# SAS® 9.3 Scalable Performance Data Engine: リファレンス

SAS® ドキュメント

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2011. *SAS® 9.3 Scalable Performance Data Engine: Reference*. Cary, NC: SAS Institute Inc.

**SAS® 9.3 Scalable Performance Data Engine: Reference**



SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

Electronic book 1, 2011 July

SAS® Publishing provides a complete selection of books and electronic products to help customers use SAS software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit the SAS Publishing Web site at **support.sas.com/publishing** or call 1-800-727-3228.

# 目次

# このドキュメントについて

## SAS 言語の構文規則

### *SAS 言語の構文規則の概要*

SAS では、SAS 言語要素の構文ドキュメントに共通の規則を使用しています。これらの規則により、SAS 構文の構成要素を簡単に識別できます。規則は、次の項目に分類されます。

- 構文の構成要素
- スタイル規則
- 特殊文字
- SAS ライブラリと外部ファイルの参照

### *構文のコンポーネント*

言語要素の多くでは、その構文の構成要素はキーワードと引数から構成されます。キーワードのみ必要な言語要素もあります。また、キーワードに等号(=)が続く言語要素もあります。

キーワード
プログラムの作成ときに使用する SAS 言語要素名です。キーワードはリテラルであり、通常、構文の先頭の単語です。CALL ルーチンでは、最初の 2 つの単語がキーワードです。

次の SAS 構文の例では、構文の最初の単語がキーワードです。

**CHAR** (*string, position*)

**CALL RANBIN** (*seed, n, p, x*);

**ALTER** (*alter-password*)

**BEST** *w.*

**REMOVE** <*data-set-name*>

次の例では、CALL ルーチンの最初の 2 つの単語がキーワードです。

**CALL RANBIN**(*seed*, *n*, *p*, *x*)

引数なしで 1 つのキーワードから構成される SAS ステートメント構文もあります。

**DO**;
*... SAS code ...*
**END;**

2 つのキーワード値のいずれか 1 つの指定が必要なシステムオプションもあります。

**DUPLEX** | **NODUPLEX**

引数
数値定数、文字定数、変数、式のいずれかです。引数は、キーワードに続くか、キーワードの後ろの等号に続きます。SAS では、引数を使用して、言語要素を処理します。引数が必須の場合もオプションの場合もあります。構文では、オプションの引数にはかぎかっこが付けられます。

次の例では、*string* と *position* がキーワード CHAR に続きます。これらの引数は、CHAR 関数の必須引数です。

**CHAR** (*string, position*)

引数ごとに値が指定されます。次の例の SAS コードでは、引数 *string* の値として 'summer'、引数 *position* の値として 4 が指定されています。`x=char('summer', 4);`

次の例では、*string* と *substring* は必須引数ですが、*modifiers* と *startpos* はオプションの引数です。

**FIND**(*string, substring* <,*modifiers*> <,*startpos*>

*注:* 通常、SAS ドキュメントのサンプルコードは、小文字の固定幅フォントを使用して表記されます。コードの作成には、大文字も、小文字も、大文字と小文字の両方も使用できます。

## *スタイル規則*

SAS 構文の説明に使用されるスタイル規則には、大文字太字、大文字、斜体の規則も含まれます。

大文字太字
関数名やステートメント名などの SAS キーワードを示します。次の例では、キーワード ERROR の表記には大文字太字が使用されています。

**ERROR**<*message*>;

大文字
リテラルの引数を示します。

次の CMPMODEL=システムオプションの例では、BOTH、CATALOG、XML がリテラルです。

**CMPMODEL** = BOTH | CATALOG | XML

*斜体*
ユーザー指定の引数または値を示します。斜体表記の項目は、ユーザー指定値であり、次のいずれかを表します。

- 非リテラルの引数。次の LINK ステートメントの例では、引数 *label* はユーザー指定値であるため、斜体で表記されています。

  **LINK** *label*;

- 引数に割り当てられる非リテラル値。

  次の FORMAT ステートメントの例では、引数 DEFAULT に変数の *default-format* が割り当てられます。

  **FORMAT** = *variable-1* <, ..., *variable-n format* ><DEFAULT = *default-format*>;

斜体表記の項目は、選択可能な引数リストの総称でもあります(*attribute-list* など)。複数の斜体表記の項目が使用される場合、項目は *item-1, ..., item-n* という形式で表記されます。

## *特殊文字*

SAS 言語要素の構文には、次の特殊文字も使用されます。

=
等号は、一部の言語要素(システムオプションなど)のリテラル値を示します。

次の MAPS システムオプションの例では、等号は MAPS の値を設定します。

**MAPS** = *location-of-maps*

< >
かぎかっこはオプションの引数を示します。かぎかっこ付きでない引数は必須引数です。

次の CAT 関数の例では、少なくとも項目が 1 つ必要です。

**CAT** (*item-1 <, ..., item-n>*)

|
縦棒は、値グループから 1 つの値を選択できることを示します。縦棒で区切られている値は、相互排他です。

次の CMPMODEL=システムオプションの例では、属性を 1 つのみ選択できます。

**CMPMODEL** = BOTH | CATALOG | XML

...
省略記号は、省略記号に続く引数や引数グループの繰り返しを示します。省略記号とその後の引数にかぎかっこが付けられている場合、その引数はオプションです。

次の CAT 関数の例では、省略記号はオプションの項目を複数指定できることを示しています。

**CAT** (*item-1 <, ..., item-n>*)

'*value*' or "*value*"
単一引用符や二重引用符付きの引数は、その値も単一引用符または二重引用符を付ける必要があることを示します。

次の FOOTNOTE ステートメントの例では、引数 *text* には引用符が付けられています。

**FOOTNOTE** <*n*> <*ods-format-options* '*text*' | "*text*">;

;
セミコロンは、ステートメントまたは CALL ルーチンの終わりを示します。

次の例では、それぞれのステートメントはセミコロンで終了しています。
```
data namegame; length color name $8; color = 'black'; name = 'jack'; game = trim(color) || name; run;
```

## *SAS ライブラリと外部ファイルへの参照*

多くの SAS ステートメントなどの言語要素では、SAS ライブラリと外部ファイルを参照します。論理名(ライブラリ参照名またはファイル参照名)から参照を作成するのか、引用符付きの物理ファイル名を使用するかを選択できます。論理名を使用する場合、通

常、関連付けに SAS ステートメント(LIBNAME または FILENAME)を使用するのか、動作環境のコントロール言語を使用するのかを選択します。複数の方法を使用して、SAS ライブラリと外部ファイルを参照できます。動作環境によっては使用できない方法があります。

SAS ドキュメントでは、外部ファイルを使用する例には斜体のフレーズ *file-specification* を使用します。また、SAS ライブラリを使用する例には斜体フレーズ *SAS-library* を使用します。*SAS-library* は引用符付きであることに注意してください。

```
infile file-specification obs = 100;
libname libref 'SAS-library';
```

# SAS 9.3 Scalable Performance Data Engine の新機能

## 概要

9.3 では、次の機能が追加され、拡張されました。

- SPD Engine ファイルのバックアップに関するセクションが追加されました。“SPD Engine ファイルのバックアップ” (23 ページ)を参照してください。

## SPD Engine システムオプション

VALIDMEMNAME=EXTEND と ALIDVARNAME=の動作は、SPD Engine と Base SAS Engine では異なります。詳細については、VALIDMEMNAME=EXTEND (83 ページ)と VALIDVARNAME= (83 ページ)を参照してください。

# 推奨資料

- *Base SAS プロシジャガイド*
- *SAS 言語リファレンス: 解説編*
- *SAS システムオプション: リファレンス*
- *SAS データセットオプション: リファレンス*

SAS の刊行物の総一覧については、support.sas.com/bookstore にてご確認ください。
必要な書籍についてのご質問は、下記までお寄せください。

SAS Books
SAS Campus Drive
Cary, NC 27513-2414
電話: 1-800-727-3228
ファクシミリ: 1-919-677-8166
電子メール: sasbook@sas.com
Web アドレス: support.sas.com/bookstore

*1 章*

# 概要: SPD Engine



## SPD Engine について

SPD Engine は、ハイパフォーマンスデータ配信向けに設計されています。これにより、アプリケーションで処理する SAS データに迅速にアクセスできるようになります。SPD Engine では、データが、効率化されたファイル形式に編成され、複数の CPU を利用し

て並列入出力機能が実行されるので、データがアプリケーションに迅速に配信されます。

SPD Engine では、*スレッド*を使用して、非常に迅速に並列処理でデータブロックが読み取られます。ソフトウェアタスクはオペレーティングシステムに関連して実行されます。このため、コンピュータの任意の使用可能 CPU でスレッドを実行できます。スレッド I/O は SPD Engine 機能の重要部分ですが、SPD Engine の真価は、ソフトウェアでの SAS データの構造化方法にあります。SPD Engine では、データが新しいファイル形式に編成されます。これにはデータの分割も含まれます。このデータ構造ではスレッド(並列実行)が許可され、I/O タスクが効率的に実行されます。

デフォルト Base SAS Engine との置換を意図したわけではありませんが、SPD Engine は、非常に大きなデータセットの処理では高速の代替方法となります。何百万ものオブザベーションを含むデータセットの読み書きが行われます。たとえば、これには、一部のオペレーティングシステムで課される 2GB のサイズ制限を超えて拡張するデータセット、ならびに SAS 分析ソフトウェアおよびプロシジャでより迅速な処理が必要なデータセットが含まれます。

SPD Engine のパフォーマンスは次の方法で向上させられます。

- GB 単位のデータのサポート
- 対称型マルチプロセッサ(SMP)コンピュータのスケーラビリティ
- 並列処理での WHERE 選択
- 並列処理でのロード
- 並列処理でのインデックス作成
- 並列処理での I/O データのアプリケーションへの配信
- BY ステートメントでの自動並べ替え

SPD Engine は、UNIX、Windows、z/OS (zFS ファイルシステムのみ)、HP Integrity サーバーの OpenVMS (ODS-5 ファイルシステムのみ)で実行します。

*注:* SAS 9.3 でのスケーラビリティの詳細については、**`http://support.sas.com/rnd/scalability`** の Scalability and Performance フォーカス領域を参照してください。システム要件については、**`http://support.sas.com/documentation/installcenter`** の Install Center を参照してください。

## SPD Engine 互換性

SAS 9.1.3 から SAS 9.3 にアップグレードする場合、動作環境が同一のままであれば、データセットを移行する必要はありません。複数のホストにわたるアップグレードの場合(例: 32 ビットから 64 ビットの Windows 動作環境へ)、データセットを移行する必要があります。データセットを移行するには、CIMPORT、COPY または CPORT を使用します。詳細については、*Base SAS プロシジャガイド*を参照してください。

移行については、**`http://support.sas.com/rnd/migration`** の Migration フォーカス領域を参照してください。

*注:* MIGRATE プロシジャでは、SPD Engine はサポートされていません。

# SMP コンピュータの使用

SPD Engine では、対称型マルチプロセッシングと呼ばれるハードウェアとソフトウェアのアーキテクチャが活用されます。SMP コンピュータには、複数の中央処理装置(CPU)、およびスレッドをサポートするオペレーティングシステムがあります。SMP コンピュータは通常、複数のコントローラ、およびコントローラごとの複数のディスクドライブで構成されます。SPD Engine では、データファイルの読み取り時に、各 CPU に対して 1 つ以上のスレッドが起動されます。これらのスレッドは、CPU ごとに 1 つ以上のコントローラによって駆動され、複数のディスクドライブから並列処理でデータを読み取ります。SMP コンピュータで実行する SPD Engine では、所定の経過時間内ではるかに多くのデータを読み取り、アプリケーションに配信する機能が提供されます。

5 つの CPU と 10 のディスクドライブを備えた SMP コンピュータでデータセットを読み取ると、シングル CPU コンピュータの I/O よりも 5 倍速く処理することが可能になります。スレッド I/O に加えて、SMP コンピュータでは、アプリケーションプロセスのスレッド化が可能になります(たとえば、SAS 9.1 以降の SORT プロシジャでのスレッド化された並べ替えなど)。

SMP コンピュータの正確な CPU 数は、製造元とモデルによって変わります。コンピュータのオペレーティングシステムも専門化され、並列実行のためにコードセグメントをスケジューリングする機能が必要とされます。オペレーティングシステムカーネルがスレッド化されると、実行スレッド間の競合が防止されるため、パフォーマンスがさらに向上します。

スレッドオペレーティングシステムで管理された SMP コンピュータ上でスレッドを実行すると、使用可能な CPU が同時に作動します。CPU とスレッドの相乗効果によって、ソフトウェアで、処理パフォーマンスのスケール変更が可能になります。その結果、スケーラビリティによって、データセットの作成、データの追加、および WHERE ステートメントを使用したデータのクエリなどのタスクの全体処理速度が大幅に増加します。

# SPD Engine を使用した SAS データの編成

### *SPD Engine の SAS データの編成方法*

SPD Engine でハイパフォーマンス処理のためにデータが編成されるので、SPD Engine データセットはデフォルト Base SAS Engine データセットと物理的に異なります。デフォルト Base SAS Engine では、ファイルのデータとデータディスクリプタ(メタデータ)の両方が含まれる単一データファイルにデータが格納されます。SPD Engine では、データとデータディスクリプタに対して別々のファイルが作成されます。さらに、データセットにインデックスが付いている場合、インデックスごとに 2 つずつインデックスファイルが作成されます。これら 4 種類のファイルのそれぞれが SPD Engine *コンポーネントファイル*と呼ばれ、それぞれの識別ファイル拡張子が存在します。

さらに、これらのコンポーネントのそれぞれを 1 つ以上の物理ファイルで構成し、コンポーネントが複数のボリュームにわたっていても 1 つの論理ファイルとして参照するようにできます。たとえば、SPD Engine では、データを含む物理ファイルを多数作成できますが、データを含むファイルを、SPD Engine データセットの単一データコンポーネントとして参照できます。*メタデータ*コンポーネントとインデックスコンポーネントは、次の 2 点がデータコンポーネントと異なります。

1. PARTSIZE=オプションを使用してデータコンポーネントファイルの固定長パーティションサイズを指定できます。ただし、メタデータパーティションまたはインデックスパーティションのサイズについては、ほとんどまたはまったく制御できません。
2. データコンポーネントファイルは、すべての定義パスにわたって周期的に作成されます。メタデータコンポーネントとインデックスコンポーネントは、シングルパスで、そのパスがいっぱいになるまで作成され、その後で次のパスが使用されます。

### *メタデータコンポーネントファイル*

SPD Engine データセットでは、説明的なメタデータが、ファイル拡張子.mdf のファイルに格納されます。通常、SPD Engine データセットの.mdf ファイルは 1 つだけです。

### *インデックスコンポーネントファイル*

ファイルにインデックスが付いている場合、SPD Engine では、インデックスごとに 2 つずつインデックスコンポーネントファイルが作成されます。これらの各ファイルにインデックスの特定ビューが含まれるので、各データセットに両方が存在します。

- .hbx ファイル拡張子の付いたインデックスファイルには、グローバルインデックスが含まれます。
- .idx ファイル拡張子の付いたインデックスファイルには、セグメントインデックスが含まれます。

### *データコンポーネントファイル*

SPD Engine データセットのデータコンポーネントでは、パスまたはデバイスごとにファイル(パーティション)が 1 つだけではなく複数存在する可能性があります。これらの各パーティションは固定長で、その長さは SPD Engine データセットの作成時に指定されます。

データコンポーネントファイルのサイズ指定によって、アプリケーションのパフォーマンスを調整できます。パーティションとはスレッド化可能単位です。すなわち、各パーティション(ファイル)は 1 スレッドで読み取られます。“SPD Engine ファイルの作成とロード” (11 ページ) では、SPD Engine でのデータ、メタデータおよびインデックスの格納に関する詳細を提供します。

## デフォルト Base SAS Engine と SPD Engine の比較

### *比較の概要*

デフォルト Base SAS Engine データセットと SPD Engine データセットには多くの類似点があります。両方とも SAS ライブラリにデータを格納します。この SAS ライブラリとは、1 つ以上のディレクトリに存在するファイルのコレクションです。ただし、SPD Engine データライブラリは複数のデバイスやファイルシステムにまたがるので、SPD Engine は非常に大きなデータセットの使用に最適です。また、SPD Engine では、LIBNAME ステートメントでコンポーネントごとに別々のディレクトリ(またはデバイス)を指定できます。“SPD Engine ファイルの作成とロード” (11 ページ) では、SPD Engine データライブラリの設計および設定に関する詳細を提供します。

### SPD Engine ライブラリとファイルシステム

SPD Engine ライブラリには、データファイル、メタデータファイルおよびインデックスファイルを含められます。SPD Engine では、カタログ、SAS ビュー、MDDB、その他のユーティリティ(バイト)ファイルはサポートされません。

SPD Engine では、z/OS 用 zFS ファイルシステム、および HP Integrity サーバーの OpenVMS 用 ODS-5 ファイルシステムが使用されます。これはつまり、これらのプラットフォームで一部の機能が多少異なる可能性があることを意味します。たとえば、z/OS の場合、ユーザーは zFS 上にホームディレクトリが必要です。

### ユーティリティファイルワークスペース

ユーティリティファイルは、追加領域を必要とする SPD Engine 操作時(並列インデックスの作成時や、非常に大きなファイルの並べ替え時など)に生成されます。すべてのプラットフォームにデフォルトの場所が存在しますが、大量の処理データがある場合、デフォルトの場所の容量が不十分になる可能性があります。SPD Engine システムオプション SPDEUTILLOC=では、ユーティリティスクラッチファイルを格納する一連のファイル場所を指定できます。詳細については、“SPDEUTILLOC=システムオプション” (89 ページ)を参照してください。

### 中間データセットの一時ストレージ

中間データセットを格納するライブラリを作成するには、LIBNAME ステートメントに SPD Engine オプション TEMP=を指定します。現在のアプリケーションで 1 レベル名を使用してこれらの中間ファイルを参照する場合は、USER=システムオプションでライブラリを指定します。

次のコード例では、中間データセットのユーザーライブラリ参照名が作成されます。これはセッションの終了時に削除されます。

```
libname user spde '/mydata' temp=yes;
data a; x=1;
run;
proc print data=a;
```

構成ファイルで USER=オプションを設定し、1 レベル名で中間データセットを参照するアプリケーションを SPD Engine で実行するようにできます。

### デフォルト Base SAS Engine データセットと SPD Engine データセットの相違

次の表は、SPD Engine 機能とデフォルト Base SAS Engine 機能の比較です。

**表 1.1** デフォルト Base SAS Engine データセットと SPD Engine データセットの比較

| 機能 | SPD Engine | デフォルト Base SAS Engine |
|---|---|---|
| 分割データセット | あり | なし |
| 並列処理での WHERE 最適化 | あり | なし |
| 最下位ロックレベル | メンバ | レコード |

| 機能 | SPD Engine | デフォルト Base SAS Engine |
|---|---|---|
| 指定データセットでの複数 SAS セッションからの同時アクセス | READ (INPUT オープンモード) | READ と WRITE (すべてオープンモード) |
| SAS/CONNECT 経由のリモートコンピューティング | なし | あり |
| SAS/CONNECT 経由のデータ転送 | なし | あり |
| SAS/CONNECT 経由の RLS (リモートライブラリサービス) | なし | あり |
| SAS/CONNECT 経由の使用可能性 | なし | あり |
| SAS/SHARE でのサポート | なし | あり |
| SAS BY 処理のための自動並べ替え (BY 処理をサポートするためにデータの一時コピーを並べ替える) | あり | なし |
| ユーザー定義の出力形式と入力形式 | あり(WHERE 以外)[1] | あり |
| カタログ | なし | あり |
| ビュー | なし | あり |
| MDDB | なし | あり |
| 一貫性制約 | なし | あり |
| データセット世代 | なし | あり |
| CEDA | なし | あり |
| 監査証跡 | なし | あり |
| NLS トランスコーディング | なし | あり |
| カウント可能なオブザベーション数 | $2^{63}$-1 (すべてのホスト) | $2^{31}$-1 (32 ビットホスト)$2^{63}$-1 (64 ビットホスト) |
| COMPRESS= | YES\|NO\|CHAR\|BINARY (ファイルが暗号化されていない場合のみ) | YES\|NO\|CHAR\|BINARY |
| DLCREATEDIR | なし | あり |
| ENCRYPT= | COMPRESS=と一緒には使用不可 | COMPRESS=と一緒に使用可能 |
| 暗号化 | データファイルのみ | あり(すべてのファイル) |

1 WHERE 処理では、ユーザー定義の出力形式と入力形式はスーパーバイザに渡されて処理されます。したがって、並列では処理されません。

| 機能 | SPD Engine | デフォルト Base SAS Engine |
|---|---|---|
| FIRSTOBS=システムオプションおよびデータセットオプション | なし | あり |
| OBS=システムオプションおよびデータセットオプション | あり(ENDOBS=または STARTOBS= SPD Engine オプションなしで使用する場合) | あり |
| 関数とコールルーチン | あり(ただし一部例外あり)[1] | あり |
| テーブルを OS ユーティリティによって異なるディレクトリまたはフォルダに移動 | なし | あり |
| 物理的な順序で返されるオブザベーション | なし(BY または WHERE が存在する場合) | あり |
| DLDMGACTION=システムオプションおよびデータセットオプション | なし [2] | あり |

# デフォルト Base SAS Engine データセットと SPD Engine データセットの相互運用性

デフォルト Base SAS Engine データセットは、SPD Engine でアクセスできるように、SPD Engine 形式に変換する必要があります。デフォルト Base SAS Engine データセットは、COPY プロシジャ、APPEND プロシジャまたは DATA ステップを使用すると容易に変換できます。(PROC MIGRATE は使用できません。)さらに、既存 SAS プログラムの大部分は、LIBNAME ステートメント以外、ほとんど修正せずに、SPD Engine ファイルに対して実行できます。“SPD Engine ファイルの作成とロード” (11 ページ) では、デフォルト Base SAS Engine データセットから SPD Engine 形式への変換に関する詳細を提供します。

# SPD Engine ファイルの共有

SPD Engine では、メンバレベルのロックがサポートされます。つまり、複数のユーザーが INPUT で同じ SPD Engine データセットを開くことができます(読み取り専用)。ただし、更新のために SPD Engine データセットを開いた場合は、そのユーザーのみアクセス可能になります。

1 WHERE 処理では、SAS 9 以降に導入された関数とコールルーチンはスーパーバイザに渡されて処理されます。したがって、並列では処理されません。

2 パーティションデータファイルまたはメタデータファイルの損傷は検出されません。ファイルは修復できません。インデックスファイルの損傷は検出されます。インデックスは、PROC DATASETS の REPAIR ステートメントで修復される場合があります。

# I/O のパフォーマンスを向上させる機能

### *I/O パフォーマンス向上の概要*

SPD Engine には、I/O パフォーマンスを向上させる新機能がいくつかあります。これらの機能によって I/O バウンドアプリケーションのパフォーマンスを劇的に高められます。このとき、大量のデータはアプリケーションに配信して処理する必要があります。

### *複数のディレクトリパス*

SPD Engine ではいくつかのボリュームにわたる複数の物理ファイルを単一の論理ファイルとして参照できるので、コンポーネントの種類ごとに複数のディレクトリパスとデバイスを指定できます。非常に大きなデータセットの場合、この機能によって、オペレーティングシステムで課される可能性があるファイルサイズ制限をすべて回避できます。

### *データファイルと関連インデックスの物理的分離*

コンポーネントファイルの各種類はそれぞれ異なる場所に格納できるので、コンポーネントファイルの格納場所の決定時にファイル依存関係を考慮する必要はありません。考慮が必要なのは、コスト、パフォーマンス、およびディスク領域の可用性のみです。

### *WHERE の最適化*

SPD Engine では、WHERE ステートメントで指定された基準を満たすためにオブザベーションを評価するのに最適なプロセスが自動的に決定されます。WHERE ステートメントの効率性は、式の変数にインデックスを付けるかどうかなどの因子によって変わります。SPD Engine に含まれる WHERE 評価プランナでは、WHERE 式評価のための最適な方法を選択できます。その方法では、インデックスを使用して評価を最適化します。

# 処理パフォーマンスを向上させる機能

### *自動的な並べ替え機能*

SPD Engine の自動並べ替え機能によって、大きなデータセットを処理する SAS アプリケーションの時間とリソースが節約されます。SPD Engine では、BY 句を指定した SAS ステートメントをサブミットする前に SORT プロシジャを呼び出す必要はありません。SPD Engine で BY 句が検出される際、データはまだ並べ替えられておらず、BY 変数のインデックスも付いていません。SPD Engine による自動的な並べ替えでは、永久データセットに影響は及ばず、新しい出力データセットも作成されません。

### *インデックスを使用したクエリ*

パフォーマンスを最大にするには大きなデータセットにインデックスを付けます。インデックスによって、インデックス付き変数に対する迅速な WHERE 式評価が可能になります。SPD Engine は、複数の CPU を利用して、インデックスコンポーネントファイルを効率的に検索します。

*注:* 変数名に次の特殊文字のいずれかが含まれている場合は、変数に対して単一インデックスも複合インデックスも作成できません。

```
" * | \ : / < > ? -
```

### *並列処理でのインデックス作成*

さらに、SPD Engine では、大きなデータセットのインデックス付けに時間がかからないように、並列インデックス作成がサポートされています。SPD Engine では、データセットの追加操作または挿入操作が、並列実行の可能な一連のステップに分解されます。並列処理のレベルは、データセットに存在するインデックスの数によって決まります。インデックスが多いほど、インデックス作成時に並列処理が多く活用されます。ただし、インデックスの作成には、ユーティリティのファイル領域とメモリリソースが必要です。

*注:* 変数名に次の特殊文字のいずれかが含まれている場合は、変数に対して単一インデックスも複合インデックスも作成できません。

```
" * | \ : / < > ? -
```

## SPD Engine のオプション

SPD Engine では、多くのデフォルト Base SAS Engine オプションが取り扱われます。それに加えて、SPD Engine でのみ使用されるオプションもあります。そのオプションを使用すると、SPD Engine のライブラリと処理の管理が促進されます。次を参照してください。

- SPD Engine データセットオプション (41 ページ)
- SPD Engine LIBNAME ステートメントオプション (25 ページ)
- SPD Engine システムオプション (81 ページ)

*2 章*

# SPD Engine ファイルの作成とロード



## SPD Engine ファイルの作成とロードについて

このセクションでは、SPD Engine ライブラリの割り当て、ならびに SPD Engine のデータとインデックスの作成およびロードについて詳細を提供します。また、これらのタスクに関連するパフォーマンスの考慮事項についても説明します。

# ライブラリ領域の割り当て

## *ライブラリ領域の割り当て方法*

SPD Engine の分割データ I/O およびスレッド機能によるパフォーマンスの向上を実現するには、SPD Engine ライブラリの構成および管理を正しく行う必要があります。SAS System 管理者はこれらのタスクを最適な形で実行します。

SPD Engine データセットでは、各種コンポーネントの格納に十分な領域を有するファイルシステムが必要です。多くの場合、そのファイルシステムには、それらのコンポーネントに対する複数のディレクトリが含まれます。通常、シングルディレクトリパス(ファイルシステムの一部)は、ファイルシステム全体のボリューム制限に制約されます。この制限とは、ファイルシステム用に構成されたディスク領域の最大容量です。

この最大ディスク領域の範囲内で、SPD Engine コンポーネントファイルすべてにとって十分な領域を割り当てる必要があります。各ライブラリに必要な記憶容量を見積もるには、各コンポーネントファイルの処理方法を理解しておくことが重要です。

## *シングルパスのすべてのコンポーネントの領域の構成*

最も単純な SPD Engine ライブラリ構成では、すべての SPD Engine コンポーネントファイル(データファイル、メタデータファイルおよびインデックスファイル)が、*プライマリパス*というシングルパスに存在します。プライマリパスは、LIBNAME ステートメントのデフォルトパス指定です。次の LIBNAME ステートメントでは、MYLIB ライブラリに対してプライマリファイルシステムが設定されます。

```
libname mylib spde '/disk1/spdedata';
```

他のパスオプションが指定されていないため、すべてのコンポーネントファイルがこのプライマリパスに作成されます。すべての種類のコンポーネントファイルをプライマリパスに格納するのは、簡単で、非常に小容量のデータセットでは効果的です。ただし、コンポーネントを別々に格納することによって達成可能なパフォーマンスの向上や、複数の CPU という利点は得られません。

*注:* SPDEngine では、完全パス名を指定する必要があります。

## *コンポーネントファイル別のライブラリ領域の構成*

ほとんどのサイトでは、SPD Engine は非常に大容量のデータの管理に使用されます。データには何千もの変数を含められ、その一部にはインデックスを付けられます。それらのサイトでは通常、各種コンポーネントに対して別々の記憶域パスが定義されます。さらに、ディスクストライピングおよび RAID (Redundant Array of Independent Disks)を使用すると、非常に効率的になる可能性があります。詳細については、**`http://support.sas.com/rnd/scalability/spde/setup.html`** で Scalability and Performance の“SPD Engine Disk I/O Setup”を参照してください。

メタデータコンポーネントファイルはすべて、複数のデバイスにまたがる場合でも、プライマリパスで始まる必要があります。さらに、データコンポーネントとインデックスコンポーネントに別々のパスを指定すると、いっそうパフォーマンスが向上します。異なるパスを指定するのは、I/O 負荷が複数のディスクドライブに分散するためです。データコンポーネントとインデックスコンポーネントを分けておくと、複雑な WHERE 評価では特に、ディスクの競合を防ぎ、達成可能な*並列処理*のレベルを上げるのに役立ちます。次のコード例では、メタデータのプライマリパスが指定されます。コードでは、

DATAPATH= (30 ページ) および INDEXPATH= (34 ページ) を使用して、データコンポーネントファイルとインデックスコンポーネントファイルに対して別々のパスが追加指定されます。

```
libname all_users spde '/disk1/metadata'
datapath= ('/disk2/userdata' '/disk3/userdata')
indexpath= ('/disk4/userindexes' '/disk5/userindexes');
```

メタデータは、プライマリパスの disk1 に格納されます。データは disk2 および disk3、インデックスは disk4 および disk5 です。すべてのパス指定において、完全パス名を指定する必要があります。

***注意:***

**プライマリパスは各ライブラリで重複しないようにする必要があります。**プライマリパスが同じで、その他のパスに相違があるライブラリ参照名を 2 つ作成した場合、データが失われる可能性があります。NFS はプライマリパス以外のパスオプションでは使用できません。

*注:* ローカルでマウントされたドライブにデータを格納し、リモートコンピュータからデータにアクセスする場合は、LIBNAME の指定時にリモートパス名を使用します。**/data01** および **/data02** が localA コンピュータにローカルでマウントされたドライブである場合は、LIBNAME ステートメントでパス名 **/nfs/localA/data01** および **/nfs/localA/data02** を使用します。

*注:* ファイルのパス名は変更できません。DATAPATH=、INDEXPATH=、METAPATH=、プライマリパス LIBNAME オプションのいずれかを指定すると、データセットの作成時に使用された同一パスがデータセットにアクセスするたびに使用されるようになります。これらの場所のパス名はデータセットの内部に格納されます。パス名を一部でも変更すると、SPD Engine でデータセットを検索できなくなったり、データセットが破損したりする可能性があります。

## *コンポーネントファイル別の領域の予想*

SPD Engine ライブラリ領域を正しく構成するには、SPD Engine コンポーネントファイルの相対サイズについて理解する必要があります。次の情報は概要を示しています。詳細については、**http://support.sas.com/rnd/scalability/spde/setup.html** で Scalability and Performance の“SPD Engine Disk I/O Setup”を参照してください。

メタデータコンポーネントファイルは相対的に小さいファイルなので、プライマリパスが、ライブラリのメタデータファイルをすべて含めるのに十分な大きさである場合があります。

インデックスコンポーネントファイル(.idx と.hbx の両方)は、各インデックスの個別の値数、およびインデックスが単一インデックスか複合インデックスかに応じて、中容量から大容量になります。インデックスコンポーネントファイルが、現在のファイルパスで使用可能な領域よりも大きくなると、次のパスに新しいコンポーネントファイルが作成されます。

データコンポーネントファイルは、データ容量と、データセットに指定するパーティションサイズによっては、非常に多数に及ぶ可能性があります。各データパーティションが別々のデータコンポーネントファイルとして格納されます。データパーティションのサイズは “PARTSIZE= LIBNAME ステートメントオプション” (36 ページ)で指定されます。パーティションサイズを指定できるコンポーネントファイルはデータファイルのみです。

## メタデータコンポーネントファイルの記憶域

### メタデータコンポーネントファイル

SPD Engine で分割データの読み取りや書き込みを効率的に行うために必要な情報の多くは、メタデータコンポーネントに格納されます。SPD Engine ではそのメタデータへの迅速なアクセスを可能にする必要があります。設計上、SPD Engine ではすべてのデータセットのメタデータコンポーネントがプライマリパスで始まることが予想されます。これらのメタデータコンポーネントファイルは、他のパス(“METAPATH= LIBNAME ステートメントオプション” (35 ページ)で指定)にオーバーフローする可能性がありますが、常にプライマリパスで始める必要があります。この概念を理解しておくことは非常に重要です。これはデータセット(および関連するメタデータファイル)をライブラリに追加可能かどうかに直接影響するからです。

たとえば、ライブラリに対して新しいデータセットが作成され、プライマリパスの領域がいっぱいになったとします。SPD Engine では、要求に応じてそのプライマリパスでメタデータコンポーネントファイルを始めることができません。作成操作は失敗し、適切なエラーメッセージが表示されます。この場合、新しいデータセットを正常に作成するには、プライマリパスの領域を解放するか、または新しいライブラリを割り当てる必要があります。METAPATH=オプションでは、新しいデータセットの最初のメタデータパーティションの領域を作成することはできません。METAPATH=で指定できるのは、プライマリパスで始まるけれども、拡張されて、プライマリパスの予約領域が埋まってしまったメタデータコンポーネント用のオーバーフロー領域のみです。メタデータコンポーネントは拡張されてファイルサイズやライブラリ領域の制限を超過する場合があります。プライマリパスに追加データセット用の領域を確保するには、ライブラリの最初の作成時に、METAPATH=オプションでメタデータのオーバーフローパスを指定します。

初期の LIBNAME ステートメントで別のパスを指定した場合でも、データコンポーネントファイルとインデックスコンポーネントファイルに対して後から追加領域を指定できます。

### パスの初期セット

次の例では、LIBNAME ステートメントで、プライマリパスとして MYLIB ディレクトリが指定されます。デフォルトでは、このパスが初期メタデータパーティションの格納に使用されます。データパーティションおよびインデックスパーティションの格納用に他のデバイスおよびディレクトリが指定されます。

```
libname myref spde 'mylib'
datapath=('/mydisk30/siteuser')
indexpath=('/mydisk31/siteuser');
```

### 後続パスの追加

後からさらに領域が必要になった場合は(大容量データの追加など)、次の例のように、追加デバイスがデータおよびインデックスに追加されます。

```
libname myref spde 'mylib'
datapath=('/mydisk30/siteuser' '/mydisk32/siteuser' '/mydisk33/siteuser')
indexpath=('/mydisk31/siteuser' '/mydisk34/siteuser');
```

### インデックスコンポーネントファイルの記憶域

インデックスコンポーネントファイルはオーバーフロー領域に基づいて格納されます。INDEXPATH=オプションで複数のファイルパスが指定されます。インデックスファイルは最初の使用可能な領域で開始され、前の領域が埋まると、次のファイルパスにオーバーフローします。メタデータコンポーネントとは違って、インデックスコンポーネントファイルはプライマリパスで始まる必要はありません。

### *データパーティションの記憶域*

サイズを指定できるファイルはデータコンポーネントパーティションのみです。分割データはスレッドで容易に処理できるので、コンピュータの複数 CPU が最大限に活用されます。

データコンポーネントのパーティションサイズは固定で、データセットの作成時に設定されます。デフォルトは 128MB ですが、PARTSIZE=オプションを使用して異なるパーティションサイズを指定できます。パフォーマンスは適切なパーティションサイズによって決まります。このためデータのサイズと用途を理解しておく必要があります。バランスの取れたオブザベーション数をもたらすパーティションサイズで SPD Engine データセットを作成できます。(詳細については、“PARTSIZE=データセットオプション” (67 ページ)を参照してください。)

指定データセットの各データパスで多くのデータパーティションを作成できます。SPD Engine では、DATAPATH=オプションで指定したファイルパスを使用して、パーティションが周期的に分散されます。SPD Engine では、指定パスの 1 つに最初のデータパーティション、次のパスに 2 番目のパーティションというように、順番に作成されます。このソフトウェアでは、データセットのすべてのデータパーティションが格納されるまで、必要な回数だけ、ファイルパス内で処理が循環します。最初のパーティションとして選択されるパスは、ランダムに選択されます。

LIBNAME ステートメントで次のように指定したとします。

```
datapath=('/data1' '/data2')
```

SPD Engine では、/DATA1 に最初のパーティション、/DATA2 に 2 番目のパーティション、/DATA1 に 3 番目のパーティション、というように格納されていきます。データパーティションの周期的分散によってディスクストライピングが行われます。これにより非常に効率がよくなる可能性があります。ディスクストライピングの詳細については、**`http://support.sas.com/rnd/scalability/spde/setup.html`** で Scalability and Performance の“SPD Engine Disk I/O Setup”を参照してください。

### *コンポーネントファイルの名前変更、コピー、移動*

***注意:***

**SPD Engine のデータセットやそのコンポーネントファイルの名前変更、コピー、移動にはオペレーティングシステムコマンドを使用しないでください。**

ある場所から別の場所に SPD Engine データセットをコピーするには COPY プロシジャ、SPD Engine データセットの名前変更や削除には DATASETS プロシジャを常に使用してください。

---

## ディスクストライピングと大容量ディスクアレイを使用した場合の効率

システム内にあるファイル作成ユーティリティで、ファイルシステムの制限を上書きし、単一のディスク上の領域より大きいファイルシステム(ボリューム)を作成できる場合があります。RAID などの複数のディスクデバイスにまたがる SPD Engine ライブラリを割り当てるには、このユーティリティを使用します。RAID 構成では、ディスクストライピングという手法によって、I/O が大幅に拡張されます。ディスクストライピングおよび RAID の詳細について
は、**`http://support.sas.com/rnd/scalability/spde/setup.html`** で Scalability and Performance の“SPD Engine Disk I/O Setup”を参照してください。

# デフォルト Base SAS Engine データセットから SPD Engine データセットへの変換

## COPY プロシジャと APPEND プロシジャの使用

既存のデフォルト Base SAS Engine データセットを SPD Engine データセットに変換するには、次のメソッドを使用します。

- PROC COPY
- PROC APPEND

いくつかの制限が適用されます。デフォルト Base SAS Engine データに一貫性制約がある場合は、ファイルが SPD Engine 形式で作成されるときに、その一貫性制約は解除されます。次のファイル特性表は、特性の保持や解除が行われる可能性があるかどうか、または結果的に変換時にエラーが発生するかどうかを示しています。

*表 2.1 Base SAS ファイル特性の変換結果*

| Base SAS ファイル特性 | 変換結果 |
|---|---|
| インデックス | SPD Engine で再作成 (ASYNCINDEX=YES の場合は並列処理) |
| デフォルト Base SAS Engine COMPRESS=YES \|CHAR \|BINARY * | データファイルが暗号化されていない場合、圧縮ありの変換 |
| デフォルト Base SAS Engine ENCRYPT=YES * | 暗号化ありの変換 |
| 一貫性制約 | エラーなしで解除 |
| 監査ファイル | エラーなしで解除 |
| 世代ファイル | エラーなしで解除 |

* デフォルト Base SAS Engine ファイルで圧縮と暗号化の両方が行われている場合、圧縮は解除されますが、暗号化は保持されます。SAS では、圧縮のかわりにデータセットのセキュリティが保持されます。

## PROC COPY を使用したデフォルト Base SAS Engine データセットの変換

既存のデフォルト Base SAS Engine データセットから SPD Engine データセットを作成するには、次の例に示すように、単に COPY プロシジャを使用します。PROC COPY ステートメントでは、デフォルト Base SAS Engine 形式のデータセット LOCAL.RACQUETS が新しい SPD Engine 形式のデータセット SPORT.RACQUETS にコピーされます。

```
libname sport spde 'conversion_area';

proc copy in=local out=sport;
```

```
select racquets;
run;
```

デフォルト Base SAS Engine データセットのインデックスが SPD Engine インデックスとして自動的に再生成される場合でも(.hdx ファイルと.idx ファイルの両方)、データセットオプション ASYNCINDEX=NO がデフォルトなので、並列作成されることはありません。

SPD Engine データセットが暗号化される場合は、データコンポーネントファイルのみ暗号化されます。メタデータファイルおよび両方のインデックスファイルは暗号化されません。

### *PROC APPEND を使用したデフォルト Base SAS Engine データセットの変換*

新しい SPD Engine データセットに対してデータセットオプションを指定する必要がある場合は、APPEND プロシジャを使用します。

次の例では、PROC APPEND を使用して、デフォルト Base SAS Engin データセットから SPD Engine データセットを作成します。ASYNCINDEX=YES データセットオプションでは、インデックスの並列作成が指定されます。PARTSIZE=オプションでは、100MB のパーティションの作成が指定されます。

```
libname spdelib spde 'old_data';
libname somelib 'old_data';
proc append base=spdelib.cars (asyncindex=yes partsize=100)
data=somelib.cars;
run;
```

## 新しい SPD Engine データセットの作成とロード

新しい SPD Engine データセットを作成するには、DATA ステップ、任意の PROC ステートメント [1] と OUT=オプション、または PROC SQL と CREATE TABLE=オプションを使用します。

次の例では、DATA ステップを使用して、report_area ディレクトリに新しい SPD Engine データセット CARDATA.OLD_AUTOS を作成します。

```
libname cardata spde '/report_area';

data cardata.old_autos(compress=no encrypt=yes pw=secret);
input year $4. @6 manufacturer $12. @18 model $12. @31 body_style $5. @37
engine_liters @42 transmission_type $1. @45 exterior_color
$10. @55 mileage @62 condition;

datalines;

1966 Ford Mustang conv 3.5 M white 143000 2
1967 Chevrolet Corvair sedan 2.2 M burgundy 70000 3
1975 Volkswagen Beetle 2door 1.8 M yellow 80000 4
1987 BMW 325is 2door 2.5 A black 110000 3
1962 Nash Metropolitan conv 1.3 M red 125000 3
;
```

1 PROC MIGRATE 以外

```
run;
```

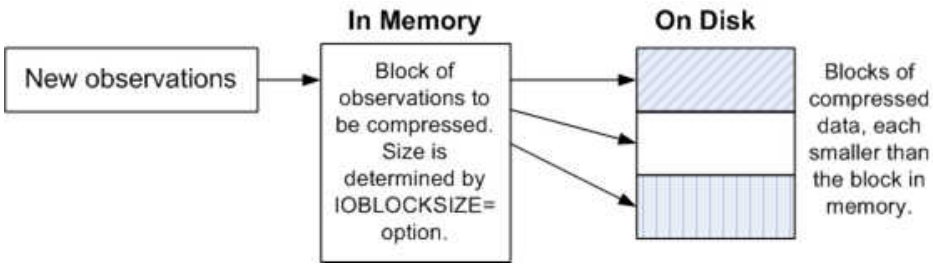
*注:* SPD Engine では、暗号化と圧縮は相互に排他的です。ENCRYPT=オプションを使用できるのは、圧縮されていない SPD Engine データファイルを作成する場合のみです。SPD Engine データセットを作成する際に暗号化と圧縮を両方とも行うことはできません。

## SPD Engine データセットの圧縮

COMPRESS=YES|BINARY|CHAR の場合、SPD Engine では、データコンポーネントファイルが、作成時にブロックで圧縮されます。SPD Engine では、ユーザー指定の圧縮はサポートされません。さらに、圧縮と暗号化の両方が行われたデフォルト Base SAS Engine データセットを移行する場合、暗号化は保持されますが、圧縮は解除されます。

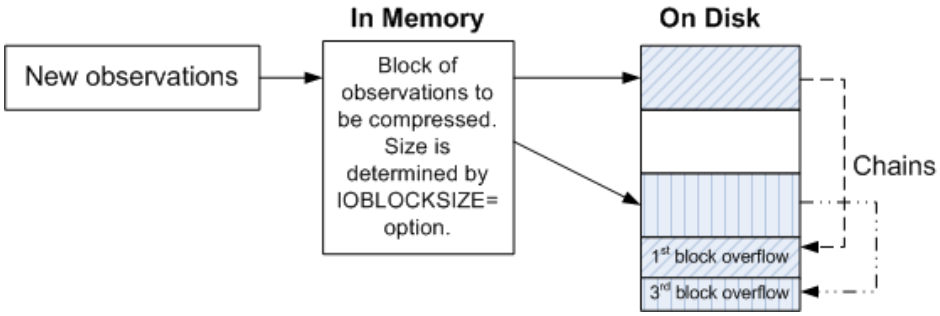
いったん圧縮データセットを作成すると、ブロックサイズは変更できません。圧縮ブロックは連続的に格納され、ブロック間に空き領域はありません。次の図は、ブロックのディスクへの格納方法を示しています。

***図 2.1*** *ディスク上の圧縮ブロック*




圧縮後のデータセットへの更新で、ブロック内の利用可能容量よりも多くの領域が必要になった場合、SPD Engine ではオーバーフローを保持するために新しいブロックフラグメントが作成されます。さらに更新をして再度オーバーフローが発生した場合、新しいブロックフラグメントが作成され、チェーンが形成されます。次の図は、更新によるディスク上でのブロックチェーンの作成を示しています。

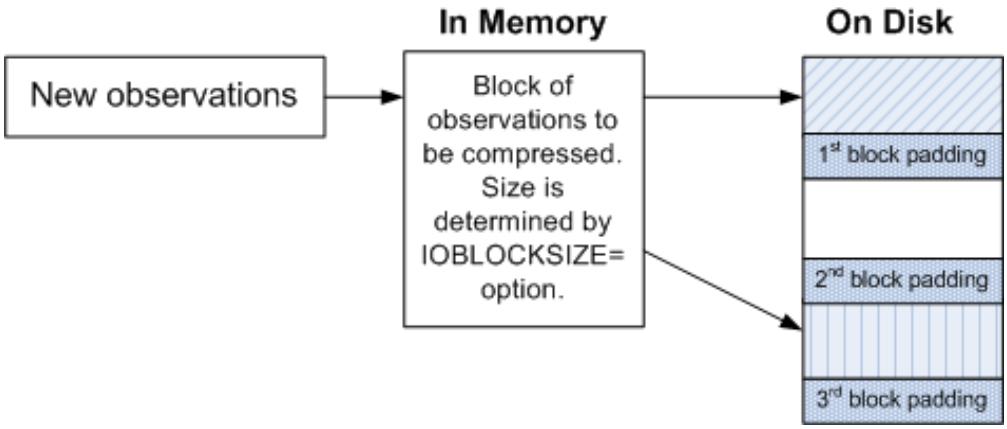
***図 2.2*** *オーバーフローのある圧縮ブロック*




チェーンが長くなりすぎると、パフォーマンスに影響します。チェーンを削除してブロックのサイズ変更を行うには、データセットを新しいデータセットにコピーし、IOBLOCKSIZE= (62 ページ) を出力データセットに適したブロックサイズに設定します。

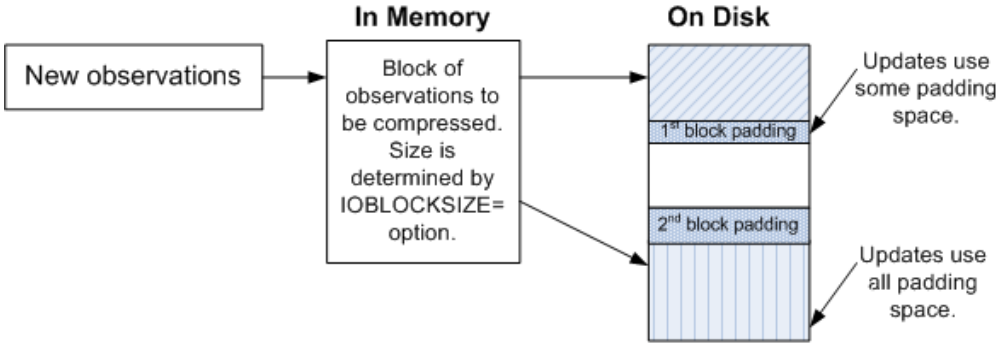
データセットの頻繁な更新が予想される場合、PADCOMPRESS= (66 ページ)の使用をお勧めします。SPD Engine では、新しいブロックフラグメントが作成されるかわりに、各ブロックに対する埋め込み領域が作成されます。次の図は、更新に対する各ブロックの領域の埋め込みを示しています。

**図 2.3** *圧縮された埋め込みブロック*




圧縮後のデータセットへの更新で、ブロック内の利用可能容量よりも多くの領域が必要になった場合、SPD Engine では、新しいブロックフラグメントが作成されるかわりに、各ブロックの埋め込み領域が使用されます。次の図は、更新による埋め込み領域の減少を示しています。

**図 2.4** *更新のある圧縮された埋め込みブロック*




CONTENTS プロシジャでは、圧縮についての情報が印刷されます。次の例では、CONTENTS プロシジャ出力の圧縮情報フィールドについて説明します。

***アウトプット 2.1*** *CONTENTS プロシジャの圧縮情報出力*

| - Compressed Info | - |
|---|---|
| Number of compressed blocks | 202 |
| Raw data blocksize | 32736 |
| Number of blocks with overflow | 5 |
| Max overflow chain length | 3 |
| Block number for max chain | 80 |
| Min overflow area | 87 |
| Max overflow area | 181 |

圧縮ブロック数
　データの格納に必要な圧縮ブロック数。

生データブロックサイズ
　IOBLOCKSIZE=データセットオプションに指定したサイズから計算されたバイト単位の圧縮ブロックサイズ。

オーバーフローのあるブロック数
　より多くの領域を必要とする圧縮ブロック数。データが更新されて、圧縮された新ブロックが圧縮された旧ブロックよりも大きくなった場合、オーバーフローブロックフラグメントが作成されます。

オーバーフローチェーンの最大長
　単一ブロックのオーバーフロー最大数。たとえば、圧縮ブロックが更新されて大きくなり、再度更新されてより大きなサイズになった場合、オーバーフローチェーンの最大長は 2 になります。

最大チェーンのブロック数
　最大数のオーバーフローブロックを含むブロック数。

最小オーバーフロー領域
　オーバーフローに必要なディスク領域最小容量。

最大オーバーフロー領域
　オーバーフローに必要なディスク領域最大容量。

## SPD Engine コンポーネントファイルの命名規則

SPD Engine データセットを作成する場合、多くのコンポーネントファイルも作成できます。SPD Engine コンポーネントファイルは次の命名規則で格納されます。

```
filename.mdf.0.p#.v#.spds9
filename.dpf.fuid.p#.v#.spds9
filename.idxsuffix.fuid.p#.v#.spds9
filename.hbxsuffix.fuid.p#.v#.spds9
```

*filename*
有効な SAS ファイル名。

mdf
メタデータコンポーネントファイルを識別します。

dpf
分割データコンポーネントファイルを識別します。

*p#*
パーティション番号です。

*v#*
バージョン番号です。[1]

*fuid*
一意のファイル ID。プライマリ(メタデータ)パスに相当する 16 進数。

idx*suffix*
インデックスのセグメント化ビューを識別します。このとき *suffix* はインデックス名です。

hbx*suffix*
インデックスのグローバルビューを識別します。このとき *suffix* はインデックス名です。

spds9
SAS 9 SPD Engine コンポーネントファイルを示します。

表 2.2 は、この LIBNAME ステートメントおよび DATA ステップの使用時に作成されるデータセットコンポーネントファイルを示しています。

```
libname sample spde '/DATA01/mydir'
datapath=('/DATA01/mydir' '/DATA02/mydir')
indexpath=('/IDX1/mydir');
data sample.mine(index=(ssn));
do i=1 to 100000;
ssn=ranuni(0);
end;
run;
```

***表 2.2*** *データセットコンポーネントファイル*

| | |
|---|---|
| mine.mdf.0.0.0.spds9 | メタデータコンポーネントファイル |
| mine.dpf.000032a6.0.1.spds9 | データファイルパーティション #1 |
| mine.dpf.000032a6.1.1.spds9 | データファイルパーティション #2 |
| mine.dpf.000032a6.*n*-1.1.spds9 | データファイルパーティション *#n* |
| mine.dpf.000032a6.*n*.1.spds9 | データファイルパーティション *#n*+1 |

1 バージョン番号は、データセットが更新されるとき、すなわち、データセットが UPDATE モードで開くときのみ増加します。PROC SORT のようなデータセットを置換する操作では、バージョン番号が増すかわりに、1 にリセットされます。

| | |
|---|---|
| mine.hbxssn.000032a6.0.1.spds9 | 変数 SSN のグローバルインデックスデータセット |
| mine.idxssn.000032a6.0.1.spds9 | 変数 SSN のセグメント化インデックスデータセット |

# SPD Engine における効率的なインデックス付け

## *インデックス付けの並列処理*

インデックスによって、WHERE 式処理および BY 式処理のパフォーマンス向上が可能になります。SPD Engine では、並列処理ができるので、インデックスの迅速な作成および更新が可能になります。

SPD Engine のインデックスは特に、さまざまなサイズのデータセットおよびデータ分散に適しています。これらのインデックスには、インデックス付き変数値のセグメント化ビューとグローバルビューの両方が含まれます。この機能を使用すると、SPD Engine で、次のクエリの両方を最適にサポートできます。

- グローバルデータビューを必要とするクエリ(BY 式処理など)
- セグメント化ビューを必要とするクエリ(WHERE 式の並列処理など)

SPD Engine データセットを暗号化する場合は、データコンポーネントファイルのみ暗号化されます。メタデータファイルやインデックスファイルなどの他のファイルは暗号化されません。

## *並列処理でのインデックス作成*

SPD Engine データのインデックスは、非同期で並列作成できます。非同期並列インデックスの作成を可能にするには、“ASYNCINDEX=データセットオプション” (44 ページ)を使用します。

インデックスを有するデフォルト Base SAS Engine データセットから SPD Engine データセットを作成する場合、このオプションを、DATA ステップの INDEX=オプションや PROC DATASETS INDEX CREATE と一緒に使用するか、PROC APPEND ステートメントで使用します。それぞれの方法で、1 回のデータセットスキャンによってすべての宣言インデックスを事前設定することができます。

*注:* 暗号化され、インデックスもあるデフォルト Base SAS Engine データセットから SPD Engine データセットを作成する場合、SPD Engine データセットではインデックスは暗号化されません。詳細については、“デフォルト Base SAS Engine データセットから SPD Engine データセットへの変換” (16 ページ)を参照してください。

次の例は、DATA ステップを使用して並列作成されたインデックスを示しています。変数 X に単一インデックス、変数 A および B に複合インデックスが作成されます。

```
data foo.mine(index=(x y=(a b)) asyncindex=yes);
x=1;
a="Doe";
b=20;
run;
```

複数のインデックスを並列作成するには、すべてのキー並べ替えを同時に行うのに十分なユーティリティディスク領域を割り当てる必要があります。十分なメモリ領域も割り当ててください。ディスク領域を割り当てるには “SPDEUTILLOC=システムオプション” (89 ページ) を使用し、追加メモリを割り当てるには構成ファイル内か起動時に “SPDEINDEXSORTSIZE=システムオプション” (87 ページ) を使用します。

DATASETS プロシジャには柔軟性があるので、複数の MODIFY グループを使用したバッチ並列インデックス作成が可能です。一度にすべてのインデックスを作成すると大量の領域が必要になりますが、そのかわりに、次の例に示すようにグループ単位でインデックスを作成できます。

```
proc datasets lib=main;
modify patients(asyncindex=yes);
index create number;
index create class;
run;
modify patients(asyncindex=yes)'
index create lastname firstname;
run;
modify patients(asyncindex=yes);
index create fullname=(lastname firstname);
index create class_sex=(class sex);
run;
quit;
```

インデックス NUMBER およびインデックス CLASS、インデックス LASTNAME およびインデックス FIRSTNAME、ならびにインデックス FULLNAME および CLASS_SEX がそれぞれ並列作成されます。

*注:* 変数名に次の特殊文字のいずれかが含まれている場合は、変数に対して単一インデックスも複合インデックスも作成できません。

```
" * | \ : / < > ? -
```

### *並列処理でのインデックス更新*

SPD Engine では、データセット追加操作時の並列インデックス更新もサポートされています。複数のスレッドによって、データストアおよびインデックスファイルの更新が可能になります。SPD Engine では、データセットの追加操作または挿入操作が、並列実行の可能な一連のステップに分解されます。並列処理の到達レベルは、データセット内のインデックス数によって決まります。並列インデックス作成と同様に、この操作では、インデックス追加処理の一部であるキー並べ替えのためにメモリおよびディスク領域が使用されます。メモリを割り当てるにはシステムオプション SPDEINDEXSORTSIZE=、ディスク領域を割り当てるには SPDEUTILLOC=を使用します。

## SPD Engine ファイルのバックアップ

SPD Engine データセットをバックアップする場合は、次の要件に注意してください。

- データセットを構成するファイルはすべて、異なるディスクやファイルシステムに存在する場合でも、まとめて同時にバックアップするようにしてください。
- 更新中のファイルがある場合、データセットのバックアップは行わないでください。
- バックアップ後には毎回テストを実行し、バックアップが成功したか確認します。

*3 章*

# SPD Engine LIBNAME ステートメントオプション



## SPD Engine LIBNAME ステートメントについて

このセクションには、SPD Engine の LIBNAME ステートメントに有効なすべての LIBNAME オプションのための参考情報が含まれています。これらの LIBNAME オプションのうちのいくつかはデータセットオプションでもあります。デフォルト Base SAS Engine でのように、両方のオプションが設定された場合、対応する LIBNAME オプションよりデータセットオプションが優先します。

## 構文

LIBNAME *libref* SPDE '*full-primary-path*' <*options*> ;

*libref*
SAS 命名規則に則った最大 8 文字長の名前。TEMP が環境変数として使用されていない場合には、SPD Engine ライブラリの libref で TEMP は指定できません。

*'full-primary-path'*
SPD Engine ライブラリ用プライマリパスの完全パス名。その名前が動作環境で認識されることが必要です。単一引用符または二重引用符で名前を囲みます。DATAPATH=オプションと INDEXPATH=オプションが指定されない場合は、イン

デックスとデータコンポーネントが同じ場所に格納されます。プライマリパスはライブラリごとに一意であることが必要です。同じプライマリパス名を参照しているが異なる Libdef は、同じライブラリであると解釈され、データが失われる場合があります。

*注:* ファイルの場所の名前は変更できません。DATAPATH=、INDEXPATH=、METAPATH=、プライマリパス LIBNAME オプションのいずれかを指定する場合、データセットのアクセス時は常にデータセット作成時に使用されたものと同一パスが使用されることを確認してください。これらの場所の名前は、データセットに内部に格納されます。

*options*
1 つ以上の SPD Engine LIBNAME ステートメントオプション。

*動作環境情報*: 有効なライブラリの指定と構文は、動作環境に固有です。詳細については、動作環境に関する SAS のドキュメントを参照してください。

## SPD Engine LIBNAME ステートメントオプションのリスト

ACCESS=READONLY
データセットの読み取りはできるが、更新や作成はできないことを指定します。

BYSORT=
BY ステートメントが検出されたとき、SPD Engine が自動並べ替えを行なうように指定します。

DATAPATH=
SPD Engine データセットのためのデータパーティション(.dpf)を格納するパスのリストを指定します。

ENDOBS=
ユーザー定義のオブザベーション処理範囲における最後のオブザベーション番号を指定します。

IDXBY
SPD Engine での BY ステートメント処理時にインデックスを使用するかどうかを指定します。

INDEXPATH=
SPD Engine データセットに関連付けられた 2 種類のインデックスコンポーネントファイル(.hbx および.idx)を格納するパスまたはパスのリストを指定します。

METAPATH=
SPD Engine データセットのためのメタデータ(.mdf)コンポーネントファイルを格納するオーバーフローパスのリストを指定します。

PARTSIZE=
データコンポーネントパーティションがとられる最大サイズを指定します。値は SPD Engine データセットの作成時に指定されます。このサイズは固定サイズです。この指定はデータコンポーネントファイルにのみ適用されます。

STARTOBS=
ユーザー定義のオブザベーション処理範囲における最初のオブザベーション番号を指定します。

TEMP=
プライマリディレクトリの一時サブディレクトリにライブラリを格納するよう指定します。

# ディクショナリ

## ACCESS= LIBNAME ステートメントオプション

データソースのアクセスレベルを決定します。

**デフォルト:** none

**エンジン:** SPD Engine only

### 構文

**ACCESS**=READONLY

#### *必須引数*

**READONLY**
データセットの読み取りはできるが、更新や作成はできないことを指定します。

### 詳細

このオプションを使用して、データソースへの書き込みを防ぎます。このオプションを省略すると、必要なデータソース権限がある場合には、データセットの読み込み、更新、および作成ができます。

## BYSORT= LIBNAME ステートメントオプション

BY ステートメントの検出時に SPD Engine で自動並べ替えを実行することを指定します。

**デフォルト:** YES

**操作:** BYNOEQUALS=

**エンジン:** SPD Engine only

### 構文

**BYSORT**=YES | NO

#### *必須引数*

**YES**
BY ステートメントの前に PROC SORT を呼び出すかわりに、BY ステートメントを検出するたびに BY 変数に基づいて自動的にデータを並べ替えるように指定します。

**NO**
データを BY 変数に基づいて並べ替えないように指定します。NO を指定する場合は、BY ステートメントの前にすでにデータが並べ替えられている必要があります。インデックスは使用されません。

## 詳細

デフォルト Base SAS Engine を使用した DATA または PROC ステップ処理では、インデックスがない場合やオブザベーションが順序どおりではない場合、BY ステートメントが発行される前にデータセットを並べ替える必要があります。それに対して、SPD Engine では、オブザベーションが順序どおりではない場合、デフォルトで、アプリケーションに返されるデータが並べ替えられます。並べ替えたデータセットが新規作成される PROC SORT とは異なり、SPD Engine の自動並べ替えでは、保存用データセットは変更されず、データセットは新規作成されません。ただし、ユーティリティファイル領域は使用されます。詳細については、“SPDEUTILLOC=システムオプション” (89 ページ)を参照してください。

デフォルトは BYSORT=YES です。BYSORT=YES 引数によって自動並べ替えが有効になり、オブザベーションは BY グループ順序で出力されます。データセットオプション BYNOEQUALS=YES の場合、グループ内のオブザベーションは、データセットでの順序とは異なる順序で出力される可能性があります。データセット順序を保持するには BYNOEQUALS=NO を設定します。

BYSORT=NO 引数の場合は、指定した BY 変数ですでにデータが並べ替えられている必要があります。前の並べ替えで PROC SORT を使用したか、または BY 変数順序でデータセットが作成されていると、このような結果になる可能性があります。BYSORT=NO の場合、グループ化されたデータはデータセット順序でアプリケーションに配信されます。インデックスは、BY 変数順序のオブザベーションの取得には使用されません。BYSORT=NO の場合、データセットオプション BYNOEQUALS=は無効になります。

LIBNAME ステートメントで BYSORT=オプションを指定する場合、PROC ステップまたは DATA ステップでの BYSORT=の指定によって上書きされる可能性があります。したがって、LIBNAME ステートメントで BYSORT=NO を設定し、その後に BY ステートメントを記述します。(事前に PROC SORT を使用するか、または並べ替え順序で作成されたために)データが並べ替え済みの場合以外は、エラーが発生します。入力モードや更新モードで開くために、LIBNAME ステートメントの BYSORT=NO を上書きするには、DATA ステップまたは PROC ステップで BYSORT=YES を設定します。重要な点は、BYSORT=NO によって、エンジンがデータの並べ替え処理をしないように指示されることです。

MSGLEVEL=I SAS システムオプションを設定した場合、BYSORT=YES および IDXWHERE=データセットオプションを使用すると、次のメッセージが SAS ログに書き込まれます。

- IDXWHERE=YES で BY 変数にインデックスがある場合、テーブルの行の並べ替えにそのインデックスが使用されます。次のメッセージが SAS ログに書き込まれます。

  ```
  Note: BY ordering was produced by using an index for table tablename.
  ```

- IDXWHERE=NO または IDXWHERE=YES で、BY 変数にインデックスがない場合、SPD Engine で自動並べ替えが実行され、テーブルの行が並べ替えられます。次のメッセージが SAS ログに書き込まれます。

  ```
  Note: BY ordering was produced by performing an automatic
  sort on table tablename.
  ```

# 例

## 例 1: BYSORT=YES を使用するデフォルトのグループフォーマット

```
libname growth spde 'D:\SchoolAge';
data growth.teens;
input Name $ Sex $ Age Height Weight;
datalines;
Alfred M 14 69.0 112.5
Carol F 14 62.8 102.5
James M 13 57.3 83.0
Janet F 15 62.5 112.5
Judy F 14 64.3 90.0
Philip M 16 72.0 150.0
William M 15 66.5 112.0
;
proc print data=growth.teens; by sex;
run;
```

BYSORT=YES がデフォルトのため、PROC SORT を使用してデータを並べ替えていなくても、エラーは発生しません。出力は次のとおりです。

***アウトプット 3.1*** *BYSORT=YES を使用するデフォルトのグループフォーマット*

**The SAS System**

**Sex=F**

| Obs | Name | Age | Height | Weight |
|---|---|---|---|---|
| 2 | Carol | 14 | 62.8 | 102.5 |
| 4 | Janet | 15 | 62.5 | 112.5 |
| 5 | Judy | 14 | 64.3 | 90.0 |

**Sex=M**

| Obs | Name | Age | Height | Weight |
|---|---|---|---|---|
| 1 | Alfred | 14 | 69.0 | 112.5 |
| 3 | James | 13 | 57.3 | 83.0 |
| 6 | Philip | 16 | 72.0 | 150.0 |
| 7 | William | 15 | 66.5 | 112.0 |

### *例 2: LIBNAME ステートメントでの BYSORT=NO の使用*

次の例では、DATA ステップまたは PROC ステップで BYSORT=YES が指定されず、LIBNAME ステートメントでの BYSORT=NO 指定が上書きされなかったので、SAS はエラーを返します。自動並べ替えが抑制される場合(BYSORT=NO)、(たとえば、PROC SORT を使用する)BY ステートメントの前にデータが BY 変数で並び替えられてる必要があります。

```
libname growth spde 'D:\SchoolAge' bysort=no;
proc print data=growth.teens;
by sex;
run;
```

```
ERROR: Data set GROWTH.TEENS is not sorted in ascending sequence.
The current by-group has Sex = M and the next by-group has Sex = F.
NOTE: The SAS System stopped processing this step because of errors.
```

## DATAPATH= LIBNAME ステートメントオプション

SPD Engine データセットのためのデータパーティション(.dpf)を格納するパスのリストを指定します。

**デフォルト:** LIBNAME ステートメントに指定されたプライマリパス

**操作:** PARTSIZE=データセットまたは LIBNAME オプション

**エンジン:** SPD Engine のみ

### 構文

**DATAPATH**=(*'path1' 'path2'...*)

#### *必須引数*

***'pathn'***
単一引用符あるいは二重引用符で囲んだ完全パス名をかっこ内に指定します。複数の引数はスペースで区切ります。

*注:* DATAPATH=オプションに指定したパス名はライブラリごとに一意であることが必要です。同じパス名を参照しているが異なる Libdef では、データが失われる場合があります。

*注:* データが zFS ファイルシステムにある場合は、パス指定は 1 つのみで必須です。zFS システムは複数の論理ボリュームに渡るパーティションを自動的に展開します。

### 詳細

SPD Engine は、データをすべて格納するために必要な数だけのパーティションを作成します。パーティションのサイズは PARTSIZE=オプションを使用して設定され、DATAPATH=オプションを使用して指定されたパスにパーティションが周期的に作成されます。

*注:* ローカルでマウントされたドライブにデータを格納し、リモートコンピュータからデータにアクセスする場合は、LIBNAME の指定時にリモートパス名を使用します。たとえば、localA コンピュータで**/data01** と**/data02** がローカルにマウントされているドライブである場合、LIBNAME ステートメントでパス名には**/nfs/localA/data01** と**/nfs/localA/data02** を使用します。これらのファイルのパス名は変更できません。DATAPATH=、INDEXPATH=、METAPATH=、プライマリパス

LIBNAME オプションのいずれかを指定すると、データセットの作成時に使用されたものと同一パスがデータセットにアクセスするたびに使用されるようになります。これらの場所の名前はデータセットの内部に格納されます。パス名を一部でも変更すると、SPD Engine でデータセットを検索できなくなったり、データセットが破損したりする可能性があります。

### 例: 最初のパーティションの DATAPATH=

最初のパーティションのパスはランダムに選択され、その後は周期的な方法で続きます。

```
libname mylib spde '/metadisk/metadata'
datapath=('/disk1/dataflow1' '/disk2/dataflow2' '/disk3/dataflow3');
```

たとえば、/DISK2/DATAFLOW2 が最初のパスとしてランダムに選択されると、最初のパーティションはそこに置かれます。2 番目のパーティションは/DISK3/DATAFLOW3 に置かれ、3 番目のパーティションは/DISK1/DATAFLOW1 というように、配置されます。

## ENDOBS= LIBNAME ステートメントオプション

ユーザー定義のオブザベーション処理範囲における最後のオブザベーション番号を指定します。

**デフォルト:** データセット内の最後のオブザベーション

**制限事項:** ENDOBS=は入力データセットにのみ使用

OBS=システムおよびデータセットオプション、または FIRSTOBS=システムおよびデータセットオプションと一緒には使用不可

**操作:** STARTOBS=

**エンジン:** SPD Engine のみ

### 構文

**ENDOBS=***n*

#### *必須引数*

***n***
終了オブザベーションの番号。

### 詳細

デフォルトでは、SPD Engine で、STARTOBS=オプションと ENDOBS=オプションを使用してオブザベーションの範囲を指定しない限り、データセット全体のすべてのオブザベーションが処理されます。STARTOBS=オプションを ENDOBS=オプションなしで使用した場合、ENDOBS=の暗黙値はデータセットの終わりになります。両方のオプションを一緒に使用する場合、ENDOBS=値を STARTOBS=値よりも大きくする必要があります。

デフォルト Base SAS Engine オプション FIRSTOBS=と違い、STARTOBS=および ENDOBS= SPD Engine オプションは LIBNAME ステートメントで使用できます。

*注:* OBS=システムオプションおよび OBS=データセットオプションは、STARTOBS=もしくは ENDOBS=データセットまたは LIBNAME オプションと一緒には使用できません。

(WHERE 処理での ENDOBS=データセットオプションの使用については、SPD Engine データセットオプション (41 ページ) を参照してください。

### 例: STARTOBS=オプションおよび ENDOBS=オプション

次の例は、WHERE 節を実行する前に STARTOBS=オプションと ENDOBS=オプションがデータをサブセット化することを示します。例では、WHERE 式(PROC PRINT にある age >13)で適合した 4 つのオブザベーションを出力します。4 つのオブザベーションは、入力データセットから 5 つのオブザベーションが処理された結果です。

```
libname growth spde 'D:\SchoolAge' endobs=5;
data growth.teens;
input Name $ Sex $ Age Height Weight;
datalines;
Alfred M 14 69.0 112.5
Carol F 14 62.8 102.5
James M 13 57.3 83.0
Janet F 15 62.5 112.5
Judy F 14 64.3 90.0
Philip M 16 72.0 150.0
William M 15 66.5 112.0
;
proc print data=growth.teens;
where age >13;
run;
```

出力は次のとおりです。

The SAS System

| Obs | Name | Sex | Age | Height | Weight |
|---|---|---|---|---|---|
| 1 | Alfred | M | 14 | 69.0 | 112.5 |
| 2 | Carol | F | 14 | 62.8 | 102.5 |
| 4 | Janet | F | 15 | 62.5 | 112.5 |
| 5 | Judy | F | 14 | 64.3 | 90.0 |

## IDXBY= LIBNAME ステートメントオプション

SPD Engine での BY ステートメント処理時にインデックスを使用するかどうかを指定します。

**デフォルト:** YES

**操作:** BYSORT=

**エンジン:** SPD Engine のみ

## 構文

**IDXBY**=YES | NO

### *必須引数*

**YES**
BY ステートメントのインデックス変数を処理するときにインデックスを使用します。

*注:* BY ステートメントに 2 つ以上の変数または DESCENDING オプションが指定されている場合は、IDXBY=YES でもインデックスは使用されません。

**NO**
BY ステートメントのインデックス変数を処理するときにインデックスを使用しません。

**注** IDXBY=NO の場合、BY ステートメントの処理時に自動並べ替えが実行されます。

## 詳細

IDXBY= LIBNAME オプションを使用する場合は、BYSORT=YES オプションを使用し、BY 変数にインデックスを付けるようにしてください。

場合によっては、データを自動的に並べ替えると、SPD Engine のパフォーマンスが向上することがあります。自動並べ替えを使用するには、BYSORT=YES を設定し、IDXBY=NO を指定する必要があります。

SAS システムオプション MSGLEVEL=I を設定し、BY 処理情報が SAS ログに書き込まれるようにします。IDXBY=LIBNAME オプションおよび BYSORT=YES オプションを使用すると、SAS ログに次のメッセージが書き込まれます。

- IDXBY=YES で BY 変数にインデックスがある場合、テーブルの行の並べ替えにそのインデックスが使用されます。次のメッセージが SAS ログに書き込まれます。

```
 NOTE: BY ordering was produced by using an index for
table tablename.
```

- IDXBY=NO を使用する場合、次のメッセージが SAS ログに書き込まれます。

```
NOTE: BY ordering was produced by performing an automatic sort
on table tablename.
```

## 例

### *例 1: IDXBY=NO LIBNAME オプションの使用*

```
libname permdata spde 'c:/sales' idxby=no;
options msglevel=i;
proc means data=permdata.customer;
var sales;
by state;
run;
```

次のメッセージが SAS ログに書き込まれます。

```
NOTE: BY ordering was produced by performing an automatic sort
on table PERMDATA.customer.
NOTE: There were 100 observations read from the data
set PERMDATA.CUSTOMER.
```

### *例 2: IDXBY=YES LIBNAME オプションの使用*

次の例は IDXBY=YES を使用しています。

```
libname permdata spde 'c:\sales' idxby=yes;
options msglevel=i;
proc means data=permdata.customer;
var sales;
by state;
run;
```

次のメッセージが SAS ログに書き込まれます。

```
NOTE: BY ordering was produced by using an index for table
PERMDATA.customer.
NOTE: There were 2981 observations read from the data set
PERMDATA.CUSTOMER.
```

## INDEXPATH= LIBNAME ステートメントオプション

SPD Engine データセットに関連付けられた 2 種類のインデックスコンポーネントファイル(.hbx および.idx)を格納するパスまたはパスのリストを指定します。

**デフォルト:** LIBNAME ステートメントに指定されたプライマリパス

**エンジン:** SPD Engine のみ

### 構文

**INDEXPATH**=(*'path1'* )<*'path2'...*>

#### *必須引数*

***'pathn'***
単一引用符あるいは二重引用符で囲んだ完全パス名をかっこ内に指定します。複数の引数はスペースで区切ります。

*注:* INDEXPATH=オプションに指定したパス名はライブラリごとに一意であることが必要です。同じパス名を参照しているが異なる Libdef では、データが失われる場合があります。

### 詳細

SPD Engine は、指定された場所に 2 つのインデックスコンポーネントファイルを作成します。複数のパス名が指定されている場合、最初のパスはランダムに選択されます。パスに十分な領域がない場合、インデックスコンポーネントファイルは指定された次のファイルパスなどにオーバーフローします。

*注:* ローカルでマウントされたドライブにデータを格納し、リモートコンピュータからデータにアクセスする場合は、LIBNAME の指定時にリモートパス名を使用します。たとえば、localA コンピュータで**/data01** と**/data02** がローカルにマウントされているドライブである場合、LIBNAME ステートメントでパス名には**/nfs/localA/data01** と**/nfs/localA/data02** を使用します。これらのファイルのパス名は変更できません。DATAPATH=、INDEXPATH=、METAPATH=、プライマリパス LIBNAME オプションのいずれかを指定すると、データセットの作成時に使用されたものと同一パスがデータセットにアクセスするたびに使用されるようになります。これらの場所の名前はデータセットの内部に格納されます。パス名を一部でも変

更すると、SPD Engine でデータセットを検索できなくなったり、データセットが破損したりする可能性があります。

### 例: インデックスコンポーネントファイルの作成

次の例では、インデックスコンポーネントファイルを、パス/DISK1/IDXFLOW1、/DISK2/IDXFLOW2、および/DISK3/IDXFLOW3 にわたって作成します。

```
libname mylib spde '/metadisk/metadata'
datapath= ('/disk1/dataflow1' '/disk2/dataflow2'
'/disk3/dataflow3')
indexpath=('/disk1/idxflow1' '/disk2/idxflow2'
'/disk3/idxflow3' );
```

最初のインデックスコンポーネントファイルのパスはランダムに選択されます。SAS は、最初の場所にインデックスコンポーネントファイルを置き、その場所がいっぱいになるまで周期的にそこを使い続けます。たとえば、/DISK2/IDXFLOW2 がランダムに選択された場合は、そこに最初のインデックスコンポーネントファイルが置かれます。その場所がいっぱいになったときインデックスコンポーネントファイルは、/DISK3/IDXFLOW3、その後/DISK1/IDXFLOW1 にオーバーフローします。

## METAPATH= LIBNAME ステートメントオプション

SPD Engine データセットのためのメタデータ(.mdf)コンポーネントファイルを格納するオーバーフローパスのリストを指定します。

**デフォルト:** LIBNAME ステートメントに指定されたプライマリパス

**エンジン:** SPD Engine のみ

### 構文

**METAPATH=**(*'path1'*)<*'path2'...*>

#### *必須引数*

***'pathn'***

単一引用符あるいは二重引用符で囲んだ完全パス名をかっこ内に指定します。複数の引数はスペースで区切ります。

*注:* METAPATH=オプションに指定したパス名はライブラリごとに一意であることが必要です。同じパス名を参照しているが異なる Libdef では、データが失われる場合があります。

### 詳細

METAPATH=オプションは、メタデータコンポーネントファイルのオーバーフロー領域専用の領域を指定します。各データセットのメタデータコンポーネントファイルは、プライマリパスで開始する必要があり、プライマリパスがいっぱいになったとき、METAPATH=に指定された場所へオーバーフローが発生します。

*注:* ローカルでマウントされたドライブにデータを格納し、リモートコンピュータからデータにアクセスする場合は、LIBNAME の指定時にリモートパス名を使用します。localA コンピュータで**/data01** と**/data02** がローカルにマウントされているドライブである場合、LIBNAME ステートメントでパス名に**/nfs/localA/data01** と**/nfs/localA/data02** を使用します。これらのファイルのパス名は変更できま

せん。DATAPATH=、INDEXPATH=、METAPATH=、プライマリパス LIBNAME オプションのいずれかを指定すると、データセットの作成時に使用されたものと同一パスがデータセットにアクセスするたびに使用されるようになります。これらの場所の名前はデータセットの内部に格納されます。パス名を一部でも変更すると、SPD Engine でデータセットを検索できなくなったり、データセットが破損したりする可能性があります。

### 例: オーバーフローメタデータファイルパーティションの作成

次の例では、必要に応じ、パス/DISK1/METAFLOW1 を使用してオーバーフローメタデータファイルを作成します。

/METADISK/METADATA がいっぱいになった場合、メタデータは/DISK1/METAFLOW1 へオーバーフローします。

```
libname mylib spde '/metadisk/metadata'
datapath=('/disk1/dataflow1' '/disk2/dataflow2')
metapath=('/disk1/metaflow1');
```

## PARTSIZE= LIBNAME ステートメントオプション

データコンポーネントパーティションに指定可能な最大サイズ(MB、GB または TB 単位)を指定します。値は SPD Engine データセットの作成時に指定されます。このサイズは固定サイズです。この指定はデータコンポーネントファイルにのみ適用されます。

**デフォルト:** 128 MB

**操作:** DATAPATH=
MINPARTSIZE=システムオプション

**エンジン:** SPD Engine のみ

### 構文

**PARTSIZE**=*n* | *n*M | *n*G | *n*T

#### *必須引数*

***n* | *n*M | *n*G | *n*T**
MB、GB または TB でのパーティションのサイズです。M、G または T なしで *n* を指定した場合のデフォルトは MB です。PARTSIZE=128 と PARTSIZE=128M は同じ意味です。最大値は、8,796,093,022,207MB です。

**制限事項** この制限は、オペレーティングシステム z/OS、Linux SLES 9 x86 および Windows ファミリの 32 ビットホストにのみ適用されます。SAS 9.3 でパーティションサイズが 2GB 以上のデータセットを作成した場合、SAS 9.2 以前のいずれのバージョンの SPD Engine でもそのデータセットを開くことはできません。次のエラーメッセージが SAS ログに書き込まれます。**ERROR: Unable to open data file because its data representation differs from the SAS session data representation.**

### 詳細

SPD Engine データは、後で並列処理されるため複数パーティションに格納される必要があります。PARTSIZE=を指定することで、SPD Engine データファイルが指定サイズ

に分割されます。実際のパーティションサイズは、*n*MB、GB または TB の指定サイズに収まるオブザベーション最大数に対応するように計算されます。オブザベーションの長さが 65K より大きいテーブルの場合、指定した PARTSIZE=と実際のパーティションサイズが一致しない可能性があります。この数字を一致させるには、32 の倍数であり、このオブザベーションの長さとなる PARTSIZE=を指定します。

SPD Engine データセットのデータ部分を固定サイズのファイルに分割(パーティション)することによって、ソフトウェアで一部の操作に高度なスケーラビリティが導入されます。SPD Engine では、スレッドを並列起動できます(たとえば、WHERE 評価に対して 1 パーティションにつき 1 スレッドまで)。また、別々のデータパーティションによって、スレッド間のファイルアクセス競合のオーバーヘッドなしでデータを処理できます。各パーティションが 1 ファイルであるため、小さなパーティションサイズの代償として、オブザベーションを格納するためにファイル(UNIX i ノードなど)の数の増加が必要になります。

PARTSIZE=を使用したスケーラビリティ制限は、DATAPATH=オプションで指定したファイルシステムを、どのように構成し、どのように複数のストライプボリュームに分散させるのかによって決まります。(個々のボリュームのストライピング構成を、ディスクアレイの複数のディスクコントローラまたは SCSI チャネルに分散させる必要があります。)構成の目的は、データ取得時の並列処理を最大限にすることです。ディスクストライピングの詳細については、**`http://support.sas.com/rnd/scalability`** の Scalability and Performance で“SPD Engine”の“I/O Setup and Validation”を参照してください。

PARTSIZE=の指定は SPD Engine システムオプション MINPARTSIZE=によって制限されます。このシステムオプションは通常システム管理者によって設定、保守されます。MINPARTSIZE=を使用すると、経験不足のユーザーが任意に小さなパーティションを作成し、その結果、多数のファイルが生成されるようなことは起こらなくなります。

パーティションサイズによって、データセット全体のスキャンを必要とする並列操作の多くの作業単位が決定されます。ただし、パーティションの増加が必ずしも処理の迅速化を意味するとは限りません。この折り合いをつけるには、データセットの格納に必要な物理ファイル(パーティション)の増加数と、パーティションの増加によって並列処理が可能になる作業量とのバランスをとる必要があります。パーティションが多くなると、データセットを処理するために開くファイルも多くなりますが、各パーティションのオブザベーション数は少なくなります。一般的なルールでは、パーティションは、1 データパスにつき 10 以下、1CPU につき 3 から 4 までとなります。

新しい SPD Engine データセットの適切なパーティションサイズを決定するには、次の点に注意する必要があります。

- データに対して実行するアプリケーションの種類
- データ量はどの程度か
- アプリケーションで使用可能な CPU はいくつあるか
- パーティションの格納に使用可能なディスクはどれか
- そのディスクと CPU との関係

たとえば、各 CPU の制御するディスクが 1 つだけの場合、適切なパーティションサイズは、各ディスクにほぼ同じ量のデータが含まれるサイズです。各 CPU の制御するディスクが 2 つの場合、適切なパーティションサイズは、負荷のバランスがとれているサイズです。各 CPU で、ほぼ同じ量の作業が行われます。

*注:* データセットの PARTSIZE=値は、データセットの作成後は変更できません。PARTSIZE=を変更するには、データセットを再作成し、LIBNAME ステートメント内か、またはその新しい(出力)データセットに対して、異なる PARTSIZE=値を指定する必要があります。

### 例: パーティションサイズの指定

LIBNAME ステートメントでパーティションサイズを指定する場合、そのライブラリに格納されるデータセットのほとんどに適切なサイズを選択する必要があります。たとえば、8 つのディスク構成を持っていると仮定します。最小のデータセットには 20 ギガバイトのデータがあり、最大のものには 50 ギガバイトのデータがあります。また、残りのデータセットにはそれぞれ 36 ギガバイトのデータがあります。1250M のパーティションサイズは、36 ギガバイトのデータセット(1 つのディスクあたり 4 つのパーティション)には最適です。20 ギガバイトのデータセットは、1 つのディスクあたり 2 つのパーティションを使用し、50 ギガバイトのデータセットは、1 つのディスク当たり 5 つのパーティションを使用します。

```
libname sales spde '/primdisk' partsize=1250M
datapath=('/disk01' '/disk02' '/disk03' '/disk04'
'/disk05' '/disk06' '/disk07' '/disk08');
```

## STARTOBS= LIBNAME ステートメントオプション

ユーザー定義のオブザベーション処理範囲における最初のオブザベーション番号を指定します。

**デフォルト:** データセット内の最初のオブザベーション

**制限事項:** STARTOBS=は入力データセットにのみ使用

OBS=システムおよびデータセットオプション、または FIRSTOBS=システムおよびデータセットオプションと一緒には使用不可

**操作:** ENDOBS=

**エンジン:** SPD Engine のみ

### 構文

**STARTOBS=***n*

#### *必須引数*

***n***

開始オブザベーションの番号。

### 詳細

デフォルトでは、SPD Engine で、STARTOBS=オプションと ENDOBS=オプションを使用してオブザベーションの範囲を指定しない限り、データセット全体のすべてのオブザベーションが処理されます。ENDOBS=オプションを STARTOBS=オプションなしで使用した場合、STARTOBS=の暗黙値は 1 になります。両方のオプションを一緒に使用する場合、STARTOBS=値を ENDOBS=値よりも小さくする必要があります。

デフォルト Base SAS Engine オプション FIRSTOBS=と違い、STARTOBS=および ENDOBS= SPD Engine オプションは LIBNAME ステートメントで使用できます。

*注:* FIRSTOBS=デフォルト Base SAS Engine オプションは、SPD Engine ではサポートされていません。OBS=システムオプションおよび OBS=データセットオプションは、STARTOBS=もしくは ENDOBS=データセットまたは LIBNAME オプションと一緒には使用できません。

(WHERE 処理での STARTOBS=データセットオプションの使用については、SPD Engine データセットオプション (41 ページ) を参照してください。)

### 例: WHERE 式の使用

次の例では、WHERE 式(PROC PRINT にある age >13)で適合した 5 つのオブザベーションを出力します。5 つのオブザベーションは、データセットにある 2 番目のオブザベーションから開始して 6 つのオブザベーションが処理された結果です。

```
libname growth spde 'D:\SchoolAge' startobs=2;
data growth.teens;
input Name $ Sex $ Age Height Weight;
datalines;
Alfred M 14 69.0 112.5
Carol F 14 62.8 102.5
James M 13 57.3 83.0
Janet F 15 62.5 112.5
Judy F 14 64.3 90.0
Philip M 16 72.0 150.0
William M 15 66.5 112.0
;
proc print data=growth.teens;
where age >13;
run;
```

出力は次のとおりです。

***アウトプット 3.2*** *STARTOBS=*

The SAS System

| Obs | Name | Sex | Age | Height | Weight |
|---|---|---|---|---|---|
| 2 | Carol | F | 14 | 62.8 | 102.5 |
| 4 | Janet | F | 15 | 62.5 | 112.5 |
| 5 | Judy | F | 14 | 64.3 | 90.0 |
| 6 | Philip | M | 16 | 72.0 | 150.0 |
| 7 | William | M | 15 | 66.5 | 112.0 |

## TEMP= LIBNAME ステートメントオプション

プライマリディレクトリの一時サブディレクトリにライブラリを格納するよう指定します。

**デフォルト:** NO

**エンジン:** SPD Engine のみ

### 構文

**TEMP**=YES | NO

### *必須引数*

**YES**
一時サブディレクトリを作成する指定です。

**NO**
一時サブディレクトリを作成しない指定です。

## 詳細

TEMP=オプションでは、LIBNAME ステートメントで指定されたプライマリディレクトリの一時サブディレクトリを作成します。そのサブディレクトリとそこに含まれていたすべてのファイルがセッションの終わりに削除されます。

SAS オプション USER=と一緒に TEMP=を使用して一時ディレクトリを作成し、単一レベル名で参照できる中間データセットを格納できます。

*注:* SAS/CONNECT ソフトウェアで SIGNON ステートメントを使用する場合、INHERITLIB=オプションでは、TEMP=オプションで定義された SPD Engine ライブラリを参照できません。

## 例: 一時ライブラリの作成

次の例では 2 つの機能を説明します。

- TEMP= LIBNAME オプションを使用して一時ライブラリを作成します。
- USER=システムオプションの使用によって、SPD Engine テーブルのための単一レベルテーブル名が使用可能になります。

**mydata** の下にディレクトリが作成されます。MASTERCOPY テーブルには、mydata に格納されているメタデータファイルがあります。MASTERCOPY のためのデータとインデックスは、DATAPATH=オプションおよび INDEXPATH=オプションで指定した位置に作成されます。

```
libname perm <masterdata>
libname mywork spde 'mydata'
datapath=('/data01/mypath' '/data02/mypath' '/data03/mypath' '/data04/mypath')
indexpath=('index/mypath') TEMP=YES;
option user=mywork;
data mastercopy (index=(lastname));
set perm.customer;
where region='W';
run;
```

*4 章*

# SPD Engine データセットオプション



## SPD Engine データセットオプションについて

SPD Engine のデータセットオプションの指定は、デフォルト Base SAS Engine または SAS/ACCESS Engine のデータセットオプションの指定と同じです。このセクションでは、SPD Engine でだけ使用されるデータセットオプションに関する詳細を提供します。SPD Engine に影響があるデフォルト Base SAS Engine データセットオプションもリストしています。

オプションを使用する際、データセットオプションが同じ名前の LIBNAME オプションの後続に使用される場合には、データセットオプションの値が優先されることに注意してください。

## 構文

(*option1=value1* <(option2=value2>...)

SAS データセット名の後に、データセットオプションをかっこで囲んで指定します。複数のデータセットオプションを指定するには、スペースで区切ります。

## SPD Engine データセットオプションのリスト

ASYNCINDEX=
SPD Engine データセットに複数のインデックスを作成する際、インデックスを並列作成するように指定します。

BYNOEQUALS=
BY 変数に同一の値を持っているデータセットオブザベーションのインデックス出力順序を指定します。

BYSORT=
BY ステートメントが検出されたとき、SPD Engine が自動並べ替えを行なうように指定します。

COMPRESS=
ディスク上の SPD Engine データセットを圧縮します。

*注:* SPD Engine では、圧縮と暗号化は相互排他的です。

ENCRYPT=
データファイルを暗号化します。

*注:* SPD Engine では、圧縮と暗号化は相互排他的です。

ENDOBS=
ユーザー定義のオブザベーション処理範囲における最後のオブザベーション番号を指定します。

IDXBY=
SPD Engine での BY ステートメント処理時にインデックスを使用するかどうかを指定します。

IDXWHERE=
SPD Engine での WHERE 式の処理時にインデックスを使用するように指定します。

IOBLOCKSIZE=
圧縮されるオブザベーションのブロックサイズを指定します。

LISTFILES=
CONTENTS プロシジャで、SPD Engine データセットのコンポーネントファイルすべての完全パス名をリストするかどうかを指定します。

PADCOMPRESS=
OUTPUT または UPDATE モードで開かれるデータセットの圧縮ブロックに追加するバイト数を指定します。

PARTSIZE=
データコンポーネントファイルの最大のパーティションサイズを指定します。
PARTSIZE=は LIBNAME オプションでもあります。

STARTOBS=
ユーザー定義のオブザベーション処理範囲における最初のオブザベーション番号を指定します。

SYNCADD=
一度に 1 つのオブザベーションを処理するのか、それとも一度にオブザベーションをブロックで処理するのかを指定します。

THREADNUM=
SPD Engine 処理に使用するスレッドの最大数を指定します。

UNIQUESAVE=
SYNCADD=NO の場合で、固有のインデックスを持つデータセットに追加あるいは挿入の際に、固有キー値ではないという理由で拒否された任意のオブザベーションを(別のファイルに)保管するために指定します。

WHERENOINDEX=
WHERE 式を評価するときに除外するインデックスのリストを指定します。

## SPD Engine とデフォルト Base SAS Engine で動作が異なる SAS データセットオプション

CNTLLEV=
値 MEM だけが使用できます。

COMPRESS=
ユーザー提供の値は使用できません。

***注意:***
**SPD Engine では、圧縮と暗号化は相互排他的です。** デフォルト Base SAS Engine データセットを SPD Engine データセットにコピーしようとするとき、そのデータセットが圧縮かつ暗号化されていた場合、圧縮は解除されます。SPD Engine データセットを作成する際に暗号化と圧縮を両方とも行うことはできません。

ENCRYPT=
データファイルを暗号化します。

***注意:***
**SPD Engine では、圧縮と暗号化は相互排他的です。**

## SPD Engine ではサポートされない SAS データセットオプション

- BUFNO=
- BUFSIZE=
- DLDMGACTION=
- ENCODING=
- FIRSTOBS=
- GENMAX=

- GENNUM=
- IDXNAME=
- OUTREP=
- POINTOBS=
- REUSE=
- TOBSNO=

# ディクショナリ

## ASYNCINDEX=データセットオプション

SPD Engine データセットに複数のインデックスを作成する際、インデックスを並列作成するように指定します。

**該当要素:** DATA ステップおよび PROC ステップ

**デフォルト:** NO

**エンジン:** SPD Engine のみ

### 構文

**ASYNCINDEX**=YES|NO

#### *必須引数*

**YES**
インデックスを並列作成します(非同期)。

**NO**
一度に 1 つのインデックスを作成します(同期)。

### 詳細

SPD Engine では、1 つのデータセットに対して複数のインデックスを同時に作成できます。SPD Engine では、作成されたインデックスごとに 1 つずつスレッドが起動され、そのスレッドが同時に処理されます。インデックスは一度に 1 つずつ作成するよりも並列作成する方がずっと迅速に処理できますが、このオプションのデフォルトは NO です。並列作成では追加ユーティリティ作業領域および追加メモリが必要になりますが、それが使用不可能な場合があります。リソース不十分のためインデックス作成が失敗した場合は、次のいずれかを実行します。

- SAS システムオプションを MEMSIZE=0 に設定 [1]
- SPDEUTILLOC=システムオプションを使用してユーティリティファイル領域のサイズを増加

SPDEINDEXSORTSIZE=システムオプションを使用して、インデックスの並べ替えに使用されるメモリ領域を増やします。インデックスの並列作成を指定する場合は、SPDEUTILLOC=システムオプションを使用して十分な容量の領域を指定します。

[1] HP Integrity サーバーの OpenVMS の場合は、ページングファイル割り当て(PGFLQUO)を増やしてください。z/OS の場合は、REGION サイズを増やします。

### 例: グループ単位でのインデックス作成

DATASETS プロシジャには柔軟性があるので、複数の MODIFY グループを使用したバッチ並列インデックス作成を行えます。一度にすべてのインデックスを作成すると大量の領域が必要になりますが、そのかわりに、次の例に示すようにグループ単位でインデックスを作成できます。

```
proc datasets lib=main;
modify patients(asyncindex=yes);
index create PatientNo PatientClass;
run;
modify patients(asyncindex=yes);
index create LastName FirstName;
run;
modify patients(asyncindex=no);
index create FullName=(LastName FirstName)
ClassSex=(PatientClass PatientSex);
run;
quit;
```

## BYNOEQUALS=データセットオプション

BY 変数に対する同一値があるデータセットオブザベーションの出力順序がデータセット順序になることを保証するかどうかを指定します。

**該当要素:** DATA ステップおよび PROC ステップ

**使用要素:** BYSORT=YES データセットオプション

**デフォルト:** NO

**エンジン:** SPD Engine のみ

### 構文

**BYNOEQUALS**=YES | NO

#### *必須引数*

**YES**
BY 変数に対する同一値があるデータセットオブザベーションの出力順序がデータセット順序になることを保証しません。

**NO**
BY 変数に対する同一値があるデータセットオブザベーションの出力順序がデータセット順序になることを保証します。

### 詳細

BYNOEQUALS=NO なので、BY ステートメントに対して同一値があるオブザベーションのグループが出力の場合、出力でのオブザベーションの順序はデータセット順序と同じになります。YES を指定すると、処理時間は短縮されますが、オブザベーションがデータセット順序で出力される保証はありません。

BYNOEQUALS=オプションは BYSORT=NO の場合は無効になるので、データセットまたは LIBNAME オプション BYSORT=は YES (デフォルト)にする必要があります。

次の表は、SPD Engine でどのような場合に出力の物理的な順序が保持されるかを示しています。

*表 4.1 SPD Engines での物理的な順序の保持*

| 条件: | データセット順序保持の有無 |
|---|---|
| BY が存在する場合 | あり(デフォルトで BYNOEQUALS=NO および BYSORT=YES) |
| BY が存在し、BYNOEQUALS=YES の場合 | なし |
| BY が存在し、BYSORT=NO の場合 | あり(自動並べ替えが発生しないため) |
| BY も WHERE も存在しない場合 | あり |
| WHERE が存在する場合 | なし |

## 例

### *例 1: BYNOEQUALS=YES*

次の例では、BYNOEQUALS=YES のため、キー変数に同一の BY 値があるオブザベーションが予測不能な順序で出力されます。

```
title 'With BYNOEQUALS=YES';
proc print data=tempdata.housreps(bynoequals=yes);
by state;
where state in ('CA' 'TX');
run;
```

出力は次のとおりです。

**アウトプット 4.1** BYNOEQUALS=YES

With BYNOEQUALS=YES

State=CA

| Obs | Representative | District |
|---|---|---|
| 1124 | Baca, Joe | 42d |
| 1125 | Becerra, Xavier | 30th |
| 1126 | Berman, Howard L. | 26th |
| 1127 | Bono, Mary | 44th |
| 1128 | Calvert, Ken | 43d |
| 1129 | Capps, Lois | 22d |
| 1130 | Condit, Gary A. | 18th |
| 1131 | Cox, Christopher | 47th |
| 1132 | Cunningham, Randy "Duke" | 51 |

State=TX

| Obs | Representative | District |
|---|---|---|
| 1190 | Armey, Richard K. | 26th |
| 1191 | Barton, Joe | 6th |
| 1192 | Bentsen, Ken | 25th |
| 1193 | Bonilla, Henry | 23d |
| 1194 | Brady, Kevin | 8th |
| 1195 | Combest, Larry | 19th |
| 1196 | Culberson, John Abney | 7th |

### 例 2: BYNOEQUALS=NO

次の例は、BYNOEQUALS=NO の出力を示しています。

```
title 'With BYNOEQUALS=NO';

proc print data=tempdata.housreps(bynoequals=no);
by state;
where state in ('CA' 'TX');
run;
```

出力は次のとおりです。

*アウトプット 4.2* BYNOEQUALS=NO

**With BYNOEQUALS=NO**

State=CA

| Obs | Representative | District |
|---|---|---|
| 1124 | Baca, Joe | 42d |
| 1125 | Becerra, Xavier | 30th |
| 1126 | Berman, Howard L. | 26th |
| 1127 | Bono, Mary | 44th |
| 1128 | Calvert, Ken | 43d |
| 1129 | Capps, Lois | 22d |
| 1130 | Condit, Gary A. | 18th |
| 1131 | Cox, Christopher | 47th |
| 1132 | Cunningham, Randy "Duke" | 51 |

State=TX

| Obs | Representative | District |
|---|---|---|
| 1190 | Armey, Richard K. | 26th |
| 1191 | Barton, Joe | 6th |
| 1192 | Bentsen, Ken | 25th |
| 1193 | Bonilla, Henry | 23d |
| 1194 | Brady, Kevin | 8th |
| 1195 | Combest, Larry | 19th |
| 1196 | Culberson, John Abney | 7th |

## BYSORT=データセットオプション

BY ステートメントの検出時に SPD Engine で自動並べ替えを実行することを指定します。

**該当要素:** DATA ステップおよび PROC ステップ

**デフォルト:** YES

**操作:** BYNOEQUALS=

**エンジン:** SPD Engine のみ

## 構文

**BYSORT**=YES | NO

### *必須引数*

**YES**
BY ステートメントの前に PROC SORT を呼び出すかわりに、BY ステートメントを検出するたびに BY 変数に基づいて自動的にデータを並べ替えるように指定します。

**NO**
データを BY 変数に基づいて並べ替えないように指定します。NO を指定する場合は、BY ステートメントの前にすでにデータが並べ替えられている必要があります。インデックスは使用されません。

## 詳細

デフォルト Base SAS Engine を使用した DATA または PROC ステップ処理では、インデックスがない場合やオブザベーションが順序どおりではない場合、BY ステートメントが発行される前にデータセットを並べ替える必要があります。それに対して、SPD Engine では、オブザベーションが順序どおりではない場合、デフォルトで、アプリケーションに返されるデータが並べ替えられます。並べ替えたデータセットが新規作成される PROC SORT とは異なり、SPD Engine の自動並べ替えでは、保存用データセットは変更されず、データセットは新規作成されません。ただし、ユーティリティファイル領域は使用されます。詳細については、“SPDEUTILLOC=システムオプション” (89 ページ)を参照してください。

デフォルトは BYSORT=YES です。BYSORT=YES 引数によって自動並べ替えが有効になり、オブザベーションは BY グループ順序で出力されます。データセットオプション BYNOEQUALS=YES の場合、グループ内のオブザベーションは、データセットでの順序とは異なる順序で出力される可能性があります。データセット順序を保持するには BYNOEQUALS=NO を設定します。

BYSORT=NO 引数の場合は、指定した BY 変数ですでにデータが並べ替えられている必要があります。前の並べ替えで PROC SORT を使用したか、または BY 変数順序でデータセットが作成されていると、このような結果になる可能性があります。BYSORT=NO の場合、グループ化されたデータはデータセット順序でアプリケーションに配信されます。インデックスは、BY 変数順序のオブザベーションの取得には使用されません。BYSORT=NO の場合、データセットオプション BYNOEQUALS=は無効になります。

LIBNAME ステートメントで BYSORT=オプションを指定する場合、PROC ステップまたは DATA ステップでの BYSORT=の指定によって上書きされる可能性があります。したがって、LIBNAME ステートメントで BYSORT=NO を設定し、その後に BY ステートメントを記述します。(事前に PROC SORT を使用するか、または並べ替え順序で作成されたために)データが並べ替え済みの場合以外は、エラーが発生します。入力モードや更新モードで開くために、LIBNAME ステートメントの BYSORT=NO を上書きするには、DATA ステップまたは PROC ステップで BYSORT=YES を設定します。重要な点は、BYSORT=NO によって、エンジンがデータの並べ替え処理をしないように指示されることです。

MSGLEVEL=I SAS システムオプションを設定した場合、BYSORT=YES および IDXWHERE=データセットオプションを使用すると、次のメッセージが SAS ログに書き込まれます。

- IDXWHERE=YES で BY 変数にインデックスがある場合、テーブルの行の並べ替えにそのインデックスが使用されます。次のメッセージが SAS ログに書き込まれます。

```
Note: BY ordering was produced by using an index for table tablename.
```

- IDXWHERE=NO または IDXWHERE=YES で、BY 変数にインデックスがない場合、SPD Engine で自動並べ替えが実行され、テーブルの行が並べ替えられます。次のメッセージが SAS ログに書き込まれます。

```
Note: BY ordering was produced by performing an automatic sort on
table tablename.
```

## 例

### *例 1: BYSORT=YES を使用するデフォルトのグループフォーマット*

```
libname growth spde 'D:\SchoolAge';
data growth.teens;
input Name $ Sex $ Age Height Weight;
datalines;
Alfred M 14 69.0 112.5
Carol F 14 62.8 102.5
James M 13 57.3 83.0
Janet F 15 62.5 112.5
Judy F 14 64.3 90.0
Philip M 16 72.0 150.0
William M 15 66.5 112.0
;
proc print data=growth.teens; by sex;
run;
```

BYSORT=YES がデフォルトのため、PROC SORT を使用してデータを並べ替えていなくても、エラーは発生しません。

出力は次のとおりです。

*アウトプット 4.3　BYSORT=YES を使用するデフォルトのグループフォーマット*

The SAS System

Sex=F

| Obs | Name | Age | Height | Weight |
|---|---|---|---|---|
| 2 | Carol | 14 | 62.8 | 102.5 |
| 4 | Janet | 15 | 62.5 | 112.5 |
| 5 | Judy | 14 | 64.3 | 90.0 |

Sex=M

| Obs | Name | Age | Height | Weight |
|---|---|---|---|---|
| 1 | Alfred | 14 | 69.0 | 112.5 |
| 3 | James | 13 | 57.3 | 83.0 |
| 6 | Philip | 16 | 72.0 | 150.0 |
| 7 | William | 15 | 66.5 | 112.0 |

### *例 2: BYSORT=NO*

PROC PRINT ステートメントに BYSORT=NO を指定すると、自動並べ替えが抑制されている場合(BYSORT=NO)、必ずエラーが返されます。BY ステートメントの前に BY 変数でデータを並べ替える必要があります(PROC SORT などを使用)。

```
libname growth spde 'D:\SchoolAge';
proc print data=growth.teens (bysort=no);
by sex;
run;
ERROR: Data set GROWTH.TEENS is not sorted in ascending sequence.
The current BY-group has Sex = M and the next BY-group has Sex = F.
NOTE: The SAS System stopped processing this step because of errors.
```

## COMPRESS=データセットオプション

SPD Engine データセットの作成時にディスク上で圧縮するように指定します。

**該当要素:** DATA ステップおよび PROC ステップ

**デフォルト:** NO

**制限事項:** ENCRYPT=YES または ENCRYPT=RC4 と一緒には使用不可

**操作:** 関連データセットオプション: “IOBLOCKSIZE=データセットオプション” (62 ページ)および “PADCOMPRESS=データセットオプション” (66 ページ)

**エンジン:** SPD Engine のみ

## 構文

**COMPRESS**= NO | YES | CHAR | BINARY

### *必須引数*

**NO**
データセットの圧縮は実行しません。

**YES | CHAR**
データセットで Run Length Compression (SPDSRLC2)を実行します。

**BINARY**
データセットで Ross Data Compression (SPDSRDC)を実行します。

## 詳細

COMPRESS=YES|BINARY|CHAR を指定した場合、SPD Engine では、データコンポーネントファイルが、作成時にブロックで圧縮されます。圧縮ブロックのサイズを指定するには、データセットの作成時に“IOBLOCKSIZE=データセットオプション” (62 ページ)を使用します。新しく圧縮したブロックに埋め込みを追加するには、データセットの作成または更新時に“PADCOMPRESS=データセットオプション” (66 ページ)を指定します。詳細については、“SPD Engine データセットの圧縮” (18 ページ)を参照してください。

## 例

### 例 1: COMPRESS=BINARY

**アウトプット 4.4** COMPRESS=BINARY オプションの使用

**The SAS System**

The CONTENTS Procedure

| | | | |
|---|---|---|---|
| Data Set Name | TEMPDATA.HOUSREPS | Observations | 1205 |
| Member Type | DATA | Variables | 3 |
| Engine | SPDE | Indexes | 0 |
| Created | Wed, Dec 15, 2010 03:31:56 PM | Observation Length | 53 |
| Last Modified | Wed, Dec 15, 2010 03:31:56 PM | Deleted Observations | 0 |
| Protection | | Compressed | BINARY |
| Data Set Type | | Point to Observations | YES |
| Label | | Sorted | NO |
| Data Representation | WINDOWS_32 | | |
| Encoding | wlatin1 Western (Windows) | | |

| Engine/Host Dependent Information | |
|---|---|
| Blocking Factor (obs/block) | 1236 |
| Disk Compression Name | SPDSRDC |
| Data Partsize | 134225892 |
| - Compressed Info | - |
| Number of compressed blocks | 2 |
| Raw data blocksize | 32754 |
| Number of blocks with overflow | 0 |
| Max overflow chain length | 0 |
| Block number for max chain | 0 |
| Min overflow area | 0 |
| Max overflow area | 0 |

Alphabetic List of Variables and Attributes

| # | Variable | Type | Len | Format | Informat | Label |
|---|---|---|---|---|---|---|
| 3 | District | Char | 21 | $21. | $21. | District |
| 1 | Representative | Char | 29 | $29. | $29. | Representative |
| 2 | State | Char | 3 | $3. | $3. | State |

### 例 2: COMPRESS=CHAR

**アウトプット 4.5** COMPRESS=CHAR オプションの使用

## The SAS System

### The CONTENTS Procedure

| | | | |
|---|---|---|---|
| Data Set Name | TEMPDATA.HOUSREPS | Observations | 1205 |
| Member Type | DATA | Variables | 3 |
| Engine | SPDE | Indexes | 0 |
| Created | Wed, Dec 15, 2010 03:30:03 PM | Observation Length | 53 |
| Last Modified | Wed, Dec 15, 2010 03:30:03 PM | Deleted Observations | 0 |
| Protection | | Compressed | CHAR |
| Data Set Type | | Point to Observations | YES |
| Label | | Sorted | NO |
| Data Representation | WINDOWS_32 | | |
| Encoding | wlatin1 Western (Windows) | | |

| Engine/Host Dependent Information | |
|---|---|
| Blocking Factor (obs/block) | 1236 |
| Disk Compression Name | SPDSRLC2 |
| Data Partsize | 134225892 |
| - Compressed Info | - |
| Number of compressed blocks | 2 |
| Raw data blocksize | 32754 |
| Number of blocks with overflow | 0 |
| Max overflow chain length | 0 |
| Block number for max chain | 0 |
| Min overflow area | 0 |
| Max overflow area | 0 |

Alphabetic List of Variables and Attributes

| # | Variable | Type | Len | Format | Informat | Label |
|---|---|---|---|---|---|---|
| 3 | District | Char | 21 | $21. | $21. | District |
| 1 | Representative | Char | 29 | $29. | $29. | Representative |
| 2 | State | Char | 3 | $3. | $3. | State |

# ENCRYPT=データセットオプション

出力 SPD Engine データセットを暗号化するかどうかを指定します。

**該当要素:** DATA ステップおよび PROC ステップ

**制限事項:** 出力データセットにのみ使用
ENCRYPT=YES は COMPRESS=と一緒には使用不可

**エンジン:** SPD Engine のみ

## 構文

**ENCRYPT**=YES | NO

### *構文の説明*

**YES**
ファイルを暗号化します。この暗号化方法では、データセットに保存されたパスワードが使用されます。ENCRYPT=YES を指定した場合、少なくとも READ=または PW=データセットオプションを同時に指定する必要があります。この暗号化方法ではパスワードが使用されるため、暗号化されたデータセット上のパスワードは、データセットを再作成せずに変更することはできません。

***注意:***
**ENCRYPT=YES の場合はすべてのパスワードを記録してください。** パスワードを忘れた場合、パスワードをリセットするには SAS のサポートが必要です。このプロセスには非常に多くの時間とリソースが必要になります。

**NO**
ファイルを暗号化しません。

## 詳細

SPD Engine では、暗号化と圧縮は相互に排他的です。

SPD Engine データセットを作成する際に暗号化と圧縮を両方とも行うことはできません。ENCRYPT=YES データセットオプションおよび COMPRESS=データセットまたは LIBNAME オプションを使用すると、次のエラーが発生します。

```
ERROR: The data set was not compressed because compression and
encryption cannot both be specified.
```

圧縮と暗号化が行われた Base SAS データセットを SPD Engine データセットにコピーすると、圧縮が解除されます。SAS では、圧縮のかわりにデータセットのセキュリティが保持されます。

## 例: データセットの暗号化

次の例では、データセットを暗号化します。

```
libname depta spde '/datasecret';
data salary(encrypt=yes read=green);
input name $ yrsal bonuspct;
datalines;
Muriel 34567 3.2
```

```
Bjorn 74644 2.5
Freda 38755 4.1
Benny 29855 3.5
Agnetha 70998 4.1
;
```

このデータセットを使用するには、READ パスワードを指定します。

```
proc contents data=salary(read=green);
run;
```

## ENDOBS=データセットオプション

ユーザー定義のオブザベーション処理範囲における終了オブザベーション番号を指定します。

**該当要素:** DATA ステップおよび PROC ステップ

**使用要素:** STARTOBS=データセットオプション

**デフォルト:** データセット内の最後のオブザベーション

**制限事項:** ENDOBS=は入力データセットにのみ使用
OBS=システムおよびデータセットオプション、または FIRSTOBS=システムおよびデータセットオプションと一緒には使用不可

**エンジン:** SPD Engine のみ

### 構文

**ENDOBS=***n*

#### *必須引数*

***n***
終了オブザベーションの番号。

### 詳細

#### *オブザベーション範囲の指定*

デフォルトでは、SPD Engine で、STARTOBS=オプションまたは ENDOBS=オプションを使用してオブザベーションの範囲を指定しない限り、データセット全体のすべてのオブザベーションが処理されます。STARTOBS=オプションを ENDOBS=オプションなしで使用した場合、ENDOBS=の暗黙値はデータセットの終わりになります。両方のオプションを一緒に使用する場合、ENDOBS=値を STARTOBS=値よりも大きくする必要があります。

SPD Engine の ENDOBS=データセットオプションは、WHERE 式で指定した場合を除いて、デフォルト Base SAS Engine の OBS=データセットオプションと同様に機能します。

#### *WHERE 式での ENDOBS=の使用*

ENDOBS=を WHERE 式で使用する場合、ENDOBS=値は、返されるオブザベーションの数ではなく、最後に処理するオブザベーションを表します。次の例でその違いを示します。

*注:* OBS=システムオプションおよび OBS=データセットオプションは、STARTOBS=もしくは ENDOBS=データセットまたは LIBNAME オプションと一緒には使用できません。

## 例

### *例 1: SPD Engine での ENDOBS=*

SPD Engine で ENDOBS=5 を指定してデータセットを作成および処理します。オブザベーション番号 5 で終わるデータセットに WHERE 式が適用されます。PRINT プロシジャでは 4 つのオブザベーションが印刷されます。これは WHERE 式に適合するオブザベーションです。

```
libname growth spde 'c:\SchoolAge';
data growth.teens;
input Name $ Sex $ Age Height Weight;
list;
datalines;
Alfred M 14 69.0 112.5
Carol F 14 62.8 102.5
James M 13 57.3 83.0
Janet F 15 62.5 112.5
Judy F 14 64.3 90.0
Philip M 16 72.0 150.0
Zeke M 14 71.1 105.0
Alice F 14 65.1 91.0
William M 15 66.5 112.0
;
proc print data=growth.teens (endobs=5);
where age >13;
title 'WHERE age > 13 using SPD Engine';
run;
```

出力は次のとおりです。

***アウトプット 4.6*** *印刷された 4 つのオブザベーション*

**WHERE age > 13 using SPD Engine**

| Obs | Name | Sex | Age | Height | Weight |
|---|---|---|---|---|---|
| 1 | Alfred | M | 14 | 69.0 | 112.5 |
| 2 | Carol | F | 14 | 62.8 | 102.5 |
| 4 | Janet | F | 15 | 62.5 | 112.5 |
| 5 | Judy | F | 14 | 64.3 | 90.0 |

### *例 2: SPD Engine での OBS=*

OBS=5 を指定して同じデータセットを処理します。PROC PRINT では 5 つのオブザベーションが印刷されます。これは WHERE 式に適合するすべてのオブザベーションで、5 番目の適合オブザベーションで終了します。

```
libname growth spde 'c:\SchoolAge';
data growth.teens;
input Name $ Sex $ Age Height Weight;
list;
datalines;
Alfred M 14 69.0 112.5
Carol F 14 62.8 102.5
James M 13 57.3 83.0
Janet F 15 62.5 112.5
Judy F 14 64.3 90.0
Philip M 16 72.0 150.0
Zeke M 14 71.1 105.1
Alice F 14 65.1 91.0
William M 15 66.5 112.0
;
proc print data=growth.teens (obs=5);
where age >13;
title 'WHERE age > 13 using V9';
run;
```

***アウトプット 4.7*** *印刷された 5 つのオブザベーション*

**WHERE age > 13 using V9**

| Obs | Name | Sex | Age | Height | Weight |
|---|---|---|---|---|---|
| 1 | Alfred | M | 14 | 69.0 | 112.5 |
| 2 | Carol | F | 14 | 62.8 | 102.5 |
| 4 | Janet | F | 15 | 62.5 | 112.5 |
| 5 | Judy | F | 14 | 64.3 | 90.0 |
| 6 | Philip | M | 16 | 72.0 | 150.0 |

## IDXBY=データセットオプション

SPD Engine での BY ステートメント処理時にインデックスを使用するかどうかを指定します。

**該当要素:** DATA ステップおよび PROC ステップ

**デフォルト:** YES

**エンジン:** SPD Engine のみ

## 構文

**IDXBY**=YES | NO

### *必須引数*

**YES**
BY ステートメントのインデックス変数を処理するときにインデックスを使用します。

*注:* BY ステートメントに 2 つ以上の変数または DESCENDING オプションが指定されている場合は、IDXBY=YES でもインデックスは使用されません。

**NO**
BY ステートメントのインデックス変数を処理するときにインデックスを使用しません。

注 IDXBY=NO の場合、BY ステートメントの処理時に自動並べ替えが実行されます。

## 詳細

IDXBY=データセットオプションを使用する場合は、BYSORT=YES オプションを使用し、BY 変数にインデックスを付けるようにしてください。

場合によっては、データを自動的に並べ替えると、SPD Engine のパフォーマンスが向上することがあります。自動並べ替えを使用するには、BYSORT=YES を設定し、IDXBY=NO を指定する必要があります。

SAS システムオプション MSGLEVEL=I を設定し、BY 処理情報が SAS ログに書き込まれるようにします。IDXBY=データセットオプションおよび BYSORT=YES オプションを使用すると、SAS ログに次のメッセージが書き込まれます。

- IDXBY=YES で BY 変数にインデックスがある場合、テーブルの行の並べ替えにそのインデックスが使用されます。次のメッセージが SAS ログに書き込まれます。

```
 NOTE: BY ordering was produced by using an index for
table tablename.
```

- IDXBY=NO の場合、次のメッセージが SAS ログに書き込まれます。

```
NOTE: BY ordering was produced by performing an automatic sort
on table tablename.
```

## 例

### *例 1: IDXBY=NO データセットオプションの使用*

```
options msglevel=i;
proc means data=permdata.customer(IDXBY=no);
by sales;
by state;
run;
```

次のメッセージが SAS ログに書き込まれます。

```
NOTE: BY ordering was produced by performing an automatic sort
on table PERMDATA.customer.
NOTE: There were 2981 observations read from the data set
PERMDATA.CUSTOMER.
```

### *例 2: IDXBY=YES データセットオプションの使用*

```
proc means data=permdata.customer(IDXBY=yes);
var sales;
by state;
run;
```

次のメッセージが SAS ログに書き込まれます。

```
NOTE: BY ordering was produced by using an index for table
PERMDATA.customer.
NOTE: There were 2981 observations read from the data set
PERMDATA.CUSTOMER.
```

## IDXWHERE=データセットオプション

SPD Engine での WHERE 式の処理時にインデックスを使用するように指定します。

**該当要素:** DATA ステップおよび PROC ステップ

**デフォルト:** YES

**制限事項:** WHERENOINDEX=オプションは IDXWHERE=NO オプションと一緒には使用不可

**エンジン:** SPD Engine のみ

### 構文

**IDXWHERE**=YES | NO

#### *必須引数*

**YES**
WHERE 式の処理時にインデックスを使用します。

**NO**
WHERE 式の処理時にインデックスを無視します。

**制限事項** IDXWHERE=NO オプションと WHERENOINDEX=オプションを一緒に使用することはできません。

### 詳細

IDXWHERE=は、WHINIT と呼ばれる、SPD Engine の WHERE 式計画ソフトウェアで使用されます。WHINIT では、さまざまなアプリケーションにおける WHERE 処理でのインデックス使用のパフォーマンスがテストされます。SAS システムオプション MSGLEVEL=I を設定し、WHERE 処理情報が SAS ログに出力されるようにします。

IDXWHERE=データセットオプションおよび BYSORT=YES オプションを使用すると、SAS ログに次のメッセージが書き込まれます。

- IDXWHERE=YES で BY 変数にインデックスがある場合、テーブルの行の並べ替えにそのインデックスが使用されます。次のメッセージが SAS ログに書き込まれます。

  ```
  Note: BY ordering was produced by using an index for
  table tablename.
  ```

- IDXWHERE=NO または IDXWHERE=YES で、BY 変数にインデックスがない場合、SPD Engine で自動並べ替えが実行され、テーブルの行が並べ替えられます。次のメッセージが SAS ログに書き込まれます。

  ```
  Note: BY ordering was produced by performing an
  automatic sort on table tablename.
  ```

SPD Engine では、4 つの WHERE 式評価方法がサポートされます。詳細については、“SPDEWHEVAL=システムオプション” (91 ページ)を参照してください。方法 1、3 および 4 では、使用可能なインデックスが使用され、WHERE 式のインデックス付き部分が実行されます。評価方法 2 では、WHERE 式の非インデックス付き部分が実行されます。

最初の例では、IDXWHERE=NO が指定されたため、WHERE 式で評価方法 2 が使用されます。2 番目の例では、IDXWHERE=YES が指定されたため、評価方法 1 が使用されました。

## 例: WHINIT ログ出力(MSGLEVEL=I)

***ログ 4.1*** *IDXWHERE=NO*

```
34 options msglevel=i;
35 proc means data=permdata.customer(idxwhere=no);
36 var sales;
37 where state="CA";
38 run;
whinit: WHERE (sstate='CA')
whinit returns: ALL EVAL2
NOTE: There were 2981 observations read from the data set
PERMDATA.CUSTOMER. WHERE state='CA';
```

***ログ 4.2*** *IDXWHERE=YES*

```
39 proc means data=permdata.customer(idxwhere=yes);
40 var sales;
41 where state="CA";
42 run;
whinit: WHERE (sstate='CA')
--
whinit: SBM-INDEX STATE uses 45% of segs (WITHIN maxsegratio 75%)
whinit returns: ALL EVAL1(w/SEGLIST)
NOTE: There were 2981 observations read from the data set
PERMDATA.CUSTOMER. WHERE state='CA';
```

*注: WHERE と BY ステートメントの両方を組み合わせて使用する場合は、インデックスの使用を任意で抑制しないでください。*両方を使用して、WHERE 式でオブザベーションのフィルタ処理、BY 式でオブザベーションの並べ替えを行う場合、WHERE 式に適合する、フィルタ処理されたオブザベーションが、並列 WHERE 式評価の一部として並べ替えステップに直接取り込まれます。結果として、最終的に並べ替えられたオブザベーションセットが作成されます。WHERE 処理でのインデックス使用によって、並べ替えステップに取り込まれるフィルタ処理のパフォーマンスが大幅に向上します。

## IOBLOCKSIZE=データセットオプション

I/O 操作で使用するオブザベーションブロックのサイズをバイト単位で指定します。

**該当要素:** DATA ステップおよび PROC ステップ

**デフォルト:** 32,768 バイト

**エンジン:** SPD Engine のみ

### 構文

**IOBLOCKSIZE=***n*

#### *必須引数*

***n***
オブザベーションブロックのバイト単位でのサイズです。

### 詳細

SPD Engine では、メモリ内のブロックを使用して、データコンポーネントファイルでの読み書き対象のオブザベーションを収集します。IOBLOCKSIZE=では、これらのブロックのサイズが指定されます。(実際のサイズは、*n* バイトの指定サイズに収まるオブザベーション最大数に対応するように計算されます。したがって、実際のサイズはオブザベーションの長さの倍数になります。)

IOBLOCKSIZE=で指定したブロックサイズは圧縮データセットにも適用されます。

いったんデータセットを作成すると、ブロックサイズは変更できません。ブロックのサイズ変更を行うには、データセットを新しいデータセットにコピーし、IOBLOCKSIZE=を出力データセットに適したブロックサイズに設定します。

デフォルト値および IOBLOCKSIZE=最小値は 32,768 バイトです。アクセス対象データを補足する IOBLOCKSIZE=値を指定します。ランダムに分散したデータへのアクセスでは、ブロックサイズが小さい方が有利です(32,768 バイトなど)。これは大きいブロックにアクセスするよりも小さいブロックにアクセスする方が処理が迅速になるためです。それとは対照的に、均等もしくは順番に分散されたデータ、またはデータセット全体のスキャンが必要なデータへのアクセスでは、ブロックサイズを大きくする必要があります(131,072 バイトなど)。

### 例: IOBLOCKSIZE=の使用

```
/*IOBLOCKSIZE set to 64K */
data sport.maillist(ioblocksize=65536);
/*IOBLOCKSIZE set to 32K */
data sport.maillist(ioblocksize=32768 compress=yes);
```

## LISTFILES=データセットオプション

CONTENTS プロシジャで、SPD Engine データセットのコンポーネントファイルすべての完全パス名をリストするかどうかを指定します。

**該当要素:** PROC CONTENTS のみ

**デフォルト:** NO

**エンジン:** SPD Engine のみ

## 構文

**LISTFILES**=YES | NO

### *必須引数*

**YES**
SPD Engine データセットのコンポーネントファイルすべての完全パス名をリストします。

**NO**
パス名をリストしません。

## 詳細

LISTFILES=データセットオプションは SPD Engine および CONTENTS プロシジャでのみ使用され、SPD エンジンデータセットのコンポーネントファイルすべての完全パス名をリストします。

## 例: LISTFILES オプション

```
proc contents data=hrdept.names (listfiles=yes);
```

次の CONTENTS プロシジャ出力には、すべてのコンポーネントファイルの完全パス名が表示されています。

**アウトプット 4.8** CONTENTS プロシジャー出力セクション 1

**The SAS System**

**The CONTENTS Procedure**

| | | | |
|---|---|---|---|
| Data Set Name | COMPANY.DEPTS | Observations | 285120 |
| Member Type | DATA | Variables | 8 |
| Engine | SPDE | Indexes | 1 |
| Created | Tuesday, December 14, 2010 04:41:29 PM | Observation Length | 152 |
| Last Modified | Tuesday, December 14, 2010 04:47:15 PM | Deleted Observations | 0 |
| Protection | | Compressed | NO |
| Data Set Type | | Sorted | NO |
| Label | | | |
| Data Representation | WINDOWS_64 | | |
| Encoding | wlatin1 Western (Windows) | | |

**アウトプット 4.9** CONTENTS プロシジャー出力セクション 2

| Engine/Host Dependent Information | |
|---|---|
| Blocking Factor (obs/block) | 431 |
| Data Partsize | 10481920 |
| - Alphabetic List of Index Info | - |
| Index | JOB1 |
| KeyValue (Min) | ACCOUNTANT |
| KeyValue (Max) | TRANSLATOR |
| Number of discrete values | 29 |
| - Metadata Files | - |
| d:\main\depts.mdf.0.0.0.spds9 | - |
| - Data Files | - |
| d:\data\depts.dpf.03bf03f7.0.161.spds9 | - |
| d:\data\depts.dpf.03bf03f7.1.161.spds9 | - |
| d:\data\depts.dpf.03bf03f7.2.161.spds9 | - |
| d:\data\depts.dpf.03bf03f7.3.161.spds9 | - |
| d:\data\depts.dpf.03bf03f7.4.161.spds9 | - |
| - Index Files | - |
| d:\indexes\depts.idxjob1.03bf03f7.0.161.spds9 | - |
| d:\indexes\depts.hbxjob1.03bf03f7.0.161.spds9 | - |

***アウトプット 4.10*** *CONTENTS プロシジャー出力セクション 3*

**Alphabetic List of Variables and Attributes**

| # | Variable | Type | Len |
|---|---|---|---|
| 4 | DEPTHEAD | Char | 15 |
| 7 | JOB1 | Char | 15 |
| 2 | LEVEL1 | Char | 16 |
| 1 | LEVEL2 | Char | 13 |
| 5 | LEVEL3 | Char | 20 |
| 6 | LEVEL4 | Char | 30 |
| 3 | LEVEL5 | Char | 30 |
| 8 | N | Num | 8 |

**Alphabetic List of Indexes and Attributes**

| # | Index | # of Unique Values |
|---|---|---|
| 1 | JOB1 | 29 |

## PADCOMPRESS=データセットオプション

OUTPUT または UPDATE モードで開かれるデータセットの圧縮ブロックに追加するバイト数を指定します。

**該当要素:** DATA ステップおよび PROC ステップ

**デフォルト:** 0

**操作:** データセットオプション関連: “COMPRESS=データセットオプション” (51 ページ)および “IOBLOCKSIZE=データセットオプション” (62 ページ)

**エンジン:** SPD Engine のみ

### 構文

**PADCOMPRESS=** *n*

#### *必須引数*

***n***
追加するバイト数です。

## 詳細

圧縮 SPD Engine データセットはディスク上で複数ブロックの領域を占有します。ブロックのサイズは、データセットの作成時に指定された IOBLOCKSIZE=データセットオプションから得られます。データセットが更新されると、その更新を保持するために新しいブロックフラグメントの作成が必要になる可能性があります。更新を重ねると新しいフラグメントが作成される可能性があり、その結果、データセットの読み取りに必要な I/O 操作の回数が増加します。

データセットへの更新が多いことが予想される状況では、ブロック埋め込みを増加させることによって断片化を最小限に抑えられます。ただし、データセットを更新しない場合は、埋め込みの追加によって領域を無駄に消費する可能性があります。

すべての圧縮ブロックの埋め込みコストと、一部の圧縮ブロックで発生の可能性がある断片化のコストとを比較検討する必要があります。

データセットの作成または更新時に PADCOMPRESS=データセットオプションを指定すると、ブロックをディスクに書き込み直す際、すべてのブロックに領域が追加されます。PADCOMPRESS=設定は、データセットのメタデータでは保持されません。

# PARTSIZE=データセットオプション

データコンポーネントパーティションに指定可能な最大サイズ(MB、GB または TB 単位)を指定します。値は SPD Engine データセットの作成時に指定されます。このサイズは固定サイズです。この指定はデータコンポーネントファイルにのみ適用されます。

**該当要素:** DATA ステップおよび PROC ステップ

**使用要素:** MINPARTSIZE=システムオプション

**デフォルト:** 128M

**操作:** DATAPATH=

**エンジン:** SPD Engine のみ

## 構文

**PARTSIZE=***n* | *n*M | *n*G | *n*T

### *必須引数*

***n* | *n*M | *n*G | *n*T**
MB、GB または TB でのパーティションのサイズです。M、G または T なしで *n* を指定した場合のデフォルトは MB です。たとえば、PARTSIZE=128 と PARTSIZE=128M は同じ意味です。最大値は、8,796,093,022,207MB です。

**制限事項** この制限は、オペレーティングシステム z/OS、Linux SLES 9 x86 および Windows ファミリの 32 ビットホストにのみ適用されます。SAS 9.3 でパーティションサイズが 2GB 以上のデータセットを作成した場合、SAS 9.2 以前のいずれのバージョンの SPD Engine でもそのデータセットを開くことはできません。次のエラーメッセージが SAS ログに書き込まれます。**`ERROR: Unable to open data file because its data representation differs from the SAS session data representation.`**

## 詳細

複数のパーティションでは、データを並列処理で読み取る必要があります。オプション PARTSIZE=では、SPD Engine データファイルに指定サイズでパーティションが切られます。実際のパーティションサイズは、*n*MB、GB または TB の指定サイズに収まるオブザベーション最大数に対応するように計算されます。オブザベーションの長さが 65K より大きいテーブルの場合、指定した PARTSIZE=と実際のパーティションサイズが一致しない可能性があります。この数字を一致させるには、PARTSIZE=に、32 およびオブザベーションの長さの倍数を指定します。

SPD Engine データセットのデータ部分を固定サイズのファイルに分割(パーティション)することによって、ソフトウェアで一部の操作に高度なスケーラビリティが導入されます。SPD Engine では、スレッドを並列起動できます(たとえば、WHERE 評価に対して 1 パーティションにつき 1 スレッドまで)。また、別々のデータパーティションによって、スレッド間のファイルアクセス競合のオーバーヘッドなしでデータを処理することも可能になります。各パーティションが 1 ファイルであるため、小さなパーティションサイズの代償として、オブザベーションを格納するためにファイル(UNIX i ノードなど)の数の増加が必要になります。

PARTSIZE=を使用したスケーラビリティ制限は、DATAPATH=オプションで指定したファイルシステムを、どのように構成し、どのように複数のストライプボリュームに分散させるのかによって決まります。(個々のボリュームのストライピング構成を、ディスクアレイの複数のディスクコントローラまたは SCSI チャネルに分散させる必要があります。)ハードウェアレベルでの構成の目標は、データ取得時の並列処理を最大限にすることです。ディスクストライピングの詳細については、**`http://support.sas.com/rnd/scalability`** の Scalability and Performance で“SPD Engine”の“I/O Setup and Validation”を参照してください。

PARTSIZE=の指定は SPD Engine システムオプション MINPARTSIZE=によって制限されます。このシステムオプションは通常システム管理者によって保守されます。MINPARTSIZE=を使用すると、経験不足のユーザーが任意に小さなパーティションを作成し、その結果、多数のデータファイルが生成されるようなことは起こらなくなります。

パーティションサイズによって、データセット全体のスキャンを必要とする並列操作の多くの作業単位が決定されます。ただし、パーティションの増加が必ずしも処理の迅速化を意味するとは限りません。この折り合いをつけるには、データセットの格納に必要な物理ファイル(パーティション)の増加数と、パーティションの増加によって並列処理が可能になる作業量とのバランスをとる必要があります。パーティションが多くなると、データセットを処理するために開くファイルも多くなりますが、各パーティションのオブザベーション数は少なくなります。一般的なルールでは、パーティションは、1 データパスにつき 10 以下、1CPU につき 3 から 4 までとなります。(一部のオペレーティングシステムでは、使用可能な開いたファイルの数に制限がかかることもあります。)

新しい SPD Engine データセットの適切なパーティションサイズを決定するには、次の点に注意する必要があります。

- データに対して実行するアプリケーションの種類
- データ量はどの程度か
- アプリケーションで使用可能な CPU はいくつあるか
- パーティションの格納に使用可能なディスクはどれか
- そのディスクと CPU との関係

たとえば、各 CPU の制御するディスクが 1 つだけの場合、適切なパーティションサイズは、各ディスクにほぼ同じ量のデータが含まれるサイズです。各 CPU の制御するディスクが 2 つの場合、適切なパーティションサイズは、負荷のバランスがとれているサイズです。各 CPU で、ほぼ同じ量の作業が行われます。

*注:* データセットの PARTSIZE=値は、データセットの作成後は変更できません。PARTSIZE=を変更するには、データセットを再作成し、LIBNAME ステートメント内か、またはその新しい(出力)データセットに対して、異なる PARTSIZE=値を指定する必要があります。

### 例: PROC SQL の使用

100GB のデータと 8 つのディスクがあり、1 ディスクにつき 12.5GB を格納できるとします。パーティションは 1 ディスクにつき 3 つから 4 つまでが最適です。3.125GB のパーティションサイズが適切です。そのため、PARTSIZE=3200M を指定します。

```
data salecent.sw (partsize=3200m);
```

データ量は同じで、1 年以内にデータ量が 2 倍になることが予想されるとします。その場合、同じ PARTSIZE=を指定して 1 ディスクにつきおよそ 7 パーティションにするか、または PARTSIZE=を 5000M に増やして 1 ディスクにつき 5 パーティションにします。

## STARTOBS=データセットオプション

ユーザー定義のオブザベーション処理範囲における開始オブザベーション番号を指定します。

**該当要素:** DATA ステップおよび PROC ステップ

**デフォルト:** データセット内の最初のオブザベーション

**制限事項:** STARTOBS=は入力データセットにのみ使用

OBS=システムおよびデータセットオプション、または FIRSTOBS=システムおよびデータセットオプションと一緒には使用不可

**エンジン:** SPD Engine のみ

### 構文

**STARTOBS=***n*

#### *必須引数*

***n***
開始オブザベーションの番号。

### 詳細

#### *オブザベーション範囲の指定*

デフォルトでは、SPD Engine で、STARTOBS=オプションと ENDOBS=オプションを使用してオブザベーションの範囲を指定しない限り、データセット全体のすべてのオブザベーションが処理されます。ENDOBS=オプションを STARTOBS=オプションなしで使用した場合、STARTOBS=の暗黙値は 1 になります。両方のオプションを一緒に使用する場合、STARTOBS=値を ENDOBS=値よりも小さくする必要があります。

SPD Engine の STARTOBS=データセットオプションは、WHERE 式で指定した場合を除いて、デフォルト Base SAS Engine の FIRSTOBS= SAS データセットオプションと同様に機能します。

*注:* FIRSTOBS= SAS データセットオプションは、SPD Engine ではサポートされていません。OBS=システムオプションおよび OBS=データセットオプションは、

STARTOBS=もしくは ENDOBS=データセットまたは LIBNAME オプションと一緒には使用できません。

### *WHERE 式での STARTOBS=の使用*

STARTOBS=を WHERE 式で使用する場合、STARTOBS=値は、WHERE 式を適用する最初のオブザベーションを表します。この値をデフォルト Base SAS Engine データセットオプション FIRSTOBS=と比較します。FIRSTOBS=では、WHERE 式に適合するデータのサブセット内の開始オブザベーション番号が指定されます。

## 例

### *例 1: SPD Engine での STARTOBS=*

SPD Engine で STARTOBS=5 を指定してデータセットを作成および処理します。オブザベーション番号 5 で始まるデータセットに WHERE 式が適用されます。PRINT プロシジャでは 6 つのオブザベーションが印刷されます。これは WHERE 式に適合するオブザベーションです。

```
libname growth spde 'c:\temp';
data growth.teens;
input Name $ Sex $ Age Height Weight;
list;
datalines;
Alfred M 14 69.0 112.5
Carol F 14 62.8 102.5
James M 13 57.3 83.0
Janet F 15 62.5 112.5
Judy F 14 64.3 90.0
Philip M 16 72.0 150.0
Zeke M 14 71.1 105.1
Alice F 14 65.1 91.0
William M 15 66.5 112.0
Mike M 16 67.0 105.1
;
proc print data=growth.teens (startobs=5);
where age >13;
title 'WHERE age>13 using SPD Engine';
run;
```

*アウトプット 4.11　印刷された 6 つのオブザベーション*

**WHERE age > 13 using SPD Engine**

| Obs | Name | Sex | Age | Height | Weight |
|---|---|---|---|---|---|
| 5 | Judy | F | 14 | 64.3 | 90.0 |
| 6 | Philip | M | 16 | 72.0 | 150.0 |
| 7 | Zeke | M | 14 | 71.1 | 105.1 |
| 8 | Alice | F | 14 | 65.1 | 91.0 |
| 9 | William | M | 15 | 66.5 | 112.0 |
| 10 | Mike | M | 16 | 67.0 | 105.1 |

### *例 2: デフォルト Base SAS Engine での FIRSTOBS=*

デフォルト Base SAS Engine で FIRSTOBS=5 を指定して同じデータセットを処理します。PROC PRINT では 5 つのオブザベーションが印刷されます。これは WHERE 式に適合するすべてのオブザベーションで、5 番目の適合オブザベーションから開始されます。FIRSTOBS=は、SPD Engine ではサポートされていません。

```
libname growth v9 'c:\temp';
data growth.teens;
input Name $ Sex $ Age Height Weight;
list;
datalines;
Alfred M 14 69.0 112.5
Carol F 14 62.8 102.5
James M 13 57.3 83.0
Janet F 15 62.5 112.5
Judy F 14 64.3 90.0
Philip M 16 72.0 150.0
Zeke M 14 71.1 105.1
Alice F 14 65.1 91.0
William M 15 66.5 112.0
Mike M 16 67.0 105.1
;
proc print data=growth.teens (firstobs=5);
where age >13;
title 'WHERE age>13 using the V9 Engine';
run;
```

***アウトプット 4.12*** *印刷された 5 つのオブザベーション*

WHERE age > 13 using the V9 Engine

| Obs | Name | Sex | Age | Height | Weight |
|---|---|---|---|---|---|
| 6 | Philip | M | 16 | 72.0 | 150.0 |
| 7 | Zeke | M | 14 | 71.1 | 105.1 |
| 8 | Alice | F | 14 | 65.1 | 91.0 |
| 9 | William | M | 15 | 66.5 | 112.0 |
| 10 | Mike | M | 16 | 67.0 | 105.1 |

## SYNCADD=データセットオプション

一度に 1 つのオブザベーションを処理するのか、それとも一度に複数のオブザベーションを処理するのかを指定します。

**該当要素:** PROC SQL

**デフォルト:** NO

**操作:** UNIQUESAVE=

**エンジン:** SPD Engine のみ

### 構文

**SYNCADD**=YES|NO

#### *必須引数*

**YES**
一度に 1 つのオブザベーションを処理します(同期)。

**NO**
一度に複数のオブザベーションを処理します(非同期)。

### 詳細

SYNCADD=YES の場合、オブザベーションは一度に 1 つずつ処理されます。PROC SQL では、重複しないインデックスが付いたデータセットにオブザベーションを追加した結果、SPD Engine で重複する値を含むオブザベーションが検出された場合、次の処理が発生します。

- 追加操作を停止
- 直前に追加したトランザクションをすべて取り消し
- ディスク上の元のデータセットは変更なし

SYNCADD=NO の場合、オブザベーションがブロック単位で追加されます(パイプライン処理)。通常はこちらの方が迅速です。重複しないインデックスが付いたデータセットにオブザベーションを追加した結果、SPD Engine で重複インデックス値を含むオブザベーションが検出された場合、次の処理が発生します。

- SPD Engine でオブザベーションを拒否
- SPD Engine で処理を続行
- 追加操作または挿入操作の終了時にのみステータスコードを発行

拒否されたオブザベーションを別のデータセットに保存するには、UNIQUESAVE=データセットオプションを YES に設定します。

## 例: SQL プロシジャを使用した重複しない複合インデックスの作成

次の例では、UQ01A と UQ01B という 2 つのデータセットが作成されます。UQ01A では、PROC SQL によって、重複しない複合インデックスが作成され、SYNCADD=NO(ブロックデータの挿入)で新しい値がデータセットに挿入されます。UNIQUESAVE=が YES に設定されているため、重複値は別のファイルに格納されます。

次に、PROC SQL によって、UQ01B で重複しない複合インデックスが作成され、SYNCADD=YES で新しい値が挿入されます。PROC SQL は重複値が検出されると停止し、データセットを復元します。UNIQUESAVE=YES でも、その指定は無視されます。SAS ログは次のとおりです。

```
1097 libname userfile spde 'c:\temp';
NOTE: Libref SPDS USERFILE was successfully assigned as follows:
Engine: SPD Engine
Physical Name: d3727.na.sas.com:528c:\temp\
1098
1099 data uq01a uq01b;
1100 input z $ 1-20 x y;
1101 list;
1102 datalines;
RULE:----+----1----+----2----+----3----+----4----+----5----+----6----+----7
1103 one 1 10
1104 two 2 20
1105 three 3 30
1106 four 4 40
1107 five 5 50
NOTE: The data set USER.UQ01A has 5 observations and 3 variables.
NOTE: The data set USER.UQ01B has 5 observations and 3 variables.
NOTE: DATA statement used (Total process time):
real time 0.51 seconds
cpu time 0.06 seconds
1108 ;
1109
1110
1111 proc sql sortseq=ascii exec noerrorstop;
1112 create unique index comp
1113 on uq01a (x, y);
NOTE: Composite index comp has been defined.
1114 insert into uq01a(syncadd=no,uniquesave=yes)
1115 values('rollback1', -80, -80)
1116 values('rollback2',-90, -90)
```

```
1117 values('nonunique', 2, 20)
1118 ;
NOTE: 3 observations were inserted into USER.UQ01A.
WARNING: Duplicate values not allowed on index comp for file USER.UQ01A.
(Occurred 1 times.)
NOTE: Duplicate records have been stored in file USER._D2DAAF7.
NOTE: PROCEDURE SQL used (Total process time):
real time 0.99 seconds
cpu time 0.05 seconds
1119 proc sql sortseq=ascii exec noerrorstop;
1120 create unique index comp
1121 on uq01b (x, y);
NOTE: Composite index comp has been defined.
1122 insert into uq01b(syncadd=yes,uniquesave=yes)
1123 set z='rollback3', x=-60, y=-60
1124 set z='rollback4', x=-70, y=-70
1125 set z='nonunique', x=2, y=20;
ERROR: Duplicate values not allowed on index comp for file UQ01B.
NOTE: Deleting the successful inserts before error noted above to restore
data set to a consistent state.
1126
NOTE: PROCEDURE SQL used (Total process time):
real time 0.26 seconds
cpu time 0.17 seconds
1127 proc compare data=uq01a compare=uq01b;run;
NOTE: There were 7 observations read from the data set USER.UQ01A.
NOTE: There were 5 observations read from the data set USER.UQ01B.
NOTE: PROCEDURE COMPARE used (Total process time):
real time 0.51 seconds
cpu time 0.05 seconds
```

## THREADNUM=データセットオプション

SPD Engine データセットの処理のために SPD Engine で起動可能な I/O スレッド最大数を指定します。

**該当要素:** DATA ステップおよび PROC ステップ

**デフォルト:** SPDEMAXTHREADS=を設定する場合はその値。設定しない場合、デフォルトはコンピュータ上の CPU 数の 2 倍。

**操作:** SPDEMAXTHREADS=

**エンジン:** SPD Engine のみ

### 構文

**THREADNUM=***n*

#### *必須引数*

***n***
スレッド数を指定します。

## 詳細

THREADNUM=では、SPD Engine データセットの処理のために SPD Engine で起動される I/O スレッドの最大数を指定できます。THREADNUM=値は、次の SPD Engine I/O 処理のいずれにも適用されます。

- WHERE 式の処理
- 並列処理でのインデックス作成
- スレッド対応アプリケーションで要求される I/O

THREADNUM=の調整によって、システム管理者は、SPD Engine で任意のプロセスに使用可能な CPU リソースのレベルを調整できます。たとえば、64 ビットプロセッサシステムで THREADNUM=4 を設定すると、プロセスが多くとも 4CPU までに制限されるので、他のユーザーまたはアプリケーションのスループットを高められます。

THREADNUM=が 1 より大きい場合は、並列処理が発生する可能性が高くなります。したがって、物理的な順序が出力では保持されない場合があります。

また、このオプションを使用すると、WHERE 式評価のスケーラビリティも探索できます。

設定可能なシステムオプションである SPDEMAXTHREADS=では、システムリソースの消費に上限が課されるので、THREADNUM=値が制約されます。

*注:* SAS システムオプション NOTHREADS は SPD Engine には影響しません。

*注:* THREADNUM=1 を設定すると、並列処理は発生しなくなります。これはデフォルト Base SAS Engine と一致した動作です。

## 例: %MACRO の使用

SPD Engine システムオプション SPDEMAXTHREADS=がセッションに対して 128 に設定されます。次の例では、SAS マクロで並列処理の効果が示されます。

```
%macro dotest(maxthr);
%do nthr=1 %to &maxthr;
data _null_;
set spde cen.precs(threadnum= &nthr);
where occup= '022'
and state in('37','03','06','36');
run;
%mend dotest;
```

# UNIQUESAVE=データセットオプション

重複しないインデックスが付いたデータセットにオブザ ベーションを追加または挿入するときに、重複するキー値を含むオブザベーション(拒否されたオブザベーション)を別のデータセットに保存するように指定します。

**該当要素:** PROC APPEND および PROC SQL

**使用要素:** SPDSUSDS 自動マクロ変数

**デフォルト:** NO

**操作:** SYNCADD=NO

**エンジン:** SPD Engine のみ

## 構文

**UNIQUESAVE**=YES|NO

### *必須引数*

**YES**
SYNCADD=NO の場合、拒否されたオブザベーションがシステム作成された別のデータセットに書き込まれます。このデータセットにアクセスするには、マクロ変数 SPDSUSDS を参照します。

**NO**
拒否されたオブザベーションは別のデータセットに書き込まれません。

## 詳細

インデックスが重複しないデータセットにオブザベーションを追加するときに、データセットオプション SYNCADD=NO が設定されている場合は、UNIQUESAVE=YES を使用します。

SYNCADD=NO では、追加操作または挿入操作で、オブザベーションを一度に 1 つずつではなくブロック単位で処理(パイプライン処理)するように指定されます。すべてのオブザベーションがデータセットに適用されて初めて、重複インデックス値が検出されます。UNIQUESAVE=YES の場合、拒否されたオブザベーションは別のデータセットに保存され、その名前が SPD Engine マクロ変数 SPDSUSDS に格納されます。データセット名のかわりにそのマクロ変数を指定して、拒否されたオブザベーションを識別できます。

*注:* SYNCADD=YES の場合、UNIQUESAVE=オプションは無視されます。詳細については、SYNCADD=データセットオプションを参照してください。

## 例: APPEND プロシジャでの UNIQUESAVE=オプションの使用

次の例では、変数 NAME に重複しないインデックスが付いた 2 つのデータセットを作成し、PROC APPEND で UNIQUESAVE=YES を指定して一緒に追加します。SAS ログは次のとおりです。

```
1 libname employee spde 'c:\temp';
NOTE: Libref EMPLOYEE was successfully assigned as follows:
Engine: SPD Engine
Physical Name: c:\temp\
2 data employee.emp1 (index=(name/unique));
3 input name $ exten;
4 list; datalines;
RULE:----+----1----+----2----+----3----+----4----+----5----+----6----+
5 Jill 4344
6 Jack 5589
7 Jim 8888
8 Sam 3334
NOTE: The data set EMPLOYEE.EMP1 has 4 observations and 2 variables.
NOTE: DATA statement used (Total process time):
real time 9.98 seconds
cpu time 1.28 seconds
9 run;
10 data employee.emp2 (index=(name/unique));
11 input name $ exten;
12 list; datalines;
```

```
RULE:----+----1----+----2----+----3----+----4----+----5----+----6----+
13 Jack 4443
14 Ann 8438
15 Sam 3334
16 Susan 5321
17 Donna 3332
NOTE: The data set EMPLOYEE.EMP2 has 5 observations and 2 variables.
NOTE: DATA statement used (Total process time):
real time 0.04 seconds
cpu time 0.04 seconds
18 run;
19 proc append data=employee.emp2 base=employee.emp1
20 (syncadd=no uniquesave=yes);
21 run;
NOTE: Appending EMPLOYEE.EMP2 to EMPLOYEE.EMP1.
NOTE: There were 5 observations read from the data set EMPLOYEE.EMP2.
NOTE: 3 observations added.
NOTE: The data set EMPLOYEE.EMP1 has 7 observations and 2 variables.
WARNING: Duplicate values not allowed on index name for file
EMPLOYEE.EMP1. (Occurred 2 times.)
NOTE: Duplicate records have been stored in file EMPLOYEE._D3596FF.
NOTE: PROCEDURE APPEND used (Total process time):
real time 6.25 seconds
cpu time 1.26 seconds
22 proc print data=employee.emp1;
23 title 'Listing of Final Data Set';
24 run;
NOTE: There were 7 observations read from the data set EMPLOYEE.EMP1.
NOTE: PROCEDURE PRINT used (Total process time):
real time 2.09 seconds
cpu time 0.40 seconds
25
26 proc print data=&spdsusds;
27 title 'Listing of Rejected observations';
28 run;
NOTE: There were 2 observations read from the data set EMPLOYEE._D3596FF.
NOTE: PROCEDURE PRINT used (Total process time):
real time 0.01 seconds
cpu time 0.01 seconds
```

***アウトプット 4.13*** *UNIQUESAVE=YES*

**Listing of Final Data Set**

| Obs | name | exten |
|---|---|---|
| 1 | Jack | 4443 |
| 2 | Ann | 8438 |
| 3 | Sam | 3334 |
| 4 | Susan | 5321 |
| 5 | Donna | 3332 |

***アウトプット 4.14*** *拒否されたオブザベーション*

**Listing of Rejected Observations**

| Obs | name | exten | XXX00000 |
|---|---|---|---|
| 1 | Jack | 4443 | name |
| 2 | Ann | 8438 | name |
| 3 | Sam | 3334 | name |
| 4 | Susan | 5321 | name |
| 5 | Donna | 3332 | name |

## WHERENOINDEX=データセットオプション

WHERE 式の評価時に取り除くインデックスのリストを指定します。

**該当要素:** DATA ステップおよび PROC ステップ

**デフォルト:** 空白

**制限事項:** IDXWHERE=NO データセットオプションと一緒には使用不可

**エンジン:** SPD Engine のみ

### 構文

**WHERENOINDEX**=(*name1 name2...*)

### *必須引数*

**(*name1 name2...*)**
WHERE プランナから取り除くインデックス名のリスト。

## 例: インデックスを取り除く

データセット PRECS がインデックスと一緒に定義されます。

```
proc datasets lib=spde cen
modify precs;
index create stser=(state serialno) occind=(occup industry) hour89;
quit;
```

次のクエリの評価時に、SPD Engine では、STATE 変数でも HOUR89 変数でもインデックスは使用されません。

この場合、OCCUP 変数と INDUSTRY 変数の条件の AND 組み合わせでは、出力対象は非常に少なくなります。条件を満たすオブザベーションは少数です。クエリの完全インデックス評価に必要とされる余分なインデックス I/O(コンピュータ時間)を回避するには、次の SAS コードを使用します。

```
proc sql;
create data set hr80spde
as select state, age, sex, hour89, industry, occup from spde cen.precs
(wherenoindex=(stser hour89))
where occup='022'
and state in('37','03','06','36')
and industry='012'
and hour89 > 40;
quit;
```

*注:* WHERENOINDEX リストでは、変数名ではなく、インデックス名を指定します。前述の例では、STATE 変数の複合インデックス STSER と単一インデックス HOUR89 の両方が取り除かれます。

*5 章*

# SPD Engine システムオプション



## SPD Engine システムオプションについて

SAS システムオプションは、SAS セッションに影響を与える指示です。SAS システムの初期化、ハードウェアやソフトウェアのインターフェイス、ジョブと SAS ファイルの入力、処理および出力などの、SAS が操作を実施する方法を制御します。SPD Engine システムオプションは SAS システムオプションと同じ方法で機能します。このセクションでは、SPD Engine でだけ使用されるシステムオプション、および SPD Engine では異なる動作をする Base SAS システムオプションについて説明します。

## 構文

OPTIONS *option1* <*...option-n*>;

*option*
変更する 1 つ以上の SPD Engine システムオプションを指定します。

次の例では、SPD Engine システムオプション MAXSEGRATIO=を指定します。

```
options maxsegratio=50;
```

*動作環境情報* コマンドラインまたは構成ファイルでは、動作環境に固有の構文を使用します。詳細については、動作環境に関する SAS のドキュメントを参照してください。

## SPD Engine システムオプションのリスト

COMPRESS=
SPD Engine データセットの作成時にディスク上で圧縮するように指定します。

MAXSEGRATIO=
WHERE 式を処理する前に、候補セグメントとして識別するインデックスセグメントのパーセンテージを制御します。これは、インデックス付き変数を含む WHERE 式を評価するときに起こります。

MINPARTSIZE=
SPD Engine データセットの作成時に使用する最少パーティションサイズを指定します。

SPDEINDEXSORTSIZE=
インデックスの作成で値を並べ替えるとき、並べ替えユーティリティが使用できるメモリ領域のサイズを指定します。

SPDEMAXTHREADS=
SPD Engine が I/O 処理のために起動できるスレッドの最大数を指定します。

SPDESORTSIZE=
SPD Engine によって使用される並べ替え処理操作に必要なメモリ領域サイズを指定します。

SPDEUTILLOC=
SPD Engine がユーティリティファイルを一時格納できる 1 つ以上のファイルシステムの場所を指定します。

SPDEWHEVAL=
どのオブザベーションが WHERE 式の条件を満たすかを判断するプロセスを指定します。

## SPD Engine では動作が異なる SAS システムオプション

MSGLEVEL=
値 I は、WHINIT のプランナ出力を可能にします。

MSGLEVEL=I
SAS ログに WHERE 最適化情報を出力します。

COMPRESS=
ユーザー定義の圧縮は実施できません。

DLCREATEDIR
SPD Engine では機能しません。

DLDMGACTION=
SPD Engine には影響を与えません。SPD Engine データセットが損傷した場合、システムバックアップファイルから復元する必要があります。

FIRSTOBS=
SPD Engine では使用できません。

VALIDMEMNAME=
VALIDMEMNAME=EXTEND を使用する場合は、メンバ名に次の制約があります。

- *class.group など、ピリオド付きのメンバ名は使用不可*
- *$class など、$から始まるメンバ名は使用不可*

VALIDVARNAME=
変数名に次の特殊文字のいずれかが含まれている場合は、インデックスまたは複合インデックスをその変数には作成できません。

" * | \ : / < > ? -

# ディクショナリ

## COMPRESS=システムオプション

SPD Engine データセットの作成時にディスク上で圧縮するように指定します。

**該当要素:** 構成ファイル、SAS 起動時、OPTIONS ステートメント、システムオプションウィンドウ

**デフォルト:** NO

**制限事項:** ENCRYPT=YES と一緒には使用不可

**エンジン:** SPD Engine のみ

### 構文

**COMPRESS**= NO | YES | CHAR | BINARY

#### *必須引数*

**NO**
データセットの圧縮は実行しません。

**YES | CHAR**
データセットで Run Length Compression (SPDSRLC2)を実行します。

**BINARY**
データセットで Ross Data Compression (SPDSRDC)を実行します。

### 詳細

COMPRESS=YES|BINARY|CHAR を指定した場合、SPD Engine では、データコンポーネントファイルが、作成時にブロックで圧縮されます。圧縮ブロックのサイズを指定するには、データセットの作成時に“IOBLOCKSIZE=データセットオプション” (62 ページ)を使用します。新しく圧縮したブロックに埋め込みを追加するには、データセットの作成または更新時に“PADCOMPRESS=データセットオプション” (66 ページ)を指定します。詳細については、“SPD Engine データセットの圧縮” (18 ページ)を参照してください。

SPD Engine では、ユーザー指定の圧縮はサポートされません。圧縮と暗号化の両方が行われたデフォルト Base SAS Engine データセットを移行する場合、暗号化は保持されますが、圧縮は解除されます。

CONTENTS プロシジャでは、圧縮についての情報が印刷されます。次の例では、CONTENTS プロシジャ出力の圧縮情報フィールドについて説明します。

***アウトプット 5.1*** *PROC CONTENTS 圧縮情報セクション*

| - Compressed Info | - |
|---|---|
| Number of compressed blocks | 202 |
| Raw data blocksize | 32736 |
| Number of blocks with overflow | 5 |
| Max overflow chain length | 3 |
| Block number for max chain | 80 |
| Min overflow area | 87 |
| Max overflow area | 181 |

圧縮ブロック数
　データの格納に必要な圧縮ブロック数。

生データブロックサイズ
　IOBLOCKSIZE=データセットオプションに指定したサイズから計算されたバイト単位の圧縮ブロックサイズ。

オーバーフローのあるブロック数
　より多くの領域を必要とする圧縮ブロック数。データが更新されて、圧縮された新ブロックが圧縮された旧ブロックよりも大きくなった場合、オーバーフローブロックフラグメントが作成されます。

オーバーフローチェーンの最大長
　単一ブロックのオーバーフロー最大数。たとえば、圧縮ブロックが更新されて大きくなり、再度更新されてより大きなサイズになった場合、オーバーフローチェーンの最大長は 2 になります。

最大チェーンのブロック数
　最大数のオーバーフローブロックを含むブロック数。

最小オーバーフロー領域
　オーバーフローに必要なディスク領域最小容量。

最大オーバーフロー領域
　オーバーフローに必要なディスク領域最大容量。

## MAXSEGRATIO=システムオプション

WHERE 式を処理する前に、候補セグメントとして識別されるインデックスセグメントのパーセンテージを制御します。これは、インデックス付き変数を含む WHERE 式を評価するときに起こります。

**該当要素:** 構成ファイル、SAS 起動時、OPTIONS ステートメント、システムオプションウィンドウ

**デフォルト:** 75

**エンジン:** SPD Engine のみ

## 構文

**MAXSEGRATIO=***n*

### *必須引数*

***n***

SPD Engine が、WHERE 式で参照される値を含むと識別するインデックスセグメントのパーセンテージの上限を指定します。デフォルトは 75 で、SPD Engine が次を実施することを指定します。

- インデックスを使用して、特定の WHERE 式の値を含んでいるセグメントを識別します。
- 全セグメントの 75%より多くがその値を含むことが判明した場合、候補セグメントの識別を停止します。

有効な値の範囲は 0 と 100 の間の整数です。n=0 の場合、SPD Engine は候補セグメントを識別しようとせず、かわりに、すべてのセグメントへ WHERE 式を適用します。n=100 の場合は、SPD Engine が 100%のセグメントをチェックして候補セグメントを識別し、次に、それらの候補セグメントにだけ WHERE 式を適用します。

## 詳細

インデックス付き変数での WHERE 照会では、SPD Engine は、WHERE 式の 1 つ以上の条件と一致する 1 つ以上の変数値を含むインデックスセグメントの数を判別します。WHERE 式がその WHERE 式を満たすオブザベーションを含んでいるセグメントにのみ適用される場合、実質的パフォーマンスの向上が実現する場合がしばしばあります。

SPD Engine は、MAXSEGRATIO=の値を使用して、すべてのセグメントに WHERE 式を適用するコストが、候補セグメントを識別し続けるコストより少ない点がどこかを決定します。計算された比率が MAXSEGRATIO=で指定した比率を超過したとき、SPD Engine は候補セグメントの識別を停止し、すべてのセグメントへ WHERE 式を適用します。

*注:* テーブルによっては、75%が最適な設定にならない場合もあります。よりよい設定を決定するためには、パフォーマンスベンチマークを実行してパーセンテージを調整し、再度パフォーマンスベンチマークを実行してください。結果を比較することで、照会している特定のデータ母集団がインデックスセグメント比率の変化にどのように応答するかがわかります。

## 例

### *例 1: インデックスセグメントの識別*

次の例では、SPD Engine は WHERE 式を満たすインデックスセグメントの識別を開始し、セグメントの総数と比較して、識別されたセグメントのパーセンテージが 65 を超過するまで実施します。パーセンテージが 65 を超えた場合、SPD Engine は候補セグメントの識別を停止し、すべてのセグメントへ WHERE 式を適用します。

```
options maxsegratio=65;
```

### 例 2: WHERE 式の全セグメントへの適用

次の例では、SPD Engine がいずれの候補セグメントも最初から識別せず、WHERE 式を全セグメントへ適用します。

```
options maxsegratio=0;
```

### 例 3: 全インデックスセグメントの無停止評価

次の例では、SPD Engine がインデックスセグメントの識別を開始し、全セグメントが評価されるまで停止しません。その後で、識別されたすべての候補セグメントに WHERE 式が適用されます:。

```
options maxsegratio=100;
```

## MINPARTSIZE=システムオプション

データコンポーネントパーティションの最小サイズを指定します。値は SPD Engine データセットの作成時に指定されます。

**該当要素:** 構成ファイル、SAS 起動時

**デフォルト:** 16M

**エンジン:** SPD Engine のみ

### 構文

**MINPARTSIZE=***n* | *n*K | *n*M | *n*G

#### 必須引数

***n***

パーティションのサイズは、バイト、KB、MB、GB のいずれかです。最小パーティションサイズの最大値は 2GB–1、すなわち 2047MB です。

**制限事項** この制限は、オペレーティングシステム z/OS、Linux SLES 9 x86 および Windows ファミリの 32 ビットホストにのみ適用されます。SAS 9.3 でパーティションサイズが 2GB 以上のデータセットを作成した場合、SAS 9.2 以前のいずれのバージョンの SPD Engine でもそのデータセットを開くことはできません。次のエラーメッセージが SAS ログに書き込まれます。**ERROR: Unable to open data file because its data representation differs from the SAS session data representation.**

### 詳細

MINPARTSIZE=の指定によって、PARTSIZE=オプションで指定したパーティションサイズに下限を設定することができます。MINPARTSIZE=の指定は、パーティションがほぼ同数のオブザベーションで作成されるかどうかに影響する場合があります。パーティションサイズが小さい場合は、処理時により多くのファイルがオープンされることを意味します。使用するオペレーティングシステムによっては、使用ファイルのオープン数が制限される可能性があります。

# SPDEINDEXSORTSIZE=システムオプション

インデックスの作成で値を並べ替えるとき、並べ替えユーティリティが使用できるメモリ領域のサイズを指定します。

**該当要素:** 構成ファイル、SAS 起動時、OPTIONS ステートメント、**システムオプション**ウィンドウ

**デフォルト:** 32M

**操作:** MEMSIZE=

**エンジン:** SPD Engine のみ

## 構文

**SPDEINDEXSORTSIZE=***n* | *n*K | *n*M | *n*G

### *必須引数*

***n***

バイト、KB、MB、GB のいずれかでのメモリ領域のサイズです。*n*=0 のとき、並べ替えユーティリティではデフォルトが使用されます。有効な値の範囲は、1,048,576 から 10,736,369,664 バイトまでです。

## 詳細

SPDEINDEXSORTSIZE=オプションは、インデックスの作成で値を並べ替えるときに使用できるメモリーの最大量を指定します。インデックスが(ASYNCINDEX=YES により)並列処理で作成される場合、SPDEINDEXSORTSIZE=に指定される値は、同時に処理されるすべてのインデックス作成スレッド間で分割されます。

メモリ不足のためインデックス作成が失敗した場合は、次のいずれかを実行します。

- SAS システムオプションを MEMSIZE=0 にして SAS を再起動 [1]
- SPDEUTILLOC=システムオプションを使用してユーティリティファイル領域のサイズを増加

SPDEINDEXSORTSIZE=オプションを使用して、インデックスの作成で値を並べ替えるときに使用されるメモリ領域を増やします。インデックスの並列作成を指定する場合は、SPDEUTILLOC=システムオプションを使用して十分な容量の領域を指定します。

最大の SPDEINDEXSORTSIZE=値は 10 GB ですが、2GB までに制限されているホストシステムではこの値は尊重されません。64 ビットの LONG データタイプを持つホストシステムでは、SPD Engine は 2GB より大きな値を有効とします。32 ビットの LONG データタイプを持つホストシステムでは、SPD Engine はメモリのみ 2GB まで使用できます。SPDEINDEXSORTSIZE オプションの値をより大きな値に設定できますが、より大きな値は有効ではありません。

*注:* より大きな値が有効ではないのに使用されている場合には、SAS ログに警告を受け取ります。

1 HP Integrity サーバーの OpenVMS の場合は、ページングファイル割り当て(PGFLQUO)を増やしてください。z/OS の場合は、REGION サイズを増やします。

## SPDEMAXTHREADS=システムオプション

SPD Engine が I/O 処理のために起動できるスレッドの最大数を指定します。

**該当要素:** 構成ファイル、SAS 起動時

**デフォルト:** 0

**エンジン:** SPD Engine のみ

### 構文

**SPDEMAXTHREADS**=*n*

#### *必須引数*

***n***

SPD Engine が起動できるスレッドの最大数です。有効な値の範囲は、0 から 65,536 です。デフォルトサイズは 0 で、THREADNUM=の値が設定されている場合には SPD Engine はそれを使用します。そうでない場合には、起動するスレッドの数を、稼働するコンピュータの CPU 数の 2 倍に相当する数に SPD Engine が設定します。

### 詳細

SPDEMAXTHREADS=の指定によって、SPD Engine 処理のために起動するスレッドの数に上限を設定します。それには次が含まれます。

- WHERE 式の処理
- 並列処理でのインデックス作成
- SAS スレッド対応プロシジャなどのスレッド対応アプリケーションによって要求された任意の I/O 処理

SPDEMAXTHREADS=は、THREADNUM=データセットオプションを抑制します。

## SPDESORTSIZE=システムオプション

SPD Engine によって使用される並べ替え処理操作に必要なメモリ領域サイズを指定します。

**該当要素:** 構成ファイル、SAS 起動時、OPTIONS ステートメント、**システムオプション**ウィンドウ

**デフォルト:** 32M

**エンジン:** SPD Engine のみ

### 構文

**SPDESORTSIZE**=*n* | *n*K | *n*M | *n*G

### *必須引数*

***n***

バイト、KB、MB、GB のいずれかでのメモリ領域のサイズです。*n*=0 のとき、並べ替えユーティリティではデフォルトが使用されます。有効な値の範囲は、1,048,576 から 10,736,369,664 バイトまでです。

## 詳細

SPD Engine は、自動並べ替えを並列処理で実行できます。SPDESORTSIZE=に指定する並べ替えサイズは、並列プロセス数の倍数にする必要があります。この並べ替えサイズの合計は、プロセスで使用できる物理メモリより小さくする必要があります。SPDESORTSIZE=の適切な指定により、動作環境で制御されるメモリのスワップが抑えられ、パフォーマンスが改善される場合があります。

並べ替え処理が指定したよりも多くのメモリを必要とする場合は、次のうちの一つを実施してください。

- SAS システムオプションを MEMSIZE=0 にして SAS を再起動(HP Integrity サーバーでの OpenVMS の場合はページングファイルの割り当て(PGFLQUO)を増やし、z/OS の場合は、REGION サイズを増やす。)
- SPDEUTILLOC=システムオプションを使用して、ユーティリティファイル領域のサイズを増やします。

SPDEINDEXSORTSIZE=オプションを使用して、インデックスの作成で値を並べ替えるときに使用されるメモリを増やします。インデックスの並列作成を指定する場合は、SPDEUTILLOC=システムオプションを使用して十分な容量の領域を指定します。

*注:* デフォルト Base SAS Engine 用ドキュメントに説明されている SORTSIZE=オプションは、PROC SORT 操作に影響します。SPDESORTSIZE=の指定は SPD Engine に特有の並べ替え操作に影響を与えます。

最大の SPDESORTSIZE=値は 10 GB ですが、2GB までに制限されているホストシステムではこの値は尊重されません。64 ビットの LONG データタイプを持つホストシステムでは、SPD Engine は 2GB より大きな値を有効とします。32 ビットの LONG データタイプを持つホストシステムでは、SPD Engine はメモリのみ 2GB まで使用できます。SPDESORTSIZE オプションの値をより大きな値に設定できますが、より大きな値は有効ではありません。

*注:* より大きな値が有効ではないのに使用されている場合には、SAS ログに警告を受け取ります。

# SPDEUTILLOC=システムオプション

SPD Engine がユーティリティファイルを一時格納できる 1 つ以上のファイルシステムの場所を指定します。

**該当要素:** 構成ファイル、SAS 起動時

**エンジン:** SPD Engine のみ

**参照項目:** システムオプションの指定方法は、使用する動作環境に合った SAS ドキュメントに説明されています。

## 構文

**SPDEUTILLOC**=*location* | *(location-1 ...location-n)*

### *必須引数*

***location***
ユーティリティファイルが作成される既存のディレクトリです。

***(location-1 ...location-n)***
ユーティリティファイルが作成される一連の既存のディレクトリです。

注 *Location* は、単一引用符または二重引用符で囲むことができます。*location* にブランクが埋め込まれている場合は引用符で囲む必要があります。

## 詳細

*動作環境の情報*
システムオプションの指定方法は、使用するオペレーティングシステム用 SAS ドキュメントに説明されています。

SPD Engine は、自動並べ替えやインデックス作成などの特定の処理中に一時ユーティリティファイルを作成します。処理を正常に完了させるには、ユーティリティファイルを格納するために十分な領域が必要です。SPDEUTILLOC=システムオプションを使用して、処理のために十分な適当量の領域を指定できます。ただし、HP Integrity サーバーの OpenVMS では、ライブラリは ODS-5 ファイルである必要があります。SPDEUTILLOC=システムオプションに複数のディレクトリが指定されている場合、最初のユーティリティファイルのディレクトリは処理の開始時にランダムに選択されます。周期的な方法で他のディレクトリへ選択が続きます。ユーティリティファイルは一時的で、処理が完了した後に削除されます。

*注:* 構文エラーを回避するためには、構成ファイルで複数のディレクトリを指定してください。

ユーティリティファイルの作成処理に十分な領域を確保するため、SPDEUTILLOC=オプションあるいは UTILLOC=オプションを常に指定することをお勧めます。SPDEUTILLOC=システムオプションあるいは UTILLOC=SAS システムオプションが指定されず、SPD Engine が SAS WORK ディレクトリを検出できない(あるいはそれに対する書き込み許可がない)場合、一時ユーティリティファイル格納のための場所は動作環境ごとに定義されます。次の表はデフォルトのユーティリティファイルの場所を示します。

***表 5.1*** *ユーティリティファイルのデフォルト場所*

| 動作環境 | デフォルト場所 1 | デフォルト場所 2 | デフォルト場所 3 |
|---|---|---|---|
| | | | |
| UNIX | UTILLOC= SAS システムオプション(指定がある場合) | SAS ワークライブラリ | /tmp |
| Windows | UTILLOC= SAS システムオプション(指定がある場合) | SAS ワークライブラリ | TEMP=環境変数で指定された場所 |
| z/OS | UTILLOC= SAS システムオプション(指定がある場合) | SAS ワークライブラリ | /tmp |

| 動作環境 | デフォルト場所 1 | デフォルト場所 2 | デフォルト場所 3 |
|---|---|---|---|
| | | | |
| HP Integrity サーバーの OpenVMS | UTILLOC= SAS システムオプション(指定がある場合) | WORK=SAS システムオプション(指定がある場合)、および ODS-5 ディレクトリ (WORK=SAS システムオプションが ODS-5 ディレクトリを指定していない場合、および SAS セッションが SASROOT の ODS-5 ファイル指定で開始した場合、ユーティリティファイルは SASROOT ディレクトリに作成されます。それ以外の場合は、デフォルトの場所はないので、LIBNAME 割り当てが失敗します。) | sys$scratch: |

## SPDEWHEVAL=システムオプション

どのオブザベーションが WHERE 式の条件を満たすかを判断するプロセスを指定します。

**該当要素:** 構成ファイル、SAS 起動時、OPTIONS ステートメント、システムオプションウィンドウ

**デフォルト:** COST

**エンジン:** SPD Engine のみ

### 構文

**SPDEWHEVAL**=COST | EVAL1 | EVAL3EVAL4

#### *必須引数*

**COST**
WHERE 式を最適化するために使用する評価方法を SPD Engine が決定することを指定します。この処理では、使用するスレッド数を計算します。これにより、十分に活用されなかったスレッドを起動するオーバーヘッドを最少化されます。これがデフォルト設定です。

**EVAL1**
マルチスレッドのインデックス評価方法で、複数のスレッドを使用し、WHERE 式を満たす行を迅速に判断できます。オブザベーションを取得するために起動されるスレッドの数は、THREADNUM=の値と同じです。

**EVAL3EVAL4**
単一スレッドのインデックス評価方法で、すべてのキー変数が単純なインデックスを持ち、かつ非同等性を検査する条件は持たない、単純あるいは複合の WHERE

式に使用されます。オブザベーションの取得にはマルチスレッドが使用される場合もあります。

## 詳細

COST は SPDEWHEVAL=のデフォルト設定で、WHERE 式および任意の使用可能なインデックスを分析します。分析に基づき、SPD Engine が WHERE 式を最適化する評価方法を選択します。評価方法は、EVAL1、EVAL3EVAL4、または、インデックスが使用可能でない場合や、インデックスの使用では処理時間を改善できないという分析が示されている場合には、データを順次読み込む方法が可能です。

COST は、WHERE 式の処理に使用するスレッドの数を最適化します。COST は、効率的に使用できるスレッドの数を決定して起動します。THREADNUM=の値に基づき、COST は活用されていないスレッドを起動しないことで処理時間を大幅に節約できます。

その WHERE 式が他の評価方法基準に厳密に適合する場合以外は、COST が SPDEWHEVAL=のための推奨値です。COST 以外の値がより効率的かどうかを判断するためには、ベンチマークテストの使用を強くお勧めします。

EVAL1 は、WHERE 式が複雑で、変数に複数のインデックスがあるときにはより効率的な場合があります。EVAL1 では、複数のスレッドを起動して、どのセグメントが WHERE 式の条件を満たすかを判断します。また、オブザベーションの取得にも複数のスレッドを使用できます。

*注:* COST が最適でない場合も多少あります。EVAL1 または EVAL3EVAL4 に値を変更することでよりよいパフォーマンスを産めるかどうかは、パフォーマンスベンチマークを実行し、値を変更して再度パフォーマンスベンチマークを実行することで判断できます。結果を比較することで、照会している特定のデータ母集団が、計画中の規則に基づいた WHERE にどのように応答するかがわかります。

# 用語集

**CPU バウンドアプリケーション**
パフォーマンスがデータ計算の実行可能速度に制約されるアプリケーション。複数の CPU とスレッドテクノロジによってこの問題を軽減できます。

**I/O バウンドアプリケーション**
パフォーマンスが処理データの配信可能速度に制約されるアプリケーション。複数の CPU、分割 I/O、スレッドテクノロジ、RAID (redundant array of independent disks)テクノロジ、またはこれらの組み合わせによって、この問題を軽減できます。

**RAID**
多数のディスクから成る記憶システムの一種で、カリフォルニア大学バークレイ校で開発されたインターリーブ記憶技術を実装しています。RAID では複数のレベルを使用できます。たとえば、レベル 0 の RAID では、2 つ以上のハードドライブが結合されて 1 つの論理ディスクドライブになります。さまざまな RAID レベルで、さまざまなレベルの冗長性と記憶容量が提供されます。RAID では、大容量のデータ記憶域が安価で提供されます。また、同じデータが異なる場所に格納されるため、I/O 操作を同時進行させられます。その結果、パフォーマンスが向上する場合があります。略称: RAID。

**redundant array of independent disks**
RAID を参照。

**sasroot**
サイトまたはコンピュータで SAS がインストールされているディレクトリまたはフォルダの名前を示す表記。

**SASROOT**
ユーザーのサイトまたはコンピュータで SAS がインストールされているディレクトリまたはフォルダの名前を表す用語。

**Scalable Performance Data Engine**
この SAS エンジンでは、効率化されたファイル形式にデータが編成されるため、データを迅速にアプリケーションに配信できます。略称: SPD Engine。

**SMP**
対称型マルチプロセッシングを参照。

**SPD Engine**
Scalable Performance Data Engine を参照。

**SPD Engine データセット**
SPD Engine で作成されるデータセットで、コンポーネントファイルが 4 つまで含まれます(データ用が 1 つ、メタデータ用が 1 つ、任意のインデックス用が 2 つ)。最小コンポーネントファイル数は、データとメタデータの 2 つです。SPD Engine のファイル編成では、データとメタデータは分けられます。

**SPD Engine データファイル**
SPD Engine データセットのデータコンポーネント。SAS データファイルとは対照的に、SPD Engine データファイルに含まれるのはデータのみです。メタデータは含まれません。SPD Engine では、データビューはサポートされません。

**WHERE 式**
オブザベーションの選択基準を定義します。

**解決時間**
タスクの完了に必要な経過時間。解決時間の測定値は、異なるコンピューティング環境におけるソフトウェアアプリケーションのパフォーマンス比較に使用されます。言い換えれば、スケーラビリティの測定に使用されます。

**起動**
プロセスまたはプロセススレッド(軽量プロセススレッド(LWPT)など)を開始すること。

**軽量プロセススレッド**
通常はオペレーティングシステムコールによって個別に作成および制御される単一スレッドサブプロセス。複数の軽量プロセススレッドは、対称型マルチプロセッシング(SMP)ハードウェアかまたはスレッド対応オペレーティングシステムで一度にアクティブにできます。

**コントローラ**
コンピュータと、ディスクや RAID などの周辺デバイスとの間の相互作用を管理するコンピュータコンポーネント。たとえば、コントローラでは、CPU とディスクドライブとの間のデータ I/O が管理されます。コンピュータには多くのコントローラを含められます。1 つの CPU で 2 つ以上のコントローラに命令できます。また、1 つのコントローラで複数のディスクに命令できます。

**サーバースケーラビリティ**
複数のクライアント要求を同時に処理するために SMP ハードウェアとスレッド処理を利用するサーバー機能。したがって、SMP ハードウェアで提供される計算能力が増加すると、それに比例して時間単位ごとの処理可能トランザクション数が増加します。

**冗長性**
障害またはエラー、あるいはその両方の影響を最小限に抑えるために、複数の交換可能な構成要素が提供されるというコンピューティングシステムの特性。たとえば、データが(RAID などで)冗長的に格納されている場合、1 つのディスクが失われても、そのデータは別のディスクではなおも使用可能です。

**スケーラビリティ**
実行する計算や演算の量の変化、およびコンピューティング環境の変化にもかかわらず、パフォーマンスをほとんど低下させず十分に機能させられるソフトウェアアプリケーションの機能。スケーラブルなソフトウェアでは、SMP ハードウェアおよびスレッド処理の使用によってもたらされるような計算機能の増大を最大限に活用できます。

**スケーラブルなソフトウェア**
SMP ハードウェアの計算機能の増大に予測どおりに対応するソフトウェア。たとえば、CPU 数を増加した場合、CPU バウンドの問題解決までの時間は、それに比例して減少します。また、I/O システムのスループットが増加した場合、I/O バウンドの問題解決までの時間は、それに比例して減少します。

**スレッド**
CPU のコアで動作するプロセス実行のシングルパス。

**スレッド I/O**
速度増加のために複数のスレッドによって実行される I/O。スレッド I/O でパフォーマンスを大幅に向上させるには、I/O を実行しているアプリケーションにもデータの迅速な処理能力が必要です。

**スレッド処理**
CPU バウンドアプリケーションの速度を向上させるために複数のスレッドで実行される処理。

**スレッド処理**
データ処理かまたはデータ I/O のハイパフォーマンステクノロジです。この場合、タスクがスレッドに分割され、1 つ以上の CPU の複数のコアで同時実行されます。

**スレッド対応オペレーティングシステム**
複数の CPU による共有メインメモリ領域への対称アクセスの調整が可能なオペレーティングシステム。この調整アクセスによって、同じプロセスのスレッドが非常に効率的にデータを共有できるようになります。

**スレッド対応プロシジャ**
スレッド I/O またはスレッド処理をサポートする SAS プロシジャ。

**ソートインジケータ**
データセットを並べ替えるかどうか、どのように並べ替えたか、および並べ替えが検証されたかどうかを示すデータファイルの属性。ソートインジケータは特に次の情報を示します。1) 並べ替えに使用された BY 変数。2) 文字変数に使用された文字セット。3) 使用された文字変数の照合順序。4) 並べ替え情報が検証されたかどうか。この属性は、データファイルディスクリプタ情報に格納されます。プロセスの一部としてデータの並べ替えを必要とする SAS プロシジャはすべて、ソートインジケータを使用します。

**対称型マルチプロセッシング**
I/O 速度と処理速度の向上を可能にするハードウェアとソフトウェアのアーキテクチャ。SMP マシンには、複数の CPU とスレッド対応オペレーティングシステムが搭載されています。SMP マシンは通常、複数のコントローラ、およびコントローラごとの複数のディスクドライブで構成されます。略称: SMP。

**データパーティション**
データが含まれる物理ファイルで、SAS Scalable Performance Data Engine データセットのデータコンポーネントから成る物理ファイルのコレクションの一部。

**パーティション**
複数のデバイスやディレクトリにわたる論理ファイルの一部または全部。SPD Engine では、パーティションは 1 つの物理ファイルです。データファイル、インデックスファイル、メタデータファイルはすべて分割可能で、その結果、それぞれデータパーティション、インデックスパーティション、メタデータパーティションが生成されます。ファイルの分割によって、非常に大きなデータセットでパフォーマンスの向上が可能になります。

**複合 WHERE 式**
WHERE X=1 と Y>3 などのように、2 つ以上の演算子を含む WHERE 式。

**プライマリパス**
SPD Engine メタデータファイルが格納される場所。複数 CPU のパフォーマンス向上を活用するために、その他の SPD Engine コンポーネントファイル(データファイルとインデックスファイル)は別の記憶域パスに格納されます。

**ブロック**
データセット内のオブザベーションのグループ。ブロックを使用すると、スレッド対応アプリケーションで、オブザベーションの読み取り、書き込みおよび処理が、個別オブザベーションとして配信される場合よりも迅速に行えます。

**並列 I/O**
複数の CPU と複数のコントローラを利用した入出力方法で、コントローラごとに複数のディスクを使用して、独立したスレッドでデータの読み取りや書き込みを行います。

**並列処理**
大きなジョブを分割し、複数の CPU で並列実行可能な複数の小さなジョブにする処理方法。

# キーワード