

WPS による各種ファイル・データベース入出力方法

(WPS バージョン 3.1.1)

第1版

2014 年 7 月

株式会社ブレインパッド

目次

1. テキストファイルの入出力.....	3
1.1. インポートウィザードを使用したテキストファイルの入力.....	4
[インポートウィザードの使用例 (可変長テキスト入力)]	4
[インポートウィザードの使用例 (固定長テキスト入力)]	8
1.2. エクスポートウィザードを使用した WPS データセットのテキストファイル出力	
10	
[エクスポートウィザードの使用例]	10
1.3. IMPORT プロシジャによるテキストデータ入力	12
[IMPORT プロシジャの指定方法 (区切り文字テキスト)]	12
[IMPORT プロシジャの使用例 (区切り文字テキスト)]	14
1.4. EXPORT プロシジャによるテキストデータ出力.....	17
[EXPORT プロシジャの指定方法 (区切り文字テキスト)]	17
[EXPORT プロシジャの使用例 (区切り文字テキスト)]	19
2. Excel シートの入出力	20
[準備]	20
2.1. Excel シートの入力.....	21
[IMPORT プロシジャの指定方法 (Excel シート)]	21
[IMPORT プロシジャの使用例 (Excel シート)]	22
[IMPORT プロシジャの追加指定ステートメント (Excel シート)]	24
2.2. Excel シートへの出力.....	24
[EXPORT プロシジャの指定方法 (Excel シート)]	25
[EXPORT プロシジャの使用例 (Excel シート)]	26
2.3. libname ステートメントを用いた Excel シート入出力.....	27
[libname ステートメントによる EXCEL の入出力]	28
3. Access データベースの入出力.....	29
[準備]	29
3.1. Access データベースの入力.....	30
[IMPORT プロシジャの指定方法 (Access)]	30
[IMPORT プロシジャの使用例 (Access)]	31
[IMPORT プロシジャの追加指定ステートメント (Access)]	32
3.2. Access データベースへの出力	33
[EXPORT プロシジャの指定方法 (Access)]	33
[EXPORT プロシジャの使用例 (Access)]	34
[EXPORT プロシジャの追加指定ステートメント (Access)]	35

3.3.	libname ステートメントを用いた Access データ入出力	36
	[libname ステートメントによる ACCESS の入出力]	37
4.	各種データベースの入出力	38
	[WPS がサポートしているデータベース]	38
	[AdventureWorks2012 サンプル DB]	40
4.1.	ネイティブドライバー経由の DB テーブルの入出力	40
	[SQL 言語を使用した DB 読み取り]	41
	[libname ステートメントによる DB 入力]	43
	[WPS データセットの DB への出力方法]	47
4.2.	ODBC ドライバー経由の DB テーブルの入出力	49
	[ODBC ドライバのインストール]	49
	[PC 側の ODBC ドライバーの設定]	50
	[SQL 言語を使用した DB 読み取り]	52
	[libname ステートメントによる DB 入力]	55
	[WPS データセットの DB への出力方法]	58

WPS3.1 基本プロダクト (WPS CORE) には、テキスト形式のファイル (.txt, .dat, .csv)、Microsoft 社の Excel シートや Access データテーブルを簡単に読み書きするための IMPORT プロシジャと EXPORT プロシジャが用意されています。また、WPS ワークベンチを使う場合は、テキスト形式のファイル (.txt, .dat, .csv) を読み書きするためのテキストデータ読み取りウィザードとテキストデータ出力ウィザードが備わっています。その他 Oracle, DB2, SQL Server, Teradata, MY SQL など主要なデータベース上のデータの入出力を行う DB インタフェースが別途用意されています。

本資料は、Windows 版 WPS3.1 を WPS ワークベンチから起動し、SAS 言語の DATA ステッププログラミングを用いなくて、これらのデータを読み書きする方法を説明しています。

なお、本資料では、主に SHIFT-JIS エンコード (正確には MS932 エンコード) の日本語データを取扱っています。そのため、WPS ワークベンチの設定が、以下のとおり設定されていることを前提としています。

- ・ 起動オプションに `encoding="SHIFT-JIS"` ¹
- ・ ワークスペースのテキストファイルエンコードに MS932²

なお、プログラムをバッチ実行する場合は、`-encoding SHIFT-JIS` (起動オプション) を指定して実行することが必要です。

1. テキストファイルの入出力

1 「WPS サーバーエクスプローラー」ビューの「ローカルサーバー」を右クリック → 「プロパティ(R)」をクリック → 「プロパティ: ローカルサーバー」画面の左領域の「起動オプション」をクリック → 右領域の「追加」をクリック → 「起動オプション」画面出現 → 「選択」をクリック → 出現した「起動オプションを選択」画面のオプションリストの中から「ENCODING」を探してクリック → 「OK」をクリック → 「起動オプション」画面の「名前:」欄に ENCODING が入っていることを確認 → 「値:」欄に SHIFT-JIS を入力 → 「OK」。なお、バージョン 3.1.1 から LOCALE="JAPANESE_JAPAN"を指定できるようになり、この指定を行うと、ENCODING="SHIFT-JIS" に自動設定されるようになりました。

2 「メニューバー」の「ウィンドウ(W)」 → 「設定(P)」 → 左領域の「一般」 → 「ワークスペース」をクリック → 右領域の左下部にある「テキスト・ファイル・エンコード(T)」の「その他(O)」ラジオボタンをクリック → 入力欄に MS932 とキー入力 → 「OK」。なお、バージョン 3.1.1 では既定の設定になりました。

ユーザーが SAS 言語の DATA ステップのプログラムを書かずに、テキストファイルを WPS データセットに入力したり、WPS データセットをテキストファイルに出力したりするには、以下の方法が利用できます。

- (1) インポートウィザード (WPS ワークベンチのみ)
- (2) エクスポートウィザード (WPS ワークベンチのみ)
- (3) IMPORT プロシジャ
- (4) EXPORT プロシジャ

1.1. インポートウィザードを使用したテキストファイルの入力

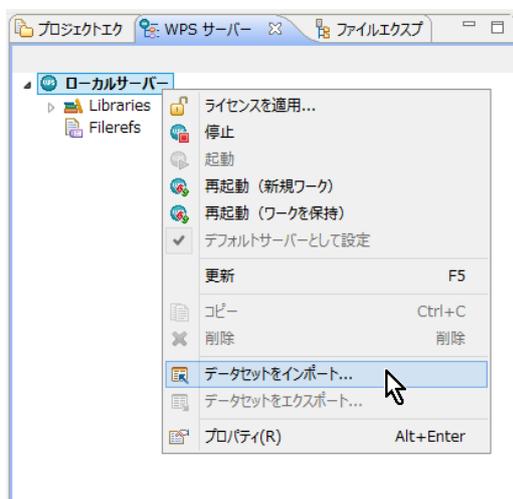
[インポートウィザードの使用例(可変長テキスト入力)]

以下の内容を持つ CSV (カンマ区切りテキストデータ) 形式のデータテキストファイルが C:¥Users¥USER1¥employees.csv に保存されているものとします。

employees.csv ファイルの内容

```
name,age,job
ベス,41,販売マネージャ
アン,28,マーケティング
アンドリュー,36,技術サポート
ウィリアム,35,経理
スティーブ,32,販売
ドーン,26,販売
```

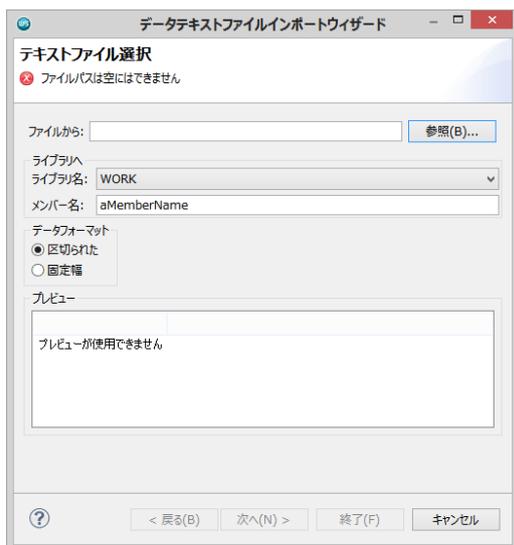
WPS ワークベンチの WPS サーバーエクスプローラービューの ローカルサーバー を右クリック → 「データセットをインポート...」 をクリックします。



※ ローカルサーバーの下の **Libraries** を右クリックしても同じです。また **Libraries** の下の特定のライブラリ (**SASUSER** や **WORK**) を右クリック → 「データセット... をインポート」をクリックすると、そのライブラリ内にデータセットが作成されます。

データテキストファイルインポートウィザードが開きます。

「参照(B)...」をクリックします。



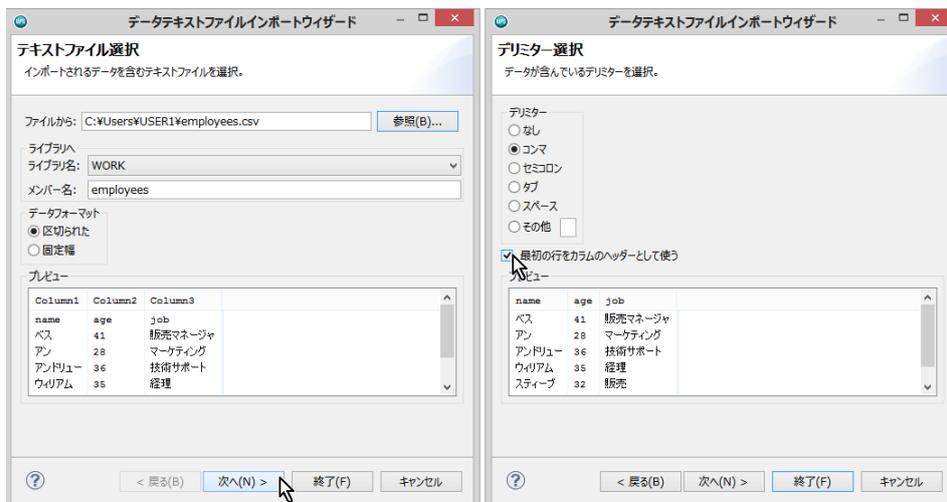
インポートしたいテキストファイルを選択し、「開く (O)」をクリックし、**C:¥Users¥USER1¥employees.csv** を選択します。

データテキストファイルインポートウィザードが表示され、読み込むファイルのプレビュー画面が表示されます。(下の左図)

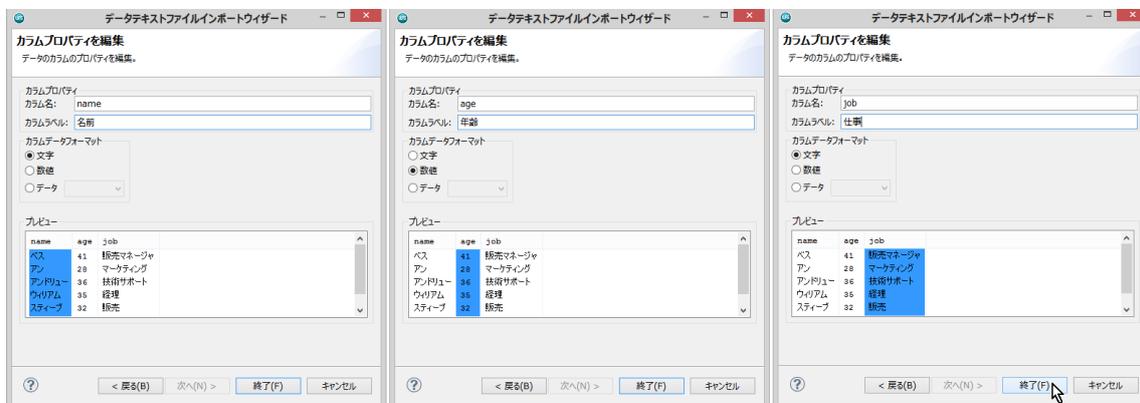
プレビューを見て、正しくデータが読み込めているかどうかを確認し、これで良ければ「終了(F)」をクリックします。

しかし、この場合は、データの1行目に入っている項目名 (**name, age, job**) がデータ値として読み込まれようとしていることがプレビューからわかります。これを修正するため、「終了(F)」ではなく「次へ(N)」をクリックし、「最初の行をカラムヘッダーとして使う」にチェックを入れます。(下の右図)

データ行の1行目がヘッダー (変数名) に用いられる形にプレビューが変化します。



プレビューに表示された通りにデータを読み取るのであれば「終了(F)」をクリックします。ここでは、さらに、「次へ(N)」をクリックして、「カラムプロパティを編集」画面に進みます。プレビューのヘッダー名の表示をクリックすると変更対象カラムが選択でき、そのカラムの変数名、変数ラベル、読み取りフォーマット（文字、数値、データ（日付）のいずれかのタイプ）が変更できます。ここでは、3つの変数の変数ラベルのみを変更することになります。



最後に「終了(F)」を押すと、テキストデータを読み取るプログラムが自動生成され実行されます。

生成されたプログラムの内容と実行がエラーなく行われたかどうかを、ログを見て確認します。

```

Local Server のログ
1 ODS_ALL_CLOSE;
2 FILENAME WPSWBHTML TEMP;
3 ODS HTML(ID=WBHTML) BODY=WPSWBHTML GPATH="C:\Users\USER1\AppData
  ! ry Data\TDR212";
NOTE: Writing HTML(WBHTML) Body file WPSWBHTML
4 DATA WORK.employees;
5 INFILE 'C:\Users\USER1\employees.csv' DLM = ',' DSD MISSOVER LR
6 FORMAT name $12.;
7 FORMAT age 3.;
8 FORMAT job $14.;
9 INFORMAT name $12.;
10 INFORMAT age 3.;
11 INFORMAT job $14.;
12 LABEL name='名前';
13 LABEL age='年齢';
14 LABEL job='仕事';
15 INPUT name$
16 age
17 job$;

NOTE: The file 'C:\Users\USER1\employees.csv' is:
      File Name 'C:\Users\USER1\employees.csv',
      Lrecl=28, Recfm=v

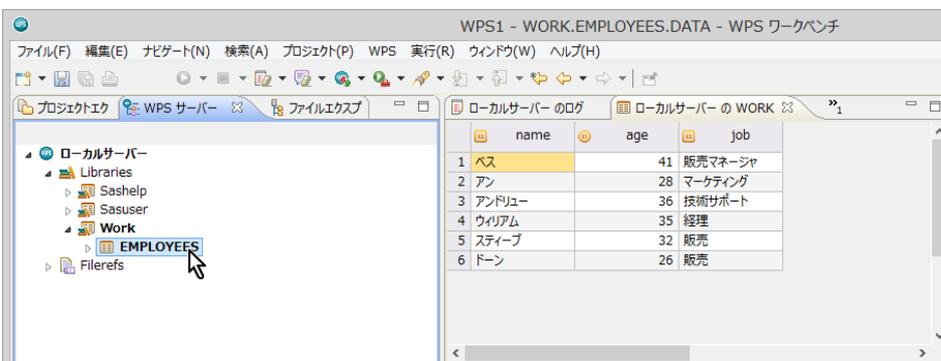
NOTE: 6 records were read from file 'C:\Users\USER1\employees.csv'
      The minimum record length was 14
      The maximum record length was 28
NOTE: Data set "WORK.employees" has 6 observation(s) and 3 variable(s)
NOTE: The data step took:
      real time : 0.035
      cpu time : 0.000

18 quit; run;
19 ODS_ALL_CLOSE;

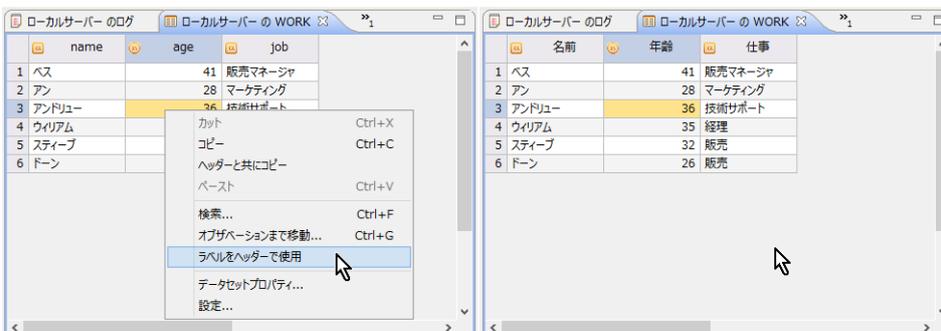
```

ログを確認したら、作成された WPS データセットの中身を確認します。

WPS サーバーエクスプローラービューの Libraries の中の WORK の中の EMPLOYEES データセットをダブルクリックすると、データセットビューが表示されます。



データセットビュー内で右クリック → 「ラベルをヘッダーで使用」をクリックすると、変数名の表示が変数ラベル表示に切り替わります。



[インポートウィザードの使用例(固定長テキスト入力)]

現在のバージョンのWPS ワークベンチのインポートウィザードは、ダブルバイト(DBCS) またはマルチバイト(MBCS) データを含む固定長テキストデータの読み取りを行うと問題³が生じます。そこで、ここでは、シングルバイトテキストのみを含む固定長テキストの読み取り例をとり上げます。

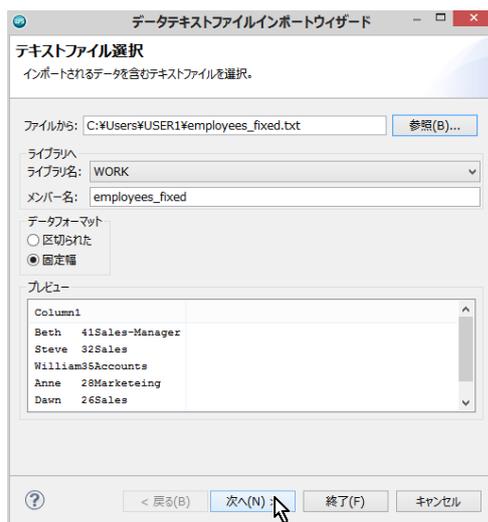
以下の内容を持つ固定長データテキストファイルが C:¥Users¥USER1¥employees_fixed.dat に保存されているものとします。

employees_fixed.dat ファイルの内容

```
Beth 41Sales-Manager
Steve 32Sales
William35Accounts
Anne 28Marketeing
Dawn 26Sales
Andrew 36Support
```

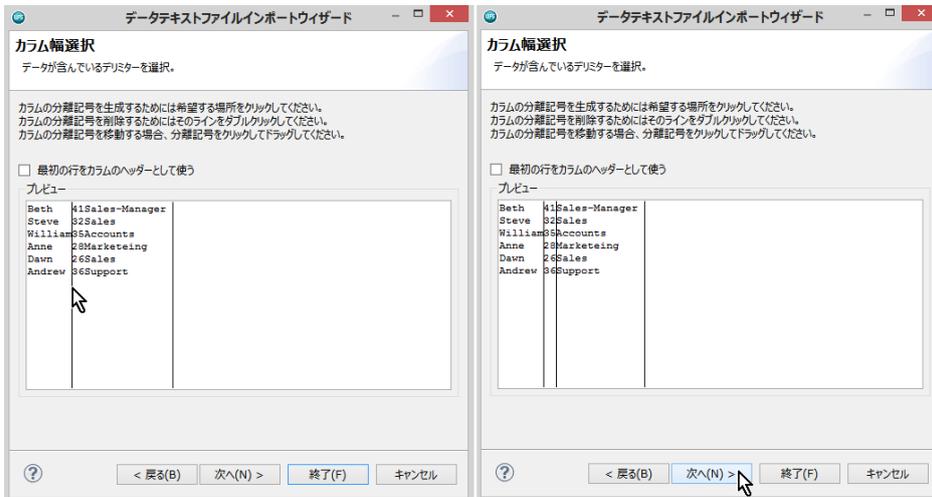
WPS ワークベンチの WPS サーバーエクスプローラービューの ローカルサーバー を右クリック → データセットをインポート... → データテキストファイルインポートウィザードで

「参照(B)...」をクリック → C:¥Users¥USER1 ディレクトリに保存しておいた employees_fixed.txt を選択し「開く(O)」をクリックすると、データのプレビュー画面が表示されます。



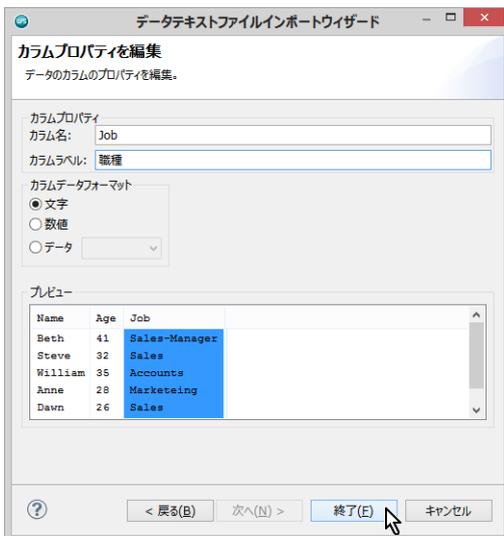
「次へ(N)」をクリックし、カラム分離位置を指定します。

3 カラム位置にズレが生じるという問題です。



※ウィザード内の説明にあるように、分離線をダブルクリックすると線を削除でき、選択してドラッグすると移動できます。

「次へ(N)」に進み、コラム名とラベルを指定します。



最後に「終了(F)」を押すと、テキストデータを読み取るプログラムが自動生成され実行されます。

生成されたプログラムの内容とエラーなく行われたかどうかは、ログを見て確認します。

```
プログラム1.sas ローカルサーバーのログ
39 ODS _ALL_ CLOSE;
40 FILENAME WPSMBHTM TEMP;
41 ODS HTML(ID=WMBHTML) BODY=WPSMBHTM GPATH="C:\Users\USER1\AppData\Local\Temp\WPS Tempora
41 ! ry Data\_TD6348";
NOTE: Writing HTML (WBHTML) Body file WPSMBHTM
42 DATA WORK.employees_fixed;
43 INFILE 'C:\Users\USER1\employees_fixed.txt' MISSEVER LRECL=22 FIRSTOBS=1;
44 FORMAT Name $7.;
45 FORMAT Age 2.;
46 FORMAT Job $13.;
47 LABEL Name='名前';
48 LABEL Age='年齢';
49 LABEL Job='仕事';
50 INPUT @1 Name$ 7.
51 @8 Age 2.
52 @10 Job$ 13.;
```

```
NOTE: The file 'C:\Users\USER1\employees_fixed.txt' is:
File Name 'C:\Users\USER1\employees_fixed.txt',
Lrecl=22, Recfm=V

NOTE: 6 records were read from file 'C:\Users\USER1\employees_fixed.txt'
The minimum record length was 22
The maximum record length was 22
NOTE: Data set "WORK.employees_fixed" has 6 observation(s) and 3 variable(s)
NOTE: The data step took :
real time : 0.006
cpu time : 0.000
```

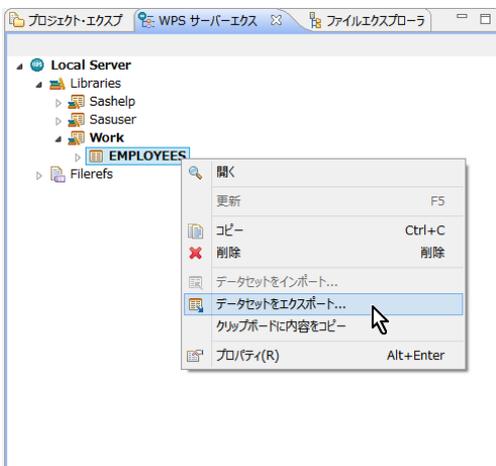
ログを確認したら、作成された WPS データセットの中身を確認します。

1.2. エクスポートウィザードを使用した WPS データセットのテキストファイル出力

WPS ワークベンチのエクスポートウィザードを使うと、WPS データセットを簡単にテキストファイルに出力できます。

[エクスポートウィザードの使用例]

インポートウィザードでテキストファイルから WPS データセットに読み込んだ EMPLOYEES データセットを、今度は、逆に、テキストファイルに出力してみましょう。WPS サーバーエクスプローラビュー内の出力したいデータセットを右クリック → 「データセットをエクスポート...」をクリックします。

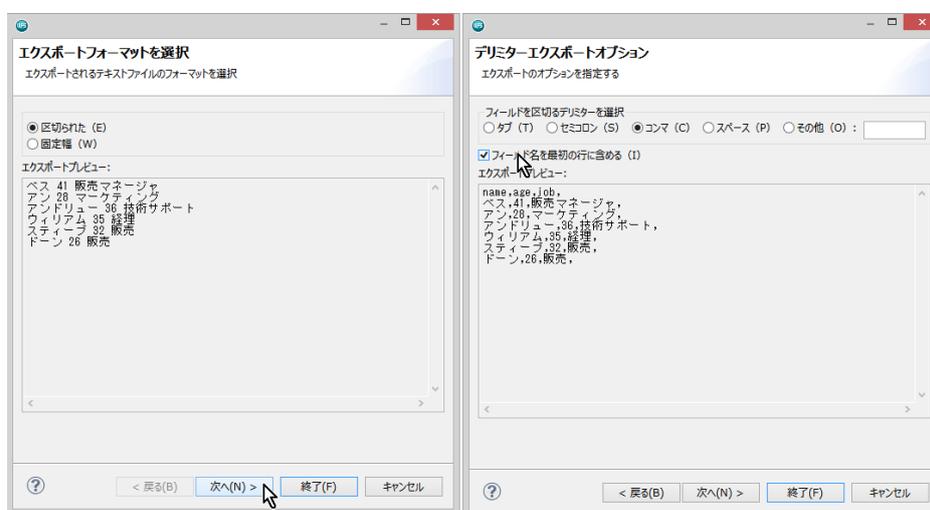


※メニュー内の「クリップボードに内容をコピー」を選択すると、データはヘッダー付きでクリップボードにコピーされて、そのまま Excel シートなどに貼り付けできる状態になります。小さなデータセットの場合はこのやり方も便利です。

エクスポートウィザードが開き、データのエクスポートプレビューが表示されます。

デリミターで区切られたデータフォーマットで出力するか、固定長フォーマット形式で出力するかを聞いてきます。ここでは、CSV ファイルとして出力することを想定し、既定の「区切られた (E)」のままにしておいて「次へ(N)」を押します。(左図)

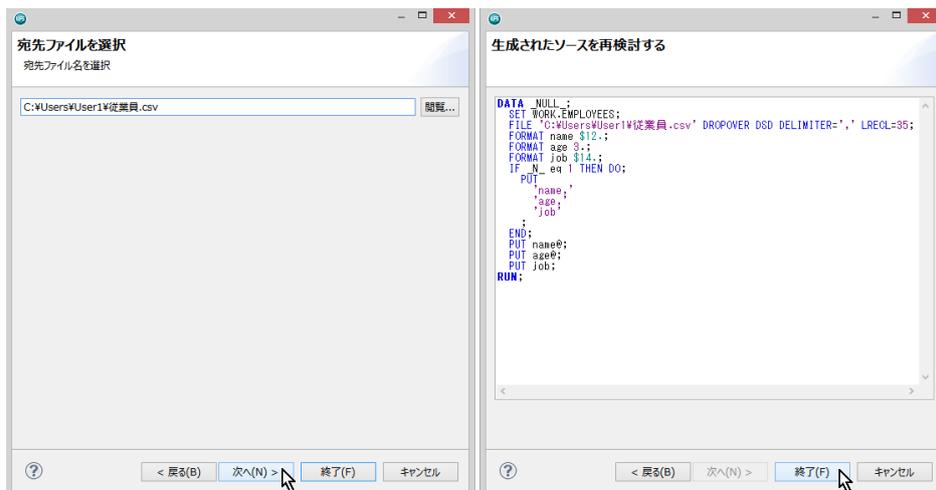
デリミター (区切り文字) として「コンマ (C)」を選択し、「フィールド名を最初の行に含める」にチェックを入れます。エクスポートプレビューが変わるのを確認します。(右図)



※ 固定幅 (W) を選択すると、固定長形式でファイル出力できますが、インポートウィザードの場合と同じく、全角文字 (マルチバイト文字) を含む場合は問題があります。

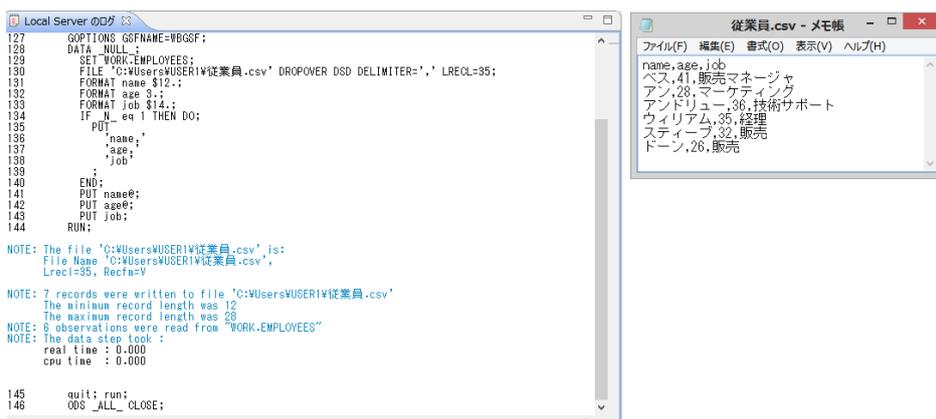
「次へ(N)」を押すと「宛先ファイルを選択」画面に変わります。ここでは C:\¥Users¥USER1¥従業員.csv という名前のファイルに出力することにします。(左図)

「次へ(N)」を押すと、生成されたプログラムが表示されます。(右図)



「終了(F)」を押すと、プログラムが実行されます。

ログを確認し (左図)、作成された「従業員.csv」の内容を確認してみましょう。(右図)



1.3. IMPORT プロシジャによるテキストデータ入力

IMPORT プロシジャは可変長タイプのテキストデータや Excel シート、Access データテーブルを WPS データセットに変換します。

Excel シート、Access データテーブルを WPS データセットに変換する IMPORT プロシジャの使い方については別の章で説明することにして、この節では、区切り文字によりデータが区切られた可変長タイプのテキストデータの入力方法についてのみ説明します。

[IMPORT プロシジャの指定方法(区切り文字テキスト)]

```

proc import datafile="入力テキストファイル名" dbms=データタイプ out=出力WPSデータ
セット名 replace;
  getnames=1行目を変数名（ヘッダー）として用いるかどうかを指定(YES/NO);
  datarow=データとして読み取りを開始する行;
  guessingrows=変数の型の判断に用いるデータ行数;
  delimiter="区切り文字";
run;

```

・ datafile="入力テキストファイル名"

入力テキストデータファイル名は、通常、ディレクリパスを含んだフルパスを拡張子（.csv, .txt など）付きで指定します。ディレトリパスを省略すると、ファイルはカレントディレクトリ（WPS ワークベンチ実行の場合は現在のワークスペース）に存在するものとみなされます。（必須指定）

なお、先行する filename ステートメントで"入力テキストファイル名"のファイル参照名を定義している場合は、datafile=ファイル参照名 という指定が可能です。

・ dbms=データタイプ

オプション指定です。dbms=データタイプはテキストファイルでは以下が指定可能です。この指定により、入力ファイルの区切り文字は暗黙的に以下のように認識されます。

データタイプ	区切り文字
dbms=csv	カンマ ('2C'x)
dbms=dlm	スペース ('20'x)
dbms=tab	タブ ('09'x)

この認識で良ければ delimiter=指定は省略できます。もしも delimiter=指定が指定された場合は、delimiter=指定が優先され、指定された区切り文字がデータとデータの間の区切り文字として認識されます。

なお、datafile=指定に以下の拡張子を持つテキストファイルを拡張子を付けて直接指定する場合のみ、dbms=指定も省略することができます。この場合は、入力テキストデータファイルの拡張子によって暗黙的に以下のように認識されます。

拡張子	データタイプ	区切り文字
.csv	dbms=csv	カンマ ('2C'x)
.txt	dbms=tab	タブ ('09'x)

・ out=出力 WPS データセット名

出力 WPS データセットの名前を指定します。（必須指定）

- `replace`

`proc import` ステートメントの `replace` 指定は「`out=`出力 WPS データセットに指定した WPS データセットが既に存在した場合は上書きしてください」という意味のオプションです。指定しない場合は、`out=`に指定した WPS データセットが存在した場合エラーとなり既存の WPS データセットは更新されません。

以下はオプションステートメントです。必要に応じて指定します。

- `getnames=YES/NO;`

データファイルの最初の 1 行目を変数名（ヘッダー）として使うかどうかを指定するオプションステートメントです。既定は `getnames=YES` です。

なお、`=`の代わりにスペースを書いてもかまいません。（`getnames YES;` 以下の指定も同様）

- `datarow=`データとして読み取を行う開始行の指定;

既定は `getnames=YES` の場合 `datarow=2` 、 `getnames=NO` の場合 `datarow=1` です。

- `guessingrows=`変数の型と長さを判断するために検査するデータ行数;

既定では `guessingrows=20`。IMPORT プロシジャは、最初にデータファイルの 1 行目から 20 行目までを検査して各変数の型と長さを決定してから、全データの読み取りを行います。したがって、例えば、20 行目までは数字、21 行目に文字が入っているデータを IMPORT プロシジャで読むと、既定では数値タイプの変数として定義されるため、20 行目までは有効な値、21 行目の値は欠損値として読み込まれます。

できるだけ正しくデータの型を決定したい場合は、実行時間がその分長くなりますが、`guessingrows=`の値を大きくします。なお、`guessingrows=`は `getnames=`や `datarow=`の指定とは無関係で、物理的なデータ行数の指定を意味します。

- `delimiter="区切り文字";`

明示的にデータ値の区切り文字を指定します。IMPORT プロシジャでは 2 つ以上の文字を指定することもできます。例えば、カンマとスペースのいずれかの文字をデータ間の区切り文字に指定する場合は、

```
delimiter=", " または delimiter="2C20"x;
```

と指定します。

[IMPORT プロシジャの使用例(区切り文字テキスト)]

以下の CSV 形式のデータファイルが `C:¥Users¥USER1¥employees.csv` に保存されてい

るものとしてします。

employees.csv ファイルの内容

```
name,age,job
ベス,41,販売マネージャ
アン,28,マーケティング
アンドリュー,36,技術サポート
ウィリアム,35,経理
スティーブ,32,販売
ドーン,26,販売
```

以下のコーディングにより、このデータファイルを読み込み、WPS データセット employees を作成します。

```
proc import datafile="c:\%users%\USER1\employees.csv" out=employees replace;
run;
```

もしくは、以下のように指定します。

```
filename in1 "c:\%users%\USER1\employees.csv";
proc import datafile=in1 out=employees dbms=csv replace;run;
```

プログラムを実行すると、以下のような SAS 言語プログラムが生成され実行され、ログビューに実行ログが表示されます。

```
172      proc import datafile="c:\%users%\USER1\employees.csv" out=employees replace;
173      run;
NOTE: Procedure import step took :
      real time : 0.000
      cpu time   : 0.000

174      data employees;
175          infile 'c:\%users%\USER1\employees.csv' delimiter=',' MISSOVER DSD firstobs=2 LRECL=32760;
176          informat name $15.;
177          informat age BEST32.;
178          informat job $14.;
179          format name $15.;
180          format age BEST12.;
181          format job $14.;
182          label name = 'name';
183          label age = 'age';
184          label job = 'job';
185          input   name $
186                age
187                job $
188                ;
189          run;

NOTE: The file 'c:\%users%\USER1\employees.csv' is:
      File Name 'c:\%users%\USER1\employees.csv',
      Lrecl=32760, Recfrm=V
```

NOTE: 6 records were read from file 'c:\%users%\USER1\employees.csv'
 The minimum record length was 14
 The maximum record length was 28
 NOTE: Data set "WORK.employees" has 6 observation(s) and 3 variable(s)
 NOTE: The data step took :
 real time : 0.006
 cpu time : 0.000

WPS データセット WORK.EMPLOYEES の内容を確認してください。

	name	age	job
1	ベス	41	販売マネージャ
2	アン	28	マーケティング
3	アンドリュー	36	技術サポート
4	ウィリアム	35	経理
5	スティーブ	32	販売
6	ドーン	26	販売

なお、ヘッダー行を読まないようにするには、`getnames=NO; datarow=2;`を指定します。
 出力 WPS データセットの変数名は自動的に VAR1, VAR2, ... という名前が付けられます。
 なお、ヘッダー行に日本語が含まれる場合もこの指定を行う必要があります。

(プログラム)

```
filename in1 "c:\%users%\USER1\employees.csv";
proc import datafile=in2 out=employees2 dbms=csv replace;
  getnames=NO;
  datarow=2;
run;
```

(ログ)

```
4      filename in1 "c:\%users%\USER1\employees.csv";
5      proc import datafile=in1 out=employees2 dbms=csv replace;
6          getnames=NO;
7          datarow=2;
8      run;
NOTE: Procedure import step took :
      real time : 0.015
      cpu time : 0.015

9      data employees2;
10         infile in1 delimiter=';' MISSOVER DSD firstobs=2 LRECL=32760;
11         informat VAR1 $15.;
12         informat VAR2 $3.;
13         informat VAR3 $18.;
14         format VAR1 $15.;
15         format VAR2 $3.;
16         format VAR3 $18.;
17         label VAR1 = 'VAR1';
18         label VAR2 = 'VAR2';
19         label VAR3 = 'VAR3';
20         input  VAR1 $
```

```
21          VAR2 $
22          VAR3 $
23          ;
24          run;
```

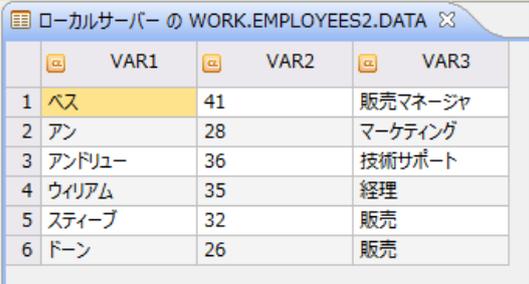
NOTE: The file in1 is:
File Name 'c:\users\USER1\employees.csv',
Lrecl=32760, Recfm=V

NOTE: 6 records were read from file in1
The minimum record length was 14
The maximum record length was 28

NOTE: Data set "WORK.employees2" has 6 observation(s) and 3 variable(s)

NOTE: The data step took :
real time : 0.000
cpu time : 0.000

(出力 WPS データセット)



	VAR1	VAR2	VAR3
1	ベス	41	販売マネージャ
2	アン	28	マーケティング
3	アンドリュー	36	技術サポート
4	ウィリアム	35	経理
5	スティーブ	32	販売
6	ドーン	26	販売

1.4. EXPORT プロシジャによるテキストデータ出力

EXPORT プロシジャは WPS データセットを可変長タイプのテキストデータや Excel シート、Access データテーブルに変換します。

WPS データセットを Excel シート、Access データテーブルへ変換する EXPORT プロシジャの使い方については別の章で説明することにして、この節では、区切り文字によりデータが区切られた可変長タイプのテキストデータへの出力方法についてのみ説明します。

[EXPORT プロシジャの指定方法(区切り文字テキスト)]

```
proc export data=入力WPSデータセット名 outfile="出力テキストファイル名" dbms=データタイプ replace label;  
  delimiter="区切り文字";  
run;
```

・ data=入力 WPS データセット名

入力 WPS データセット名を指定します。(指定が無ければ、同じセッションで最後に作成

された WPS データセットが暗黙的に用いられます)

• **outfile**="出力テキストファイル名"

出力テキストデータファイル名を、通常、ディレクトリパスを含んだフルパスを拡張子 (.csv, .txt など) 付きで指定します。(ディレクトリパスを省略すると、カレントディレクトリ (WPS ワークベンチ実行の場合は現在のワークスペース) に作成するものとみなされます。)(必須指定)

• **dbms**= データタイプ

オプション指定です。この指定により、出力ファイルにはデータ間の区切り文字が以下のように書き込まれます。

dbms=指定	区切り文字
dbms=csv	カンマ ('2C'x)
dbms=dlm	スペース ('20'x)
dbms=tab	タブ ('09'x)

この設定で良ければ **delimiter**=指定は省略できます。もしも **delimiter**=指定を指定すると、**delimiter**=指定が優先され、指定された文字がデータ間の区切り文字として書き込まれます。

なお、**outfile**=指定に以下の拡張子を持つテキストファイルを拡張子を付けて直接指定する場合のみ、**dbms**=指定も省略することができます。この場合は、出力テキストデータファイルに付けた拡張子によって暗黙的に以下のように認識されます。

拡張子	暗黙設定	区切り文字
.csv	dbms=csv	カンマ ('2C'x)
.txt	dbms=tab	タブ ('09'x)

• **replace**

proc export ステートメントの **replace** 指定は「**outfile**="出力テキストファイル名"に指定したファイルが既に存在した場合は上書きしてください」という意味のオプションです。指定しない場合は、**outfile**=に指定したテキストファイルが存在した場合はエラーとなり既存のテキストファイルは更新されません。

• **label**

label 指定は、出力ファイルの最初の行に WPS データセットの変数名の代わりに、変数ラベルを書き出すよう指定するオプションです。変数ラベルが定義されていない変数については変数名が書き出されます。

• **delimiter**="区切り文字";

明示的にデータ値の区切り文字を指定します。例えば、コロン(:)をデータとデータの間に書き込みたい場合は、

```
delimiter=":" または delimiter="3A"x;
```

と指定します。なお、EXPORT プロシジャでは1バイト文字1個のみを指定します。

[EXPORT プロシジャの使用例(区切り文字テキスト)]

WPS データセット WORK.employees を EXPORT プロシジャを使って CSV 形式のテキストファイルに変換してみましょう。

(プログラム)

```
proc export data=employees outfile="c:¥users¥USER1¥employees.csv"
replace;run;
```

(ログ)

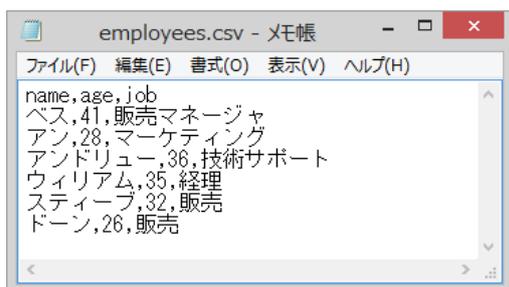
```
80      proc export data=employees outfile="c:¥users¥USER1¥employees.csv" replace;run;
NOTE: Procedure export step took :
      real time : 0.000
      cpu time  : 0.000

81      data _null_;
82      set employees;
83      file 'c:¥users¥USER1¥employees.csv' delimiter=',' DSD DROPOVER LRECL=50;
84      format name $15.;
85      format age BEST12.;
86      format job $14.;
87      if _n_=1 then
88          do;          put
89              'name'      ','
90              'age'       ','
91              'job';
92          end;
93          put name @;
94          put age @;
95          put job;

NOTE: The file 'c:¥users¥USER1¥employees.csv' is:
      File Name 'c:¥users¥USER1¥employees.csv',
      Lrecl=50, Recfm=V

NOTE: 7 records were written to file 'c:¥users¥USER1¥employees.csv'
      The minimum record length was 12
      The maximum record length was 28
NOTE: 6 observations were read from "WORK.employees"
NOTE: The data step took :
      real time : 0.002
      cpu time  : 0.000
```

(出力された CSV 形式のテキストファイル employees.csv)



2. Excel シートの入出力

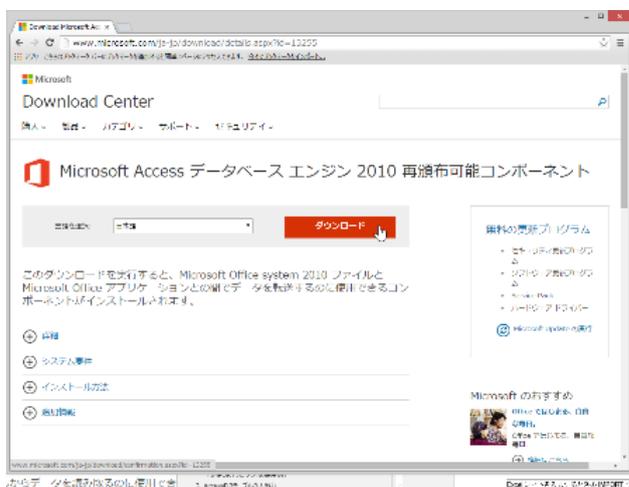
IMPORT プロシジャ（入力）と EXPORT プロシジャ（出力）を使えば、SAS 言語の DATA ステップのプログラムを書かずに Excel シートの指定範囲の値を直接 WPS データセットに入力したり、WPS データセットのデータを直接 Excel シートに出力することができます。

[準備]

ここに記載した Excel シートへのアクセスが「クラスが登録されていません」というエラーが出現しアクセスが失敗する場合は、「マイクロソフトデータ接続コンポーネント」を検索ワードとしてインターネット検索を行い、マイクロソフト社ダウンロードセンターにアクセスし、リソースへ接続するために必要なコンポーネントプログラムをインストールしてください。

導入されている Office のバージョンによって、以下のいずれかのデータ接続コンポーネントをインストールします。

(Office2010 以降の場合) 64 ビット版と 32 ビット版があります。



(Office2007 の場合)



2.1. Excel シートの入力

Excel シートを入力するときの IMPORT プロシジャの指定方法は次のとおりです。

[IMPORT プロシジャの指定方法(Excel シート)]

```
proc import datafile="入力Excelファイル名" dbms=Excelバージョン out=出力WPSデータセット名 replace;
  getnames=1行目を変数名 (ヘッダー) として用いるかどうかを指定(YES/NO);
  range="シート名$セル範囲";
  (または sheet="シート名");
run;
```

・ datafile="入力 Excel ファイル名"

入力 Excel ファイル名は、通常、ディレクリパスを含んだフルパスを拡張子 (.xls, .xlsx) 付きで指定します。ディレクリパスを省略すると、ファイルはカレントディレクトリ (WPS ワークベンチ実行の場合は現在のワークスペース) に存在するものとみなされます。(必須指定)

・ dbms= Excel バージョン

オプションの指定です。以下の指定が可能です。

dbms=EXCEL / EXCEL95 / EXCEL97 / EXCEL2000 / EXCEL2002 / EXCEL2003 / EXCEL2007 / EXCEL2010

- ・ **out=**出力 WPS データセット名

出力 WPS データセットの名前を指定します。(必須指定)

- ・ **replace**

proc import ステートメントの **replace** 指定は「**out=**出力 WPS データセットに指定した WPS データセットが既に存在した場合は上書きしてください」という意味のオプションです。指定しない場合は、**out=**に指定した WPS データセットが既に存在している場合はエラーとなり既存の WPS データセットは更新されません。

- ・ **getnames=**データファイルの最初の 1 行目を変数名 (ヘッダー) として使うかどうかの指定(YES/NO);

既定は **getnames=YES** です。

なお、**=**の代わりにスペースを書いてもかまいません。(getnames YES; 以下の指定も同様)

- ・ **range="**シート名\$セル範囲";

複引用符の中に読み取りたい「シート名」、「\$」、「セル範囲」の順に指定します。

(例) **range="sheet1\$A1:Q120";**

- ・ **sheet="**シート名"

読み込むシート名を明示的に指定します。

(例) **sheet=sheet1;**

sheet=の指定は特定のシート全体を読み込む場合に便利です。

※ **range=**指定; または **sheet=**指定; のいずれか一方のみを指定してください。

[IMPORT プロシジャの使用例(Excel シート)]

以下の **employees.xlsx** ファイル (C:\¥Users\¥USER1¥ employees.xlsx) の **employees** シートの **A1:C7** の範囲のデータを WPS データセットに変換します。ただし、1 行目は変数名として用います。

	A	B	C	D	E
1	name	age	job		
2	ベス	41	販売マネージャ		
3	アン	28	マーケティング		
4	アンドリュー	36	技術サポート		
5	ウィリアム	35	経理		
6	スティーブ	32	販売		
7	ドーン	26	販売		
8					
9					

(プログラム)

```
proc import datafile="C:\Users\USER1\社員.xlsx" out=employees_sheet replace;
  range="employees$A1:D7";
  getnames=YES;
run;
```

(ログ)

```
356      proc import datafile="C:\Users\USER1\社員.xlsx" out=employees_sheet replace;
357          range="employees$A1:D7";
358          getnames=YES;
359      run;

NOTE: Procedure import step took :
      real time : 0.000
      cpu time  : 0.000

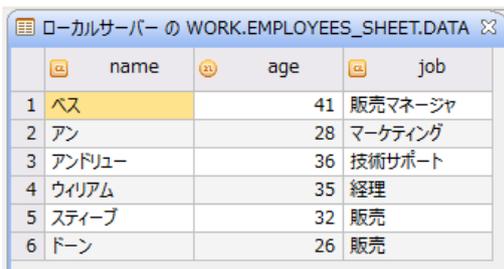
360          libname _EXCIMP excel "C:\Users\USER1\社員.xlsx" mixed=NO header=YES
msengine=ACE use_
360      ! datatype=YES scan_textsize=YES scan_timetype=YES dbmax_text=1024
NOTE: Library _EXCIMP assigned as follows:
      Engine:      OLEDB
      Physical Name:

361      ;
362      data employees_sheet;
363      set _EXCIMP.employees$A1:D7'n;
364      run;

NOTE: 6 observations were read from "_EXCIMP.employees$A1:D7"
NOTE: Data set "WORK.employees_sheet" has 6 observation(s) and 3 variable(s)
NOTE: The data step took :
      real time : 0.234
      cpu time  : 0.046

NOTE: Libref _EXCIMP has been deassigned.
365      libname _EXCIMP clear;
366      quit; run;
```

(WPS データセット employees_sheet)



	name	age	job
1	ベス	41	販売マネージャ
2	アン	28	マーケティング
3	アンドリュー	36	技術サポート
4	ウィリアム	35	経理
5	スティーブ	32	販売
6	ドーン	26	販売

[IMPORT プロシジャの追加指定ステートメント(Excel シート)]

以下のオプションステートメントが Excel シート入力の場合に指定可能です。

- `mixed=YES/NO`;

Excel の文字/数値混在列の値は WPS データセットに変換するとすべて文字変数値として読み込まれますが、Excel では数値タイプで定義されたセル値をどう取り扱うかの指定。`mixed=YES` なら数値を文字に自動変換して読み込みます。既定の `mixed=NO` では数値は文字変数の欠損値 (ブランク) となります。(※ しかし、実際に試すと `mixed=` の指定に関わらず数値は文字に変換されて読み込まれます。)

- `textsize=`テキストフィールド読み取り最大長さの指定;

既定は `textsize=1024`; より長いテキストフィールドを読むためには、より大きな値を指定する必要があります。(1~32767)

- `scantext=YES/NO`;

テキストフィールドの値を読むかどうかの指定。既定は `scantext=YES`; (※ しかし、実際には `scantext=` の指定に関わらずテキストフィールドは常に読み込むことができます。)

- `scantime=YES/NO`;

既定は `scantime=YES`; 読み取る Excel シートに含まれる日付/時間フィールドの値を検査して、時間のみを含むフィールドであれば SAS 言語の時間値を適用するようにする指定。

- `usedate=YES/NO`;

既定の `usedate=YES` では読み取る Excel シートに含まれる日付/時間フィールドの値を日付値(日付情報のみ持つ値)として読み取ります。 `usedate=NO` を指定すると、日時値(日付と時間の両方の情報を持つ値)として読み取られます。

2.2. Excel シートへの出力

WPS データセットを Excel シートへ出力するときの EXPORT プロシジャの指定方法は次

のとおりです。

[EXPORT プロシジャの指定方法(Excel シート)]

```
proc export data=入力WPSデータセット名 outfile="出力Excelファイル名" dbms=Excel  
バージョン replace label;  
  sheet=出力シート名;  
run;
```

- data=入力 WPS データセット名

入力 WPS データセット名を指定します。(指定が無ければ、同じセッションで最後に作成された WPS データセットが暗黙的に用いられます)

- outfile="出力 Excel ファイル名"

出力 Excel ファイル名を、通常、ディレクトリパスを含んだフルパスを拡張子 (.xls, .xlsx) 付きで指定します。(ディレクトリパスを省略すると、カレントディレクトリ (WPS ワークベンチ実行の場合は現在のワークスペース) に作成するものとみなされます。)(必須指定)

- dbms=Excel バージョン

オプション指定です。以下の指定が可能です。

dbms=EXCEL / EXCEL97 / EXCEL2000 / EXCEL2002 / EXCEL2003 / EXCEL2007 / EXCEL2010

- replace

proc export ステートメントの replace 指定は「outfile="出力 Excel ファイル名"の中に sheet="出力 Excel シート名"に指定したシートが既に存在した場合はそのシートの内容を上書きしてください」という意味のオプションです。sheet=を指定しなかった場合は outfile="出力 Excel ファイル名"の中の入力 WPS データセット名の名前がついたシートが存在チェックの対象となります。対象 Excel シートが既に存在した場合はエラーとなり既存の Excel ファイルの内容は全く更新されません。

- label

label 指定は、出力ファイルの最初の行に WPS データセットの変数名の代わりに、変数ラベルを書き出すよう指定するオプションです。変数ラベルが定義されていない変数については変数名が書き出されます。

- sheet=出力シート名;

出力 Excel ファイルのシート名を指定するオプションステートメントです。

※ sheet=を指定しなかった場合は、入力 WPS データセット名をシート名とする Excel シートが作成されます。

[EXPORT プロシジャの使用例(Excel シート)]

WPS データセット WORK.employees を EXPORT プロシジャを使って Excel ファイル (C:¥Users¥USER1¥社員 2.xlsx) に変換してみましょう。

(プログラム)

```
proc export data=employees outfile="c:¥users¥USER1¥社員 2.xlsx" dbms=Excel
replace;
run;
```

(ログ)

```
124      proc export data=employees outfile="c:¥users¥USER1¥社員 2.xlsx" dbms=Excel replace;
125      run;
NOTE: Procedure export step took :
      real time : 0.000
      cpu time  : 0.000

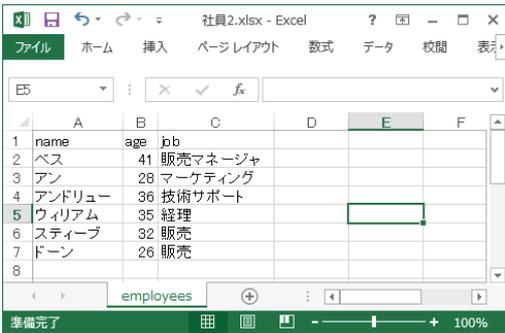
126      libname _EXCEXP excel 'c:¥users¥USER1¥社員 2.xlsx' msengine=ACE  replace;
NOTE: Library _EXCEXP assigned as follows:
      Engine:      OLEDB
      Physical Name:

127      data _EXCEXP.'employees'n;
128      set employees;run;

NOTE: 6 observations were read from "WORK.employees"
NOTE: Data set "_EXCEXP.employees" has an unknown number of observation(s) and 3 variable(s)
NOTE: The data step took :
      real time : 0.265
      cpu time  : 0.062

NOTE: Libref _EXCEXP has been deassigned.
129      libname _EXCEXP clear;
130      quit; run;
```

出力された Excel ファイル 社員 2.xlsx (シート名は既定の WPS データセット名になっています)



2.3. libname ステートメントを用いた Excel シート入出力

もう 1 つの Excel シート入出力方法として、libname ステートメントで Excel エンジンを設定しそのファイルを占有モードでアクセスする方法があります。以下のステートメントを実行すると、Excel ファイルをあたかも既存の WPS データライブラリ、個々のシートを既存の WPS データセットとみなした取扱いができるようになります。

```
libname ライブラリ参照名 EXCEL "Excelファイル物理パス名";
```

- ・ライブラリ参照名

ここで指定する Excel ファイルの参照名を、8 文字以内のアルファベットまたはアンダースコア(_)または数字で指定します。ただし、先頭の 1 文字はアルファベットまたはアンダースコア(_)でなければなりません。

- ・ EXCEL

Excel ファイルのアクセスの場合は、エンジン名として EXCEL と指定します。

- ・ "Excel ファイル物理パス名"

アクセスしたい Excel ファイルのフルパスを拡張子 (.xls, .xlsx) 付きで指定します。なお、新たに Excel ファイルを作成したいときは、存在しないファイル名を指定します。

また、ライブラリ参照を削除して、ファイルの占有状態を解除するには、以下の指定を行います。

```
libname ライブラリ参照名;
```

または、

```
libname ライブラリ参照名 CLEAR;
```

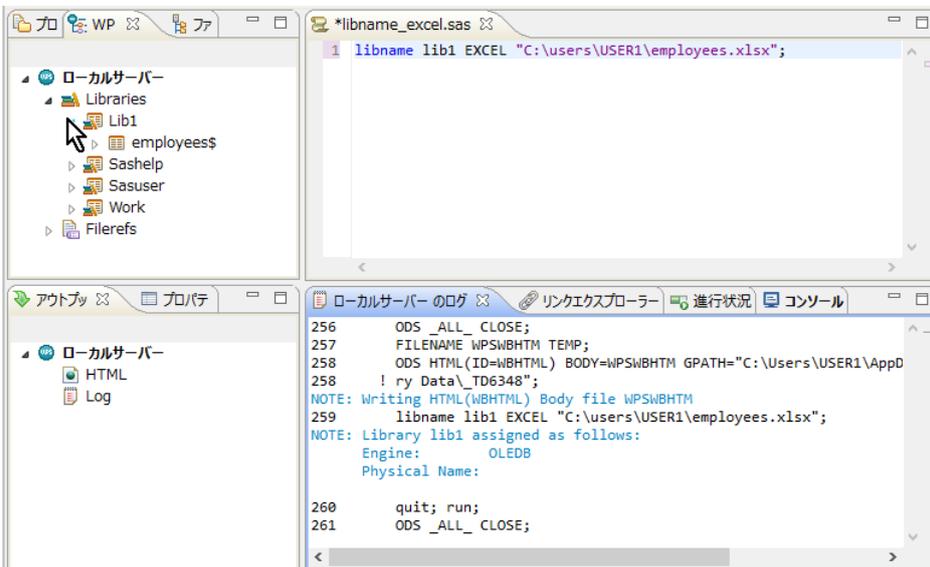
[libname ステートメントによる EXCEL の入出力]

この例は、Excel ファイル (c:\users\USER1\employees.xlsx) を EXCEL エンジンを使って直接アクセスする例です。

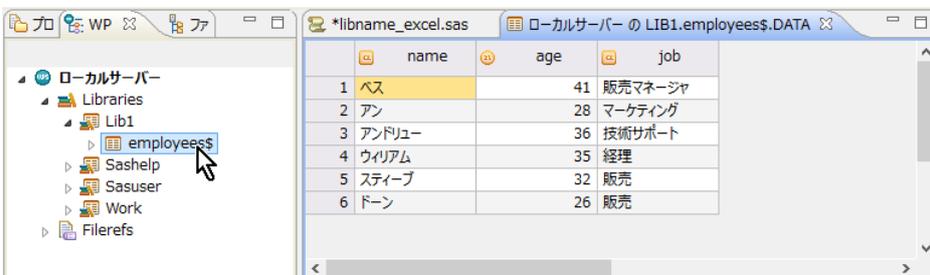
以下のプログラムをエディタビューに入力し、選択してから実行します。

```
libname lib1 EXCEL "C:\users\USER1\employees.xlsx";
```

以下のように、WPS サーバーエクスプローラービューの ローカルサーバー の下に lib1 ライブラリが出現します。



lib1 ライブラリを展開すると、employees シートが出現し、これをダブルクリックすると employees シートの内容を見ることができます。



EXECL エンジンを使う方法は、Excel シートを一旦 WPS データセットに変換する手順をユーザーが省略できるというメリットがあります。しかし、何度も同じシートにアクセスす

る場合は、IMPORT プロシジャで WPS データセットに変換した方が効率的な場合があります。また、既存のファイル内容を変更（シートの削除や追加）してしまう可能性もありますので、注意が必要です。

3. Access データベースの入出力

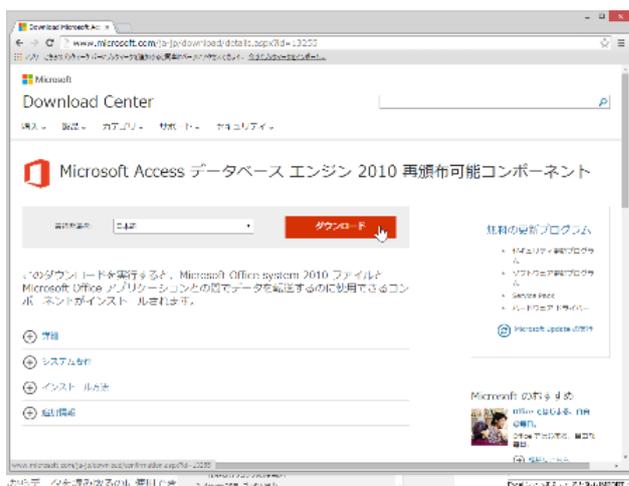
IMPORT プロシジャ（入力）と EXPORT プロシジャ（出力）を使えば、SAS 言語の DATA ステップのプログラムを書かずに Access データベースのテーブルの値を直接 WPS データセットに入力したり、WPS データセットのデータを直接 Access データベースに出力することができます。

[準備]

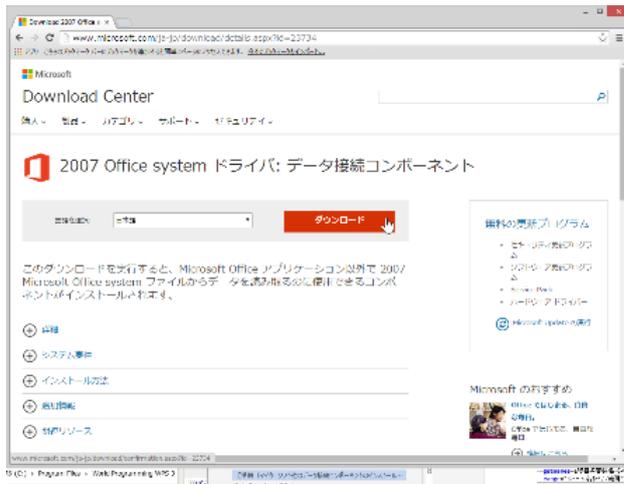
ここに記載した Access データベースへのアクセスが「クラスが登録されていません」というエラーが出現しアクセスが失敗する場合は、「マイクロソフトデータ接続コンポーネント」を検索ワードとしてインターネット検索を行い、マイクロソフト社ダウンロードセンターにアクセスし、リソースへ接続するために必要なコンポーネントプログラムをインストールしてください。

導入されている Office のバージョンによって、以下のいずれかのデータ接続コンポーネントをインストールします。

(Office2010 以降の場合) 64 ビット版と 32 ビット版があります。



(Office2007 の場合)



3.1. Access データベースの入力

Access データベースを入力するときの IMPORT プロシジャの指定方法は次のとおりです。

[IMPORT プロシジャの指定方法(Access)]

```
proc import datafile="入力Accessデータベース名" dbms=Accessバージョン
datatable="入力Accessデータテーブル名"
out=出力WPSデータセット名 replace;
run;
```

- datafile="入力 Access データベース名"

入力 Access データベース名は、通常、ディレクリパスを含んだフルパスを拡張子 (.mdb, .accdb) 付きで指定します。ディレトリパスを省略すると、ファイルはカレントディレクトリ (WPS ワークベンチ実行の場合は現在のワークスペース) に存在するものとみなされます。(必須指定)

- dbms= Access バージョン

オプションの指定です。以下の指定が可能です。

dbms=ACCESS / ACCESS97 / ACCESS2000 / ACCESS2002 / ACCESS2003 / ACCESS2007 / ACCESS2010

- datatable="入力 Access データテーブル名"

入力する Access データテーブル名を指定します。(必須指定)

・ out=出力 WPS データセット名

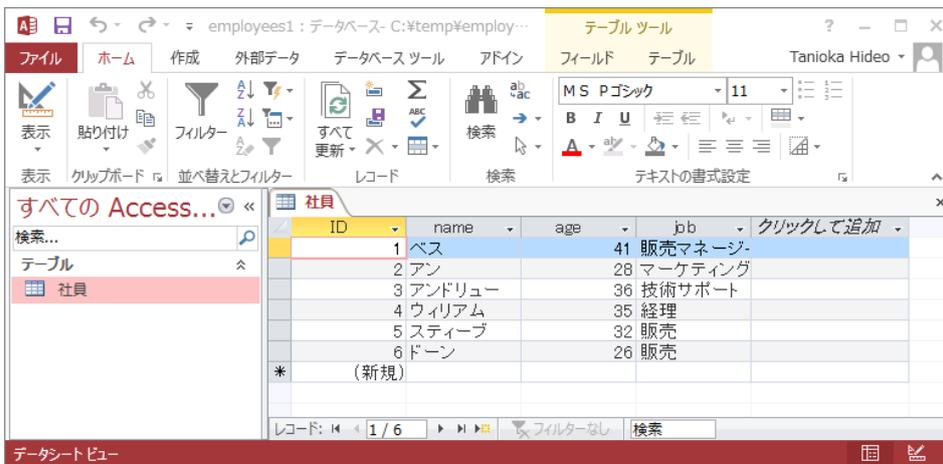
出力 WPS データセットの名前を指定します。(必須指定)

・ replace

proc import ステートメントの replace 指定は「out=出力 WPS データセットに指定した WPS データセットが既に存在した場合は上書きしてください」という意味のオプションです。指定しない場合は、out=に指定した WPS データセットが既に存在している場合はエラーとなり既存の WPS データセットは更新されません。

[IMPORT プロシジャの使用例(Access)]

以下のデータが Access データベース employees1.accdb (C:¥temp¥employees1.accdb) の社員テーブルにあるものとします。これを IMPORT プロシジャを使って WPS データセット shain に変換します。



ID	name	age	job
1	ベス	41	販売マネージ-
2	アン	28	マーケティング
3	アンドリュー	36	技術サポート
4	ウィリアム	35	経理
5	スティーブ	32	販売
6	ドーン	26	販売

(プログラム)

```
proc import datafile="C:¥temp¥employees.accdb" dbms=ACCESS datatable="社員"  
out=shain replace;  
run;
```

(ログ)

```
201      proc import datafile="C:¥temp¥employees.accdb" dbms=ACCESS datatable="社員" out=shain  
201      ! replace;  
202      run;  
NOTE: Procedure import step took :  
      real time : 0.000  
      cpu time  : 0.000
```

```

203      libname _ACCIMP ACCESS "C:¥temp¥employees.accdb" msengine=ACE use_datatype=NO
scan_tex
203      ! tsize=YES scan_timetype=YES dbmax_text=1024;
NOTE: Library _ACCIMP assigned as follows:
      Engine:      OLEDB
      Physical Name:

204      data shain;
205      set _ACCIMP.'社員'n;run;

NOTE: 6 observations were read from "_ACCIMP.社員"
NOTE: Data set "WORK.shain" has 6 observation(s) and 4 variable(s)
NOTE: The data step took :
      real time : 0.175
      cpu time  : 0.015

NOTE: Libref _ACCIMP has been deassigned.
206      libname _ACCIMP clear;
207      quit; run;

```

(WPS データセット shain)

ID	name	age	job
1	ヘス	41	販売マネージャ
2	アン	28	マーケティング
3	アンドリュー	36	技術サポート
4	ウィリアム	35	経理
5	スティーブ	32	販売
6	ドーン	26	販売

[IMPORT プロシジャの追加指定ステートメント(Access)]

Access データベースへのアクセスを制限している場合はアクセス条件（データベースパスワード、ユーザー名など）の入力を促すプロンプト画面が出現します。その場合は、画面から入力します。

また、**proc import** ステートメントと **run** ステートメントの間に、以下のデータベースアクセス用のオプションステートメントを追加することにより、プロンプト無しにデータベースへアクセスすることもできます。

- **database**="データベース名";
- **dbpassword**="データベースパスワード";
- **user**="ユーザー名";
- **password**="ユーザーパスワード";
- **wgdb**="アクセスシステムファイル名";

また、以下のオプションステートメントを必要に応じて追加指定することができます。

- **memosize**=メモ型フィールドの読み取り最大長さの指定;

既定は **memosize=1024**; より長いテキストがメモ型フィールドに格納されている場合は、より大きな値を指定してください。(1~32767)

- **scanmemo**=YES/NO;

既定は **scanmemo=YES**; メモ型フィールドを読むかどうかの指定。

- **scantime**=YES/NO;

既定は **scantime=YES**; 読み取る **Access** データテーブルに含まれる日付/時間フィールドの値を検査して、時間のみを含むフィールドであれば **SAS** 言語の時間値を適用するようにする指定。

- **usedate**=YES/NO;

usedate=YES を指定すると読み取る **Access** データテーブルに含まれる日付/時間フィールドの値を日付値(日付情報のみ持つ値)として読み取ります。既定の **usedate=NO** の場合は、日時値(日付と時間の両方の情報を持つ値)として読み取られます。

3.2. Access データベースへの出力

WPS データセットを **Access** データベースへ出力するときの **EXPORT** プロシジャの指定方法は次のとおりです。

[EXPORT プロシジャの指定方法(Access)]

```
proc export data=入力WPSデータセット outfile="出力Accessファイル名" dbms=Access
バージョン outtable="出力Accessデータテーブル名" replace;
run;
```

- **data**=入力 WPS データセット名

入力 WPS データセット名を指定します。(指定が無ければ、同じセッションで最後に作成された WPS データセットが暗黙的に用いられます)

- **outfile**="出力 Access データベース名"

出力 **Access** データベース名を、通常、ディレクトリパスを含んだフルパスを拡張子 (.mdb, .accdb) 付きで指定します。(ディレクトリパスを省略すると、カレントディレクトリ (WPS ワークベンチ実行の場合は現在のワークスペース) に作成するものとみなされます。)(必須指定)

・ dbms=Access バージョン

オプション指定です。以下の指定が可能です。

dbms=ACCESS / ACCESS97 / ACCESS2000 / ACCESS2002 / ACCESS2003 / ACCESS2007 / ACCESS2010

・ outtable="出力 Access データテーブル名"

作成する Access データテーブル名を指定します。

・ replace

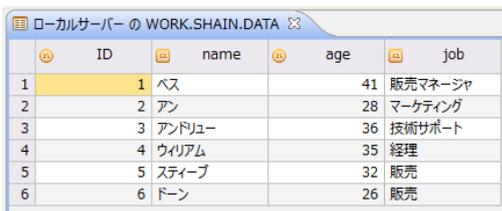
proc export ステートメントの replace 指定は「outfile="出力 Access データベース名"に指定した Access データベースファイルの中の outtable="出力 Access データテーブル名"に指定したテーブルが既に存在した場合は上書きしてください」という意味のオプションです。指定しない場合は、outtable =に指定したテーブル名が既に存在した場合はエラーとなり既存の Access データテーブルは更新されません。

・ label

label 指定は、Access データテーブルのカラム名として WPS データセットの変数名の代わりに、変数ラベルを用いるよう指定するオプションです。変数ラベルが定義されていない変数については変数名が使われます。

[EXPORT プロシジャの使用例 (Access)]

以下の WPS データセット (WORK.SHAIN) を EXPORT プロシジャを使って Access データベースファイル"C:¥temp¥shain.accdb"の中のテーブル"社員 2"に変換します。



ID	name	age	job
1	ベス	41	販売マネージャ
2	アン	28	マーケティング
3	アンドリュー	36	技術サポート
4	ウイリアム	35	経理
5	スティーブ	32	販売
6	ドーン	26	販売

(プログラム)

```
proc export data=shain outfile="C:¥temp¥shain.accdb" dbms=ACCESS outtable="社員 2" replace;  
run;
```

(ログ)

```

393      proc export data=shain outfile="C:¥temp¥shain.accdb" dbms=ACCESS outtable="社員 2" rep
393      ! lace;
394      run;
NOTE: Procedure export step took :
      real time : 0.000
      cpu time  : 0.000

395      libname _ACCIMP ACCESS 'C:¥temp¥shain.accdb' msengine=ACE replace;
NOTE: Library _ACCIMP assigned as follows:
      Engine:      OLEDB
      Physical Name:

396      data _ACCIMP.'社員 2 'n;
397      set shain;
398      run;

NOTE: 6 observations were read from "WORK.shain"
NOTE: Data set "_ACCIMP.社員 2" has an unknown number of observation(s) and 4 variable(s)
NOTE: The data step took :
      real time : 0.250
      cpu time  : 0.031

NOTE: Libref _ACCIMP has been deassigned.
399      libname _ACCIMP clear;
400      quit; run;

```

(Access データテーブル社員 2)

ID	name	age	job
1	ベス	41	販売マネージ-
2	アン	28	マーケティング
3	アンドリュー	36	技術サポート
4	ウィリアム	35	経理
5	スティーブ	32	販売
6	ドーン	26	販売

[EXPORT プロシジャの追加指定ステートメント(Access)]

Access データベースへのアクセスを制限している場合はアクセス条件（データベースパスワード、ユーザー名など）の入力を促すプロンプト画面が出現します。その場合は、画面から入力します。

また、proc export ステートメントと run ステートメントの間に、以下のデータベースアクセス用のオプションステートメントを追加することにより、プロンプト無しにデータベー

スへアクセスすることもできます。

- `database="データベース名";`
- `dbpassword="データベースパスワード";`
- `user="ユーザー名";`
- `password="ユーザーパスワード";`
- `wgdb="アクセスシステムファイル名";`

3.3. libname ステートメントを用いた Access データ入出力

もう 1 つの Access データベースの入出力方法として、libname ステートメントを用いる方法があります。以下の指定により、Access データベースをあたかも既存の WPS データライブラリ、Access データテーブルを既存の WPS データセットとみなした取扱いができるようになります。

```
libname ライブラリ参照名 ACCESS "Accessデータベース物理パス名";
```

- ライブラリ参照名

ここで指定する Access データベースの参照名を、8 文字以内のアルファベットまたはアンダースコア(_)または数字で指定します。ただし、先頭の 1 文字はアルファベットまたはアンダースコア(_)でなければなりません。

- ACCESS

Access データベースのアクセスの場合はエンジン名として ACCESS と指定します。

- "Access データベース物理パス名"

アクセスしたい Access データベースのフルパスを拡張子 (.mdb, .accdb) 付きで指定します。なお、新たに Access データベースを作成したいときは、存在しないファイル名を指定します。

また、ライブラリ参照を削除して、ファイルの占有状態を解除するには、以下の指定を行います。

```
libname ライブラリ参照名;
```

または、

```
libname ライブラリ参照名 CLEAR;
```

[libname ステートメントによる ACCESS の入出力]

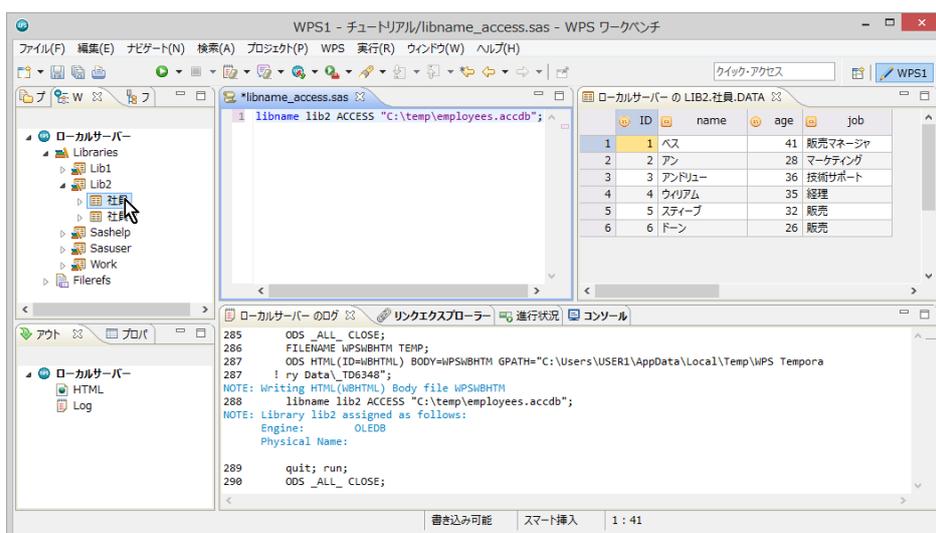
この例は、Access データベース (C:\temp\employees1.accdb) を ACCESS エンジンを使って直接アクセスする例です。

以下のプログラムをエディタビューに入力し、選択してから実行します。

```
libname lib2 ACCESS "C:\temp\employees.accdb";
```

WPS サーバーエクスプローラービューの ローカルサーバー の下に lib2 ライブラリが出現し、その中に既存の Access データテーブルの名前が表示されます。

ここで、「社員」アイテムをダブルクリックすると、データテーブル社員の内容がビューに表示されます。



続いて、以下のプログラムをエディタに入力し、選択してから実行します。

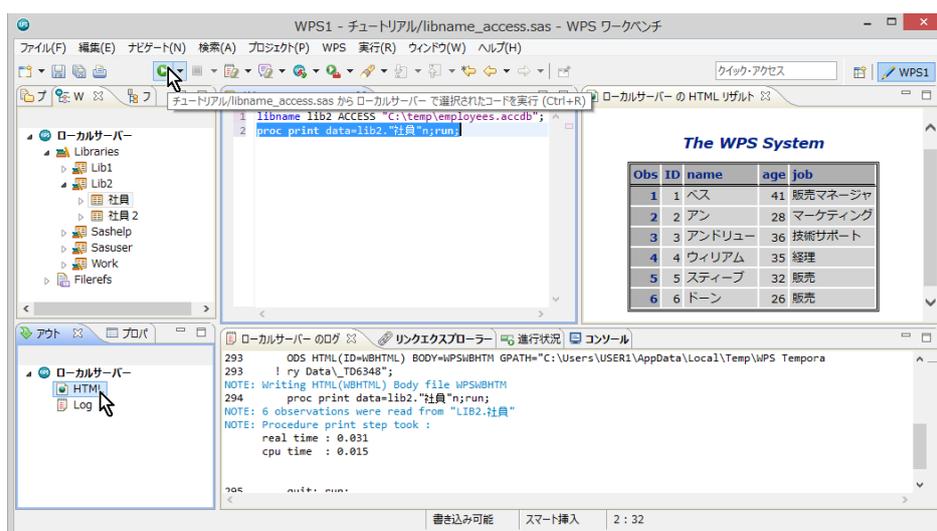
PRINT プロシジャは data=に指定したデータセットの中身をリストの形でプリント表示するプロシジャです。

```
proc print data=lib2."社員"n;run;
```

※ "社員"n の表記は ネームリテラル表記と呼ばれます。このように "データセット名"n、または、"変数名"n という書き方をすることにより、半角英数字以外のテキスト (全角文字

や半角スペース、特殊文字) を含む WPS データセット名や変数名を定義できます。ただし、WPS セッション内で `validvarname=any` というオプションが有効である必要があります。

実行後、アウトプットエクスプローラビューの中の ローカルサーバー の下の HTML アイテムをダブルクリックすると、下記のように、上記 PRINT プロシジャの結果出力が HTML 形式で表示されます。



4. 各種データベースの入出力

WPS は多数の商用、またはオープンソースのデータベース (以下 DB と表記) に対するインターフェース機能 (API) をサポートしています。インターフェース機能に含まれるのは、データの読み書き、テーブルの作成・編集・削除を含みます。サポートするインタフェース方式は、DB ごとに固有のコマンドライン接続方式 (CLI ドライバー、ネイティブドライバ) に加えて、多くの DB に対してそれぞれの DB ベンダーから提供されているオープンデータベース接続方式 (ODBC ドライバ) の両方です。

[WPS がサポートしているデータベース]

WPS3.1 がサポートしている DB とそのインターフェース機能の種類とアクセスの種類は以下のとおりです。

(インタフェース機能)

データベース	読み取り	書き込み	更新	新規 テーブルの 作成	パスス ルー (暗黙)	パスス ルー (明示)	バルクロード	バルクアップ ロード
Action Matrix	●	●	●	●	●	●	×	×
DB2	●	●	●	●	●	●	● (z/OS以外)	×
Greenplum	●	●	●	●	●	●	●	×
Informix	●	●	●	●	●	●	×	×
Kognitio	●	●	×	●	●	●	×	×
MySQL	●	●	●	●	●	●	×	×
Netezza	●	●	●	●	●	●	●	●
ODBC	●	●	●	●	●	●	● (SQL Serverのみ)	×
OLEDB	●	●	●	●	●	●	×	×
Oracle	●	●	●	●	●	●	●	×
PostgreSQL	●	●	●	●	●	●	●	×
SAND	●	●	●	●	●	●	●	×
SQL Server	●	●	●	●	●	●	●	×
Sybase	●	●	●	●	●	●	×	×
Teradata	●	●	●	●	●	●	×	×
Vertica	●	●	●	●	●	●	×	×

OS 別サポート状況は以下のとおりです。

(OS環境)

データベース	AIX pSeries(P ower)	Linux x86	Mac OS X	Solaris SPARC	Solaris x86	Windows x86	Linux System z	z/OS System z
Action Matrix	×	●	×	×	×	●	×	×
DB2	●	●	×	●	●	●	●	●
Greenplum	●	●	●	×	●	●	×	×
Informix	×	×	×	×	×	●	×	×
Kognitio	×	●	×	×	●	●	×	×
MySQL	●	●	●	●	●	●	●	×
Netezza	●	●	●	●	●	●	×	×
ODBC	●	●	●	●	●	●	×	×
OLEDB	×	×	×	×	×	●	×	×
Oracle	●	●	●	●	●	●	●	×
PostgreSQL	×	●	●	×	●	●	●	×
SAND	●	●	×	●	●	×	×	×
SQL Server	×	×	×	×	×	●	×	×
Sybase	●	●	×	●	●	●	×	×
Teradata	●	●	×	●	●	●	×	●
Vertica	●	●	×	×	●	●	×	×

※ ODBC 接続については OS や対象 DB によってサードパーティ製品の導入が必要になる場合もあります。

本資料では、Windows8.1 上に WPS3.1 と SQL Server2012 Express Edition が導入されている環境において、CLI ドライバ経由および ODBC ドライバ経由で WPS から SQL Server2012 の DB テーブルの入出力を行う場合を例として WPS の DB インタフェース機能の使い方を説明します。なお、ここに示した例は、Windows 認証方式でローカルの SQL Server に簡単にログインできる環境で実行していますが、ネットワーク環境や、ログイン

WPS の SQLSERVER エンジンネイティブドライバを経由して SQL Server とアクセスを行う仕組みを提供します。

[SQL 言語を使用した DB 読み取り]

DB へのアクセスは SQL 言語を用いて行うことが多く、WPS でも SQL プロシジャを用いて SQL 言語による DB アクセスが可能です。

SQL 言語を用いて WPS からネイティブドライバ経由で SQL Server の DB アクセスを行う場合は、以下のように、エンジン名として SQLSERVER を指定し、CONNECT 文のカッコ内に server=サーバー名、database=DB 名などの接続オプションを指定します。

以下の例は AdventureWorks2012 データベースの Sales.Customer テーブルから、TerritoryID=1 に該当するレコードのみ抽出した全カラムを含む WPS データセット WORK.Customer を作成しています。

入力後、実行する範囲を選択してから実行します。

```
proc sql;
  connect to SQLSERVER(server="PC3¥SQLEXPRESS" database=AdventureWorks2012);
  create table Customer as
  select *
  from connection to SQLSERVER
  (
    select *
    from Sales.Customer
    where TerritoryID = 1;
  );
  disconnect from SQLSERVER;
quit;
```

※ 3行目の Create table 文はWPSデータセット WORK.Customerを作成する指定です。
8 行目の from 節で指定した Sales.Customer は AdventureWorks2012 データベースの sales スキーマの Customer テーブルを参照しています。

(WPS ワークベンチの実行ログ)

```

1 proc sql;
2 connect to SQLSERVER(server="PC3\SQLXPRESS" database=AdventureWorks2012);
3 create table Customer as
4 select *
5 from connection to SQLSERVER
6 (
7 select *
8 from Sales.Customer
9 where TerritoryID = 1;
10 );
11 disconnect from SQLSERVER;
12 quit;
13
14
15
16 quit; run;
17 ODS _ALL_ CLOSE;

```

```

2 FILENAME WPSWBHTM TEMP;
3 ODS HTML(ID=WBHTML) BODY=WPSWBHTM GPATH="C:\Users\USER1\AppData\Local\Temp\WPS Tempora
4 ! ry Data\TD17292";
NOTE: Writing HTML(WBHTML) Body file WPSWBHTM
4 proc sql;
5 connect to SQLSERVER(server="PC3\SQLXPRESS" database=AdventureWorks2012);
NOTE: Successfully connected to database SQLSERVER as alias SQLSERVER.
6 create table Customer as
7 select *
8 from connection to SQLSERVER
9 (
10 select *
11 from Sales.Customer
12 where TerritoryID = 1;
13 );
NOTE: Data set "WORK.Customer" has 3520 observation(s) and 7 variable(s)
14 disconnect from SQLSERVER;
NOTE: Successfully disconnected from database SQLSERVER.
15 quit;
NOTE: Procedure sql step took :
real time : 0.078
cpu time : 0.000
16 quit; run;
17 ODS _ALL_ CLOSE;

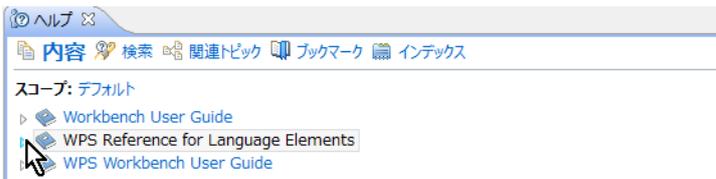
```

作成された WPS データセット WORK.Customer データセットの内容を確認します。

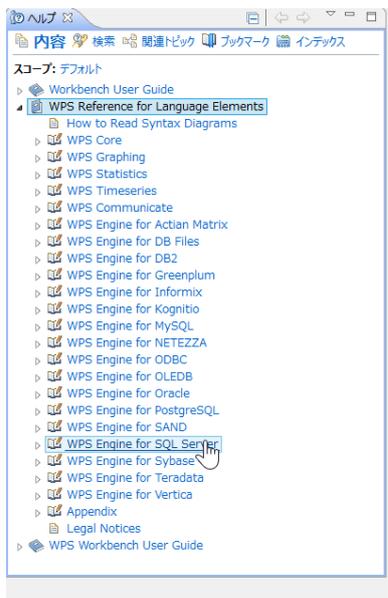
CustomerID	PersonID	StoreID	TerritoryID	AccountNumber	rowguid	ModifiedDate
1	1	.	934	1 AW00000001	3F5AE95E-B87D-4AED-	13OCT2008:11:15:07
2	2	.	1028	1 AW00000002	E552F657-A9AF-4A7D-	13OCT2008:11:15:07
3	7	.	930	1 AW00000007	029E273E-B193-44BE-9	13OCT2008:11:15:07
4	19	.	420	1 AW00000019	604E8D43-31BE-4B76-4	13OCT2008:11:15:07
5	20	.	1016	1 AW00000020	E010C10A-F1C3-48BA-	13OCT2008:11:15:07
6	37	.	938	1 AW00000037	C6EBB29A-CC67-459C-	13OCT2008:11:15:07
7	38	.	1004	1 AW00000038	E5EDA3F3-4EF1-4806-4	13OCT2008:11:15:07
8	43	.	1290	1 AW00000043	9E44903E-5D79-4F65-4	13OCT2008:11:15:07
9	55	.	1282	1 AW00000055	F5D39978-A006-46A0-	13OCT2008:11:15:07
10	56	.	992	1 AW00000056	57081682-98B1-4359-4	13OCT2008:11:15:07
11	73	.	1270	1 AW00000073	9D5C135E-BD06-46E0-	13OCT2008:11:15:07
12	74	.	528	1 AW00000074	73485E57-076D-4B10-4	13OCT2008:11:15:07
13	91	.	1258	1 AW00000091	9F314FC3-9644-4412-4	13OCT2008:11:15:07
14	92	.	632	1 AW00000092	672412C3-5D5D-488B-	13OCT2008:11:15:07
15	109	.	1890	1 AW00000109	73857908-6024-4520-4	13OCT2008:11:15:07
16	110	.	1888	1 AW00000110	82C273F0-6388-46CC-4	13OCT2008:11:15:07
17	115	.	1320	1 AW00000115	080F7047-8396-4CEC-4	13OCT2008:11:15:07
18	127	.	1328	1 AW00000127	EF55103F-ACC7-48FD-	13OCT2008:11:15:07
19	128	.	1876	1 AW00000128	03F78E74-8DE5-421E-4	13OCT2008:11:15:07
20	145	.	522	1 AW00000145	F8C5609D-0A2B-4485-4	13OCT2008:11:15:07
21	146	.	518	1 AW00000146	5475E9DD-98CA-4989-	13OCT2008:11:15:07
22	163	.	660	1 AW00000163	9899F3A-7CB8-4083-4	13OCT2008:11:15:07
23	164	.	818	1 AW00000164	5A323ACA-74A2-4CA0-	13OCT2008:11:15:07

※ 注意 : SQLSERVER ドライバーによる DB へのアクセス制限等の設定によっては、SQL プロシジャの CONNECT 文の () 内のオプションに、その他の指定が必要となる場合があります。指定可能なオプションは、以下のように、ヘルプビューで確認できます。

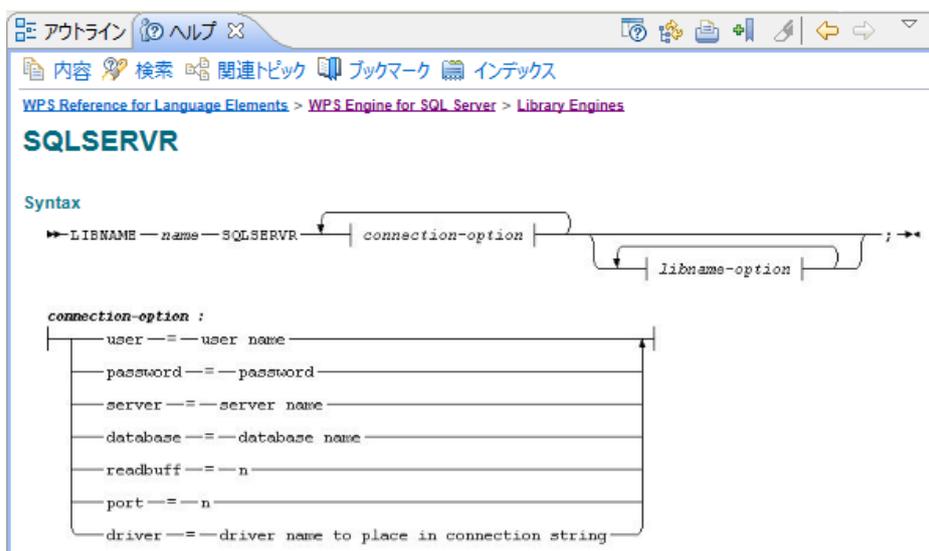
まず、ヘルプビューの左上にある「内容」をクリックします。ヘルプ全体の目次が表示されますので、WPS Reference for Language Elements をクリックします。



WPS レファレンスを展開し、WPS Engine for SQL Server を選択します。



その後、展開を何度か行くとデータベース接続オプション (connection option) が表示されます。



これらの接続オプション (user= password= server= database= readbuff= port= driver=) は SQLSERVER ドライバー経由でのデータベースアクセスの際の CONNECT 文の DBMS オプション指定としても、また、次に節で説明する libname ステートメントによる DB 接続オプションとしても用いることができ、必要に応じて指定します。

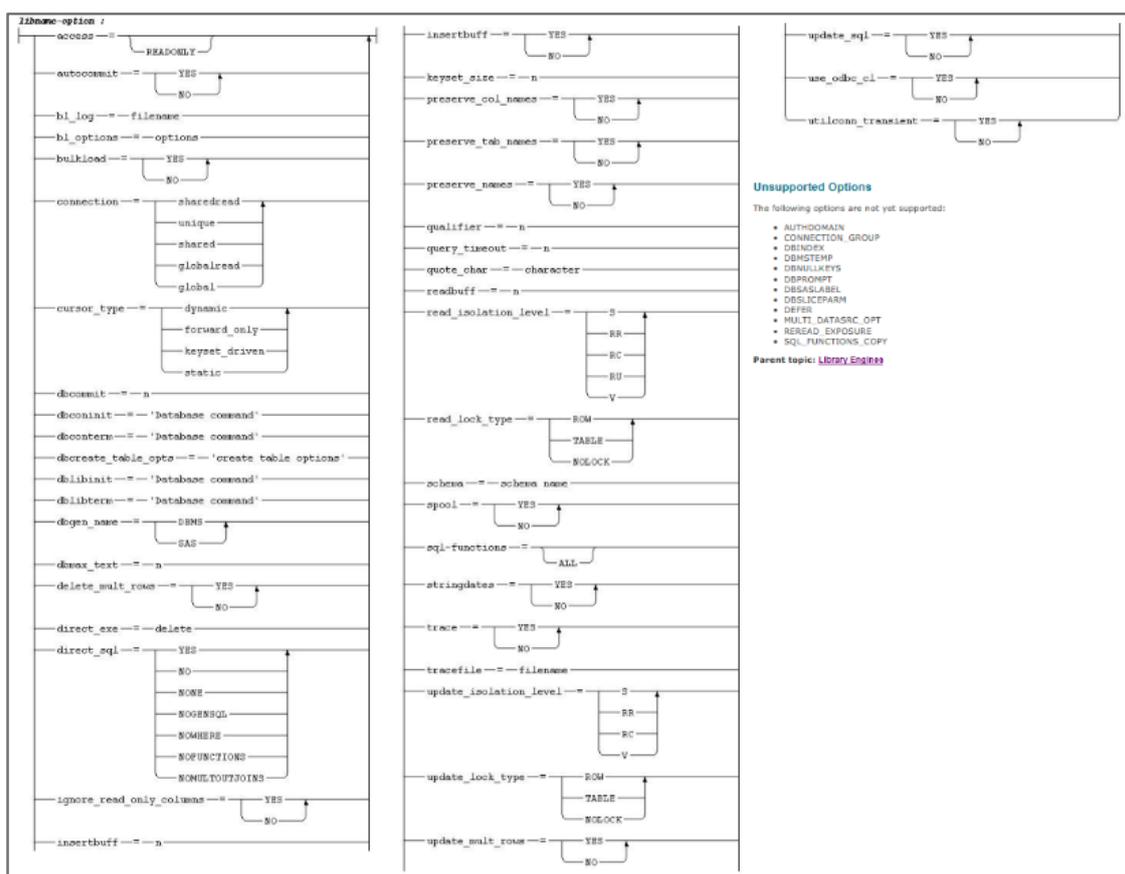
[libname ステートメントによる DB 入力]

WPS のエディタから以下のプログラムを入力し、実行します。

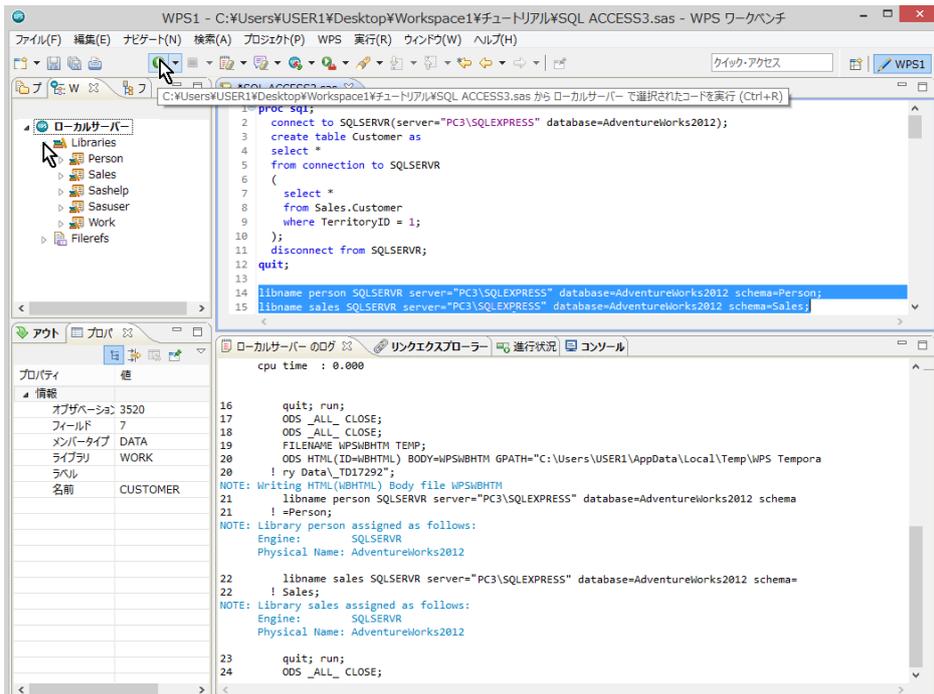
```
libname person SQLSERVER server="PC3¥SQLSERVER" database=AdventureWorks2012
schema=Person;
libname sales SQLSERVER server="PC3¥SQLSERVER" database=AdventureWorks2012
schema=Sales;
```

この指定は、WPS がデータベース AdventureWorks2012 に SQLSERVER ネイティブドライバ経由で接続し、Person スキーマのテーブルとビューをライブラリ名 person で、Sales スキーマのテーブルとビューをライブラリ名 sales で、それぞれ参照できるようにする指定です。(入出力両方可能)

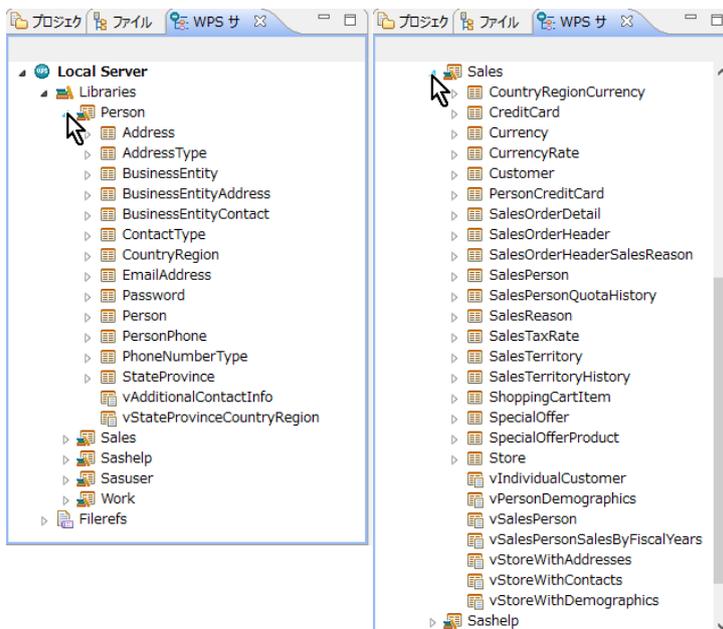
なお、server=サーバー名、database=DB 名指定は接続オプション、schema=指定は libname オプションです。以下のような libname オプションが指定可能です。(前出の SQLSERVER エンジンのヘルプ画面から確認できます。)



さて、実行すると、ログに person と sales 2 個の「ライブラリ名が割り当てられた」というメッセージが表示されます。このメッセージは SQLSERVER エンジン経由で DB と接続できたことを表します。

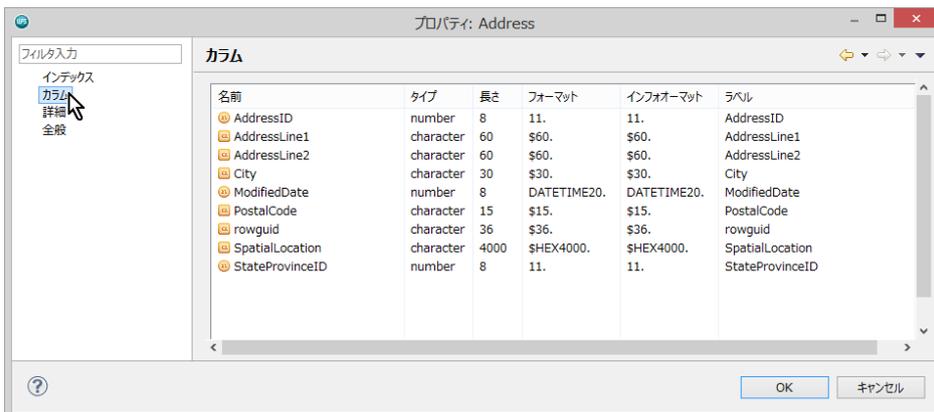
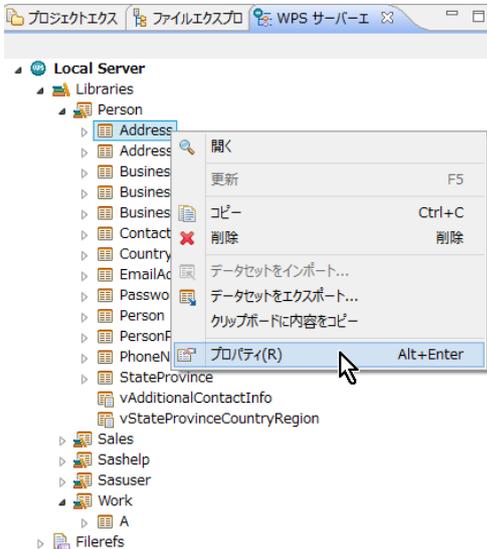


実行後、WPS サーバーエクスプローラービュー内の ローカルサーバー の下の Libraries を展開し、Person ライブラリと Sales ライブラリがあることを確認してください。これらの中に AdventureWorks2012 データベースの Person スキーマ、Sales スキーマに含まれるテーブル名およびビュー名がそれぞれ含まれていることがわかります。



※ 右矢印がついたアイテムはテーブル名（展開すると、テーブルに含まれる項目名と項目タイプを表すアイコンが表示されます。）、付いていないのはビュー名です。

これらのテーブルやビューは、WPS データセットに対する操作と同じファイル操作が可能です。例えば、以下のように、テーブル名を右クリックして出現するメニューの「プロパティ」を選択すると、カラム名などを確認することができます。

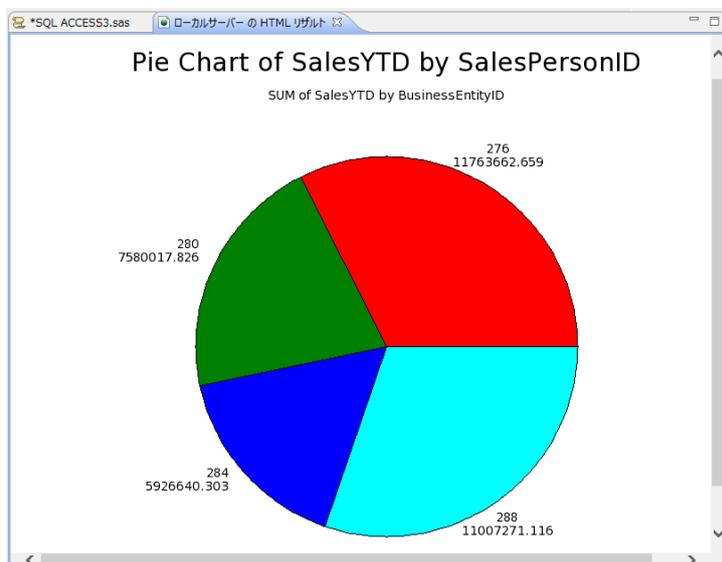


※ 注意：テーブル名をダブルクリックすると、データセットビューアが起動し、テーブルの中身を表示しようとしています。この操作は WPS セッションが動かなくなったり不意に終了してしまうなどの危険を伴いますので、できるだけ避けてください。

さて、SQLSERVER エンジンに libname ステートメントで指定した後は、以下のプログラムのように、DB テーブルの値を WPS データセットに変換することなく、直接プロシジャ入りに用いることもできます。この例では、Sales.Salesperson テーブルを入力としてビジネスエリア別の売り上げ高の円グラフを描いています。

```
goptions cback=white;
title "Pie Chart of SalesYTD by SalesPersonID";
```

```
proc gchart data=sales.salesperson;  
  pie businessentityid/sumvar=salesYTD;  
run;
```



※ 注意：グラフィック表示は WPS3.1 では日本語の表示はサポートされていません。

また、以下のように、SAS 言語の DATA ステップでも DB テーブルを読むことができるようになります。また、SQL プロシジャを使用して処理することも可能ですので、libname ステートメントを使って DB アクセスを行うときは、SAS 言語と SQL 言語いずれも使用可能になるというメリットがあります。

下の例は、Person.Address テーブルから 3 つの都市名のいずれかに該当するレコードを選択した WPS データセット WORK.address_subset を DATA ステップで作成しています。

```
data address_subset;  
  set person.address(keep=AddressID AddressLine1 City);  
  where City in ('San Francisco', 'Los Angeles', 'Seattle');  
run;
```

[WPS データセットの DB への出力方法]

以下のいずれかの方法で WPS データセットを SQL Server のテーブルへ出力できます。

- ① DBLOAD プロシジャ
- ② SQL プロシジャ
- ③ DATA ステップ

①の方法では DB への接続方法を DBLOAD プロシジャで指定しますので、libname ステートメントで SQLSRV エンジン指定する必要はありません。一方、②と③の方法では、事前に DB への接続に用いるエンジン名と接続先 (DB 名やスキーマ名) を libname ステートメントで指定しておく必要があります。

① DBLOAD プロシジャ

前節の最後のプログラムで Person.Address テーブルから 3 つの都市名のいずれかに該当するレコードを選択した WPS データセット WORK.address_subset を SQL Server の Person スキーマ内の Address_Subset テーブルへアップロードしてみましょう。以下のようにプログラムを書いて実行します。

```
proc dbload data=address_subset dbms=SQLSERVER;  
  server="PC3¥SQLEXPRESS";  
  database=AdventureWorks2012;  
  table=Person.Address_Subset;  
  load;  
run;quit;
```

※ 注意 : SQLSERVER ネイティブエンジンを使って DB にアクセスするための接続オプション (user=, password=, port=など) がさらに必要な場合は、指定を追加してください。

② SQL プロシジャによるテーブルへの読み書き

Libname ステートメントでエンジン名を指定すると、DB テーブルを WPS データセットと同じように扱えるようになります。

以下の例は AdventureWorks2012 データベースの Sales.Customer テーブルを讀んで、TerritoryID=1 に該当するレコードのみ抽出した全カラムを含むテーブル Sales.Customer_Territory_1 を作成しています。

```
libname sales SQLSERVER server="PC3¥SQLEXPRESS" database=AdventureWorks2012  
schema=Sales;  
proc sql;  
  create table Sales.Customer_Territory_1 as  
  select *  
  from Sales.Customer  
  where TerritoryID = 1;  
quit;
```

※ この例では、Sales ライブラリは SQL Server DB の Sales スキーマを意味するよう最初の libname ステートメントで定義しています。したがって、SQL プロシジャの中の Sales ライブラリは WPS データライブラリではなく、AdventureWorks2012 DB の Sales スキーマ

マを表すものと解釈され、読み取りも書き出しも DB テーブルが対象になります。

③ DATA ステップ

以下の例は②の SQL プロシジャを使った処理と同じことを SAS 言語の DATA ステッププログラムで行っています。

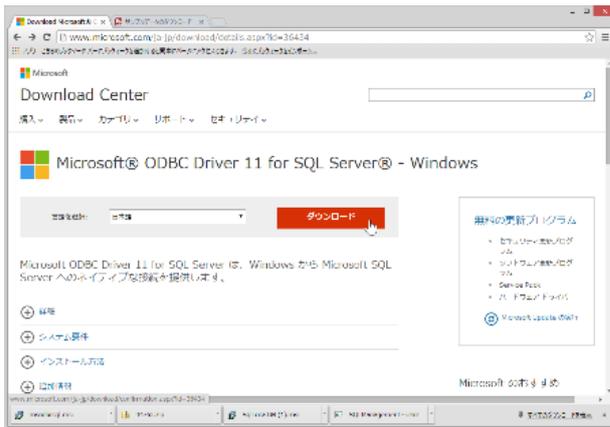
```
libname sales SQLSERVER server="DMT-PC3¥SQLEXPRESS"  
database=AdventureWorks2012 schema=Sales;  
/* 同じ名前のテーブルを一旦削除 */  
proc datasets lib=sales;  
  delete Customer_Territory_1;  
run;quit;  
/* DATAステップでDBテーブルを読み書き */  
data Sales.Customer_Territory_1;  
  set Sales.Customer;  
  where TerritoryID = 1;  
run;
```

4.2. ODBC ドライバー経由の DB テーブルの入出力

ODBC ドライバーは、Windows 環境や Linux 環境から各種 DB とのインタフェースを共通方式で行うためのソフトウェアです。ほとんどの DB の ODBC ドライバが DB ベンダやサードパーティから提供されており、ODBC ドライバの設定を行えば、DB ごとの細かいアクセス条件等の設定を気にせずに、共通の構文で DB アクセスが実現できるという利点があります。

[ODBC ドライバのインストール]

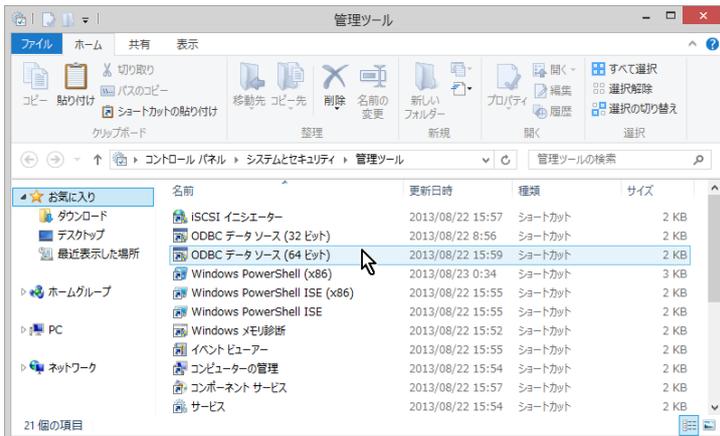
DB ベンダーのサイトから ODBC ドライバーをダウンロードし、インストールしておきます。ここでは、Microsoft のサイトから SQL Server の ODBC ドライバーをダウンロードしています。



[PC 側の ODBC ドライバーの設定]

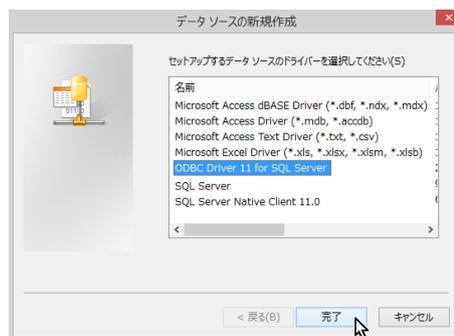
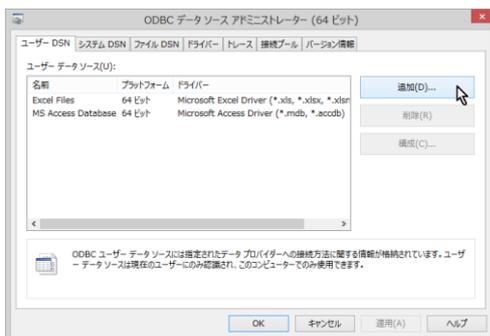
WPS が導入されている PC 側では、以下のような手順で、上記 SQL Server 2012 の AdventureWorks2012 データベースを ODBC データソースとして登録しておきます。

システムとセキュリティ → 管理ツールで ODBC データソースを選択します。



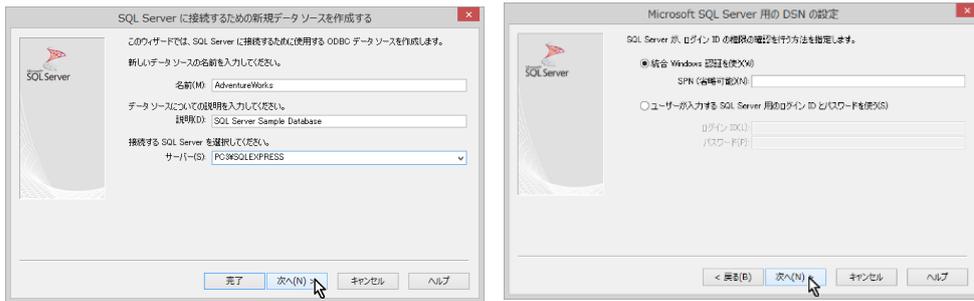
(左図) 「追加」をクリックします。

(右図) ダウンロードした SQL Server 用 ODBC ドライバーを選択し、「完了」を押します。



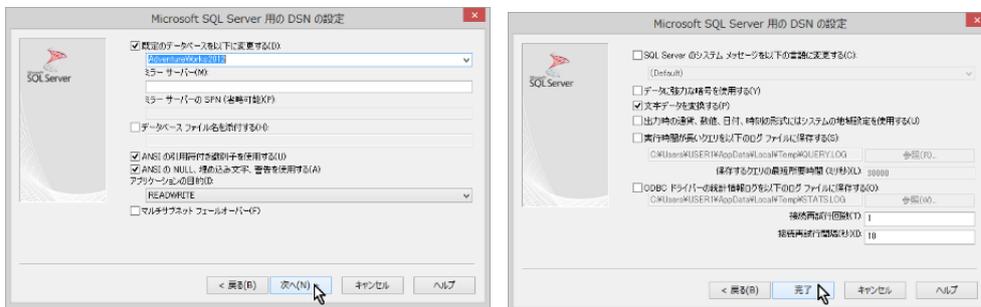
(左図) 新規データソースの名前と説明にはユーザーが自由に入力できます。ここではわかりやすく、名前に「AdventureWorks」、説明には「SQL Server Sample Database」と入力を行いました。「接続する SQL Server を選択してください」には、SQL Server のサーバー名を入力します。「次へ(N)」をクリックします。

(右図) ログイン方法の設定を確認して「次へ(N)」をクリックします。



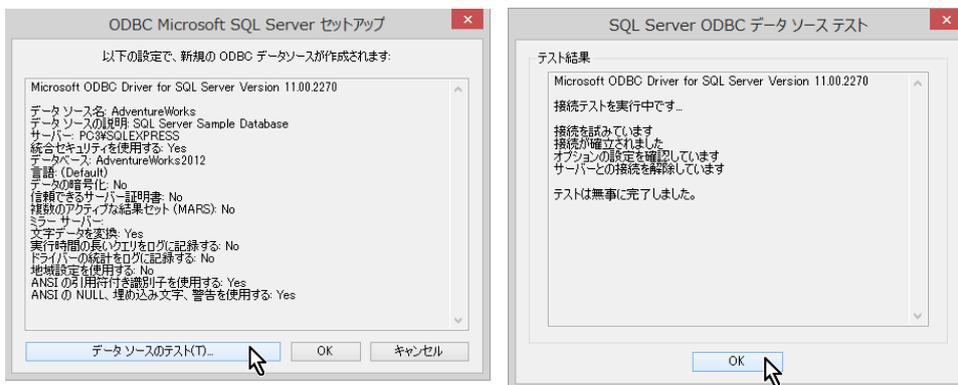
(左図) この画面で「既定のデータベースを以下に変更する」にチェックを入れて、アクセスしたいデータベース名(ここでは AdventureWorks2012)をプルダウンリストから選択し、「次へ(N)」をクリックします。

(右図) そのまま「完了」をクリックします。

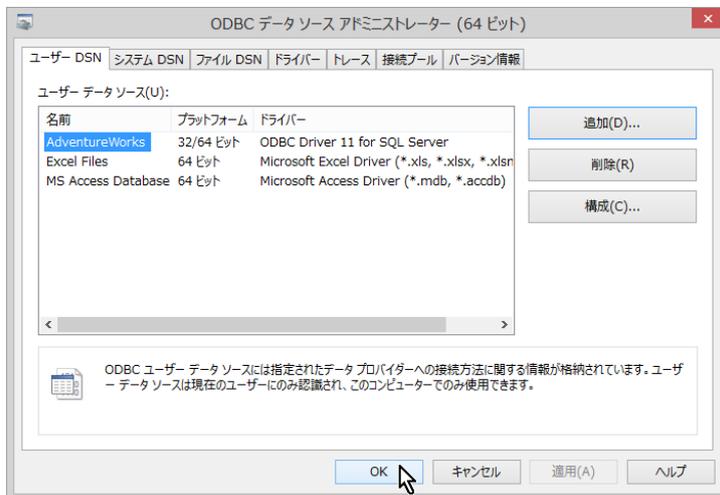


(左図) 「データソースのテスト」をクリックし接続テストを行います。

(右図) テスト結果を確認し、「OK」をクリックし、戻ります。



AdventureWorks が ODBC データソースに追加されたことを確認し、「OK」を押して設定終了です。



[SQL 言語を使用した DB 読み取り]

DB へのアクセスは SQL 言語を用いて行うことが多く、WPS でも SQL プロシジャを用いて SQL 言語による DB アクセスが可能です。

下記は、AdventureWorks2012 データベースの Person.Address テーブルの 3 つのカラム (AddressID, AddressLine1, City) を City の値が 'San Francisco' か 'Los Angeles' か 'Seattle' のいずれかに該当するレコードを抽出し、AddressID の小さい方から並べた WPS データセット WORK.A を作成する例となっています。

```
proc sql;
  connect to ODBC(datasrc=AdventureWorks);
  create table A as
  select *
  from connection to ODBC
  (
    select AddressID, AddressLine1, City
    from Person.Address
    where City in ('San Francisco', 'Los Angeles', 'Seattle')
    order by AddressID
  );
  disconnect from ODBC;
quit;
```

(WPS ワークベンチの実行ログ)

```

1 proc sql;
2   connect to ODBC(datasrc=AdventureWorks);
3   create table A as
4   select *
5   from connection to ODBC
6   (
7     select AddressID, AddressLine1, City
8     from Person.Address
9     where City in ('San Francisco', 'Los Angeles', 'Seattle')
10    order by AddressID
11  );
12  disconnect from ODBC;
13  quit;
14
15
16
17  quit; run;

```

```

4   proc sql;
5   connect to ODBC(datasrc=AdventureWorks);
NOTE: Successfully connected to database ODBC as alias ODBC.
6   create table A as
7   select *
8   from connection to ODBC
9   (
10    select AddressID, AddressLine1, City
11    from Person.Address
12    where City in ('San Francisco', 'Los Angeles', 'Seattle')
13    order by AddressID
14  );
NOTE: Data set "WORK.A" has 285 observation(s) and 3 variable(s)
15  disconnect from ODBC;
NOTE: Successfully disconnected from database ODBC.
16  quit;
NOTE: Procedure sql step took :
    real time : 0.609
    cpu time  : 0.140
17  quit; run;

```

WORK.A をダブルクリックして、作成された WORK.A データセットの内容を確認します。

AddressID	AddressLine1	City
1	23 6657 Sand Pointe Lane	Seattle
2	26 5725 Glaze Drive	San Francisco
3	91 7166 Brock Lane	Seattle
4	92 7126 Ending Ct.	Seattle
5	93 4598 Manila Avenue	Seattle
6	94 5666 Hazelnut Lane	Seattle
7	95 1220 Bradford Way	Seattle
8	96 5375 Clearland Circle	Seattle
9	97 2639 Anchor Court	Seattle
10	98 502 Alexander Pl.	Seattle
11	99 5802 Ampersand Drive	Seattle
12	100 5125 Cotton Ct.	Seattle
13	101 3243 Buckingham Dr.	Seattle
14	102 3029 Pastime Dr	Seattle
15	103 9537 Ridgewood Drive	Seattle
16	104 9964 North Ridge Drive	Seattle
17	105 1619 Stillman Court	Seattle
18	106 2144 San Rafael	Seattle
19	107 7403 N. Broadway	Seattle
20	108 7842 Ygnacio Valley Road	Seattle
21	109 874 Olivera Road	Seattle
22	110 1064 Slow Creek Road	Seattle
23	111 77 Birchwood	Seattle
24	112 7765 Sunshine Drive	Seattle
25	113 1102 Ravenwood	Seattle
26	114 1398 Yorba Linda	Seattle
27	115 4948 West 4th St	Seattle
28	116 8700 Marmaret Ct	Seattle

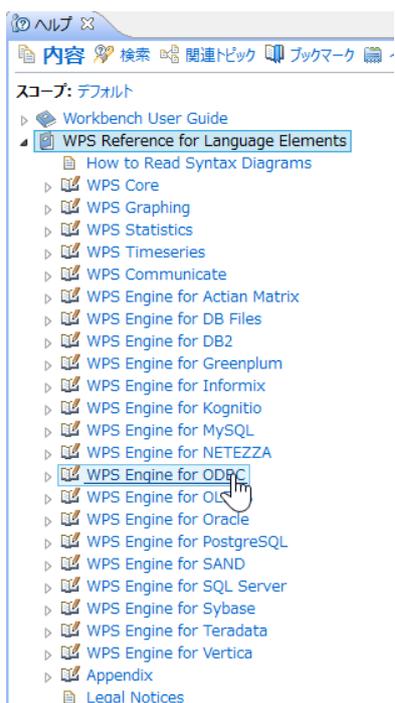
※ ODBC ドライバーによる DB へのアクセス制限等の設定によっては、SQL プロシジャの CONNECT 文の () 内のオプションに、その他の指定が必要となる場合があります。以下のようにヘルプビューで指定可能なオプションが確認できます。

ヘルプビューの左上にある「内容」をクリックします。

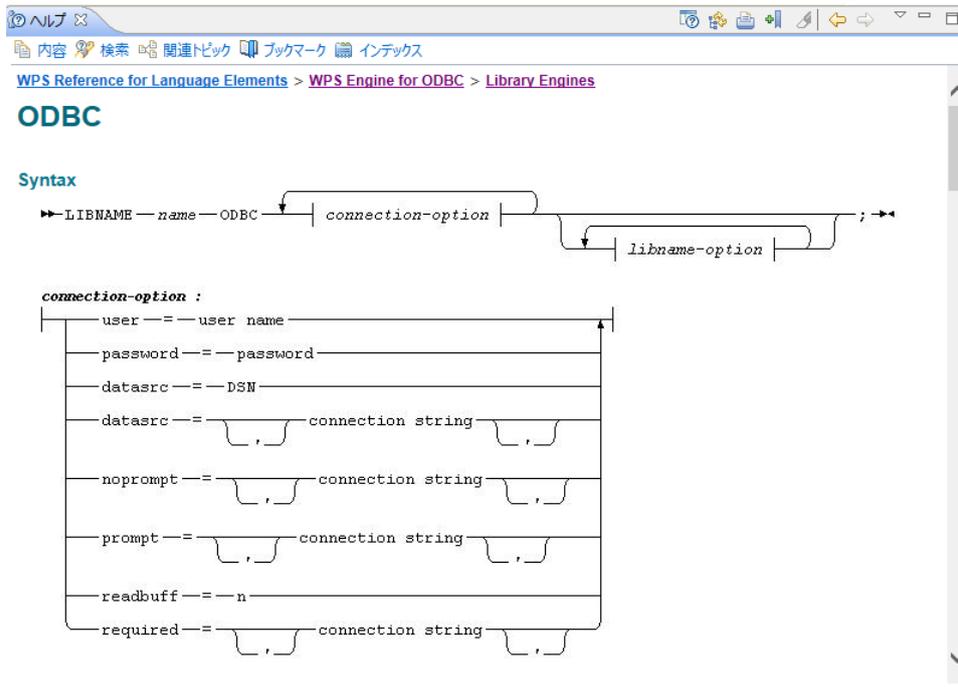
ヘルプ全体の目次が表示されます。WPS Reference for Language Elements をクリックします。



WPS レファレンスを展開し、WPS Engine for ODBC を選択します。



その後、展開を何度か行くと接続オプション (connection option) が表示されます。



これらのオプション (user= password= datasrc= noprompt= prompt= readbuff= required=) は ODBC ドライバー経由でのデータベースアクセスの際の CONNECT 文の DBMS オプション指定としても、また、次に説明する libname ステートメントによるデータベースアクセスの接続オプションとしても用いることができ、必要に応じて指定します。

[libname ステートメントによる DB 入力]

WPS のエディタから以下のプログラムを実行します。

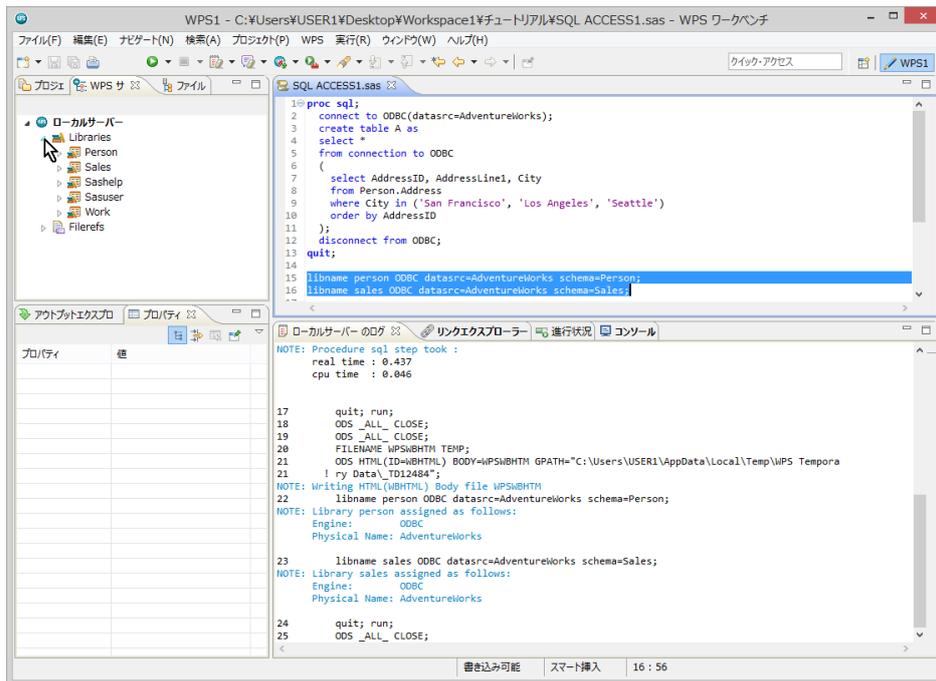
```
libname person ODBC datasrc=AdventureWorks schema=Person;
libname sales ODBC datasrc=AdventureWorks schema=Sales;
```

この指定は、WPS がデータソース AdventureWorks に ODBC ドライバー経由で接続し、Person スキーマをライブラリ名 person で、Sales スキーマのテーブルをライブラリ名 sales で、それぞれ参照できるようにする指定です。(入出力両方可能)

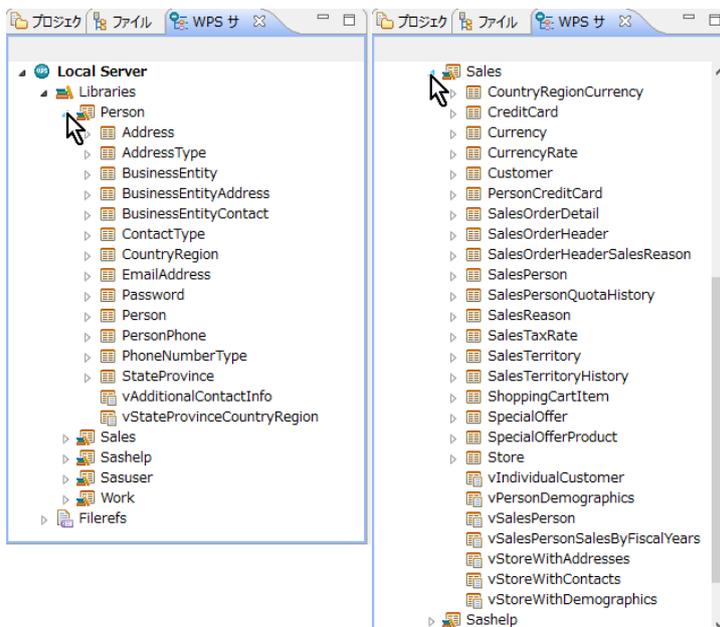
なお、datasrc=指定は DB 接続オプション、shema=指定は libname オプションです。

libname オプションは SQLSERVER エンジンの libname オプションと同じものが指定可能です。(前出の ODBC エンジンのヘルプ画面から確認できます。)

実行すると、ログに person と sales 2 個の「ライブラリ名が割り当てられた」というメッセージが表示されます。このメッセージは ODBC 経由で DB と接続できたことを表します。

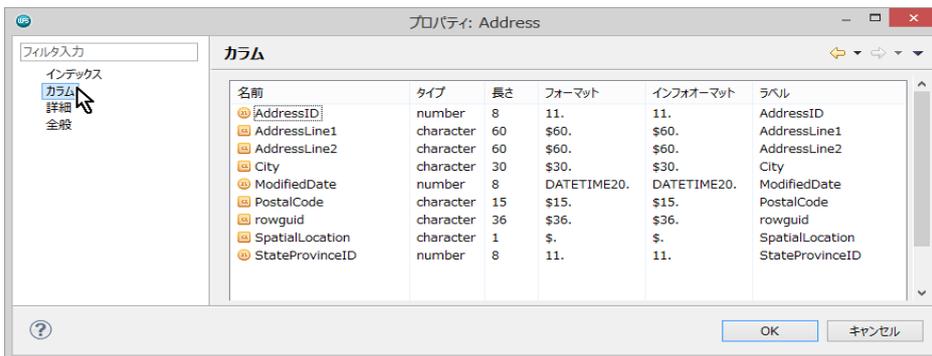
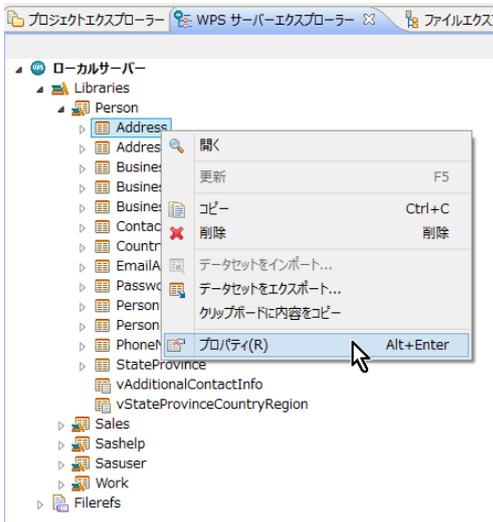


実行後、WPS サーバーエクスプローラービュー内の ローカルサーバー の下の Libraries を展開し、Person ライブラリと Sales ライブラリがあることを確認してください。これらの中に AdventureWorks2012 データベースの Person スキーマ、Sales スキーマに含まれるテーブル名およびビュー名がそれぞれ含まれていることがわかります。



※ 右矢印がついたアイテムはテーブル名（展開すると、テーブルに含まれる項目名と項目タイプを表すアイコンが表示されます。）、付いていないのはビュー名です。

これらのテーブルやビューは、WPS データセットに対する操作と同じファイル操作が可能です。例えば、以下のように、テーブル名を右クリックして出現するメニューの「プロパティ」を選択すると、カラム名などを確認することができます。

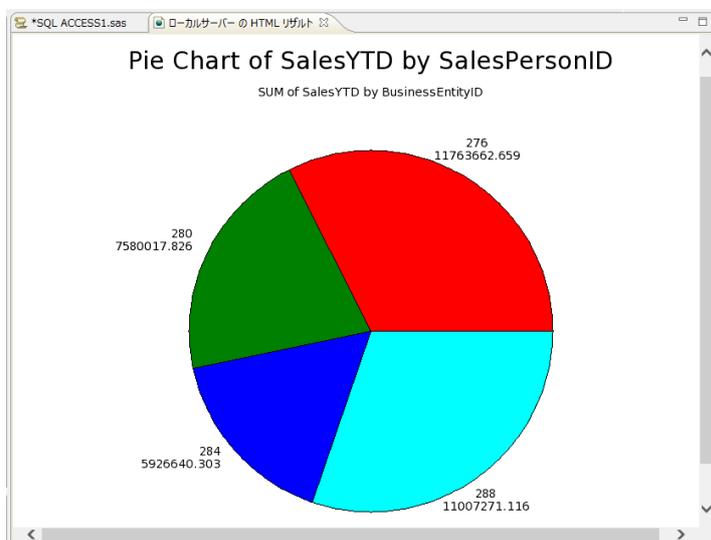


※ 注意：テーブル名をダブルクリックすると、データセットビューアが起動し、テーブルの中身を表示しようとします。この操作は WPS セッションが動かなくなったり不意に終了してしまうなどの危険を伴いますので、できるだけ避けてください。

さて、ODBC アクセスエンジンを libname ステートメントで指定すると、以下のプログラムのように、DB テーブルの値を WPS データセットに変換することなく、直接プロシジャ入力に用いることもできます。この例は、sales.salesperson テーブルを入力としてビジネスエリア別の売り上げ高の円グラフを描いています。

```
goptions cback=white;
title "Pie Chart of SalesYTD by SalesPersonID";
proc gchart data=sales.salesperson;
```

```
pie businessentityid/sumvar=salesYTD;  
run;
```



※ 注意：グラフィック表示は WPS3.1 では日本語の表示はサポートされていません。

また、以下のように、SAS 言語の DATA ステップでも DB テーブルを処理することができますようになります。また、SQL プロシジャを使用して処理することも可能ですので、libname ステートメントを使って DB アクセスを行うときは、SAS 言語と SQL 言語いずれも使用可能になるというメリットがあります。

下の例は、前の節で SQL プロシジャで作成した WORK.A と同じ内容のデータセット WORK.B を DATA ステップで作成しています。

```
data B;  
  set person.address(keep=AddressID AddressLine1 City);  
  where City in ('San Francisco', 'Los Angeles', 'Seattle');  
run;
```

[WPS データセットの DB への出力方法]

以下のいずれかの方法で WPS データセットを SQL Server のテーブルへ出力できます。

- ① DBLOAD プロシジャ
- ② SQL プロシジャ
- ③ DATA ステップ

①の方法では DB への接続方法を DBLOAD プロシジャで指定しますので、libname ステータス

トメントで ODBC エンジン指定する必要はありません。一方、②と③の方法では、事前に DB への接続方法を `libname` ステートメントで指定しておく必要があります。

① DBLOAD プロシジャ

前節の最後のプログラムで `Person.Address` テーブルから 3 つの都市名のいずれかに該当するレコードを選択した WPS データセット `WORK.A` を SQL Server の `Person` スキーマ内の `Address_Subset_A` テーブルへアップロードしてみましょう。以下のようにプログラムを書いて実行します。

```
proc dbload data=A dbms=ODBC;  
  datasrc=AdventureWorks;  
  table=Person.Address_Subset_A;  
  load;  
run;
```

※ 注意：ODBC エンジンを使って DB にアクセスするための接続オプション (`user=`, `password=`, `required=` など) がさらに必要な場合は、指定を追加してください。

② SQL プロシジャによるテーブルへの読み書き

`Libname` ステートメントで ODBC エンジン指定すると、DB テーブルを WPS データセットと同じように扱えるようになります。

以下の例は AdventureWorks2012 データベースの `Sales.Customer` テーブルを読んで、`TerritoryID=1` に該当するレコードのみ抽出した全カラムを含むテーブル `Sales.Customer_Territory_1` を作成しています。

```
libname sales ODBC datasrc=AdventureWorks schema=Sales;  
proc sql;  
  create table Sales.Customer_Territory_1 as  
  select *  
  from Sales.Customer  
  where TerritoryID = 1;  
quit;
```

※ この例では、`Sales` ライブラリは SQL Server DB の `Sales` スキーマを意味するよう最初の `libname` ステートメントで定義しています。したがって、SQL プロシジャの中の `Sales` ライブラリは WPS データライブラリではなく、AdventureWorks2012 DB の `Sales` スキーマを表すものと解釈され、読み取りも書き出しも DB テーブルが対象になります。

③ DATA ステップ

以下の例は②の SQL プロシジャを使った処理と同じことを SAS 言語の DATA ステッププログラムで行っています。

```
libname sales ODBC datasrc=AdventureWorks2012 schema=Sales;
/* 同じ名前のテーブルを一旦削除 */
proc datasets lib=sales;
  delete Customer_Territory_1;
run;quit;
/* DATAステップでDBテーブルを読み書き */
data Sales.Customer_Territory_1;
  set Sales.Customer;
  where TerritoryID = 1;
run;
```

以上で WPS による各種ファイル・データベース入出力方法の説明は終了です。

(第一版 変更履歴)

2014/7/17 バージョン 3.1.1 の WPS ワークベンチの表示に変更。一部の記述誤りを訂正。