ibi™ WebFOCUS®

ibi™ WebFOCUS® メタデータリファレンス

バージョン 9.0.0 以降 | December 2023

目次

1

データソース記述の理解

TIBCO WebFOCUS の製品には柔軟性のあるデータ記述言語が用意されています。この言語は、次のようなさまざまなタイプのデータソースに使用することができます。

- リレーショナル型 (例、DB2、Oracle、Sybase、Teradata)
- 階層型 (例、IMS、FOCUS、XFOCUS)
- ネットワーク型 (例、CA-IDMS)
- シーケンシャル型 (固定フォーマットおよびカンマ区切りフォーマット)
- マルチディメンション (多次元) 型 (例、Essbase)

また、このデータ記述言語および関連機能を使用して、次の操作を行うことができます。

- 異なるタイプのデータソースを結合して一時的なデータ構造を作成し、そのデータ構造に対してデータの読み取りをリクエストする。
- 複数のフィールドで構成されたサブセットを定義してユーザに提供する。
- データソースを論理的に再編成して異なる順序でデータにアクセスする。

トピックス

データソースの用語に関する注意

データソースのタイプが異なる場合、類似した概念をそれぞれ異なる用語で表現することがあります。たとえば、多くの階層型データベース管理システムおよびインデックスを使用したデータアクセスではデータの有意な最小単位を「フィールド」と呼びますが、リレーショナル型データベース管理システムでは「カラム」と呼びます。

共通した概念を異なる用語で表現する場合はほかにもあります。このマニュアルでは、説明を簡単にするために全体を通して標準的な用語を使用します。たとえば、データの有意な最小単位はデータソースのタイプに関係なく一般に「フィールド」と呼びます。ただし、意味を明確にする必要がある場合は、各データソースに固有の用語を使用します。このマニュアルでは、標準的な用語をはじめて使用する際に、用語の意味を定義し、他のデータソースタイプで使用される類似した用語と比較します。

データソース記述の概要

アプリケーションがデータソースにアクセスする場合、アクセス先のデータをアプリケーション自体が解釈する必要があります。アプリケーションが解釈する必要のある情報には次のものがあります。

- データ構造の概要。たとえば、リレーショナルデータ、階層データ、シーケンシャルデータなどのデータのタイプ。また、それぞれの構造でのデータの編成方法やインデックスの有無。
- 特定のデータ要素。たとえば、データソースに格納されているフィールド名や各フィールドのデータタイプ (例、文字、日付、整数)。

必要な情報を取得するために、アプリケーションは、「シノニム」とも呼ばれるデータソースの記述を読み取ります。シノニムの主コンポーネントはマスターファイルです。マスターファイルにはデータソースの構造およびその構造を構成するフィールドが記述されています。たとえば、フィールド名やデータタイプなどの情報があります。

データソースの中には、アクセスファイルを使用してマスターファイルを補足するものもあります。アクセスファイルには、データソース記述を補足する追加情報が記述されています。これには、データソースのフルネームやフルパス名などがあります。データソースを記述するには、マスターファイルが 1 つと、データソースによってはアクセスファイルが 1 つ必要です。

アプリケーションによるデータソース記述の使用

マスターファイルおよびアクセスファイルは、関係付けられたデータソースとは別の場所に格納されます。アプリケーションは、データソースのマスターファイルおよびアクセスファイルを使用して次のようにデータソースを解釈します。

1. リクエストで指定されたデータソースのマスターファイルを識別し、場所を特定してファイルを読み取ります。

   マスターファイルがメモリに存在する場合、アプリケーションはそのメモリイメージを使用してデータソースの場所の特定と読み取りを行います。

   マスターファイルがメモリに存在しない場合、アプリケーションは記憶装置上のマスターファイルの場所を特定してメモリにロードします。このとき、メモリに存在する既存のマスターファイルは上書きされます。

   マスターファイルが他のデータソースをクロスリファレンスセグメントとして参照する場合、またはマスターファイルで JOIN コマンドが有効な場合は、クロスリファレンスマスターファイルもメモリに読み込まれます。

2. マスターファイルのプロファイルが存在する (FILE 宣言に MFD_PROFILE 属性が記述されている) 場合、そのプロファイルが実行されます。
3. データソースにマスターファールデータソースセキュリティ (DBA) が設定されている場合は、そのセキュリティルールを読み取り、指定された DBA セキュリティに基づいてユーザのアクセス権限を確認します。
4. リクエストで指定されたデータソースにアクセスファイルが必要な場合は、アクセスファイルの場所を特定してファイルを読み取ります。
5. データソースの場所を特定して、データを読み取ります。

   データソースの内容は、マスターファイルおよびアクセスファイルの情報に基づいて解釈されます。

注意：ENCRYPT コマンドを使用して、マスターファイルを暗号化することができます。詳細は、362 ページの 「 制限規則の非表示 - ENCRYPT コマンド 」 を参照してください。ただし、暗号化するマスターファイルの 1 行目は、68 バイト以下にする必要があります。68 バイトを超える場合、複数行に分割する必要があります。

マスターファイルに記述する内容

マスターファイルを使用して次のことを実行することができます。

- ❏ データソースの名前およびタイプを識別する。
- ❏ フィールドグループを識別して関係付ける。

- 個々のフィールドを記述する。

注意

- マスターファイルには、セグメント宣言およびフィールド宣言をそれぞれ 1 つ以上指定する必要があります。 必須の属性に値を割り当てない場合は、プレースホルダとしてカンマ (,) を使用する必要があります。
- 構造でサポートされるセグメントの最大数は 1024 個です。この構造は、JOIN または COMBINE でデータソースを結合することにより、作成できます。

  単一 FOCUS データソースのセグメントの最大数は 64 個です。XFOCUS データソースのセグメントの最大数は 512 個です。単一 FOCUS データソースのセグメントとインデックスの合計は、最大 191 個です。
- フィールドすべての合計の長さは最大 256 キロバイトです。

データソースの識別

アプリケーションがデータを解釈するには、データソースを識別する名前およびタイプをアプリケーション自体が認識する必要があります。データソースのタイプには、たとえば DB2、Oracle、FOCUS などがあります。

詳細は、23 ページの 「 データソースの識別 」 を参照してください。

フィールドグループの識別と関係付け

マスターファイルでは、1 対 1 の関係が成立するフィールドグループを識別して互いに関係付けることができます。このグループは、マスターファイルでは「セグメント」、リレーショナルでは「テーブル」と呼ばれます。

データソースは、他のデータソースと結合することができます。データソースのタイプが同一の場合はマスターファイルまたは JOIN コマンドを使用し、タイプが異なる場合は JOIN コマンドを使用します。たとえば、2 つの DB2 データソースを FOCUS データソースに結合することも可能です。

フィールドグループの定義およびグループ間の関係についての詳細は、55 ページの 「 フィールドグループの記述 」 を参照してください。

フィールドの記述

各フィールドにはさまざまな特性があり、これらの特性をマスターファイルに記述する必要があります。たとえば、データタイプ、データ長、データスケールなどがあります。また、マスターファイルにはオプションのフィールド特性を指定することもできます。たとえば、マスターファイルでミッシング値を許可するフィールドを指定したり、特定のフィールドに関する説明情報を追加したりすることができます。

通常、マスターファイルにはデータソースのすべてのフィールドを記述します。ただし、場合によってはデータソースの特定のフィールドで構成された論理ビューをサブセットとして作成し、マスターファイルにこれらのフィールドのみを記述することもできます。

詳細は、97 ページの 「 フィールドの記述 」 を参照してください。

データソース記述の作成

データソースのマスターファイルおよびアクセスファイルはさまざまな方法で作成することができます。これらの作成方法は、既存のデータソース記述の有無により異なります。

- 既存のデータソース記述が存在する場合。たとえば、ネイティブのスキーマまたはカタログなど。この場合は、特定のツールを使用して既存のデータソース記述からマスターファイルおよびアクセスファイルを自動的に作成することができます。
- 既存のデータソース記述が存在しない場合。マスターファイルおよびアクセスファイルを作成するには、WebFOCUS のデータソース記述言語を使用してコーディングし、任意のテキストエディタで属性を指定します。

テキストエディタによるマスターファイルとアクセスファイルの作成

マスターファイルおよびアクセスファイルは次の方法で作成することができます。

- **コーディング**　テキストエディタを使用してコーディングします。すべての WebFOCUS 製品で、この方法を使用することができます。マスターファイルの構文に必要な情報はこのマニュアルに記載されています。アクセスファイルの構文についての詳細は、データアダプタのマニュアルまたは 231 ページの 「 FOCUS データソースの記述 」 を参照してください。

  マスターファイルを編集した場合は、CHECK FILE コマンドを発行して新しいマスターファイルを検証し、セッションイメージを更新する必要があります。

- **属性の指定**　シノニムエディタを使用して、属性を指定します。

マスターファイルの内容

マスターファイルでは、次の宣言を使用してデータソースを記述します。

- データソース宣言
- データソース内の各セグメントのセグメント宣言
- セグメント内の各フィールドのフィールド宣言
- オプションとしてマスターファイルに追加可能なその他のオブジェクト宣言 (例、FILTER、DEFINE、COMPUTE、SORTOBJ 定義)

データソースのタイプにより詳細は異なりますが、アクセスファイルでの指定方法もこれに類似しています。アクセスファイルの必要性の有無および必要なアクセスファイル属性については、対応するアダプタのマニュアルを参照してください。

構文　宣言の指定

各宣言には、一連の属性を次の形式で指定します。

```
attribute = value, attribute = value, ... ,$
```

説明

`attribute`

ファイル、セグメント、フィールドのプロパティを識別するマスターファイルのキーワードです。マスターファイルのすべての属性は、属性のフルネーム、エイリアス、一意のフィールド短縮名のいずれかで指定することができます。たとえば、属性のフルネームである FILENAME を使用したり、短縮名の FILE を使用したりすることができます。

`value`

属性の値です。

属性の割り当てごとにカンマ (,) を追加し、ドル記号 ($) でフィールド宣言を終了します。データソース宣言およびセグメント宣言では、宣言の末尾に使用するカンマ (,) およびドル記号 ($) は必須ではありません。

各宣言は、必ず新しい行から開始します。また、1 つの宣言を複数の行に拡張して記述することができます。1 つの宣言で属性の割り当てごとに改行したり、複数の属性または宣言全体をまとめて 1 行に記述したりすることができます。

- データソース宣言についての詳細は、23 ページの 「 データソースの識別 」 を参照してください。

- セグメント宣言についての詳細は、55 ページの 「 フィールドグループの記述 」 を参照してください。
- フィールド宣言についての詳細は、97 ページの 「 フィールドの記述 」 を参照してください。

注意：マスターファイルでは、すべての属性の名前に英数文字を使用する必要がありますが、属性の値にはサポートされている任意の言語を使用することができます。

可読性の向上

各属性の割り当ては任意の位置から開始します。宣言内の要素と要素の間にはブランクを使用することができます。ブランクを使用してセグメント宣言またはフィールド宣言を字下げすると、マスターファイルの可読性が向上します。テキストの配置場所を調整する場合は、タブ記号ではなくブランクを使用します。

宣言と宣言の間にブランク行を追加することもできます。ブランクおよびブランク行の使用は必須ではありません。また、使用してもアプリケーションでは無視されます。

例　ブランクおよびブランク行による可読性の向上

次の宣言は、宣言内および宣言間にブランクおよびブランク行を追加して可読性を向上させた例を示しています。

```
SEGNAME=EMPINFO, SEGTYPE=S1 ,$
  FIELDNAME=EMP_ID, ALIAS=EID, USAGE=A9 ,$

SEGNAME=EMPINFO, SEGTYPE=S1 ,$
  FIELDNAME = EMP_ID, ALIAS = EID, USAGE = A9 ,$

SEGNAME=EMPINFO,SEGTYPE=S1,$
  FIELDNAME = EMP_ID, ALIAS = EID, USAGE = A9 ,$
```

例　宣言の複数行記述による可読性の向上

次の宣言は、フィールド宣言を複数行に拡張して記述した例を示しています。

```
FIELDNAME = MEMBERSHIP, ALIAS = BELONGS, USAGE = A1, MISSING = ON,
  DESCRIPTION = This field indicates the applicant's membership status,
  ACCEPT = Y OR N, FIELDTYPE = I,
  HELPMESSAGE = 'Please enter Y for Yes or N for No' ,$
```

コメントの追加

次の方法で任意の宣言にコメントを追加することができます。

- 宣言の行で、終了を表すドル記号 ($) の後にコメントを入力する。

- ❏ 行の先頭にドル記号 ($) を配置してコメント専用の行を作成する。

宣言がまだ終了していない位置にコメント行を追加すると、1 つ前の宣言がそこで終了します。ドル記号 ($) の後に続く記述はすべて無視されます。

ドル記号 ($) の後に追加したコメントは、マスターファイルのソースコードを閲覧するユーザにのみ役立ちます。このコメントは、グラフィカルツールには表示されません。REMARKS 属性または DESCRIPTION 属性を使用してグラフィカルツールに説明を表示する方法についての詳細は、23 ページの 「 データソースの識別 」 および 97 ページの 「 フィールドの記述 」 を参照してください。

例　マスターファイルでのコメントの追加

次の例では、2 つのコメントが追加されています。最初のコメントはデータソース宣言のドル記号 ($) の後に入力されています。2 つ目のコメントはデータソース宣言の後のコメント専用行に入力されています。

```
FILENAME = EMPLOYEE, SUFFIX = FOC ,$ This is the personnel data source.
$ This data source tracks employee salaries and raises.
SEGNAME = EMPINFO, SEGTYPE = S1 ,$
```

マスターファイルの編集と確認

マスターファイルを手動で作成または編集した場合、CHECK FILE コマンドを発行してファイルを確認する必要があります。CHECK FILE コマンドは、新規または編集済みのマスターファイルをメモリに読み込んで、マスターファイル内のエラーを表示します。これにより、データソースを読み取る前にエラーを修正することができます。

CHECK FILE PICTURE コマンドは、データソースの構造を図示します。また、このコマンドを使用してマスターファイル内の情報を表示することもできます。たとえば、セグメントおよびフィールドの名前、特定のデータソースに対してリクエストを実行した際に取得するデータの順序などの情報があります。

詳細は、319 ページの 「 マスターファイルの確認と変更 - CHECK 」 を参照してください。

2

データソースの識別

アプリケーションがデータを解釈するには、データソースを識別する名前およびタイプをアプリケーション自体が認識する必要があります。データソースのタイプには、たとえば DB2、Teradata、FOCUS などがあります。

トピックス

データソースの識別 - 概要

データソースの名前およびタイプは、マスターファイルのデータソース宣言で識別します。データソース宣言には次の属性を含めることができます。

- FILENAME 属性 - データソースの名前を識別します。
- SUFFIX 属性 - データソースのタイプを識別します。
- REMARKS 属性 - データソースに関する説明情報を指定します。この情報は、グラフィカルツールに表示されます。
- ACCESSFILE 属性 - FOCUS データソースに使用するオプションのアクセスファイルの名前を識別します。詳細は、231 ページの 「 FOCUS データソースの記述 」 を参照してください。

- DATASET 属性 - データソースの名前が標準外の場合に物理ファイル名を識別します。

マスターファイルの処理中に実行するマスターファイルプロファイル (MFD_PROFILE) プロシジャを識別することができます。詳細は、35 ページの 「 マスターファイルプロファイルの作成および使用 」 を参照してください。

必要に応じて、次のデータソース属性を使用して日付の変移枠を指定し、下 2 桁の西暦年で格納された日付に上 2 桁の西暦年の値を割り当てることができます。

- FDEFCENT 属性 - 上 2 桁の西暦年を識別します。
- FYRTHRESH 属性 - 下 2 桁の西暦年を識別します。

データソース名の指定 - FILENAME

マスターファイルに FILENAME 属性を記述してデータソースの名前を指定します。これは、マスターファイルで最初に指定する属性です。FILENAME 属性の省略名である FILE を使用することもできます。

注意：ENCRYPT コマンドを使用して、マスターファイルを暗号化することができます。詳細は、362 ページの 「 制限規則の非表示 - ENCRYPT コマンド 」 を参照してください。ただし、暗号化するマスターファイルの 1 行目は、68 バイト以下にする必要があります。68 バイトを超える場合、複数行に分割する必要があります。

構文　　データソース名の指定

```
FILE[NAME] = data_source_name
```

説明

`data_source_name`
マスターファイルで記述するデータソースの名前です。名前の最大長は 64 バイトです。

ファイル名には 1 つ以上の文字を含める必要があります。それ以外は、文字、数字、アンダースコア (_) を任意に組み合わせて使用することができます。

例　　データソース名の指定

次の例では、「EMPLOYEE」という名前のデータソースを指定しています。

```
FILENAME = EMPLOYEE
```

データソースタイプの識別 - SUFFIX

SUFFIX 属性を使用してデータソースのタイプを識別します。たとえば、DB2 や FOCUS などのデータソースがあります。データソースへのアクセスに使用するデータアダプタは、この SUFFIX 属性の値に基づいて選択されます。

SUFFIX 属性はほとんどのデータソースタイプに必要です。この属性は、固定フォーマットのシーケンシャルデータソースではオプションとして指定します。ただし、JOIN コマンドで固定フォーマットのシーケンシャルデータソースを参照する場合は、マスターファイルで SUFFIX 属性を宣言する必要があります。

構文　データソースタイプの識別

```
SUFFIX = data_source_type
```

説明

`data_source_type`
データソースのタイプまたはカスタマイズしたデータアクセスモジュールの名前を指定します。デフォルト値は FIX です。これは、固定フォーマットのシーケンシャルデータソースを表します。

例　FOCUS データソースタイプの指定

次の例では、データソースタイプとして FOC を指定しています。これは、4 キロバイトのデータベースページを持つ FOCUS データソースです。

```
SUFFIX = FOC
```

次の例では、データソースタイプとして XFOCUS を指定しています。これは、16 キロバイトのデータベースページを持つ XFOCUS データソースです。

```
SUFFIX = XFOCUS
```

参照　SUFFIX 値

下表は、サポートされているデータソースタイプの SUFFIX 値を示しています。

データソースタイプ	SUFFIX 値
ADABAS	`ADBSIN or ADBSINX`
ALLBASE/SQL	`SQLALB or ALLBASE`

データソースタイプ	SUFFIX 値
CA-Datacom/DB	DATACOM
CA-IDMS/DB	IDMSR
CA-IDMS/SQL	SQLIDMS
C-ISAM	C-ISAM
DB2	DB2 or SQLDS
DB2/2	SQLDBM
DB2/400	SQL400 (SQL アクセス) DBFILE (ネイティブアクセス)
DB2/6000	DB2
DBMS	DBMS
DMS	FIX (キーなしアクセス)
Digital Standard MUMPS	DSM
Enscribe	ENSC
固定フォーマットのシーケンシャル	FIX - これがデフォルト値です。 PRIVATE (FOCSAM ユーザイグジット用)
FOCUS	FOC
自由フォーマットのシーケンシャル (カンマ区切り)	COM, COMMA, COMT
Image/SQL	IMAGE
IMS	IMS
INFOAccess	SQLIAX

データソースタイプ	SUFFIX 値
Information/Management	INFOMAN
Informix	SQLINF
Ingres	SQLING
KSAM	KSAM
Micronetics Standard MUMPS	MSM
Model 204	M204IN
Native Interface	SQLSAP
NETISAM	C-ISAM
NOMAD	NMDIN
NonStop SQL	NSSQL
Nucleus	SQLNUC
ODBC	SQLODBC
OpenIngres	SQLING
Open M/SQL	SQLMSQ
Oracle	SQLORA
PACE	VSAM
Progress	SQLPRO
Rdb	SQLRDB
Red Brick	SQLRED

データソースタイプ	SUFFIX 値
RMS	RMS
SQL/DS	SQLDS
SQL Server	SQLMSS
StorHouse	SQLSTH
Sybase	SQLSYB
タブ区切り	TABT, TAB
Teradata	SQLDBC
トークン区切り	DFIX
TurboIMAGE	IMAGE OMNIDEX (OMNIDEX IMS インデックスの使用)
Unify	SQLUNIFY
uniVerse	UNIVERSE
標準外のデータソース	カスタマイズしたデータアクセスルーチンの名前

マスターファイルでのコードページの指定

マスターファイルの FILE 宣言では、このファイル内でのデータの取得に使用可能なコードページを指定することができます。ファイルがクラスタ JOIN のセグメントとして追加される場合、コードページを指定しておくことで、クラスタ内での FILE 宣言で異なるコードページが指定されている場合や、デフォルトコードページが使用される場合でも、このセグメントのデータが正しいコードページで取得されることが確実になります。

構文　マスターファイルでのコードページの指定

CODEPAGE = codepage

説明

CODEPAGE
ファイルの読み取り時に使用するコードページです。

バイトオーダーの指定

数値データを格納する際に、最上位バイトから順に格納するか、最下位バイトから順に格納するかは、オペレーティング環境によって異なります。最上位バイトから順に格納する方法は「ビッグエンディアン (BE)」、最下位バイトから順に格納する方法は「スモールエンディアン (SE)」と呼ばれます。

異なるオペレーティング環境からデータを読み取る際は、マスターファイルでバイトオーダーの指定が必要になる場合があります。

構文　バイトオーダーの指定

```
BYTEORDER={BE|SE}
```

説明

BE
最上位バイトから順に格納されていることを示します。この順序を使用するハードウェアには、IBM zSeries、POWER があります。

SE
最下位バイトから順に格納されていることを示します。この順序は「逆バイト」とも呼ばれます。この順序を使用するハードウェアには、Intel x86、x86-64、DEC Alpha があります。

データタイプの指定 - IOTYPE

IOTYPE は、HOLD ファイルの作成時に生成され、FILEDEF または割り当てに LRECL/RECFM 情報が存在しない場合に、データファイルの読み取り方法を WebFOCUS に指示します。これにより、HOLD ファイルの作成時にファイルに格納されたデータのタイプが識別されます。この指定は、HOLD コマンドの結果として自動的に生成されるマスターファイルに対してのみ使用することをお勧めします。次のいずれかの値です。

- BINARY - 数値データは、生成されたファイルにバイナリフォーマットで格納されます。
- STREAM - 生成されたファイルのデータはすべて、文字フォーマットで格納されます。

IOTYPE は、次の HOLD フォーマットにのみ適用されます。

FORMAT	SUFFIX	IOTYPE
ALPHA	FIX	STREAM
BINARY	FIX	BINARY
INTERNAL	FIX	BINARY
フォーマットなし	FIX	HOLDFORMAT パラメータに応じて異なります。 ❑ HOLDFORMAT が BINARY の場合、IOTYPE は BINARY です。 ❑ HOLDFORMAT が ALPHA の場合、IOTYPE は STREAM です。

データソース説明情報の追加 - REMARKS

オプションの REMARKS 属性を使用して、データソースに関する説明情報を追加します。この説明情報は、グラフィカルツールに表示されます。

説明情報は、マスターファイル内でドル記号 ($) の後にコメントとして追加することもできます。詳細は、15 ページの 「 データソース記述の理解 」 を参照してください。ただし、コメントとして追加した説明情報はマスターファイルのソースコードを閲覧するユーザにのみ役立ちます。このコメントは、グラフィカルツールには表示されません。

マスターファイルの REMARKS 属性および DESCRIPTION 属性の値には複数の言語がサポートされます。詳細は、196 ページの 「 マスターファイルでの多言語説明の使用 」 を参照してください。

構文　データソース説明情報の追加

```
REMARKS = 'descriptive_text'
```

説明

`descriptive_text`

データソースに関する説明情報です。テキストの最大長は 2 キロバイトです。テキストは一重引用符 (') で囲む必要があります。

マスターファイルに記述する説明テキストは 1 行以内に収める必要があります。必要に応じて、REMARKS 属性のみを新しい行に記述します。説明テキストが長くなる場合は、宣言を複数の行に分割し、それぞれの行に説明テキストのみを記述します。

例　Oracle テーブルの説明情報の追加

次の例は、「ORDERS」という Oracle テーブルのデータソース宣言を示しています。このデータソース宣言には、テーブルに関する説明情報が追加されています。

```
FILENAME=ORDERS, SUFFIX=SQLORA,
REMARKS='This Oracle table tracks daily, weekly, and monthly orders.' ,$
```

この例では、データソース属性と説明情報が 1 行に収まらないため、REMARKS 属性のみを新しい行に記述しています。

物理ファイル名の指定 - DATASET

マスターファイルに DATASET 属性を追加して、割り当てるデータソースの物理パスを指定することができます。また、DATASET 属性を使用すると、データソースのデフォルトパスの検索メカニズムをバイパスすることができます。DATASET 属性を使用した場合は、JCL、FILEDEF、DYNAM、USE コマンドでデータソースを割り当てる必要がなくなります。

UNIX、Windows でのユーザ割り当てコマンドは FILEDEF です。

DATASET を HOLD コマンドに含めることもできます。その場合、DATASET をオプションの 1 つとして追加します。HOLD コマンドで生成されたマスターファイルには、ファイルレベルの DATASET 属性が含められます。HOLD コマンドについての詳細は、『TIBCO WebFOCUS Language リファレンス』を参照してください。

注意：同一のマスターファイルで ACCESSFILE 属性と DATASET 属性を同時に使用することはできません。ACCESSFILE 属性についての詳細は、231 ページの 「 FOCUS データソースの記述 」 を参照してください。

FOCUS データソースでの DATASET の動作

DATASET 属性は、FOCUS (XFOCUS を含む)、固定フォーマットシーケンシャルマスターファイルのファイルレベルで使用することができます。DATASET 属性は、FOCUS マスターファイルのセグメントレベルで使用することができます。DATASET 属性を FOCUS マスターファイルのセグメントレベルで指定する方法についての詳細は、231 ページの 「 FOCUS データソースの記述 」 を参照してください。

USE リストにマスターファイル名が存在する場合、またはユーザが明示的にデータファイルを割り当てた場合、DATASET 属性は無視されます。

データソースが FOCUS Database Server で管理される場合、TABLE リクエストを処理する際に、FOCUS Database Server がマスターファイルを読み取らないため、マスターファイルに DATASET 属性を使用すると、サーバ側では DATASET 属性は無視されます。

マスターファイルで指定する DATASET 属性は、最も低い優先度で処理されます。

- 最初に割り当てコマンドを発行し、次に CHECK FILE コマンドを発行してこれまでの DATASET 割り当てをクリアした場合、ユーザの明示的な割り当てにより DATASET 属性が上書きされます。
- FOCUS データソースに USE コマンドを使用した場合、DATASET 属性および明示的な割り当てが上書きされます。

FOCUS データソースの割り当てに DATASET 属性を使用する代わりに、アクセスファイルを使用する方法もあります。詳細は、231 ページの 「 FOCUS データソースの記述 」 を参照してください。

注意：DATASET 属性の割り当てが有効な場合、その割り当てを明示的な割り当てコマンドで上書きするには CHECK FILE コマンドを発行する必要があります。CHECK FILE コマンドは、DATASET 属性の割り当てを解除します。

構文　ファイルレベルでの DATASET 属性の使用

```
{DATASET|DATA}='filename [ON sinkname]'
```

UNIX での構文は次のとおりです。

```
{DATASET|DATA}='/filesystem/filename.foc [ON sinkname]'
```

Windows での構文は次のとおりです。

```
{DATASET|DATA}='drive:¥directory¥filename.foc [ON sinkname]'
```

説明

`filename`
プラットフォームに依存するデータソースの物理名です。

`sinkname`
データソースが FOCUS Database Server に存在することを示します。この属性は FOCUS または XFOCUS データソースでのみ有効です。

例　DATASET 属性による FOCUS データソースの割り当て

次の例は、DATASET 属性を使用して FOCUS データソースを割り当てる方法を示しています。

UNIX の場合

```
FILENAME=CAR,SUFFIX=FOC,
DATASET='/filesystem/filename.foc'
SEGNAME=ORIGIN,SEGTYPE=S1
FIELDNAME=COUNTRY,COUNTRY,A10,FIELDTYPE=I,$
SEGNAME=COMP,SEGTYPE=S1,PARENT=ORIGIN
FIELDNAME=CAR,CARS,A16,$
SEGNAME=CARREC,SEGTYPE=S1,PARENT=COMP
.
.
.
```

Windows の場合

```
FILENAME=CAR,SUFFIX=FOC,
DATASET='drive:¥directory¥filename.foc'
SEGNAME=ORIGIN,SEGTYPE=S1
FIELDNAME=COUNTRY,COUNTRY,A10,FIELDTYPE=I,$
SEGNAME=COMP,SEGTYPE=S1,PARENT=ORIGIN
FIELDNAME=CAR,CARS,A16,$
SEGNAME=CARREC,SEGTYPE=S1,PARENT=COMP
.
.
.
```

例　FOCUS Database Server でのデータソースの割り当て

次の例は、DATASET 属性に ON sink を使用して FOCUS データソースを割り当てる方法を示しています。

UNIX の場合

```
FILENAME=CAR,SUFFIX=FOC, DATASET='filename ON sink'
SEGNAME=ORIGIN,SEGTYPE=S1
FIELDNAME=COUNTRY,COUNTRY,A10,FIELDTYPE=I,$
SEGNAME=COMP,SEGTYPE=S1,PARENT=ORIGIN
FIELDNAME=CAR,CARS,A16,$
SEGNAME=CARREC,SEGTYPE=S1,PARENT=COMP
.
.
.
```

Windows の場合

```
FILENAME=CAR,SUFFIX=FOC,
DATASET='filename ON sink'
SEGNAME=ORIGIN,SEGTYPE=S1
FIELDNAME=COUNTRY,COUNTRY,A10,FIELDTYPE=I,$
SEGNAME=COMP,SEGTYPE=S1,PARENT=ORIGIN
FIELDNAME=CAR,CARS,A16,$
SEGNAME=CARREC,SEGTYPE=S1,PARENT=COMP
.
.
.
```

固定フォーマットシーケンシャルデータソースでの DATASET の動作

固定フォーマットシーケンシャルファイルの DATASET 属性は、マスターファイルのファイル宣言レベルでのみ使用することができます。ただし、ON sink を含めることはできません。DATASET 属性に ON sink を含めると、処理が終了してメッセージが発行されます。

DATASET 属性が検出されると、そのマスターファイルの明示的なデータ割り当てが確認されます。明示的な割り当てが存在する場合、DATASET 属性は無視されます。このマスターファイルの名前が割り当てられていない場合は、割り当てを実行するための内部コマンドが発行されます。この割り当ては一時的に保存され、新しいマスターファイルを使用するか、セッションを終了した際にこの割り当てが解除されます。

DATASET 属性には、UNIX、Windows プラットフォームで必要な 2 つ目のパラメータがあります。このパラメータは SUFFIX=FIX データソースに使用して、データソースに格納されたデータのバイナリ、テキストのいずれかを識別します。このパラメータは、データソース内に改行文字が存在するかどうかを表す情報を提供します。データソースのバイナリとテキストはこの方法で区別します。デフォルト値はバイナリです。

構文　固定フォーマットデータソースでの DATASET 属性の使用

```
{DATASET|DATA}='filename {BINARY|TEXT}'
```

説明

`filename`
　プラットフォームに依存するデータソースの物理名です。

`BINARY`
　バイナリデータソースであることを示します。デフォルト値は BINARY です。

`TEXT`
　テキストデータソースであることを示します。

マスターファイルで指定する DATASET 属性は、最も低い優先度で処理されます。DATASET 属性は、ユーザの明示的な割り当てで上書きされます。

注意：DATASET 属性の割り当てが有効な場合、その割り当てを明示的な割り当てコマンドで上書きするには CHECK FILE コマンドを発行する必要があります。CHECK FILE コマンドは、DATASET 属性の割り当てを解除します。

例　DATASET 属性による固定フォーマットデータソースの割り当て

次の例は、DATASET 属性を使用して固定フォーマットのデータソースを割り当てる方法を示しています。

Windows の場合

```
FILE=XX,  SUFFIX=FIX,  DATASET='C:¥DATA¥FILENAME.FTM   TEXT'
   .
   .
   .
```

UNIX の場合

```
FILE=XX,  SUFFIX=FIX,  DATASET='/u22/class/data/filename.ftm'
   .
   .
   .
```

バイナリデータソースの場合

```
FILE=XX,  SUFFIX=FIX,  DATASET='/u22/class/data/filename.ftm   BINARY'
   .
   .
   .
```

マスターファイルプロファイルの作成および使用

このバージョンでは、マスターファイル内を参照することができ、マスターファイルの処理中に実行されるプロファイルが導入されました。マスターファイルプロファイル (MFD_PROFILE) とは、リクエストのマスターファイル処理を一時停止し、停止後に実行され、再度マスターファイル処理に戻すプロシジャです。このプロファイルには多くの用途がありますが、特に次の場合に役立ちます。

- マスターファイルで定義されたグローバル変数の値を設定する。
- マスターファイルの DEFINE コマンドまたは DBA 属性の参照ファイルを作成する。
- 認証情報の外部テーブルを読み取ることにより、接続ユーザの DBA ルールを動的に作成する。

- DBAFILE を動的に作成する。このファイルは、外部データソースから取得したり、マスターファイルを参照するリクエストの実行中のアクセス制限に使用したりできます。

  注意: すべてのユーザを対象とした DBA ルールの DBAFILE を作成する代わりに、特定のユーザを対象とした DBA ルールをマスターファイルで動的に作成することもできます。

構文　マスターファイルプロファイルの呼び出し

マスターファイルの FILE 宣言に MFD_PROFILE 属性を追加します。

```
FILE = filename, SUFFIX = suffix, MFD_PROFILE = app/fexname,$
```

説明

`filename`
有効な任意のファイル名です。

`suffix`
マスターファイルで定義されたファイルタイプを指定する SUFFIX 値です。MFD_PROFILE は、あらゆるファイルタイプでサポートされます。

`app`
実行するプロシジャが格納されているアプリケーションの名前です。アプリケーション名を指定することで、同一名の別のプロシジャがアプリケーションパスの上位に存在する場合でも、実行するプロファイルが正しく特定されます。

`fexname`
MFD_PROFILE プロシジャの名前です。

参照　MFD_PROFILE 使用時の注意

- MFD_PROFILE は、MFD_PROFILE 属性が含まれているマスターファイルに対して TABLE、TABLEF、MATCH、GRAPH、CREATE、DEFINE、CHECK、?F、?FF リクエストが発行されるたびに実行されます。

  MATCH リクエストまたは MORE を使用したリクエストでは、そのリクエストに関連するマスターファイルで指定されている MFD_PROFILE のすべてが、リクエストの前に実行されます。これらのプロファイルは、リクエストで記述された逆の順序で実行されます。つまり、リクエストで最後に記述されているマスターファイルのプロファイルが最初に実行されます。

- MFD_PROFILE は、-READFILE コマンドの結果として実行されません。
- MFD_PROFILE は、WebFOCUS ツールからセグメントまたはフィールドを選択した場合には実行されません。

- 複数の MFD_PROFILE が同一のグローバル変数の値を設定した場合、結果の変数値は、これらのプロファイルプロシジャが実行された順序に基づいて設定されます。
- 複数のマスターファイルで MFD_PROFILE 属性が指定され、これらのマスターファイルが JOIN で結合されている場合、すべてのプロファイルが実行されます。
- 呼び出し元のマスターファイルの名前が最初のパラメータ (&1) として MFD_PROFILE に渡されます。
- MFD_PROFILE に機密情報が含まれている場合は、このプロファイルを暗号化する必要があります。このプロファイルで作成されるファイル (例、DBAFILE) に機密情報が含まれる場合は、そのファイルも暗号化する必要があります。
- MFD_PROFILE で作成されたファイルを呼び出し元のマスターファイルとともに使用する場合は、これらのファイルがアプリケーションパス上に存在する必要があります。この場合、可能な限りパスの上位に配置することをお勧めします。ユーザのログアウト時にファイルが削除されるようにするには、そのファイルを edatemp に配置します。
- MFD_PROFILE がアプリケーションパス上に存在しない場合、次の警告メッセージが発行されます。

  ```
  FOC(36373)  警告：MFD_PROFILE が存在しません
  ```

  プロファイルが見つからない場合に処理を停止するよう設定するには、ERROROUT パラメータを ON にします。
- MFD_PROFILE プロシジャでは、このプロシジャを 1 回だけ実行するためのカウンタを追加する場合を除いて、その呼び出し元マスターファイルに対するリクエストは発行しないでください。カウンタを追加しない場合、無限ループが発生します。MFD_PROFILE プロシジャで、それ自体に対するリクエストを実行する必要がある場合は、次の例のようにダイアログマネージャのグローバル変数をカウンタとして追加し、プロシジャが 1 回だけ実行されるようにします。

  ```
  -DEFAULT &&COUNTER=1;
  -IF &&COUNTER EQ 1 THEN GOTO START;
  -SET &&COUNTER= 2;
  -GOTO DONE
  -START
  MFD_PROFILE request against same Master as original request END
  -SET &&COUNTER=2;
  -DONE
  ```

MFD_PROFILE の最初の呼び出し時に &&COUNTER が 1 に設定されます。次に同一マスターファイルを参照する MFD_PROFILE リクエストの一部が実行されると、&&COUNTER が 2 に設定されます。このリクエストの実行で MFD_PROFILE のマスターファイルが参照されるため、MFD_PROFILE が再度呼び出されます。ただし、この時点で &&COUNTER が 2 に設定されているため、MFD_PROFILE リクエストは、同一マスターファイルを再度参照する部分に分岐します。これにより、MFD_PROFILE が無限ループで呼び出されることが回避されます。

- ビジネスビューに対するリクエストは、元のマスターファイルから MFD_PROFILE を呼び出します。
- MFD_PROFILE が DBAFILE を作成する場合、WebFOCUS ツール、または TABLE、TABLEF、GRAPH、MATCH 以外のコマンドを使用する前に、権限を所有するユーザおよび制限が SEGMENT または FIELD レベルが指定された最初の DBAFILE が存在する必要があります。最初に作成された DBAFILE には、VALUE 制限が含まれていない場合があります。実際の TABLE リクエストを実行するか、ツールの [実行] ボタンをクリックすると、MFD_PROFILE が実行され、VALUE 制限が適用されます。
- MFD_PROFILE の実行前に構築された環境に影響するコマンドは発行しないようにする必要があります。たとえば、MFD_PROFILE で JOIN を作成し、この JOIN を MFD_PROFILE の終了前にクリアする場合は、JOIN に一意の名前を付けておき、この JOIN のみをクリアする JOIN CLEAR コマンドを発行します。JOIN CLEAR * コマンドは発行しないでください。このコマンドを発行すると、MFD_PROFILE の実行前に作成された JOIN がすべてクリアされます。

例　MFD_PROFILE による参照ファイルの作成とグローバル変数の設定

ここでは、EMPDATA マスターファイルを次のように編集します。

- 「DDBAEMP」という MFD_PROFILE を指定します。
- PIN フィールドの TITLE 属性として使用する変数を定義します。
- JOBS 参照ファイルの値に基づいて従業員をフルタイムとパートタイムに分類するための TYPE_EMP および EMP_TYPE フィールドを定義します。
- DBA 制限を追加します。HR3 ユーザの場合、VALUE 句により JOBS 参照ファイルの値が検索されます。

編集後の EMPDATA マスターファイルは次のようになります。

```
FILENAME=EMPDATA, SUFFIX=FOC, MFD_PROFILE=baseapp/DDBAEMP,$
VARIABLE NAME = Emptitle, USAGE=A30, DEFAULT=EMPID,$
SEGMENT=EMPDATA,SEGTYPE=S0, $
 FIELDNAME=PIN    , ALIAS=ID, USAGE=A9, INDEX=I,   TITLE='&&Emptitle',$
 FIELDNAME=LASTNAME,      ALIAS=LN,        FORMAT=A15,             $
 FIELDNAME=FIRSTNAME,     ALIAS=FN,        FORMAT=A10,             $
 FIELDNAME=MIDINITIAL,    ALIAS=MI,        FORMAT=A1,              $
 FIELDNAME=DIV,           ALIAS=CDIV,      FORMAT=A4,              $
 FIELDNAME=DEPT,          ALIAS=CDEPT,     FORMAT=A20,             $
 FIELDNAME=JOBCLASS,      ALIAS=CJCLAS,    FORMAT=A8,              $
 FIELDNAME=TITLE,         ALIAS=CFUNC,     FORMAT=A20,             $
 FIELDNAME=SALARY,        ALIAS=CSAL,      FORMAT=D12.2M,          $
 FIELDNAME=HIREDATE,      ALIAS=HDAT,      FORMAT=YMD,             $
$
DEFINE AREA/A13=DECODE DIV (NE 'NORTH EASTERN' SE 'SOUTH EASTERN'
CE 'CENTRAL' WE 'WESTERN' CORP 'CORPORATE' ELSE 'INVALID AREA');$
DEFINE TYPE_EMP/I1 = DECODE JOBCLASS(JOBS ELSE 1);,$
DEFINE EMP_TYPE/A10 = IF TYPE_EMP EQ 1
          THEN 'FULL_TIME'  ELSE 'PART_TIME';
END
DBA=USERD,$
USER=USER1,ACCESS=R,RESTRICT=FIELD,NAME=SALARY,$
USER=USER2,ACCESS=R,RESTRICT=VALUE,NAME=SYSTEM,
                VALUE=DEPT EQ SALES OR MARKETING,$
USER=HR1,ACCESS=R,RESTRICT=VALUE,NAME=SYSTEM,
            VALUE=SALARY FROM 20000 TO 35000,$
USER=HR2,ACCESS=R,RESTRICT=VALUE,NAME=EMPDATA,VALUE=SALARY GT 0,$
USER=HR3,ACCESS=R,RESTRICT=VALUE,NAME=SYSTEM,VALUE=JOBCLASS EQ (JOBS),$
```

DDBAEMP プロシジャは、グローバル変数 &&Emptitle の値を設定し、JOBS 参照ファイルを作成します。

```
FILEDEF JOBS DISK jobs.ftm


-RUN
-SET &&Emptitle = 'Employee ID';
TABLE FILE JOBLIST
PRINT JOBCLASS
WHERE JOBDESC  CONTAINS '2ND' OR '3RD'
ON TABLE HOLD AS JOBS
END
```

次のリクエストは、EMPDATA データソースに対して実行され、JOBS ファイルを割り当て、ユーザパスワードを H3 に設定します。EMP_TYPE フィールドおよび HR3 ユーザの DBA VALUE 制限では、MFD_PROFILE により作成された JOBS ファイルが参照テーブルとして使用されます。

```
FILEDEF JOBS DISK jobs.ftm
```

```
-SET &PASS = 'HR3';
SET PASS = &PASS
-RUN
TABLE FILE EMPDATA
" Password used is &PASS "
" "
"USER1 -- Can't see Salary, reject request"
"USER2 -- Can see Sales and Marketing departments only"
"HR1   -- Can see salaries from 20 TO 35 K "
"HR2   -- Can see everyone "
"HR3   -- Can see Part Time only "
" "
PRINT PIN SALARY DEPT EMP_TYPE
ON TABLE SET PAGE NOPAGE
ON TABLE SET STYLE *
TYPE=REPORT, GRID=OFF, FONT=ARIAL,$
END
```

出力結果では、PIN フィールドの列タイトルに、MFD_PROFILE で設定された &&Emptitle 変数の値が表示され、このプロファイルで作成された JOBS ファイルに基づいてレポート出力がパートタイム従業員のみに限定されています。これは、HR3 ユーザに表示許可が与えられた情報です。

Password used is HR3

USER1 -- Can't see Salary, reject request
USER2 -- Can see Sales and Marketing departments only
HR1 -- Can see salaries from 20 TO 35 K
HR2 -- Can see everyone
HR3 -- Can see Part Time only

Employee ID	SALARY	DEPT	EMP_TYPE
000000090	$33,000.00	PERSONNEL	Part Time
000000100	$32,400.00	ACCOUNTING	Part Time
000000110	$19,300.00	CUSTOMER SUPPORT	Part Time
000000180	$25,400.00	ADMIN SERVICES	Part Time
000000240	$33,300.00	PERSONNEL	Part Time
000000400	$26,400.00	ACCOUNTING	Part Time

例

MFD_PROFILE による DBAFILE の作成

ここでは、EMPDATA マスターファイルを編集して、「DDEMP2」という MFD_PROFILE および「DBAEMP2」という DBAFILE を指定します。MFD_PROFILE は、security.data というシーケンシャルファイルからセキュリティ属性を読み取ることで DBAFILE を作成します。

マスターファイルは次のとおりです。

```
FILENAME=EMPDATA, SUFFIX=FOC, MFD_PROFILE=baseapp/DDEMP2,$
SEGMENT=EMPDATA,SEGTYPE=S0, $
 FIELDNAME=PIN    , ALIAS=ID, USAGE=A9, INDEX=I,   TITLE='Employee Id',$
 FIELDNAME=LASTNAME,      ALIAS=LN,       FORMAT=A15,             $
 FIELDNAME=FIRSTNAME,     ALIAS=FN,       FORMAT=A10,             $
 FIELDNAME=MIDINITIAL,    ALIAS=MI,       FORMAT=A1,              $
 FIELDNAME=DIV,           ALIAS=CDIV,     FORMAT=A4,              $
 FIELDNAME=DEPT,          ALIAS=CDEPT,    FORMAT=A20,             $
 FIELDNAME=JOBCLASS,      ALIAS=CJCLAS,   FORMAT=A8,              $
 FIELDNAME=TITLE,         ALIAS=CFUNC,    FORMAT=A20,             $
 FIELDNAME=SALARY,        ALIAS=CSAL,     FORMAT=D12.2M,          $
 FIELDNAME=HIREDATE,      ALIAS=HDAT,     FORMAT=YMD,             $
$
DEFINE AREA/A13=DECODE DIV (NE 'NORTH EASTERN' SE 'SOUTH EASTERN'
CE 'CENTRAL' WE 'WESTERN' CORP 'CORPORATE' ELSE 'INVALID AREA');$
END
DBA=USERD,DBAFILE=DBAEMP2,$
```

セキュリティ属性が格納されたファイル (security.data) は次のとおりです。セキュリティ属性は、USER、ACCESS、RESTRICT、NAME、VALUE 属性です。

```
USER1       R   NOPRINT     SALARY
USER2       R   VALUE       SYSTEM  DEPT EQ SALES OR MARKETING
HR1         R   VALUE       SYSTEM  SALARY FROM 20000 TO 35000
HR1         W   SEGMENT     EMPDATA
HR2         R   VALUE       EMPDATA SALARY GT 0
```

これらの属性に基づいて、次のパスワードでユーザが分類されます。

- USER1 は、SALARY フィールドの値を表示することはできません。
- USER2 は、SALES および MARKETING 部門のみを表示することができます。
- HR1 は、20000 から 35000 ドルまでの給与のみを表示することができます。HR1 は、EMPDATA セグメントに書き込むこともできます。
- HR2 は、EMPDATA セグメントのすべてを表示することができます。

DDEMP2 プロファイルプロシジャは次のとおりです。

- ダイアログマネージャコマンドの -WRITE を使用して、DBAEMP2 マスターファイルにレコードを書き込みます。最初に FILE 宣言、SEGMENT 宣言、FIELD 宣言、END 宣言を書き込み、ここからマスターファイルの DBA セクションが開始され、DBA パスワードが後に続きます。このパスワードは、EMPDATA マスターファイルの DBA パスワードと同一にする必要があります。EDAEMP2 マスターファイルは edatemp に書き込まれます。

  DBA セクションの残りの部分は、security.data ファイルから各レコードを読み込み、対応する DBA レコードを DBAEMP2 マスターファイルに書き込むという方法で作成されます。

- 引数 &1 として渡されたマスターファイル名を使用して、EMPDATA マスターファイルの FILE 宣言を書き込みます。この宣言の後に、そのマスターファイルに固有の DBA 宣言が書き込まれます。

- ダイアログマネージャコマンドの -READFILE を使用して、security.data ファイルの各レコードを順に読み取ります。RESTRICT 属性が存在するかどうかを確認し、存在する場合は、VALUE 属性が含まれているかどうかを確認します。存在する属性に応じて、DBAEMP2 マスターファイルにレコードを書き込みます。

DDEMP2 プロファイルプロシジャは次のとおりです。

```
-* FILEDEF the input security.data file (Windows)
FILEDEF SECURITY DISK c:¥ibi¥apps¥baseapp¥security.data (LRECL 81

-* DYNAM the output DBAEMP2 Master File and the input file (z/OS)
DYNAM OUTFI DA USER1.DBAEMP2.MASTER SHR REU
DYNAM SECURITY DA USER1.SECURITY.DATA SHR REU

-RUN
-* Write out the first part of the DBAEMP2 Master File
-WRITE OUTFI FILE=DBAEMP2,SUFFIX=FIX,$
-WRITE OUTFI SEGNAME=ONE,SEGTYPE=S0
-WRITE OUTFI FIELD=ONE,,A1,A1,$
-WRITE OUTFI END
-WRITE OUTFI DBA=USERD,$
-* Write out a FILE declaration for the calling Master File, passed as &1
-WRITE OUTFI FILE=&1,$
-* Initialize the variables to be read from the security.data file
-SET &USER=' ';
-SET &ACCESS=' ';
-SET &RESTRICT=' ';
-SET &NAME = ' ';
-SET &VALUE = ' ';
```

```
-* Establish the loop for each record of the security.data file
-SET &DONE =  N ;
-REPEAT ENDLP WHILE &DONE EQ N ;
-* Read a record from security.data
-READFILE SECURITY
-* Check if the end of the security.data file was reached and,
-*   if so, branch out of the loop
-SET &DONE =  IF &IORETURN EQ 1 THEN 'Y' ELSE 'N';
-IF &DONE EQ 'Y' GOTO ENDLP1;
-* If there is a RESTRICT attribute, go to the label -CHKSTR.
-IF &RESTRICT NE ' ' THEN GOTO CHKRSTR;
-* If there is no RESTRICT attribute,
-*  write the USER and ACCESS attributes, and loop for the next record
-WRITE OUTFI USER=&USER , ACCESS=&ACCESS ,$
-GOTO ENDLP

-CHKRSTR
-* If there is a RESTRICT attribute, check if it has a VALUE attribute
-* and, if so, go to the label -CHKVAL
-IF &VALUE NE ' ' THEN GOTO CHKVAL;
-* If there is no VALUE attribute,
-*  write USER, ACCESS, RESTRICT, and NAME, and loop for next record
-WRITE OUTFI USER=&USER, ACCESS=&ACCESS, RESTRICT=&RESTRICT, NAME=&NAME,$
-GOTO ENDLP

-CHKVAL
-* If there is a VALUE attribute, write out USER, ACCESS, RESTRICT,
-*  NAME, and VALUE, and loop for next record
-WRITE OUTFI  USER=&USER, ACCESS=&ACCESS,RESTRICT=&RESTRICT,NAME=&NAME, VALUE =
&VALUE ,$
-ENDLP
-ENDLP1
```

このプロシジャを実行すると、次の DBAFILE が作成されます。

```
FILE=DBAEMP2,SUFFIX=FIX,$
SEGNAME=ONE,SEGTYPE=S0
  FIELD=ONE,,A1,A1,$
END
DBA=USERD,$
FILE=EMPDATA,$
USER=USER1,ACCESS=R,RESTRICT=NOPRINT,NAME=SALARY   ,$
USER=USER2, ACCESS=R, RESTRICT=VALUE, NAME=SYSTEM,
  VALUE=DEPT EQ SALES OR MARKETING ,$
USER=HR1,   ACCESS=R, RESTRICT=VALUE, NAME=SYSTEM,
  VALUE = SALARY FROM 20000 TO 35000,$
USER=HR1,   ACCESS=W, RESTRICT=SEGMENT, NAME=EMPDATA  ,$
USER=HR2,   ACCESS=R, RESTRICT=VALUE, NAME=EMPDATA,
  VALUE = SALARY GT 0 ,$
```

次のリクエストは、EMPDATA データソースの PIN、SALARY、TITLE、DEPT フィールドを表示します。

```
TABLE FILE EMPDATA
PRINT SALARY TITLE DEPT
BY PIN
WHERE PIN GE '000000010'  AND PIN LE  '000000200'
ON TABLE SET PAGE NOPAGE
ON TABLE PCHOLD FORMAT PDF
END
```

このリクエストを実行するには、最初に有効なユーザパスワードを設定する必要があります。MFD_PROFILE プロシジャが最初に実行され、dbaemp2.mas という DBAFILE が作成されます。

最初に SET PASS=USER1 コマンドを発行してこのリクエストを実行すると、次のレポートが生成されます。SALARY フィールドに RESTRICT=NOPRINT 属性が指定されているため、このレポートの給与には 0 (ゼロ) が表示されます。

Employee Id	SALARY	TITLE	DEPT
000000010	$.00	MARKETING EXECUTIVE	MARKETING
000000020	$.00	INDUSTRIAL MARKETER	MARKETING
000000030	$.00	SALES MANAGER	SALES
000000040	$.00	MARKETING DIRECTOR	MARKETING
000000050	$.00	EXECUTIVE MANAGER	SALES
000000060	$.00	MARKETING DIRECTOR	MARKETING
000000070	$.00	MANAGER	ACCOUNTING
000000080	$.00	SENIOR SALES EXEC.	MARKETING
000000090	$.00	EMPLOYEE COORDINATOR	PERSONNEL
000000100	$.00	SUPERVISOR OF AP/AR	ACCOUNTING
000000110	$.00	PRODUCT DISTRIBUTOR	CUSTOMER SUPPORT
000000120	$.00	CORPORATE CONSULTANT	CONSULTING
000000130	$.00	MANAGER	MARKETING
000000140	$.00	MANAGER	CUSTOMER SUPPORT
000000150	$.00	SENIOR CONSULTANT	CONSULTING
000000160	$.00	MNGR OF CUST SUPPORT	CUSTOMER SUPPORT
000000170	$.00	ADMINISTRATOR	ADMIN SERVICES
000000180	$.00	ASST ADMINISTRATOR	ADMIN SERVICES
000000190	$.00	SALES SPECIALIST	SALES
000000200	$.00	VICE PRES	SALES

最初に SET PASS=USER2 コマンドを発行してこのリクエストを実行すると、次のレポートが生成されます。DEPT フィールドに VALUE 制限が指定されているため、このレポートには SALES および MARKETING 部門のみが表示されます。

Employee Id	SALARY	TITLE	DEPT
000000010	$55,500.00	MARKETING EXECUTIVE	MARKETING
000000020	$62,500.00	INDUSTRIAL MARKETER	MARKETING
000000030	$70,000.00	SALES MANAGER	SALES
000000040	$62,500.00	MARKETING DIRECTOR	MARKETING
000000050	$54,100.00	EXECUTIVE MANAGER	SALES
000000060	$55,500.00	MARKETING DIRECTOR	MARKETING
000000080	$43,400.00	SENIOR SALES EXEC.	MARKETING
000000130	$62,500.00	MANAGER	MARKETING
000000190	$39,000.00	SALES SPECIALIST	SALES
000000200	$115,000.00	VICE PRES	SALES

最初に SET PASS=HR1 コマンドを発行してこのリクエストを実行すると、次のレポートが生成されます。DEPT フィールドに VALUE 制限が指定されているため、このレポートには 20000 から 35000 ドルの給与のみが表示されます。

Employee Id	SALARY	TITLE	DEPT
000000090	$33,000.00	EMPLOYEE COORDINATOR	PERSONNEL
000000100	$32,400.00	SUPERVISOR OF AP/AR	ACCOUNTING
000000170	$30,800.00	ADMINISTRATOR	ADMIN SERVICES
000000180	$25,400.00	ASST ADMINISTRATOR	ADMIN SERVICES

例　マスターファイルでの動的 DBA ルールの作成

次のシーケンシャルデータソース (VALTEST.DATA) には、ユーザ名およびそれに関連付けられた VALUE 制限が含まれています。

```
SALLY               CURR_SAL LT 20000
JOHN                DEPARTMENT EQ PRODUCTION
TOM                 CURR_SAL GE 20000
```

このファイルを読み取る前に、FILEDEF でファイルを定義するか、ファイルを割り当てておく必要があります。

```
FILEDEF VALTEST DISK baseapp/valtest.data
```

次のマスターファイル EMPDBA は、EMPLOYEE データソースのビューを示しています。このマスターファイルの DBA セクションでは、USER 属性にグローバル変数 &&UID が使用され、EMPINFO セグメントに対する値テストにグローバル変数 &&VAL が使用されています。また、「DBAEMP3」というマスターファイルプロファイルが指定されています。このプロファイルは、VALTEST.DATA ファイルを読み取ることで、接続ユーザのユーザ ID を取得し、正しい VALUE 制限を特定します。これらのグローバル変数を正しい値に設定することで、そのユーザに対応した DBA ルールがマスターファイルに挿入されます。

注意：グローバル変数 &&UID の代わりに、システム変数 &FOCSECUSER を使用することもできます。

```
FILENAME=EMPLOYEE, SUFFIX=FOC, MFD_PROFILE=DBAEMP3,$
VARIABLE NAME=&&UID, USAGE=A8 , $
VARIABLE NAME=&&VAL, USAGE=A25, $

SEGNAME=EMPINFO,  SEGTYPE=S1
 FIELDNAME=EMP_ID,       ALIAS=EID,     FORMAT=A9,       $
 FIELDNAME=LAST_NAME,    ALIAS=LN,      FORMAT=A15,      $
 FIELDNAME=FIRST_NAME,   ALIAS=FN,      FORMAT=A10,      $
 FIELDNAME=HIRE_DATE,    ALIAS=HDT,     FORMAT=I6YMD,    $
 FIELDNAME=DEPARTMENT,   ALIAS=DPT,     FORMAT=A10,      $
 FIELDNAME=CURR_SAL,     ALIAS=CSAL,    FORMAT=D12.2M,   $
 FIELDNAME=CURR_JOBCODE, ALIAS=CJC,     FORMAT=A3,       $
 FIELDNAME=ED_HRS,       ALIAS=OJT,     FORMAT=F6.2,     $
END
DBA=DBAUSER1,$
USER=&&UID,ACCESS=R,RESTRICT=VALUE,NAME=EMPINFO,VALUE=&&VAL,$
```

MFD_PROFILE プロシジャは次のとおりです。

```
SET MESSAGE = OFF
-SET &VALUETEST = 'NOTFOUND';
-* Find the user ID of the connected user
-SET &&UID = GETUSER('A20');
-SET &&UID = TRUNCATE(&&UID);
-* Create a HOLD file with the value test for the connected user
TABLE FILE VALTEST
PRINT VALUETEST
WHERE USERNAME EQ '&&UID'
ON TABLE HOLD AS USERVAL FORMAT ALPHA
END
-RUN
-READ USERVAL &VALUETEST.A30
-* If the user name was not in the file, type a message and exit
-IF &VALUETEST NE 'NOTFOUND' GOTO SETVALUE;
-TYPE USER WASN'T THERE
-EXIT
-SETVALUE
-* Set the global variable for the value test to the correct test
-SET &&VAL = ''|&VALUETEST||'';
-* Set the USER parameter to the user ID of the connected user
SET USER = &&UID
```

次のリクエストは、EMPLOYEE データソースの EMPDBA ビューを使用してレポートを表示します。

```
USE
EMPLOYEE AS EMPDBA
END
-RUN
TABLE FILE EMPDBA
PRINT LN FN CURR_SAL
BY DEPARTMENT
ON TABLE SET PAGE NOPAGE
END
```

接続ユーザが SALLY の場合、このリクエストを実行すると、給与が 20,000 ドル未満の従業員のレポートが生成されます。

```
DEPARTMENT  LAST_NAME        FIRST_NAME         CURR_SAL
----------  ---------        ----------         --------
MIS         SMITH            MARY             $13,200.00
            JONES            DIANE            $18,480.00
            MCCOY            JOHN             $18,480.00
            GREENSPAN        MARY              $9,000.00
PRODUCTION  STEVENS          ALFRED           $11,000.00
            SMITH            RICHARD           $9,500.00
            MCKNIGHT         ROGER            $16,100.00
```

接続ユーザが TOM の場合、このリクエストを実行すると、給与が 20,000 ドル以上の従業員のレポートが生成されます。

```
DEPARTMENT  LAST_NAME        FIRST_NAME          CURR_SAL
----------  ---------        ----------          --------
MIS         BLACKWOOD        ROSEMARIE         $21,780.00
            CROSS            BARBARA           $27,062.00
PRODUCTION  BANNING          JOHN              $29,700.00
            IRVING           JOAN              $26,862.00
            ROMANS           ANTHONY           $21,120.00
```

接続ユーザが JOHN の場合、このリクエストを実行すると、PRODUCTION 部門のみが含まれたレポートが生成されます。

```
DEPARTMENT  LAST_NAME        FIRST_NAME          CURR_SAL
----------  ---------        ----------          --------
PRODUCTION  STEVENS          ALFRED            $11,000.00
            SMITH            RICHARD            $9,500.00
            BANNING          JOHN              $29,700.00
            IRVING           JOAN              $26,862.00
            ROMANS           ANTHONY           $21,120.00
            MCKNIGHT         ROGER             $16,100.00
```

メタデータのローカライズと翻訳ファイルの使用

ローカライズしたフィールドタイトル、説明、プロンプトを一元的に管理し、これらを複数のマスターファイルに適用する場合は、翻訳ファイルセットを作成し、マスターファイルの TRANS_FILE 属性を使用して、これらのファイルを呼び出すことができます。これらのファイルは手動で設定することも、LNGPREP ユーティリティを使用して翻訳ファイルを準備することもできます。

LNGPREP ユーティリティ - メタデータ言語ファイルの準備

LNGPREP ユーティリティを使用すると、アプリケーションのマスターファイルから TITLE、DESCRIPTION、CAPTION、PROMPT 属性の値が抽出され、使用する各言語の特別なフォーマットの言語翻訳ファイルに格納されます。これらの言語翻訳ファイルのコンテンツを翻訳すると、各ユーザがそれぞれ選択した言語でアプリケーションを実行することができます。

LNGPREP は、2 つの処理を実行します。LNGPREP は、マスターファイルから属性値を抽出して言語ファイルに格納するとともに、そのマスターファイルの TRANS_FILE 属性に、各言語ファイルの格納先アプリケーションフォルダを識別する値と、一連の言語ファイル名に使用する接頭語を挿入 (または更新) します。マスターファイルがクラスタの一部の場合、LNGPREP はそのクラスタで参照されている各マスターファイルから翻訳可能文字列を抽出するとともに、各マスターファイルの TRANS_FILE 属性を同一値で更新します。

LNGPREP には、使用する各言語の 3 文字コードが記述された入力ファイルが必要です。

各言語ファイルの名前は、TRANS_FILE 値で指定された接頭語で開始し、次に 3 文字言語コードが続き、.lng 拡張子で終了します。

たとえば、言語入力ファイルでフランス語とスペイン語の言語コードが指定されている場合を想定します。

```
fre
spa
```

マスターファイルでは、次のように指定されています。

```
trans_file = xlate/xl_
```

これらの言語翻訳ファイルは、xlate アプリケーションフォルダに、次のファイル名で作成されます。

- xl_fre.lng - フランス語
- xl_spa.lng - スペイン語

参照 ベース言語ファイル

各マスターファイルには、TITLE、DESCRIPTION、CAPTION、PROMPT 属性が指定された、1 つのベース言語が必要です。この言語が英語である必要はありません。

LNGPREP は、これらの属性値を抽出し、ベース言語ファイルに格納します。ベース言語コードは eng ですが、これは慣例的にこの言語コードが使用されているためです。この場合、eng は英語を表すものではありません。これは、単に「マスターファイルが記述されている言語」を表します。

ベース言語ファイル (*prefix*eng.lng) は、手動で編集しないでください。その他の .lng ファイルはすべて、翻訳者がベース言語から別の言語に文字列値を翻訳するために手動で編集します。

アプリケーションを英語に翻訳

言語コードの eng はベース言語を表す予約語のため、ベース言語が英語以外の場合に、この言語コードを使用してアプリケーションの英語翻訳を含めることはできません。英語翻訳を含めるには、他のいずれかの英語系言語コードを使用します (例、AME、UKE、CAE、AUE)。たとえば、ベース言語がドイツ語の場合、言語入力ファイルで AME を指定し、LNGPREP を実行すると、*prefix*eng.lng ファイルと *prefix*ame.lng ファイルの両方がドイツ語で生成されます。*prefix*ame.lng ファイルのコンテンツを英語に翻訳します。*prefix*eng.lng ファイルは、そのままにします。

参照

翻訳済みマスターファイル属性の表示

各言語ファイルには、関連する一連のマスターファイルから抽出された属性値に対応する行が追加されます。各属性値には、一意のインデックス番号が割り当てられます。たとえば、マスターファイルに「FIELDNAME=PRODUCT_CATEGORY, TITLE='Product,Category'」という記述があり、TITLE が翻訳可能な 39 番目の属性値の場合、LNGPREP が生成する .lng ファイルのすべてに次の行が追加されます。

```
39 = Product,Category
```

たとえば、フランス語の翻訳者は、*prefix*fre.lng ファイルを編集し、インデックス番号は変更せずに、文字列の「Product,Category」を翻訳します。

```
39 = Produit,Catégorie
```

実行時に PRODUCT_CATEGORY フィールドの TITLE を表示し、WebFOCUS の構成が LANG=FRE に設定されている場合、WebFOCUS が最初に *prefix*eng.lng ファイル内の「Product,Category」を検索し、インデックス番号の「39」を特定した後、*prefix*fre.lng ファイル内の「39」を検索し、TITLE として「Produit,Catégorie」を表示します。

LNGPREP のモード

LNGPREP は、Web コンソールの [翻訳ファイルのプリペア] オプションを使用して実行することも、構文を使用して実行することもできます。いずれの場合でも、作成する各翻訳ファイルの言語を表す 3 文字言語コードが記述された構成ファイルを作成し、各言語コードをそれぞれ別の行に記述する必要があります。特定のマスターファイルに対する LNGPREP の 1 回目の実行時は、そのマスターファイルおよび関連するすべてのマスターファイルに TRANS_FILE 属性が追加され、これらのマスターファイルから属性値を読み取ることでベース言語ファイルが作成されます。同時に、ベース言語ファイルのコピーも作成され、指定された名前が付けられます。次に、翻訳者が追加言語ごとにベース言語の属性値を翻訳し、その言語ファイルに対応した言語にします。

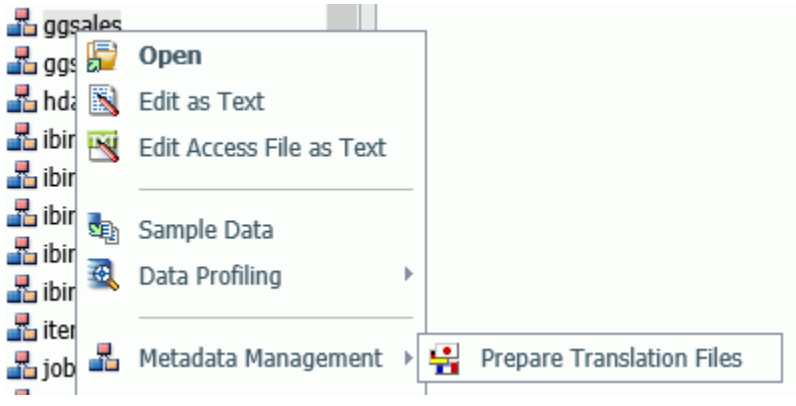
LNGPREP の 2 回目以降の実行時は、関連する一連のマスターファイルのリストおよび属性値に変更があるかどうかが確認され、必要に応じてこれらのマスターファイルが更新されます。次に、翻訳者が言語ファイルに追加された属性値を翻訳します。

参照　LNGPREP のベストプラクティス

ベストプラクティスとして、最初に .lng ファイルを格納する専用のアプリケーションディレクトリを作成し、LNGPREP コマンドの実行時は常に、そのアプリケーション名および共通の接頭語を使用することをお勧めします。さらに、このアプリケーションフォルダに言語の *fn*.cfg 入力ファイルを格納します。これにより、すべてのアプリケーションに使用される .lng ファイル一式が同一フォルダに作成され、翻訳に要する時間と労力が最小限に抑えられます。

手順　Web コンソールでメタデータ言語ファイルを準備するには

1. 下図のように、シノニムを右クリックし、[メタデータ管理]、[翻訳ファイルのプリペア] を順に選択します。



下図のように、[翻訳ファイルを設定] ページが開きます。

2. 次の値を入力するか、デフォルト値を受容します。

翻訳ファイルのアプリケーション
: 言語ファイルを格納するアプリケーションの名前です。参照ボタンをクリックして、現在のアプリケーションパスに存在するアプリケーションを選択することができます。デフォルト設定では、シノニムが存在するアプリケーションです。

接頭語
: 選択したシノニムの翻訳ファイルに使用する接頭語です。

言語ファイル
: 翻訳ファイルを準備する言語コードのリストが記述されたファイルです。このファイルは、拡張子を .cfg にし、アプリケーションパス上のアプリケーションディレクトリに格納する必要があります。また、各言語コードをそれぞれ別の行に記述します。参照ボタンをクリックして、言語ファイルが格納されているアプリケーションを選択することができます。

3. [OK] をクリックします。

   指定したアプリケーション、接頭語、言語構成ファイルを使用して、言語ファイルの準備が完了しました。ステータスウィンドウが開き、作成された言語ファイルおよび処理されたマスターファイルのリストが表示されます。

構文　構文による LNGPREP コマンドの実行

```
LNGPREP FILE n_part_name LNGAPP appname LNGPREFIX prefix
        LNGFILE appname/fn
```

説明

`n_part_name`
: マスターファイルの n 構成要素 (app1/app2...) の名前を指定します。

`appname`
: .lng ファイルの書き込み先および更新先となるディレクトリを指定します。

`prefix`
: .lng ファイルの名前で、3 文字の言語コードの前に付ける文字列を指定します。

`appname/fn`
: 3 文字の言語コードのリスト (1 言語につき 1 行) が記述された、ユーザ作成の .cfg ファイルの格納先アプリケーション名およびファイル名を指定します。たとえば、次の langretail.cfg ファイルには、英語 (米国)、フランス語、日本語の言語コードが記述されています。

  ```
  ame
  fre
  jpn
  ```

例　サンプル LNGPREP コマンド

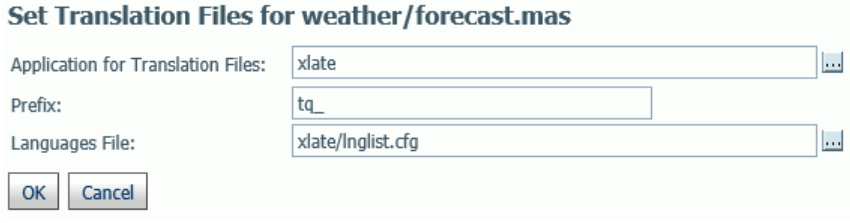
ここでは、lnglist.cfg ファイルに、fre (フランス語) と spa (スペイン語) の言語コードが記述されていることを前提にします。

```
fre
spa
```

次の LNGPREP コマンドを発行します。

```
LNGPREP FILE weather/forecast LNGAPP xlate LNGPREFIX tq_ LNGFILE  xlate/lnglist
```

別の方法として、forecast シノニムを右クリックし、[メタデータ管理]、[翻訳ファイルのプリペア] を順に選択することもできます。下図のように、[翻訳ファイルを設定] ページが開きます。下図のように値を入力し、[OK] をクリックします。



次の言語ファイルが作成されます。

- xlate / tq_eng.lng
- xlate / tq_fre.lng
- xlate / tq_spa.lng

weather/forecast.mas マスターファイルは、次の属性で更新されます。

```
TRANS_FILE= xlate/tq_
```

翻訳者は、xlate/tq_fre.lng および xlate/tq_spa.lng ファイルの値を翻訳する必要があります。

3 フィールドグループの記述

データソースのフィールドの中には、互いに 1 対 1 の関係を持つものがあります。これらのフィールドで構成されたグループは、別のグループに関係付けることができます。特定のデータソースタイプでは、サブセットとしてグループの論理的なビューを定義することができます。これらのグループおよびグループ間の関係を識別するには、マスターファイルおよびアクセスファイルで属性を指定するか、JOIN コマンドなどの関連機能を使用します。

ここでは、フィールド間の関係およびマスターファイルの属性を使用してこれらの関係を実装する方法について説明します。使用するデータソースのタイプによってはアクセスファイルが必要になる場合があります。アクセスファイルでグループを定義し、グループ間の関係を構築する方法についての補足情報は、該当するデータアダプタのマニュアルを参照してください。

トピックス

単一フィールドグループの定義

データソースのフィールドの中には、互いに 1 対 1 の関係を持つものがあります。この関係では、一方のフィールドの 1 つの値に対して他方のフィールドにもそれに対応する値が 1 つ存在します。たとえば、従業員データが格納された EMPLOYEE データソースについて考察します。

- 各従業員は、他の従業員と重複しない ID 番号を 1 つだけ持っています。

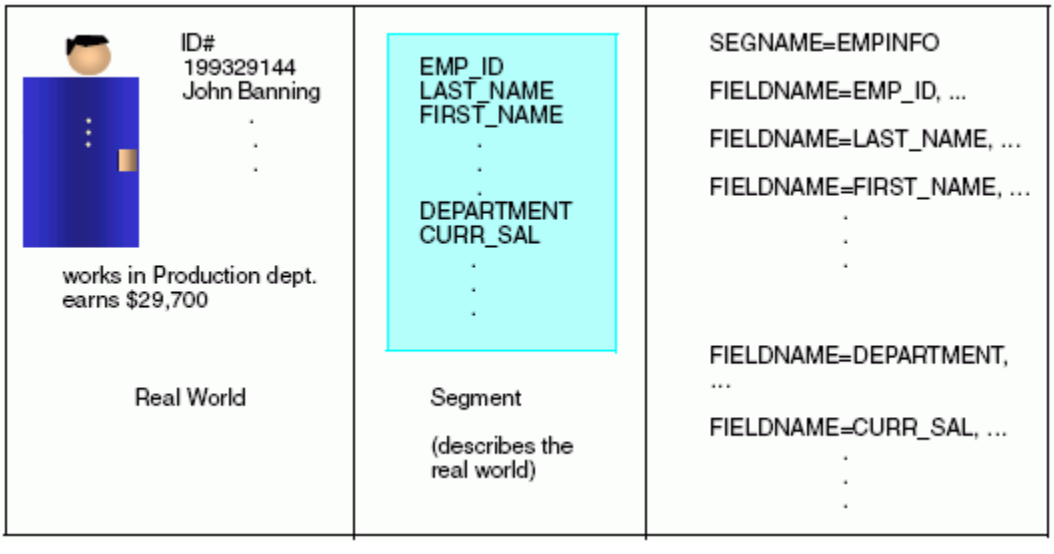
- 各 ID 番号には、その従業員の姓、名、入社日、所属部門、現在の給与のデータがそれぞれ 1 つ存在します。

このデータソースでは、それぞれのフィールドが各従業員の個々の特性を表しています。つまり、従業員の情報は、これらのフィールドのグループとして表されます。マスターファイルでは、このグループを「セグメント」と呼びます。

セグメントの理解

セグメントは、互いに 1 対 1 の関係を持つフィールドのグループです。通常は、それぞれの特性が関連するフィールドのグループを記述します。リレーショナルデータソースではセグメントはテーブルに相当します。セグメントは大規模なデータ構造の基本的要素です。異なるセグメントを互いに関係付けて新しいデータ構造を記述することができます。詳細は、61 ページの 「 複数フィールドグループの関係作成 」 を参照してください。

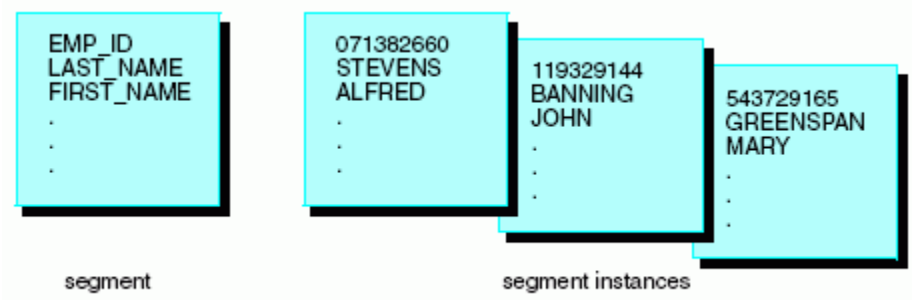
下図は、セグメントの概念を示しています。



セグメントインスタンスの理解

データを抽象的に記述したものがセグメントであるのに対し、各セグメントに対応する実際のデータがインスタンスです。インスタンスは、データソースに存在するセグメントの値です。リレーショナルデータソースでは、セグメントインスタンスはテーブルの行に相当します。単一セグメントのデータソースでは、セグメントインスタンスはレコードに相当します。

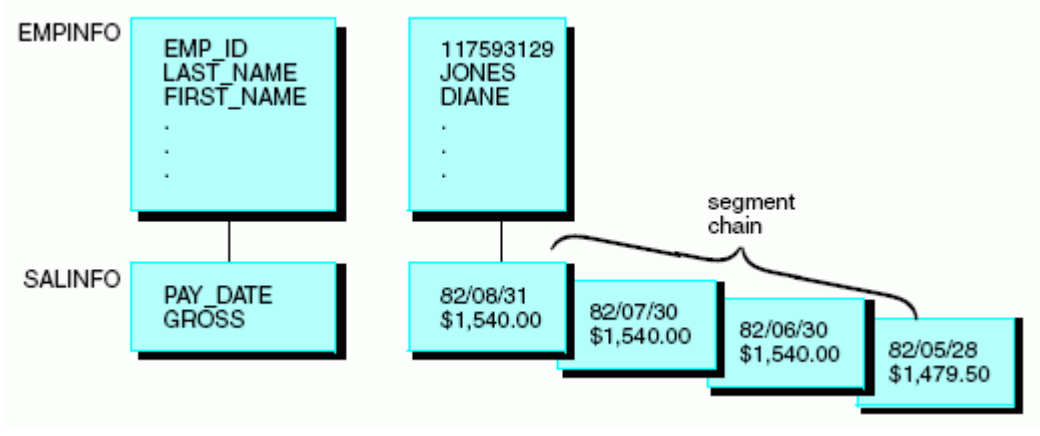
下図は、セグメントとインスタンスの関係を示しています。



セグメント連鎖の理解

1 つの親インスタンスの下位にあるセグメントインスタンスは、総称して「セグメント連鎖」と呼ばれます。特別な場合として、親セグメントを持たないルートセグメントではすべてのルートインスタンスで連鎖が形成されます。親子関係についての詳細は、65 ページの 「 論理的な依存関係 - 親子関係 」 を参照してください。

下図は、セグメント連鎖の概念を示しています。



マスターファイルにセグメントを記述するには、SEGNAME 属性および SEGTYPE 属性を使用します。SEGNAME 属性についての詳細は、58 ページの 「 セグメントの識別 - SEGNAME 」 を参照してください。

キーフィールドの識別

ほとんどのセグメントにはキーフィールドが存在します。キーフィールドとは、セグメントインスタンスを他のインスタンスと重複することなく識別できる 1 つまたは複数のフィールドのことです。たとえば、EMPLOYEE データソースでは、各従業員が ID 番号を 1 つだけ持ち、それぞれの ID 番号が重複して存在することはないため、ID 番号がキーフィールドになります。このデータソースでは、この ID 番号は EMP_ID フィールドとして表されます。

データソースがアクセスファイルを使用する場合、キーフィールドに指定するフィールドをアクセスファイルで識別することがあります。リレーショナルデータソースでも、マスターファイルで主キーとして指定するフィールドはそのセグメントの先頭のフィールドとして記述する必要があります。つまり、主キーのフィールド宣言を最初に記述してからセグメントの他のフィールドをその後に続けます。

FOCUS データソースでは、マスターファイルに SEGTYPE 属性を使用してキーフィールドおよびソート順を記述します。詳細は、231 ページの 「 FOCUS データソースの記述 」 を参照してください。キーフィールドは、セグメントの先頭フィールドに配置します。

セグメントの識別 - SEGNAME

セグメントを識別するには SEGNAME 属性を使用します。この属性は、セグメント宣言で指定する最初の属性です。エイリアスは SEGMENT です。

FOCUS データソースの場合、セグメント名の最大長は 8 バイトです。XFOCUS データソースの場合、セグメント名の最大長は 64 バイトです。FOCUS 以外のデータソースでも DBMS でサポートされている場合は、最大で 64 バイトのセグメント名を使用することができます。マスターファイルを自動生成するには、ユーザまたはネイティブのファイルマネージャが理解できる名前を SEGNAME に設定します。たとえば、DB2 テーブルを記述する場合、SEGNAME にはそのテーブルの名前 (または省略名) を割り当てます。

1 つのマスターファイル内で同一のセグメント名を重複して使用することはできません。ただし、この規則には例外が 1 つあります。FOCUS または XFOCUS データソースでは、マスターファイルで定義された JOIN のクロスリファレンスセグメント名を、これとは別のマスターファイルで定義された JOIN のクロスリファレンスセグメント名と同一にすることができます。同一の名前を持つ場合でも、CRSEGNAME 属性を使用することにより、これらの名前を互いに区別することができます。詳細は、271 ページの「 マスターファイルでの JOIN の定義 」を参照してください。

FOCUS または XFOCUS データソースでは、データソースにデータを入力した後で SEGNAME の値を変更することはできません。他のすべてのデータソースタイプでは、SEGNAME への参照をすべて変更する場合に限り、SEGNAME の値を変更することができます。たとえば、マスターファイルおよびアクセスファイルの参照をすべて変更します。

データソースがマスターファイルだけでなくアクセスファイルも使用する場合は、その両方に同一のセグメント名を指定する必要があります。

構文　セグメントの識別

```
{SEGNAME|SEGMENT} = segment_name
```

説明

`segment_name`
セグメントの識別名です。FOCUS データソースの場合、セグメント名の最大長は 8 バイトです。XFOCUS データソースの場合、セグメント名の最大長は 64 バイトです。FOCUS 以外のデータソースでも DBMS でサポートされている場合は、最大で 64 バイトのセグメント名を使用することができます。

名前の先頭には文字を使用する必要があります。それ以外には、文字、数字、アンダースコア (_) を任意に組み合わせることができます。異なるオペレーティングシステム環境や、数式を計算する際に問題を起こす可能性があるため、これ以外の文字は使用しないことをお勧めします。

例　セグメントの識別

あるセグメントが「TICKETS」というリレーショナルテーブルに対応する場合、次のように SEGNAME 属性を使用してそのセグメントに同一の名前を付けます。

```
SEGNAME = TICKETS
```

論理ビューの識別 - セグメントの再定義

通常、定義するセグメントは、データソース内で基本となるグループと一致します。たとえば、1 つのセグメントは、リレーショナルデータソースのテーブル 1 つの場合もあります。

ただし、セグメントの使用がネイティブデータソースで最初に定義された範囲に制限されるわけではありません。セグメント内の一部のフィールドのみで構成されたサブセットを論理ビューとして定義したり (リレーショナル表示に相当)、不要なフィールドを未使用フィールドとして定義したりすることができます。この方法は、たとえばアプリケーションまたはユーザが使用するフィールドを特定のセグメントフィールドのみに制限したい場合に役立ちます。

これらの方法は、次のデータソースタイプに使用します。

- **リレーショナルデータソース** 不要なフィールドをマスターファイルのセグメント記述から除外します。
- **非リレーショナルデータソース** 不要なフィールドを 1 つまたは複数の未使用フィールドとして定義します。

DBA 機能を使用すると、ユーザ ID やフィールド値などの特性に基づいて、ファイル、セグメント、フィールドの各レベルで明示的にアクセスを制限することができます。詳細は、329 ページの 「 データソースのセキュリティ設定 - DBA 」 を参照してください。

例 フィールドの除外 - セグメントサブセットの作成

リレーショナルデータソースの論理ビューを定義するには、マスターファイルのセグメント記述から不要なフィールドを除外します。ここでは、「EMPFACTS」という Oracle テーブルのマスターファイルについて考察します。

```
FILENAME = EMPFACTS, SUFFIX = SQLORA ,$
 SEGNAME = EMPFACTS, SEGTYPE = S0 ,$
  FIELDNAME= EMP_NUMBER, ALIAS= ENUM,  USAGE= A9,      ACTUAL= A9   ,$
  FIELDNAME= LAST_NAME,  ALIAS= LNAME, USAGE= A15,     ACTUAL= A15  ,$
  FIELDNAME= FIRST_NAME, ALIAS= FNAME, USAGE= A10,     ACTUAL= A10  ,$
  FIELDNAME= HIRE_DATE,  ALIAS= HDT,   USAGE= I6YMD,   ACTUAL= DATE ,$
  FIELDNAME= DEPARTMENT, ALIAS= DPT,   USAGE= A10,     ACTUAL= A10  ,$
  FIELDNAME= SALARY,     ALIAS= SAL,   USAGE= D12.2M,  ACTUAL= D8   ,$
  FIELDNAME= JOBCODE,    ALIAS= JCD,   USAGE= A3,      ACTUAL= A3   ,$
  FIELDNAME= OFFICE_NUM, ALIAS= OFN,   USAGE= I8,      ACTUAL= I4   ,$
```

従業員 ID および従業員名フィールドのみを参照するアプリケーションを作成し、アプリケーションのセグメント表示にそれを反映させる場合は、次のように必要なフィールドのみを指定した代替マスターファイルを記述することができます。

```
FILENAME = EMPFACTS, SUFFIX = SQLORA ,$
  SEGNAME = EMPFACTS, SEGTYPE = S0 ,$
    FIELDNAME= EMP_NUMBER, ALIAS= ENUM,  USAGE= A9,  ACTUAL= A9   ,$
    FIELDNAME= LAST_NAME,  ALIAS= LNAME, USAGE= A15, ACTUAL= A15  ,$
    FIELDNAME= FIRST_NAME, ALIAS= FNAME, USAGE= A10, ACTUAL= A10  ,$
```

例 フィールドの再定義 - 未使用フィールドの作成

シーケンシャルや FOCUS などのデータソースで論理ビューを定義するには、表示から除外するフィールドを未使用フィールドとして定義します。フィールドのフォーマットは、タイプは文字、長さは基本となるフィールドを合計したバイト数、名前およびエイリアスはブランクとしてそれぞれを定義します。フィールド宣言およびフィールドの長さについての詳細は、97 ページの 「 フィールドの記述 」 を参照してください。

ここでは、EMPLOYEE データソースの EMPINFO セグメントについて考察します。

```
SEGNAME = EMPINFO, SEGTYPE = S1 ,$
  FIELDNAME = EMP_ID,       ALIAS = EID,  USAGE = A9     ,$
  FIELDNAME = LAST_NAME,    ALIAS = LN,   USAGE = A15    ,$
  FIELDNAME = FIRST_NAME,   ALIAS = FN,   USAGE = A10    ,$
  FIELDNAME = HIRE_DATE,    ALIAS = HDT,  USAGE = I6YMD  ,$
  FIELDNAME = DEPARTMENT,   ALIAS = DPT,  USAGE = A10    ,$
  FIELDNAME = CURR_SAL,     ALIAS = CSAL, USAGE = D12.2M ,$
  FIELDNAME = CURR_JOBCODE, ALIAS = CJC,  USAGE = A3     ,$
  FIELDNAME = ED_HRS,       ALIAS = OJT,  USAGE = F6.2   ,$
```

従業員 ID および従業員名のフィールドのみを参照するアプリケーションを作成し、アプリケーションのセグメント表示にそれを反映させる場合は、次のように必要なフィールドのみを明示的に指定した代替マスターファイルを記述することができます。

```
SEGNAME = EMPINFO, SEGTYPE = S1 ,$
  FIELDNAME = EMP_ID,     ALIAS = EID, USAGE = A9  ,$
  FIELDNAME = LAST_NAME,  ALIAS = LN,  USAGE = A15 ,$
  FIELDNAME = FIRST_NAME, ALIAS = FN,  USAGE = A10 ,$
  FIELDNAME =,            ALIAS =,     USAGE = A29 ,$
```

未使用フィールドは、29 バイトの文字フィールドとして定義します。この長さは、このフィールドで置換されるすべてのフィールドの長さを合計したものに相当します。これらのフィールドには、HIRE_DATE (4 バイト)、DEPARTMENT (10 バイト)、CURR_SAL (8 バイト)、CURR_JOBCODE (3 バイト)、ED_HRS (4 バイト) があります。

複数フィールドグループの関係作成

セグメントを記述した後、複数のセグメントを互いに関係付けて、より合理的なデータ構造を構築することができます。次の処理を実行することができます。

- **物理関係の記述**　フィールドグループがすでにデータソース内で物理的に関係付けられている場合は、その関係を記述することができます。
- **論理関係の記述**　2 つのセグメントに共通フィールドが 1 つ以上存在する場合は、これらのセグメントを結合して両者の論理的な関係を記述します。この場合、基本となるデータ構造は物理的に結合されませんが、あたかも単一データ構造の一部のように扱われます。この新しい構造には、同一タイプのデータソースのセグメントを使用することも、異なるタイプのデータソースのセグメントを使用することもできます。

  FOCUS データソースを新しく作成する場合、設計の目的に合わせてセグメントの関係を複数の方法で構築することができます。詳細は、231 ページの 「 FOCUS データソースの記述 」 を参照してください。

複数のセグメントで構成されたデータ構造を記述する場合、それが複数セグメントのデータソースであるか、結合された複数のデータソースであるかに関係なく、次のことに着目する必要があります。

- 関係セグメント間の論理的な依存性
- 関係を持たないセグメント間の論理的な非依存性

セグメント間の関係指定機能

セグメント間の関係を指定する機能がいくつか用意されています。この章では、マスターファイルおよびアクセスファイルを使用して関係を指定する方法について説明します。JOIN コマンドで複数のセグメントを結合して 1 つの構造を構築し、そこからレポートを実行することもできます。この方法についての詳細は、『TIBCO WebFOCUS Language リファレンス』を参照してください。

関連機能の MATCH FILE コマンドを使用すると、さまざまなタイプの関係を構築することができます。最初に一連の抽出と結合の条件で関係を記述しておき、次に関係付けるデータを結合して新しい単一セグメントデータソースにします。結合した結果は JOIN 構造ではありませんが、完全に新しいデータソースとして、後の処理に使用できるようになります。なお、元のデータソースは変更されずに保持されます。MATCH FILE ファイルについての詳細は、『TIBCO WebFOCUS Language リファレンス』を参照してください。

親セグメントの識別 - PARENT

PARENT 属性を使用して、親セグメントを識別します。PARENT 属性は、マスターファイルのセグメント宣言で指定します。ルートセグメントには親セグメントが存在しないため、ルートを宣言する場合は PARENT セグメントを指定しません。

親セグメントは、マスターファイルで子セグメントより前に宣言しておく必要があります。

FOCUS データソースなどのデータソース構造内で親子関係が永続的に実装されている場合は、データソースの基本的な構造を変更することなく PARENT 属性を変更することはできません。ただし、たとえばマスターファイルで複数のリレーショナルテーブルを結合するなど、一時的な親子関係の場合は、PARENT 属性を変更することができます。

構文　親セグメントの識別

```
PARENT = segment_name
```

説明

`segment_name`
マスターファイルで PARENT 属性を指定しない場合、デフォルト設定では、各セグメントの親セグメントは、マスターファイル内でそれぞれの 1 つ前のセグメントになります。PARENT 属性を指定した場合、明示的に指定しない限り、後に続くすべてのセグメントの親セグメントは、この属性で指定したセグメントになります。

SEGTYPE=U のユニークセグメントには PARENT 属性を使用することをお勧めします。

例　親セグメントの識別

EMPLOYEE データソースでは、DEDUCT の親セグメントが SALINFO であることから、DEDUCT のセグメント宣言には次の属性が指定されています。

```
PARENT = SALINFO
```

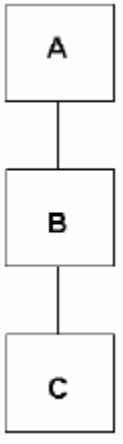
関係タイプの識別 - SEGTYPE

SEGTYPE 属性を使用して、セグメントとその親セグメント間の関係タイプを指定します。SEGTYPE 属性はセグメント宣言の一部として使用しますが、その使用方法はデータソースタイプにより異なります。FOCUS データソースでの使用方法は、231 ページの 「 FOCUS データソースの記述 」 を参照してください。その他のデータソースタイプでの使用方法は、それぞれのデータアダプタのマニュアルを参照してください。

最小参照サブツリーの効率性の理解

複数のテーブルまたはセグメントで構成されたデータベース構造の場合、WebFOCUS はデータベース構造全体のサブセットである最小参照サブツリーのみにアクセスしてデータを取得します。最小参照サブツリーは、参照するフィールドが格納されたセグメントと、全体構造を構築するために必要な中間セグメントで構成されます。

ここでは、次のように 3 つのセグメント A、B、C で構成されたデータベース構造について考察します。A は B の親で、B は C の親です。セグメント A は「ルートセグメント」とも呼ばれます。この関係は、3 つの異なるテーブルが結合された構造とも、単一セグメントまたは複数セグメントで構成された構造とも考えられます。



データベースリクエストで参照されるフィールドがセグメント A にのみ存在する場合は、セグメント A のデータのみが取得されます。同様に、リクエストで参照されるフィールドがセグメント A および B にのみ存在する場合は、セグメント A および B のデータのみが取得されます。各リクエストが 3 つのセグメントすべてを同時に取得した場合ほどの負荷は発生しません。

JOIN 構造では、ルートセグメントが暗黙的に参照されるため、データベースのリクエストでは常にルートセグメントが取得されます。セグメント B のみを参照するリクエストに JOIN 構造が関与している場合は、セグメント B にルートセグメント A が暗黙的に結合しているため、セグメント A と B の両方のデータが取得されます。また、セグメント C のフィールドのみを参照する場合でも、セグメント C にセグメント A と B が暗黙的に結合しているため、3 つのセグメントすべてからデータが取得されます。リクエストを処理する際に中間セグメントも同時に取得する場合は、取得時の負荷が増大します。

1 つのマスターファイルで複数のセグメントを定義した構造では、ルートセグメントを暗黙的に参照することはありません。このタイプの構造に関与するリクエストがセグメント C のフィールドのみを参照する場合、セグメント C のみが取得されます。ただし、セグメント A と C の両方のフィールドを参照する場合は、セグメント B が中間セグメントとして両セグメントの間に介在して全体構造を構築する必要があるため、3 つのセグメントすべてが取得されます。使用する予定があるすべてのデータベース関係を 1 つのマスターファイルに記述しておくと、参照しないセグメントの取得時の負荷を排除することができます。

論理的な依存関係 - 親子関係

セグメント間の論理的な依存関係は親子関係として表現されます。子セグメントはその親セグメントに依存します。子セグメントのインスタンスが存在できるのは、そのインスタンスに関係するインスタンスが親セグメントにも存在する場合に限られます。親セグメントは親子関係において論理的に優先度が高いため、データソースにアクセスした場合は親セグメントが最初に取得されます。

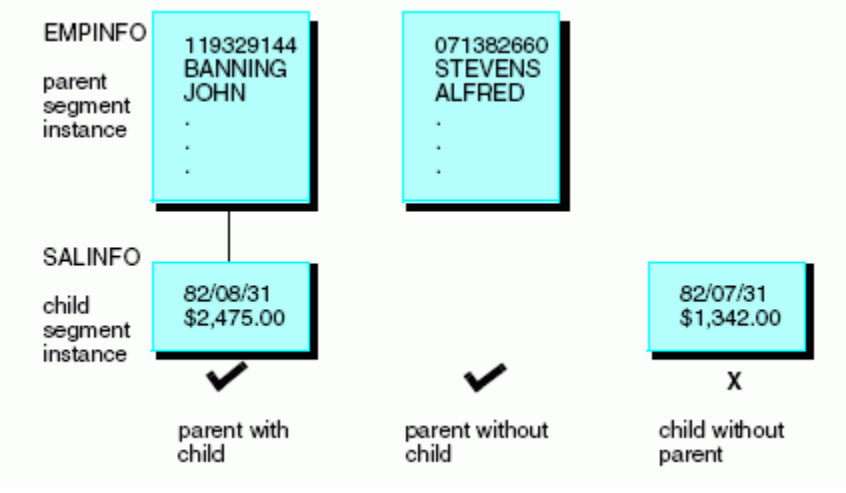
親子関係が物理的な関係ではなく、JOIN 構造のような論理的な関係で成立している場合は、関係する親インスタンスが存在しなくても子インスタンスが存在することができます。この場合、子インスタンスに直接アクセスすることはできても、JOIN 構造を経由してアクセスすることはできません。

JOIN 構造に親子関係のセグメントが関与している場合、その親セグメントは「ホストセグメント」、子セグメントは「クロスリファレンスセグメント」と呼ばれます。JOIN 構造のフィールド、つまりホストセグメントおよびクロスリファレンスセグメントの各フィールドは、それぞれ「ホストフィールド」および「クロスリファレンスフィールド」と呼ばれます。

単純な親子関係

EMPLOYEE データソースでは、EMPINFO セグメントと SALINFO セグメントとの間に関係が成立します。EMPINFO は各従業員の ID 番号であり、SALINFO には各従業員の給与履歴データが格納されています。EMPINFO が親セグメントで、その親セグメントに依存する SALINFO が子セグメントです。この関係では、従業員の ID 番号と名前が存在しても、その従業員の給与情報が入力されていないという状況、つまり、子インスタンスのない親インスタンスが存在するという状況があり得ます。逆に、給与情報は入力されていても、その情報の従業員データが存在しないという状況、つまり、親インスタンスのない子インスタンスが存在する場合、その情報は意味を持ちません。

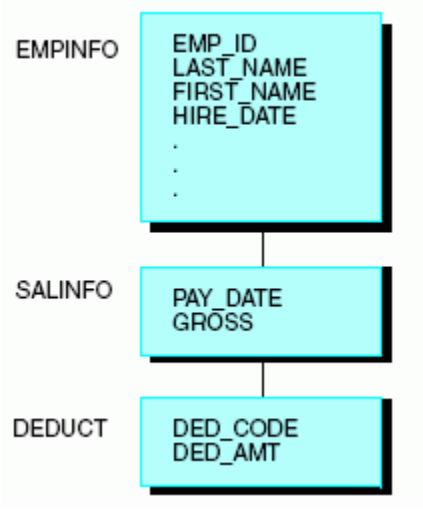
下図は、親子関係の概念を示しています。



複数セグメントの親子関係

一般的な親子関係は、3 つ以上のセグメントで構成されたデータ構造です。ここでは、EMPLOYEE データソースの一部である EMPINFO、SALINFO、DEDUCT の各セグメントの関係について考察します。DEDUCT には、各給与に対する控除額の情報が含まれています。

下図は、複数のセグメントで構成された親子関係の概念を示しています。

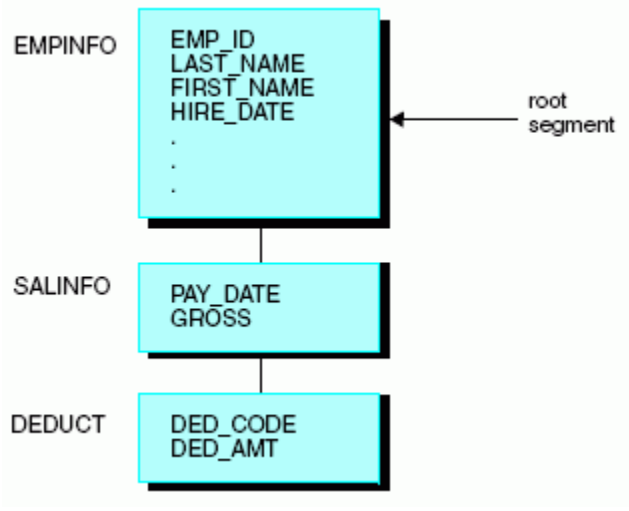


EMPINFO は SALINFO に関係しています。この関係では、EMPINFO が親セグメントで、SALINFO が子セグメントになります。また、SALINFO は DEDUCT とも関係しています。この 2 つ目の関係では、SALINFO が親セグメントで、DEDUCT が子セグメントになります。SALINFO が EMPINFO に依存するように、DEDUCT は SALINFO に依存します。

ルートセグメントの理解

データ構造全体で論理的に優先度が高いセグメント、つまり構造全体の親セグメントは「ルートセグメント」と呼ばれます。これはデータ構造がツリー状に分岐され、ツリーの根元にあたるルートセグメントがその構造の起点になるためです。

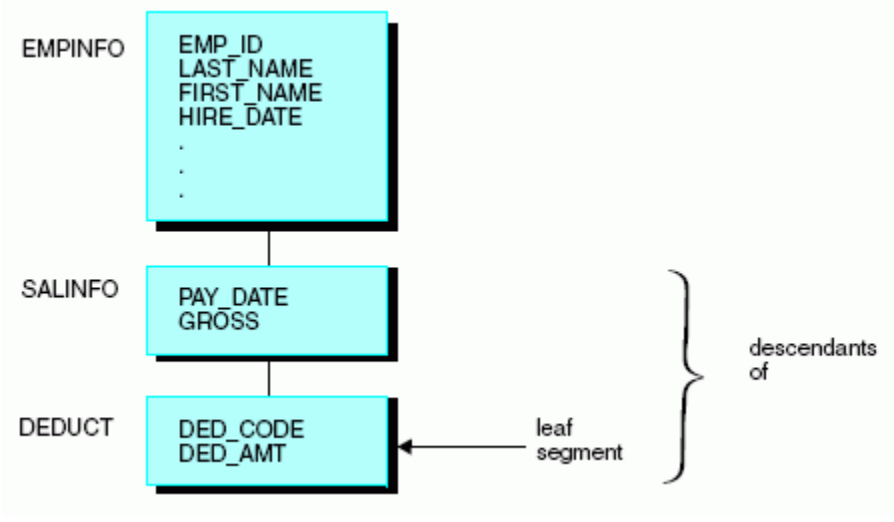
下図では、EMPINFO がルートセグメントです。このセグメントの親セグメントは存在せず、この構造の他のすべてのセグメントは直接的 (SALINFO) または間接的 (DEDUCT) にその子セグメントになります。



下位セグメントの理解

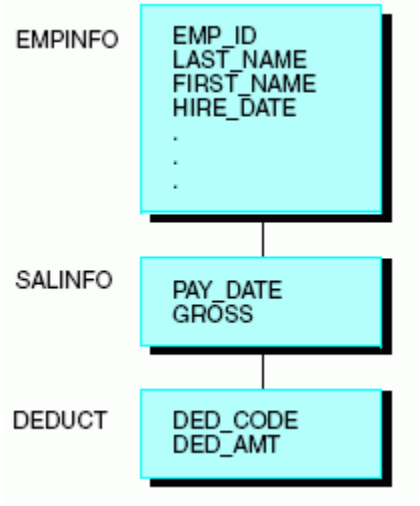
セグメントの直接的、間接的な子セグメントは、総称して「下位セグメント」と呼ばれます。上記の例では、SALINFO および DEDUCT は、EMPINFO の下位セグメントです。また、DEDUCT は SALINFO の下位セグメントです。子セグメントを持たない下位セグメントは、分岐したデータ構造ツリーの末端が「葉 (リーフ)」の部分に相当するため、「リーフセグメント」と呼ばれます。DEDUCT はリーフセグメントです。

下図は、下位セグメントの概念を示しています。



上位セグメントの理解

セグメントの直接的、間接的な親セグメントは、そのセグメントの「上位セグメント」と呼ばれます。下図では、SALINFO および EMPINFO は DEDUCT の上位セグメントです。

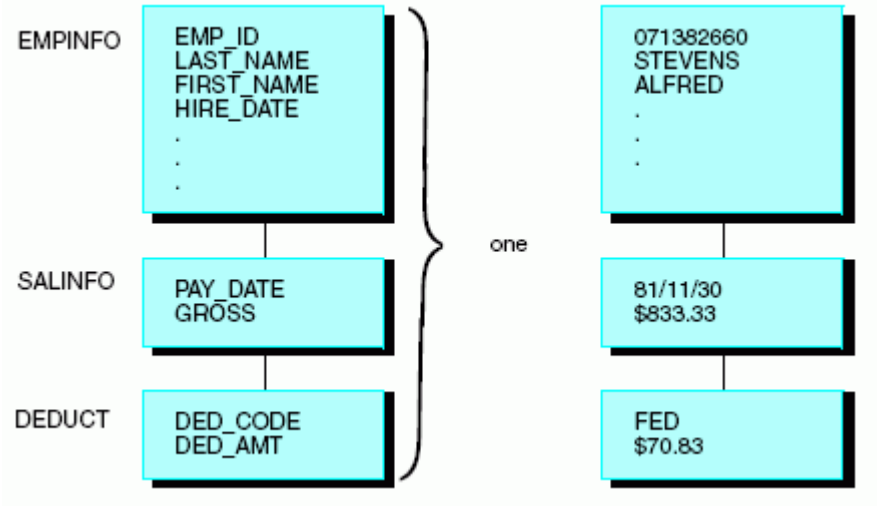


論理的な非依存関係 - 複数パス

ルートセグメントからリーフセグメントに至るまで、親子関係の系列として互いに関係付けられたセグメントのグループは「パス」と呼ばれます。パスが親子関係の系列を表すことから、各セグメントはそのパスの中で上位にあるすべてのセグメントに論理的に依存しているといえます。

単一パスの理解

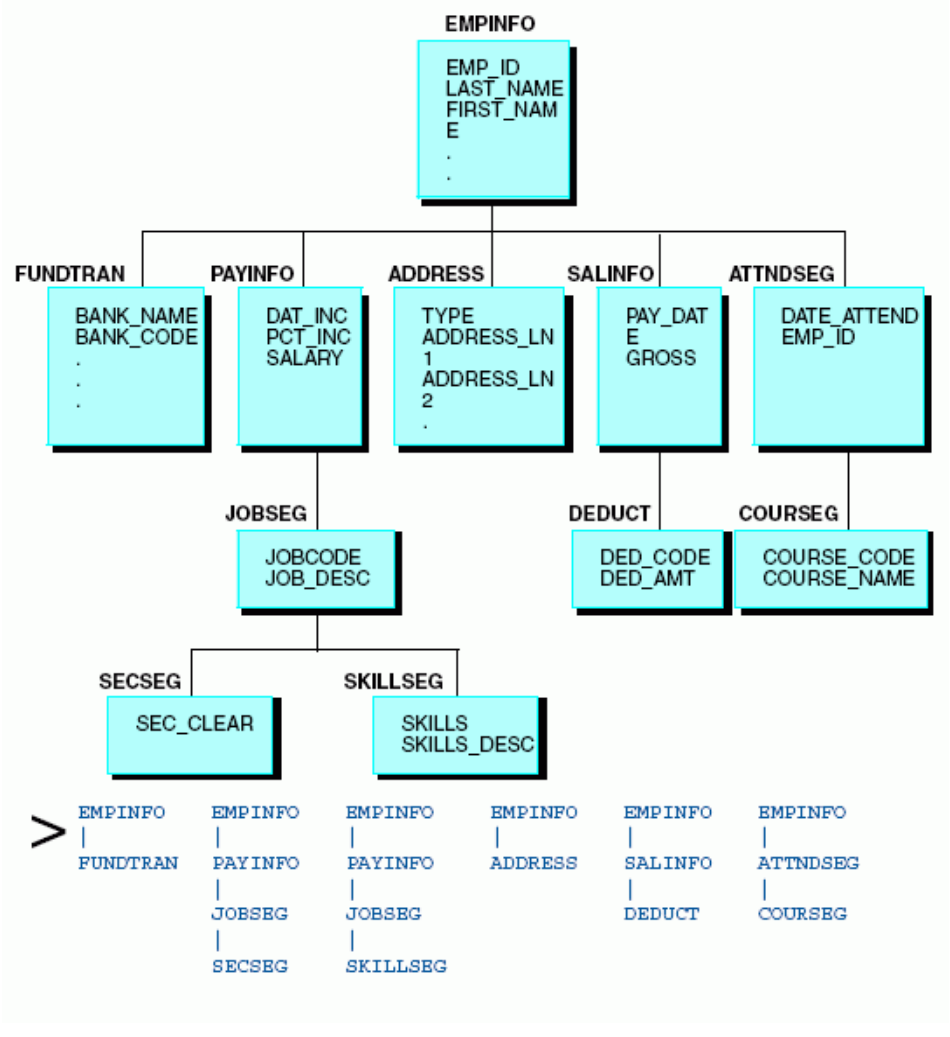
次に示す EMPLOYEE データソースの例では、EMPINFO、SALINFO、DEDUCT の 3 つのセグメントによりパスが形成されます。DEDUCT (給与控除) のインスタンスが存在できるのは、関係するインスタンスが SALINFO (給与) に存在する場合に限られます。同様に、SALINFO のインスタンスが存在できるのは、関係するインスタンスが EMPINFO (従業員) に存在する場合に限られます。



複数パスの理解

ここでは、EMPLOYEE の全体構造について考察します。この構造は、EMPLOYEE データソース、それに結合された JOBFILE および EDUCFILE データソースで構成されています。

この構造は、複数パスのデータ構造です。この構造は複数のパスで構成され、それぞれのパスがルートセグメントで始まり、リーフセグメントで終了します。リーフセグメントは、それぞれのパスの末端に位置するセグメントです。下図は、複数パスのデータ構造の概念を示しています。



論理的な非依存関係の理解

上記の EMPLOYEE データ構造は、6 つのパスで構成されています。これらのパスは EMPINFO セグメント (ルート) で始まり、次のセグメントで終了します。

- FUNDTRAN セグメント
- SECSEG セグメント
- SKILLSEG セグメント
- ADDRESS セグメント
- DEDUCT セグメント
- COURSEG セグメント

それぞれのパスは、互いに論理的に非依存の関係にあります。たとえば、DEDUCT のインスタンスはその上位セグメントの SALINFO および EMPINFO のインスタンスに依存しますが、ADDRESS セグメントは別のパスに存在するため、DEDUCT は ADDRESS に依存しません。

これは、従業員の控除額がその計算元となる給与額により識別され、給与額を先に入力してからでないと控除額の情報をデータソースに入力できないためです。一方、控除額は従業員の住所では識別されません。従業員の給与控除額は、住所が分からなくても入力することができます。逆に、給与と控除額をデータソースに入力する前に、住所情報を入力することもできます。

セグメント間の基本的な関係

データグループ間の基本的な関係としてサポートされているタイプには次のものがあります。

- 1 対 1 (1:1)
- 1 対 n (1:n)
- n 対 n (n:n)

これらの関係は、次のデータ間で定義することができます。

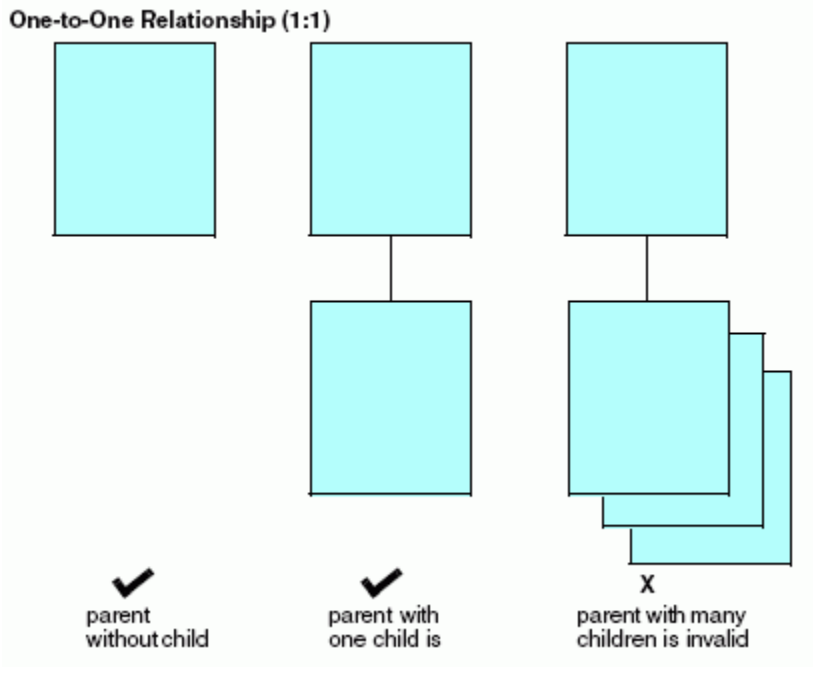
- 異なるセグメントのインスタンス。
- 同一セグメントのインスタンス (再帰的な関係)。
- 同一データソースタイプのセグメント。

- 異なるデータソースタイプのセグメント。たとえば、Oracle テーブルと FOCUS データソースの間で関係を定義することができます。なお、異なるタイプのデータソースを結合する場合は、マスターファイルまたはアクセスファイルで JOIN を定義するのではなく、JOIN コマンドを使用する必要があります。
- ネットワーク型のデータソースを使用する場合は、定義したデータソースを「回転」させて代替ビューを作成し、データ関係のいくつかを反転させたり、異なる順序でセグメントにアクセスしたりすることもできます。

1 対 1 の関係

セグメント内ではフィールドとフィールドが互いに 1 対 1 の関係になっています。また、セグメントとセグメントが互いに 1 対 1 の関係になる場合もあります。次の図では、親セグメントの 1 つのインスタンスが、子セグメントの 1 つのインスタンスに関係しています。これが 1 対 1 の関係であることから、1 つの親インスタンスが 2 つ以上の子インスタンスと関係することはありません。ただし、親インスタンスの中には、そのインスタンスに対応する子インスタンスが存在しない場合もあります。

1 対 1 の関係にある子セグメントは、子インスタンスが 2 つ以上存在することはないため「ユニークセグメント」と呼ばれます。下図は、1 対 1 の関係の概念を示しています。

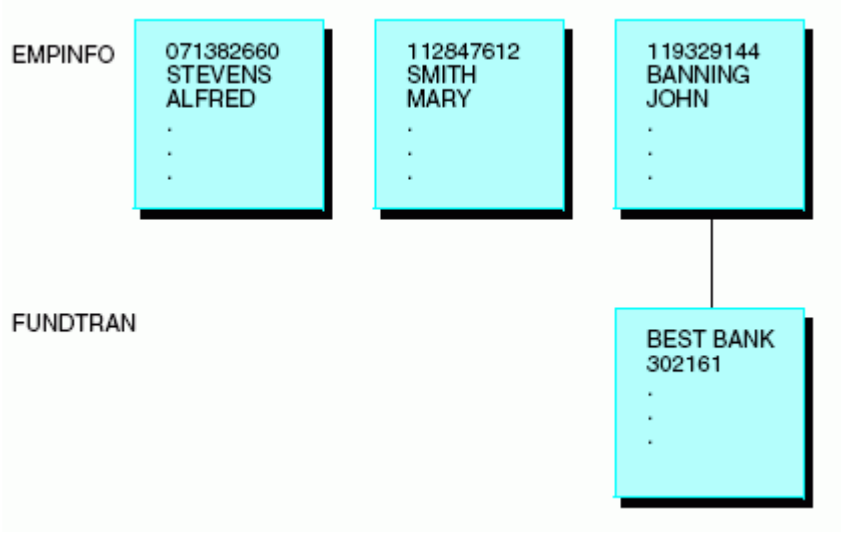


例　1 対 1 関係の理解

EMPLOYEE データソースには、EMPINFO セグメントの各インスタンスに、各従業員の ID 番号、名前、現在の給与、その他の関連情報が記述されています。従業員の中には、銀行への振り込み制度に加入して、給与を自分の銀行口座に直接振り込むようにしている人もいます。これらの従業員に対しては、データソースに銀行名と口座番号の情報も保存されています。

銀行情報は各従業員に対して 1 つだけ必要なことから (各従業員の給与振込み先は 1 つの口座に限定されている)、従業員 ID と銀行情報の 2 つのフィールドには 1 対 1 の関係が成立します。この振り込み制度に加入した人が限られているため、すべての従業員が銀行情報を持っているわけではありません。ほとんどの従業員には、銀行情報のフィールドを使用する必要はありません。

このデータソースは、格納効率を考慮して設計されているため、銀行情報のフィールドは「FUNDTRAN」と呼ばれる別のセグメントに定義されています。空き領域は銀行情報のみに使用され、必要に応じて FUNDTRAN のインスタンスが作成されます。ただし、銀行情報のフィールドが親セグメント (EMPINFO) で使用される場合、これらのフィールドのほとんどが空の状態であっても、各従業員の EMPINFO セグメントにはこのフィールド用の領域が確保されます。下図は、この概念を示しています。



1 対 1 関係の使用

データを取得する際に、ユニークセグメントであることを指定して、強制的に 1 対 1 の関係を構築することができます。

ユニークセグメントとして指定されたセグメントからデータを取得する場合、リクエストはそのセグメントに親セグメントが存在するものとして取り扱います。ユニークセグメントに複数のインスタンスが存在する場合は、リクエストはそのうち 1 つだけを取得します。ユニークセグメントにインスタンスが 1 つも存在しない場合、リクエストはそのセグメントフィールドにミッシングのデフォルト値を代入します。デフォルト値は、数値フィールドは 0 (ゼロ)、文字フィールドはブランク、MISSING 属性が指定されたフィールドはそのミッシング値です。MISSING 属性についての詳細は、97 ページの 「 フィールドの記述 」 を参照してください。

リレーショナルデータソースでの 1 対 1 関係の実装

この関係を記述するには、マスターファイルで複数のテーブルを結合し、子テーブルに対して SEGTYPE=U を指定します。マスターファイルでテーブルを結合する方法についての詳細は、該当するデータアダプタのマニュアルを参照してください。テーブルを結合する別の方法として、ALL (または MULTIPLE) オプションを使用せずに (または UNIQUE オプションを使用して) JOIN コマンドを発行し、SET OPTIMIZATION コマンドで SQL 最適化機能をオフにすることもできます。

シーケンシャルデータソースでの 1 対 1 関係の実装

2 つのレコード間でこの関係を指定するには、ALL (または MULTIPLE) オプションを使用せずに (または UNIQUE オプションを使用して) JOIN コマンドを発行します。

FOCUS データソースでの 1 対 1 関係の実装

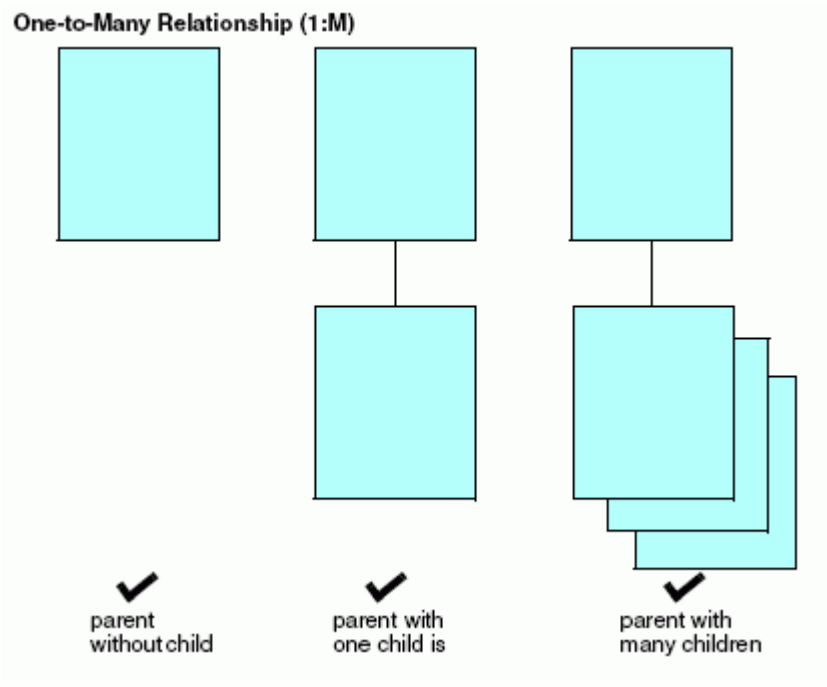
この関係を記述するには、子セグメントに対して SEGTYPE=U を指定します。セグメントを結合する別の方法として、ALL (または MULTIPLE) オプションを使用せずに (または UNIQUE オプションを使用して) JOIN コマンドを発行したり、SEGTYPE=KU (静的 JOIN) または SEGTYPE=DKU (動的 JOIN) を使用してマスターファイルでユニーク JOIN を指定したりすることもできます。SEGTYPE の値についての詳細は、231 ページの 「 FOCUS データソースの記述 」 を参照してください。

また、マスターファイルに関係を記述するか、JOIN コマンドを使用して、セグメントの 1 対 1 の関係を 1 対 n の関係として記述することもできます。この方法は柔軟性に富んでいますが、レポートの実行またはデータの入力時に強制的に 1 対 1 の関係を構築せず、効率的にリソースを使用するものではありません。

1 対 n の関係

2 つのセグメント間で最も一般的な関係は、1 対 n の関係です。親セグメントの 1 つのインスタンスは、子セグメントの 1 つまたは複数のインスタンスと関係を形成することができます。ただし、親インスタンスの中には、そのインスタンスに対応する子インスタンスが存在しない場合もあります。

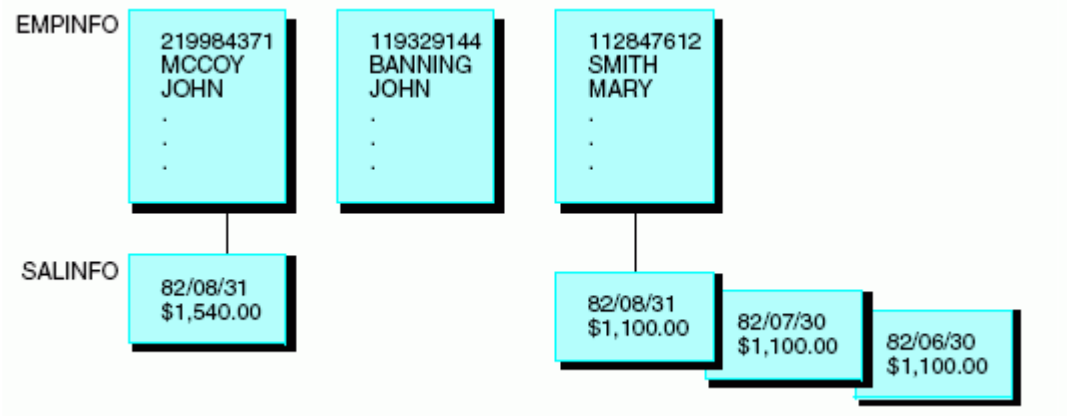
下図は、1 対 n の関係の概念を示しています。



例　1 対 n 関係の理解

EMPLOYEE データソースには、EMPINFO セグメントの各インスタンスに、各従業員の ID 番号、名前、現在の給与、その他の関連情報が記述されています。SALINFO セグメントの各インスタンスには、各従業員の毎月の給与総額が入力されています。ほとんどの従業員は何か月かの勤務実績があるため、EMPINFO と SALINFO の関係は 1 対 n になります。

下図は、1 対 n の関係の概念を示しています。



リレーショナルデータソースでの 1 対 n 関係の実装

この関係を記述するには、マスターファイルで複数のテーブルを結合し、子テーブルに対して SEGTYPE=S0 を指定します。マスターファイルでテーブルを結合する方法についての詳細は、該当するデータアダプタのマニュアルを参照してください。テーブルを結合する別の方法として、ALL または MULTIPLE オプションを使用して JOIN コマンドを発行することもできます。

シーケンシャルデータソースでの 1 対 n 関係の実装

1 つのレコードとそのレコード内で複数回出現するフィールドとの間に 1 対 n の関係を記述することができます。

- OCCURS 属性 - 特定のフィールドの出現回数を指定します。
- POSITION 属性 - 特定のフィールドの位置がレコードの末尾以外の場合に、そのフィールドがレコード内で出現する位置を指定します。
- ORDER フィールド - 特定のフィールドの出現ごとに連続番号を指定します。
- PARENT 属性 - 1 回または複数回出現するフィールド間の関係を指定します。

異なるレコード間に 1 対 n の関係を記述するには、RECTYPE フィールドを使用して各レコードのタイプを指定し、PARENT 属性で異なるレコード間の関係を指定します。

異なるデータソースの 2 つのレコード間に 1 対 n の関係を指定するには、ALL または MULTIPLE オプションを使用して JOIN コマンドを発行するか、マスターファイルで JOIN を定義します。JOIN コマンドについての詳細は、『TIBCO WebFOCUS Language リファレンス』を参照してください。また、マスターファイルで JOIN を定義する方法についての詳細は、271 ページの 「 マスターファイルでの JOIN の定義 」 を参照してください。

FOCUS データソースでの 1 対 n 関係の実装

この関係を記述するには、子セグメントに対して SEGTYPE=Sn または SEGTYPE=SHn を指定します。セグメントを結合する別の方法として、ALL または MULTIPLE オプションを使用して JOIN コマンドを発行するか、SEGTYPE=KM (静的 JOIN) または SEGTYPE=DKM (動的 JOIN) を使用してマスターファイルで JOIN を指定することもできます。SEGTYPE の値についての詳細は、231 ページの 「 FOCUS データソースの記述 」 を参照してください。

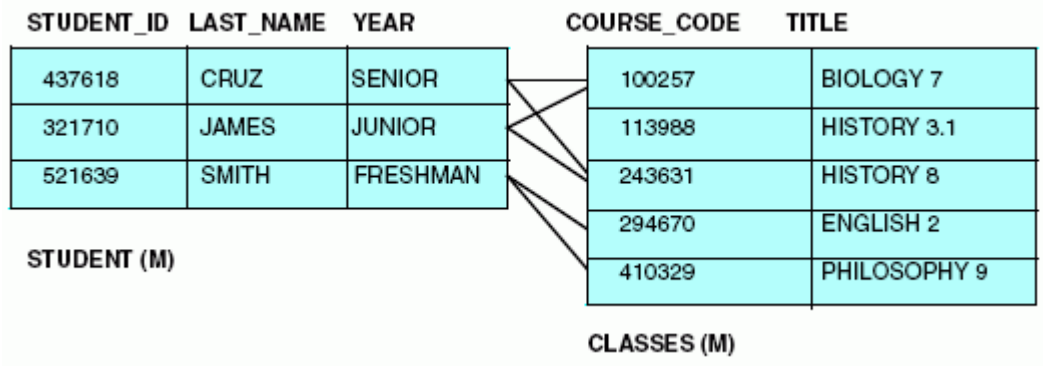
n 対 n の関係

一般に、n 対 n の関係はあまり使用されません。この関係では、1 つ目のセグメントの各インスタンスを 2 つ目のセグメントの 1 つまたは複数のインスタンスと関係させたり、2 つ目のセグメントのインスタンスを 1 つ目のセグメントの 1 つまたは複数のインスタンスに関係させたりすることができます。この関係は、2 つのリレーショナルテーブル間では直接実装し、異なるデータソースタイプ間では間接実装することができます。

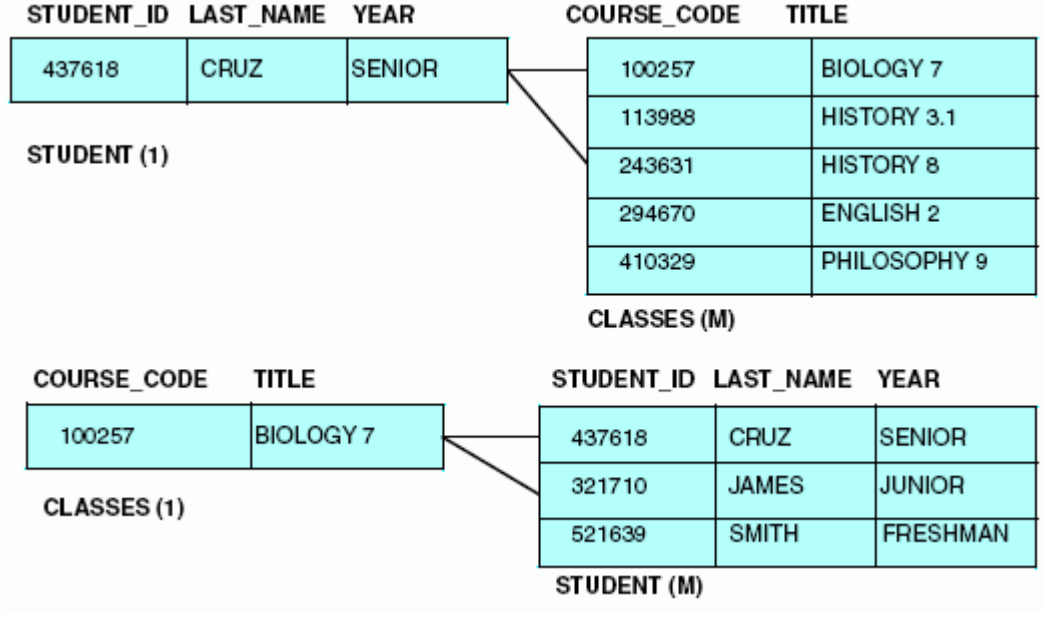
n 対 n 関係の直接実装

2 つのリレーショナルテーブル間では、n 対 n の関係を直接実装することができます。次の例では、大学に在籍する各学生の情報が STUDENT テーブルの 1 行に、大学で開講される各クラスの情報が CLASSES テーブルの 1 行に格納されています。1 名の学生が複数のクラスを受講することも、1 つのクラスを複数の学生が受講することもできます。

下図は、n 対 n の関係の概念を示しています。



n 対 n の関係を 2 つのテーブルのどちらか一方から見た場合、1 対 n の関係のように見えます。たとえば、STUDENT テーブルから見た場合、1 名の学生が複数のクラスを受講していることから 1 対 n の関係に見えます。また、CLASSES テーブルから見た場合、1 つのクラスを複数の学生が受講していることから 1 対 n の関係に見えます。下図は、このタイプの関係を示しています。



テーブルからレポートを実行したり、テーブルを更新したりする場合、n 対 n の関係をどちらか一方のテーブルから見ることができます。つまり、n 対 n の関係はどの時点においても 1 対 n の関係で見ることが可能です。どちらのテーブルから見るかを決定するには、マスターファイルまたは JOIN コマンドでそのテーブルを親セグメント (ホストセグメント) に指定します。通常の 1 対 n の関係の場合と同様に、マスターファイルで JOIN を記述するか、JOIN コマンドを使用します。

例　n 対 n 関係の直接実装

STUDENT テーブルから見た場合の関係を記述するには、次のように JOIN コマンドを使用します。

```
JOIN STUDENT_ID IN STUDENT TO ALL STUDENT_ID IN CLASSES
```

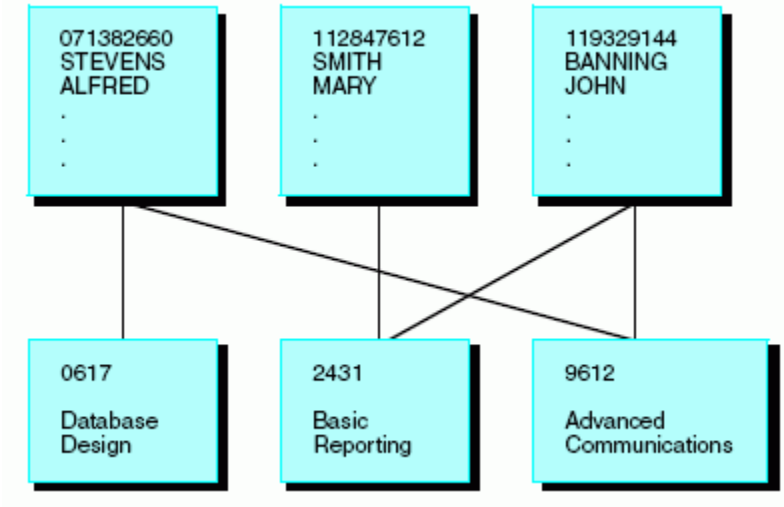
逆に、CLASSES テーブルから見た場合の関係を記述するには、次のように記述します。

```
JOIN COURSE_CODE IN CLASSES TO ALL COURSE_CODE IN STUDENT
```

n 対 n 関係の間接実装

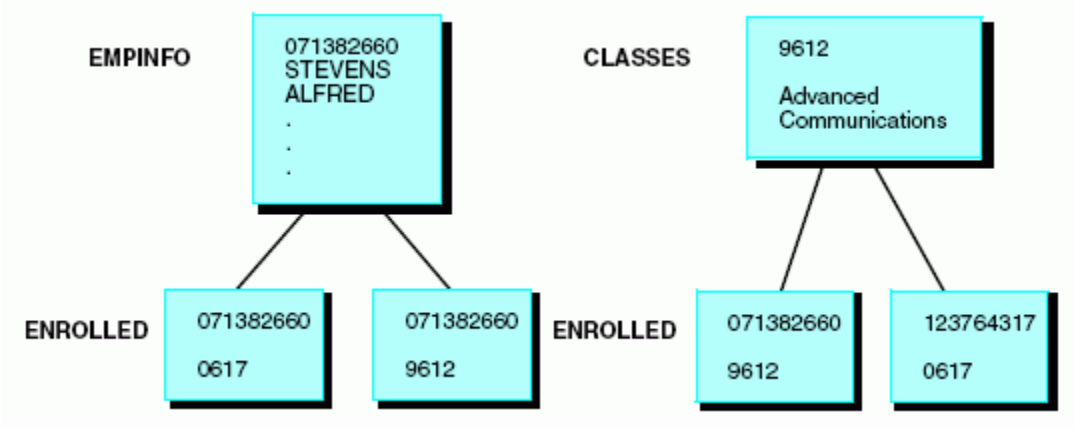
非リレーショナルデータソースの中には、n 対 n の関係を直接実装できないものがあります。そのようなデータソースでも、n 対 n の関係を間接実装して表すことができます。

ここでは、EMPLOYEE データソースの EMPINFO セグメントおよび架空の SCHOOL データソースの CLASSES セグメントについて考察します。EMPINFO のインスタンスには各従業員の情報、CLASSES のインスタンスには各コースの情報が格納されています。1 名の従業員が複数のコースを受講することも、1 つのコースを複数の従業員が受講することもできます。つまり、n 対 n の関係になっています。下図は、このタイプの関係を示しています。



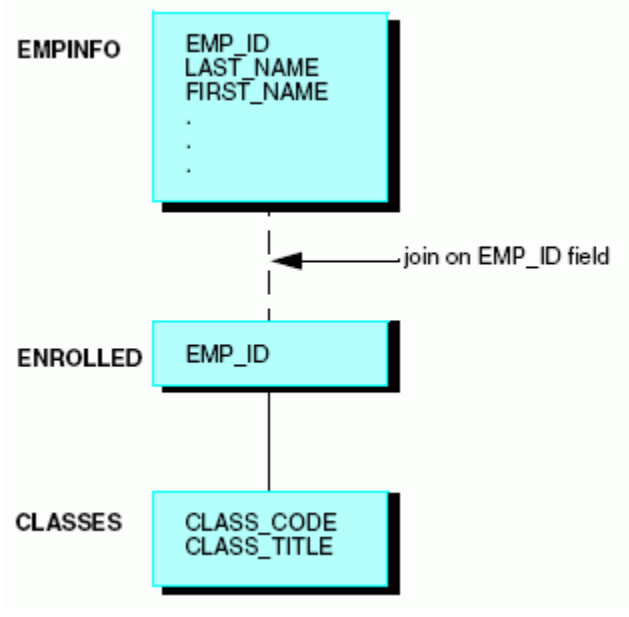
なお、データソースのタイプによってはこのような関係を直接表現できないものがあるため、その場合は「ENROLLED」と呼ばれる中間セグメントを導入する必要があります。この新しいセグメントには、オリジナルセグメントである EMP_ID および CLASS_CODE のキーが含まれており、これにより 2 つのオリジナルセグメント間の関係が表されます。このセグメントにより、1 つの n 対 n の関係が 2 つの 1 対 n の関係に分割されます。1 名の従業員が複数のクラスを受講できるため、EMPINFO の 1 つのインスタンスを ENROLLED の複数のインスタンスに関係付けることができます。同様に、1 つのクラスを複数の従業員が受講できるため、CLASSES の 1 つのインスタンスを ENROLLED の複数のインスタンスに関係付けることができます。

下図は、これらの関係を示しています。



次に、この中間セグメントを 2 つのオリジナルセグメントのどちらかの子セグメントとして指定します。たとえば、CLASSES をルートセグメントとし、その子セグメントを ENROLLED とするように SCHOOL データソースを設計することができます。なお、2 つのオリジナルセグメントのキー (EMP_ID および CLASS_CODE) は、ENROLLED をまだ結合していない段階で明示的に記述したものです。SCHOOL データソースの一部として CLASS_CODE が CLASSES との親子関係で暗黙的に実装されたため、ENROLLED から CLASS_CODE を削除することができます。続いて EMPINFO と ENROLLED を結合することができます。

下図は、このタイプの結合を示しています。



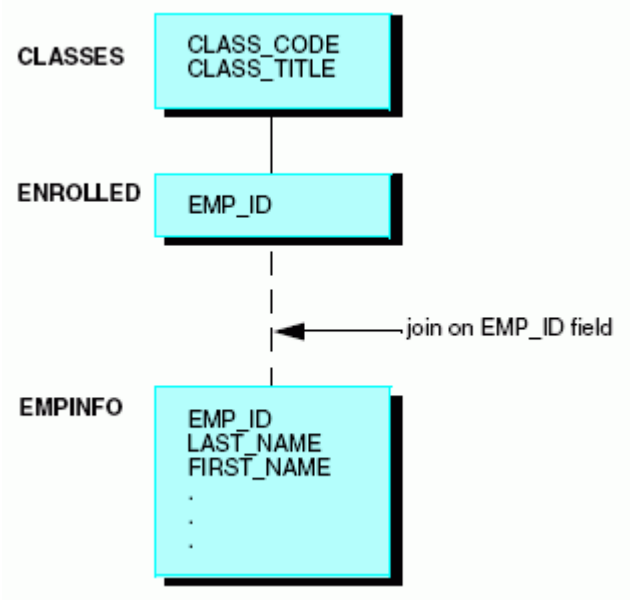
元の n 対 n の関係をこの視点から見た場合、1 対 n 対 1 の関係のように見えます。つまり、1 名の従業員が (EMPINFO) 受講する回数 (ENROLLED)、および 1 つのクラスの受講者数 (CLASSES) に関係が成立します。

新しい構造からレポートを実行したり、構造を更新したりする場合、どの時点でもどちらか一方のオリジナルセグメントからこの関係を見ることができます。この場合は、EMPINFO、CLASSES のいずれかの視点です。どのセグメントの視点で見るかを決定するには、そのセグメントを JOIN の親セグメントに指定します。JOIN を記述するには、JOIN コマンドを使用するか、FOCUS データソースではマスターファイルを使用します。この場合の ENROLLED にあたる中間セグメントを JOIN の子セグメント (クロスリファレンスセグメント) として指定するには、通常の 1 対 n として関係を実装します。また、中間セグメントを親セグメント (ホストセグメント) として指定するには、通常の 1 対 1 の JOIN として関係を実装します。

たとえば、JOIN コマンドを使用して CLASSES セグメントの視点から関係を記述し、ENROLLED をその JOIN のホストセグメントにすることができます。

```
JOIN EMP_ID IN ENROLLED TO EMP_ID IN EMPINFO
```

下図は、新しい構造を示しています。



マスターファイルで定義した JOIN を使用する例は、サンプルの FOCUS データソースの EMPLOYEE と EDUCFILE にも示されています。その例では、EMPINFO と COURSEG の中間セグメントは ATTNDSEG です。

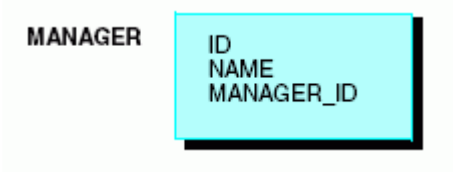
再帰的な関係

通常、異なるデータソースに存在する 2 つのセグメントを結合するには、1 対 1 の関係および 1 対 n の関係を使用します。場合によっては、データソースをそのデータソース自体に結合したり、セグメントをそのセグメント自体に結合したりすることもあります。この方法は「再帰的 JOIN」と呼ばれます。

再帰的 JOIN についての詳細は、『TIBCO WebFOCUS Language リファレンス』を参照してください。

例 単一セグメントの再帰的 JOIN

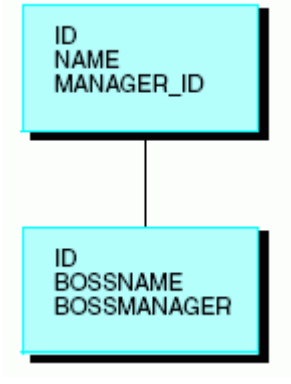
ここでは、「MANAGER」という単一セグメントのデータソースについて考察します。下図のように、このデータソースには従業員の ID 番号、従業員の名前、マネージャの ID 番号が格納されています。



各従業員の ID 番号と名前、各マネージャの ID 番号と名前を表示するレポートを作成する場合、セグメントをこのセグメント自体に結合する必要があります。次のコマンドを発行します。

```
JOIN MANAGER_ID IN MANAGER TO ID IN MANAGER AS BOSS
```

次のような構造が作成されます。



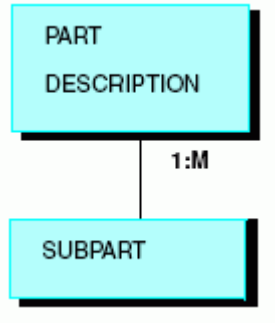
注意：JOIN 名の最初の 4 バイトを接頭語として使用すると (この例では BOSS)、クロスリファレンス再帰的セグメントでフィールドを一意に参照できるようになります。ただし、例外としてクロスリファレンスフィールドに対しては、フィールド名の代わりにエイリアスが接頭語として使用されます。

JOIN コマンドを発行後、次のようなアンサーセットが生成されます。

```
ID      NAME              MANAGER_ID     BOSSNAME
--      ----              ----------     --------
026255  JONES             837172         CRUZ
308743  MILBERG           619426         WINOKUR
846721  YUTANG            294857         CAPRISTI
743891  LUSTIG            089413         SMITH
585693  CAPRA             842918         JOHNSON
```

例　複数セグメントの再帰的 JOIN

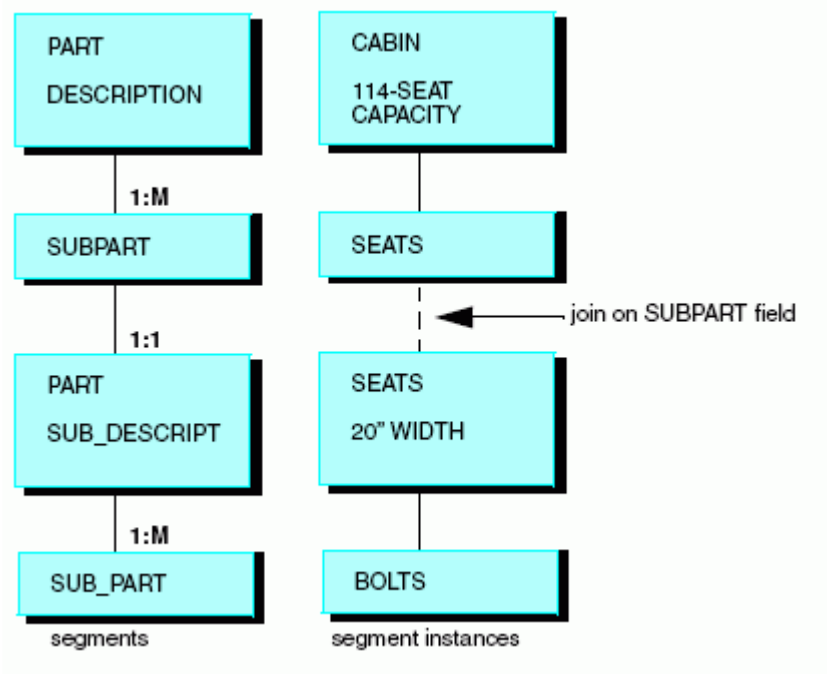
再帰的 JOIN は大規模な構造にも使用することができます。ここでは、航空会社の部品表が格納された「AIRCRAFT」という 2 つのセグメントで構成されたデータソースについて考察します。ルートセグメントには部品の名前と説明、子セグメントにはその構成部品の名前が格納されています。1 つの部品に対して複数の構成部品が存在する場合もあります。下図は、このタイプの JOIN 構造を示しています。



飛行機の組み立て工程において大規模部品の多くが特定の工程レベルの構成部品から組み立てられるの対して、これらの構成部品 (例、ボルト) のいくつかはさまざまな工程レベルで広範囲に使用されます。それぞれの構成部品を個別のセグメントインスタンスとして取り扱うと、重複する部分が増大します。そこで、前述のように 2 つのセグメントで構成される構造を利用して、このデータソースをこのデータソース自体に結合します。

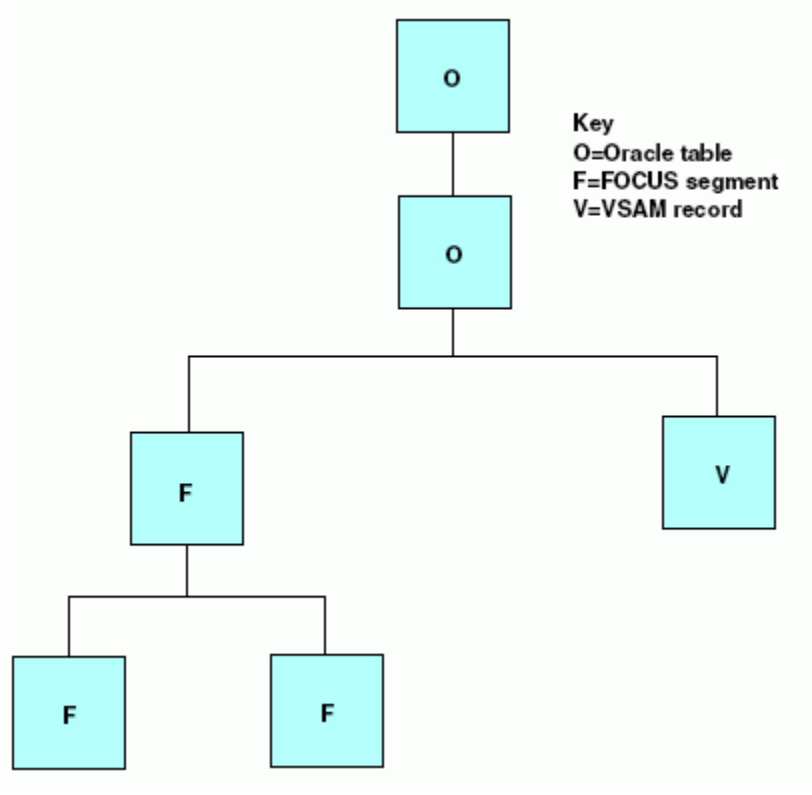
```
JOIN SUBPART IN AIRCRAFT TO PART IN AIRCRAFT AS SUB_PART
```

この結合により、次のようなデータ構造が作成されます。



異なるデータソースタイプのセグメントの関係付け

JOIN コマンドを使用すると、異なるデータソースのセグメントを結合して一時的なデータ構造を作成し、互換性のないソースの関連情報をその構造に含めることができます。たとえば、下図のように 2 つの Oracle データソースを FOCUS データソースに結合することができます。

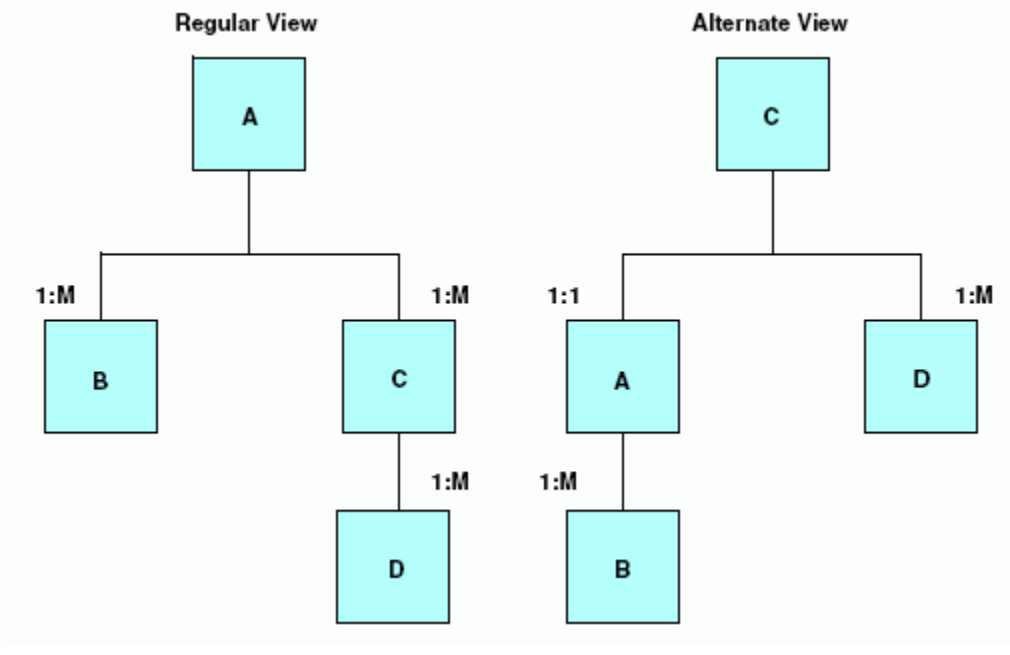


固定フォーマットデータソースの結合は、マスターファイルでもサポートされます。この方法についての詳細は、271 ページの 「 マスターファイルでの JOIN の定義 」 を参照してください。

異なるデータソースタイプで JOIN コマンドを使用する方法についての詳細は、『TIBCO WebFOCUS Language リファレンス』を参照してください。

データソースの回転 - 代替ビュー

ネットワーク型データソースまたは FOCUS などの特定の階層型データソースを使用する場合、データソースを定義した後でそれを回転させることができます。この回転によりセグメントの関係を切り替える代替ビューが作成されるため、異なる順序でセグメントにアクセスすることができます。この方法でアクセスパスをカスタマイズすれば、特定のアプリケーションで、より簡単にアクセスできるようになります。下図は、このタイプの代替ビューを示しています。



階層型およびネットワーク型のデータソースを結合した後、JOIN 構造の代替ビューを作成するには、ホストデータソースから新しいルートセグメントを選択します。

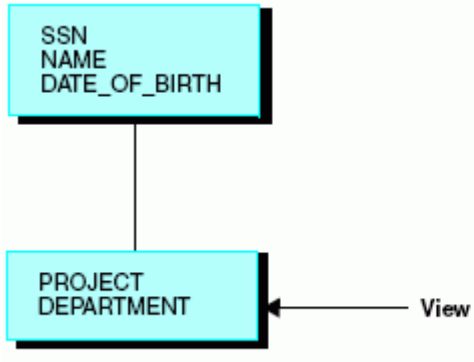
レポートを作成する際に下位のセグメント (上図のセグメント C など) に存在するフィールドを基準にレコード選択条件を使用する場合は、この代替ビューを使用すると役立ちます。レポートは、この下位セグメントをルートセグメントとする代替ビューから作成することができます。FOCUS は、関係のない上位セグメントを読み込まずに、関係のあるセグメントを基準にレコード選択を開始します。

代替ビューを使用してデータソースからレポートを作成する場合、次の両方の条件を満足すれば、より効率的にデータにアクセスすることができます。

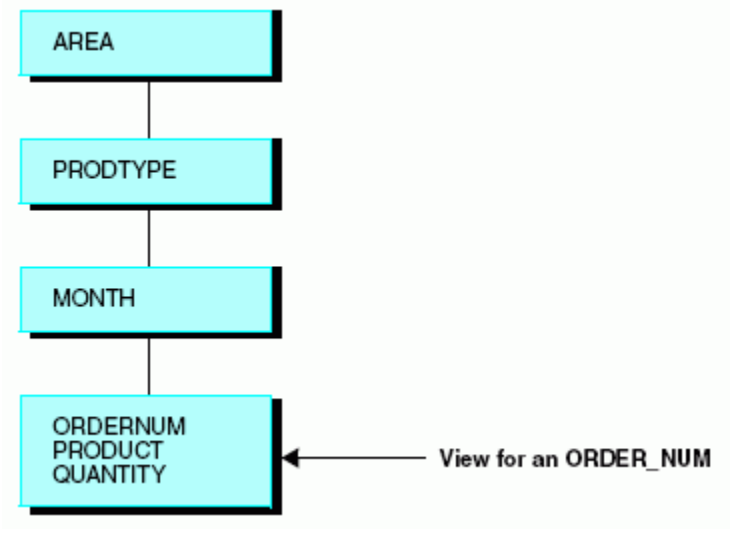
- 代替ビューの基準となるフィールドがインデックスフィールドである。FOCUS データソースの場合、代替ビューフィールドにはマスターファイルで INDEX = I を指定しておく必要があります。
- WHERE 句または IF 句を使用し、等価または範囲テストを選択基準とするレコード選択テストでフィールドを使用する。

代替ビューのリクエストは、クロスリファレンスセグメントを除いて、データソース内の任意のセグメントで行うことができます。TABLE コマンドを使用して代替ビューをリクエストするには、新しいルートセグメントとして表示するセグメントのフィールド名を指定します。代替ビューをリクエストする際の唯一の制約は、リクエストの対象となるフィールドがデータソース内の実フィールドであることです。一時項目 (DEFINE) を使用することはできません。

下図は、このタイプの代替ビューを示しています。



下図は、このタイプの代替ビューを詳細に示しています。



構文　代替ビューの指定

次の構文を使用して、レポートコマンドのファイル名にフィールド名を追加します。

```
TABLE FILE filename.fieldname
```

説明

`filename`

代替ビューを定義するデータソースの名前です。

`fieldname`

代替のルートセグメントとして定義するセグメント内のフィールドです。このフィールドには実フィールドを指定する必要があります。DEFINE 属性、DEFINE コマンド、COMPUTE コマンドで定義した一時項目を指定することはできません。

マスターファイルのフィールド宣言で FIELDTYPE=I 属性を指定し、レポートで代替ビューを使用する場合、等価 (例、EQ) または範囲指定の選択テストでこのフィールドを使用する必要があります。

例　代替ビューの指定

EMPLOYEE データソースからレポートを作成する際に、DEDUCT セグメントを代替ルートセグメントとする代替ビューを使用するには、次の TABLE FILE コマンドを使用します。

```
TABLE FILE EMPLOYEE.DED_CODE
```

フィールドタイトル接頭語の定義

マスターファイルのフィールドに TITLE 属性が定義されている場合、このタイトルがレポート出力ファイルの列見出しとして使用されます。ただし、リクエストで AS 名が指定されている場合を除きます。

マスターファイルのセグメントレベルで、このセグメントのフィールドタイトル接頭語を定義することができます。この機能は、クラスタマスターファイルでベースシノニムが複数回使用されるなど、異なるセグメントのフィールド名に同一タイトルが付けられている場合に、レポート出力に表示されているフィールドの識別に役立ちます。

参照　フィールドタイトル接頭語の定義

```
SEGMENT=segname, ... , SEG_TITLE_PREFIX='prefix', $
```

説明

`segname`
有効なセグメント名です。

`'prefix'`
レポート出力のフィールドタイトルの接頭語として使用するテキストです。SEG_TITLE_PREFIX と TITLE string の合計の長さが 512 バイトを超えることはできません。テキストをカンマ (,) で区切ることにより、最大で 5 つのタイトル行に分割することができます。フィールドタイトルの末尾にブランクを含める場合は、そのブランクの位置にスラッシュ (/) を配置します。カンマ (,) または先頭にブランクを含む文字列は、一重引用符 (') で囲む必要があります。

HOLD ファイルを SET HOLDATTRS ON で生成する場合、HOLD ファイルの TITLE 値を生成する際に、元の TITLE 値の前に SEG_TITLE_PREFIX が追加されます。

例　フィールドタイトル接頭語の定義

次の例は、WF_RETAIL_TIME_LITE テーブルを参照する WF_RETAIL__LITE クラスタマスターファイルの 3 つのセグメントを示しています。WF_RETAIL_TIME_SALES セグメントの SEG_TITLE_PREFIX は 'Sale' です。WF_RETAIL_TIME_DELIVERED セグメントの SEG_TITLE_PREFIX は'Delivery' です。WF_RETAIL_TIME_SHIPPED セグメントの SEG_TITLE_PREFIX は 'Shipped' です。

```
SEGMENT=WF_RETAIL_TIME_SALES, CRFILE=wfretail82/dimensions/wf_retail_time_lite,
CRSEGMENT=WF_RETAIL_TIME_LITE, CRINCLUDE=ALL,
    DESCRIPTION='Time Sales Dimension', SEG_TITLE_PREFIX='Sale,', $
   PARENT=WF_RETAIL_SALES, SEGTYPE=KU, CRJOINTYPE=LEFT_OUTER,
    JOIN_WHERE=WF_RETAIL_SALES.ID_TIME EQ WF_RETAIL_TIME_SALES.ID_TIME;, $
     ...
   SEGMENT=WF_RETAIL_TIME_DELIVERED, SEGTYPE=KU, PARENT=WF_RETAIL_SHIPMENTS,
   CRFILE=ibisamp/dimensions/wf_retail_time_lite, CRSEGMENT=WF_RETAIL_TIME_LITE,
CRINCLUDE=ALL, CRJOINTYPE=LEFT_OUTER,
    JOIN_WHERE=ID_TIME_DELIVERED EQ WF_RETAIL_TIME_DELIVERED.ID_TIME;,
    DESCRIPTION='Shipping Time Delivered Dimension', SEG_TITLE_PREFIX='Delivery,', $
     ...
  SEGMENT=WF_RETAIL_TIME_SHIPPED, SEGTYPE=KU, PARENT=WF_RETAIL_SHIPMENTS,
    CRFILE=ibisamp/dimensions/wf_retail_time_lite, CRSEGMENT=WF_RETAIL_TIME_LITE,
CRINCLUDE=ALL, CRJOINTYPE=LEFT_OUTER,
    JOIN_WHERE=ID_TIME_SHIPPED EQ WF_RETAIL_TIME_SHIPPED.ID_TIME;,
    DESCRIPTION='Shipping Time Shipped Dimension', SEG_TITLE_PREFIX='Shipped,', $
```

これら 3 つのセグメントには、すべて同一のフィールドが格納されています。SEG_TITLE_PREFIX はレポート出力に表示され、フィールドの格納先のセグメントが示されます。次のリクエストは、DAYSDELAYED を TIME_QTR ごとに集計します。

```
TABLE FILE wf_retail_lite
SUM DAYSDELAYED
BY TIME_QTR
ON TABLE SET PAGE NOLEAD
ON TABLE SET STYLE *
GRID=OFF,$
END
```

出力の列見出しは、TIME_QTR 値が WF_RETAIL_TIME_SALES セグメントに格納されていたことを示しています。これは、WF_RETAIL_LITE マスターファイルで、このセグメントが、このフィールド名が所属する最上位のセグメントであるためです。

Sale Quarter	Quantity Sold
1	4,188
2	4,105
3	4,007
4	1,623

WF_RETAIL_TIME_DELIVERED などの異なるセグメントを指定するには、リクエストで修飾フィールド名を指定します。

```
TABLE FILE wf_retail_lite
SUM DAYSDELAYED
BY WF_RETAIL_TIME_DELIVERED.TIME_QTR
ON TABLE SET PAGE NOLEAD
ON TABLE SET STYLE *
GRID=OFF,$
END
```

出力の列見出しは、TIME_QTR 値が WF_RETAIL_TIME_DELIVERED セグメントに格納されていたことを示しています。

Delivery Quarter	Days Delayed
1	1,436
2	1,637
3	1,517
4	765

例 SEG_TITLE_PREFIX 指定セグメントを含む HOLD ファイルの生成

WF_RETAIL_TIME_LITE マスターファイルの TIME_QTR のフィールドタイトルは 'Quarter' です。WF_RETAIL_LITE マスターファイルの WF_RETAIL_TIME_DELIVERED セグメントの SEG_TITLE_PREFIX は 'Delivery' です。次のリクエストは HOLDATTRS パラメータを ON に設定し、「SEGPREFIX」という名前の HOLD ファイルを生成します。

```
APP HOLD baseapp
TABLE FILE wf_retail_lite
SUM DAYSDELAYED
BY WF_RETAIL_TIME_DELIVERED.TIME_QTR
ON TABLE SET HOLDATTRS ON
ON TABLE HOLD AS SEGPREFIX
END
```

このリクエストを実行すると、SEGPREFIX マスターファイルが生成されます。TIME_QTR フィールドのタイトルとして 'Delivery,Quarter' が指定されます。このタイトルは、WF_RETAIL_LITE マスターファイルの SEG_TITLE_PREFIX 値を、WF_RETAIL_TIME_LITE マスターファイルのタイトル値と連結することで生成されます。

```
 FIELDNAME=TIME_QTR, ALIAS=E01, USAGE=I2, ACTUAL=I04,
      MISSING=ON,
      TITLE='Delivery,Quarter', DESCRIPTION='Quarter', $
```

4

フィールドの記述

フィールドは、データソースに格納されたデータの有意な最小単位ですが、フィールドごとにさまざまな特性を表すことができます。これらの特性を記述するには、マスターファイルの属性を使用します。

トピックス

フィールドの特性

マスターファイルでは、次のフィールド特性を記述します。

- FIELDNAME 属性で識別するフィールド名。

- フィールドの別名。これは、ネイティブデータ管理システムで定義された元の名前、特定のデータソースタイプで独自に選択した別名、特別な場合としてフィールドを解釈するために事前に定義された値のいずれかです。これらの別名をリクエストで使用することができます。この別名は ALIAS 属性で定義します。
- フィールドがデータを格納して表示する方法。これらの特性は、ACTUAL、USAGE、MISSING 属性で指定します。

  ACTUAL 属性は、データソースにデータを実際に格納する際のデータタイプおよび長さを指定します。たとえば、長さが 15 バイトの文字フィールドとしてフィールドを定義します。FOCUS データソースでは ACTUAL 属性は使用されず、代わりに USAGE 属性を使用してデータのフォーマット設定を指定します。データの格納は、WebFOCUS によって処理されます。

  USAGE 属性は、フィールドをレポートに表示する際のフィールドのフォーマットを指定します。この属性のエイリアスは FORMAT です。また、日付フォーマット、通貨記号表示 (浮動)、ゼロサプレス (ゼロを非表示) などの編集オプションを指定することもできます。

  MISSING 属性は、FOCUS データソースや多くのリレーショナルデータソースなどの Null データをサポートするデータソースで、Null 値のフィールドへの 書き込みおよびフィールドからの読み取りを有効にします。
- 一時項目を作成するオプション。一時項目は、データソースに格納されている値ではなく、データソースに存在する情報から算出された値を持つフィールドです。一時項目 (DEFINE) は DEFINE 属性で指定します。
- オプションとして使用する開発者向けのフィールドの説明。この説明は DESCRIPTION 属性で指定します。
- 受容可能なフィールドのデータ入力値。この特性は ACCEPT 属性で指定します。
- フィールドの代替レポートフィールドタイトル。この特性は TITLE 属性で指定します。
- フィールドに格納された下 2 桁の西暦年に上 2 桁の世紀の値を割り当てる 100 年。この時間枠は DEFCENT および YRTHRESH の 2 つの属性で定義します。

フィールドの名前 - FIELDNAME

FIELDNAME 属性を使用してフィールドを識別します。この属性は、マスターファイルのフィールド宣言で最初に指定する属性です。フィールドには、ネイティブデータソースでの名前に関係なく、任意の名前を割り当てることができます。同様に、FOCUS データソースでは、新しいデータソースのフィールドに任意の名前を割り当てることができます。

レポートを生成すると、デフォルト設定でレポートのフィールドタイトルにはフィールド名が表示されるため、レポートを閲覧するユーザに分かりやすい名前を付けておくと役立ちます。別の方法として、リクエストで AS 句を使用して、レポートに表示する別のフィールドタイトルを指定することもできます。詳細は、『TIBCO WebFOCUS Language リファレンス』および188 ページの 「 代替レポートフィールドタイトル - TITLE 」 を参照してください。

構文　フィールド名の識別

```
FIELD[NAME] = field_name
```

説明

`field_name`

このフィールドに適用する名前です。1 バイト文字セットの場合、最大値は 512 文字です。Unicode 環境の場合、この長さは各文字が表すバイト数による影響を受けます。詳細は、『TIBCO WebFOCUS サーバ管理者ガイド』の「Unicode サポート」を参照してください。12 バイトを超える名前には、いくつかの制限が適用されます。詳細は、100 ページの「 フィールド名の制限事項 」を参照してください。名前には 1 つ以上の文字を含める必要があります。それ以外は、文字、数字、アンダースコア (_) を任意に組み合わせて使用することができます。異なるオペレーティングシステム環境や、数式を計算する際に問題を起こす可能性があるため、これ以外の文字は使用しないことをお勧めします。

また、タイプを表す Cn、En、Xn は、レポートフィールド、HOLD ファイルフィールド、その他の特別なオブジェクトを参照する際に使用されるため、フィールド名には使用しないことをお勧めします (ここで、n は任意の 1 桁または 2 桁の連続番号)。

フィールドのレポートフィールドタイトルに特殊文字を使用する必要がある場合は、マスターファイルの TITLE 属性でタイトルを指定する方法もあります。詳細は、188 ページの「 代替レポートフィールドタイトル - TITLE 」 を参照してください。

参照　FIELDNAME 属性使用時の注意

FIELDNAME 属性の使用時には次の規則が適用されます。

- **エイリアス**　FIELDNAME 属性のエイリアスは FIELD です。
- **変更**　FOCUS データソースでは、フィールドにインデックスが作成されている場合 (INDEX=I 属性)、データソースを再構築しない限り、フィールド名を変更することはできません。これ以外の場合は名前を変更することができます。

参照　フィールド名の制限事項

12 バイトを超えるフィールド名およびエイリアスには、次の制限が適用されます。

- JOIN コマンドで使用するクロスリファレンスファイルが FOCUS データソースの場合、12 バイトを超えるフィールド名を使用してクロスリファレンスフィールドを指定することはできません。
- FOCUS データソースのインデックスフィールドおよびテキストフィールドには 12 バイトを超える長さのフィールド名を含めることはできません。XFOCUS マスターファイルのテキストフィールドおよびインデックスフィールドには 12 バイトの制限は適用されません。長い名前のエイリアスは両タイプのデータソースでサポートされます。
- 代替ファイルビューで指定されたフィールド名を修飾することはできません。
- CHECK FILE コマンドの PICTURE および HOLD オプションは、表示された図または HOLD ファイルに長い名前の先頭の 11 バイトを表示します。12 バイト目の位置にキャレット (>) が表示された場合、これは実際の名前が表示された部分より長いことを意味します。
- ?FF、? HOLD、? DEFINE

  これらのコマンドは、名前の 31 バイト分に加えて、長いフィールド名を示すキャレット (>) を 32 バイト目に表示します。

修飾フィールド名の使用

リクエストでは、参照されているフィールド名およびエイリアスのすべてを、ファイル名またはセグメント名で修飾することができます。この方法は、重複したフィールド名がマスターファイルのセグメントや結合されたデータソースに存在する場合に役立ちます。

FOCUS マスターファイルでは、テキストフィールドおよびインデックスフィールドの名前は 12 バイトに制限されます。XFOCUS マスターファイルでは、テキストフィールドおよびインデックスフィールド名には 12 バイトの制限は適用されません。ただし、テキストフィールドおよびインデックスフィールドのエイリアスの最大長は 512 バイトです。TITLE 属性または AS 句が使用されていない場合は、最大で 512 バイトのフィールド名が TABLE レポートのフィールドタイトルに表示されます。

SET FIELDNAME コマンドのデフォルト値である SET FIELDNAME=NEW により、長いフィールド名および修飾フィールド名は有効になっています。構文についての詳細は、『TIBCO WebFOCUS アプリケーション作成ガイド』を参照してください。

構文　リクエストでの修飾フィールド名の指定

```
[filename.][segname.]fieldname
```

説明

`filename`
マスターファイルまたはタグの名前です。タグ名は JOIN および COMBINE コマンドで使用します。

`segname`
フィールドが存在するセグメントの名前です。

`fieldname`
フィールド名です。

例　フィールド修飾名の指定

EMPLOYEE データソースの EMPINFO セグメントに存在する EMP_ID フィールドの完全修飾名は次のとおりです。

```
EMPLOYEE.EMPINFO.EMP_ID
```

構文　修飾文字の変更

```
SET QUALCHAR = qualcharacter
```

デフォルトの修飾文字はピリオド (.) です。SET QUALCHAR コマンドおよび有効な修飾文字 (. : ! % | ¥) についての詳細は、『TIBCO WebFOCUS アプリケーション作成ガイド』を参照してください。

重複フィールド名の使用

1 つのフィールド名またはエイリアスで複数のフィールドを参照できる場合、重複したフィールド名と見なされます。次のような場合にフィールドの重複が発生します。

- 1 つの名前がマスターファイル内で複数回出現する場合。
- 複数のマスターファイルで定義された JOIN、または再帰的 JOIN。
- 接頭語を指定せずに COMBINE を発行した場合。

同一セグメント内で重複フィールドを使用することはできません (同一のフィールド名およびエイリアスを持つフィールド)。2 回目に出現したフィールドにはアクセスできないため、CHECK および CREATE FILE を発行すると次のメッセージが生成されます。

```
(FOC1829) 警告。セグメントに重複フィールド名があります：fieldname
```

マスターファイルでは、異なるセグメントに重複フィールド名を含めることができます。リクエストでそのフィールドを取得するには、フィールド名をセグメント名で修飾する必要があります。マスターファイルに複数回出現するフィールドをリクエストで修飾しない場合は、マスターファイルで最初に見つかったフィールドが取得されます。

注意：マスターファイルに実フィールドまたは一時項目 (DEFINE) の重複フィールド名が含まれる場合、フィールドの取得時には次のロジックが適用されます。

- 一時項目 (DEFINE) のみが重複している場合は、最後の一時項目 (DEFINE) が取得されます。
- 実フィールドのみが重複している場合は、最初の実フィールドが取得されます。
- マスターファイルに同一名の実フィールドと 1 つまたは複数の一時項目 (DEFINE) が存在する場合は、最後の一時項目 (DEFINE) が取得されます。
- マスターファイルの外部で定義したフィールドの名前がマスターファイルの一時項目または実フィールドの名前と同一の場合は、マスターファイルの外部で定義した最後の一時項目が取得されます。

レポートには修飾名をフィールドタイトルとして含めることができます。レポートに重複フィールド名の修飾フィールドタイトルを表示するかどうかを指定するには、SET QUALTITLES コマンドを使用します。このコマンドについての詳細は、『TIBCO WebFOCUS アプリケーション作成ガイド』を参照してください。SET QUALTITLES=ON に設定すると、リクエストで修飾名を指定しなくても、重複フィールド名の修飾フィールドタイトルが表示されます。デフォルト値は OFF で、修飾フィールドタイトルの表示は無効になっています。

修飾フィールド名評価時の規則

修飾フィールド名の評価には次の規則が適用されます。

- 最大の修飾フィールド名は filename.segname.fieldname です。以下はその例です。

```
TABLE FILE EMPLOYEE
PRINT EMPLOYEE.EMPINFO.EMP_ID
END
```

ここでは、完全修飾フィールドとして EMP_ID が指定されています。ファイル名の EMPLOYEE およびセグメント名の EMPINFO はフィールド修飾子です。

重複した修飾子名を使用することもできます。以下はその例です。

```
FILENAME=CAR, SUFFIX=FOC
  SEGNAME=ORIGIN, SEGTYPE=S1
      FIELDNAME=COUNTRY, COUNTRY, A10, $
  SEGNAME=COMP, SEGTYPE=S1, PARENT=ORIGIN
      FIELDNAME=CAR, CARS, A16, $
        .
        .
        .
TABLE FILE CAR
PRINT CAR.COMP.CAR
END
```

このリクエストは、「CAR」というエイリアス付きでフィールドを出力します。この場合、ファイル名とフィールド名の両方が CAR です。

フィールド名は、そのファイル名、セグメント名のいずれか一方の修飾子で修飾することができます。以下はその例です。

```
FILENAME=CAR, SUFFIX=FOC
   SEGNAME=ORIGIN, SEGTYPE=S1
      FIELDNAME=COUNTRY, COUNTRY, A10, $
   SEGNAME=COMP, SEGTYPE=S1, PARENT=ORIGIN
      FIELDNAME=CAR, CARS, A16, $
        .
        .
        .
TABLE FILE CAR
PRINT COMP.CAR AND CAR.CAR
END
```

このリクエストは、「CAR」というエイリアス付きでフィールドを 2 回出力します。

修飾子が 1 つの場合は、セグメント名がファイル名より優先されます。そのため、ファイル名とセグメント名が同一の場合は、セグメント名で修飾された方のフィールドが取得されます。

- フィールド名の先頭文字が演算接頭語と同一の場合、リクエストの参照先がそのフィールド名全体を指しているのか、接頭語演算子が適用されたフィールド名を指しているのかを判断できない場合があります。この場合は、演算接頭語をフィールドに適用して計算した値ではなく、全体をフィールド名とした値が取得されます。次の例では、修飾されていない「CNT.COUNTRY」というフィールド名および「COUNTRY」というフィールド名が指定されています。

```
FILENAME=CAR, SUFFIX=FOC
   SEGNAME=ORIGIN, SEGTYPE=S1
      FIELDNAME=CNT.COUNTRY, ACNTRY, A10, $
      FIELDNAME=COUNTRY, BCNTRY, A10, $
TABLE FILE CAR
SUM CNT.COUNTRY
END
```

  このリクエストの「CNT.COUNTRY」という文字列は、演算接頭語の CNT. が適用された「COUNTRY」というフィールドを指しているのではなく、「CNT.COUNTRY」というフィールドを指しているものと解釈されます。そのため、このリクエストではエイリアスが ACNTRY のフィールドの方が集計されます。フィールド名の CNT.COUNTRY にはピリオド (.) が含まれていますが、これは修飾されていないフィールド名です。これは修飾名でもフィールド名に付ける演算接頭語でもありません。両者ともマスターファイルで使用することはできません。このリクエストでは、エイリアスが BCNTRY のフィールドのインスタンス数は集計されません。

- マスターファイルで定義したファイル名、セグメント名のいずれかが演算接頭語と同一の場合でも、リクエストはフィールドに演算接頭語を適用して値を計算するのではなく、セグメント内のフィールド自体の値を取得します。

  以下はその例です。

```
FILENAME=CAR, SUFFIX=FOC
   SEGNAME=ORIGIN, SEGTYPE=S1
      FIELDNAME=COUNTRY, COUNTRY, A10, $
   SEGNAME=PCT, SEGTYPE=S1, PARENT=ORIGIN
      FIELDNAME=CAR, CARS, I2, $
 TABLE FILE CAR
SUM PCT.CAR PCT.PCT.CAR
BY COUNTRY
END
```

  このリクエストは、最初にエイリアスが CAR のフィールドを集計し、次に COUNTRY に対する CAR のパーセント値を計算します。

- 修飾フィールド名が 2 つの修飾レベルのどちらかでも成り立つ場合は、上位にある修飾フィールド名が優先されます。

  次の例では、修飾されていないフィールド名 (ORIGIN セグメントの ORIGIN.COUNTRY フィールド) とセグメント名で修飾されているフィールド名 (ORIGIN セグメントの COUNTRY フィールド) のどちらも成り立ちます。ここでは、セグメント名が修飾されているフィールドが取得されます。

```
FILENAME=CAR, SUFFIX=FOC
   SEGNAME=ORIGIN, SEGTYPE=S1
      FIELDNAME=ORIGIN.COUNTRY, OCNTRY, A10, $
      FIELDNAME=COUNTRY, CNTRY, A10, $
TABLE FILE CAR
PRINT ORIGIN.COUNTRY
END
```

  このリクエストは、エイリアスが CNTRY のフィールドを表示します。エリアスが OCNTRY のフィールドを取得するには、次のようにフィールド名の ORIGIN.COUNTRY をそのセグメント名の ORIGIN で修飾する必要があります。

```
PRINT ORIGIN.ORIGIN.COUNTRY
```

- 修飾フィールド名が修飾レベルの等しい 2 つのフィールド名のどちらでも成り立つ場合は、基本フィールド名の長さが短い方のフィールド名が優先されます。以下はその例です。

```
FILENAME=CAR, SUFFIX=FOC
   SEGNAME=CAR, SEGTYPE=S1
      FIELDNAME=CAR.CAR, CAR1, A10, $
   SEGNAME=CAR.CAR, SEGTYPE=S1, PARENT=CAR
      FIELDNAME=CAR, CAR2, A10, $
TABLE FILE CAR
PRINT CAR.CAR.CAR
END
```

  この例では、「CAR.CAR.CAR」という文字列が、CAR セグメントの CAR.CAR というフィールドを指しているのか、CAR.CAR セグメントの CAR というフィールドを指しているのかを判断することができません。この場合、両方とも「CAR.CAR」という名前はピリオド (.) を含む修飾されていない名前で、修飾名ではありません。修飾名をマスターファイルで使用することはできません。

  いずれの場合も修飾フィールド名は同一で、修飾レベルでの明確な差はありません。

  この場合は、エイリアスが CAR2 のフィールドの基本フィールド名の方が短いため、CAR2 が表示されます。この例は、2 つの修飾レベルで比較した前回の例とは異なります。CAR1 フィールドを取得するには、そのエイリアスを指定する必要があります。

フィールドエイリアス - ALIAS

すべてのフィールドに別名 (エイリアス) を割り当てることができます。フィールドのエイリアスには、そのネイティブデータソースで定義された元の名前、ユーザが独自に選択した任意の名前、特別な場合として定義済みの値のいずれかを割り当てることができます。エイリアスの割り当て方法は、データソースのタイプおよび特別な場合としてデータソースでのフィールドの役割により異なります。エイリアスを割り当てると、通常のフィールド名の別名としてエイリアスをリクエストに使用することができます。エイリアスを割り当てるには、ALIAS 属性を使用します。

例　フィールドエイリアスの使用

EMPLOYEE データソースでは、FIELDNAME 属性を使用してフィールドに「CURR_SAL」という名前を割り当て、ALIAS 属性を使用してこのフィールドに「CSAL」というエイリアスを割り当てています。

```
FIELDNAME = CURR_SAL, ALIAS = CSAL, USAGE = D12.2M,  $
```

リクエストでは、有効な名前として両者を同等に使用することができます。次の 2 つの TABLE リクエストは、両者が同一のフィールドを参照し、機能的に同等で、同一の結果を生成することを示しています。

```
TABLE FILE EMPLOYEE
PRINT CURR_SAL BY EMP_ID
END

TABLE FILE EMPLOYEE
PRINT CSAL BY EMP_ID
END
```

注意：抽出ファイル (HOLD および PCHOLD) のフィールドの識別には、エイリアスではなく、フィールド名が使用されます。

フィールドエイリアスの実装

ALIAS 属性に値を割り当てる場合、特記されていない限り、FIELDNAME 属性と同一の規則に従って名前を付ける必要があります。次の方法で各データソースタイプの ALIAS 属性に値を割り当てします。

- **リレーショナルデータソース** ALIAS 属性には、リレーショナルテーブルで定義された元のフィールド名と同一の名前を指定します。
- **シーケンシャルデータソース** ALIAS 属性には、リクエストでフィールドの識別に使用するエイリアス (別名) を指定します。エイリアスとして任意の名前を割り当てることができます。多くのユーザは、フィールド実名の短縮名を選択します。たとえば、フィールド名が LAST_NAME の場合、エイリアスは LN にします。マスターファイルでは ALIAS 属性を指定する必要がありますが、その値をブランクにすることができます。

  なお、連続した繰り返しフィールドには ALIAS の値を ORDER に指定してこの属性を別の方法で使用します。また、データソースに複数のレコードタイプが含まれている場合に RECTYPE および MAPVALUE フィールドにも使用します。
- **FOCUS データソース** ALIAS 属性には、リクエストでフィールドの識別に使用するエイリアス (別名) を指定します。 エイリアスとして任意の名前を割り当てることができます。多くのユーザは、フィールド実名の短縮名を選択します。たとえば、フィールド名が LAST_NAME の場合、エイリアスは LN にします。マスターファイルでは ALIAS 属性を指定する必要がありますが、その値をブランクにすることができます。 詳細は、231 ページの「 FOCUS データソースの記述 」を参照してください。エイリアスは、データソースを再構築せずに変更することができます。エイリアスが他のデータソースで参照されている場合、そのマスターファイルでも同様の変更が必要な場合があります。

表示データタイプ - USAGE

USAGE 属性は、レポートに表示するフィールドまたは計算に使用するフィールドのフォーマットを記述します。この属性は、「FORMAT 属性」とも呼ばれます。

表示フォーマットの指定

FOCUS データソースでは ACTUAL 属性を使用しないため、USAGE 属性を使用してフィールドの格納方法も指定します。他のタイプのデータソースでは、ACTUAL 値に対応する USAGE 値を割り当てて、フィールドをデータソースに格納する場合のデータタイプと同一のデータタイプでフィールドを識別します。データが文字として格納されている場合は、レポート内でのフィールドの表示方法に基づいて USAGE を割り当てます。変換は自動的に実行されます。USAGE 値に対応する ACTUAL 値についての詳細は、使用するデータアダプタのマニュアルを参照してください。その他のデータソースタイプについては、アダプタのマニュアルを参照してください。

フィールドのデータタイプおよび長さを選択する以外に、日付フォーマット、通貨記号表示 (浮動)、ゼロサプレス (ゼロを非表示) などの表示オプションを指定することもできます。これらのオプションを使用して、レポートにフィールドを表示する方法をカスタマイズします。

構文　表示フォーマットの指定

```
USAGE = tl[d]
```

説明

`t`
データタイプです。有効値には、A (文字)、F (単精度浮動小数点数)、D (倍精度浮動小数点数)、X (拡張 10 進数浮動小数点数)、I (整数)、P (パック 10 進数)、TX (テキスト) があります。また、D、W、M、Q、Y (日付) の有効な組み合わせを使用することもできます。

`l`
長さの指定です。指定はデータタイプにより異なります。詳細は、各データタイプのセクションを参照してください。なお、日付フォーマットのフィールドには長さは指定しません。

`d`
1 つまたは複数の表示オプションです。データタイプごとに表示オプションが異なります。詳細は、各データタイプのセクションを参照してください。

USAGE 値の最大長は 8 バイトです。

フィールドのタイプおよび長さを指定する値は、フィールドを表示または格納する際に割り当てられる出力位置の値に反映されます。表示オプションは、フィールドを表示または印刷する場合にのみ影響します。抽出ファイルなどの表示に関係しないフィールドの取得時には表示オプションは無効です。

注意：指定した USAGE フォーマットで数値フィールドを表示できない場合は、アスタリスク (*) が表示されます (例、集計結果の桁数が大きすぎる場合)。

詳細は、各フォーマットタイプのセクションを参照してください。

参照　USAGE 属性使用時の注意

USAGE 属性の使用時には次の規則が適用されます。

- **エイリアス**　USAGE 属性のエイリアスは FORMAT です。
- **変更**　ほとんどのデータソースでは、USAGE 属性で指定するタイプおよび長さは、そのフィールドの ACTUAL 属性で有効なタイプおよび長さの範囲内でのみ変更することができます。表示オプションはいつでも変更することができます。

  FOCUS データソースでは、タイプの指定を変更することはできません。フィールドのデータタイプが I、F、D、P の場合、長さの指定は格納時には影響せず、表示のみに反映されるため、長さの指定を変更することができます。P フィールドでは長さ指定の小数部を変更することはできません。A (文字) フィールドの長さ指定は、REBUILD 機能を使用する場合にのみ変更することができます。表示オプションはいつでも変更することができます。

データタイプのフォーマット

次のフォーマットタイプを指定することができます。

- **数値**　数値フォーマットには、整数、単精度浮動小数点数、倍精度浮動小数点数、拡張 10 進浮動小数点数 (XMATH)、10 進浮動小数点数 (MATH)、パック 10 進数の 6 種類があります。詳細は、116 ページの 「 数値の表示オプション 」 を参照してください。
- **文字**　文字フォーマットは、数字、文字、およびその他の文字で構成され、一連の文字として解釈されるすべての値に使用することができます。
- **16 進数**　この USAGE フォーマットは、実フォーマットの文字フォーマットとともに使用し、文字フィールド値を 16 進数表現として表示および保存する場合に使用します。
- **文字列**　リレーショナルデータソースでデータタイプが STRING の文字データには、文字列フォーマットを使用することができます。
- **日付**　日付フォーマットを使用して、年、四半期、月、日、曜日などの日付構成要素を定義し、次のことを行えます。
  - 日付別のソート
  - 日付による比較および計算
  - トランザクション時の日付の自動確認

たとえば、DECODE 関数を使用して日付値を割り当てるようなアプリケーションでは、部分的な日付機能を備えた日付表示オプションを使用して文字、整数、パック 10 進数フィールドを使用することもできます。

- **日付時間** 多くのリレーショナルデータソースで使用するタイムスタンプデータタイプと同様に、日付時間フォーマットは日付と時間の両方をサポートします。日付時間フィールドは、8、10、12 バイトのいずれかで格納されます。日付には 4 バイトが使用され、時間にはミリ秒またはナノ秒をフォーマットに指定するかどうかにより 4、6、8 バイトのいずれかが使用されます。計算は、データタイプ内で直接割り当てた場合にのみ実行可能です。その他のすべての演算は一連の日付時間関数で実行されます。
- **テキスト** テキストフィールドを使用して、大量のデータを格納したり、そのデータに改行を追加して表示したりすることができます。

整数フォーマット

整数には整数フォーマットを使用することができます。整数は、小数点を含まない、0 から 9 までの数字で構成された任意の値です。

整数フィールドと日付表示オプションを使用して、限定的な日付サポートを活用することもできます。整数フィールドの使用についての詳細は、149 ページの 「 日付表示オプションを使用した文字および数値フォーマット 」 を参照してください。

整数の USAGE タイプは I です。詳細は、116 ページの 「 数値の表示オプション 」 を参照してください。長さ指定のフォーマットは次のとおりです。

n

説明

n

表示する桁数です。最大長は、32 ビット版の WebFOCUS では 11、64 ビット版では 22 です。この長さには、桁数とマイナス符号 (フィールドに負の値が含まれている場合) を含める必要があります。また、小数点以下の桁数 (最大で n-1 桁) を指定することもできます。数値は、その桁数の前に小数点が付いて表示されます。

以下はその例です。

フォーマット	表示
I6	4316

フォーマット	表示
I6.2	43.16
I2	22
I4	-617

倍精度浮動小数点数フォーマット

倍精度浮動小数点数フォーマットは、0 から 9 までの数字およびオプションの小数点で構成された任意の値に対して使用することができます。

倍精度浮動小数点数の USAGE タイプは D です。表示オプションについての詳細は、116 ページの 「 数値の表示オプション 」 を参照してください。長さ指定のフォーマットは次のとおりです。

```
t[.s]
```

説明

`t`

表示する桁数で、最大長は 33 桁です。この長さには、数値の桁数、オプションの小数点、負の値を持つフィールド先頭のマイナス符号 (-) が含まれます。サポートされる有効数字の桁数は、オペレーティング環境により異なります。

`s`

小数点の後に続く桁数です。最大で 31 桁ですが、t より小さい桁数を指定する必要があります。

以下はその例です。

フォーマット	表示
D8.2	3,187.54
D8	416

たとえば D8.2 の場合、数字の 8 は小数点および小数点以下の桁数を含めた全体の最大桁数を表します。数字の 2 は、全体の 8 桁のうち、小数点以下の桁数を表します。カンマ (,) は自動的に表示され、桁数には含まれません。

単精度浮動小数点数フォーマット

単精度浮動小数点数フォーマットは、小数部を含む任意の数値に対して使用することができます。この数値は、0 から 9 までの数字、およびオプションの小数点で構成されます。倍精度浮動小数点数フォーマットとは異なり、このフォーマットは長さが 9 桁を超えることができないため、主に桁数の少ない小数に使用します。

単精度浮動小数点数の USAGE タイプは F です。表示オプションについての詳細は、116 ページの 「 数値の表示オプション 」 を参照してください。長さ指定のフォーマットは次のとおりです。

```
t[.s]
```

説明

`t`
: 表示する桁数で、最大長は 33 桁です。この長さには、数値の桁数、オプションの小数点、負の値を持つフィールド先頭のマイナス符号 (-) が含まれます。サポートされる有効数字の桁数は、オペレーティング環境により異なります。

`s`
: 小数点の後に続く桁数です。最大で 31 桁ですが、t より小さい桁数を指定する必要があります。

以下はその例です。

フォーマット	表示
`F5.1`	`614.2`
`F4`	`318`

拡張 10 進数浮動小数点数 (XMATH) フォーマット

拡張 10 進数浮動小数点数フォーマットは、小数部を含む任意の数値に対して使用することができます。この数値は、0 から 9 までの数字、およびオプションの小数点で構成されます。この長さは、有効桁数 37 桁を超えることはできません。拡張 10 進数浮動小数点数値は、2 進数として格納される倍精度および単精度の浮動小数点数とは異なり、ベース 10 (10 進数) の数値として格納されます。デフォルト設定では、すべての数値処理は、倍精度浮動小数点数を使用して実行されます。このデフォルト設定は、SET FLOATMAPPING コマンドを使用して変更することができます。

拡張 10 進数浮動小数点数の USAGE タイプは X です。互換性のある表示オプションについての詳細は、116 ページの 「 数値の表示オプション 」 を参照してください。長さ指定のフォーマットは次のとおりです。

```
t[.s]
```

説明

`t`
表示する桁数で、最大長は 44 桁です。この長さには、数値の桁数、オプションの小数点、負の値を持つフィールド先頭のマイナス符号 (-) が含まれます。サポートされる有効数字の桁数は、オペレーティング環境により異なります。

`s`
小数点の後に続く桁数です。最大で 37 桁ですが、t より小さい桁数を指定する必要があります。

以下はその例です。

フォーマット	表示
`X5.1`	`614.2`
`X4`	`318`

10 進数浮動小数点数 (MATH) フォーマット

10 進数浮動小数点数フォーマットは、小数部を含む任意の数値に対して使用することができます。この数値は、0 から 9 までの数字、およびオプションの小数点で構成されます。このフォーマットは、桁数の少ない小数に使用します。拡張 10 進数浮動小数点数とは異なり、長さが有効桁数 15 桁を超えることはできません。10 進数浮動小数点数値は、2 進数として格納される倍精度および単精度の浮動小数点数とは異なり、ベース 10 (10 進数) の数値として格納されます。デフォルト設定では、すべての数値処理は、倍精度浮動小数点数を使用して実行されます。このデフォルト設定は、SET FLOATMAPPING コマンドを使用して変更することができます。

10 進数浮動小数点数の USAGE タイプは M です。互換性のある表示オプションについての詳細は、116 ページの 「 数値の表示オプション 」 を参照してください。長さ指定のフォーマットは次のとおりです。

```
t[.s]
```

説明

`t`
: 表示する桁数で、最大長は 34 桁です。この長さには、数値の桁数、オプションの小数点、負の値を持つフィールド先頭のマイナス符号 (-) が含まれます。サポートされる有効数字の桁数は、オペレーティング環境により異なります。

`s`
: 小数点の後に続く桁数です。最大で 31 桁ですが、t より小さい桁数を指定する必要があります。

以下はその例です。

フォーマット	表示
`M5.1`	`614.2`
`M4`	`318`

パック 10 進数フォーマット

パック 10 進数フォーマットは、小数部を含む任意の数値に対して使用することができます。小数部は、0 から 9 までの数字、およびオプションの小数点で構成された任意の値です。

パック 10 進数フィールドを日付表示オプションとともに使用して、限定的な日付サポートを活用することもできます。詳細は、149 ページの 「 日付表示オプションを使用した文字および数値フォーマット 」 を参照してください。

パック 10 進数の USAGE タイプは P です。表示オプションについての詳細は、116 ページの「 数値の表示オプション 」 を参照してください。

長さ指定のフォーマットは次のとおりです。

```
t[.s]
```

説明

`t`
表示する桁数で、最大長は 33 桁です。この長さには、最大 31 桁の数字、オプションの小数点、負の値を持つフィールド先頭のマイナス符号 (-) が含まれます。

`s`
小数点の後に続く桁数です。最大で 31 桁ですが、t より小さい桁数を指定する必要があります。

以下はその例です。

フォーマット	表示
`P9.3`	`4168.368`
`P7`	`617542`

P フィールドには、8 バイト (15 桁までをサポート) および 16 バイト (33 桁までをサポート) の 2 つの内部長が含まれています。USAGE の P1 から P15 までは、8 バイトで構成される内部ストレージとして、自動的に割り当てられます。USAGE の P16 以上は、16 バイトで構成される内部ストレージとして割り当てられます。

USAGE が格納済みの数値の表示に必要な桁数を満たさない場合は、数値の代わりにアスタリスク (*) が表示されます。これは、必ずしもフィールドのオーバーフローを示しているわけではなく、フィールドの表示に必要な桁数を指定しなかったことのみを意味します。

オーバーフローは、割り当て済みの内部ストレージに適用可能な桁数よりも多くの桁数を含む数値を格納しようとした場合に発生します。この種のオーバーフローは、すべてのオペレーティング環境で、すべてが 9 で構成される数値を格納することで示されます。したがって、内部ストレージとして 8 バイトが割り当てられたパック 10 進数フィールドに 16 桁で構成される数値を格納しようとすると、このフィールドには、代わりに数値 999999999999999 (9 桁を 15 回繰り返す) が格納されます。

この種のフィールドに P1 から P14 までの USAGE フォーマットを割り当てた場合、このフィールドに格納された 15 桁は表示できないため、アスタリスク (*) が表示されます。ただし、このフィールドに USAGE フォーマットの P15 を割り当てた場合、フィールドに格納された 15 桁の数値が表示可能になるため、値 999999999999999 が表示されます。P15 フィールドにこの数値が表示される場合は、実際の数値の可能性も、適合しなかった代替数値の可能性もあります。

数値の表示オプション

表示オプションを使用して、数値フォーマットを編集することができます。これらのオプションは、フィールドのデータを印刷または画面に表示する場合にのみ反映され、データソースに保存する場合には影響しません。

編集オプション	説明	効果
-	マイナス符号	負の数値データの右側にマイナス符号 (-) を表示します。 **注意：**フォーマットオプションの B、E、R、T、DMY、MDY、YMD ではサポートされません。
%	パーセント記号	数値データの横にパーセント記号 (%) を表示します。パーセント計算は実行されません。
p	パーセント	数値を 100 倍し、パーセント記号 (%) を追加することで、数値をパーセント値に変換します。I フォーマットおよび P フォーマットではサポートされません。

編集オプション	説明	効果
A	マイナス符号の非表示	数値の絶対値を表示しますが、格納されている値には影響しません。 ❑ マイナス符号を非表示にする USAGE 属性を含むフィールドを HOLD ファイルに出力すると、HOLD ファイルには符号付きの値が格納されます。また、マイナス符号を非表示にする USAGE 属性も HOLD ファイルに出力されるため、この HOLD ファイルに対してリクエストを実行すると、レポート出力でマイナス符号 (-) が非表示になります。 ❑ マイナス符号の非表示オプションは、次の表示オプションと併用することはできません。 ❑ B (ブラケットネガティブ) ❑ R (クレジットネガティブ) ❑ - (右側のマイナス符号)
a	自動短縮	数値の規模に基づいて、表示に使用する適切な短縮形 (K、M、B、T) を計算します。このオプションでは、現在行の値に応じた短縮形が使用されるため、それぞれの行で使用される短縮形が異なる場合があります。たとえば、1234567890 は 1.23B と表示され、1234567890000 は 1.23T と表示されます。
b	10 億の短縮形	数値を 10 億単位で表示します。たとえば、1234567890 は 1.23B と表示されます。
B	ブラケットネガティブ	負の値を括弧で囲みます。

編集オプション	説明	効果
c	カンマサプレス	カンマ (,) を非表示にします。 数値フォーマットオプションの M (ドル記号、浮動) および N (ドル記号、固定)、データフォーマットの D (倍精度浮動小数点数) と組み合わせて使用します。
C	カンマ挿入	有効数字 3 桁ごとにカンマ (,) を挿入します。また、コンチネンタル 10 進数表記を適用している場合はピリオド (.) を挿入します。
DMY	日-月-年	日/月/年の形式で文字または整数データを日付として表示します。
E	指数表記	有効数字のみを表示します。
k	1000 の短縮形	数値を 1000 単位で表示します。たとえば、12345 は 12.35K と表示されます。
L	リーディングゼロ	先頭に 0 (ゼロ) を追加します。
m	100 万の短縮形	数値を 100 万単位で表示します。たとえば、1234567 は 1.23M と表示されます。

編集オプション	説明	効果
M	通貨記号表示 (浮動) (US コードページの場合は $)	有効数字の最上位桁の左側に浮動通貨記号を表示します。デフォルトの通貨記号は、コードページにより異なります。SET CURRSYMB=*symbol* コマンドを使用して、通貨記号として最大 4 文字または次の通貨コードのいずれかをすることができます。 USD または '$' - 米ドルを指定します。 GBP - イギリスポンドを指定します。 JPY - 日本円または中国元を指定します。 EUR - ユーロを指定します。
MDY	月-日-年	月/日/年の形式で文字または整数データを日付として表示します。
N	固定 $ (US コードページの場合)	フィールドの左側に通貨記号を表示します。通貨記号は、各ページの最初の詳細行にのみ表示されます。デフォルトの通貨記号は、コードページにより異なります。SET CURRSYMB=*symbol* コマンドを使用して、通貨記号として最大 4 文字または次の通貨コードのいずれかをすることができます。 USD または '$' - 米ドルを指定します。 GBP - イギリスポンドを指定します。 JPY - 日本円または中国元を指定します。 EUR - ユーロを指定します。
R	クレジット (CR) ネガティブ	負の値の後に「CR」を表示します。

編集オプション	説明	効果
S	ゼロサプレス	データ値が 0 (ゼロ) の場合、その位置にブランクを表示します。
t	兆の短縮形	数値を兆単位で表示します。たとえば、1234567890000 は 1.23T と表示されます。
T	月の変換	月を 3 文字の省略形で表示します。
YMD	年-月-日	年/月/日の形式で文字または整数データを日付として表示します。

注意：短縮形オプションの k、m、b、t、a では、次のことに注意してください。

- 短縮された値は、値の格納先フィールドのフォーマットで指定された小数点以下の桁数で表示されます。
- これらのフォーマットオプションは、値の表示形式を変更するのみで、格納されている値には影響しません。
- 値は、短縮形で表示される前に端数処理されます。そのため、EXTENDNUM ON 設定を使用してパック 10 進数または整数フォーマットの値を短縮表示した際に、オーバーフロー値が正しい結果と間違えられる可能性があります。
- これらの表示オプションは、すべての数値フォーマットでサポートされ、追加の表示オプション (例、-、B、R、C、c、M、N) とともに使用することができます。

例　数値表示オプションの使用

下表は、数値フィールドで使用可能な表示オプションの例を示しています。

オプション	フォーマット	データ	表示
マイナス符号	I2- D7- F7.2-	-21 -6148 -8878	21- 6148- 8878.00-
パーセント記号	I2% D7% F3.2%	21 6148 48	21% 6,148% 48.00%

オプション	フォーマット	データ	表示
カンマサプレス	D6c D7Mc D7Nc	41376 6148 6148	41376 $6148 $ 6148
カンマ挿入	I6C	41376	41,376
ゼロサプレス	D6S	0	
ブラケットネガティブ	I6B	-64187	(64187)
クレジット (CR) ネガティブ	I8R	-3167	3167 CR
リーディングゼロ	F4L	31	0031
通貨記号表示 (浮動)	D7M	6148	$6,148
通貨記号表示 (固定)	D7N	5432	$ 5,432
指数表記	D12.5E	1234.5	0.12345D+04
年/月/日	I6YMD I8YYMD	980421 19980421	98/04/21 1998/04/21
月/日/年	I6MDY I8MDYY	042198 04211998	04/21/98 04/21/1998
日/月/年	I6DMY I8DMYY	210498 21041998	21/04/98 21/04/1998
月の変換	I2MT	07	JUL

次のように、複数の表示オプションを組み合わせることができます。

フォーマット	データ	表示
I5CB	-61874	(61,874)

すべてのオプションは、任意の順序で指定することができます。数値フォーマットオプションの M (ドル記号、浮動) および N (ドル記号、固定)、データフォーマットの D (倍精度浮動小数点数) を指定した場合は、自動的にオプション C (カンマ) が有効になります。オプションの L と S を同時に使用することはできません。文字または整数の USAGE 指定に表示オプションの M (月) が含まれている場合は、その指定の任意の場所にオプションの T (変換) を挿入することができます。浮動小数点数フィールドで使用できない日付表示オプション (D、M、T、Y) についての詳細は、149 ページの 「 日付表示オプションを使用した文字および数値フォーマット 」 を参照してください。

国際単位系 (SI) 数値フォーマットの短縮形オプション

国際単位系では、桁数の非常に大きい数値 (小数を含む) の短縮形が使用されます。

WebFOCUS では、SI 準拠の次の数値短縮形がサポートされます。SI 準拠フォーマットは 2 バイトの表示コードで、小文字の「*n*」と SI 数値フォーマットの短縮形で構成されます。

接頭語	WebFOCUS フォーマットコード	サイズ	例	英語名 (米国/英国)
yotta	nY	10**24	1000000000000000000000000	septillion/quadrillion
zetta	nZ	10**21	1000000000000000000000	sextillion/trilliard
exa	nE	10**18	1000000000000000000	quintillion/trillion
peta	nP	10**15	1000000000000000	quadrillion/billiard
tera	nT	10**12	1000000000000	trillion/billion
giga	nG	10**9	1000000000	billion/milliard
mega	nM	10**6	1000000	million
kilo	nK	10**3	1000	thousand
milli	nm	10**(-3)	0.001	thousandth
micro	nu	10**(-6)	0.000001	millionth
nano	nn	10**(-9)	0.000000001	billionth/milliardth
pico	np	10**(-12)	0.000000000001	trillionth/billionth
femto	nf	10**(-15)	0.000000000000001	quadrillionth/billiardth

接頭語	WebFOCUS フォーマットコード	サイズ	例	英語名 (米国/英国)
atto	na	10**(-18)	0.000000000000000001	quintillionth/trillionth
zepto	nz	10**(-21)	0.000000000000000001	sextillionth/trilliardth
yocto	ny	10**(-24)	0.000000000000000000000001	septillionth/quadrillionth

次のリクエストでは、mega および giga のフォーマットオプションが使用されています。小数部桁数は、フォーマットで制御されます。この場合は、SUM コマンドでフォーマットが再設定されています。

```
DEFINE FILE GGSALES
NEWDOLL/D12.2 = DOLLARS * 100;
END
TABLE FILE GGSALES
SUM DOLLARS NEWDOLL/D12.5nM AS Millions NEWDOLL/D12.3nG AS Billions
BY CATEGORY
ON TABLE SET PAGE NOLEAD
ON TABLE SET STYLE *
GRID=OFF,$
ENDSTYLE
END
```

出力結果は次のとおりです。

Category	Dollar Sales	Millions	Billions
Coffee	17231455	1,723.14550M	1.723G
Food	17229333	1,722.93330M	1.723G
Gifts	11695502	1,169.55020M	1.170G

拡張通貨記号の表示オプション

国際言語サポート (NLS) でデフォルトの通貨記号を構成した場合でも、レポート出力に表示する通貨記号を任意に選択することができます。その場合は、表示オプションの [通貨記号表示 (浮動) - M] または [通貨記号表示 (固定) - N] の代わりに拡張通貨記号フォーマットを使用します。表示オプションの [通貨記号表示 (浮動) - M] または [通貨記号表示 (固定) - N] を使用する場合は、デフォルトのコードページに関連付けられた通貨記号が表示されます。たとえば、英語 (米国) コードページを使用する場合は、ドル記号 ($) が表示されます。

一方、拡張通貨記号フォーマットを使用すると、ドル記号 ($) 以外の記号を表示することができます。たとえば、米国ドル、イギリスポンド、日本円、中国元、またはユーロの記号を表示することができます。拡張通貨記号は、数値フォーマット (I、D、F、P) でサポートされます。

拡張通貨記号フォーマットは、数値表示フォーマットの末尾に 2 文字を組み合わせて指定します。この組み合わせの最初の文字には、感嘆符 (!) またはコロン (:) のいずれかを使用することができます。コロン (:) はすべてのコードページで正しく動作するため、コロン (:) を使用することを推奨します。感嘆符 (!) は、すべてのコードページで共通する記号でないため、使用するコードページによっては感嘆符 (!) が別の記号に変換されて予期しない動作が発生する場合があります。

また、SET コマンドの SET CURSYM_D、SET CURSYM_E、SET CURSYM_F、SET CURSYM_G、SET CURSYM_L、SET CURSYM_Y を使用して、拡張通貨記号フォーマットのデフォルト表示文字を再定義することもできます。たとえば、数値の右側にユーロ記号を表示し、数値とユーロ記号の間にブランクを追加するには、リクエストまたはマスターファイルで SET CURSYM_F コマンドを発行し、拡張通貨記号フォーマットの「:F」を使用します。

```
SET CURSYM_F = ' €'
```

詳細は、『TIBCO WebFOCUS アプリケーション作成ガイド』を参照してください。

下表は、サポートされる拡張通貨表示オプションを示しています。

表示オプション	説明	例
: または !	通貨記号は、ロケールの設定により決定されます。表示は、次のパラメータで制御されます。 ❏ CURRENCY_PRINT_ISO - 通貨記号と ISO コードのどちらを表示するかを指定します。 ❏ CURRENCY_DISPLAY - 通貨記号または ISO コードを数値との相対で表示するかどうかを制御します。 ❏ CURRENCY_ISO_CODE - ISO コードに設定することで、使用する通貨記号を設定します。	D12.2:C
:d または !d	ドル記号 (固定)	D12.2:d

表示オプション	説明	例
:D または !D	ドル記号 (浮動)	D12.2:D
:e または !e	ユーロ記号 (固定)	F9.2:e
:E または !E	ユーロ記号 (浮動) (左側)	F9.2:E
:F または !F	ユーロ記号 (浮動) (右側)	F9.2:F
:G または !G	ドル記号 (浮動) (右側)	F9.2:G
:I または !I	イギリスポンド記号 (固定)	D12.1:I
:L または !L	イギリスポンド記号 (浮動)	D12.1:L
:y または !y	日本円または中国元記号 (固定)	I9:y
:Y または !Y	日本円または中国元記号 (浮動)	I9:Y

参照　拡張通貨記号フォーマット

次のガイドラインが適用されます。

- フォーマット指定の最大長は 8 バイトです。
- 拡張通貨オプションは、フォーマットの最後のオプションとして指定する必要があります。
- 拡張通貨記号フォーマットに、表示オプションの [通貨記号表示 (浮動) - M] または [通貨記号表示 (固定) - N] を含めることはできません。

- 固定通貨記号は、レポートページの最初の行にのみ表示されます。(次の例のように) レポートの列に複数の通貨記号を表示するために、フィールドベースのフォーマット再設定を使用する場合、その 1 行目に関連した記号のみが表示されます。この場合は、固定通貨記号を使用することはできません。
- 端末 I/O プロシジャにより、小文字は大文字として送信されます。そのため、固定の拡張通貨記号はプロシジャでのみ指定することができます。
- 拡張通貨記号フォーマットは、単精度浮動小数点数、倍精度浮動小数点数、パック 10 進数、整数のフォーマットが設定されたフィールドに使用することができます。文字フォーマットおよび可変長文字フォーマットを使用することはできません。

構文　　拡張通貨記号の選択

```
numeric_format {:|!}option
```

説明

`numeric_format`
有効な数値フォーマットです (データタイプ I、D、F、P)。

`: または !`
この記号を必ず追加します。感嘆符 (!) は、すべてのコードページで共通する記号でないため、使用するコードページによっては感嘆符 (!) が別の記号に変換されて予期しない動作が発生する場合があります。コロン (:) はすべてのコードページで不変の記号のため、この記号の使用をお勧めします。

`option`
表示する通貨記号、および浮動、固定のいずれかを指定します。利用可能な値には、次のものがあります。

- **C** - ロケールの設定で特定された通貨記号を表示します。
- **d** - ドル記号 (固定) を表示します。
- **D** - ドル記号 (浮動) を表示します。
- **e** - ユーロ記号 (固定) を表示します。
- **E** - ユーロ記号 (浮動) を左側に表示します。
- **F** - ユーロ記号 (浮動) を右側に表示します。
- **I** - イギリスポンド記号 (固定) を表示します。

- **L** - イギリスポンド記号 (浮動) を表示します。
- **y** - 日本円または中国元 (固定) を表示します。
- **Y** - 日本円または中国元 (浮動) を表示します。

拡張通貨オプションは、フォーマットの最後に配置する必要があります。

例 拡張通貨記号の表示

次のリクエストは、ユーロ記号を表示します。

```
SET PAGE-NUM = OFF
TABLE FILE CENTORD
PRINT PRODNAME QUANTITY PRICE/D10.2!E
BY ORDER_DATE
WHERE QUANTITY GT 700;
ON TABLE SET STYLE *
TYPE = REPORT, GRID = OFF,$
ENDSTYLE
END
```

出力結果は次のとおりです。

Date Of Order:	Product Name:	Quantity:	Price:
2001/10/16	R5 Micro Digital Tape Recorder	726	€89.00
	ZT Digital PDA - Commercial	726	€499.00
2002/03/20	R5 Micro Digital Tape Recorder	751	€89.00
	ZT Digital PDA - Commercial	751	€499.00
2002/04/03	ZC Digital PDA - Standard	751	€299.00
2002/06/07	R5 Micro Digital Tape Recorder	751	€89.00
	ZT Digital PDA - Commercial	751	€499.00
	ZC Digital PDA - Standard	751	€299.00
	R5 Micro Digital Tape Recorder	702	€89.00
	ZT Digital PDA - Commercial	702	€499.00
	ZC Digital PDA - Standard	702	€299.00
2002/06/18	R5 Micro Digital Tape Recorder	751	€89.00
	ZT Digital PDA - Commercial	751	€499.00
2002/10/16	R5 Micro Digital Tape Recorder	798	€89.00
	ZT Digital PDA - Commercial	798	€499.00
2002/12/19	2 Hd VCR LCD Menu	701	€179.00

参照

通貨値のロケール表示オプション

CURRENCY_ISO_CODE、CURRENCY_DISPLAY パラメータは、マスターファイルの DEFINE、DEFINE コマンド、または COMPUTE で、:C 表示オプションを使用する表示パラメータとして、フィールドレベルに適用することができます。

注意：この設定は、FORMAT EXL2K レポート出力ではサポートされません。

構文は次のとおりです。

```
fld/fmt:C(CURRENCY_DISPLAY='pos',
   CURRENCY_ISO_CODE='iso')= expression;
```

説明

`fld`
パラメータを適用するフィールドです。

`fmt`
通貨の値をサポートする数値フォーマットです。

`pos`
数値の位置を基準にして通貨記号の相対位置を定義します。デフォルト値は default です。この設定では、現在有効になっているフォーマットと通貨記号の位置が使用されます。有効な値には、次のものがあります。

- **LEFT_FIXED** 通貨記号を数値の前に左揃えで配置します。
- **LEFT_FIXED_SPACE** 通貨記号を数値の前に左揃えで配置し、記号と数値の間に少なくともブランクを 1 つ挿入します。
- **LEFT_FLOAT** 通貨記号を数値の前に配置し、記号と数値の間にブランクを挿入しません。
- **LEFT_FLOAT_SPACE** 通貨記号を数値の前に配置し、記号と数値の間にブランクを 1 つ挿入します。
- **TRAILING** 通貨記号を数値の後に配置し、記号と数値の間にブランクを挿入しません。
- **TRAILING_SPACE** 通貨記号を数値の後に配置し、記号と数値の間にブランクを 1 つ挿入します。

`iso`
標準の 3 文字通貨コードです (例、米ドルは USD、日本円は JPY)。デフォルト値は default です。この設定では、構成済み言語コードの通貨コードが使用されます。

`expression`
一時項目 (DEFINE) を作成する式です。

注意：複数のレベルで通貨パラメータが指定されている場合、次の優先順位が適用されます。

1. フィールドレベルのパラメータ
2. リクエストで設定されたパラメータ (ON TABLE SET)
3. リクエスト外部のプロシジャで設定されたパラメータ
4. プロファイルで設定されたパラメータ (プロファイル処理の優先実行を使用)。

例 DEFINE での通貨パラメータの指定

次のリクエストは、「Currency_parms」という一時項目 (DEFINE) を作成し、そのフィールド値の右側に、日本円の ISO コード (JPY) を使用して通貨記号を表示します。

```
DEFINE FILE WF_RETAIL_LITE
Currency_parms/D20.2:C(CURRENCY_DISPLAY='TRAILING',CURRENCY_ISO_CODE='JPY')
= COGS_US;
END
TABLE FILE WF_RETAIL_LITE
SUM COGS_US Currency_parms
BY BUSINESS_REGION AS 'Region'
ON TABLE SET PAGE NOLEAD
ON TABLE SET STYLE *
GRID=OFF,$
END
```

出力結果は次のとおりです。

Region	Cost of Goods	Currency_parms
EMEA	$1,247,925.00	1,247,925.00¥
North America	$1,457,020.00	1,457,020.00¥
Oceania	$9,613.00	9,613.00¥
South America	$235,800.00	235,800.00¥

端数処理

小数部を含む値を数値フィールドに割り当てた場合、その値の小数部がフィールドの長さとして指定された桁数を超えると、値を格納または表示する前に許容する長さに端数処理されます。その場合、余分な小数部の最初の桁が 5 未満の場合はその数字が切り捨てられます。また、余分な小数部の最初の桁が 5 以上の場合はその数字が切り上げられます。ただし、浮動小数点数の値では考慮する点がほかにもあります。

たとえば、小数点以下 2 桁のパック 10 進数のフィールドについて考察します。

```
FIELDNAME = PRESSURE, FORMAT = P8.2, $
```

このフィールドに、次のような小数点以下 4 桁の値を割り当てます。

```
PRESSURE = 746.1289
```

余分な小数部の最初の桁、つまり指定したフィールドの長さを超える最初の桁の値は 8 です。この場合、8 は 5 以上の数字のため、PRESSURE の値は切り上げられて次のようになります。

```
746.13
```

各数値フォーマットでは、次のように端数処理されます。

- **整数フォーマット**　小数部を含む値を整数フィールドに割り当てた場合、値を格納する前に端数処理されます。DEFINE または COMPUTE コマンドを使用して値を割り当てると、値の小数部は格納する前に切り捨てられます。
- **パック 10 進数フォーマット**　パック 10 進数フィールドに値を割り当てた場合、その値の小数部がフィールドのフォーマットで指定された桁数を超えると、値を格納する前に端数処理されます。
- **単精度および倍精度浮動小数点数フォーマット**　これらのフィールドのいずれかに値を割り当てた場合、その値の小数部がフィールドのフォーマットで指定された桁数を超えても、値全体がフィールドに格納されます。この場合、フィールドの内部格納領域タイプで定義された桁数が上限です。ただし、この値を表示する場合は端数処理されます。

  なお、浮動小数点数フィールドの値の小数部が内部的に 16 進数表記の循環小数の場合、循環小数である 16 進数は、その値よりわずかに小さい非循環小数として解決され、このわずかに小さい値がフィールド値として格納されます。この場合は、元の値で端数処理の対象となる桁の値が 5 (本来は切り上げ対象) であっても、この値が格納される場合は 4 (切り捨て対象) として扱われます。

  たとえば、小数点以下 1 桁の倍精度浮動小数点数フィールドについて考察します。

  ```
  FIELDNAME = VELOCITY, FORMAT = D5.1, $
  ```

  このフィールドに、次のような小数点以下 2 桁の値を割り当てます。

  ```
  VELOCITY = 1.15
  ```

  この値は、上記の浮動小数点演算の特殊な事情により、わずかに小さい値として格納されます。

  ```
  1.149999
  ```

元の値 1.15 を端数処理すると 1.2 になりますが (余分な小数部の最初の桁が 5 以上)、格納された値は 1.15 よりもわずかに小さい「1.149999」という値になり、この値は端数処理されて 1.1 になります (余分な小数部の最初の桁が 5 未満)。この処理を要約すると次のようになります。

```
format:    D5.1
entered:   1.15
stored:    1.149999
rounded:   1.1
displayed: 1.1
```

文字フォーマット

文字フォーマットは、数字、文字、およびその他の文字で構成され、一連の文字として解釈されるすべての値に使用することができます。

文字フィールドと日付表示オプションを使用して、限定的な日付サポートを利用することもできます。文字フィールドの使用方法についての詳細は、149 ページの 「 日付表示オプションを使用した文字および数値フォーマット 」 を参照してください。

文字の USAGE タイプは A です。長さ指定のフォーマットは n です。ここで、n はフィールドの最大バイト数を表します。FOCUS ファイルセグメントでの文字フィールドの最大長は 3968 バイトで、XFOCUS ファイルセグメントでの最大長は 4096 バイトです。固定フォーマットのシーケンシャルデータソースでは、最大長は 4095 バイトです。フィールドの長さは、マスターファイル、DEFINE FILE コマンド、COMPUTE コマンドで定義することができます。

以下はその例です。

フォーマット	表示
A522	The minutes of today's meeting were submitted...
A2	B3
A24	127-A429-BYQ-49

標準の数値表示オプションは、文字データフォーマットに使用することはできません。ただし、実行時にパターンを指定して、文字データの表示を制御することができます。たとえば、製品コードを複数の区分で表示し、各区分をハイフン (-) で分割する場合、DEFINE コマンドに次の記述を含めます。

```
PRODCODE/A11 = EDIT (fieldname,'999-999-999') ;
```

説明

`fieldname`
　既存のフィールド名です。新しく定義したフィールド名ではありません。

たとえば、値が 716431014 の場合、PRODCODE には 716-431-014 と表示されます。詳細は、『TIBCO WebFOCUS Language リファレンス』を参照してください。

参照　4 キロバイト文字フィールド使用時の注意

- 長い文字フィールドにインデックスを付けることはできません。
- FOCUS データソースでは、セグメントを 4 キロバイトページに収める必要があります。そのため、文字フィールドの最大長は、そのセグメント内の他のフィールドの長さにより異なります。
- 長い文字フィールドを印刷またはホールドすることはできますが、オンラインで参照することはできません。
- 長い文字フィールドはキーとして使用することができます。

16 進数フォーマット

USAGE フォーマットの U*n* を使用して、文字データを 16 進数で表示することができます。対応する ACTUAL フォーマットは A で、長さは USAGE フォーマットの 2 倍になります。

16 進数フォーマットでは、次の処理がサポートされます。

- 表示
- フォーマット再設定
- DEFINE
- COMPUTE
- HOLD (文字出力を生成)
- SAVE

- 文字 (文字フォーマット) に対する選択
- 16 進数フィールドと文字フィールド間での割り当て

たとえば、次のリクエストは文字フォーマットの「Category」フィールドの 16 進数表現を表示します。

```
TABLE FILE GGSALES
SUM CATEGORY/U10 AS Hex,Category DOLLARS UNITS
BY CATEGORY
ON TABLE SET PAGE NOLEAD
ON TABLE SET STYLE *
GRID=OFF,$
ENDSTYLE
END
```

出力結果は次のとおりです。

Category	Hex Category	Dollar Sales	Unit Sales
Coffee	436F6666656520202020	17231455	1376266
Food	466F6F64202020202020	17229333	1384845
Gifts	47696674732020202020	11695502	927880

次のリクエストは、DEFINE コマンドにより 16 進数のフィールドを作成し、HOLD コマンドを追加します。

```
APP HOLD baseapp
DEFINE FILE GGSALES
HEXCAT/U10 = CATEGORY;
END

TABLE FILE GGSALES
SUM HEXCAT DOLLARS UNITS
BY CATEGORY
ON TABLE HOLD AS HOLDHEX FORMAT ALPHA
ON TABLE SET PAGE NOLEAD
ON TABLE SET STYLE *
GRID=OFF,$
ENDSTYLE
END
```

生成されたマスターファイルの内容は次のとおりです。HEXCAT フィールドには、USAGE=U10 および ACTUAL=A20 が記述されています。

```
FILENAME=HOLDHEX , SUFFIX=FIX      , IOTYPE=STREAM, $
  SEGMENT=HOLDHEX, SEGTYPE=S1, $
    FIELDNAME=CATEGORY, ALIAS=E01, USAGE=A11, ACTUAL=A11, $
    FIELDNAME=HEXCAT, ALIAS=E02, USAGE=U10, ACTUAL=A20, $
    FIELDNAME=DOLLARS, ALIAS=E03, USAGE=I08, ACTUAL=A08, $
    FIELDNAME=UNITS, ALIAS=E04, USAGE=I08, ACTUAL=A08, $
```

整数フォーマット

一部のリレーショナルデータソースでは、長さ制限なしの文字データを格納するための STRING データタイプがサポートされます。このデータタイプは、TX データタイプにマッピングすることができます。ただし、WebFOCUS ソート句および選択句で使用する際に、テキストフィールドに制限事項があります。

STRING フィールドのフォーマットには、長さ指定はありません。長さは、取得時に決定されます。以下はその例です。

```
FIELD1/STRING
```

STRING データタイプには、WebFOCUS での文字データタイプの機能がすべて適用されます。STRING フィールド値の最大長さは 2 ギガバイトです。このデータタイプは、STRING データタイプが存在するリレーショナルデータソースと、Delimited HOLD ファイルに継承することが可能で、USAGE と ACTUAL フォーマットの両方が STRING として生成されます。

日付フォーマット

日付フォーマットを使用すると、日付フィールドを定義した後、フィールド値を操作して正しい日付の値として表示することができます。日付フォーマットを使用すると、次の操作を実行することができます。

- 日付構成要素 (例、年、四半期、月、日、曜日) を定義し、日付フィールドから日付構成要素を簡単に抽出する。
- 日付の表示方法に関係なく、レポートを日付順でソートする。
- 特別な日付処理関数を使用せずに、日付演算を実行して日付を比較する。
- 表示または編集フォーマットを考慮せずに、一般的な表記方法で日付を指定する (例、JAN 1 1995)。
- トランザクションの日付を自動検証する。

日付表示オプション

日付フォーマットではタイプや長さは指定しません。その代わりに、日付構成要素オプション (D、W、M、Q、Y、YY) および表示オプションを指定します。下表は、これらのオプションの一覧です。

表示オプション	説明	効果
D	日	日の値を 1 から 31 の範囲で表示します。
M	月	月の値を 1 から 12 の範囲で表示します。
Y	年	年を 2 桁で表示します。
YY	4 桁の年	年を 4 桁で表示します。
T	月または日の変換	USAGE 設定に M が含まれる場合、月の 3 文字の省略形を大文字で表示します。M が含まれていない場合、曜日を表示します。
t	月または日の変換	上記の大文字の T の場合と同様に機能します。ただし、月または日付の最初の文字が大文字で、後に続く文字は小文字になります。*
TR	月または日の変換	上記の大文字の T の場合と同様に機能します。ただし、省略名の代わりに月または日の完全名が表示されます。
tr	月または日の変換	上記の小文字の t の場合と同様に機能します。ただし、省略名の代わりに月または日の完全名が表示されます。*
Q	四半期	Q1 から Q4 を表示します。

表示オプション	説明	効果
W	曜日	USAGE 設定に他の日付構成要素オプションとともに使用して、3 文字の曜日の省略名を大文字で表示します。USAGE 設定にこの日付構成要素オプションのみを指定した場合は、曜日を数字で表示します (1 から 7 までの数字、月曜日が 1)。
w	曜日	上記の大文字の W の場合と同様に機能します。ただし、最初の文字が大文字で、後に続く文字が小文字になります。*
WR	曜日	上記の大文字の W の場合と同様に機能します。ただし、省略名の代わりに曜日の完全名が表示されます。
wr	曜日	上記の小文字の w の場合と同様に機能します。ただし、省略名の代わりに曜日の完全名が表示されます。*
J[UL]or JULIAN	ユリウス暦	ユリウス暦フォーマットで日付を表示します。
YYJ[UL]	ユリウス暦	ユリウス暦フォーマットの日付を YYYYDDD フォーマットで表示します。この 7 桁のフォーマットは、4 桁の年に続けて 1 月 1 日からの通算日を表示します。たとえば、ユリウス暦フォーマットの 2001 年 1 月 3 日は 2001003 になります。

注意：これらの表示オプションを使用する場合は、マスターファイルで実際にフィールドが小文字で保存されていることを確認してください。

次の日付構成要素の組み合わせは、日付フォーマットではサポートされません。

```
I2M, A2M, I2MD, A2MD
```

参照

フィールドフォーマット Y、YY、M、W の格納方法

Y、YY、M は SmartDate フォーマットではありません。SmartDate フォーマットの YMD および YYMD は、基準日「12/31/1900」からのオフセットとして格納されます。SmartDate フォーマットの YM、YQ、YYM、YYQ は、基準日 「12/1900 」からのオフセットとして格納されます。W フォーマットは、曜日を表す 1 から 7 までの 1 桁の整数で格納されます。Y、YY、M フォーマットは、整数として格納されます。Y および M の表示上の長さは 2 です。YY の表示上の長さは 4 です。Y および YY フィールドフォーマットを使用する場合は、次の 2 点に注意する必要があります。

- Y フォーマットは、DEFCENT および YRTHRESH 設定に基づいてソートされません。Y フォーマットのフィールドは、位置をずらした数字ではなく 4 桁の整数のため、YY フィールドとは異なります。
- DEFCENT および YRTHRESH を使用して、フィールドフォーマットを Y から YY に変換することができます。

参照

日付リテラルの解析表

下表は、日付フォーマットの動作を示しています。日付フォーマットの入力桁数がそれぞれの列に示されています。フィールドの USAGE またはフォーマットがそれぞれの行に示されています。行と列の交差部は、入力およびフォーマットの結果を示しています。

日付フォーマット	1	2	3	4
YYMD	*	*	CC00/0m/dd	CC00/mm/dd
MDYY	*	*	*	*
DMYY	*	*	*	*
YMD	*	*	CC00/0m/dd	CC00/mm/dd
MDY	*	*	*	*
DMY	*	*	*	*
YYM	CC00/0m	CC00/mm	CC0y/mm	CCyy/mm

日付フォーマット	1	2	3	4
MYY	*	*	*	*
YM	CC00/0m	CC00/mm	CC0y/mm	CCyy/mm
MY	*	*	0m/CCyy	mm/CCyy
M	0m	mm	*	*
YYQ	CC00/q	CC0y/q	CCyy/q	0yyy/q
QYY	*	*	q/CCyy	*
YQ	CC00/q	CC0y/q	CCyy/q	0yyy/q
QY	*	*	q/CCyy	*
Q	q	*	*	*
JUL	00/00d	00/0dd	00/ddd	0y/ddd
YYJUL	CC00/00d	CC00/0dd	CC00/ddd	CC0y/ddd
YY	000y	00yy	0yyy	yyyy
Y	0y	yy	*	*
D	0d	dd	*	*
W	w	*	*	*

日付フォーマット	5	6	7	8
YYMD	CC0y/mm/dd	CCyy/mm/dd	0yyy/mm/dd	yyyy/mm/dd
MDYY	0m/dd/CCyy	mm/dd/CCyy	0m/dd/yyyy	mm/dd/yyyy

日付フォーマット	5	6	7	8
DMYY	0d/mm/CCyy	dd/mm/CCyy	0d/mm/yyyy	dd/mm/yyyy
YMD	CC0y/mm/dd	CCyy/mm/dd	0yyy/mm/dd	yyyy/mm/dd
MDY	0m/dd/CCyy	mm/dd/CCyy	0m/dd/yyyy	mm/dd/yyyy
DMY	0d/mm/CCyy	dd/mm/CCyy	0d/mm/yyyy	dd/mm/yyyy
YYM	0yyy/mm	yyyy/mm	*	*
MYY	0m/yyyy	mm/yyyy	*	*
YM	0yyy/mm	yyyy/mm	*	*
MY	0m/yyyy	mm/yyyy	*	*
M	*	*	*	*
YYQ	yyyy/q	*	*	*
QYY	q/yyyy	*	*	*
YQ	yyyy/q	*	*	*
QY	q/yyyy	*	*	*
Q	*	*	*	*
JUL	yy/ddd	*	*	*
YYJUL	CCyy/ddd	0yyy/ddd	yyyy/ddd	*
YY	*	*	*	*
Y	*	*	*	*
D	*	*	*	*

日付フォーマット	5	6	7	8
W	*	*	*	*

注意

- CC は、DFC/YRT 設定で指定された世紀に相当する 2 桁の値を表します。
- アスタリスク (*) は、メッセージ FOC177 (無効な日付定数) を表します。
- 日付リテラルは右から左に読み取られます。日付リテラルおよびフィールドは、計算式に使用することもできます。詳細は、『TIBCO WebFOCUS Language リファレンス』を参照してください。

日付区切り記号の制御

日付を表示する際に、日付の区切り記号を制御することができます。基本的な日付フォーマット (例、YMD、MDYY) では、日付構成要素をスラッシュ記号 (/) で区切ります。年-月フォーマットでも同様に、年と四半期をブランクで区切ります (例、94 Q3、Q3 1994)。単一構成要素のフォーマットでは、1 つの数字または名前のみが表示されます。

区切り記号としてピリオド (.)、ダッシュ (-)、ブランクを指定したり、区切り記号を完全に除外したりすることができます。下表は、区切り記号を変更する際の USAGE 設定を示しています。

フォーマット	表示
YMD	93/12/24
Y.M.D	93.12.24
Y-M	93-12
YBMBD	93 12 24 (文字 B はブランクを表します)
Y|M|D	931224 (連結記号「|」は区切り記号を省略します)

注意

- 日付区切り記号の変更が可能な日付フォーマットには、YYMD、MDYY、DMYY、YMD、MDY、DMY、YYM、MYY、YM、MY、YYQ、QYY、YQ、QY があります。
- 日付変換オプションを含むフォーマットで日付区切り記号を変更することはできません。
- 日付表示オプション (例、I8YYMD) を含む文字または数値フォーマットで日付区切り記号のスラッシュ (/) を変更することはできません。

日付の変換

数値で表示した月および曜日は、JAN、January、Wed、Wednesday などの文字に変換することができます。変換した月または曜日には、3 文字の省略名または完全名を使用することができます。名前は大文字、小文字のいずれかで表示することができます。また、曜日 (例、Monday) は日付の前後いずれかに追加することができます。これらのすべてのオプションはそれぞれ独立して機能します。

変換	表示
`MT`	`JAN`
`Mt`	`Jan`
`MTR`	`JANUARY`
`Mtr`	`January`
`WR`	`MONDAY`
`wr`	`Monday`

例　日付フォーマットの使用

下表は、FOCUS 以外のデータソースに格納したデータの USAGE および ACTUAL フォーマットのサンプルを示しています。[値] 列には実際のデータ値、[表示] 列には表示時のデータ値が示されています。

USAGE	ACTUAL	値	表示
wrMtrDYY	A6YMD	990315	Monday, March 15 1999
YQ	A6YMD	990315	99 Q1
QYY	A6YMD	990315	Q1 1999
YMD	A6	990315	99/03/15
MDYY	A6YMD	990315	03/15/1999

なお、ACTUAL フォーマットの日付属性は、FOCUS 以外のデータソースに日付を格納する際の順序を指定します。ACTUAL フォーマットで月、日、年の順序を指定しない場合、USAGE フォーマットの指定から推定されます。

日付フィールドの使用

日付フォーマットに指定したフィールドは、日付の入力時に値が自動検証されます。日付の値として、一般表記の日付リテラル (例、JAN 12 1999) を入力することも、数値の日付リテラル (例、011299) を入力することもできます。

一般表記の日付リテラルを使用すると、日付構成要素間にブランクを挿入したり、月の省略名を使用したりして、簡単で分かりやすい方法で日付を指定することができます。たとえば、1999 年 4 月 25 日は次のいずれかの方法で一般表記の日付リテラルを指定することができます。

```
APR 25 1999
25 APR 1999
1999 APR 25
```

一般表記の日付リテラルは、すべての日付計算およびデータソースの更新機能で使用することができます。次のコードはその一例です。

```
In WHERE screening                    WHERE MYDATE IS 'APR 25 1999'
In arithmetic expressions             MYDATE - '1999 APR 25'
In computational date comparisons     IF MYDATE GT '25 APR 1999'

In comma-delimited data          ...,MYDATE = APR 25 1999, ...
```

下表は、一般表記の日付リテラルのフォーマットを示しています。

リテラル	フォーマット
年-月-日	4 桁の年、大文字 3 文字の省略名または完全名の月、1 桁または 2 桁の日付で構成されます (例、1999 APR 25、APRIL 25 1999)。
年-月	年および月は上記のとおりです。
年-四半期	年は上記のとおりです。四半期には Q および四半期数が追加されます (例、1999 Q3)。
月	月は上記のとおりです。
四半期	四半期は上記のとおりです。
曜日	曜日を表す大文字 3 文字の省略名または完全名です (例、MON、MONDAY)。

一般表記の日付けリテラルの日付構成要素は、ターゲットフィールドの USAGE 設定で指定した順序に関係なく、任意の順序で指定することができます。複数の日付構成要素は、1 つまたは複数のブランクで区切ります。

たとえば、日付フィールドの USAGE 設定が YM の場合、フィールドに入力する一般表記の日付リテラルには、年および月を任意の順序で記述することができます。「MAY 1999」および「1990 APR」はいずれも有効なリテラルです。

数値日付リテラル

数値日付リテラルは、単純な数字の配列という点で、一般表記の日付リテラルとは異なります。数値日付リテラルで入力する日付構成要素の順序は、それに対応する USAGE 設定の日付構成要素の順序に一致させる必要があります。また、数値日付リテラルには、USAGE 設定に含まれているすべての日付構成要素を含める必要があります。たとえば、USAGE 設定が DMY の場合、1999 年 4 月 25 日は次のように記述する必要があります。

```
250499
```

数値日付リテラルは、すべての日付計算およびデータソースの更新機能で使用することができます。

演算式の日付フィールド

演算式で日付フィールドを操作する際の一般的な規則として、1 つの演算式に複数の日付フィールドを使用する場合は、共通の日付構成要素を指定する必要があります。日付構成要素は任意の順序で指定することができます。この場合、表示オプションは無視されます。Y、YY のいずれか、および Q、M、W、D は有効な構成要素です。

なお、四半期、月、曜日に割り当てた演算式は、それぞれ 4、12、7 で剰余計算されるため、第 5 四半期や 13 番目の月などの異常な出力結果が防止されます。

たとえば、NEWQUARTER および THISQUARTER の両方の USAGE 設定が Q で、THISQUARTER の値が 2 の場合、次のステートメントを記述すると、NEWQUARTER の値は 1 になります (5 を 4 で整数除算した剰余)。

```
NEWQUARTER = THISQUARTER + 3
```

日付フィールドの変換

日付フィールドは、フォーマット変換および日付構成要素変換という 2 つの方法で変換することができます。フォーマット変換では、日付フォーマットフィールドの値を日付表示オプションを使用する文字フィールドまたは整数フィールドに割り当てることができます (詳細は、次のセクションを参照)。逆の変換も可能です。

日付構成要素変換では、USAGE で日付構成要素を指定するフィールドを、異なる日付構成要素を指定するもう 1 つのフィールドに割り当てることができます。

たとえば、REPORTDATE (DMY) の値を ORDERDATE (Y) に割り当てることができます。この場合、年は REPORTDATE から抽出されます。REPORTDATE が Apr 27 99 の場合、ORDERDATE は 99 です。

ORDERDATE の値を REPORTDATE に割り当てることもできます。ORDERDATE の値が 99 の場合、REPORTDATE の値は Jan 1 99 です。この場合、ミッシング日付構成要素の値が REPORTDATE に適用されます。

構文　日付フィールドの変換

```
field1/format = field2;
```

説明

`field1`
日付フォーマットフィールド、または日付表示オプションを使用する文字あるいは整数フォーマットフィールドです。

`format`
field1 (ターゲットフィールド) の USAGE (または FORMAT) 設定です。

`field2`
日付フォーマットフィールド、または日付表示オプションを使用する文字あるいは整数フォーマットフィールドです。*field1* および *field2* のフォーマットタイプ (文字、整数、日付) と日付構成要素 (YY、Y、Q、M、W、D) は一致する必要はありません。

日付フィールドの内部表現

日付フィールドは、内部的には 4 バイトのバイナリ整数で表現されます。これは、日付フォーマットの基準日からの経過時間を表します。各フィールドでは、経過時間の単位としてフィールドの最小日付構成要素が使用されます。

たとえば、REPORTDATE の USAGE 設定が MDY の場合、経過時間は日単位で測定され、内部的にはフィールドに入力した日付と基準日との間の経過日数が格納されます。たとえば、February 13, 1964 の数値リテラル (つまり 021364) を入力し、そのフィールドを出力すると、02/13/64 と表示されます。これを式に使用すると次のようになります。

```
NEWDATE = 'FEB 28 1964' - REPORTDATE ;
DAYS/D = NEWDATE ;
```

DAYS の値は 15 になります。ただし、REPORTDATE の内部表現は、December 31, 1900 と February 13, 1964 との間の日数を表す 4 バイトのバイナリ整数になります。

経過時間と同様に、基準日もフィールドの最小日付構成要素の単位に基づいて測定されます。たとえば、YQ の場合、経過時間は四半期単位で計算されますが、基準日は、1900 年の最終四半期です。YM の場合は、経過時間は月単位で計算されますが、基準日は、1900 年の最終月です。

レポートでブランクまたは実際の基準日を表示するには、SET DATEDISPLAY コマンドを使用します。詳細は、『TIBCO WebFOCUS アプリケーション作成ガイド』を参照してください。デフォルト値の OFF は、日付が基準日と一致する場合にブランクを表示します。この値を ON に設定すると、実際の基準日の値が表示されます。

通常は、日付フォーマットの内部表現を意識する必要はありません。ただし、基準日に設定したすべての日付はブランクとして表示され、ブランクまたはすべて 0 (ゼロ) と入力したすべての日付フィールドは確認時に受容されて基準日として解釈されます。これらのフィールドはブランクとして表示されますが、日付計算および式では基準日として解釈されます。

日付構成要素を表示するには、いくつかのタイプのフォーマットが用意されています。値やオフセットの表現方法は、フォーマットのタイプによって異なります。

- 完全日付フォーマットは、基準日 1900 年 12 月 31 日からの経過日数として格納されます。
- 単一日付構成要素フォーマットの Y、YY、M、W、D は、基準日からのオフセットではなく、整数として格納されます。
- 部分日付フォーマットの YM、YQ、YYM、YYQ は、基準日 1900 年 12 月からのオフセットとして格納されます。これらの基準日は、完全構成要素の日付の基準日 (1900 年 12 月 31 日) とは異なります。オフセットは、YM および YYM の場合は基準日からの月数、YQ および YYQ の場合は基準日からの四半期数として表現されます。
- YYMDy、YYMDm、YYMDq など、完全日付フォーマットで構成要素の表示オプションを指定する場合は、YYQ や YM などの構成要素フォーマットと同一の値が表示されます。ただし、これらは完全日付フォーマット （基準日 1900 年 12 月 31 日からの経過日数） として格納されます。このため、構成要素の日付フォーマット (例、YM、YYM、YQ、YYQ) と同一タイプの日付が表示されますが、内部オフセット値は同じではありません。これらは異なるタイプの日付フォーマットと見なされるため、比較や差分の計算に使用することはできません。
- 部分日付または日付構成要素が完全日付フォーマットのフィールドに割り当てられた場合、ミッシング構成要素には値 01 が割り当てられます。

例　完全日付フォーマットおよび日付構成要素フォーマットの使用

次のリクエストは、現在の日付を「FULLDATE」という名前の完全日付フィールドとして取得します。このリクエストは、完全日付フィールドから、「PARTIALDATE」という名前の YYM フォーマットの部分日付フィールド、および「FULLCOMPONENT」という名前の YYMDm フォーマットの構成要素日付フィールドを作成します。さらに、一方のフィールドに部分日付、他方のフィールドに構成要素日付を割り当てることで、「FULLDATE2」および「FULLDATE3」という名前の 2 つの完全日付を作成します。

```
DEFINE FILE GGSALES
FULLDATE/YYMD = '2017/09/12';
PARTIALDATE/YYM =FULLDATE;
FULLCOMPONENT/YYMDm = FULLDATE;
FULLDATE2/YYMD = FULLCOMPONENT;
FULLDATE3/YYMD = PARTIALDATE;
END
TABLE FILE GGSALES
PRINT FULLDATE PARTIALDATE FULLCOMPONENT FULLDATE2 FULLDATE3
BY CATEGORY
WHERE RECORDLIMIT EQ 1
ON TABLE SET PAGE NOLEAD
ON TABLE SET STYLE *
GRID=OFF,$
ENDSTYLE
END
```

下図は、出力結果を示しています。部分日付および構成要素日付が FULLDATE2 および FULLDATE3 に割り当てられる際、割り当て日付は、両者ともに 01 であることに注意してください。

Category	FULLDATE	PARTIALDATE	FULLCOMPONENT	FULLDATE2	FULLDATE3
Coffee	2017/09/12	2017/09	2017/09	2017/09/01	2017/09/01

標準外日付フォーマットの表示

デフォルト設定では、FOCUS 以外のデータソースの日付フィールドに無効な日付が含まれている場合、メッセージが表示され、そのレコード全体がレポートに表示されません。たとえば、A6 の ACTUAL および YMD の USAGE の日付フィールドに '980450' が含まれている場合、そのフィールドを含むレコードは表示されません。無効な日付を含むレコードの残りの部分を表示するには、SET ALLOWCVTERR コマンドを使用します。

注意：ALLOWCVTERR パラメータは、一時項目 (DEFINE) ではサポートされません。

構文　ALLOWCVTERR の呼び出し

```
SET ALLOWCVTERR = {ON|OFF}
```

説明

ON
　無効な日付を含むフィールドを表示します。

OFF
　無効な日付を検知した場合に診断メッセージを生成し、その日付を含むレコードを表示しません。デフォルト値は OFF です。

無効な日付が検知された場合、ALLOWCVTERR は MISSING=ON の設定に基づいて、フィールドの値を MISSING、基準日のいずれかに設定します。

下表は、ALLOWCVTERR=ON に設定された状態で無効な日付が存在すると仮定した場合の DATEDISPLAY および MISSING の間の相互作用の結果を示しています。

	MISSING=OFF	MISSING=ON
DATEDISPLAY=ON	基準日の 19001231 または 1901/1 を表示します。	
DATEDISPLAY=OFF	ブランクを表示します。	

DATEDISPLAY は、基準日の表示方法にのみ影響します。DATEDISPLAY についての詳細は、『TIBCO WebFOCUS アプリケーション作成ガイド』を参照してください。

日付フォーマットのサポート

日付フォーマットフィールドは、次の機能では特別な方法で使用されます。

- **ダイアログマネージャ** 変数を一般表記の日付リテラルに設定すると、その変数は日付フィールドとして機能します。以下はその例です。

```
-SET &NOW = 'APR 25 1960' ;
-SET &LATER = '1990 25 APR' ;
-SET &DELAY = &LATER - &NOW ;
```

  この場合、&DELAY の値は、2 つの日付間の日数 (10,957) になります。

- **抽出ファイル** SAVB ファイルおよびフォーマットが設定されていない HOLD ファイルの日付フィールドは、フィールドの実際の値と標準基準日との差を表す 4 バイトのバイナリ整数として格納されます。SAVE ファイルおよびフォーマットが設定された HOLD ファイルの日付フィールド (例、USAGE WP) は、表示オプションなしで格納されます。

- **グラフ** 日付フィールドは、ACROSS および BY 句のソートフィールドとしてサポートされません。

- **FML** 日付フィールドは、RECAP ステートメントではサポートされません。

日付表示オプションを使用した文字および数値フォーマット

標準の日付フォーマットを使用して日付を表す以外に、文字、整数、パック 10 進数フィールドで日付表示オプション (D、M、Y、T) を使用することもできます。ただし、標準の日付フォーマットで使用可能な日付サポートの一部は使用することができません。

文字および整数フィールドで日付表示オプションを使用すると、特別な日付関数を使用した場合に、限定的な日付機能が提供されます。詳細は、『TIBCO WebFOCUS Language リファレンス』を参照してください。

日付表示オプションを使用して文字または数値フィールドを日付として表現する場合は、年、月、日を指定することができます。これらの 3 つの要素をすべて指定すると、日付は 6 桁 (年を 4 桁で表示する場合は 8 桁) になり、USAGE には次のフォーマットを設定することができます。

フォーマット	表示
I6MDY	04/21/98
I6YMD	98/04/21

フォーマット	表示
P6DMY	21/04/98
I8DMYY	21/04/1998

フォーマットの M の直後に T を追加して、月の値 (1 から 12) をそれぞれ対応する月の名前に変換することができます。以下はその例です。

フォーマット	データ	表示
I6MTDY	05/21/98	MAY 21 98
I4MTY	0698	JUN 98
I2MT	07	JUL

日付の要素が月のみの場合、I2MT フォーマットでは、値 4 は APR と表示されます。この方法は、レポートで列や行を月別にソートする場合に役立ちます。これらの値は、カレンダー順に正しく表示されます。たとえば、JAN、FEB、MAR は、文字ではなく、数値に基づいてソートされます。なお、T 表示オプションを使用しない場合、I2M は通貨記号表示 (浮動) の整数として解釈されます。

日付時間フォーマット

多くのリレーショナルデータソースで使用するタイムスタンプデータタイプと同様に、日付時間フォーマットは日付と時間の両方をサポートします。

日付時間フィールドは、8、10、12 バイトのいずれかで格納されます。日付には 4 バイトが使用され、時間にはミリ秒またはナノ秒をフォーマットに指定するかどうかにより 4、6、8 バイトのいずれかが使用されます。

演算処理では、同一データタイプ間の直接代入、つまり、文字から文字へ、数値から数値へ、日付から日付へ、日付時間から日付時間への代入処理のみを行えます。その他のすべての演算は一連の日付時間関数で実行されます。詳細は、『TIBCO WebFOCUS 関数リファレンス』を参照してください。

日付時間フォーマットは、ISO 8601:2000 日付時間表記標準に準拠した出力値の生成および入力値の受容もサポートします。この表記を有効にするには、SET パラメータおよび特定のフォーマットオプションを使用します。

構文　ISO 標準日付時間表記の有効化

```
SET DTSTANDARD = {OFF|ON|STANDARD|STANDARDU}
```

説明

OFF
: ISO 8601:2000 標準日付時間表記との互換性を提供しません。デフォルト値は OFF です。

ON|STANDARD
: ISO 標準フォーマットの認識および出力を有効にします。これを有効にすると、日付と時間の区切り文字として T を使用すること、秒を桁数で表すときの区切り文字にピリオド (.) およびカンマ (,) を使用すること、世界時の末尾に Z を使用すること、タイムゾーン情報付きの入力の受容が可能になります。STANDARD は ON の同義語です。

STANDARDU
: ISO 標準フォーマットを有効にし (STANDARD と同様)、可能な場合は入力文字列を対応する世界時 (従来のグリニッジ標準時) に変換します。このオプションにより、アプリケーションですべての日付時間値を同一の方法で格納することが可能になります。

例　SET DTSTANDARD の使用

次のリクエストは、ISO 8601:2000 日付時間標準フォーマットで入力された日付時間値を表示します。SET DTSTANDARD=OFF の場合、リクエストが終了し、エラーメッセージ「(FOC177) 無効な日付データです：」が表示されます。

```
SET DTSTANDARD = &STAND
DEFINE FILE EMPLOYEE
-* The following input is format YYYY-MM-DDThh:mm:ss.sTZD
DT1/HYYMDs  =     DT(2004-06-01T19:20:30.45+01:00);
-* The following input has comma as the decimal separator
DT2/HYYMDs  =     DT(2004-06-01T19:20:30,45+01:00);
DT3/HYYMDs  =     DT(20040601T19:20:30,45);
DT4/HYYMDUs =     DT(2004-06-01T19:20:30,45+01:00);
END
TABLE FILE EMPLOYEE
HEADING CENTER
"DTSANDARD = &STAND "
" "
SUM CURR_SAL NOPRINT DT1 AS 'DT1: INPUT = 2004-06-01T19:20:30.45+01:00'
OVER DT2 AS 'DT2: INPUT = 2004-06-01T19:20:30,45+01:00'
OVER DT3 AS 'DT3: INPUT = 20040601T19:20:30,45'
OVER DT4 AS 'DT4: OUTPUT  FORMAT HYYMDUs'
END
```

DTSTANDARD= STANDARD の場合、出力結果では入力値は受容されていますが、DT1、DT2、DT4 のタイムゾーンオフセット (+01:00) は無視されています。DT4 フォーマットの「U」という文字により、日付と時間の間に「T」という区切り文字が使用されています。

```
                    DTSANDARD = STANDARD

DT1: INPUT = 2004-06-01T19:20:30.45+01:00  2004-06-01 19:20:30.450
DT2: INPUT = 2004-06-01T19:20:30,45+01:00  2004-06-01 19:20:30.450
DT3: INPUT = 20040601T19:20:30,45          2004-06-01 19:20:30.450
DT4: OUTPUT  FORMAT HYYMDUs                2004-06-01T19:20:30.450
```

DTSTANDARD= STANDARDU の場合、出力結果ではタイムゾーンオフセット (+01:00) が減算され、DT1、DT2、DT4 の値がユニバーサル時間に変換されています。

```
                    DTSANDARD = STANDARDU

DT1: INPUT = 2004-06-01T19:20:30.45+01:00  2004-06-01 18:20:30.450
DT2: INPUT = 2004-06-01T19:20:30,45+01:00  2004-06-01 18:20:30.450
DT3: INPUT = 20040601T19:20:30,45          2004-06-01 19:20:30.450
DT4: OUTPUT  FORMAT HYYMDUs                2004-06-01T18:20:30.450
```

日付時間フィールドの記述

マスターファイルの USAGE (または FORMAT) 属性を使用して、日付時間フィールド値をレポート出力およびフォームに表示する方法、これらの値が式および関数で動作する方法を指定します。FOCUS データソースでは、この属性で値を格納する方法も指定します。

日付時間フィールドはフォーマットタイプの H で指定します。日付時間フィールドの USAGE 属性に H フォーマットコードを設定して、フィールドの長さまたは関連する日時表示オプションのいずれかを指定することができます。

日付時間フィールドの MISSING 属性には、ON または OFF を指定することができます。この属性を OFF に設定すると、日付時間フィールドに値が存在しない場合にデフォルトのブランクが表示されます。

構文　表示オプションを使用しない数値日付時間値の記述

このフォーマットは、文字 HOLD ファイルまたはトランザクションファイルでの使用に適しています。

```
USAGE = Hn
```

説明

n

1 から 23 バイトまでのフィールドの長さです。この長さには、日付を表示する最大 8 バイトの長さ、および時間を表示する最大 9、12、15 バイトのいずれかの長さが含まれます。長さが 20 バイト未満の場合、日付は右側で切り捨てられます。

8 バイトの日付 YYYYMMDD には、年に 4 桁、月に 2 桁、日付に 2 桁が含まれます。

9 バイトの時間 HHMMSSsss には、時間に 2 桁、分に 2 桁、秒に 2 桁、ミリ秒に 3 桁が含まれます。ミリ秒構成要素には、秒の小数部が 3 桁で表されます。

12 バイトの時間 HHMMSSsssmmm には、時間に 2 桁、分に 2 桁、秒に 2 桁、ミリ秒に 3 桁、マイクロ秒に 3 桁が含まれます。ミリ秒構成要素には、秒の値の小数部が 3 桁で表されます。マイクロ秒構成要素には、ミリ秒の値に加えてさらに 3 桁の小数部が表されます。

15 バイトの時間 HHMMSSsssmmmnnn には、時間に 2 桁、分に 2 桁、秒に 2 桁、ミリ秒に 3 桁、マイクロ秒に 3 桁、ナノ秒に 3 桁が含まれます。ミリ秒構成要素には、秒の値の小数部が 3 桁で表されます。マイクロ秒構成要素には、ミリ秒の値に加えてさらに 3 桁の小数部が表されます。ナノ秒構成要素には、マイクロ秒の値に加えてさらに 3 桁の小数部が表されます。

このフォーマットでは、日付と時間構成要素間のブランク、構成要素内の小数点、ブランク、区切り記号はありません。時間は 24 時間制で入力します。たとえば、19991231225725333444 の値は 1999/12/31 10:57:25.333444PM を表します。

構文

時間値のみの記述

```
USAGE = Htimefmt1
```

説明

`timefmt1`
時間のみを表示するための USAGE フォーマットです。時間、分、秒の各構成要素は常にコロン (:) で区切り、ブランクは挿入しません。時間値では、am/pm 標識の直前にブランクを配置できます。詳細は、154 ページの 「 時間値のみの表示オプション 」 を参照してください。

参照

時間値のみの表示オプション

下表は、時間のみの USAGE 属性に使用する有効な時間表示オプションの一覧です。ここでは、時間の値として 2:05:27.123456444 a.m. を想定します。

オプション	説明	効果
`H`	時間 (2 桁)。 オプションの a、b、A、B のいずれかがフォーマットに含まれている場合、時間の値は 01 から 12 になります。 それ以外の場合、時間の値は 00 から 23 になり、00 が午前零時を表します。	2 桁の時間を表示します。以下はその例です。 `USAGE = HH` では、`02` と表示されます。
`h`	ゼロサプレス (ゼロを非表示) を使用した時間。 オプションの a、b、A、B のいずれかがフォーマットに含まれている場合、時間の値は 1 から 12 になります。 それ以外の場合は、時間は 0 (ゼロ) から 23 です。	ゼロサプレス (ゼロを非表示) を使用して時間を表示します。以下はその例です。 `USAGE = Hh` では、`2` と表示されます。

オプション	説明	効果
I	分 (2 桁)。 分の値は 00 から 59 です。	2 桁の分を表示します。以下はその例です。 USAGE = HHI では、02:05 と表示されます。
i	ゼロサプレス (ゼロを非表示) を使用した分。 分の値は 0 から 59 です。	ゼロサプレス (ゼロを非表示) を使用して分を表示します。これを時間フォーマット (H または h) とともに使用することはできません。以下はその例です。 USAGE = Hi では、5 と表示されます。
S	秒 (2 桁)。 秒の値は 00 から 59 です。	2 桁の秒を表示します。以下はその例です。 USAGE = HHIS では、02:05:27 と表示されます。
s	ミリ秒 (秒の小数点以下 3 桁)。 000 から 999	秒を小数点以下 3 桁に表示します。以下はその例です。 USAGE = HHISs では、02:05:27.123 と表示されます。
m	マイクロ秒 (ミリ秒以下の追加の 3 桁)。 000 から 999	秒を小数点以下 6 桁に表示します。以下はその例です。 USAGE = HSsm では、27.123456 と表示されます。
n	ナノ秒 (マイクロ秒以下の追加の 3 桁)。 000 から 999	秒を小数点以下 9 桁に表示します。以下はその例です。 USAGE = HSsn では、27.123456444 と表示されます。

オプション	説明	効果
x	S、s、m、n を使用する代わりに、x オプションを使用して、小数部 9 桁を最大とする秒の桁数を指定することもできます。ここで、x は 1 から 9 までの数値です。別の方法として、s、m、n フォーマットを使用して、小数部を 3 桁、6 桁、9 桁で表示することもできます。	`USAGE = HHI1` では、`02:05:27.1` と表示されます。
A	大文字の AM または PM を含む 12 時間表示。	01 から 12 の時間に AM または PM を続けて表示します。以下はその例です。 `USAGE = HHISA` では、`02:05:27AM` と表示されます。
a	小文字の am または pm を含む 12 時間表示。	01 から 12 の時間に AM または PM を続けて表示します。以下はその例です。 `USAGE = HHISa` では、`02:05:27am` と表示されます。
B	大文字の AM または PM を含む 12 時間表示。AM または PM の前にブランクを 1 つ追加します。	01 から 12 の時間の後にブランク 1 つと AM または PM を続けて表示します。以下はその例です。 `USAGE = HHISB` では、`02:05:27 AM` と表示されます。
b	小文字の am または pm を含む 12 時間表示。am または pm の前にブランクを 1 つ追加します。	01 から 12 の時間の後にブランク 1 つと am または pm を続けて表示します。以下はその例です。 `USAGE = HHISb` では、`02:05:27 am` と表示されます。

オプション	説明	効果
Z	ユニバーサル時間を示す Z を含めた 24 時間表示。Z を AM/PM とともに指定することはできません。	01 から 24 の時間に Z を続けて表示します。以下はその例です。 USAGE = HHISZ では、14:30[:20.99]Z と表示されます。

フォーマットに複数の時間表示オプションを含める場合は、次の規則が適用されます。

- これらのオプションは、時間、分、秒、ミリ秒、マイクロ秒、ナノ秒の順に指定する必要があります。
- 最初のオプションには、時間、分、秒のいずれかを指定する必要があります。
- 中間の構成要素を省略することはできません。時間を指定した場合、次のオプションは分にする必要があります。秒にすることはできません。

注意：AM/PM 時間表示オプションのいずれかを指定しない限り、時間構成要素では 24 時間制が使用されます。

構文　日付時間値の記述

```
USAGE = Hdatefmt [separator] [timefmt2]
```

説明

`datefmt`
日付時間フィールドの日付の部分を表示するための USAGE フォーマットです。詳細は、158 ページの「 日付時間フィールドの日付構成要素に使用する表示オプション 」を参照してください。

`separator`
日付構成要素間の区切り文字です。デフォルトの区切り文字はスラッシュ (/) です。その他の有効な区切り文字には、ピリオド (.)、ハイフン (-)、ブランク (B)、なし (N) があります。変換された月では、これらの区切り文字は k オプションの未使用時にのみ指定可能です。

STANDARD と STANDARDU の設定では、日付の区切り文字は常にハイフン (-) です。日付と時間のデフォルトの区切り文字はブランクです。ただし、文字 U を区切り文字オプションとして指定する場合、日付と時間は文字 T で区切ります。

`timefmt2`
日付の後に続く時間のフォーマットです。時間と日付はブランクで区切ります。時間構成要素間はコロン (:) で区切ります。 時間のみに使用するフォーマットと異なり、日付フォーマットに続く時間フォーマットは最大 2 文字で構成されます。最初の文字が表示する時間構成要素のすべてを表し、2 つ目の文字が AM/PM オプションを表します。詳細は、161 ページの 「 日付時間フィールドの時間構成要素に使用する表示オプション 」 を参照してください。

参照 日付時間フィールドの日付構成要素に使用する表示オプション

下表は、日付フォーマットに使用可能な表示オプションの有効な組み合わせを示しています。ここでは、日付として February 5, 1999 を想定します。

オプション	説明	例
`Y`	2 桁の年	`99`
`YY`	4 桁の年	`1999`
`M`	2 桁の月 (01 から 12 まで)	`02`
`MT`	月の完全名	`February`
`Mt`	月の短縮名	`Feb`
`D`	2 桁の日	`05`
`d`	ゼロを非表示にした日。0 (ゼロ) はブランクで置き換えられます。	`5`
`e`	ゼロを削除した日。日の数値が左側に移動し、その右側にある構成要素もすべて左側に移動します。 日付区切り文字が必要です。	`5`

オプション	説明	例
o	ゼロを削除した月。このオプションを使用すると、e オプション (ゼロを削除した日) が自動的に実装されます。月および日の数値が左側に移動し、その右側にある構成要素もすべて左側に移動します。 日付区切り文字が必要です。	5
k	月または日の後に年が続き、月を短縮名または完全名に変換するフォーマットを使用する場合は、k を指定してカンマ (,) およびブランクで年と日を区切ります。それ以外の場合、区切り文字はブランクになります。	USAGE = HMtDkYY prints Feb 05, 1999

注意：AM/PM 時間表示オプションのいずれかを指定しない限り、時間構成要素には 24 時間制が使用されます。

例　日付時間、月、日付を表す数値でのゼロ削除の使用

次のリクエストは、「01/01/2013」という日付時間値を作成します。次に、この値を次のフォーマットで表示します。

- 月と日付の通常の表示 - HMDYY フォーマット
- 月のゼロ削除、日付のゼロ削除 - HoeYY フォーマット
- 月のゼロ削除、日付のゼロ削除なし (日付にゼロ削除が強制的に適用される) - HodYY フォーマット

- 日付のゼロ削除、月のゼロ削除なし - HMeYY フォーマット。この場合、日付のゼロ削除により、月のゼロ削除は強制的に適用されません。

```
DEFINE FILE GGSALES
DATE1A/HMDYY = DT(01/01/2013);
DATE1B/HoeYY = DATE1A;
DATE1C/HodYY = DATE1A;
DATE1D/HMeYY = DATE1A;
END
TABLE FILE GGSALES
SUM DOLLARS NOPRINT
DATE1A AS 'HMDYY'
DATE1B AS 'HoeYY'
DATE1C AS 'HodYY'
DATE1D AS 'HMeYY'
ON TABLE SET PAGE NOPAGE
END
```

出力結果は次のとおりです。

```
HMDYY       HoeYY       HodYY       HMeYY
-----       -----       -----       -----
01/01/2013  1/1/2013    1/1/2013    01/1/2013
```

例　ゼロサプレスとゼロ削除の比較

次のリクエストは、日付時間フォーマットで 2 つの日付を作成します。この日構成要素には、リーディングゼロ (01) が含まれています。1 つ目の日付では、日構成要素が先頭の構成要素で、左端に表示されます。2 つ目の日付では、日構成要素が 2 番目の構成要素で、中央に表示されます。このリクエストは、これらの日付を次のフォーマットで表示します。

- すべてのゼロを表示 - HDMYY フォーマット
- 日構成要素にゼロサプレスを使用 - HdMYY フォーマット

❑ 日構成要素にゼロ削除を使用 - HeMYY フォーマット

```
DEFINE FILE GGSALES
DATE1A/HDMYY = DT(01/12/2012);
DATE2A/HMDYY = DT(12/01/2012);
DATE1B/HdMYY = DATE1A;
DATE2B/HMdYY = DATE2A;
DATE1C/HeMYY = DATE1A;
DATE2C/HMeYY = DATE2A;
END
TABLE FILE GGSALES
SUM DOLLARS NOPRINT
DATE1A AS 'HDMYY'
DATE2A AS ''      OVER
DATE1B AS 'HdMYY'
DATE2B AS ''      OVER
DATE1C AS 'HeMYY'
DATE2C AS ''
ON TABLE SET PAGE NOPAGE
```

出力結果では、1 行目のすべての日付に 0 (ゼロ) が表示されています。2 行目では、日付を表す数値にゼロサプレスが適用されています。ここでは、0 (ゼロ) がブランクに置き換えられ、すべての構成要素の位置が、1 行目の構成要素の位置と一致しています。3 行目では、ゼロ削除が適用されています。日付を表す数値から 0 (ゼロ) が削除され、残りの数値がすべて左側へ移動しています。

```
HDMYY  01/12/2012    12/01/2012
HdMYY   1/12/2012    12/ 1/2012
HeMYY  1/12/2012     12/1/2012
```

参照 **日付時間フィールドの時間構成要素に使用する表示オプション**

下表は、有効な表示オプションの一覧を示しています。ここでは、日付および時間をそれぞれ February 5, 1999 および 02:05:25.444555333 a.m. と想定します。

オプション	説明	例
H	時間を表示します。	USAGE = HYYMDH では、1999/02/05 02 と表示されます。
I	時間：分を表示します。	USAGE = HYYMDI では、1999/02/05 02:05 と表示されます。

オプション	説明	例
S	時間：分：秒を表示します。	USAGE = HYYMDS では、1999/02/05 02:05:25 と表示されます。
s	時間：分：秒.ミリ秒を表示します。	USAGE = HYYMDs では、1999/02/05 02:05:25.444 と表示されます。
m	時間：分：秒.マイクロ秒を表示します。	USAGE = HYYMDm では、1999/02/05 02:05:25.444555 と表示されます。
n	時間：分：秒.ナノ秒を表示します。	USAGE = HYYMDn では、1999/02/05 02:05:25.444555333 と表示されます。
x	S、s、m、n を使用する代わりに、x オプションを使用して、小数部 9 桁を最大とする秒の桁数を指定することもできます。ここで、x は 1 から 9 までの数値です。別の方法として、s、m、n フォーマットを使用して、小数部を 3 桁、6 桁、9 桁で表示することもできます。	USAGE = HYYMD1 では、1999/02/05 02:05:25.4 と表示されます。
A	AM または PM を表示します。このオプションでは 12 時間制が使用され、時間をゼロサプレスで表示します (ゼロを非表示)。	USAGE = HYYMDSA では、1999/02/05 2:05:25AM と表示されます。
a	am または pm を表示します。このオプションでは 12 時間制が使用され、時間をゼロサプレスで表示します (ゼロを非表示)。	USAGE = HYYMDSa では、1999/02/05 2:05:25am と表示されます。

オプション	説明	例
B	先頭にブランクを 1 つ挿入して AM または PM を表示します。このオプションでは 12 時間制が使用され、時間をゼロサプレスで表示します (ゼロを非表示)。	`USAGE = HYYMDSB` では、`1999/02/05 2:05:25 AM` と表示されます。
b	先頭にブランクを 1 つ挿入して am または pm を表示します。このオプションでは 12 時間制が使用され、時間をゼロサプレスで表示します (ゼロを非表示)。	`USAGE = HYYMDSb` では、`1999/02/05 2:05:25 am` と表示されます。
Z	ユニバーサル時間を示す Z を表示します。このオプションでは 12 時間制が使用されます。Z を AM/PM とともに指定することはできません。	`USAGE = HHISZ` では、 `14:30[:20.99]Z` と表示されます。

日付構成要素で指定可能な組み合わせおよび順序は次のとおりです。

- 先頭に年を指定する組み合わせは Y、YY、YM、YYM、YMD、YYMD です。
- 先頭に月を指定する組み合わせは M、MD、MY、MYY、MDY、MDYY です。
- 先頭に日を指定する組み合わせは D、DM、DMY、DMYY です。

参照 日付時間フォーマット使用時の注意

- 時間構成要素を使用するには、日付構成要素を指定する必要があります。
- k オプションを使用する場合は、日付区切り文字を変更することはできません。

文字フォーマット A*n*V

文字フォーマット A*n*V は、FOCUS、XFOCUS、リレーショナルデータソースのマスターファイルでサポートされます。このフォーマットは、リレーショナルデータベース管理システムでサポートされる VARCHAR (可変長文字) データタイプを表すために使用します。

リレーショナルデータソースでは、AnV は VARCHAR フィールドの実際の長さの情報を保持します。この情報は、異なる RDBMS の VARCHAR フィールドに値を入力する際に重要です。この情報は、文字列連結時に末尾のブランクを保持するかどうかに影響します。また Oracle の場合は、文字列の比較に影響します。なお、Oracle 以外のリレーショナルエンジンでは、文字列連結の末尾のブランクは無視されます。

FOCUS および XFOCUS データソースでは、AnV は、実際の可変長文字列サポートを提供するわけではありません。これは、フィールドに格納されたデータの実際の長さを保持するために、2 つの先行バイトが追加された固定長文字フィールドです。この長さは、2 バイトを占有する短整数値として格納されます。AnV フィールドの一部として入力した末尾のブランクもフィールドの長さとして計上されます。

注意：2 バイトのオーバーヘッドおよびそれらの削除に必要な追加処理のために、非リレーショナルデータソースで AnV フォーマットは使用しないことをお勧めします。

構文　マスターファイルでの A*n*V フィールドの指定

```
FIELD=name, ALIAS=alias, USAGE=AnV [,ACTUAL=AnV] , $
```

説明

`n`

フィールドのサイズ (最大長) です。1 から 4093 までの値を指定することができます。なお、この長さの格納には追加の 2 バイトが使用されるため、実際の A4093V フィールドの長さは 4095 バイトになります。サイズゼロ (0、A0V) はサポートされません。フィールドのインスタンスの長さは 0 (ゼロ) にすることができます。

注意：HOLD FORMAT ALPHA を使用すると、マスターファイル内に AnW の ACTUAL フォーマットが作成されます。詳細は、166 ページの 「HOLD ファイルへの AnV フィールドの継承」を参照してください。

例　マスターファイルでの A*n*V フォーマットの指定

次の例は、サイズが 200 の DB2 データソースのマスターファイルで指定した VARCHAR フィールドを示しています。

```
$ VARCHAR FIELD USING AnV
  FIELD=VARCHAR200, ALIAS=VC200, USAGE=A200V, ACTUAL=A200V,MISSING=ON ,$
```

次の例は、サイズが 200 の FOCUS データソースのマスターファイルで指定した AnV フィールドを示しています。

```
FIELD=ALPHAV, ALIAS=AV200, USAGE=A200V, MISSING=ON ,$
```

データソースに AnV フィールドがある場合は、次のように記述して長さを指定せずに HOLD FORMAT ALPHA ファイルを作成します。

```
FIELD=ALPHA, USAGE=A25, ACTUAL=A25V, $
```

または

```
DEFINE ...
  ALPHA/A25 = VARCHAR ;
END
```

または

```
COMPUTE ALPHA/A25 = VARCHAR ;
```

マスターファイルを変更または作成して AnV を追加するには、データを変換して、フィールドの先頭にその長さを追加する必要があります。たとえば、次のようにフィールドを記述する場合に HOLD コマンドを発行します。

```
FIELD=VARCHAR, ,USAGE=A25V, ACTUAL=A25, $
```

または

```
DEFINE ...
  VARCHAR/A25V = ALPHA ;
END
```

または

```
COMPUTE VARCHAR/A25V = ALPHA ;
```

参照　A*nV* フォーマット使用時の注意

- ここに記載された制限を除いて、AnV は An が使用可能な任意の場所で使用することができます。
- このデータタイプでは、リレーショナルデータソースの CREATE FILE を含めて、FOCUS および SQL のすべての演算がサポートされます。
- JOIN は、An と AnV フィールドの間ではサポートされません。
- DBCS 文字はサポートされます。An フォーマットと同様に、文字数は 4 キロバイトデータ領域内に収める必要があります。
- COMPUTE および DEFINE は、左辺で指定されたデータタイプを生成します。
- AnV と TX フィールド間の変換はサポートされません。

- AnV フィールドに日付表示オプションを含めることはできません。

参照

HOLD ファイルへの A*n*V フィールドの継承

HOLD FORMAT ALPHA コマンドを使用して AnV フィールドをシーケンシャルデータソースに継承する場合、2 バイトの整数長は 6 桁の文字長に変換されます。HOLD ファイルのフィールドは、文字データが続くこの 6 桁の数で構成されます。このフィールドのフォーマット属性は次のとおりです。

```
... USAGE=AnV, ACTUAL=AnW
```

AnW は、HOLD FORMAT ALPHA の結果として生成され、必要に応じて読み取りおよび入力に使用することができます。HOLD ファイルで、このフィールドが占有するバイト数は 6+*n* です。

例

HOLD ファイルへの A*n*V フィールドの継承

たとえば、「TITLEV」という A39V フィールドは、次のように HOLD ファイルに継承されます。

```
FIELDNAME = TITLEV ,E03 ,A39V ,A39W ,$
```

バイナリ HOLD ファイルでは ACTUAL フォーマットには完全な 4 バイトワードになるまでブランクが追加されますが、USAGE および ACTUAL フォーマットは AnV です。HOLD ファイルで、このフィールドが占有するバイト数は 2+*n* です。

AnV フィールドをデータソースに入力する場合、設定した長さを超えて入力したすべてのバイトは無視されます。これらのバイトは入力処理の一部としてブランクに設定されます。

HOLD FORMAT *sqlengine* コマンドを使用してリレーショナルデータソースを作成すると、AnV フィールドによりそのリレーショナルデータソース内に VARCHAR フィールドが生成されます。

たとえば、名前が TITLEV の A39V フィールドは、HOLD FORMAT DB2 ファイルに次のように継承されます。

```
FIELDNAME = 'TITLEV', 'TITLEV', A39V, A39V ,$
```

テキストフィールドのフォーマット

任意の文字の組み合わせをテキストフィールドとして格納することができます。

構文　マスターファイルでのテキストフィールドの指定

```
FIELD = fieldname, ALIAS = aliasname, USAGE = TXn[F],$
```

説明

`fieldname`
テキストフィールドに割り当てる名前です。

`aliasname`
フィールド名の別名です。

`n`
TABLE 内で、テキストフィールドの出力表示の長さです。表示する長さの範囲は 1 から 256 バイトです。

すべての文字、数字、特殊文字は、このフォーマットで格納することができます。下表は、テキストフィールドフォーマットのサンプルを示しています。

フォーマット	表示
TX50	This course provides the DP professional with the skills needed to create, maintain, and report from FOCUS data sources.
TX35	This course provides the DP professional with the skills needed to create, maintain, and report from FOCUS data sources.

テキストフィールドフォーマットでは、標準の編集オプションを使用することはできません。

参照　テキストフィールドフォーマット使用時の注意

- テキストフィールドと文字フィールド間の変換は、DEFINE および COMPUTE コマンドでサポートされます。
- 複数のテキストフィールドがサポートされ、これらのフィールドのセグメント内での位置に制限はありません。

格納データタイプ - ACTUAL

ACTUAL 属性は、データを実際にデータソースに格納する際のデータタイプおよび長さを記述します。文字データタイプなどのいくつかのデータタイプがユニバーサルであるのに対し、他のデータタイプはデータソースのタイプにより異なります。ユニークデータタイプをサポートするデータソースもあります。そのため、ACTUAL 属性に割り当て可能な値は、データソースのタイプにより異なります。

ACTUAL 属性

この属性は、データが実際のデータソースに存在する際のデータのタイプおよび長さを記述します。この情報は、データソースの現在の記述に基づきます。ACTUAL 属性は、FOCUS 以外のデータソースのマスターファイルに記述する特別な特性の 1 つです。この属性は FOCUS 以外のデータ構造のフォーマットを記述する場合にのみ使用するため、FOCUS データ構造のマスターファイルでは使用されません。

データソースに文字フィールドとして格納された日付があり、その日付を WebFOCUS の日付に変換してレポートでソートまたは集計する必要がある場合は、マスターファイルで DATEPATTERN 属性を使用することができます。WebFOCUS は、指定されたパターンを使用して文字日付を WebFOCUS の日付に変換します。

構文　ACTUAL 属性の指定

```
ACTUAL = format
```

説明

`format`
下表のいずれかの値で構成されます。この表には、読み取り可能なデータタイプのコードが示されています。

ACTUAL タイプ	説明
DATE	入力する日付と日付フォーマットの基準日との差を表す内部的な 4 バイトの整数フォーマットです。

ACTUAL タイプ	説明
An	ここで、*n* の値は固定フォーマットのシーケンシャルでは 1 から 4095、FOCUS 以外のその他のデータソースでは 1 から 256 です。A から Z、0 から 9 の英数文字です。 A*n* は、すべての日付時間文字列フォーマットおよび H*n* 表示フォーマットを受容します。ACTUAL=A*n* は、文字 HOLD ファイルまたは SAVE ファイルに存在する日付時間フィールドも受容します。 文字フォーマットは、16 進数の USAGE (U) で使用することもできます。ACTUAL の長さは、USAGE の長さの 2 倍にする必要があります。
D8	内部的に 8 バイトで格納された倍精度浮動小数点数です。
F4	内部的に 4 バイトで格納された単精度浮動小数点数です。
Hn	H8、H10、H12 は、バイナリ HOLD ファイルまたは SAVB ファイルに存在する日付時間フィールドを受容します。
In	バイナリ整数は次のとおりです。 I1 = 1 バイトのバイナリ整数。 I2 = 2 バイトのバイナリ整数。 I4 = 4 バイトのバイナリ整数。 I8 = 8 バイトのバイナリ整数。 **注意：**USAGE に P または D が指定されている必要があります。小数値は、USAGE が P または D の小数値に変換されて受容されます。
M8	内部的に 8 バイトで格納された 10 進数浮動小数点数 (MATH) です。

ACTUAL タイプ	説明
Pn	*n* = 1-16。パック 10 進数の内部フォーマット。*n* はバイト数を表し、末尾のバイトに格納する 1 桁と符号 (+ または -) を除いて、それぞれのバイトに 2 桁ずつ格納されます。たとえば、P6 は 11 桁と 1 つの符号を表します。
STRING	データタイプが STRING のリレーショナルデータソースです。長さの指定はありません。
X16	内部的に 16 バイトで格納された拡張 10 進数浮動小数点数 (XMATH) です。
Zn	*n* は 1 から 31 です。ゾーン 10 進数の内部フォーマットです。*n* は桁数を表し、それぞれ 1 バイトの格納領域を使用します。末尾の桁には、1 桁と符号が格納されます。 フィールドに想定小数点が含まれる場合は、Z*n* の ACTUAL フォーマットおよび P*m.d* の USAGE フォーマットを使用してフィールドを表します。ここで、*m* は表示する桁数と想定小数点の合計、*d* は小数部の桁数を表します。また、*m* には *n* より 少なくとも 1 つ大きい値を指定する必要があります。たとえば、ACTUAL=Z5 で小数点以下 1 桁のフィールドには、USAGE=P6.1 または P7.1 以上が必要です。

注意

- データソースをプログラムで作成した場合を除いて、すべての文字はタイプ A (文字)、タイプ Z (ゾーン 10 進数) のいずれかになります。

- 日付時間値でサポートされる ACTUAL フォーマットは、A*n*、H8、H10、H12 です。A*n* は、すべての日付時間文字列および H*n* USAGE 表示フォーマットを受容します。ACTUAL=H8、H10、H12 は、バイナリ HOLD ファイルまたは SAVB ファイルに存在する日付時間フィールドを受容します。ACTUAL=A*n* は、文字 HOLD ファイルまたは SAVE ファイルに存在する日付時間フィールドも受容します。
- 日付時間フィールドを含むデータソースからバイナリ HOLD ファイルを作成すると、そのフィールドの ACTUAL フォーマットは H*n* になります。日付時間フィールドを含むデータソースから文字 HOLD ファイルを作成すると、そのフィールドの ACTUAL フォーマットは A*n* になります。

参照 **ACTUAL から USAGE への変換**

下表の ACTUAL フォーマットから USAGE (表示) フォーマットへの変換は自動的に処理され、関数を呼び出す必要はありません。

ACTUAL	USAGE
A	A, D, F, I, P, date format, date-time format
D	D
DATE	date format
F	F
H	H
I	I, date format
M	M
P	P, date format
X	X
Z	D, F, I, P

フィールドへの地理的役割の追加

フィールドが地理的な位置や座標を表す場合、マスターファイルで GEOGRAPHIC_ROLE 属性を使用することで、正確な地理的役割を特定することができます。

GEOGRAPHIC_ROLE 属性

この属性は、このフィールドが表すロケーションインテリジェンスデータのタイプを指定します。

構文　地理的役割の指定

```
GEOGRAPHIC_ROLE = georole
```

説明

`georole`

有効な地理的役割です。地理的役割には、名前、郵便番号、ISO (国際標準化機構) コード、FIPS (連邦情報処理標準) コード、NUTS (地域統計分類単位) コードのいずれかを指定することができます。以下は、サポートされている地理的役割のリストを示しています。

- ADDRESS_FULL　完全な住所です。
- ADDRESS_LINE　番地です。
- CITY　都市名です。
- CONTINENT　大陸名です。
- CONTINENT_ISO2　ISO-3166 大陸コードです。
- COUNTRY　国名です。
- COUNTRY_FIPS　FIPS 国コードです。
- COUNTRY_ISO2　ISO-3166-2 国コードです。
- COUNTRY_ISO3　ISO-3166-3 国コードです。
- GEOMETRY_AREA　面ジオメトリです。
- GEOMETRY_LINE　線ジオメトリです。
- GEOMETRY_POINT　点ジオメトリです。
- LATITUDE　緯度です。

- LONGITUDE　経度です。
- NUTS0　国名です (NUTS レベル 0)。
- NUTS0_CC　国コードです (NUTS レベル 0)。
- NUTS1　地域名です (NUTS レベル 1)。
- NUTS1_CC　地域コードです (NUTS レベル 1)。
- NUTS2　州名です (NUTS レベル 2)。
- NUTS2_CC　州コードです (NUTS レベル 2)。
- NUTS3　地区名です (NUTS レベル 3)。
- NUTS3_CC　地区コードです (NUTS レベル 3)。
- POSTAL_CODE　郵便番号です。
- STATE　州名です。
- STATE_FIPS　FIPS 州コードです。
- STATE_ISO_SUB　米国の ISO 州行政区分コードです。
- USSCITY　米国の都市名です。
- USCITY_FIPS　米国の FIPS 都市コードです。
- USCOUNTY　米国の郡名です。
- USCOUNTY_FIPS　米国の FIPS 国コードです。
- USSTATE　米国の州名です。
- USSTATE_ABBR　米国の州の短縮名です。
- USSTATE_FIPS　米国の FIPS 州コードです。
- ZIP3　米国の 3 桁の郵便番号です。
- ZIP5　米国の 5 桁の郵便番号です。

ミッシング値 (Null 値) - MISSING

セグメントインスタンスは存在するが、セグメント内のフィールドのいずれかにデータが入力されていない場合、そのフィールドは値が欠落した状態になります。いくつかのデータソースタイプではデータの欠落をブランクまたは 0 (ゼロ) として表示しますが、他のデータソースタイプではこのデータの欠落を Null インジケータまたは特別な Null 値として明示的に指定します。Null 値 は「ミッシングデータ」とも呼ばれ、レポートアプリケーションで平均値の計算のような集計関数を実行する場合に重要です。

FOCUS データソースおよびほとんどのリレーショナルデータソースのように、ミッシングデータをサポートするデータソースタイプを使用する場合は、オプションの MISSING 属性を使用してフィールドに Null 値を入力したり、フィールドから Null 値を読み取ったりすることができます。次のような場合に MISSING 属性を使用すると役立ちます。

- **新しいセグメントインスタンスを作成する場合**　マスターファイルまたは DEFINE/COMPUTE 定義で MISSING 属性が ON に設定されたフィールドに値が入力されていない場合、そのフィールドにはミッシング値が割り当てられます。
- **レポートを生成する場合**　Null 値を含むフィールドを取得する場合、そのフィールド値は平均や合計などの集計計算に使用されません。このフィールドの値を呼び出してレポートに表示すると、ミッシング値を示す特殊文字が表示されます。デフォルト文字はピリオド (.) ですが、SET NODATA コマンドを使用するか、HOLD ファイルの場合は SET HNODATA コマンドを使用して、デフォルト文字を任意の文字に変更することができます。詳細は、『TIBCO WebFOCUS アプリケーション作成ガイド』を参照してください。

構文　ミッシング値の指定

```
MISSING = {ON|OFF}
```

説明

ON
: 新しいセグメントインスタンスの作成時およびレポートの実行時に、意図的に入力したブランクまたは 0 (ゼロ) とミッシング値を区別します。

OFF
: 新しいセグメントインスタンスの作成時およびレポートの実行時に、ブランクまたは 0 (ゼロ) とミッシング値を区別しません。デフォルト値は OFF です。

参照

MISSING 属性使用時の注意

MISSING 属性の使用時には次の規則が適用されます。

- **エイリアス** MISSING 属性にはエイリアスはありません。
- **値** ミッシング特性がデータソースの作成時に明示的に設定されているか、デフォルトで設定されているかに関係なく、MISSING 属性の設定は、フィールドに定義済みのミッシング特性に一致させることをお勧めします。たとえば、リレーショナルテーブルフィールドがミッシングデータを受容するように作成されている場合は、このフィールドの MISSING 属性を ON に設定してミッシング値を正しく解釈するように記述します。

  FOCUS データソースでは、MISSING=ON もサポートされます。この設定は、ミッシング値に内部フラグを設定します。
- **変更** MISSING 属性はいつでも変更することができます。なお、MISSING 属性を変更しても、以前の設定で入力して実際に格納されているデータ値には影響を与えません。ただし、データの解釈方法には影響します。MISSING=ON の設定でミッシングデータを入力した後、MISSING=OFF の設定に変更すると、最初にミッシング値として入力したデータはブランク (文字フィールド) または 0 (ゼロ) (数値フィールド) として解釈されます。唯一の例外として、FOCUS データソースでミッシングとして入力された元のデータはそのデータタイプの内部ミッシング値として解釈されます。詳細は、『TIBCO WebFOCUS メタデータリファレンス』および231 ページの 「 FOCUS データソースの記述 」 を参照してください。

ミッシング値の使用

ここでは、次の 4 つのレコードで示されたフィールド値について考察します。

		1	3

フィールド宣言に MISSING 属性を指定せずにミッシング値の平均を計算すると、2 つのブランクレコードに対して 0 (ゼロ) の値が自動的に入力されます。そのため、この 4 つのレコードの平均は (0+0+1+3)/4 で 1 になります。MISSING 属性を ON に設定すると、この 2 つのブランクレコードが計算に使用されないため、平均は (1+3)/2 で 2 になります。

ユニークセグメントのミッシング値の場合でも、MISSING 属性の設定に基づいて 0 (ゼロ)、ブランク、ミッシング値のいずれかが自動的に入力されます。ユニークセグメントのミッシング値が他の値と異なる点は、ミッシング値が格納されないことです。個数または平均値を計算するユニークセグメントのフィールドには MISSING 属性を指定する必要はありません。

Null 値 (ミッシング値とも呼ばれる) の使用についての詳細は、『TIBCO WebFOCUS Language リファレンス』を参照してください。このマニュアルでは、WHERE 句に MISSING 選択演算子を使用したり、DEFINE FILE コマンドに SOME または ALL 句を使用して一時項目 (DEFINE) を作成したりして、レポートでこれらの値を区別する別の方法について説明しています。

FML 階層の記述

FML (Financial Modeling Language) では、階層データ構造に対する動的なレポート作成がサポートされます。

FML を使用して、マスターファイルのフィールド間に階層関係を定義し、これらのフィールドを自動的に表示することができます。また、指定した階層フィールドの値の代わりに、説明キャプションをレポートに表示することもできます。

2 つのフィールド間に階層関係を定義するには、マスターファイルで PROPERTY=PARENT_OF および REFERENCE=hierarchyfld 属性を指定します。

親フィールドと子フィールドに同一の FORMAT または USAGE が指定され、その親子関係が階層型である必要があります。親フィールドおよび子フィールドの両方のフォーマットは、数値、文字のいずれかに設定する必要があります。

構文　マスターファイルでのフィールド間の階層の指定

```
FIELD=parentfield,...,PROPERTY=PARENT_OF, REFERENCE=[seg.]hierarchyfld,$
```

説明

`parentfield`

階層の親フィールドです。

`PROPERTY=PARENT_OF`

このフィールドを、階層内で参照されるフィールドの親として指定します。

これらの属性は、フィールドごとに指定することができます。そのため、1 つのマスターファイルに複数の階層を定義することができます。ただし、各フィールドに指定できる親はそれぞれ 1 つです。同一階層フィールドの CAPTION 属性が複数のフィールドで指定されている場合は、その階層構造を上から下、左から右に検索して最初に見つかった親がこの場合の親として使用されます。

`seg`

階層フィールドのセグメントの位置です。hierarchyfield フィールドが複数のセグメントに存在する場合に必要です。

hierarchyfld

階層の子フィールドです。

PARENT_OF は、マスターファイルの一時項目 (DEFINE) にも使用することができます。

```
DEFINE name/fmt=expression;,PROPERTY=PARENT_OF,REFERENCE=hierarchyfld,$
```

構文　階層フィールド値への説明キャプションの割り当て

マスターファイルで階層フィールドにキャプションを割り当てるには、次の属性を使用します。

```
FIELD=captionfield,..., PROPERTY=CAPTION, REFERENCE=[seg.]hierarchyfld,$
```

説明

captionfield

階層フィールドの説明キャプションを追加するフィールド名です。たとえば、従業員 ID が階層フィールドの場合、従業員 ID の代わりに従業員の名前 (姓) を説明テキストとしてレポートに表示することができます。

PROPERTY=CAPTION

このフィールドに、階層フィールド値の代わりにレポートに表示する説明キャプションが指定されていることを示します。

キャプションはフィールドごとに指定することができますが、各フィールドに追加できるキャプションは 1 つです。同一階層フィールドの CAPTION 属性が複数のフィールドで指定されている場合は、その階層構造を上から下、左から右に検索して最初に見つかった親がキャプションとして使用されます。

seg

階層フィールドのセグメントの位置です。hierarchyfield フィールドが複数のセグメントに存在する場合に必要です。

hierarchyfld

階層フィールドです。

CAPTION は、マスターファイルの一時項目 (DEFINE) にも使用することができます。

```
DEFINE name/format=expression;,PROPERTY=CAPTION,REFERENCE=hierarchyfld,$
```

例　マスターファイルでの階層の定義

CENTGL マスターファイルには、勘定科目表の階層が格納されています。GL_ACCOUNT_PARENT は、階層の親フィールドです。GL_ACCOUNT フィールドは、階層フィールドです。次の GL_ACCOUNT_CAPTION フィールドは、階層フィールドの説明キャプションとして使用することができます。

```
FILE=CENTGL     ,SUFFIX=FOC
SEGNAME=ACCOUNTS,SEGTYPE=S01
FIELDNAME=GL_ACCOUNT,           ALIAS=GLACCT,  FORMAT=A7,
          TITLE='Ledger,Account', FIELDTYPE=I, $
FIELDNAME=GL_ACCOUNT_PARENT,    ALIAS=GLPAR,   FORMAT=A7,
          TITLE=Parent,
          PROPERTY=PARENT_OF, REFERENCE=GL_ACCOUNT, $
FIELDNAME=GL_ACCOUNT_TYPE,      ALIAS=GLTYPE,  FORMAT=A1,
          TITLE=Type,$
FIELDNAME=GL_ROLLUP_OP,         ALIAS=GLROLL,  FORMAT=A1,
          TITLE=Op, $
FIELDNAME=GL_ACCOUNT_LEVEL,     ALIAS=GLLEVEL, FORMAT=I3,
          TITLE=Lev, $
FIELDNAME=GL_ACCOUNT_CAPTION,   ALIAS=GLCAP,   FORMAT=A30,
          TITLE=Caption,
          PROPERTY=CAPTION,   REFERENCE=GL_ACCOUNT, $
FIELDNAME=SYS_ACCOUNT,          ALIAS=ALINE,   FORMAT=A6,
          TITLE='System,Account,Line', MISSING=ON, $
```

ディメンションの定義 - WITHIN

OLAP モデルは、フィールド名の WITHIN 属性を使用して、マスターファイルにディメンションを事前に定義することによりデータ構造を整理します。ディメンションは、「要素」と呼ばれる互いに関係するフィールドのグループまたはリストです。

WITHIN 属性を使用して、階層ディメンションでドリルアップおよびドリルダウン機能を有効にすることができます。OLAP を有効にしたフィールドを選択し、ドリルダウンしてディメンション階層の他のレベルを表示するようにレポートを操作することができます。

たとえば、製品の販売地域の階層をマスターファイルで GEOGRAPHY ディメンションとして定義し、このディメンションに 「Region」(地域)、「State」(州)、「City」(都市) というフィールド (要素) を降順に指定することができます。階層の最上位要素である Region には、GEOGRAPHY ディメンション内のすべての地域のリストが含まれます。階層の次の上位要素である State には、各地域で有効な州名のリストが含まれます。ディメンションは、サポートされている任意のデータソースのマスターファイルで定義することができます。

OLAP を有効にしたデータソースで複数のディメンション階層を組み合わせたマトリックスは、「マルチディメンション」と呼ばれます。たとえば、製品を州内で販売する場合でも、製品を州と同一のディメンションでグループ化する必要はありません。代わりに、製品カテゴリや製品名という要素は「PRODUCT」と呼ばれるディメンションでグループ化します。州という要素は、地域や都市を含めることもできる GEOGRAPHY ディメンションのメンバーになります。これらのディメンションは、マトリックスで統合されます。たとえば Northeast 地域のコーヒーの売上のように、条件が交差する点で特定の値が取得されます。

ACCEPT 属性を使用して、各ディメンション要素 (フィールド) の許容値のリストを指定することができます。この指定には、マスターファイル、参照ファイルのいずれかでハードコードしたリストを使用します。ACCEPT 属性についての詳細は、184 ページの 「 データの確認 - ACCEPT 」 を参照してください。

構文　ディメンションの定義

```
WITHIN='*dimensionname'
WITHIN=field
```

説明

`'*dimensionname'`

ディメンションの名前です。ディメンションは、階層の最上位に位置するフィールドのフィールド宣言で定義します。名前の先頭にアスタリスク (*) を追加し、名前を一重引用符 (') で囲む必要があります。名前の先頭には文字を使用する必要がありますが、それ以外は、文字、数字、アンダースコア (_)、ピリオド (.) を任意に組み合わせて使用することができます。特殊文字および埋め込みブランクは使用しないでください。

`field`

指定したディメンションに含める追加要素で階層関係を定義するために使用します。階層の最上位の位置でディメンション名を定義した後、各要素 (フィールド) に WITHIN 属性を使用して階層内で直接上位に位置するフィールドに結合します。WITHIN 属性は、フィールド名、エイリアスのいずれかでフィールドを参照することができます。なお、フィールドは 1 つのディメンションにのみ所属することができ、2 つのフィールドが上位の同一フィールドを参照することはできません。

例　ディメンションの定義

次の例は、OSALES マスターファイルで PRODUCT ディメンションを定義する方法を示しています。

```
PRODUCT Dimension
           ===> Product Category
               ===> Product Name
FILENAME=OSALES,  SUFFIX=FOC
  SEGNAME=SALES01,  SEGTYPE=S1
     FIELD=PRODCAT,     ALIAS=PCAT,      FORMAT=A11,
       WITHIN='*PRODUCT',$
     FIELD=PRODNAME,    ALIAS=PNAME,     FORMAT=A16,
       WITHIN=PRODCAT,$
```

例　マルチディメンションの定義

次の例は、OSALES マスターファイルで PRODUCT、GEOGRAPHY、TIME ディメンションを定義する方法を示しています。この例には注釈が付いています。

```
PRODUCT Dimension
           ===> Product Category
               ===> Product Name
GEOGRAPHY Dimension
           ===> Region
               ===> State
                   ===> City
                         ===> Store Name (from the OSTORES Master File)
TIME Dimension
           ===> Year
               ===> Quarter
                   ===> Month
                       ===> Date
```

OSALES マスターファイル

```
FILENAME=OSALES, SUFFIX=FOC
 SEGNAME=SALES01, SEGTYPE=S1
 FIELD=SEQ_NO,    ALIAS=SEQ,    FORMAT=I5,    TITLE='Sequence#',
  DESC='Sequence number in database',$
 FIELD=PRODCAT,   ALIAS=PCAT,  FORMAT=A11,INDEX=I, TITLE='Category',
  DESC='Product category',
  ACCEPT='Coffee' OR 'Food' OR 'Gifts',
1.    WITHIN='*PRODUCT',$
 FIELD=PRODCODE, ALIAS=PCODE, FORMAT=A4, INDEX=I, TITLE='Product ID',
  DESC='Product Identification code (for sale)',$
 FIELD=PRODNAME, ALIAS=PNAME, FORMAT=A16,         TITLE='Product',
  DESC='Product name',
2.  ACCEPT='Espresso' OR 'Latte' OR 'Cappuccino' OR 'Scone' OR
        'Biscotti' OR 'Croissant' OR 'Mug' OR 'Thermos' OR
        'Coffee Grinder' OR 'Coffee Pot',
    WITHIN=PRODCAT,$
 FIELD=REGION,   ALIAS=REG,   FORMAT=A11, INDEX=I,TITLE='Region',
  DESC='Region code',
  ACCEPT='Midwest' OR 'Northeast' OR 'Southwest' OR 'West',
3.    WITHIN='*GEOGRAPHY',$
 FIELD=STATE,    ALIAS=ST,    FORMAT=A2, INDEX=I,TITLE='State',
  DESC='State',
4.  ACCEPT=(OSTATE),
    WITHIN=REGION,$
 FIELD=CITY,      ALIAS=CTY,    FORMAT=A20,        TITLE='City',
  DESC='City',
    WITHIN=STATE,$
 FIELD=STORE_CODE, ALIAS=STCD,  FORMAT=A5, INDEX=I, TITLE='Store ID',
  DESC='Store identification code (for sale)',$
 FIELD=DATE,       ALIAS=DT,    FORMAT=I8YYMD,      TITLE='Date',
  DESC='Date of sales report',
    WITHIN=MO,$
5. FIELD=UNITS,     ALIAS=UN,    FORMAT=I8,     TITLE='Unit Sales',
  DESC='Number of units sold',$
 FIELD=DOLLARS,    ALIAS=DOL,   FORMAT=I8,     TITLE='Dollar Sales',
  DESC='Total dollar amount of reported sales',$
 FIELD=BUDUNITS,   ALIAS=BUNIITS, FORMAT=I8,   TITLE='Budget Units',
  DESC='Number of units budgeted',$
 FIELD=BUDDOLLARS, ALIAS=BDOLLARS,FORMAT=I8,   TITLE='Budget Dollars',
  DESC='Total sales quota in dollars',$
```

```
6. DEFINE ADATE/A8 = EDIT(DATE);$
   DEFINE YR/I4 = EDIT (EDIT(ADATE,'9999$$$$')); WITHIN='*TIME',$
   DEFINE MO/I2 = EDIT (EDIT(ADATE,'$$$$99$$')); WITHIN=QTR,$
   DEFINE QTR/I1 = IF MO GE 1 AND MO LE 3
     THEN 1 ELSE IF MO GE 4 AND MO LE 6
     THEN 2 ELSE IF MO GE 7 AND MO LE 9
     THEN 3 ELSE IF MO GE 10 AND MO LE 12
     THEN 4 ELSE 0;
     WITHIN=YR,$
7. SEGNAME = STORES01, SEGTYPE = KU, PARENT = SALES01,
     CRFILE = OSTORES, CRKEY = STORE_CODE, $
```

1. PRODUCT ディメンションを宣言します。名前の先頭にアスタリスク (*) を追加し、名前を一重引用符 (') で囲む必要があります。
2. 必要に応じて、ディメンション要素 (フィールド) ごとに許容値のリストを定義することもできます。マスターファイル、外部 Flat File のいずれかでハードコードされたリストを使用して ACCEPT 属性を指定します。許容値のリストは、選択可能な条件値として表示されます。この例では、製品名の値に Espresso、Latte、Cappuccino、Scone、Biscotti、Croissant、Mug、Thermos、Coffee Grinder、Coffee Pot を指定する必要があります。ACCEPT 属性についての詳細は、187 ページの 「 ディメンション許容値の指定 」 を参照してください。
3. GEOGRAPHY ディメンションを宣言し、GEOGRAPHY のディメンション階層を定義します。GEOGRAPHY 内に Region (階層の最上位)、Region 内に State、State 内に City をそれぞれ定義します。
4. この例では、ACCEPT 属性は外部 Flat File (OSTATE) を使用して、State (ST フィールド) に使用可能なすべてのデータ値を指定しています。
5. UNITS、DOLLARS、BUDUNITS、BUDDOLLARS の 4 つのフィールドは、メジャーフィールドの例です。通常、メジャーフィールドは数量を定義する分析に使用します。たとえば、Units、Dollars、Budget Units、Budget Dollars は、販売ユニット数、ドル建て報告総販売額、販売予定ユニット数、ドル建て総販売目標額をそれぞれ指定するメジャーです。

6. 次の項目を表示します。

   一時項目 (DEFINE) は、ディメンションの任意レベルに含めることができます。この例では、TIME ディメンション内に YR、MO、QTR フィールドを 定義しています。一時項目 (DEFINE) に使用する WITHIN 属性は、式を終了するセミコロン (;) と同一行に配置する必要があります。

   TIME ディメンションのディメンション階層の定義では、Time 内に Year、Year 内に Quarter、Quarter 内に Month、Month 内に Date を定義します。

   階層内のフィールドは、マスターファイルで任意の順序に配置することができます。

7. ディメンションには、修飾名を使用して動的または静的 JOIN 構造を含めることができます。この例では、OSALES データソースは、STORE_CODE を共通フィールドとする、OSTORES データソースの静的クロスリファレンスデータソースです。OLAP アプリケーションは、この結合を通して OSTORES データソースから店舗名の値を取得することができます。なお、OSTORES マスターファイルの STORE_NAME は、OSALES マスターファイルで定義された GEOGRAPHY ディメンションの要素です。OSTORES マスターファイルは次のとおりです。

```
FILENAME=OSTORES, SUFFIX=FOC
SEGNAME=STORES01, SEGTYPE=S1
 FIELD=STORE_CODE, ALIAS=STCD,  FORMAT=A5, INDEX=I,TITLE='Store ID',
   DESC='Franchisee ID Code',$
 FIELD=STORE_NAME, ALIAS=SNAME, FORMAT=A23,  TITLE='Store Name',
   DESC='Store Name', WITHIN=SALES01.CITY,$
 FIELD=ADDRESS1,   ALIAS=ADDR1, FORMAT=A19,  TITLE='Contact',
   DESC='Franchisee Owner',$
 FIELD=ADDRESS2,   ALIAS=ADDR2, FORMAT=A31,  TITLE='Address',
   DESC='Street Address',$
 FIELD=CITY,       ALIAS=CTY,   FORMAT=A22,  TITLE='City',
   DESC='City',$
 FIELD=STATE,      ALIAS=ST,    FORMAT=A2,   TITLE='State',
   DESC='State',$
 FIELD=ZIP,        ALIAS=ZIP,   FORMAT=A6,   TITLE='Zip Code',
   DESC='Postal Code',$
```


- 一時項目 (DEFINE) は、ディメンションの任意レベルに含めることができます。この例では、TIME ディメンション内に YR、MO、QTR フィールドを 定義しています。一時項目 (DEFINE) に使用する WITHIN 属性は、式を終了するセミコロン (;) と同一行に配置する必要があります。
- TIME ディメンションのディメンション階層の定義では、Time 内に Year、Year 内に Quarter、Quarter 内に Month、Month 内に Date を定義します。
- 階層内のフィールドは、マスターファイルで任意の順序に配置することができます。

データの確認 - ACCEPT

オプションとして指定する ACCEPT 属性を使用して、パラメータプロンプト画面に入力されたデータの有効性を確認することができます。

注意：Suffix および FIX データソースは、複数の RECTYPE 値を特定するために ACCEPT 属性を使用する場合があります。

ACCEPT 属性では、次のタイプのオプションがサポートされます。

- ACCEPT = *value1* OR *value2* ...

  このオプションは、受容可能な 1 つ以上の値を指定する場合に使用します。

- ACCEPT = *value1* TO *value2*

  このオプションは、受容可能な値の範囲を指定する場合に使用します。

- ACCEPT = FIND

  このオプションは、別のデータソースで保守管理を行う際に、FOCUS データソースの値と入力トランザクションデータを比較検証する場合に使用します。FIND は、FOCUS データソースでのみサポートされ、OLAP を有効にしたシノニムには適用されません。

- ACCEPT = DECODE

  このオプションは、オートプロンプト用の値の組み合わせを提供する場合に使用します。値の組み合わせは、データソースでの検索用の値と、それに対応する表示用の値で構成されます。

- ACCEPT = FOCEXEC

  このオプションは、プロシジャを実行して参照フィールド値および表示フィールド値を取得する場合に使用します。出力の各行には、参照フィールドの値が 1 つと、それに対応する表示フィールドの値が含まれている必要があります。これらの値は、行の任意の位置に任意の順序で存在することができます。このプロシジャで、その他のフィールド値を返すこともできます。

- ACCEPT = SYNONYM

  このオプションは、別のデータソースで値を検索し、それに対応する表示値を取得する場合に使用します。両方のデータソースに参照フィールド値が存在する必要がありますが、これらのフィールド名が一致している必要はありません。このオプションでは、シノニム名、参照フィールド名、表示フィールド名を指定します。

構文　データの確認

```
ACCEPT = list

ACCEPT = value1 TO value2

ACCEPT = FIND (field [AS name] IN file)

ACCEPT=SYNONYM(lookup_field AS display_field IN lookup_synonym)

ACCEPT=FOCEXEC(lookup_field AS display_field IN lookup_focexec)
```

説明

`list`
許容値のリストです。構文は次のとおりです。

```
value1 OR value2 OR value3...
```

たとえば、ACCEPT = RED OR WHITE OR BLUE のように記述します。項目の区切り文字としてブランクを使用することもできます。許容値のリストが 1 行に収まらない場合は、そのまま次の行に続けます。リストはカンマ (,) で終了します。

`value1 TO value2`
許容値の範囲を設定します。たとえば、ACCEPT = 150 TO 1000 のように記述します。

`FIND`
他のインデックスフィールドの値に対して、入力データを確認します。詳細は、231 ページの 「 FOCUS データソースの記述 」 を参照してください。

`SYNONYM`
別のデータソースで値を検索し、その値に対応する表示値を取得します。両方のデータソースに参照フィールド値が存在する必要がありますが、これらのフィールド名が一致している必要はありません。このオプションでは、シノニム名、参照フィールド名、表示フィールド名を指定します。

プロシジャ
プロシジャを実行して、参照フィールド値および表示フィールド値を取得します。各行には、参照フィールドの値が 1 つと、それに対応する表示フィールドの値が、行の任意の位置に任意の順序で含まれている必要があります。このオプションでは、プロシジャ名、参照フィールド名、表示フィールド名を指定します。

`lookup_field`
: lookup_synonym で指定するシノニムのフィールド、または lookup_focexec で指定するプロシジャから返されるフィールドです。このフィールド値が、[選択条件 (WHERE) フィルタの演算] ダイアログボックスまたはオートプロンプト機能で使用され、ACCEPT 属性が指定されたフィールドと比較されます。

`display_field`
: lookup_synonym で指定するシノニムのフィールド、または lookup_focexec で指定するプロシジャから返されるフィールドです。このフィールド値が、[選択条件 (WHERE) フィルタの演算] ダイアログボックスまたはオートプロンプトドロップダウンリストの選択項目として表示されます。

`lookup_synonym`
: 参照データを記述したシノニムの名前です。

`lookup_focexec`
: 参照フィールド値および表示フィールド値を任意の順序で返すプロシジャの名前です。このプロシジャは、その他のフィールド値を返すプロシジャにすることもできます。

ACCEPT 属性に埋め込みブランクが含まれる場合 (例、Great Britain)、その値は一重引用符 (') で囲む必要があります。

フィールド宣言で ACCEPT 属性を指定し、SET コマンドパラメータである ACCBLN の値が OFF の場合、ブランクおよび 0 (ゼロ) の値は ACCEPT 属性に明示的にコーディングされている場合にのみ使用することができます。SET ACCBLN についての詳細は、『TIBCO WebFOCUS アプリケーション作成ガイド』を参照してください。

例　埋め込みブランクを含むリストの指定

```
ACCEPT = SPAIN OR ITALY OR FRANCE OR 'GREAT BRITAIN'
```

参照　ACCEPT 属性使用時の注意

ACCEPT 属性の使用時には次の規則が適用されます。

- **エイリアス**　ACCEPT 属性にはエイリアスはありません。
- **変更**　ACCEPT 属性の情報はいつでも変更することができます。
- **一時項目 (DEFINE)**　ACCEPT 属性を使用して、DEFINE 属性で作成した一時項目 (DEFINE) を確認することはできません。
- **HOLD ファイル**　HOLD ファイルのマスターファイルに ACCEPT 属性を継承するには、SET HOLDATTR コマンドを使用します。HOLD ファイルについての詳細は、『TIBCO WebFOCUS Language リファレンス』を参照してください。

- 複数のプロシジャで共用する確認リストを 1 つ作成しておくと役立ちます。FIND 関数は、値のリストが大規模な場合や変更頻度が高い場合に役立ちます。

ディメンション許容値の指定

オプションの ACCEPT 属性を使用して、OLAP を有効にしたマスターファイルで使用するための許容値リストを指定することができます。

ACCEPT 属性は、データフィールドを複数のレポートで参照する場合に役立ちます。マスターファイルに ACCEPT 属性を 1 つ指定しておくと、プロシジャごとに許容値リストを提供する必要がなくなります。

ACCEPT 属性により指定が可能なものは次のとおりです。

- 許容データ値のリスト。
- 許容データ値の範囲。
- 外部 FLAT (参照) データソースに含まれる任意の値と一致する許容データ値。

参照オプションの使用は、値のリストが大きい場合や変更頻度が高い場合に役立ちます。

構文　ディメンションの許容値リストの指定

```
ACCEPT=value1 OR value2 OR value3...
```

説明

`value1, value2, value3...`

許容値のリストです。埋め込みブランクを含む値が ACCEPT リストに存在する場合は、その値を一重引用符 (') で囲む必要があります。埋め込みブランクを含まない値には、オプションとして一重引用符 (') を使用します。

例　ディメンションの許容製品リストの指定

次の例は、マスターファイルで指定する ACCEPT 構文を示しています。この構文で指定する許容製品リストには、Espresso、Latte、Cappuccino、Scone、Biscotti、Croissant、Mug、Thermos、Coffee Grinder、Coffee Pot が含まれます。

```
ACCEPT='Espresso' OR 'Latte' OR 'Cappuccino' OR 'Scone' OR 'Biscotti' OR
   'Croissant' OR 'Mug' OR 'Thermos' OR 'Coffee Grinder' OR 'Coffee Pot'
```

構文　ディメンションの許容値範囲の指定

```
ACCEPT=value1 TO value2
```

説明

`value1`

範囲内の先頭の値です。

`value2`

範囲内の末尾の値です。

例　ディメンションに使用する許容値範囲の指定

次の例は、マスターファイルで 150 から 1000 までの許容データ値の範囲を指定する ACCEPT 構文を示しています。

```
ACCEPT=150 TO 1000
```

構文　外部フラットデータソースの値に対するデータ確認

```
ACCEPT=(ddname)
```

説明

`ddname`

許容値リストの値が格納された既存のデータソースの完全修飾名を示す ddname です。この ddname の最大長は 8 バイトです。なお、FILEDEF または ALLOCATE ステートメントを発行して、外部データソースへの ddname を割り当てる必要があります。任意の有効なプロファイルで FILEDEF または ALLOCATE ステートメントを発行することができます。

例　State フィールドの許容値に対するデータ確認

OSTATE データソースには、State フィールドで許容されるすべての値のリストが格納されています。外部 OSTATE データソースに対して State 値を確認するマスターファイルの ACCEPT 構文は次のとおりです。

```
ACCEPT=(OSTATE)
```

代替レポートフィールドタイトル - TITLE

レポートを作成すると、デフォルト設定でレポートの各フィールドタイトルにはフィールド名が表示されます。フィールドの TITLE 属性オプションを指定して、デフォルトのフィールドタイトルを変更することができます。

リクエストで AS 句を使用ことにより、個別リクエストのフィールドタイトルに異なるフィールドタイトルを指定することもできます。詳細は、『TIBCO WebFOCUS Language リファレンス』を参照してください。

なお、AVE. などの演算接頭語をフィールドに使用した場合、TITLE 属性の指定はレポートに反映されません。演算接頭語をフィールドに使用した場合は、AS 句を使用して代替フィールドタイトルを指定することができます。

マスターファイルは TITLE 属性を多言語でサポートします。詳細は、191 ページの 「 多言語メタデータ 」 を参照してください。

構文　代替タイトルの指定

```
TITLE = 'text'
```

説明

`text`
1 バイト文字セットの場合、最大値は 512 文字です。Unicode 環境の場合、この長さは各文字が表すバイト数による影響を受けます。詳細は、『TIBCO WebFOCUS サーバ管理者ガイド』の「Unicode サポート」を参照してください。テキストをカンマ (,) で区切ることにより、最大で 5 つのタイトル行に分割することができます。フィールドタイトルの末尾にブランクを含める場合は、そのブランクの位置にスラッシュ (/) を配置します。カンマ (,) または先頭にブランクを含む文字列は、一重引用符 (') で囲む必要があります。

例　デフォルトフィールドタイトルの置換

たとえば、次のように FIELD 宣言を記述します。

```
FIELD = LNAME, ALIAS = LN, USAGE = A15, TITLE = 'Client,Name',$
```

これにより、デフォルトフィールドタイトルの LNAME が次のタイトルに置換されます。

```
Client
Name
------
```

参照　TITLE 属性使用時の注意

TITLE 属性の使用時には次の規則が適用されます。

- **エイリアス**　TITLE 属性にはエイリアスはありません。

- **変更** TITLE 属性の情報はいつでも変更することができます。TITLE 属性を上書きするには、リクエストに AS 句を指定するか、SET TITLES=OFF コマンドを使用して TITLE 属性をオフにします。
- **一時項目 (DEFINE)** DEFINE 属性で作成した一時項目 (DEFINE) に TITLE 属性を使用する場合は、DEFINE 式を終了するセミコロン (;) を TITLE キーワードと同一行に配置する必要があります。
- **HOLD ファイル** HOLD ファイルのマスターファイル内に TITLE 属性を継承するには、SET HOLDATTR コマンドを使用します。HOLD ファイルについての詳細は、『TIBCO WebFOCUS Language リファレンス』を参照してください。

フィールドの説明 - DESCRIPTION

オプションの DESCRIPTION 属性を使用して、マスターファイル内のフィールドにコメントおよび他の説明を追加することができます。追加するコメントは、最大で 2 キロバイト (2048 バイト) です。

なお、フィールド宣言、セグメント宣言、ファイル宣言にもコメントを追加することできます。その場合は、フィールドにコメントを入力し、終了を表すドル記号 ($) を配置します。また、コメント専用の行を作成するには、宣言の後に新しい行を挿入し、その行の先頭にドル記号 ($) を配置します。マスターファイルの作成に使用する構文および規則についての詳細は、15 ページの 「 データソース記述の理解 」 を参照してください。

FOCUS データソースに使用する DESCRIPTION 属性は、データソースを再構築せずにいつでも変更することができます。

マスターファイルでは、多言語の DESCRIPTION 属性がサポートされます。詳細は、191 ページの 「 多言語メタデータ 」 を参照してください。

構文 フィールド説明の指定

```
DESC[RIPTION] = text
```

説明

`DESCRIPTION`

DESC に短縮することもできます。キーワードを短縮しても、機能面での影響はありません。

`text`

最大で 2 キロバイト (2048 バイト) の任意のテキストです。テキストにカンマ (,) が含まれる場合は、テキスト全体を一重引用符 (') で囲む必要があります。

例　フィールド説明の指定

次のように FIELD 宣言を記述して DESCRIPTION 属性を指定します。

```
FIELD=UNITS,ALIAS=QTY,USAGE=I6, DESC='QUANTITY SOLD, NOT RETURNED',$
```

参照　DESCRIPTION 属性使用時の注意

DESCRIPTION 属性の使用時には次の規則が適用されます。

- **エイリアス**　DESCRIPTION 属性のエイリアスは DEFINITION です。
- **変更**　DESCRIPTION 属性はいつでも変更することができます。
- **一時項目 (DEFINE)**　DEFINE 属性で作成した一時項目 (DEFINE) に DESCRIPTION 属性を使用することができます。

多言語メタデータ

マスターファイルは、多言語のフィールドタイトルをサポートします。

使用されるタイトルまたは説明は、LANG パラメータの値およびマスターファイルで指定された TITLE_*ln* または DESC_*ln* 属性の有無、あるいはマスターファイルに対応する翻訳ファイルセットの有無により異なります。ここで、*ln* はフィールドタイトルまたは説明に適用される言語を識別します。

マスターファイルでは、フィールドタイトルは次の項目から取得されます。

1. AS 句を使用してレポートリクエストに指定されたタイトル
2. マスターファイルの TITLE 属性 (リクエストに AS 句が指定されておらず、SET TITLES=ON の場合)
3. マスターファイルで指定したフィールド名 (AS 句と TITLE 属性のいずれも指定されておらず、SET TITLES=OFF の場合)

構文　言語使用の有効化

サポートされるプロファイル、プロシジャで次の構文を発行します。

```
SET LANG = lng
```

または

```
SET LANG = ln
```

説明

`lng`
言語の 3 文字の略名です。

`ln`
2 文字の ISO 言語コードです。

注意：SET LANG がプロシジャ内に使用されている場合、この値は nlscfg.err ファイルまたはすべてのプロファイルに設定されている値を上書きします。

参照 NLS 構成ファイルでの言語の有効化

nlscfg.err 構成ファイルで、次のコマンドを発行します。

```
LANG = lng
```

参照 言語および言語コードの略名

言語名	2 文字の言語コード	3 文字の言語の略名
アラビア語	ar	ARB
バルト言語	lt	BAL
中国語 (簡体字)	zh	PRC
中国語 (繁体字)	tw	ROC
チェコ語	cs	CZE
デンマーク語	da	DAN
オランダ語	nl	DUT
英語 (米国)	en	AME または ENG
英語 (英国)	uk	UKE
フィンランド語	fi	FIN

言語名	2 文字の言語コード	3 文字の言語の略名
フランス語 (カナダ)	fc	FRE
フランス語 (フランス)	fr	FRE
ドイツ語 (オーストリア)	at	GER
ドイツ語 (ドイツ)	de	GER
ギリシャ語	el	GRE
ヘブライ語	iw	HEW
イタリア語	it	ITA
日本語 (ASCII: Shift-JIS(cp942)	ja	JPN
日本語 (UNIX では、ASCII: EUC(cp10942))	je	JPE
韓国語	ko	KOR
ノルウェー語	no	NOR
ポーランド語	pl	POL
ポルトガル語 (ブラジル)	br	POR
ポルトガル語 (ポルトガル)	pt	POR
ロシア語	ru	RUS
スペイン語	es	SPA
スウェーデン語	sv	SWE
タイ語	th	THA
トルコ語	tr	TUR

マスターファイルへの多言語メタデータの直接指定

マスターファイルに TITLE_ln および DESCRIPTION_ln 属性を直接指定することができます。ここで、ln は言語コードを表します。

注意：言語翻訳ファイルセットを作成しておき、マスターファイルのファイルレベルで trans_file 属性を追加することもできます。この方法についての詳細は、48 ページの 「 メタデータのローカライズと翻訳ファイルの使用 」 を参照してください。

構文　マスターファイルでの多言語メタデータの指定

```
FIELDNAME = field, ...
   .
   .
   .
TITLE= default_column_heading, TITLE_ln = column_heading_for_ln,
   .
   .
   .
DESC= default_desc, DESC_ln = desc_for_ln,
   .
   .
   .
```

説明

`field`
マスターファイル内のフィールドです。

`default_column_heading`
SET TITLES=ON が設定され、かつ LANG パラメータがサーバのデフォルトに設定されている、または別の言語に設定されているが、マスターファイルのフィールドに対応する TITLE_*ln* 属性が指定されていない場合に使用するフィールド見出しです。このフィールドタイトルは、ln 値が無効な場合にも使用されます。

`default_desc`
この説明テキストは、LANG パラメータがサーバのデフォルトの言語に設定されている場合、またはその他の言語に設定されているがマスターファイルのフィールドに対応する DESC_ln 属性がない場合に使用されます。この説明は、ln 値が無効な場合にも使用されます。

`TITLE_ln = column_heading_for_ln`

フィールドタイトルに使用する言語、およびその言語でのフィールドタイトルのテキストを指定します。このフィールドタイトルは、SET TITLES=ON の設定で、LANG パラメータがサーバのデフォルト以外の言語に設定され、マスターファイルに対応する TITLE_ln 属性がある場合に使用されます。ここで、ln は LANG パラメータで指定された 2 バイトの ISO 639 言語コードの略名です。詳細は、192 ページの「 言語および言語コードの略名 」を参照してください。

`DESC_ln = desc_for_ln`

説明に使用する言語、およびその言語での説明テキストを指定します。この説明は、LANG パラメータがサーバのデフォルト言語以外の言語に設定され、マスターファイルに対応する DESC_ln 属性がある場合に使用されます。*ln* の有効な値は、2 バイトの ISO 639 言語コードの略名です。

参照　多言語メタデータ使用時の注意

- 文字を正しく生成するには、使用するすべての言語が、サーバの起動時に指定されたコードページと一致する必要があります。コードページを変更するには、一度サーバを停止し、新しいコードページで再起動する必要があります。
- マスターファイルは、サーバのコードページで格納しておく必要があります。
- 多言語の説明は、一時項目 (DEFINE および COMPUTE) を含むマスターファイルに記述されたすべてのフィールドでサポートされます。
- SET HOLDATTR=ON を指定して HOLD コマンドを 発行した場合、1 つの TITLE 属性のみが HOLD マスターファイルに継承されます。この値は、レポートの出力に表示されるフィールドタイトルです。

例　マスターファイルでの多言語説明の使用

次の例では、CENTINV データソースのマスターファイルで、PROD_NUM および PRODNAME フィールドの説明にフランス語 (DESC_FR)、スペイン語 (DESC_ES)、デフォルト言語 (DESC) が指定されています。

```
FILE=CENTINV, SUFFIX=FOC, FDFC=19, FYRT=00
 SEGNAME=INVINFO, SEGTYPE=S1, $
  FIELD=PROD_NUM, ALIAS=PNUM, FORMAT=A4, INDEX=I,
   DESCRIPTION='Product Number'
   DESC='Product Number',
   DESC_ES='Numero de Producto',
   DESC_FR='Nombre de Produit', $
  FIELD=PRODNAME, ALIAS=PNAME, FORMAT=A30,
   WITHIN=PRODCAT,
   DESCRIPTION='Product Name'
   DESC_FR='Nom de Produit',
   DESC_ES='Nombre de Producto', $
  FIELD=QTY_IN_STOCK, ALIAS=QIS, FORMAT=I7,
   DESCRIPTION='Quantity In Stock', $
  FIELD=PRICE, ALIAS=RETAIL, FORMAT=D10.2,
   TITLE='Price:',
   DESCRIPTION=Price, $
```

例　リクエストでの多言語タイトルの使用

次の例では、CENTINV データソースのマスターファイルで、PROD_NUM および PRODNAME フィールドのタイトルにフランス語 (TITLE_FR)、スペイン語 (TITLE_ES)、デフォルト言語 (TITLE) が指定されています。

```
FILE=CENTINV, SUFFIX=FOC, FDFC=19, FYRT=00
 SEGNAME=INVINFO, SEGTYPE=S1, $
  FIELD=PROD_NUM, ALIAS=PNUM, FORMAT=A4, INDEX=I,
   TITLE='Product,Number:',
   TITLE_FR='Nombre,de Produit:',
   TITLE_ES='Numero,de Producto:',
   DESCRIPTION='Product Number', $
  FIELD=PRODNAME, ALIAS=PNAME, FORMAT=A30,
   WITHIN=PRODCAT,
   TITLE='Product,Name:',
   TITLE_FR='Nom,de Produit:',
   TITLE_ES='Nombre,de Producto:'
   DESCRIPTION='Product Name', $
  FIELD=QTY_IN_STOCK, ALIAS=QIS, FORMAT=I7,
   TITLE='Quantity,In Stock:',
   DESCRIPTION='Quantity In Stock', $
  FIELD=PRICE, ALIAS=RETAIL, FORMAT=D10.2,
   TITLE='Price:',
   DESCRIPTION=Price, $
```

サーバコードページのデフォルト言語は英語です。デフォルトの状態では、SET TITLES=ON に設定されています。そのため、次のリクエストは TITLE 属性を使用して、すべてのフィールドタイトルを英語で作成します。

```
TABLE FILE CENTINV
PRINT PROD_NUM PRODNAME PRICE
WHERE PRICE LT 200
END
```

出力結果は次のとおりです。

```
Product  Product
Number:  Name:                                Price:
-------  -------                              ------
1004     2 Hd VCR LCD Menu                    179.00
1008     DVD Upgrade Unit for Cent. VCR       199.00
1026     AR3 35MM Camera 10 X                 129.00
1028     AR2 35MM Camera 8 X                  109.00
1030     QX Portable CD Player                169.00
1032     R5 Micro Digital Tape Recorder        89.00
```

次のコマンドを発行して言語をスペイン語に設定し、上記のリクエストを実行します。

```
SET LANG = SPA
```

出力結果には、PROD_NUM および PRODNAME フィールドに存在する TITLE_ES 属性がフィールドタイトルとして表示されます。PRICE フィールドにはスペイン語のタイトルが存在しないため、TITLE 属性のフィールドタイトルが表示されます。

```
Numero        Nombre
de Producto:  de Producto:                        Price:
------------  ------------                        ------
1004          2 Hd VCR LCD Menu                   179.00
1008          DVD Upgrade Unit for Cent. VCR      199.00
1026          AR3 35MM Camera 10 X                129.00
1028          AR2 35MM Camera 8 X                 109.00
1030          QX Portable CD Player               169.00
1032          R5 Micro Digital Tape Recorder       89.00
```

一時項目 (DEFINE) の記述 - DEFINE

オプションの DEFINE 属性を使用して、レポートの作成時に使用する一時項目 (DEFINE) を作成します。一時項目 (DEFINE) の値は、データソースに存在する情報、つまり永続フィールドから取得することができます。一時項目 (DEFINE) の一般的な使用例は次のとおりです。

- データレコードに存在しない新しい数値を計算する。
- 他の文字列から新しい文字列を計算する。

- データ値を範囲またはグループに分類する。
- 計算内でサブルーチンを呼び出す。

一時項目 (DEFINE) は、データソースを使用してレポートを作成する際に常に使用することができます。

構文　一時項目 (DEFINE) の記述

```
DEFINE fieldname/format [(GEOGRAPHIC_ROLE = georole)]
  [REDEFINES field2] = expression;
  [,TITLE='title',]
  [TITLE_ln='titleln', ... ,]
  [,DESC[CRIPTION]='desc',]
  [DESC_ln='descln', ... ,]$
```

説明

`fieldname`
一時項目 (DEFINE) の名前です。この名前は、FIELDNAME 属性を使用して割り当てる名前の規則に従います。FIELDNAME 属性についての詳細は、98 ページの 「 フィールドの名前 - FIELDNAME 」 を参照してください。

`format`
フィールドのフォーマットです。このフォーマットは、USAGE 属性を使用して割り当てるフォーマットと同様の方法で指定します。詳細は、107 ページの 「 表示データタイプ - USAGE 」 を参照してください。特定のフォーマットを指定しない場合は、デフォルトの D12.2 が割り当てられます。

`georole`
有効な地理的役割です。地理的役割には、名前、郵便番号、ISO (国際標準化機構) コード、FIPS (連邦情報処理標準) コード、NUTS (地域統計分類単位) コードのいずれかを指定することができます。以下は、サポートされている地理的役割のリストを示しています。

- ADDRESS_FULL　完全な住所です。
- ADDRESS_LINE　番地です。
- CITY　都市名です。
- CONTINENT　大陸名です。
- CONTINENT_ISO2　ISO-3166 大陸コードです。
- COUNTRY　国名です。

- COUNTRY_FIPS　FIPS 国コードです。
- COUNTRY_ISO2　ISO-3166-2 国コードです。
- COUNTRY_ISO3　ISO-3166-3 国コードです。
- GEOMETRY_AREA　面ジオメトリです。
- GEOMETRY_LINE　線ジオメトリです。
- GEOMETRY_POINT　点ジオメトリです。
- LATITUDE　緯度です。
- LONGITUDE　経度です。
- NUTS0　国名です (NUTS レベル 0)。
- NUTS0_CC　国コードです (NUTS レベル 0)。
- NUTS1　地域名です (NUTS レベル 1)。
- NUTS1_CC　地域コードです (NUTS レベル 1)。
- NUTS2　州名です (NUTS レベル 2)。
- NUTS2_CC　州コードです (NUTS レベル 2)。
- NUTS3　地区名です (NUTS レベル 3)。
- NUTS3_CC　地区コードです (NUTS レベル 3)。
- POSTAL_CODE　郵便番号です。
- STATE　州名です。
- STATE_FIPS　FIPS 州コードです。
- STATE_ISO_SUB　米国の ISO 州行政区分コードです。
- USSCITY　米国の都市名です。
- USCITY_FIPS　米国の FIPS 都市コードです。
- USCOUNTY　米国の郡名です。
- USCOUNTY_FIPS　米国の FIPS 国コードです。
- USSTATE　米国の州名です。

- USSTATE_ABBR　米国の州の短縮名です。
- USSTATE_FIPS　米国の FIPS 州コードです。
- ZIP3　米国の 3 桁の郵便番号です。
- ZIP5　米国の 5 桁の郵便番号です。

`field2`
フィールド名が複数のセグメントに存在する場合に、そのフィールドの再定義または再計算を行えるようにします。

`expression`
有効な式です。式はセミコロン (;) で終了する必要があります。式についての詳細は、『TIBCO WebFOCUS Language リファレンス』を参照してください。

なお、一時項目 (DEFINE) の式に IF-THEN 句を使用する場合は、ELSE 句を含める必要があります。

`TITLE='title'`
デフォルト言語での一時項目 (DEFINE) のフィールドタイトルです。

`TITLE_ln='titleln'`
言語コード ln で指定された言語での一時項目 (DEFINE) のフィールドタイトルです。

`DESC[CRIPTION]='desc'`
デフォルト言語での一時項目 (DEFINE) の説明です。

`DESC_ln='descln'`
言語コード ln で指定された言語での一時項目 (DEFINE) の説明です。

そのセグメントのすべてのフィールド設営の後に各 DEFINE 属性を配置します。

例　一時項目 (DEFINE) の記述

次の例は、CAR セグメントに「PROFIT」という一時項目 (DEFINE) を記述する方法を示しています。

```
SEGMENT = CARS ,SEGTYPE = S1 ,PARENT = CARREC, $
   FIELDNAME = DEALER_COST ,ALIAS = DCOST ,USAGE = D7, $
   FIELDNAME = RETAIL_COST ,ALIAS = RCOST ,USAGE = D7, $
   DEFINE PROFIT/D7 = RETAIL_COST - DEALER_COST; $
```

参照　マスターファイルの一時項目 (DEFINE) 使用時の注意

DEFINE 属性の使用時には次の規則が適用されます。

- **エイリアス**　DEFINE 属性にはエイリアスはありません。

- **変更**　一時項目 (DEFINE) の宣言はいつでも変更することができます。
- DEFINE FILE コマンドは、マスターファイルに同一名で記述した一時項目 (DEFINE) より優先されます。
- 一時項目 (DEFINE) の値を取得する式が関数を呼び出す場合、USERFCHK パラメータを FULL に設定していないとパラメータの個数とタイプの確認は実行されません。

一時項目 (DEFINE) の使用

DEFINE 属性では、修飾フィールド名を式の左辺に使用することはできません。左辺に WITH 句を使用して、選択した任意の実フィールドと同一セグメントに一時項目 (DEFINE) を配置します。これにより、DEFINE 式がどの時点で評価されるかが決定されます。

DEFINE の右辺の式は、同一パスにある任意セグメントのフィールドを参照することができます。マスターファイルの DEFINE ステートメントの右辺の式には、修飾フィールド名を使用することができます。

マスターファイルの DEFINE 属性は、そのパスにあるフィールドのみを参照することができます。複数のパスにあるフィールドから値を取得する一時項目 (DEFINE) を作成するには、レポートリクエストを発行する前に、代替ビューを使用して DEFINE FILE を使用して一時項目 (DEFINE) を作成する必要があります。詳細は、『TIBCO WebFOCUS Language リファレンス』を参照してください。DEFINE FILE コマンドは、1 回だけ使用する一時項目 (DEFINE) を作成し、マスターファイルにその一時項目 (DEFINE) の宣言を追加したくない場合に使用すると役立ちます。

マスターファイルに記述した一時項目 (DEFINE) は、実際に格納されたフィールドのように扱い、データソースを使用する際に常に使用することができます。そのため、マスターファイルに記述した一時項目 (DEFINE) はレポートリクエストでクリアすることはできません。

一時項目 (DEFINE) は、JOIN 内のクロスリファレンスとして使用することはできません。ただし、ホストフィールドとして使用することはできます。

一時項目 (COMPUTE) の記述 - COMPUTE

COMPUTE コマンドをマスターファイルに含めて、後に続く TABLE リクエストで参照することができます。これにより、式を 1 回だけ作成して、複数のリクエストで再利用することができます。

構文　　マスターファイルへの COMPUTE コマンドの追加

```
COMPUTE fieldname/fmt [(GEOGRAPHIC_ROLE = georole)]
 =expression;
  [,TITLE='title',]
  [TITLE_ln='titleln', ... ,]
  [,DESC[CRIPTION]='desc',]
  [DESC_ln='descln', ... ,]$
```

説明

`fieldname`
一時項目 (COMPUTE) の名前です。

`fmt`
一時項目 (COMPUTE) のフォーマットおよび長さです。

`georole`
有効な地理的役割です。地理的役割には、名前、郵便番号、ISO (国際標準化機構) コード、FIPS (連邦情報処理標準) コード、NUTS (地域統計分類単位) コードのいずれかを指定することができます。以下は、サポートされている地理的役割のリストを示しています。

- ADDRESS_FULL　完全な住所です。
- ADDRESS_LINE　番地です。
- CITY　都市名です。
- CONTINENT　大陸名です。
- CONTINENT_ISO2　ISO-3166 大陸コードです。
- COUNTRY　国名です。
- COUNTRY_FIPS　FIPS 国コードです。
- COUNTRY_ISO2　ISO-3166-2 国コードです。
- COUNTRY_ISO3　ISO-3166-3 国コードです。
- GEOMETRY_AREA　面ジオメトリです。
- GEOMETRY_LINE　線ジオメトリです。
- GEOMETRY_POINT　点ジオメトリです。
- LATITUDE　緯度です。

- LONGITUDE　経度です。
- NUTS0　国名です (NUTS レベル 0)。
- NUTS0_CC　国コードです (NUTS レベル 0)。
- NUTS1　地域名です (NUTS レベル 1)。
- NUTS1_CC　地域コードです (NUTS レベル 1)。
- NUTS2　州名です (NUTS レベル 2)。
- NUTS2_CC　州コードです (NUTS レベル 2)。
- NUTS3　地区名です (NUTS レベル 3)。
- NUTS3_CC　地区コードです (NUTS レベル 3)。
- POSTAL_CODE　郵便番号です。
- STATE　州名です。
- STATE_FIPS　FIPS 州コードです。
- STATE_ISO_SUB　米国の ISO 州行政区分コードです。
- USSCITY　米国の都市名です。
- USCITY_FIPS　米国の FIPS 都市コードです。
- USCOUNTY　米国の郡名です。
- USCOUNTY_FIPS　米国の FIPS 国コードです。
- USSTATE　米国の州名です。
- USSTATE_ABBR　米国の州の短縮名です。
- USSTATE_FIPS　米国の FIPS 州コードです。
- ZIP3　米国の 3 桁の郵便番号です。
- ZIP5　米国の 5 桁の郵便番号です。

`expression`
一時項目 (COMPUTE) の計算に使用する式です。

`TITLE='title'`
デフォルト言語での一時項目 (COMPUTE) のフィールドタイトルです。

`TITLE_ln='titleln'`
言語コード ln で指定された言語での一時項目 (COMPUTE) のフィールドタイトルです。

`DESC[CRIPTION]='desc'`
デフォルト言語での一時項目 (COMPUTE) の説明です。

`DESC_ln='descln'`
言語コード ln で指定された言語での一時項目 (COMPUTE) の説明です。

参照　マスターファイル COMPUTE 使用時の注意

すべてのインスタンスにおいて、マスターファイルで指定する COMPUTE には一時項目 (COMPUTE) と同一の機能および制限が適用されます。特に、マスターファイルの COMPUTE フィールドは次の規則に従う必要があります。

- JOIN、DEFINE、ACROSS 句で使用することはできません。また、演算接頭語とともに使用することはできません。
- 選択条件として使用する場合、構文は IF TOTAL フィールドまたは WHERE TOTAL フィールドです。
- ソートフィールドとして使用する場合、構文は「BY TOTAL COMPUTE field」です。
- 一時項目 (COMPUTE) を見出し、脚注に挿入するには、HEADING または FOOTING コマンドより前に参照する必要があります。

例　一時項目 (COMPUTE) のマスターファイルでのコーディングおよびアクセス

一時項目 (COMPUTE) をマスターファイルにコーディングするには、標準の COMPUTE 構文を使用します。これにより、以降の TABLE リクエストで、その一時項目 (COMPUTE) を参照してアクセスすることができます。動詞オブジェクトとして使用する場合 (次の例を参照)、構文は「SUM (または PRINT) COMPUTE field」です。

次の例は、SALESTES マスターファイル (埋め込まれた COMPUTE で変更された SALES FILE) です。

```
FILENAME=SALESTES, SUFFIX=FOC,
SEGNAME=STOR_SEG, SEGTYPE=S1,
   FIELDNAME=STORE_CODE,  ALIAS=SNO,  FORMAT=A3,   $
   FIELDNAME=CITY,        ALIAS=CTY,  FORMAT=A15,  $
   FIELDNAME=AREA,        ALIAS=LOC,  FORMAT=A1,   $

SEGNAME=DATE_SEG, PARENT=STOR_SEG, SEGTYPE=SH1,
   FIELDNAME=DATE,        ALIAS=DTE,  FORMAT=A4MD, $

SEGNAME=PRODUCT, PARENT=DATE_SEG, SEGTYPE=S1,
   FIELDNAME=PROD_CODE,     ALIAS=PCODE,   FORMAT=A3,    FIELDTYPE=I, $
   FIELDNAME=UNIT_SOLD,     ALIAS=SOLD,    FORMAT=I5,    $
   FIELDNAME=RETAIL_PRICE,  ALIAS=RP,      FORMAT=D5.2M, $
   FIELDNAME=DELIVER_AMT,   ALIAS=SHIP,    FORMAT=I5,    $
   FIELDNAME=OPENING_AMT,   ALIAS=INV,     FORMAT=I5,    $
   FIELDNAME=RETURNS,       ALIAS=RTN,     FORMAT=I3,    MISSING=ON, $
   FIELDNAME=DAMAGED,       ALIAS=BAD,     FORMAT=I3,    MISSING=ON, $

   COMPUTE REVENUE/D12.2M=UNIT_SOLD*RETAIL_PRICE;
```

次の TABLE リクエストは REVENUE フィールドを使用します。

```
TABLE FILE SALESTES
HEADING CENTER
"NEW YORK PROFIT REPORT"
" "
SUM UNIT_SOLD AS 'UNITS,SOLD' RETAIL_PRICE AS 'RETAIL_PRICE'
COMPUTE REVENUE;
BY PROD_CODE AS 'PROD,CODE'
WHERE CITY EQ 'NEW YORK'
END
```

出力結果は次のとおりです。

```
           NEW YORK PROFIT REPORT

  PROD  UNITS
  CODE  SOLD   RETAIL_PRICE          REVENUE
  ----  ----   ------------          -------
  B10      30          $.85           $25.50
  B17      20         $1.89           $37.80
  B20      15         $1.99           $29.85
  C17      12         $2.09           $25.08
  D12      20         $2.09           $41.80
  E1       30          $.89           $26.70
  E3       35         $1.09           $38.15
```

フィルタの記述 - FILTER

ブール一時項目 (DEFINE) (True または False を評価する DEFINE フィールド) を、レコード選択条件として使用することができます。一時項目 (DEFINE) を主としてレコード選択に使用する場合、DEFINE ではなく FILTER を使用して式を保存することにより、マスターファイルでこの目的を明確にし、一時項目 (DEFINE) を分類することができます。フィルタは次の機能を提供します。

- 頻繁に使用する選択条件をマスターファイルに分類、保存して、ビジネスビューでグループ化し、複数のリクエストとツールで再使用することができます。
- フロントエンドツールはこれらを作成し、リクエストの主な機能に適切に使用します。

構文　マスターファイルのフィルタ宣言

```
FILTER  filtername = expression; [MANDATORY={YES|NO}]
  [, DESC[RIPTION]='desc']
  [, DESC_ln='descln', ... ] ,$
```

説明

`filtername`
フィルタに割り当てる名前です。フィルタには内部的に I1 フォーマットが割り当てられ、変更することはできません。

`expression`
True (1 をフィルタフィールドに割り当て) または False (0 をフィルタフィールドに割り当て) を評価する論理式です。その他の式のフィールドは、すべてマスターファイルの標準数値一時項目 (DEFINE) になります。ダイアログマネージャ変数は、標準のマスターファイル DEFINE と同様、フィルタ式で使用することができます。

`MANDATORY={YES|NO}`
シノニムに対するリクエストで参照されていない場合でも、フィルタを適用するかどうかを指定します。YES は、シノニムに対するすべてのリクエストにフィルタを適用します。NO は、リクエストで参照されている場合にのみフィルタを適用します。デフォルト値は NO です。

注意：FILTER FILE コマンドで作成するフィルタが ON と OFF で切り替え可能であることと異なり、この設定を無効にするには、常にマスターファイルの値を削除または変更する必要があります。

`DESC[CRIPTION]='desc'`
デフォルト言語でのソートオブジェクトの説明です。

```
DESC_ln='descln'
```
言語コード ln で指定された言語でのソートオブジェクトの説明です。

構文　リクエストでのマスターファイルフィルタの使用

```
TABLE FILE filename
   .
   .
   .
{WHERE|IF} expression_using_filters
```

説明

`expression_using_filters`
フィルタを参照する論理式です。WHERE 句では、論理式で 1 つまたは複数のフィルタおよび一時項目 (DEFINE) を参照することができます。

参照　マスターファイルフィルタ使用時の注意

- フィルタフィールド名には内部的に I1 フォーマットが割り当てられ、変更することはできません。
- フィルタは、標準の数値一時項目 (DEFINE) としてレポートリクエストの任意の位置に使用することができますが、WHERE TOTAL テストではサポートされません。
- 必須のフィルタを使用することにより、リクエストで参照されていないセグメント (例、クラスタシノニムのテーブル) へのアクセスを、強制的に実行することができます。

例　マスターファイルフィルタの定義と使用

ここでは、MOVIES マスターファイルに次のフィルタ宣言を追加する場合について考察します。

```
FILTER G_RATING = RATING EQ 'G' OR 'PG'; $
```

次のリクエストは G_RATING を適用します。

```
TABLE FILE MOVIES
HEADING CENTER
"Rating G and PG"
PRINT TITLE CATEGORY RATING
WHERE G_RATING
ON TABLE SET PAGE NOPAGE
ON TABLE SET GRID OFF

ON TABLE SET STYLE *
type=report, style=bold, color=black, backcolor=yellow, $
type=data, backcolor=aqua, $
ENDSTYLE
END
```

下図は、出力結果を示しています。

Rating G and PG		
TITLE	CATEGORY	RATING
JAWS	ACTION	PG
CABARET	MUSICALS	PG
BABETTE'S FEAST	FOREIGN	G
SHAGGY DOG, THE	CHILDREN	G
REAR WINDOW	MYSTERY	PG
VERTIGO	MYSTERY	PG
BACK TO THE FUTURE	COMEDY	PG
GONE WITH THE WIND	CLASSIC	G
AIRPLANE	COMEDY	PG
ALICE IN WONDERLAND	CHILDREN	G
ANNIE HALL	COMEDY	PG
FIDDLER ON THE ROOF	MUSICALS	G
BIG	COMEDY	PG
TOP GUN	ACTION	PG
FAMILY, THE	FOREIGN	PG
BAMBI	CHILDREN	G
DEATH IN VENICE	FOREIGN	PG

例　日付フィールドの使用

ここでは、MOVIES マスターファイルに次のフィルタ宣言を追加する場合について考察します。

```
FILTER G_RATING = RATING EQ 'G' OR 'PG'; MANDATORY=YES ,$
```

次のリクエストでは、G_RATING フィルタへの参照は含まれていません。

```
TABLE FILE MOVIES
HEADING CENTER
"Rating G and PG"
PRINT TITLE CATEGORY RATING
ON TABLE SET PAGE NOPAGE
ON TABLE SET GRID OFF

ON TABLE SET STYLE *
type=report, style=bold, color=black, backcolor=yellow, $
type=data, backcolor=aqua, $
ENDSTYLE
END
```

下図は、出力結果を示しています。G_RATING フィルタは、リクエストで参照されていませんが、適用されていることに注意してください。

Rating G and PG		
TITLE	CATEGORY	RATING
JAWS	ACTION	PG
CABARET	MUSICALS	PG
BABETTE'S FEAST	FOREIGN	G
SHAGGY DOG, THE	CHILDREN	G
REAR WINDOW	MYSTERY	PG
VERTIGO	MYSTERY	PG
BACK TO THE FUTURE	COMEDY	PG
GONE WITH THE WIND	CLASSIC	G
AIRPLANE	COMEDY	PG
ALICE IN WONDERLAND	CHILDREN	G
ANNIE HALL	COMEDY	PG
FIDDLER ON THE ROOF	MUSICALS	G
BIG	COMEDY	PG
TOP GUN	ACTION	PG
FAMILY, THE	FOREIGN	PG
BAMBI	CHILDREN	G
DEATH IN VENICE	FOREIGN	PG

ソートオブジェクトの記述 - SORTOBJ

マスターファイル内のソート句および属性をソートオブジェクトとして定義することで、このマスターファイルに対するリクエストでこれらの項目をソートオブジェクト名で参照することができます。ソートオブジェクトのテキスト全体は、TABLE 内のソートオブジェクトが参照されている位置で置換されます。ソートオブジェクト内のソート句の妥当性は、この置換の前に確認されません。マスターファイルの SORTOBJ レコードにソートオブジェクト名および等号 (=) が存在することのみが確認されます。

参照　マスターファイルでのソートオブジェクト使用時の注意

- ソートオブジェクト宣言は、最初の SEGNAME/SEGMENT レコード後の任意の位置に配置することができます。ただし、ソートオブジェクト宣言は、マスターファイルでソートオブジェクトに含めるフィールドのすべて (一時項目 (DEFINE) を含む) が定義された後に配置する必要があります。
- ソートオブジェクトには、マスターファイルフィールドとローカルの一時項目 (DEFINE) の両方を使用することができます。
- マスターファイルに配置するソートオブジェクト宣言の数には制限はありませんが、TABLE リクエストで参照する数は、リクエストで使用するソート句の最大数以下にする必要があります。
- ソートオブジェクト宣言の後には、オプションの属性を続けることができます。
- ソートオブジェクトにフィールドと同一の名前を使用した場合、リクエストでその名前が参照された際にソートオブジェクトが使用されます。

構文　マスターファイルでのソートオブジェクトの宣言

```
FILE= ...
SEG= ...
FIELD= ...
SORTOBJ sortname = {BY|ACROSS} sortfield1 [attributes]
  [{BY|ACROSS} sortfield2 ... ];
  [,DESC[CRIPTION]='desc',]
  [DESC_ln='descln', ... ,]$
```

説明

`sortname`

ソートオブジェクトの名前です。

`sortfield1, sortfield2 ..`

レポート出力のソートに使用する、マスターファイルのフィールドまたはローカルの一時項目 (DEFINE) です。

`attributes`

任意の有効なソート属性です。

`;`

ソートオブジェクト式の末尾を区切るために必要な構文です。

```
DESC[CRIPTION]='desc'
```

デフォルト言語でのソートオブジェクトの説明です。

```
DESC_ln='descln'
```

言語コード ln で指定された言語でのソートオブジェクトの説明です。

構文　リクエストでのソートオブジェクトの参照

```
TABLE FILE ...
   .
   .
   .
BY sortname   .
   .
   .
END
```

説明

```
sortname
```

リクエストに挿入するソートオブジェクトです。

例　ソートオブジェクトの宣言と参照

次の CRSORT は、GGSALES マスターファイルのソートオブジェクトです。このソートオブジェクトには、次の 2 つのソート句が定義されています。

- BY 句で REGION フィールドをソートします。SKIP-LINE 属性が指定されています。
- ACROSS 句で CATEGORY フィールドをソートします。

```
SORTOBJ CRSORT = ACROSS CATEGORY BY REGION SKIP-LINE ; ,$
```

次のリクエストでは、CRSORT ソートオブジェクトが参照されています。

```
TABLE FILE GGSALES
SUM DOLLARS
BY CRSORT
ON TABLE SET PAGE NOPAGE
END
```

出力結果は次のとおりです。

```
                    Category
Region       Coffee         Food          Gifts
-------------------------------------------------
Midwest      4178513        4404483       2931349
Northeast    4201057        4445197       2848289
Southeast    4435134        4308731       3037420
West         4493483        4204333       2977092
```

マスターファイルでの DEFINE FUNCTION の呼び出し

マスターファイル内の一時項目 (DEFINE)、一時項目 (COMPUTE)、および FILTER フィールドの式で、DEFINE FUNCTION を参照することができます。この DEFINE FUNCTION は、関連付けられた式がリクエストで使用される際にメモリにロードされます。

注意：DEFINE FUNCTION は、複数ルートのマスターファイルで使用することはできません。

構文　マスターファイルの式での DEFINE FUNCTION の呼び出し

```
DF.[appname/]filename.functionname(parm1, parm2, ...);
  [DESCRIPTION='description',$
```

説明

`appname`

DEFINE FUNCTION プロシジャが格納されているアプリケーションの名前です (オプション)。

`filename`

DEFINE FUNCTION 定義が記述されたプロシジャの名前です。このプロシジャには、複数の DEFINE FUNCTION 定義を含めることができます。

`functionname(parm1, parm2, ...)`

式に使用する関数名およびパラメータです。

`'description'`

一重引用符で囲まれた説明です (オプション)。

例　マスターファイルでの DEFINE FUNCTION の使用

次の DEFINE FUNCTION は、DMFUNCS プロシジャに格納されています。このプロシジャは、姓 (Lastname) および名 (Firstname) からフルネームを「Lastname, Firstname」のフォーマットで生成します。

```
DEFINE FUNCTION DMPROPER
DESCRIPTION 'Convert name to proper case and last, first format‘
(LASTNAME/A17, FIRSTNAME/A14)
DMPROPER/A34V=LCWORD(17, LASTNAME, 'A17')
             || (', ' | LCWORD(14, FIRSTNAME, 'A14'));
END
```

次の WHOLENAME という一時項目 (DEFINE) は、WF_RETAIL_CUSTOMER マスターファイルに追加されています。この一時項目 (DEFINE) から上記の DEFINE FUNCTION が呼びされます。

```
DEFINE WHOLENAME/A40 = DF.DMFUNCS.DMPROPER(LASTNAME, FIRSTNAME);
     DESCRIPTION = 'Calls DMPROPER to create full name',$
```

次のリクエストでは、上記の WHOLENAME 一時項目 (DEFINE) を使用します。

```
TABLE FILE WF_RETAIL_CUSTOMER
PRINT WHOLENAME AS Whole,Name
BY ID_CUSTOMER
WHERE ID_CUSTOMER LT 600
ON TABLE SET PAGE NOPAGE
END
```

出力結果は次のとおりです。

```
             Whole
ID Customer  Name
-----------  -----
          15  Nolan, Tyler
          20  Bull, Joshua
          78  Wood, Zara
         124  Mckenzie, Callum
         125  Charlton, Bradley
         132  Griffiths, Henry
         152  Rowe, Anthony
         161  Storey, Max
         185  Thomas, Evie
         201  Birch, Brandon
         213  Parry, Maisie
         239  Barrett, Taylor
         258  Lord, Harvey
         270  Bell, Jay
         312  Dunn, Daisy
         352  Mckenzie, Callum
         379  Fisher, Leo
         454  Day, Zak
         472  Howarth, Molly
         503  Barrett, Daniel
         531  Hargreaves, Chloe
         566  Fitzgerald, Bethany
```

マスターファイル DEFINE による日付システム変数の使用

マスターファイルの DEFINE フィールドでは、ダイアログマネージャのシステム日付変数を使用して、リクエストでマスターファイルを解析するたびにシステムの日付を取得することができます。

日付変数の戻り値のフォーマットは、変数名で指定されたフォーマットと同一です。たとえば、&DATEYYMD の日付戻り値のフォーマットは YYYMD です。例外として、変数名の &DATE および &TOD の場合は文字の値を返すため、文字フォーマットのフィールドに割り当てる必要があります。また、DEFINE 式では変数名の &DATE および &TOD を一重引用符 (') で囲む必要があります。

以下は、マスターファイルの DEFINE でサポートされる変数です。

- &DATE
- &TOD
- &DATEMDY
- &DATEDMY

- &DATEYMD
- &DATEMDYY
- &DATEDMYY
- &DATEYYMD
- &DMY
- &YMD
- &MDY
- &YYMD
- &MDYY
- &DMYY

上記以外の変数はマスターファイルではサポートされません。

例　マスターファイルの DEFINE での日付変数 &DATE の使用

次の EMPLOYEE マスターファイルには、「TDATE」という一時項目 (DEFINE) が追加されています。TDATE のフォーマットは A12 で、&DATE の値を取得します。戻り値は文字であり、この変数は一重引用符 (') で囲む必要があります。

```
FILENAME=EMPLOYEE, SUFFIX=FOC
SEGNAME=EMPINFO,  SEGTYPE=S1
 FIELDNAME=EMP_ID,        ALIAS=EID,     FORMAT=A9,       $
 FIELDNAME=LAST_NAME,     ALIAS=LN,      FORMAT=A15,      $
 FIELDNAME=FIRST_NAME,    ALIAS=FN,      FORMAT=A10,      $
 FIELDNAME=HIRE_DATE,     ALIAS=HDT,     FORMAT=I6YMD,    $
 FIELDNAME=DEPARTMENT,    ALIAS=DPT,     FORMAT=A10,      $
 FIELDNAME=CURR_SAL,      ALIAS=CSAL,    FORMAT=D12.2M,   $
 FIELDNAME=CURR_JOBCODE, ALIAS=CJC,      FORMAT=A3,       $
 FIELDNAME=ED_HRS,        ALIAS=OJT,     FORMAT=F6.2,     $
DEFINE TDATE/A12   ='&DATE';, $
   .
   .
   .
```

次のリクエストは TDATE の値を表示します。

```
TABLE FILE EMPLOYEE
PRINT LAST_NAME FIRST_NAME HIRE_DATE TDATE AS 'TODAY''S,DATE'
WHERE LAST_NAME EQ 'BANNING'
END
```

出力結果は次のとおりです。

LAST_NAME	FIRST_NAME	HIRE_DATE	TODAY'S DATE
BANNING	JOHN	82/08/01	06/17/04

例　マスターファイルの DEFINE での日付変数 &YYMD の使用

次の EMPLOYEE マスターファイルには、「TDATE」という一時項目 (DEFINE) が追加されています。TDATE のフォーマットは YYMD で、&YYMD の値を取得します。

```
FILENAME=EMPLOYEE, SUFFIX=FOC
SEGNAME=EMPINFO,  SEGTYPE=S1
 FIELDNAME=EMP_ID,        ALIAS=EID,     FORMAT=A9,       $
 FIELDNAME=LAST_NAME,     ALIAS=LN,      FORMAT=A15,      $
 FIELDNAME=FIRST_NAME,    ALIAS=FN,      FORMAT=A10,      $
 FIELDNAME=HIRE_DATE,     ALIAS=HDT,     FORMAT=I6YMD,    $
 FIELDNAME=DEPARTMENT,    ALIAS=DPT,     FORMAT=A10,      $
 FIELDNAME=CURR_SAL,      ALIAS=CSAL,    FORMAT=D12.2M,   $
 FIELDNAME=CURR_JOBCODE, ALIAS=CJC,      FORMAT=A3,       $
 FIELDNAME=ED_HRS,        ALIAS=OJT,     FORMAT=F6.2,     $
DEFINE TDATE/YYMD   = &YYMD ;, $
   .
   .
   .
```

次のリクエストは TDATE の値を表示します。

```
TABLE FILE EMPLOYEE
PRINT LAST_NAME FIRST_NAME HIRE_DATE TDATE AS 'TODAY''S,DATE'
WHERE LAST_NAME EQ 'BANNING'
END
```

出力結果は次のとおりです。

LAST_NAME	FIRST_NAME	HIRE_DATE	TODAY'S DATE
BANNING	JOHN	82/08/01	2004/06/17

参照　マスターファイル の DEFINE での日付システム変数用のメッセージ

マスターファイルの一時項目 (DEFINE) でサポートされていない変数を使用すると、次のメッセージが表示されます。

```
(FOC104) マスターファイルの DEFINE フィールドに誤りがあります。
```

変数を使用したマスターファイルおよびアクセスファイルのパラメータ化

マスターファイルでグローバル変数を定義し、その変数を使用してマスターファイルおよびそれに対応するアクセスファイル内の特定の属性をパラメータ化することができます。たとえば、アクセスファイル内の接続属性をマスターファイルで定義した変数でパラメータ化し、実行時に実際の接続名を指定することができます。

構文　マスターファイル変数の作成

マスターファイルの FILE 宣言の前に、次の変数定義を追加します。

```
VARIABLE NAME=[&&]var, USAGE=Aln, [DEFAULT=defvalue,][QUOTED={OFF|ON},]
[PERSISTENT={ON|OFF},]$
```

説明

`[&&]var`

グローバル変数に割り当てる名前です。マスターファイルまたはアクセスファイル内の変数を参照するときは、2 つのアンパサンド (&) を名前の先頭に追加する必要があります。ただし、変数を定義する場合は、これらのアンパサンドはオプションとして追加します。

`ln`

変数値の最大の長さです。

`defvalue`

指定した変数のデフォルト値です。実行時に値が設定されていない場合は、この値が使用されます。

`QUOTED = {OFF|ON}`

ON を指定すると、変数に割り当てられた文字列が一重引用符 (') で囲まれます。文字列内の一重引用符は、2 つの一重引用符に変換されます。デフォルト値は OFF です。

`PERSISTENT = {ON|OFF}`

ON は、この変数に割り当てられたデフォルト値が変更できないことを示し、この変数をフィルタの選択として使用することはできません。この設定には、デフォルトで ON が適用されます。

OFF は、この変数に割り当てられたデフォルト値が変更できることを示し、この変数をフィルタの選択として使用することができます。

参照　マスターファイルおよびアクセスファイルの属性での変数サポート

マスターファイルでは、POSITION、OCCURS、REMARKS、DESCRIPTION、TITLE、HELPMESSAGE 属性を、変数でパラメータ化することができます。

マスターファイルの DBA セクションでは、USER、VALUE 属性をパラメータ化することができます。マスターファイルプロファイルでこれらの変数を使用して動的 DBA ルールを作成する方法についての詳細は、35 ページの 「マスターファイルプロファイルの作成および使用 」を参照してください。

アクセスファイルでは、CONNECTION、TABLENAME、WORKSHEET (Excel 直接取得)、START、CHKPT_SAVE、CHKPT_FILE、POLLING、TIMEOUT、MAXLUWS、ACTION、MSGLIMIT、DIRECTORY、NAME、EXTENSION、DATA_ORIGIN、MAXFILES、MAXRECS、 OBJECT、PICKUP、TRIGGER、DISCARD、ARCHIVE 属性を、変数でパラメータ化することができます。

注意：複数の変数を連結して、1 つの属性値を作成することができます。

例　マスターファイルおよびアクセスファイルでの属性のパラメータ化

次のリクエストは、EMPLOYEE という名前の FOCUS データソースから、ORAEMP という名前の Oracle テーブルを作成します。

```
TABLE FILE EMPLOYEE
SUM LAST_NAME FIRST_NAME CURR_SAL CURR_JOBCODE DEPARTMENT
BY EMP_ID
ON TABLE HOLD AS ORAEMP FORMAT SQLORA
END
```

このリクエストで作成されるマスターファイルは次のとおりです。

```
FILENAME=ORAEMP , SUFFIX=SQLORA , $
  SEGMENT=SEG01, SEGTYPE=S0, $
    FIELDNAME=EMP_ID, ALIAS=EID, USAGE=A9, ACTUAL=A9, $
    FIELDNAME=LAST_NAME, ALIAS=LN, USAGE=A15, ACTUAL=A15, $
    FIELDNAME=FIRST_NAME, ALIAS=FN, USAGE=A10, ACTUAL=A10, $
    FIELDNAME=CURR_SAL, ALIAS=CSAL, USAGE=D12.2M, ACTUAL=D8, $
    FIELDNAME=CURR_JOBCODE, ALIAS=CJC, USAGE=A3, ACTUAL=A3, $
    FIELDNAME=DEPARTMENT, ALIAS=DPT, USAGE=A10, ACTUAL=A10, $
```

このリクエストで作成されるアクセスファイルは次のとおりです。

```
SEGNAME=SEG01, TABLENAME=ORAEMP, KEYS=01, WRITE=YES, $
```

次の変数定義をマスターファイルに追加して、アクセスファイル内の TABLENAME 属性およびマスターファイル内の EMP_ID フィールドの TITLE 属性をパラメータ化します。

```
FILENAME=ORAEMP, SUFFIX=SQLORA , $
VARIABLE NAME=table, USAGE=A8, DEFAULT=EDUCFILE, $
VARIABLE NAME=emptitle, USAGE=A30, DEFAULT=empid,$
```

マスターファイルで、EMP_ID の FIELD 宣言に TITLE 属性を追加します。

```
FIELDNAME=EMP_ID, ALIAS=EID, USAGE=A9, ACTUAL=A9,
      TITLE='&&emptitle', $
```

アクセスファイルで、TABLENAME 属性の値を変数名に置換します。

```
SEGNAME=SEG01, TABLENAME=&&table, KEYS=01, WRITE=YES, $
```

次のリクエストは、変数の値を設定し、TABLE リクエストを発行します。

```
-SET &&table = ORAEMP;
-SET &&emptitle = 'Id,number';
TABLE FILE ORAEMP
PRINT EMP_ID LAST_NAME FIRST_NAME DEPARTMENT
END
```

ここで、-SET コマンド内の &&emptitle の値が一重引用符 (') で囲まれています。これは、この値に特殊文字のカンマ (,) が含まれているためです。一重引用符は文字列の一部ではなく、またレポート出力には表示されません。変数定義に QUOTED=ON 属性を含めた場合は、フィールドタイトルが一重引用符に囲まれた状態で表示されます。

レポート出力では、EMP_ID フィールドのフィールドタイトルに、&&emptitle で設定された値が表示されます。また、このリクエストがアクセスするテーブルは、この例の最初の手順で作成した ORAEMP テーブルになります。

```
Id
number     LAST_NAME       FIRST_NAME  DEPARTMENT
------     ---------       ----------  ----------
071382660  STEVENS         ALFRED      PRODUCTION
112847612  SMITH           MARY        MIS
117593129  JONES           DIANE       MIS
119265415  SMITH           RICHARD     PRODUCTION
119329144  BANNING         JOHN        PRODUCTION
123764317  IRVING          JOAN        PRODUCTION
126724188  ROMANS          ANTHONY     PRODUCTION
219984371  MCCOY           JOHN        MIS
326179357  BLACKWOOD       ROSEMARIE   MIS
451123478  MCKNIGHT        ROGER       PRODUCTION
543729165  GREENSPAN       MARY        MIS
818692173  CROSS           BARBARA     MIS
```

例　変数の連結による属性値の作成

次の例の TABLENAME 属性には、データベース名、オーナー ID、テーブル接頭語、および変数接尾語を含む固定テーブル名で構成される複数部分名を指定する必要があります。ここでは、各部分の変数を別々に定義し、それらの変数を連結します。

最初に、各部分の変数を別々に定義します。

```
VARIABLE NAME=db,USAGE=A8,DEFAULT=mydb,$
VARIABLE NAME=usr,USAGE=A8,DEFAULT=myusrid,$
VARIABLE NAME=tprf,USAGE=A4,DEFAULT=test_,$
VARIABLE NAME=tsuf,USAGE=YYM,$
```

アクセスファイルで、これらの変数を連結して TABLENAME 属性を作成します。各部分の区切り文字はピリオド (.) ですが、変数名を連結してピリオドを保持するためには、2 つのピリオドを使用する必要があります。

```
TABLENAME=&db..&usr..&tprf.table&tsuf,
```

デフォルト値に基づいて、TABLENAME は次のようになります。

```
TABLENAME=mydb.myusrid.test_table
```

リクエストでは、それぞれの変数に次の値を設定します。

```
I-SET &&db=db1;
-SET &&tprf=prod_;
-SET &&tsuf=200801;
```

これらの値により、使用される TABLENAME は次のようになります。

```
TABLENAME=db1.myusrid.prod_table200801
```

文字日付の WebFOCUS 日付への変換

データソースには、日付の値が文字フォーマットで格納され、特別な規格が存在せず、年、四半期、月などの構成要素の任意の組み合わせ、および任意の区切り文字を使用するものがあります。レポートにソートが設定されている場合、このようなデータはアルファベット順にソートされ、結果は実務上の意味を持ちません。データフィールドのソート、集計、レポート実行を適切に行うため、WebFOCUS では、「DATEPATTERN」というマスターファイル属性で指定する変換パターンを使用して、文字フォーマットの日付を標準 WebFOCUS 日付フォーマットに変換することができます。

パターンの各要素は、実際に入力される特定の文字、または日付構成要素を表す変数です。マスターファイルの USAGE 属性を編集して、日付構成要素の日付パターンを記述する必要があります。DATEPATTERN 文字列の最大長は 64 バイトです。

参照　DATEPATTERN 使用時の注意

- 元の日付に WebFOCUS の USAGE フォーマットと同等の構成要素が含まれていない場合、変換後の日付は元のデータのような表示にはなりません。その場合、元のデータを表示するには、OCCURS セグメントを使用して元の文字フォーマットでフィールドを再定義することで、リクエストでそのフィールドを表示できる場合があります。
- DATEPATTERN では、ACTUAL フォーマットから USAGE フォーマットへの変換が必要です。そのため、DATEPATTERN は SUFFIX=FOC および SUFFIX=XFOC データソースではサポートされません。

日付パターン変数の指定

有効な日付構成要素 (変数) は、年、四半期、月、日、曜日です。日付パターン内の変数は、大括弧 ([ ]) で囲みます。これらの大括弧は、入力または出力の一部ではありません。データに大括弧が含まれる場合は、日付パターン内でエスケープ文字を使用して、データの大括弧を変数を囲む大括弧と区別する必要があります。

構文　日付パターンに年を指定

`[YYYY]`

4 桁の年を指定します。

`[YYYY]`

4 桁の年を指定します。

`[YY]`

2 桁の年を指定します。

`[yy]`

0 (ゼロ) を非表示にする 2 桁の年を指定します (例、2008 年の場合は 8)。

`[by]`

ブランクでパディングされた 2 桁の年を指定します。

構文　日付パターンに月を表す数値を指定

`[MM]`

月を表す数値を 2 桁で指定します。

`[mm]`

0 (ゼロ) を非表示にする月を表す数値を指定します。

`[bm]`

ブランクでパディングされた月を表す数値を指定します。

構文　日付パターンに月名を指定

`[MON]`

月を表す 3 文字を大文字で指定します。

`[mon]`

月を表す 3 文字を小文字で指定します。

`[Mon]`

月を表す 3 文字を先頭大文字で指定します。

`[MONTH]`

月の完全名を大文字で指定します。

`[month]`

月の完全名を小文字で指定します。

`[Month]`

月の完全名を先頭大文字で指定します。

構文　日付パターンに日付を指定

`[DD]`

2 桁の月単位の日付を指定します。

`[dd]`

0 (ゼロ) を非表示にする月単位の日付を指定します。

`[bd]`

ブランクでパディングされた月単位の日付を指定します。

構文　　日付パターンにユリウス暦の日付を指定

`[DDD]`

3 桁の年単位の日付を指定します。

`[ddd]`

0 (ゼロ) を非表示にする年単位の日付を指定します。

`[bdd]`

ブランクでパディングされた年単位の日付を指定します。

構文　　日付パターンに曜日を指定

`[WD]`

1 桁の曜日を指定します。

`[DAY]`

曜日を表す 3 文字を大文字で指定します。

`[day]`

曜日を表す 3 文字を小文字で指定します。

`[Day]`

曜日を表す 3 文字を先頭大文字で指定します。

`[WDAY]`

曜日の完全名を大文字で指定します。

`[wday]`

曜日の完全名を小文字で指定します。

`[Wday]`

曜日の完全名を先頭大文字で指定します。

曜日の場合、WEEKFIRST の設定で、週の開始日を定義します。

構文　　日付パターンに四半期を指定

`[Q]`

1 桁の四半期番号を指定します (1、2、3、4 のいずれか)。

Q2 や Q02 などの文字列の場合は、[Q] の前に定数を使用します (例、Q0[Q])。

日付パターン定数の指定

変数の間に、任意の定数値を挿入することができます。

通常は変数の一部として解釈される文字を挿入する場合は、円記号 (¥) を使用します。以下はその例です。

- 定数値として左大括弧 ([) を指定するには、¥[を使用します。
- 定数値として円記号 (¥) を指定するには、¥¥ を使用します。

一重引用符 (') の場合は、連続する 2 つの一重引用符 ('') を使用します。

日付パターンサンプル

データソースの日付が「CY 2001 Q1」の形式の場合、DATEPATTERN 属性は次のとおりです。

```
DATEPATTERN = 'CY [YYYY] Q[Q]'
```

データソースの日付が「Jan 31, 01」の形式の場合、DATEPATTERN 属性は次のとおりです。

```
DATEPATTERN = '[Mon] [DD], [YY]'
```

データソースの日付が「APR-06」の形式の場合、DATEPATTERN 属性は次のとおりです。

```
DATEPATTERN = '[MON]-[YY]'
```

データソースの日付が「APR - 06」の形式の場合、DATEPATTERN 属性は次のとおりです。

```
DATEPATTERN = '[MON] - [YY]'
```

データソースの日付が「APR '06」の形式の場合、DATEPATTERN 属性は次のとおりです。

```
DATEPATTERN = '[MON] ''[YY]'
```

データソースの日付が「APR [06]」の形式の場合、DATEPATTERN 属性は次のとおりです。

```
DATEPATTERN = '[MON] ¥[[YY]¥]' (or '[MON] ¥[[YY]]'
```

右大括弧に、エスケープ文字を追加する必要はありません。

例　文字の日付によるソート

次の例で、 date1.ftm は、以下のデータが格納されたシーケンシャルファイルです。

```
June 1, '02
June 2, '02
June 3, '02
June 10, '02
June 11, '02
June 12, '02
June 20, '02
June 21, '02
June 22, '02
June 1, '03
June 2, '03
June 3, '03
June 10, '03
June 11, '03
June 12, '03
June 20, '03
June 21, '03
June 22, '03
June 1, '04
June 2, '04
June 3, '04
June 4, '04
June 10, '04
June 11, '04
June 12, '04
June 20, '04
June 21, '04
June 22, '04
```

DATE1 マスターファイルの DATE1 フィールドには、USAGE フォーマットおよび ACTUAL フォーマットが設定されます。これらは両者とも文字 (A18) フォーマット です。

```
FILENAME=DATE1   , SUFFIX=FIX ,
  DATASET = c:¥tst¥date1.ftm    , $
  SEGMENT=FILE1, SEGTYPE=S0, $
    FIELDNAME=DATE1, ALIAS=E01, USAGE=A18, ACTUAL=A18, $
```

次のリクエストは、DATE1 FIELD でソートします。

```
TABLE FILE DATE1
PRINT DATE1 NOPRINT
BY DATE1
ON TABLE SET PAGE NOPAGE
END
```

出力で、文字の日付は、日付順ではなくアルファベット順にソートされています。

```
DATE1
-----
June 1, '02
June 1, '03
June 1, '04
June 10, '02
June 10, '03
June 10, '04
June 11, '02
June 11, '03
June 11, '04
June 12, '02
June 12, '03
June 12, '04
June 2, '02
June 2, '03
June 2, '04
June 20, '02
June 20, '03
June 20, '04
June 21, '02
June 21, '03
June 21, '04
June 22, '02
June 22, '03
June 22, '04
June 3, '02
June 3, '03
June 3, '04
June 4, '04
```

日付を正しくソートするため、マスターファイルに DATEPATTERN 属性を追加して、WebFOCUS によって日付が WebFOCUS の日付フィールドに変換されるようにします。また、USAGE フォーマットを編集し、そのフィールドを WebFOCUS の日付フォーマットにする必要もあります。適切なパターンを構成するため、格納されている日付のすべての要素を記述する必要があります。文字の日付には、次の変数と定数があります。

- 変数 - 先頭大文字の月の完全名 [Month]。
- 定数 - ブランク。
- 変数 - ゼロを非表示にした月を表す数値 [dd]。
- 定数 - カンマ (,)、ブランク、アポストロフィ (') (パターンでは 2 つのアポストロフィとして記述) の順に入力。
- 変数 - 2 桁の年 [YY]。

編集後のマスターファイルは、次のようになります。DEFCENT 属性を追加して、2 桁の年を 4 桁の年に変換します。

```
FILENAME=DATE1   , SUFFIX=FIX ,
  DATASET = c:¥tst¥date1.ftm    , $
  SEGMENT=FILE1, SEGTYPE=S0, $
    FIELDNAME=DATE1, ALIAS=E01, USAGE=MtrDYY, ACTUAL=A18,
      DEFCENT=20,
      DATEPATTERN = '[Month] [dd], ''[YY]', $
```

同一のリクエストを発行すると、出力は次のようになります。DATE1 は、USAGE で指定した MtrDYY フォーマットの WebFOCUS の日付に変換されています。

```
DATE1
-----
June  1, 2002
June  2, 2002
June  3, 2002
June 10, 2002
June 11, 2002
June 12, 2002
June 20, 2002
June 21, 2002
June 22, 2002
June  1, 2003
June  2, 2003
June  3, 2003
June 10, 2003
June 11, 2003
June 12, 2003
June 20, 2003
June 21, 2003
June 22, 2003
June  1, 2004
June  2, 2004
June  3, 2004
June  4, 2004
June 10, 2004
June 11, 2004
June 12, 2004
June 20, 2004
June 21, 2004
June 22, 2004
```

5

FOCUS データソースの記述

ここでは、FOCUS データソースに特化したデータ記述について説明します。

- **設計のヒント**　新規の FOCUS データソースの設計時および既存のデータソースの変更時に必要な情報を提供します。
- **セグメントの記述**　FOCUS データソースのマスターファイルで指定するセグメント宣言について説明します。たとえば、SEGTYPE 属性を使用してセグメント関係、キー、ソート順を定義する方法や LOCATION 属性を使用してセグメントを異なる場所に格納する方法などがあります。
- **フィールドの記述**　FOCUS データソースのマスターファイルで指定するフィールド宣言について説明します。たとえば、ACCEPT 属性の FIND オプション、INDEX 属性を使用したインデックスフィールド、GROUP 属性を使用したフィールド順序の再定義、FORMAT 属性で定義するデータタイプ別の内部格納領域および MISSING 属性で定義するミッシング値の要件などがあります。
- **分割データソースの記述**　マスターファイルおよびアクセスファイルの宣言を使用して高度な方法で FOCUS データソースを分割する方法について説明します。

トピックス

FOCUS データソースのタイプ

作成する FOCUS データソースのタイプは、必要とする格納領域の容量により決定します。

- FOCUS データソース (SUFFIX = FOC) は、4 キロバイトのデータベースページで構成されます。
- XFOCUS データソース (SUFFIX = XFOCUS) は、16 キロバイトのデータベースページで構成されます。

下表は、FOCUS データおよび XFOCUS データのセグメントで使用可能なバイト数を示しています。

セグメントタイプ	FOCUS データソース	XFOCUS データソース
単一セグメントマスターファイルのセグメント	3968	16284
複数セグメントマスターファイルのルートまたはリーフ	3964	16280
その他すべてのセグメント	3960	16276

SUFFIX=FOC データソースの使用

FOCUS データソースの容量は、物理データファイルごとに最大で 2 ギガバイトです。分割することにより、1 つの論理 FOCUS データソースに最大で 2 ギガバイトの物理ファイルを 1022 個まで含めることができます。

XFOCUS データソースの使用

XFOCUS データソースは従来の FOCUS データソースの流れを汲んだ新しいデータベース構造で、パフォーマンスおよびデータ許容量が大幅に向上し、より優れた機能を提供します。

- セグメントインスタンス当たり 4 倍以上のデータ許容量。
- 物理ファイル当たり 16 倍となる最大 32 ギガバイトのデータ格納。

XFOCUS データソースは 16 キロバイトのデータベースページで構成されます。マスターファイルで指定する SUFFIX は XFOCUS です。FOCUS データソース (SUFFIX=FOC) は 4 キロバイトのデータベースページで構成されます。

FOCUS ファイルで動作する既存のコマンドはすべて XFOCUS ファイルでも動作します。

FOCUS データソースを XFOCUS データソースに変換するには、マスターファイルの SUFFIX 属性を変更し、REBUILD ユーティリティを適用します。

構文　XFOCUS データソースの指定

```
FILE = filename, SUFFIX = XFOCUS, $
```

説明

`filename`
任意の有効なファイル名です。

構文　XFOCUS データソースバッファのページ数の設定

FOCUS データソースは BINS 設定で割り当てられたバッファページを使用します。XFOCUS データソースのバッファページは XFOCUSBINS 設定で割り当てます。

```
SET XFOCUSBINS = n
```

説明

`n`
XFOCUS データソースのバッファに使用するページ数です。有効値は 16 から 1023 です。デフォルト値は 64 です。

XFOCUS データソースがセッションで使用されるまで、実際にメモリが割り当てられることはありません。そのため、? SET XFOCUSBINS クエリコマンドを発行すると、XFOCUS バッファに設定されているページ数および実際にメモリが割り当てられているかどうかを確認することができます (割り当て前が passive、割り当て済みが active)。

手順　XFOCUS データソースを作成するには

XFOCUS データソースを作成するには 2 通りの方法があります。CREATE FILE コマンドを発行する方法と HOLD FORMAT XFOCUS コマンドを使用する方法があります。

CREATE FILE コマンドを使用して XFOCUS データソースを作成するには、次の手順を実行します。

1. SUFFIX=XFOCUS を指定したマスターファイルを作成します。
2. CREATE FILE コマンドを発行します。

```
CREATE FILE name
```

説明

`name`
SUFFIX=XFOCUS を指定したマスターファイルの名前です。

参照　XFOCUS データソース使用時の注意

- UNIX および Windows での拡張子は .foc です。
- フォーマットの A4096 を使用した文字フィールドがサポートされます。SIUFFIX=FOC での制限は A3968 です。
- CALCFILE ユーティリティは、SUFFIX タイプに合わせて演算アルゴリズムを自動的に調整します。このユーティリティは、ファイルのサイズ調整に使用することができます。
- USE コマンドは SUFFIX タイプの XFOCUS および FOC が混在する 1022 個のファイルを扱うことができます。ただし、それぞれのファイルに対応するマスターファイルに SUFFIX (FOCUS は FOC、XFOCUS は XFOCUS) を正しく指定しておく必要があります。

  同一の USE で SUFFIX=FOC のデータソースに USE...AS を指定し、SUFFIX=XFOCUS のデータソースに別の AS を指定します。
- 以前のバージョンでは SUFFIX=XFOCUS が有効な値として認識されないため、以前のバージョンを使用して、SUFFIX=XFOCUS のデータソースにアクセスしようとすると、エラーが発生します。
- SUFFIX=FOC および SUFFIX=XFOCUS のデータソース間では、JOIN コマンドを使用することができます。マスターファイルのクロスリファレンスは、他の SUFFIX =XFOCUS のデータソースに対してのみ使用することができます。

- COMBINE コマンドを SUFFIX=XFOCUS データソースに使用することができます。SUFFIX=FOC と SUFFIX XFOCUS データソースを 1 つの COMBINE で結合することができます。

FOCUS データソースの設計

データベース管理システムを使用すると、高度な階層データ構造を作成することができます。ここでは、効果的かつ効率的に利用可能な FOCUS データソースを設計する方法および作成したデータソースの設計を変更する方法について説明します。

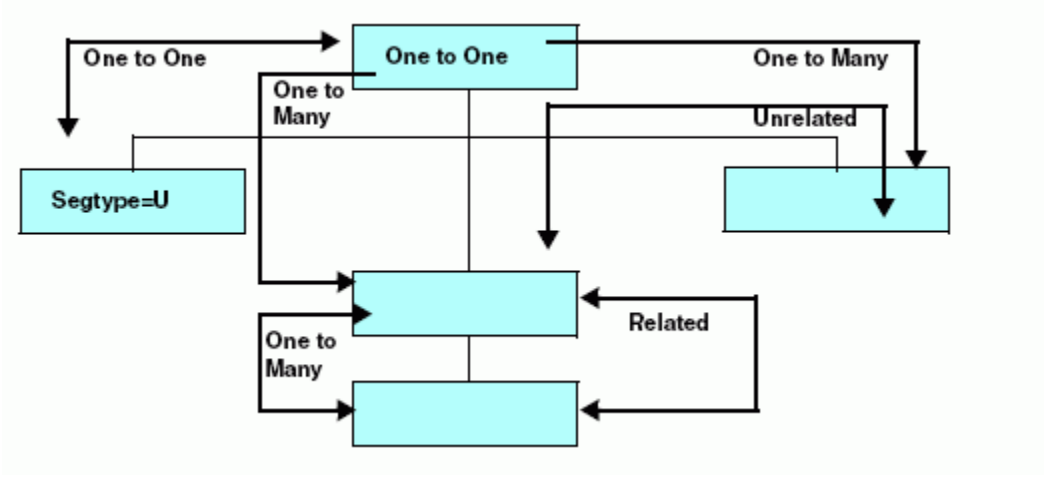
データ間の関係

データソースを設計する場合、さまざまなフィールド間の関係を最初に考慮する必要があります。マスターファイルを作成する前に、これらの関係を図式化します。たとえば、関係が成立するフィールドを特定します。次に、フィールド間に関係がある場合、それが 1 対 1 の関係であるか、1 対 n の関係であるかを特定します。また、関係するデータが他のデータソースにすでに存在する場合、そのデータソースとこれから作成するデータソースを結合できるかどうかを特定します。

一般的なガイドラインは次のとおりです。

- 特定のレコードで 1 回だけ出現するすべての情報は、ルートセグメントまたはユニーク子セグメントに配置します。
- 通常、2 つのデータソースを結合して取得可能なすべての情報は、結合したデータソースから取得し、2 つのデータソースで同一の情報を重複して保守することはありません。
- 特定のセグメントとの情報に n 対 1 の関係を持つすべての情報は、そのセグメントの下位セグメントに格納します。
- 子セグメントで互いに関係するデータは同一パスに、関係しないデータは別のパスに格納します。

下図は、データ関係を構築する際に考慮する規則を図示したものです。



JOIN に関する考慮事項

2 つのセグメントを結合する場合、ホストフィールドとクロスリファレンスフィールドのフォーマットが一致していなければなりません。また、クロスリファレンスフィールドには INDEX 属性を使用してインデックスを付ける必要があります。また、マスターファイルで指定するクロスリファレンスでは、ホストフィールドとクロスリファレンスフィールドが共通の名前を持つ必要があります。この場合、両方のフィールドの名前またはエイリアスが完全に一致するか、一方のフィールドの名前が他方のエイリアスと一致する必要があります。

一般的な効率性に関する考慮事項

クエリを実行してレコードを検索する場合、FOCUS データソースは最初にルートセグメントを読み取り、次に階層内の他のセグメントを読み取ります。ルートセグメントを小さくすると、1 回に読み取れるルートセグメントインスタンス数が多くなるため、クエリを処理してレコードを選択する速度が向上します。

AUTOPATH を設定して、レコードの変換効率を向上させることもできます。AUTOPATH は、TABLE FILE ddname.fieldname 構文を自動実行するものです。ここで、fieldname はインデックスフィールド以外のフィールドで、物理的な検索はこのフィールドのセグメントから開始されます。AUTOPATH についての詳細は、『TIBCO WebFOCUS アプリケーション作成ガイド』を参照してください。

情報処理の問題でよく見られるように、効率的な FOCUS データソースを設計しようとすると必ず相反することが発生します。ルートセグメントのサイズを小さくしてレコードの検索速度を上げることと、レコードの選択テストに使用するフィールドを可能な限り階層データ構造の上位に配置してレコードの選択速度を上げることは相反します。この場合は両者のバランスを考慮する必要があります。WHERE または IF テストに使用するフィールドのセグメントロケーションは、リクエストの処理効率に大きく影響します。フィールドのレコード選択テストが失敗した場合、そのセグメントまたは下位セグメントのインスタンスに対して追加の処理は行われません。データ構造内の選択フィールドの位置が上位になるほど、セグメントを読み取ってレコードのステータスを決定する際のセグメント数は少なくなります。

データソースを設計して作成した後、データ構造の下位フィールドに基づいてレコードを選択する場合は、代替ビューを使用してデータ構造を回転し、そのフィールドを一時的に上位に配置することができます。代替ビューについての詳細は、55 ページの「 フィールドグループの記述 」 を参照してください。レポートリクエストで代替ビューを使用する方法についての詳細は、『TIBCO WebFOCUS Language リファレンス』を参照してください。

効率的なデータ構造を設計するには、次のガイドラインを参考にしてください。

- ルートセグメントに追加する情報は、レコードの識別に必要な情報および選択条件でよく使用する情報のみに制限します。
- 不要なキーフィールドは省略します。SEGTYPE=S1 のセグメントは、たとえば SEGTYPE=S9 のようなセグメントより効率的に処理されます。
- 日付シーケンスでデータの追加または保守を行う場合は、SEGTYPE=SH1 のセグメントを使用します。この場合、SEGTYPE=SH1 により、最新の日付が論理的にデータソースの末尾ではなく先頭に配置されます。
- レコード選択テストによく使用するフィールドがセグメントに含まれている場合は、キーフィールド、選択フィールド、レポートによく使用フィールドに限定することでセグメントを小さくします。
- ユニークインスタンスの検索をよく実行する場合は、そのフィールドにインデックスを付けます。フィールドにインデックスを指定する場合は、データ値およびそのデータ値に対応するデータソース内の物理ロケーションで構成されたテーブルを作成し、保守します。このため、フィールドにインデックスを付けると、検索速度が向上します。

  インデックスは任意のフィールドに付けることができますが、インデックスには追加の格納領域が必要になるため、データソース内のインデックス数に制限を設定することをお勧めします。格納領域の増加と検索速度の向上のバランスを考慮する必要がります。

FOCUS データソースの変更

FOCUS データソースを設計および作成した後、マスターファイルの属性を編集してデータソースの特性の一部を変更することができます。データソースを作成後に編集できる属性タイプについては、各属性の説明をを参照してください。

REBUILD 機能を使用してデータソースを再構築する場合、属性を編集できなくてもその特性を変更できるものがあります。詳細は、367 ページの 「 データソースの作成と再構築 」 を参照してください。また REBUILD を使用して、データソースに新しいフィールドを追加することもできます。

セグメントの記述

セグメント記述では、キーフィールド、ソート順、セグメントの関係を定義することができます。セグメントの最大数は、FOCUS データソースでは 64、XFOCUS データソースでは 512 です。この個数には、セグメント数、インデックス数、TEXT ロケーションファイル数が含まれます (各 TEXT ロケーションファイルには複数の TEXT フィールドを記述可能)。

JOIN コマンドで作成される構造には、最大で 1024 個のセグメントを含めることができます。FOCUS データソースは、最大 1024 セグメントで構成される JOIN 構造の一部として使用することができます。

インデックスビューを使用すると、その構造に使用されるセグメントとインデックスの合計の最大数が 191 に減少します。AUTOINDEX を ON に設定している場合は、明示的に指定しなくてもインデックスビューを自動的に使用していることがあります。

マスターファイルで LOCATION セグメントを記述して別の物理ファイルロケーションを指定すれば、ファイルサイズを拡張することができます。

また、AUTODATE を使用して、セグメントを変更した日時のタイムスタンプを記録するフィールドを作成することもできます。

FOCUS セグメント間で JOIN を定義するには、「CRFILE」、「CRKEY」、「CRSEGNAME」という 3 つの追加の属性を使用します。詳細は、271 ページの「 マスターファイルでの JOIN の定義 」を参照してください。

キーフィールド、ソート順、セグメント関係の記述 - SEGTYPE

FOCUS データソースでは、SEGTYPE 属性を使用してセグメントのキーフィールド、ソート順、親セグメントとの関係を記述します。

SEGTYPE 属性は、SUFFIX=FIX データソースにも使用され、データソースの論理的なキー順を指定します。SEGTYPE についての詳細は、55 ページの「 フィールドグループの記述 」 を参照してください。

構文　セグメントの記述

FOCUS データソースに使用する SEGTYPE 属性の構文は次のとおりです。

```
SEGTYPE = segtype
```

有効な値には、次のものがあります。

`SH[n]`
: セグメントの先頭の *n* フィールドの値に基づいて、セグメントのインスタンスが上位の値から下位の値にソートされることを示します。*n* には 1 から 99 までの任意の値を指定することができます。指定しない場合は、デフォルトの 1 に設定されます。

`S[n]`
: セグメントの先頭の *n* フィールドの値に基づいて、セグメントのインスタンスが下位の値から上位の値にソートされることを示します。*n* には 1 から 255 までの任意の値を指定することができます。指定しない場合は、デフォルトの 1 に設定されます。

`S0`
: セグメントにキーフィールドは存在せず、ソートされていないことを示します。新しいインスタンスはセグメント連鎖の末尾に追加されます。すべての検索は現在位置から開始されます。

  S0 セグメントは、入力したテキストを入力した順に検索するアプリケーションでテキストを格納する場所としてよく使用されます。このアプリケーションは特定のインスタンスを検索する必要はありません。

`(blank)`
: セグメントにキーフィールドは存在せず、ソートされていないことを示します。新しいインスタンスはセグメント連鎖の末尾に追加されます。すべての検索はセグメント連鎖の先頭から開始されます。

  SEGTYPE=blank セグメントは、セグメントインスタンスが極端に少ない場合、およびセグメントに格納された情報にキーとしての役割を担えるフィールドが含まれていない場合によく使用されます。

  なお、ルートセグメントを SEGTYPE=blank セグメントにすることはできません。

`U`
: 親と 1 対 1 の関係を持つユニークセグメントであることを示します。なお、SEGTYPE=U が指定されたユニークセグメントは下位セグメントを持つことはできません。

KM
: マスターファイルで定義した静的 JOIN でデータソースに結合されたクロスリファレンスセグメントであり、ホストセグメントと 1 対 n の関係があることを示します。マスターファイルで定義する JOIN についての詳細は、271 ページの 「マスターファイルでの JOIN の定義」 を参照してください。親子関係のポインタはデータソースに格納されます。

KU
: マスターファイルで定義した静的 JOIN でデータソースに結合されたクロスリファレンスセグメントであり、ホストセグメントと 1 対 1 の関係 (ユニークセグメント) があることを示します。マスターファイルで定義する JOIN についての詳細は、271 ページの 「マスターファイルでの JOIN の定義」を参照してください。親子関係のポインタはデータソースに格納されます。

DKM
: マスターファイルで定義した動的 JOIN でデータソースに結合されたクロスリファレンスセグメントであり、ホストセグメントと 1 対 n の関係があることを示します。マスターファイルで定義する JOIN についての詳細は、271 ページの 「マスターファイルでの JOIN の定義」 を参照してください。親子ポインタは実行時に決定されるため、再構築せずに新しいインスタンスを追加することができます。

DKU
: マスターファイルで定義した動的 JOIN でデータソースに結合されたクロスリファレンスセグメントであり、ホストセグメントと 1 対 1 の関係 (ユニークセグメント) があることを示します。マスターファイルで定義する JOIN についての詳細は、271 ページの 「マスターファイルでの JOIN の定義」を参照してください。親子ポインタは実行時に決定されるため、再構築せずに新しいインスタンスを追加することができます。

KL
: マスターファイル定義の JOIN でクロスリファレンスデータソースの KM、KU、DKM、DKU セグメントの下位として記述されたセグメントで、ホストセグメントと 1 対 n の関係あることを示します。

KLU
: マスターファイル定義の JOIN でクロスリファレンスデータソースの KM、KU、DKM、DKU セグメントの下位として記述されたセグメントで、ホストセグメントと 1 対 1 の関係 (ユニークセグメント) にあることを示します。

参照　SEGTYPE 使用時の注意

FOCUS データソースで SEGTYPE 属性を使用する場合には次の規則が適用されます。

- **エイリアス**　SEGTYPE 属性のエイリアスはありません。

- **変更** SEGTYPE の S[n] または SH[n] を S0 または b に変更するか、キーフィールド数を増やすことができます。SEGTYPE にその他の変更を加える場合は、REBUILD 機能を使用する必要があります。

キーフィールドの記述

SEGTYPE 属性を使用して、セグメント内でキーフィールドとなるフィールドを記述します。これらのフィールドの値は、セグメントインスタンスの配列順序を決定します。キーフィールドは、セグメントの先頭フィールドにする必要があります。下位の値から上位の値にソートするセグメント (SEGTYPE = S*n*) では最大で 255 のキー、上位の値から下位の値にソートするセグメント (SEGTYPE = SH*n*) では最大で 99 のキーを指定することができます。検索効率を最大にするには、指定するキーの数を各レコードが一意となるのに必要な数に抑えることをお勧めします。また、キーフィールドが存在しないセグメントとして指定することもできます (SEGTYPE = S0 および SEGTYPE = blank)。

注意：テキストフィールドをキーフィールドとして使用することはできません。

ソート順の記述

キーフィールドを持つセグメントには、SEGTYPE 属性を使用してセグメントのソート順を記述します。セグメントのインスタンスは 2 つの方法でソートすることができます。

- **下位から上位へ** SEGTYPE S*n* (*n* はキー数) を指定すると、インスタンスは先頭の *n* フィールドの連結値を使用して下位の値から上位の値にソートされます。
- **上位から下位へ** SEGTYPE S*n* (*n* はキー数) を指定すると、インスタンスは先頭の *n* フィールドの連結値を使用して上位の値から下位の値にソートされます。

  セグメント連鎖では、最新の日付を持つセグメントインスタンスが最初に検索されるため、日付フィールドを持つセグメントには上位から下位へのソートがよく使用されます。

ソート順の理解

ここでは、セグメントに部署コードおよび社員の名前 (姓) で構成された次のようなフィールドについて考察します。

06345 Jones	19887 Smith	19887 Frank	23455 Walsh	21334 Brown

SEGTYPE を S1 に設定した場合、部署コードがキーになります。ここでは、この例の後半で S2 セグメントのポイントを説明するために 2 つのレコードに重複したキー値を使用していますが、S1 および SH1 セグメントには重複したキー値を使用しないことをお勧めします。セグメントインスタンスは次のようにソートされます。

06345 Jones	19887 Smith	19887 Frank	21334 Brown	23455 Walsh

SEGTYPE を S1 に指定した状態で姓フィールドを部署コードの前に移動すると、姓フィールドがキーになります。セグメントインスタンスは次のようにソートされます。

Brown 21334	Frank 19887	Jones 06345	Smith 19887	Walsh 23455

部署コードを先頭フィールドに配置した状態で SEGYPE を S2 に設定すると、次のように最初に部署コードを基準にセグメントがソートされ、続いて姓フィールドを基準にソートされます。

06345 Jones	19887 Frank	19887 Smith	21334 Brown	23455 Walsh

セグメント関係の記述

SEGTYPE 属性は、特定のセグメントとその親セグメントとの関係を記述します。

- 通常、物理的な 1 対 1 の関係を指定するには、SEGTYPE を U に設定します。セグメントがマスターファイル定義の JOIN でクロスリファレンスセグメントの下位セグメントとして記述されている場合は SEGTYPE を JOIN 定義で KLU に設定します。
- 物理的な 1 対 n の関係を指定するには、SEGTYPE を S で始まる任意の有効値 (例、S0、SH*n*、S*n*) または blank に設定します。また、セグメントがマスターファイル定義の JOIN でクロスリファレンスセグメントの下位セグメントとして記述されている場合は SEGTYPE を JOIN 定義で KL に設定します。

- マスターファイルで定義した 1 対 1 の JOIN を指定するには、SEGTYPE を KU または DKU に設定します。詳細は、271 ページの 「 マスターファイルでの JOIN の定義 」 を参照してください。
- マスターファイルで定義した 1 対 1 の JOIN を指定するには、SEGTYPE を KM または DKM に設定します。詳細は、271 ページの 「 マスターファイルでの JOIN の定義 」 を参照してください。

別ロケーションへのセグメント格納 - LOCATION

デフォルト設定では、FOCUS データソースに存在するすべてのセグメントは 1 つの物理ファイルに格納されます。たとえば、EMPLOYEE データソースのすべてのセグメントは、「EMPLOYEE」というデータソースに格納されます。

LOCATION 属性を使用して、主データソースファイルとは別の物理ファイルに格納する 1 つまたは複数のセグメントを指定します。LOCATION ファイルは、「水平パーティション」とも呼ばれます。マスターファイルごとに最大で 64 個の LOCATION ファイルを使用することができます。ルートセグメントを除いて、1 つのセグメントに対して 1 つの LOCATION が使用可能です。この方法は、FOCUS データソースファイルの制限を超えるデータソースを作成する場合やセキュリティなどの理由でデータソースの一部を別のロケーションに格納する場合に役立ちます。

LOCATION 属性は、少なくとも 2 つの状況で使用されます。

- 物理ファイルの最大容量には制限があります。LOCATION 属性を使用して、ファイルの最大容量の条件を満たす範囲でデータソースを複数の物理ファイルに分割することにより、データソースの容量を増加することができます。複数のデータソースを JOIN 構造として構成した方が効率的な場合もあります。詳細は、271 ページの 「 マスターファイルでの JOIN の定義 」 を参照してください。
- 実際にはレポートに表示するセグメントのみが必要になるため、このデータのみを別の物理ファイルに格納しておくこともできます。別のデータソースにセグメントが格納され、参照されていないものは、別の媒体に保存して領域を節約したり、別のセキュリティ機能を実装したりすることができます。場合によっては、複数のセグメントを異なるデータソースに分割して、それぞれを異なるディスクドライブで使用することもできます。

データソースを分割する場合は、ファイルの管理上の注意が必要です。特に、データのバックアップなど、それぞれ別の手順でデータソースを分割する場合は要注意です。たとえば、関係付けられた 2 つのデータソースを火曜日に半分、木曜日に残り半分をバックアップするという方法で FOCUS 構造を再構築する場合、データの不一致を検知する手段はありません。

構文　別ロケーションへのセグメント格納

```
LOCATION = filename [,DATASET = physical_filename]
```

説明

`filename`
　セグメントを格納するファイルの ddname です。

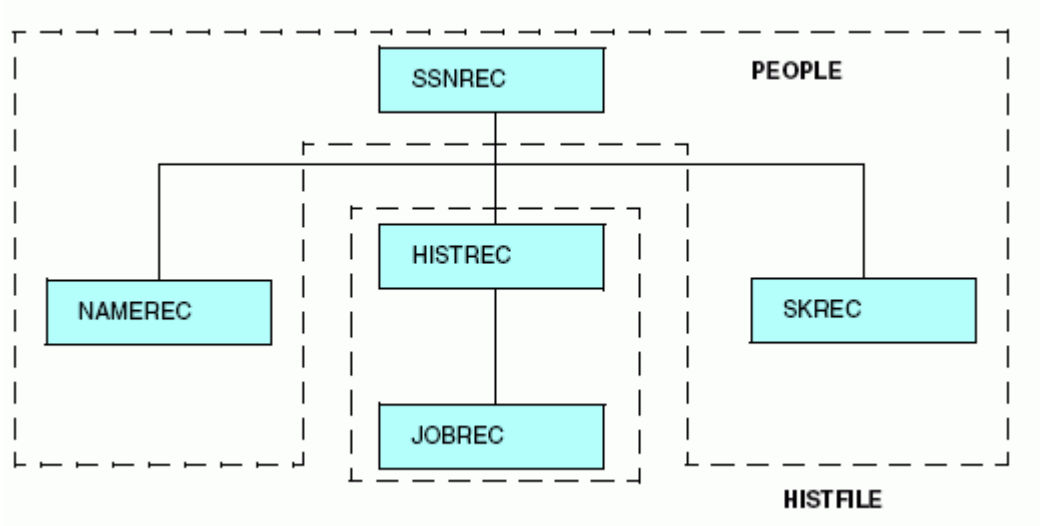
`physical_filename`
　使用するプラットフォームに依存するデータソースの物理名です。

例　セグメントのロケーションの指定

次の例は、LOCATION 属性の使用方法を示しています。

```
FILENAME = PEOPLE, SUFFIX = FOC, $
SEGNAME = SSNREC,  SEGTYPE = S1, $
 FIELD = SSN,     ALIAS = SOCSEG, USAGE = I9,  $
SEGNAME = NAMEREC, SEGTYPE = U, PARENT = SSNREC, $
 FIELD = LNAME,   ALIAS = LN,     USAGE = A25, $
SEGNAME = HISTREC, SEGTYPE = S1,PARENT = SSNREC, LOCATION = HISTFILE, $
FIELD = DATE,    ALIAS = DT,     USAGE = YMD, $
SEGNAME = JOBREC,  SEGTYPE = S1,PARENT = HISTREC,$
 FIELD = JOBCODE, ALIAS = JC,     USAGE = A3,  $
SEGNAME = SKREC,   SEGTYPE = S1,PARENT = SSNREC, $
 FIELD = SCODE,   ALIAS = SC,     USAGE = A3,  $
```

この記述は、下図のように 5 つのセグメントを 2 つの物理ファイルに分類します。



ここで、LOCATION 属性が指定されていない「SKREC」というセグメントは PEOPLE データソースに格納されています。セグメントに LOCATION 属性が指定されていない場合、そのセグメントはデフォルト設定で親セグメントと同一のファイルに配置されます。この例では、SKREC セグメントの宣言に LOCATION 属性を指定して、このセグメントを別のファイルに割り当てることができます。デフォルト設定を使用するのではなく、LOCATION 属性を指定することをお勧めします。

大規模なテキストフィールドの分割

テキストフィールドは、デフォルト設定でテキストフィールドを持たない 1 つの物理ファイルに格納されます。ただし、セグメントの場合と同様に、1 つのテキストフィールドを専用の物理ファイルに配置したり、複数のテキストフィールドを組み合わせて 1 つまたは複数の物理ファイルに配置したりすることができます。テキストフィールドを別のファイルに格納するには、フィールド記述に LOCATION 属性を指定します。

次の例では、DESCRIPTION のテキストは、「CRSEDESC」という別の物理ファイルに格納されています。

```
FIELD = DESCRIPTION, ALIAS = CDESC, USAGE = TX50, LOCATION = CRSEDESC ,$
```

複数のテキストフィールドがある場合、各フィールドをそれぞれ別のファイルに格納したり、複数のテキストフィールドを 1 つのファイルに格納したりすることができます。

次の例では、DESCRIPTION および TOPICS テキストフィールドは、「CRSEDESC」という 1 つの LOCATION ファイルに格納されています。また、PREREQUISITE テキストフィールドは、別の PREREQS ファイルに格納されています。

```
FIELD = DESCRIPTION , ALIAS = CDESC, USAGE = TX50, LOCATION = CRSEDESC,$
FIELD = PREREQUISITE, ALIAS = PREEQ, USAGE = TX50, LOCATION = PREREQS ,$
FIELD = TOPICS,       ALIAS =      , USAGE = TX50, LOCATION = CRSEDESC,$
```

セグメントの場合と同様に、LOCATION 属性はテキストフィールドが長い場合に使用することができます。ただし、LOCATION セグメントとは異なり、テキストフィールドの LOCATION ファイルは、リクエストの実行時にテキストフィールドが参照されていない場合でも正しい場所に配置しておく必要があります。

LOCATION 属性は、セグメントとテキストフィールドにそれぞれ独立して使用することができます。たとえば、LOCATION 属性をセグメントには使用せずに、テキストフィールドのみに使用することもできます。また、同一のマスターファイルでセグメントとテキストフィールドの両方に LOCATION 属性を使用することもできます。

注意：FOCUS マスターファイルでは、テキストフィールドのフィールド名の最大長さは 12 バイトです。XFOCUS マスターファイルでは、テキストフィールドのフィールド名には 12 バイトの制限は適用されません。ただし、両方のデータソースでこれらのフィールドのエイリアスの最大長は 66 バイトです。

セグメント、LOCATION ファイル、インデックス、テキストフィールドの個数制限

マスターファイルで定義するセグメントの最大数は 64 です。指定可能なロケーションセグメントおよびテキストの LOCATION ファイルの数には制限があります。この制限は、FOCUS および XFOCUS データソースの FDT (ファイルディレクトリテーブル) で許可されたエントリ数に基づいて決定されます。FDT には、データソース内のセグメント名、インデックスフィールド名、テキストフィールドの LOCATION ファイル名が格納されます。

参照　FOCUS または XFOCUS データソースの FDT エントリ

FDT には 189 件のエントリを含めることができ、その中の最大 64 件のエントリにセグメントおよび LOCATION ファイルを割り当てることができます。1 つの LOCATION ファイルが 1 つの FDT エントリとして数えられます。

次の計算式を使用して、データソースの LOCATION ファイルの最大数を決定します。

```
Available FDT entries = 189 - (Number of Segments + Number of Indexes)
Location files = min (64, Available FDT entries)
```

説明

`Location files`
LOCATION セグメントおよびテキスト LOCATION ファイルの最大数です (最大 64)。

`Number of Segments`
マスターファイル内のセグメントの数です。

`Number of Indexes`
インデックスフィールドの数です。

たとえば、2 つのインデックスフィールドを持つ 10 セグメントのデータソースでは、最大で 52 の LOCATION セグメントおよびテキストフィールドの LOCATION ファイルを指定することができます (189 - (10 + 2))。計算式の結果は 177 になります。ただし、テキストの LOCATION ファイルの最大数は常に 64 です。

注意：LOCATION 属性にテキストフィールドを指定する場合は、テキストの LOCATION ファイルの数に主ファイルも数えられます。

セグメント物理ファイル名の指定 - DATASET

FOCUS マスターファイルのファイルレベルで DATASET 属性を指定する以外に、この属性をセグメントレベルで指定して、LOCATION セグメントまたはフィールドが再定義されたクロスリファレンスセグメントの物理ファイル名を指定することができます。

DATASET 属性をファイルレベルで指定する方法についての詳細は、23 ページの 「 データソースの識別 」 を参照してください。

注意

- USE コマンドを発行したり、ファイルの割り当てを明示したりすると、DATASET 属性が無視されることを告げる警告メッセージが生成されます。
- 同一のマスターファイルで ACCESSFILE 属性と DATASET 属性の両方を使用することはできません。

DATASET 属性は、LOCATION セグメント、クロスリファレンスセグメントのいずれかに指定する必要があります。クロスリファレンスセグメントに指定する際の注意は以下のとおりです。

- クロスリファレンスフィールドにフィールド宣言が指定されている場合、クロスリファレンスマスターファイルは読み取られず、そのマスターファイルで指定した DATASET 属性は呼び出すことができないため、物理ファイルの指定には DATASET 属性を使用する方法のみが有効です。
- クロスリファレンスフィールドにフィールド宣言が指定されていない場合は、DATASET 属性はクロスリファレンスマスターファイルのファイルレベルに配置します。この場合、ホストマスターファイルのセグメントレベルに指定した DATASET の値と、クロスリファレンスマスターファイルのファイルレベルで指定した DATASET の値が一致していないと、競合が発生して (FOC1998) メッセージが表示されます。

データソースが FOCUS Database Server で管理される場合、TABLE リクエストを処理する際に、FOCUS Database Server がマスターファイルを読み取らないため、マスターファイルに DATASET 属性を使用すると、サーバ側では DATASET 属性は無視されます。

マスターファイルで指定する DATASET 属性は、最も低い優先度で処理されます。

- DATASET 属性は、ユーザの明示的な割り当てで上書きされます。
- FOCUS データソースに USE コマンドを使用した場合、DATASET 属性および明示的な割り当てが上書きされます。

注意：DATASET 属性の割り当てが有効な場合、この割り当てを明示的な割り当てコマンドで上書きするには、CHECK FILE コマンドを発行する必要があります。CHECK FILE コマンドは、DATASET 属性の割り当てを解除します。

構文　セグメントレベルでの DATASET 属性の使用

LOCATION セグメントに使用する場合は次のように記述します。

```
SEGNAME=segname, SEGTYPE=segtype, PARENT=parent, LOCATION=filename,
                 DATASET='physical_filename [ON sinkname]',$
```

クロスリファレンスセグメントに使用する場合は次のように記述します。

```
SEGNAME=segname, SEGTYPE=segtype, PARENT=parent, [CRSEGNAME=crsegname,]
[CRKEY=crkey,] CRFILE=crfile, DATASET='filename1 [ON sinkname]',
  FIELD=...
```

説明

`filename`
LOCATION ファイルの論理名です。

`physical_filename`
プラットフォームに依存するデータソースの物理名です。

`sinkname`
データソースが FOCUS Database Server に存在することを示します。この属性は、FOCUS データソースで有効です。

UNIX での構文は次のとおりです。

```
{DATASET|DATA}='path/filename'
```

WINDOWS での構文は次のとおりです。

```
{DATASET|DATA}='path¥filename'
```

例　DATASET 属性を使用したセグメントの割り当て

UNIX/USS の場合

```
FILE = ...
 SEGNAME=BDSEG,SEGTYPE=KU,CRSEGNAME=IDSEG,CRKEY=PRODMGR,
  CRFILE=PERSFILE,DATASET='/u2/prod/user1/idseg.foc',
   FIELD=NAME,ALIAS=FNAME,FORMAT=A12,INDEX=I, $
```

Windows の場合

```
FILE = ...
 SEGNAME=BDSEG,SEGTYPE=KU,CRSEGNAME=IDSEG,CRKEY=PRODMGR,
  CRFILE=PERSFILE,DATASET='¥u2¥prod¥user1¥idseg.foc',
   FIELD=NAME,ALIAS=FNAME,FORMAT=A12,INDEX=I, $
```

FOCUS セグメントのタイムスタンプ - AUTODATE

FOCUS データソースのセグメントには、セグメントが最後に変更された日付と時間を記録するタイムスタンプフィールドを追加することができます。このフィールドには任意の名前を割り当てることができますが、USAGE フォーマットは AUTODATE に設定する必要があります。このフィールドには、セグメントインスタンスが更新されるたびに時間が記録されます。タイムスタンプは HYYMDS フォーマットで格納されますが、日付時間関数のいずれかを使用してレポート作成用にタイムスタンプを操作することができます。

FOCUS データソースの各セグメントで、USAGE = AUTODATE を指定したフィールドを定義することができます。AUTODATE フィールドをセグメントのキーフィールドの一部にすることはできません。そのため、SEGTYPE = S2 の場合は、AUTODATE フィールドをセグメント内の先頭または 2 番目のフィールドとして定義することはできません。

AUTODATE フォーマットの指定はマスターファイルで定義する実フィールドでのみサポートされ、DEFINE および COMPUTE コマンドまたはマスターファイルに記述する一時項目 (DEFINE) ではサポートされません。ただし、DEFINE または COMPUTE コマンドを使用して、AUTODATE フィールドに格納された値を操作したり、フォーマットを再設定したりできます。

セグメントに AUTODATE フィールドを追加した後、REBUILD を使用してデータソースを再構築する必要があります。REBUILD を実行してもフィールドにはタイムスタンプは追加されません。セグメントインスタンスが挿入または更新されるまでフィールドには値は存在しません。

ユーザが作成したプロシジャが AUTODATE フィールドを更新する場合、セグメントインスタンスがデータソースに書き込まれる際にユーザ指定の値が上書きされます。値が上書きされたことをユーザに通知するメッセージは生成されません。

AUTODATE フィールドにインデックスを付けることができます。ただし、セグメントインスタンスが変更されるたびにインデックスを更新するには、インデックスのオーバヘッドがかかるため、インデックスが必要であるかどうかは確認することを推奨します。

AUTODATE フィールドを含む HOLD ファイルを作成する場合、AUTODATE フィールドは HYYMDS フォーマットの日付時間フィールドとして HOLD ファイルに継承されます。

構文　セグメントの AUTODATE フィールドの定義

```
FIELDNAME = fieldname, ALIAS = alias, {USAGE|FORMAT} = AUTODATE ,$
```

説明

`fieldname`
　有効な任意のフィールド名です。

`alias`
　有効な任意のエイリアスです。

例　AUTODATE フィールドの定義

EMPLOYEE データソースの REBUILD DUMP および EMPDATE データソースへの REBUID LOAD を実行して EMPDATE データソースを作成します。EMPDATE のマスターファイルは EMPLOYEE のマスターファイルと同一ですが、FILENAME が変更され、DATECHK フィールドが追加されます。

```
FILENAME=EMPDATE, SUFFIX=FOC
SEGNAME=EMPINFO,  SEGTYPE=S1
 FIELDNAME=EMP_ID,     ALIAS=EID,  FORMAT=A9,        $
 FIELDNAME=DATECHK,    ALIAS=DATE, USAGE=AUTODATE,   $
 FIELDNAME=LAST_NAME, ALIAS=LN,    FORMAT=A15,       $
   .
   .
   .
```

次のプロシジャを実行して、タイムスタンプ情報を EMPDATE に追加します。

```
SET TESTDATE = 20010715
TABLE FILE EMPLOYEE
PRINT EMP_ID CURR_SAL
ON TABLE HOLD
END
MODIFY FILE EMPDATE
FIXFORM FROM HOLD
MATCH EMP_ID
ON MATCH COMPUTE CURR_SAL = CURR_SAL + 10;
ON MATCH UPDATE CURR_SAL
ON NOMATCH REJECT
DATA ON HOLD
END
```

続いて、DEFINE または COMPUTE コマンドで AUTODATE フィールドを参照するか、表示コマンドを使用してこのフィールドを表示します。次のリクエストは、2001 年 7 月 31 日と DATECHK フィールドの日付の日数差を計算します。

```
DEFINE FILE EMPLOYEE
DATE_NOW/HYYMD = DT(20010731);
DIFF_DAYS/D12.2 =  HDIFF(DATE_NOW, DATECHK, 'DAY', 'D12.2');
END
TABLE FILE EMPDATE
PRINT DATECHK DIFF_DAYS
WHERE LAST_NAME EQ 'BANNING'
END
```

出力結果は次のとおりです。

```
DATECHK                      DIFF_DAYS
-------                      ---------
2001/07/15 15:10:37          16.00
```

参照　AUTODATE 使用時の注意

- PRINT * および PRINT.SEG. *fld* は AUTODATE フィールドを表示します。
- FOCUS Database Server は、サーバの日付時間を使用して、セグメントごとに AUTODATE フィールドを更新します。
- AUTODATE フィールドでの DBA 使用は許可されますが、セグメント内で制限されていないフィールドを更新すると、システムが AUTODATE フィールドを更新します。
- AUTODATE フィールドでは MISSING、ACCEPT、HELPMESSAGE 属性はサポートされません。

GROUP 属性

グループを使用すると、1 つまたは複数の連続したフィールドに使いやすい別名を割り当てることができます。グループはフィールドの順序を再定義をしますが、グループを格納するための領域は必要ありません。

フィールドのグループを使用すると、次の操作を実行することができます。

- クロスリファレンスデータソースの複数フィールドに結合する。別のフィールドをグループとして再定義し、グループキーのインデックスを付け、グループフィールドに結合します。

- グループの各コンポーネントでの等価テストに基づいてレコードを選択する際に、TABLE リクエストにインデックスビューを使用する。TABLE はこのタイプの検索に 1 つのインデックスのみを使用するため、グループにインデックスを付けて検索効率を向上することができます。

グループフィールドとフォーマットの記述

コンポーネントフィールドには、文字または数値データを含めることができます。ただし、グループフィールドには必ず文字フォーマットを使用します。グループフィールドの長さは、コンポーネントフィールドの実長の合計にする必要があります。たとえば、レポートに表示する文字数を設定する USAGE フォーマットに関係なく、整数フィールドは常に実際の長さの 4 バイトになります。

構文　グループフィールドとそのフォーマットの記述

```
GROUP=groupname, [ALIAS=groupalias,] USAGE=An, [FIELDTYPE=I,] $
 FIELDNAME=field1, ALIAS=alias1, USAGE=fmt1,$
 FIELDNAME=field2, ALIAS=alias2, USAGE=fmt2,$
 .
 .
 .
 FIELDNAME=fieldn, ALIAS=aliasn, USAGE=fmtn,$
```

説明

`groupname`
グループ名です。

`groupalias`
オプションとして指定するグループのエイリアスです。

`An`
グループフィールドのフォーマットです。この長さは、コンポーネントフィールドの内部長の合計です。

- タイプ I のフィールドの内部長は 4 です。
- タイプ F のフィールドの内部長は 4 です。
- USAGE フォーマットの P15、P16、またはそれ以下のタイプ P のフィールドの内部長は 8 で、USAGE フォーマットの P17 以上の長さのフィールドの内部長は 16 です。
- タイプ D のフィールドの内部長は 8 です。
- タイプ D のフィールドの内部長は、USAGE フォーマット内の文字数と同数です。

タイプ A 以外のフォーマットでグループフィールドを記述してもエラーメッセージは生成されませんが、予期しない結果が発生する可能性があるため、使用しないことをお勧めします。

`field1, ..., fieldn`
グループ内のコンポーネントフィールドです。

`alias1, ..., aliasn`
グループ内のコンポーネントフィールドのエイリアスです。

`fmt1, ..., fmtn`
グループ内のコンポーネントフィールドの USAGE フォーマットです。

`FIELDTYPE=I`
グループに別のインデックスを作成します。

参照　FOCUS データソースグループフィールド使用時の注意

- グループのいずれかのコンポーネントが数値または日付フォーマットの場合、グループフィールドは正しく表示されません。ただし、グループフィールドの値および個々のフィールドは正しく表示されます。TABLE リクエストにこのタイプのグループを使用して、複数のコンポーネントフィールドでインデックス検索を実行したり、複数のコンポーネントフィールドに結合したりします。
- マスターファイルで指定したキーフィールド以外のグループフィールド定義を、他に影響を与えることなくいつでも追加または削除することができます。
- グループフィールドでは MISSING 属性はサポートされません。
- 数値コンポーネントを含むグループフィールドを選択条件に使用する場合は、各コンポーネントの値をスラッシュ (/) で区切ります。ただし、グループコンポーネントの値にスラッシュ (/) が含まれる場合は、区切り文字としてスラッシュ (/) を使用することができないため、この値に対するテストは実行されません。

  なお、コンポーネントフィールドに末尾のブランクが含まれる場合、スラッシュ (/) を使用するとこのブランクを計上する必要がなくなるため値の指定が容易になります。

  グループフィールドの選択条件では、先頭コンポーネントに続くすべてのグループコンポーネントの任意の文字の組み合わせを受容するマスクのみがサポートされます。たとえば、FULL_NAME グループが LAST_NAME および FIRST_NAME で構成されている場合、次のマスクを使用することができます。

  ```
  WHERE FULL_NAME EQ '$R$$$$*'
  ```

- 文字コンポーネントのみを含むグループフィールドを選択条件に使用する場合は、各コンポーネントの値を区切るスラッシュ (/) はオプションとして使用します。

- グループフォーマットでは日付表示オプションがサポートされます。
- グループとグループコンポーネントの両方にインデックスが指定されている場合は、グループコンポーネントを一致させずにグループレベルで MATCH を実行することができます。
- グループおよびそのグループの後に続くフィールドがキーの一部の場合、SEGTYPE 属性に指定する数には、グループフィールド、コンポーネントフィールド、グループに続くフィールドのすべてを計上する必要があります。たとえば、グループキーにコンポーネントが 2 つあり、グループの後にキーの一部にもなっている別のフィールドが続く場合、SEGTYPE を S4 にする必要があります。

```
GROUP = ...,                ,$
  FIELDNAME = FIELDG1, ... ,$
  FIELDNAME = FIELDG2, ... ,$
  FIELDNAME = FIELD3, ...  ,$
```

例　文字グループフィールドの表示

次のグループフィールド定義では、グループ長は LAST_NAME と FIRST_NAME のフィールドの長さを合計して 25 になります。

```
FILENAME=EMPLOYEE, SUFFIX=FOC
SEGNAME=EMPINFO,  SEGTYPE=S1
 FIELDNAME=EMP_ID,        ALIAS=EID,       FORMAT=A9,       $
 GROUP=FULL_NAME,                 ,       FORMAT=A25,      $
FIELDNAME=LAST_NAME,    ALIAS=LN,       FORMAT=A15,      $
  FIELDNAME=FIRST_NAME,   ALIAS=FN,       FORMAT=A10,      $
```

両方のコンポーネントフィールドが文字であるため、グループフィールドの WHERE テストには、各コンポーネント値を区別するスラッシュ (/) は必要ありません。

```
TABLE FILE EMPLOYEE
PRINT EMP_ID HIRE_DATE
BY FULL_NAME
WHERE FULL_NAME GT 'CROSSBARBARA'
END
```

出力結果は次のとおりです。

```
FULL_NAME                    EMP_ID     HIRE_DATE
---------                    ------     ---------
GREENSPAN      MARY          543729165   82/04/01
IRVING         JOAN          123764317   82/01/04
JONES          DIANE         117593129   82/05/01
MCCOY          JOHN          219984371   81/07/01
MCKNIGHT       ROGER         451123478   82/02/02
ROMANS         ANTHONY       126724188   82/07/01
SMITH          MARY          112847612   81/07/01
SMITH          RICHARD       119265415   82/01/04
STEVENS        ALFRED        071382660   80/06/02
```

例　整数コンポーネントのグループフィールドの選択

次のグループフィールド定義では、グループ長は LAST_NAME、FIRST_NAME と HIRE_DATE のフィールドの長さを合計して 29 になります。HIRE_DATE は整数フィールドのため、内部長は 4 です。

```
FILENAME=EMPLOYEE, SUFFIX=FOC
SEGNAME=EMPINFO,  SEGTYPE=S1
 FIELDNAME=EMP_ID,       ALIAS=EID,      FORMAT=A9,    $
 GROUP=FULL_NAME,                ,       FORMAT=A29,   $
FIELDNAME=LAST_NAME,    ALIAS=LN,      FORMAT=A15,   $
  FIELDNAME=FIRST_NAME,   ALIAS=FN,      FORMAT=A10,   $
  FIELDNAME=HIRE_DATE,    ALIAS=HDT,     FORMAT=I6YMD, $
```

次のリクエストでは、WHERE テストの各コンポーネントフィールドの値をスラッシュ (/) で区切る必要があります。整数コンポーネントが正しく表示されないため、このリクエストではグループフィールドは表示されません。

```
TABLE FILE EMPGROUP
PRINT EMP_ID LAST_NAME FIRST_NAME HIRE_DATE
BY FULL_NAME NOPRINT
WHERE FULL_NAME GT 'CROSS/BARBARA/811102'
END
```

出力結果は次のとおりです。

```
EMP_ID     LAST_NAME       FIRST_NAME  HIRE_DATE
------     ---------       ----------  ---------
543729165  GREENSPAN       MARY         82/04/01
123764317  IRVING          JOAN         82/01/04
117593129  JONES           DIANE        82/05/01
219984371  MCCOY           JOHN         81/07/01
451123478  MCKNIGHT        ROGER        82/02/02
126724188  ROMANS          ANTHONY      82/07/01
112847612  SMITH           MARY         81/07/01
119265415  SMITH           RICHARD      82/01/04
071382660  STEVENS         ALFRED       80/06/02
```

要素群のグループフィールド記述

マスターファイルの GROUP 宣言を使用して、複数のフィールドを 1 つのグループとして記述します。複数のフィールドを 1 つのグループ名として参照すると、レポート作成が容易になる場合があります。

従来の方法で GROUP フィールドを記述する場合は、GROUP フィールドの USAGE および ACTUAL フォーマットの両方が文字フォーマットであるにも関わらず、グループの USAGE フォーマットの長さを各コンポーネントの長さの合計として計算する必要があります。ここで、整数または単精度フィールドは 4 バイト、倍精度フィールドは 8 バイト、パック 10 進数フィールドはそのサイズにより 8 バイトまたは 16 バイトとして計上する必要があります。

GROUP ELEMENTS オプションを使用すると、USAGE および ACTUAL フォーマットを使用せずに一連の要素をグループとして記述できるため、これらの長さを合計する必要がなくなります。

構文　　要素群のグループフィールド記述

```
GROUP=group1, ALIAS=g1alias,ELEMENTS=n1,$
   FIELDNAME=field11, ALIAS=alias11, USAGE=ufmt11, ACTUAL=afmt11, $
   .
   .
   .
   FIELDNAME=field1h, ALIAS=alias1h, USAGE=ufmt1h, ACTUAL=afmt1h, $
GROUP=group2,ALIAS=g2alias,ELEMENTS=n2,$
   FIELDNAME=field21, ALIAS=alias21, USAGE=ufmt21, ACTUAL=afmt21, $
   .
   .
   .
   FIELDNAME=field2k, ALIAS=alias2k, USAGE=ufmt2k, ACTUAL=afmt2k, $
```

説明

`group1, group2`
　フィールドのグループに割り当てらる有効な名前です。グループ名は、フィールド名の規則に準拠します。

`n1, n2`
　グループを構成する要素 (フィールドおよびグループ) の数です。グループが別のグループ内で定義されている場合、サブグループ (すべての要素を含む) は親グループの要素の 1 つとして計上されます。

`field11, field2k`
　有効なフィールド名です。

`alias11, alias2k`
　有効なエイリアス名です。

`ufmt11, ufmt2k`
　各フィールドの USAGE フォーマットです。

`afmt11, afmt2k`
　各フィールドの ACTUAL フォーマットです。

参照 GROUP ELEMENTS 属性使用時の注意

- ELEMENTS 属性を使用する場合は、GROUP フィールド宣言でグループ名および要素数のみを指定する必要があります。
  - グループ宣言で ELEMENTS 属性を指定せずに USAGE および ACTUAL を指定すると、USAGE および ACTUAL 属性が正しくない場合でもこれらの属性が受容されます。
  - グループ宣言に ELEMENTS 属性とともに USAGE および ACTUAL 属性を指定した場合は、ELEMENTS 属性が優先されます。
- 各サブグループは 1 つの要素として計上されます。サブグループ内の個々のフィールドおよびサブグループは、親グループの要素数には計上されません。

例 ELEMENTS によるグループの宣言

次のマスターファイルでは、GRP2 は FIELDA および FIELDB フィールドの 2 つの要素で構成されています。GRP1 は、GRP2 および FIELDC フィールドの 2 つの要素で構成されています。FIELDD フィールドはグループには属しません。

```
FILENAME=XYZ     , SUFFIX=FIX     , $
  SEGMENT=XYZ, SEGTYPE=S2, $
GROUP=GRP1,ALIAS=CCR,ELEMENTS=2,$
   GROUP=GRP2,ALIAS=CC,ELEMENTS=2,$
    FIELDNAME=FIELDA, ALIAS=E01, USAGE=A10, ACTUAL=A10, $
    FIELDNAME=FIELDB, ALIAS=E02, USAGE=A16, ACTUAL=A16, $
    FIELDNAME=FIELDC, ALIAS=E03, USAGE=P27, ACTUAL=A07, $
    FIELDNAME=FIELDD, ALIAS=E04, USAGE=D7, ACTUAL=A07, $
```

下表は、これらのフィールドのオフセットおよびフォーマットを示しています。

フィールド番号	フィールド名	オフセット	USAGE	ACTUAL
1	GRP1	0	A42 - FIELDC (P27) の 16 バイトをサポートします。	A33
2	GRP2	0	A26	A26
3	FIELDA	0	A10	A10
4	FIELDB	10	A16	A16
5	FIELDC	26	P27	A7

フィールド番号	フィールド名	オフセット	USAGE	ACTUAL
6	FIELDD	42	D7	A7

ACCEPT 属性

データソースのすべてのタイプにこの属性を使用する方法についての詳細は、97 ページの「 フィールドの記述 」 を参照してください。ただし、ACCEPT には FOCUS データソースにのみ使用可能な「FIND」という特別なオプションがあります。FIND を使用すると、入力データが別のフィールドに格納された値と照合されます。

構文　データ確認の指定

```
ACCEPT = list
ACCEPT = range
ACCEPT = FIND (sourcefield [AS targetfield] IN file)
```

説明

`list`
許容値のリストです。詳細は、97 ページの 「 フィールドの記述 」 を参照してください。

`range`
許容値の範囲を設定します。詳細は、97 ページの 「 フィールドの記述 」 を参照してください。

`FIND`
入力データを FOCUS データソースのインデックス値と照合します。

`sourcefield`
ACCEPT 属性を適用するフィールド名、または同一セグメントまたはそのセグメントへのパスに存在するその他の任意のフィールド名です。エイリアスや短縮名ではなく、実際のフィールド名を指定する必要があります。

`targetfield`
許容データ値を含むフィールドの名前です。このフィールドはインデックスフィールドである必要があります。

`file`
許容値のインデックスフィールドが格納されたデータソースを記述するファイルの名前です。

INDEX 属性

フィールド宣言に INDEX 属性またはその FIELDTYPE のエイリアスを含めることにより、フィールド値にインデックスを付けます。インデックスは、内部的に格納、保守されるデータ値とロケーションのテーブルのため、検索が高速に実行されます。次の操作を行うには、インデックスを作成する必要があります。

- 等価性に基づいて 2 つのセグメントを結合する。結合した FOCUS データソースのクロスリファレンスフィールドにはインデックスを付ける必要があります。マスターファイルで定義する JOIN についての詳細は、238 ページの 「 セグメントの記述 」 を参照してください。また、JOIN コマンドを使用して定義する JOIN についての詳細は、『TIBCO WebFOCUS Language リファレンス』を参照してください。
- 代替ビューを作成して検索を高速にする。詳細は、55 ページの 「 フィールドグループの記述 」 を参照してください。
- 指定したフィールド値に基づいてセグメントの選択および検索を高速に実行する。詳細は、『TIBCO WebFOCUS Language リファレンス』を参照してください。

構文　フィールドインデックスの指定

マスターファイルで指定する INDEX 属性の構文は次のとおりです。

```
INDEX = I or FIELDTYPE = I
```

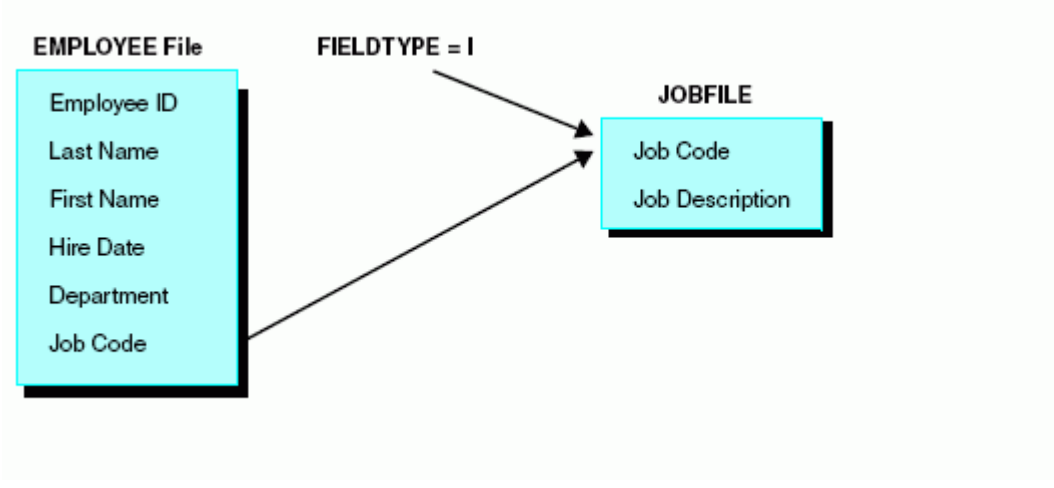
テキストフィールドにインデックスを付けることはできません。FOCUS データソースでは、インデックスフィールドのフィールド名の最大長は 12 バイトです。XFOCUS データソースでは、インデックスフィールドのフィールド名の最大長は 66 バイトです。

例　JOBCODE フィールドのインデックスの指定

```
FIELDNAME = JOBCODE, ALIAS = CJC, FORMAT = A3, INDEX = I, $
```

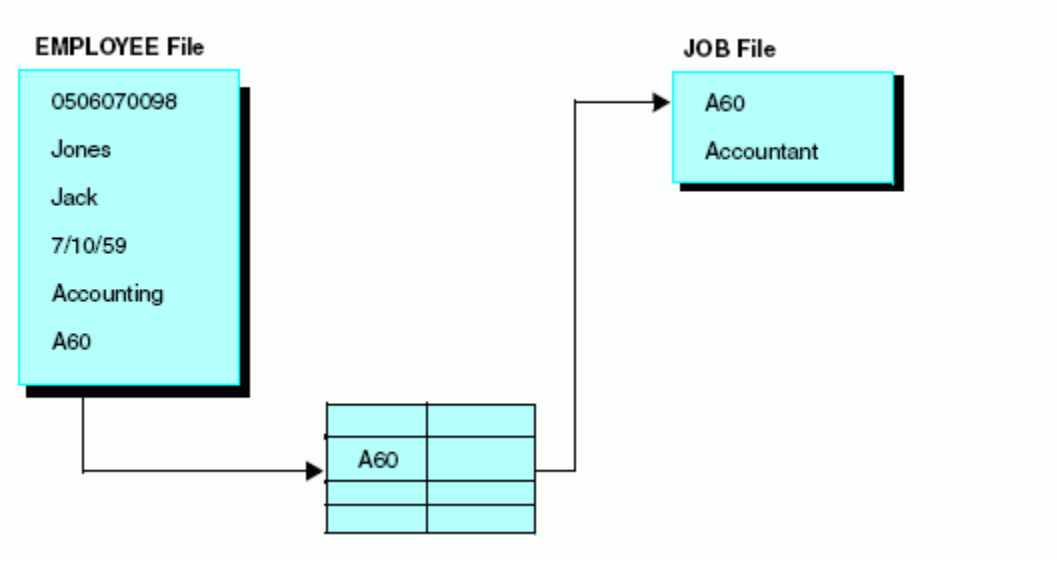
JOIN と INDEX 属性

静的クロスリファレンス、動的クロスリファレンス、equijoin を使用したクロスリファレンスとしてセグメントを使用するには、そのセグメント内の少なくとも 1 つのフィールドがインデックスフィールドである必要があります。このフィールドは「クロスリファレンスフィールド」と呼ばれ、ホストデータソースとフィールド値を共有します。下図のように、インデックスフィールドはクロスリファレンスセグメントのみで必要になります。



他のデータソースは、このインデックスを利用してセグメントを特定して使用します。各セグメントには任意の数のインデックスフィールドを指定することができますが、データソース内で指定可能なインデックスフィールド数を制限することをお勧めします。

下図のように、EMPLOYEE データソースの JOBCODE フィールド値は、JOBFILE データソースの JOBCODE フィールドのインデックスを使用して JOBFILE データソースの JOBCODE フィールド値に一致させることができます。



インデックスは、FOCUS データソースの一部として格納、保守されます。インデックスは、クロスリファレンス機能の処理で重要な役割を果たします。インデックスを使用することにより、任意の数の外部データソースを特定してセグメントを共有することができます。既存データソースのインデックスフィールドと共通のデータ項目を持つ新しいデータソースは、いつでも追加することができます。

参照　INDEX 属性使用時の注意

INDEX 属性の使用時には次の規則が適用されます。

- **エイリアス**　INDEX 属性のエイリアスは FIELDTYPE です。
- **変更**　INDEX 属性をフィールドから削除したり、この属性にブランクと同等の値を割り当てたりすると、インデックスの指定は解除されます。インデックスを使用する必要がなくなった場合は、INDEX 属性を削除した後、REBUILD 機能の REORG オプションを使用して、インデックスが占有していた領域を復元します。REBUILD についての詳細は、367 ページの「 データソースの作成と再構築 」 を参照してください。

たとえば大規模なデータをデータソースに高速でロードする場合など、インデックスの使用を一時的に中断する場合は、データをロードする前に INDEX 属性を一度削除し、次にこの属性を復元してから、REBUILD コマンドの INDEX オプションを使用してインデックスを作成することができます。この方法は、「データソースのポストインデックスの作成」と呼ばれます。

データソースを作成し、レコードを格納した後でフィールドにインデックスを追加するには、REBUILD 機能の INDEX オプションを使用します。

- **最大数** インデックス、テキストフィールド、セグメントの合計が 189 を超えることはできません。このうち、セグメントおよびテキスト LOCATION ファイルの最大数は 64 です。

FORMAT および MISSING 属性 - 内部格納の要件

アプリケーション開発者が各データタイプおよび値がどのように表示され、内部的にどのように格納されるかを知ることは重要です。

- 整数フィールドは、4 バイトのバイナリ整数として格納されます。
- 倍精度浮動小数点数フィールドは、倍精度浮動小数点数 (8 バイト) の数値として格納されます。
- 単精度浮動小数点数フィールドは、単精度浮動小数点数 (4 バイト) の数値として格納されます。
- パック 10 進数フィールドは、最大で 31 桁の 10 進数を表す 8 または 16 バイトの数値として格納されます。
- 日付フィールドは、指定した日付と日付フォーマットの基準日の差を表す 4 バイトのバイナリ整数として格納されます。基準日は、日付フォーマットにより 1900 年 12 月 31 日、1901 年 1 月 1 日のいずれかです。
- 日付時間フィールドは、時間構成要素にミリ秒が指定されているかどうかに応じて、8 または 10 バイトで格納されます。
- 文字フィールドは、指定されたバイト数の文字として格納されます。
- 可変長文字フィールドは、指定されたバイト数に加え、フィールドに格納された文字列の長さを指定するための 2 バイトが合算されたバイト数の文字として格納されます。
- ミッシング値は、内部的にはフラグで表現されます。

FOCUS 分割データソースの記述

FOCUS データソースには、最大で 1022 の物理ファイルを含めることができます。水平パーティションは、データソース構造全体の 1 つまたは複数のセグメントを水平に分割したものです。ただし、1 つの FOCUS データソースに関連する物理ファイルの数は、パーティション数と LOCATION ファイル数の合計になります。その合計は 1022 以下にする必要があります。将来的に FOCUS データソースのサイズが大きくなる場合は、アプリケーションの要件に基づいてデータソースを分割することができます。

高度な分割

FOCUS データソースでは、高度な分割がサポートされます。これは、垂直方向に分割した各パーティションに、特定のデータ値または値の範囲の完全なデータソース構造が含まれることを意味します。高度な分割を使用すると、データを最大で 1022 の物理ファイルに分割できるだけでなく、アクセスファイルを作成して WHERE 条件で実際のデータ値を各パーティションに記述することもできます。レポートリクエストを処理する際に、リクエストの選択条件がアクセスファイルの WHERE 条件と比較され、データの取得に必要なパーティションが特定されます。

分割の効果が期待できるアプリケーションを特定する場合は、USE コマンドを採用してデータソースを連結しているアプリケーションや、データ値または値の範囲に基づいてデータを分割できるアプリケーション (例、月単位や部署別に格納されたデータ) を選択します。高度な分割は、高度な USE コマンドのように機能します。高度な分割はレポートリクエストを処理する際に読み取り先のパーティションを特定しますが、USE コマンドはリストのすべてのファイルを読み取ります。この高度な機能で入出力が減少するため、パフォーマンスが向上します。

分割機能を使用するには、次の手順を実行する必要があります。

- マスターファイルを編集して ACCESSFILE 属性を追加する。
- テキストエディタを使用してアクセスファイルを作成する。

複数のパーティションの連結は、レポートの作成用としてのみサポートされます。ロードまたは再構築の処理は、物理パーティションごとに実行する必要があります。次の場合は、ロード処理で参照する各パーティションのマスターファイルを別々に作成することも、分割したデータソースに対するレポート作成用のマスターファイルを 1 つだけ使用することもできます。

- 明示的な割り当てコマンドを発行して、マスターファイルをパーティションごとに順にリンクさせる。
- パーティションごとにロード処理を順に実行する。

注意：ユーザが操作しなくても、レポートリクエストは必要なすべてのパーティションを自動的に読み取ります。

FOCUS マスターファイルでのアクセスファイルの指定

分割機能を使用するには、マスターファイルを編集して、ACCESSFILE 属性を追加する必要があります。

アクセスファイルの名前はマスターファイルの名前と同一にし、マスターファイルの ACCESS= 属性で指定する必要があります。

構文　FOCUS データソースのアクセスファイル指定

```
FILENAME=filename, SUFFIX=FOC, ACCESS[FILE]=accessfile,
.
.
.
```

説明

`filename`
分割データソースのファイル名です。

`accessfile`
アクセスファイルの名前です。この名前には、マスターファイルと同一の名前を指定する必要があります。

参照

分割 FOCUS データソース使用時の注意

- レポートを作成する場合に限り、単一リクエストで複数のパーティションを連結することができます。分割データソースに対して REBUID を実行する場合は、1 回ごとに特定のパーティションを明示的に割り当ててから REBUILD を実行する必要があります。
- データソースの割り当てに適用される優先順位は次のとおりです。
  - USE コマンドが有効な場合は、このコマンドが最優先されます。このコマンドは、データソースのアクセスファイルまたは明示的な割り当てを上書きします。
  - アクセスファイルは、データソースの明示的な割り当てを上書きします。
- DATASET 属性は、ACCESSFILE 属性と同一のマスターファイルに使用することはできません。
- データソース書き替えコマンド (例、REBUILD) は、アクセスファイルを使用しません。データソース書き替えコマンドを指定した ACCESSFILE 属性を含むマスターファイルを使用すると、次の警告メッセージが表示されます。

  ```
  (FOC1968) マスター %1 内のアクセスファイル情報は無視されます。
  ```

  このメッセージは HTML ページではなく、「ソース表示」に表示される場合があります。
- CREATE FILE コマンドは、作成するデータソースの動的割り当てを自動的に発行します。この割り当ては、ACCESSFILE 属性より優先されます。CREATE ファイルコマンドを発行後に ACCESSFILE 属性を使用するには、最初にこの自動割り当てを解除する必要があります。
- コマンドのタイプをデータソースの読み取りから書き込みに変更した場合、またはその逆の場合、マスターファイルが再解析されます。
- クロスリファレンスマスターファイルに ACCESSFILE 属性が含まれている場合、ホストマスターファイルでクロスリファレンスフィールドの名前を変更することはできません。

FOCUS アクセスファイル属性

すべてのリクエストには必ずマスターファイル名を指定します。データソースにアクセスする場合、このマスターファイルが読み取られ、マスターファイル内の宣言が使用されます。マスターファイルに ACCESS= 属性が含まれている場合、アクセスファイルが読み取られ、正しいデータソースが特定されます。アクセスファイルには、マスターファイルと同一の名前が指定されている必要があります。マスターファイルの ACCESS= 属性で指定された名前のアクセスファイルが存在しない場合、リクエストはマスターファイルのみを使用して処理されます。

参照 FOCUS アクセスファイルのアクセスファイル属性

FOCUS アクセスファイルでは、1 つのマスターファイルに対応するファイルおよび MDI を記述します。そのマスターファイル名は、アクセスファイル名と一致する必要があります。

属性と値の組み合わせはすべて等号 (=) で区切り、宣言内での各組み合わせはカンマ (,) で区切ります。各宣言は、カンマとドル記号の組み合わせ (,$) で終了します。

1. 各アクセスファイルは、対応するマスターファイルの名前を指定する宣言で開始します。
2. 次に、物理ファイルの位置を記述する DATA 宣言を続けます。物理ファイルが分割されている場合は、複数の DATA 宣言を記述します。

   物理ファイルが高度な分割で作成され、各パーティションに含めるデータ値が式によって記述されている場合は、その式を指定する WHERE 句を DATA 宣言に追加します。
3. データソースに LOCATION セグメントが含まれている場合、LOCATION 宣言で各 LOCATION セグメント名を指定します。この宣言に対応する DATA 宣言で、物理 LOCATION ファイルが指定されます。
4. データソースに MDI が含まれている場合、アクセスファイルの MDI 宣言で MDI およびそのターゲットセグメントを指定し、次に MDI のディメンション名を指定する宣言、物理 MDI ファイルを指定する MDIDATA 宣言を順に続けます。MDI が分割されている場合は、その MDI に対して複数の MDIDATA 宣言を記述します。

構文 FOCUS アクセスファイルの作成

マスターファイル宣言

```
MASTER=mastername,$
```

説明

`mastername`
このアクセスファイルが関連付けられているマスターファイルの名前です。マスターファイルの ACCESS=filename 属性で指定された値と同一の値を指定します。この属性により、アクセスファイルが存在すること、およびアクセスファイル名が特定されます。

```
DATA=file_specification,
   [WHERE= expression; ,]$
[DATA=file_specification,
   [WHERE= expression; ,]$ ...]
```

説明

`file_specification`
ファイルの場所を指定します。これは完全なファイル名の指定です。1 つのアクセスファイルに最大で 1022 個の DATA 宣言 (パーティション) を含めることができます。高度な分割は、WHERE 句に基づく機能です。式はセミコロン (;) で終了し、宣言全体はカンマとドル記号の組み合わせ (,$) で終了します。次のタイプの WHERE 式がサポートされます。

```
WHERE= field  operator value1 [ OR  value2...]; ,$

WHERE= field FROM value1 TO value2 [AND FROM value3 TO value4];,$
```

AND 演算子を使用して複数の式を組み合わせることができます。

ロケーションファイル宣言

```
LOCATION=location_segment_name,
  DATA=location_segment_file_specification,$
```

説明

`location_segment_name`
ロケーションファイルに格納されているセグメントの名前です。

`location_segment_file_specification`
セグメントが格納されている物理ファイルの完全なファイル名です。

MDI 宣言

```
MDI=mdiname, TARGET_OF = segname,$
   DIM = [filename.]fieldname [, MAXVALUES = n] [, WITHIN = dimname1],$
   [DIM = [filename.]fieldname [, MAXVALUES = n] [, WITHIN = dimname1] ,
$ ...]
 MDIDATA=mdi_file_specification,$
 [MDIDATA=mdi_file_specification,$ ...]
```

説明

`mdiname`
MDI の名前です。

`segname`
ターゲットセグメントの名前です。

`filename`
MDI ディメンションが存在するファイルの名前です。

`fieldname`
MDI のディメンションとして使用されるフィールドの名前です。

`n`
ディメンション値の種類数です。MDI の作成時に、実際のディメンション値は長さが 1、2、4 バイトのいずれかの整数値に変換され、その値がインデックスリーフに格納されます。

`mdi_file_specification`
物理 MDI ファイルの完全修飾名です。MDI が分割されている場合、MDI の 1 つのパーティションの名前を指定します。1 つの MDI には最大で 250 個の MDIDATA 宣言 (パーティション) を含めることができます。1 つのアクセスファイルに含めることのできる MDI 数には制限がありません。

`dimname`
ディメンションの階層を定義します。このディメンションは *dimname* ディメンション内に定義されます。 たとえば、CITY WITHIN STATE のように指定します。

6

マスターファイルでの JOIN の定義

1 つ以上の共通フィールドを持つ 2 つのセグメントを結合して新しい関係を作成する方法について説明します。これらのデータ構造は物理的には結合されませんが、レポートの作成時にあたかも単一のデータ構造のように扱うことができます。ここでは、FOUCUS、固定フォーマット、シーケンシャルデータソースのマスターファイルで JOIN を定義する方法について説明します。その他のデータソースタイプのマスターファイルで JOIN の定義が可能かどうかについては、使用するデータアダプタのマニュアルを参照してください。

トピックス

JOIN のタイプ

次の方法で 2 つのデータソースを結合することができます。

- **JOIN コマンドで動的に結合**　作成した JOIN はセッション中に限り有効になり (またはセッション中に JOIN を明示的にクリアするまで)、両方のデータソースに存在するすべてのセグメントで構成された一時的なデータビューを作成します。また、JOIN コマンドを使用して任意のタイプの 2 つのデータソースを結合することができます。たとえば、FOCUS データソースを FOCUS 以外のデータソースに結合することも可能です。JOIN コマンドについての詳細は、『TIBCO WebFOCUS Language リファレンス』を参照してください。

- **マスターファイルで静的に結合**　この方法は JOIN 構造に頻繁にアクセスする場合に役立ちます。JOIN の実装に必要なリンク (ポインタ) 情報が永続的に格納されるため、リクエストでレコードを取得するたびに呼び出す必要がなく、時間を節約することができます。マスターファイルで定義する動的 JOIN と同様に、任意の時点で JOIN を使用し、指定したセグメントのみを取得します。詳細は、272 ページの 「 マスターファイルでの静的 JOIN の定義 - SEGTYPE = KU、KM 」 を参照してください。この方法は FOCUS データソースでのみサポートされます。
- **マスターファイルで動的に結合**　データソースを結合するたびに JOIN コマンドを発行する手間を省くとともに、JOIN 構造内で利用可能なセグメントを自由に選択することができます。詳細は、285 ページの 「 マスターファイルでの動的 JOIN の定義 - SEGTYPE = DKU、DKM 」 を参照してください。

ヒント：設計時にこれらの JOIN を効果的に活用するには、最初に動的 JOIN (JOIN コマンドの発行またはマスターファイルでの定義) を使用してデータベースのプロトタイプを設計します。その設計が安定していることを確認した後、頻繁に使用する動的 JOIN をマスターファイルで定義する静的 JOIN に変更します。静的 JOIN を使用することにより、データソースへのアクセスが高速化されます。静的 JOIN は、ターゲットデータソースおよびクロスリファレンスデータソースの内容が変更されない場合に使用します。動的 JOIN を静的 JOIN に変更するには、REBUILD 機能を使用します。

注意：マスターファイルで定義する JOIN は「クロスリファレンス」とも呼ばれます。

マスターファイルでの静的 JOIN の定義 - SEGTYPE = KU、KM

静的 JOIN を使用すると、異なる FOCUS データソースのセグメントを永続的に関係付けることができます。静的 JOIN は、ホストデータソースのマスターファイルで定義します。

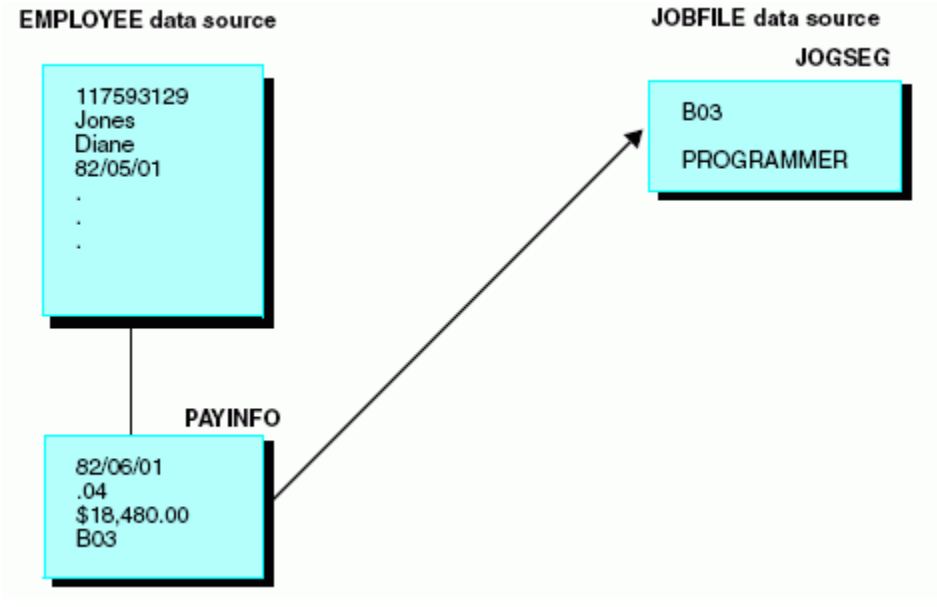
静的 JOIN には、1 対 1 の JOIN (SEGTYPE KU) と 1 対 n の JOIN (SEGTYPE KM) の 2 種類があります。

- 1 対 1 の JOIN (ユニーク JOIN) - JOIN 構造からレコードを取得する際に、ホストデータソースの各レコードインスタンスに対応するクロスリファレンスデータソースのレコードが多くても 1 つの場合はこの JOIN を指定します。
- 1 対 n の JOIN - クロスリファレンスデータソースから 1 つまたは複数のレコードインスタンスを取得する場合はこの JOIN を指定します。

ユニーク JOIN の記述 - SEGTYPE = KU

EMPLOYEE データソースには、PAYINFO セグメント内に「JOBCODE」というフィールドがあります。JOBCODE フィールドには、従業員の業務を識別するコードが格納されています。

業務の具体的な説明およびその他の関連情報は、「JOBFILE」という別のデータソースに格納されています。下図のように、JOBFILE データソースから業務説明を取得するには、EMPLOYEE データソースの JOBCODE 値に対応するレコードの場所を特定します。

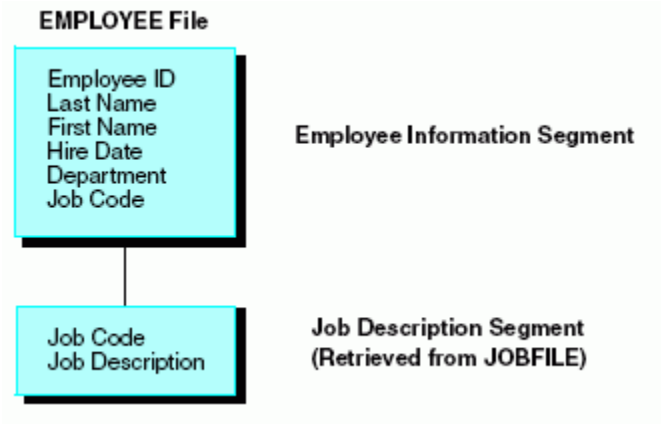


この場合、JOIN を使用することにより、EMPLOYEE データソースの各レコードに業務説明を入力したり、それを修正したりする必要がなくなります。その代わりに、JOBFILE データソースを使用して、有効なすべての業務説明を保守することができます。業務の説明に加える変更は JOBFILE データソースで 1 回だけ行い、その変更は結合された EMPLOYEE データソースのすべてのレコードに反映されます。

業務コードと業務説明の関係が変わる可能性は低いため、この場合は静的 JOIN として結合を実装する方が効率的です。

従業員情報および業務説明はそれぞれ別のデータソースに格納されていますが、レポートの作成時には JOBFILE データソースの業務説明のセグメントがあたかも EMPLOYEE データソースに格納されているように扱われます。JOIN 構造は JOBFILE データソースの実際の構造には影響しません。

EMPLOYEE データソースは次のように表現されます。



構文　静的ユニーク JOIN の指定

```
SEGNAME = segname, SEGTYPE = KU, PARENT = parent,
CRFILE = db_name, CRKEY = field, [CRSEGNAME = crsegname,]
[DATASET = physical_filename,] $
```

説明

`segname`

ホストデータソースでクロスリファレンスセグメントを認識するための名前です。クロスリファレンスデータソースの元のセグメント名などの有効なセグメント名を任意に指定することができます。

`parent`

ホストセグメントの名前です。

`db_name`

クロスリファレンスデータソースの名前です。この名前は、データソースを再構築せずに変更することができます。

`field`

ホストフィールドでクロスリファレンスフィールドで共通の名前です (フィールド名またはエイリアス)。ホストフィールドのフィールド名またはエイリアスは、クロスリファレンスフィールドのフィールド名と一致する必要があります。SEGTYPE が変更されていない場合は、データソースを再構築せずにフィールド名を変更することができます。

両方のフィールドのフォーマットタイプおよび長さが一致している必要があります。

クロスリファレンスフィールドはインデックスフィールドである必要があります (FIELDTYPE=I または INDEX=I)。

`crsegname`

クロスリファレンスセグメントの名前です。この名前を指定しない場合は、SEGNAME に割り当てられたデフォルト値が使用されます。クロスリファレンスデータソースにデータを入力した場合、この名前を変更するにはデータソースを再構築する必要があります。

`physical_filename`

CRFILE に使用するプラットフォームに依存したデータソースの物理名です。この名前はオプションとして指定します。

SEGTYPE 値の KU は Keyed Unique (キー付きユニーク) を表します。

例 静的ユニーク JOIN の作成

```
SEGNAME = JOBSEG, SEGTYPE = KU, PARENT = PAYINFO,
   CRFILE = JOBFILE, CRKEY = JOBCODE, $
```

次の記述は、EMPLOYEE マスターファイルから関連する部分を抜粋したものです。ただし、不要なフィールドおよびセグメントは省略されています。

```
FILENAME = EMPLOYEE, SUFFIX = FOC, $
SEGNAME = EMPINFO, SEGTYPE = S1, $
   .
   .
   .
SEGNAME = PAYINFO, SEGTYPE = SH1, PARENT = EMPINFO, $
   FIELDNAME = JOBCODE,    ALIAS = JBC, FORMAT = A3, $
   .
   .
   .
SEGNAME = JOBSEG, SEGTYPE = KU, PARENT = PAYINFO,   CRFILE = JOBFILE,
   CRKEY = JOBCODE, $
```

ここで指定するのは、クロスリファレンスセグメントの名前のみです。そのセグメントのフィールド名は、クロスリファレンスデータソースのマスターファイルですでに識別されています。この例では、JOBFILE がそれに該当します。この例では、CRSEGNAME 属性は SEGNAME 属性に割り当てられた名前と同一のため、この属性は省略されています。

クロスリファレンスデータソースのマスターファイルおよびそのデータソース自体は、ホストデータソースを使用する場合は常にアクセス可能な状態にしておく必要があります。クロスリファレンスデータソースにデータが存在する必要はありません。

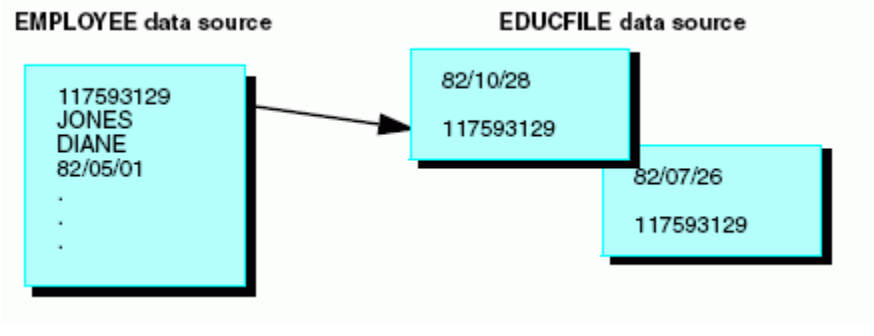
ユニーク JOIN によるデコード

デコードとは、コード (上記の例の業務コード) とコードの情報 (上記の例の業務説明) を一致させるプロセスです。それぞれのコードは一意の情報を持つため、コードと情報との JOIN は 1 対 1、つまりユニークな関係になります。例に示したように、デコードを実行するには、JOIN を使用することも、DEFINE コマンドで DECODE 関数を使用することもできます。詳細は、『TIBCO WebFOCUS Language リファレンス』を参照してください。コードの数が多い場合は、JOIN を使用する方法をお勧めします。

非ユニーク JOIN の記述 - SEGTYPE = KM

ホストセグメントの 1 つのインスタンスにクロスリファレンスセグメントの複数インスタンスが関係している場合は、1 対 n の JOIN、つまりユニーク JOIN を使用します。ここでは、上記の EMPLOYEE のデータソースを使用し、さらに従業員の研修実績を管理する「EDUCFILE」という研修データソースが作成されている場合を想定します。このデータソースのセグメントの 1 つである ATTNDSEG には、各従業員が各研修を受講した日付が格納されています。このセグメントには、受講日でキーが付けられています。受講者を識別する EMP_ID フィールドには、EMPLOYEE データソースの EMPINFO セグメント内の EMP_ID フィールドと同一の ID 番号が格納されています。

従業員の研修実績を表示する場合は、EMPINFO セグメントの EMP_ID フィールドを ATTNDSEG セグメントの EMP_ID フィールドに結合することができます。この場合、特定の従業員 ID に関係するすべての研修受講インスタンスを取得するため、1 対 n の JOIN を使用する必要があります。



構文　静的複数 JOIN の指定

1 対 n の JOIN を記述する構文は、274 ページの 「 静的ユニーク JOIN の指定 」 の 1 対 1 の JOIN の構文に類似しています。ただし、次のように SEGTYPE 属性には「KM」(キー付き複合) という値を指定する必要があります。

```
SEGTYPE = KM
```

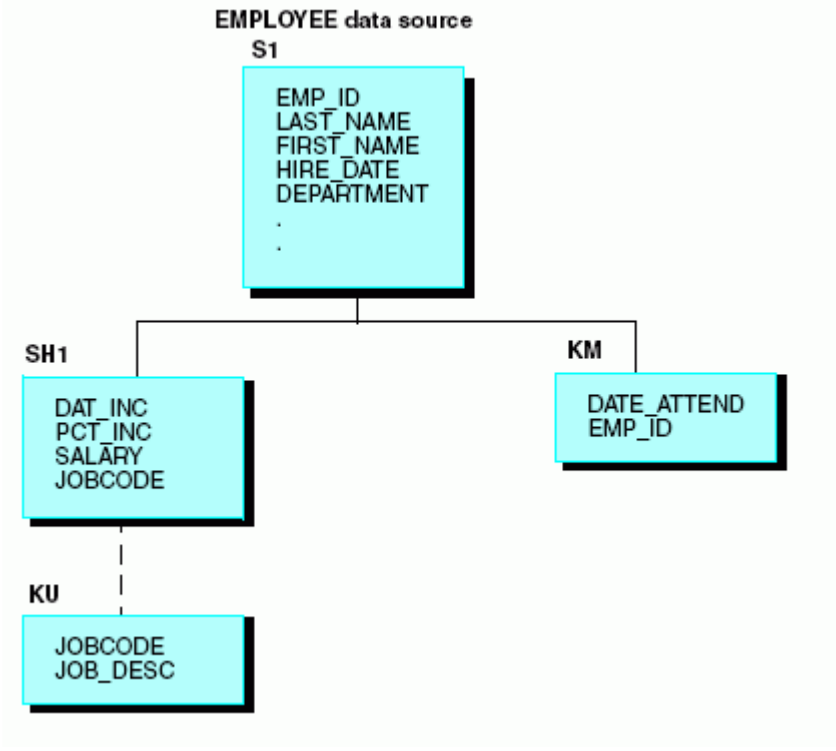
例　静的複数 JOIN の指定

```
SEGNAME = ATTNDSEG, SEGTYPE = KM, PARENT = EMPINFO,
   CRFILE = EDUCFILE, CRKEY = EMP_ID, $
```

次の記述は、EMPLOYEE マスターファイルから関連する部分を抜粋したものです。ただし、不要なフィールドおよびセグメントは省略されています。

```
FILENAME = EMPLOYEE, SUFFIX = FOC, $
SEGNAME = EMPINFO,  SEGTYPE = S1, $
   FIELDNAME = EMP_ID, ALIAS = EID, FORMAT = A9, $
   .
   .
   .
SEGNAME = PAYINFO,  SEGTYPE = SH1, PARENT = EMPINFO, $
   FIELDNAME = JOBCODE, ALIAS = JBC, FORMAT = A3, $
   .
   .
   .
SEGNAME = JOBSEG,   SEGTYPE = KU,  PARENT = PAYINFO, CRFILE = JOBFILE,
   CRKEY = JOBCODE, $
   .
   .
   .
SEGNAME = ATTNDSEG, SEGTYPE = KM,  PARENT = EMPINFO, CRFILE = EDUCFILE,
   CRKEY = EMP_ID, $
```

レポートリクエストでは、クロスリファレンスデータソースの JOBFILE および EDUCFILE は、EMPLOYEE データソースの一部のように扱われます。データ構造は次のようになります。



クロスリファレンス下位セグメントの使用 - SEGTYPE = KL、KLU

2 つのデータソースを結合した場合、クロスリファレンスセグメントのみにアクセスが限定されるのではなく、クロスリファレンスデータソースに存在するすべてのセグメントにアクセスすることができます。クロスリファレンスセグメント以外のセグメントは「リンクセグメント」とも呼ばれます。ホストデータソースから見ると、すべてのリンクセグメントは、クロスリファレンスセグメントの下位セグメントで、クロスリファレンスセグメントをルートとするクロスリファレンスデータソースの代替ビューのように見えます。リンクセグメントにアクセスするには、ホストデータソースのマスターファイルでリンクセグメントを宣言します。

構文　　クロスリファレンス下位セグメントの識別

```
SEGNAME = segname, SEGTYPE = {KL|KLU}, PARENT = parentname,
CRFILE = db_name, [CRSEGNAME = crsegname,]
[DATASET = physical_filename,] $
```

説明

`segname`

ホストデータソースのクロスリファレンスセグメントに割り当てられた名前です。

`KL`

ホストデータソースから見ると、このセグメントはクロスリファレンスデータソースの下位セグメントのように見え、その親セグメントと 1 対 n の関係になります。KL は Keyed through Linkage の略語です。

`KLU`

ホストデータソースから見ると、このセグメントはクロスリファレンスデータソースの下位セグメントのように見え、その親セグメントと 1 対 1 の関係になります。KLU は Keyed through Linkage, Unique の略語です。

`parentname`

ホストデータソースから見た場合のクロスリファレンスデータソースのセグメントの親セグメントの名前です。

`db_name`

クロスリファレンスデータソースの名前です。この名前は、データソースを再構築せずに変更することができます。

`crsegname`

クロスリファレンスセグメントの名前です。この名前を指定しない場合は、SEGNAME に割り当てられたデフォルト値が使用されます。

`physical_filename`

CRFILE に使用するプラットフォームに依存したデータソースの物理名です。この名前はオプションとして指定します。

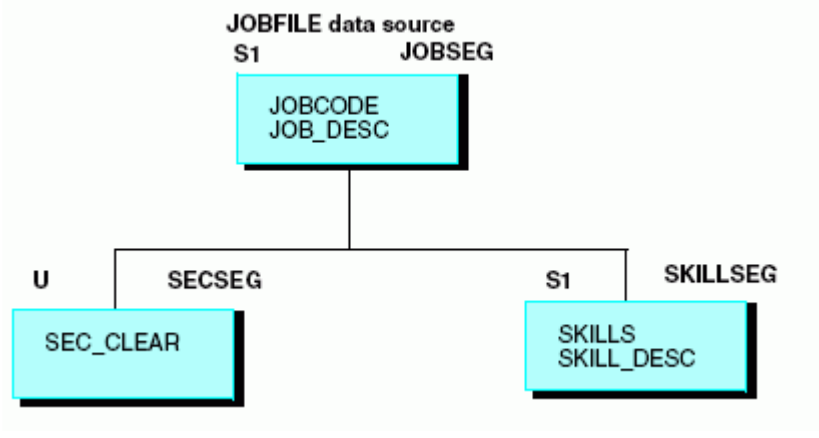
例 クロスリファレンス下位セグメントの識別

```
SEGNAME = SECSEG,   SEGTYPE = KLU, PARENT = JOBSEG, CRFILE = JOBFILE, $
SEGNAME = SKILLSEG, SEGTYPE = KL,  PARENT = JOBSEG, CRFILE = JOBFILE, $
```

CRKEY で識別する共通の JOIN フィールドはクロスリファレンスセグメントに対してのみ指定する必要があるため、リンクセグメントの宣言には CRKEY 属性は使用しません。

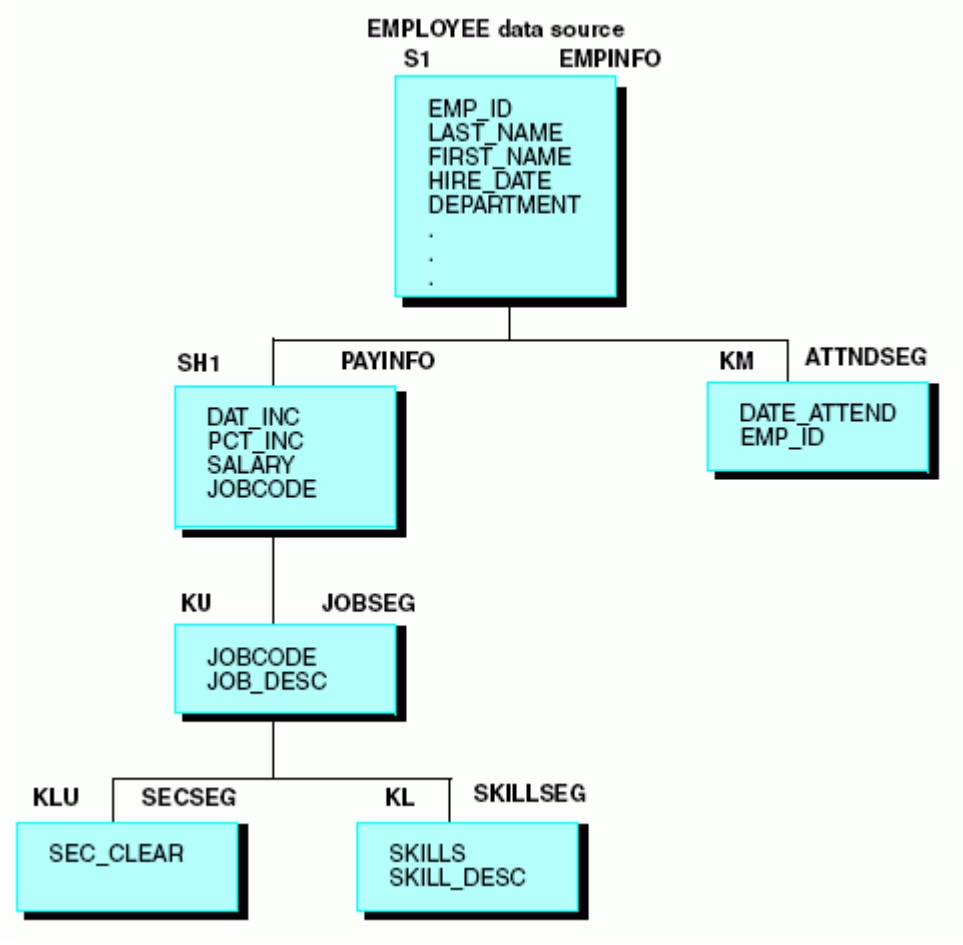
例　クロスリファレンス下位セグメントの使用

ここでは、上記の EMPLOYEE データソースについて考察します。JOBFILE は、複数セグメントのデータソースです。



ここでは、EMPLOYEE データソースのアプリケーションで、SECSEG セグメントに格納されたセキュリティ情報および SKILLSEG セグメントに格納された業務技能情報が必要な場合を想定します。JOIN を作成した後、ホストデータソースから見て 1 対 n の関係を表す SEGTYPE=KL を使用するか、1 対 n の関係を表す SEGTYPE=KLU を使用して、クロスリファレンスデータソース内の一部またはすべてのセグメントにアクセスすることができます。KL および KLU は、静的 JOIN (KM) と動的 JOIN (DKM) の両方のクロスリファレンスデータソース内の下位セグメントへのアクセスに使用することができます。

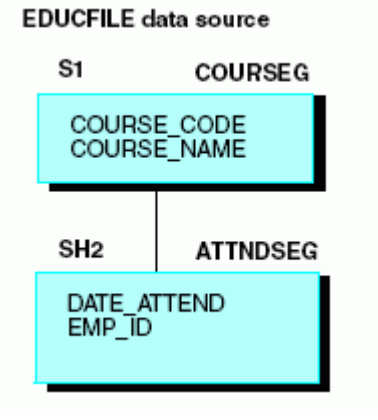
JOBFILE データソースから JOBSEG セグメントを取得する場合、EMPLOYEE マスターファイルで SEGTYPE=KL または KLU を指定して宣言したすべての JOBSEG の子セグメントも取得されます。



例　クロスリファレンス上位セグメントの使用

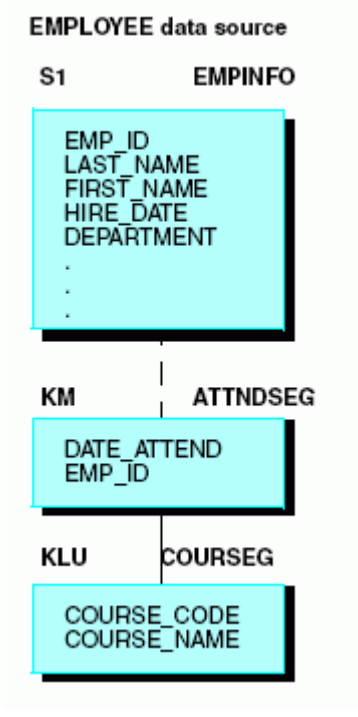
クロスリファレンスデータソースのすべてのセグメントを取得する場合、クロスリファレンスセグメントの下位セグメントだけでなく、上位セグメントも取得することができます。ホストデータソースから見た場合、1 つのセグメントには 1 つの親セグメントのみが存在するため (1 対 1 の関係)、上位セグメントはホストマスターファイルに SEGTYPE=KLU を指定して宣言する必要があります。

ここでは、上記の EDUCFILE データソースについて考察します。COURSEG はルートセグメントで、研修コースの情報が記述されています。ATTNDSEG は子セグメントで、従業員の受講情報が格納されています。



EMPLOYEE データソースの EMPINFO セグメントを EDUCFILE データソースの ATTNDSEG セグメントに結合する場合、COURSEG セグメントをリンクセグメントとして宣言することにより、そのセグメントの研修コースの説明にアクセスすることができます。

下図のように、この視点で見ると、COURSEG セグメントが ATTNDSEG セグメントの下位セグメントのように見えます。

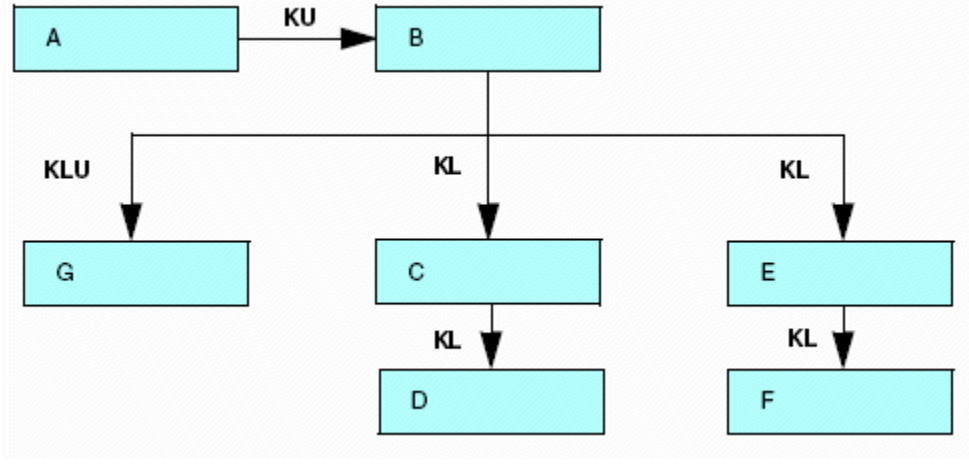


次の例は、EMPLOYEE マスターファイルから関連する部分を抜粋したものです。ただし、不要なフィールドおよびセグメントは省略されています。

```
FILENAME = EMPLOYEE, SUFFIX = FOC, $
SEGNAME = EMPINFO, SEGTYPE = S1, $
   FIELDNAME = EMP_ID, ALIAS = EID, FORMAT = A9, $
   .
   .
   .
SEGNAME = PAYINFO, SEGTYPE = SH1, PARENT = EMPINFO, $
   FIELDNAME = JOBCODE, ALIAS = JBC, FORMAT = A3, $
   .
   .
   .
SEGNAME = JOBSEG,  SEGTYPE = KU, PARENT = PAYINFO,  CRFILE = JOBFILE,
   CRKEY = JOBCODE, $
SEGNAME = SECSEG,  SEGTYPE = KLU,PARENT = JOBSEG,   CRFILE = JOBFILE, $
SEGNAME = SKILLSEG,SEGTYPE = KL, PARENT = JOBSEG,   CRFILE = JOBFILE, $
SEGNAME = ATTNDSEG,SEGTYPE = KM, PARENT = EMPINFO,  CRFILE = EDUCFILE,
   CRKEY = EMP_ID, $
SEGNAME = COURSEG, SEGTYPE = KLU,PARENT = ATTNDSEG, CRFILE = EDUCFILE, $
```

リンクセグメントの階層

1 つのリンクセグメント (KL) が、結果として別のリンクセグメント (KL) になることもあります。下図は、その関係を図示したものです。



矢印上の文字は、SEGTYPE を表しています。

ここで、G セグメントは、B のユニーク子セグメントになることも、B の親セグメントのユニーク子セグメントになることもできます。

マスターファイルでの動的 JOIN の定義 - SEGTYPE = DKU、DKM

SEGTYPE 属性を使用して、マスターファイルで動的 JOIN を定義することができます。マスターファイルで定義できる動的 JOIN には、1 対 1 の JOIN (SEGTYPE=KU) および 1 対 n の JOIN (SEGTYPE=KM) の 2 種類があります。

- 1 対 1 の JOIN (ユニーク JOIN) - 静的 JOIN の場合と同様に、JOIN 構造からレコードを取得する際に、ホストデータソースの各レコードインスタンスに対応するクロスリファレンスデータソースのレコードが多くても 1 つの場合はこの JOIN を指定します。
- 1 対 n の JOIN - クロスリファレンスデータソースから 1 つまたは複数のレコードインスタンスを取得する場合はこの JOIN を指定します。

静的 JOIN と動的 JOIN は、格納方法、速度、柔軟性の点で異なります。

- 静的 JOIN のリンク (ポインタ) は、1 回だけ取得されてホストデータソースに永続的に格納されます。また、必要に応じて自動更新されます。
- 動的 JOIN のリンクは、どこにも格納されないため、各レポートリクエストでレコードごとにリンクを取得する必要があります。

そのため、静的 JOIN は動的 JOIN に比べてより高速ですが、変更が難しくなります。静的 JOIN を再定義または削除する場合は、必ず REBUILD 機能を使用します。動的 JOIN の再定義または削除は、マスターファイルを編集することで、いつでも行うことができます。

構文　マスターファイルでの動的 JOIN の指定

マスターファイルで定義する動的 JOIN の指定方法は、274 ページの「 静的ユニーク JOIN の指定 」で説明した静的 JOIN の指定方法に類似しています。ただし、クロスリファレンスセグメントの SEGTYPE 属性には、1 対 1 の JOIN では DKU (Dynamic Keyed Unique) で、1 対 n の JOIN では DKM (Dynamic Keyed Multiple) を使用します。

以下はその例です。

```
SEGNAME = JOBSEG, SEGTYPE = DKU, PARENT = PAYINFO,
   CRFILE = JOBFILE, CRKEY = JOBCODE, $
```

動的 JOIN でのリンクセグメントの宣言は、静的 JOIN の場合と同様の方法で行います。両方の JOIN で、ユニークリンクセグメントには SEGTYPE=KLU、非ユニークリンクセグメントには SEGTYPE=KL をそれぞれ指定します。

例　マスターファイルでの動的 JOIN の指定

次のマスターファイルは、EMPLOYEE に使用するセグメントおよびそれに結合されたセグメントに関する部分を抜粋したものです。ただし、静的 JOIN が動的 JOIN に変更されており、不要なフィールドおよびセグメントは省略されています。

```
FILENAME = EMPLOYEE, SUFFIX = FOC, $
SEGNAME = EMPINFO, SEGTYPE = S1, $
   FIELDNAME = EMP_ID,    ALIAS = EID, FORMAT = A9, $
   .
   .
   .
SEGNAME = PAYINFO, SEGTYPE = SH1,  PARENT = EMPINFO, $
   FIELDNAME = JOBCODE,    ALIAS = JBC, FORMAT = A3, $
   .
   .
   .
SEGNAME = JOBSEG,  SEGTYPE = DKU,  PARENT = PAYINFO, CRFILE = JOBFILE,
   CRKEY = JOBCODE, $
SEGNAME = SECSEG,  SEGTYPE = KLU,  PARENT = JOBSEG,  CRFILE = JOBFILE, $
SEGNAME = SKILLSEG,SEGTYPE = KL,   PARENT = JOBSEG,  CRFILE = JOBFILE, $
SEGNAME = ATTNDSEG,SEGTYPE = DKM,  PARENT = EMPINFO, CRFILE = EDUCFILE,
   CRKEY = EMP_ID, $
SEGNAME = COURSEG, SEGTYPE = KLU,  PARENT = ATTNDSEG,CRFILE = EDUCFILE,$
```

マスターファイルでの条件付き JOIN の定義

条件付き (WHERE ベース) の JOIN は、特定の条件を定義し、その条件に基づいて 2 つのデータソースの行を関連付ける方法です。このタイプの埋め込み JOIN では、一方のデータソースのマスターファイルに、他方のデータソースのマスターファイルへのクロスリファレンスを記述します。FOCUS 以外のデータソースを関連付ける場合、条件付き埋め込み JOIN には複数テーブルのアクセスファイルは必要ありません。

構文　マスターファイルでの条件付き JOIN の定義

JOIN で指定された条件は、マスターファイル内の一時項目 (DEFINE) と見なされます。JOIN のタイプを指定するには、CRJOINTYPE 属性を使用します。

```
FILENAME=filename, SUFFIX=suffix   [,$]
 SEGNAME=file1, SEGTYPE= {S0|KL} [,CRFILE=crfile1] [,$]
  FIELD=name1,...,$
  .
  .
  .
SEGNAME=seg,  SEGTYPE=styp,  PARENT=parseg,
       CRFILE=xmfd,  [CRSEG=xseg,  ], [CRJOINTYPE = {INNER|LEFT_OUTER}]
       JOIN_WHERE=expression; [JOIN_WHERE=expression; ...] ,$
```

説明

`filename`

マスターファイルの名前です。

`suffix`

SUFFIX 値です。

`file1`

親セグメントの SEGNAME 値です。

`name`

任意のフィールド名です。

`seg`

結合されるセグメントのセグメント名です。クロスリファレンスマスターファイルに複数のセグメントが記述されている場合でも、このセグメントのみが JOIN に含まれます。

`styp`

結合されるセグメントのセグメントタイプです。マスターファイルの従来のクロスリファレンスと同様に、DKU、DKM、KU、KM のいずれかを指定することができます。

注意：ホストファイルとクロスリファレンスファイルの関係が 1 対 n の場合、ユニーク JOIN を指定すると、予期しない結果になります。

`parseg`

親セグメント名です。

`xmfd`

クロスリファレンスマスターファイルです。

`xseg`

seg で指定したセグメント名がクロスリファレンスマスターファイルの SEGNAME 値と異なる場合、クロスリファレンスセグメントを指定します。

`expression`

DEFINE FILE コマンドで有効な任意の式です。すべての式で参照されているフィールドすべてが単一パスに存在する必要があります。

例　マスターファイルでの条件付き JOIN の使用

次の EMPDATAJ1 マスターファイルでは、EMPDATA データソースと JOBHIST データソース間の条件付き JOIN が定義されています。

```
FILENAME=EMPDATA, SUFFIX=FOC  , DATASET=ibisamp/empdata.foc
SEGNAME=EMPDATA,  SEGTYPE=S1
 FIELDNAME=PIN,          ALIAS=ID,       FORMAT=A9,  INDEX=I,    $
 FIELDNAME=LASTNAME,     ALIAS=LN,       FORMAT=A15,             $
 FIELDNAME=FIRSTNAME,    ALIAS=FN,       FORMAT=A10,             $
 FIELDNAME=MIDINITIAL,   ALIAS=MI,       FORMAT=A1,              $
 FIELDNAME=DIV,          ALIAS=CDIV,     FORMAT=A4,              $
 FIELDNAME=DEPT,         ALIAS=CDEPT,    FORMAT=A20,             $
 FIELDNAME=JOBCLASS,     ALIAS=CJCLAS,   FORMAT=A8,              $
 FIELDNAME=TITLE,        ALIAS=CFUNC,    FORMAT=A20,             $
 FIELDNAME=SALARY,       ALIAS=CSAL,     FORMAT=D12.2M,          $
 FIELDNAME=HIREDATE,     ALIAS=HDAT,     FORMAT=YMD,             $
SEGNAME=JOBHIST, PARENT=EMPDATA,  SEGTYPE=DKM, CRFILE=ibisamp/jobhist,
CRJOINTYPE=INNER,$
   JOIN_WHERE = EMPDATA.JOBCLASS CONTAINS '257' AND JOBHIST.JOBCLASS
CONTAINS '019';$
```

次のリクエストでは、JOIN が定義された上記のマスターファイルを使用します。

```
TABLE FILE EMPDATAJ1
SUM SALARY TITLE AS 'Empdata Title' FUNCTITLE AS 'Jobhist Title'
BY LASTNAME
BY FIRSTNAME
BY EMPDATA.JOBCLASS AS 'Empdata Job'
BY JOBHIST.JOBCLASS AS 'Jobhist Job'
WHERE LASTNAME LT 'D'
ON TABLE SET PAGE NOPAGE

ON TABLE SET STYLE *
GRID=OFF,$
FONT=ARIAL, SIZE=8,$
TYPE=TITLE, STYLE=BOLD,$
END
```

下図では、JOIN 条件で指定されたとおり、EMPDATA セグメントのすべての JOBCLASS 値が「257」で始まり、JOBHIST セグメントのすべての JOBCLASS 値が「019」始まっています。

LASTNAME	FIRSTNAME	Empdata Job	Jobhist Job	SALARY	Empdata Title	Jobhist Title
ADAMS	RUTH	257PTB	019PTB	$250,000.00	MARKETING DIRECTOR	MNGR OF SALESPEOPLE
			019PUA	$250,000.00	MARKETING DIRECTOR	EXECUTIVE MANAGER
			019PUB	$62,500.00	MARKETING DIRECTOR	ASST VICE PRESIDENT
			019PVA	$62,500.00	MARKETING DIRECTOR	INTERNAL VICE PRES
			019PVB	$62,500.00	MARKETING DIRECTOR	EXEC VICE PRES
BELLA	MICHAEL	257PSB	019PTB	$250,000.00	INDUSTRIAL MARKETER	MNGR OF SALESPEOPLE
			019PUA	$250,000.00	INDUSTRIAL MARKETER	EXECUTIVE MANAGER
			019PUB	$62,500.00	INDUSTRIAL MARKETER	ASST VICE PRESIDENT
			019PVA	$62,500.00	INDUSTRIAL MARKETER	INTERNAL VICE PRES
			019PVB	$62,500.00	INDUSTRIAL MARKETER	EXEC VICE PRES
CHISOLM	HENRY	257PRB	019PTB	$172,000.00	SALES SPECIALIST	MNGR OF SALESPEOPLE
			019PUA	$172,000.00	SALES SPECIALIST	EXECUTIVE MANAGER
			019PUB	$43,000.00	SALES SPECIALIST	ASST VICE PRESIDENT
			019PVA	$43,000.00	SALES SPECIALIST	INTERNAL VICE PRES
			019PVB	$43,000.00	SALES SPECIALIST	EXEC VICE PRES
CONTI	MARSHALL	257PRA	019PTB	$129,200.00	ASST MKTG REP	MNGR OF SALESPEOPLE
			019PUA	$129,200.00	ASST MKTG REP	EXECUTIVE MANAGER
			019PUB	$32,300.00	ASST MKTG REP	ASST VICE PRESIDENT
			019PVA	$32,300.00	ASST MKTG REP	INTERNAL VICE PRES
			019PVB	$32,300.00	ASST MKTG REP	EXEC VICE PRES

静的 JOIN と動的 JOIN の比較

2 つの FOCUS データソースを結合する場合、2 つの JOIN タイプ (静的または動的) のいずれかを選択し、JOIN を定義する 2 つの方法 (マスターファイルでの定義または JOIN コマンドの発行) のいずれかを選択することができます。

- 静的 JOIN では、ホストセグメントインスタンスからクロスリファレンスセグメントインスタンスへのリンクは 1 回だけ作成され、永続的に格納されてホストデータソースで自動更新されます。
- 動的 JOIN では、このリンクは必要に応じて取得されます。このため、リンクの作成が 1 回だけの静的 JOIN の方が動的 JOIN より高速です。その反面、静的 JOIN の再定義および削除には REBUILD 機能を使用したり、ファイルを再ロードしたりする必要があるため柔軟性が低いといえます。

動的 JOIN では、JOIN コマンドを手軽に使用できるため、JOIN の指定を変更するたびにマスターファイルを変更する必要はありません。また、ホストデータソースから見た場合のリンクセグメントを個別に記述する必要はありません。一方、マスターファイルで定義する動的 JOIN では、不要なクロスリファレンスセグメントを省略することができます。

特定の JOIN をよく使用する場合は、その JOIN を静的 JOIN として実装しておくと効率的です。REBUILD 機能を使用して、静的 JOIN を動的 JOIN に変更したり、静的 JOIN を動的 JOIN に変更したりすることができます。

下表は、マスターファイルで定義する静的 JOIN と動的 JOIN、および JOIN コマンドを発行して定義する動的 JOIN の実装を比較したものです。

JOIN タイプ	長所	短所
マスターファイルでの静的 JOIN **(SEGTYPE = KU または KM)**	2 回目以降での処理が高速化されます。リンクの作成は 1 回だけです。 常に使用可能です。 特定のリンクセグメントを選択したり、他のリンクセグメントを省略したりすることができます。	データソースを作成する前に指定するか、REBUILD 機能を使用して再ロードする必要があります。 変更するには REBUILD 機能を使用する必要があります。 インスタンスごとに 4 バイトのファイルスペースが必要です。 ユーザは、リンクセグメントの関係 (KL、KLU) を指定する方法に精通している必要があります。
マスターファイルでの動的 JOIN **(SEGTYPE =DKU または DKM)**	任意の時点で指定することができます。 常に使用可能です。データソースで余分な領域を必要としません。 REBUILD 機能を使用せずに、必要に応じて変更および削除することができます。 特定のリンクセグメントを選択したり、他のリンクセグメントを省略したりすることができます。	処理が遅い。レポートリクエストごとに各レコードのリンクが取得されます。 ユーザは、リンクセグメントの関係 (KL、KLU) を指定する方法に精通している必要があります。

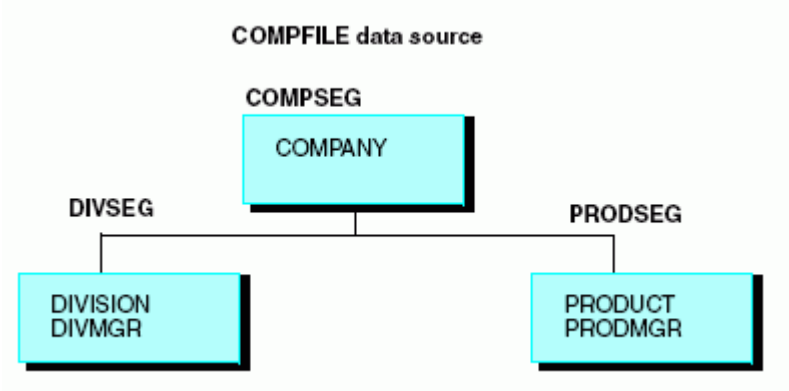
JOIN タイプ	長所	短所
JOIN コマンドでの動的 JOIN	任意の時点で指定することができます。 データソースで余分な領域を必要としません。 REBUILD 機能を使用せずに、必要に応じて変更および削除することができます。 ユーザがリンクセグメントの関係を指定する必要はありません。	処理が遅い。レポートリクエストごとに各レコードのリンクが取得されます。 JOIN コマンドは、JOIN を使用するセッションごとに発行する必要があります。 セグメントの使用の有無に関係なく、ターゲットデータソースのすべてのセグメントが含まれます。

複数ホストセグメントからの単一クロスリファレンスセグメントへの結合

状況によっては、ホストデータソースの複数の異なるセグメントから 1 つのクロスリファレンスセグメントに結合する場合があります。また、2 つの異なるホストデータソースから 1 つのクロスリファレンスセグメントに同時に結合する場合もあります。これらのデータ構造は、マスターファイルで定義する JOIN を使用して対処します。

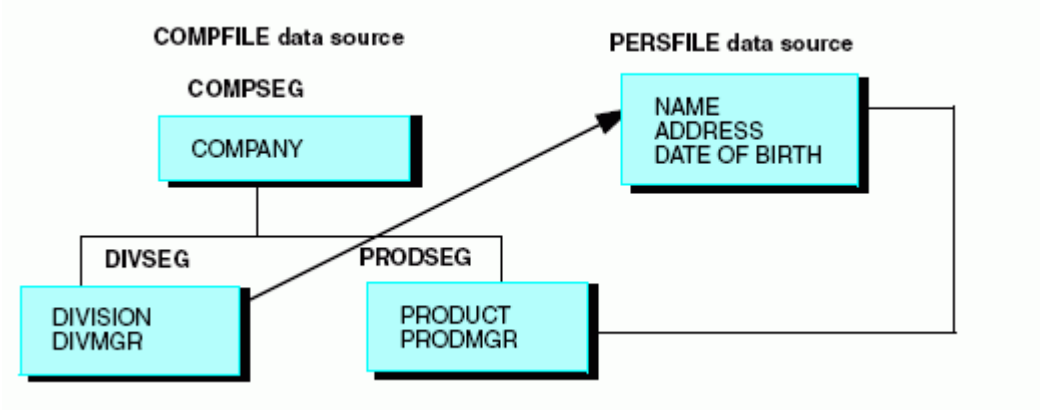
単一ホストデータソース複数セグメントからの結合

特定のアプリケーションでは、1 つのクロスリファレンスセグメントをデータソースの異なる場所で繰り返し使用する場合があります。ここでは、COMPFILE データソースを使用して、会社経営に関するデータを保守する場合を想定します。



DIVSEG セグメントには、各事業部のインスタンスが格納され、事業部名およびマネージャ名のフィールドが含まれています。同様に、PRODSEG セグメントには、各製品および製品マネージャの名前のインスタンスが格納されています。

下図のように、1 つの人事データソースから製品マネージャおよび事業部マネージャに関する個人情報を取得します。



このホストデータソースには 2 つのクロスリファレンスキー (PRODMGR および DIVMGR) が存在し、PRODSEG セグメントから取得した住所と生年月日、DIVSEG セグメントから取得した住所と生年月日をレポート内で区別する必要があるため、この情報を、標準のマスターファイルで定義した JOIN で取得することはできません。

ホストデータソースに存在する複数のセグメントから 1 つのクロスリファレンスセグメントに結合する JOIN を実装する方法があります。フィールド名のエイリアスでクロスリファレンスフィールドとホストフィールドを一致させ、それらのフィールドに一意のフィールド名を割り当てます。

PERSFILE のマスターファイルは次のように記述されています。

```
FILENAME = PERSFILE, SUFFIX = FOC, $
SEGNAME = IDSEG, SEGTYPE = S1, $
   FIELD = NAME,      ALIAS = FNAME, FORMAT = A12,      INDEX=I, $
   FIELD = ADDRESS,   ALIAS = DAS,   FORMAT = A24,               $
   FIELD = DOB,       ALIAS = IDOB,  FORMAT = YMD,               $
```

次のマスターファイルを使用して、PERSFILE と COMPFILE を結合します。ここでは、クロスリファレンスセグメント宣言の後にはレコード終了文字 ($) は配置されていません。

```
FILENAME = COMPFILE, SUFFIX = FOC, $
 SEGNAME = COMPSEG, SEGTYPE = S1, $
   FIELD = COMPANY,    ALIAS = CPY,      FORMAT = A40,           $
 SEGNAME = DIVSEG,  PARENT = COMPSEG, SEGTYPE = S1, $
   FIELD = DIVISION,  ALIAS = DV,       FORMAT = A20,           $
   FIELD = DIVMGR,    ALIAS = NAME,     FORMAT = A12,           $
 SEGNAME = ADSEG,   PARENT = DIVSEG,  SEGTYPE = KU,
  CRSEGNAME = IDSEG, CRKEY = DIVMGR, CRFILE = PERSFILE,
   FIELD = NAME,       ALIAS = FNAME,   FORMAT = A12, INDEX = I,$
   FIELD = DADDRESS,  ALIAS = ADDRESS, FORMAT = A24,           $
   FIELD = DDOB,       ALIAS = DOB,     FORMAT = YMD,           $
 SEGNAME = PRODSEG, PARENT = COMPSEG, SEGTYPE = S1, $
   FIELD = PRODUCT,    ALIAS = PDT,      FORMAT = A8,            $
   FIELD = PRODMGR,    ALIAS = NAME,     FORMAT = A12,           $
 SEGNAME = BDSEG,   PARENT = PRODSEG, SEGTYPE = KU,
  CRSEGNAME = IDSEG, CRKEY = PRODMGR, CRFILE = PERSFILE,
   FIELD = NAME,       ALIAS = FNAME,   FORMAT = A12, INDEX = I,$
   FIELD = PADDRESS,  ALIAS = ADDRESS, FORMAT = A24,           $
   FIELD = PDOB,       ALIAS = DOB,     FORMAT = YMD,           $
```

DIVMGR および PRODMGR は、CRKEY として記述されています。PERSFILE データソースの「NAME」というフィールド名に合わせて、共通のエイリアス名が自動的に NAME になります。また、JOIN 情報の後に続くフィールド宣言により、ADDRESS および DOB フィールドの名前がレポートで参照される名前にそれぞれ変更されます。PERSFILE の実フィールド名がエイリアスに使用されます。

なお、NAME フィールドは共通の JOIN フィールドのため、この名前を変更することはできません。この名前はクロスリファレンスデータソースに記述されているため、名前を変更するフィールドとともに宣言に含める必要があります。ただし、このフィールドのエイリアス名は変更できること、およびこのフィールドがレポートに使用されないことから、フィールド名を変更できなくても問題になりません。NAME フィールドは JOIN フィールドのため、このフィールドには DIVMGR および PRODMGR と同一の情報が格納されます。

次の規則に従う必要があります。

- ホストデータソースの共通の JOIN フィールドである FIELDNAME または ALIAS を、クロスリファレンスデータソースの FIELDNAME と一致させる必要があります。
- 共通の JOIN フィールドの名前を変更することはできませんが、エイリアス名は変更することができます。クロスリファレンスセグメントのその他のフィールドは名前を変更することができます。

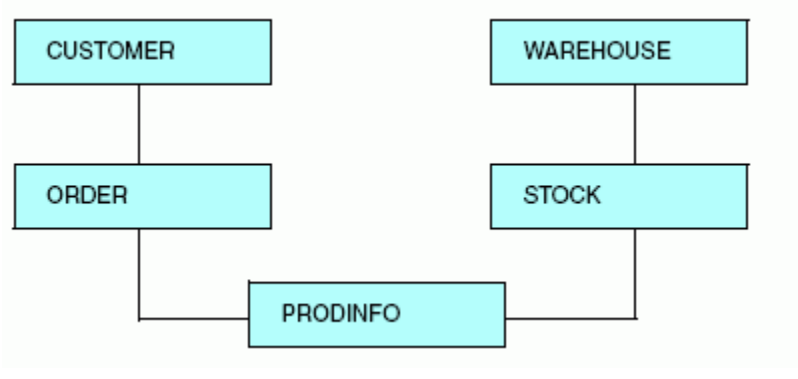
- ホストデータソースのマスターファイルで、クロスリファレンス情報の後にクロスリファレンスセグメントの各フィールド宣言をフィールドの出現順に配置します。次のように、ホストマスターファイルでは、クロスリファレンスセグメント宣言の末尾のレコード終了文字 ($) は省略します。

```
SEGNAME = BDSEG, PARENT = PRODSEG, SEGTYPE = KU,
 CRSEGNAME = IDSEG, CRKEY = PRODMGR, CRFILE = PERSFILE,
  FIELD = NAME,     ALIAS = FNAME,   FORMAT = A12 ,INDEX=I, $
  FIELD = PADDRESS, ALIAS = ADDRESS, FORMAT = A24         , $
  FIELD = PDOB,     ALIAS = DOB,     FORMAT = YMD         , $
```

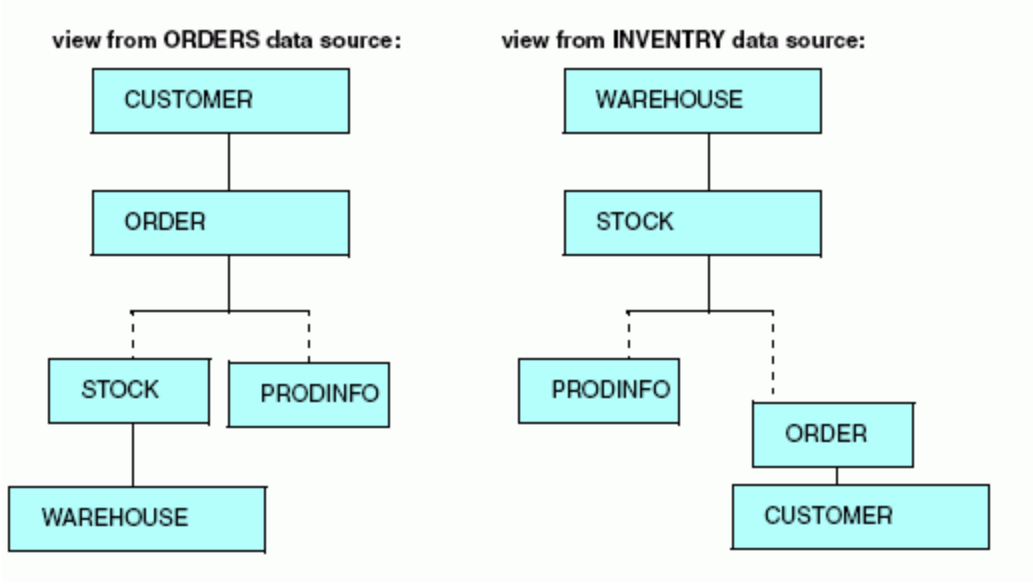
複数ホストデータソース複数セグメントからの結合 - 複数の親

状況によっては、2 つの異なるホストデータソースから 1 つのクロスリファレンスセグメントに同時に結合する場合があります。このような構造を 1 つのデータソースとして記述する場合、1 つのセグメントに対して 2 つの異なる親セグメントを指定する必要があると考えられますが、その方法は無効です。この場合は、JOIN を使用してそれぞれを異なるデータソースの情報として記述することにより、同様の結果を得ることができます。

ここでは、部品の顧客注文数、部品の在庫数、部品の一般情報を管理するアプリケーションについて考察します。このアプリケーションを 1 つのデータソースとして記述すると、次のような構造になります。



複数のデータソースを結合して、これと同様の構造を作成することができます。以下はその例です。



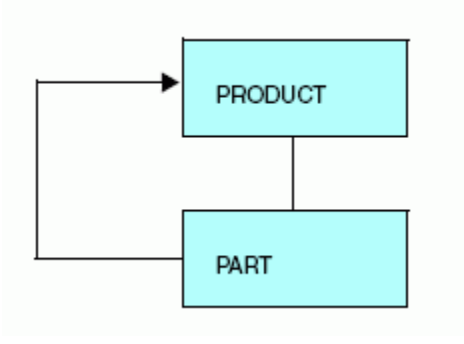
ORDERS データソースには CUSTOMER および ORDER セグメント、INVENTRY データソースには WAREHOUSE および STOCK セグメント、PRODUCTS データソースには PRODINFO セグメントがそれぞれ格納されています。INVENTRY および ORDERS データソースは、PRODUCTS データソースとの間に 1 対 1 の JOIN が構築されています。INVENTRY データソースでは STOCK がホストセグメントで、ORDERS データソースでは ORDER がホストセグメントです。

また、INVENTRY データソースの STOCK セグメントから ORDERS データソースの ORDER セグメントへ 1 対 n の結合があり、ORDERS データソースの ORDER セグメントから、INVENTRY データソースの STOCK セグメントへの逆の 1 対 n の結合があります。

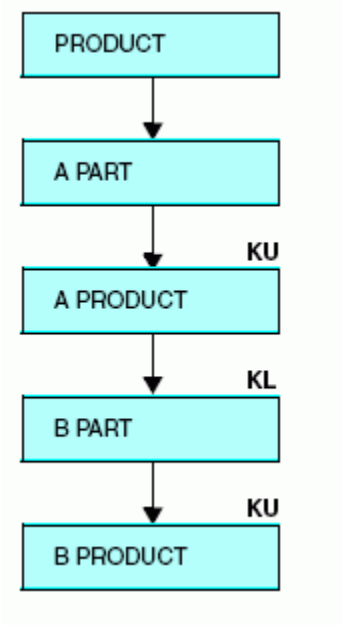
上記の 2 つのホストデータソースからこれらの 3 つのデータソース間の JOIN 見ると、前述の複数の親構造に類似しているといえます。

再帰的セグメントの使用

データソースがそのデータソース自体をクロスリファレンスにする場合がまれにあります。ここでは、製品情報およびその製品を構成する部品の情報が格納されたデータソースについて考察します。部品自体が 1 つの製品として構成部品を持つ場合もあります。この関係を図示すると次のようになります。



この場合、次のように 2 つのレベルの構成部品として表わされます。



再帰的 JOIN についての詳細は、『TIBCO WebFOCUS Language リファレンス』を参照してください。

クラスタマスターファイルの作成

クラスタマスターファイルは、データソースのタイプに関係なく、複数の異なるデータソースが、それぞれ個別のセグメントとして記述されたマスターファイルです。特定のセグメントの SUFFIX が最上位セグメントの SUFFIX と異なる場合、そのセグメントのタイプは SEGSUF 属性で指定します。

注意：タイプの異なる SUFFIX 値がクラスタマスターファイルに含まれている場合、FOCUS セグメントを最上位セグメントにすることはできません。

フィールドの区切り値の個別行としての読み取り

SEGSUF=DFIX (Delimited Flat File) が指定されたセグメントを追加した場合、複数の値が特定の文字で区切られたフィールド (例、複数の Email アドレスがブランクで区切られている場合) を回転 (ピボット) することで、それぞれの値を個別の行として読み取ることができます。

構文　フィールドの区切り値の個別行としての読み取り

マスターファイルでは、セグメント定義として SEGTYPE=S0、SEGSUF=DFIX、POSITION 属性を追加します。POSITION 属性では、区切り値が含まれたフィールドを指定します。

```
SEGNAME=parentseg, SUFFIX=suffix, SEGTYPE=S1,$
 FIELD=FIELD1,  ...,$
 FIELD=delimitedfield, ALIAS=alias1, USAGE=fmt, ACTUAL=afmt,$
   ...
SEGNAME=dfixsegname, PARENT=parentseg, SEGSUF=DFIX,
  POSITION=delimitedfield,$
  FIELD=name, ALIAS=alias2, USAGE=fmt, ACTUAL=afmt,$
   ...
```

アクセスファイルを作成し、DFIX セグメントの行区切り文字、およびその他の DFIX 属性を記述します。

```
 SEGNAME=dfixsegname, RDELIMITER='delimiter', $
```

説明

`delimitedfield`

区切り付きフィールドの名前です。

`alias1`

区切り付きフィールドのエイリアスです。

`fmt`

区切り付きフィールドの USAGE フォーマットです。

`afmt`

区切り付きフィールドの ACTUAL フォーマットです。

`dfixsegname`

区切り付きフィールド用に追加されたセグメント定義のセグメント名です。

`parentseg`

区切り付きフィールドが実際に属しているセグメントの名前です。

`name`

区切り付きフィールドの構成要素の名前です。

`alias2`

区切り付きフィールドの構成要素のエイリアスです。

`delimiter`

区切り付きフィールドで使用されている区切り文字です。

リクエストを発行すると、この区切り文字に基づいて、フィールド値が個別の行として処理されます。

例　フィールドの区切り値の個別行としての読み取り

次の COUNTRYL.FTM ファイルには、国名のリストと、各国の首都の経度と緯度が含まれています。経度と緯度の値はカンマ (,) で区切られ、LNGLAT という単一フィールドに格納されています。

```
Argentina      -64.0000000,-34.0000000
Australia      133.0000000,-27.0000000
Austria        13.3333000,47.3333000
Belgium        4.0000000,50.8333000
Brazil         -55.0000000,-10.0000000
Canada         -95.0000000,60.0000000
Chile          -71.0000000,-30.0000000
China          105.0000000,35.0000000
Colombia       -72.0000000,4.0000000
Denmark        10.0000000,56.0000000
Egypt          30.0000000,27.0000000
Finland        26.0000000,64.0000000
France         2.0000000,46.0000000
Germany        9.0000000,51.0000000
Greece         22.0000000,39.0000000
Hungary        20.0000000,47.0000000
India          77.0000000,20.0000000
Ireland        -8.0000000,53.0000000
Israel         34.7500000,31.5000000
Italy          12.8333000,42.8333000
Japan          138.0000000,36.0000000
Luxembourg     6.1667000,49.7500000
Malaysia       112.5000000,2.5000000
Mexico         -102.0000000,23.0000000
Netherlands    5.7500000,52.5000000
Norway         10.0000000,62.0000000
Philippines    122.0000000,13.0000000
Poland         20.0000000,52.0000000
Portugal       -8.0000000,39.5000000
Singapore      103.8000000,1.3667000
South Africa   24.0000000,-29.0000000
South Korea    127.5000000,37.0000000
Spain          -4.0000000,40.0000000
Sweden         15.0000000,62.0000000
Switzerland    8.0000000,47.0000000
Taiwan         121.0000000,23.5000000
Thailand       100.0000000,15.0000000
Tunisia        9.0000000,34.0000000
Turkey         35.0000000,39.0000000
United Kingdom -.1300000,51.5000000
United States  -97.0000000,38.0000000
```

以下は、元の COMMA1 マスターファイルです。

```
FILENAME=COMMA1  , SUFFIX=FIX, IOTYPE=STREAM
 DATASET=appname/countryl.ftm, $
   SEGNAME=COU, SEGTYPE=S1, $
     FIELDNAME=COUNTRY, ALIAS=E01, USAGE=A15, ACTUAL=A15, $
     FIELDNAME=LNGLAT, ALIAS=LNGLAT,USAGE=A25, ACTUAL=A25, $
```

以下は、DFIX セグメントが追加された COMMA2 マスターファイルです。

```
FILENAME=COMMA2  , SUFFIX=FIX, IOTYPE=STREAM,
 DATASET=appname/countryl.ftm, $
   SEGNAME=COU, SEGTYPE=S1, $
     FIELDNAME=COUNTRY, ALIAS=E01, USAGE=A15, ACTUAL=A15, $
     FIELDNAME=LNGLAT, ALIAS=LNGLAT,USAGE=A25, ACTUAL=A25, $
   SEGNAME=COMMA2, SEGTYPE=S0, SEGSUF=DFIX,PARENT=COU,POSITION=LNGLAT,$
     FIELD=COORD, ALIAS = XY, USAGE=A25, ACTUAL=A25,$
```

以下は、COMMA2 アクセスファイルです。

```
SEGNAME=COMMA2, RDELIMITER=',', HEADER=NO, PRESERVESPACE=NO, $
```

次のリクエストは、COMMA2 マスターファイルを使用して値を出力します。

```
TABLE FILE COMMA2
PRINT COORD
BY COUNTRY
END
```

出力結果では、LNGLAT フィールドが 2 つの個別のレコードとして処理されています。以下は、出力の一部を示しています。

```
COUNTRY          COORD
-------          -----
Argentina        -64.0000000
                 -34.0000000
Australia        133.0000000
                 -27.0000000
Austria          13.3333000
                 47.3333000
Belgium          4.0000000
                 50.8333000
Brazil           -55.0000000
                 -10.0000000
Canada           -95.0000000
                 60.0000000
Chile            -71.0000000
                 -30.0000000
China            105.0000000
                 35.0000000
Colombia         -72.0000000
```

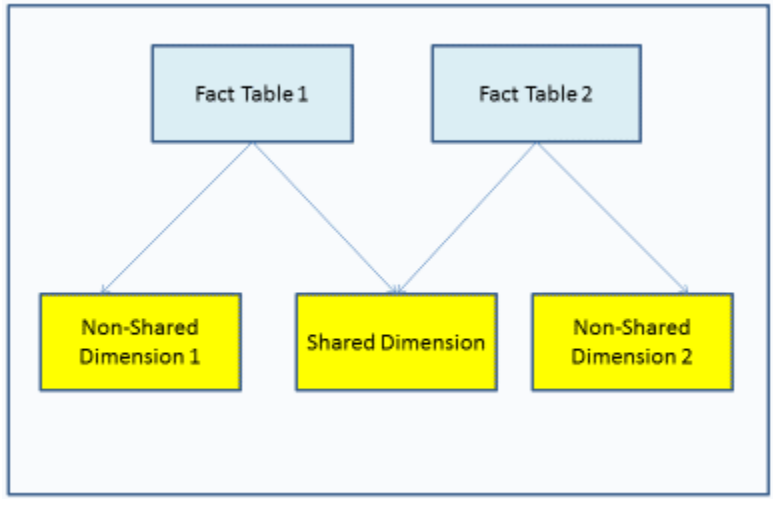
マルチルートクラスタマスターファイルの作成

クラスタマスターファイルは、ベースシノニムを指定する CRFILE 属性参照により、各セグメントがクラスタに追加されたシノニムです。子セグメントは、JOIN WHERE 属性によって、親セグメントに結合されます。クラスタマスターファイルは、複数のルートセグメントを持つことができます。この場合、スタースキーマで見られるように、通常、ルートセグメントはファクトテーブルであり、子セグメントはディメンションテーブルです。この種の構造は、「マルチファクトクラスタ」呼ばれます。

クラスタのルートであるファクトテーブルのそれぞれは、マスターファイルでルートセグメントを識別する PARENT=. 属性を指定する必要があります。

ディメンションテーブルは、(共有ディメンションと呼ばれる) マルチファクトテーブルの子、または (非共有ディメンションと呼ばれる) シングルファクトテーブルにすることができます。共有セグメントのそれぞれには、マスターファイルの PARENT 属性が複数指定されます。

下図は、単純なマルチファクト構造を示しています。



マルチファクトマスターファイルを使用するレポート作成についての詳細は、『TIBCO WebFOCUS Language リファレンス』を参照してください。

構文　マルチファクトクラスタの定義

ルートセグメントのそれぞれは、マスターファイルで PARENT=. を記述する必要があります。次の構文は、マルチファクトクラスタのルートセグメントを定義するために必要な属性を示しています。その他すべてのセグメント属性もサポートされています。

```
SEGMENT=rsegname, PARENT=., CRFILE=[rapp/]rfilename,
        CRINCLUDE=ALL,$
```

説明

`crsegname`
ルートセグメントの名前です。

`PARENT=.`
このセグメントをマルチファクトクラスタのルートセグメントとして定義します。

`CRFILE=[rapp/]rfilename`
ルートファクトテーブルを記述するアプリケーションパスおよびマスターファイルの名前です (オプション)。

`CRINCLUDE=ALL`
このクラスタマスターファイルを使用して、ファクトテーブルのすべてのフィールドをアクセス可能にします。この属性を省略する場合は、このマスターファイルを使用してアクセス可能にするファクトテーブルのすべてのフィールドをリスト化する必要があります。

次の例は、wfretail アプリケーションフォルダに格納されている WF_RETAIL_LITE マスターファイルのルートセグメントの記述を示しています。

```
SEGMENT=WF_RETAIL_SALES, PARENT=., CRFILE=wfretail/facts/wf_retail_sales,
      CRINCLUDE=ALL, DESCRIPTION='Sales Fact', $
```

共有ディメンションのそれぞれに対して、マスターファイルで複数の PARENT 属性を指定する必要があります。また、親のそれぞれに対して、JOIN WHERE を指定する必要があります。次の構文は、マルチファクトクラスタマスターファイルで共有ディメンションを定義するために必要な属性を示しています。

```
 SEGMENT=dsegname, CRFILE=[dapp/]dfilename,
          [CRSEGMENT=crsegname,] CRINCLUDE=ALL,$
     PARENT=parent1, SEGTYPE=KU, CRJOINTYPE=jointype1,
       JOIN_WHERE=expression1;, $
    PARENT=parent2, SEGTYPE=KU, JOIN_TYPE=jointype2,
       JOIN_WHERE=expression2;,
    . . . $
```

説明

`SEGMENT=dsegname`
共有ディメンションの名前です。

`CRFILE=[dapp/]dfilename`
ディメンションテーブルを記述するアプリケーションパスおよびマスターファイルの名前です (オプション)。

`CRSEGMENT=crsegname`
ディメンションマスターファイルで結合するセグメントの名前です。ディメンションマスターファイルが単一セグメントで構成されている場合、この指定はオプションです。

`CRINCLUDE=ALL`
このクラスタマスターファイルを使用して、ディメンションテーブルのすべてのフィールドをアクセス可能にします。この属性を省略する場合は、このマスターファイルを使用してアクセス可能にするファクトテーブルのすべてのフィールドをリスト化する必要があります。

`PARENT=parent1 PARENT=parent2 ...`
共有ディメンションの親セグメントの名前です。

`CRJOINTYPE=jointype1`
共有ディメンションと最初の親セグメントを結合する、サポートされる JOIN のタイプです。有効な値は、INNER、LEFT-OUTER、RIGHT-OUTER、FULL-OUTER です。指定する JOIN のタイプは、テーブルを定義するリレーショナルエンジンでサポートされている必要があります。

`JOIN_TYPE=jointype2`
共有ディメンションと後続親セグメントを結合する、サポートされる JOIN のタイプです。有効な値は、INNER、LEFT-OUTER、RIGHT-OUTER、FULL-OUTER です。指定する JOIN のタイプは、テーブルを定義するリレーショナルエンジンでサポートされている必要があります。

`JOIN_WHERE=expression1; JOIN_WHERE=expression2;`
親セグメントと共有ディメンションのそれぞれを結合する、サポートされる JOIN 式です。

スタースキーマを記述するシノニムの場合、通常、式のそれぞれは等価条件 (演算子 EQ を使用) および 1 対 n 関係を記述します。

次の例は、WF_RETAIL_LITE マスターファイルでの共有ディメンションセグメントの定義を示しています。このシノニムは、wfretail アプリケーションフォルダに格納されています。

```
SEGMENT=WF_RETAIL_CUSTOMER, CRFILE=wfretail/dimensions/wf_retail_customer,
       CRINCLUDE=ALL, DESCRIPTION='Customer Dimension', $
   PARENT=WF_RETAIL_SALES, SEGTYPE=KU, CRJOINTYPE=LEFT_OUTER,
    JOIN_WHERE=WF_RETAIL_SALES.ID_CUSTOMER EQ
      WF_RETAIL_CUSTOMER.ID_CUSTOMER;, $
   PARENT=WF_RETAIL_SHIPMENTS, SEGTYPE=KU, JOIN_TYPE=LEFT_OUTER,
    JOIN_WHERE=WF_RETAIL_SHIPMENTS.ID_CUSTOMER EQ
       WF_RETAIL_CUSTOMER.ID_CUSTOMER;, $
```

次の例は、WF_RETAIL_LITE マスターファイルでの非共有ディメンションセグメントの定義を示しています。このシノニムは、wfretail アプリケーションフォルダに格納されています。

```
SEGMENT=WF_RETAIL_TIME_DELIVERED, SEGTYPE=KU, PARENT=WF_RETAIL_SHIPMENTS,
      CRFILE=wfretail/dimensions/wf_retail_time_lite,
      CRSEGMENT=WF_RETAIL_TIME_LITE,
      CRINCLUDE=ALL, CRJOINTYPE=LEFT_OUTER,
      JOIN_WHERE=ID_TIME_DELIVERED EQ WF_RETAIL_TIME_DELIVERED.ID_TIME;,
      DESCRIPTION='Shipping Time Delivered Dimension',
      SEG_TITLE_PREFIX='Delivery,', $
```

7 マスターファイルビジネスビューの作成

マスターファイルのビジネスビュー (BV) は、データソース内の項目を物理的な位置でグループ化するのではなく、アプリケーションのビジネスロジックに基づいて互いに関連する項目をグループ化します。ビジネスビューはデータの情報モデルを物理的な格納メカニズムから分離するため、開発者およびユーザは複雑なデータベース構造を意識せずにビジネスの問題解決に必要な情報を取得することができます。

トピックス

ビジネスビューでのビジネスロジックのグループ化

ビジネスビューは、実マスターファイルの一部として格納された、ユーザがアクセス可能な限定的なフィールドセットです。ビジネスビューを定義することで、ユーザに限定的なデータビューが提供されるとともに、アプリケーションの保守が簡素化され、追加のセキュリティが提供されます。

ビジネスビューは「フォルダ」と呼ばれる仮想セグメントに編成され、これらはマスターファイルの実セグメントおよびクロスリファレンスセグメントの下に定義されます。各フォルダには、フィールドおよび他のフォルダのグループを含めることができます。フォルダをブランクにすることもできます。フォルダ内のフィールドは、元のマスターファイル内の異なるセグメントから取得することもできます。フィールドが元のマスターファイルの単一パスに存在せず、リクエストに元のマスターファイルの別のパスにあるフィールドが含まれる場合、リクエストの実行時に警告メッセージが表示される可能性があります。

フォルダ階層は、下位フォルダの宣言で PARENT 属性を指定することで定義できます。

リクエストでビジネスビューを使用すると、すべての実フィールドおよびセキュリティ情報が元のマスターファイルから取得されます。元のマスターファイルのクロスリファレンスセグメントからフィールドを取得する場合、すべてのクロスリファレンスセグメントおよびキー情報は元のマスターファイルに保持されます。

ビジネスビューが含まれたマスターファイルを WebFOCUS のツールで開いた場合、ビジネスビューのフォルダのみが表示されます。

ビジネスビューには、元のマスターファイルの実フィールド、一時項目 (COMPUTE)、一時項目 (DEFINE)、フィルタを含めることができます。

ビジネスビューは、リレーショナルデータおよび FOCUS データソースのビューとして最も役立ちます。

ビジネスビューのフィールド名およびセグメント名は、元のセグメントの名前と同一にすることも、新しい名前を割り当てることもできます。ビジネスビューで新しいフィールド名を割り当てる場合は、ALIAS 値を元のフィールド名にする必要があります。

構文　フォルダの定義

```
FOLDER = folder_name, [PARENT=parent_folder_name
    [,DESCRIPTION="default_desc"
    [, DESC_ln="desc_for_ln" ...]] $
     [FIELD = bv_field_name, [ALIAS=real_field_name] ,
     [BELONGS_TO_SEGMENT= real_segment_name]  ,
     [, TITLE = "default_title"
     [, TITLE_ln="title_for_ln" ...]]]
     [, DESC="default_desc"
     [, DESC_ln="desc_for_ln" ...], $]]
      FIELD= bv_field_name[, ALIAS = real_field_name],
          BELONGS_TO_SEGMENT = real_segment_name
          [,TITLE = default_title]
          [,TITLE_ln = title_for_ln]
          [,DESCRIPTION = default_description ]
          [,DESCRIPTION_ln = desc_for_ln]  ,$
```

説明

`folder_name`
ビュー内の仮想セグメントの名前です。フォルダはブランクにすることもできます。

`parent_folder_name`
仮想セグメントの親の名前です。

`bv_field_name`
割り当てるフィールド名です。実フィールドと同一名または別のフィールド名を指定することができます。実フィールド名と異なる場合、実フィールドは ALIAS 属性で識別されます。

`real_field_name`
元のマスターファイルに記述されたフィールド名です。このフィールドには、実フィールドまたは一時項目 (DEFINE/COMPUTE) のいずれかを指定することができます。
bv_field_name が real_field_name と一致する場合は、ALIAS 属性を省略することができます。ALIAS を指定しない場合は、ビジネスビューのフィールド名を元のマスターファイルのフィールド名と一致させる必要があります。

`real_segment_name`
元のマスターファイルに存在するフィールドのセグメント名です。元のマスターファイルで実フィールド名が一意の場合は、BELONGS_TO_SEGMENT 属性を省略することができます。元のマスターファイルで実フィールド名が一意でない場合に BELONGS_TO_SEGMENT 属性を省略すると、元のマスターファイルで名前が一致するフィールドの最初のフィールドが使用されます。

`default_title`
LANG パラメータがサーバのデフォルト言語に設定されている場合、またはデフォルト以外の言語に設定されているがマスターファイルのフィールドに対応する TITLE_*ln* 属性が存在しない場合に使用するフィールドタイトルです。このフィールドタイトルは、*ln* 値が無効な場合にも使用されます。

`default_desc`
LANG パラメータがサーバのデフォルトの言語に設定されている場合、またはデフォルト以外の言語に設定されているがマスターファイルのフィールドに対応する DESC_*ln* 属性が存在しない場合に使用する説明テキストです。この説明は、*ln* 値が無効な場合にも使用されます。この説明はフロントエンドユーザインターフェースに表示されます。

`ln`
タイトルまたは説明を適用する言語を指定します。*ln* の有効な値は、2 バイトの ISO 639 言語コードの略名です。

`title_for_ln`
LANG パラメータがサーバのデフォルト言語以外の言語に設定され、マスターファイルに対応する TITLE_*ln* 属性が存在する場合に使用するタイトルです。ここで、*ln* は LANG パラメータで指定した言語の 2 文字のコードを表します。

`desc_for_ln`
LANG パラメータがサーバのデフォルト言語以外の言語に設定され、マスターファイルに対応する DESC_*ln* 属性が存在する場合に使用する説明です。ここで、*ln* は LANG パラメータにより指定された言語の 2 文字のコードを表します。

参照　言語名と言語コード

言語名	2 文字の言語コード	3 文字の言語の略名
アラビア語	ar	ARB
バルト言語	lt	BAL

言語名	2 文字の言語コード	3 文字の言語の略名
中国語 (簡体字)	zh	PRC
中国語 (繁体字)	tw	ROC
チェコ語	cs	CZE
デンマーク語	da	DAN
オランダ語	nl	DUT
英語 (米国)	en	AME または ENG
英語 (英国)	uk	UKE
フィンランド語	fi	FIN
フランス語 (カナダ)	fc	FRE
フランス語 (フランス)	fr	FRE
ドイツ語 (オーストリア)	at	GER
ドイツ語 (ドイツ)	de	GER
ギリシャ語	el	GRE
ヘブライ語	iw	HEW
イタリア語	it	ITA
日本語 (ASCII: Shift-JIS(cp942)	ja	JPN
日本語 (UNIX では、ASCII: EUC(cp10942))	je	JPE
韓国語	ko	KOR
ノルウェー語	no	NOR
ポーランド語	pl	POL
ポルトガル語 (ブラジル)	br	POR

言語名	2 文字の言語コード	3 文字の言語の略名
ポルトガル語 (ポルトガル)	pt	POR
ロシア語	ru	RUS
スペイン語	es	SPA
スウェーデン語	sv	SWE
タイ語	th	THA
トルコ語	tr	TUR

参照　ビジネスビュー使用時の注意

- USAGE フォーマットや ACTUAL フォーマット、インデックスなどの詳細情報は、元のセグメントに保持されます。
- ビジネスビューに複数のマスターファイルのフィールドを含める場合は、ビジネスビューが格納されているマスターファイルに、すべてのセグメントがクロスリファレンスまたはクラスタ JOIN として含まれている必要があります。
- ビジネスビューが格納されているマスターファイルで定義済みの DBA 属性は、このビジネスビューに適用されます。クロスリファレンスセグメントの DBA 属性を適用するには、SET DBASOURCE=ALL コマンドを有効にする必要があります。
- ビジネスビューでは、データソース保守コマンドはサポートされません。
- 複数のファイルを結合した場合など、マスターファイルに記述した複数のフィールドが同一名の場合、ビジネスビューで参照されているフィールド名のインスタンスは BELONGS_TO_SEGMENT 属性で識別されます。
- フォルダはブランクにすることができます。
- ビジネスビューでは USE コマンドがサポートされます。
- マスターファイルプロファイル (MFD_PROFILE 属性) は、アクセスされる各マスターファイルに対して実行されます。
- SORTOBJ 宣言および STYLEOBJ 宣言は、元のマスターファイルでサポートされません。
- ビジネスビューに対して SQL SELECT コマンドを発行することができます。ただし、ダイレクト SQL パススルーリクエストはサポートされません。

- ビジネスビューは、ENCRYPT コマンドと DECRYPT コマンドにより、暗号化および復号化することができます。
- ビジネスビューでは、代替ファイルビューおよび完全修飾フィールド名がサポートされます。
- ビジネスビューフォルダに対して SEG. 演算子を使用すると、実セグメント内のすべてのフィールドではなく、そのフォルダ内のすべてのフィールドが表示されます。
- ビジネスビューでは、すべての HOLD フォーマットがサポートされます。
- FOCUS 以外のデータソースのすべてのアダプタは、ビジネスビューに対する検索リクエストをサポートします。

例　ビジネスビューの作成

次の例は、3 つのフォルダで構成された EMPLOYEE データソースのビジネスビューを示しています。

- 最初のフォルダには、EMPLOYEE マスターファイルの EMP_ID、LAST_NAME、FIRST_NAME、CURR_SAL、CURR_JOBCODE が含まれています。
- 2 番目のフォルダには、元のマスターファイルの PAYINFO セグメントの SALARY、および昇給記録のフィールド (DAT_INC、PCT_INC) が含まれています。
- 3 番目のフォルダには、クロスリファレンスセグメントである JOBSEG セグメントの JOBCODE フィールドおよび JOB_DESC フィールドが含まれています。

「JOBCODE」という名前のフィールドは 2 番目と 3 番目のフォルダに含まれています。BELONGS_TO_SEGMENT 属性は、EMPLOYEE マスターファイルの PAYINFO セグメントの JOBCODE フィールドと JOBSEG セグメントの JOBCODE フィールドを区別します。また、実セグメントとは異なる名前で定義されたフィールドには、実セグメントで指定されたフィールド名が ALIAS 値として記述されていることに注意してください。

```
FILENAME=EMPLOYEE, SUFFIX=FOC, REMARKS='Legacy Metadata Sample: employee',$
SEGNAME=EMPINFO,  SEGTYPE=S1
 FIELDNAME=EMP_ID,        ALIAS=EID,     FORMAT=A9,       $
 FIELDNAME=LAST_NAME,     ALIAS=LN,      FORMAT=A15,      $
 FIELDNAME=FIRST_NAME,    ALIAS=FN,      FORMAT=A10,      $
 FIELDNAME=HIRE_DATE,     ALIAS=HDT,     FORMAT=I6YMD,    $
 FIELDNAME=DEPARTMENT,    ALIAS=DPT,     FORMAT=A10,      $
 FIELDNAME=CURR_SAL,      ALIAS=CSAL,    FORMAT=D12.2M,   $
 FIELDNAME=CURR_JOBCODE, ALIAS=CJC,     FORMAT=A3,       $
 FIELDNAME=ED_HRS,        ALIAS=OJT,     FORMAT=F6.2,     $
SEGNAME=FUNDTRAN, SEGTYPE=U,   PARENT=EMPINFO
 FIELDNAME=BANK_NAME,     ALIAS=BN,      FORMAT=A20,      $
 FIELDNAME=BANK_CODE,     ALIAS=BC,      FORMAT=I6S,      $
 FIELDNAME=BANK_ACCT,     ALIAS=BA,      FORMAT=I9S,      $
 FIELDNAME=EFFECT_DATE,   ALIAS=EDATE,   FORMAT=I6YMD,    $
SEGNAME=PAYINFO,  SEGTYPE=SH1, PARENT=EMPINFO
 FIELDNAME=DAT_INC,       ALIAS=DI,      FORMAT=I6YMD,    $
 FIELDNAME=PCT_INC,       ALIAS=PI,      FORMAT=F6.2,     $
 FIELDNAME=SALARY,        ALIAS=SAL,     FORMAT=D12.2M,   $
 FIELDNAME=JOBCODE,       ALIAS=JBC,     FORMAT=A3,       $
SEGNAME=ADDRESS,  SEGTYPE=S1,  PARENT=EMPINFO
 FIELDNAME=TYPE,          ALIAS=AT,      FORMAT=A4,       $
 FIELDNAME=ADDRESS_LN1,   ALIAS=LN1,     FORMAT=A20,      $
 FIELDNAME=ADDRESS_LN2,   ALIAS=LN2,     FORMAT=A20,      $
 FIELDNAME=ADDRESS_LN3,   ALIAS=LN3,     FORMAT=A20,      $
 FIELDNAME=ACCTNUMBER,    ALIAS=ANO,     FORMAT=I9L,      $
SEGNAME=SALINFO,  SEGTYPE=SH1, PARENT=EMPINFO
 FIELDNAME=PAY_DATE,      ALIAS=PD,      FORMAT=I6YMD,    $
 FIELDNAME=GROSS,         ALIAS=MO_PAY,  FORMAT=D12.2M,   $
SEGNAME=DEDUCT,   SEGTYPE=S1,  PARENT=SALINFO
 FIELDNAME=DED_CODE,      ALIAS=DC,      FORMAT=A4,       $
 FIELDNAME=DED_AMT,       ALIAS=DA,      FORMAT=D12.2M,   $
SEGNAME=JOBSEG,  SEGTYPE=KU ,PARENT=PAYINFO, CRFILE=JOBFILE, CRKEY=JOBCODE,$
SEGNAME=SECSEG,  SEGTYPE=KLU,PARENT=JOBSEG,  CRFILE=JOBFILE,$
SEGNAME=SKILLSEG,SEGTYPE=KL, PARENT=JOBSEG,  CRFILE=JOBFILE,$
SEGNAME=ATTNDSEG,SEGTYPE=KM, PARENT=EMPINFO, CRFILE=EDUCFILE,CRKEY=EMP_ID,$
SEGNAME=COURSEG, SEGTYPE=KLU,PARENT=ATTNDSEG,CRFILE=EDUCFILE,$
```

```
FOLDER=FOLDER1, $
 FIELDNAME=EMPID, ALIAS=EMP_ID,
  BELONGS_TO_SEGMENT=EMPINFO, $
FIELDNAME=LASTNAME, ALIAS=LAST_NAME,
  BELONGS_TO_SEGMENT=EMPINFO, $
FIELDNAME=FIRSTNAME,
  ALIAS=FIRST_NAME,
  BELONGS_TO_SEGMENT=EMPINFO, $
 FIELDNAME=DEPARTMENT,
  BELONGS_TO_SEGMENT=EMPINFO, $
 FIELDNAME=CURRSAL, ALIAS=CURR_SAL,
  BELONGS_TO_SEGMENT=EMPINFO, $
 FIELDNAME=CURR_JOBCODE,
  BELONGS_TO_SEGMENT=EMPINFO, $

FOLDER=FOLDER3, PARENT=FOLDER1, $
 FIELDNAME=DAT_INC,
  BELONGS_TO_SEGMENT=PAYINFO, $
 FIELDNAME=PCT_INC,
  BELONGS_TO_SEGMENT=PAYINFO, $
 FIELDNAME=SALARY,
  BELONGS_TO_SEGMENT=PAYINFO, $
 FIELDNAME=JOBCODE,
  BELONGS_TO_SEGMENT=PAYINFO, $

FOLDER=FOLDER2, PARENT=FOLDER1, $
 FIELDNAME=JOBCODE,
  BELONGS_TO_SEGMENT=JOBSEG, $
 FIELDNAME=JOB_DESC,
  BELONGS_TO_SEGMENT=JOBSEG, $
```

フォルダの 1 つには、JOBSEG セグメントの JOBCODE フィールドが格納されています。EMPLOYEE の JOBSEG セグメントは、JOBFILE マスターファイルを指定するクロスリファレンスセグメントです。JOBFILE マスターファイルは次のとおりです。

```
FILENAME=JOBFILE  ,SUFFIX=FOC, $
SEGNAME=JOBSEG   ,SEGTYPE=S1
 FIELD=JOBCODE     ,ALIAS=JC          ,USAGE=A3          ,INDEX=I,$
 FIELD=JOB_DESC    ,ALIAS=JD          ,USAGE=A25                ,$
SEGNAME=SKILLSEG ,SEGTYPE=S1   ,PARENT=JOBSEG
 FIELD=SKILLS      ,ALIAS=            ,USAGE=A4                 ,$
 FIELD=SKILL_DESC  ,ALIAS=SD          ,USAGE=A30                ,$
SEGNAME=SECSEG   ,SEGTYPE=U    ,PARENT=JOBSEG
 FIELD=SEC_CLEAR   ,ALIAS=SC          ,USAGE=A6                 ,$
```

次のプロシジャはビジネスビューを参照します。

```
TABLE FILE EMPLOYEE
PRINT FOLDER3.JOBCODE JOB_DESC
BY LASTNAME BY FIRSTNAME
BY HIGHEST 1 DAT_INC  NOPRINT
END
```

出力結果は次のとおりです。

```
LAST_NAME       FIRST_NAME  JOBCODE  JOB_DESC
---------       ----------  -------  --------
BANNING         JOHN        A17      DEPARTMENT MANAGER
BLACKWOOD       ROSEMARIE   B04      SYSTEMS ANALYST
CROSS           BARBARA     A17      DEPARTMENT MANAGER
GREENSPAN       MARY        A07      SECRETARY
IRVING          JOAN        A15      ASSIST.MANAGER
JONES           DIANE       B03      PROGRAMMER ANALYST
MCCOY           JOHN        B02      PROGRAMMER
MCKNIGHT        ROGER       B02      PROGRAMMER
ROMANS          ANTHONY     B04      SYSTEMS ANALYST
SMITH           MARY        B14      FILE QUALITY
                RICHARD     A01      PRODUCTION CLERK
STEVENS         ALFRED      A07      SECRETARY
```

次に、EMPLOYEE マスターファイルにフィルタを追加して、ビジネスビューの FOLDER 1 に含めます。

マスターファイル

```
FILTER DFILTER  WITH EMPINFO.EMP_ID=DEPARTMENT EQ 'MIS'; $
```

ビジネスビュー

```
FIELDNAME=DFILTER, ALIAS=DFILTER, BELONGS_TO_SEGMENT=EMPINFO, $
```

次のリクエストにはフィルタが使用されています。

```
TABLE FILE EMPLOYEE
PRINT FOLDER3.JOBCODE JOB_DESC
BY LASTNAME BY FIRSTNAME
BY HIGHEST 1 DAT_INC  NOPRINT
WHERE DFILTER
END
```

出力結果は次のとおりです。

```
LAST_NAME       FIRST_NAME  JOBCODE  JOB_DESC
---------       ----------  -------  --------
BLACKWOOD       ROSEMARIE   B04      SYSTEMS ANALYST
CROSS           BARBARA     A17      DEPARTMENT MANAGER
GREENSPAN       MARY        A07      SECRETARY
JONES           DIANE       B03      PROGRAMMER ANALYST
MCCOY           JOHN        B02      PROGRAMMER
SMITH           MARY        B14      FILE QUALITY
```

ビジネスビューの DV ロール

従来のビジネスビューでは、データソース内の関連項目を物理的な位置でグループ化するのではなく、アプリケーションのビジネスロジックを表すフォルダごとに関連項目をグループ化することで、データソースのカスタム論理ビューをユーザに提供していました。ただし、これらのフォルダに分類された各フィールドは、リクエストでのフィールドの役割を示すものではありませんでした。

一方、従来のディメンションビューでは、リクエストでの各フィールドの役割に基づいてフィールドが分類されていました。各メジャーはメジャーグループに配置され、各階層はディメンション内で、各レベルは階層内で、各属性はレベル内でそれぞれ構成されていました。その結果、Designer のレポートキャンバスまたはグラフキャンバスでフィールドをダブルクリック、ドラッグした場合に、そのフィールドは、ディメンションビュー構造の位置に基づいてソートフィールドまたは集計フィールドとして追加されました。ただし、ディメンションビューでは、データソースのカスタム論理ビューを作成することはできませんでした。

ビジネスビューで DV ロールを使用することにより、フィールドをフォルダ単位でグループ化し、各フィールドに対してリクエストでの役割を示すロールを割り当てることが可能になります。構文は明確かつ簡潔であり、構造上の任意の位置にフォルダを作成したり、結果として生成されるフィールドを複数のフォルダで利用したりと、柔軟に活用することができます。

たとえば、フィールドに [ディメンション] ロールを割り当てた場合、フィールドをダブルクリックするか、レポートキャンバスまたはグラフキャンバスにドラッグすると、そのフィールドがレポートの [BY] フィールドコンテナまたはグラフの横軸に自動的に追加されます。また、フォルダ内の一連のフィールドに [ドリルレベル] ロールを割り当てた場合、AUTODRILL をオンに設定しておくと、生成された出力結果に最上位から最下位までのオートドリルダウンが生成されます。

シノニムは、Reporting Server Web コンソール、データ管理コンソール、または メタデータキャンバスで作成、編集することができます。

ディメンションビューロールの割り当て

ビジネスビューでは、シノニムのビューを提供するフォルダ (セグメントとして機能) を定義し、アクセス可能なフィールド群およびそれらの関係を定義します。フォルダの関係はセグメントの関係と同一で、上位フォルダ、下位フォルダ、同位フォルダで構成されます。

データソースの任意のフィールドを使用して柔軟に構造を定義できますが、シノニムに対するレポートリクエストを発行する際は、データの検索パスが、DBMS エンティティダイアグラムによる制約および WebFOCUS の検索規則による制約に従う必要があります。

WebFOCUS ツールでクラスタシノニムを開く場合、デフォルト設定では、ディメンションノードはビジネスビューのフォルダとして作成されます。ノードまたはフォルダを追加することはできますが、存在する場合は、ディメンションビュー構造を使用することをお勧めします。

WebFOCUS ツールには、実セグメントではなく、フォルダのみが表示され、レポートの作成にはフォルダ構造内のフィールドのみがアクセス可能になります。

フォルダまたはフィールドにディメンションビューロールを割り当てるには、フォルダまたはフィールドを右クリックし、ディメンションビューロールのいずれかを選択します。

ディメンションビューロールは、フォルダまたはフィールドを明示的に割り当てることも、上位項目から自動的に継承されるよう指定することもできます。ディメンションビューロールを明示的に割り当てた場合、ビジネスビュープラス (BV+) 構造内で項目を別の場所に移動すると、その項目とともにロールも移動されます。ディメンションビューロールを明示的に割り当てない場合、項目を別の場所に移動すると、その項目のロールも変更されます。ただし、項目を [ドリルレベル] ロールのフィールド上にドロップした場合は例外です。[ドリルレベル] ロールのフィールド上にドロップすると、移動したフィールドに [ドリルレベル] ロールが継承されます。

割り当て可能なディメンションビューロールには、次のものがあります。

- **ディメンション**　ディメンションフィールドは、WebFOCUS ツールでダブルクリックするか、レポートキャンバスまたはグラフキャンバスにドラッグすることで、BY ソートフィールドとしてリクエストに自動的に追加されます。

  フォルダには [ディメンション] ロールを割り当てることができます。

  フィールドには [ディメンション (スタンドアロン)] または [ディメンション (ドリルレベル)] ロールを割り当てることができます。フィールドに [ディメンション (ドリルレベル)] ロールを割り当てた場合、そのフィールドは、フォルダ内のフィールドの順序に応じてレベルが決定される階層の一部になります。さらに、AUTODRILL をオンにすると、レポートまたはグラフ出力にオートドリルダウンが作成されます。

  フォルダに [ディメンション] ロールを割り当てた場合またはフィールドに [ディメンション (スタンドアロン)] ロールを割り当てた場合、シノニムのフォルダ宣言またはフィールド宣言に次の属性が追加されます。

  ```
  DV_ROLE=DIMENSION
  ```

フィールドに [ディメンション (ドリルレベル)] ロールを割り当てた場合、シノニムのフィールド宣言に次の属性が追加されます。

```
DV_ROLE=LEVEL
```

フォルダには、1 つのドリルダウン階層のみを含めることができます。ただし、各階層をそれぞれ異なるフォルダに配置することで、複数の階層に存在する同一フィールドを使用することはできます。ドリルレベル階層が含まれたフォルダは、階層として使用することに限定されません。このフォルダには、異なる DV_ROLE のフィールドを追加することもできます。

- **メジャー** メジャーフィールドは、WebFOCUS ツールでダブルクリックするか、レポートキャンバスまたはグラフキャンバスにドラッグすることで、集計 (SUM) フィールドとしてリクエストに自動的に追加されます (数値フィールドの場合)。メジャーが文字フィールドの場合は、BY ソートフィールドとして追加されます。フォルダまたはフィールドには [メジャー] ロールを割り当てることができます。

  フォルダまたはフィールドに [メジャー] ロールを割り当てた場合、シノニムのフォルダ宣言またはフィールド宣言に次の属性が追加されます。

  ```
  DV_ROLE=MEASURE
  ```

- **属性** 属性フィールドは、WebFOCUS ツールでダブルクリックするか、レポートキャンバスまたはグラフキャンバスにドラッグすることで、数値フィールドの場合は集計 (SUM) フィールドとして、文字フィールドの場合は BY ソートフィールドとしてリクエストに自動的に追加されます。フォルダまたはフィールドには、ロール属性を割り当てることができます。

  フォルダまたはフィールドに [属性] ロールを割り当てた場合、シノニムのフォルダ宣言またはフィールド宣言に次の属性が追加されます。

  ```
  DV_ROLE=ATTRIBUTE
  ```

- **フォルダ** ビジネスビューでは、フォルダは仮想セグメントです。フォルダには、[ディメンション]、[メジャー]、[属性] ロールのいずれかを割り当てることができます。

  注意：フォルダがフィールドの子として挿入された場合、その関係が PARENT_FIELD 属性で定義されます。デフォルト設定では、そのようなフォルダおよびそのフォルダ内のフィールドは、[属性] ロールと見なされます。

- **なし** フォルダまたはフィールドにロールが割り当てられていない場合、その項目には上位項目のロールが継承されます。ロールがすでに割り当てられている場合、そのロールを削除するには、上位項目からロールを継承するオプションを選択します。

例　ディメンションフォルダ定義のサンプル

PRODUCT フォルダの DV_ROLE は DIMENSION です。

```
FOLDER=PRODUCT, PARENT=FOLDER1,
   DV_ROLE=DIMENSION,
   DESCRIPTION='Product and Vendor', $
```

8

マスターファイルの確認と変更 - CHECK

CHECK コマンドを使用して、マスターファイルが正しく記述されていることを確認することができます。マスターファイルを作成した場合はこの操作を必ず実行します。CHECK コマンドを発行しないと、解析済みコピーが存在する場合にマスターファイルがメモリに再ロードされないことがあります。

トピックス

データソース記述の確認

CHECK コマンドを実行するとマスターファイル内のエラーが出力されるため、データソースを読み取る前にエラーを修正することができます。エラーを修正した後、CHECK コマンドを再実行してマスターファイルが正しく記述されていることを確認します。

構文　データソース記述の確認

```
CHECK FILE filename[.field] [PICTURE [RETRIEVE]] [DUPLICATE]
[HOLD [AS name] [ALL]]
```

説明

`filename`

作成したマスターファイル名です。

`field`

マスターファイルの代替ビューに使用されます。

`PICTURE`

データソースの全体構造を図示するオプションです。キーワードの PICTURE は PICT に短縮することができます。このオプションについての詳細は、323 ページの 「PICTURE オプション」 を参照してください。

RETRIEVE

TABLE または TABLEF コマンドを発行した際のセグメントの取得順序を表す図に変更します。なお、ユニークセグメントは親セグメントの論理的拡張として表示されます。キーワードの RETRIEVE は RETR に短縮することができます。

DUPLICATE

指定したデータソースの重複フィールド名をリスト表示します。キーワードの DUPLICATE は DUPL に短縮することができます。

HOLD

データソースのフィールドに関する情報が格納された一時的な HOLD ファイルおよび HOLD マスターファイルを生成します。HOLD ファイルを使用してレポートを作成することができます。データソースのフィールドの名前を指定するには AS オプションを使用します。このオプションについての詳細は、325 ページの 「 HOLD オプション 」 を参照してください。

name

HOLD ファイルおよび HOLD マスターファイルの名前です。

ALL

ファイルレベルの FDEFCENT および FYRTHRESH の値、フィールドレベルの DEFCENT および YRTHRESH の値を HOLD ファイルに追加します。

CHECK コマンドの出力

マスターファイルに構文エラーがある場合、CHECK コマンドがそれぞれのエラーに対応するメッセージを表示します。

参照 CHECK FILE コマンドの出力

データソース記述に構文エラーがない場合、CHECK コマンドは次のメッセージを表示します。

```
エラーの数 =      0
セグメントの数 =    n  (  実 =     n 仮想 =     n )
フィールドの数 =      n  インデックス =    n ファイル =       n
一時項目 =     n
全フィールドの長さの合計 =  n
```

説明

NUMBER OF ERRORS

マスターファイルの構文エラーの数を示します。

NUMBER OF SEGMENTS
マスターファイル内のセグメントの数です。ここには、クロスリファレンスセグメントも含まれます。

REAL
クロスリファレンス以外のセグメントの数です。このセグメントタイプには、Sn、SHn、U、ブランクがあります。

VIRTUAL
クロスリファレンスセグメントの数です。このセグメントタイプには、KU、KLU、KM、KL、DKU、DKM があります。

NUMBER OF FIELDS
マスターファイル内のフィールドの数です。

INDEXES
インデックスフィールドの数です。このフィールドには、マスターファイルで FIELDTYPE=I または INDEX=I 属性が指定されています。

FILES
フィールドが格納されているデータソースの数です。

NUMBER OF DEFINES
マスターファイル内の一時項目 (DEFINE) の数です。このメッセージは、一時項目 (DEFINE) が記述されている場合にのみ表示されます。

TOTAL LENGTH
FORMAT 属性 (FOCUS データソースの場合)、ACTUAL 属性 (FOCUS 以外のデータソース場合) のいずれかを指定してマスターファイルで定義されたすべてのフィールドの長さの合計です。

例 CHECK FILE コマンドの使用

次のコマンドを入力します。

```
CHECK FILE EMPLOYEE
```

次の情報が生成されます。

```
エラーの数 =      0
セグメントの数 =  11      ( 実 =    6 仮想 =  5 )
フィールドの数 =     34      インデックス =  0 ファイル =     3
全フィールドの長さの合計 = 365
```

一般的なエラーの原因

- FOCUS 以外のデータソースを使用する場合は、「全フィールドの長さの合計 (TOTAL LENGTH OF ALL FIELDS)」の値をチェックして、フィールドの長さが正確に指定されていることを確認してください。FOCUS 以外のデータソースからレポートを生成する場合の一般的なエラー原因の 1 つは、フィールドの長さを正しく指定していないことです。すべてのフィールドの長さ合計は、FOCUS 以外のデータソースの論理レコード長と一致していなければなりません。

  一般に、すべてのフィールドの長さ合計が FOCUS 以外のデータソースの論理レコード長と一致していない場合、少なくとも 1 つのフィールドの長さが正しく指定されていません。このエラーを修正しないと、外部データを正しく読み取れない場合があります。

- 次の警告メッセージが表示された場合は、同一セグメントに重複フィールド (同一フィールド名およびエイリアスを持つフィールド) が指定されています。2 回目に出現したフィールドはアクセスされません。

  ```
  (FOC1829) 警告。セグメントに重複フィールド名があります：fieldname
  ```

  同一セグメントに重複フィールドが存在するデータソースに対して CHECK コマンドを発行すると、FOC1829 メッセージが生成され、次のように重複フィールド名にアクセスできないことを示す警告メッセージが表示されます。

  ```
  (FOC1829) 警告。セグメントに重複フィールド名があります：BB
  同じセグメント内に同じフィールド名とエイリアス名をもつフィールドが 2 個あります。
  BB  IN SEGMENT SEGA        (VS SEGB
  ```

  DUPLICATE オプションが追加されている場合、出力結果に、最初の重複フィールドが存在する位置を示す警告メッセージが表示されます。

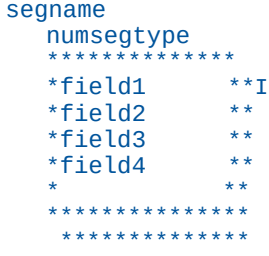
  ```
  WARNING: FOLLOWING FIELDS APPEAR MORE THAN ONCE
  AA IN SEGMENT SEGB        (VS SEGA)
  ```

PICTURE オプション

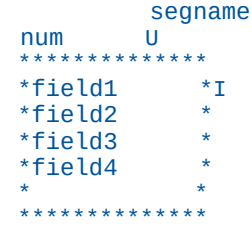
PICTURE オプションを使用すると、マスターファイルで定義された構造が図示されます。各セグメントがボックス単位で表示されます。ボックスには 4 つのタイプがあり、非ユニークセグメント、ユニークセグメント、実セグメント、クロスリファレンスセグメントに分類されます。4 つのタイプのボックスは次のとおりです。

実セグメント

(非ユニークセグメント)

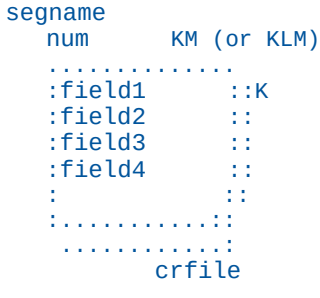


ユニークセグメント

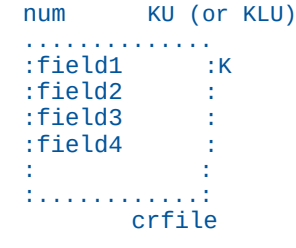


クロスリファレンスセグメント

(非ユニークセグメント)



ユニークセグメント



説明

`num`
構造内のセグメントに割り当てられた番号です。

`segname`
セグメントの名前です。

`segtype`
実セグメントで非ユニークのセグメントタイプです。このタイプには、Sn、SHn、N があります (N はブランクのセグメントタイプ)。

`field1...`
セグメント内のフィールドの名前です。フィールド名が 12 バイトを超える場合は、CHECK FILE PICTURE の実行時に末尾が切り捨てられて 12 バイトになります。その場合は、末尾に > 記号が表示されて実際の名前が表示された部分より長いことを示します。

`I`

インデックスフィールドであることを示します。

`K`

クロスリファレンスセグメントのキーフィールドであることを示します。

`crfile`

クロスリファレンスセグメントが存在する場合、クロスリファレンスデータソースの名前です。

上の図は、セグメント間の関係も示しています (次の例を参照)。親セグメントの図は子セグメントの図の上側に表示され、それぞれが直線で結合されています。

例　CHECK FILE PICTURE オプションの使用

下図は、SALARY データソースに結合された JOB データソースの構造を図示しています。

```
JOIN EMP_ID IN JOB TO EMP_ID IN SALARY
>
CHECK FILE JOB PICTURE
 NUMBER OF ERRORS=    0
 NUMBER OF SEGMENTS=  2  ( REAL=    1  VIRTUAL=   1 )
 NUMBER OF FIELDS=    7  INDEXES=   0  FILES=     2
 TOTAL LENGTH OF ALL FIELDS=   86
SECTION 01
             STRUCTURE OF FOCUS    FILE JOB      ON 01/31/03 AT 12.33.04

         JOBSEG
 01      S1
**************
*EMP_ID      **
*FIRST_NAME  **
*LAST_NAME   **
*JOB_TITLE   **
*            **
***************
 **************
       I
       I
       I
       I SALSEG
 02    I KU
..............
:EMP_ID      :K
:SALARY      :
:EXEMPTIONS  :
:            :
:            :
:............:
JOINED  SALARY
```

HOLD オプション

HOLD オプションを使用すると、一時的な HOLD ファイルが生成されます。HOLD ファイルについての詳細は、『TIBCO WebFOCUS Language リファレンス』を参照してください。この HOLD ファイルには、ファイル属性、セグメント属性、フィールド属性に関する詳細情報が格納され、TABLE リクエストを使用してこの情報をレポートに表示することができます。

この HOLD ファイルのフィールドの中には特別なフィールドがあります。特に記述がない限り、これらのフィールドにはマスターファイルで指定した属性と同一の名前が付けられます。各フィールドには、それぞれに対応する属性の値が格納されます。各フィールドは、ファイル属性、セグメント属性、フィールド属性に分類することができます。

ファイル属性

`FILENAME, SUFFIX, FDEFCENT, FYRTHRESH`

FDEFCENT および FYRTHRESH 属性は、元のマスターファイルに存在し、ALL オプションが指定されている場合に HOLD ファイルに格納されます。

セグメント属性

`SEGNAME, SEGTYPE`

このフィールドにはセグメントのキーフィールド数は含まれません。S1 および S2 などのセグメントタイプはタイプ S として表示されます。SHn セグメントタイプについても同様です。

`SKEYS`

セグメントのキーフィールド数です。たとえば、セグメントタイプが S2 の場合、SKEYS 値は 2 になります。

`SEGNO`

構造内でセグメントに割り当てられた番号です。この番号は図に表示されます。

`LEVEL`

構造内のセグメントのレベルです。ルートセグメントはレベル 1、その子セグメントはレベル 2 になります。

`PARENT, CRKEY, FIELDNAME`

フィールド属性

`ALIAS, FORMAT, ACTUAL`

TABLE リクエストに FORMAT フィールドを含める場合は、「FORMAT」というフィールド名を使用することはできません。その代わりに、USAGE のエイリアスまたは FORMAT フィールド名の一意の短縮名を使用します。一意の短縮名は FO です。

`DEFCENT, YRTHRESH`

これらの属性が元のマスターファイルに存在し、ALL オプションが指定されている場合に HOLD ファイルに格納されます。

例 CHECK FILE HOLD オプションの使用

次のサンプルプロシジャは、EMPLOYEE データソースの情報が記述された HOLD ファイルを作成します。続いて、EMPLOYEE データソース内のクロスリファレンスセグメント名、セグメントタイプ、フィールドの属性 (フィールド名、エイリアス、フォーマット) を表示するレポートを出力します。

```
CHECK FILE EMPLOYEE HOLD
TABLE FILE HOLD
HEADING
"FIELDNAMES, ALIASES, AND FORMATS"
"OF CROSS-REFERENCED FIELDS IN THE EMPLOYEE DATA SOURCE"
" "
PRINT FIELDNAME/A12 ALIAS/A12 USAGE BY SEGNAME BY SEGTYPE
WHERE SEGTYPE CONTAINS 'K'
END
```

出力結果は次のとおりです。

```
PAGE 1

FIELDNAMES, ALIASES, AND FORMATS
OF CROSS-REFERENCED FIELDS IN THE EMPLOYEE DATA SOURCE

SEGNAME    SEGTYPE            FIELDNAME      ALIAS   FORMAT
-------    -------            ---------      -----   ------
ATTNDSEG   KM                 DATE_ATTEND    DA      I6YMD
                              EMP_ID         EID     A9
COURSEG    KLU                COURSE_CODE    CC      A6
                              COURSE_NAME    CD      A30
JOBSEG     KU                 JOBCODE        JC      A3
                              JOB_DESC       JD      A25
SECSEG     KLU                SEC_CLEAR      SC      A6
SKILLSEG   KL                 SKILLS                 A4
                              SKILL_DESC     SD      A30
```

例　CHECK FILE HOLD ALL オプションの使用

ここでは、EMPLOYEE データソースに次のファイル宣言が記述されていることを前提にします。

```
FILENAME = EMPLOYEE, SUFFIX = FOC, FDEFCENT = 19, FYRTHRESH = 50
```

次のリクエストを実行します。

```
CHECK FILE EMPLOYEE HOLD ALL
TABLE FILE HOLD
PRINT FDEFCENT FYRTHRESH
END
```

次の出力が生成されます。

```
FDEFCENT    FYRTHRESH
--------    ---------
      19           50
```

HOLD オプションによる代替ファイル名の指定

CHECK コマンドで生成される一時的 な HOLD ファイルに AS 名を指定することができます。名前を指定しない場合はデフォルト名の HOLD が使用され、この名前のファイルが存在する場合は、上書きされます。

注意：他の CHECK オプションとの組み合わせて AS オプションを指定する場合は、AS holdname を最後に指定する必要があります。

TITLE 属性、HELPMESSAGE 属性、TAG 属性

CHECK コマンドの HOLD オプションを使用する場合、TITLE テキストは FLDATTR セグメントの TITLE フィールドに、HELPMESSAGE テキストは FLDATTR セグメントの HELPMESSAGE フィールドに、TAG 名は SEGATTR セグメントの TAGNAME フィールドにそれぞれ配置されます。

有効な JOIN が存在しない場合、または TAG 名を指定せずに JOIN コマンドを発行した場合、デフォルト設定で TAGNAME フィールドには CHECK コマンドで指定したデータソース名が配置されます。JOIN コマンドを TAGNAME 機能とともに発行した場合、TAGNAME フィールドにはホストデータソースおよびクロスリファレンスデータソースの TAG 名が配置されます。

マスターファイルの一時項目

HOLD オプションでは、一時項目 (DEFINE) は、データソースの実フィールドであった場合に実フィールドとして本来格納されるべきセグメントに配置されます。これはマスターファイルのフィールドの物理ロケーションとは限りませんが、フィールドを定義する式を評価するためにアクセスが必要な最下位セグメントです。フィールドの値が取得した値に依存しない場合、そのフィールドはデフォルト設定で最上位セグメントになります。これらのフィールドでは、FLDATTR セグメントの FLDSEG の値に 0 (ゼロ) が設定されます。マスターファイルでは FLDSEG フォーマットは I2S です。このフォーマットでは、0 (ゼロ) はブランクとしてレポートに表示されます。0 (ゼロ) を表示するには、TABLE リクエスト (FLDSEG/I2) で FLDSET のフォーマットを動的に再設定します。

データソースにデータを入力した後は、マスターファイルを自由に変更することができなくなります。任意の時点で変更しても影響を与えないものありますが、それ以外の変更はデータを再入力またはデータソースを再構築しない限り禁止されています。また、1 つの変更に関連する変更を同時に行う場合に限り許可される変更もあります。

テキストエディタを使用して、許可された範囲内でマスターファイルに変更を加えることができます。変更後は CHECK コマンドを使用する必要があります。

9

データソースのセキュリティ設定 - DBA

データベース管理者は、DBA セキュリティ機能を使用して FOCUS データソースにセキュリティを設定することができます。このセキュリティ機能を使用すると、ユーザが 1 つのレポートでリクエスト可能なレコード数または読み取り数を制限することができます。

DBA セキュリティ機能は、FOCUS 以外のデータソースにセキュリティを設定する場合にも使用することができます。なお、DBA セキュリティ機能では、WebFOCUS 以外からアクセスするデータソースを保護することはできません。

注意：FOCUS データソースに関するすべての説明は、XFOCUS データソースにも適用されます。

トピックス

データソースセキュリティの概要

DBA 機能には、次のようなセキュリティオプションが用意されています。

- USER 属性 - 特定のデータソースにアクセス可能なユーザを限定します。この属性についての詳細は、334 ページの 「 アクセス権限によるユーザの識別 - USER 属性 」 を参照してください。

- ACCESS 属性 - ユーザのアクセス権限を、読み取り、書き込み、更新のいずれかに制限します。この属性についての詳細は、339 ページの 「 アクセス権限タイプの指定 - ACCESS 属性 」 を参照してください。
- RESTRICT 属性 - ユーザアクセスを特定のフィールドまたはセグメントに制限します。この属性についての詳細は、343 ページの 「 データソースのアクセス制限 - RESTRICT 属性 」 を参照してください。
- RESTRICT 属性 - 取得するレコードを、確認テストにパスしたレコードのみに制限します。この属性についての詳細は、343 ページの 「 データソースのアクセス制限 - RESTRICT 属性 」 を参照してください。
- RESTRICT 属性 - ユーザがデータソースにアクセスして読み取り、書き込み、または変更できる値を制限します。この属性についての詳細は、343 ページの 「 データソースのアクセス制限 - RESTRICT 属性 」 を参照してください。
- SET DBASOURCE コマンド - マルチファイル構造でのアクセス制限のソースを制御することができます。このコマンドについての詳細は、350 ページの 「 マルチファイル構造でのアクセス制限のソース制御 」 を参照してください。
- DBAFILE 属性 - 別のマスターファイルに格納されているパスワードおよび制限を参照するように指定します。この属性についての詳細は、354 ページの 「 主マスターファイルへのセキュリティ情報の追加 」 を参照してください。
- WebFOCUS DBA イグジットルーチン - 外部セキュリティシステムで WebFOCUS パスワードを設定することができます。詳細は、『TIBCO WebFOCUS セキュリティ管理ガイド』を参照してください。
- プロシジャにセキュリティを設定することができます。この方法について詳細は、364 ページの 「 プロシジャのセキュリティ 」 WebFOCUS を参照してください。

データソースセキュリティの実装

WebFOCUS セキュリティは、ファイル単位で実装します。次の項目を指定して、DBA セキュリティ機能を簡単に実装することができます。

- データソースへのアクセス権限を与える WebFOCUS ユーザの名前またはパスワード。
- ユーザに与えるアクセス権限のタイプ。
- ユーザのアクセスを制限するセグメント、フィールド、データ値の範囲。

マスターファイルの END コマンドの後に宣言 (セキュリティ宣言と呼ばれる) を記述して、指定したデータソースにセキュリティが必要であること、および必要になるセキュリティタイプに関する情報を WebFOCUS に指示します。セキュリティ宣言は、次の 1 つまたは複数の属性で構成されます。

- DBA 属性 - データソースのデータベース管理者の名前またはパスワードを指定します。データベース管理者には、指定したデータソースおよびそのマスターファイルに無制限のアクセスが与えられます。
- USER 属性 - データソースの正規ユーザとなるユーザを識別します。セキュリティが設定された FOCUS データソースのマスターファイルでユーザ名またはパスワードが指定されている場合は、そのユーザのみがデータソースにアクセスすることができます。
- ACCESS 属性 - アクセス権限が与えられたユーザのアクセスタイプを定義します。次の 4 つのアクセスタイプを使用することができます。

  RW - データソースの読み取りと書き込み

  R - データソースの読み取り専用

  W - 新しいセグメントインスタンスのデータソースへの書き込み専用

  U - データソースのレコードの更新専用

- RESTRICT 属性 - ユーザにアクセス権限を与えないセグメントまたはフィールドを指定します。この属性は、ユーザが表示またはトランザクションを実行できるデータ値を制限する場合にも使用することができます。
- NAME 属性および VALUE 属性 - RESTRICT 宣言の一部として使用します。

データソースにセキュリティを設定するには、マスターファイルでこれらの属性値をカンマ (,) 区切りのフォーマットで指定します。これは、マスターファイルで他の属性を指定する場合と同様です。

マスターファイルで END のみが 1 行に配置されている場合、これはセグメントおよびフィールドの属性がそこで終了し、アクセス制限がその後に続くことを示しています。マスターファイルに END を配置する場合、少なくとも 1 つの DBA 属性をその後に続ける必要があります。

例　マスターファイルへのデータソースセキュリティの実装

次の例は、WebFOCUS DBA の記述例です。

```
FILENAME = PERS, SUFFIX = FOC,$
SEGMENT = IDSEG, SEGTYPE = S1,$
 FIELD = SSN          ,ALIAS = SSN    ,FORMAT = A9   ,$
 FIELD = FULLNAME     ,ALIAS = FNAME  ,FORMAT = A40  ,$
 FIELD = DIVISION     ,ALIAS = DIV    ,FORMAT = A8   ,$
SEGMENT=COMPSEG, PARENT=IDSEG, SEGTYPE=S1,$
 FIELD = SALARY       ,ALIAS = SAL    ,FORMAT = D8   ,$
 FIELD = DATE         ,ALIAS = DATE   ,FORMAT = YMD  ,$
 FIELD = INCREASE     ,ALIAS = INC    ,FORMAT = D6   ,$
END
DBA=JONES76,$
USER=TOM    ,ACCESS=RW, $
USER=BILL   ,ACCESS=R  ,RESTRICT=SEGMENT  ,NAME=COMPSEG    ,$
USER=JOHN   ,ACCESS=R  ,RESTRICT=FIELD    ,NAME=SALARY     ,$
                                           NAME=INCREASE   ,$
USER=LARRY  ,ACCESS=U  ,RESTRICT=FIELD    ,NAME=SALARY     ,$
USER=TONY   ,ACCESS=R  ,RESTRICT=VALUE    ,NAME=IDSEG,
   VALUE=DIVISION EQ 'WEST' ,$
USER=MARY   ,ACCESS=W  ,RESTRICT=VALUE    ,NAME=SALTEST,
   VALUE=INCREASE+SALARY GE SALARY,$
                                           NAME=HISTTEST,
   VALUE=DIV NE ' ' AND DATE GT 0,$
```

参照　データソースセキュリティを実装する際の特別な注意

- JOIN コマンドを使用する場合、データソースの DBA 情報が無視される可能性があります。JOIN 構造では、DBA 情報がホストマスターファイルから読み取られるため、このようなセキュリティ障害が発生します。この問題を解決するには、DBAFILE 機能を使用します。この方法についての詳細は、354 ページの 「 主マスターファイルへのセキュリティ情報の追加 」 を参照してください。JOIN 構造のすべてのデータソースは、DBAFILE でコーディングされたセキュリティ情報を取得します。
- マスターファイルの DBA セクションにコメントを使用することはできません。

データベース管理者の識別 - DBA 属性

セキュリティ属性として最初に指定するのは、データベース管理者を識別するパスワードです。このパスワードの最大長は 64 バイトで、大文字と小文字の区別はありません。このパスワードには、特殊文字を使用することができます。DBA パスワードにブランクを含める場合は、そのパスワードを一重引用符 (') で囲む必要があります。この行を終了するには、単に通常の区切り文字 (,$) を配置します。

注意

- データソースにアクセス制限が設定されている場合、そのデータソースには DBA 属性を指定する必要があります。
- 複数のクロスリファレンスデータソースが存在する場合、そのすべてのデータソースに同一の DBA 属性値を指定する必要があります。
- 分割データソースを USE コマンドで一括読み取りする場合、これらのデータソースには同一の DBA 属性値を指定する必要があります。
- データベース管理者には、データソースとクロスリファレンスデータソースすべてに無制限のアクセスが与えられます。そのため、DBA 属性を使用して、フィールド、セグメント、値の制限を指定することはできません。
- マスターファイルの暗号化および復号化や既存のデータソースの制限を行うには、DBA パスワードが必要です。
- データソースを使用する前に、すべてのセキュリティ属性を十分にテストしておく必要があります。特に、値の制限をテストして、エラーが発生しないことを確認しておくことが重要です。値をテストするには、仮の選別条件を追加したり、各リクエスト文の後に VALIDATE ステートメントを記述したりします。ユーザは値の制限について認識していないため、値の制限によってエラーが発生した場合、ユーザはその原因を理解できない場合があります。

例 DBA 属性によるデータベース管理者の識別

```
DBA=JONES76,$
```

手順 DBA パスワードを変更するには

データベース管理者は、すべてのセキュリティ属性を自由に変更することができます。既存の FOCUS データソースのマスターファイルで DBA パスワードを変更する場合は、RESTRICT コマンドを使用して、この変更で影響を受けるすべての FOCUS データソースにも変更後の DBA パスワードを格納する必要があります。この操作を怠ると、WebFOCUS はこの新しい記述が制限規則を無視していると見なします。影響を受けるすべてのデータソースに対して次の手順を実行します。

1. マスターファイルを編集して、DBA の古い値を新しい値に変更します。
2. 次のコマンドを発行します。

   ```
   SET PASS=old_DBA_password
   ```

3. 次のコマンドを発行します。

```
RESTRICT
mastername
END
```

4. 次のコマンドを発行します。

```
SET PASS=new_DBA_password
```

HOLD ファイルへの DBA 属性の追加

SET HOLDSTAT コマンドを使用して、DBA 情報およびコメントが格納されたデータソースを識別し、その情報を HOLD マスターファイルおよび PCHOLD マスターファイルに自動的に追加することができます。SET HOLDSTAT コマンドについての詳細は、『TIBCO WebFOCUS アプリケーション作成ガイド』を参照してください。

アクセス権限によるユーザの識別 - USER 属性

USER 属性は、データソースに正規アクセスを与えるユーザを識別するためのパスワードです。USER 属性は、この属性単独で使用することはできません。この属性の後に 1 つ以上の ACCESS 制限を追加して、ユーザに与えるアクセスタイプを指定する必要があります。詳細は、339 ページの 「 アクセス権限タイプの指定 - ACCESS 属性 」 を参照してください。

セキュリティが設定されたデータソースを使用する前に、ユーザは SET PASS または SET USER コマンドを使用してパスワードを入力する必要があります。ユーザが入力したパスワードがマスターファイルに存在しない場合、データソースへのアクセスは拒否されます。ユーザにパスワードが与えられていない場合、またはパスワードは与えられているがリクエストしたアクセスタイプに適合しない場合、次のメッセージが表示されます。

```
(FOC047) ユーザに十分なアクセス権限がありません。ファイル：filename
```

構文　USER 属性の設定

マスターファイルでユーザ名またはパスワードが宣言されていないユーザは、データソースへのアクセスが拒否されます。USER 属性の構文は次のとおりです。

```
USER = name
```

説明

`name`

ユーザのパスワードです。パスワードの最大長は 64 バイトです。パスワードには特殊文字を使用することができ、大文字と小文字の区別はありません。パスワードにブランクを含める場合は、パスワードを一重引用符 (') で囲む必要があります。

ブランクのパスワードを指定することができます。変更しない限り、これがデフォルト値です。ブランクのパスワードを指定すると、ユーザは SET PASS= コマンドを発行する必要はありません。ブランクのパスワードを指定した場合でもアクセス制限を設定できるため、多数のユーザに同一のアクセス権限を与える場合にこのパスワードが役立ちます。

例　USER 属性の設定

```
USER=TOM,...
```

次の例では、ユーザのパスワードをブランクに指定し、アクセス権限を読み取り専用に設定しています。

```
USER= , ACCESS=R,$
```

上書き禁止のユーザパスワード (SET PERMPASS)

PERMPASS パラメータを使用して、セッション中または接続中は継続して有効となるユーザパスワードを設定します。この設定は、サポートされているすべてのプロファイルで発行できますが、特にユーザプロファイルで発行すると、特定のユーザに限定してパスワードを作成できるため便利です。このパラメータは、ON TABLE 句で設定することはできません。このパラメータがすべてのユーザに適用されることを回避するため、EDASPROF では設定しないことをお勧めします。

PERMPASS が有効な場合は、既存のマスターファイルの DBA セクションで設定したすべてのセキュリティルールが適用されます。ユーザは、SET PASS または SET USER コマンドを発行して、別のセキュリティルールに従うユーザパスワードに変更を加えることはできません。この変更を加えようとすると、次のメッセージが表示されます。

```
(FOC32409) 恒久 PASS が有効化されています。これ以外は無視されます。 値は変更されていません。
```

注意: 1 回のセッションで設定できる永続パスワードは 1 つです。パスワードを一度設定すると、そのセッション内で変更することはできません。

構文　上書き禁止のユーザパスワードの設定

```
SET PERMPASS=userpass
```

説明

`userpass`
データソースに関連付けられたマスターファイルで DBA セキュリティルールが確立されている場合に、そのデータソースへのアクセスに使用するユーザパラメータです。

例　上書き禁止のユーザパスワードの設定

次の例は、DBA ルールを有効にした MOVIES マスターファイルを示しています。

```
DBA=USER1,$
USER = USERR,  ACCESS = R ,$
USER = USERU,  ACCESS = U ,$
USER = USERW,  ACCESS = W ,$
USER = USERRW, ACCESS = RW,$
```

次のプロシジャで永続パスワードが設定されます。

```
SET PERMPASS = USERU
TABLE FILE MOVIES
PRINT TITLE BY DIRECTOR
END
```

このユーザのアクセス権限は ACCESS=U に設定されているため、ファイルに対して TABLE リクエストを発行することはできません。

```
(FOC047) ユーザに十分なアクセス権限がありません。ファイル：MOVIES
コマンドの終わりまで処理をバイパスします
```

永続パスワードを変更することはできません。

```
SET PERMPASS = USERRW
```

```
(FOC32409) 恒久 PASS が有効化されています。これ以外は無視されます。
値は変更されていません
```

ユーザパスワードを変更することはできません。

```
SET PASS = USERRW
```

```
(FOC32409) 恒久 PASS が有効化されています。これ以外は無視されます。
値は変更されていません
```

パスワードの大文字小文字の区別

データベース管理者またはユーザが SET USER、SET PERMPASS、SET PASS コマンドのいずれかを発行すると、DBA 属性を含むマスターファイルすべてのデータソースへのアクセスを許可する前に、このユーザ ID の認証情報が確認されます。このパスワードは、プロシジャを暗号化または復号化する際にも確認されます。

SET DBACSENSITIV コマンドは、確認の前にパスワードを大文字に変換するかどうかを指定します。

構文　パスワードの大文字小文字の切り替え

```
SET DBACSENSITIV = {ON|OFF}
```

説明

ON
　パスワードを大文字に変換しません。ユーザ定義のパスワードと、マスターファイルまたはプロシジャのパスワードを比較する際は、常に大文字と小文字が区別されます。

OFF
　確認前にパスワードを大文字に変換します。ユーザ定義のパスワードと、マスターファイルまたはプロシジャのパスワードを比較する際は、大文字と小文字は区別されません。デフォルト値は OFF です。

例　パスワードの大文字小文字の切り替え

ここでは、EMPLOYEE マスターファイルに次の DBA 宣言を追加する場合について考察します。

```
USER = User2, ACCESS = RW,$
```

User2 は EMPLOYEE データソースからレポートを作成する必要があるため、次のコマンドを発行します。

```
SET USER = USER2
```

DBACSENSITIV を OFF に設定しているため、入力したパスワードの大文字と小文字がマスターファイルと一致しなくても、User2 はリクエストを実行することができます。

DBACSENSITIV を ON に設定すると、User2 は次のメッセージを受信します。

```
(FOC047) ユーザに十分なアクセス権限がありません。ファイル：filename
```

DBACSENSITIV を ON に設定する場合、ユーザは次のコマンドを発行する必要があります。

```
SET USER = User2
```

ユーザ ID の設定

セキュリティが設定された FOCUS データソースを使用する場合、ユーザはパスワードを入力する必要があります。1 名のユーザが、複数のファイルでそれぞれ異なるパスワードを入力する場合もあります。たとえば、ファイル ONE ではパスワード BILL の権限、ファイル TWO ではパスワード LARRY の権限をそれぞれ適用するような場合です。パスワードを設定するには、SET PASS コマンドを使用します。

構文　ユーザ ID の設定

```
SET {PASS|USER} = name [[IN {file|* [NOCLEAR]}], name [IN file] ...]
```

説明

`name`
ユーザ名またはパスワードです。 パスワードに使用する文字がオペレーティングシステム環境で特別な意味を持つ場合 (例、エスケープ文字)、プロシジャ内で SET USER コマンドを発行し、そのプロシジャを実行してパスワードを設定することができます。SET USER コマンドを発行する場合は、パスワードにブランクが含まれている場合でもパスワードを一重引用符 (') で囲む必要はありません。

`file`
パスワードを適用するマスターファイルの名前です。

`*`
すべてのファイルのパスワードを name の値で置換します。

`NOCLEAR`
特定のパスワードリストは変更せずに、有効なパスワードリストのパスワードをすべて置換します。

例　ユーザ ID の設定

次の例では、「TOM」というパスワードが、特定のパスワードが設定されていないすべてのデータソースで有効になります。

```
SET PASS=TOM
```

次の例では、ファイル ONE には「BILL」というパスワード、ファイル TWO には「LARRY」というパスワードが有効になります。その他のファイルにはパスワードは設定されていません。

```
SET PASS=BILL IN ONE, LARRY IN TWO
```

次の例では、ファイル SIX および SEVEN には「DAVE」というパスワード、それ以外のすべてのファイルには「SALLY」というパスワードが有効になります。

```
SET PASS=SALLY, DAVE IN SIX
SET PASS=DAVE IN SEVEN
```

次の例では、ファイル FIVE には「MARY」というパスワード、それ以外のすべてファイルには「FRANK」というパスワードが有効になります。

```
SET PASS=MARY IN FIVE,FRANK
```

特定のファイルに固有のパスワードを設定した場合、それらのファイルのリストは保持されます。このファイルリストを表示するには次のコマンドを発行します。

```
? PASS
```

パスワードを IN * (すべてのファイル) に設定すると、有効なパスワードテーブルはファイル名が関連付けられていない 1 つの値に集約されます。ファイル名のリストを保持するには、NOCLEAR オプションを使用します。

次の例では、すべてのファイルで有効なすべてのパスワードが「KEN」というパスワードに変更され、有効なパスワードテーブルが 1 つの値に集約されます。

```
SET PASS=KEN IN *
```

次の例では、ファイル NINE および TEN の現在有効なパスワードテーブルのすべてのパスワードが「MARY」というパスワードに変更され、その他のすべてのファイルでは「FRANK」というパスワードが有効になります。NOCLEAR オプションを使用すると、指定したファイルリストのすべてのパスワードをすばやく変更することができます。

```
SET PASS=BILL IN NINE,TOM IN TEN
SET PASS=MARY IN * NOCLEAR,FRANK
```

注意：COMBINE コマンドで結合されたデータソースが別のパスワードでセキュリティ設定されている場合は、FIND 関数を使用することはできません。

セキュリティ設定されたデータソースを使用する場合、ユーザはセッションごとに SET PASS コマンドでパスワードを発行する必要があります。ユーザはいつでもパスワードを発行して特定のデータソースにアクセスし、その後、別のパスワードを発行して他のデータソースにアクセスすることができます。

アクセス権限タイプの指定 - ACCESS 属性

ACCESS 属性を使用して、ユーザに与えるアクセス権限のタイプを指定します。DBA 宣言以外のすべてのセキュリティ宣言では、USER 属性および ACCESS 属性を必ず指定します。

次の例は、USER 属性および ACCESS 属性を含むセキュリティ宣言全体を示しています。

```
USER=TOM, ACCESS=RW,$
```

この宣言では、ユーザ Tom に対してデータソースの読み取りおよび書き込み (新しいセグメントインスタンスの追加) のアクセス権限が与えられます。

ACCESS 属性には 4 つの値のいずれかを割り当てることができます。その値は次のとおりです。

ACCESS=R	読み取り専用
ACCESS=W	書き込み専用
ACCESS=RW	データソースの読み取り、新しいセグメントの書き込み
ACCESS=U	更新専用

ユーザが発行できるコマンドの種類は、各ユーザに与えられたアクセスレベルにより異なります。ユーザに割り当てるアクセスレベルを決定する前に、各ユーザに必要なコマンドを検討する必要があります。ユーザがコマンドを使用するのに十分なアクセス権限が与えられていない場合、次のメッセージが表示されます。

```
(FOC047) ユーザに十分なアクセス権限がありません。ファイル：filename
```

ACCESS 属性で指定したアクセスレベルにより、ユーザがデータソースに対して実行できる操作が異なります。ユーザがアクセスできるフィールド、値、セグメントを制限するには、RESTRICT 属性を使用します。詳細は、343 ページの 「 データソースのアクセス制限 - RESTRICT 属性 」 を参照してください。すべての USER 属性に対して ACCESS 属性を割り当てる必要があります。RESTRICT 属性はオプションとして設定します。この属性を設定していないと、ユーザはデータソース内のフィールドおよびセグメントに無制限にアクセスすることができます。

アクセス権限のタイプ

下表のように、アクセス権限のタイプに応じて使用可能な WebFOCUS コマンドの種類が異なります。複数のアクセス権限のタイプに印が付いている場合、それは部分的にでもそのコマンドを使用できることを表しています。ただし、同一のコマンドでもユーザに与えられたアクセス権限のタイプにより使用方法が異なる場合がよくあります。

コマンド	R	W	RW	U	DBA
CHECK	X	X	X	X	X
CREATE			X		X
DECRYPT					X

コマンド	R	W	RW	U	DBA
DEFINE	X		X		X
ENCRYPT					X
MATCH	X		X		X
REBUILD			X		X
RESTRICT					X
TABLE	X		X		X

CHECK コマンド　DBA パスワードを持っていないユーザ、または読み取りと書き込み (RW) のアクセス権限が与えられていないユーザは、CHECK コマンドを制限付きで使用することができます。ただし、HOLD オプションが指定されている場合は、「パスワードによりアクセスが制限されました (ACCESS LIMITED BY PASSWORD)」という警告が表示され、DBA RESTRICT 属性に基づいて制限付きのフィールドが HOLD ファイルに継承されます。RESTRICT 属性についての詳細は、343 ページの 「 データソースのアクセス制限 - RESTRICT 属性 」 を参照してください。

CREATE コマンド　DBA パスワードを持っているユーザ、または読み取りと書き込み (RW) の権限が与えられたユーザのみが CREATE コマンドを発行することができます。

DECRYPT コマンド　DBA パスワードを持っているユーザのみが DECRYPT コマンドを発行することができます。

DEFINE コマンド　すべてのレポートコマンドと同様に、読み取り専用 (R) のアクセス権限が与えられたユーザは DEFINE コマンドを使用することができます。読み取り専用 (R) のアクセス権限が与えられた場合、ユーザはデータソースからレコードを読み取り、レポートの作成準備を行うことができます。書き込み専用 (W) または更新専用 (U) のアクセス権限が与えられた場合、ユーザは DEFINE コマンドを使用することはできません。

ENCRYPT コマンド　DBA パスワードを持っているユーザのみが ENCRYPT コマンドを使用することができます。

REBUILD コマンド　DBA パスワードを持っているユーザ、または読み取りと書き込み (RW) のアクセス権限が与えられているユーザのみが REBUILD コマンドを発行することができます。このコマンドは、FOCUS データソース専用です。

RESTRICT コマンド　DBA パスワードを持っているユーザのみが RESTRICT コマンドを使用することができます。

TABLE または MATCH コマンド　読み取り専用 (R) または読み取りと書き込み (RW) のアクセス権限が与えられたユーザは TABLE コマンドを使用することができます。書き込み専用 (W) または更新専用 (U) のアクセス権限が与えられたユーザはこのコマンドを使用することはできません。

参照　RESTRICT 属性のキーワード

RESTRICT 属性のキーワードは、CHECK コマンドで作成される HOLD ファイルに次のような影響を与えます。

`FIELD`
NAME パラメータで指定されたフィールドは HOLD ファイルに含まれません。

`SEGMENT`
NAME パラメータで指定されたセグメントは HOLD ファイルに含まれますが、そのセグメント内のフィールドは含まれません。

`SAME`
NAME パラメータで指定されたユーザと同一の動作特性になります。

`NOPRINT`
NAME または SEGNAME パラメータで指定されたフィールドは、ユーザがこれらのフィールドを参照できるため、HOLD ファイルに含まれます。

`VALUE`
VALUE パラメータで指定されたフィールドは、ユーザがこれらのフィールドを参照できるため、HOLD ファイルに含まれます。

CHECK コマンドを PICTURE オプションとともに発行すると、RESTRICT 属性のキーワードは結果の図に次のような影響を与えます。

`FIELD`
NAME パラメータで指定されたフィールドは図には含まれません。

`SEGMENT`
NAME パラメータで指定されたセグメントにはボックスが表示されますが、セグメント内のフィールドには表示されません。

`SAME`
NAME パラメータで指定されたユーザと同一の動作特性になります。

`NOPRINT`
このオプションは図に影響しません。

VALUE
このオプションは図に影響しません。

データソースのアクセス制限 - RESTRICT 属性

ACCESS 属性を使用して、ユーザがデータソースに対して実行可能な操作を制御します。

また、オプションの RESTRICT 属性を指定して、特定のフィールド、値、セグメントへのユーザアクセスを制限します。

RESTRICT=VALUE 属性は、IF 句でサポートされている条件で使用することができます。RESTRICT=VALUE_WHERE 属性は、WHERE 句でサポートされているすべての条件で使用することができます (例、フィールド間での比較、関数の使用)。WHERE 式は、可能な場合に構成済みのアダプタに渡されます。

構文　データソースのアクセス制限

```
...RESTRICT=level, NAME={name|SYSTEM} [,VALUE=test],$
```

または

```
...RESTRICT=VALUE_WHERE, NAME=name,  VALUE=expression; ,$
```

説明

`level`
次のいずれかの値です。

- **FIELD**　ユーザは NAME パラメータで指定されたフィールドにアクセスすることはできません。
- **SEGMENT**　ユーザは NAME パラメータで指定されたセグメントにアクセスすることはできません。
- **PROGRAM**　ユーザがデータソースを使用するたびに NAME パラメータで指定されたプログラムが呼び出されます。
- **SAME**　ユーザには NAME パラメータで指定されたユーザと同一の制限が適用されます。ネストされた SAME ユーザは 4 名まで有効です。
- **NOPRINT**　NAME または SEGMENT パラメータで指定されたフィールドをリクエストステートメントに記述することはできますが、そのフィールドは表示されません。VALUE テストを使用して、制限の対象を式の条件を満たす値のみに限定することができます。たとえば、次のように RESTRICT=NOPRINT 宣言を記述した場合、ユーザ MARY が表示できるのは、給与が 10000 未満の従業員の ID のみです。

```
USER=MARY   ,ACCESS=R  ,RESTRICT=NOPRINT  ,NAME=EMP_ID ,
   VALUE=CURR_SAL LT 10000;,$
```

注意：RESTRICT=NOPRINT が設定されたフィールドは、表示コマンド (動詞) で参照することはできますが、すべての種類のフィルタコマンド (例、IF、WHERE、FIND、LOOKUP、VALIDATE) で参照することはできません。

`name`
制限の対象とするフィールドまたはセグメントの名前です。これを NOPRINT の後に使用すると、フィールドの名前のみが対象になります。値のテストでのみ使用可能な NAME=SYSTEM は、下位のセグメントを含めてデータソース内のすべてのセグメントが制限の対象になります。制限の対象として複数のフィールドまたはセグメントを指定するには、1 名のユーザに対して RESTRICT 属性を複数回発行します。

注意：値の制限を使用する場合、NAME=segment を指定すると、代替ファイルビューにより検索ビューが変更されるかどうかに関係なく、指定されたセグメントと、階層内でそのセグメントの下位にあるセグメントすべてが制限の対象になります。つまり、親セグメントに値の制限があり、JOIN または代替ファイルビューにより子セグメントが新しいルートになった場合でも、元の親セグメントに対する値の制限が新しいルートにそのまま適用されます。

`VALUE`
ユーザは test パラメータで指定されたテスト条件を満たす値にのみアクセスすることができます。

`test`
データがこの値のテスト条件を満たす場合に限り、ユーザはこの値にアクセスすることができます。このテスト条件は、IF 句でサポートされる式です。

`VALUE_WHERE`
ユーザは expression パラメータで指定されたテスト条件を満たす値にのみアクセスすることができます。

`expression;`
データがこの値のテスト条件を満たす場合に限り、ユーザはこの値にアクセスすることができます。このテスト条件は、WHERE 句でサポートされる式です。

注意：セミコロン (;) が必要です。

例　VALUE_WHERE による値へのアクセス制限

次の DBA 宣言を GGSALES マスターファイルの末尾に追加します。これらの宣言により、USER1 に West 地域および「C」という文字で始まる製品へのアクセス権限が与えられます。

```
END
DBA = USERD,$
USER = USER1, ACCESS = R, NAME = SALES01, RESTRICT = VALUE_WHERE,
       VALUE = REGION EQ 'West' AND PRODUCT LIKE 'C%'; ,$
```

次のリクエストは、USER1 にパスワードを設定し、REGION、CATEGORY、PRODUCT 別に売上および個数の合計を計算します。

```
SET USER = USER1
TABLE FILE GGSALES
SUM DOLLARS UNITS
BY REGION
BY CATEGORY
BY PRODUCT
END
```

出力結果には、マスターファイル内の WHERE 式を満たす地域および製品のみが表示されます。

```
Region       Category     Product           Dollar Sales  Unit Sales
------       --------     -------           ------------  ----------
West         Coffee       Capuccino               915461       72831
             Food         Croissant              2425601      197022
             Gifts        Coffee Grinder          603436       48081
                          Coffee Pot              613624       47432
```

RESTRICT=VALUE_WHERE 属性を RESTRICT=VALUE 属性に変更した場合、この式は無効になります。次のメッセージが表示され、リクエストは実行されません。

```
(FOC002) 無効な語句です：LIKE 'C%'
```

例　データソースのアクセス制限

```
USER=BILL ,ACCESS=R ,RESTRICT=SEGMENT ,NAME=COMPSEG,$
```

フィールドまたはセグメントのアクセス制限

RESTRICT 属性を使用して、ユーザのアクセスを制限するフィールドまたはセグメントを識別します。RESTRICT 属性で指定されていないフィールドまたはセグメントはすべてアクセス可能になります。

RESTRICT 属性を使用しない場合、ユーザはデータソース全体にアクセスすることができます。ユーザのアクセス権限を新しいレコードの読み取り、書き込み、更新のいずれかに限定することはできますが、データソースのすべてのレコードは処理に使用することができます。

構文　フィールドまたはセグメントのアクセス制限

```
...RESTRICT=level,  NAME=name,$
```

説明

`level`
次のいずれかの値です。

`FIELD` - ユーザは NAME パラメータで指定されたフィールドにアクセスすることはできません。

`SEGMENT` - ユーザは NAME パラメータで指定されたセグメントにアクセスすることはできません。

`SAME` - ユーザには NAME パラメータで指定されたユーザと同一の制限が適用されます。

`NOPRINT` - NAME または SEGMENT パラメータで指定されたフィールドをリクエストステートメントに記述することはできますが、そのフィールドは表示されません。これを NOPRINT の後に使用すると、フィールドの名前のみが対象になります。
RESTRICT=NOPRINT が設定されたフィールドは、表示コマンド (動詞) で参照することはできますが、すべての種類のフィルタコマンド (例、IF、WHERE、FIND、LOOKUP、VALIDATE) で参照することはできません。

`name`
制限の対象とするフィールドまたはセグメントの名前です。これを NOPRINT の後に使用すると、フィールドの名前のみが対象になります。

値のテストでのみ使用可能な NAME=SYSTEM は、下位のセグメントを含めてデータソース内のすべてのセグメントが制限の対象になります。制限の対象として複数のフィールドまたはセグメントを指定するには、1 名のユーザに対して RESTRICT 属性を複数回発行します。

注意

- NAME 属性にフィールドまたはセグメントが記述されている場合、ユーザがそのフィールドまたはセグメントを取得することはできません。リクエストステートメントでこのようなフィールドまたはセグメントが記述されている場合、そのリクエストはユーザのアクセス権限を超えているとして拒否されます。NOPRINT を使用した場合、フィールドまたはセグメントを記述することはできますが、データは表示されません。このデータは、文字フォーマットのフィールドではブランク、数値フォーマットのフィールドでは 0 (ゼロ) として表示されます。RESTRICT=NOPRINT が設定されたフィールドは、表示コマンド (動詞) で参照することはできますが、すべての種類のフィルタコマンド (例、IF、WHERE、FIND、LOOKUP、VALIDATE) で参照することはできません。
- 複数のフィールドまたはセグメントを制限するには、RESTRICT ステートメントを複数回記述します。たとえば、Harry に対してフィールド A とセグメント B の使用を制限するには、次のアクセス制限を発行します。

```
USER=HARRY, ACCESS=R, RESTRICT=FIELD,    NAME=A,$
   RESTRICT=SEGMENT, NAME=B,$
```

- フィールドとセグメントのアクセス制限は、必要に応じていくつでも追加することができます。
- RESTRICT=SAME は、複数のパスワードに共通した一連の制限を繰り返し使用する場合に役立ちます。新しいユーザに RESTRICT=SAME を指定し、USER 属性の NAME 値で指定されたユーザ名またはパスワードを使用すると、このユーザにはこの NAME 属性で指定されたユーザと同一の制限が適用されます。必要に応じて、他の制限を後から追加することができます。

例　セグメントのアクセス制限

次の例では、「Bill」というユーザに、データソースの COMPSEG セグメントを除くすべてのデータへの読み取り専用のアクセス権限が与えられています。

```
USER=BILL ,ACCESS=R ,RESTRICT=SEGMENT ,NAME=COMPSEG,$
```

例　共通のアクセス制限の再利用

次の例では、Sally および Harry に Bill と同一のアクセス権限が与えられています。また、Sally は SALARY フィールドの読み取りが制限されています。

```
USER=BILL, ACCESS=R, RESTRICT=VALUE, NAME=IDSEG,
     VALUE=DIVISION EQ 'WEST',$
USER=SALLY, ACCESS=R, RESTRICT=SAME, NAME=BILL,$
                      RESTRICT=FIELD, NAME=SALARY,$
USER=HARRY, ACCESS=R, RESTRICT=SAME, NAME=BILL,$
```

注意：特定のセグメントへのアクセス制限は、その下位にも影響します。

値のアクセス制限

RESTRICT 属性にテスト条件を設定して、ユーザアクセスを特定の値のみに制限することができます。ユーザが使用できる値は、テスト条件に一致する値に限定されます。

値を制限する方法は 2 つあります。ユーザがデータソースから読み取れる値を制限する方法と、ユーザがデータソースに書き込める値を制限する方法です。これらの 2 つの制限は、互いに独立して機能します。ACCESS 属性を使用して、ユーザが読み取れる値と書き込める値のどちらを制限するかを指定します。

ユーザが読み取れる値を制限するには、ACCESS=R および RESTRICT=VALUE を設定します。このタイプの制限を設定すると、ユーザが参照できるデータは RESTRICT 属性で指定されたテスト条件に一致する値に限定され、その他すべての値は参照できなくなります。RESTRICT 属性に ACCESS=R を使用すると、レポートリクエストで指定する強制 IF ステートメントのように機能します。そのため、ACCESS=R で値を制限する構文は、レポートリクエストの IF テストの規則に従う必要があります。

構文　ユーザが読み取り可能な値の制限

```
...ACCESS=R, RESTRICT=VALUE, NAME=name, VALUE=test,$
```

説明

`name`
テスト条件を有効にするセグメントの名前です (テスト条件として参照されている場合)。データソースのすべてのセグメントを指定するには、NAME=SYSTEM を使用します。

`test`
実行するテストです。

例　ユーザが読み取り可能な値の制限

```
USER=TONY, ACCESS=R, RESTRICT=VALUE, NAME=IDSEG,
     VALUE=DIVISION EQ 'WEST',$
```

この制限では、Tony は WEST 地区からのレコードのみを読み取ることができます。

テスト条件の式は VALUE= の後に入力します。テスト条件の構文は、レコードの選別時に TABLE コマンドで使用する構文と同一です。ただし、この句の前に「IF」という語句は使用しません。TABLE コマンドで使用する選別条件についての詳細は、『TIBCO WebFOCUS Language リファレンス』を参照してください。複数のフィールドでテストを実行する場合は、VALUE 属性をさらに追加する必要があります。各テストには、テストを適用するセグメント名をそれぞれ指定する必要があります。以下はその例です。

```
USER=DICK, ACCESS=R, RESTRICT=VALUE, NAME=IDSEG,
     VALUE=DIVISION EQ 'EAST' OR 'WEST',$
     NAME=IDSEG,
     VALUE=SALARY LE 10000,$
```

1 つのテスト条件が行の許容する長さを超える場合は、複数のセクションに分割して記述することができます。各セクションは VALUE= 属性で開始し、終了文字 (,$) で終了する必要があります。以下はその例です。

```
USER=SAM, ACCESS=R, RESTRICT=VALUE, NAME=IDSEG,
     VALUE=DIVISION EQ 'EAST' OR 'WEST',$
     VALUE=OR 'NORTH' OR 'SOUTH',$
```

注意：値の制限を指定した 2 行目以降は、キーワードの OR で開始する必要があります。

テスト条件は、テストを適用するデータセグメントの親セグメントに対して適用することができます。ここでは、次の例について考察します。

```
USER=DICK, ACCESS=R, RESTRICT=VALUE, NAME=IDSEG,
     VALUE=DIVISION EQ 'EAST' OR 'WEST',$
     NAME=IDSEG,
     VALUE=SALARY LE 10000,$
```

「SALARY」というフィールドは、実際には COMPSEG というセグメントの一部です。NAME=IDSEG でテストを指定しているため、このテストは親セグメントの IDSEG に対するリクエストで有効になります。この場合、「PRINT FULLNAME」というリクエストでは、このテストが下位セグメント IDSEG の一部となるフィールドで実行されたにも関わらず、このテスト条件に一致する従業員、つまり給与が $10,000 以下の従業員の氏名のみが表示されます。ただし、このテストが COMPSEG 上 (つまり NAME=COMPSEG) で有効になった場合、データソースの全員の氏名が取得されますが、このテスト条件に一致する給与情報のみが取得されます。

値の読み取りと書き込みの制限

ユーザに適用する値の制限として、ACCESS=W (データ更新用) と ACCESS=R (データ検索用) の両方を発行すると便利な場合が多くあります。これにより、ユーザがデータソースに書き込める値とユーザが実際に閲覧できる値の両方が制限されます。この制限を適用するには、RESTRICT=VALUE 属性の発行時に ACCESS=R を追加します。これにより、ユーザはテスト条件で指定された値以外の値を参照できなくなります。さらに、RESTRICT=VALUE 属性の発行時に ACCESS=W を使用すると、このユーザに書き込み制限が追加されます。ACCESS=RW を使用してこの設定を行うことはできません。

注意：このマニュアルに説明はありませんが、書き込み制限はデータ管理機能に適用されます。

マルチファイル構造でのアクセス制限のソース制御

DBASOURCE パラメータにより、マルチファイル構造へのアクセスを許可するセキュリティ属性が指定されます。デフォルト設定では、アクセス制限は JOIN 構造のホストファイル、または COMBINE 構造の最後のファイルに基づいて決定されます。DBASOURCE パラメータを「ALL」に設定した場合、JOIN または COMBINE 構造のファイルすべてのアクセス制限が適用されます。

注意：JOIN 構造および COMBINE 構造のファイルすべてのアクセス制限を格納して適用するには、DBAFILE を作成して実行することもできます。主マスターファイルにアクセス制限を含める方法についての詳細は、354 ページの 「 主マスターファイルへのセキュリティ情報の追加 」 を参照してください。

SET DBASOURCE コマンドは 1 回のセッションまたは接続につき、一度だけ発行することができます。それ以上コマンドを発行しようとしても、2 回目以降は無視されます。値がプロファイルで設定されている場合、セッション中にユーザが変更することはできません。

DBASOURCE=ALL の場合は、次のようになります。

- JOIN 構造に対する TABLE リクエストでは、ユーザが構造内の各ファイルに最低読み取り権限を所有している場合に限り、クロスリファレンスファイルおよびクロスリファレンスセグメントへのアクセスが許可されます。

DBASOURCE=HOST の場合は、次のようになります。

- TABLE リクエストでは、ユーザは JOIN 構造のホストファイルの読み取り権限が必要です。セキュリティ制限は、すべてホストファイルから適用されます。構造内のファイルのセキュリティ制限を適用するには、DBAFILE を作成して有効にすることもできます。

構文　JOIN または COMBINE 構造のアクセス制限適用制御

```
SET DBASOURCE = {HOST|ALL}
```

説明

HOST
DBAFILE を使用して、構造内の別ファイルのアクセス制限を適用する場合を除き、JOIN 構造のホストファイル、または COMBINE 構造の最後のファイルに、アクセス制限を限定します。デフォルト値は HOST です。

ALL
ユーザには、JOIN 構造および COMBINE 構造のファイルすべてに読み取り権限が必要です。このファイルに INCLUDE、DELETE、または UPDATE コマンドを発行する場合、ユーザに WRITE、UPDATE、または READ/WRITE アクセス権限が必要です。

参照　SET DBASOURCE 使用時の注意

- JOIN 構造および COMBINE 構造のファイルには、すべて同一の DBA パスワードが設定されている必要があります。DBA 属性が同一でない場合、構造にアクセスする方法はありません。
- SET DBASOURCE コマンドがセッション中に複数回発行された場合は、次のメッセージが表示され、値は変更されません。

```
(FOC32575) DBASOURCE を再設定できません。
値は、変更されていません
```

例　JOIN 内のアクセス制限制御

次のリクエストにより、TRAINING データソースが EMPDATA データソースと COURSE データソースに結合され、JOIN 構造に対してリクエストが発行されます。

```
JOIN CLEAR *
JOIN COURSECODE IN TRAINING TO COURSECODE IN COURSE AS J1
JOIN PIN IN TRAINING TO PIN IN EMPDATA AS J2
TABLE FILE TRAINING
PRINT COURSECODE AS 'CODE' CTITLE
   LOCATION AS 'LOC'
BY LASTNAME
WHERE COURSECODE NE '    '
WHERE LOCATION EQ 'CA' OR LOCATION LIKE 'N%'
END
```

マスターファイルに DBA 属性が存在しない場合、出力は次のようになります。

```
LASTNAME        CODE     CTITLE                                LOC
--------        ----     ------                                ---
ADAMS           EDP750   STRATEGIC MARKETING PLANNING          NJ
CASTALANETTA    EDP130   STRUCTURED SYS ANALYSIS WKSHP         NY
                AMA130   HOW TO WRITE USERS MANUAL             CA
CHISOLM         EDP690   APPLIED METHODS IN MKTG RESEARCH      NJ
FERNSTEIN       MC90     MANAGING DISTRIBUTOR SALE NETWORK     NY
GORDON          SFC280   FUND OF ACCTG FOR SECRETARIES         NY
LASTRA          MC90     MANAGING DISTRIBUTOR SALE NETWORK     NY
MARTIN          EDP130   STRUCTURED SYS ANALYSIS WKSHP         CA
MEDINA          EDP690   APPLIED METHODS IN MKTG RESEARCH      NJ
OLSON           PU168    FUNDAMNETALS OF MKTG COMMUNICATIONS   NY
RUSSO           PU168    FUNDAMNETALS OF MKTG COMMUNICATIONS   NY
SO              BIT420   EXECUTIVE COMMUNICATION               CA
WANG            PU440    GAINING COMPETITIVE ADVANTAGE         NY
WHITE           BIT420   EXECUTIVE COMMUNICATION               CA
```

さらに、次の DBA 属性を TRAINING マスターファイルの最後に追加します。

```
END
DBA = DBA1,$
USER = TUSER, ACCESS =R,$
```

同一のリクエストを実行すると、次のメッセージが生成されます。

```
(FOC047) ユーザに十分なアクセス権限がありません。ファイル：TRAINING
コマンドの終わりまで処理をバイパスします
```

さらに、次の SET PASS コマンドを発行します。

```
SET PASS = TUSER
```

次の DBA 属性を COURSE マスターファイルの最後に追加します。

```
END
DBA = DBA1,$
USER = CUSER, ACCESS = R,$
```

次の DBA 属性を EMPDATA マスターファイルの最後に追加します。

```
END
DBA = DBA1,$
USER = EUSER, ACCESS = R,$
```

DBA 属性の値は、すべてのマスターファイルで共通です。

リクエストを再度実行します。セキュリティ違反は発生せずに、レポート出力が生成されます。DBASOURCE パラメータは、デフォルト設定で「HOST」に設定されているため、ホストファイル内のみで有効なパスワードを使用して、リクエストを実行することができます。

さらに、DBASOURCE パラメータを「ALL」に設定します。

```
SET DBASOURCE = ALL
SET PASS = TUSER
```

TUSER は COURSE データソースで有効なユーザではないため、リクエストを実行すると、次のようなメッセージが生成されます。

```
(FOC052) フィールドに対するパスワードが無効です
```

さらに、次の SET PASS コマンドを発行して、各ファイルに有効なパスワードを設定します。

```
SET PASS = TUSER IN TRAINING, CUSER IN COURSE, EUSER IN EMPDATA
```

ここで、リクエストを実行し、レポート出力を生成することができます。

SET DBASOURCE コマンドの発行後、値を変更することはできません。次の SET コマンドは、値の「HOST」への変更を試みますが、クエリコマンド出力は、変更が実行されなかったことを示しています。

```
>  > set dbasource = host
(FOC32575) DBA SOURCE を再設定できません。
値は、変更されていません
```

JOIN 条件への DBA 制限の追加

複数セグメント構造のリクエストに DBA 制限を適用すると、デフォルト設定で、その制限がリクエストの WHERE 条件として追加されます。DBAJOIN パラメータを ON に設定すると、DBA 制限が、それらが指定されているファイルまたはセグメントに対して内部的な制限として処理され、JOIN 構文に追加されます。

制限の対象とするファイルまたはセグメントに構造上の親が含まれ、その JOIN が OUTER JOIN または UNIQUE JOIN の場合、この違いが重要になります。

制限がレポートフィルタとして処理される場合、そのフィルタに一致しない下位セグメントインスタンスは、ホストセグメントとともにレポート出力から省略されます。ホストセグメントが省略されるため、出力には OUTER JOIN または UNIQUE JOIN が正しく反映されません。

制限が JOIN 条件として処理される場合、その JOIN 条件に一致しない下位セグメントインスタンスはミッシング値として表示され、レポート出力にはホスト行がすべて表示されます。

詳細は、『TIBCO WebFOCUS Language リファレンス』を参照してください。

主マスターファイルへのセキュリティ情報の追加

DBAFILE 属性を使用して、複数のマスターファイルのすべてのパスワードおよびセキュリティ制限を 1 つのファイルに集約することができます。個々のマスターファイルは、この主マスターファイルを参照します。複数のマスターファイルで同一の DBA パスワードを使用する場合は、共通の DBAFILE でこの DBA パスワードを共有することができます。

この方法には次のような利点があります。

- パスワードを複数のデータソースに適用する場合でもパスワードを別々に格納する必要がないため、パスワードの管理が簡略化されます。
- JOIN または COMBINE コマンドを使用して、異なるユーザパスワードを使用するデータソースを結合することができます。また、JOIN または COMBINE で結合された各データソースでそれぞれの DBA 情報は保持されます。

主 DBAFILE は標準的なマスターファイルです。DBAFILE 属性で主マスターファイルの名前を指定することにより、他のマスターファイルでも主マスターファイルのパスワードおよびセキュリティ制限を使用することができます。

注意

- 同一の DBAFILE を指定したすべてのマスターファイルは、共通の DBA パスワードを持ちます。
- 管理用の DBAFILE では、通常のマスターファイルのように END ステートメントの前に 1 つ以上のセキュリティ宣言と 1 つのフィールド宣言を記述して、DBA 情報が含まれていることを示す必要があります。必須の属性に特定の値を割り当てない場合でも、そのことをカンマ (,) で明示する必要があります。DBAFILE の DBA パスワードは、このファイルを参照する他のすべてのマスターファイルのパスワードと共通です。これにより、個人が各自のセキュリティを変更することを防止します。すべてのマスターファイルは暗号化する必要があります。
- DBAFILE には、DBA パスワードの後にパスワードおよび制限を列記することができます。これらのパスワードは、この DBAFILE を参照するすべてのデータソースに適用されます。
- DBAFILE では、共通パスワードの後にデータソース固有のパスワードおよび一般パスワードの追加分を指定することができます。この機能を実装するには、DBAFILE の DBA セクションに FILENAME 属性を追加します (例、FILENAME=TWO)。FILENAME 属性についての詳細は、358 ページの 「 DBAFILE ファイル名の規則 」 を参照してください。

- データソースに固有の制限は、指定したデータソースに適用された一般の制限を上書きします。これらの制限が競合する場合は、FILENAME セクションのパスワードが優先されます。たとえば、DBAFILE の共通セクションに ACCESS=RW が指定されている場合でも、特定のデータソースに対して FILENAME セクションを追加し、同一のパスワードに ACCESS=R を指定することができます。
- 追加した値の制限はすべて累積されるため、データを取得する場合はすべての値の制限を満足させなければなりません。以降の例では、PASS=JOE を 2 回使用しています。JOE はすべてのデータソースに適用された共通パスワードですが、FILENAME=THREE では「RESTRICT=...」というデータソース THREE にのみ適用される別の制限が追加されています。

構文　主マスターファイルへのセキュリティ属性の追加

```
END
DBA=dbaname, DBAFILE=filename ,$
```

説明

`dbaname`
主マスターファイルの dbaname と同一の名前です。

`filename`
主マスターファイル名です。

DBAFILE でパスワードと制限を指定し、その DBAFILE を参照するすべてのマスターファイルにそのパスワードと制限を適用することができます。また、DBAFILE に FILENAME 属性を追加して、特定のマスターファイルにのみ適用するパスワードと制限を指定することもできます。

例　主マスターファイルへのセキュリティ属性の追加

次の例は、複数のマスターファイルで「FOUR」という共通の DBAFILE を共有する方法を示しています。

```
ONE MASTER
FILENAME=ONE
  .
  .
END
DBA=ABC, DBAFILE=FOUR,$
```

```
TWO MASTER
FILENAME=TWO
  .
  .
END
DBA=ABC, DBAFILE=FOUR,$

THREE MASTER
FILENAME=THREE
  .
  .
END
DBA=ABC,
DBAFILE=FOUR,$

FOUR MASTER
FILENAME=FOUR,$
SEGNAME=mmmmm,$
FIELDNAME=fffff,$
END
DBA=ABC,$
   PASS=BILL,ACCESS=R,$
   PASS=JOE,ACCESS=R,$
FILENAME=TWO,$
   PASS=HARRY,ACCESS=RW,$
FILENAME=THREE,$
   PASS=JOE,ACCESS=R,RESTRICT=...,$
   PASS=TOM,ACCESS=R,$
```

例　JOIN 構造での DBAFILE の使用

次のリクエストにより、TRAINING データソースが EMPDATA データソースと COURSE データソースに結合され、JOIN 構造に対してリクエストが発行されます。

```
JOIN CLEAR *
JOIN COURSECODE IN TRAINING TO COURSECODE IN COURSE AS J1
JOIN PIN IN TRAINING TO PIN IN EMPDATA AS J2
TABLE FILE TRAINING
PRINT COURSECODE AS 'CODE' CTITLE
   LOCATION AS 'LOC'
BY LASTNAME
WHERE COURSECODE NE '   '
WHERE LOCATION EQ 'CA' OR LOCATION LIKE 'N%'
END
```

マスターファイルに DBA 属性が存在しない場合、出力は次のようになります。

```
LASTNAME        CODE     CTITLE                                LOC
--------        ----     ------                                ---
ADAMS           EDP750   STRATEGIC MARKETING PLANNING          NJ
CASTALANETTA    EDP130   STRUCTURED SYS ANALYSIS WKSHP         NY
                AMA130   HOW TO WRITE USERS MANUAL             CA
CHISOLM         EDP690   APPLIED METHODS IN MKTG RESEARCH      NJ
FERNSTEIN       MC90     MANAGING DISTRIBUTOR SALE NETWORK     NY
GORDON          SFC280   FUND OF ACCTG FOR SECRETARIES         NY
LASTRA          MC90     MANAGING DISTRIBUTOR SALE NETWORK     NY
MARTIN          EDP130   STRUCTURED SYS ANALYSIS WKSHP         CA
MEDINA          EDP690   APPLIED METHODS IN MKTG RESEARCH      NJ
OLSON           PU168    FUNDAMNETALS OF MKTG COMMUNICATIONS   NY
RUSSO           PU168    FUNDAMNETALS OF MKTG COMMUNICATIONS   NY
SO              BIT420   EXECUTIVE COMMUNICATION               CA
WANG            PU440    GAINING COMPETITIVE ADVANTAGE         NY
WHITE           BIT420   EXECUTIVE COMMUNICATION               CA
```

EMPDATA マスターファイルがリクエストの主 DBAFILE になります。次の DBA 属性を EMPDATA マスターファイルの最後に追加します。

```
END
DBA=DBA1,$
USER = EUSER, ACCESS = R,$
FILENAME = COURSE
USER = CUSER2, ACCESS=RW,$
```

これらの属性により、EUSER ユーザには、DBAFILE として EMPDATA を使用するすべてのファイルへの読み取りアクセス権限が与えられます。CUSER2 ユーザには、COURSE データソースへの読み取りと書き込みのアクセス権限が与えられます。

次のセキュリティ属性を COURSE マスターファイルの最後に追加します。これらの属性により、EMPDATA マスターファイルが主ファイルになり、このファイル内のセキュリティ情報が COURSE データソースへのアクセスに使用されます。これにより、DBA 属性の値が、EMPDATA マスターファイルの DBA 属性の値と同一になります。

```
END
DBA = DBA1, DBAFILE=EMPDATA,$
```

次のセキュリティ属性を TRAINING マスターファイルの最後に追加します。これらの属性により、EMPDATA マスターファイルが主ファイルになり、このファイル内のセキュリティ情報が TRAINING データソースへのアクセスに使用されます。これにより、DBA 属性の値が、EMPDATA マスターファイルの DBA 属性の値と同一になります。

```
END
DBA = DBA1, DBAFILE=EMPDATA,$
```

これで、この JOIN 構造に対してリクエストを実行するには、JOIN 構造の各ファイルへの読み取りアクセス権限を持つ現在のユーザパスワードが必要になります。次の SET PASS コマンドを発行してリクエストを実行します。

```
SET PASS = EUSER
```

または

```
SET PASS = EUSER IN *
```

EUSER ユーザは JOIN 構造の各ファイルで有効なユーザであるため、リクエストが実行され、出力結果が生成されます。

EMPDATA マスターファイルは CUSER2 ユーザを COURSE マスターファイルの有効なユーザとして識別するため、次の SET PASS コマンドを発行してリクエストを実行することもできます。

```
SET USER = EUSER IN EMPDATA, EUSER IN TRAINING, CUSER2 IN COURSE
```

JOIN 構造の各ファイルに無効なパスワードを指定して SET PASS コマンドを発行すると、次のいずれかのメッセージが生成され、リクエストは実行されません。

```
(FOC052) フィールドに対するパスワードが無効です：フィールド名
```

```
(FOC047) ユーザに十分なアクセス権限がありません。ファイル：filename
```

DBAFILE ファイル名の規則

DBAFILE に FILENAME 属性を追加して特定のマスターファイルを指定する場合、参照する側のマスターファイルの FILENAME 属性と、DBAFILE の DBA セクションで指定する FILENAME 属性は一致させる必要があります。これにより、ユーザがマスターファイルの名前を変更して、DBAFILE が理解できない名前になることが回避されます。

例 DBAFILE 名の規則

```
ONE MASTER
FILENAME=XONE
   .
   .
   .
END
DBA=ABC, DBAFILE=FOUR,$


FOUR MASTER
FILENAME=XFOUR
   .
   .
   .
END
DBA=ABC,$
   .
   .
   .
FILENAME=XONE,$
   .
   .
   .
```

ONE MASTER は、リクエストでは「TABLE FILE ONE」として参照されます。ただし、ONE MASTER および DBAFILE である FOUR MASTER の DBA セクションの両方で FILENAME=XONE を指定します。

セキュリティ上の理由から、DBAFILE 情報を含むマスターファイルの FILENAME 属性には、マスターファイル名とは異なる名前を指定することをお勧めします。なお、FOUR MASTER では FILENAME 属性に「XFOUR」という名前を指定しています。

DBAFILE による既存 DBA システムへの接続

既存のシステムでは、新しい属性である DBAFILE を使用しない場合、システムの特性に変更はありません。現在のシステムでは、複数のデータソースを JOIN コマンドで結合した場合、リスト内の最初のデータソースがコントロールデータソースになります。そのデータソースのパスワードのみが検証の対象になります。複数のデータソースを COMBINE コマンドで結合した場合は、最後のデータソースのパスワードのみが有効になります。この場合、すべてのデータソースで DBA パスワードを一致させる必要があります。

新しいシステムでは、JOIN または COMBINE で結合したすべてのデータソースの DBA セクションが検証の対象になります。マスターファイルに DBAFILE が含まれている場合は、そのパスワードおよびセキュリティ制限が読み込まれます。JOIN または COMBINE のリストでデータソースの DBA セクションを有効にするには、そのデータソースの DBAFILE を指定します。

新しいシステムの使用を開始した段階で、すべてのマスターファイルを変換します。データベース管理者が既存のシステムを変換する際に別の物理 DBAFILE を作成したくない場合は、DBAFILE 属性でデータソース自体を指定することもできます。

例 DBAFILE による既存の DBA システムへの接続

```
FILENAME=SEVEN,
  SEGNAME=..
    FIELDNAME=...
     .
     .
     .
END
DBA=ABC,DBAFILE=SEVEN,$    (OR DBAFILE= ,$)
 PASS=...
 PASS=...
```

DBAFILE によるアプリケーションの結合

各データソースにはそれぞれ独自の制限が設定されているため、異なるアプリケーションのデータソースや異なるユーザパスワードを持つデータソースを取り扱う場合でも、これらのデータソースを JOIN コマンドおよび COMBINE コマンドで結合することができます。ここでの要件は、各データソースで有効なパスワードのみです。ここで、現在のシステムのパスワードを割り当てることにより、異なるデータベース管理者の管理下のアプリケーションに別のアプリケーションのアクセス権限を与えることができます。

データソースに選別条件を割り当て、データソースにアクセスするすべてのレポートリクエストにその選別条件を自動的に適用することができます。詳細は、『TIBCO WebFOCUS Language リファレンス』を参照してください。

セキュリティ属性の概要

下表は、WebFOCUS で使用するセキュリティ属性の一覧です。

属性	エイリアス	最大の長さ	説明
DBA	DBA	8	割り当てる値は、データソースに無制限でアクセス可能なデータベース管理者 (DBA) のコード名です。
USER	PASS	8	割り当てる値は、セキュリティ制限を設定するユーザを識別するための任意のコード名です。

属性	エイリアス	最大の長さ	説明
ACCESS	ACCESS	8	このユーザのアクセスレベルです。次の値があります。 R - 読み取り専用 W - 新しいセグメントの書き込み専用 RW - 読み取りと書き込み U - 値の更新専用
RESTRICT	RESTRICT	8	このアクセスレベルに適用する制限のタイプです。次の値があります。 SEGMENT FIELD VALUE SAME NOPRINT
NAME	NAME	66	制限の対象となるセグメントまたはフィールドの名前、あるいは呼び出すプログラムの名前です。
VALUE	VALUE	制限なし	制限のタイプとして RESTRICT=VALUE を指定した場合に、結果が TRUE となるテスト条件の式です。
DBAFILE	DBAFILE	8	使用するパスワードおよび制限が格納されたマスターファイルの名前です。

制限規則の非表示 - ENCRYPT コマンド

FOCUS データソースの制限情報はマスターファイルに保存されるため、ユーザがその制限規則を検査できないようにマスターファイルを暗号化する必要があります。記述した内容を暗号化できるのはデータベース管理者のみです。ENCRYPT コマンドを発行する前に、PASS=DBAname を設定しておく必要があります。なお、ENCRYPT コマンドの構文は、オペレーティングシステムの種類により異なります。

注意：暗号化するマスターファイルの 1 行目は、68 バイト以下にする必要があります。68 バイトを超える場合、複数行に分割する必要があります。

構文　制限規則の非表示 - ENCRYPT コマンド

```
ENCRYPT FILE filename
```

説明

`filename`
　暗号化するファイルの名前です。

例　マスターファイルの暗号化と復号化

次の例は、プロシジャ全体を示しています。

```
SET PASS=JONES76
ENCRYPT FILE PERS
```

制限を変更する場合は、上記の逆の手順を実行します。その場合は、DECRYPT コマンドを使用して、記述内容を読み取り可能な形式で保存します。

ファイルを復号化する前に、SET コマンドで DBA パスワードを発行しておく必要があります。以下はその例です。

```
SET PASS=JONES76
DECRYPT FILE PERS
```

データの暗号化

マスターファイルに ENCRYPT パラメータを使用して、すべてのセグメントまたはその一部を暗号化することもできます。暗号化したファイルを外部メディア (ディスクまたはテープ) に保存した場合でも、各ファイルは許可されていないアクセスに対してセキュリティが確保されます。

暗号化はセグメントレベルで実行されます。つまり、セグメント全体が暗号化されます。暗号化をリクエストするには、マスターファイルで ENCRYPT 属性を ON に設定します。

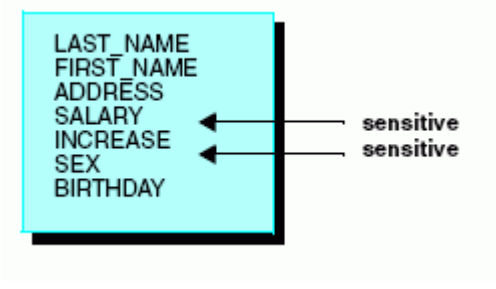
例　データの暗号化

```
SEGMENT=COMPSEG, PARENT=IDSEG, SEGTYPE=S1, ENCRYPT=ON,$
```

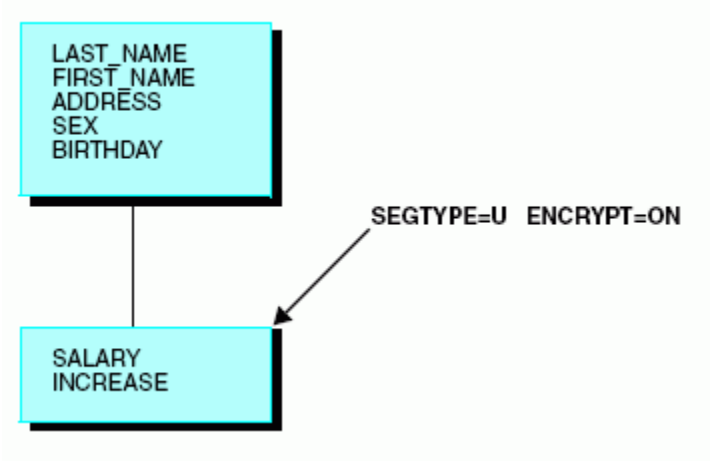
データソースにデータを入力する前に、ENCRYPT パラメータを指定する必要があります。暗号化をはじめてリクエストする場合、「NEW FILE...」というメッセージが表示されます。マスターファイルに変更を加えて暗号化を後からリクエストすることはできません。また、リクエスト後またはデータソースに何らかのデータを入力した後では暗号化を解除することはできません。

暗号化したデータのパフォーマンスに関する注意

データを暗号化すると処理効率がわずかに低下します。処理効率の低下を最小限に抑えるには、セグメント上で暗号化の必要なデータフィールドのみをグループ化し、それを独立したユニークセグメント (SEGTYPE=U) として元のセグメントの下に配置します。たとえば、セグメント上に次のようなデータ項目があることを想定します。



次のようにグループ化します。



注意：DBA パスワードを変更するには、RESTRICT コマンドを発行する必要があります。詳細は、333 ページの 「 DBA パスワードを変更するには 」 を参照してください。

プロシジャのセキュリティ

データセキュリティに関する問題の大部分は、WebFOCUS DBA イグジットルーチンを使用して解決するのが最善の方法です。WebFOCUS DBA イグジットルーチンについての詳細は、『TIBCO WebFOCUS セキュリティ管理ガイド』を参照してください。また、プロシジャを暗号化および復号化することもできます。

プロシジャの暗号化と復号化

プロシジャの実行をユーザに許可した場合でも、プロシジャ内のテキストデータを機密扱いにする必要があります。それは、プロシジャに機密情報が含まれていたり、権限のないユーザによるプロシジャの変更を回避するためです。保存したプロシジャを権限のないユーザから保護するには、ENCRYPT コマンドを使用します。

暗号化したプロシジャは、すべてのユーザが実行できますが、その内容を参照するにはプロシジャを復号化する必要があります。プロシジャを復号化できるのは、暗号化パスワードを持っているユーザのみです。

プロシジャを暗号化または復号化するためにユーザが選択したパスワードはエディタには表示されず、使用するファイルの DBA パスワードとは関連性はありません。

構文　プロシジャの暗号化と復号化

次のプロシジャを使用して「SALERPT」という名前のプロシジャを暗号化します。

```
SET PASS = DOHIDE
ENCRYPT FILE SALERPT FOCEXEC
```

次のプロシジャを使用して「SALERPT」という名前のプロシジャを復号化します。

```
SET PASS = DOHIDE
DECRYPT FILE SALERPT FOCEXEC
```

10

データソースの作成と再構築

CREATE コマンドを使用して、新規のデータソースを作成したり、既存のデータソースの初期化を再実行したりすることができます。

データソースの作成後、ディスク領域を効率的に使用するためにデータソースの再構成が必要になる場合があります。また、コンテンツ、インデックス、データソース構造の変更や、レガシー日付フィールドの SmartDate フィールドへの変更が必要になる場合もあります。REBUILD コマンドを使用すると、これらの操作をすべて実行することができます。

CREATE および REBUILD コマンドは、FOCUS、XFOCUS データソースで使用することができます。また、CREATE コマンドを使用して、データアダプタが正しくインストールされたデータソースにリレーショナルテーブルを作成することもできます。

この章の FOCUS データソースに関するすべての説明は、XFOCUS データソースにも適用されます。

トピックス

新しいデータソースの作成 - CREATE コマンド

CREATE コマンドを使用して、マスターファイルに新しいブランクの FOCUS データソースを作成することができます。また、CREATE コマンドを使用して、既存の FOCUS データソースのデータを削除することもできます。

データアダプタが正しくインストールされている場合は、CREATE コマンドはリレーショナルテーブル (例、DB2、Teradata) にも使用することができます。詳細は、関連するデータアダプタのマニュアルを参照してください。

データソースがすでに存在する場合に CREATE FILE コマンドを発行すると、FOCUS または XFOCUS データソースでは次のメッセージが表示されます。

```
(FOC441) 警告。すでにファイルが存在します。CREATE しますか?
```

CREATE FILE コマンドの DROP オプションは、このメッセージの表示を省略してデータソースを作成します。この場合、必要に応じて既存のテーブルを最初に削除し、マスターファイルが変更された場合はマスターファイルを再解析します。

構文　CREATE コマンドの使用

```
CREATE FILE mastername [DROP]
```

説明

`mastername`
　データソースを記述するマスターファイル名です。

`DROP`
　必要に応じて、CREATE コマンドを実行する前に既存のファイルを削除し、マスターファイルを再解析します。警告メッセージは生成されません。

　外部インデックスまたは MDI を含む FOCUS または XFOCUS データソースに対して CREATE FILE *filename* DROP コマンドを発行した場合は、データソースの作成後にインデックスを再構築する必要があります。

CREATE コマンドを入力すると、以下の行が表示されます。

```
NEW FILE name ON date AT time
```

説明

`name`
　新しいデータソースの完全な名前です。

```
ON date AT time
```
データソースが作成または再作成された日時です。

データソースがすでに存在する場合、DROP オプションを指定せずに CREATE コマンドを発行すると、次のメッセージが表示されます。

```
(FOC441) 警告。すでにファイルが存在します。CREATE しますか?
```

データソースを削除してブランクの新しいデータソースを作成するには「Y」と入力します。コマンドを取り消してデータソースを変更しない場合は、「END」、「QUIT」、「N」のいずれかを入力します。

データソースのファイルの整合性を保持するには、CREATE コマンドを発行する前に次のコマンドを発行します。

```
SET SHADOW=ON
```

例 Windows での FOCUS データソースの再作成

EMPLOYEE データソースを再作成するには、次のコマンドを発行します。

```
CREATE FILE EMPLOYEE
```

次のメッセージが表示されます。

```
(FOC441) 警告。すでにファイルが存在します。CREATE しますか?
```

次のように入力します。

```
YES
```

次のメッセージが表示されます。

```
NEW FILE C:EMPLOYEE.FOC     ON 01/03/2003 AT 15.48.57
```

EMPLOYEE データソースはディスクに存在しますが、レコードは格納されていません。

データソースの再構築 - REBUILD コマンド

FOCUS データソースを作成した後、REBUILD コマンドを使用してデータソースの構造に変更を加えることができます。REBUILD コマンドおよびそのサブコマンドのいずれかを使用して、次の操作を実行することができます。使用可能なサブコマンドには、REBUILD、REORG、INDEX、EXTERNAL INDEX、CHECK、TIMESTAMP、DATE NEW、MDINDEX があります。

- 分散されたデータソースを再構築する (REBUILD)。

- 選択条件に基づいてインスタンスを削除する (REBUILD または REORG)。
- 既存のデータソースを再設計する。この再設計には、セグメントの追加と削除、データフィールドの追加と削除、複数フィールドへのインデックスの追加、文字データフィールドの長さの変更などがあります (REORG)。
- データソースを再構築または作成した後に、新しいフィールドにインデックスを付ける (INDEX)。
- レコードを結合またはレコードの場所を特定する際にインデックス検索を容易にする外部インデックスデータベースを作成する (EXTERNAL INDEX)。
- データソース構造の整合性を確認する (CHECK)。FOCUS データソースが最後に変更された日時を確認する (TIMESTAMP)。
- レガシー日付フォーマットを SmartDate フォーマットに変換する (DATE NEW)。
- マルチディメンションインデックス (MDI) を構築または編集する (MDINDEX)。

REBUILD 機能は、次の方法で使用することができます。

- バッチ処理として REBUILD コマンド、サブコマンド、サブコマンドの応答を順に入力する。処理のプロンプトがそれぞれ別の行に表示されます。

REBUILD 機能を使用する前に、ファイルの割り当て、セキュリティ認可、バックアップに関する必須要件および推奨要件について考慮する必要があります。

参照 REBUILD を使用する前の要件

REBUILD 機能を使用する前に、いくつかの要件について考慮する必要があります。

- **割り当て** 通常は REBUILD コマンドを使用する前にワークスペースを割り当てる必要はありません。ワークスペースは自動的に割り当てられます。ただし、適切なサイズのワークスペースが必要になります。一般に、REBUILD および REORG オプションには既存のファイルサイズより 10 パーセントから 20 パーセント大きいサイズを使用します。

  ワークスペースには、常に「REBUILD」というファイル名が割り当てられます。REORG コマンドの DUMP フェーズでは、LOAD フェーズを別の時点で実行するための割り当てステートメントが表示されます。

- **セキュリティ認可** 再構築するデータソースがデータベース管理者によって保護されている場合、REBUILD 機能を実行するには読み取りおよび書き込みアクセス権限が必要です。データソースのセキュリティについての詳細は、329 ページの 「 データソースのセキュリティ設定 - DBA 」 を参照してください。

- **バックアップ**　これは必須要件ではありませんが、REBUILD サブコマンドを使用する前に、元のマスターファイルおよびデータソースのバックアップコピーを作成しておくことをお勧めします。

手順　REBUILD 機能を使用するには

次の手順は、REBUILD 機能の使用方法を示しています。

1. 次のコマンドを入力して REBUILD 機能を初期化します。

```
REBUILD
```

2. サブコマンドの名前または番号を入力してサブコマンドのいずれかを選択します。サブコマンドの名前および対応する番号は次のとおりです。

```
1. REBUILD          (データベース構造最適化)
2. REORG            (データベース構造の再編成)
3. INDEX            (インデックスの追加/変更)
4. EXTERNAL INDEX   (外部インデックスの追加/変更)
5. CHECK            (データベース構造のチェック)
6. TIMESTAMP        (タイムスタンプの変更)
7. DATE NEW         (OLD DATE フォーマットを SMARTDATE フォーマットに変換)
8. MDINDEX          (多次元インデックスの追加/変更)
```

選択するサブコマンドにより、次に続く応答が異なります。通常はデータソース名の入力のみが要求されますが、その他の情報を入力する場合もあります。

REBUILD メッセージ表示頻度の制御

REBUILD がデータソースを処理する場合、ステータスメッセージを定期的に表示します。たとえば、「REFERENCE..AT SEGMENT 1000」というメッセージは再構築の進捗状況を示します。REBUILD の取得フェーズおよびロードフェーズでのデフォルト表示頻度は、セグメントインスタンスの処理数 1000 件につき 1 回です。表示されるメッセージ数は、再構築する FOCUS データソースのセグメント数を表示頻度で除算した値になります。

構文　REBUILD メッセージ表示頻度の制御

REBUILD は「REFERENCE..AT SEGMENT *segnum*」メッセージを定期的に表示して、データソースの処理状況を示します。REBUILD がこのメッセージを表示する頻度を制御するには、次のコマンドを発行します。

```
SET REBUILDMSG = {n|1000}
```

説明

n

1,000 から 99,999,999 までの整数または 0 (ゼロ) です。0 を指定するとメッセージが無効になります。

1000 未満の値を設定すると、次のように動作します。

- 有効な値が 0 (ゼロ) または 1000 以上であることを知らせる警告メッセージを生成する。
- 現在の設定をそのまま使用する。現在の設定は、デフォルトの 1000 または前回の REBUILDMSG 設定の 1000 以上の整数値です。

例 REBUILD メッセージ表示頻度の制御

次の例は、REBUILD CHECK で生成されたメッセージを示しています。ここでは、REBUILDMSG が 4000 に設定され、データソースには 19,753 件のレコードが格納されています。

```
STARTING..
REFERENCE..AT SEGMENT    4000
REFERENCE..AT SEGMENT    8000
REFERENCE..AT SEGMENT   12000
REFERENCE..AT SEGMENT   16000
NUMBER OF SEGMENTS RETRIEVED=   19753
CHECK COMPLETED...
```

ファイルサイズの最適化 - REBUILD サブコマンド

REBUILD サブコマンドは、次の 2 つの目的で使用されます。第 1 の目的は、データのアクセス時間および格納領域の効率性を向上することです。大量のデータを削除すると、データの物理構造と論理構造が一致しなくなります。REBUILD の REBUILD サブコマンドは、データを一時的なワークスペースにダンプした後、データを再ロードします。この処理でインスタンスを最適な論理順序に戻します。第 2 の目的は、REBUILD の REBUILD サブコマンドを使用して一連の選択条件でセグメントインスタンスを削除することです。

通常、REBUILD サブコマンドはデータソースのクリーンな状態を維持する目的で使用します。データソースの再構築が必要がどうかを確認するには、? FILE コマンドを入力します。詳細は、387 ページの 「? FILE と TABLEF による構造的な整合性の確認」 を参照してください。

```
? FILE filename
```

データソースが分散されている場合は、次のメッセージが表示されます。

```
FILE APPEARS TO NEED THE -REBUILD-UTILITY
REORG PERCENT IS A MEASURE OF FILE DISORGANIZATION
0 PCT IS PERFECT -- 100 PCT IS BAD
REORG PERCENT x%
```

このメッセージは、REORG PERCENT の値が 30 パーセントを超える場合に必ず表示されます。REORG PERCENT の値は、データソースの物理的な配置が論理的な配置と異なる度合いを示します。

&FOCDISORG 変数は、? FILE コマンドの直後に使用して、データソースの分散状態をパーセント (%) で示します。&FOCDISORG 変数は、データソースの分散状態が 30 パーセント未満の場合にもこの値を表示します。詳細は、『TIBCO WebFOCUS アプリケーション作成ガイド』を参照してください。

手順 REBUILD サブコマンドを使用するには

次の手順は、REBUILD サブコマンドの使用方法を示しています。

1. 次のコマンドを入力して REBUILD 機能を初期化します。

   ```
   REBUILD
   ```

   次のオプションがあります。

   ```
   1. REBUILD         (データベース構造最適化)
   2. REORG           (データベース構造の再編成)
   3. INDEX           (インデックスの追加/変更)
   4. EXTERNAL INDEX (外部インデックスの追加/変更)
   5. CHECK           (データベース構造のチェック)
   6. TIMESTAMP       (タイムスタンプの変更)
   7. DATE NEW        (OLD DATE フォーマットを SMARTDATE フォーマットに変換)
   8. MDINDEX         (多次元インデックスの追加/変更)
   ```

2. 次のように入力して REBUILD サブコマンドを選択します。

   ```
   REBUILD or 1
   ```

3. 再構築するデータソースの名前を入力します。

   UNIX、Windows では *filename* を入力します。 再構築するデータソースを参照するには、USE コマンドを使用します。USE コマンドが動作しない場合は、EDAPATH 変数を使用してデータソースを検索することができます。

4. 選択テストを指定せずにデータソースの再構築のみを実行する場合は、次のように入力します。

   ```
   NO
   ```

   REBUILD 処理が即時に実行されます。

   また、REBUILD サブコマンドに選別条件を追加する場合は、次のように入力します。

   ```
   YES
   ```

最終行に ,$ を追加して選択テストを入力します。

テスト間の関係には、EQ、NE、LE、GE、LT、GT、CO (含む)、OM (含まない) のいずれかを使用することができます。複数のテストは AND 演算子で結合し、リテラルのリストは OR で結合することができます。テストを終了するには、カンマとドル記号の組み合わせ (,$) を配置する必要があります。

たとえば、次のように入力します。

```
A EQ A1 OR A2 AND B LT 100 AND
C GT 400 AND D CO 'CUR',$
```

REBUILD の REBUILD サブコマンドの処理が完了すると、取得したセグメント数および再構築したデータソースに存在するセグメント数が統計値として表示されます。

例 Windows での REBUILD サブコマンドの使用

手順は次のとおりです。

```
1. REBUILD
2. REBUILD
3. EMPLOYEE
4. NO
```

1. REBUILD 機能を初期化します。
2. REBUILD サブコマンドを指定します。
3. 構築するデータソースの名前を入力します。
4. レコード選択テストが必要でないことを示します。

データソースが再構築され、統計が生成されます。

データソース構造の変更 - REORG サブコマンド

FOCUS データソースにデータを入力後、REORG サブコマンドを使用してマスターファイルにさまざまな変更を加えることができます。REORG サブコマンドは 2 つの処理で構成されます。最初にデータを一時的なワークスペースにダンプし、次にそのデータを新しいマスターファイルに再ロードします。

REORG サブコマンドを使用して次の操作を実行することができます。

- 新規セグメントを既存セグメントの下位セグメントとして追加する。
- セグメントを削除する。

- 新規データフィールドを既存セグメントの下位データフィールドとして追加する。

  注意：このフィールドはキーフィールドの後に追加する必要があります。
- データフィールドを削除する。
- セグメント内のキーフィールド以外のデータフィールドの順序を変更する。キーフィールドを変更することはできません。
- ユニークセグメントのフィールドを親セグメントに昇格させる。
- 親セグメントのフィールドを下位ユニークセグメントに降格させる。
- フィールドにインデックスを付ける、またはフィールドからインデックスを削除する。
- 文字データフィールドの長さを変更する。

次の操作は REORG サブコマンドで実行することはできません。

- フィールドのフォーマットタイプを変更する (例、文字から数値、数値から文字、数値フォーマットタイプの変更)。
- SEGNAME 属性の値を変更する。
- SEGTYPE 属性の値を変更する。
- インデックスフィールドの名前を変更する。

手順　REORG サブコマンドを使用するには

次の手順は、REORG コマンドを使用する方法を示しています。

1. 元のマスターファイルを変更する前に、別のファイル名でコピーを作成します。
2. エディタを使用して、マスターファイルのコピーを編集します。
3. 次のコマンドを入力して REBUILD 機能を初期化します。

   ```
   REBUILD
   ```

   次のオプションがあります。

   ```
   1. REBUILD          (データベース構造最適化)
   2. REORG            (データベース構造の再編成)
   3. INDEX            (インデックスの追加/変更)
   4. EXTERNAL INDEX   (外部インデックスの追加/変更)
   5. CHECK            (データベース構造のチェック)
   6. TIMESTAMP        (タイムスタンプの変更)
   7. DATE NEW         (OLD DATE フォーマットを SMARTDATE フォーマットに変換)
   8. MDINDEX          (多次元インデックスの追加/変更)
   ```

4. 次のように入力して REORG サブコマンドを選択します。

   ```
   REORG or 2
   ```

   次のオプションがあります。

   ```
   1. DUMP          (データベース内容の DUMP)
   2. LOAD          (データベースにデータを LOAD)
   ```

5. 次のように入力して、この手順の DUMP フェーズを初期化します。

   ```
   DUMP or 1
   ```

6. ダンプ元のデータソースの名前を入力します。このフェーズで使用する元のマスターファイル名を必ず入力します。

   UNIX、Windows では *filename* を入力します。 再構築するデータソースを参照するには、USE コマンドを使用します。USE コマンドが動作しない場合は、EDAPATH 変数を使用してデータソースを検索することができます。

7. 選択テストを指定するには「YES」と入力します。ここで指定した条件を満足するデータのみがダンプされます。ほとんどの場合はデータソース全体をダンプします。その場合は次のように入力します。

   ```
   NO
   ```

   DUMP フェーズでは、ダンプされたセグメント数およびデータの保持に使用された一時的なファイルの名前が統計値として表示されます。

8. DUMP フェーズが完了すると、REORG サブコマンドの次のフェーズである LOAD フェーズの開始します。次のように入力します。

   ```
   REBUILD
   ```

9. 次のように入力して REORG サブコマンドを選択します。

   ```
   REORG or 2
   ```

   次のオプションがあります。

   ```
   1. DUMP     (データベース内容の DUMP)
   2. LOAD     (データベースにデータを LOAD)
   ```

10. 次のように入力して、この処理の LOAD フェーズを初期化します。

    ```
    LOAD or 2
    ```

11. DUMP フェーズで作成された一時的ファイルからロードするデータソースの名前を入力します。ほとんどの場合、これは新しいデータソースの名前です。

この時点で、指定したデータを元のマスターファイルからここで指定した新しいデータソースにロードしたことになります。ここで注意することは、元のマスターファイルおよびそのデータソースが保持されていることです。次の 3 つの選択肢があります。

- 混乱を回避するため、元のマスターファイルおよびデータソースの名前を変更する。
- 新規のマスターファイルおよびデータソースを元の名前に変更する。この結果、既存のプロシジャが参照する元の名前は、すべて新規データソースに対して実行されます。
- 新規マスターファイルおよびデータソースが正しく作成されたことを確認した後、元のマスターファイルおよびデータソースを削除する。

既存のデータソースの名前 (元のマスターファイル) を入力した場合、既存のデータソースにデータを追加してこのまま続行するかどうかの確認メッセージが表示されます。

REBUILD 実行を中止するには「N」と入力します。一時的な REBUILD ファイルのレコードを元の FOCUS データソースに追加するには「Y」と入力します。

マスターファイルでフィールド名が重複する場合は、REORG サブコマンドを使用することはできません。

例 Windows での REORG サブコマンドの使用

手順は次のとおりです。

```
1.  COPY EMPLOYEE.FOC EMPOLD.FOC
2.  REBUILD
3.  REORG
4.  DUMP
5.  EMPLOYEE
6.  NO
7.  ERASE EMPLOYEE.FOC
8.  REBUILD
9.  REORG
10. LOAD
11. EMPLOYEE
```

1. データソースのコピーを作成します。
2. REBUILD 機能を初期化します。
3. REORG サブコマンドを指定します。
4. DUMP フェーズを初期化します。
5. ダンプするデータソースの名前を指定します。
6. レコード選択テストが必要でないことを示します。

   データソースがダンプされて統計が生成されます。

7. EMPLOYEE データソースを削除します。
8. REBUILD 機能を初期化します。
9. REORG サブコマンドを指定します。
10. LOAD フェーズを初期化します。
11. ロードするデータソースの名前を指定します。

    データソースがロードされて統計が生成されます。

フィールドインデックスの追加 - INDEX サブコマンド

データソースにデータを入力後、フィールドにインデックスを付けるには INDEX サブコマンドを使用します。マスターファイルで事前に指定されたインデックスや、最後の REBUILD または CREATE コマンド以降に指定したインデックス以外にも、フィールドにインデックスを付けることができます。ただし、ここで指定する各フィールドには、FIELDTYPE=I (または INDEX=I) 属性をマスターファイルに記述しておく必要があります。

INDEX オプションは、オペレーティングシステムのソートプログラムを使用します。書き込みを行うためのディスク領域を確保する必要があります。必要なディスク領域の容量を計算するには、インデックスフィールドの長さに 8 バイトを加算し、その合計をセグメントインスタンス数の 2 倍の数で乗算します。

```
(LENGTH + 8) * 2n
```

説明

n
    セグメントインスタンスの数です。

FIELDTYPE=I 属性を指定してフィールドにインデックスを追加する操作は、データのロードが完了してから行うこともできます。これは、データソースを作成する際にデータのロードとインデックスの指定を同時に実行するより、これらを別々に実行した方が処理が高速化するためです。特に大規模なデータソースを作成する場合にこの差が顕著に現れます。

ソートライブラリおよびワークスペースを使用可能な状態にしておく必要があります。INDEX サブコマンドを発行する前に、DDNAME の SORTIN および SORTOUT を割り当てる必要があります。

手順　INDEX サブコマンドを使用するには

次の手順は、INDEX サブコマンドを使用する方法を示しています。

1. マスターファイルで、インデックスを付けるフィールドに FIELDTYPE=I 属性を追加します。
2. 次のコマンドを入力して REBUILD 機能を初期化します。

   ```
   REBUILD
   ```

   次のオプションがあります。

   ```
   1. REBUILD          (データベース構造最適化)
   2. REORG            (データベース構造の再編成)
   3. INDEX            (インデックスの追加/変更)
   4. EXTERNAL INDEX   (外部インデックスの追加/変更)
   5. CHECK            (データベース構造のチェック)
   6. TIMESTAMP        (タイムスタンプの変更)
   7. DATE NEW         (OLD DATE フォーマットを SMARTDATE フォーマットに変換)
   8. MDINDEX          (多次元インデックスの追加/変更)
   ```

3. 次のように入力して INDEX サブコマンドを選択します。

   ```
   INDEX or 3
   ```

4. FIELDTYPE=I または INDEX=I 属性を追加するマスターファイルの名前を入力します。
5. インデックスを付けるフィールドの名前を入力します。FIELDTYPE=I 属性のすべてのフィールドにインデックスを付ける場合は、アスタリスク (*) を使用します。

INDEX サブコマンドの処理が完了すると、インデックスが付けられたフィールドの名前およびそのフィールド内のインデックス値に関する統計が表示されます。

例　Windows での INDEX サブコマンドの使用

手順は次のとおりです。

```
1. REBUILD
2. INDEX
3. EMPLOYEE
4. EMP_ID
```

1. REBUILD 機能を初期化します。
2. INDEX サブコマンドを指定します。
3. マスターファイルの名前を指定します。
4. インデックスを付けるフィールドの名前を指定します。

フィールドにインデックスが追加されて統計が生成されます。

外部インデックスの作成 - EXTERNAL INDEX サブコマンド

ローカルの FOCUS データソースの読み取り権限を持つユーザは、レコードの結合やレコードの位置を特定する際にインデックス検索を容易にするための外部インデックスデータベースを作成することができます。外部インデックスは、指定した 1 つまたは複数の FOCUS データソースのインデックス、フィールド、セグメントの情報が格納された FOCUS データソースです。外部インデックスは、それに関係付けられた FOCUS データソースから独立して動作します。外部インデックスは、データの取得および分析において永続的なインデックスと同等のパフォーマンスを提供します。

外部インデックスを使用すると、連結した FOCUS データソース、実フィールドおよび一時項目 (DEFINE)、WHERE または IF テストで選択したレコードにインデックスを追加することができます。外部インデックスは、永続的なデータセットとして事前に割り当てられていない限り、一時的なデータセットとして作成されます。インデックスデータが変更されても外部インデックス更新はされません。

外部インデックスを作成するには、REBUILD コマンドを使用します。REBUILD の内部的な処理は、最初にインデックスを構成するデータベースを読み取り、次にインデックス情報を収集し、最後にすべてのフィールド、フォーマット、セグメント、パスの情報が格納されたインデックスデータベースを作成します。

指定する情報には次のものがあります。

- 連結したデータベースからインデックスデータベースに新しいレコードを追加するかどうかの選択。
- 作成する外部インデックスデータベースの名前。
- インデックス情報の取得元になるデータソースの名前。
- インデックスの作成元になるフィールドの名前。
- インデックスフィールドを特定のセグメント内に配置するかどうかの選択。
- 有効な WHERE または IF レコード選択テスト。

ソートライブラリおよびワークスペースを使用可能な状態にしておく必要があります。

手順　EXTERNAL INDEX サブコマンドを使用するには

連結したデータベースから外部インデックスを作成するには次の手順を実行します。

1. ここでは、次の USE が有効になっていることを前提にします。

```
USE CLEAR *
USE
EMPLOYEE
EMP2 AS EMPLOYEE
JOBFILE
EDUCFILE
END
```

EMPLOYEE と EMP2 が連結され、これを EMPLOYEE マスターファイルに記述することができます。

2. 次のコマンドを入力して REBUILD 機能を初期化します。

```
REBUILD
```

次のオプションがあります。

```
1. REBUILD        (データベース構造最適化)
2. REORG          (データベース構造の再編成)
3. INDEX          (インデックスの追加/変更)
4. EXTERNAL INDEX (外部インデックスの追加/変更)
5. CHECK          (データベース構造のチェック)
6. TIMESTAMP      (タイムスタンプの変更)
7. DATE NEW       (OLD DATE フォーマットを SMARTDATE フォーマットに変換)
8. MDINDEX        (多次元インデックスの追加/変更)
```

3. 次のように入力して EXTERNAL INDEX サブコマンドを選択します。

```
EXTERNAL INDEX or 4
```

4. 次のいずれかを入力して、新規のインデックスデータソースを作成するか、既存のインデックスデータソースに追加するかを選択します。

```
NEW
ADD
```

この例では、新規のインデックスデータベースを作成するため、次のように入力します。

```
NEW
```

5. 外部インデックスデータベースの名前を指定します。

```
EMPIDX
```

6. インデックスレコードの取得先のデータソースの名前を指定します。

```
EMPLOYEE
```

7. インデックスを付けるフィールドの名前を指定します。

   ```
   CURR_JOBCODE
   ```

8. 「YES」または「NO」と入力して、インデックスを特定のフィールドに関係付けるかどうかを指定します。この例では次のように入力します。

   ```
   NO
   ```

9. 「YES」または「NO」と入力して、レコード選択テストが必要かどうかを指定します。

   この例では次のように入力します。

   ```
   NO
   ```

   「YES」と入力した場合は、別の行にレコード選択テストを入力し、END コマンドで終了します。

   以下はその例です。

   ```
   IF DEPARTMENT EQ 'MIS'
   END
   ```

EXTERNAL INDEX サブコマンドの処理が完了すると、インデックスデータソースに関する統計 (? FDT クエリの出力) が表示されます。このクエリは、インデックスデータベースの内容を確認するために、EXTERNAL INDEX サブコマンドの処理の最後に自動的に発行されます。

参照 REBUILD EXTERNAL INDEX の特別な考慮事項

外部インデックスを使用する際の考慮事項は次のとおりです。

- USE リストでは、WITH ステートメントを使用して最大で 8 つのインデックスを有効にすることができます。USE CLEAR コマンドに続けて新しい USE ステートメントを発行した場合は、セッション内で 9 つ以上のインデックスを有効にすることができます。
- 連結したファイルでは、最大で 256 個のファイルにインデックスを付けることができます。ただし、同時に有効にできるインデックス数は 8 です。
- コンポーネントファイルは、ファイル作成日時のタイムスタンプで確認されます。コピーしたファイルの日時のタイプスタンプが他のファイルと重複する場合は、次のメッセージが表示されます。

  ```
  (FOC995) 外部インデックスエラー：コンポーネントが重複しています。
  ```

- フィールド名が 12 バイトを超えるフィールドにインデックスを作成することはできません。

- テキストフィールドをインデックスフィールドとして使用することはできません。
- USE オプションの NEW、READ、ON、LOCAL、AS *master* ON *userid* READ は外部インデックスデータベースではサポートされません。
- CREATE FILE が一時的な割り当てを自動的に実行するため、外部インデックスデータベースを割り当てる必要はありません。永続的なデータベースが必要な場合は、REBUILD の EXTERNAL INDEX サブコマンドを発行する前にインデックスデータベースの割り当てを実行しておく必要があります。
- REBUILD の EXTERNAL INDEX サブコマンド処理で作成される SORTIN および SORTOUT 作業ファイルは、領域を十分に確保して割り当てる必要があります。次の計算式を使用して、必要な領域を予測することができます。

  ```
  bytes = (field_length + 20) * number_of_occurrences
  ```

インデックスデータベースの連結

外部インデックス機能を使用して、連結した FOCUS データソースからインデックス検索を行うことができます。インデックスを含むデータベースを連結する場合は、REBUILD を発行する前に USE コマンドを発行する必要があります。この USE コマンドには、すべてのクロスリファレンスファイルおよび LOCATION ファイルを含める必要があります。EXTERNAL INDEX サブコマンドには、連結したデータベースの新規インデックスレコードのみをインデックスデータベースに追加する ADD 関数が含まれているため、インデックスデータベースを再作成する必要はありません。

インデックスレコードを追加する際に、インデックスが作成された元のデータソースが USE リストに含まれていない場合があります。その場合は、EXTERNAL INDEX サブコマンドが次のメッセージを表示します。

```
(FOC999) 警告。外部インデックスコンポーネントが再使用されました：ddname
```

インデックスフィールドの配置

外部インデックス機能は、検索パフォーマンスを向上させるために、特定のセグメント内で一時項目 (DEFINE) のインデックス値の検索位置を指定する際に役立ちます。インデックスフィールドはそのソースセグメントの外部にあるデータに関係付けられているため、階層内の下位のセグメントに配置すると、インデックスフィールドで取得されるデータが影響を受けます。これにより、ソースセグメントとターゲットセグメントの関係を構築することが可能になります。ソースセグメントは、インデックスフィールドが配置されたセグメントとして定義されます。ターゲットセグメントは、同一パス内でソースセグメントの上位または下位に存在するセグメントとして定義されます。

ターゲットセグメントが同一パスに存在しない場合、次のメッセージが表示されます。

```
(FOC974) 外部インデックスエラー：該当セグメントに誤りがあります。
```

一時項目 (DEFINE) を上位のセグメントに配置することはできません。

ソースセグメントをクロスリファレンスセグメントまたは LOCATION セグメントにすることは可能ですが、ターゲットセグメントをクロスリファレンスセグメントにすることはできません。クロスリファレンスセグメントにターゲットセグメントを配置しようとすると、次のメッセージが表示されます。

```
(FOC1000) クロスリファレンスフィールドの使用に誤りがあります。
```

インデックスを特定のフィールドに関連付けない場合、ソースセグメントとターゲットセグメントは同一のセグメントになります。

外部インデックスの有効化

外部インデックスデータベースを作成した後、そのデータベースを作成元のデータソースに関係付ける必要があります。この関係付けには USE コマンドを使用します。外部インデックスデータベースを作成する前に発行した USE 構文と同一の構文を使用します。ただし、WITH または INDEX オプションは必須です。

構文　外部インデックスの有効化

```
USE  [ADD|REPLACE]
database_name [AS masternamе]
index_database_name  [WITH|INDEX]  masternamе   .
   .
   .
END
```

説明

`ADD`
1 つまたは複数の新規データベースを既存の USE リストに追加します。ADD オプションを使用しない場合は、既存の USE リストがクリアされて、現在の USE データベースのリストに置換されます。

`REPLACE`
USE リスト内の既存の *database_name* を置換します。

`database_name`
データソースの名前です。

UNIX、Windows では *filename* を入力します。 再構築するデータソースを参照するには、USE コマンドを使用します。USE コマンドが動作しない場合は、EDAPATH 変数を使用してデータソースを検索することができます。

マスターファイルで指定されたすべてのクロスリファレンスファイルおよび LOCATION ファイルのデータソース名を USE リストに追加する必要があります。

`AS`
この句をマスターファイル名に付けてデータソースを連結します。

`mastername`
マスターファイルの名前を指定します。

`index_database_name`
外部インデックスデータベースの名前です。

UNIX、Windows では [*pathname*]*filename.foc* を入力します。再構築するデータソースを参照するには、USE コマンドを使用します。USE コマンドが動作しない場合は、EDAPATH 変数を使用してデータソースを検索することができます。

`WITH|INDEX`
コンポーネントデータソースとインデックスデータベースの関係を作成するキーワードです。INDEX は WITH の同義語です。

データソース整合性の確認 - CHECK サブコマンド

FOCUS データソースが構造的に破損して整合性が失われることはほとんどありません。ただし、ドライブの不良や不適切なマスターファイルを使用した場合は構造的に破損する場合があります。その場合は、REBUILD の CHECK サブコマンドを使用して次の 2 つの重要なタスクを実行します。

- データソース内のポインタを確認する。
- エラーが発生した場合は、メッセージを表示し、障害のあるセグメントまたはインスタンスからの分岐を試みる。

CHECK サブコマンドは、損失対象のデータを報告することはできますが、データ損失を回避するために FOCUS データソースのバックアップを頻繁に行うことが大切です。

CHECK サブコマンドで構造的な破損を発見できない場合があります。CHECK サブコマンドでデータ損失が報告されていなくてもデータソースが破損していると考えられる場合は、別の方法でデータソースの整合性を確認することができます。この方法では、? FILE および TABLEF コマンドを使用します。この方法は REBUILD 機能ではありませんが、CHECK サブコマンドに関連するため、このセクションの最後で説明しています。

手順　CHECK サブコマンドを使用するには

次の手順は、CHECK サブコマンドを使用する方法を示しています。

1. 次のコマンドを入力して REBUILD 機能を初期化します。

   ```
   REBUILD
   ```

   次のオプションがあります。

   ```
   1. REBUILD        (データベース構造最適化)
   2. REORG          (データベース構造の再編成)
   3. INDEX          (インデックスの追加/変更)
   4. EXTERNAL INDEX (外部インデックスの追加/変更)
   5. CHECK          (データベース構造のチェック)
   6. TIMESTAMP      (タイムスタンプの変更)
   7. DATE NEW       (OLD DATE フォーマットを SMARTDATE フォーマットに変換)
   8. MDINDEX        (多次元インデックスの追加/変更)
   ```

2. 次のように入力して CHECK サブコマンドを選択します。

   ```
   CHECK or 5
   ```

3. 確認するデータソースの名前を入力します。

   UNIX、Windows では *filename* を入力します。 再構築するデータソースを参照するには、USE コマンドを使用します。USE コマンドが動作しない場合は、EDAPATH 変数を使用してデータソースを検索することができます。

CHECK サブコマンドの処理中に統計が表示されます。

- エラーが見つからない場合は、取得されたセグメント数が統計に表示されます。
- エラーが見つかった場合は、各エラーのタイプおよび発生場所が統計に表示されます。

  DELETE は、データが削除され、取得されなかったことを示します。

  OFFPAGE は、このセグメントが所有するページにデータのアドレスが存在しないことを示します。

  INVALID は、リンクタイプを識別できないことを示します。データソースが部分的に破損している可能性があります。

例　Windows での CHECK サブコマンドの使用 (未破損ファイル)

手順は次のとおりです。

```
1. REBUILD
2. CHECK
3. EMPLOYEE
```

1. REBUILD 機能を初期化します。
2. CHECK サブコマンドを指定します。
3. 確認するデータソースの名前を入力します。

   データソースが確認されて統計が生成されます。

? FILE と TABLEF による構造的な整合性の確認

REBUILD の CHECK サブコマンドでデータ破損が報告されていなくてもデータソースが破損していると考えられる場合は、? FILE および TABLEF コマンドを呼び出して、報告されたセグメントインスタンス数を比較します。インスタンス数が一致しない場合、構造的な問題があります。

手順　? FILE と TABLEF を使用して REBUILD CHECK を確認するには

1. 次のコマンドを発行します。

   ```
   ? FILE filename
   ```

   説明

   `filename`
   確認する FOCUS データソースの名前です。

   データソースのステータスに関する情報がレポートに表示されます。各セグメントのインスタンス数が [ACTIVE COUNT] 列に表示されます。

2. 次のように TABLEF コマンドを発行して、短いパスに存在するセグメントも含めたすべてのインスタンス数を集計します。

   ```
   SET ALL = ON
   ```

3. 次のように入力します。

```
TABLEF FILE filenameCOUNT field1 field2END
```

説明

`filename`
FOCUS データソースのマスターファイル名です。

`field1...`
データソース内のフィールドの名前です。各セグメントからフィールドを 1 つ指定します。セグメント内の任意のフィールドを選択します。

レポートが生成され、選択したフィールドの出現回数および各セグメントのセグメントインスタンス数が表示されます。これらの数字が、? FILE コマンドで報告されたセグメントインスタンス数と一致している必要があります。ただし、TABLEF コマンドで報告された親セグメントに存在するインスタンスのユニークセグメントは除外します。この数字が一致しない場合、または ? FILE コマンドまたは TABLEF コマンドが異常終了した場合は、データソースが破損している可能性があります。

例 EMPLOYEE データソース整合性の確認

ここでは、ユーザの入力を太字で、コンピュータの応答を大文字で表しています。

```
? FILE
STATUS OF FOCUS FILE: c:employee.foc   ON 01/31/2003 AT 16.17.32
                ACTIVE  DELETED    DATE OF    TIME OF    LAST TRANS
SEGNAME         COUNT   COUNT      LAST CHG   LAST CHG   NUMBER

EMPINFO           12              05/13/1999   16.17.22      448
FUNDTRAN           6              05/13/1999   16.17.22       12
PAYINFO           19              05/13/1999   16.17.22       19
ADDRESS           21              05/13/1999   16.17.22       21
SALINFO           70              05/13/1999   16.17.22      448
DEDUCT           448              05/13/1999   16.17.22      448
TOTAL SEGS       576
TOTAL CHAR      8984
TOTAL PAGES        8
LAST CHANGE                       05/13/1999   16.17.22      448
SET ALL = ON
TABLEF FILE EMPLOYEE
COUNT EMP_ID BANK_NAME DAT_INC TYPE PAY_DATE DED_CODE
END

PAGE     1

 EMP_ID  BANK_NAME  DAT_INC  TYPE   PAY_DATE  DED_CODE
 COUNT   COUNT      COUNT    COUNT  COUNT     COUNT
 ------  ---------  -------  -----  --------  --------
     12         12       19     21        70       448
NUMBER OF RECORDS IN TABLE=      488 LINES= 1
```

このレポートでは、TABLEF レポートの BANK_NAME の個数と ? FILE クエリによるレポートの FUNDTRAN インスタンス数が一致していません。これは、FUNDTRAN がユニークセグメントで、その親の拡張セグメントとして解釈されるためです。

データソース作成日時の変更 - TIMESTAMP サブコマンド

FOCUS データソースのタイムスタンプは、CREATE、REBUILD によりデータソースが変更されるたびに更新されます。REBUILD の TIMESTAMP サブコマンドを使用すると、データソースを変更せずにデータソースのタイムスタンプを更新することができます。

手順 TIMESTAMP サブコマンドを使用するには

次の手順は、TIMESTAMP サブコマンドを使用する方法を示しています。

1. 次のコマンドを入力して REBUILD 機能を初期化します。

   ```
   REBUILD
   ```

   次のオプションがあります。

   ```
   1. REBUILD        (データベース構造最適化)
   2. REORG          (データベース構造の再編成)
   3. INDEX          (インデックスの追加/変更)
   4. EXTERNAL INDEX (外部インデックスの追加/変更)
   5. CHECK          (データベース構造のチェック)
   6. TIMESTAMP      (タイムスタンプの変更)
   7. DATE NEW       (OLD DATE フォーマットを SMARTDATE フォーマットに変換)
   8. MDINDEX        (多次元インデックスの追加/変更)
   ```

2. 次のように入力して TIMESTAMP サブコマンドを選択します。

   ```
   TIMESTAMP or 6
   ```

3. タイムスタンプを更新するデータソースの名前です。

   UNIX、Windows では *filename* を入力します。 再構築するデータソースを参照するには、USE コマンドを使用します。USE コマンドが動作しない場合は、EDAPATH 変数を使用してデータソースを検索することができます。

4. 次のオプションのいずれかを入力して、日付時間の基準日時を指定します。

   T - 今日の日付。データソースのタイムスタンプを現在の日時で更新します。

   D - ファイルを検索した日付。データソースのタイムスタンプをデータソースが最後に変更された日時で更新します。データソースの各ページが検索され、ページに記録された最新の日時がデータソースに適用されます。この方法は、? FILE クエリを発行する場合と同様に、大規模なデータソースではかなりの時間を要します。このオプションは、外部インデックスデータベースをそのコンポーネントデータソースに同期させる際に使用します。

   MMDDYY HHMMSS - ユーザが指定する日時です。REBUILD はこの値を使用してデータソースのタイムスタンプを更新します。入力する日付時間には、mmddyy hhmmss、mmddyyyy hhmmss のいずれかのフォーマットを使用する必要があります。また、日付と時間の間にブランクを入れる必要があります。2 桁の年の値を使用した場合、REBUILD はその値を DEFCENT および YRTHRESH に使用して西暦年を特定します。

   無効な日付または時間を入力した場合は、次のメッセージが表示されます。

   (FOC961) REBUILD で指定した日付に誤りがあります。

レガシー日付の変換 - DATE NEW サブコマンド

REBUILD の DATE NEW サブコマンドを使用して、FOCUS データソースに存在するレガシー日付 (日付表示オプションを使用した文字、整数、パック 10 進数のフィールド) を SmartDate 日付 (日付フォーマットのフィールド) に変換することができます。

このユーティリティは、「Update-In-Place」というテクノロジを使用します。このテクノロジは、データソースを更新して新しいマスターファイルを作成しますが、データソースの構造およびサイズは変更しません。REBUILD の DATE NEW サブコマンドを実行する前に、データソースのバックアップを作成しておく必要があります。このユーティリティを使用する場合は、コピーしたデータソースに対してユーティリティを実行し、実行後に元のファイルを更新後のバックアップファイルで上書きするという方法をお勧めします。

DATE NEW によるレガシー日付の変換

REBUILD の DATE NEW サブコマンドを使用すると、元のレガシー日付フィールド (日付表示オプションを使用した文字、整数、パック 10 進数のフィールド) が SmartDate 日付 (日付フォーマットのフィールド) で上書きされます。レガシー日付の格納サイズが SmartDate 日付の格納サイズである 4 バイトより大きい場合は、データソースの日付フィールドの後に未使用フィールドが追加されます。

- A6YMD、A6MDY、A6DMY フォーマットはそれぞれ YMD、MDY、DMY フォーマットに変換され、マスターファイルに 2 バイトの未使用フィールドが追加されます。
- 整数日付 (例、I6YMD、I6MDY) の格納サイズは 4 バイトのため、未使用フィールドは追加されません。
- すべてのパック 10 進数フィールドおよび A8 日付には 4 バイトの未使用フィールドが追加されます。

日付がキーフィールド (セグメントの最後のキーではない) で未使用フィールドを必要とする場合は、未使用フィールドが必要なそれぞれの日付フィールドで SEGTYPE のキーの番号が 1 つだけ大きくなります。

DATE NEW サブコマンドは、レガシー日付を SmartDate 日付に変換する場合にのみ使用します。マスターファイルのフィールドフォーマットは、次のいずれかのフォーマットである必要があります。なお、月変換編集オプションの T および TR をフォーマットに含めることができます。

```
A8YYMD A8MDYY A8DMYY A6YMD A6MDY A6DMY A6YYM A6MYY A4YM A4MY

I8YYMD I8MDYY I8DMYY I6YMD I6MDY I6DMY I6YYM I6MYY I4YM I4MY

P8YYMD P8MDYY P8DMYY P6YMD P6MDY P6DMY P6YYM P6MYY P4YM P4MY
```

日付値を格納するフィールドが上記のどのフォーマットにも該当しない場合、DATE NEW サブコマンドは変換を実行しません。変換を実行したくないフィールドのフォーマットが上記のいずれかのフォーマットに該当する場合は、そのフォーマットから日付編集オプションを一時的に削除し、DATE NEW サブコマンドを実行した後、フォーマットに編集オプションを戻します。

参照 DATE NEW サブコマンド使用時の注意

- REBUILD を発行する前に、データソースの DBA パスワードを発行しておく必要があります。
- 元のマスターファイルが暗号化されている場合は使用することはできません。

- REBUILD を実行する際に、LOCATION ファイルを含むすべてのファイルをローカルで使用可能な状態にしておく必要があります。
- マスターファイルに GROUP フィールドが存在する場合は使用することはできません。
- すべてのエラー番号は &&FOCREBUILD で、部分的なエラー番号は &FOCERRNUM で取得することができます。データソースを構築するためのプロシジャを作成する場合は、&&FOCREBUILD と &FOCERRNUM の両方のエラーテストを実行します。
- 問題の発生を事前に回避するため、REBUILD を発行する前にすべての LET および JOIN をクリアします。
- DEFCENT および YRTHRESH は、グローバルレベル、データソースレベル、フィールドレベルで評価されます。
- DATE NEW サブコマンドを実行する前に、データソース内の無効な日付をすべて修正します。このユーティリティは、無効な日付をすべて 0 (ゼロ) に変換します。無効な日付がキーとして使用された場合、結果としてデータソース内のキーが重複する場合があります。
- 一時的な REBUILD ファイルを格納するための十分なワークスペースを確保する必要があります。一般に、既存のファイルサイズより 10 パーセントから 20 パーセント大きいサイズを使用します。
- インデックスが存在する場合は、INDEX サブコマンドが自動的に実行されます。
- キーが日付の場合は、REBUILD サブコマンドが DATE NEW サブコマンドの後に自動的に実行されます。
- INDEX サブコマンドの場合と同様に、ソートライブラリおよびワークスペースを使用可能な状態にしておく必要があります。

DATE NEW で変換されない項目

REBUILD の DATE NEW サブコマンドは、FOUCS データソースおよび日付フィールドのみを修復するツールです。次の項目は修復の対象外です。

- マスターファイルの DEFINE 属性
- マスターファイルの ACCEPT 属性
- マスターファイルまたは主セキュリティリポジトリ (DBAFILE) で設定した DBA 制限 (例、VALUE 制限)
- このマスターファイルまたは他のマスターファイルで指定した他の日付フィールドへのクロスリファレンス

- プロシジャで指定した日付フィールドへの参照

例 Windows での DATE NEW サブコマンドの使用

手順は次のとおりです。

```
1. SET DFC = 19, YRT = 50
2. REBUILD
3. DATE NEW
4. NEWEMP.MAS
5. YES
```

1. DEFCENT および YRTHRESH パラメータを設定して、各日付に使用する世紀を指定します。
2. REBUILD 機能を初期化します。
3. DATE NEW サブコマンドを指定します。
4. 日付を変換するマスターファイルの名前を入力します。
5. データソースがバックアップされたことを示します。

日付が変換されて統計が生成されます。この統計には、変更されたセグメント数が表示されます。

新しいマスターファイルは、元のマスターファイルのコピーを更新したファイルです。次の点について考慮する必要があります。

- レガシー日付フィールドの USAGE フォーマットが更新され、フォーマットおよび長さが削除されます。日付編集オプションは保持されます。たとえば、A6YMDTR は YMDTR になります。
- 必要に応じて、日付に未使用フィールドが追加されます。

  ```
  FIELDNAME= ,ALIAS= ,FORMAT=An,$ PAD FIELD ADDED BY REBUILD
  ```

  説明

  n

  未使用フィールドの長さです (2 または 4 バイト)。FIELDNAME および ALIAS はブランクになります。
- 日付に未使用フィールドが必要だが、その日付がキー内の最終フィールドでない場合は、キーとして修復された日付を持つセグメントの SEGTYPE 属性が更新されます。SEGTYPE の番号が、キーに追加された未使用フィールド数だけ大きくなります。

- セグメントに SEGTYPE が存在しない場合は、そのセグメントの終了文字 ($) の直前に次の行が追加されます。

  ```
  SEGTYPE=segtype,$ OMITTED SEGTYPE ADDED BY REBUILD
  ```

  説明

  `segtype`
  REBUILD により決定されます。

- フィールドに USAGE 属性が存在しない場合は (日付フィールドも含む)、そのフィールドの終了文字 ($) の直前に次の行が追加されます。

  ```
  USAGE=fmt,$ OMITTED USAGE ADDED BY REBUILD
  ```

  説明

  `fmt`
  マスターファイルで以前に指定したフィールドのフォーマットです。USAGE= ステートメントを明示的に指定しない場合は、REBUILD が以前のフィールドフォーマットをここで記述したすべてのフィールドに自動的に割り当てます。

DATE NEW で作成した新しいマスターファイルの使用

REBUILD の DATE NEW サブコマンドは、データソースへの変更が反映された更新済みのマスターファイルを作成します。データソースを再構築した段階で、元のマスターファイルはそのデータソースに対して使用できなくなります。DATE NEW サブコマンドで作成した新しいマスターファイルを使用する必要があります。

例　サンプルマスターファイル - DATE NEW による変換前と変換後の比較

変換前	変換後
`FILE=filename`	`FILE=filename`
`SEGNAME=segname, SEGTYPE=S2`	`SEGNAME=segname, SEGTYPE=S3`
`FIELD=KEY1,,USAGE=A6YMD,$`	`FIELD=KEY1,,USAGE= YMD,$` `FIELD=, ,USAGE=A2,$ PAD FIELD ADDED BY REBUILD`
`FIELD=KEY2,,USAGE=I6MDY,$`	`FIELD=KEY2,,USAGE= MDY,$`

変換前	変換後
FIELD=FIELD3,,USAGE=A8YYMD,$	FIELD=FIELD3,,USAGE= YYMD,$ FIELD=, ,USAGE=A4,$ PAD FIELD ADDED BY REBUILD

REBUILD の DATE NEW サブコマンドを使用すると、サンプルマスターファイルは次のように変換されます。

- SEGTYPE が S2 から S3 に変更され、2 バイトの未使用フィールドが追加されます。
- A6YMD フォーマットが SmartDate 日付の YMD フォーマットに変更されます。
- マスターファイルのブランクのフィールド名およびエイリアスに 2 バイトの未使用フィールドが追加されます。
- I6MDY フォーマットが SmartDate 日付の MDY フォーマットに変更されます (未使用フィールドは必要ありません)。
- A8YYMD フォーマットが SmartDate 日付の YYMD フォーマットに変更されます。
- マスターファイルのブランクのフィールド名およびエイリアスに 4 バイトの未使用フィールドが追加されます。

DATE NEW 処理での日付フィールドへの実行アクション

日付フィールドがキーフィールドまたはインデックスフィールドの場合、DATE NEW サブコマンドを実行すると、REBUILD または INDEX サブコマンドが自動的に実行されます。下表は、DATE NEW サブコマンドを処理中に日付フィールドに実行されるアクションについて示しています。

日付がキー	インデックス	結果
いいえ	なし	変更されたセグメント数 = *n*
いいえ	はい	日付フィールドで INDEX サブコマンドが実行される。
はい	なし	REBUILD サブコマンドが実行される。

日付がキー	インデックス	結果
はい	任意のフィールド	REBUILD サブコマンドが実行される。 インデックスフィールドに対して INDEX サブコマンドが実行される。

マルチディメンションインデックスの作成 - MDINDEX サブコマンド

MDINDEX サブコマンドは、マルチディメンションインデックス (MDI) の作成および保守に使用します。

A

マスターファイルと構造図

この付録では、マニュアルの事例で使用するサンプルデータソースの記述およびその構造図について説明します。

EMPLOYEE データソース

EMPLOYEE データソースには、企業の従業員に関するサンプルデータが保存されています。次のセグメントが記述されています。

EMPINFO

従業員の ID 番号、氏名、役職が格納されています。

FUNDTRAN

従業員の自動振込み口座情報が格納されています。これは、ユニークセグメントです。

PAYINFO

従業員の給与履歴が格納されています。

ADDRESS

従業員の自宅の住所が格納されています。

SALINFO

従業員の毎月の給与額が格納されています。

DEDUCT

従業員の毎月の給与控除額が格納されています。

EMPLOYEE データソースには、JOBFILE ファイルおよび EDUCFILE ファイルに属するクロスリファレンスセグメントも含まれています。この付録では、これらのファイルについても説明します。ここでは、次のセグメントが記述されています。

JOBSEG (JOBFILE から)

各従業員に適用する役職名を記述します。

SKILLSEG (JOBFILE から)

各役職に要求されるスキルを列記します。

SECSEG (JOBFILE から)

各役職に要求される機密区分を指定します。

ATTNDSEG (EDUCFILE から)

従業員が社内コースを受講した日付を列記します。

COURSEG (EDUCFILE から)

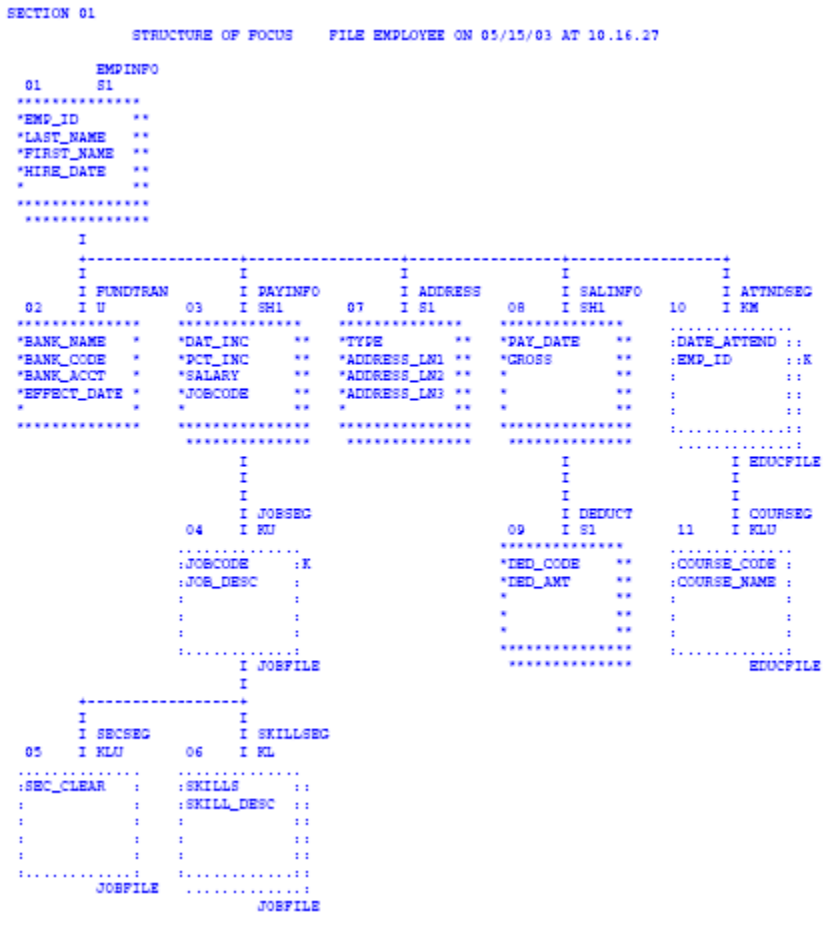
従業員が受講したコース名を列記します。

EMPLOYEE マスターファイル

```
FILENAME=EMPLOYEE, SUFFIX=FOC
 SEGNAME=EMPINFO,  SEGTYPE=S1
  FIELDNAME=EMP_ID,         ALIAS=EID,     FORMAT=A9,           $
  FIELDNAME=LAST_NAME,      ALIAS=LN,      FORMAT=A15,          $
  FIELDNAME=FIRST_NAME,     ALIAS=FN,      FORMAT=A10,          $
  FIELDNAME=HIRE_DATE,      ALIAS=HDT,     FORMAT=I6YMD,        $
  FIELDNAME=DEPARTMENT,     ALIAS=DPT,     FORMAT=A10,          $
  FIELDNAME=CURR_SAL,       ALIAS=CSAL,    FORMAT=D12.2M,       $
  FIELDNAME=CURR_JOBCODE,   ALIAS=CJC,     FORMAT=A3,           $
  FIELDNAME=ED_HRS,         ALIAS=OJT,     FORMAT=F6.2,         $
 SEGNAME=FUNDTRAN,  SEGTYPE=U,   PARENT=EMPINFO
  FIELDNAME=BANK_NAME,      ALIAS=BN,      FORMAT=A20,          $
  FIELDNAME=BANK_CODE,      ALIAS=BC,      FORMAT=I6S,          $
  FIELDNAME=BANK_ACCT,      ALIAS=BA,      FORMAT=I9S,          $
  FIELDNAME=EFFECT_DATE,    ALIAS=EDATE,   FORMAT=I6YMD,        $
 SEGNAME=PAYINFO,   SEGTYPE=SH1, PARENT=EMPINFO
  FIELDNAME=DAT_INC,        ALIAS=DI,      FORMAT=I6YMD,        $
  FIELDNAME=PCT_INC,        ALIAS=PI,      FORMAT=F6.2,         $
  FIELDNAME=SALARY,         ALIAS=SAL,     FORMAT=D12.2M,       $
  FIELDNAME=JOBCODE,        ALIAS=JBC,     FORMAT=A3,           $
 SEGNAME=ADDRESS,   SEGTYPE=S1,  PARENT=EMPINFO
  FIELDNAME=TYPE,           ALIAS=AT,      FORMAT=A4,           $
  FIELDNAME=ADDRESS_LN1,    ALIAS=LN1,     FORMAT=A20,          $
  FIELDNAME=ADDRESS_LN2,    ALIAS=LN2,     FORMAT=A20,          $
  FIELDNAME=ADDRESS_LN3,    ALIAS=LN3,     FORMAT=A20,          $
  FIELDNAME=ACCTNUMBER,     ALIAS=ANO,     FORMAT=I9L,          $
 SEGNAME=SALINFO,   SEGTYPE=SH1, PARENT=EMPINFO
  FIELDNAME=PAY_DATE,       ALIAS=PD,      FORMAT=I6YMD,        $
  FIELDNAME=GROSS,          ALIAS=MO_PAY,  FORMAT=D12.2M,       $
 SEGNAME=DEDUCT,    SEGTYPE=S1,  PARENT=SALINFO
  FIELDNAME=DED_CODE,       ALIAS=DC,      FORMAT=A4,           $
  FIELDNAME=DED_AMT,        ALIAS=DA,      FORMAT=D12.2M,       $
 SEGNAME=JOBSEG,    SEGTYPE=KU,  PARENT=PAYINFO, CRFILE=JOBFILE,
  CRKEY=JOBCODE,$
 SEGNAME=SECSEG,    SEGTYPE=KLU, PARENT=JOBSEG,   CRFILE=JOBFILE, $
 SEGNAME=SKILLSEG,  SEGTYPE=KL,  PARENT=JOBSEG,   CRFILE=JOBFILE, $
 SEGNAME=ATTNDSEG,  SEGTYPE=KM,  PARENT=EMPINFO,  CRFILE=EDUCFILE,
  CRKEY=EMP_ID,$
 SEGNAME=COURSEG,   SEGTYPE=KLU, PARENT=ATTNDSEG, CRFILE=EDUCFILE,$
```

EMPLOYEE 構造図

EMPLOYEE 構造図は次のとおりです。



JOBFILE データソース

JOBFILE データソースには、企業の役職に関するサンプルデータが保存されています。次のセグメントが記述されています。

JOBSEG

各従業員に適用する役職名を記述します。このセグメントの JOBCODE フィールドはインデックスフィールドです。

SKILLSEG

各役職に要求されるスキルを列記します。

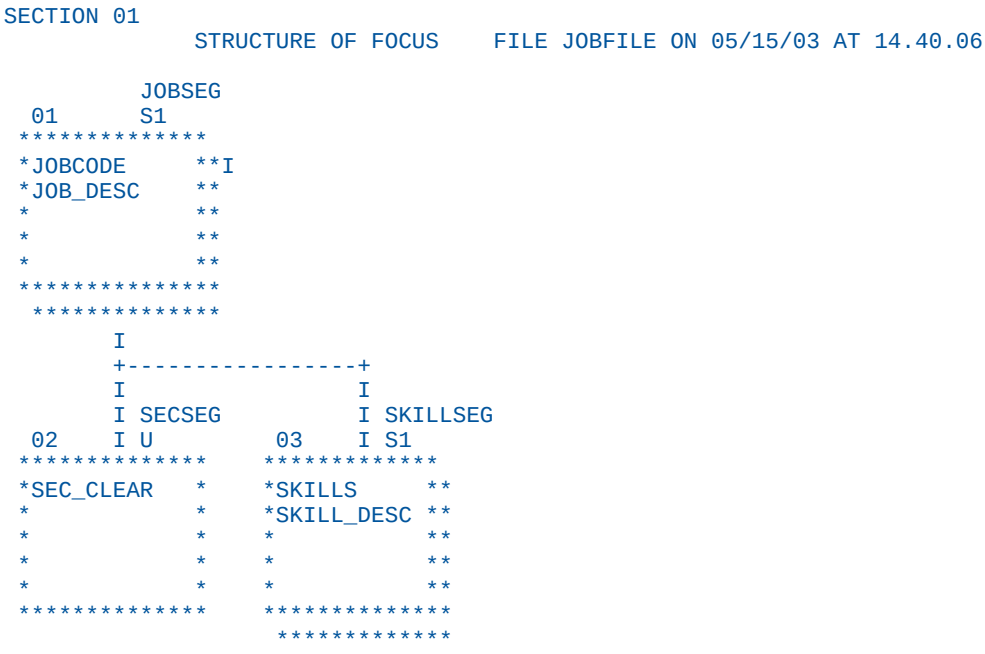
SECSEG

各役職に要求される機密区分を指定します (必要な場合)。これは、ユニークセグメントです。

JOBFILE マスターファイル

```
FILENAME=JOBFILE,  SUFFIX=FOC
 SEGNAME=JOBSEG,   SEGTYPE=S1
  FIELDNAME=JOBCODE,    ALIAS=JC, FORMAT=A3,    INDEX=I,$
  FIELDNAME=JOB_DESC,   ALIAS=JD, FORMAT=A25           ,$
 SEGNAME=SKILLSEG, SEGTYPE=S1,  PARENT=JOBSEG
  FIELDNAME=SKILLS,     ALIAS=,   FORMAT=A4            ,$
  FIELDNAME=SKILL_DESC, ALIAS=SD, FORMAT=A30           ,$
 SEGNAME=SECSEG,   SEGTYPE=U,   PARENT=JOBSEG
  FIELDNAME=SEC_CLEAR,  ALIAS=SC, FORMAT=A6            ,$
```

JOBFILE 構造図



EDUCFILE データソース

EDUCFILE データソースには、企業の社内コースに関するサンプルデータが保存されています。次のセグメントが記述されています。

COURSEG

各コース名が格納されています。

ATTNDSEG

各コースを受講した従業員を特定します。このセグメントの両方のフィールドがキーフィールドです。このセグメントの EMP_ID フィールドはインデックスフィールドです。

EDUCFILE マスターファイル

```
FILENAME=EDUCFILE, SUFFIX=FOC
 SEGNAME=COURSEG,  SEGTYPE=S1
  FIELDNAME=COURSE_CODE,  ALIAS=CC,   FORMAT=A6,          $
  FIELDNAME=COURSE_NAME,  ALIAS=CD,   FORMAT=A30,         $
 SEGNAME=ATTNDSEG,  SEGTYPE=SH2,  PARENT=COURSEG
  FIELDNAME=DATE_ATTEND,  ALIAS=DA,   FORMAT=I6YMD,       $
  FIELDNAME=EMP_ID,       ALIAS=EID,  FORMAT=A9, INDEX=I, $
```

EDUCFILE 構造図

```
SECTION 01
              STRUCTURE OF FOCUS    FILE EDUCFILE ON 05/15/03 AT 14.45.44

          COURSEG
  01      S1
 **************
 *COURSE_CODE **
 *COURSE_NAME **
 *            **
 *            **
 *            **
 ***************
  **************
        I
        I
        I
        I ATTNDSEG
  02    I SH2
 **************
 *DATE_ATTEND **
 *EMP_ID      **I
 *            **
 *            **
 *            **
 ***************
  **************
```

SALES データソース

SALES データソースには、提携したチェーン店を持つ酪農企業に関するサンプルデータが保存されています。次のセグメントが記述されています。

`STOR_SEG`

製品を購入する店舗を列記します。

DATE_SEG
在庫日データが格納されています。

PRODUCT
各製品の日付別の売上データが格納されています。PROD_CODE フィールドはインデックスフィールドです。RETURNS フィールドと DAMAGED フィールドには、MISSING=ON 属性が指定されています。

SALES マスターファイル

```
FILENAME=KSALES,    SUFFIX=FOC
 SEGNAME=STOR_SEG, SEGTYPE=S1
  FIELDNAME=STORE_CODE,   ALIAS=SNO,    FORMAT=A3,    $
  FIELDNAME=CITY,         ALIAS=CTY,    FORMAT=A15,   $
  FIELDNAME=AREA,         ALIAS=LOC,    FORMAT=A1,    $
 SEGNAME=DATE_SEG, PARENT=STOR_SEG, SEGTYPE=SH1,
  FIELDNAME=DATE,         ALIAS=DTE,    FORMAT=A4MD, $
 SEGNAME=PRODUCT,  PARENT=DATE_SEG, SEGTYPE=S1,
  FIELDNAME=PROD_CODE,    ALIAS=PCODE, FORMAT=A3,    FIELDTYPE=I,$
  FIELDNAME=UNIT_SOLD,    ALIAS=SOLD,  FORMAT=I5,    $
  FIELDNAME=RETAIL_PRICE,ALIAS=RP,     FORMAT=D5.2M,$
  FIELDNAME=DELIVER_AMT, ALIAS=SHIP,   FORMAT=I5,    $
  FIELDNAME=OPENING_AMT, ALIAS=INV,    FORMAT=I5,    $
  FIELDNAME=RETURNS,      ALIAS=RTN,    FORMAT=I3,    MISSING=ON,$
  FIELDNAME=DAMAGED,      ALIAS=BAD,    FORMAT=I3,    MISSING=ON,$
```

SALES 構造図

```
SECTION 01
            STRUCTURE OF FOCUS    FILE SALES ON 05/15/03 AT 14.50.28

          STOR_SEG
 01       S1
**************
*STORE_CODE  **
*CITY        **
*AREA        **
*            **
*            **
***************
 **************
       I
       I
       I
       I DATE_SEG
 02    I SH1
**************
*DATE        **
*            **
*            **
*            **
*            **
***************
 **************
       I
       I
       I
       I PRODUCT
 03    I S1
**************
*PROD_CODE   **I
*UNIT_SOLD   **
*RETAIL_PRICE**
*DELIVER_AMT **
*            **
***************
 **************
```

CAR データソース

CAR データソースには、車の仕様および売上情報に関するサンプルデータが保存されています。次のセグメントが記述されています。

ORIGIN

車の製造国を列記します。COUNTRY フィールドはインデックスフィールドです。

COMP

車の名前が格納されています。

CARREC

車の型式が格納されています。

BODY

車体の種類、座席、ディーラー、小売価格、売上台数を列記します。

SPECS

車の仕様を列記します。これは、ユニークセグメントです。

WARANT

保証のタイプを列記します。

EQUIP

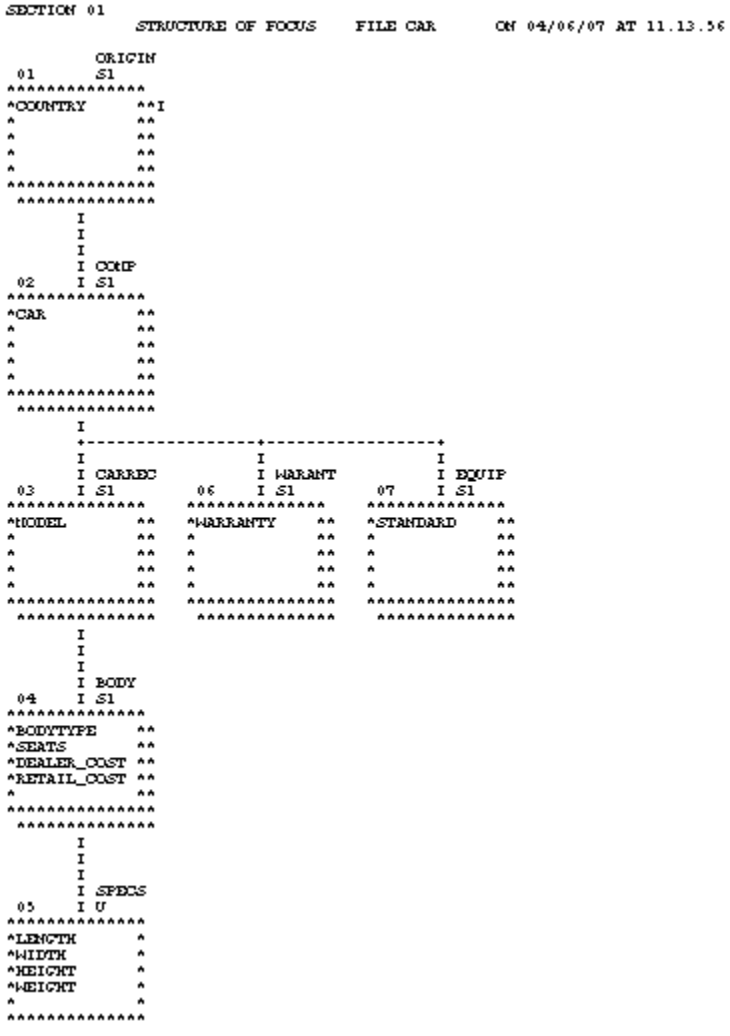
標準装備を列記します。

CAR マスターファイルでは、「ALIAS=」という記述をせずに ALIAS が指定されています。

CAR マスターファイル

```
FILENAME=CAR,SUFFIX=FOC
 SEGNAME=ORIGIN,SEGTYPE=S1
  FIELDNAME=COUNTRY,COUNTRY,A10,FIELDTYPE=I,$
 SEGNAME=COMP,SEGTYPE=S1,PARENT=ORIGIN
  FIELDNAME=CAR,CARS,A16,$
 SEGNAME=CARREC,SEGTYPE=S1,PARENT=COMP
  FIELDNAME=MODEL,MODEL,A24,$
 SEGNAME=BODY,SEGTYPE=S1,PARENT=CARREC
  FIELDNAME=BODYTYPE,TYPE,A12,$
  FIELDNAME=SEATS,SEAT,I3,$
  FIELDNAME=DEALER_COST,DCOST,D7,$
  FIELDNAME=RETAIL_COST,RCOST,D7,$
  FIELDNAME=SALES,UNITS,I6,$
 SEGNAME=SPECS,SEGTYPE=U,PARENT=BODY
  FIELDNAME=LENGTH,LEN,D5,$
  FIELDNAME=WIDTH,WIDTH,D5,$
  FIELDNAME=HEIGHT,HEIGHT,D5,$
  FIELDNAME=WEIGHT,WEIGHT,D6,$
  FIELDNAME=WHEELBASE,BASE,D6.1,$
  FIELDNAME=FUEL_CAP,FUEL,D6.1,$
  FIELDNAME=BHP,POWER,D6,$
  FIELDNAME=RPM,RPM,I5,$
  FIELDNAME=MPG,MILES,D6,$
  FIELDNAME=ACCEL,SECONDS,D6,$
 SEGNAME=WARANT,SEGTYPE=S1,PARENT=COMP
  FIELDNAME=WARRANTY,WARR,A40,$
 SEGNAME=EQUIP,SEGTYPE=S1,PARENT=COMP
  FIELDNAME=STANDARD,EQUIP,A40,$
```

CAR 構造図



LEDGER データソース

LEDGER データソースには、サンプルの会計データが保存されています。このデータソースは、「TOP」という 1 つのセグメントで構成されています。このデータソースは、当初は財務レポートの事例として指定されたものです。このマスターファイルのフィールドにはエイリアスは存在せず、カンマ (,) がプレースホルダとして使用されています。

LEDGER マスターファイル

```
FILENAME=LEDGER, SUFFIX=FOC,$
 SEGNAME=TOP,    SEGTYPE=S2,$
  FIELDNAME=YEAR    ,   , FORMAT=A4, $
  FIELDNAME=ACCOUNT,   , FORMAT=A4, $
  FIELDNAME=AMOUNT ,   , FORMAT=I5C,$
```

LEDGER 構造図

```
SECTION 01
           STRUCTURE OF FOCUS    FILE LEDGER   ON 05/15/03 AT 15.17.08

          TOP
  01      S2
 **************
 *YEAR         **
 *ACCOUNT      **
 *AMOUNT       **
 *             **
 *             **
 ***************
  ***************
```

FINANCE データソース

FINANCE データソースには、バランスシート用のサンプル財務データが保存されています。このデータソースは、「TOP」という 1 つのセグメントで構成されています。このデータソースは、当初は財務レポートの事例として指定されたものです。このマスターファイルのフィールドにはエイリアスは存在せず、カンマ (,) がプレースホルダとして使用されています。

FINANCE マスターファイル

```
FILENAME=FINANCE, SUFFIX=FOC,$
 SEGNAME=TOP,     SEGTYPE=S2,$
  FIELDNAME=YEAR    ,   , FORMAT=A4,  $
  FIELDNAME=ACCOUNT,   , FORMAT=A4,  $
  FIELDNAME=AMOUNT ,   , FORMAT=D12C,$
```

FINANCE 構造図

```
SECTION 01
           STRUCTURE OF FOCUS    FILE FINANCE  ON 05/15/03 AT 15.17.08

          TOP
  01      S2
 **************
 *YEAR         **
 *ACCOUNT      **
 *AMOUNT       **
 *             **
 *             **
 ***************
  ***************
```

REGION データソース

REGION データソースには、国の東部地区および西部地区のサンプル会計データが保存されています。このデータソースは、「TOP」という 1 つのセグメントで構成されています。このデータソースは、当初は財務レポートの事例として指定されたものです。このマスターファイルのフィールドにはエイリアスは存在せず、カンマ (,) がプレースホルダとして使用されています。

REGION マスターファイル

```
FILENAME=REGION, SUFFIX=FOC,$
 SEGNAME=TOP,    SEGTYPE=S1,$
  FIELDNAME=ACCOUNT,   , FORMAT=A4, $
  FIELDNAME=E_ACTUAL, , FORMAT=I5C,$
  FIELDNAME=E_BUDGET, , FORMAT=I5C,$
  FIELDNAME=W_ACTUAL, , FORMAT=I5C,$
  FIELDNAME=W_BUDGET, , FORMAT=I5C,$
```

REGION 構造図

```
SECTION 01
           STRUCTURE OF FOCUS    FILE REGION   ON 05/15/03 AT 15.18.48

          TOP
  01      S1
 **************
 *ACCOUNT     **
 *E_ACTUAL    **
 *E_BUDGET    **
 *W_ACTUAL    **
 *            **
 ***************
  **************
```

EMPDATA データソース

EMPDATA データソースには、企業の従業員に関するサンプルデータが保存されています。このデータソースは、「EMPDATA」という 1 つのセグメントで構成されています。PIN フィールドはインデックスフィールドです。AREA フィールドは一時項目です。

EMPDATA マスターファイル

```
FILENAME=EMPDATA, SUFFIX=FOC
 SEGNAME=EMPDATA, SEGTYPE=S1
  FIELDNAME=PIN,          ALIAS=ID,       FORMAT=A9,  INDEX=I,    $
  FIELDNAME=LASTNAME,     ALIAS=LN,       FORMAT=A15,             $
  FIELDNAME=FIRSTNAME,    ALIAS=FN,       FORMAT=A10,             $
  FIELDNAME=MIDINITIAL,   ALIAS=MI,       FORMAT=A1,              $
  FIELDNAME=DIV,          ALIAS=CDIV,     FORMAT=A4,              $
  FIELDNAME=DEPT,         ALIAS=CDEPT,    FORMAT=A20,             $
  FIELDNAME=JOBCLASS,     ALIAS=CJCLAS,   FORMAT=A8,              $
  FIELDNAME=TITLE,        ALIAS=CFUNC,    FORMAT=A20,             $
  FIELDNAME=SALARY,       ALIAS=CSAL,     FORMAT=D12.2M,          $
  FIELDNAME=HIREDATE,     ALIAS=HDAT,     FORMAT=YMD,             $
$
DEFINE AREA/A13=DECODE DIV (NE 'NORTH EASTERN' SE 'SOUTH EASTERN'
CE 'CENTRAL' WE 'WESTERN' CORP 'CORPORATE' ELSE 'INVALID AREA');$
```

EMPDATA 構造図

```
SECTION 01
            STRUCTURE OF FOCUS    FILE EMPDATA  ON 05/15/03 AT 14.49.09

          EMPDATA
  01      S1
 **************
 *PIN          **I
 *LASTNAME     **
 *FIRSTNAME    **
 *MIDINITIAL   **
 *             **
 ***************
  **************
```

TRAINING データソース

TRAINING データソースには、従業員のトレーニングコースに関するサンプルデータが保存されています。このデータソースは、「TRAINING」という 1 つのセグメントで構成されています。PIN フィールドはインデックスフィールドです。EXPENSES、GRADE、LOCATION の各フィールドには、MISSING=ON 属性が指定されています。

TRAINING マスターファイル

```
FILENAME=TRAINING, SUFFIX=FOC
 SEGNAME=TRAINING, SEGTYPE=SH3
  FIELDNAME=PIN,           ALIAS=ID,     FORMAT=A9,    INDEX=I,    $
  FIELDNAME=COURSESTART,   ALIAS=CSTART, FORMAT=YMD,               $
  FIELDNAME=COURSECODE,    ALIAS=CCOD,   FORMAT=A7,                $
  FIELDNAME=EXPENSES,      ALIAS=COST,   FORMAT=D8.2, MISSING=ON  $
  FIELDNAME=GRADE,         ALIAS=GRA,    FORMAT=A2,   MISSING=ON, $
  FIELDNAME=LOCATION,      ALIAS=LOC,    FORMAT=A6,   MISSING=ON, $
```

TRAINING 構造図

```
SECTION 01
            STRUCTURE OF FOCUS    FILE TRAINING ON 05/15/03 AT 14.51.28

          TRAINING
  01      SH3
 **************
 *PIN          **I
 *COURSESTART  **
 *COURSECODE   **
 *EXPENSES     **
 *             **
 ***************
  **************
```

COURSE データソース

COURSE データソースには、教育コースに関するサンプルデータが保存されています。このデータソースは、CRSELIST セグメントのみで構成されています。

COURSE マスターファイル

```
FILENAME=COURSE,   SUFFIX=FOC
 SEGNAME=CRSELIST,  SEGTYPE=S1
  FIELDNAME=COURSECODE,  ALIAS=CCOD,   FORMAT=A7,   INDEX=I,    $
  FIELDNAME=CTITLE,      ALIAS=COURSE, FORMAT=A35,              $
  FIELDNAME=SOURCE,      ALIAS=ORG,    FORMAT=A35,              $
  FIELDNAME=CLASSIF,     ALIAS=CLASS,  FORMAT=A10,              $
  FIELDNAME=TUITION,     ALIAS=FEE,    FORMAT=D8.2, MISSING=ON, $
  FIELDNAME=DURATION,    ALIAS=DAYS,   FORMAT=A3,   MISSING=ON, $
  FIELDNAME=DESCRIPTN1,  ALIAS=DESC1,  FORMAT=A40,              $
  FIELDNAME=DESCRIPTN2,  ALIAS=DESC2,  FORMAT=A40,              $
  FIELDNAME=DESCRIPTN2,  ALIAS=DESC3,  FORMAT=A40,              $
```

COURSE 構造図

```
SECTION 01
            STRUCTURE OF FOCUS    FILE COURSE   ON 05/15/03 AT 12.26.05

         CRSELIST
 01      S1
**************
*COURSECODE  **I
*CTITLE      **
*SOURCE      **
*CLASSIF     **
*            **
***************
 **************
```

JOBHIST データソース

JOBHIST データソースには、従業員の職務に関する情報が保存されています。PIN および JOBUSTART フィールドの両方がキーフィールドです。PIN フィールドはインデックスフィールドです。

JOBHIST マスターファイル

```
FILENAME=JOBHIST, SUFFIX=FOC
SEGNAME=JOBHIST, SEGTYPE=SH2
 FIELDNAME=PIN,          ALIAS=ID,       FORMAT=A9,      INDEX=I ,$
 FIELDNAME=JOBSTART,     ALIAS=SDAT,     FORMAT=YMD,              $
 FIELDNAME=JOBCLASS,     ALIAS=JCLASS,   FORMAT=A8,               $
 FIELDNAME=FUNCTITLE,    ALIAS=FUNC,     FORMAT=A20,              $
```

JOBHIST 構造図

```
SECTION 01
     STRUCTURE OF FOCUS    FILE JOBHIST  ON 01/22/08 AT 16.23.46
    JOBHIST
  01      SH2
 **************
 *PIN         **I
 *JOBSTART    **
 *JOBCLASS    **
 *FUNCTITLE   **
 *            **
 ***************
  ***************
```

JOBLIST データソース

JOBLIST データソースには、職務に関する情報が保存されています。JOBCLASS フィールドはインデックスフィールドです。

JOBLIST マスターファイル

```
FILENAME=JOBLIST, SUFFIX=FOC
SEGNAME=JOBSEG,   SEGTYPE=S1
 FIELDNAME=JOBCLASS,     ALIAS=JCLASS,   FORMAT=A8,    INDEX=I ,$
 FIELDNAME=CATEGORY,     ALIAS=JGROUP,   FORMAT=A25,             $
 FIELDNAME=JOBDESC,      ALIAS=JDESC,    FORMAT=A40,             $
 FIELDNAME=LOWSAL,       ALIAS=LSAL,     FORMAT=D12.2M,          $
 FIELDNAME=HIGHSAL,      ALIAS=HSAL,     FORMAT=D12.2M,          $
DEFINE GRADE/A2=EDIT  (JCLASS,'$$$99');$
DEFINE LEVEL/A25=DECODE GRADE (08 'GRADE 8' 09 'GRADE 9' 10
'GRADE 10' 11 'GRADE 11' 12 'GRADE 12' 13 'GRADE 13' 14 'GRADE 14');$
```

JOBLIST 構造図

```
SECTION 01
              STRUCTURE OF FOCUS    FILE JOBLIST  ON 01/22/08 AT 16.24.52
          JOBSEG
  01      S1
 **************
 *JOBCLASS     **I
 *CATEGORY     **
 *JOBDESC      **
 *LOWSAL       **
 *             **
 ***************
  **************
```

LOCATOR データソース

LOCATOR データソースには、従業員の勤務場所および電話番号に関する情報が保存されています。PIN フィールドはインデックスフィールドです。

LOCATOR マスターファイル

```
FILENAME=LOCATOR, SUFFIX=FOC
SEGNAME=LOCATOR,   SEGTYPE=S1,
 FIELDNAME=PIN,           ALIAS=ID_NO,    FORMAT=A9,   INDEX=I, $
 FIELDNAME=SITE,          ALIAS=SITE,     FORMAT=A25,           $
 FIELDNAME=FLOOR,         ALIAS=FL,       FORMAT=A3,            $
 FIELDNAME=ZONE,          ALIAS=ZONE,     FORMAT=A2,            $
 FIELDNAME=BUS_PHONE,     ALIAS=BTEL,     FORMAT=A5,            $
```

LOCATOR 構造図

```
SECTION 01
              STRUCTURE OF FOCUS    FILE LOCATOR  ON 01/22/08 AT 16.26.55
          LOCATOR
  01      S1
 **************
 *PIN          **I
 *SITE         **
 *FLOOR        **
 *ZONE         **
 *             **
 ***************
  **************
```

PERSINFO データソース

PERSINFO データソースには、従業員の個人情報が保存されています。PIN フィールドはインデックスフィールドです。

PERSINFO マスターファイル

```
FILENAME=PERSINFO, SUFFIX=FOC
SEGNAME=PERSONAL, SEGTYPE=S1
 FIELDNAME=PIN,           ALIAS=ID,       FORMAT=A9,   INDEX=I,    $
 FIELDNAME=INCAREOF,      ALIAS=ICO,      FORMAT=A35,              $
 FIELDNAME=STREETNO,      ALIAS=STR,      FORMAT=A20,              $
 FIELDNAME=APT,           ALIAS=APT,      FORMAT=A4,               $
 FIELDNAME=CITY,          ALIAS=CITY,     FORMAT=A20,              $
 FIELDNAME=STATE,         ALIAS=PROV,     FORMAT=A4,               $
 FIELDNAME=POSTALCODE,    ALIAS=ZIP,      FORMAT=A10,              $
 FIELDNAME=COUNTRY,       ALIAS=CTRY,     FORMAT=A15,              $
 FIELDNAME=HOMEPHONE,     ALIAS=TEL,      FORMAT=A10,              $
 FIELDNAME=EMERGENCYNO,   ALIAS=ENO,      FORMAT=A10,              $
 FIELDNAME=EMERGCONTACT, ALIAS=ENAME,     FORMAT=A35,              $
 FIELDNAME=RELATIONSHIP, ALIAS=REL,       FORMAT=A8,               $
 FIELDNAME=BIRTHDATE,     ALIAS=BDAT,     FORMAT=YMD,              $
```

PERSINFO 構造図

```
SECTION 01
     STRUCTURE OF FOCUS    FILE PERSINFO ON 01/22/08 AT 16.27.24
   PERSONAL
  01      S1
 **************
 *PIN          **I
 *INCAREOF     **
 *STREETNO     **
 *APT          **
 *             **
 ***************
  **************
```

SALHIST データソース

SALHIST には、従業員の給与履歴に関するデータが格納されています。PIN フィールドはインデックスフィールドです。PIN および EFFECTDATE フィールドの両方がキーフィールドです。

SALHIST マスターファイル

```
FILENAME=SALHIST,  SUFFIX=FOC
SEGNAME=SLHISTRY,  SEGTYPE=SH2
 FIELDNAME=PIN,         ALIAS=ID,     FORMAT=A9,     INDEX=I,  $
 FIELDNAME=EFFECTDATE, ALIAS=EDAT,    FORMAT=YMD,              $
 FIELDNAME=OLDSALARY,  ALIAS=OSAL,    FORMAT=D12.2,            $
```

SALHIST 構造図

```
SECTION 01
     STRUCTURE OF FOCUS    FILE SALHIST  ON 01/22/08 AT 16.28.02
    SLHISTRY
  01      SH2
 **************
 *PIN          **I
 *EFFECTDATE   **
 *OLDSALARY    **
 *             **
 *             **
 ***************
  **************
```

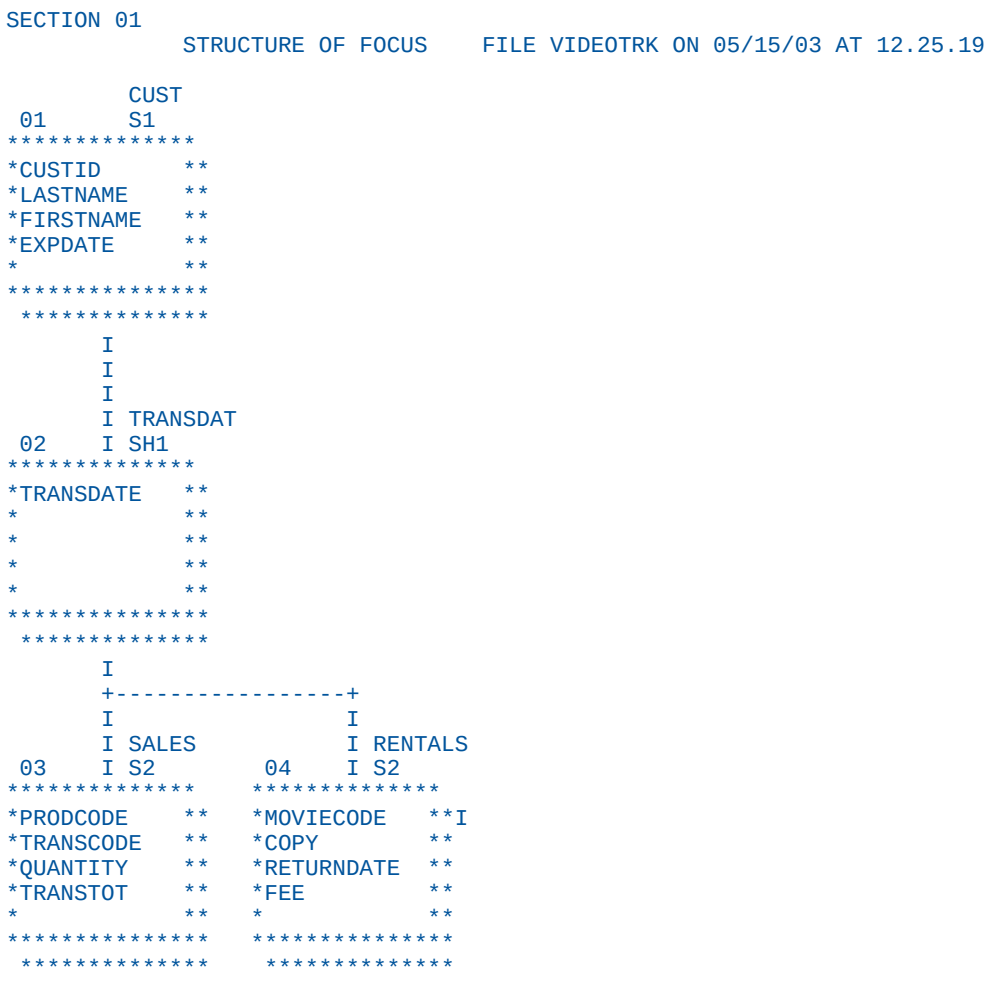
VIDEOTRK、MOVIES、ITEMS データソース

VIDEOTRK データソースには、レンタルビデオ事業の顧客、レンタル商品、注文情報に関するサンプルデータが保存されています。このデータソースは、MOVIES データソースまたは ITEMS のデータソースと結合することができます。

VIDEOTRK マスターファイル

```
FILENAME=VIDEOTRK, SUFFIX=FOC
 SEGNAME=CUST,     SEGTYPE=S1
  FIELDNAME=CUSTID,     ALIAS=CIN,          FORMAT=A4,    $
  FIELDNAME=LASTNAME,   ALIAS=LN,           FORMAT=A15,   $
  FIELDNAME=FIRSTNAME,  ALIAS=FN,           FORMAT=A10,   $
  FIELDNAME=EXPDATE,    ALIAS=EXDAT,        FORMAT=YMD,   $
  FIELDNAME=PHONE,      ALIAS=TEL,          FORMAT=A10,   $
  FIELDNAME=STREET,     ALIAS=STR,          FORMAT=A20,   $
  FIELDNAME=CITY,       ALIAS=CITY,         FORMAT=A20,   $
  FIELDNAME=STATE,      ALIAS=PROV,         FORMAT=A4,    $
  FIELDNAME=ZIP,        ALIAS=POSTAL_CODE,  FORMAT=A9,    $
SEGNAME=TRANSDAT,  SEGTYPE=SH1,  PARENT=CUST
  FIELDNAME=TRANSDATE,  ALIAS=OUTDATE,      FORMAT=YMD,   $
SEGNAME=SALES,     SEGTYPE=S2,   PARENT=TRANSDAT
  FIELDNAME=PRODCODE,   ALIAS=PCOD,         FORMAT=A6,    $
  FIELDNAME=TRANSCODE,  ALIAS=TCOD,         FORMAT=I3,    $
  FIELDNAME=QUANTITY,   ALIAS=NO,           FORMAT=I3S,   $
  FIELDNAME=TRANSTOT,   ALIAS=TTOT,         FORMAT=F7.2S, $
SEGNAME=RENTALS,   SEGTYPE=S2,   PARENT=TRANSDAT
  FIELDNAME=MOVIECODE,  ALIAS=MCOD,         FORMAT=A6, INDEX=I, $
  FIELDNAME=COPY,       ALIAS=COPY,         FORMAT=I2,    $
  FIELDNAME=RETURNDATE, ALIAS=INDATE,       FORMAT=YMD,   $
  FIELDNAME=FEE,        ALIAS=FEE,          FORMAT=F5.2S, $
```

VIDEOTRK 構造図



MOVIES マスターファイル

```
FILENAME=MOVIES,     SUFFIX=FOC
 SEGNAME=MOVINFO,    SEGTYPE=S1
  FIELDNAME=MOVIECODE,    ALIAS=MCOD,  FORMAT=A6, INDEX=I, $
  FIELDNAME=TITLE,        ALIAS=MTL,   FORMAT=A39,  $
  FIELDNAME=CATEGORY,     ALIAS=CLASS, FORMAT=A8,   $
  FIELDNAME=DIRECTOR,     ALIAS=DIR,   FORMAT=A17,  $
  FIELDNAME=RATING,       ALIAS=RTG,   FORMAT=A4,   $
  FIELDNAME=RELDATE,      ALIAS=RDAT,  FORMAT=YMD,  $
  FIELDNAME=WHOLESALEPR, ALIAS=WPRC,  FORMAT=F6.2, $
  FIELDNAME=LISTPR,       ALIAS=LPRC,  FORMAT=F6.2, $
  FIELDNAME=COPIES,       ALIAS=NOC,   FORMAT=I3,   $
```

MOVIES 構造図

```
SECTION 01
            STRUCTURE OF FOCUS    FILE MOVIES   ON 05/15/03 AT 12.26.05

         MOVINFO
 01      S1
**************
*MOVIECODE   **I
*TITLE       **
*CATEGORY    **
*DIRECTOR    **
*            **
***************
 **************
```

ITEMS マスターファイル

```
FILENAME=ITEMS,    SUFFIX=FOC
 SEGNAME=ITMINFO, SEGTYPE=S1
  FIELDNAME=PRODCODE,   ALIAS=PCOD,  FORMAT=A6,  INDEX=I, $
  FIELDNAME=PRODNAME,   ALIAS=PROD,  FORMAT=A20,          $
  FIELDNAME=OURCOST,    ALIAS=WCOST, FORMAT=F6.2,         $
  FIELDNAME=RETAILPR,   ALIAS=PRICE, FORMAT=F6.2,         $
  FIELDNAME=ON_HAND,    ALIAS=NUM,   FORMAT=I5,           $
```

ITEMS 構造図

```
SECTION 01
            STRUCTURE OF FOCUS    FILE ITEMS    ON 05/15/03 AT 12.26.05

         ITMINFO
 01      S1
**************
*PRODCODE    **I
*PRODNAME    **
*OURCOST     **
*RETAILPR    **
*            **
***************
 **************
```

Gotham Grinds データソース

Gotham Grinds は複数のデータソースで構成され、それぞれのデータソースには特選品を扱う企業に関するサンプルデータが保存されています。

- GGDEMOG データソースには、コーヒー、グルメ菓子、ギフトなどの特選品を販売する会社である Gotham Grinds の顧客に関する統計データが保存されています。詳細は、GGDEMOG を参照してください。このデータソースは、DEMOG01 セグメントのみで構成されています。
- GGORDER データソースには、Gotham Grinds の注文情報が保存されています。詳細は、GGORDER を参照してください。このデータソースは、ORDER01 と ORDER02 の 2 つのセグメントで構成されています。
- GGPRODS データソースには、Gotham Grinds の製品情報が保存されています。詳細は、GGPRODS を参照してください。このデータソースは、PRODS01 セグメントのみで構成されています。
- GGSALES データソースには、Gotham Grinds の売上情報が保存されています。詳細は、GGSALES を参照してください。このデータソースは、SALES01 セグメントのみで構成されています。
- GGSTORES データソースには、Gotham Grinds が米国内に持つ 12 店舗の個別情報が保存されています。STORES01 セグメントのみで構成されています。

GGDEMOG マスターファイル

```
FILENAME=GGDEMOG, SUFFIX=FOC
 SEGNAME=DEMOG01, SEGTYPE=S1
  FIELD=ST,       ALIAS=E02, FORMAT=A02, INDEX=I,TITLE='State',
   DESC='State',$
  FIELD=HH,       ALIAS=E03, FORMAT=I09, TITLE='Number of Households',
   DESC='Number of Households',$
  FIELD=AVGHHSZ98,ALIAS=E04, FORMAT=I09, TITLE='Average Household Size',
   DESC='Average Household Size',$
  FIELD=MEDHHI98, ALIAS=E05, FORMAT=I09, TITLE='Median Household Income',
   DESC='Median Household Income',$
  FIELD=AVGHHI98, ALIAS=E06, FORMAT=I09, TITLE='Average Household Income',
   DESC='Average Household Income',$
  FIELD=MALEPOP98,ALIAS=E07, FORMAT=I09, TITLE='Male Population',
   DESC='Male Population',$
  FIELD=FEMPOP98, ALIAS=E08, FORMAT=I09, TITLE='Female Population',
   DESC='Female Population',$
  FIELD=P15TO1998,ALIAS=E09, FORMAT=I09, TITLE='15 to 19',
   DESC='Population 15 to 19 years old',$
  FIELD=P20TO2998,ALIAS=E10, FORMAT=I09, TITLE='20 to 29',
   DESC='Population 20 to 29 years old',$
  FIELD=P30TO4998,ALIAS=E11, FORMAT=I09, TITLE='30 to 49',
   DESC='Population 30 to 49 years old',$
  FIELD=P50TO6498,ALIAS=E12, FORMAT=I09, TITLE='50 to 64',
   DESC='Population 50 to 64 years old',$
  FIELD=P65OVR98, ALIAS=E13, FORMAT=I09, TITLE='65 and over',
   DESC='Population 65 and over',$
```

GGDEMOG 構造図

```
SECTION 01
     STRUCTURE OF FOCUS    FILE GGDEMOG  ON 05/15/03 AT 12.26.05

         GGDEMOG
 01      S1
**************
*ST           **I
*HH           **
*AVGHHSZ98    **
*MEDHHI98     **
*             **
***************
 **************
```

GGORDER マスターファイル

```
FILENAME=GGORDER, SUFFIX=FOC,$
 SEGNAME=ORDER01, SEGTYPE=S1,$
  FIELD=ORDER_NUMBER, ALIAS=ORDNO1,   FORMAT=I6,  TITLE='Order,Number',
   DESC='Order Identification Number',$
  FIELD=ORDER_DATE,   ALIAS=DATE,     FORMAT=MDY, TITLE='Order,Date',
   DESC='Date order was placed',$
  FIELD=STORE_CODE,   ALIAS=STCD,     FORMAT=A5,  TITLE='Store,Code',
   DESC='Store Identification Code (for order)',$
  FIELD=PRODUCT_CODE, ALIAS=PCD,      FORMAT=A4,  TITLE='Product,Code',
   DESC='Product Identification Code (for order)',$
  FIELD=QUANTITY,     ALIAS=ORDUNITS, FORMAT=I8,  TITLE='Ordered,Units',
   DESC='Quantity Ordered',$
SEGNAME=ORDER02, SEGTYPE=KU, PARENT=ORDER01, CRFILE=GGPRODS, CRKEY=PCD,
CRSEG=PRODS01  ,$
```

GGORDER 構造図

```
SECTION 01
     STRUCTURE OF FOCUS    FILE GGORDER  ON 05/15/03 AT 16.45.48

          GGORDER
  01      S1
 **************
 *ORDER_NUMBER**
 *ORDER_DATE  **
 *STORE_CODE  **
 *PRODUCT_CODE**
 *            **
 **************
  **************
        I
        I
        I
        I ORDER02
  02    I KU
 ..............
 :PRODUCT_ID  :K
 :PRODUCT_DESC:
 :VENDOR_CODE :
 :VENDOR_NAME :
 :            :
 :............:
```

GGPRODS マスターファイル

```
FILENAME=GGPRODS, SUFFIX=FOC
 SEGNAME=PRODS01, SEGTYPE=S1
  FIELD=PRODUCT_ID, ALIAS=PCD, FORMAT=A4, INDEX=I, TITLE='Product,Code',
   DESC='Product Identification Code',$
  FIELD=PRODUCT_DESCRIPTION, ALIAS=PRODUCT, FORMAT=A16, TITLE='Product',
   DESC='Product Name',$
  FIELD=VENDOR_CODE, ALIAS=VCD, FORMAT=A4, INDEX=I, TITLE='Vendor ID',
   DESC='Vendor Identification Code',$
  FIELD=VENDOR_NAME, ALIAS=VENDOR, FORMAT=A23, TITLE='Vendor Name',
   DESC='Vendor Name',$
  FIELD=PACKAGE_TYPE, ALIAS=PACK, FORMAT=A7, TITLE='Package',
   DESC='Packaging Style',$
  FIELD=SIZE, ALIAS=SZ, FORMAT=I2, TITLE='Size',
   DESC='Package Size',$
  FIELD=UNIT_PRICE, ALIAS=UNITPR, FORMAT=D7.2, TITLE='Unit,Price',
   DESC='Price for one unit',$
```

GGPRODS 構造図

```
SECTION 01
     STRUCTURE OF FOCUS    FILE GGPRODS  ON 05/15/03 AT 12.26.05

         GGPRODS
 01      S1
**************
*PRODUCT_ID  **I
*PRODUCT_DESC**I
*VENDOR_CODE **
*VENDOR_NAME **
*            **
***************
 **************
```

GGSALES マスターファイル

```
FILENAME=GGSALES, SUFFIX=FOC
 SEGNAME=SALES01, SEGTYPE=S1
  FIELD=SEQ_NO, ALIAS=SEQ, FORMAT=I5, TITLE='Sequence#',
   DESC='Sequence number in database',$
  FIELD=CATEGORY, ALIAS=E02, FORMAT=A11, INDEX=I, TITLE='Category',
   DESC='Product category',$
  FIELD=PCD, ALIAS=E03, FORMAT=A04, INDEX=I, TITLE='Product ID',
   DESC='Product Identification code (for sale)',$
  FIELD=PRODUCT, ALIAS=E04, FORMAT=A16, TITLE='Product',
   DESC='Product name',$
  FIELD=REGION, ALIAS=E05, FORMAT=A11, INDEX=I, TITLE='Region',
   DESC='Region code',$
  FIELD=ST, ALIAS=E06, FORMAT=A02, INDEX=I, TITLE='State',
   DESC='State',$
  FIELD=CITY, ALIAS=E07, FORMAT=A20, TITLE='City',
   DESC='City',$
  FIELD=STCD, ALIAS=E08, FORMAT=A05, INDEX=I, TITLE='Store ID',
   DESC='Store identification code (for sale)',$
  FIELD=DATE, ALIAS=E09, FORMAT=I8YYMD, TITLE='Date',
   DESC='Date of sales report',$
  FIELD=UNITS, ALIAS=E10, FORMAT=I08, TITLE='Unit Sales',
   DESC='Number of units sold',$
  FIELD=DOLLARS, ALIAS=E11, FORMAT=I08, TITLE='Dollar Sales',
   DESC='Total dollar amount of reported sales',$
  FIELD=BUDUNITS, ALIAS=E12, FORMAT=I08, TITLE='Budget Units',
   DESC='Number of units budgeted',$
  FIELD=BUDDOLLARS, ALIAS=E13, FORMAT=I08, TITLE='Budget Dollars',
   DESC='Total sales quota in dollars',$
```

GGSALES 構造図

```
SECTION 01
     STRUCTURE OF FOCUS    FILE GGSALES  ON 05/15/03 AT 12.26.05

         GGSALES
 01      S1
**************
*SEQ_NO      **
*CATEGORY    **I
*PCD         **I
*PRODUCT     **I
*            **
***************
 **************
```

GGSTORES マスターファイル

```
FILENAME=GGSTORES, SUFFIX=FOC
 SEGNAME=STORES01, SEGTYPE=S1
  FIELD=STORE_CODE, ALIAS=E02, FORMAT=A05, INDEX=I, TITLE='Store ID',
   DESC='Franchisee ID Code',$
  FIELD=STORE_NAME, ALIAS=E03, FORMAT=A23, TITLE='Store Name',
   DESC='Store Name',$
  FIELD=ADDRESS1, ALIAS=E04, FORMAT=A19, TITLE='Contact',
   DESC='Franchisee Owner',$
  FIELD=ADDRESS2, ALIAS=E05, FORMAT=A31, TITLE='Address',
   DESC='Street Address',$
  FIELD=CITY, ALIAS=E06, FORMAT=A22, TITLE='City',
   DESC='City',$
  FIELD=STATE, ALIAS=E07, FORMAT=A02, INDEX=I, TITLE='State',
   DESC='State',$
  FIELD=ZIP, ALIAS=E08, FORMAT=A06, TITLE='Zip Code',
   DESC='Postal Code',$
```

GGSTORES 構造図

```
SECTION 01
     STRUCTURE OF FOCUS    FILE GGSTORES ON 05/15/03 AT 12.26.05

         GGSTORES
 01      S1
**************
*STORE_CODE  **I
*STORE_NAME  **
*ADDRESS1    **
*ADDRESS2    **
*            **
**************
 **************
```

Century Corp データソース

Century Corp は、世界中の小売店を通して製品を流通する家電製品のメーカーです。この企業は、世界中の工場、倉庫、オフィスに何千人もの従業員を配置しています。従業員の役割は、品質のよい製品とサービスを顧客に提供することです。

Century Corp データソースは複数のデータソースで構成され、これらのデータソースには財務、人材、在庫、注文に関するデータが保存されています。最後の 3 つのデータソースは、勘定科目の階層図と併せて使用するよう設計されています。

- CENTCOMP マスターファイルには、店舗のロケーション情報が格納されています。このデータソースは、COMPINFO セグメントのみで構成されています。
- CENTFIN マスターファイルには、財務情報が格納されています。このデータソースは、ROOT_SEG セグメントのみで構成されています。

- CENTHR マスターファイルには、人事情報が格納されています。このデータソースは、EMPSEG セグメントのみで構成されています。
- CENTINV マスターファイルには、在庫情報が格納されています。このデータソースは、INVINFO セグメントのみで構成されています。
- CENTORD マスターファイルには、注文情報が格納されています。このデータソースは、OINFO、STOSEG、PINFO、INVSEG の 4 つのセグメントで構成されています。
- CENTQA マスターファイルには、障害情報が格納されています。このデータソースは、PROD_SEG、INVSEG、PROB_SEG の 3 つのセグメントで構成されています。
- CENTGL マスターファイルには、勘定科目表が格納されています。GL_ACCOUNT_PARENT は、階層の親フィールドです。GL_ACCOUNT フィールドは、階層フィールドです。GL_ACCOUNT_CAPTION フィールドは、階層フィールドの説明キャプションとして使用することができます。
- CENTSYSF マスターファイルには、詳細レベルの財務データが格納されています。CENTSYSF は、異なる勘定科目体系 (SYS_ACCOUNT) を使用しますが、CENTGL の SYS_ACCOUNT フィールドと結合することができます。このデータは、通例の記号 (費用はプラス、収益はマイナス) を使用しています。

CENTCOMP マスターファイル

```
FILE=CENTCOMP, SUFFIX=FOC, FDFC=19, FYRT=00
  SEGNAME=COMPINFO, SEGTYPE=S1, $
  FIELD=STORE_CODE, ALIAS=SNUM, FORMAT=A6, INDEX=I,
   TITLE='Store Id#:',
   DESCRIPTION='Store Id#', $
  FIELD=STORENAME, ALIAS=SNAME, FORMAT=A20,
   WITHIN=STATE,
   TITLE='Store,Name:',
   DESCRIPTION='Store Name', $
  FIELD=STATE, ALIAS=STATE, FORMAT=A2,
   WITHIN=PLANT,
   TITLE='State:',
   DESCRIPTION=State, $
  DEFINE REGION/A5=DECODE STATE ('AL' 'SOUTH' 'AK' 'WEST' 'AR' 'SOUTH'
  'AZ' 'WEST' 'CA' 'WEST' 'CO' 'WEST' 'CT' 'EAST'
  'DE' 'EAST' 'DC' 'EAST' 'FL' 'SOUTH' 'GA' 'SOUTH' 'HI' 'WEST'
  'ID' 'WEST' 'IL' 'NORTH' 'IN' 'NORTH' 'IA' 'NORTH'
  'KS' 'NORTH' 'KY' 'SOUTH' 'LA' 'SOUTH' 'ME' 'EAST' 'MD' 'EAST'
  'MA' 'EAST' 'MI' 'NORTH' 'MN' 'NORTH' 'MS' 'SOUTH' 'MT' 'WEST'
  'MO' 'SOUTH' 'NE' 'WEST'  'NV' 'WEST' 'NH' 'EAST' 'NJ' 'EAST'
  'NM' 'WEST' 'NY' 'EAST' 'NC' 'SOUTH' 'ND' 'NORTH' 'OH' 'NORTH'
  'OK' 'SOUTH' 'OR' 'WEST' 'PA' 'EAST' 'RI' 'EAST' 'SC' 'SOUTH'
  'SD' 'NORTH' 'TN' 'SOUTH' 'TX' 'SOUTH' 'UT' 'WEST' 'VT' 'EAST'
  'VA' 'SOUTH' 'WA' 'WEST' 'WV' 'SOUTH' 'WI' 'NORTH' 'WY' 'WEST'
  'NA' 'NORTH' 'ON' 'NORTH' ELSE ' ');,
   TITLE='Region:',
   DESCRIPTION=Region, $
```

CENTCOMP 構造図

```
SECTION 01
       STRUCTURE OF FOCUS    FILE CENTCOMP ON 05/15/03 AT 10.20.49

          COMPINFO
  01      S1
 **************
 *STORE_CODE  **I
 *STORENAME   **
 *STATE       **
 *            **
 *            **
 ***************
  **************
```

CENTFIN マスターファイル

```
FILE=CENTFIN, SUFFIX=FOC, FDFC=19, FYRT=00
  SEGNAME=ROOT_SEG, SEGTYPE=S4, $
  FIELD=YEAR, ALIAS=YEAR, FORMAT=YY,
   WITHIN='*Time Period', $
  FIELD=QUARTER, ALIAS=QTR, FORMAT=Q,
   WITHIN=YEAR,
   TITLE=Quarter,
   DESCRIPTION=Quarter, $
  FIELD=MONTH, ALIAS=MONTH, FORMAT=M,
   TITLE=Month,
   DESCRIPTION=Month, $
  FIELD=ITEM, ALIAS=ITEM, FORMAT=A20,
   TITLE=Item,
   DESCRIPTION=Item, $
  FIELD=VALUE, ALIAS=VALUE, FORMAT=D12.2,
   TITLE=Value,
   DESCRIPTION=Value, $
  DEFINE ITYPE/A12=IF EDIT(ITEM,'9$$$$$$$$$$$$$$$$$$$') EQ 'E'
   THEN 'Expense' ELSE IF EDIT(ITEM,'9$$$$$$$$$$$$$$$$$$$') EQ 'R'
   THEN 'Revenue' ELSE 'Asset';,
   TITLE=Type,
   DESCRIPTION='Type of Financial Line Item',$
  DEFINE MOTEXT/MT=MONTH;,$
```

CENTFIN 構造図

```
SECTION 01
       STRUCTURE OF FOCUS    FILE CENTFIN  ON 05/15/03 AT 10.25.52

          ROOT_SEG
  01      S4
 **************
 *YEAR         **
 *QUARTER      **
 *MONTH        **
 *ITEM         **
 *             **
 ***************
  **************
```

CENTHR マスターファイル

```
FILE=CENTHR, SUFFIX=FOC
  SEGNAME=EMPSEG, SEGTYPE=S1, $
  FIELD=ID_NUM, ALIAS=ID#, FORMAT=I9,
   TITLE='Employee,ID#',
   DESCRIPTION='Employee Identification Number', $
  FIELD=LNAME, ALIAS=LN, FORMAT=A14,
   TITLE='Last,Name',
   DESCRIPTION='Employee Last Name', $
  FIELD=FNAME, ALIAS=FN, FORMAT=A12,
   TITLE='First,Name',
   DESCRIPTION='Employee First Name', $
  FIELD=PLANT, ALIAS=PLT, FORMAT=A3,
   TITLE='Plant,Location',
   DESCRIPTION='Location of the manufacturing plant',
   WITHIN='*Location', $
  FIELD=START_DATE, ALIAS=SDATE, FORMAT=YYMD,
   TITLE='Starting,Date',
   DESCRIPTION='Date of employment',$
  FIELD=TERM_DATE, ALIAS=TERM_DATE, FORMAT=YYMD,
   TITLE='Termination,Date',
   DESCRIPTION='Termination Date', $
  FIELD=STATUS, ALIAS=STATUS, FORMAT=A10,
   TITLE='Current,Status',
   DESCRIPTION='Job Status', $
  FIELD=POSITION, ALIAS=JOB, FORMAT=A2,
   TITLE=Position,
   DESCRIPTION='Job Position',  $
  FIELD=PAYSCALE, ALIAS=PAYLEVEL, FORMAT=I2,
   TITLE='Pay,Level',
   DESCRIPTION='Pay Level',
   WITHIN='*Wages',$
  DEFINE POSITION_DESC/A17=IF POSITION EQ 'BM' THEN
   'Plant Manager' ELSE
   IF POSITION EQ 'MR' THEN 'Line Worker' ELSE
   IF POSITION EQ 'TM' THEN 'Line Manager' ELSE
   'Technician';
   TITLE='Position,Description',
   DESCRIPTION='Position Description',
   WITHIN='PLANT',$
  DEFINE BYEAR/YY=START_DATE;
   TITLE='Beginning,Year',
   DESCRIPTION='Beginning Year',
   WITHIN='*Starting Time Period',$
```

```
DEFINE BQUARTER/Q=START_DATE;
 TITLE='Beginning,Quarter',
 DESCRIPTION='Beginning Quarter',
 WITHIN='BYEAR',
DEFINE BMONTH/M=START_DATE;
 TITLE='Beginning,Month',
 DESCRIPTION='Beginning Month',
 WITHIN='BQUARTER',$
DEFINE EYEAR/YY=TERM_DATE;
 TITLE='Ending,Year',
 DESCRIPTION='Ending Year',
 WITHIN='*Termination Time Period',$
DEFINE EQUARTER/Q=TERM_DATE;
 TITLE='Ending,Quarter',
 DESCRIPTION='Ending Quarter',
 WITHIN='EYEAR',$
DEFINE EMONTH/M=TERM_DATE;
 TITLE='Ending,Month',
 DESCRIPTION='Ending Month',
 WITHIN='EQUARTER',$
DEFINE RESIGN_COUNT/I3=IF STATUS EQ 'RESIGNED' THEN 1
 ELSE 0;
 TITLE='Resigned,Count',
 DESCRIPTION='Resigned Count',$
DEFINE FIRE_COUNT/I3=IF STATUS EQ 'TERMINAT' THEN 1
 ELSE 0;
 TITLE='Terminated,Count',
 DESCRIPTION='Terminated Count',$
DEFINE DECLINE_COUNT/I3=IF STATUS EQ 'DECLINED' THEN 1
 ELSE 0;
 TITLE='Declined,Count',
 DESCRIPTION='Declined Count',$
DEFINE EMP_COUNT/I3=IF STATUS EQ 'EMPLOYED' THEN 1
 ELSE 0;
 TITLE='Employed,Count',
 DESCRIPTION='Employed Count',$
DEFINE PEND_COUNT/I3=IF STATUS EQ 'PENDING' THEN 1
 ELSE 0;
 TITLE='Pending,Count',
 DESCRIPTION='Pending Count',$
DEFINE REJECT_COUNT/I3=IF STATUS EQ 'REJECTED' THEN 1
 ELSE 0;
 TITLE='Rejected,Count',
 DESCRIPTION='Rejected Count',$
DEFINE FULLNAME/A28=LNAME||', '|FNAME;
 TITLE='Full Name',
 DESCRIPTION='Full Name: Last, First', WITHIN='POSITION_DESC',$
```

```
DEFINE SALARY/D12.2=IF BMONTH LT 4 THEN PAYLEVEL * 12321
 ELSE IF BMONTH GE 4 AND BMONTH LT 8 THEN PAYLEVEL * 13827
 ELSE PAYLEVEL * 14400;,
 TITLE='Salary',
 DESCRIPTION='Salary',$
DEFINE PLANTLNG/A11=DECODE PLANT (BOS 'Boston' DAL 'Dallas'
 LA 'Los Angeles' ORL 'Orlando' SEA 'Seattle' STL 'St Louis'
 ELSE 'n/a');$
```

CENTHR 構造図

```
SECTION 01
     STRUCTURE OF FOCUS    FILE CENTHR   ON 05/15/03 AT 10.40.34

          EMPSEG
  01      S1
 **************
 *ID_NUM      **
 *LNAME       **
 *FNAME       **
 *PLANT       **
 *            **
 ***************
  **************
```

CENTINV マスターファイル

```
FILE=CENTINV, SUFFIX=FOC, FDFC=19, FYRT=00
 SEGNAME=INVINFO, SEGTYPE=S1, $
  FIELD=PROD_NUM, ALIAS=PNUM, FORMAT=A4, INDEX=I,
   TITLE='Product,Number:',
   DESCRIPTION='Product Number', $
  FIELD=PRODNAME, ALIAS=PNAME, FORMAT=A30,
   WITHIN=PRODCAT,
   TITLE='Product,Name:',
   DESCRIPTION='Product Name', $
  FIELD=QTY_IN_STOCK, ALIAS=QIS, FORMAT=I7,
   TITLE='Quantity,In Stock:',
   DESCRIPTION='Quantity In Stock', $
  FIELD=PRICE, ALIAS=RETAIL, FORMAT=D10.2,
   TITLE='Price:',
   DESCRIPTION=Price, $
  FIELD=COST, ALIAS=OUR_COST, FORMAT=D10.2,
   TITLE='Our,Cost:',
   DESCRIPTION='Our Cost:', $
  DEFINE PRODCAT/A22 = IF PRODNAME CONTAINS 'LCD'
   THEN 'VCRs' ELSE IF PRODNAME
   CONTAINS 'DVD' THEN 'DVD' ELSE IF PRODNAME CONTAINS 'Camcor'
   THEN 'Camcorders'
   ELSE IF PRODNAME CONTAINS 'Camera' THEN 'Cameras' ELSE IF PRODNAME
   CONTAINS 'CD' THEN 'CD Players'
   ELSE IF PRODNAME CONTAINS 'Tape' THEN 'Digital Tape Recorders'
   ELSE IF PRODNAME CONTAINS 'Combo' THEN 'Combo Players'
   ELSE 'PDA Devices'; WITHIN=PRODTYPE, TITLE='Product Category:' ,$
  DEFINE PRODTYPE/A19 = IF PRODNAME CONTAINS 'Digital' OR 'DVD' OR 'QX'
   THEN 'Digital' ELSE 'Analog';,WITHIN='*Product Dimension',
   TITLE='Product Type:',$
```

CENTINV 構造図

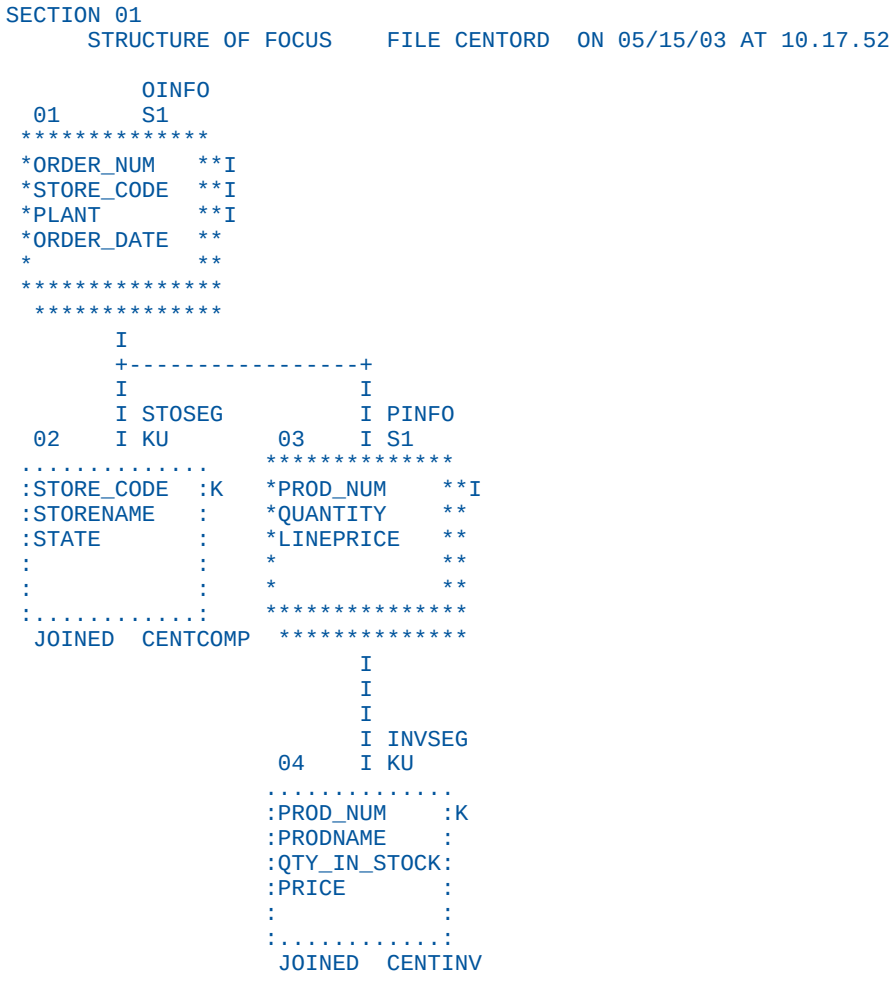
```
SECTION 01
       STRUCTURE OF FOCUS    FILE CENTINV  ON 05/15/03 AT 10.43.35

          INVINFO
  01      S1
 **************
 *PROD_NUM    **I
 *PRODNAME    **
 *QTY_IN_STOCK**
 *PRICE       **
 *            **
 ***************
  **************
```

CENTORD マスターファイル

```
FILE=CENTORD, SUFFIX=FOC
 SEGNAME=OINFO, SEGTYPE=S1, $
  FIELD=ORDER_NUM, ALIAS=ONUM, FORMAT=A5, INDEX=I,
   TITLE='Order,Number:',
   DESCRIPTION='Order Number', $
  FIELD=ORDER_DATE, ALIAS=ODATE, FORMAT=YYMD,
   TITLE='Date,Of Order:',
   DESCRIPTION='Date Of Order', $
  FIELD=STORE_CODE, ALIAS=SNUM, FORMAT=A6, INDEX=I,
   TITLE='Company ID#:',
   DESCRIPTION='Company ID#', $
  FIELD=PLANT, ALIAS=PLNT, FORMAT=A3, INDEX=I,
   TITLE='Manufacturing,Plant',
   DESCRIPTION='Location Of Manufacturing Plant',
   WITHIN='*Location',$
  DEFINE YEAR/YY=ORDER_DATE;,
   WITHIN='*Time Period',$
  DEFINE QUARTER/Q=ORDER_DATE;,
   WITHIN='YEAR',$
  DEFINE MONTH/M=ORDER_DATE;,
   WITHIN='QUARTER',$
 SEGNAME=PINFO, SEGTYPE=S1, PARENT=OINFO, $
  FIELD=PROD_NUM, ALIAS=PNUM, FORMAT=A4,INDEX=I,
   TITLE='Product,Number#:',
   DESCRIPTION='Product Number#', $
  FIELD=QUANTITY, ALIAS=QTY, FORMAT=I8C,
   TITLE='Quantity:',
   DESCRIPTION=Quantity, $
  FIELD=LINEPRICE, ALIAS=LINETOTAL, FORMAT=D12.2MC,
   TITLE='Line,Total',
   DESCRIPTION='Line Total', $
  DEFINE LINE_COGS/D12.2=QUANTITY*COST;,
   TITLE='Line,Cost Of,Goods Sold',
   DESCRIPTION='Line cost of goods sold', $
  DEFINE PLANTLNG/A11=DECODE PLANT (BOS 'Boston' DAL 'Dallas'
   LA 'Los Angeles' ORL 'Orlando' SEA 'Seattle' STL 'St Louis'
   ELSE 'n/a');
 SEGNAME=INVSEG, SEGTYPE=DKU, PARENT=PINFO, CRFILE=CENTINV,
  CRKEY=PROD_NUM,   CRSEG=INVINFO,$
 SEGNAME=STOSEG, SEGTYPE=DKU, PARENT=OINFO, CRFILE=CENTCOMP,
  CRKEY=STORE_CODE, CRSEG=COMPINFO,$
```

CENTORD 構造図


SECTION 01
STRUCTURE OF FOCUS FILE CENTORD ON 05/15/03 AT 10.17.52
OINFO
01 S1
*ORDER_NUM **I
*STORE_CODE **I
*PLANT **I
*ORDER_DATE **
STOSEG
02 KU
:STORE_CODE :K
:STORENAME :
:STATE :
JOINED CENTCOMP
PINFO
03 S1
*PROD_NUM **I
*QUANTITY **
*LINEPRICE **
INVSEG
04 KU
:PROD_NUM :K
:PRODNAME :
:QTY_IN_STOCK:
:PRICE :
JOINED CENTINV

CENTQA マスターファイル

```
FILE=CENTQA, SUFFIX=FOC, FDFC=19, FYRT=00
 SEGNAME=PROD_SEG, SEGTYPE=S1, $
  FIELD=PROD_NUM, ALIAS=PNUM, FORMAT=A4, INDEX=I,
   TITLE='Product,Number',
   DESCRIPTION='Product Number', $
 SEGNAME=PROB_SEG, PARENT=PROD_SEG, SEGTYPE=S1, $
  FIELD=PROBNUM, ALIAS=PROBNO, FORMAT=I5,
   TITLE='Problem,Number',
   DESCRIPTION='Problem Number',
   WITHIN=PLANT,$
  FIELD=PLANT, ALIAS=PLT, FORMAT=A3, INDEX=I,
   TITLE=Plant,
   DESCRIPTION=Plant,
   WITHIN=PROBLEM_LOCATION,$
  FIELD=PROBLEM_DATE, ALIAS=PDATE, FORMAT=YYMD,
   TITLE='Date,Problem,Reported',
   DESCRIPTION='Date Problem Was Reported', $
  FIELD=PROBLEM_CATEGORY, ALIAS=PROBCAT, FORMAT=A20, $
   TITLE='Problem,Category',
   DESCRIPTION='Problem Category',
   WITHIN=*Problem,$
  FIELD=PROBLEM_LOCATION, ALIAS=PROBLOC, FORMAT=A10,
   TITLE='Location,Problem,Occurred',
   DESCRIPTION='Location Where Problem Occurred',
   WITHIN=PROBLEM_CATEGORY,$
  DEFINE PROB_YEAR/YY=PROBLEM_DATE;,
   TITLE='Year,Problem,Occurred',
   DESCRIPTION='Year Problem Occurred',
   WITHIN=*Time Period,$
  DEFINE PROB_QUARTER/Q=PROBLEM_DATE;
   TITLE='Quarter,Problem,Occurred',
   DESCRIPTION='Quarter Problem Occurred',
   WITHIN=PROB_YEAR,$
  DEFINE PROB_MONTH/M=PROBLEM_DATE;
   TITLE='Month,Problem,Occurred',
   DESCRIPTION='Month Problem Occurred',
   WITHIN=PROB_QUARTER,$
  DEFINE PROBLEM_OCCUR/I5 WITH PROBNUM=1;,
   TITLE='Problem,Occurrence'
   DESCRIPTION='# of times a problem occurs',$
  DEFINE PLANTLNG/A11=DECODE PLANT (BOS 'Boston' DAL 'Dallas'
   LA 'Los Angeles' ORL 'Orlando' SEA 'Seattle' STL 'St Louis'
   ELSE 'n/a');$
 SEGNAME=INVSEG, SEGTYPE=DKU, PARENT=PROD_SEG, CRFILE=CENTINV,
  CRKEY=PROD_NUM, CRSEG=INVINFO,$
```

CENTQA 構造図

```
SECTION 01
       STRUCTURE OF FOCUS    FILE CENTQA   ON 05/15/03 AT 10.46.43

          PROD_SEG
  01      S1
 **************
 *PROD_NUM    **I
 *            **
 *            **
 *            **
 *            **
 ***************
  ***************
        I
        +-----------------+
        I                 I
        I INVSEG          I PROB_SEG
  02    I KU        03    I S1
 ..............    **************
 :PROD_NUM    :K   *PROBNUM     **
 :PRODNAME    :    *PLANT       **I
 :QTY_IN_STOCK:    *PROBLEM_DATE**
 :PRICE       :    *PROBLEM_CAT>**
 :            :    *             **
 :............:    ***************
  JOINED  CENTINV   **************
```

CENTGL マスターファイル

```
FILE=CENTGL ,SUFFIX=FOC
 SEGNAME=ACCOUNTS, SEGTYPE=S1
  FIELDNAME=GL_ACCOUNT, ALIAS=GLACCT, FORMAT=A7,
   TITLE='Ledger,Account', FIELDTYPE=I, $
  FIELDNAME=GL_ACCOUNT_PARENT, ALIAS=GLPAR, FORMAT=A7,
   TITLE=Parent,
   PROPERTY=PARENT_OF, REFERENCE=GL_ACCOUNT, $
  FIELDNAME=GL_ACCOUNT_TYPE, ALIAS=GLTYPE, FORMAT=A1,
   TITLE=Type,$
  FIELDNAME=GL_ROLLUP_OP, ALIAS=GLROLL, FORMAT=A1,
   TITLE=Op, $
  FIELDNAME=GL_ACCOUNT_LEVEL, ALIAS=GLLEVEL, FORMAT=I3,
   TITLE=Lev, $
  FIELDNAME=GL_ACCOUNT_CAPTION, ALIAS=GLCAP, FORMAT=A30,
   TITLE=Caption,
   PROPERTY=CAPTION, REFERENCE=GL_ACCOUNT, $
  FIELDNAME=SYS_ACCOUNT, ALIAS=ALINE, FORMAT=A6,
   TITLE='System,Account,Line', MISSING=ON, $
```

CENTGL 構造図

```
SECTION 01
       STRUCTURE OF FOCUS    FILE CENTGL   ON 05/15/03 AT 15.18.48

          ACCOUNTS
  01      S1
 **************
 *GL_ACCOUNT  **I
 *GL_ACCOUNT_>**
 *GL_ACCOUNT_>**
 *GL_ROLLUP_OP**
 *            **
 ***************
  **************
```

CENTSYSF マスターファイル

```
FILE=CENTSYSF ,SUFFIX=FOC
 SEGNAME=RAWDATA ,SEGTYPE=S2
  FIELDNAME = SYS_ACCOUNT ,  ,A6    , FIELDTYPE=I,
   TITLE='System,Account,Line', $
  FIELDNAME = PERIOD      ,  ,YYM   , FIELDTYPE=I,$
  FIELDNAME = NAT_AMOUNT  ,  ,D10.0 , TITLE='Month,Actual', $
  FIELDNAME = NAT_BUDGET  ,  ,D10.0 , TITLE='Month,Budget', $
  FIELDNAME = NAT_YTDAMT  ,  ,D12.0 , TITLE='YTD,Actual',   $
  FIELDNAME = NAT_YTDBUD  ,  ,D12.0 , TITLE='YTD,Budget',   $
```

CENTSYSF 構造図

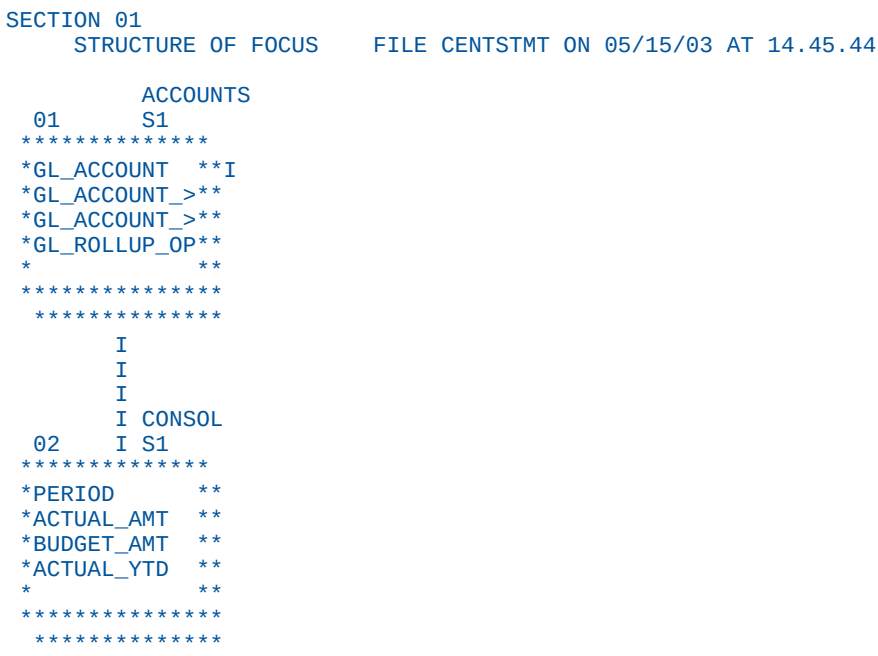
```
SECTION 01
       STRUCTURE OF FOCUS    FILE CENTSYSF   ON 05/15/03 AT 15.19.27

          RAWDATA
  01      S2
 **************
 *SYS_ACCOUNT **I
 *PERIOD      **I
 *NAT_AMOUNT  **
 *NAT_BUDGET  **
 *            **
 ***************
  **************
```

CENTSTMT マスターファイル

```
FILE=CENTSTMT, SUFFIX=FOC
 SEGNAME=ACCOUNTS, SEGTYPE=S1
  FIELD=GL_ACCOUNT, ALIAS=GLACCT,  FORMAT=A7,
   TITLE='Ledger,Account', FIELDTYPE=I, $
  FIELD=GL_ACCOUNT_PARENT, ALIAS=GLPAR, FORMAT=A7,
   TITLE=Parent,
   PROPERTY=PARENT_OF, REFERENCE=GL_ACCOUNT, $
  FIELD=GL_ACCOUNT_TYPE, ALIAS=GLTYPE, FORMAT=A1,
   TITLE=Type,$
  FIELD=GL_ROLLUP_OP, ALIAS=GLROLL, FORMAT=A1,
   TITLE=Op, $
  FIELD=GL_ACCOUNT_LEVEL, ALIAS=GLLEVEL, FORMAT=I3,
   TITLE=Lev, $
  FIELD=GL_ACCOUNT_CAPTION, ALIAS=GLCAP, FORMAT=A30,
   TITLE=Caption,
   PROPERTY=CAPTION, REFERENCE=GL_ACCOUNT, $
 SEGNAME=CONSOL, SEGTYPE=S1, PARENT=ACCOUNTS, $
  FIELD=PERIOD, ALIAS=MONTH, FORMAT=YYM, $
  FIELD=ACTUAL_AMT, ALIAS=AA, FORMAT=D10.0, MISSING=ON,
   TITLE='Actual', $
  FIELD=BUDGET_AMT, ALIAS=BA, FORMAT=D10.0, MISSING=ON,
   TITLE='Budget', $
  FIELD=ACTUAL_YTD, ALIAS=AYTD, FORMAT=D12.0, MISSING=ON,
   TITLE='YTD,Actual', $
  FIELD=BUDGET_YTD, ALIAS=BYTD, FORMAT=D12.0, MISSING=ON,
   TITLE='YTD,Budget', $
```

CENTSTMT 構造図



B

エラーメッセージ

エラーメッセージに関するテキストまたは説明は、エラーファイルに表示することができます。すべての WebFOCUS エラーメッセージは、8 つのシステムエラーファイルに格納されています。

- UNIX、Windows では、ファイルの拡張子は .err です。

メッセージの表示

メッセージのテキストおよび説明を表示するには、プロシジャで次のクエリコマンドを発行します。

```
? n
```

説明

`n`
メッセージ番号です。

メッセージ番号およびテキストが、メッセージの詳細説明 (存在する場合) とともに表示されます。たとえば、次のコマンドを発行します。

```
? 210
```

次のメッセージが表示されます。

```
(FOC210) 数値以外のデータがあります：
数値でなければならないところに英文字があります。
```

C

WebFOCUS の端数処理

ここでは、WebFOCUS での数値フィールドの格納および表示方法、演算での端数処理の方法、およびフォーマット変換時の動作について説明します。

データの格納および表示

データの端数処理は、フォーマットに応じて、格納または表示の前に実行されます。整数フィールド (フォーマット I) およびパック 10 進数フィールド (フォーマット P) は、格納前に端数処理されます。浮動小数点数フィールド (フォーマット F、D) および 10 進数浮動小数点数フィールド (フォーマット M、X) は、入力どおりに格納され、表示時に端数処理されます。

一般に、末尾の数値が 5 より小さい場合、データ値は切り捨てられます。末尾の数値が 5 以上の場合、データ値は切り上げられます。端数処理には、次のアルゴリズムが使用されます。

1. 入力値を 10 で乗算する。
2. ターゲットフォーマットの小数点以下の桁数と同じ回数分だけ乗算を繰り返す。たとえば、小数点以下 1 桁のパック 10 進数フィールドに 123.78 が入力された場合、以下のように、この数値は一度 10 で乗算されます。

   ```
   1237.8
   ```

3. 次に、以下のように、入力値が正の数の場合は 0.5 が加算され、負の数の場合は 0.5 が減算されます。

   ```
   1237.8 + 0.5 = 1238.3
   ```

   入力値が -123.78 の場合は、次のようになります。

   ```
   -1237.8 - 0.5 = -1238.3
   ```

4. 値は切り捨てられて、小数点が左側に移動します。

   ```
   123.8
   ```

   元の値が負の値の場合は、次のようになります。

   ```
   -123.8
   ```

下表は、WebFOCUS 数値フィールドフィーマット間での端数処理の相違を示しています。これらの相違についての詳細は、次のトピックで説明します。

フォーマット	タイプ	フォーマット	入力	格納	表示
I	整数	I3	123.78	0124	124
F	単精度浮動小数点数	F5.1 F3	123.78 123.78	123.7800 123.7800	124 123.8
D	倍精度浮動小数点数	D3 D5.1	123.78 123.78	123.78000000000000 123.78000000000000	124 123.8
M	10 進数浮動小数点数	M5.1 M3	123.78 123.78	123.7800 123.7800	124 123.8
X	拡張 10 進数浮動小数点数	X3 X5.1	123.78 123.78	123.78000000000000 123.78000000000000	124 123.8
P	パック 10 進数	P3 P5.1	123.78 123.78	0000124 00001238	124 123.8

注意：浮動小数点数 (D または F フォーマット) の場合、小数点以下の数値の格納値は、16 進数であり、実際の 10 進数の値よりも、わずかに小さい値に変換されることがあります。最も右側の位の値が 5 の場合、これらの数字が切り捨て (切り上げではなく) られます。SET FLOATMAPPING コマンドを使用して、倍精度の数値を 10 進数の数値として処理することにより、この問題を軽減することができます。

整数フィールド - I フォーマット

小数部の数値が含まれない整数値は、入力したとおりに格納されます。

トランザクションを介して小数部を含む値を整数フィールドに入力した場合、値が端数処理された上で、結果が格納されます。小数点以下の部分が 0.5 より小さい場合は値が切り捨てられ、0.5 以上の場合は切り上げられます。

ただし、整数フィールドが計算により求められる場合は、結果値の小数部は切り捨てられ、整数部分が格納 (表示) されます。たとえば、計算結果が 123.78 の場合、格納値は 123 です。

浮動小数点数フィールド - F、D フォーマット

F フォーマットタイプは、内部的に 4 バイトで格納される単精度浮動小数点数です。D フォーマットタイプは、内部的に 8 バイトで格納される倍精度浮動小数点数です。

F フォーマット、D フォーマットは、フィールドに割り当てられた格納サイズに達するまで、入力どおりに任意の数の小数部を格納します。数値が大きくなる場合、D フォーマットは、F フォーマットに比べてより正確です。これは、D フィールドには最大で 15 有効桁数を格納することができるのに対して、F フォーマットは、最大の 8 桁を超えると正確ではなくなるためです。浮動小数点数フィールドは、対数フォーマットで格納されます。先頭のバイトは指数を格納します。残りの 3 または 7 バイトは、仮数または値を格納します。

入力小数部の数がフォーマットで指定されている数を超える場合、F フォーマットおよび D フォーマットのフィールド値は、桁数の最大値に達するまでは、入力どおりに格納されます。これらの値の表示時には、フィールドフォーマットに応じて端数処理されます。たとえば、小数点以下 1 桁の浮動小数点数フォーマットのフィールドに 123.78 が入力された場合、123.78 が格納され、123.8 が表示されます。

10 進数浮動小数点フィールド - M、X フォーマット

M フォーマットタイプは、内部的に 8 バイトで格納される 10 進数浮動小数点数です。X フォーマットタイプは、内部的に 16 バイトで格納される拡張 10 進数浮動小数点数です。

M フォーマット、X フォーマットは、フィールドに割り当てられた精度の最大値に達するまで、入力どおりに任意の数の小数部を格納します。数値が大きくなる場合、X フォーマットは、M フォーマットに比べてより正確です。これは、X フィールドには最大で 37 有効桁数を格納することができるのに対して、M フォーマットは、最大の 15 桁を超えると正確ではなくなるためです。F フォーマットおよび D フォーマットと同様に、これらのフィールドは対数フォーマットで格納されます。先頭のバイトは指数を格納します。残りのバイトは、仮数または値を 2 進数フォーマットで格納します。M、X フォーマットと F、D フォーマットとの主要な相違は、F、D フォーマットは、指数を適用する際の底となる点です。M、X フォーマットは (10 を底とする) ベース 10 を使用するため、ベース 16 を使用する F、D フォーマットで見られる端数処理の問題が発生しません。

入力の小数部がフォーマットの格納小数部よりも大きい場合、M フィールドおよび X フィールドの値は、精度の最大値まで、入力どおりに格納されます。これらの値の表示時には、フィールドフォーマットに応じて端数処理されます。たとえば、小数点以下 1 桁の M または X フィールドに 123.78 が入力された場合、123.78 が格納され、123.8 が表示されます。

後続の HOLD ファイルへの浮動小数点数の格納方法を自動的に決定するには、SET FLOATMAPPING = {D|M|X} コマンドを使用することができます。デフォルトフォーマットは D です。

パック 10 進数フィールド - P フォーマット

パック 10 進数フォーマット (P フォーマットタイプ) では、各バイトに 2 桁が格納されます。ただし、最終バイトは例外で、ここには 1 桁の数値と符号 (D が負の数、C が正の数) が格納されます。

パック 10 進数フィールドの値は、格納される前に、フィールドフォーマットで指定された桁数に端数処理されます。入力の小数部がフォーマットの格納小数部よりも大きい場合、P フィールドの値は、最初に端数処理され、格納または表示されます。

パック 10 進数フィールドは、格納値に十分大きさの小数部がある場合に、正確になります。そうでない場合は、値は格納前に端数処理されることから、表示する小数部の桁数を増加しても、正確性が向上することはありません。たとえば、小数点以下 1 桁のパック 10 進数フィールドに 123.78 を入力した場合、123.8 が格納されます。さらに、COMPUTE または DEFINE を使用してフィールドフォーマットを P6.2 に変更した場合、123.80 が表示されます。フォーマットをマスターファイルで P6.2 に変更した場合、12.38 が表示されます。

注意：これまで継続的に数値表現ハンドラを改良し、より効率的で正確な結果が得られていますが、パック 10 進数フィールドを使用する際の端数処理に若干の差異が見られる場合があります。さまざまな改良により、パック 10 進数フィールドの数値を処理する際の計算精度が向上しています。パック 10 進数フィールドの端数処理は格納時に実行されるため、実際の数値が変更されます。この動作は、精度ベースのフィールドと異なります。精度ベースのフィールドの端数処理は表示時に実行されるため、元の数値が保持されます。

例　　値の格納および表示

浮動小数点数 (F または D フォーマット) の場合、小数点以下の数値の格納値は、16 進数であり、実際の 10 進数の値よりも、わずかに小さい値に変換されることがあります。最も右側の位の値が 5 の場合、これらの数字が切り捨て (切り上げではなく) られます。

次の例では、小数点以下 2 桁の入力値を、小数点以下 2 桁のパック 10 進数フィールド、小数点以下 1 桁のパック 10 進数フィールド、小数点以下 1 桁の D フィールド、小数点以下 1 桁の F フィールド、小数点以下 1 桁の M フィールド、小数点以下 1 桁の X フィールドに格納します。

マスターファイル

```
FILE=FIVE, SUFFIX=FOC
 SEGNAME=ONLY, SEGTYPE=S1,$
  FIELD=PACK2,,P5.2,$
  FIELD=PACK1,,P5.1,$
  FIELD=DOUBLE1,,D5.1,$
  FIELD=FLOAT1,,F5.1,$
  FIELD=MATH1,,M5.1,$
  FIELD=XMATH1,,X5.1,$
```

データロードプログラム

```
CREATE FILE FIVE
MODIFY FILE FIVE
FIXFORM  PACK2/5 PACK1/5 DOUBLE1/5 FLOAT1/5 MATH1/5 XMATH1/5
MATCH PACK2
  ON MATCH REJECT
  ON NOMATCH INCLUDE
DATA
1.05 1.05 1.05 1.05 1.05 1.05
1.15 1.15 1.15 1.15 1.15 1.15
1.25 1.25 1.25 1.25 1.25 1.25
1.35 1.35 1.35 1.35 1.35 1.35
1.45 1.45 1.45 1.45 1.45 1.45
1.55 1.55 1.55 1.55 1.55 1.55
1.65 1.65 1.65 1.65 1.65 1.65
1.75 1.75 1.75 1.75 1.75 1.75
1.85 1.85 1.85 1.85 1.85 1.85
1.95 1.95 1.95 1.95 1.95 1.95
END
```

TABLE リクエスト

TABLE リクエストは、これら 6 つのフィールドの値および合計を表示します。

```
TABLE FILE FIVE
PRINT PACK2 PACK1 DOUBLE1 FLOAT1 MATH1 XMATH1
ON TABLE SUMMARIZE
ON TABLE SET PAGE NOLEAD
END
```

次のレポートが表示されます。

```
PACK2  PACK1  DOUBLE1  FLOAT1  MATH1  XMATH1
-----  -----  -------  ------  -----  ------
 1.05    1.1      1.1     1.1    1.1     1.1
 1.15    1.2      1.1     1.1    1.2     1.2
 1.25    1.3      1.3     1.3    1.3     1.3
 1.35    1.4      1.4     1.4    1.4     1.4
 1.45    1.5      1.4     1.4    1.5     1.5
 1.55    1.6      1.6     1.6    1.6     1.6
 1.65    1.7      1.6     1.6    1.7     1.7
 1.75    1.8      1.8     1.8    1.8     1.8
 1.85    1.9      1.9     1.9    1.9     1.9
 1.95    2.0      1.9     1.9    2.0     2.0

TOTAL
15.00   15.5     15.0    15.0   15.0    15.0
```

PACK2 の値 1.15 は、単精度および倍精度の浮動小数点数フィールドが 1.1 に端数処理されます。これは、値 1.15 を 2 進数に正確に変換することができないためで、1.15 よりもわずかに小さい値 (例、1.1499999) として格納された上で、(切り上げではなく) 切り捨てられます。1.25 および 1.75 を除くすべての単精度および倍精度の値は、16 進数の繰り返し小数として格納されます。

PACK2 の値は、端数処理されません。これらは入力どおりに格納、表示されます。

PACK1 の値は格納の前に端数処理されるため、PACK1 の合計は PACK2 の合計よりも 0.5 大きくなっています。

D フィールドの値は入力時に格納され、表示用に端数処理されるため、D フィールドの合計は、PACK2 と同一です。

演算および変換での端数処理

ほとんどの演算は、浮動小数点数演算として処理されます。パック 10 進数フィールドは、内部的に D (倍精度浮動小数点数) フォーマットに変換され、P (パック 10 進数) フォーマットに戻されます。オペレーティングシステムがサポートする場合は、整数またはパック 10 進数 (8 バイト) フォーマットの加算および減算には、ネイティブ演算が使用されます。長パック 10 進数 (16 バイト) フォーマットは、拡張精度の数値に変換された上で計算されます。

小数点以下の数値を含むフィールドを計算して整数を求める場合、小数部は切り捨てられ、結果として得られる値は、入力値の整数部分になります。

小数点以下の数値を含むフィールドのフォーマットを別のフォーマットに変更する場合は、ネイティブ演算を使用する場合を除き、2 つの変換が実行されます。

1. 最初に、フィールドは内部的に浮動小数点数表記に変換されます。

2. 次に、この変換の結果が指定されたフォーマットに変換されます。この時点で、前述した端数処理アルゴリズムが適用されます。

例　フィールドフォーマットの再定義

次の例は、パック 10 進数フィールド、浮動小数点数フィールド、固定小数点数フィールド、整数フィールドの格納方法および表示方法の相違を示しています。また、フォーマットの再定義により小数点以下の桁数が増加されたデータベースの値も示しています。

マスターファイル

```
FILE=EXAMPLE, SUFFIX=FOC
 SEGNAME=ONLY, SEGTYPE=S1,$
  FIELD=PACKED2,,P9.2,$
  FIELD=DOUBLE2,,D9.2,$
  FIELD=FLOAT2,, F9.2,$
  FIELD=INTEGER,,I9  ,$
  FIELD=MATH2,,  M9.2,$
  FIELD=XMATH2,, X9.2,$
```

データロードプログラム

次に、これと同一のデータ値が各フィールドにロードされます。

```
CREATE FILE EXAMPLE
MODIFY FILE EXAMPLE
FIXFORM PACKED2/9 X1 DOUBLE2/9 X1 FLOAT2/9 X1 INTEGER/9 X1
FIXFORM MATH2/9 X1 XMATH2/9
MATCH PACKED2
ON MATCH REJECT
ON NOMATCH INCLUDE
DATA
1.6666666 1.6666666 1.6666666 1.6666666 1.6666666 1.6666666
125.16666 125.16666 125.16666 125.16666 125.16666 125.16666
5432.6666 5432.6666 5432.6666 5432.6666 5432.6666 5432.6666
4.1666666 4.1666666 4.1666666 4.1666666 4.1666666 4.1666666
5.5       5.5       5.5       5.5       5.5       5.5
106.66666 106.66666 106.66666 106.66666 106.66666 106.66666
7.2222222 7.2222222 7.2222222 7.2222222 7.2222222 7.2222222
END
```

TABLE リクエスト

DEFINE コマンドによって、PACKED2、DOUBLE2、FLOAT2、MATH2、XMATH2 と同等の一時フィールドが作成されます。これらのフィールドのフォーマットは、小数点以下の桁数が 2 から 4 に再設定されます。これらの DEFINE フィールドは、パック 10 進数フィールド、浮動小数点数フィールド、固定小数点数フィールドの格納方法および表示方法の相違を示しています。

リクエストは、これら 6 つのデータベースフィールド、および 5 つの DEFINE フィールドの値と合計を表示します。

```
DEFINE FILE EXAMPLE
PACKED4/P9.4=PACKED2;
DOUBLE4/D9.4=DOUBLE2;
FLOAT4/D9.4=FLOAT2;
MATH4/M9.4 = MATH2;
XMATH4/X9.4=XMATH2;
END

TABLE FILE EXAMPLE
PRINT PACKED2 PACKED4 DOUBLE2 DOUBLE4 FLOAT2 FLOAT4 MATH2 MATH4 XMATH2
XMATH4 INTEGER
ON TABLE SUMMARIZE
ON TABLE SET STYLE *
GRID=OFF,$
ENDSTYLE
END
```

下図は、Windows での出力結果を示しています。

PACKED2	PACKED4	DOUBLE2	DOUBLE4	FLOAT2	FLOAT4	MATH2	MATH4	XMATH2	XMATH4	INTEGER
1.66	1.6600	1.67	1.6667	1.67	1.6667	1.67	1.6667	1.67	1.6667	1
125.16	125.1600	125.17	125.1667	125.17	125.1667	125.17	125.1667	125.17	125.1667	125
5432.66	5432.6600	5,432.67	5,432.6666	5432.67	5,432.6665	5,432.67	5,432.6666	5,432.67	5,432.6666	5432
4.16	4.1600	4.17	4.1667	4.17	4.1667	4.17	4.1667	4.17	4.1667	4
5.50	5.5000	5.50	5.5000	5.50	5.5000	5.50	5.5000	5.50	5.5000	5
106.66	106.6600	106.67	106.6667	106.67	106.6667	106.67	106.6667	106.67	106.6667	106
7.22	7.2200	7.22	7.2222	7.22	7.2222	7.22	7.2222	7.22	7.2222	7
TOTAL										
5683.02	5683.0200	5,683.06	5,683.0555	5683.06	5,683.0554	5,683.06	5,683.0555	5,683.06	5,683.0555	5680

この例で、PACKED2 の合計は表示値の合計であり、格納値と一致します。PACKED4 の値と合計は、PACKED2 値と一致します。

DOUBLE2 の合計は、Windows では .04 の差異が生じています。これは、表示値の合計ではなく、格納値の合計を端数処理した値です。DOUBLE4 の値から、DOUBLE2 の値が格納値を端数処理して求められていることが分かります。DOUBLE4 の値および合計には、より多くの小数点以下の桁数が表示されています。DOUBLE2 の値は、この値を端数処理して求められています。

DOUBLE2 の合計と同様、FLOAT2 の合計は、FLOAT2 の格納値の合計を端数処理して求めらています。FLOAT4 列が示すように、F フィールドは 8 桁目以降が不正確です。

整数の合計は正確です。パック 10 進数フィールドと同様、格納値と表示値は同一です。

MATH2 と XMATH2 の合計では、.01 の差異が生じています。これらは表示値の合計ではなく、格納値の合計を端数処理して求めらています。MATH4 と XMATH4 の値から、MATH2 と XMATH2 の値が、格納値を端数処理して求められていることが分かります。MATH4 と XMATH4 の値および合計には、より多くの小数点以下の桁数が表示されています。MATH2 および XMATH2 の値は、この値を端数処理して求められています。

次のリクエストは、浮動小数点数と MATH データタイプとの相違を示しています。

```
DEFINE FILE ROUND1
DOUBLE20/D32.20=DOUBLE2;
MATH20/M32.20=MATH2;
END
TABLE FILE ROUND1
PRINT DOUBLE20 MATH20
ON TABLE SUMMARIZE
END
```

出力結果は、ほとんどの場合、倍精度浮動小数点数は入力値と正確には一致しないことを示しています。正確な 2 進法に対応していない値に対しては、余分な桁数が設定されています。MATH の値は、入力値を正確に表しています。

```
                          DOUBLE20                              MATH20
                          --------                              ------
             1.66666660000000010911                1.66666660000000000000
           125.16666600000000025238              125.16666600000000000000
         5,432.66659999999956198735            5,432.66660000000000000000
             4.16666660000000010911                4.16666660000000000000
             5.50000000000000000000                5.50000000000000000000
           106.66666000000000025238              106.66660000000000000000
             7.22222220000000003637                7.22222220000000000000

TOTAL
         5,683.05547539999770378927            5,683.05547540000000000000
```

一時項目 （DEFINE） および 一時項目 （COMPUTE）

一時項目 （DEFINE） および 一時項目 （COMPUTE）では、端数処理されたフィールドに対する結果が異なる場合があります。一時項目 (DEFINE) はデータソースフィールドと同等に処理される一方、一時項目 (COMPUTE) は、TABLE リクエストの表示コマンドの結果に対して計算を実行します。次の例は、この相違を示しています。

```
DEFINE FILE EXAMPLE
DEFP3/P9.3=PACKED2/4;
END
```

```
TABLE FILE EXAMPLE
PRINT PACKED2 DEFP3
COMPUTE COMPP3/P9.3=PACKED2/4;
ON TABLE SUMMARIZE
END
```

次のレポートが表示されます。

```
PAGE     1


 PACKED2      DEFP3     COMPP3
 -------      -----     ------
    1.67       .417       .417
  125.17     31.292     31.292
 5432.67   1358.167   1358.167
    4.17      1.042      1.042
    5.50      1.375      1.375
  106.67     26.667     26.667
    7.22      1.805      1.805

TOTAL
 5683.07   1420.765   1420.767
```

DEFP3 フィールドは、DEFINE の結果です。これらの値はデータソースフィールドの値と同等に処理されます。表示合計の 1420.765 は、DEFP3 表示値の合計です。これは、PACKED2 の合計が PACKED2 の値の合計であることと同様です。

COMPP3 フィールドは、COMPUTE の結果です。表示合計の 1420.767 は、PACKED2 の合計から計算されます (5683.07/4)。

SOME CLOUD SOFTWARE GROUP, INC. (“CLOUD SG”) SOFTWARE AND CLOUD SERVICES EMBED, BUNDLE, OR OTHERWISE INCLUDE OTHER SOFTWARE, INCLUDING OTHER CLOUD SG SOFTWARE (COLLECTIVELY, “INCLUDED SOFTWARE”). USE OF INCLUDED SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED CLOUD SG SOFTWARE AND/OR CLOUD SERVICES. THE INCLUDED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER CLOUD SG SOFTWARE AND/OR CLOUD SERVICES OR FOR ANY OTHER PURPOSE.

USE OF CLOUD SG SOFTWARE AND CLOUD SERVICES IS SUBJECT TO THE TERMS AND CONDITIONS OF AN AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER AGREEMENT WHICH IS DISPLAYED WHEN ACCESSING, DOWNLOADING, OR INSTALLING THE SOFTWARE OR CLOUD SERVICES (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH LICENSE AGREEMENT OR CLICKWRAP END USER AGREEMENT, THE LICENSE(S) LOCATED IN THE “LICENSE” FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE SAME TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

ibi, the ibi logo, ActiveMatrix BusinessWorks, BusinessConnect, Enterprise Message Service, FOCUS, Hawk, iWay, Maporama, Omni-Gen, Omni-HealthData, TIBCO, the TIBCO logo, the TIBCO O logo, TIBCO Administrator, TIBCO Designer, and WebFOCUS are either registered trademarks or trademarks of Cloud Software Group, Inc. in the United States and/or other countries.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only. You acknowledge that all rights to these third party marks are the exclusive property of their respective owners. Please refer to Cloud SG’s Third Party Trademark Notices (https://www.cloud.com/legal) for more information.

This document includes fonts that are licensed under the SIL Open Font License, Version 1.1, which is available at: https://scripts.sil.org/OFL.

Copyright (c) Paul D. Hunt, with Reserved Font Name Source Sans Pro and Source Code Pro.

Cloud SG software may be available on multiple operating systems. However, not all operating system platforms for a specific software version are released at the same time. See the “readme” file for the availability of a specific version of Cloud SG software on a specific operating system platform.