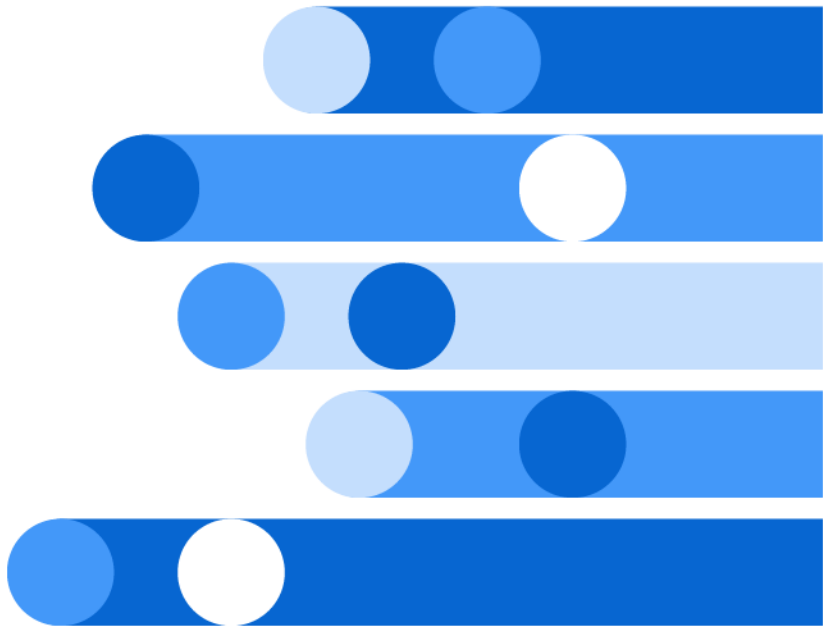
SAS® FedSQL 言語リファレンス

2021.1.6 - 2026.08*



* このドキュメントは、ソフトウェアの追加バージョンに適用される場合があります。このドキュメントを SAS Help Center で開き、バナーのバージョンをクリックすると、使用できるすべてのバージョンが表示されます。

SAS® ドキュメント
2026 年 9 月 18 日

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2020. *SAS® FedSQL 言語リファレンス*. Cary, NC: SAS Institute Inc.

SAS® FedSQL 言語リファレンス

Copyright © 2020, SAS Institute Inc., Cary, NC, USA

All Rights Reserved. Produced in the United States of America.

For a hard copy book: No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

For a web download or e-book: Your use of this publication shall be governed by the terms established by the vendor at the time you acquire this publication.

The scanning, uploading, and distribution of this book via the Internet or any other means without the permission of the publisher is illegal and punishable by law. Please purchase only authorized electronic editions and do not participate in or encourage electronic piracy of copyrighted materials. Your support of others' rights is appreciated.

U.S. Government License Rights; Restricted Rights: The Software and its documentation is commercial computer software developed at private expense and is provided with RESTRICTED RIGHTS to the United States Government. Use, duplication, or disclosure of the Software by the United States Government is subject to the license terms of this Agreement pursuant to, as applicable, FAR 12.212, DFAR 227.7202-1(a), DFAR 227.7202-3(a), and DFAR 227.7202-4, and, to the extent required under U.S. federal law, the minimum restricted rights as set out in FAR 52.227-19 (DEC 2007). If FAR 52.227-19 is applicable, this provision serves as notice under clause (c) thereof and no other notice is required to be affixed to the Software or documentation. The Government’s rights in Software and documentation shall be only those set forth in this Agreement.

SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

August 2026

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

v_001-P1:fedsqlref

目次

1 部 概要 1

1 章 / FedSQL 言語について **3**
- FedSQL 言語について 3
- FedSQL の利点 4
- FedSQL ステートメントのサブミット 4
- データソースサポート 5
- このドキュメントの目的 6
- 対象とするユーザー 7
- 文字体裁の規則 7
- 構文規則 8

2 部 FedSQL 言語リファレンス 11

2 章 / FedSQL 言語の概念 **13**
- データ型 14
- 識別子 18
- リテラル 20
- FedSQL が null および SAS 欠損値を処理する方法 21
- 型変換 26
- NLS トランスコーディングの失敗 30
- 結合操作 30
- FedSQL 式 46
- FedSQL の日付と時間 59
- DICTIONARY テーブル 67
- FedSQL 暗黙的パススルー機能 68
- FedSQL 明示的パススルー機能 70
- FedSQL でのトランザクション 71
- FedSQL 予約語 72

3 章 / FedSQL 出力形式 **77**
- 出力形式の概要 79
- 一般的な出力形式構文 80
- FedSQL での出力形式の使用 81
- FedSQL 出力形式の検証 82
- FedSQL 出力形式の例 83
- ユーザー定義出力形式の使用 84
- 出力形式カテゴリ 85
- FedSQL でサポートされる NLS 出力形式 86

PUT 関数でサポートされている出力形式(カテゴリ別) 89
ディクショナリ 96

4 章 / FedSQL 関数 249
FedSQL 関数の概要 256
一般的な関数構文 257
FedSQL 関数の使用 258
集計関数 260
関数カテゴリ 262
SAS Output Delivery System と FedSQL 262
カテゴリ別 FEDSQL 関数 263
ディクショナリ 281

5 章 / FedSQL の式と述語 897
式と述語の概要 897
ディクショナリ 898

6 章 / FedSQL 入力形式 923
入力形式の定義 923
一般的な入力形式構文 923
FedSQL での入力形式の使用方法 924
FedSQL で入力形式を指定する方法 926
FedSQL 入力形式の検証 926
FedSQL 入力形式の例 926

7 章 / FedSQL ステートメント 927
ステートメントの概要 927
カテゴリ別 FedSQL ステートメント 928
ディクショナリ 929

8 章 / FedSQL ステートメントテーブルオプション 1011
ステートメントテーブルオプションの概要 1014
テーブルオプションと他のタイプのオプションとの相互作用 1014
FedSQL ステートメントテーブルオプションの構文 1015
BULKLOAD テーブルオプションについて 1016
データソース別の FedSQL ステートメントテーブルオプション 1016
ディクショナリ 1032

9 章 / FedSQL とシステムオプション 1163
システムオプションの FedSQL サポート 1163

10 章 / FedSQL バージョン PROC SQL 1165
言語の概要 1165
SQL プロシジャと FedSQL の高レベルの相違点 1166
ステートメントの比較 1168
引用要件 1170
比較演算子 1172
DICTIONARY テーブル 1174
FedSQL での一般的な PROC SQL タスクの実行 1175

3 部 付録 1193

付録 1 / トラブルシューティング: FedSQL エラーの解決 1195
FedSQL エラーの解決 1195

付録 2 / データ型リファレンス 1199
SAS データセットのデータ型 1200
SPD Engine データセットのデータ型 1201
SPD Server テーブルのデータ型 1202
Amazon Redshift のデータ型 1204
UNIX および PC ホストでの DB2 のデータ型 1206
Google BigQuery のデータ型 1207
Greenplum のデータ型 1209
Hive のデータ型 1210
Impala のデータ型 1213
JDBC のデータ型 1214
MDS のデータ型 1216
Microsoft SQL Server のデータ型 1218
MongoDB のデータ型 1220
MySQL のデータ型 1222
Netezza のデータ型 1224
ODBC のデータ型 1225
Oracle のデータ型 1227
PostgreSQL のデータ型 1229
Salesforce から読み取るためのデータ型 1231
Salesforce に書き込むためのデータ型 1233
SAP のデータ型 1235
SAP HANA のデータ型 1237
SAP IQ のデータ型 1238
SingleStore のデータ型 1240
Snowflake のデータ型 1242
Spark のデータ型 1244
Teradata のデータ型 1246
Vertica のデータ型 1248
Yellowbrick のデータ型 1251

付録 3 / FedSQL と DS2 の使用 1253
DS2 から FedSQL ステートメントを動的に実行する 1253

付録 4 / 例で使用されているテーブル 1255
概要 1256
AFewWords 1256
Countries 1256
Customers 1258
CustonLine 1259
Densities 1260
Depts 1261
Employees 1262
EmpSales 1263
GrainProducts 1264
Houses 1265

Integers . . . 1265
Numbers1 . . . 1266
Numbers2 . . . 1267
Numbers3 . . . 1267
Payroll . . . 1268
Products . . . 1270
Sales . . . 1271
Staff . . . 1272
Stores . . . 1274
Survey . . . 1275
WorldCityCoords . . . 1276
WorldTemps . . . 1277

付録 5 / DICTIONARY テーブルの説明 . . . 1279
DICTIONARY.CATALOGS . . . 1279
DICTIONARY.COLUMNS . . . 1280
DICTIONARY.COLUMN_STATISTICS . . . 1283
DICTIONARY.STATISTICS . . . 1285
DICTIONARY.TABLES . . . 1290

付録 6 / FedSQL と ANSI 規格 . . . 1293
準拠 . . . 1293
FedSQL の拡張 . . . 1294
FedSQL の制限事項 . . . 1296

付録 7 / 使用上の注意 . . . 1297
使用上の注意 . . . 1297

付録 8 / ICU ライセンス . . . 1299
UNICODE ライセンス V3 . . . 1299
Third-Party Software Licenses: ICU 1.8.1–ICU 57 and ICU4J 1.3.1–ICU4J 57 . . . 1300
Unicode, Inc. License Agreement - Data Files and Software: ICU 63 and Later . . . 1307

1 部

概要

1 章
FedSQL 言語について 3

1

FedSQL 言語について

FedSQL 言語について . . . 3
FedSQL の利点 . . . 4
FedSQL ステートメントのサブミット . . . 4
データソースサポート . . . 5
このドキュメントの目的 . . . 6
対象とするユーザー . . . 7
文字体裁の規則 . . . 7
構文規則 . . . 8

FedSQL 言語について

SAS FedSQL は、ANSI SQL:1999 コア規格の SAS 独自の実装です。新しいデータ型やその他の ANSI 1999 コア準拠機能と独自の拡張機能をサポートしています。FedSQL は、複数のデータソース内のリレーショナルデータにアクセス、管理、および共有するためのスケーラブルでスレッド化された高性能な方法を提供します。大規模な操作を解決するために、可能な場合、FedSQL クエリはマルチスレッドアルゴリズムで最適化されます。FedSQL は、一部の非リレーショナルデータソースの読み取りもサポートしています。

アプリケーションには、FedSQL はすべてのデータソースに 共通の SQL 構文を提供します。つまり、FedSQL はベンダーニュートラルな SQL ダイアレクトであり、データソースに固有の SQL ダイアレクトでクエリをサブミットすることなく、さまざまなデータソースからのデータにアクセスすることができます。さらに、単一の FedSQL クエリは、複数のデータソースのデータをターゲットにして、単一の結果セットを返すことができます。

SQL:1999 コア準拠の構文と SAS 拡張機能の詳細については、付録 6, “FedSQL と ANSI 規格” (1293 ページ)を参照してください。

FedSQL の利点

FedSQL は、SQL プロシジャで提供されるよりも多くの機能が必要な環境で作業している場合、多くの利点を提供します。

- FedSQL は、ANSI SQL:1999 コア規格に準拠しています。この準拠により、独自の言語と、規格に準拠する他のデータソースのネイティブ言語でクエリを処理できます。
- FedSQL は、以前の SAS SQL 実装よりも多くのデータ型をサポートしています。SAS/ACCESS を介した従来のデータソースアクセスは、ターゲットデータソースのデータ型を 2 つのレガシー SAS データ型(SAS 数値と SAS 文字)との間で変換します。FedSQL がデータソースに接続すると、言語は必要に応じて、ターゲットデータソースの定義をこれらのデータ型に一致または変換します。これにより、精度が向上します。サポートされているデータ型については、“データ型” (14 ページ)で説明されています。
- FedSQL はフェデレーションクエリを処理します。フェデレーションクエリは、2 つ以上のデータソースのデータにアクセスし、単一の結果セットを返すクエリです。たとえば、SAS データセット、PostgreSQL テーブル、および Snowflake テーブルからデータをクエリできます。データはデータソースに保存されたままになります。従来の DATA ステップまたは SQL プロシジャを使用すると、SAS/ACCESS LIBNAME エンジンは、目的のデータソースのデータのみにアクセスできます。
- FedSQL 言語は、ターゲットデータソースがクエリで表されていない場合でも、サポートされている任意のデータソースでデータを作成できます。これにより、アプリケーションのニーズに最も近いデータソースにデータを保存できます。

FedSQL と PROC SQL の違いの詳細については、PROC SQL と FedSQL の高レベルの比較 (1165 ページ)を参照してください。

FedSQL ステートメントのサブミット

FedSQL ステートメントは、次のいずれかの方法でサブミットできます。

- FEDSQL プロシジャを使用して、SAS Data and AI Studio(2026.06 より前は SAS Studio という名前でした)を介します。PROC FEDSQL を使用すると、FedSQL ステートメントを SAS ライブラリエンジンまたは CAS サーバーにサブミットできます。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。
- PROC CAS を使用して SAS Data and AI Studio を介します。PROC CAS を使用する場合、CASL 言語を使用して CAS アクションを CAS サーバーにサブミットします。FedSQL 言語ステートメントは、fedSQL.execDirect アクションで指定さ

れます。fedSQL.execDirect アクションの詳細については、*SAS Viya プラットフォーム: システムプログラミングガイド*を参照してください。

- fedSql.execDirect アクションを使用して、Lua、Python、または R プログラムからサブミットします。詳細については、*SAS Viya プラットフォームプログラミング: CAS 用 Lua 入門ガイド*、*SAS Viya プラットフォームプログラミング: CAS 用 Python 入門ガイド*、および *SAS \Viya プラットフォームプログラミング: CAS 用 R 入門ガイド*を参照してください。
- SAS Federation Server を使用して、JDBC、ODBC、または OLE DB クライアントからサブミットします。*SAS Federation Server: 管理者ガイド*を参照してください。
- DS2 プログラムからサブミットします。付録 3, “FedSQL と DS2 の使用” (1253 ページ)を参照してください。
- SAS Federation Server の SAS LIBNAME エンジンは、SAS 言語呼び出しを FedSQL に内部的に変換して、サーバー上で実行します。PROC SQL EXECUTE ステートメントを使用して、FedSQL ステートメントを SAS Federation Server にサブミットすることもできます。詳細については、*SAS LIBNAME Engine for SAS Federation Server: ユーザーガイド*を参照してください。

データソースサポート

FedSQL は、次のデータソースにアクセスできます。

- SAS データセット
- CAS テーブル
- SAS Scalable Performance Data Engine (SPD Engine)データセット
- SAS Scalable Performance Data Server (SPD Server)テーブル
- Amazon Redshift
- DB2
- Google BigQuery
- Greenplum
- Hadoop (Hive)
- Impala
- JDBC に準拠したデータベース
- メモリデータストア(MDS)
- Microsoft SQL Server
- MongoDB
- MySQL
- Netezza

- ODBC に準拠したデータベース
- Oracle
- PostgreSQL
- Salesforce
- SAP (読み取り専用)
- SAP HANA
- SAP IQ
- SingleStore
- Snowflake
- Spark
- Teradata
- Vertica
- Yellowbrick

注: SAS ライブラリエンジン、CAS サーバー、および SAS Federation Server では、さまざまなデータソースがサポートされています。SAS エンジンおよび CAS でのデータソースサポートについては、"FEDSQL プロシジャ" (*Base SAS プロシジャガイド*)を参照してください。SAS Federation Server でサポートされているデータソースについては、*SAS Federation Server: 管理者ガイド*を参照してください。

このドキュメントの目的

*SAS FedSQL 言語リファレンス*には、Federated Query Language (FedSQL)に関する次の情報が記載されています。

- FedSQL 言語に関する概念情報
- SAS ライブラリエンジンおよび SAS Federation Server での使用がサポートされている主要な言語要素の詳細な参照情報。これらには次が含まれます。
  - データ型
  - 出力形式
  - 関数
  - 式、演算子および述語
  - ステートメント
  - テーブルオプション
- FedSQL と PROC SQL の高レベル比較

CAS でサポートされている FedSQL ステートメントについては、*SAS Viya プラットフォーム: SAS Cloud Analytic Services の FedSQL プログラミング*を参照してください。

対象とするユーザー

このドキュメントは、次のユーザーを対象としています。

- クライアントアプリケーションを作成し、FedSQL 言語を理解する必要があるアプリケーション開発者。テーブルの作成、テーブルのバルクロード、テーブルの操作、データのクエリを行うアプリケーションを作成します。
- FedSQL を使用して永続化された SAS データと DBMS データの組み合わせを作成、更新、およびクエリするプログラマ。

文字体裁の規則

FedSQL 構文のドキュメントでは、書体に特別な意味があります。次のリストは、構文の書体規則を説明しています。

UPPERCASE BOLD (大文字の太字)
ステートメントや関数の名前などの SAS キーワードを識別します(例: CREATE TABLE)。

UPPERCASE ROMAN (大文字のローマン体)
リテラルの引数と値を識別します(例: FROM)。

italic (斜体)
ユーザーが指定する引数や値を識別します。斜体の項目は、次のいずれかのユーザー指定の値を表すことができます。

- 引数に割り当てらる非リテラル値(例: END=*variable*)。
- 非リテラル引数(例: AS *alias*)。

斜体の項目を複数使用できる場合、その項目は *item* [, ...*item*]として表されます。

monospace
SAS コードの例を識別します。

構文規則

*SAS FedSQL 言語リファレンス*では、バッカスナウア記法(BNF)を使用しています。具体的には、Jim Melton が *SQL:1999 Understanding Relational Language Components* で使用したものと同じ構文表記法を使用しています。

従来の SAS 構文と SAS FedSQL 言語リファレンスドキュメントで使用されている構文の主な違いは、オプションの構文引数の表示方法です。従来の SAS 構文では、オプションの構文を示すために山かっこ(< >)が使用されます。FedSQL 構文では、角かっこ([ ])がオプションの構文を示すために使用され、山かっこは非終端コンポーネントを示すために使用されます。

FedSQL 構文では、次の記号が使用されます。

::=
: この記号は、"で構成される"または"として定義されている"と解釈できます。

< >
: 山かっこは、非終端コンポーネントを示します。つまり、下位レベルの構文文法にさらに分解できる構文コンポーネントです。

[ ]
: 角かっこはオプション引数を示します。角かっこで囲まれていない引数はすべて必須引数です。角かっこは、その前にリテラルであることを示すバックスラッシュ(\)が付いていない限り、入力しないでください。

{ }
: 中かっこは、必要な複数単語の引数を区別する方法を提供します。中かっこは、その前にリテラルであることを示すバックスラッシュ(\)が付いていない限り、入力しないでください。

|
: 縦棒は、グループの中から 1 つの値を選択できることを示しています。棒で区切られた値は相互に排他的です。

...
: 省略記号は、省略記号に続く引数または引数のグループが何度でも繰り返されることを示しています。省略記号と次の引数が角かっこで囲まれている場合はオプションです。

\
: バックスラッシュは、次の文字がリテラルであることを示します。

次の例は、このセクションで説明されている構文規則を示しています。これらの例には、完全な構文ではなく、選択された構文要素が含まれています。

CREATE TABLE {*table* | _NULL_}
[\{OPTIONS*SAS-table-option=value*[...*SAS-table-option=value*] \}]]

(<column-definition> [...<column-definition> | <table-constraint>])

| AS *query-expression*

;

<column-definition>::=

column data-type [<column-constraint><DEFAULT value>[HAVING[FORMAT format][INFORMAT informat][LABEL='label']]]

- **CREATE TABLE** はステートメントの名前であるため、大文字の太字で示されています。
- *table* は、指定できる引数であるため、斜体で示しています。
- _NULL_は、リテラル引数であるため、大文字のローマン体です。
- バックスラッシュが前にある中かっこ(リテラル引数 OPTIONS の前後)は、中かっこがリテラルであることを示します。つまり、OPTIONS ステートメントは中かっこ内に指定する必要があります。
- <column-definition>は山かっこで囲まれています。これは、下位レベルの構文文法にさらに分解される非終端引数であるためです。
- <column-definition>の 2 番目のインスタンスを囲む角かっこと省略記号は、それがオプションであることを示しており、この引数をカンマで区切って何度でも繰り返すことができます。
- <column-definition>または AS *query-expression* を指定できますが、両方を指定することはできません。
- AS は、リテラル引数であるため、大文字のローマン体です。
- <column-definition>::=は非リテラル展開を示します。

2 部

FedSQL 言語リファレンス

2 章
FedSQL 言語の概念 13

3 章
FedSQL 出力形式 77

4 章
FedSQL 関数 249

5 章
FedSQL の式と述語 897

6 章
FedSQL 入力形式 923

7 章
FedSQL ステートメント 927

8 章
FedSQL ステートメントテーブルオプション 1011

9 章
FedSQL とシステムオプション 1163

10 章
FedSQL バージョン PROC SQL 1165

2

FedSQL 言語の概念

データ型 **14**

識別子 **18**
- 標準識別子 18
- 区切り識別子 18
- 非ラテン文字のサポート 19
- 区切り識別子の内側の引用符 19
- 区切り識別子でのマクロ変数の参照 19
- データソース固有の命名要件 19

リテラル **20**
- リテラルの指定規則 20
- 文字列リテラルの引用符とスラッシュ 20

FedSQL が null および SAS 欠損値を処理する方法 **21**
- 存在しないデータの FedSQL モード 21
- SAS 欠損モードのデータ型 22
- null 値の処理と SAS 欠損値の処理の違い 22
- ANSI モードでの存在しないデータの読み取りと書き込み 23
- SAS モードでの存在しないデータの読み取りと書き込み 24
- null および SAS 欠損値のテストと変更 25

型変換 **26**
- 型変換の定義 26
- 型変換の概要 27
- 単項式の型変換 27
- 論理式の型変換 28
- 算術式の型変換 28
- 関係式の型変換 29
- 連結式の型変換 29

NLS トランスコーディングの失敗 **30**

結合操作 **30**
- 結合操作の概要 30
- 結合操作について 32
- 内部結合タイプと外部結合タイプ 41
- 異種データの結合 45

FedSQL 式 **46**
- クエリ式とサブクエリ 46
- FedSQL 値式 58

FedSQL の日付と時間 **59**

FedSQL の日付と時間の概要 . . . 59
FedSQL の日付、時間、および日時の定数 . . . 60
SAS データセット、SPD Engine データセット、および SPD Server テーブルでの日付と時間の使用 . . . 61
日付、時間、および日時関数 . . . 62
SAS データセット、SPD Engine データセット、および SPD Server テーブルの日付、時間、および日時出力形式 . . . 63

DICTIONARY テーブル . . . **67**

FedSQL 暗黙的パススルー機能 . . . **68**
概要 . . . 68
FedSQL 暗黙的パススルーの使用方法 . . . 69
単一ソースの FedSQL 暗黙的パススルー . . . 69
複数ソースの FedSQL 暗黙的パススルー . . . 69

FedSQL 明示的パススルー機能 . . . **70**
概要 . . . 70
FROM 句の CONNECTION TO コンポーネント . . . 70
EXECUTE ステートメント . . . 71

FedSQL でのトランザクション . . . **71**

FedSQL 予約語 . . . **72**

データ型

データ型とは、テーブル内の各列の属性で、その列が保存するデータの型を指定します。たとえば、データ型は、文字列、整数、浮動小数点数、日付や時間であることを示すデータの特性です。また、データ型は、列の値に割り当てるメモリの量も決定します。

次の表は、FedSQL でサポートされているデータ型を示します。各データソースのテーブルストレージにすべてのデータ 型を使用できるわけではないことに注意してください。

表 2.1 FedSQL データ型

データ型	説明
BIGINT	19 桁の精度で、大きな符号付きの正確な整数が保存されます。整数の範囲は、-9,223,372,036,854,775,808 から 9,223,372,036,854,775,807 までです。整数データ型は小数値を保存しません。小数部分は破棄されます。
BINARY(*n*)	固定長のバイナリデータが保存されます。*n* は保存される最大バイト数です。値の実際のサイズに関係なく、各値を保存するには最大バイト数が必要です。
CHAR(*n*)[1]	固定長の文字列が保存されます。*n* は保存される最大文字数です。各文字に 1 バイトのストレージが使用されます。値の実際のサイズ

データ型	説明
	に関係なく、各値を保存するには最大文字数が必要です。**char(10)**が指定されていて、文字列の長さが 5 文字しかない場合、値はスペースで右埋めされます。
DATE	カレンダー日付を保存します。日付リテラルは *yyyy-mm-dd* の形式で指定します。4 桁の年(0001 から 9999)、2 桁の月(01 から 12)、および 2 桁の日(01 から 31)です。たとえば、1975 年 9 月 24 日の日付は **1975-09-24** と指定されます。 FedSQL は、日付に関する ANSI SQL:1999 規格に準拠しています。ただし、すべてのデータソースが 日付の全範囲をサポートしているわけではありません。たとえば、0001-01-01 から 1582-12-31 までの日付は、SAS データセット、SPD Engine データセット、および SPD Server テーブルの有効な日付ではありません。
DOUBLE	符号付き近似倍精度浮動小数点数が保存されます。大きな数値を許可し、小数点の右側に多くの桁の精度を必要とする計算を許可します。
DECIMAL|NUMERIC(*p*,*s*)	ユーザー指定の精度とスケールで、符号付きの正確な固定小数点 10 進数を保存します。小数点の位置は、精度とスケールによって決まります。精度は、小数点の左右に保存できる最大桁数で、範囲は 1 から 52 です。スケールは、小数点以下に保存できる最大桁数です。スケールは精度以下でなければなりません。たとえば、**decimal(9,2)** は、1234567.89 のように、2 桁の固定小数点小数部分を含む 9 桁までの 10 進数を保存します。 **注:** DECIMAL データ型は、操作をデータソースに完全に渡すことができない場合を除き、列の定義、列へのデータの挿入、およびフェッチ操作でサポートされています。データソースに渡すことができない操作については、“FedSQL 暗黙的パススルー機能” (68 ページ)で説明されています。これらの条件下では、DECIMAL 列は DOUBLE として処理され、精度に影響を与える可能性があります。要求は、DECIMAL を完全にサポートするか、DOUBLE に変換されます。DECIMAL を使用するには、要求がデータソースに完全に渡される必要があります。
FLOAT(*p*)	符号付き、近似、単精度、または倍精度の浮動小数点数が保存されます。ユーザー指定の精度によって、データ型が単精度数または倍精度数のどちらを保存するかが決まります。指定された精度が 25 以上の場合、値は倍精度数(DOUBLE)として保存されます。指定された精度が 25 未満の場合、値は単精度数(REAL)として保存されます。たとえば、**float(10)**は最大 10 桁を保存するように指定し、結果として REAL データ型になります。
INTEGER	10 桁の精度で、通常サイズの符号付きの正確な整数が保存されます。整数の範囲は-2,147,483,648 から 2,147,483,647 までです。整数データ型は小数値を保存しません。小数部分は破棄されます。

データ型	説明
	注: ゼロによる整数除算は、すべてのオペレーティングシステムで同じ結果になるわけではありません。ゼロによる整数除算は避けることをお勧めします。
NCHAR(*n*)[1]	CHAR などの固定長文字列を保存しますが、Unicode 各国語文字セットを使用します。ここで、*n* は保存するマルチバイト文字の最大数です。プラットフォームに応じて、Unicode 文字は 1 文字あたり 2 バイトまたは 4 バイトを使用し、すべての国際文字をサポートします。
NVARCHAR(*n*)[1]	VARCHAR などの可変長文字列を保存しますが、Unicode 各国語文字セットを使用します。ここで、*n* は保存するマルチバイト文字の最大数です。プラットフォームに応じて、Unicode 文字は 1 文字あたり 2 バイトまたは 4 バイトを使用し、すべての国際文字をサポートできます。
REAL	符号付き近似単精度浮動小数点数が保存されます。
SMALLINT	5 桁の精度で、小さな符号付きの正確な整数が保存されます。整数の範囲は-32,768 から 32,767 までです。整数データ型は小数値を保存しません。小数部分は破棄されます。
TIME(*p*)	時間値を保存します。時間リテラルは、*hh:mm:ss[.nnnnnnnnn]*の形式で指定します。2 桁の時間 00 から 23、2 桁の分 00 から 59、および 2 桁の秒 00 から 59 で、オプションの小数値があります。たとえば、午前 6:30 は **06:30:00** と指定されます。データソースでサポートされている場合、*p* パラメーターは秒の精度を指定します。秒の精度は、最大 9 桁の長さのオプションの小数値です。
TIMESTAMP(*p*)	日付と時間の両方の値を保存します。タイムスタンプリテラルは、*yyyy-mm-dd:hh:mm:ss[.nnnnnnnnn]*の形式で指定します。4 桁の年 0001 から 9999、2 桁の月 01 から 12、2 桁の日 01 から 31、2 桁の時間 00 から 23、2 桁の分 00 から 59、2 桁の秒 00 から 59 で、オプションの小数値があります。たとえば、1975 年 9 月 24 日午前 6:30 の日時は **1975-09-24:06:30:00** と指定されます。データソースでサポートされている場合、*p* パラメーターは秒の精度を指定します。秒の精度は、最大 9 桁の長さのオプションの小数値です。
TINYINT	3 桁の精度で、非常に小さな符号付きの正確な整数が保存されます。整数の範囲は、-128 から 127 までです。整数データ型は小数値を保存しません。小数部分は破棄されます。
VARBINARY(*n*)	可変長のバイナリデータが保存されます。*n* は保存される最大バイト数です。各値の保存に最大バイト数は必要ありません。**varbinary(10)**が指定され、バイナリ文字列が 5 バイトのみを使用する場合、列には 5 バイトのみが保存されます。
VARCHAR(*n*)[1]	可変長の文字列が保存されます。*n* は保存される最大文字数です。各文字に 1 バイトのストレージが使用されます。各値の保存に最大

データ型	説明
	文字数は必要ありません。**varchar(10)**が指定されていて、文字列の長さが 5 文字しかない場合、列には 5 文字しか保存されません。

1 データ型が CHAR、NCHAR、NVARCHAR、および VARCHAR の場合、FEDSQL プロシジャは、入力として使用される列、または出力に含まれる列について、最大長の 32,767 文字をサポートします。

データ型を定義するときは、FedSQL でサポートされているデータ型またはターゲットデータベースの SQL 言語でサポートされているデータ型のいずれかに対してデータ型キーワードを使用します。つまり、次のように処理します。

- FedSQL ステートメントをサブミットする場合は、FedSQL データ型キーワードを使用します。
- 明示的な SQL パススルーを要求し、特定のデータソースによって実装されている SQL 言語を使用して SQL ステートメントをサブミットする場合は、ターゲットデータベースのデータ型名を使用します。特定のデータソースに対して SQL 言語を使用してデータ型を定義する方法については、そのデータソースのドキュメントを参照してください。

データを保存するには、そのデータソースのデータストレージでデータ型を使用できる必要があることに注意してください。FedSQL はいくつかのデータ型をサポートしていますが、特定のテーブルに定義できるデータ型はデータソースによって異なります。これは、各データソースが必ずしもすべての FedSQL データ型をサポートしているわけではないためです。さらに、データソースは標準 SQL データ型のバリエーションをサポートします。つまり、指定した特定のデータ型が別のデータ型にマッピングされる可能性があり、基になるデータソースで別の属性を持つ可能性もあります。これは、データソースが特定のデータ型をネイティブにサポートしていないが、同様のデータ型のデータ値をデータ損失なしで変換できる場合に発生します。たとえば、INTEGER データ型をサポートするために、SAS データセットはデータ型定義を DOUBLE である SAS 数値にマッピングします。

各データ型のデータソースの実装の詳細については、付録 2, “データ型リファレンス” (1199 ページ)を参照してください。

さらに、データ型のマッピング方法を制御する CT_PRESERVE=接続引数は、データ型を定義できるかどうかに影響する可能性があります。値 FORCE (デフォルト)および FORCE_COL_SIZE は、データ型を定義できるかどうかには影響しません。要求されたデータ型がデータソースにネイティブでない場合、または指定された精度またはスケールがデータソースの範囲内にない場合、値 STRICT および SAFE はエラーになる可能性があります。CT_PRESERVE=接続引数の詳細については、*SAS Federation Server: 管理者ガイド*を参照してください。

FedSQL は関係式のブール演算をサポートしています。論理式でのブール演算のサポートはデータソースに依存します。FedSQL は、NOT 演算子を含む SELECT ステートメントの検索条件でブールをサポートしません。SELECT ステートメントの検索条件には、値 TRUE、FALSE、または UNKNOWN に解決される 3 方向のロジックが適用されます。詳細については、“<search-condition>” (1006 ページ)を参照してください。

識別子

FedSQL は、標準識別子と区切り識別子の両方について、ANSI SQL:1999 コア規格をサポートしています。ANSI SQL:1999 識別子をサポートすることにより、FedSQL は、同様に ANSI SQL:1999 識別子をサポートするデータソースとも互換性があります。

標準識別子

標準識別子には次の特徴があります。

- 長さは 1 - 256 文字です。
- 文字、数字、およびアンダースコア(_)を含めることができます。
- 標準識別子の最初の文字は英字にする必要があります。後続の文字には、英字、数字、またはアンダースコアを使用できます。
- 標準識別子では大文字と小文字は区別されません。名前の書き方に関係なく、FedSQL は出力を大文字で表示します。つまり、識別子 NAME、name、および NaMe はすべて NAME として表示されます。

区切り識別子

区切り識別子には次の特徴があります。

- 長さは 1 - 256 文字です。
- 二重引用符で始まり、二重引用符で終わります。
- スペースや特殊文字を含む、最初と最後の二重引用符の間の文字シーケンスで構成されます。
- 大文字と小文字が区別されます。出力では、FedSQL は区切り識別子を記述どおりに表示します。すなわち、区切り識別子"NAME"、"name"、および"NaMe"は、NAME、name、および NaMe として表示されます。

> ヒント　大文字と小文字を保持したり、テーブル、列、またはカタログ名に特殊文字を含めたりするには、エンティティに名前を付けるときに区切り識別子を使用します。

そのままでは予約語となる単語についても、区切り識別子が使用されます。たとえば、文字宣言以外で"char"という単語を使用するには、区切り識別子"char"として指定します。詳細については、“FedSQL 予約語” (72 ページ)を参照してください。

非ラテン文字のサポート

非ラテン文字は区切り識別子として指定する必要があります。標準識別子で使用できるのはラテン文字のみです。

区切り識別子の内側の引用符

外側の引用符との混同を避けるために、区切り識別子の内側の二重引用符(")はエスケープする必要があります。二重引用符は、二重引用符を二重にすることでエスケープされます。たとえば、区切り識別子"identifier abc"def"の二重引用符は、FedSQLでは次のようにエスケープされます。

```
create table test ("identifier abc""def" char(100));
```

区切り識別子でのマクロ変数の参照

区切り識別子で SAS マクロ変数を参照するには、SAS マクロ関数%TSLIT を使用します。これにより、リテラル文字列を二重引用符で囲む必要がなくなり、入力値を単一引用符で囲みます。

%TSLIT マクロ関数は、デフォルトの自動呼び出しマクロライブラリに保存されています。構文は次のとおりです。

```
%TSLIT(&macro-name)
```

注: SAS マクロ変数は、PROC FEDSQL を使用して Base SAS セッションで実行される FedSQL プログラムでのみ使用できます。SAS Federation Server など、SAS の外部で実行される FedSQL プログラムでは使用できません。SAS マクロ変数を使用して、ソースデータを準備して実行する前に前処理することができます。SYMPUT()やSYMGET()などの関数は FedSQL では機能しません。

データソース固有の命名要件

サポートされている各データソースには独自の命名規則があり、そのすべてが FedSQL で受け入れられます。プログラムに特定のデータソースに固有の識別子が含まれている場合は、そのデータソースの命名規則に従う必要があります。命名規則については、データベースのドキュメントを参照してください。

SAS データセット、SPD Engine データセット、および SPD Server テーブルの識別子は、32 文字に制限されています。標準識別子と区切り識別子の FedSQL ルール

は、他の SAS 言語のものとは異なります。詳細については、“引用要件” (1170 ページ)を参照してください。

複数のデータソースが関係する場合、識別子の最大長は、すべてのデータソースと FedSQL でサポートされる最小の最大長によって決まります。たとえば、データソースが最大 32 文字の SAS データセットと、最大 64 文字の MySQL である場合、識別子の最大長は 32 文字になります。

リテラル

リテラルの指定規則

*リテラル*という用語は、*定数*の類義語であり、固定データ値を指します。固定データ値を列値として、および SQL 式で指定します。FedSQL は、文字列リテラル、各国語文字列リテラル、数値リテラル、日時リテラルをサポートしています。

文字列リテラルは、文字値として扱われる 0 個以上の英数字です。文字列リテラルは、単一引用符で囲んで指定する必要があります。文字列リテラルの例は、'Tuesday'、'John Jamison'、'abc123'、'1'、' ' (空白)です。

(オプション) NCHAR(*n*)列および NVARCHAR(*n*)列の文字値を格納するために、各国語文字列リテラルを使用できます。各国語文字列リテラルは、文字 N で始まる単一引用符で囲んで指定します。各国語文字列リテラルの例を次に示します: n'Schrödinger'および n'美丽的'。文字定数の先頭の N は、文字列のエンコーディングを Unicode として設定します。

注: 文字列リテラルとして指定された数値は計算に使用できません。

数値リテラルは、計算で使用できる数値です。数値リテラルは、単に数値または整数として指定します。

日時リテラルは、日付値、時間値、または日時値です。日付、時間、および日時値は、FedSQL の日付、時間、または日時定数として指定されます。詳細については、“FedSQL の日付と時間” (59 ページ)を参照してください。

文字列リテラルの引用符とスラッシュ

文字列リテラル内でエスケープが必要な埋め込み文字は、単一引用符(')とバックスラッシュ(\)だけです。埋め込まれた各文字は、文字を 2 倍にしてエスケープされます。つまり、次のとおりです。

- 内側の単一引用符は、別の単一引用符('')でエスケープされます。文字列リテラルの外側の引用符との混同を避けるために、内側の単一引用符はエスケープされます。

- 内側のバックスラッシュは、別のバックスラッシュでエスケープされます(\\)。

例として、このコードを紹介します。

```
proc fedsql;
  create table test (col1 char(20));
  insert into test values ('literal P''A"3');
  insert into test values ('literal P\\3/A');
  select * from test;
quit;
```

次の結果が生成されます。

COL1
literal P'A"3
literal P\3/A

内側の二重引用符とスラッシュ(/)はエスケープする必要はありません。

FedSQL が null および SAS 欠損値を処理する方法

存在しないデータの FedSQL モード

存在しないデータは、SAS データセット、SPD Engine データセット、および SPD Server テーブルの SAS 欠損値によって表されます。他のすべてのデータソースでは、存在しないデータは ANSI SQL null 値で表されます。SAS 欠損値インジケーター、.、._、.A - .Z、と'␣'は、存在しないデータを示す既知の値です。ANSI null を持つテーブルデータには、実際のデータ値はありません。不明な値を示すメタデータです。

null 値と SAS 欠損値の処理には大きな違いがあるため、FedSQL には、存在しないデータを処理するための 2 つのモードがあります。ANSI SQL null モード(ANSI モード)と SAS 欠損値モード(SAS モード)です。

存在しないデータの動作は、データソースへの接続方法によって異なります。

- デフォルトでは、JDBC や ODBC などのクライアント側ドライバーを介してデータソースに接続するクライアントアプリケーションは、ANSI モードを使用してデータを処理します。
- デフォルトでは、PROC FEDSQL をサブミットする Base SAS セッションは、SAS モードを使用してデータを処理します。PROC FEDSQL は、ANSI モードでデータを処理するために ANSIMODE オプションを提供します。

ほとんどの場合、存在しないデータを処理するためにモードを変更する必要はありません。次の状況では、モードを変更する必要がある場合があります。

- クライアントアプリケーションが SAS データセット、SPD Engine データセット、または SPD Server テーブルを処理し、存在しないデータのモードが ANSI モードである場合
- SAS データセット、SPD Engine データセット、または SPD Server テーブルの処理が完了し、クライアントアプリケーションが ANSI モードに戻る準備ができたとき

注意
目的の結果が得られるようにモードが設定されていない場合、データは失われます。
ANSI モードでは、FedSQL が数値の SAS 欠損データ値をデータソースから読み取るときに、DOUBLE のデータ型に変換します。SAS 欠損データ値が .A などの特殊欠損値である場合、 .A は null に変換されるときに失われます。null 値が SAS データセット、SPD Engine データセット、または SPD サーバーテーブルに書き込まれると、FedSQL はそれを SAS 欠損値であるピリオド(.)に変更します。SAS モードでは、CHAR 型の null がデータソースから読み取られると、FedSQL はそれを空白文字に変換します。空白文字が SAS データセット、SPD Engine データセット、または SPD Server テーブルに保存されている場合、その値は ANSI モードで不明な値として解釈されなくなりました。

ANSI モードおよび SAS モードで FedSQL を設定する方法については、クライアント環境の SAS ドキュメントを参照してください。

SAS 欠損モードのデータ型

現在、FedSQL は、SAS 欠損モードを DOUBLE および CHAR データ型の列と、それらのデータ型を含む比較に適用します。SAS 欠損モードは、FedSQL DATE、TIME、および TIMESTAMP データ型で定義された列には適用されません。これらのデータ型は DOUBLE に変換され、一部のデータソースには SAS 出力形式が割り当てられます。FedSQL での日付値と時間値の処理方法については、“FedSQL の日付と時間” (59 ページ)を参照してください。FedSQL は、これらのソースの日付と時間の値を含む列を、他のすべての方法で DATE、TIME、および TIMESTAMP データ型のように扱います。

null 値の処理と SAS 欠損値の処理の違い

SAS 欠損値の処理は、null 値の処理とは異なり、次の状況で重要な意味を持ちます。

- データをフィルタリングする場合(たとえば、WHERE 句、HAVING 句、または外部結合 ON 句)。SAS モードは null 値を既知の値である SAS 欠損値として解釈しますが、ANSI モードは null 値を不明な値として解釈します。
- ANSI モードで外部結合をサブミットすると、内部処理によって中間結果セットに null が生成される場合があります。FedSQL は、中間結果セットの SAS モードで SAS 欠損値を生成する場合があります。したがって、中間結果セットの場合、null は ANSI モードでは不明な値として解釈され、SAS モードでは欠損値は既知の値として解釈されます。

- 空白文字を比較する場合。SAS モードは、空白文字を欠損値として解釈します。ANSI モードでは、空白文字は空白文字です。特別な意味はありません。

次の表に、null 値と SAS 欠損値の属性と動作の違いを示します。

表 2.2 null 値と SAS 欠損値の属性と動作の違い

属性または動作	null 値	SAS 欠損値
内部表現	メタデータ	浮動小数点または文字
論理演算子による評価	不明な値であり、解決値が True、False、および Unknown となる 3 値ロジックを使用して比較されます。たとえば、WHERE col1 = null は UNKNOWN を返します。	既知の値であり、比較すると、解決されてブール結果になります。
照合順序	最小値として表示されます。	最小値として表示されます。

null 値に対する論理演算の結果については、“<search-condition>” (1006 ページ)を参照してください。

ANSI モードでの存在しないデータの読み取りと書き込み

Oracle や DB2 などの多くのリレーショナルデータベースは、ANSI SQL の null 値を実装しています。したがって、FedSQL を使用した null 値の概念は、ANSI SQL をサポートするデータベースで SQL 言語を使用する場合と同じです。データが失われる可能性があるため、FedSQL が SAS 欠損値を処理する方法を理解することが重要です。

SAS 欠損値のデータ型は、DOUBLE または CHAR のみです。したがって、これらのデータ型の変換のみが示されています。次の表は、FedSQL が ANSI モードでデータソースから null 値または SAS 欠損値を読み取るときに、クライアントアプリケーションに返される値を示しています。

表 2.3 ANSI モードでの存在しないデータ値の読み取り

列のデータ型	存在しないデータ値	アプリケーションに返される値
DOUBLE	.、._、または.A - .Z	null
DOUBLE	null	null

列のデータ型	存在しないデータ値	アプリケーションに返される値
CHAR	'␣'	'␣'
CHAR	null	null

注: 値'␣'は単一引用符の間の空白スペースであり、ANSI モードでは空白スペースであり、存在しないデータではありません。

次の表は、存在しないデータ値が ANSI モードでデータソースに書き込まれたときに保存される値を示しています。

表 2.4 ANSI モードでの存在しないデータ値の書き込み

列のデータ型	存在しないデータ値	次に保存される値: SAS データセット、SPD Engine データセット、または SPD Server テーブル	ANSI SQL null サポートデータソースに保存される値
DOUBLE	.、._、または.A - .Z	.	null
DOUBLE	null	.	null
CHAR	'␣'	'␣'	'␣'
CHAR	null	'␣'	null

注: 値'␣'は単一引用符の間の空白スペースであり、ANSI モードでは空白スペースであり、存在しないデータではありません。

SAS モードでの存在しないデータの読み取りと書き込み

クライアントアプリケーションが SAS モードを使用する場合、Base SAS 環境では存在しないデータ値が SAS 欠損値のように扱われます。

次の表は、データ型 DOUBLE および CHAR の存在しないデータ値が SAS モードでどのように読み取られるかを示しています。

表 2.5 SAS モードでの存在しないデータ値の読み取り

列のデータ型	存在しないデータ値	アプリケーションに返される値
DOUBLE	.、._、または.A - .Z	.、._、または.A - .Z
DOUBLE	null	.
CHAR	'␣'	'␣'
CHAR	null	'␣'

注: 値'␣'は単一引用符の間の空白スペースであり、SAS モードでは存在しないデータです。

次の表は、存在しないデータ値が SAS モードでデータソースにどのように書き込まれるかを示しています。

表 2.6 SAS モードでの存在しないデータ値の書き込み

列のデータ型	存在しないデータ値	次に保存される値: SAS データセット、SPD Engine データセット、または SPD Server テーブル	ANSI SQL null サポートデータソースに保存される値
DOUBLE	.、._、または.A - .Z	.、._、または.A - .Z	null
DOUBLE	null	.	null
CHAR	'␣'	'␣'	'␣'
CHAR	null	'␣'	'␣'

注: 値'␣'は単一引用符の間の空白スペースであり、SAS モードでは存在しないデータです。

null および SAS 欠損値のテストと変更

FedSQL は、null 値をテストするための IFNULL 関数と、null 値を変更するための NULLIF 式を提供します。

IFNULL 関数は、引数として 2 つの式を取ります。最初の式が null 値の場合、2 番目の式を返します。それがない場合は、関数から最初の値が返されます。

IFNULL(*expression*, *return_value_if_null_expression*)

expression の値が null の場合、関数は *return_value_if_null_expression* の値を返します。

この例では、numCopies の値が不明な本について、すべての本の名前が返されます。

```
select bookName when ifnull(numCopies, 'T') = 'T';
```

NULLIF 式も引数として 2 つの式を取ります。2 つの式が等しい場合、返される値は null 値です。それ以外の場合、返される値は最初の SQL 式です。

NULLIF(*expression*, *test_value_expression*);

ここで、numCopies の値が負の値-1 の場合、不明な値であることを示すために null に置き換えられます。

```
update books set numCopies = nullif(numCopies, –1);
```

詳細については、“IFNULL 関数” (573 ページ)および “NULLIF 式” (917 ページ)を参照してください。

型変換

型変換の定義

バイナリデータ型
: VARBINARY および BINARY データ型を指します。

文字データ型
: CHAR、VARCHAR、NCHAR、NVARCHAR データ型を指します。

強制型変換可能なデータ型
: 文字データ型だけでなく、複数のデータ型に変換できるデータ型。

日付/時間データ型
: DATE、TIME、および TIMESTAMP データ型を指します。

強制型変換できないデータ型
: 文字データ型にのみ変換できるデータ型。

数値データ型
: DECIMAL、NUMERIC、DOUBLE (または FLOAT)、REAL、BIGINT、INT、SMALLINT、TINYINT データ型を指します。

標準文字変換
: 式が文字データ型でない場合は、CHAR データ型に変換されます。

標準数値変換
: 式が強制型変換可能な非数値データ型の場合、DOUBLE データ型に変換されます。

型変換の概要

FedSQL が式を解決するには、式のオペランドが同じ一般的なデータ型(数値、文字、バイナリ、日付/時間)である必要があります。必要な場合、FedSQL は式のオペランドと演算子に応じて、オペランドのデータ型を別のデータ型に変換します。このプロセスは*型変換*と呼ばれます。型変換は、基になるデータソースがサポートしている場合にのみ発生します。たとえば、連結演算子(||)は文字データ型を操作します。データ型 INTEGER および CHAR をサポートするデータベースの場合、文字列"First"と数値整数 1 の連結では、連結が行われる前に、オペランド 1 の INTEGER データ型が CHAR データ型に変換されます。

オペランドのデータ型が同じ一般的なデータ型内で変換される場合、オペランドのデータ型はプロモートされます。SMALLINT および TINYINT のデータ型のオペランドは INTEGER にプロモートされ、REAL 型のオペランドは DOUBLE にプロモートされます。メソッドおよび関数式の引数を含む、SMALLINT、TINYINT、および REAL のすべての操作に対して型プロモーションが実行されます。

数値および文字データ型は強制型変換できます。BINARY、VARBINARY、および日付/時間データ型の DATE、TIME、および TIMESTAMP は強制型変換不可であり、PUT 関数を使用していずれかの文字データ型にのみ変換できます。

FedSQL が式を評価するとき、オペランドのデータ型が正確に一致する場合、型の変換やプロモーションは必要なく、式は解決されます。それ以外の場合、各オペランドは、演算子に応じて、標準数値変換または標準文字変換を行う必要があります。

数値式または文字式の結果は、データ型の優先順位に基づきます。両方のオペランドが同じ一般的なデータ型内で異なる型を持つ場合、式のデータ型は、優先順位の高いオペランドのデータ型になります。ここで、1 が最高の優先順位です。たとえば、数値データ型の場合、DOUBLE のデータ型が最も優先されます。式に INTEGER 型のオペランドと DOUBLE 型のオペランドがある場合、式のデータ型は DOUBLE です。優先順位のリストは、該当する場合は、さまざまなタイプの式の後続のトピックにあります。

単項式の型変換

++1 や-444 などの単項式では、標準数値変換がオペランドに適用されます。次の表は、単項式のデータ型を示しています。

表 2.7　単項式のデータ型変換

式のタイプ	式のデータ型
単項プラス	オペランドと同じ、または変換されたオペランドの DOUBLE

式のタイプ	式のデータ型
単項マイナス	オペランドと同じ、または変換されたオペランドの DOUBLE
単項否定	INTEGER

論理式の型変換

(a <> start) OR (f = finish)などの論理式では、標準数値変換が各オペランドに適用されます。次の表は、式のデータ型を決定するために使用される優先順位を示しています。1 が最高の優先順位で、3 が最低です。式のデータ型は、優先順位の高いオペランドのデータ型です。

表 2.8 論理式のデータ型変換

優先順位	いずれかのオペランドのデータ型	式のデータ型
1	DOUBLE	DOUBLE
2	BIGINT	BIGINT
3	他のすべての数値データ型	INTEGER

算術式の型変換

a<>b や a + (b * c)などの算術式では、標準数値変換が各オペランドに適用されます。

次の表は、加算、減算、乗算、除算演算子の算術式のデータ型を決定するために使用される優先順位を示しています。1 が最高の優先順位で、3 が最低です。式のデータ型は、優先順位の高いオペランドのデータ型です。

表 2.9 加算、減算、乗算、除算式の型変換

優先順位	いずれかのオペランドのデータ型	式のデータ型
1	DOUBLE	DOUBLE
2	BIGINT	BIGINT
3	他のすべての数値データ型	INTEGER

次の表は、べき乗演算子を使用する算術式のデータ型を示しています。

表 2.10　最小、最大、およびべき乗演算子式のデータ型変換

演算子	演算子のデータ型	式のデータ型
**	すべての数値データ型	DOUBLE

関係式の型変換

x <= y や i > 4 などの関係式では、適用される標準変換はオペランドのデータ型によって異なります。次の表に示すように、式のデータ型は常にブールです。

表 2.11　IN 式以外の関係式のデータ型変換

データ型解決の順序	いずれかのオペランドのデータ型	標準変換	式のデータ型
1	任意の数値データ型	数値	BOOLEAN
2	CHAR/NCHAR	文字	BOOLEAN
3	DATE、TIME、TIMESTAMP	なし、データ型が一致する必要がある	BOOLEAN
4	他のすべてのデータ型	なし、エラーが返される	適用できません

表 2.12　IN 式のデータ型変換

オペランド	オペランド変換	式のデータ型
すべて	標準数値または標準文字	BOOLEAN

連結式の型変換

a || b や x !! y などの連結式では、標準文字変換が各オペランドに適用されます。次の表は、式のデータ型を決定するために使用される優先順位を示しています。1 が最高の優先順位で、2 が最低です。式のデータ型は、優先順位の高いオペランドのデータ型です。

表 2.13 連結式のデータ型変換

優先順位	いずれかのオペランドのデータ型	式のデータ型
1	どちらかが NCHAR 型の場合	NCHAR
2	CHAR	CHAR

NLS トランスコーディングの失敗

トランスコーディングは、文字データをあるエンコーディングから別のエンコーディングに変換するプロセスです。NLS トランスコーディングの失敗は、行の入力または出力操作中、または文字列の割り当て中に発生する可能性があります。デフォルトでは、このランタイムエラーにより、行の処理が停止します。次のいずれかのオプションを使用して、デフォルトの動作を変更できます。

- SAS Federation Server: DEFAULT_ATTR=接続オプションを XCODE_WARN=*n* ステートメントハンドルオプションとともに指定します。
- PROC FEDSQL および PROC DS2: XCODE=オプションを設定します。

オプションを使用して、エラーを無視して行の処理を続行することを選択できます。

詳細については、SAS Federation Server および SAS プロシジャドキュメントを参照してください。

結合操作

結合操作の概要

結合操作とは、2 つ以上のテーブルまたはビューのデータを、通常はそれらのテーブル内のデータ間の関係に基づいて結合するクエリのことです。SELECT ステートメントの FROM 句に複数のテーブル指定がリストされている場合、それらは 1 つの結果セットを形成するように処理されます。結果セットには各寄与テーブルのデータが含まれており、テーブルとして保存したり、そのまま使用したりすることができます。ほとんどの結合操作には、FROM 句または WHERE 句のいずれかに結合条件が少なくとも 1 つ含まれています。

たとえば、両方のテーブルに存在する列の値に基づいて、2 つのテーブルのデータを結合できます。次のクエリは、2 つのテーブル Products と Sales を結合します。

FedSQL は、列 Prodid の値が一致する列 Product および Totals のデータを取得することにより、結果セットを作成します。

注: これらのコード例では、FedSQL ステートメントは PROC FEDSQL を使用して SAS に送信されます。コードは、SAS Work ライブラリで作成されたテーブルで実行されます。

```
proc fedsql;
  select products.product, sales.totals
    from products, sales
    where products.prodid=sales.prodid;
quit;
```

アウトプット 2.1　テーブル Products と Sales の結合結果セット

PRODUCT	TOTALS
Rice	$189,400
Corn	$555,789
Wheat	$781,183
Wheat	$2,789,654
Barley	$899,453

ほとんどの結合は 2 つのテーブルです。ただし、3 つ以上のテーブルも結合できます。3 つ以上のテーブルの結合操作を実行するには、FedSQL は最初に結合条件に基づいて 2 つのテーブルを結合します。次に、FedSQL は、結合条件に基づいて結果を別のテーブルに結合します。このプロセスは、すべてのテーブルが 結果セットに結合されるまで続きます。次のクエリは、最初にテーブル Products テーブルと Sales テーブルを結合して結果セットを生成します。次に、その結果セットとテーブル Customers (1258 ページ)を結合して、最終的な結果セットを生成します。

```
proc fedsql;
  select products.product, sales.totals, customers.city
    from products, sales, customers
    where products.prodid=sales.prodid and sales.custid=customers.custid;
quit;
```

アウトプット 2.2　テーブル Products、Sales、および Customers の結合結果セット

PRODUCT	TOTALS	CITY
Rice	$189,400	Boulder
Corn	$555,789	Nagasaki
Wheat	$781,183	Tokyo
Wheat	$2,789,654	Little Rock
Barley	$899,453	Buenos Aires

FedSQL は、単純結合、等価結合、クロス結合、限定結合、自然結合など、いくつかの結合操作をサポートしています。適切な構文により、結合操作のタイプが決ま

ります。さらに、限定結合と自然結合の操作は、内部結合または外部結合のいずれかの結合タイプを指定することで影響を受ける可能性があります。

注: このセクションの結合操作の例では、テーブル Customers、Products、および Sales を使用します。テーブルを表示するには、付録 4, “例で使用されているテーブル” (1255 ページ)を参照してください。

結合操作について

単純結合

単純結合は、カンマで区切られた複数のテーブルが SELECT ステートメントの FROM 句にリストされる最も基本的なタイプの結合です。結合条件はありません。この方法でテーブルを結合すると、1 番目のテーブルの各行が 2 番目のテーブルの各行と結合された結果セットが生成されます。

単純結合の例では、テーブル Products および Sales からすべての列とすべての行を選択します。

```
proc fedsql;
  select * from products, sales;
quit;
```

アウトプット 2.3　2 つのテーブルの単純結合

PRODID	PRODUCT	PRODID	CUSTID	TOTALS	COUNTRY
3234	Rice	3234	1	$189,400	United States
1424	Corn	3234	1	$189,400	United States
3421	Wheat	3234	1	$189,400	United States
3422	Oat	3234	1	$189,400	United States
3975	Barley	3234	1	$189,400	United States
3234	Rice	1424	3	$555,789	Japan
1424	Corn	1424	3	$555,789	Japan
3421	Wheat	1424	3	$555,789	Japan
3422	Oat	1424	3	$555,789	Japan
3975	Barley	1424	3	$555,789	Japan
3234	Rice	3421	4	$781,183	Japan
1424	Corn	3421	4	$781,183	Japan
3421	Wheat	3421	4	$781,183	Japan
3422	Oat	3421	4	$781,183	Japan
3975	Barley	3421	4	$781,183	Japan
3234	Rice	3421	2	$2,789,654	United States
1424	Corn	3421	2	$2,789,654	United States
3421	Wheat	3421	2	$2,789,654	United States
3422	Oat	3421	2	$2,789,654	United States

この例も単純結合ですが、SELECT ステートメントでは 3 つのテーブルのそれぞれから 1 つの列を指定しています。最初のテーブルの各行は 2 番目のテーブルの各行と結合され、次に 3 番目のテーブルの各行と結合されます。結果的に、基本的に意味のない大きな結果セットになります。次の出力は、結果セットの一部のみを示しています。

```
proc fedsql;
  select products.product, sales.totals, customers.country
    from products, sales, customers;
quit;
```

アウトプット 2.4　3 つのテーブルの単純結合

PRODUCT	TOTALS	COUNTRY
Rice	$189,400	United States
Corn	$189,400	United States
Wheat	$189,400	United States
Oat	$189,400	United States
Barley	$189,400	United States
Rice	$555,789	United States
Corn	$555,789	United States
Wheat	$555,789	United States
Oat	$555,789	United States
Barley	$555,789	United States
Rice	$781,183	United States
Corn	$781,183	United States
Wheat	$781,183	United States
Oat	$781,183	United States
Barley	$781,183	United States
Rice	$2,789,654	United States

等価結合

等価結合は、WHERE 句を含むサブセットである単純結合です。結合条件は等価比較でなければなりません。等価結合では、等価性テストを満たす行のみが返されるため、単純結合よりも意味のある結果が生成されます。AND 演算子を使用すると、複数の一致基準を指定できます。複数の一致基準が指定されている場合、すべての等価性テストを満たす行のみが返されます。

この等価結合の例では、テーブル Products および Sales から、両方のテーブルに存在する列 Prodid の値が一致するすべての列が選択されます。すべての列が*表記で選択されているため、結果セットでは Prodid 列が重複しています。

```
proc fedsql;
  select * from products, sales
    where products.prodid=sales.prodid;
quit;
```

アウトプット 2.5　すべての列の等価結合

PRODID	PRODUCT	PRODID	CUSTID	TOTALS	COUNTRY
3234	Rice	3234	1	$189,400	United States
1424	Corn	1424	3	$555,789	Japan
3421	Wheat	3421	4	$781,183	Japan
3421	Wheat	3421	2	$2,789,654	United States
3975	Barley	3975	5	$899,453	Argentina

列 Prodid、Product、および Totals を SELECT ステートメントで指定すると、列 Prodid は Products テーブルと Sales テーブルの両方に存在しても重複しません。結果セットには、列 Prodid の値が一致するデータが含まれます。

```
proc fedsql;
  select products.prodid, products.product, sales.totals
    from products, sales
    where products.prodid=sales.prodid;
quit;
```

アウトプット 2.6　指定された列の等価結合

PRODID	PRODUCT	TOTALS
3234	Rice	$189,400
1424	Corn	$555,789
3421	Wheat	$781,183
3421	Wheat	$2,789,654
3975	Barley	$899,453

この等価結合の例では、テーブル Product、Sales、および Customers から列 Product、Totals、および City を選択します。まず、結果セットには、テーブル Products および Sales に存在する列 Prodid の値が一致するデータが含まれます。次に、結果セットは、テーブル Sales および Customers に存在する列 Custid の値が一致するデータと結合されます。

```
proc fedsql;
  select products.product, sales.totals, customers.city
    from products, sales, customers
    where products.prodid=sales.prodid and sales.custid=customers.custid;
quit;
```

アウトプット 2.7　3 つのテーブルの等価結合

PRODUCT	TOTALS	CITY
Rice	$189,400	Boulder
Corn	$555,789	Nagasaki
Wheat	$781,183	Tokyo
Wheat	$2,789,654	Little Rock
Barley	$899,453	Buenos Aires

クロス結合

クロス結合は、2 つのテーブルのデカルト積を生成するリレーショナル結合です。クロス結合は、構文 CROSS JOIN で要求されます。クロス結合は WHERE 句でサブセット化できますが、ON 句は使用できません。

このクロス結合の例では、テーブル Products および Sales からすべての列とすべての行を選択します。これにより、2 つのテーブルの単純結合と同じ結果が得られます。

```
proc fedsql;
  select * from products cross join sales;
quit;
```

アウトプット 2.8　2 つのテーブルのクロス結合

PRODID	PRODUCT	PRODID	CUSTID	TOTALS	COUNTRY
3234	Rice	3234	1	$189,400	United States
1424	Corn	3234	1	$189,400	United States
3421	Wheat	3234	1	$189,400	United States
3422	Oat	3234	1	$189,400	United States
3975	Barley	3234	1	$189,400	United States
3234	Rice	1424	3	$555,789	Japan
1424	Corn	1424	3	$555,789	Japan
3421	Wheat	1424	3	$555,789	Japan
3422	Oat	1424	3	$555,789	Japan
3975	Barley	1424	3	$555,789	Japan
3234	Rice	3421	4	$781,183	Japan
1424	Corn	3421	4	$781,183	Japan
3421	Wheat	3421	4	$781,183	Japan
3422	Oat	3421	4	$781,183	Japan
3975	Barley	3421	4	$781,183	Japan
3234	Rice	3421	2	$2,789,654	United States
1424	Corn	3421	2	$2,789,654	United States
3421	Wheat	3421	2	$2,789,654	United States
3422	Oat	3421	2	$2,789,654	United States

このクロス結合の例では、テーブル Products および Sales から列 Prodid、Product、および Totals を選択します。結果セットには、列 Prodid の値が一致するデータが含まれます。この結果は、2 つのテーブルの等価結合と同じです。

```
proc fedsql;
  select products.prodid, products.product, sales.totals
    from products cross join sales
    where products.prodid=sales.prodid;
quit;
```

アウトプット 2.9 WHERE 句を使用したクロス結合

PRODID	PRODUCT	TOTALS
3234	Rice	$189,400
1424	Corn	$555,789
3421	Wheat	$781,183
3421	Wheat	$2,789,654
3975	Barley	$899,453

限定結合

限定結合によって、結果セットに表示される行を制御する簡単な方法が提供されます。任意の列を使用して、あるテーブルの行を別のテーブルの行とマッチングできます。限定結合は、構文 JOIN を使用して要求され、次に構文 ON または USING を使用して結合条件を指定します。WHERE 句を使用すると、クエリ結果をさらにサブセット化できます。

- ON 句は、データをフィルタリングするための結合条件を指定します。ON 句は、WHERE 句のような条件式などの検索条件を受け入れます。ON 句は、両方のテーブルで列名が一致しないテーブルを結合します。両方のテーブルに存在する列の場合、ON 句は、結合された各テーブルの列を結果セットに個別に保持します。
- USING 句では、等価性をテストする列が指定されます。USING 句にリストされている列は、両方のテーブルに存在している必要があります。USING 句は、修飾子を指定せずに結合条件を定義するための簡略化方法のようなものです。USING 句は、左テーブルの各列が右テーブルの同名の列と比較される結合条件と同等です。両方のテーブルに存在する列の場合、USING 句は、結合されたテーブルの列を単一列にマージします。

限定結合には内部結合と外部結合があり、構文 INNER または OUTER で要求されます。結合タイプの指定を省略すると、内部結合が暗黙的に指定されます。“内部結合タイプと外部結合タイプ” (41 ページ)を参照してください。

この限定結合の例では、テーブル Products および Sales からすべての列を選択します。返される行は、Sales テーブルの列 Country に基づいて、Country の値が United States に等しいという条件でフィルタリングされます。列 Prodid は両方のテーブルに存在し、結果セットで重複しています。

```
proc fedsql;
  select * from products join sales
    on (sales.country='United States');
quit;
```

アウトプット 2.10　ON 句を使用した限定結合

PRODID	PRODUCT	PRODID	CUSTID	TOTALS	COUNTRY
3234	Rice	3234	1	$189,400	United States
3234	Rice	3421	2	$2,789,654	United States
1424	Corn	3234	1	$189,400	United States
1424	Corn	3421	2	$2,789,654	United States
3421	Wheat	3234	1	$189,400	United States
3421	Wheat	3421	2	$2,789,654	United States
3422	Oat	3234	1	$189,400	United States
3422	Oat	3421	2	$2,789,654	United States
3975	Barley	3234	1	$189,400	United States
3975	Barley	3421	2	$2,789,654	United States

この限定結合の例では、テーブル Products および Sales からすべての列を選択します。返された行は、両方のテーブルに存在する列 Prodid に一致する値を選択することによってフィルタリングされます。USING 句は、修飾子を指定せずに結合条件を定義するための簡略化方法のようなものです。USING 句は、左テーブルの各列が右テーブルの同名の列と比較される結合条件と同等です。等価結合やクロス結合とは異なり、列 Prodid は結果セットで重複しません。

```
proc fedsql;
  select * from products join sales
    using (prodid);
quit;
```

アウトプット 2.11　USING 句を使用した限定結合

PRODID	PRODUCT	CUSTID	TOTALS	COUNTRY
3234	Rice	1	$189,400	United States
1424	Corn	3	$555,789	Japan
3421	Wheat	4	$781,183	Japan
3421	Wheat	2	$2,789,654	United States
3975	Barley	5	$899,453	Argentina

この限定結合の例では、テーブル Products および Sales から列 Prodid、Product、および Totals を選択します。返された行は、列 Country に基づいて、値が United States に等しいという条件でフィルタリングされます。返された行は、さらに Product の値が Rice に等しいという条件でサブセット化されます。

```
proc fedsql;
  select products.prodid, products.product, sales.totals
    from products join sales
    on (sales.country='United States')
    where products.product='Rice';
quit;
```

アウトプット 2.12 ON 句と WHERE 句を使用した限定結合

PRODID	PRODUCT	TOTALS
3234	Rice	$189,400
3234	Rice	$2,789,654

自然結合

自然結合では、同じ名前と同じ型を共有する列の値が等しい 2 つのテーブルから行が選択されます。自然結合は、構文 NATURAL JOIN で要求されます。類似列が見つからない場合は、クロス結合が実行されます。自然結合では ON 句を使用しないでください。自然結合を使用すると、ON 句が暗黙的に指定され、類似列がすべて一致します。WHERE 句を使用すると、クエリ結果をサブセット化できます。自然結合は、USING 句を使用した限定結合と同じように機能します。自然結合は USING の簡略化方法です。USING と同様に、類似列は結果セットに一度だけ表示されます。

自然結合には内部結合と外部結合があり、構文 INNER または OUTER で要求されます。結合タイプの指定を省略すると、内部結合が暗黙的に指定されます。“内部結合タイプと外部結合タイプ” (41 ページ)を参照してください。

この自然結合の例では、テーブル Products および Sales からすべての列を選択します。結果セットには、両方のテーブルに存在する列 Prodid の値が一致するデータが含まれます。クロス結合や 2 つのテーブルの単純結合とは異なり、自然結合の結果セットには、重複する Prodid 列は含まれません。

```
proc fedsql;
  select * from products natural join sales;
quit;
```

アウトプット 2.13 すべての列の自然結合

PRODID	PRODUCT	CUSTID	TOTALS	COUNTRY
3234	Rice	1	$189,400	United States
1424	Corn	3	$555,789	Japan
3421	Wheat	4	$781,183	Japan
3421	Wheat	2	$2,789,654	United States
3975	Barley	5	$899,453	Argentina

この自然結合の例では、テーブル Sales および Customers から列 City および Totals を選択します。結果セットには、両方のテーブルに存在する列 Custid と Country の値が一致するデータが含まれます。返された行は、Country の値が United States に等しいサブセットです。

```
proc fedsql;
  select customers.city, sales.totals
    from sales natural join customers
    where customers.country='United States';
quit;
```

アウトプット 2.14　WHERE 句を使用した自然結合

CITY	TOTALS
Boulder	$189,400
Little Rock	$2,789,654

内部結合タイプと外部結合タイプ

内部結合タイプと外部結合タイプの概要

限定結合と自然結合の結果セットは、内部結合または外部結合のいずれかの結合タイプを指定することで影響を受ける可能性があります。デフォルトでは、限定結合と自然結合は内部結合として機能します。

内部結合

内部結合では、2 番目のテーブルの行と一致する 1 番目のテーブルのすべての 行を含む結果セットが返されます。内部結合では、結合条件を満たす行のみが返されます。不一致行は両方のテーブルから破棄されます。デフォルトでは、限定結合と自然結合は内部結合として機能します。構文 INNER を含めても、結果セットに追加の影響はありません。つまり、これらの SELECT ステートメントの結果は同じです。

```
select * from products inner join sales
  on (sales.country='United States');

select customers.city, sales.totals
  from sales natural inner join customers
  where country='United States';
```

外部結合

外部結合では、結合条件を満たすすべての行と、一方または両方のテーブルの不一致行を含む結果セットが返されます。外部結合には、左外部結合、右外部結合、完全外部結合があります。内部結合では、結合条件が満たされない行は破棄されますが、外部結合では、不一致行の一部またはすべてが維持されます。

外部結合の場合、指定された WHERE 句が結合の実行後に適用され、WHERE 句を満たさない行はすべて削除されます。WHERE 句は外部結合で保持される行そのものを削除するため、WHERE 句を外部結合に適用すると目的を達成できないことがあります。

- 左外部結合は、SELECT ステートメントにリストされた最初のテーブルである左テーブルからの不一致行を保持します。左外部結合では、結合条件を満たすすべ

ての行と、結合条件に一致しない左テーブルの行を含む結果セットが返されます。したがって、左外部結合では、左テーブルからはすべての 行、右テーブルからは一致する行のみが返されます。左外部結合は、構文 LEFT [OUTER]で要求されます。

この限定結合の例では、結合条件を満たす両方のテーブルのすべての 行を含む結果セットが返されます。結合条件で、値が United States と等しい列 Country に基づいて行がフィルタリングされます。結果セットには、結合条件に一致しない Customers テーブルの行も含まれます。左外部結合として、Customers テーブルのすべての行が返されます。

```
proc fedsql;
  select customers.city, sales.totals
    from customers left outer join sales
    on (customers.country='United States');
quit;
```

CITY	TOTALS
Boulder	$189,400
Little Rock	$189,400
Boulder	$555,789
Little Rock	$555,789
Boulder	$781,183
Little Rock	$781,183
Boulder	$2,789,654
Little Rock	$2,789,654
Boulder	$899,453
Little Rock	$899,453
Nagasaki	.
Tokyo	.
Buenos Aires	.

この自然結合の例では、結合条件を満たす両方のテーブルのすべての 行を含む結果セットが返されます。これには、列 Prodid の値が一致するデータが含まれます。結果セットには、結合条件に一致しない Sales テーブルの行も含まれます。左外部結合として、Sales テーブルのすべての 行が返されます。

```
proc fedsql;
  select * from sales natural left outer join products;
quit;
```

PRODID	CUSTID	TOTALS	COUNTRY	PRODUCT
3234	1	$189,400	United States	Rice
1424	3	$555,789	Japan	Corn
3421	4	$781,183	Japan	Wheat
3421	2	$2,789,654	United States	Wheat
3975	5	$899,453	Argentina	Barley

- 右外部結合は、SELECT ステートメントにリストされた 2 番目のテーブルである右テーブルからの不一致行を保持します。右外部結合では、結合条件を満たすすべての行と、結合条件に一致しない右テーブルの行を含む結果セットが返されます。したがって、右外部結合では、右テーブルからはすべての 行、左テーブルからは一致する行のみが返されます。右外部結合は、構文 RIGHT [OUTER]で要求されます。

  この限定結合の例では、結合条件を満たす両方のテーブルのすべての 行を含む結果セットが返されます。結合条件で、値が United States と等しい列 Country に基づいて行がフィルタリングされます。結果セットには、結合条件に一致しない Sales テーブルの行も含まれます。右外部結合として、Sales テーブルのすべての行が返されます。

```
proc fedsql;
  select * from products right outer join sales
    on (sales.country='United States');
quit;
```

PRODID	PRODUCT	PRODID	CUSTID	TOTALS	COUNTRY
3234	Rice	3234	1	$189,400	United States
3234	Rice	3421	2	$2,789,654	United States
1424	Corn	3234	1	$189,400	United States
1424	Corn	3421	2	$2,789,654	United States
3421	Wheat	3234	1	$189,400	United States
3421	Wheat	3421	2	$2,789,654	United States
3422	Oat	3234	1	$189,400	United States
3422	Oat	3421	2	$2,789,654	United States
3975	Barley	3234	1	$189,400	United States
3975	Barley	3421	2	$2,789,654	United States
.		1424	3	$555,789	Japan
.		3421	4	$781,183	Japan
.		3975	5	$899,453	Argentina

この自然結合の例では、結合条件を満たす両方のテーブルのすべての 行を含む結果セットが返されます。これには、列 Prodid の値が一致するデータが含まれます。結果セットには、結合条件に一致しない Sales テーブルの行も含まれます。右外部結合として、Sales テーブルのすべての 行が返されます。

```
proc fedsql;
```

```
  select * from products natural right outer join sales;
quit;
```

PRODID	PRODUCT	CUSTID	TOTALS	COUNTRY
3234	Rice	1	$189,400	United States
1424	Corn	3	$555,789	Japan
3421	Wheat	4	$781,183	Japan
3421	Wheat	2	$2,789,654	United States
3975	Barley	5	$899,453	Argentina

- 完全外部結合では、両方のテーブルの不一致行が保持されます。つまり、完全外部結合では、左と右のテーブルからすべての 一致行と不一致行が返されます。完全外部結合は、構文 FULL [OUTER]で要求されます。

  この限定結合の例では、結合条件を満たす両方のテーブルのすべての 行を含む結果セットが返されます。結合条件で、値 Rice を含む列 Product に基づいて行がフィルタリングされます。結果セットには、結合条件に一致しない両方のテーブルのすべての行も含まれます。完全外部結合として、両方のテーブルのすべての行が返されます。

```
proc fedsql;
  select * from products full outer join sales
    on (products.product='Rice');
quit;
```

PRODID	PRODUCT	PRODID	CUSTID	TOTALS	COUNTRY
3234	Rice	3234	1	$189,400	United States
3234	Rice	1424	3	$555,789	Japan
3234	Rice	3421	4	$781,183	Japan
3234	Rice	3421	2	$2,789,654	United States
3234	Rice	3975	5	$899,453	Argentina
1424	Corn	.	.	.	
3421	Wheat	.	.	.	
3422	Oat	.	.	.	
3975	Barley	.	.	.	

  この自然結合の例では、結合条件を満たす両方のテーブルのすべての 行を含む結果セットが返されます。これには、列 Prodid の値が一致するデータが含まれます。結果セットには、結合条件に一致しない Sales テーブルの行と Products テーブルの行も含まれます。完全外部結合として、両方のテーブルのすべての 行が返されます。

```
proc fedsql;
  select * from products natural full outer join sales;
quit;
```

PRODID	PRODUCT	CUSTID	TOTALS	COUNTRY
1424	Corn	3	$555,789	Japan
3234	Rice	1	$189,400	United States
3421	Wheat	4	$781,183	Japan
3421	Wheat	2	$2,789,654	United States
3422	Oat	.	.	
3975	Barley	5	$899,453	Argentina

異種データの結合

一般的な組織は複数のデータベースにデータを保存するため、FedSQL は異種データの結合をサポートしています。異種結合は、結合操作のテーブルが異なるデータソースに存在する場合に発生します。異種結合は、フェデレーションクエリの一種です。

すべての FedSQL 結合操作は、異種結合として実行できます。たとえば、次の自然結合は、Oracle テーブルと Teradata テーブル間の異種結合です。

```
select * from oracle.product natural left outer join tera.sales;
```

この異種結合の例では、SAS データセットと Oracle テーブルの両方に対してクエリを実行します。PROC FEDSQL は、libref の属性を使用して 2 つのデータソースに接続します。SAS データセット MyBase.Products には、列 Prodid および Product が含まれています。Oracle テーブル Oracle.Sales には、列 Prodid、Totals、および Country が含まれています。SELECT ステートメントは、両方のテーブルに存在する列 Prodid に基づいて、両方のテーブルの列と行を結合します。

```
libname mybase base '/tmp/Base';
libname myoracle oracle user=scott password=tiger path=oraclev11;

proc fedsql;
  select mybase.products.prodid, mybase.products.product, myoracle.sales.totals
    from mybase.products, myoracle.sales
    where products.prodid=sales.prodid;
quit;
```

アウトプット 2.15　SAS データセットと Oracle テーブルの FedSQL 異種結合

PRODID	PRODUCT	TOTALS
3234	Rice	$189,400
1424	Corn	$555,789
3421	Wheat	$781,183
3421	Wheat	$2,789,654
3975	Barley	$899,453

この異種結合の例では、3 つのテーブル(2 つの Oracle テーブルと 1 つの SAS データセット)に対してクエリを実行します。クエリでは、MyOracle.Products と MyOracle.Customers の結合は Oracle によって実行されます。Oracle 結果セットと SAS データセット MyBase.Sales の結合は、FedSQL によって実行されます。

```
libname mybase base '/tmp/Base';
libname myoracle oracle user=scott password=tiger path=oraclev11;

proc fedsql;
  select myoracle.products.product, mybase.sales.totals, myoracle.customers.city
    from myoracle.products, mybase.sales, myoracle.customers
    where products.prodid=sales.prodid and sales.custid=customers.custid;
quit;
```

アウトプット 2.16　2 つの Oracle テーブルと SAS データセットの FedSQL 異種結合

PRODUCT	TOTALS	CITY
Rice	$189,400	Boulder
Corn	$555,789	Nagasaki
Wheat	$781,183	Tokyo
Wheat	$2,789,654	Little Rock
Barley	$899,453	Buenos Aires

FedSQL 式

クエリ式とサブクエリ

クエリ式の概要

*クエリ式*または*クエリ*は、結果セットを生成する 1 つ以上の SELECT ステートメントで構成されます。複数の SELECT ステートメントを組み合わせる方法の 1 つに、サブクエリの使用があります。もう 1 つは、セット演算子を使用する方法です。このセクションでは、セット演算子について説明します。サブクエリについては、“サブクエリの概要” (51 ページ)を参照してください。

FedSQL は、UNION、EXCEPT、および INTERSECT セット演算子をサポートしています。

- UNION セット演算子は、2 つのクエリの結果を結合して、両方のクエリからの一意の行を含む単一の結果セットにします。

- EXCEPT 演算子は、2 つのクエリの結果を結合して、最初のクエリからの一意の行を含む単一の結果セットにします。
- INTERSECT 演算子は、2 つのクエリの結果を結合して、両方のクエリに共通の行の一意のインスタンスを含む単一の結果セットにします。

デフォルトでは、列は、列名に関係なく、入力テーブル内の位置によって結合されます。2 つのクエリで同じ相対位置にある列は、互換性のあるデータ型と長さを持っている必要があり、各クエリは、セット操作を成功させるために同じ数の列を持っている必要があります。結果セットには、クエリ式の結果である一意の行のみが含まれます。つまり、セット演算子は結果セットから重複する行を自動的に削除します。

セット演算子は、デフォルト動作を変更するためのオプションキーワードをサポートしています。

表 2.14 UNION、EXCEPT、および INTERSECT セット演算子のオプションキーワード

キーワード	目的
ALL	一意の行だけではなく、重複も含めて、両方のクエリからすべての行を返します。
CORRESPONDING	クエリの列を、位置ではなく列名で結合します。列名が不一致の列はすべて無視されます。CORRESPONDING キーワードは、一致する名前を持つすべての列の行を返します。
CORRESPONDING および BY	列を列名で結合し、列リストから列のみを返します。

セット演算子を使用したクエリ式は、次のように評価されます。

- 各 SELECT ステートメントが評価され、仮想の中間結果セットが生成されます。
- 次に、各中間結果セットは、式を形成するためにセット演算子にリンクされたオペランドになります(例: A UNION B)。(CORRESPONDING キーワードと BY キーワードを一緒に指定すると、列リストにない列は削除されます。)
- クエリ式に 2 つを超える SELECT ステートメントが含まれる場合、最初の 2 つのクエリの結果は、(A UNION B) EXCEPT C、((A UNION B) EXCEPT C) INTERSECT D など、次のセット演算子およびオペランドのオペランドになります。
- クエリ式を評価すると、単一の出力結果セットが生成されます。

デフォルトのクエリ式の動作

クエリ式がどのように機能するかを理解するために、次の例を考えてみましょう。この例では、テーブル Numbers1 と Numbers2 を操作します。次の図は、これらのテーブルのコンテンツを示しています。

図 2.1 テーブル Numbers1 と Numbers2 のコンテンツ

Numbers1

X	Y	Z
one	1	10
three	3	30
four	4	40
four	4	40
five	5	50
five	5	50

Numbers2

X	Y	Z
four	4	40
five	5	50
five	5	50
one	1	10
two	2	20

これらのテーブル表の列は同じ型で同じ位置にあることに注意してください。X は文字列、Y と Z は数値列です。

次のコード例では、UNION 演算子を指定しています。

```
proc fedsql;
  select * from numbers1 union select * from numbers2;
quit;
```

UNION 演算子は、両方のテーブルから一意の行を返し、単一の仮想テーブルとして表示します。

アウトプット 2.17 UNION 操作からの出力

X	Y	Z
five	5	50
four	4	40
one	1	10
three	3	30
two	2	20

次のコード例では、EXCEPT 演算子を指定しています。

```
proc fedsql;
  select * from numbers1 except select * from numbers2;
quit;
```

EXCEPT 演算子は、最初のテーブルには存在するが 2 番目のテーブルには存在しない一意の行を返します。

アウトプット 2.18 EXCEPT 操作からの出力

X	Y	Z
three	3	30

次のコード例では、INTERSECT 演算子を指定しています。

```
proc fedsql;
  select * from numbers1 intersect select * from numbers2;
quit;
```

INTERSECT 演算子は、指定されたテーブルに共通する行の一意のインスタンスを返します。

アウトプット 2.19　INTERSECT 操作からの出力

X	Y	Z
five	5	50
four	4	40
one	1	10

次に、3 番目のテーブル Numbers3 について考えてみましょう。

図 2.2　Numbers3 テーブルのコンテンツ

Z	Y	X
40	4	four
60	6	six
70	7	seven

テーブル Numbers3 には、テーブル Numbers1 および Numbers2 と同じ名前と種類の列が含まれていますが、列の順序は異なります。テーブル Numbers1 または Numbers2 をテーブル Numbers3 と結合する要求は失敗します: ERROR: types CHAR and DOUBLE cannot be matched.

CORRESPONDING キーワードを使用したクエリの結合

CORRESPONDING キーワードは、入力テーブルの列を列名で結合するため、列が同じ位置にある必要はありません。また、一致する列は互換性のあるデータ型である必要があります。たとえば、VARCHAR 型の列の行を CHAR 型の列の行と結合することができます。入力テーブルの列数が同じである必要はありません。

CORRESPONDING キーワードを使用する UNION セット操作の例を次に示します。要求は、テーブル Numbers1 と Numbers3 の行を結合します。これは、“デフォルトのクエリ式の動作” (47 ページ)で紹介されています。

```
proc fedsql;
  select * from Numbers1 UNION CORRESPONDING select * from Numbers3;
quit;
```

アウトプット 2.20 CORRESPONDING キーワードを使用した UNION からの出力

X	Y	Z
five	5	50
four	4	40
one	1	10
seven	7	70
six	6	60
three	3	30

UNION CORRESPONDING 操作は、両方の列から一意の行を返します。結果セットの列は、式の最初のクエリの列順序を使用します。

CORRESPONDING キーワードを使用する、同じテーブルに対する EXCEPT セット操作の例を次に示します。

```
proc fedsql;
  select * from Numbers1 EXCEPT CORRESPONDING select * from Numbers3;
quit;
```

アウトプット 2.21 CORRESPONDING キーワードを使用した EXCEPT 操作からの出力

X	Y	Z
five	5	50
one	1	10
three	3	30

EXCEPT CORRESPONDING 操作は、最初のクエリの列順序を使用して、最初のクエリに固有の行を返します。

CORRESPONDING キーワードを指定するテーブル Numbers1 および Number3 に対する INTERSECT セット操作の例を次に示します。

```
proc fedsql;
  select * from Numbers1 INTERSECT CORRESPONDING select * from Numbers3;
quit;
```

アウトプット 2.22 CORRESPONDING キーワードを使用した INTERSECT 操作からの出力

X	Y	Z
four	4	40

結果セットには、2 つの入力テーブルに共通する一意の行のインスタンスが含まれます。

BY キーワードも指定する INTERSECT CORRESPONDING BY セット操作の例を次に示します。このコードは、列 Y の値を要求します。

```
proc fedsql;
  select * from Numbers1 INTERSECT CORRESPONDING BY (Y) select * from Numbers3;
quit;
```

アウトプット 2.23　CORRESPONDING および BY キーワードを使用した INTERSECT からの出力

Y
4

セット演算子の結合

セット演算子を一緒に使用する方法を説明する例を次に示します。

```
proc fedsql;
  select * from numbers1
    intersect select * from numbers2
    union corresponding select * from numbers3;
quit;
```

最初のクエリ式は、テーブル Numbers1 と Numbers2 に共通する行の一意のインスタンスを選択します。2 番目のクエリ式は、テーブル Numbers3 の一意の行を中間結果セットとマージします。CORRESPONDING キーワードを使用すると、テーブル Numbers3 を中間結果セットに対して評価できます。

サブクエリの概要

*サブクエリ*とは、別のクエリ式の一部としてネストされているクエリ式です。かっこ内に指定され、値を返すことを目的としています。サブクエリは、アトミック値(1 列に 1 行が含まれ、*スカラークエリ*とも呼ばれる)、行値(1 つまたは複数の列に対して 1 行)、またはテーブル値(1 つまたは複数の列に対して 1 つまたは複数の行)を返すことができます。

サブクエリは、SELECT、INSERT、UPDATE、DELETE ステートメントで使用できます。サブクエリの目的は、あるテーブルのコンテンツが別のテーブルのクエリまたはアクションに影響を与えることを可能にすることです。

サブクエリは、クエリ内のさまざまな場所で使用できます。

- SELECT ステートメント
- WHERE 句
- HAVING 句
- FROM 句

スカラーサブクエリは、スカラー値を使用できる場所であればどこでも、4 つの場所すべてで指定できます。行値を返すサブクエリは、通常、WHERE 句で指定されます。テーブル値を返すサブクエリは、FROM 句で指定されます。

サブクエリは、外部クエリに依存することも、独立することもできます。サブクエリで追跡される情報が、外部クエリで認識されているデータになんらかの形で依存している場合、そのデータは外部クエリと相関があると言います。*相関サブクエリ*は通常、外部クエリのデータを相関名で参照するか、EXISTS または IN 述語を使用して、外部クエリのデータを使用します。外部クエリによって処理されるデータが変更されると、相関サブクエリによって取得される情報も変更されます。相関サブクエリは、外部クエリによって識別される行ごとに評価されるため、サブクエリはリソースを大量に消費します。多くの相関クエリは、結合操作に関して言い換えることができます。

外部クエリに依存しないサブクエリは、非相関クエリと呼ばれます。非相関サブクエリには、残りのクエリで蓄積されるデータとの相互作用はあまりありません。非相関サブクエリは一度だけ評価され、その結果は外部クエリの評価で繰り返し使用されます。最も重要なのは、外部クエリによって処理されたデータが変更されても、サブクエリの結果が変わらないことです。

サブクエリはネストできます。クエリ式で複数のサブクエリが使用されている場合、最初に最も内側のクエリが評価され、次に内側から 2 番目のクエリが続き、その後も同様に外側に向かって順番に処理されます。

相関サブクエリの例

これは、 EXISTS 述語と Depts を指定する相関サブクエリの例です。Employees と Depts テーブルのデータを使用します。 これらのテーブルを作成する方法については、“Employees” (1262 ページ)および“Depts” (1261 ページ)を参照してください。

```
proc fedsql;
  select *
  from employees e
  where exists(select * from depts d
        where d.deptno = e.dept
        and e.pos <> 'Manager');
quit;
```

EXISTS 述語は、内部クエリにも存在する値に対してのみ、外部クエリのテーブルから情報を返すことを規定しています。2 番目の WHERE 句が列参照 *e.dept* を使用していることに注意してください。外部クエリの FROM 句を見ると、*e* が外部クエリで使用されている Employees テーブルに関連付けられた相関名であることがわかります。

アウトプット 2.24　EXISTS 述語を使用した相関クエリの出力

EMPID	DEPT	EMP_NAME	POS	HIRE_DATE
5	20	Greg Welty	Developer	26NOV2004
6	20	Penny Jackson	Developer	26NOV2004
7	10	Edward Murray	Sales Associate	26NOV2004
8	10	Ronald Thomas	Sales Associate	26NOV2004
9	30	Elsie Marks	Executive Assistant	11FEB2005
10	40	Bruno Kramer	Grounds support technician	02NOV2005

このクエリを記述するより効率的な方法を次に示します。

```
proc fedsql;
  select *
    from employees
    where dept in (select deptno from depts
      where pos <> 'Manager');
quit;
```

このクエリには相関名は必要ありません。Dept 列と Deptno 列の内容は、IN 述語で比較されます。クエリは、前のクエリと同じ結果を返します。一方、内部クエリを一度実行すれば、その結果を Employees テーブルの各行と比較できます。

スカラーサブクエリの例

スカラーサブクエリは、1 つの列に対して 1 つの値を返します。そのため、集計クエリに役立ちます。

HAVING 句のスカラーサブクエリの例を次に示します。この例では、“相関サブクエリの例” (52 ページ)で使用された Employees テーブルと Depts テーブルを使用します。

```
proc fedsql;
  select dept, count(emp_name)
    from mybase.employees e
    group by dept
    having dept in (select deptno from mybase.depts);
quit;
```

これは単純な要求です。Employees テーブルの従業員数をカウントし、そのカウントを部門ごとにグループ化し、Depts (1261 ページ)テーブルの Deptno 列に一致する値を持つ部門に関する情報のみを返します。

アウトプット 2.25　部門別従業員数

DEPT	COUNT
40	2
20	3
30	2
10	3

SELECT ステートメントで指定されたスカラーサブクエリの例を次に示します。

```
proc fedsql;
  select d.deptno,
     d.deptname,
     d.manager,
     (select count (*) from employees e
     where e.dept = d.deptno and
     e.pos <> 'Manager') as Employees
     from depts d;
quit;
```

サブクエリは、Employees テーブルのレコード数をカウントし、マネージャーに属する従業員レコードを削除してから、両方のテーブルで一致する部門コードごとに集計値を返します。つまり、部門ごとに 1 つの値が返されます。これも相関サブクエリです。

アウトプット 2.26　マネージャー別従業員数

DEPTNO	DEPTNAME	MANAGER	EMPLOYEES
10	Sales	Jim Barnes	2
20	Research	Clifford James	2
30	Accounting	Barbara Sandman	1
40	Operations	William Baylor	1

WHERE 句のスカラーサブクエリの例を次に示します。

```
proc fedsql;
  select *
  from depts d
  where (select COUNT(*) FROM employees e
     WHERE d.deptno = e.dept
     and e.pos <> 'Manager') > 1;
quit;
```

前の例のように、サブクエリは、Employees (1262 ページ)テーブルのすべての 行からマネージャーを表す行を除いたものをカウントし、Depts テーブルの DeptNo 列に関連付けます。 ただし、WHERE 句はクエリをさらに修飾して、複数の従業員がいる部門に関する情報のみを取得します。

アウトプット 2.27　複数の従業員がいる部門

DEPTNO	DEPTNAME	MANAGER
10	Sales	Jim Barnes
20	Research	Clifford James

INSERT ステートメントのスカラーサブクエリの例を次に示します。この例では、Population という列を持つテーブル Densities (1260 ページ)のデータを使用します。新しいテーブル Summary が作成され、Densities テーブルの Population 列から集計された値が入力されます。Densities テーブルの詳細については、“Densities” (1260 ページ)を参照してください。

```
proc fedsql;
  drop table summary force;
  create table summary (
    sum_pop double having format comma12.,
    max_pop double having format comma12.,
    min_pop double having format comma12.,
    avg_pop double having format comma12.);

  insert into summary (
    sum_pop,
    max_pop,
    min_pop,
    avg_pop)
    values (
      (select sum(population) from densities),
      (select max(population) from densities),
      (select min(population) from densities),
      (select avg(population) from densities)
);
select * from summary;
quit;
```

INSERT ステートメントの各サブクエリは、スカラーサブクエリです。各サブクエリは、Population という 1 つの列のデータから 1 つの値(合計値、最大値、最小値、平均値)を返し、新しいテーブルに入力します。

アウトプット 2.28　Summary テーブルのコンテンツ

SUM_POP	MAX_POP	MIN_POP	AVG_POP
122,775,442	34,248,705	64,634	12,277,544

非相関クエリの例

非相関サブクエリは外部クエリの前に実行され、サブクエリは 1 回だけ実行されます。外部クエリとサブクエリのデータは独立しており、サブクエリを 1 回実行すると、外部クエリのすべての行が処理されます。

このセクションの例では、次の 3 つのテーブルのデータを使用します。

```
data All1;
```

```
  input x c $;
datalines;
1 one
2 two
3 three
4 four
5 five
6 six
;
run;

data One;
  x=1;
run;

data Two;
  x=2;
  c="two";
run;
```

アウトプット 2.29　テーブル All1、One、および Two

Table All1

x	c
1	one
2	two
3	three
4	four
5	five
6	six

Table One

x
1

Table Two

x	c
2	two

このコード例では、SELECT ステートメントでサブクエリを指定しています。サブクエリは、テーブル One に存在するテーブル All1 の値のみを表示するように指定します。テーブル All1 は変更されません。クエリは、出力された行のみに影響します。

```
proc fedsql;
  select * from all1 where x in (select x from one);
quit;
```

アウトプット 2.30　SELECT サブクエリの出力

x	c
1	one

次のコード例では、INSERT ステートメントでサブクエリを指定しています。

```
proc fedsql;
  insert into two select * from all1 where exists (select * from one);
  select * from two;
quit;
```

この例では、テーブル Two のコンテンツが拡張され、テーブル One にも存在するテーブル All1 のコンテンツが含まれます。

アウトプット 2.31　INSERT サブクエリの結果

x	c
2	two
1	one
2	two
3	three
4	four
5	five
6	six

このコード例では、DELETE ステートメントでサブクエリを指定しています。

```
proc fedsql;
  delete from all1 where x in (select x from one);
  select * from all1;
quit;
```

このコードは、テーブル One の列 X の値と一致する値をテーブル All1 から削除するように指定します。

アウトプット 2.32　DELETE サブクエリの結果

x	c
2	two
3	three
4	four
5	five
6	six

次のコード例では、UPDATE ステートメントでサブクエリを指定しています。

```
proc fedsql;
  update all1 set x = (select x from one);
  select * from all1;
quit;
```

テーブル All1 の列 X は、テーブル Two の列 X の値で更新されます。

アウトプット 2.33 UPDATE サブクエリの結果

x	c
1	two
1	three
1	four
1	five
1	six

非相関サブクエリを使用すると、2 つ以上のテーブル間で関連のないデータブロックを選択、挿入、削除、変更できます。

FedSQL 値式

数値式

*数値式*を使用すると、加算(+)、減算(–)、乗算(*)、除算(/)演算子を使用して数値を計算できます。数値は数値リテラルにすることができます。これらの値は、列名、変数、またはサブクエリが数値に評価される限り、列名、変数、またはサブクエリにすることもできます。

数値式の結果のデータ型は、オペランドのデータ型に基づいています。

数値式の例を次に示します。

- -6
- salary * 1.07
- cost + (exp – discount)

行値式

行値式、または*行値コンストラクター*は、かっこで囲まれた 1 つ以上の値式です。複数の値式はカンマで区切られます。

行値コンストラクターには、次の値を含めることができます。

- *value-expression*
- NULL
- DEFAULT
- *row-subquery*

- (*row-value-constructor*, *row-value-constructor*, ...)
- ROW (*row-value-constructor*, *row-value-constructor*, ...)

NULL は、テーブルの対応する列の値を null にします。DEFAULT は、対応する列の値をデフォルト値にします。ARRAY[ ]は、宛先が配列であり、空の配列を作成する場合にのみ有効です。NULL、DEFAULT、および ARRAY []以外の行コンストラクター値は、単純な値または値式にすることができます。

行値コンストラクターは、単一の値や列ではなく、値または列のリストを操作します。一度に行全体または行のサブセットを操作できます。

ここでは、INSERT ステートメントで行値コンストラクターを使用してテーブルに複数の値を追加できる例を紹介します。

```
INSERT INTO inventory
  (prodname, qty, price)
  VALUES ('rice', 3849, .37);
```

この例では、WHERE 句で行値コンストラクターを使用して列値を比較します。

```
SELECT * FROM inventory
  WHERE (inventory.prodname, inventory.price)
    =
      (competitor.prodname, inventory.price);
```

FedSQL の日付と時間

FedSQL の日付と時間の概要

FedSQL は、DATE、TIME、および TIMESTAMP データ型を使用して、日付、時間、および日時に関する業界標準の規則をサポートしています。FedSQL でサポートされているすべてのサードパーティデータソースは 、これらのデータ型をサポートしています。これらのデータ型を使用して、SAS データセット、SPD Engine データセット、および SPD Server テーブルを作成できますが、FedSQL は、これらのデータ型を SAS 日付、時間、または日時出力形式を持つ DOUBLE に変換します。

FedSQL は、SAS 日付、時間、および日時値を、データ型が DOUBLE の値としてのみ読み取り、処理できます。FedSQL は、データ型が DOUBLE の SAS 日付、時間、または日時値を、データ型が DATE、TIME、または TIMESTAMP の値に変換できません。

FedSQL の日付、時間、および日時関数は、ローカルの現在の時間と GMT の現在の時間を提供できます。

FedSQL 出力形式を使用すると、日付、時間、日時をさまざまな方法でフォーマットできます。

FedSQL の日付、時間、および日時の定数

次の構文を使用して、FedSQL の日付、時間、または日時定数を記述します。

DATE'*yyyy-mm-dd*'

TIME'*hh*:*nn*:*ss*[.*fraction*]'

TIMESTAMP' *yyyy-mm-dd hh*:*nn*:*ss*[.*fraction*]'

ここでは

- *yyyy* は 4 桁の年
- *mm* は 2 桁の月(01-12)
- *dd* は 2 桁の日(01-31)
- *hh* は 2 桁の 24 時間表示の時間(00-23)
- *nn* は 2 桁の分(00-59)
- *ss* は 2 桁の秒(00-59)
- *fraction* は 1-10 桁の 0–9 で、オプションであり、秒の小数部を表す

DATE、TIME、または TIMESTAMP キーワードの後の値の文字列部分は、単一引用符で囲む必要があります。

日付定数では、ハイフンが必要であり、日付文字列の長さは 8 以上にする必要があります。1 桁の日付の前にゼロ(0)を付ける必要はありません。

時間定数では、コロンが必要です。秒の小数部が存在しない場合、時間文字列は 8 文字の長さである必要があり、ピリオドは含めることも除外することもできます。秒の小数部が存在する場合、その小数部は最大 9 桁の長さになります。時間定数は 8 文字から 18 文字までの長さにすることができます。

タイムスタンプ定数では、日付のハイフンと時間のコロンが必要です。秒の小数部が存在しない場合、ピリオドは含めることも除外することもできます。秒の小数部が存在する場合、その小数部は最大 9 桁の長さになります。タイムスタンプ定数は 19 文字から 29 文字までの長さにすることができます。

FedSQL の日付、時間、およびタイムスタンプ定数の例を次に示します。

```
date'2008-01-31'
date'2008-1-1'
time'20:44:59'
timestamp'2007-02-02 07:00:00.7569'
```

キーワード DATETIME は、タイムスタンプ定数の TIMESTAMP のエイリアスとして使用できます。次に例を示します: datetime'2007-02-02 07:00:00.7569'。

日時値を含むテーブルを作成する例を次に示します。

```
create table bikerace (name char(30), entry_number int,
  registration_date timestamp);
  insert into bikerace values ('Andersen, Mark', 342,
    timestamp'2007-03-15 12:27:33');
  insert into bikerace values ('Steinbek, Mark', 244,
```

```
timestamp'2006-11-27 13:26:19');
```

日付またはタイムスタンプ定数の年値が 4 桁を超える場合、FedSQL はエラーメッセージを返しません。たとえば、値 timestamp'20211-01-01 12:34:56'を挿入しても、エラーメッセージは返されません。長すぎる年値が挿入されると、結果の出力値が異なる場合があります。また、結果が予測不能で不正確になる可能性もあります。したがって、データを挿入する前に、年値が 4 桁のみであることを確認してください。

SAS データセット、SPD Engine データセット、および SPD Server テーブルでの日付と時間の使用

BASE ドライバーまたは SPD ドライバーを使用してテーブルを作成する場合、DATE、TIME、および TIMESTAMP データ型を使用して、日付、時間、および日時値の列を指定します。FedSQL は、これらのデータ型の値を、これらの SAS 出力形式を持つ DOUBLE のデータ型に変換します。

表 2.15 SAS データセット、SPD Engine データセット、および SPD Server テーブルの日付と時間の値に割り当てられた SAS 出力形式

FedSQL 日付/時間データ型	SAS 出力形式
DATE	DATE9.
TIME	TIME8.
TIMESTAMP	DATETIME19.2

これらの SAS 出力形式は、FedSQL 言語の ALTER TABLE ステートメントを使用して変更することはできません。これらは、Base SAS のプロトコルを使用して、SAS データセット、SPD Engine データセット、または SPD Server テーブルの列に割り当てられた出力形式を変更することによってのみ変更できます。

次の例では、DATE、TIME、および TIMESTAMP データ型を使用して日付、時間、および日時の値を含む SAS データセットを作成し、SAS 出力形式を使用してそれらがどのように表示されるかを示します。

```
proc fedsql;
  create table basedt (d date, t time, ts timestamp);
  insert into basedt values (date '2013-03-14', time '10:31:22',
    timestamp '2013-03-14 13:30:33.222');
  select * from basedt;
quit;
```

出力は次のとおりです。

アウトプット 2.34　日付、時間、およびタイムスタンプの結果

D	T	TS
14MAR2013	10:31:22	14MAR13:13:30:33.22

これらの日付、時間、および日時の値に指定する出力形式は、これらの値で保存されている出力形式をオーバーライドします。次の SELECT ステートメントでは、PUT 関数を使用して、日時値を DATETIME21.2 出力形式でフォーマットし、4 文字の 2008 年を表示します。

```
proc fedsql;
  select put(ts,datetime21.2) from basedt;
quit;
```

出力は次のとおりです。

アウトプット 2.35　PUT ステートメントを使用した結果

column
14MAR2013:13:30:33.22

日付、時間、および日時関数

FedSQL の日付、時間、および日時関数のリストと説明を次に示します。

表 2.16　日付、時間、および日時関数

関数	説明
CURRENT_DATE	タイムゾーンの現在の日付を返します。
CURRENT_TIME	タイムゾーンの現在の時間を返します。
CURRENT_TIME_GMT()	現在の GMT 時間を返します。
CURRENT_TIMESTAMP	タイムゾーンの現在の日時を返します。
CURRENT_TIMESTAMP_GMT()	現在の GMT 日時を返します。
DAY	日付または日時値から月の日を数値で返します。
HOUR	時間または日時値から時間を返します。
MINUTE	時間または日時値から分を返します。

関数	説明
MONTH	日付または日時値から月を数値で返します。
SECOND	時間または日時値から秒を返します。
YEAR	日付または日時値から年を返します。

SAS データセット、SPD Engine データセット、および SPD Server テーブルの日付、時間、および日時出力形式

SAS データセット、SPD Engine データセット、および SPD Server テーブルのフォーマットには、次の日付、時間、および日時出力形式を使用できます。

表 2.17 種類別に並べた日付、時間、および日時関数

種類	言語要素	入力	フォーマットされた結果
日付出力形式	DATE.	date'2013-03-17'	17MAR13
	DATE9.	date'2013-03-17'	17MAR2013
	DATEAMPM21.	timestamp'2013-03-17 14:22:21'	17MAR13:02:22:21 PM
	DATEAMPM23.	timestamp'2013-03-17 14:22:21'	17MAR2013:02:22:21 PM
	DAY.	date'2013-03-17'	17
	DDMMYY.	date'2013-03-17'	2017/03/13
	DDMMYB.	date'2013-03-17'	17 03 13
	DDMMYYC.	date'2013-03-17'	17:03
	DDMMYYD10.	date'2013-03-17'	17-03-2013
	DDMMYYN.	date'2013-03-17'	17032013

種類	言語要素	入力	フォーマットされた結果
	DDMMYYP.	date'2013-03-17'	17.03.13
	DDMMYYS.	date'2013-03-17'	2017/03/13
	DOWNAME.	date'2013-03-17'	Monday
	JULIAN.	date'2013-03-17'	13077
	MMDDYY.	date'2013-03-17'	2013/03/17
	MMDDYY10.	date'2013-03-17'	2013/03/17
	MMDDYYB.	date'2013-03-17'	03 17 13
	MMDDYC.	date'2013-03-17'	03:17
	MMDDYD.	date'2013-03-17'	03-17-13
	MMDDYN.	date'2013-03-17'	03172013
	MMDDYYP.	date'2013-03-17'	03.17.13
	MMDDYYS.	date'2013-03-17'	2013/03/17
	MMYY.	date'2013-03-17'	03M2013
	MMYYC.	date'2013-03-17'	03:2013
	MMYYD.	date'2013-03-17'	03-2013
	MMYYN.	date'2013-03-17'	032013
	MMYYP.	date'2013-03-17'	03.2013
	MMYYS.	date'2013-03-17'	03/2013
	MONNAME.	date'2013-03-17'	March
	MONTH.	date'2013-03-17'	3
	MONYY.	date'2013-03-17'	MAR13
	WEEKDATE.	date'2013-03-17'	2013/03/17
	WEEKDATX.	date'2013-03-17'	2013/03/17

種類	言語要素	入力	フォーマットされた結果
	WEEKDAY.	date'2013-03-17'	2
日時出力形式	DTDATE.	timestamp'2013-03-18 14:22:21'	17MAR13
	DTMONYY.	timestamp'2013-03-18 14:22:21'	MAR13
	DTWKDATX.	timestamp'2013-03-18 14:22:21'	2013/03/17
	DTYEAR.	timestamp'2013-03-18 14:22:21'	2013
	DTYYQC.	timestamp'2013-03-18 14:22:21'	13:1
四半期出力形式	QTR.	date'2013-03-17'	1
	QTRR.	date'2013-03-17'	I
時間出力形式	HHMM.	time'14:22:21'	14:22
	HOUR.	time'14:22:21'	14
	TIME.	time'14:22:21'	14:22
	TIMEAMPM.	time'14:22:21'	14:22
	TOD.	time'14:22:21	14:22
年出力形式	YEAR.	date'2013-03-17'	2013
	YYMM.	date'2013-03-17'	2013M03
	YYMMC.	date'2013-03-17'	2013:03
	YYMMD.	date'2013-03-17'	2013-03
	YYMMN.	date'2013-03-17'	201303
	YYMMP.	date'2013-03-17'	2013.03
	YYMMS.	date'2013-03-17'	2013/03

種類	言語要素	入力	フォーマットされた結果
	YYMMDD.	date'2013-03-17'	13-03-17
	YYMMDDB.	date'2013-03-17'	13 03 17
	YYMMDDC.	date'2013-03-17'	13:03
	YYMMDDD.	date'2013-03-17'	13-03-17
	YYMMDDN.	date'2013-03-17'	20130317
	YYMMDDP.	date'2013-03-17'	13.03.17
	YYMMDDS.	date'2013-03-17'	2013/03/17
	YYMON.	date'2013-03-17'	2013MAR
年/四半期出力形式	YYQ.	date'2013-03-17'	2013Q1
	YYQC.	date'2013-03-17'	2013:1
	YYQD.	date'2013-03-17'	2013-1
	YYQN.	date'2013-03-17'	20131
	YYQP.	date'2013-03-17'	2013.1
	YYQS.	date'2013-03-17'	2013/1
	YYQR.	date'2013-03-17'	2013QI
	YYQRC.	date'2013-03-17'	2013:I
	YYQRD.	date'2013-03-17'	2013-I
	YYQRN.	date'2013-03-17'	2013I
	YYQRP.	date'2013-03-17'	2013.I
	YYQRS.	date'2013-03-17'	2013/I

DICTIONARY テーブル

FedSQL DICTIONARY テーブルは、Base SAS DICTIONARY テーブルと似ていますが、異なります。FedSQL DICTIONARY テーブルは、列、テーブル、およびカタログに関する情報と、テーブルおよびそれらに関連付けられたインデックスに関する統計量を含む読み取り専用のテーブルです。

次の表では、使用可能な DICTIONARY テーブルについて説明します。DICTIONARY テーブルのすべての 列の完全な説明については、付録 5, “DICTIONARY テーブルの説明” (1279 ページ)を参照してください。

表 2.18 DICTIONARY テーブル

DICTIONARY テーブル	説明
CATALOGS	現在の接続のカタログに関する情報が含まれています。
COLUMNS	既知のすべてのテーブルの 列に関する情報が含まれています。
COLUMN_STATISTICS	テーブルの列に基づくテーブルおよびインデックスの統計量が含まれています。
STATISTICS	テーブルに関連付けられているテーブルとインデックスの統計量が含まれています。
TABLES	テーブルタイプに関する情報と、データソース内のテーブル、カタログ、またはテーブルのスキーマ名のリストが含まれています。

DICTIONARY テーブルは、他のテーブルをクエリするのと同じ方法でクエリできます。CATALOG を除くすべての DICTIONARY テーブルでは、WHERE 句によるサブセット化と結果の順序付けを含めることができます。識別子は引用符で囲む必要があります。次のクエリはその例です。

> ヒント FedSQL は、CREATE TABLE ステートメントで名前を引用するか、データソースが小文字の保存しかサポートしない場合を除き、テーブルの作成時に、テーブルと列の名前を基になるデータベースに大文字で送信します。DICTIONARY クエリでは、大文字と小文字が区別されます。テーブル名と列名は、特に指定がない限り、大文字で指定します。

```
select * from dictionary.catalogs where catalog='SALECAT';

select * from dictionary.tables;
```

```
select * from dictionary.columns where table_name='YEAREND';

select * from dictionary.columns where table_name='view-name';

select distinct table_cat, table_schem from dictionary.tables;
```

注: DICTIONARY テーブルは、その情報についてデータソースサポートに依存しています。データソースが基になる関数呼び出しをサポートしていない場合、行は返されません。たとえば、一部のデータソースは、STATISTICS および COLUMN_STATISTICS 辞書テーブルの情報を返しません。

WHERE 句を指定しない DICTIONARY クエリは、特にデータソースに多数のテーブルがある場合、処理に時間がかかることがあります。

Google BigQuery ユーザー: カタログとスキーマで修飾されていない列とテーブルのクエリは、接続を介して使用可能なスキーマごとに個別のクエリを発行します。最良の結果を得るには、次のようにカタログとスキーマを使用して辞書クエリを修飾します。

```
select * from dictionary.columns where table_cat='catalog-name'
and table_name='table-name';
```

MongoDB ユーザー: デフォルトのスキーマ名または SCHEMA_COLLECTION=オプションで割り当てられた名前をテーブルスキーマとして指定します。

```
select * from dictionary.statistics
  WHERE table_schem='sas-default-schema' and table_name='mytable';
```

FedSQL 暗黙的パススルー機能

概要

暗黙的パススルー(IP)とは、SQL クエリコードを同等のデータソース固有の SQL コードに変換して、データソースに直接渡して処理できるようにするプロセスです。IP は、クエリの応答時間を向上させ、セキュリティを強化します。

IP によって提供されるパフォーマンス上の利点は、データ転送量の削減とデータソース固有の機能の活用という 2 つの主なカテゴリに分けることができます。転送されるデータの量は、データソースに対してクエリを実行することで削減されます。データソースから FedSQL に転送される行数を大幅に削減できるため、全体のクエリ処理時間を短縮することができます。超並列処理などのデータソース固有の機能の活用は、データソースに固有のものです。特別な機能のその他の例としては、高度な結合技術、データパーティション分割、テーブル統計、列統計があります。これらの機能により、データソースが FedSQL よりも迅速に SQL クエリを実行できるようになることがよくあります。

IP のセキュリティ上の利点は、処理可能な IP クエリのすべての部分がデータソース側で処理されることです。これにより、機密情報を含む可能性のある関連テーブルをクエリ処理のために FedSQL 側に送信する必要がなくなります。

FedSQL 暗黙的パススルーの使用方法

FedSQL IP は自動的に実行されます。IP を使用するためにオプションを指定する必要はありません。

単一ソースの FedSQL 暗黙的パススルー

クエリが単一のデータソースにアクセスしている場合、完全なクエリが暗黙的にデータソースに渡されるか、述語(WHERE 句など)が渡されて、ローカル処理のために FedSQL に転送する必要がある行がサブセット化されます。FedSQL は、SQL 構文が ANSI 準拠である場合にのみ、暗黙的にクエリを渡すことができます。次の制限により、IP が妨げられる可能性があります。

- PUT などの FedSQL 固有の関数。
- SKEWNESS、STUDENTS_T、NMISS、KURTOSIS、CSS、USS、PROBT などの特定の集計統計量。
- SIN、COS、ATAN、TAN などの数学関数。
- ANSI 準拠の FedSQL 構文は、データソースがその領域で ANSI に準拠していない場合、IP を妨げる可能性があります。

暗黙的パススルーを使用してマージおよび結合するには、データベーステーブルが同じカタログ内にある必要があります。異なるカタログに存在するデータベーステーブルのマージと結合は、クライアントで処理されます。複数のスキーマ(ただしカタログは 1 つだけ)への参照を含むクエリは、データソースに渡すことができます。

複数ソースの FedSQL 暗黙的パススルー

FedSQL は、複数のデータソースを含むクエリに対して IP を実行できます。これは、クエリを複数のクエリに分割し、これらの個々のクエリをそれぞれのドライバーに渡すことによって実現されます。“単一ソースの FedSQL 暗黙的パススルー” (69 ページ)に記載されている制限に加えて、複数ソース IP には次の追加の制限があります。

- 1 つのカンマ結合に最大 10 個のテーブルを含めることができます。
- 複数ソース IP が正しく実行されるようにするには、セット操作(UNION、INTERSECT、および EXCEPT など)の各側に複数のソースからのテーブルが必要です。
- クエリに相関サブクエリが含まれている場合、複数ソース IP は試行されません。

FedSQL 明示的パススルー機能

概要

FedSQ 明示的パススルー機能を使用すると、データソースに接続し、SQL ステートメントをそのデータソースに直接送信して実行することができます。この機能により、データソースの構文を使用することも可能になり、データソースでサポートされている非 ANSI 規格 SQL がサポートされます。

言語プロセッサへの接続を作成できます。この接続は、標準の SQL 構文をサポートしています。この接続は、SELECT ステートメントの FROM 句の CONNECTION TO コンポーネントと EXECUTE ステートメントを使用して、ネイティブ SQL 構文を受け入れることもできます。SAS Federation Server では、セキュリティ上の制限により、DSN が個人ログインを使用するように構成されている場合にのみ、これらのステートメントの使用が許可されることに注意してください。

FROM 句の CONNECTION TO コンポーネント

結果セットを生成するネイティブ SQL 要求をサブミットするには、SELECT ステートメントの FROM 句の CONNECTION TO コンポーネントを使用する必要があります。

CONNECTION TO コンポーネントの構文は次のとおりです。

FROM CONNECTION TO*catalog* (*native-syntax*) [[AS]*alias*]

引数

catalog	既存の FedSQL 接続のカタログの名前を指定します。
native-syntax	カタログのドライバー上で実行する選択型のクエリ(DDL ではない)を指定します。
alias	ネイティブクエリで生成される結果セットの名前を提供します。

例:

```
select oo.i, oo.rank, ff.onoff
   from connection to catalog_a
      ( select i, rank() over (order by j) rank from table_a ) oo,
   connection to catalog_b
      ( select distinct i, iif(k > 0.5, 1, 0) as onoff from table_a ) ff
   where oo.i = ff.i
order by 1;
```

詳細については、“SELECT ステートメント” (977 ページ)を参照してください。

EXECUTE ステートメント

DML ステートメントや DDL ステートメントなど、ネイティブ SQL が結果セットを生成しない場合は、EXECUTE ステートメントを使用する必要があります。さらに、EXECUTE ステートメントは、結果セットを生成するネイティブ SQL を受け入れます。詳細については、“EXECUTE ステートメント” (970 ページ)を参照してください。

FedSQL でのトランザクション

一部のアプリケーションでは、特定のデータソースに対する更新のグループを 1 つの単位として扱う必要があります。つまり、更新のグループ全体がデータに適用されるか、いずれも適用されないかのどちらかです。アプリケーションは、エラーをテストし、特定のコミットおよびロールバック操作を実行することで、その機能を提供できます。

FedSQL は COMMIT ステートメントと ROLLBACK ステートメントの両方をサポートします。これらのステートメントは、操作が中断されたときに、更新が完全に適用されるか、トランザクション前の状態にロールバックされることを保証することにより、データ保護を提供します。

*トランザクション*は、アトミックな作業単位です。つまり、トランザクションは完全に成功するか、まったく効果がないかのどちらかです。論理的で一貫性のある一連の変更が発生した後、トランザクションは、変更をコミットして永続的にするか、変更をキャンセルしてトランザクションによって変更された値を元の状態に戻すかのどちらかで終了します。

データベースへの接続が確立されると、自動コミット機能がデフォルトになります。つまり、個々の FedSQL ステートメントはトランザクションとして扱われます。ステートメントが実行されるとすぐに、リターンコードが検出されない場合、トランザクションは自動的にコミットされます。更新の問題が検出された場合、FedSQL はトランザクションのロールバックを開始します。

トランザクションに複数の FedSQL ステートメントを含めることができるようにするには、アプリケーションで自動コミット機能をオフにする必要があります。自動コミット機能がオンの場合、COMMIT および ROLLBACK ステートメントは効果がありません。AUTOCOMMIT オプションを使用して、PROC FEDSQL の自動コミット機能を制御できます。詳細については、FEDSQL プロシジャを参照してください。

すべてのデータソースがトランザクション管理を提供するわけではありません。たとえば、トランザクション管理は、SAS データセット、SPD Engine データセット、および SPD Server テーブルでは使用できません。ただし、Microsoft SQL Server、Oracle、Snowflake、Teradata などのデータソースではトランザクションサポートを使用できます。自動コミット機能をオフにする方法については、サーバー管理のドキュメントを参照してください。たとえば、*SAS Federation Server: 管理者ガイド* などを参照してください。

FedSQL 予約語

次の語は FedSQL 言語のキーワードとして予約されており、変数名やその他の方法で使用することはできません。

注: 区切り識別子は、予約語になる可能性のある用語に使用できます。たとえば、文字宣言以外で"char"という単語を使用するには、区切り識別子"char"として使用します。 詳細については、“区切り識別子” (18 ページ)を参照してください。

表 2.19 FedSQL 予約語 A - D

A
ABORT
ABSOLUTE
ACCESS
ACTION
ADD
AFTER
AGGREGATE
ALL
ALLOCATE
ALTER
ANALYSE
ANALYZE
AND
ANY
ARE
ARRAY
AS
ASC
ASENSITIVE
ASSERTION
ASSIGNMENT
ASYMMETRIC
AT
ATOMIC

BINARY
BIT
BLOB
BOOLEAN
BOTH
BY

C
CACHE
CALL
CALLED
CARDINALITY
CASCADE
CASCADED
CASE
CAST
CHAIN
CHAR
CHAR_LENGTH
CHARACTER
CHARACTER_LENGTH
CHARACTERISTICS
CHECK
CHECKPOINT
CLASS

COMMENT
COMMIT
COMMITTED
CONDITION
CONNECT
CONSTRAINT
CONSTRAINTS
CONVERSION
CONVERT
COPY
CORR
CORRESPONDING
COVAR_POP
COVAR_SAMP
CREATE
CREATEDB
CREATEUSER
CROSS
CUBE
CUME_DIST
CURRENT
CURRENT_DATE
CURRENT_DEFAULT_TRANSFORM_GROUP
CURRENT_PATH

D
DATABASE
DAY
DEALLOCATE
DEC
DECIMAL
DECLARE
DEFAULT
DEFAULTS
DEFERRABLE
DEFERRED
DEFINER
DELETE
DELIMITER
DELIMITERS
DENSE_RANK
DEREF
DESC
DESCRIBE
DETERMINISTIC
DISCONNECT
DISTINCT
DO
DOMAIN
DOUBLE

AUTORIZATION
B
BACKWARD
BEFORE
BEGIN
BETWEEN
BIGINT
CLOB
CLOSE
CLUSTER
COALESCE
COLLATE
COLLECT
COLUMN
CURRENT_ROLE
CURRENT_TIME
CURRENT_TIMESTAMP
CURRENT_TRANSFORM
_GROUP_FOR_TYPE
CURRENT_USER
CURSOR
CYCLE
DROP
DYNAMIC

表 2.20 FedSQL 予約語 E - O

E
EACH
ELEMENT
ELSE
ENCODING
ENCRYPTED
END
END-EXEC
ESCAPE
EVERY
EXCEPT
EXCLUDING
EXCLUSIVE
EXEC
EXECUTE
EXISTS
EXPLAIN
EXTERNAL
EXTRACT
F
FALSE
FETCH
FILTER
FIRST
FLOAT
FOR
FORCE
G
GET
GLOBAL
GRANT
GROUP
GROUPING
H
HANDLER
HAVING
HOLD
I
ILIKE
IMMEDIATE
IMMUTABLE
IMPLICIT
IN
INCLUDING
INCREMENT
INDEX
INDICATOR
INHERITS
INITIALLY
INNER
INOUT
INPUT
INSENSITIVE
INSERT
J
JOIN
K
KEY
L
LANCOMPILER
LANGUAGE
LARGE
LAST
LATERAL
LEADING
LEFT
LEVEL
LIKE
LIMIT
LISTEN
LOAD
LOCAL
LOCALTIME
LOCALTIMESTAMP
LOCATION
LOCK
M
MATCH
MAXVALUE
MEMBER
N
NAMES
NATIONAL
NATURAL
NCHAR
NCLOB
NEW
NEXT
NO
NOCREATEDB
NOCREATEUSER
NONE
NORMALIZE
NOT
NOTHING
NOTIFY
NOTNULL
NULL
NULLIF
NUMERIC
O
OF
OFF
OFFSET
OIDS
OLD
ON

FOREIGN
FORWARD
FREE
FREEZE
FROM
FULL
FUNCTION
FUSION
INSTEAD
INT
INTEGER
INTERSECT
INTERSECTION
INTERVAL
INTO
INVOKER
IS
ISNULL
ISOLATION
MERGE
METHOD
MINUTE
MINVALUE
MODE
MODIFIES
MODULE
MONTH
MOVE
MULTISET
ONLY
OPEN
OPERATOR
OPTION
OR
ORDER
OUT
OUTER
OVER
OVERLAPS
OVERLAY
OWNER

表 2.21 FedSQL 予約語 P - Z

P
PARAMETER
PARTIAL
PARTITION
PASSWORD
PATH
PENDANT
PERCENT_RANK
PERCENTILE_CONT
PERCENTILE_DESC
PLACING
POSITION
PRECISION
PREPARE
PRESERVE
PRIMARY
PRIOR
PRIVILEGES
PROCEDURAL
PROCEDURE

R
RANK
READ
REPLACE
RESET
RESTART
RESTRICT
RESULT
RETURN
RETURNS
REVOKE
RIGHT
ROLLBACK
ROLLUP
ROW
ROWS
ROW_NUMBER
RULE

S
SCHEMA
SCOPE
SCROLL
SEARCH
SECOND
SECURITY
SELECT
STATEMENT
STATIC
STATISTICS
STDDEV_POP
STDDEV_SAMP
STDIN
STDOUT
STORAGE
STRICT
SUBMULTISET
SUBSTRING
SYMMETRIC
SYSID
SYSTEM
SYSTEM_USER

T
TABLE
TABLESAMPLE
TEMP
TEMPLATE
TEMPORARY
THEN
TIME
UNION
UNIQUE
UNKNOWN
UNLISTEN
UNNEST
UNTIL
UPDATE
USAGE
USER
USING

V
VACUUM
VALID
VALIDATOR
VALUE
VALUES
VARCHAR
VARYING
VAR_POP
VAR_SAMP
VERBOSE
VERSION
VIEW

READS
REAL
RECHECK
RECURSIVE
REF
REFERENCES
REFERENCING
REGR_AVGX
REGR_AVGY
REGR_COUNT
REGR_INTERCEPT
REGR_R2
REGR_SLOPE
REGR_SXX
REGR_SXY
REGR_SYY
REINDEX
RELATIVE
RENAME

SENSITIVE
SPECIFIC
SPECIFICTYPE
SEQUENCE
SERIALIZABLE
SESSION
SESSION_USER
SET
SETOF
SHARE
SHOW
SIMILAR
SIMPLE
SMALLINT
SOME
SQLEXCEPTION
SQLSTATE
SQLWARNING
STABLE
START

TIMESTAMP
TIMEZONE_HOUR
TIMEZONE_MINUTE
TO
TOAST
TRAILING
TRANSACTION
TRANSLATE
TRANSLATION
TREAT
TRIGGER
TRIM
TRUE
TRUNCATE
TRUSTED
TYPE

U

UESCAPE
UNENCRYPTED

VOLATILE

W

WHEN
WHENEVER
WHERE
WIDTH_BUCKET
WINDOW
WITH
WITHIN
WITHOUT
WORK
WRITE

Y

YEAR

Z

ZONE

3

FedSQL 出力形式

出力形式の概要 ... **79**

一般的な出力形式構文 ... **80**

FedSQL での出力形式の使用 ... **81**
保存出力形式の保存、変更、削除、使用方法 ... 81
PUT 関数で出力をフォーマットする方法 ... 81

FedSQL 出力形式の検証 ... **82**

FedSQL 出力形式の例 ... **83**
新しいテーブルの列の出力形式を指定する ... 83
数値の出力形式 ... 83
日時値のフォーマット ... 83

ユーザー定義出力形式の使用 ... **84**

出力形式カテゴリ ... **85**

FedSQL でサポートされる NLS 出力形式 ... **86**

PUT 関数でサポートされている出力形式(カテゴリ別) ... **89**

ディクショナリ ... **96**
$BASE64Xw. 出力形式 ... 96
$BINARYw. 出力形式 ... 97
$CHARw. 出力形式 ... 98
$CSTRw. 出力形式 ... 99
$HEXw. 出力形式 ... 100
$OCTALw. 出力形式 ... 101
$QUOTEw. 出力形式 ... 102
$REVERJw. 出力形式 ... 104
$REVERSw. 出力形式 ... 105
$UPCASEw. 出力形式 ... 106
$UUIDw. 出力形式 ... 107
$w. 出力形式 ... 108
B8601DAw. 出力形式 ... 109
B8601DNw. 出力形式 ... 110
B8601DTw.d 出力形式 ... 111
B8601DZw. 出力形式 ... 113
B8601LZw. 出力形式 ... 115
B8601TMw.d 出力形式 ... 117

B8601TZw. 出力形式 . . . 118
BESTDOTXw. 出力形式 . . . 120
BESTw. 出力形式 . . . 122
BESTDw.p 出力形式 . . . 124
BINARYw. 出力形式 . . . 126
COMMAw.d 出力形式 . . . 127
COMMAXw.d 出力形式 . . . 128
Dw.p 出力形式 . . . 130
DATEAMPMw.d 出力形式 . . . 131
DATETIMEw.d 出力形式 . . . 133
DATEw. 出力形式 . . . 135
DAYw. 出力形式 . . . 137
DDMMYYw. 出力形式 . . . 138
DDMMYYxw. 出力形式 . . . 139
DOLLARw.d 出力形式 . . . 141
DOLLARXw.d 出力形式 . . . 143
DOWNAMEw. 出力形式 . . . 144
DTDATEw. 出力形式 . . . 145
DTMONYYw. 出力形式 . . . 147
DTWKDATXw. 出力形式 . . . 148
DTYEARw. 出力形式 . . . 150
DTYYQCw. 出力形式 . . . 151
Ew. 出力形式 . . . 152
E8601DAw. 出力形式 . . . 153
E8601DNw. 出力形式 . . . 154
E8601DTw.d 出力形式 . . . 156
E8601DZw. 出力形式 . . . 157
E8601LZw. 出力形式 . . . 159
E8601TMw.d 出力形式 . . . 161
E8601TZw.d 出力形式 . . . 163
EUROw.d 出力形式 . . . 165
EUROXw.d 出力形式 . . . 166
FLOATw.d 出力形式 . . . 168
FRACTw. 出力形式 . . . 169
HEXw. 出力形式 . . . 170
HHMMw.d 出力形式 . . . 171
HOURw.d 出力形式 . . . 173
IEEEw.d 出力形式 . . . 175
JULIANw. 出力形式 . . . 176
MDYAMPMw.d 出力形式 . . . 177
MMDDYYw. 出力形式 . . . 179
MMDDYYxw. 出力形式 . . . 181
MMSSw.d 出力形式 . . . 183
MMYYw. 出力形式 . . . 185
MMYYxw. 出力形式 . . . 186
MONNAMEw. 出力形式 . . . 188
MONTHw. 出力形式 . . . 189
MONYYw. 出力形式 . . . 190
NEGPARENw.d 出力形式 . . . 192
NENGOw. 出力形式 . . . 193
OCTALw. 出力形式 . . . 194
PERCENTw.d 出力形式 . . . 196
PERCENTNw.d 出力形式 . . . 197
QTRw. 出力形式 . . . 199

QTRRw. 出力形式 . . . 200
ROMANw. 出力形式 . . . 201
SIZEKw.d 出力形式 . . . 201
SIZEKMGw.d 出力形式 . . . 203
TIMEw.d 出力形式 . . . 204
TIMEAMPMw.d 出力形式 . . . 206
TODw.d 出力形式 . . . 208
VAXRBw.d 出力形式 . . . 210
w.d 出力形式 . . . 211
WEEKDATEw. 出力形式 . . . 212
WEEKDATXw. 出力形式 . . . 214
WEEKDAYw. 出力形式 . . . 216
WEEKUw. 出力形式 . . . 217
WEEKVw. 出力形式 . . . 218
WEEKWw. 出力形式 . . . 220
YEARw. 出力形式 . . . 222
YENw.d 出力形式 . . . 223
YYMMw. 出力形式 . . . 224
YYMMxw. 出力形式 . . . 225
YYMMDDw. 出力形式 . . . 227
YYMMDDxw. 出力形式 . . . 229
YYMONw. 出力形式 . . . 232
YYQw. 出力形式 . . . 233
YYQxw. 出力形式 . . . 234
YYQRw. 出力形式 . . . 236
YYQRxw. 出力形式 . . . 238
YYQZw. 出力形式 . . . 240
YYWEEKUw. 出力形式 . . . 242
YYWEEKVw. 出力形式 . . . 243
YYWEEKWw. 出力形式 . . . 245
Zw.d 出力形式 . . . 246

出力形式の概要

出力形式とは、FedSQL がデータ値を書き込むために使用する命令のことです。出力形式を使用して、書き込まれるデータ値の表示を制御したり、場合によっては、分析のためにデータ値をグループ化したりします。たとえば、数値をローマ数字に変換する ROMANw.出力形式は、数値 2013 を **MMXIII** として書き込みます。

FedSQL を使用して SAS データセット、SPD Engine データセット、または SPD Server テーブルを作成する場合、フォーマット命令をデータセットとともに保存し、データセットが Base SAS セッションで表示される際に SAS Output Delivery System (ODS)によって自動的に適用することができます。他の SAS/ACCESS エンジンの PUT 関数では、出力形式を明示的に指定する必要があります。これは、データソースドライバーが出力形式と入力形式に関する情報を保存しないためです。

一般的な出力形式構文

FedSQL 出力形式の構文は次のとおりです。

[$] *format* [*w*] . [*d*]

引数

$
文字出力形式であることを表します。数値入力形式には使用されません。

format
出力形式名です。出力形式には、SAS 出力形式、FedSQL 出力形式、または PROC FORMAT の INVALUE ステートメントを使用して事前に定義されたユーザー定義の出力形式があります。ユーザー定義の出力形式の詳細については、*Base SAS プロシジャガイド*の PROC FORMAT を参照してください。

w
出力形式幅を指定します。ほとんどの出力形式では、入力データのカラム数です。

d
小数点以下の桁数です(数値出力形式)。FedSQL は、入力データを 10 の *d* 乗で除算します。

ヒント *d* の値が 15 より大きい場合、小数点以下 15 桁以降の桁の小数値が正確でない場合があります。

出力形式は、名前の一部として必ずピリオド(.)を含みます。出力形式の *w* 値と *d* 値を省略すると、デフォルト値が使用されます。出力形式で指定する *d* 値は、データの小数点以下の桁数に関係なく、その小数点以下の桁数を表示するように FedSQL に指示します。出力形式は、内部的に保存されているデータ値は変更または切り捨てるものではありません。

たとえば、DOLLAR10.2 では、*w* 値に 10 が指定されているため、値の最大カラム数は 10 となります。*d* 値には 2 が指定されているため、10 カラムのうち 2 カラムが値の小数部分に使用されます。値の残りのすべての文字には 8 カラムが使用されます。これには、小数点、小数以外の数値、マイナス記号(値が負の場合)、ドル記号、およびカンマが必要に応じて含まれます。

値を表すには出力形式の幅が狭すぎる場合、FedSQL では、値は利用可能なスペースに合うように圧縮されます。文字出力形式では、値の右側が切り捨てられます。数値出力形式では、BESTw.出力形式が使用される場合があります。BESTw.出力形式は、数値を書き込むためのデフォルトの出力形式です。BESTw.は値を丸め、SAS が指定された幅内で小数点以下の部分に少なくとも 1 つの有効数字を表示できる場合、BESTw.は結果を 10 進数で生成します。それ以外の場合は、指数表記で結果が生成されます。SAS では、表現に使用する出力形式にかかわらず、常に完全な値を保存します。BESTw.出力形式を適用するには、少なくとも 3 カラムが使用可能である必要があります。十分な幅を指定していない場合、FedSQL では、空白が出力されます。例としては、次の要求:

```
select put(12345, 3.);
```

は出力 **1E4** を返します。一方、この要求:

```
select put(12345, 2.);
```

は空白値を返します。

数値出力形式を使用して文字値を書き出すなど、矛盾する出力形式を使用した場合、FedSQL では、まず他の種類の類似した出力形式が試されます。これが実行不能な場合は、問題を説明するエラーメッセージが SAS ログに表示されます。

FedSQL での出力形式の使用

保存出力形式の保存、変更、削除、使用方法

出力形式メタデータのストレージは、SAS データセット、SPD Engine データセット、および SPD Server テーブルでのみサポートされます。その結果、Base SAS で使用すると、FedSQL で作成された SAS データセットは、SAS で作成されたデータセットと同じように動作します。

CREATE TABLE ステートメントで HAVING 句の属性として出力形式を指定します。詳細については、“CREATE TABLE ステートメント” (939 ページ)を参照してください。たとえば、次のステートメントでは、列 *profit* が EURO13.2 出力形式で宣言されています。

```
create table monthly (profit double having format euro13.2);
```

保存出力形式を変更または削除するには、Base SAS を使用する必要があります。FedSQL でテーブルを読み取るときに、保存出力形式値を持つ列を別の出力形式で表示する場合は、PUT 関数を使用します。

FedSQL は、次の保存出力形式をサポートしています。

- ユーザー定義の出力形式と SAS が提供する出力形式の両方を保存できます。
- SAS が提供するすべての出力形式を保存できます。出力形式のリストについては、*SAS 出力形式と入力形式: リファレンス*を参照してください。FedSQL は出力形式を検証しません。保存出力形式が無効な場合、エラーが発生しますが、クライアントアプリケーションで無効な出力形式が使用されている場合のみです。
- 出力形式は、データ型が CHAR および DOUBLE の列に対してのみ保存できます。
- 保存出力形式にアクセスするには、Base SAS セッションが使用可能である必要があります。Base SAS セッションには、SAS 出力形式の定義が含まれています。
- ユーザー定義の出力形式は、SAS セッションで使用できる必要があります。

PUT 関数で出力をフォーマットする方法

PUT 関数を使用すると、SAS データだけでなく、サードパーティデータソースのデータにも出力形式を関連付けることができます。出力形式は、PUT 関数の引数として指定され、フォーマットされたデータを生成します。次の例では、PUT 関数は DOLLAR17.2 出力形式を使用してフォーマット値 **4503945867** を返します。この例では、値**$4,503,945,867.00** が返されます。

```
select put(4503945867, dollar17.2);
```

FedSQL では、PUT 関数で指定された出力形式を次のようにサポートしています。

- PUT 関数を出力形式なしで使用すると、エラーが発生します。
- PUT 関数は、FedSQL 言語が Base SAS セッションの外で実行されるときに Base SAS で使用可能な出力形式のサブセットをサポートしています。リストについては、“PUT 関数でサポートされている出力形式(カテゴリ別)” (89 ページ)を参照してください。
- 出力形式は、FedSQL でサポートされている任意のデータ型に関連付けることができます。ただし、データ型は変換されます。数値出力形式で PUT 関数に渡される値はすべて、NVARCHAR、VARBINARY、または BINARY に変換されます。型変換は、出力形式名に基づいて実行されます。文字出力形式で PUT 関数に渡される値はすべて、NVARCHAR に変換されます。
- PUT で指定される出力形式は一時的です。PUT 関数は、保存データには影響しません。
- PUT 関数では、Base SAS セッションが使用可能である必要はありません。ただし、セッションが使用できない場合、機能は制限されます。

詳細については、“PUT 関数” (788 ページ)も参照してください。

FedSQL 出力形式の検証

FedSQL を使用して出力形式がデータセットに保存される場合、検証は行われません。列のメタデータが要求されると、検証なしで出力形式名が返されます。出力形式は実行時に検証されます。

PUT 関数は、使用時に指定された出力形式を検証します。

FedSQL 出力形式の例

新しいテーブルの列の出力形式を指定する

この例では、PROD、UNIT_PRICE、および TOTAL_SALES 列を含むテーブル Work.UnitSales を作成します。文字出力形式$10 が列 PROD に適用され、数値出力形式 DOLLAR12.2 が列 UNIT_PRICE および TOTAL_SALES に適用されます。

```
proc fedsql;
  create table work.UnitSales(
    prod char(10) having format $10.,
    unit_price double having format dollar12.2,
    total_sales double having format dollar12.2
  );
  insert into work.UnitSales VALUES('Desk', 357.98, 452844.7);
  insert into work.UnitSales VALUES('Chair', 149.78, 697225.9);
  insert into work.UnitSales VALUES('Table', 158.99, 168211.42);
  select * from work.UnitSales;
quit;
```

アウトプット 3.1　Work.UnitSales テーブル

PROD	UNIT_PRICE	TOTAL_SALES
Desk	$357.98	$452,844.70
Chair	$149.78	$697,225.90
Table	$158.99	$168,211.42

数値の出力形式

この例では、PUT 関数 (788 ページ)を使用して、出力形式 COMMA10 を列 UnitSales に適用します。この列は、Total_Sales をテーブル UnitSales の Unit_Price で除算することによって計算されます。

```
proc fedsql;
  select Prod as "Product",
    put(total_sales/unit_price, comma10.) as "Units Sold"
    from Work.UnitSales;
quit;
```

アウトプット 3.2　結果

Product	Units Sold
Desk	1,265
Chair	4,655
Table	1,058

日時値のフォーマット

この例では、DATETIME 関数 (442 ページ)によって返される値をフォーマットして、列 Current Date/Time を作成します。ここでは、PUT 関数 (788 ページ)を使用して、出力形式 DATETIME20 を戻り値に適用します。

```
proc fedsql;
  select put(datetime(), datetime20.) as "Current Date/Time";
quit;
```

アウトプット 3.3　結果

Current Date/Time
03JUL2025:08:08:46

ユーザー定義出力形式の使用

SAS FORMAT プロシジャを使用して、生データ値をフォーマット文字値に置き換えるカスタム出力形式を定義できます。たとえば、次の PROC FORMAT コードでは、部門コードを対応する部門名にマッピングする DEPTNO.というカスタム数値出力形式が作成されます。

```
proc format;
  value deptno
  10 = 'Sales'
  20 = 'Research'
  30 = 'Accounting'
  40 = 'Operations';
run;
```

結果のユーザー定義の出力形式は、SAS データセット、SPD Engine データセット、または SPD Server テーブルに保存するか、PUT 関数を使用してサードパーティデータソースに適用できます。次のコードでは、PUT 関数と DEPTNO を使用します。出力形式を上記で作成し、EMPLOYEES テーブルの DEPT 列に数値部門コードを対応する文字ベースの部門名として表示します。

```
proc fedsql;
```

```
    select emp_name, hire_date, put(dept, deptno.) as dept
    from employees limit 4;
quit;
```

アウトプット 3.39 (85 ページ)は、ソース EMPLOYEES テーブルのいくつかの行を表示します。PUT 関数の出力はアウトプット 3.40 (85 ページ)に表示されます。

アウトプット 3.4　ソース Employees テーブルのコンテンツ

EMP_NAME	HIRE_DATE	DEPT
Jim Barnes	26NOV2000	10
Clifford James	26NOV2000	20
Barbara Sandman	26NOV2000	30
William Baylor	26NOV2000	40

アウトプット 3.5　PUT 関数が適用された後の Employees テーブルのコンテンツ

EMP_NAME	HIRE_DATE	DEPT
Jim Barnes	26NOV2000	Sales
Clifford James	26NOV2000	Research
Barbara Sandman	26NOV2000	Accounting
William Baylor	26NOV2000	Operations

SAS で独自の出力形式を作成する方法の詳細については、*Base SAS プロシジャガイド*の PROC FORMAT を参照してください。

出力形式カテゴリ

出力形式は、操作する値の種類によって分類できます。各 FedSQL 出力形式は、次のいずれかのカテゴリに属します。

文字
　文字変数から文字データ値を書き出します。

日付と時間
　文字変数から文字データ値を書き出します。

数値
　数値変数から数値データ値を書き出します。

FedSQL でサポートされる NLS 出力形式

各国語サポート(NLS)は、ソフトウェア製品が対象となるすべてのグローバル市場で適切に機能することを可能にする一連の機能です。SAS System には、NLS 機能が含まれており、SAS アプリケーションをローカル言語の規則に準拠するように記述できます。通常、英語で記述されたソフトウェアは、英語と、米国で遵守されている規則を使用してフォーマットされたデータの両方を使用するユーザーに適しています。ただし、NLS がないと、これらの製品は世界の他の地域のユーザーにとってはうまく機能しない可能性があります。SAS の NLS を使用すると、アジアやヨーロッパなどの地域で、その母国語と環境でデータを正常に処理できます。FedSQL 言語は、次の NLS 出力形式をサポートしています。詳細については、*SAS 各国語サポート(NLS): リファレンスガイド*を参照してください。

カテゴリ	言語要素	説明
日付と時間	NLDATE*w*.	SAS 日付値を指定ロケールの日付値に変換し、その日付値を日付として書き込みます。
	NLDATEMD*w*.	SAS 日付値を指定ロケールの日付値に変換し、その値を月名と日付として書き込みます。
	NLDATEMN*w*.	SAS 日付値を指定ロケールの日付値に変換し、その値を月名として書き込みます。
	NLDATEW*w*.	SAS 日付値を指定ロケールの日付値に変換し、その値を日付と曜日として書き込みます。
	NLDATEWN*w*.	SAS 日付値を指定ロケールの日付値に変換し、その日付値を曜日として書き込みます。
	NLDATEYM*w*.	SAS 日付値を指定ロケールの日付値に変換し、その日付値を年と月名として書き込みます。
	NLDATEYQ*w*.	SAS 日付値を指定ロケールの日付値に変換し、その日付値を年と四半期として書き込みます。
	NLDATEYR*w*.	SAS 日付値を指定ロケールの日付値に変換し、その日付値を年として書き込みます。

カテゴリ	言語要素	説明
	NLDATEYW*w*.	SAS 日付値を指定ロケールの日付値に変換し、その日付値を年と週として書き込みます。
	NLDATMAP*w*.	SAS 日時値を指定ロケールの日時値に変換し、その値を午前または午後の日時として書き込みます。
	NLDATMDT*w*.	SAS 日時値を指定ロケールの日時値に変換し、その値を月名、日付、および年として書き込みます。
	NLDATMMD*w*.	SAS 日時値を指定ロケールの日時値に変換し、その値を月名と日付として書き込みます。
	NLDATMMN*w*.	SAS 日時値を指定ロケールの日時値に変換し、その値を月名として書き込みます。
	NLDATMTM*w*.	SAS 日時値の時間部分を指定ロケールの時刻値に変換し、その値を時刻として書き込みます。
	NLDATM*w*.	SAS 日時値を指定ロケールの日時値に変換し、その値を日時として書き込みます。
	NLDATMW*w*.	SAS 日時値を曜日と日時として、ロケールに依存する日時文字列に変換します。
	NLDATMWN*w*.	SAS 日時値を指定ロケールの日時値に変換し、その値を曜日として書き込みます。
	NLDATMYM*w*.	SAS 日時値を指定ロケールの日時値に変換し、その値を年と月名として書き込みます。
	NLDATMYQ*w*.	SAS 日時値を指定ロケールの日時値に変換し、その値を年と四半期として書き込みます。
	NLDATMYR*w*.	SAS 日時値を指定ロケールの日時値に変換し、その値を年として書き込みます。
	NLDATMYW*w*.	SAS 日時値を指定ロケールの日時値に変換し、その値を年と週名として書き込みます。
	NLTIMAP*w*.	SAS 時間値を指定ロケールの時間値に変換し、その値を午前または午後の時間値として

カテゴリ	言語要素	説明
		書き込みます。NLTIMAP は、SAS 日時値も変換します。
	NLTIME*w.*	SAS 時間値を指定ロケールの時間値に変換し、その値を時間値として書き込みます。NLTIME は、SAS 日時値も変換します。
数値	NLBEST*w.*	ロケールに基づいて最適な数値表記を書き込みます。
	NLMNY*w.d*	現地通貨を使用して、指定ロケールの現地表現の通貨出力形式を書き込みます。
	NLMNYI*w.d*	指定ロケールの国際表現の通貨出力形式を書き込みます。
	NLNUM*w.d*	指定ロケールの現地表現の数値出力形式を書き込みます。
	NLNUMI*w.d*	指定ロケールの国際表現の数値出力形式を書き込みます。
	NLPCT*w.d*	指定ロケールの現地表現のパーセンテージデータを書き込みます。
	NLPCTI*w.d*	指定ロケールの国際表現のパーセンテージデータを書き込みます。
	NLPCTN*w.d*	負の値にマイナス記号を使用して、パーセンテージを生成します。
	NLPCTP*w.d*	ロケール固有の数値をパーセンテージで書き込みます。ロケール固有の数値をパーセンテージで書き込みます。
	NLPVALUE*w.d*	指定ロケールの現地表現の p 値を書き込みます。
	NLSTRMON*w.d*	指定ロケールの月名を書き込みます。
	NLSTRQTR*w.d*	指定ロケールの四半期として数値を書き込みます。
	NLSTRWK*w.d*	指定ロケールの曜日として数値を書き込みます。

PUT 関数でサポートされている出力形式(カテゴリ別)

カテゴリ	言語要素	説明
ISO 8601	B8601DAw. 出力形式 (p. 109)	ISO 8601 規格の基本表記 yyyymmdd を使用して日付値を書き出します。
	B8601DNw. 出力形式 (p. 110)	ISO 8601 規格の基本表記 yyyymmdd を使用して、日時値から日付を書き出します。
	B8601DTw.d 出力形式 (p. 111)	ISO 8601 規格の基本表記 yyyymmddThhmmss<ffffff>を使用して日時値を書き出します。
	B8601DZw. 出力形式 (p. 113)	ISO 8601 規格の日時とタイムゾーンの基本表記 yyyymmddThhmmss+0000 を使用して、基準子午線の協定世界時(UTC)の日時値を書き出します。
	B8601LZw. 出力形式 (p. 115)	ISO 8601 規格の基本表記 hhmmss+|–hhmm を使用して、ローカル時間と UTC 間のタイムゾーンオフセット差を追加することにより、時間値をローカル時間として書き出します。
	B8601TMw.d 出力形式 (p. 117)	ISO 8601 規格の基本表記 hhmmss<ffff>を使用して、時間値を書き出します。
	B8601TZw. 出力形式 (p. 118)	ISO 8601 規格の基本時間表記 hhmmss+|–hhmm を使用して、時間値を協定世界時(UTC)に調整し、時間値を書き出します。
	E8601DAw. 出力形式 (p. 153)	ISO 8601 規格の拡張表記 yyyy-mm-dd を使用して日付値を書き出します。
	E8601DNw. 出力形式 (p. 154)	ISO 8601 規格の拡張表記 yyyy-mm-dd を使用して、SAS 日時値から日付を書き出します。
	E8601DTw.d 出力形式 (p. 156)	ISO 8601 規格の拡張表記 yyyy-mm-ddThh:mm:ss.ffffff を使用して日時値を書き出します。
	E8601DZw. 出力形式 (p. 157)	ISO 8601 規格の日時とタイムゾーンの拡張表記 yyyy-mm-ddThh:mm:ss+00:00 を使用して、基準子午線の協定世界時(UTC)の日時値を書き出します。
	E8601LZw. 出力形式 (p. 159)	ISO 8601 規格の拡張時間表記 hh:mm:ss+|–hh:mm を使用して、ローカル SAS セッションに対応した協定世界時(UTC)のオフセットを追加することにより、時間値をローカル時間として書き出します。

カテゴリ	言語要素	説明
	E8601TMw.d 出力形式 (p. 161)	ISO 8601 規格の拡張表記 hh:mm:ss.ffffff を使用して時間値を書き出します。
	E8601TZw.d 出力形式 (p. 163)	時間値を協定世界時(UTC)に調整し、時間値を ISO 8601 規格の拡張表記 hh:mm:ss.<fff>+|–hh:mm を使用して書き出します。
数値	BESTDOTXw. 出力形式 (p. 120)	SAS が最適な表記を選び、小数点としてドット(.)を使うように指定します。
	BESTw. 出力形式 (p. 122)	SAS により最適な出力が選択されます。
	BESTDw.p 出力形式 (p. 124)	同じような大きさの数値に対しては小数点以下の桁数を揃えて出力します。整数は小数なしで出力します。
	BINARYw. 出力形式 (p. 126)	数値を 2 進表現に変換します。
	COMMAw.d 出力形式 (p. 127)	3 桁ごとにカンマを入れ、小数部分との区切りにはピリオドを使用して数値を出力します。
	COMMAXw.d 出力形式 (p. 128)	3 桁ごとにピリオドを入れ、小数部分との区切りにはカンマを使用して数値を出力します。
	Dw.p 出力形式 (p. 130)	値の桁数をできる限り最大にし、同じような大きさの数値に対しては小数点以下の桁数を揃えて出力します。
	DOLLARw.d 出力形式 (p. 141)	先頭にドル($)記号を付け、3 桁ごとにカンマを入れ、小数部分との区切りにはピリオドを使用して数値を出力します。
	DOLLARXw.d 出力形式 (p. 143)	先頭にドル($)記号を付け、3 桁ごとにピリオドを入れ、小数部分との区切りにはカンマを使用して数値を出力します。
	Ew. 出力形式 (p. 152)	数値を指数表記で書き出します。
	EUROw.d 出力形式 (p. 165)	先頭にユーロ記号(E)を付け、3 桁ごとにカンマを入れ、小数部分との区切りにはピリオドを使用して数値を出力します。
	EUROXw.d 出力形式 (p. 166)	先頭にユーロ記号(E)を付け、3 桁ごとにピリオドを入れ、小数部分との区切りにはカンマを使用して数値を出力します。
	FLOATw.d 出力形式 (p. 168)	数値に 10 の d 乗を掛けた、単精度のネイティブ浮動小数点値を生成します。
	FRACTw. 出力形式 (p. 169)	数値を分数に変換します。
	HEXw. 出力形式 (p. 170)	バイナリ実数(浮動小数点)値を 16 進表現に変換します。
	IEEEw.d 出力形式 (p. 175)	数値に 10 の d 乗を掛けて、IEEE 浮動小数点値を生成します。
	NEGPARENw.d 出力形式 (p. 192)	負の数値をかっこで囲んで書き出します。
	OCTALw. 出力形式 (p. 194)	数値を 8 進表現に変換します。

カテゴリ	言語要素	説明
	PERCENTw.d 出力形式 (p. 196)	数値を百分率として書き出します。
	PERCENTNw.d 出力形式 (p. 197)	百分率を書き出します。負の値にはマイナス符号を使用します。
	ROMANw. 出力形式 (p. 201)	数値をローマ数字で書き出します。
	SIZEKw.d 出力形式 (p. 201)	キロバイトを表す nK の形式で数値を書き込みます。
	SIZEKMGw.d 出力形式 (p. 203)	キロバイトの場合は nKB、メガバイトの場合は nMB、ギガバイトの場合は nGB の形式で数値を書き込みます。
	VAXRBw.d 出力形式 (p. 210)	VMS 形式のバイナリ実数(浮動小数点)を書き出します。
	w.d 出力形式 (p. 211)	標準数値でデータを書き出します(1 バイト 1 桁)。
	YENw.d 出力形式 (p. 223)	円記号、カンマ、小数点を使用して数値を書き込みます。
	Zw.d 出力形式 (p. 246)	先頭に 0 を含む標準数値データを書き出します。
日付と時間	B8601DAw. 出力形式 (p. 109)	ISO 8601 規格の基本表記 yyyymmdd を使用して日付値を書き出します。
	B8601DNw. 出力形式 (p. 110)	ISO 8601 規格の基本表記 yyyymmdd を使用して、日時値から日付を書き出します。
	B8601DTw.d 出力形式 (p. 111)	ISO 8601 規格の基本表記 yyyymmddThhmmss<ffffff>を使用して日時値を書き出します。
	B8601DZw. 出力形式 (p. 113)	ISO 8601 規格の日時とタイムゾーンの基本表記 yyyymmddThhmmss+0000 を使用して、基準子午線の協定世界時(UTC)の日時値を書き出します。
	B8601LZw. 出力形式 (p. 115)	ISO 8601 規格の基本表記 hhmmss+|–hhmm を使用して、ローカル時間と UTC 間のタイムゾーンオフセット差を追加することにより、時間値をローカル時間として書き出します。
	B8601TMw.d 出力形式 (p. 117)	ISO 8601 規格の基本表記 hhmmss<ffff>を使用して、時間値を書き出します。
	B8601TZw. 出力形式 (p. 118)	ISO 8601 規格の基本時間表記 hhmmss+|–hhmm を使用して、時間値を協定世界時(UTC)に調整し、時間値を書き出します。
	DATEAMPMw.d 出力形式 (p. 131)	SAS 日時値を、AM または PM を付けて ddmmmyy:hh:mm:ss.ss 形式で書き出します。
	DATETIMEw.d 出力形式 (p. 133)	SAS 日時値を ddmmmyy:hh:mm:ss.ss 形式で書き出します。
	DATEw. 出力形式 (p. 135)	SAS 日付値を ddmmmyy、ddmmmyyyy または dd-mmm-yyyy 形式で書き出します。

カテゴリ	言語要素	説明
	DAYw. 出力形式 (p. 137)	SAS 日付値の日付部分を書き出します。
	DDMMYYw. 出力形式 (p. 138)	SAS 日付値を ddmm<yy>yy または dd/mm/<yy>yy 形式で書き出します。スラッシュが区切り文字として使用され、年は 2 桁または 4 桁の数字で書き出されます。
	DDMMYYxw. 出力形式 (p. 139)	SAS 日付値を ddmm<yy>yy または dd-mm-yy<yy>形式で書き出します。出力形式名の x は、日、月、年を区切る特殊文字を表す文字で、ハイフン(-)、ピリオド(.)、空白文字、スラッシュ(/)、コロン(:)、または区切り文字なしになります。年は 2 桁または 4 桁のいずれかになります。
	DOWNAMEw. 出力形式 (p. 144)	SAS 日付値から曜日名を書き出します。
	DTDATEw. 出力形式 (p. 145)	入力として SAS 日時値が必要です。SAS 日付値を ddmmmyy または ddmmmyyyy 形式で書き出します。
	DTMONYYw. 出力形式 (p. 147)	SAS 日時値の日付部分を月と年として mmmyy または mmmyyyy 形式で書き出します。
	DTWKDATXw. 出力形式 (p. 148)	SAS 日時値の日付部分を曜日と日付として day-of-week、dd month-name yy (または yyyy)形式で書き出します。
	DTYEARw. 出力形式 (p. 150)	SAS 日時値の日付部分を年として yy または yyyy 形式で書き出します。
	DTYYQCw. 出力形式 (p. 151)	SAS 日時値から、年と四半期をコロン(:)で区切って書き出します。
	E8601DAw. 出力形式 (p. 153)	ISO 8601 規格の拡張表記 yyyy-mm-dd を使用して日付値を書き出します。
	E8601DNw. 出力形式 (p. 154)	ISO 8601 規格の拡張表記 yyyy-mm-dd を使用して、SAS 日時値から日付を書き出します。
	E8601DTw.d 出力形式 (p. 156)	ISO 8601 規格の拡張表記 yyyy-mm-ddThh:mm:ss.ffffff を使用して日時値を書き出します。
	E8601DZw. 出力形式 (p. 157)	ISO 8601 規格の日時とタイムゾーンの拡張表記 yyyy-mm-ddThh:mm:ss+00:00 を使用して、基準子午線の協定世界時(UTC)の日時値を書き出します。
	E8601LZw. 出力形式 (p. 159)	ISO 8601 規格の拡張時間表記 hh:mm:ss+|–hh:mm を使用して、ローカル SAS セッションに対応した協定世界時(UTC)のオフセットを追加することにより、時間値をローカル時間として書き出します。
	E8601TMw.d 出力形式 (p. 161)	ISO 8601 規格の拡張表記 hh:mm:ss.ffffff を使用して時間値を書き出します。
	E8601TZw.d 出力形式 (p. 163)	時間値を協定世界時(UTC)に調整し、時間値を ISO 8601 規格の拡張表記 hh:mm:ss.<fff>+|–hh:mm を使用して書き出します。

カテゴリ	言語要素	説明
	HHMMw.d 出力形式 (p. 171)	SAS 時間値を時間および分として hh:mm 形式で書き出します。
	HOURw.d 出力形式 (p. 173)	SAS 時間値を時間および時間の小数として書き出します。
	JULIANw. 出力形式 (p. 176)	SAS 日付値を yyddd または yyyyddd 形式のユリウス日付として書き出します。
	MDYAMPMw.d 出力形式 (p. 177)	SAS 日時値を mm/dd/yy<yy> hh:mm AM|PM 形式で書き出します。年は 2 桁または 4 桁のいずれかです。
	MMDDYYw. 出力形式 (p. 179)	SAS 日付値を mmdd<yy>yy または mm/dd/<yy>yy 形式で書き出します。スラッシュが区切り文字として使用され、年は 2 桁または 4 桁の数字で書き出されます。
	MMDDYYxw. 出力形式 (p. 181)	SAS 日付値を mmdd<yy>yy または mm-dd-<yy>yy 形式で書き出します。ここで、出力形式名の x は、月、日、年を区切る特殊文字を表す文字です。特殊文字は、ハイフン(-)、ピリオド(.)、空白文字、スラッシュ(/)、コロン(:)または区切り文字なしになります。年は 2 桁または 4 桁のいずれかになります。
	MMSSw.d 出力形式 (p. 183)	SAS 時間値を午前 0 時から経過した分数および秒数として書き出します。
	MMYYw. 出力形式 (p. 185)	SAS 日付値を mmM<yy>yy 形式で書き出します。M は区切り文字を表し、年は 2 桁または 4 桁の数字で書き出されます。
	MMYYxw. 出力形式 (p. 186)	SAS 日付値を mm<yy>yy または mm-<yy>yy 形式で書き出します。出力形式名の x は、月と年を区切る特殊文字を表す文字で、ハイフン(-)、ピリオド(.)、空白文字、スラッシュ(/)、コロン(:)、または区切り文字なしになります。年は 2 桁または 4 桁のいずれかになります。
	MONNAMEw. 出力形式 (p. 188)	SAS 日付値を月の名前として書き出します。
	MONTHw. 出力形式 (p. 189)	SAS 日付値を月として書き出します。
	MONYYw. 出力形式 (p. 190)	SAS 日付値を月および年として mmmyy または mmmyyyy 形式で書き出します。
	NENGOw. 出力形式 (p. 193)	SAS 日付値を e.yymmdd 形式で日本の日付として書き出します。
	QTRw. 出力形式 (p. 199)	SAS 日付値を年の四半期として書き出します。
	QTRRw. 出力形式 (p. 200)	SAS 日付値を年の四半期としてローマ数字で書き出します。
	TIMEw.d 出力形式 (p. 204)	SAS 時間値を時間、分、および秒として hh:mm:ss.ss 形式で書き出します。
	TIMEAMPMw.d 出力形式 (p. 206)	SAS 時間値を時間、分、および秒として hh:mm:ss.ss 形式で AM または PM を付けて書き出します。

カテゴリ	言語要素	説明
	TODw.d 出力形式 (p. 208)	SAS 時間値と、SAS 日時値の時間部分を hh:mm:ss.ss 形式で書き出します。
	WEEKDATEw. 出力形式 (p. 212)	SAS 日付値を曜日と日付として day-of-week, month-name dd, yy (または yyyy)形式で書き出します。
	WEEKDATXw. 出力形式 (p. 214)	SAS 日付値を曜日と日付として day-of-week, dd month-name yy (または yyyy)形式で書き出します。
	WEEKDAYw. 出力形式 (p. 216)	SAS 日付値から曜日の値を書き出します。
	WEEKUw. 出力形式 (p. 217)	U アルゴリズムを使用して、10 進数の週番号を書き出します。
	WEEKVw. 出力形式 (p. 218)	V アルゴリズムを使用して、10 進数の週番号を書き出します。
	WEEKWw. 出力形式 (p. 220)	W アルゴリズムを使用して、10 進数の週番号を書き出します。
	YEARw. 出力形式 (p. 222)	SAS 日付値から年部分を書き出します。
	YYMMw. 出力形式 (p. 224)	SAS 日付値を<yy>yyMmm 形式で書き出します。M は月番号が M の後に続くことを示す区切り文字で、年は 2 桁または 4 桁の数字で書き出されます。
	YYMMxw. 出力形式 (p. 225)	SAS 日付値を<yy>yymm または<yy>yy-mm 形式で書き出します。出力形式名の x は、年と月を区切る特殊文字を表す文字で、ハイフン(-)、ピリオド(.)、空白文字、スラッシュ(/)、コロン(:)、または区切り文字なしになります。年は 2 桁または 4 桁の数字で書き出されます。
	YYMMDDw. 出力形式 (p. 227)	SAS 日付値を<yy>yymmdd または<yy>yy–mm–dd 形式で書き出します。ハイフンが区切り文字として使用され、年は 2 桁または 4 桁の数字で書き出されます。
	YYMMDDxw. 出力形式 (p. 229)	SAS 日付値を<yy>yymmdd または<yy>yy-mm-dd 形式で書き出します。出力形式名の x は、年、月、日を区切る特殊文字を表す文字です。特殊文字は、ハイフン(-)、ピリオド(.)、空白文字、スラッシュ(/)、コロン(:)または区切り文字なしになります。年は 2 桁または 4 桁のいずれかになります。
	YYMONw. 出力形式 (p. 232)	SAS 日付値を yymmm または yyyymmm 形式で書き出します。
	YYQw. 出力形式 (p. 233)	SAS 日付値を<yy>yyQq 形式で書き出します。Q は区切り文字として使用され、年は 2 桁または 4 桁の数字で書き出されます。また、q は四半期を表します。
	YYQxw. 出力形式 (p. 234)	SAS 日付値を<yy>yyq または<yy>yy-q 形式で書き出します。出力形式名の x は、年と四半期を区切る特殊文字を表す文字で、ハイフン(-)、ピリオド(.)、空白文字、スラッシュ(/)、コロン(:)、または区切り文字なしになります。年は 2 桁または 4 桁の数字で書き出されます。

カテゴリ	言語要素	説明
	YYQRw. 出力形式 (p. 236)	SAS 日付値を<yy>yyQqr 形式で書き出します。Q は区切り文字として使用され、年は 2 桁または 4 桁の数字で書き出されます。また qr はローマ数字表現の四半期を表します。
	YYQRxw. 出力形式 (p. 238)	SAS 日付値を<yy>yyqr または<yy>yy-qr 形式で書き出します。出力形式名の x は、年と四半期を区切る特殊文字を表す文字で、ハイフン(-)、ピリオド(.)、空白文字、スラッシュ(/)、コロン(:)、または区切り文字なしになります。年は 2 桁または 4 桁の数字で書き出されます。また qr は、ローマ数字表現の四半期です。
	YYQZw. 出力形式 (p. 240)	SAS 日付値を<yy><qq>形式で書き出します。年は 2 桁または 4 桁の数字で書き出され、qq は四半期を表します。
	YYWEEKUw. 出力形式 (p. 242)	U アルゴリズムを使用して、曜日情報を除いて 10 進数の週番号を書き出します。
	YYWEEKVw. 出力形式 (p. 243)	V アルゴリズムを使用して、曜日情報を除いて 10 進数の週番号を書き出します。
	YYWEEKWw. 出力形式 (p. 245)	W アルゴリズムを使用して、曜日情報を除いて 10 進数の週番号を書き出します。
文字	$BASE64Xw. 出力形式 (p. 96)	Base 64 エンコーディングを使用して、文字データを ASCII テキストに変換します。
	$BINARYw. 出力形式 (p. 97)	文字データを 2 進表現に変換します。
	$CHARw. 出力形式 (p. 98)	標準文字データを書き出します。
	$CSTRw. 出力形式 (p. 99)	文字引数の最後の空白ではない文字を探し、末尾の空白でない文字の後に null ターミネータを持つ文字列のコピーを渡します。
	$HEXw. 出力形式 (p. 100)	文字データを 16 進表現に変換します。
	$OCTALw. 出力形式 (p. 101)	文字データを 8 進表現に変換します。
	$QUOTEw. 出力形式 (p. 102)	データ値を二重引用符で囲んで書き込みます。
	$REVERJw. 出力形式 (p. 104)	文字データの末尾から順に書き込みます。空白も保持されます。
	$REVERSw. 出力形式 (p. 105)	文字データの末尾から順に、左揃えで書き込みます。
	$UPCASEw. 出力形式 (p. 106)	文字データを大文字に変換します。
	$UUIDw. 出力形式 (p. 107)	ユニバーサル一意識別子(UUID)形式で文字データを書き込みます。
	$w. 出力形式 (p. 108)	標準文字データを書き出します。

ディクショナリ

$BASE64Xw. 出力形式

Base 64 エンコーディングを使用して、文字データを ASCII テキストに変換します。

カテゴリ: 文字

配置: 左

構文

$BASE64X*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 1

範囲 1-32767

注 幅なしで$BASE64X.出力形式を指定すると、デフォルトの幅 1 が使用され、出力を生成するには不十分です。結果は返されません。

詳細

Base 64 は、ポジション指定スキームに基づき、ASCII 文字のみ使用したエンコード文字を指定するエンコーディング方式です。複数の Base 64 エンコーディングスキームが、メールやコンテンツマスキングなどの特定の用途に定義されています。ポジション 0 - 61 を文字 A - Z、a - z、0 - 9 にマップします。ポジション 62 は文字+にマップし、ポジション 63 は文字/にマップします。

次に、Base 64 エンコーディングの使用例を示します。

- XML ファイルの埋め込みバイナリデータ
- パスワードのエンコード
- URL のエンコード

エンコード結果の'='文字は、結果にゼロビットが埋め込まれたことを示します。エンコード文字をデコードするには、'='をデコードする値に含める必要があります。

例

次の例は、BASE64X*w*.出力形式を示しています。

ステートメント	結果
select put ('FCA01A7993BC', $base64x64.);	RkNBMDFBNzk5M0JD
select put ('MyPassword', $base64x64.);	TXlQYXNzd29yZA==
select put ('www.mydomain.com/myhiddenURL', $base64x64.);	d3d3Lm15ZG9tYWluLmNvbi9teWhpZGRlblVSTA==

$BINARYw. 出力形式

文字データを 2 進表現に変換します。

カテゴリ: 文字

配置: 左

構文

$BINARY*w*.

必須引数

w

出力フィールドの幅を指定します。

デフォルト: デフォルトの幅は、出力対象となる変数の長さに基づいて計算されます。

範囲: 1-32767

ヒント: 文字値はそれぞれ 8 つの 2 進表現の文字を生成するため、*w* の値には文字値の長さを 8 倍した値を指定します。

比較

$BINARY*w*.出力形式は、文字値を 2 進表現に変換します。$BINARY*w*.出力形式は、数値を 2 進表現に変換します。

例

例の表では、入力値 AB を使用して、$BINARY*w*.出力形式の効果を示しています。

ステートメント	結果	
select put('AB', $binary16.);	ASCII	EBCDIC
	0100000101000010	1100000111000010

関連項目

出力形式:

- “BINARYw. 出力形式” (126 ページ)

$CHARw. 出力形式

標準文字データを書き出します。

カテゴリ: 文字

配置: 左

別名: $*w*.と$F*w*.

構文

$CHAR*w*.

必須引数

w

出力フィールドの幅を指定します。数値または列範囲を指定できます。

デフォルト 変数の長さが未定義の場合は 8。それ以外の場合は、変数の長さ。

範囲 1-32767

比較

- $CHAR*w*.出力形式は、$*w*.および$F*w*.出力形式と同じです。

- $CHAR*w*.、$F*w*.、および$*w*.出力形式は先頭の空白を切り捨てません。先頭の空白を切り捨てるには、LEFT 関数を使用して、出力前に文字データを左揃えにします。

例

例の表では、入力値 XYZ を使用して、$CHAR*w*.出力形式の効果を示しています。

ステートメント	結果
select put ('XYZ',$char1.);	X
select put ('XYZ',$char2.);	XY
select put ('XYZ',$char.);	XYZ

$CSTRw. 出力形式

文字引数の最後の空白ではない文字を探し、末尾の空白でない文字の後に null ターミネータを持つ文字列のコピーを渡します。

カテゴリ: 文字

構文

$CSTR*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 1

範囲 1-32767

詳細

$CSTRw.出力形式を使用して、文字引数を null で終了された文字列として渡します。$CSTR は文字引数の最後の空白ではない文字を探し、末尾の空白でない文字の後に null ターミネータを持つ文字列のコピーを渡します。

例

例の表では、入力値 XYZ を使用して、$CSTR*w*.出力形式の効果を示しています。

ステートメント	結果
select put('XYZ',$cstr10.);	XYZ

$HEXw. 出力形式

文字データを 16 進表現に変換します。

カテゴリ: 文字

配置: 左

構文

$HEX*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト: デフォルトの幅は、出力対象となる変数の長さに基づいて計算されます。

範囲: 1-32767

注: *w* が変換したい変数の長さの 2 倍を超えている場合、$HEX*w*.により空白が埋め込まれます。

ヒント: 文字値はそれぞれ 2 つの 16 進表現の文字を生成するため、*w* の値には文字値の長さを 2 倍した値を指定します。

詳細

$HEX*w*.出力形式は、各文字を 2 つの 16 進文字に変換します。各空白は、末尾の空白を含めて 1 文字としてカウントされます。

比較

HEX*w*.出力形式は、バイナリ実数を 16 進数の表現に変換します。

例

次の例は、$HEX*w*.出力形式を示しています。

ステートメント	結果
select put ('AB',$hex5.);	4142

関連項目

出力形式:

- “HEXw. 出力形式” (170 ページ)

$OCTALw. 出力形式

文字データを 8 進表現に変換します。

カテゴリ: 文字

配置: 左

構文

$OCTAL*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト: デフォルトの幅は、出力対象となる変数の長さに基づいて計算されます。

範囲: 1-32767

ヒント: 文字値はそれぞれ 3 つの 8 進表現の文字を生成するため、*w* の値には文字値の長さを 3 倍した値を指定します。

比較

$OCTALw.出力形式は、文字値をその文字コードの 8 進表現に変換します。
OCTALw.出力形式は、数値を 8 進表現に変換します。

例

次の例は、$OCTALw.出力形式を示しています。

ステートメント	結果
select put ('art', $octal15.);	141162164
select put ('rice', $octal15.);	162151143145
select put ('bank', $octal15.);	142141156153

関連項目

出力形式:

- “OCTALw. 出力形式” (194 ページ)

$QUOTEw. 出力形式

データ値を二重引用符で囲んで書き込みます。

カテゴリ: 文字

配置: 左

構文

$QUOTE*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 変数の長さが未定義の場合は 2。それ以外の場合は、変数の長さ + 2。

範囲	2-32767
ヒント	*w* の幅を左と右の引用符を含められるよう十分な大きさにします。

詳細

次のリストは、$QUOTE*w*.出力形式を使用した場合に作成される出力です。

- データ値が引用符で囲まれていない場合、出力が二重引用符で囲まれます。
- データ値が引用符で囲まれていないが値に単一引用符が含まれている場合、次のアクションが実行されます。
  - データ値を二重引用符で囲む
  - 単一引用符を変更しない
- データ値が単一引用符で始まり終わっていて、値に二重引用符が含まれている場合、次のアクションが実行されます。
  - データ値を二重引用符で囲む
  - データ値に見つかった二重引用符を複製する
  - 単一引用符を変更しない
- データ値が単一引用符で始まり終わっていて、値に 2 つの隣接した単一引用符が含まれている場合、次のアクションが実行されます。
  - 値を二重引用符で囲む
  - 単一引用符を変更しない
- データ値が単一引用符で始まり終わっていて、二重引用符と隣接した単一引用符が含まれている場合、次のアクションが実行されます。
  - 値を二重引用符で囲む
  - データ値に見つかった二重引用符を複製する
  - 単一引用符を変更しない
- ターゲットフィールドの長さが文字列と引用符を含めるのに十分でない場合、そのフィールドの長さに合う長さの引用符付き文字列が返されます。たとえば、指定された値が ABCDE で、$QUOTE5.を指定した場合、値の DE は切り捨てられ、結果は"ABC"になります。

  注: 埋め込まれた引用符は、追加の引用符で囲む必要があります。A"B の値を指定すると、$QUOTE5.は値の B を切り捨てて、"A""と書き込みます。

例

次の例は、$QUOTE*w*.出力形式を示しています。

ステートメント	結果
select put('SAS',$quote.);	"SAS"
select put('SAS''s',$quote.);	"SAS's"
select put('ad''verb',$quote16.);	"ad'verb"
select put('"ad"""verb"',$quote20.);	"""ad""""verb"""

$REVERJw. 出力形式

文字データの末尾から順に書き込みます。空白も保持されます。

カテゴリ: 文字

配置: 右

構文

$REVERJ*w*.

必須引数

w

出力フィールドの幅を指定します。

デフォルト *w* が指定されていない場合は 1。

範囲 1-32767

比較

$REVERJ*w*.出力形式は、$REVERS*w*.出力形式とほぼ同じですが、$REVERS*w*.では先頭の空白がすべて削除され、結果を左揃えにします。

例

次の例は、$REVERJ*w*.出力形式を示しています。

ステートメント	結果
select put('ABCD###',$reverj7.);	###DCBA
select put('###ABCD',$reverj7.);	DCBA###

関連項目

出力形式:

- “$REVERSw. 出力形式” (105 ページ)

$REVERSw. 出力形式

文字データの末尾から順に、左揃えで書き込みます。

カテゴリ: 文字

配置: 左

構文

$REVERS*w*.

必須引数

w

出力フィールドの幅を指定します。

デフォルト *w* が指定されていない場合は 1。

範囲 1-32767

比較

$REVERS*w*.出力形式は、$REVERJ*w*.出力形式とほぼ同じですが、$REVERJ*w*. では結果を左揃えにしません。

例

次の例は、$REVERS*w*.出力形式を示しています。

ステートメント	結果
select put('ABCD###',$revers7.);	###DCBA
select put('###ABCD',$revers7.);	DCBA###

関連項目

出力形式:

- “$REVERJw. 出力形式” (104 ページ)

$UPCASEw. 出力形式

文字データを大文字に変換します。

カテゴリ: 文字

配置: 左

構文

$UPCASE*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 変数の長さが未定義の場合は 8。それ以外の場合は、変数の長さ。

範囲 1-32767

詳細

ハイフンやその他の記号などの特殊文字は、変更されません。

例

次の例は、$UPCASE*w.*出力形式を示しています。

ステートメント	結果
select put('coxe-ryan',$upcase9.);	COXE-RYAN

$UUIDw. 出力形式

ユニバーサル一意識別子(UUID)形式で文字データを書き込みます。

カテゴリ: 文字

配置: 左

構文

$UUID*w.*

必須引数

w
出力フィールドの幅を指定します。

範囲 1-200

詳細

必要な値を 16 進形式で指定する必要があります。それ以外の場合は、元の文字の 16 進表現が UUID として返されます。

注: 値を 16 進リテラルとして指定し、先頭に'x'を付ける場合、値は単一引用符でのみ囲むことができます。二重引用符は使用できません。

例

次の例は、$UUID*w.*出力形式を示しています。

ステートメント	結果
select put(x'1548611fb8cb6a47a721a6fd284e74f2', $uuid36.);	1548611f-b8cb-6a47-a721-a6fd284e74f2

$w. 出力形式

標準文字データを書き出します。

カテゴリ: 文字

配置: 左

別名: $F*w*.と$CHAR*w*.

構文

$*w*.

必須引数

w
出力フィールドの幅を指定します。数値または列範囲を指定できます。

デフォルト 変数の長さが未定義の場合は 1。それ以外の場合は、変数の長さ。

範囲 1-32767

比較

$*w*.、$F*w*.、および$CHAR*w*.出力形式は同じで、先頭の空白を切り捨てません。先頭の空白を切り捨てるには、LEFT 関数を使用して、出力前に文字データを左揃えにします。

例

次の例は、$*w*.出力形式を示しています。(文字#は空白を表します)。

ステートメント	結果
select put('#Cary',$5.);	#Cary
select put('#Cary',$f5.);	#Cary
select put('Carolina',$5.);	Carol

関連項目

関数:

- “LEFT 関数” (635 ページ)

B8601DAw. 出力形式

ISO 8601 規格の基本表記 *yyyymmdd* を使用して日付値を書き出します。

カテゴリ: 日付と時間
ISO 8601

配置: 左

制限事項: UTC タイムゾーンオフセット値はサポートされていません。

サポート: ISO 8601 Element 5.2.1.1, complete representation

構文

B8601DA*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 10

範囲 8-10

詳細

B8601DA 出力形式は、ISO 8601 規格の基本表記 *yyyymmdd* を使用して日付値を書き出します。

yyyy
4 桁の年です。

mm
01 から 12 までの 2 桁の月です(ゼロ埋め込み)。

dd
0 から 31 までの 2 桁の日です(ゼロ埋め込み)。

例

次の例は、B8601DA*w.*出力形式を示しています。次の例では、入力値として 20999 を使用します。20999 は、2017 年 6 月 29 日に対応する SAS 日付値です。

ステートメント	結果
select put (20999,b8601da.);	20170629

関連項目

"ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の処理" (*SAS 出力形式と入力形式: リファレンス*)

B8601DNw. 出力形式

ISO 8601 規格の基本表記 *yyyymmdd* を使用して、日時値から日付を書き出します。

カテゴリ: 日付と時間
ISO 8601

配置: 左

制限事項: UTC タイムゾーンオフセット値はサポートされていません。

サポート: ISO 8601 Element 5.2.1.1, complete representation

構文

B8601DN*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 10

範囲 8-10

詳細

B8601DN 出力形式は、ISO 8601 規格の基本表記 *yyyymmdd* を使用して、日時値から日付を書き出します。

yyyy
4 桁の年です。

mm
01 から 12 までの 2 桁の月です(ゼロ埋め込み)。

dd
01 から 31 までの 2 桁の日です(ゼロ埋め込み)。

例

次の例は、B8601DN*w.*出力形式を示しています。次の例では、入力値として 1793308532 を使用します。1793308532 は、2016 年 10 月 28 日に対応する SAS 日付値です。

ステートメント	結果
select put (1793308532,b8601dn.);	20161028

関連項目

"ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の処理" (*SAS 出力形式と入力形式: リファレンス*)

B8601DTw.d 出力形式

ISO 8601 規格の基本表記 *yyyymmdd*T*hhmmss*<*ffffff*>を使用して日時値を書き出します。

カテゴリ: 日付と時間
ISO 8601

配置: 左

制限事項: UTC タイムゾーンオフセット値はサポートされていません。

サポート: ISO 8601 Element 5.4.1, complete representation

構文

B8601DT*w.d*

必須引数

w
出力フィールドの幅を指定します。

デフォルト 19

範囲 15-26

d
秒の端数を表す秒値の小数点以下の桁数を指定します。この引数はオプションです。

デフォルト 0

範囲 0-6

詳細

B8601DT 出力形式は、ISO 8601 規格の基本表記 *yyyymmdd*T*hhmmss*<*ffffff*>を使用して、日時値を書き出します。

yyyy
4 桁の年です。

mm
01 から 12 までの 2 桁の月です(ゼロ埋め込み)。

dd
01 から 31 までの 2 桁の日です(ゼロ埋め込み)。

hh
00 から 23 までの 2 桁の時間です(ゼロ埋め込み)。

mm
00 から 59 までの 2 桁の分です(ゼロ埋め込み)。

ss
00 から 59 までの 2 桁の秒です(ゼロ埋め込み)。

ffffff
6 桁までの有効桁数を持つ、オプションの秒の分数です。各桁は 0 から 9 までです。

例

次の例は、B8601DT*w.d.*出力形式を示しています。次の例では、入力値として 1793308532 を使用します。1793308532 は、2016 年 10 月 28 日の午後 9:15:32 に対応する SAS 日時値です。

ステートメント	結果
select put (1793308532, b8601dt.);	20161028T211532

関連項目

"ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の処理" (*SAS 出力形式と入力形式: リファレンス*)

B8601DZw. 出力形式

ISO 8601 規格の日時とタイムゾーンの基本表記 *yyyymmdd*T*hhmmss*+0000 を使用して、基準子午線の協定世界時(UTC)の日時値を書き出します。

カテゴリ: 日付と時間
ISO 8601

配置: 左

サポート: ISO 8601 Element 5.4.1, complete representation

構文

B8601DZ*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 26

範囲 20-35

詳細

UTC 値は、イギリスのグリニッジの基準子午線に基づく時間とタイムゾーンを示します。B8601DZ 出力形式は、次の ISO 8601 規格の基本日時表記のいずれかを使用して、基準子午線の日時に対応した SAS 日時値を書き出します。

- *yyyymmdd*T*hhmmss*+0000

  注: この形式は、*w* の長さがこのタイムゾーン表記に十分な場合に使用します。

- *yyyymmdd*T*hhmmss*Z

  注: この形式は、*w* の長さが+0000 タイムゾーン表記に十分でない場合に使用します。

yyyy
4 桁の年です。

mm
01 から 12 までの 2 桁の月です(ゼロ埋め込み)。

dd
01 から 31 までの 2 桁の日です(ゼロ埋め込み)。

hh
00 から 23 までの 2 桁の時間です(ゼロ埋め込み)。

mm
00 から 59 までの 2 桁の分です(ゼロ埋め込み)。

ss
00 から 59 までの 2 桁の秒です(ゼロ埋め込み)。

+0000
基準子午線(イギリスのグリニッジ)の UTC 時間を示します。

タイムゾーンオフセットが指定された ISO 8601 規格の時間値または日時値は、オフセットに示された時間数と分数によって調整されます。その後、タイムゾーンオフセットが基準子午線(イギリスのグリニッジ)の時間または日時として処理されます。B8601DZ 出力形式は、常に基準子午線オフセット値の+0000 を使用して日時値を書き出します。+0000 以外のタイムゾーンオフセットを使用した日時を書き出すには、“B8601LZw. 出力形式” (115 ページ)を参照してください。

制約: 短い形式+00 はサポートされていません。

Z
時間が基準子午線(イギリスのグリニッジ)に対応した時間、つまり+0000UTC 時間であることを示します。出力形式の幅が+0000 表記をサポートしない場合に Z が使用されます。

例

次の例は、B8601DZ*w.*出力形式を示しています。次の例では、入力値として 1793308532 を使用します。1793308532 は、2016 年 10 月 28 日の午後 9:15:32 に対応する SAS 日付値です。

ステートメント	結果
select put (1793308532,b8601dz.);	20161028T211532+0000

関連項目

"ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の処理" (*SAS 出力形式と入力形式: リファレンス*)

B8601LZw. 出力形式

ISO 8601 規格の基本表記 *hhmmss+|–hhmm* を使用して、ローカル時間と UTC 間のタイムゾーンオフセット差を追加することにより、時間値をローカル時間として書き出します。

カテゴリ: 日付と時間
ISO 8601

配置: 左

サポート: ISO 8601 Elements 5.3.3 and 5.3.4.2

構文

B8601LZ*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 14

範囲 9-20

詳細

B8601LZ 出力形式は、ISO 8601 規格の基本表記 *hhmmss+|–hhmm* を使用して時間値を調整せずに書き出し、ローカル SAS セッションに対応した UTC タイムゾーンオフセットを追加します。

hh
00 から 23 までの 2 桁の時間です(ゼロ埋め込み)。

mm
00 から 59 までの 2 桁の分です(ゼロ埋め込み)。

ss
00 から 59 までの 2 桁の秒です(ゼロ埋め込み)。

+|–*hhmm*
基準子午線時間からの時間と分の符号付きオフセットです。オフセットは+|–*hhmm* (つまり+または–を伴う 4 文字)である必要があります。

+は基準子午線の東部タイムゾーンに、-は基準子午線の西部タイムゾーンにそれぞれ使用します。たとえば、+0200 は基準子午線の東部での 2 時間差を表し、-0600 は基準子午線の西部での 6 時間差を表します。

制約: 短い形式+|–*hh* はサポートされていません。

SAS では UTC 時間を B8601TZ 入力形式を使用して読み込む際に、オフセット調整後の時間が 24 時間を超えていたり 00 時間未満である場合、オフセット調整後の時間が 000000 と 235959 の間になるように値を調整します。B8601LZ 出力形式をこの時間範囲外の時間に適用する場合、時間はアスタリスクを使用してフォーマットされ、その値が範囲外であることが示されます。

例

次の例は、B8601LZ*w.*出力形式を示しています。この例では、20999 を入力値として使用します。これは、ローカル時間値 054959-0000 です。

ステートメント	結果
select put (20999,b8601lz.);	054959-0000

関連項目

“ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の処理” (*SAS 出力形式と入力形式: リファレンス*)

B8601TMw.d 出力形式

ISO 8601 規格の基本表記 *hhmmss<ffff>*を使用して、時間値を書き出します。

カテゴリ: 日付と時間
ISO 8601

配置: 左

制限事項: UTC タイムゾーンオフセット値はサポートされていません。

サポート: ISO 8601 Element 5.3.1.1, complete representation

構文

B8601TM*w.d*

必須引数

w
出力フィールドの幅を指定します。

デフォルト 8

範囲 6-15

d
秒の端数を表す秒値の小数点以下の桁数を指定します。この引数はオプションです。

デフォルト 0

範囲 0-6

詳細

B8601TM 出力形式は、ISO 8601 規格の基本時間表記 *hhmmss<ffffff>*を使用して、SAS 時間値を書き出します。

hh
00 から 23 までの 2 桁の時間です(ゼロ埋め込み)。

mm
00 から 59 までの 2 桁の分です(ゼロ埋め込み)。

ss
00 から 59 までの 2 桁の秒です(ゼロ埋め込み)。

fffff
6 桁までの有効桁数を持つ、オプションの秒の分数です。各桁は 0 から 9 までです。

例

次の例は、B8601TM*w.d.*出力形式を示しています。この例では、054959 の時間値に対応する 20999 の入力値を使用しています。

ステートメント	結果
select put (20999,b8601tm.);	054959

関連項目

"ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の処理" (*SAS 出力形式と入力形式: リファレンス*)

B8601TZw. 出力形式

ISO 8601 規格の基本時間表記 *hhmmss+|–hhmm* を使用して、時間値を協定世界時(UTC)に調整し、時間値を書き出します。

カテゴリ: 日付と時間
ISO 8601

配置: 左

サポート: ISO 8601 Elements 5.3.3 and 5.3.4

構文

B8601TZ*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 14

範囲 9-20

詳細

UTC 時間値は、イギリスのグリニッジの基準子午線に基づく時間とタイムゾーンを示します。B8601TZ 出力形式は、時間値を基準子午線の時間に調整し、その時間値を次の ISO 8601 規格の基本時間表記のいずれかで書き出します。

- *hhmmss*+|–*hhmm*

  注: この形式は、*w* の長さがこの時間表記に十分である場合に使用します。

- *hhmmss*Z

  注: この形式は、*w* の長さが+|–*hhmm* タイムゾーン表記に十分でない場合に使用します。

hh
00 から 23 までの 2 桁の時間です(ゼロ埋め込み)。

mm
00 から 59 までの 2 桁の分です(ゼロ埋め込み)。

ss
00 から 59 までの 2 桁の秒です(ゼロ埋め込み)。

+|–*hh:mm*
基準子午線時間からの時間と分の符号付きオフセットです。オフセットは+|–*hhmm* (つまり+または–を伴う 4 文字)である必要があります。

+は基準子午線の東部タイムゾーンに、-は基準子午線の西部タイムゾーンにそれぞれ使用します。たとえば、+0200 は基準子午線の東部での 2 時間差を表し、-0600 は基準子午線の西部での 6 時間差を表します。

制約: 短い形式+|–*hh* はサポートされていません。

Z
時間が基準子午線(イギリスのグリニッジ)に対応した時間、つまり+0000 UTC 時間であることを示します。出力形式の幅が+0000 表記をサポートしない場合に Z が使用されます。

SAS では UTC 時間を B8601TZ 入力形式を使用して読み込む際に、オフセット調整後の時間が 24 時間を超えていたり 00 時間未満である場合、オフセット調整後の時間が 000000 と 240000 の間になるように値を調整します。B8601TZ 出力形式をこの時間範囲外の時間に適用する場合、時間はアスタリスクを使用してフォーマットされ、その値が範囲外であることが示されます。

比較

- 000000 と 240000 の間の時間値に対して、B8601TZ 出力形式は時間値を基準子午線の時間に調整し、調整後の値を国際規格の拡張時間表記で書き出します。
- B8601LZ 出力形式は時間を調整せず、ローカル SAS セッションの UTC タイムゾーンオフセットを使用して時間値を国際規格の拡張時間表記で書き出します。

例

次の例は、B8601TZ*w.*出力形式を示しています。この例では、054959+0000 の時間値に対応する 20999 の入力値を使用しています。

ステートメント	結果
select put (20999,b8601tz.);	054959+0000

関連項目

“ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の処理” (*SAS 出力形式と入力形式: リファレンス*)

BESTDOTXw. 出力形式

SAS が最適な表記を選び、小数点としてドット(.)を使うように指定します。

カテゴリ: 数値

操作: DECIMALCONV=システムオプションを STDIEEE に設定した場合、この出力形式を使用して書き出される出力が前のリリースと多少異なる可能性があります。詳細については、“DECIMALCONV= システムオプション” (*SAS システムオプション: リファレンス*)を参照してください。

構文

BESTDOTX.*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 12

範囲 1-32

ヒント 0 から.01 までの数字を出力する場合、過度に丸めないように 7 以上のフィールド幅を使用します。0 から-.01 までの数字を出力する場合は、8 以上のフィールド幅を使用します。

詳細

NLDECSEPARATOR システムオプションが無効である場合、BEST*w*.と BESTDOTX*w*.はデータを同じように処理します。NLDECSEPARATOR システムオプションが有効であれば、BEST*w*.か BESTDOTX*w*.によって出力形式が異なります。違いについては次の表を参照してください。

LOCALE オプション	ロケールに応じたデフォルトの小数点文字	NLDECSEPARATOR オプション	BEST*w*.の場合の小数点文字	BESTDOTX*w*.の場合の小数点文字
en_US	ドット(.)	無効(デフォルト)	ドット(.)	ドット(.)
		有効	ドット(.)	ドット(.)
fr_FR	カンマ(,)	無効(デフォルト)	ドット(.)	ドット(.)
		有効	カンマ(,)	ドット(.)

NLDECSEPARATOR システムオプションの詳細については、*SAS 各国語サポート (NLS): リファレンスガイド*を参照してください。

比較

- BESTDOTX*w* 出力形式の場合、出力フィールド内にできるだけ多くの有効数字を収めようとします。整数は小数点付きで出力されます。
- BEST*w*.出力形式は、出力フィールドに可能な限りの有効桁数を書き出しますが、数字の大きさが異なる場合に小数点以下の桁数を揃えません。整数は小数点なしで出力されます。

例

次のプログラムは、BESTDOTX*w*.出力形式を示しています。

```
options nld locale=en_US;
title 'Locale en_US';
proc fedsql;
drop table test force;
create table test (x double);
insert into test values (12.34);
select x as "default", put(x, best.) as "best",
  put(x, bestdotx.) as "bestdotx" from test;
quit;

options nld locale=fr_FR;
title 'Locale fr_FR';
proc fedsql;
drop table test2 force;
```

```
create table test2 (x double);
insert into test2 values (12.34);
select x as "default", put(x, best.) as "best",
  put(x, bestdotx.) as "bestdotx" from test2;
quit;
```

関連項目

出力形式:

- “BESTw. 出力形式” (122 ページ)

BESTw. 出力形式

SAS により最適な出力が選択されます。

カテゴリ: 数値

操作: DECIMALCONV=システムオプションを STDIEEE に設定した場合、この出力形式を使用して書き出される出力が前のリリースと多少異なる可能性があります。詳細については、“DECIMALCONV= システムオプション” (*SAS システムオプション: リファレンス*)を参照してください。

構文

BEST*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 12

範囲 1-32

ヒント 0 から.01 までの数字を出力する場合、過度に丸めないように 7 以上のフィールド幅を使用します。0 から-.01 までの数字を出力する場合は、8 以上のフィールド幅を使用します。

詳細

BEST*w*.出力形式は、数値を書き込むためのデフォルトの出力形式です。出力形式の指定がない場合、SAS は、使用可能なフィールド幅に従って、値に関する最も多く

の情報を提供する出力形式を選択します。BEST*w*.は値を丸め、SAS が指定された幅内で小数点以下の部分に少なくとも 1 つの有効数字を表示できる場合、BEST*w*.は結果を 10 進数で生成します。それ以外の場合は、指数表記で結果が生成されます。SAS では、表現に使用する出力形式にかかわらず、常に完全な値を保存します。

比較

- BEST*w*.出力形式は、出力フィールドに可能な限りの有効桁数を書き出しますが、数字の大きさが異なる場合に小数点以下の桁数を揃えません。整数は小数点なしで出力されます。
- BESTDOTX*w*.出力形式の場合、出力フィールド内にできるだけ多くの有効数字を収めようとします。整数は小数点付きで出力されます。
- D*w.p* 出力形式は数字を指定した有効桁数で書き出し、BEST*w*.出力形式よりも小数点揃えを行います。
- *w.d* 出力形式は小数点揃えを行いますが、すべての数字が必ずしも同じ有効桁数で表示されません。

例

例の表では、値 1257000 を使用して、BEST*w*.出力形式の効果を示しています。

ステートメント	結果
select put(1257000,best6.);	1.26E6
select put(1257000,best3.);	1E6

関連項目

出力形式:

- “BESTDw.p 出力形式” (124 ページ)
- “BESTDOTXw. 出力形式” (120 ページ)
- “Dw.p 出力形式” (130 ページ)
- “w.d 出力形式” (211 ページ)

BESTDw.p 出力形式

同じような大きさの数値に対しては小数点以下の桁数を揃えて出力します。整数は小数なしで出力します。

カテゴリ: 数値

操作: DECIMALCONV=システムオプションを STDIEEE に設定した場合、この出力形式を使用して書き出される出力が前のリリースと多少異なる可能性があります。詳細については、“DECIMALCONV= システムオプション” (*SAS システムオプション: リファレンス*)を参照してください。

構文

BESTD*w*.[*p*]

必須引数

w
出力フィールドの幅を指定します。

デフォルト 12

範囲 1-32

オプション引数

p
有効桁数を指定します。

デフォルト 3

範囲 0 - *w*–1

要件 *w* より小さい数にします。

ヒント *p* が省略される、または 0 が指定されると、*p* は 3 に設定されます。

詳細

BESTD*w.p* 出力形式は、同じような大きさの値グループ内で小数点を揃えて値を書き出します。整数は小数点なしで出力されます。*p* の値が大きくなると、出力されるデータ値の有効桁数が増え、小数点揃えのシフトが増す可能性があります。*p* の値が小さくなると、出力されるデータ値の有効桁数が減り、小数点揃えの可能性が高くなります。

出力形式は、値の範囲に対して、値をより少ない小数点以下の桁数で表示できる場合でも、出力に適した小数点以下の桁数を選択します。

比較

- BEST*w*.出力形式は、出力フィールドに可能な限りの有効桁数を書き出しますが、数字の大きさが異なる場合に小数点以下の桁数を揃えません。整数は小数点なしで出力されます。
- BESTD*w.p* 出力形式は、BEST*w*.出力形式と D*w.p* 出力形式を組み合わせたもので、すべての数値データに適用されます。また、BEST*w*.出力形式よりも小数点揃えに優れています。
- D*w.p* 出力形式は数字を指定した有効桁数で書き出し、BEST*w* 出力形式よりも小数点揃えを行います。
- *w.d* 出力形式は小数点揃えを行いますが、すべての数字が必ずしも同じ有効桁数で表示されません。

例

次の例は、BESTD*w.p* 出力形式を示しています。

ステートメント	結果
select put (12345, bestd14.);	12345
select put (123.45, bestd14.);	123.4500000
select put (1.2345, bestd14.);	1.2345000
select put (.12345, bestd14.);	0.1234500
select put (1.23456789, bestd14.);	1.23456789

関連項目

出力形式:

- “BESTw. 出力形式” (122 ページ)
- “Dw.p 出力形式” (130 ページ)
- “w.d 出力形式” (211 ページ)

BINARYw. 出力形式

数値を 2 進表現に変換します。

カテゴリ: 数値

配置: 左

構文

BINARY*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 8

範囲 1-64

比較

BINARY*w*. は、数値を 2 進表現に変換します。$BINARY*w*.出力形式は、文字値を 2 進表現に変換します。

例

次の例は、BINARY*w*.出力形式を示しています。

ステートメント	結果
select put (123.45, binary8.);	01111011
select put (123, binary8.);	01111011
select put (–123, binary8.);	10000101

関連項目

出力形式:

- “$BINARYw. 出力形式” (97 ページ)

COMMAw.d 出力形式

3 桁ごとにカンマを入れ、小数部分との区切りにはピリオドを使用して数値を出力します。

カテゴリ: 数値

配置: 右

操作: DECIMALCONV=システムオプションを STDIEEE に設定した場合、この出力形式を使用して書き出される出力が前のリリースと多少異なる可能性があります。詳細については、“DECIMALCONV= システムオプション” (*SAS システムオプション: リファレンス*)を参照してください。

構文

COMMA*w*.[*d*]

必須引数

w
出力フィールドの幅を指定します。

デフォルト 6

範囲 1-32

ヒント *w* の幅に数値、カンマ、オプションの小数点の書き出しに十分な値を指定します。

オプション引数

d
数値の小数点以下の桁数を指定します。

範囲 0-31

要件 *w* より小さい数にします。

比較

- COMMA*w.d* 出力形式は COMMAX*w.d* 出力形式とほぼ同じですが、COMMAX*w.d* 出力形式では小数点とカンマの役割が逆になっています。この規則はヨーロッパの国で共通です。
- COMMA*w.d* 出力形式は DOLLAR*w.d* 出力形式とほぼ同じですが、COMMA*w.d* 出力形式は先頭のドル記号を出力しません。

例

次の例は、COMMA*w.d* 出力形式を示しています。

ステートメント	結果
select put (23451.23,comma10.2);	23,451.23
select put (123451.234,comma10.2);	123,451.23

関連項目

出力形式:

- “COMMAXw.d 出力形式” (128 ページ)
- “DOLLARw.d 出力形式” (141 ページ)

COMMAXw.d 出力形式

3 桁ごとにピリオドを入れ、小数部分との区切りにはカンマを使用して数値を出力します。

カテゴリ: 数値

配置: 右

操作: DECIMALCONV=システムオプションを STDIEEE に設定した場合、この出力形式を使用して書き出される出力が前のリリースと多少異なる可能性があります。詳細については、“DECIMALCONV= システムオプション” (*SAS システムオプション: リファレンス*)を参照してください。

構文

COMMAX[*w*.*d*]

必須引数

w
出力フィールドの幅を指定します。

デフォルト 6

範囲 1-32

ヒント *w* の幅に数値、カンマ、オプションの小数点の書き出しに十分な値を指定します。

オプション引数

d
数値の小数点以下の桁数を指定します。

範囲 0-31

要件 *w* より小さい数にします。

比較

COMMA*w.d* 出力形式は COMMAX*w.d* 出力形式とほぼ同じですが、COMMAX*w.d* 出力形式では小数点とカンマの役割が逆になっています。この規則はヨーロッパの国で共通です。

例

次の例は、COMMAX*w.d* 出力形式を示しています。

ステートメント	結果
select put (23451.23,commax10.2);	23.451,23
select put (123451.234,commax10.2);	123.451,23

関連項目

出力形式:

- “COMMAw.d 出力形式” (127 ページ)

Dw.p 出力形式

値の桁数をできる限り最大にし、同じような大きさの数値に対しては小数点以下の桁数を揃えて出力します。

カテゴリ: 数値

配置: 右

操作: DECIMALCONV=システムオプションを STDIEEE に設定した場合、この出力形式を使用して書き出される出力が前のリリースと多少異なる可能性があります。詳細については、“DECIMALCONV= システムオプション” (*SAS システムオプション: リファレンス*)を参照してください。

構文

D[*w*.*p*]

必須引数

w
出力フィールドの幅を指定します。

デフォルト 12

範囲 1-32

p
有効桁数を指定します。

デフォルト 3

範囲 0-16

要件 *w* より小さい数にします。

詳細

D*w*.*p* 出力形式は、同じような大きさの値グループ内で小数点を揃えて値を書き出します。*p* の値が大きくなると、出力されるデータ値の有効桁数が増え、小数点揃えのシフトが増す可能性があります。*p* の値が小さくなると、出力されるデータ値の有効桁数が減り、小数点揃えの可能性が高くなります。

比較

- BEST*w*.出力形式は、出力フィールドに可能な限りの有効桁数を書き出しますが、数字の大きさが異なる場合に小数点以下の桁数を揃えません。
- D*w.p* は数字を指定した有効桁数で書き出し、BEST*w*.出力形式よりも小数点揃えを行います。

例

次の例は、D*w.p* 出力形式を示しています。

ステートメント	結果
select put (12345,d10.4);	12345.0
select put (1234.5,d10.4);	1234.5
select put (123.45,d10.4);	123.45000
select put (12.345,d10.4);	12.34500
select put (1.2345,d10.4);	1.23450
select put (.12345,d10.4);	0.12345

関連項目

出力形式:

- “BESTw. 出力形式” (122 ページ)

DATEAMPMw.d 出力形式

SAS 日時値を、AM または PM を付けて *ddmmmyy:hh:mm:ss.ss* 形式で書き出します。

カテゴリ: 日付と時間

配置: 右

操作: DECIMALCONV=システムオプションを STDIEEE に設定した場合、この出力形式を使用して書き出される出力が前のリリースと多少異なる可能性があります。詳細については、“DECIMALCONV= システムオプション” (*SAS システムオプション: リファレンス*)を参照してください。

構文

DATEAMPM*w*.[*d*]

必須引数

w
出力フィールドの幅を指定します。

デフォルト 19

範囲 7-40

ヒント AM または PM を出力するには、*w* に少なくとも 13 を指定する必要があります。幅の値が 10 と 12 の間の場合、24 時間表記が出力されます。

オプション引数

d
秒値の小数点以下の桁数を指定します。

範囲 0-39

要件 *w* より小さい数にします。

注 *w*–*d*< 17 の場合、SAS は小数値を切り捨てます。

詳細

DATEAMPM*w.d* 出力形式は、SAS 日時値を *ddmmmyy:hh:mm:ss.ss* 形式で書き出します。構文の説明は次のとおりです。

dd
月の日を表す整数です。

mmm
月名の最初の 3 文字になります。

yy
年を表す 2 桁の整数です。

hh
時間を表す整数です。

mm
分を表す整数です。

ss.ss
小数点以下 2 桁の秒数です。

比較

DATEAMPM*w.d* 出力形式は DATETIME*w.d* 出力形式とほぼ同じですが、DATEAMPM*w.d* は時間の最後に AM または PM を出力します。

例

例の表では 1679344494 を入力値として使用します。この値は 2013 年 3 月 19 日の午後 08:34:54 に相当する SAS 日時値です。

ステートメント	結果
select put (1679344494,dateampm.);	19MAR13:08:34:54 PM
select put (1679344494,dateampm7.);	19MAR13
select put (1679344494,dateampm10.);	19MAR13:20
select put (1679344494,dateampm13.);	19MAR13:08 PM
select put (1679344494,dateampm22.2.);	19MAR13:08:34:54.00 PM

関連項目

出力形式:

- “DATETIMEw.d 出力形式” (133 ページ)

DATETIMEw.d 出力形式

SAS 日時値を *ddmmmyy:hh:mm:ss.ss* 形式で書き出します。

カテゴリ: 日付と時間

配置: 右

操作: DECIMALCONV=システムオプションを STDIEEE に設定した場合、この出力形式を使用して書き出される出力が前のリリースと多少異なる可能性があります。詳細については、“DECIMALCONV= システムオプション” (*SAS システムオプション: リファレンス*)を参照してください。

構文

DATETIME*w*.[*d*]

必須引数

w
出力フィールドの幅を指定します。

デフォルト 16

範囲 7-40

ヒント SAS では、日付、時間、秒を含む SAS 日時値を書き込むために、最低でも *w* 値を 16 にする必要があります。オプションの小数点以下の秒数で値を返すには、*w* を 2 桁増やし、*d* に値を追加します。

オプション引数

d
秒値の小数点以下の桁数を指定します。

範囲 0-39

要件 *w* より小さい数にします。

詳細

DATETIME*w.d* 出力形式は、SAS 日時値を *ddmmmyy:hh:mm:ss.ss* 形式で書き出します。構文の説明は次のとおりです。

dd
月の日を表す整数です。

mmm
月名の最初の 3 文字になります。

yy
年を表す 2 桁の整数です。

hh
時間を 24 時間表記で表す整数です。

mm
分を表す整数です。

ss.ss
小数点以下 2 桁の秒数です。

注: *w*–*d*<17 の場合、SAS は小数値を切り捨てます。

比較

DATEAMPM*w.d* 出力形式は DATETIME*w.d* 出力形式とほぼ同じですが、DATEAMPM*w.d* は時間の最後に AM または PM を出力します。

例

例の表では 1699674559 を入力値として使用します。この値は 2013 年 11 月 10 日の午前 3:49:19 に相当する SAS 日時値です。

ステートメント	結果
select put (1699674559,datetime.);	10NOV13:03:49:19
select put (1699674559,datetime7.);	10NOV13
select put (1699674559,datetime12.);	10NOV13:03
select put (1699674559,datetime18.);	10NOV13:03:49:19
select put (1699674559,datetime18.1);	10NOV13:03:49:19.0
select put (1510285759,datetime19.);	10NOV2013:03:49:19
select put (1699674559,datetime20.1);	10NOV2013:03:49:19.0
select put (1699674559,datetime21.2);	10NOV2013:03:49:19.00

関連項目

出力形式:

- “DATEw. 出力形式” (135 ページ)
- “DATEAMPMw.d 出力形式” (131 ページ)
- “TIMEw.d 出力形式” (204 ページ)

関数:

- “DATETIME 関数” (442 ページ)

DATEw. 出力形式

SAS 日付値を *ddmmmyy*、*ddmmmyyyy* または *dd-mmm-yyyy* 形式で書き出します。

カテゴリ: 日付と時間

配置: 右

構文

DATE*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 7

範囲 5-11

ヒント 4 桁の年を日、月、年の間を区切らずに出力する場合は、幅に 9 を指定します。4 桁の年を日、月、年の間をハイフンで区切って出力する場合は、幅に 11 を指定します。

詳細

DATE*w*.出力形式は、SAS 日付値を *ddmmmyy*、*ddmmmyyyy* または *dd-mmm-yyyy* 形式で書き出します。構文の説明は次のとおりです。

dd
月の日を表す整数です。

mmm
月名の最初の 3 文字になります。

yy または *yyyy*
年を表す 2 桁または 4 桁の整数です。

例

例の表では、19431 を入力値として使用します。この値は 2013 年 3 月 14 日に相当する SAS 日付値です。

ステートメント	結果
select put (19431,date5.);	14MAR
select put (19431,date6.);	14MAR
select put (19431,date7.);	14MAR13

ステートメント	結果
select put (19431,date8.);	14MAR13
select put (19431,date9.);	14MAR2013

関連項目

関数:

- “DATE 関数” (439 ページ)

DAYw. 出力形式

SAS 日付値の日付部分を書き出します。

カテゴリ: 日付と時間

配置: 右

構文

DAY*w*.

必須引数

w

出力フィールドの幅を指定します。

デフォルト 2

範囲 2-32

例

例の表では、19523 を入力値として使用します。この値は 2013 年 6 月 14 日に相当する SAS 日付値です。

ステートメント	結果
select put(19523,day2.);	14

DDMMYYw. 出力形式

SAS 日付値を *ddmm<yy>yy* または *dd/mm/<yy>yy* 形式で書き出します。スラッシュが区切り文字として使用され、年は 2 桁または 4 桁の数字で書き出されます。

カテゴリ: 日付と時間

配置: 右

構文

DDMMYY*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 8

範囲 2-10

操作 *w* の値が 2 から 5 までの場合、日と月が可能な範囲で表示されます。*w* が 7 の場合、年を 2 桁で表し、スラッシュを使わずに出力します。

詳細

DDMMYY*w*.出力形式は、SAS 日付値を *ddmm<yy>yy* または *dd/mm/<yy>yy* 形式で書き出します。構文の説明は次のとおりです。

dd
月の日を表す整数です。

/
区切り文字です。

mm
月を表す整数です。

<yy>yy
年を表す 2 桁または 4 桁の整数です。

例

次の例では、19704 を入力値として使用します。この値は、2013 年 12 月 12 日に相当する SAS 日付値です。

ステートメント	結果
select put(19704,ddmmyy.);	12/12/13
select put(19704,ddmmyy5.);	12/12
select put(19704,ddmmyy6.);	121213
select put(19704,ddmmyy7.);	121213
select put(19704,ddmmyy8.);	12/12/13
select put(19704,ddmmyy10.);	12/12/2013

関連項目

出力形式:

- “DATEw. 出力形式” (135 ページ)
- “DDMMYYxw. 出力形式” (139 ページ)
- “MMDDYYw. 出力形式” (179 ページ)
- “YYMMDDw. 出力形式” (227 ページ)

関数:

- “MDY 関数” (664 ページ)

DDMMYYxw. 出力形式

SAS 日付値を *ddmm<yy>yy* または *dd-mm-yy<yy>*形式で書き出します。出力形式名の *x* は、日、月、年を区切る特殊文字を表す文字で、ハイフン(-)、ピリオド(.)、空白文字、スラッシュ(/)、コロン(:)、または区切り文字なしになります。年は 2 桁または 4 桁のいずれかになります。

カテゴリ: 日付と時間

配置: 右

構文

DDMMYY*xw*.

必須引数

x
区切り文字を示します。または、区切り文字を日、月、年の間に挿入しないことを示します。有効な値は次のとおりです。

B
空白で区切ります。

C
コロンで区切ります。

D
ハイフンで区切ります。

N
区切り文字なしを表します。

P
ピリオドで区切ります。

S
スラッシュで区切ります。

w
出力フィールドの幅を指定します。

デフォルト　8

範囲　2-10

操作　*w* の値が 2 から 5 までの場合、日と月が可能な範囲で表示されます。*w* が 7 の場合、日付の年は 2 桁で、区切り文字を使用せずに表示されます。

x の値が N の場合、幅の範囲は 2–8 に変化します。

詳細

DDMMYY*xw*.出力形式は、SAS 日付値を *ddmm<yy>yy* または *dd*X*mm*X*<yy>yy* 形式で書き出します。構文の説明は次のとおりです。

dd
月の日を表す整数です。

X
指定された区切り文字です。

mm
月を表す整数です。

<yy>yy
年を表す 2 桁または 4 桁の整数です。

例

次の例では、19431 を入力値として使用します。この値は、2013 年 03 月 14 日に相当する SAS 日付値です。

ステートメント	結果
select put(19431,ddmmyyc5.);	14:03
select put(19431,ddmmyyd8.);	14-03-13
select put(19431,ddmmyyn8.);	14032013
select put(19431,ddmmyyp10.);	14.03.2013

関連項目

出力形式:

- “DATEw. 出力形式” (135 ページ)
- “DDMMYYw. 出力形式” (138 ページ)
- “MMDDYYxw. 出力形式” (181 ページ)
- “YYMMDDxw. 出力形式” (229 ページ)

関数:

- “DAY 関数” (443 ページ)
- “MDY 関数” (664 ページ)
- “MONTH 関数” (676 ページ)
- “YEAR 関数” (887 ページ)

DOLLARw.d 出力形式

先頭にドル($)記号を付け、3 桁ごとにカンマを入れ、小数部分との区切りにはピリオドを使用して数値を出力します。

カテゴリ: 数値

配置: 右

操作: DECIMALCONV=システムオプションを STDIEEE に設定した場合、この出力形式を使用して書き出される出力が前のリリースと多少異なる可能性があります。詳細については、“DECIMALCONV= システムオプション” (*SAS システムオプション: リファレンス*)を参照してください。

構文

DOLLAR*w*.[*d*]

必須引数

w
出力フィールドの幅を指定します。

デフォルト 6

範囲 2-32

オプション引数

d
数値の小数点以下の桁数を指定します。

範囲 0-31

要件 *w* より小さい数にします。

詳細

DOLLAR*w.d* 出力形式は、先頭にドル($)記号を付け、3 桁ごとにカンマを入れ、小数部分との区切りにはピリオドを使用して数値を出力します。

比較

- DOLLAR*w.d* 出力形式は DOLLARX*w.d* 出力形式とほぼ同じですが、DOLLARX*w.d* 出力形式では小数点とカンマの役割が逆になっています。この規則はヨーロッパの国で共通です。
- DOLLAR*w.d* 出力形式は COMMA*w.d* 出力形式とほぼ同じですが、COMMA*w.d* 出力形式は先頭のドル記号を出力しません。

例

次の例は、DOLLAR*w.d* 出力形式を示しています。

ステートメント	結果
select put(1254.71,dollar10.2);	$1,254.71

関連項目

出力形式:

- “COMMAw.d 出力形式” (127 ページ)
- “DOLLARXw.d 出力形式” (143 ページ)

DOLLARXw.d 出力形式

先頭にドル($)記号を付け、3 桁ごとにピリオドを入れ、小数部分との区切りにはカンマを使用して数値を出力します。

カテゴリ: 数値

配置: 右

操作: DECIMALCONV=システムオプションを STDIEEE に設定した場合、この出力形式を使用して書き出される出力が前のリリースと多少異なる可能性があります。詳細については、“DECIMALCONV= システムオプション” (*SAS システムオプション: リファレンス*)を参照してください。

構文

DOLLARX*w*.[*d*]

必須引数

w
出力フィールドの幅を指定します。

デフォルト 6

範囲 2-32

オプション引数

d
数値の小数点以下の桁数を指定します。

デフォルト 0

範囲	0-31
要件	*w* より小さい数にします。

詳細

DOLLARX*w.d* 出力形式は、先頭にドル($)記号を付け、3 桁ごとにカンマを入れ、小数部分との区切りにはピリオドを使用して数値を出力します。

比較

- DOLLARX*w.d* 出力形式は DOLLAR*w.d* 出力形式とほぼ同じですが、DOLLARX*w.d* 出力形式では小数点とカンマの役割が逆になっています。この規則はヨーロッパの国で共通です。
- DOLLARX*w.d* 出力形式は COMMAX*w.d* 出力形式とほぼ同じですが、COMMA*w.d* 出力形式は先頭のドル記号を出力しません。

例

次の例は、DOLLARX*w.d* 出力形式を示しています。

ステートメント	結果
select put(1254.71,dollarx10.2);	$1.254,71

関連項目

出力形式:

- “COMMAXw.d 出力形式” (128 ページ)
- “DOLLARw.d 出力形式” (141 ページ)

DOWNAMEw. 出力形式

SAS 日付値から曜日名を書き出します。

カテゴリ:	日付と時間
配置:	右

構文

DOWNAME*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 9

範囲 1-32

ヒント *w* を省略すると、 曜日名全体が出力されます。

詳細

必要に応じて、SAS により出力形式の幅に合うように曜日名が切り捨てられます。たとえば、出力形式 DOWNAME2.を指定すると、曜日名の最初の 2 文字が出力されます。

例

例の表では、19431 を入力値として使用します。この値は 2013 年 3 月 14 日に相当する SAS 日付値です。

ステートメント	結果
select put(19431,downame.);	Thursday

関連項目

出力形式:

- “WEEKDAYw. 出力形式” (216 ページ)

DTDATEw. 出力形式

入力として SAS 日時値が必要です。SAS 日付値を *ddmmmyy* または *ddmmmyyyy* 形式で書き出します。

カテゴリ: 日付と時間

配置: 右

構文

DTDATE*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 7

範囲 5-9

ヒント 4 桁の年を出力する場合は幅に 9 を指定します。

詳細

DTDATE*w*.出力形式は、SAS 日付値を *ddmmmyy* または *ddmmmyyyy* 形式で書き出します。構文の説明は次のとおりです。

dd
月の日を表す整数です。

mmm
月名の最初の 3 文字になります。

yy または *yyyy*
年を表す 2 桁または 4 桁の整数です。

比較

DTDATE*w*.出力形式は、DATE*w*.出力形式と同じ種類の出力を作成します。相違点は、DTDATE*w*.出力形式には SAS 日時値が必要なことです。

例

例の表では、2017 年 10 月 9 日の午前 11:25:25 に相当する SAS 日時値である 1823167525 の入力値を使用し、DTDATE*w*.出力形式に対応した 2 桁の年と 4 桁の年を出力します。

ステートメント	結果
select put(1823167525,dtdate.);	09OCT17
select put(1823167525,dtdate9.);	09OCT2017

関連項目

出力形式:

- “DATEw. 出力形式” (135 ページ)

DTMONYYw. 出力形式

SAS 日時値の日付部分を月と年として *mmmyy* または *mmmyyyy* 形式で書き出します。

カテゴリ: 日付と時間

配置: 右

構文

DTMONYY*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 5

範囲 5-7

詳細

DTMONYY*w*.出力形式は、SAS 日時値を *mmmyy* または *mmmyyyy* 形式で書き出します。構文の説明は次のとおりです。

mmm
月名の最初の 3 文字になります。

yy または *yyyy*
年を表す 2 桁または 4 桁の整数です。

比較

DTMONYY*w.* 出力形式と MONYY*w.*出力形式は、どちらも日付値を書き出すという点でほぼ同じです。相違点は、DTMONYY*w.*が入力として SAS 日時値を必要とし、MONYY*w.*は SAS 日付値が必要であることです。

例

例の表では 1823167525 を入力値として使用します。この値は 2017 年 10 月 09 日、午前 11:25:25 に相当する SAS 日時値です。

ステートメント	結果
select put(1823167525,dtmonyy.);	2017:4
select put(1823167525,dtmonyy5.);	OCT17
select put(1823167525,dtmonyy6.);	OCT17
select put(1823167525,dtmonyy7.);	OCT2017

関連項目

出力形式:

- “DATETIMEw.d 出力形式” (133 ページ)
- “MONYYw. 出力形式” (190 ページ)

DTWKDATXw. 出力形式

SAS 日時値の日付部分を曜日と日付として *day-of-week*、*dd month-name yy* (または *yyyy*)形式で書き出します。

カテゴリ: 日付と時間

配置: 右

構文

DTWKDATX*w.*

必須引数

w
出力フィールドの幅を指定します。

デフォルト 29

範囲 3-37

詳細

DTWKDATX*w*.出力形式は、SAS 日付値を *day-of-week*、*dd*、*month-name*、*yy* または *yyyy* 形式で書き出します。構文の説明は次のとおりです。

day-of-week
曜日名の最初の 3 文字か、曜日名全体です。

dd
月の日を表す整数です。

month-name
月名の最初の 3 文字か、月名全体です。

yy または *yyyy*
年を表す 2 桁または 4 桁の整数です。

比較

DTWKDATX*w*. 出力形式と WEEKDATX*w*. 出力形式は、どちらも日付値を書き出すという点でほぼ同じです。相違点は、DTWKDATX*w*.が入力として SAS 日時値を必要とし、WEEKDATX*w*.は SAS 日付値が必要であることです。

例

例の表では 1823167525 を入力値として使用します。この値は 2017 年 10 月 09 日、午前 11:25:25 に相当する SAS 日時値です。

ステートメント	結果
select put(1823167525,dtwkdatx.);	Monday, 9 October 2017
select put(1823167525,dtwkdatx3.);	Mon
select put(1823167525,dtwkdatx8.);	Mon
select put(1823167525,dtwkdatx25.);	Monday, 9 Oct 2017

関連項目

出力形式:

- “DATETIMEw.d 出力形式” (133 ページ)
- “WEEKDATXw. 出力形式” (214 ページ)

DTYEARw. 出力形式

SAS 日時値の日付部分を年として *yy* または *yyyy* 形式で書き出します。

カテゴリ: 日付と時間

配置: 右

構文

DTYEAR*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 4

範囲 2-4

比較

DTYEAR*w*.出力形式と YEAR*w*.出力形式は、どちらも日付値を書き出すという点でほぼ同じです。相違点は、DTYEAR*w*.が入力として SAS 日時値を必要とし、YEAR*w*.は SAS 日付値が必要であることです。

例

例の表では 1823167525 を入力値として使用します。この値は 2017 年 10 月 09 日、午前 11:25:25 に相当する SAS 日時値です。

ステートメント	結果
select put(1823167525,dtyear.);	2017

ステートメント	結果
select put(1823167525,dtyear2.);	17
select put(1823167525,dtyear3.);	17
select put(1823167525,dtyear4.);	2017

関連項目

出力形式:

- “DATETIMEw.d 出力形式” (133 ページ)
- “YEARw. 出力形式” (222 ページ)

DTYYQCw. 出力形式

SAS 日時値から、年と四半期をコロン(:)で区切って書き出します。

カテゴリ: 日付と時間

配置: 右

構文

DTYYQC*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 4

範囲 4-6

詳細

DTYYQC*w*.出力形式は、SAS 日時値を *yy* または *yyyy* 形式の年にコロン(:)と四半期を示す数値を続けて書き出します。

例

例の表では 1823167525 を入力値として使用します。この値は 2017 年 10 月 09 日、午前 11:25:25 に相当する SAS 日時値です。

ステートメント	結果
	----+----1
select put(1823167525,dtyyqc.);	17:4
select put(1823167525,dtyyqc4.);	17:4
select put(1823167525,dtyyqc5.);	17:4
select put(1823167525,dtyyqc6.);	2017:4

関連項目

出力形式:

- “DATETIMEw.d 出力形式” (133 ページ)

Ew. 出力形式

数値を指数表記で書き出します。

カテゴリ: 数値

配置: 右

操作: DECIMALCONV=システムオプションを STDIEEE に設定した場合、この出力形式を使用して書き出される出力が前のリリースと多少異なる可能性があります。詳細については、“DECIMALCONV= システムオプション” (*SAS システムオプション: リファレンス*)を参照してください。

構文

E*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 12

範囲 7-32

詳細

SAS は、結果の最初のカラムをマイナス記号を書き込む場所として確保します。

例

次の例は、*ew.*出力形式を示しています。

ステートメント	結果
select put(1257,e10.);	1.257E+03
select put(-1257,e10.);	-1.257E+03

E8601DAw. 出力形式

ISO 8601 規格の拡張表記 *yyyy-mm-dd* を使用して日付値を書き出します。

カテゴリ: 日付と時間
ISO 8601

配置: 左

別名: IS8601DA*w.*

制限事項: UTC タイムゾーンオフセット値はサポートされていません。

サポート: ISO 8601 Element 5.2.1.1, complete representation

構文

E8601DA*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 10

要件 10 である必要があります。

詳細

E8601DA 出力形式は、ISO 8601 規格の拡張表記 *yyyy-mm-dd* を使用して日付を書き出します。

yyyy
4 桁の年です。

mm
01 から 12 までの 2 桁の月です(ゼロ埋め込み)。

dd
01 から 31 までの 2 桁の日です(ゼロ埋め込み)。

例

次の例は、E8601DA*w.*出力形式を示しています。次の例では、入力値として 20999 を使用します。20999 は、2017 年 6 月 29 日に対応する SAS 日付値です。

ステートメント	結果
select put(20999,e8601da.);	2017-06-29

関連項目

“ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の処理” (*SAS 出力形式と入力形式: リファレンス*)

E8601DNw. 出力形式

ISO 8601 規格の拡張表記 *yyyy-mm-dd* を使用して、SAS 日時値から日付を書き出します。

カテゴリ: 日付と時間
ISO 8601

配置: 左

別名: IS8601DN

制限事項: UTC タイムゾーンオフセット値はサポートされていません。

サポート: ISO 8601 Element 5.2.1.1, complete representation

構文

E8601DN*w*.

必須引数

w
入力幅を指定します。

デフォルト 10

要件 入力幅は 10 である必要があります。

詳細

E8601DN 出力形式は、ISO 8601 規格の拡張日付表記 *yyyy-mm-dd* を使用して日付を書き出します。

yyyy
4 桁の年です。

mm
01 から 12 までの 2 桁の月です(ゼロ埋め込み)。

dd
01 から 31 までの 2 桁の日です(ゼロ埋め込み)。

例

次の例は、E8601DN*w.*出力形式を示しています。次の例では、入力値として 1793308532 を使用します。1793308532 は、2016 年 10 月 28 日に対応する SAS 日付値です。

ステートメント	結果
select put(1793308532,e8601dn.);	2016-10-28

関連項目

“ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の処理” (*SAS 出力形式と入力形式: リファレンス*)

E8601DTw.d 出力形式

ISO 8601 規格の拡張表記 *yyyy-mm-dd*T*hh:mm:ss.ffffff* を使用して日時値を書き出します。

カテゴリ: 日付と時間
ISO 8601

配置: 左

別名: IS8601DT*w.d*

制限事項: UTC タイムゾーンオフセット値はサポートされていません。

サポート: ISO 8601 Element 5.4.1, complete representation

構文

E8601DT*w.d*

必須引数

w
入力幅を指定します。

デフォルト 19

範囲 16-26

d
秒値の小数点以下の桁数を指定します。この引数はオプションです。

デフォルト 0

範囲 0-6

詳細

E8601DT 出力形式は、ISO 8601 規格の拡張日付表記 *yyyy-mm-dd*T*hh:mm:ss.ffffff* を使用して日時値を書き出します。

yyyy
4 桁の年です。

mm
01 から 12 までの 2 桁の月です(ゼロ埋め込み)。

dd
01 から 31 までの 2 桁の日です(ゼロ埋め込み)。

hh
00 から 23 までの 2 桁の時間です(ゼロ埋め込み)。

mm
00 から 59 までの 2 桁の分です(ゼロ埋め込み)。

ss
00 から 59 までの 2 桁の秒です(ゼロ埋め込み)。

ffffff
6 桁までの有効桁数を持つ、オプションの秒の分数です。各桁は 0 から 9 までです。

注: 幅を 16 に指定すると、SAS は秒の値を 0 と見なして出力から省略します。

例

次の例は、E8601DT*w.d* 出力形式を示しています。次の例では、入力値として 1793308532 を使用します。1793308532 は、2016 年 10 月 28 日の午後 9:15:32 に対応する SAS 日時値です。

ステートメント	結果
select put(1793308532,e8601dt.);	2016-10-28T21:15:32

関連項目

"ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の処理" (*SAS 出力形式と入力形式: リファレンス*)

E8601DZw. 出力形式

ISO 8601 規格の日時とタイムゾーンの拡張表記 *yyyy-mm-dd*T*hh:mm:ss*+00:00 を使用して、基準子午線の協定世界時(UTC)の日時値を書き出します。

カテゴリ: 日付と時間
ISO 8601

配置: 左

別名: IS8601DZ*w.*

サポート: ISO 8601 Element 5.4.1, complete representation

構文

E8601DZ*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 26

範囲 20-35

詳細

UTC 値は、イギリスのグリニッジの基準子午線に基づく時間とタイムゾーンを示します。E8601DZ 出力形式は、SAS 日時値を次の ISO 8601 規格の拡張日時表記のいずれかを使用して書き出します。

- *yyyy-mm-dd*T*hh:mm:ss*+00:00

  注: この形式は、*w* の長さがこのタイムゾーン表記に十分な場合に使用します。

- *yyyy-mm-dd*T*hh:mm:ss*Z

  注: この形式は、*w* の長さが+00:00 タイムゾーン表記に十分でない場合に使用します。

yyyy
4 桁の年です。

mm
01 から 12 までの 2 桁の月です(ゼロ埋め込み)。

dd
01 から 31 までの 2 桁の日です(ゼロ埋め込み)。

hh
00 から 24 までの 2 桁の時間です(ゼロ埋め込み)。

mm
00 から 59 までの 2 桁の分です(ゼロ埋め込み)。

ss
00 から 59 までの 2 桁の秒です(ゼロ埋め込み)。

+00:00
時間が基準子午線(イギリスのグリニッジ)に対応した時間であることを示します。

タイムゾーンオフセットが指定された ISO 8601 規格の時間値または日時値は、オフセットに示された時間数と分数によって調整され、基準子午線(イギリスのグ

リニッジ)の時間または日時として処理されます。E8601DZ 出力形式は、常に +00:00 の基準子午線オフセット値を使用して日時値を書き出します。+00:00 以外の UTC オフセットを使用して日時を書き出すには、“E8601LZw. 出力形式” (*SAS 出力形式と入力形式: リファレンス*)を参照してください。

制約: 短い形式+00 はサポートされていません。

Z

時間が基準子午線(イギリスのグリニッジ)に対応した時間、つまり+00:00 UTC 時間であることを示します。出力形式の幅が+00:00 表記に十分でない場合は、Z を使用します。

例

次の例は、E8601DZ*w.*出力形式を示しています。次の例では、入力値として 1793308532 を使用します。1793308532 は、2016-10-28T21:15:32+00:00 に対応する SAS 日付値です。

ステートメント	結果
select put(1793308532,e8601dz.);	2016-10-28T21:15:32+00:00

関連項目

“ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の処理” (*SAS 出力形式と入力形式: リファレンス*)

E8601LZw. 出力形式

ISO 8601 規格の拡張時間表記 *hh:mm:ss+|–hh:mm* を使用して、ローカル SAS セッションに対応した協定世界時(UTC)のオフセットを追加することにより、時間値をローカル時間として書き出します。

カテゴリ: 日付と時間
ISO 8601

配置: 左

別名: IS8601LZ*w.*

サポート: ISO 8601 Element 5.3.1.1, complete representation

構文

E8601LZ*w.*

必須引数

w
出力フィールドの幅を指定します。

デフォルト 14

範囲 9-20

詳細

E8601LZ 出力形式は、次の ISO 8601 規格の拡張時間表記のいずれかを使用して時間値を調整せずに書き出し、ローカル SAS セッションに対応した UTC タイムゾーンオフセットを追加します。

- *hh:mm:ss*+|–*hh:mm*

  注: この形式は、*w* の長さがこの時間表記に十分である場合に使用します。

- *hh:mm:ss*Z

  注: この形式は、*w* の長さが+|– *hh:mm* タイムゾーン表記に十分でない場合に使用します。

hh
00 から 23 までの 2 桁の時間です(ゼロ埋め込み)。

mm
00 から 59 までの 2 桁の分です(ゼロ埋め込み)。

ss
00 から 59 までの 2 桁の秒です(ゼロ埋め込み)。

+|–*hh:mm*
基準子午線時間からの時間と分の符号付きオフセットです。オフセットは+|–*hh:mm* (つまり+または–に続く 5 文字)である必要があります。

+は基準子午線の東部タイムゾーンに、-は基準子午線の西部タイムゾーンにそれぞれ使用します。たとえば、+02:00 は基準子午線の東部での 2 時間差を表し、–06:00 は基準子午線の西部での 6 時間差を表します。

制約: 短い形式+|–*hh* はサポートされていません。

Z
基準子午線(イギリスのグリニッジ)に対応した時間、つまり+00:00 UTC 時間であることを示します。

SAS では、時間値を *hh:mm.ffffff* 形式を使用して書き出し、ローカル SAS セッションに対応した基準子午線とのタイムゾーンオフセットのタイムゾーンインジケーター+|–*hh:mm* または Z を追加します。Z タイムゾーンインジケーターは、出力形式の長さの値が 14 未満の場合に使用されます。

両方のゾーンインジケーターを使用して同じ時間を書き出す場合、UTC に基づく 2 つの異なる時間を示します。たとえば、ローカル SAS セッションでアメリカの東部

標準時が使用され、その時間値が 45824 である場合、12:43:44-04:00 または 12:43:44Z が書き出されます。時間 12:43:44–04:00 は、基準子午線の時間の 16:43:44+00:00 です。Z は、その時間が基準子午線の時間である、つまり 12:43:44+00:00 であることを示します。

SAS では UTC 時間を E8601TZ 入力形式を使用して読み込む際に、オフセット調整後の時間が 24 時間を超えていたり 00 時間未満である場合、オフセット調整後の時間が 00:00:00 と 24:00:00 の間になるように値を調整します。E8601LZ 出力形式をこの時間範囲外の時間に適用する場合、時間はアスタリスクを使用してフォーマットされ、その値が範囲外であることが示されます。

例

次の例は、E8601LZ*w.*出力形式を示しています。この例では、20999 を入力値として使用します。これは、ローカル時間値 05:49:59-04:00 です。

ステートメント	結果
select put(20999,e8601lz.);	05:49:59-04:00

関連項目

"ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の処理" (*SAS 出力形式と入力形式: リファレンス*)

E8601TMw.d 出力形式

ISO 8601 規格の拡張表記 *hh:mm:ss.ffffff* を使用して時間値を書き出します。

カテゴリ: 日付と時間
ISO 8601

配置: 左

別名: IS8601TM*w.d*

制限事項: UTC タイムゾーンオフセット値はサポートされていません。

サポート: ISO 8601 要素 5.3.1.1、完全な表現、および 5.3.1.3、小数の表現

構文

E8601TM*w.d*

必須引数

w
出力フィールドの幅を指定します。

デフォルト 8

範囲 8-15

d
秒値の小数点以下の桁数を指定します。この引数はオプションです。

デフォルト 0

範囲 0-6

詳細

E8601TM 出力形式は、SAS 時間値を ISO 8601 規格の拡張時間表記 *hh:mm:ss.ffffff* を使用して書き出します。

hh
00 から 23 までの 2 桁の時間です(ゼロ埋め込み)。

mm
00 から 59 までの 2 桁の分です(ゼロ埋め込み)。

ss
00 から 59 までの 2 桁の秒です(ゼロ埋め込み)。

.ffffff
6 桁までの有効桁数を持つ、オプションの秒の分数です。各桁は 0 から 9 までです。

例

次の例は、E8601TM*w.d* 出力形式を示しています。この例では、05:49:59 の時間値に対応する 20999 の入力値を使用しています。

ステートメント	結果
select put(20999,e8601tm.);	05:49:59

関連項目

“ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の処理” (*SAS 出力形式と入力形式: リファレンス*)

E8601TZw.d 出力形式

時間値を協定世界時(UTC)に調整し、時間値を ISO 8601 規格の拡張表記 *hh*:*mm*:*ss.<fff>*+|–*hh*:*mm* を使用して書き出します。

カテゴリ: 日付と時間
ISO 8601

配置: 左

別名: IS8601TZ*w.d*

サポート: ISO 8601 Element 5.3.1.1, complete representation

構文

E8601TZ*w.d*

必須引数

w
出力フィールドの幅を指定します。

デフォルト 14

範囲 9-20

d
秒値の小数点以下の桁数を指定します。この引数はオプションです。

デフォルト 0

範囲 0-6

詳細

UTC 時間値は、イギリスのグリニッジの基準子午線に基づく時間とタイムゾーンを示します。E8601TZ 出力形式は、時間値を次の ISO 8601 規格の拡張時間表記のいずれかを使用して書き出します。

- *hh:mm:ss<.fff>*+|–*hh:mm*

  注: この形式は、*w* の長さがこのタイムゾーン表記に十分な場合に使用します。

- *hh:mm:ss*Z

注: この形式は、*w* の長さが+|–*hh:mm* タイムゾーン表記に十分でない場合に使用します。

hh
00 から 23 までの 2 桁の時間です(ゼロ埋め込み)。

mm
00 から 59 までの 2 桁の分です(ゼロ埋め込み)。

ss
00 から 59 までの 2 桁の秒です(ゼロ埋め込み)。

fff
オプションの秒の端数です。

+|–*hh:mm*
基準子午線時間からの時間と分の符号付きオフセットです。オフセットは+|–*hh:mm* (つまり、+または–を伴う 5 文字)である必要があります。

制約: 短い形式+|–*hh* はサポートされていません。

+は基準子午線の東部タイムゾーンに、-は基準子午線の西部タイムゾーンにそれぞれ使用します。たとえば、+02:00 は基準子午線の東部での 2 時間差を表し、–06:00 は基準子午線の西部での 6 時間差を表します。

Z
基準子午線(イギリスのグリニッジ)に対応した時間、つまり+00:00 UTC 時間であることを示します。

SAS では UTC 時間を E8601TZ 入力形式を使用して読み込む際に、オフセット調整後の時間が 24 時間を超えていたり 00 時間未満である場合、オフセット調整後の時間が 00:00:00 と 24:00:00 の間になるように値を調整します。E8601TZ 出力形式をこの時間範囲外の時間に適用する場合、時間はアスタリスクを使用してフォーマットされ、その値が範囲外であることが示されます。

比較

00:00:00 と 24:00:00 の間の時間値に対して、B8601TZ 出力形式は時間値を基準子午線の時間に調整し、調整後の値を国際規格の拡張時間表記で書き出します。B8601LZ 出力形式は時間を調整せず、ローカル SAS セッションの UTC タイムゾーンオフセットを使用して時間値を国際規格の拡張時間表記で書き出します。

例

次の例は、E8601TZ*w.d* 出力形式を示しています。この例では、05:49:59+00:00 の時間値に対応する 20999 の入力値を使用しています。

ステートメント	結果
select put(20999,e8601tz.);	05:49:59+00:00

関連項目

"ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の処理" (*SAS 出力形式と入力形式: リファレンス*)

EUROw.d 出力形式

先頭にユーロ記号(E)を付け、3 桁ごとにカンマを入れ、小数部分との区切りにはピリオドを使用して数値を出力します。

カテゴリ: 数値

配置: 右

構文

EURO*w.d*

必須引数

w
出力フィールドの幅を指定します。

デフォルト 6

範囲 1-32

ヒント ユーロ記号を出力の一部にする場合は、必ず適切な幅を選択してください。

d
数値の小数点以下の桁数を指定します。

デフォルト 0

範囲 0-31

要件 *w* より小さい数にします。

比較

- EURO*w.d* 出力形式は EUROX*w.d* 出力形式とほぼ同じですが、EUROX*w.d* 出力形式では小数点とカンマの役割が逆になっています。この規則はヨーロッパの国で共通です。
- EURO*w.d* 出力形式は DOLLAR*w.d* 出力形式とほぼ同じですが、DOLLAR*w.d* 出力形式がユーロ記号のかわりにドル記号を先頭に書き込む点が異なります。

注: EUROX*w.d* 出力形式は、ユーロ文字(U+20AC)を使用します。DBCS バージョンの SAS と、ユーロ文字をサポートしていないエンコーディングを使用すると、エラーが発生します。このエラーを防ぐには、セッションエンコーディングをユーロ文字をサポートするエンコーディングに変更します。

例

これらの例では、金額の値として 1254.71 を使用しています。

ステートメント	結果
select put(1254.71,euro10.2);	E1,254.71
select put(1254.71,euro5.);	1,255
select put(1254.71,euro9.2);	E1,254.71
select put(1254.71,euro15.3);	E1,254.710

関連項目

出力形式:

- “EUROXw.d 出力形式” (166 ページ)

EUROXw.d 出力形式

先頭にユーロ記号(E)を付け、3 桁ごとにピリオドを入れ、小数部分との区切りにはカンマを使用して数値を出力します。

カテゴリ: 数値

配置: 右

構文

EUROX*w.d*

必須引数

w
出力フィールドの幅を指定します。

デフォルト 6

範囲 1-32

ヒント ユーロ記号を出力の一部にする場合は、必ず適切な幅を選択してください。

d
数値の小数点以下の桁数を指定します。

デフォルト 0

範囲 0-31

要件 *w* より小さい数にします。

比較

- EUROX*w.d* 出力形式は EURO*w.d* 出力形式とほぼ同じですが、EURO*w.d* 出力形式ではカンマと小数点の役割が逆になっています。この規則は英語圏の国で共通です。
- EUROX*w.d* 出力形式は DOLLARX*w.d* 出力形式とほぼ同じですが、DOLLARX*w.d* 出力形式がユーロ記号のかわりにドル記号を先頭に書き込む点が異なります。

注: EUROX*w.d* 出力形式は、ユーロ文字(U+20AC)を使用します。DBCS バージョンの SAS と、ユーロ文字をサポートしていないエンコーディングを使用すると、エラーが発生します。このエラーを防ぐには、セッションエンコーディングをユーロ文字をサポートするエンコーディングに変更します。

例

これらの例では、金額の値として 1254.71 を使用しています。

ステートメント	結果
select put(1254.71,eurox10.2);	E1.254,71
select put(1254.71,eurox5.);	1.255
select put(1254.71,eurox9.2);	E1.254,71
select put(1254.71,eurox15.3);	E1.254,710

関連項目

出力形式:

- “DOLLARXw.d 出力形式” (143 ページ)
- “EUROw.d 出力形式” (165 ページ)

FLOATw.d 出力形式

数値に 10 の *d* 乗を掛けた、単精度のネイティブ浮動小数点値を生成します。

カテゴリ: 数値

配置: 左

構文

FLOAT*w*.[*d*]

必須引数

w
出力フィールドの幅を指定します。

要件 幅は 4 である必要があります。

オプション引数

d
値を乗算する 10 のべき乗を指定します。

デフォルト 0

範囲 0-31

詳細

FLOAT4.によって書き出される値は通常、使用している動作環境で実行される、単精度値を扱う他の外部プログラムによって読み込まれるための値です。フォーマットされる値が欠損値、あるいは単精度のネイティブ浮動小数点の範囲外である場合、ゼロの単精度の値が生成されます。

例

次の例は、FLOAT*w.d* 出力形式を示しています。結果は、IEEE 形式で保存したバイナリ値を 16 進表現で表したものです。

ステートメント	結果
select put(1,float4.);	0000803F

FRACTw. 出力形式

数値を分数に変換します。

カテゴリ: 数値

配置: 右

構文

FRACT*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 10

範囲 4-32

詳細

1 を 3 で割ると、値は 0.33333333 となります。この値を 1/3 として書き出すには、FRACT*w*.出力形式を使用します。FRACT*w*. は、分数を約分して書き出します。50/100 は 1/2 となります。

例

次の例は、FRACT*w*.出力形式を示しています。

ステートメント	結果
select put(0.6666666667,fract8.);	2/3
select put(0.2784,fract8.);	174/625

HEXw. 出力形式

バイナリ実数(浮動小数点)値を 16 進表現に変換します。

カテゴリ: 数値

配置: 左

構文

HEX*w*.

必須引数

w

出力フィールドの幅を指定します。

デフォルト 8

範囲 1-16

ヒント *w* < 16 の場合、HEX*w*.出力形式はバイナリ実数を固定小数点数に変換してから 16 進表現に変換します。また、負数を 2 つの補数にして右揃えで書き出します。*w* が 16 の場合、HEX*w*.は浮動小数点数値を 16 進表現で表示します。

詳細

どの動作環境でも、HEX*w*.によって書き出される最下位バイトは右端のバイトになります。一部の動作環境では、最下位桁が最初のバイトである整数が保存されます。HEX*w*.出力形式は、バイト単位の重要度に関係なく、どの動作環境でも一貫した結果を生成します。

注: 動作環境によって浮動小数点数の保存方法は異なります。しかし、HEX16.出力形式は、使用されている動作環境での保存方法と同じように、16 進表現の浮動小数点数を書き出します。

比較

HEX*w*. 数値出力形式と$HEX*w*. 文字出力形式は、ともに値を 16 進表現に変換します。

例

次の例は、HEX*w*.出力形式を示しています。

ステートメント	結果
select put(35.4, hex8.);	00000023
select put(88, hex8.);	00000058
select put(2.33, hex8.);	00000002
select put(-150, hex8.);	FFFFFF6A

関連項目

出力形式:

- “$HEXw. 出力形式” (100 ページ)

HHMMw.d 出力形式

SAS 時間値を時間および分として *hh:mm* 形式で書き出します。

カテゴリ: 日付と時間

配置: 右

構文

HHMM*w*.[*d*]

必須引数

w
出力フィールドの幅を指定します。

デフォルト 5

範囲 2-20

オプション引数

d
分値の小数点以下の桁数を指定します。小数点以下の桁数は、分の端数を示します。

デフォルト 0

範囲 0-19

要件 *w* より小さい数にします。

詳細

HHMM*w.d* 出力形式は、SAS 日時値を *hh:mm* 形式で書き出します。構文の説明は次のとおりです。

hh
整数です。

mm
00 から 59 の範囲の時間(分)です。

時間と分は、SAS 時間値の秒値に基づき丸められます。

比較

HHMM*w.d* 出力形式は TIME*w.d* 出力形式とほぼ同じですが、HHMM*w.d* 出力形式は秒を出力しません。

HHMM*w.d* 出力形式形式と TIME*w.d* 出力形式は、1 桁の時間にはその先頭に空白を書き出します。TOD*w.d* 出力形式は、1 桁の時間にはその先頭にゼロを書き出します。

例

例の表では、46796 を入力値として使用します。この値は、午後 12:59:56 に相当する SAS 時間値です。

ステートメント	結果
select put(46796,hhmm.);	13:00

ステートメント	結果
select put(46796,hhmm8.2);	12:59.93

最初の例では、SAS 時間値の秒値に基づき、時間値が 4 秒切り上げられます。2 番目の例では、出力形式に小数点以下の桁数として 2 を追加指定することにより、56 秒が 1 分の 93%として表示されています。

関連項目

出力形式:

- “HOURw.d 出力形式” (173 ページ)
- “MMSSw.d 出力形式” (183 ページ)
- “TIMEw.d 出力形式” (204 ページ)
- “TODw.d 出力形式” (208 ページ)

関数:

- “HMS 関数” (565 ページ)
- “HOUR 関数” (570 ページ)
- “MINUTE 関数” (670 ページ)
- “SECOND 関数” (824 ページ)

HOURw.d 出力形式

SAS 時間値を時間および時間の小数として書き出します。

カテゴリ: 日付と時間

配置: 右

操作: DECIMALCONV=システムオプションを STDIEEE に設定した場合、この出力形式を使用して書き出される出力が前のリリースと多少異なる可能性があります。詳細については、“DECIMALCONV= システムオプション” (*SAS システムオプション: リファレンス*)を参照してください。

構文

HOUR*w*.[*d*]

必須引数

w
出力フィールドの幅を指定します。

デフォルト 2

範囲 2-20

オプション引数

d
時間値の小数点以下の桁数を指定します。時間の小数が出力されます。

範囲 0-19

要件 *w* より小さい数にします。

詳細

時間は、SAS 時間値の分値に基づき丸められます。

例

例の表では、41400 を入力値として使用します。この値は、午前 11:30 に相当する SAS 時間値です。

ステートメント	結果
select put(41400,hour4.1);	11.5

関連項目

出力形式:

- “HHMMw.d 出力形式” (171 ページ)
- “MMSSw.d 出力形式” (183 ページ)
- “TIMEw.d 出力形式” (204 ページ)
- “TODw.d 出力形式” (208 ページ)

関数:

- “HMS 関数” (565 ページ)

- “HOUR 関数” (570 ページ)
- “MINUTE 関数” (670 ページ)
- “SECOND 関数” (824 ページ)

IEEEw.d 出力形式

数値に 10 の *d* 乗を掛けて、IEEE 浮動小数点値を生成します。

カテゴリ: 数値

配置: 左

注意: **値が大きい浮動小数点数値や有効桁数が必要な浮動小数点数値は、IEEE 出力形式を使用して IBM メインフレームに書き出し、IEEE 入力形式で読み込みなおす場合、元の SAS 値とは異なる場合があります。**

構文

IEEE*w*.[*d*]

必須引数

w
出力フィールドの幅を指定します。

デフォルト 8

範囲 3-8

ヒント *w* が 8 の場合、IEEE の倍精度浮動小数点数が書き出されます。*w* が 5、6 または 7 の場合、相当するバイト数が切り捨てられたと仮定して、IEEE の倍精度浮動小数点数が書き出されます。*w* が 4 の場合、IEEE の単精度浮動小数点数が書き出されます。*w* が 3 の場合、1 バイトが切り捨てられたと仮定して、IEEE の単精度浮動小数点数が書き出されます。

オプション引数

d
10^d で数値を乗算するよう指定します。

デフォルト 0

範囲 0-10

詳細

この出力形式は、IEEE*w.d* が使用される浮動小数点数表現である動作環境で使用されます。また、IEEE*w.d* 出力形式を使用して、IEEE 浮動小数点数表現を使用する動作環境のプログラムによって使用されるファイルを作成することもできます。

通常プログラムは、単精度(4 バイト)または倍精度(8 バイト)で IEEE 値を生成します。プログラムは、出力ファイルのスペースを削減する場合にのみ切り捨てを実行します。機械語命令では、浮動小数点数が 4 バイトまたは 8 バイトのいずれかである必要があります。IEEE*w.d* 出力形式では、他の長さも処理できるため、スペースの削減のために切り捨てられたデータを含むファイルにデータを書き込めます。

例

次の例は、IEEE*w.d* 出力形式を示しています。

ステートメント	結果 [1]
select put(1,ieee4.);	3FF00000
select put(1,ieee5.);	3FF0000000

1 結果は、IEEE 形式で保存したバイナリ値を 16 進表現で表したものです。

JULIANw. 出力形式

SAS 日付値を *yyddd* または *yyyyddd* 形式のユリウス日付として書き出します。

カテゴリ: 日付と時間

配置: 左

構文

JULIAN*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 5

範囲 5-7

ヒント　*w* が 5 の場合、JULIAN*w*.出力形式では日付の年の部分を 2 桁で書き出します。*w* が 7 の場合、JULIAN*w*.出力形式では日付の年の部分を 4 桁で書き出します。

詳細

JULIAN*w*.出力形式では、SAS 日付値を *yyddd* または *yyyyddd* 形式で書き出します。構文の説明は次のとおりです。

yy または *yyyy*
年を表す 2 桁または 4 桁の整数です。

ddd
1 から 365 まで(うるう年の場合は 1 から 366 まで)の年の日数です。

例

例の表では、19431 を入力値として使用します。この値は 2013 年 3 月 14 日(年の 73 日目)に相当する SAS 日付値です。

ステートメント	結果
select put(19431,julian5.);	13073
select put(19431,julian7.);	2013073

関連項目

関数:

- “DATEJUL 関数” (440 ページ)

MDYAMPMw.d 出力形式

SAS 日時値を *mm/dd/yy<yy> hh:mm* AM|PM 形式で書き出します。年は 2 桁または 4 桁のいずれかです。

カテゴリ:　日付と時間

配置:　右

操作:　DECIMALCONV=システムオプションを STDIEEE に設定した場合、この出力形式を使用して書き出される出力が前のリリースと多少異なる可能性があります。詳細に

ついては、“DECIMALCONV= システムオプション” (*SAS システムオプション: リファレンス*)を参照してください。

注: デフォルトの時間は AM です。

構文

MDYAMPM*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 19

範囲 8-40

詳細

MDYAMPM*w.d* 出力形式は、SAS 日時値を次の形式で書き出します。

mm/*dd*/*yy*[yy] *hh*:*mm*[AM | PM]:

mm
月を表す 1 から 12 までの整数です。

dd
月の日を表す 1 から 31 の整数です。

yy または *yyyy*
年を表す 2 桁または 4 桁の整数です。

hh
時間を表す 00 から 23 までの整数です。

mm
分を表す 00 から 59 までの整数です。

AM | PM
00:01–12:00 正午(AM)または 12:01–12:00 深夜(PM)のいずれかです。デフォルトは AM です。

日時の区切り文字
SAS で日付要素と時間要素を区切るために使用するスラッシュ(/)、コロン(:)、空白文字などの特殊文字のいずれか 1 つです。

比較

MDYAMPM*w.*出力形式は、日時値を区切り文字付きの *mm/dd/yy<yy> hh:mm* AM | PM 形式で書き出します。日付と時間の間にスペースが挿入されます。

DATETIME*w.d* 出力形式は、日時値を区切り文字付きの *ddmmmyy<yy>: hh:mm:ss.ss* 形式で書き出します。

例

次の例では、19431 を入力値として使用します。この値は、2013 年 03 月 14 日に相当する SAS 日付値です。

ステートメント	結果
select put(19431,mmddyy2.);	03
select put(19431,mmddyy3.);	03
select put(19431,mmddyy4.);	0314
select put(19431,mmddyy5.);	03/14
select put(19431,mmddyy6.);	031413
select put(19431,mmddyy7.);	031413
select put(19431,mmddyy8.);	03/14/13
select put(19431,mmddyy10.);	03/14/2013

関連項目

出力形式:

- “DATETIMEw.d 出力形式” (133 ページ)

MMDDYYw. 出力形式

SAS 日付値を *mmdd<yy>yy* または *mm/dd/<yy>yy* 形式で書き出します。スラッシュが区切り文字として使用され、年は 2 桁または 4 桁の数字で書き出されます。

カテゴリ: 日付と時間

配置: 右

構文

MMDDYY*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 8

範囲 2-10

操作 *w* の値が 2 から 5 までの場合、月と日が可能な範囲で表示されます。*w* が 7 の場合、年を 2 桁で表し、スラッシュを使わずに出力します。

詳細

MMDDYY*w*.出力形式は、SAS 日付値を *mmdd<yy>yy* または *mm/dd/<yy>yy* 形式で書き出します。構文の説明は次のとおりです。

mm
月を表す整数です。

/
区切り文字です。

dd
月の日を表す整数です。

<yy>yy
年を表す 2 桁または 4 桁の整数です。

例

次の例では、19431 を入力値として使用します。この値は、2013 年 3 月 14 日に相当する SAS 日付値です。

ステートメント	結果
select put(19431,mmddyyc5.);	03:14
select put(19431,mmddyyd8.);	03-14-13
select put(19431,mmddyyn9.);	03132013
select put(19431,mmddyyp10.);	03.14.2013

関連項目

出力形式:

- “DATEw. 出力形式” (135 ページ)
- “DDMMYYw. 出力形式” (138 ページ)
- “MMDDYYxw. 出力形式” (181 ページ)
- “YYMMDDw. 出力形式” (227 ページ)

関数:

- “DAY 関数” (443 ページ)
- “MDY 関数” (664 ページ)
- “MONTH 関数” (676 ページ)
- “YEAR 関数” (887 ページ)

MMDDYYxw. 出力形式

SAS 日付値を *mmdd<yy>yy* または *mm-dd-<yy>yy* 形式で書き出します。ここで、出力形式名の *x* は、月、日、年を区切る特殊文字を表す文字です。特殊文字は、ハイフン(-)、ピリオド(.)、空白文字、スラッシュ(/)、コロン(:)または区切り文字なしになります。年は 2 桁または 4 桁のいずれかになります。

カテゴリ: 日付と時間

配置: 右

構文

MMDDYY*xw*.

必須引数

x
区切り文字を示します。または、区切り文字を日、月、年の間に挿入しないことを示します。有効な値は次のとおりです。

B
空白で区切ります。

C
コロンで区切ります。

D
ハイフンで区切ります。

N
区切り文字なしを表します。

P
ピリオドで区切ります。

S
スラッシュで区切ります。

w
出力フィールドの幅を指定します。

デフォルト　8

範囲　2-10

操作　*w* の値が 2 から 5 までの場合、月と日が可能な範囲で表示されます。*w* が 7 の場合、日付の年は 2 桁で、区切り文字を使用せずに表示されます。

x の値が N の場合、幅の範囲は 2–8 に変化します。

詳細

MMDDYY*xw*.出力形式は、SAS 日付値を *mmdd<yy>yy* または *mm*X*dd*X*<yy>yy* 形式で書き出します。構文の説明は次のとおりです。

mm
月を表す整数です。

X
指定された区切り文字です。

dd
月の日を表す整数です。

<yy>yy
年を表す 2 桁または 4 桁の整数です。

例

次の例では、19431 を入力値として使用します。この値は、2013 年 3 月 14 日に相当する SAS 日付値です。

ステートメント	結果
select put(19431,mmddyyc5.);	03:14
select put(19431,mmddyyd8.);	03-14-13

ステートメント	結果
select put(19431,mmddyyn9.);	03132013
select put(19431,mmddyyp10.);	03.14.2013

関連項目

出力形式:

- “DATEw. 出力形式” (135 ページ)
- “DDMMYYxw. 出力形式” (139 ページ)
- “MMDDYYxw. 出力形式” (181 ページ)
- “YYMMDDxw. 出力形式” (229 ページ)

関数:

- “DAY 関数” (443 ページ)
- “MDY 関数” (664 ページ)
- “MONTH 関数” (676 ページ)
- “YEAR 関数” (887 ページ)

MMSSw.d 出力形式

SAS 時間値を午前 0 時から経過した分数および秒数として書き出します。

カテゴリ: 日付と時間

配置: 右

操作: DECIMALCONV=システムオプションを STDIEEE に設定した場合、この出力形式を使用して書き出される出力が前のリリースと多少異なる可能性があります。詳細については、“DECIMALCONV= システムオプション” (*SAS システムオプション: リファレンス*)を参照してください。

構文

MMSS*w*.[*d*]

必須引数

w
出力フィールドの幅を指定します。

デフォルト 5

範囲 2-20

ヒント 分数と秒数を表す値を書き出すには、*w* に少なくとも 5 を指定します。

オプション引数

d
秒値の小数点以下の桁数を指定します。このため、SAS 時間値には秒の端数が含まれます。

範囲 0-19

制限事項 *w* より小さい数にします。

例

例の表では、SAS 入力値として 4530 を使用します。

ステートメント	結果
select put(4530,mmss.);	75:30

関連項目

出力形式:

- “HHMMw.d 出力形式” (171 ページ)
- “TIMEw.d 出力形式” (204 ページ)

関数:

- “HMS 関数” (565 ページ)
- “MINUTE 関数” (670 ページ)
- “SECOND 関数” (824 ページ)

MMYYw. 出力形式

SAS 日付値を *mm*M<*yy*>*yy* 形式で書き出します。M は区切り文字を表し、年は 2 桁または 4 桁の数字で書き出されます。

カテゴリ: 日付と時間

配置: 右

構文

MMYY*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 7

範囲 5-32

操作 *w* の値が 5 または 6 の場合、日付の年は下 2 桁でのみ表示されます。*w* が 7 以上の場合、日付の年は 4 桁で表示されます。

詳細

MMYY*w*.出力形式は、SAS 日付値を *mm*M<*yy*>*yy* 形式で書き出します。構文の説明は次のとおりです。

mm
月を表す整数です。

M
区切り文字です。

<*yy*>*yy*
年を表す 2 桁または 4 桁の整数です。

例

次の例では、19431 を入力値として使用します。この値は、2013 年 03 月 14 日に相当する SAS 日付値です。

ステートメント	結果
select put(19431,mmyy5.);	03M13
select put(19431,mmyy6.);	03M13
select put(19431,mmyy7.);	03M2013
select put(19431,mmyy10.);	03M2013

関連項目

出力形式:

- “MMYYxw. 出力形式” (186 ページ)
- “YYMMw. 出力形式” (224 ページ)

MMYYxw. 出力形式

SAS 日付値を *mm<yy>yy* または *mm-<yy>yy* 形式で書き出します。出力形式名の *x* は、月と年を区切る特殊文字を表す文字で、ハイフン(-)、ピリオド(.)、空白文字、スラッシュ(/)、コロン(:)、または区切り文字なしになります。年は 2 桁または 4 桁のいずれかになります。

カテゴリ: 日付と時間

配置: 右

構文

MMYY*xw*.

必須引数

x
区切り文字を示します。または、区切り文字が月と年の間に挿入しないことを示します。有効な値は次のとおりです。

C
コロンで区切ります。

D
ハイフンで区切ります。

N
区切り文字なしを表します。

P
ピリオドで区切ります。

S
スラッシュで区切ります。

w
出力フィールドの幅を指定します。

デフォルト 7

範囲 5-32

操作 *x* を N に設定すると、区切り文字なしを指定します。この場合の幅範囲は 4–32 になり、デフォルトは 6 に変化します。

x の値が C、D、P または S で、*w* の値が 5 または 6 の場合、日付の年は下 2 桁のみ表示されます。*w* が 7 以上の場合、日付の年は 4 桁で表示されます。

x の値が *N* で、*w* の値が 4 または 5 の場合、日付の年は下 2 桁のみ表示されます。*x* の値が *N* で、*w* が 6 以上の場合、日付の年は 4 桁で表示されます。

詳細

MMYY*xw*.出力形式は、SAS 日付値を *mm<yy>yy* または *mm*X*<yy>yy* 形式で書き出します。構文の説明は次のとおりです。

mm
月を表す整数です。

X
指定された区切り文字です。

<yy>yy
年を表す 2 桁または 4 桁の整数です。

例

次の例では、19560 を入力値として使用します。この値は、2013 年 7 月 14 日に相当する SAS 日付値です。

ステートメント	結果
select put(19560,mmyyc5.);	07:13
select put(19560,mmyyd7.);	07-2013

ステートメント	結果
select put(19560,mmyyn4.);	0713
select put(19560,mmyyp8.);	07.2013
select put(19560,mmyys10.);	07/2013

関連項目

出力形式:

- “MMYYw. 出力形式” (185 ページ)
- “YYMMw. 出力形式” (224 ページ)

MONNAMEw. 出力形式

SAS 日付値を月の名前として書き出します。

カテゴリ: 日付と時間

配置: 右

構文

MONNAME*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 9

範囲 1-32

ヒント MONNAME3.を使用すると、月名の最初の 3 文字が出力されます。

詳細

必要に応じて、出力形式の幅に合うように月名が切り捨てられます。

例

例の表では、19431 を入力値として使用します。この値は 2014 年 3 月 14 日に相当する SAS 日付値です。

ステートメント	結果
select put(19431,monname1.);	M
select put(19431,monname3.);	Mar
select put(19431,monname5.);	March

関連項目

出力形式:

- “MONTHw. 出力形式” (189 ページ)

MONTHw. 出力形式

SAS 日付値を月として書き出します。

カテゴリ: 日付と時間

配置: 右

構文

MONTH*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 2

範囲 1-32

詳細

MONTH*w*.出力形式は、SAS 日付値から月(1 から 12 まで)を書き出します。

例

例の表では、19431 を入力値として使用します。この値は 2014 年 3 月 14 日に相当する SAS 日付値です。

ステートメント	結果
select put(19431,month.);	3

関連項目

出力形式:

- “MONNAMEw. 出力形式” (188 ページ)

MONYYw. 出力形式

SAS 日付値を月および年として *mmmyy* または *mmmyyyy* 形式で書き出します。

カテゴリ: 日付と時間

配置: 右

構文

MONYY*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 5

範囲 5-7

詳細

MONYY*w*.出力形式は、SAS 日付値を *mmmyy* または *mmmyyyy* 形式で書き出します。構文の説明は次のとおりです。

mmm
月名の最初の 3 文字になります。

yy または *yyyy*
年を表す 2 桁または 4 桁の整数です。

比較

MONYY*w*.出力形式と DTMONYY*w*.出力形式は、どちらも日付値を書き出すという点でほぼ同じです。相違点は、MONYY*w*.が 入力として SAS 日付値を必要とし、DTMONYY*w*.は日時値が必要であることです。

例

例の表では、19704 を入力値として使用します。この値は 2013 年 12 月 12 日に相当する SAS 日付値です。

ステートメント	結果
select put(19704,monyy5.);	DEC13
select put(19704,monyy7.);	DEC2013

関連項目

出力形式:

- “DTMONYYw. 出力形式” (147 ページ)
- “DDMMYYw. 出力形式” (138 ページ)
- “MMDDYYw. 出力形式” (179 ページ)
- “YYMMDDw. 出力形式” (227 ページ)

関数:

- “MONTH 関数” (676 ページ)
- “YEAR 関数” (887 ページ)

NEGPARENw.d 出力形式

負の数値をかっこで囲んで書き出します。

カテゴリ: 数値

配置: 右

操作: DECIMALCONV=システムオプションを STDIEEE に設定した場合、この出力形式を使用して書き出される出力が前のリリースと多少異なる可能性があります。詳細については、“DECIMALCONV= システムオプション” (*SAS システムオプション: リファレンス*)を参照してください。

構文

NEGPAREN*w*.[*d*]

必須引数

w
出力フィールドの幅を指定します。

デフォルト 6

範囲 1-32

オプション引数

d
数値の小数点以下の桁数を指定します。

デフォルト 0

範囲 0-31

詳細

NEGPAREN*w.d* 出力形式は、出力値を右揃えにします。入力値が負の場合、NEGPAREN*w.d* は、指定したフィールドの幅が十分な場合は値をかっこで囲って出力を表示します。フィールドの幅が十分でない場合は、負の値を表すためにマイナス符号が使用されます。入力値が負でない場合、NEGPAREN*w.d* は、適切なカラム配置を確実にするために値を先頭および末尾の空白とともに表示します。値が正の場合でも、最後のカラムは右かっこ用に確保されます。

比較

NEGPARENw.d 出力形式と COMMAw.d 出力形式は、各値を 3 桁ごとにカンマで区切るという点では同じです。

例

次の例は、NEGPARENw.d 出力形式を示しています。

ステートメント	結果
select put(100,negparen6.);	100
select put(-200,negparen6.);	(200)
select put(-2000,negparen8.);	(2,000)

NENGOw. 出力形式

SAS 日付値を *e.yymmdd* 形式で日本の日付として書き出します。

カテゴリ: 日付と時間

配置: 左

構文

NENGO*w*.

必須引数

w

出力フィールドの幅を指定します。

デフォルト 10

範囲 2-10

詳細

NENGO*w.*出力形式では、SAS 日付値を *e.yymmdd* 形式で書き出します。構文の説明は次のとおりです。

e
元号(明治、大正、昭和、平成、令和)の頭文字です。

yy
年を表す整数です。

mm
月を表す整数です。

dd
月の日を表す整数です。

幅が小さすぎる場合、SAS はピリオドを省略します。

例

例の表では、19431 を入力値として使用します。この値は 2013 年 3 月 14 日に相当する SAS 日付値です。

ステートメント	結果
select put(19431,nengo3.);	H25
select put(19431,nengo6.);	H25/03
select put(19431,nengo8.);	H.250314
select put(19431,nengo9.);	H25/03/14
select put(19431,nengo10.);	H.25/03/14

OCTALw. 出力形式

数値を 8 進表現に変換します。

カテゴリ: 数値

配置: 左

構文

OCTAL*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 3

範囲 1-24

詳細

必要に応じて、OCTAL*w*.出力形式は数値を整数に変換してから 8 進表現で表示します。

比較

OCTAL*w*. は、数値を 8 進表現に変換します。$OCTAL*w*.出力形式は、文字値を 8 進表現に変換します。

例

次の例は、OCTAL*w*.出力形式を示しています。

ステートメント	結果
select put (3592, octal16.);	0000000000007010

関連項目

出力形式:

- “$OCTALw. 出力形式” (101 ページ)

PERCENTw.d 出力形式

数値を百分率として書き出します。

カテゴリ: 数値

配置: 右

操作: DECIMALCONV=システムオプションを STDIEEE に設定した場合、この出力形式を使用して書き出される出力が前のリリースと多少異なる可能性があります。詳細については、“DECIMALCONV= システムオプション” (*SAS システムオプション: リファレンス*)を参照してください。

構文

PERCENT*w*.[*d*]

必須引数

w
出力フィールドの幅を指定します。

デフォルト 6

範囲 4-32

オプション引数

d
数値の小数点以下の桁数を指定します。

範囲 0-31

要件 *w* より小さい数にします。

詳細

PERCENT*w.d* 出力形式は、値を 100 で乗算した後に BEST*w.d* 出力形式を適用し、適用後の値の末尾にパーセント記号(%)を追加し、負の値をかっこで囲みます。
PERCENT*w.d* 出力形式では、値が負でない場合でも、パーセント記号とかっこの余地があります。

例

次の例は、PERCENT*w.d* 出力形式を示しています。

ステートメント	結果
select put(0.1,percent10.);	10%
select put(1.2,percent10.);	120%
select put(-.05,percent10.);	(5%)

関連項目

出力形式:

- “PERCENTNw.d 出力形式” (197 ページ)

PERCENTNw.d 出力形式

百分率を書き出します。負の値にはマイナス符号を使用します。

カテゴリ: 数値

配置: 右

操作: DECIMALCONV=システムオプションを STDIEEE に設定した場合、この出力形式を使用して書き出される出力が前のリリースと多少異なる可能性があります。詳細については、“DECIMALCONV= システムオプション” (*SAS システムオプション: リファレンス*)を参照してください。

構文

PERCENTN*w*.[*d*]

必須引数

w
出力フィールドの幅を指定します。

デフォルト 6

範囲 4-32

オプション引数

d
数値の小数点以下の桁数を指定します。

範囲 0-31

要件 *w* より小さい数にします。

詳細

PERCENTN*w.d* 出力形式は、 負の値に 100 を乗算し、BEST*w.d* 出力形式を適用し、適用後の値の先頭と末尾にマイナス記号とパーセント記号(%)をそれぞれ追加します。PERCENTN*w.d* 出力形式では、値が負でない場合でも、パーセント記号とマイナス記号の余地があります。

比較

PERCENTN*w.d* 出力形式は、負の値にかっこではなくマイナス記号を使用して、百分率を生成します。PERCENT*w.d* 出力形式は、負の値にかっこを使用して、百分率を生成します。

例

次の例は、PERCENTN*w.d* 出力形式を示しています。

ステートメント	結果
select put(-0.1,percentn10.);	-10%
select put(.2,percentn10.);	20%
select put(.8,percentn10.);	80%
select put(-0.05,percentn10.);	-5%
select put(-6.3,percentn10.);	-630%

関連項目

出力形式:

- “PERCENTw.d 出力形式” (196 ページ)

QTRw. 出力形式

SAS 日付値を年の四半期として書き出します。

カテゴリ: 日付と時間

配置: 右

構文

QTR*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 1

範囲 1-32

例

例の表では、19624 を入力値として使用します。この値は 2013 年 9 月 23 日に相当する SAS 日付値です。

ステートメント	結果
select put(19624,qtr.);	3

関連項目

出力形式:

- “QTRRw. 出力形式” (200 ページ)

QTRRw. 出力形式

SAS 日付値を年の四半期としてローマ数字で書き出します。

カテゴリ: 日付と時間

配置: 右

構文

QTRR*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 3

範囲 3-32

例

例の表では、19624 を入力値として使用します。この値は 2013 年 9 月 23 日に相当する SAS 日付値です。

ステートメント	結果
select put(19624,qtrr.);	III

関連項目

出力形式:

- “QTRw. 出力形式” (199 ページ)

ROMANw. 出力形式

数値をローマ数字で書き出します。

カテゴリ: 数値

配置: 左

構文

ROMAN*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 6

範囲 2-32

詳細

ROMAN*w*.出力形式は、浮動小数点数を整数に切り捨ててから値を書き出します。

例

次の例は、ROMAN*w*.出力形式を示しています。

ステートメント	結果
select put(2006,roman.);	MMVI

SIZEKw.d 出力形式

キロバイトを表す *n*K の形式で数値を書き込みます。

カテゴリ: 数値

配置: 右

構文

SIZEK*w*.[*d*]

必須引数

w
出力フィールドの幅を指定します。

デフォルト 9

範囲 2-33

オプション引数

d
数値の小数点以下の桁数を指定します。

デフォルト 0

範囲 0-31

詳細

SIZEK*w.d* 出力形式を使用して *n*K の形式で数値を書き込むには、数値を 1,024 で割って *n* の値を計算します。記号 K は、値が 1,024 の倍数であることを示します。

例

次の例は、SIZEK*w.d* 出力形式を示しています。

ステートメント	結果
select put (1024, sizek.);	1K
select put (200943, sizek.);	197K

関連項目

出力形式:

■ “SIZEKMGw.d 出力形式” (203 ページ)

SIZEKMGw.d 出力形式

キロバイトの場合は *n*KB、メガバイトの場合は *n*MB、ギガバイトの場合は *n*GB の形式で数値を書き込みます。

カテゴリ: 数値

配置: 右

構文

SIZEKMG*w*.[*d*]

必須引数

w
出力フィールドの幅を指定します。

デフォルト 6

範囲 4-35

オプション引数

d
数値の小数点以下の桁数を指定します。

デフォルト 0

範囲 0-31

詳細

SIZEKMG*w.d* 出力形式を指定すると、SAS は最適な接尾辞を決定します。KB はキロバイト、MB はメガバイト、GB はギガバイトを表し、SAS 数値は次のいずれかの値で割られます。

KB
1024

MB
1048576

GB
1073741824

例

次の例は、SIZEKMG*w.d* 出力形式を示しています。

ステートメント	結果
select put(3688, sizekmg.);	4KB
select put(1048576, sizekmg.);	1MB
select put(83409922345, sizekmg.);	78GB

関連項目

出力形式:

- “SIZEKw.d 出力形式” (201 ページ)

TIMEw.d 出力形式

SAS 時間値を時間、分、および秒として *hh:mm:ss.ss* 形式で書き出します。

カテゴリ: 日付と時間

配置: 右

操作: DECIMALCONV=システムオプションを STDIEEE に設定した場合、この出力形式を使用して書き出される出力が前のリリースと多少異なる可能性があります。詳細については、“DECIMALCONV= システムオプション” (*SAS システムオプション: リファレンス*)を参照してください。

構文

TIME*w*.[*d*]

必須引数

w
出力フィールドの幅を指定します。

デフォルト 8

範囲 2-20

ヒント *w* には、必要な結果を生成するために十分な大きさを指定します。小数点以下 3 桁までをすべて含む時間値を取得するには、少なくとも 12 スペース(小数点の左側に 8 スペース、小数点自体に 1 スペース、秒の小数部分に 3 スペース)を指定する必要があります。

オプション引数

d
秒値の小数点以下の桁数を指定します。

デフォルト 0

範囲 0-19

要件 *w* より小さい数にします。

詳細

TIME*w.d* 出力形式は、SAS 時間値を *hh:mm:ss.ss* 形式で書き出します。構文の説明は次のとおりです。

hh
整数です。

注: *hh* が 1 桁の場合、TIME*w.d* はその桁の前に先頭の空白を置きます。たとえば、TIME*w.d.* 出力形式は、09:00 ではなく 9:00 と書き出します。

mm
00 から 59 の範囲の時間(分)です。

ss.ss
小数点以下の秒の端数を含む、00 から 59 の範囲の秒数です。

比較

TIME*w.d* 出力形式は HHMM*w.d* 出力形式と同じですが、TIME*w.d* は秒を含めます。

TIME*w.d* 出力形式と HHMM*w* は、1 桁の時間にはその先頭に空白を書き出します。TOD*w.d* 出力形式は、1 桁の時間にはその先頭にゼロを書き出します。

例

例の表では、59083 を入力値として使用します。この値は、午後 4:24:43 に相当する SAS 時間値です。

ステートメント	結果
select put(59083,time.);	16:24:43

関連項目

出力形式:

- “HHMMw.d 出力形式” (171 ページ)
- “HOURw.d 出力形式” (173 ページ)
- “MMSSw.d 出力形式” (183 ページ)
- “TODw.d 出力形式” (208 ページ)

関数:

- “HOUR 関数” (570 ページ)
- “MINUTE 関数” (670 ページ)
- “SECOND 関数” (824 ページ)

TIMEAMPMw.d 出力形式

SAS 時間値を時間、分、および秒として *hh:mm:ss.ss* 形式で AM または PM を付けて書き出します。

カテゴリ: 日付と時間

配置: 右

操作: DECIMALCONV=システムオプションを STDIEEE に設定した場合、この出力形式を使用して書き出される出力が前のリリースと多少異なる可能性があります。詳細については、“DECIMALCONV= システムオプション” (*SAS システムオプション: リファレンス*)を参照してください。

構文

TIMEAMPM*w*.[*d*]

必須引数

w
出力フィールドの幅を指定します。

デフォルト 11

範囲 2-20

オプション引数

d
秒値の小数点以下の桁数を指定します。

デフォルト 0

範囲 0-19

要件 *w* より小さい数にします。

詳細

TIMEAMPM*w.d* 出力形式は、SAS 時間値を *hh:mm:ss.ss* 形式で AM または PM を付けて書き出します。構文の説明は次のとおりです。

hh
時間を表す整数です。

mm
分を表す整数です。

ss.ss
小数点以下 2 桁の秒数です。

23:59:59 PM を超える時間は、次の日として表示されます。

w には、必要な結果を生成するために十分な大きさを指定します。小数点以下 3 桁と AM または PM をすべて含む時間値を取得するには、少なくとも 11 スペース (*hh:mm:ss* PM)を指定する必要があります。*w* が 5 未満の場合、AM または PM だけが書き出されます。

比較

- TIMEAMPMM*w.d* 出力形式は TIMEM*w.d* 出力形式と同じですが、TIMEAMPMM*w.d* は時間の最後に AM または PM を出力します。
- TIME*w.d* は 23:59:59 PM を超える時間を書き出しますが、TIMEAMPM*w.d* は書き出しません。

例

例の表では、59083 を入力値として使用します。この値は、午後 4:24:43 に相当する SAS 時間値です。

ステートメント	結果
select put(59083,timeampm3.);	PM
select put(59083,timeampm5.);	4 PM
select put(59083,timeampm7.);	4:24 PM
select put(59083,timeampm11.);	4:24:43 PM

関連項目

出力形式:

- “TIMEw.d 出力形式” (204 ページ)

TODw.d 出力形式

SAS 時間値と、SAS 日時値の時間部分を *hh:mm:ss.ss* 形式で書き出します。

カテゴリ: 日付と時間

配置: 右

操作: DECIMALCONV=システムオプションを STDIEEE に設定した場合、この出力形式を使用して書き出される出力が前のリリースと多少異なる可能性があります。詳細については、“DECIMALCONV= システムオプション” (*SAS システムオプション: リファレンス*)を参照してください。

構文

TOD*w*.[*d*]

必須引数

w
出力フィールドの幅を指定します。

デフォルト 8

範囲 2-20

ヒント 指定した幅が十分な場合は、ゼロ時間に対しゼロが書き出されます (たとえば、02:30 または 00:30)。

オプション引数

d
秒値の小数点以下の桁数を指定します。

デフォルト	0
範囲	0-19
要件	*w* より小さい数にします。

詳細

TOD*w.d* 出力形式は、SAS 日時値を *hh:mm:ss.ss* 形式で書き出します。構文の説明は次のとおりです。

hh
時間を表す整数です。

mm
分を表す整数です。

ss.ss
小数点以下 2 桁の秒数です。

比較

TOD*w.d* 出力形式は、1 桁の時間にはその先頭にゼロを書き出します。TIME*w.d* 出力形式と HHMM*w.d* 出力形式は、1 桁の時間にはその先頭に空白を書き出します。

例

次の例は、TOD*w.d* 出力形式を示しています。

ステートメント	結果
select put(1472049623,tod9.);	14:40:23

関連項目

出力形式:

- “TIMEw.d 出力形式” (204 ページ)
- “TIMEAMPMw.d 出力形式” (206 ページ)

関数:

- “TIMEPART 関数” (851 ページ)

VAXRBw.d 出力形式

VMS 形式のバイナリ実数(浮動小数点)を書き出します。

カテゴリ: 数値

配置: 右

構文

VAXRB*w*.[*d*]

必須引数

w
出力フィールドの幅を指定します。

デフォルト 8

範囲 2-8

オプション引数

d
値を除算する 10 のべき乗を指定します。

デフォルト 0

範囲 0-31

詳細

VAXRB*w.d* 出力形式を使用して、ネイティブ VAX/VMS 浮動小数点数表記のデータを書き出します。

例

次の例は、VAXRB*w.d* 出力形式を示しています。

ステートメント	結果 [1]
select put(1,vaxrb8.);	8040000000000000

1 結果は、整数を 16 進表現で表したものです。

w.d 出力形式

標準数値でデータを書き出します(1 バイト 1 桁)。

カテゴリ: 数値

配置: 右

別名: F*w.d*

操作: DECIMALCONV=システムオプションを STDIEEE に設定した場合、この出力形式を使用して書き出される出力が前のリリースと多少異なる可能性があります。詳細については、"DECIMALCONV= システムオプション" (*SAS システムオプション: リファレンス*)を参照してください。

構文

w.[*d*]

必須引数

w

出力フィールドの幅を指定します。

範囲 1-32

ヒント 必要に応じて、値、小数点、マイナス記号を書き出せる十分なスペースを指定します。

オプション引数

d

数値の小数点以下の桁数を指定します。

範囲 0-31

要件 *w* より小さい数にします。

ヒント *d* が 0、または *d* を指定しない場合、*w.d* は値を小数点なしで書き出します。

詳細

w.d 出力形式は、出力フィールドにおさまる近傍値になるように値を丸めます。*w.d* が小さすぎる場合、小数点揃えを BEST*w*.出力形式に合わせます。*w.d* 出力形式は、負の数には先頭のマイナス記号を付けて書き出します。また、*w.d* は、出力を右揃えにしてから書き出し、先頭に空白を挿入します。

比較

Z*w.d* 出力形式は、*w.d* 出力形式とほぼ同じですが、Z*w.d* は右揃えにした出力に空白ではなく 0 を挿入します。

例

次の例は、*w.d* 出力形式を示しています。

ステートメント	結果
select put (23.45, 6.3);	23.450

関連項目

出力形式:

- “Zw.d 出力形式” (246 ページ)

WEEKDATEw. 出力形式

SAS 日付値を曜日と日付として *day-of-week*, *month-name dd*, *yy* (または *yyyy*)形式で書き出します。

カテゴリ: 日付と時間

配置: 右

構文

WEEKDATE*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 29

範囲 3-37

詳細

WEEKDATE*w*.出力形式は、SAS 日付値を *day-of-week*, *month-name dd*, *yy* (または *yyyy*)形式で書き出します。構文の説明は次のとおりです。

dd
月の日を表す整数です。

yy または *yyyy*
年を表す 2 桁または 4 桁の整数です。

w が小さすぎて曜日と月の名前を完全な形で書き出すことができない場合、必要に応じて短縮形で書き出します。

比較

WEEKDATE*w*.出力形式は WEEKDATX*w*.出力形式とほぼ同じですが、WEEKDATX*w*. は月名の前に *dd* を出力します。

例

例の表では、20999 を入力値として使用します。この値は 2017 年 6 月 29 日に相当する SAS 日付値です。

ステートメント	結果
select put(20999,weekdate3.);	Thu
select put(20999,weekdate9.);	Thursday
select put(20999,weekdate15.);	Thu, Jun 29, 17
select put(20999,weekdate17.);	Thu, Jun 29, 20137

関連項目

出力形式:

- “DTWKDATXw. 出力形式” (148 ページ)
- “DATEw. 出力形式” (135 ページ)
- “DDMMYYw. 出力形式” (138 ページ)
- “MMDDYYw. 出力形式” (179 ページ)
- “TODw.d 出力形式” (208 ページ)
- “WEEKDATXw. 出力形式” (214 ページ)
- “YYMMDDw. 出力形式” (227 ページ)

関数:

- “JULDATE 関数” (626 ページ)
- “MDY 関数” (664 ページ)
- “WEEKDAY 関数” (883 ページ)

WEEKDATXw. 出力形式

SAS 日付値を曜日と日付として *day-of-week*, *dd month-name yy* (または *yyyy*)形式で書き出します。

カテゴリ: 日付と時間

配置: 右

構文

WEEKDATX*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 29

範囲 3-37

詳細

WEEKDATX*w*.出力形式は、SAS 日付値を *day-of-week*, *dd month-name*, *yy* (または *yyyy*)形式で書き出します。構文の説明は次のとおりです。

dd
月の日を表す整数です。

yy または *yyyy*
年を表す 2 桁または 4 桁の整数です。

w が小さすぎて曜日と月の名前を完全な形で書き出すことができない場合、必要に応じて短縮形で書き出します。

比較

WEEKDATE*w*.出力形式は WEEKDATX*w*.出力形式とほぼ同じですが、WEEKDATE*w*. は月名の後に *dd* を出力します。

DTWKDATX*w*.出力形式は WEEKDATX*w*.出力形式とほぼ同じですが、DTWKDATX*w*. は入力として日時値が必要です。

例

例の表では、20999 を入力値として使用します。この値は 2017 年 6 月 29 日に相当する SAS 日付値です。

ステートメント	結果
select put(20999,weekdatx.);	Thursday, 29 June 2017

関連項目

出力形式:

- “DTWKDATXw. 出力形式” (148 ページ)
- “DATEw. 出力形式” (135 ページ)
- “DDMMYYw. 出力形式” (138 ページ)
- “MMDDYYw. 出力形式” (179 ページ)
- “TODw.d 出力形式” (208 ページ)
- “WEEKDATEw. 出力形式” (212 ページ)
- “YYMMDDw. 出力形式” (227 ページ)

関数:

- “JULDATE 関数” (626 ページ)
- “MDY 関数” (664 ページ)
- “WEEKDAY 関数” (883 ページ)

WEEKDAYw. 出力形式

SAS 日付値から曜日の値を書き出します。

カテゴリ: 日付と時間

配置: 右

構文

WEEKDAY*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 1

範囲 1-32

詳細

WEEKDAY*w*.出力形式は、SAS 日付値を曜日の値(1=日曜日、2=月曜日など)として書き出します。

例

例の表では、20999 を入力値として使用します。この値は 2017 年 6 月 29 日に相当する SAS 日付値です。

ステートメント	結果
select put(20999,weekday.);	5

関連項目

出力形式:

- “DOWNAMEw. 出力形式” (144 ページ)

WEEKUw. 出力形式

U アルゴリズムを使用して、10 進数の週番号を書き出します。

カテゴリ: 日付と時間

構文

WEEKU*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 11

範囲 2-200

詳細

WEEKU*w*.出力形式では、週番号形式で書き込みます。WEEKU*w*.出力形式では、指定された幅に応じてさまざまな形式で書き込みます。U アルゴリズムでは、年内の週番号を使用して SAS 日付値を計算します(日曜日を週の最初の日と見なします)。週番号値は、先頭に 0 を付けた 0 から 53 の範囲の 10 進数として表され、最大値は 53 になります。たとえば、年の 5 週目は 05 として表されます。

幅、出力形式および例の詳細については、次の表を参照してください。

幅	出力形式	例
3-4	Www	w01
5-6	yyWww	03W01
7-8	yyWwwdd	03W0101
9–10	yyyyWwwdd	2003W0101

幅	出力形式	例
11-200	yyyy-Www-dd	2003-W01-01

比較

WEEKV*w*.出力形式では、01 から 53 の範囲の 10 進数として週番号を書き込みます。1 月の最初の月曜日が 2 日、3 日または 4 日の場合、それより前の日は前年の最後の週に組み込まれます。WEEKW*w*.出力形式では、00 から 53 の範囲の 10 進数として年の週番号を書き込みます。第 1 週の最初の日は月曜日になります。WEEKU*w*.出力形式では、先頭に 0 を付けた 0 から 53 の範囲の 10 進数として年の週番号を書き込みます(週の最初の日は日曜日になります)。

例

次のステートメントは、WEEKU*w*.出力形式を示しています。入力値 20999 は、2017 年 6 月 29 日の SAS 日付値に相当します。

ステートメント	結果
select put(20999,weeku3.);	W26
select put(20999,weeku5.);	17W26
select put(20999,weeku7.);	17W2605
select put(20999,weeku9.);	2017W2605
select put(20999,weeku11.);	2017-W26-05

WEEKVw. 出力形式

V アルゴリズムを使用して、10 進数の週番号を書き出します。

カテゴリ: 日付と時間

配置: 左

構文

WEEKV*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 11

範囲 2-200

詳細

WEEKV*w*.出力形式では、指定された幅に応じてさまざまな形式で書き込みます。V アルゴリズムでは、SAS 日付値を計算します。週番号値は、先頭に 0 を付けた 01 から 53 の範囲の 10 進数として表され、最大値が 53 になります。各週は月曜日から始まり、年の第 1 週は 1 月 4 日と年の最初の木曜日の両方を含む週です。1 月の最初の月曜日が 2 日、3 日または 4 日の場合、それより前の日は前年の最後の週に組み込まれます。たとえば、年の 5 週目は 06 として表されます。

幅、出力形式および例の詳細については、次の表を参照してください。

幅	出力形式	例
3-4	Www	w01
5-6	yyWww	03W01
7-8	yyWwwdd	03W0101
9–10	yyyyWwwdd	2003W0101
11-200	yyyy-Www-dd	2003-W01-01

比較

WEEKV*w*.出力形式では、01 から 53 の範囲の 10 進数として週番号を書き込みます。1 月の最初の月曜日が 2 日、3 日または 4 日の場合、それより前の日は前年の最後の週に組み込まれます。WEEKW*w*.出力形式では、00 から 53 の範囲の 10 進数として年の週番号を書き込みます。第 1 週の最初の日は月曜日になります。WEEKU*w*.出力形式では、先頭に 0 を付けた 0 から 53 の範囲の 10 進数として年の週番号を書き込みます(週の最初の日は日曜日になります)。

例

次のステートメントは、WEEKV*w*.出力形式を示しています。入力値 20999 は、2017 年 6 月 29 日の SAS 日付値に相当します。

ステートメント	結果
select put(20999,weekv3.);	W26
select put(20999,weekv5.);	17W26
select put(20999,weekv7.);	17W2604
select put(20999,weekv9.);	2017W2604
select put(20999,weekv11.);	2017-W26-04

WEEKWw. 出力形式

W アルゴリズムを使用して、10 進数の週番号を書き出します。

カテゴリ: 日付と時間

配置: 左

構文

WEEKW*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 11

範囲 2-200

詳細

WEEKW*w*.出力形式では、指定された幅に応じてさまざまな形式で書き込みます。W アルゴリズムでは、年内の週番号を使用して SAS 日付値を計算します(月曜日を週の最初の日と見なします)。週番号値は、先頭に 0 を付けた 0 から 53 の範囲の 10 進数

として表され、最大値は 53 になります。たとえば、年の 5 週目は 05 として表されます。

幅、出力形式および例の詳細については、次の表を参照してください。

幅	出力形式	例
3-4	Www	w01
5-6	yyWww	03W01
7-8	yyWwwdd	03W0101
9–10	yyyyWwwdd	2003W0101
11-200	yyyy-Www-dd	2003-W01-01

比較

WEEKV*w*.出力形式では、01 から 53 の範囲の 10 進数として週番号を書き込みます。各週は月曜日から始まり、年の第 1 週には 1 月 4 日と年の最初の木曜日の両方が含まれます。1 月の最初の月曜日が 2 日、3 日または 4 日の場合、それより前の日は前年の最後の週に組み込まれます。WEEKW*w*.出力形式では、00 から 53 の範囲の 10 進数として年の週番号を書き込みます。第 1 週の最初の日は月曜日になります。WEEKU*w*.出力形式では、先頭に 0 を付けた 0 から 53 の範囲の 10 進数として年の週番号を書き込みます(週の最初の日は日曜日になります)。

例

次のステートメントは、WEEKW*w*.出力形式を示しています。入力値 20999 は、2017 年 6 月 29 日の SAS 日付値に相当します。

ステートメント	結果
select put(20999,weekw3.);	W26
select put(20999,weekw5.);	17W26
select put(20999,weekw7.);	17W2604
select put(20999,weekw9.);	2017W2604
select put(20999,weekw11.);	2017-W26-04

YEARw. 出力形式

SAS 日付値から年部分を書き出します。

カテゴリ: 日付と時間

配置: 右

構文

YEAR*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 4

範囲 2-32

ヒント *w* が 4 より小さい場合、年の最後の 2 桁が出力されます。それ以外の場合、年値は 4 桁で出力されます。

比較

YEAR*w*.出力形式と DTYEAR*w*.出力形式は、どちらも日付値を書き出すという点でほぼ同じです。相違点は、YEAR*w*.が入力として SAS 日付値を必要とし、DTYEAR*w*.は SAS 日時値が必要であることです。

例

例の表では、20999 を入力値として使用します。この値は 2017 年 6 月 29 日に相当する SAS 日付値です。

ステートメント	結果
select put(20999,year2.);	17
select put(20999,year4.);	2017

関連項目

出力形式:

- “DTYEARw. 出力形式” (150 ページ)

YENw.d 出力形式

円記号、カンマ、小数点を使用して数値を書き込みます。

カテゴリ: 数値

配置: 右

構文

YEN*w.d*

必須引数

w
出力フィールドの幅を指定します。

デフォルト 8

範囲 1-32

d
数値の小数点以下の桁数を指定します。

制限事項 0 または 2 のいずれかである必要があります

ヒント *d* が 2 の場合、YEN*w.d* は小数点と小数点以下 2 桁を書き込みます。*d* が 0 の場合、YEN*w.d* は小数点と小数点以下の桁を書き込みません。

詳細

YEN*w.d* 出力形式は、先頭に円記号を付け、各値の 3 桁ごとにカンマで区切って数値を書き込みます。

円記号文字のコードの 16 進表現は、EBCDIC のシステムでは 5B、ASCII のシステムでは 5C です。これらのコードが表す通貨記号は、他の国では異なる場合があります。

例

次の例は、YEN*w.d* 出力形式を示しています。

ステートメント	結果
select put(1254.71,yen10.2);	¥1,254.71

YYMMw. 出力形式

SAS 日付値を*<yy>yy*M*mm* 形式で書き出します。M は月番号が M の後に続くことを示す区切り文字で、年は 2 桁または 4 桁の数字で書き出されます。

カテゴリ: 日付と時間

配置: 右

構文

YYMM*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 7

範囲 5-32

操作 *w* の値が 5 または 6 の場合、日付の年は下 2 桁でのみ表示されます。*w* が 7 以上の場合、日付の年は 4 桁で表示されます。

詳細

YYMM*w*.出力形式は、SAS 日付値を*<yy>yy*M*mm* 形式で書き出します。構文の説明は次のとおりです。

<yy>yy
年を表す 2 桁または 4 桁の整数です。

M
区切り文字です。

mm
月を表す整数です。

比較

- YYMM*w.d* 出力形式は YYMM*xw.d* 出力形式とほぼ同じですが、YYMM*xw.d* 出力形式では年と月の間にハイフン、コロン、スラッシュ、ピリオドなどの区切り文字が含まれている点が異なります。

例

次の例は、YYMM*w.*出力形式を示しています。例では、20999 を入力値として使用します。この値は、2017 年 6 月 29 日に相当する SAS 日付値です。

ステートメント	結果
select put(20999,yymm5.);	17M06
select put(20999,yymm6.);	17M06
select put(20999,yymm.);	2017M06
select put(20999,yymm7.);	2017M06
select put(20999,yymm10.);	2017M06

関連項目

出力形式:

- “MMYYw. 出力形式” (185 ページ)
- “YYMMxw. 出力形式” (225 ページ)

YYMMxw. 出力形式

SAS 日付値を*<yy>yymm* または*<yy>yy-mm* 形式で書き出します。出力形式名の *x* は、年と月を区切る特殊文字を表す文字で、ハイフン(-)、ピリオド(.)、空白文字、スラッシュ(/)、コロン(:)、または区切り文字なしになります。年は 2 桁または 4 桁の数字で書き出されます。

カテゴリ: 日付と時間

配置: 右

構文

YYMM*xw*.

必須引数

x
区切り文字を示します。または、区切り文字を年と月の間に挿入しないことを示します。有効な値は次のとおりです。

C
コロンで区切ります。

D
ハイフンで区切ります。

N
区切り文字なしを表します。

P
ピリオドで区切ります。

S
スラッシュで区切ります。

w
出力フィールドの幅を指定します。

デフォルト　7

範囲　5-32

操作　*x* を *N* に設定すると、区切り文字なしを指定します。この場合の幅範囲は 4–32 になり、デフォルトは 6 に変化します。

x の値が C、D、P または S で、*w* の値が 5 または 6 の場合、日付の年は下 2 桁のみ表示されます。*w* が 7 以上の場合、日付の年は 4 桁で表示されます。

x の値が *N* で、*w* の値が 4 または 5 の場合、日付の年は下 2 桁のみ表示されます。*x* の値が *N* で、*w* が 6 以上の場合、日付の年は 4 桁で表示されます。

詳細

YYMM*xw*.出力形式は、SAS 日付値を*<yy>yymm* または*<yy>yyXmm* 形式で書き出します。構文の説明は次のとおりです。

<yy>yy
年を表す 2 桁または 4 桁の整数です。

X
指定された区切り文字です。

mm
月を表す整数です。

比較

- YYMM*w.d* 出力形式は YYMM*xw.d* 出力形式とほぼ同じですが、YYMM*xw.d* 出力形式では年と月の間にハイフン、コロン、スラッシュ、ピリオドなどの区切り文字が含まれている点が異なります。

例

次の例は、YYMM*xw*.出力形式を示しています。例では、20999 を入力値として使用します。この値は、2017 年 6 月 29 日に相当する SAS 日付値です。

ステートメント	結果
select put(20999,yymmc5.);	17:06
select put(20999,yymmd.);	2017-06
select put(20999,yymmn4.);	1706
select put(20999,yymmp8.);	2017.06
select put(20999,yymms10.);	2017/06

関連項目

出力形式:

- “MMYYxw. 出力形式” (186 ページ)
- “YYMMw. 出力形式” (224 ページ)

YYMMDDw. 出力形式

SAS 日付値を*<yy>yymmdd* または*<yy>yy–mm–dd* 形式で書き出します。ハイフンが区切り文字として使用され、年は 2 桁または 4 桁の数字で書き出されます。

カテゴリ: 日付と時間

配置: 右

構文

YYMMDD*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 8

範囲 2-10

操作 *w* の値が 2 から 5 までの場合、日付の年と月が可能な範囲で表示されます。*w* が 7 の場合、日付は 2 桁の年としてハイフンなしで表示されます。

詳細

YYMMDD*w*.出力形式は、SAS 日付値を*<yy>yymmdd* または*<yy>yy–mm–dd* 形式で書き出します。構文の説明は次のとおりです。

<yy>yy
年を表す 2 桁または 4 桁の整数です。

–
区切り文字です。

mm
月を表す整数です。

dd
月の日を表す整数です。

比較

- YYMMDD*w.d* 出力形式は、YYMMDD*xw.d* 出力形式とほぼ同じですが、YYMMDD*xw.d* 出力形式では年、月、日の間にコロン、スラッシュ、ピリオドなどの区切り文字が含まれている点が異なります。

例

次の例は、YYMMDD*w*.出力形式を示しています。例では、20999 を入力値として使用します。この値は、2017 年 6 月 29 日に相当する SAS 日付値です。

ステートメント	結果
select put(20999,yymmdd2.);	17
select put(20999,yymmdd3.);	17
select put(20999,yymmdd4.);	1706
select put(20999,yymmdd5.);	17-06
select put(20999,yymmdd6.);	170629
select put(20999,yymmdd7.);	170629
select put(20999,yymmdd8.);	17-06-29
select put(20999,yymmdd10.);	2017-06-29

関連項目

出力形式:

- “DATEw. 出力形式” (135 ページ)
- “DDMMYYw. 出力形式” (138 ページ)
- “MMDDYYw. 出力形式” (179 ページ)
- “YYMMDDxw. 出力形式” (229 ページ)

関数:

- “DAY 関数” (443 ページ)
- “MDY 関数” (664 ページ)
- “MONTH 関数” (676 ページ)
- “YEAR 関数” (887 ページ)

YYMMDDxw. 出力形式

SAS 日付値を*<yy>yymmdd* または*<yy>yy-mm-dd* 形式で書き出します。出力形式名の *x* は、年、月、日を区切る特殊文字を表す文字です。特殊文字は、ハイフン(-)、ピリオド(.)、空白文字、スラッシュ(/)、コロン(:)または区切り文字なしになります。年は 2 桁または 4 桁のいずれかになります。

カテゴリ: 日付と時間

配置: 右

構文

YYMMDD*xw*.

必須引数

x
区切り文字を示します。または、区切り文字を年、月、日の間に挿入しないことを示します。有効な値は次のとおりです。

B
空白で区切ります。

C
コロンで区切ります。

D
ハイフンで区切ります。

N
区切り文字なしを表します。

P
ピリオドで区切ります。

S
スラッシュで区切ります。

w
出力フィールドの幅を指定します。

デフォルト　8

範囲　2-10

操作　*w* の値が 2 から 5 までの場合、日付の年と月が可能な範囲で表示されます。*w* が 7 の場合、日付の年は 2 桁で、区切り文字を使用せずに表示されます。

x の値が N の場合、幅の範囲は 2–8 に変化します。

詳細

YYMMDD*xw*.出力形式は、SAS 日付値を*<yy>yymmdd* または*<yy>yyXmmXdd* 形式で書き出します。構文の説明は次のとおりです。

<yy>yy
年を表す 2 桁または 4 桁の整数です。

X
指定された区切り文字です。

mm
月を表す整数です。

dd
月の日を表す整数です。

比較

- YYMMDD*w.d* 出力形式は YYMMDD*xw.d* 出力形式とほぼ同じですが、YYMMDD*xw.d* 出力形式には、年、月、日の間にコロン、スラッシュ、ピリオドなどの区切り文字が含まれています。

例

次の例は、YYMMDD*xw*.出力形式を示しています。例では、20999 を入力値として使用します。この値は、2017 年 6 月 29 日に相当する SAS 日付値です。

ステートメント	結果
select put(20999,yymmddc5.);	17:06
select put(20999,yymmddd8.);	17-06-29
select put(20999,yymmddp10.);	2017.06.29

関連項目

出力形式:

- “DATEw. 出力形式” (135 ページ)
- “DDMMYYxw. 出力形式” (139 ページ)
- “MMDDYYxw. 出力形式” (181 ページ)
- “YYMMDDw. 出力形式” (227 ページ)

関数:

- “DAY 関数” (443 ページ)
- “MDY 関数” (664 ページ)
- “MONTH 関数” (676 ページ)
- “YEAR 関数” (887 ページ)

YYMONw. 出力形式

SAS 日付値を *yymmm* または *yyyymmm* 形式で書き出します。

カテゴリ: 日付と時間

配置: 右

構文

YYMON*w*.

必須引数

w
出力フィールドの幅を指定します。出力形式の幅が小さすぎて 4 桁の年を出力できない場合、年の下 2 桁だけが出力されます。

デフォルト 7

範囲 5-32

詳細

YYMON*w*.出力形式では、月名が 3 文字に省略されます。

例

次の例は、YYMON*w*.出力形式を示しています。例では、20999 を入力値として使用します。この値は、2017 年 6 月 29 日に相当する SAS 日付値です。

ステートメント	結果
select put(20999,yymon6.);	17JUN
select put(20999,yymon7.);	2017JUN

関連項目

出力形式:

- “MMYYw. 出力形式” (185 ページ)

YYQw. 出力形式

SAS 日付値を*<yy>yy*Q*q* 形式で書き出します。Q は区切り文字として使用され、年は 2 桁または 4 桁の数字で書き出されます。また、*q* は四半期を表します。

カテゴリ: 日付と時間

配置: 右

構文

YYQ*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 6

範囲 4-32

操作 *w* の値が 4 または 5 の場合、日付の年は下 2 桁でのみ表示されます。*w* が 6 以上の場合、日付の年は 4 桁で表示されます。

詳細

YYQ*w*.出力形式は、SAS 日付値を*<yy>yy*Q*q* 形式で書き出します。構文の説明は次のとおりです。

<yy>yy
年を表す 2 桁または 4 桁の整数です。

Q
区切り文字です。

q
四半期を表す整数(1、2、3、4 のいずれか)です。

比較

YYQ*w*.出力形式は YYQ*xw*.出力形式とほぼ同じですが、YYQ*xw*.出力形式には、YY と Q の間にハイフン、スラッシュ、ピリオド、コロンなどの区切り文字があります。

例

次の例は、YYQ*w*.出力形式を示しています。例では、20999 を入力値として使用します。この値は、2017 年 6 月 29 日に相当する SAS 日付値です。

ステートメント	結果
select put(20999,yyq4.);	17Q2
select put(20999,yyq5.);	17Q2
select put(20999,yyq.);	2017Q2
select put(20999,yyq6.);	2017Q2
select put(20999,yyq10.);	2017Q2

関連項目

出力形式:

- “YYQxw. 出力形式” (234 ページ)
- “YYQRw. 出力形式” (236 ページ)
- “YYQRxw. 出力形式” (238 ページ)
- “YYQZw. 出力形式” (240 ページ)

YYQxw. 出力形式

SAS 日付値を*<yy>yyq* または*<yy>yy-q* 形式で書き出します。出力形式名の *x* は、年と四半期を区切る特殊文字を表す文字で、ハイフン(-)、ピリオド(.)、空白文字、スラッシュ(/)、コロン(:)、または区切り文字なしになります。年は 2 桁または 4 桁の数字で書き出されます。

カテゴリ: 日付と時間

配置: 右

構文

YYQ*xw*.

必須引数

x
区切り文字を示します。または、区切り文字を年と四半期の間に挿入しないことを示します。有効な値は次のとおりです。

C
コロンで区切ります。

D
ハイフンで区切ります。

N
区切り文字なしを表します。

P
ピリオドで区切ります。

S
スラッシュで区切ります。

w
出力フィールドの幅を指定します。

デフォルト 6

範囲 4-32

操作 *x* を *N* に設定すると、区切り文字なしを指定します。この場合の幅範囲は 3–32 になり、デフォルトは 5 に変化します。

w の値が 4 または 5 の場合、日付の年は下 2 桁でのみ表示されます。*w* が 6 以上の場合、日付の年は 4 桁で表示されます。

x の値が *N* で、*w* の値が 3 または 4 の場合、日付の年は下 2 桁のみ表示されます。*x* の値が *N* で、*w* が 5 以上の場合、日付の年は 4 桁で表示されます。

詳細

YYQ*xw*.出力形式は、SAS 日付値を*<yy>yyq* または*<yy>yyXq* 形式で書き出します。構文の説明は次のとおりです。

<yy>yy
年を表す 2 桁または 4 桁の整数です。

X
指定された区切り文字です。

q
四半期を表す整数(1、2、3、4 のいずれか)です。

比較

YYQ*w*.出力形式は YYQ*xw*.出力形式とほぼ同じですが、YYQ*xw*.出力形式には、YY と Q の間にハイフン、スラッシュ、ピリオド、コロンなどの区切り文字があります。

例

次の例は、YYQ*xw*.出力形式を示しています。例では、20999 を入力値として使用します。この値は、2017 年 6 月 29 日に相当する SAS 日付値です。

ステートメント	結果
select put(20999,yyqc4.);	17:2
select put(20999,yyqd.);	2017-2
select put(20999,yyqn3.);	172
select put(20999,yyqp6.);	2017.2
select put(20999,yyqs8.);	2017/2

関連項目

出力形式:

- “YYQw. 出力形式” (233 ページ)
- “YYQRw. 出力形式” (236 ページ)
- “YYQRxw. 出力形式” (238 ページ)
- “YYQZw. 出力形式” (240 ページ)

YYQRw. 出力形式

SAS 日付値を<*yy*>*yy*Q*qr* 形式で書き出します。Q は区切り文字として使用され、年は 2 桁または 4 桁の数字で書き出されます。また *qr* はローマ数字表現の四半期を表します。

カテゴリ: 日付と時間

配置: 右

構文

YYQR*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 8

範囲 6-32

操作 *w* の値が小さすぎて 4 桁の年を書き出せない場合、日付の年は下 2 桁でのみ表示されます。

詳細

YYQR*w*.出力形式は、SAS 日付値を*<yy>yy*Q*qr* 形式で書き出します。構文の説明は次のとおりです。

<yy>yy
年を表す 2 桁または 4 桁の整数です。

Q
区切り文字です。

qr
四半期を表すローマ数字表現(I、II、III、IV のいずれか)です。

比較

YYQR*w*.出力形式は YYQR*xw*.出力形式とほぼ同じですが、YYQR*xw*.出力形式には、YY と QR の間にハイフン、スラッシュ、ピリオド、コロンなどの区切り文字があります。

例

次の例は、YYQR*w*.出力形式を示しています。例では、20999 を入力値として使用します。この値は、2017 年 6 月 29 日に相当する SAS 日付値です。

ステートメント	結果
select put(20999,yyqr6.);	17QII
select put(20999,yyqr7.);	2017QII
select put(20999,yyqr.);	2017QII
select put(20999,yyqr8.);	2017QII
select put(20999,yyqr10.);	2017QII

関連項目

出力形式:

- “YYQw. 出力形式” (233 ページ)
- “YYQRxw. 出力形式” (238 ページ)
- “YYQxw. 出力形式” (234 ページ)
- “YYQZw. 出力形式” (240 ページ)

YYQRxw. 出力形式

SAS 日付値を*<yy>yyqr* または*<yy>yy-qr* 形式で書き出します。出力形式名の *x* は、年と四半期を区切る特殊文字を表す文字で、ハイフン(-)、ピリオド(.)、空白文字、スラッシュ(/)、コロン(:)、または区切り文字なしになります。年は 2 桁または 4 桁の数字で書き出されます。また *qr* は、ローマ数字表現の四半期です。

カテゴリ: 日付と時間

配置: 右

構文

YYQR*xw*.

必須引数

x
区切り文字を示します。または、区切り文字を年と四半期の間に挿入しないことを示します。有効な値は次のとおりです。

C
コロンで区切ります。

D
ハイフンで区切ります。

N
区切り文字なしを表します。

P
ピリオドで区切ります。

S
スラッシュで区切ります。

w
出力フィールドの幅を指定します。

デフォルト 8

範囲 6-32

操作 *x* を *N* に設定すると、区切り文字なしを指定します。この場合の幅範囲は 5–32 になり、デフォルトは 7 に変化します。

w の値が小さすぎて 4 桁の年を書き出せない場合、日付の年は下 2 桁でのみ表示されます。

詳細

YYQR*xw*.出力形式は、SAS 日付値を*<yy>yyqr* または*<yy>yy*X*qr* 形式で書き出します。構文の説明は次のとおりです。

<yy>yy
年を表す 2 桁または 4 桁の整数です。

X
指定された区切り文字です。

qr
四半期を表すローマ数字表現(I、II、III、IV のいずれか)です。

比較

YYQR*w*.出力形式は YYQR*xw*.出力形式とほぼ同じですが、YYQR*xw*.出力形式には、YY と QR の間にハイフン、スラッシュ、ピリオド、コロンなどの区切り文字があります。

例

次の例は、YYQR*xw*.出力形式を示しています。例では、20999 を入力値として使用します。この値は、2017 年 6 月 29 日に相当する SAS 日付値です。

ステートメント	結果
select put(20999,yyqrc6.);	17:II
select put(20999,yyqrd.);	2017-II
select put(20999,yyqrn5.);	17II
select put(20999,yyqrp8.);	2017.II
select put(20999,yyqrs10.);	2017/II

関連項目

出力形式:

- “YYQxw. 出力形式” (234 ページ)
- “YYQRw. 出力形式” (236 ページ)
- “YYQw. 出力形式” (233 ページ)
- “YYQZw. 出力形式” (240 ページ)

YYQZw. 出力形式

SAS 日付値を<*yy*><*qq*>形式で書き出します。年は 2 桁または 4 桁の数字で書き出され、*qq* は四半期を表します。

カテゴリ: 日付と時間

配置: 右

構文

YYQZ*w*.

必須引数

Z
区切り文字を年と四半期の間に挿入しないことを示します。

w
出力フィールドの幅を指定します。

デフォルト 4

範囲 4-6

詳細

YYQZ*w*.出力形式は、SAS 日付値を<*yy*> <*qq*>形式で書き出します。構文の説明は次のとおりです。

<*yy*>
年を表す 2 桁または 4 桁の整数です。

Z
区切り文字がないことを示します。

<*qq*>
四半期を表す整数(01、02、03、04 のいずれか)です。

例

次の例は、YYQZ*w*.出力形式を示しています。例では、20999 を入力値として使用します。この値は、2017 年 6 月 29 日に相当する SAS 日付値です。

ステートメント	結果
select put (20999,yyqz6.);	201702
select put (20999,yyqz4.);	1702

関連項目

出力形式:

- “YYQw. 出力形式” (233 ページ)
- “YYQxw. 出力形式” (234 ページ)
- “YYQRw. 出力形式” (236 ページ)
- “YYQRxw. 出力形式” (238 ページ)

YYWEEKUw. 出力形式

U アルゴリズムを使用して、曜日情報を除いて 10 進数の週番号を書き出します。

カテゴリ: 日付と時間

構文

YYWEEKU*w*

必須引数

w
出力フィールドの幅を指定します。

デフォルト 7

範囲 2-8

詳細

YYWEEKU*w*.出力形式では、週番号形式で書き出します。YYWEEKU*w*.出力形式では、指定した幅に応じてさまざまな形式を書き出します。U アルゴリズムでは、年内の週番号を使用して SAS 日付値を計算します(日曜日を週の最初の日と見なします)。

幅、出力形式、および例の詳細については、次の表を参照してください。

幅	出力形式	例
3-4	Www	W01
5-6	yyWww	07W01
7	yyyyWww	2007W01
8	yyyy-Www	2007-W01
9 以上	無効	無効

比較

YYWEEKU*w*.出力形式は WEEKU*w*.出力形式と似ていますが、YYWEEKU*w*.出力形式では曜日情報を示しません。また、YYWEEKU*w*.出力形式では、8 より大きい幅は受け付けません。

例

次の例は、YYWEEKU*w*.出力形式を示しています。例では、20999 を入力値として使用します。この値は、2017 年 6 月 29 日に相当する SAS 日付値です。

ステートメント	結果
select put(20999,yyweeku5.);	17W26
select put(20999,yyweeku6.);	17W26
select put(20999,yyweeku7.);	2017W26
select put(20999,yyweeku8.);	2017-W26

YYWEEKVw. 出力形式

V アルゴリズムを使用して、曜日情報を除いて 10 進数の週番号を書き出します。

カテゴリ: 日付と時間

構文

YYWEEKV*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 7

範囲 2-8

詳細

YYWEEKV*w*.出力形式では、指定された幅に応じてさまざまな形式で書き込みます。V アルゴリズムでは、SAS 日付値を計算します。週番号値は、先頭に 0 を付けた 01 から 53 の範囲の 10 進数として表され、最大値が 53 になります。各週は月曜日から始まり、年の第 1 週は 1 月 4 日と年の最初の木曜日の両方を含む週です。1 月の最初の月曜日が 2 日、3 日または 4 日の場合、それより前の日は前年の最後の週に組み込まれます。たとえば、年の 5 週目は 06 として表されます。

幅、出力形式および例の詳細については、次の表を参照してください。

幅	出力形式	例
3-4	www	w01
5-6	yyWww	07W01
7	yyyyWww	2007W01
8	yyyy-Www	2007-W01
9-above	無効	無効

比較

YYWEEKV*w*.出力形式は WEEKV*w*.出力形式と似ていますが、YYWEEKV*w*.出力形式では曜日情報を示しません。また、YYWEEKV*w*.出力形式では、8 より大きい幅は受け付けません。

例

次の例は、YYWEEKV*w*.出力形式を示しています。例では、20999 を入力値として使用します。この値は、2017 年 6 月 29 日に相当する SAS 日付値です。

ステートメント	結果
select put(20999,yyweekv3.);	W26
select put(20999,yyweekv4.);	W26
select put(20999,yyweekv5.);	17W26
select put(20999,yyweekv6.);	17W26
select put(20999,yyweekv7.);	2017W26

ステートメント	結果
select put(20999,yyweekv8.);	2017-W26

YYWEEKWw. 出力形式

W アルゴリズムを使用して、曜日情報を除いて 10 進数の週番号を書き出します。

カテゴリ: 日付と時間

配置: 左

構文

YYWEEKW*w*.

必須引数

w
出力フィールドの幅を指定します。

デフォルト 7

範囲 2-8

詳細

YYWEEKW*w*.出力形式では、指定された幅に応じてさまざまな形式で書き込みます。W アルゴリズムでは、年内の週番号を使用して SAS 日付値を計算します。

幅、出力形式および例の詳細については、次の表を参照してください。

幅	出力形式	例
3-4	Www	W01
5-6	yyWww	07W01
7	yyyyWww	2007W01
8	yyyy-Www	2007-W01
9-above	無効	無効

比較

YYWEEKW*w*.出力形式は WEEKW*w*.出力形式と似ていますが、YYWEEKW*w*.出力形式では曜日情報を示しません。また、YYWEEKW*w*.出力形式では、8 より大きい幅は受け付けません。

例

次の例は、YYWEEKW*w*.出力形式を示しています。例では、20999 を入力値として使用します。この値は、2017 年 6 月 29 日に相当する SAS 日付値です。

ステートメント	結果
select put(20999,yyweekw3.);	W26
select put(20999,yyweekw4.);	W26
select put(20999,yyweekw5.);	17W26
select put(20999,yyweekw6.);	17W26
select put(20999,yyweekw7.);	2017W26
select put(20999,yyweekw8.);	2017-W26

Zw.d 出力形式

先頭に 0 を含む標準数値データを書き出します。

カテゴリ: 数値

配置: 右

操作: DECIMALCONV=システムオプションを STDIEEE に設定した場合、この出力形式を使用して書き出される出力が前のリリースと多少異なる可能性があります。詳細については、“DECIMALCONV= システムオプション” (*SAS システムオプション: リファレンス*)を参照してください。

構文

Z*w*.[*d*]

必須引数

w
出力フィールドの幅を指定します。

デフォルト 1

範囲 1-32

ヒント 必要に応じて、値、小数点、マイナス記号を書き出せる十分なスペースを指定します。

オプション引数

d
数値の小数点以下の桁数を指定します。

デフォルト 0

範囲 0-31

ヒント *d* が 0 の場合、または *d* の指定を省略する場合、Z*w.d* は値を小数点なしで書き出します。

詳細

Z*w.d* 出力形式は、標準数値(1 バイト 1 桁)を書き出し、データ値の左側に 0 を入力します。

Z*w.d* 出力形式は、出力フィールドにおさまる近傍値になるように値を丸めます。*w.d* が大きすぎておさまらない場合、小数点揃えを BESTw.出力形式に合わせます。Z*w.d* 出力形式は、負の数には先頭のマイナス記号を付けて書き出します。また、出力を右揃えしてから書き出し、先頭にゼロを挿入します。

例

次の例は、Z*w.d* 出力形式を示しています。

ステートメント	結果
select put(1350,z8.);	00001350

4

FedSQL 関数

FedSQL 関数の概要 **256**

一般的な関数構文 **257**
集計関数の構文 257
非集計関数の構文 257

FedSQL 関数の使用 **258**
関数の引数の制限 258
式での DS2 パッケージの使用 258
FedSQL での文字型を返す関数の使用 258

集計関数 **260**
集計関数の概要 260
FedSQL 集計関数のかわりに Base SAS 関数を呼び出す 260

関数カテゴリ **262**

SAS Output Delivery System と FedSQL **262**

カテゴリ別 FEDSQL 関数 **263**

ディクショナリ **281**
ABS 関数 281
AIRY 関数 282
ANYALNUM 関数 283
ANYALPHA 関数 285
ANYDIGIT 関数 287
ANYFIRST 関数 288
ANYLOWER 関数 291
ANYNAME 関数 292
ANYPUNCT 関数 294
ANYSPACE 関数 296
ANYUPPER 関数 298
ANYXDIGIT 関数 300
ARCOS 関数 302
ARCOSH 関数 303
ARSIN 関数 304
ARSINH 関数 305
ARTANH 関数 307
ATAN 関数 308
ATAN2 関数 309

AVG 関数 311
BAND 関数 313
BETA 関数 314
BETAINV 関数 315
BLACKCLPRC 関数 317
BLACKPTPRC 関数 319
BLKSHCLPRC 関数 321
BLKSHPTPRC 関数 324
BLSHIFT 関数 326
BNOT 関数 327
BOR 関数 328
BRSHIFT 関数 329
BXOR 関数 330
BYTE 関数 331
CAST 関数 332
CDF 関数 334
CDF BERNOULLI 分布 関数 337
CDF BETA 分布 関数 338
CDF BINOMIAL 分布 関数 340
CDF CAUCHY 分布 関数 342
CDF カイ 2 乗分布 関数 344
CDF Conway-Maxwell-Poisson 分布 関数 346
CDF 指数分布 関数 347
CDF F 分布 関数 349
CDF GAMMA 分布 関数 351
CDF 一般化 Poisson 分布 関数 352
CDF GEOMETRIC 分布 関数 354
CDF HYPERGEOMETRIC 分布 関数 355
CDF LAPLACE 分布 関数 357
CDF LOGISTIC 分布 関数 359
CDF LOGNORMAL 分布 関数 361
CDF NEGBINOMIAL 分布 関数 362
CDF NORMAL 分布 関数 364
CDF NORMALMIX 分布 関数 366
CDF PARETO 分布 関数 368
CDF POISSON 分布 関数 369
CDF T 分布 関数 371
CDF TWEEDIE 分布 関数 372
CDF UNIFORM 分布 関数 375
CDF WALD (逆ガウス)分布 関数 376
CDF WEIBULL 分布 関数 378
CEIL 関数 380
CEILZ 関数 381
CHARACTER_LENGTH 関数 382
CNONCT 関数 384
COALESCE 関数 386
COALESCEC 関数 387
COMB 関数 389
COMPARE 関数 390
COMPBL 関数 394
COMPOUND 関数 395
COMPRESS 関数 396
CONSTANT 関数 400
CONVX 関数 405

CONVXP 関数 407
COS 関数 409
COSH 関数 410
COT 関数 411
COUNT 関数 413
COUNTC 関数 414
COUNTW 関数 418
CSC 関数 422
CSS 関数 423
CUMIPMT 関数 425
CUMPRINC 関数 426
CURRENT_DATE 関数 428
CURRENT_LOCALE 関数 429
CURRENT_TIME 関数 429
CURRENT_TIME_GMT 関数 431
CURRENT_TIMESTAMP 関数 432
CURRENT_TIMESTAMP_GMT 関数 433
CV 関数 434
DAIRY 関数 435
DATDIF 関数 436
DATE 関数 439
DATEJUL 関数 440
DATEPART 関数 441
DATETIME 関数 442
DAY 関数 443
DEGREES 関数 444
DEQUOTE 関数 445
DEVIANCE 関数 447
DHMS 関数 451
DIGAMMA 関数 453
DUR 関数 454
DURP 関数 456
E 関数 458
EFFRATE 関数 459
ERF 関数 460
ERFC 関数 461
EXP 関数 462
FACT 関数 464
FINANCE 関数 465
FINANCE ACCRINT 関数 469
FINANCE ACCRINTM 関数 471
FINANCE AMORDEGRC 関数 472
FINANCE AMORLINC 関数 474
FINANCE COUPDAYBS 関数 476
FINANCE COUPDAYS 関数 477
FINANCE COUPDAYSNC 関数 479
FINANCE COUPNCD 関数 480
FINANCE COUPNUM 関数 482
FINANCE COUPPCD 関数 483
FINANCE CUMIPMT 関数 485
FINANCE CUMPRINC 関数 486
FINANCE DB 関数 487
FINANCE DDB 関数 489
FINANCE DISC 関数 490

FINANCE DOLLARDE 関数 491
FINANCE DOLLARFR 関数 492
FINANCE DURATION 関数 493
FINANCE EFFECT 関数 495
FINANCE FV 関数 495
FINANCE FVSCHEDULE 関数 497
FINANCE INTRATE 関数 498
FINANCE IPMT 関数 499
FINANCE IRR 関数 501
FINANCE ISPMT 関数 501
FINANCE MDURATION 関数 502
FINANCE MIRR 関数 504
FINANCE NOMINAL 関数 505
FINANCE NPER 関数 506
FINANCE NPV 関数 507
FINANCE ODDFPRICE 関数 508
FINANCE ODDFYIELD 関数 510
FINANCE ODDLPRICE 関数 512
FINANCE ODDLYIELD 関数 514
FINANCE PMT 関数 516
FINANCE PPMT 関数 517
FINANCE PRICE 関数 519
FINANCE PRICEDISC 関数 521
FINANCE PRICEMAT 関数 522
FINANCE PV 関数 524
FINANCE RATE 関数 525
FINANCE RECEIVED 関数 526
FINANCE SLN 関数 528
FINANCE SYD 関数 529
FINANCE TBILLEQ 関数 530
FINANCE TBILLPRICE 関数 531
FINANCE TBILLYIELD 関数 532
FINANCE VDB 関数 533
FINANCE XIRR 関数 534
FINANCE XNPV 関数 535
FINANCE YIELD 関数 537
FINANCE YIELDDISC 関数 538
FINANCE YIELDMAT 関数 540
FLOOR 関数 542
FLOORZ 関数 543
FMTINFO 関数 545
FNONCT 関数 546
FUZZ 関数 549
GAMINV 関数 550
GAMMA 関数 551
GARKHCLPRC 関数 552
GARKHPTPRC 関数 555
GCD 関数 557
GEOMEAN 関数 559
GEOMEANZ 関数 560
HARMEAN 関数 562
HARMEANZ 関数 564
HMS 関数 565
HOLIDAY 関数 567

HOUR 関数 . . . 570
IBESSEL 関数 . . . 571
IFNULL 関数 . . . 573
INDEX 関数 . . . 574
INDEXC 関数 . . . 576
INPUTC 関数 . . . 577
INPUTN 関数 . . . 579
INT 関数 . . . 580
INTCINDEX 関数 . . . 582
INTCK 関数 . . . 585
INTCYCLE 関数 . . . 591
INTFIT 関数 . . . 595
INTGET 関数 . . . 596
INTINDEX 関数 . . . 599
INTNX 関数 . . . 603
INTRR 関数 . . . 609
INTSEAS 関数 . . . 611
INTSHIFT 関数 . . . 615
INTTEST 関数 . . . 617
INTZ 関数 . . . 619
IPMT 関数 . . . 621
IQR 関数 . . . 623
IRR 関数 . . . 624
JBESSEL 関数 . . . 625
JULDATE 関数 . . . 626
JULDATE7 関数 . . . 628
KURTOSIS 関数 . . . 629
LARGEST 関数 . . . 631
LCM 関数 . . . 632
LCOMB 関数 . . . 633
LEFT 関数 . . . 635
LFACT 関数 . . . 636
LGAMMA 関数 . . . 637
LOG 関数 . . . 638
LOG1PX 関数 . . . 639
LOG2 関数 . . . 640
LOG10 関数 . . . 641
LOGBETA 関数 . . . 642
LOGCDF 関数 . . . 643
LOGISTIC 関数 . . . 646
LOGPDF 関数 . . . 647
LOGSDF 関数 . . . 649
LOWCASE 関数 . . . 652
MAD 関数 . . . 653
MAKEDATE 関数 . . . 654
MAKETIME 関数 . . . 655
MAKETIMESTAMP 関数 . . . 656
MARGRCLPRC 関数 . . . 657
MARGRPTPRC 関数 . . . 660
MAX 関数 . . . 663
MDY 関数 . . . 664
MEAN 関数 . . . 666
MEDIAN 関数 . . . 667
MIN 関数 . . . 668

MINUTE 関数 . . . 670
MOD 関数 . . . 671
MODZ 関数 . . . 673
MONTH 関数 . . . 676
MORT 関数 . . . 677
N 関数 . . . 679
NETPV 関数 . . . 680
NMISS 関数 . . . 682
NOMRATE 関数 . . . 684
NOTALNUM 関数 . . . 685
NOTALPHA 関数 . . . 687
NOTDIGIT 関数 . . . 689
NOTFIRST 関数 . . . 691
NOTLOWER 関数 . . . 693
NOTNAME 関数 . . . 695
NOTPUNCT 関数 . . . 697
NOTSPACE 関数 . . . 699
NOTUPPER 関数 . . . 701
NOTXDIGIT 関数 . . . 703
NPV 関数 . . . 704
NWKDOM 関数 . . . 706
OCTET_LENGTH 関数 . . . 708
ORDINAL 関数 . . . 710
PCTL 関数 . . . 711
PDF 関数 . . . 712
PDF BERNOULLI 分布 関数 . . . 715
PDF BETA 分布 関数 . . . 716
PDF BINOMIAL 分布 関数 . . . 718
PDF CAUCHY 分布 関数 . . . 720
PDF カイ 2 乗分布 関数 . . . 721
PDF Conway-Maxwell-Poisson 分布 関数 . . . 723
PDF EXPONENTIAL 分布 関数 . . . 726
PDF F 分布 関数 . . . 727
PDF GAMMA 分布 関数 . . . 729
PDF 一般化 Poisson 分布 関数 . . . 731
PDF GEOMETRIC 分布 関数 . . . 732
PDF 超幾何分布 関数 . . . 734
PDF LAPLACE 分布 関数 . . . 736
PDF LOGISTIC 分布 関数 . . . 737
PDF LOGNORMAL 分布 関数 . . . 739
PDF NEGBINOMIAL 分布 関数 . . . 740
PDF NORMAL 分布 関数 . . . 742
PDF NORMALMIX 分布 関数 . . . 743
PDF PARETO 分布 関数 . . . 745
PDF POISSON 分布 関数 . . . 747
PDF T 分布 関数 . . . 748
PDF TWEEDIE 分布 関数 . . . 750
PDF UNIFORM 分布 関数 . . . 752
PDF Wald (逆ガウス)分布 関数 . . . 754
PDF WEIBULL 分布 関数 . . . 755
PERM 関数 . . . 757
PI 関数 . . . 758
PMT 関数 . . . 759
POISSON 関数 . . . 761

POWER 関数 762
PROBBETA 関数 763
PROBBNML 関数 764
PROBBNRM 関数 765
PROBCHI 関数 767
PROBF 関数 768
PROBGAM 関数 769
PROBHYPR 関数 770
PROBIT 関数 772
PROBMC 関数 773
PROBMED 関数 783
PROBNEGB 関数 785
PROBNORM 関数 786
PROBT 関数 787
PUT 関数 788
PVP 関数 790
QTR 関数 792
QUANTILE 関数 793
QUOTE 関数 797
RADIANS 関数 798
RANGE 関数 799
RANK 関数 801
REPEAT 関数 802
REVERSE 関数 803
RMS 関数 804
ROUND 関数 805
ROUNDZ 関数 808
SAVING 関数 810
SAVINGS 関数 811
SCAN 関数 814
SDF 関数 819
SEC 関数 823
SECOND 関数 824
SIGN 関数 825
SIN 関数 826
SINH 関数 827
SKEWNESS 関数 829
SMALLEST 関数 830
SQUANTILE 関数 832
SQRT 関数 835
STD 関数 836
STDDEV 関数 837
STDERR 関数 838
STUDENTS_T 関数 840
SUBSTR 関数 842
SUBSTRING 関数 844
SUM 関数 846
SUMABS 関数 847
SYSGET 関数 848
TAN 関数 849
TANH 関数 850
TIMEPART 関数 851
TIMEVALUE 関数 852
TINV 関数 854

TNONCT 関数 . . . 856
TODAY 関数 . . . 858
TRANSLATE 関数 . . . 859
TRANSTRN 関数 . . . 861
TRANWRD 関数 . . . 863
TRIGAMMA 関数 . . . 865
TRIM 関数 . . . 866
TRUNC 関数 . . . 868
UPCASE 関数 . . . 869
URLDECODE 関数 . . . 870
URLENCODE 関数 . . . 872
USS 関数 . . . 873
UUIDGEN 関数 . . . 875
VAR 関数 . . . 876
VARIANCE 関数 . . . 877
VERIFY 関数 . . . 878
WEEK 関数 . . . 880
WEEKDAY 関数 . . . 883
WHICHC 関数 . . . 884
WHICHN 関数 . . . 885
YEAR 関数 . . . 887
YIELDP 関数 . . . 888
YRDIF 関数 . . . 890
YYQ 関数 . . . 893

FedSQL 関数の概要

FedSQL 関数は、FedSQL 式に対して計算を実行し、FedSQL 式が列の場合は単一の値または値のセットを返します。列に対して計算を実行する関数は*集計関数*です。他の SQL 環境では、集計関数は集合関数とも呼ばれます。他のほとんどの関数は、ユーザーが指定した引数を使用しますが、動作環境から引数を取得する関数もあります。

FedSQL 関数は、FedSQL アプリケーション、または埋め込み FedSQL が DS2 アプリケーションの一部である DS2 プログラムで使用できます。詳細については、“FedSQL 式” (46 ページ)および “SET ステートメント” (*SAS DS2 言語リファレンス*) を参照してください。

関数式の引数のデータ型が FedSQL 関数で期待されるものではない場合、FedSQL は引数に対して型変換を実行して、適切なデータ型になるようにします。型変換が成功すると、関数が実行されます。それ以外の場合は、エラーが発行されます。詳細については、“型変換” (26 ページ)を参照してください。

注: Base SAS で同様の名前が付けられている FedSQL 関数の場合、これらの関数は DOUBLE および文字引数を取ります。FedSQL は、ANSI 規格の日時値を含むすべての数値入力を DOUBLE に変換します。日時値は、SAS 日時値になります。FedSQL はすべての文字入力を CHAR に変換します。文字の戻り値は NVARCHAR です。ただし、数値の戻り値は DOUBLE のままです。特に、日付、時間、および日時の値は、SAS の日付、時間、および日時の値のままです。

CAST 関数を使用して、数値出力を別のデータ型にキャストできます。たとえば、SAS 関数 INTNX は、SAS 日時値を出力として返します。次のように、CAST 関数演算子::を使用して、INTNX 関数の出力を ANSI DATE または ANSI TIMESTAMP データ型に変換できます。

```
select intnx('month', date'2013-01-01', 5, 'beginning'):: date
select intnx('month', date'2013-01-01', 5, 'beginning'):: timestamp
```

一般的な関数構文

集計関数の構文

集計関数の構文は次のとおりです。

aggregate-function (*expression*)

aggregate-function
関数の名前を指定します。

expression
列名に評価される式を指定します。式には、次のいずれかの形式を含めることができます。

table.column
FedSQL ステートメントによって処理されているテーブルを識別する修飾列であり、その後にピリオドと列の名前が続きます。

例 select avg(densities.density)from densities;

column
暗黙的なテーブル修飾子を持つ修飾されていない列名です。暗黙的なテーブル修飾子は、FedSQL が処理しているテーブルです。

例 select avg(density)from densities;

注: 一部の集計関数では、複数の引数を指定できます。その場合、Base SAS 関数が使用され、関数は集計関数と見なされなくなります。詳細については、“FedSQL 集計関数のかわりに Base SAS 関数を呼び出す” (261 ページ)を参照してください。

非集計関数の構文

関数の構文は次のとおりです。

function (*argument* [, ...*argument*])

function
関数の名前を指定します。

argument
変数名、定数、または別の関数を含む任意の FedSQL 式にすることができます。FedSQL で許可される引数の数と種類は個別の関数と併せて記載しています。複数の引数はカンマで区切ります。

例 select sqrt(1500);

select ifnull(AvgHigh, 0) from worldtemps;

FedSQL 関数の使用

関数の引数の制限

引数の値が無効な場合、FedSQL は結果を null または欠損値に設定します。関数の引数に関する一般的な制限は次のとおりです。

- 一部の関数では、引数を特定の範囲内に制限する必要があります。たとえば、LOG 関数の引数は 0 より大きくする必要があります。
- ほとんどの関数は、null または欠損値を引数として許可しません。一部の記述統計関数と IFNULL 関数は例外です。
- 一般に、引数の許容範囲は、EXP 関数など、プラットフォームによって異なります。

式での DS2 パッケージの使用

DS2 パッケージメソッドを FedSQL SELECT ステートメントの関数として呼び出すことができます。式の構文を次に示します。

[*catalog*.] [*schema*.] *package.method* (*argument-1* [,... *argument-n*])

メソッドパラメーターには、DOUBLE、DATE、TIME、TIMESTAMP、任意のサイズの整数、文字列のいずれかを指定できます。

FedSQL の式で DS2 パッケージを使用する場合、次の制限が適用されます。

- メソッドは値を返す必要があります。
- パッケージ名を指定する必要があります。
- FedSQL SELECT ステートメントの引数の型と順序は、DS2 パッケージメソッドのパラメーターの型と順序と一致する必要があります。
- PROC FEDSQL から実行されるパッケージメソッドは、メソッドに入力引数のみを持つことができます。

次の例では、メソッド **BAR** が DS2 で作成され、FedSQL SELECT ステートメント呼び出しで関数として使用されます。

```
/* create table */
data dataset;
  x=1; y=1; z=1; output;
  x=1; y=1; z=2; output;
  x=2; y=2; z=2; output;
  x=2; y=2; z=8; output;
  x=2; y=3; z=13; output;
run;

/*  DS2 code */
proc ds2;
  package pkga;
    method bar(double x, double y) returns double;
      return x*x + y*y;
    end;
  endpackage;
 run;
quit;

/* fedsql code */
proc fedsql;
  select pkga.bar(1,2) as five,
    cot(radians(45)) as one,
    degrees(pi()) as one_eighty,
    power(3,4) as eighty_one,
    sign(-42) as minus1,
    sign(0) as zero,
    sign(42) as plus1;
    select * from dataset where pkga.bar(x,y) = z;
quit;
```

FedSQL SELECT ステートメントからの出力は次のようになります。

FIVE	ONE	ONE_EIGHTY	EIGHTY_ONE	MINUS1	ZERO	PLUS1
5	1	180	81	-1	0	1

x	y	z
1	1	2
2	2	8
2	3	13

詳細については、*“パッケージメソッド式” (SAS DS2 プログラマガイド)*を参照してください。

FedSQL での文字型を返す関数の使用

文字型を返す FedSQL 関数は、出力文字列が 20 文字を超える場合、最大列サイズを使用する場合があります。このような場合、次のような情報が SAS ログに書き込まれます。

NOTE: Column UUIDGEN has a maximum length of 131068, which is larger than the largest supported by PROC FEDSQL (32767).Some data might be truncated.

必要な戻り値が 32767 文字未満であることがわかっている場合、この情報は無視できます。マルチバイト文字セットを使用する文字エンコーディングで関数を実行する場合も無視できます。この情報を非表示するには、文字型を固定文字長列で返す関数を実行します。または、CAST 関数を使用して文字長を設定します。

集計関数

集計関数の概要

FedSQL 集計関数は、テーブル列内の式のすべての値を操作し、単一の結果を返します。集計関数が GROUP BY ステートメントで処理される場合、集計関数はグループごとに単一の結果を返します。COUNT 関数の COUNT(*)構文を除き、null 値と SAS 欠損値は演算では考慮されません。関数で指定するテーブル列は、列名に評価される任意の FedSQL 式にすることができます。

次は、FedSQL 集計関数です。

- “AVG 関数” (311 ページ)
- “COUNT 関数” (413 ページ)
- “CSS 関数” (423 ページ)
- “KURTOSIS 関数” (629 ページ)
- “MAX 関数” (663 ページ)
- “MIN 関数” (668 ページ)

- “NMISS 関数” (682 ページ)
- “PROBT 関数” (787 ページ)
- “RANGE 関数” (799 ページ)
- “SKEWNESS 関数” (829 ページ)
- “STD 関数” (836 ページ)
- “STDDEV 関数” (837 ページ)
- “STDERR 関数” (838 ページ)
- “STUDENTS_T 関数” (840 ページ)
- “SUM 関数” (846 ページ)
- “USS 関数” (873 ページ)
- “VARIANCE 関数” (877 ページ)

注: 現在、MongoDB では集計関数はサポートされていません。

Salesforce では、限定された集計関数のセットがサポートされています。サポートされている関数には、AVG()、COUNT()、MIN()、MAX()、SUM()が含まれます。

デフォルトの SAS/ACCESS 設定(Salesforce SOAP API を使用)で Salesforce に接続すると、FedSQL パススルーを介して処理するために関数を Salesforce に渡すことができます。BULK_API=オプション(Salesforce Bulk API を呼び出す)でバルクロードが有効になっている場合、関数は SAS クライアントで処理されます。

以下は、テーブル WorldTemps の AvgLow 列における AVG、COUNT、および MAX 集計関数の機能を示す例です。

```
/* Get the average of the average low temperatures  */
select avg(AvgLow) as AvgTemp from worldtemps;

/* Get the number of different average low temperatures */
/* and group them by the average low temperature */
select AvgLow, count(AvgLow) from worldtemps group by AvgLow;

/* Get the highest average low temperature */
select max(AvgLow) from worldtemps;
```

FedSQL 集計関数のかわりに Base SAS 関数を呼び出す

集計関数の引数として複数の列が指定され、同様の名前の Base SAS 関数が存在する場合、Base SAS 関数が使用されます。これらの引数に対して計算される統計量は、現在の行のものです。関数は集計関数とは見なされなくなります。例としては、MIN、MAX、および SUM 関数があります。

集計関数に複数の引数が指定され、同様の名前の Base SAS 関数がない場合、エラーが返されます。例として、AVG 関数があります。

関数カテゴリ

関数は、操作する値の種類によって分類できます。各 FedSQL 関数は、次のいずれかのカテゴリに属します。

集計
: テーブル列の値を操作します。

ビットごとの論理演算
: 個々のビットのレベルで 1 つ以上のビットパターンまたは 2 進数を操作します。

文字
: 文字データと SQL 式を操作します。

日付と時間
: 日時 SQL 式を操作します。

記述統計
: 中心傾向、値間の変動、および分布値の形状を測定する値を操作します。

財務
: 株式のヨーロッパオプションの利息、定期支払い、減価償却、価格などの財務価値を計算します。

数学
: 値を操作して一般的な数学計算を実行します。

確率
: 確率計算を返します。

特殊
: null 値と SAS 欠損値を操作します。

三角関数
: 値を操作して三角関数の計算を実行します。

切り捨て
: 多くの場合ファジングを使用して、数値を切り捨てて数値を返します。

SAS Output Delivery System と FedSQL

SAS Output Delivery System (ODS)は、デフォルトで、PROC FEDSQL でサブミットされた関数からの数値出力を 8 文字に丸めます。FedSQL で可能な限り完全な精度で数値出力を表示するには、次のように BEST16.出力形式を指定して PUT 関数を使用します。

```
select PUT (beta(5,3), best16.) as Beta;
```

カテゴリ別 FEDSQL 関数

カテゴリ	言語要素	説明
Web ツール	URLDECODE 関数 (p. 870)	URL のエスケープ構文を使用してデコードされた文字列を返します。
	URLENCODE 関数 (p. 872)	URL のエスケープ構文を使用してエンコードされた文字列を返します。
確率	CDF 関数 (p. 334)	さまざまな連続確率分布および離散確率分布の左累積分布関数を計算します。
	CDF BERNOULLI 分布 関数 (p. 337)	Bernoulli 累積確率分布の値を返します。
	CDF BETA 分布 関数 (p. 338)	ベータ累積確率分布の値を返します。
	CDF BINOMIAL 分布 関数 (p. 340)	二項累積確率分布の値を返します。
	CDF CAUCHY 分布 関数 (p. 342)	Cauchy 累積確率分布の値を返します。
	CDF カイ 2 乗分布 関数 (p. 344)	カイ 2 乗累積確率分布の値を返します。
	CDF Conway-Maxwell-Poisson 分布 関数 (p. 346)	Conway-Maxwell-Poisson 累積確率分布の値を返します。
	CDF 指数分布 関数 (p. 347)	指数累積確率分布の値を返します。
	CDF F 分布 関数 (p. 349)	F 累積確率分布の値を返します。
	CDF GAMMA 分布 関数 (p. 351)	ガンマ累積確率分布の値を返します。
	CDF 一般化 Poisson 分布 関数 (p. 352)	一般化 Poisson 累積確率分布の値を返します。
	CDF GEOMETRIC 分布 関数 (p. 354)	幾何累積確率分布の値を返します。
	CDF HYPERGEOMETRIC 分布 関数 (p. 355)	超幾何累積確率分布の値を返します。
	CDF LAPLACE 分布 関数 (p. 357)	Laplace 累積確率分布の値を返します。

カテゴリ	言語要素	説明
	CDF LOGISTIC 分布 関数 (p. 359)	ロジスティック累積確率分布の値を返します。
	CDF LOGNORMAL 分布 関数 (p. 361)	対数正規累積確率分布の値を返します。
	CDF NEGBINOMIAL 分布 関数 (p. 362)	負の二項累積確率分布の値を返します。
	CDF NORMAL 分布 関数 (p. 364)	正規累積確率分布の値を返します。
	CDF NORMALMIX 分布 関数 (p. 366)	正規混合累積確率分布の値を返します。
	CDF PARETO 分布 関数 (p. 368)	パレート累積確率分布の値を返します。
	CDF POISSON 分布 関数 (p. 369)	Poisson 累積確率分布の値を返します。
	CDF T 分布 関数 (p. 371)	T 累積確率分布の値を返します。
	CDF TWEEDIE 分布 関数 (p. 372)	Tweedie 累積確率分布の値を返します。
	CDF UNIFORM 分布 関数 (p. 375)	一様累積確率分布の値を返します。
	CDF WALD (逆ガウス)分布 関数 (p. 376)	Wald (逆ガウスとも呼ばれる)累積確率分布から値を返します。
	CDF WEIBULL 分布 関数 (p. 378)	Weibull 累積確率分布の値を返します。
	LOGCDF 関数 (p. 643)	左累積分布関数の対数を返します。
	LOGPDF 関数 (p. 647)	さまざまな連続分布および離散分布から確率密度(質量)関数の対数を計算します。
	LOGSDF 関数 (p. 649)	生存関数の対数を返します。
	PDF 関数 (p. 712)	確率密度(質量)分布の値を返します。
	PDF BERNOULLI 分布 関数 (p. 715)	Bernoulli 確率密度(質量)分布の値を返します。
	PDF BETA 分布 関数 (p. 716)	ベータ確率密度(質量)分布の値を返します。
	PDF BINOMIAL 分布 関数 (p. 718)	二項確率密度(質量)分布の値を返します。
	PDF CAUCHY 分布 関数 (p. 720)	Cauchy 確率密度(質量)分布の値を返します。

カテゴリ	言語要素	説明
	PDF カイ 2 乗分布 関数 (p. 721)	カイ 2 乗確率密度(質量)分布の値を返します。
	PDF Conway-Maxwell-Poisson 分布 関数 (p. 723)	Conway-Maxwell-Poisson 確率密度(質量)分布の値を返します。
	PDF EXPONENTIAL 分布 関数 (p. 726)	指数確率密度(質量)分布の値を返します。
	PDF F 分布 関数 (p. 727)	F 確率密度(質量)分布の値を返します。
	PDF GAMMA 分布 関数 (p. 729)	ガンマ確率密度(質量)分布の値を返します。
	PDF 一般化 Poisson 分布 関数 (p. 731)	一般化 Poisson 確率密度(質量)分布の値を返します。
	PDF GEOMETRIC 分布 関数 (p. 732)	幾何確率密度(質量)分布の値を返します。
	PDF 超幾何分布 関数 (p. 734)	超幾何確率密度(質量)分布の値を返します。
	PDF LAPLACE 分布 関数 (p. 736)	Laplace 確率密度(質量)分布の値を返します。
	PDF LOGISTIC 分布 関数 (p. 737)	ロジスティック確率密度(質量)分布の値を返します。
	PDF LOGNORMAL 分布 関数 (p. 739)	対数正規確率密度(質量)分布の値を返します。
	PDF NEGBINOMIAL 分布 関数 (p. 740)	負の二項確率密度(質量)分布の値を返します。
	PDF NORMAL 分布 関数 (p. 742)	正規確率密度(質量)分布の値を返します。
	PDF NORMALMIX 分布 関数 (p. 743)	正規混合確率密度(質量)分布の値を返します。
	PDF PARETO 分布 関数 (p. 745)	パレート確率密度(質量)分布の値を返します。
	PDF POISSON 分布 関数 (p. 747)	Poisson 確率密度(質量)分布の値を返します。
	PDF T 分布 関数 (p. 748)	T 確率密度(質量)分布の値を返します。
	PDF TWEEDIE 分布 関数 (p. 750)	Tweedie 確率密度(質量)分布の値を返します。
	PDF UNIFORM 分布 関数 (p. 752)	一様確率密度(質量)分布の値を返します。

カテゴリ	言語要素	説明
	PDF Wald (逆ガウス)分布 関数 (p. 754)	Wald (逆ガウスとも呼ばれる)確率密度(質量)分布から値を返します。
	PDF WEIBULL 分布 関数 (p. 755)	Weibull 確率密度(質量)分布の値を返します。
	POISSON 関数 (p. 761)	Poisson 分布の確率を返します。
	PROBBETA 関数 (p. 763)	ベータ分布の確率を返します。
	PROBBNML 関数 (p. 764)	二項分布の確率を返します。
	PROBBNRM 関数 (p. 765)	2 変量正規分布の確率を返します。
	PROBCHI 関数 (p. 767)	カイ 2 乗分布の確率を返します。
	PROBF 関数 (p. 768)	F 分布の確率を返します。
	PROBGAM 関数 (p. 769)	ガンマ分布の確率を返します。
	PROBHYPR 関数 (p. 770)	超幾何分布の確率を返します。
	PROBMC 関数 (p. 773)	平均値の多重比較を行うために各種の分布から確率または分位点を返します。
	PROBMED 関数 (p. 783)	サンプル中央値の累積確率を計算します。
	PROBNEGB 関数 (p. 785)	負の二項分布の確率を返します。
	PROBNORM 関数 (p. 786)	標準正規分布の確率を返します。
	SDF 関数 (p. 819)	生存関数を返します。
記述統計	AVG 関数 (p. 311)	列内のすべての値の平均を返します。
	CSS 関数 (p. 423)	式のすべての値の修正済み平方和を返します。
	CV 関数 (p. 434)	変動係数を返します。
	GEOMEAN 関数 (p. 559)	幾何平均を返します。
	GEOMEANZ 関数 (p. 560)	ゼロファジーを使用して、幾何平均を返します。
	HARMEAN 関数 (p. 562)	調和平均を返します。
	HARMEANZ 関数 (p. 564)	ゼロファジーを使用して、調和平均を返します。
	IQR 関数 (p. 623)	四分位範囲を返します。
	KURTOSIS 関数 (p. 629)	式のすべての値の尖度を返します。
	LARGEST 関数 (p. 631)	k 番目に大きい非 null または非欠損値を返します。

カテゴリ	言語要素	説明
	MAD 関数 (p. 653)	中央値からの中央絶対偏差を返します。
	MAX 関数 (p. 663)	列の最大値を返します。
	MEAN 関数 (p. 666)	非 null または非欠損引数の算術平均(平均)を返します。
	MEDIAN 関数 (p. 667)	中央値を返します。
	MIN 関数 (p. 668)	式の最小値を返します。
	ORDINAL 関数 (p. 710)	値のリストを並べ替え、リスト内の位置に基づく値を返します。
	PCTL 関数 (p. 711)	パーセントに対応するパーセント点を返します。
	PROBT 関数 (p. 787)	式の値の t 分布から確率を返します。
	RANGE 関数 (p. 799)	式の値の間の範囲を返します。
	RMS 関数 (p. 804)	2 乗平均平方根を返します。
	SKEWNESS 関数 (p. 829)	式のすべての値の歪度を返します。
	SMALLEST 関数 (p. 830)	k 番目に小さい非 null 値または非欠損値を返します。
	STD 関数 (p. 836)	標準偏差を返します。
	STDDEV 関数 (p. 837)	式のすべての値の統計標準偏差を返します。
	STDERR 関数 (p. 838)	式のすべての値の統計標準誤差を返します。
	STUDENTS_T 関数 (p. 840)	式の値のスチューデントの t 分布を返します。
	SUM 関数 (p. 846)	式のすべての値の合計を返します。
	SUMABS 関数 (p. 847)	非欠損引数の絶対値の合計を返します。
	USS 関数 (p. 873)	式のすべての値の無修正平方和を返します。
	VAR 関数 (p. 876)	分散を返します。
	VARIANCE 関数 (p. 877)	式のすべての値のばらつきの尺度を返します。
切り捨て	CEIL 関数 (p. 380)	数値式以上の最小の整数を返します。
	CEILZ 関数 (p. 381)	ゼロファジーを使用して、引数より大きいか等しい整数のうち最小の値を返します。
	FLOOR 関数 (p. 542)	数値式以下の最大の整数を返します。
	FLOORZ 関数 (p. 543)	ゼロファジーを使用して、引数より小さいか等しい整数のうち最大の値を返します。

カテゴリ	言語要素	説明
	FUZZ 関数 (p. 549)	引数が整数の 1E-12 以内の場合、最も近い整数を返します。
	INT 関数 (p. 580)	予期しない浮動小数点の結果を避けるためにファジー処理された整数を返します。
	INTZ 関数 (p. 619)	ゼロファジーを使用して、引数の整数部を返します。
	ROUND 関数 (p. 805)	第 1 引数を第 2 引数の最も近い倍数(第 2 引数が省略された場合は最も近い整数)に丸めます。
	ROUNDZ 関数 (p. 808)	第 1 引数を第 2 引数の最も近い倍数にゼロファジーを使用して丸めます。
	TRUNC 関数 (p. 868)	数値を指定した長さに切り捨てます。
組み合わせ	COMB 関数 (p. 389)	n 個の要素を同時に r 個使用するときの組み合わせの数を計算します。
	LCOMB 関数 (p. 633)	COMB 関数の対数を計算します。これは、n 個のオブジェクトから一度に r 個を取り出した組み合わせ数の対数です。
	LFACT 関数 (p. 636)	FACT(階乗)関数の対数を計算します。
	PERM 関数 (p. 757)	n 個の項目から一度に r 個の項目を抜き出す場合の順列数を計算します。
財務	BLACKCLPRC 関数 (p. 317)	Black モデルに基づき、先物のヨーロピアンオプションのコール価格を計算します。
	BLACKPTPRC 関数 (p. 319)	Black モデルに基づき、先物のヨーロピアンオプションのプット価格を計算します。
	BLKSHCLPRC 関数 (p. 321)	Black-Scholes モデルに基づき、株式のヨーロピアンオプションのコール価格を計算します。
	BLKSHPTPRC 関数 (p. 324)	Black-Scholes モデルに基づき、株式のヨーロピアンオプションのプット価格を計算します。
	COMPOUND 関数 (p. 395)	複利パラメーターを返します。
	CONVX 関数 (p. 405)	列挙キャッシュフローのコンベクシティを返します。
	CONVXP 関数 (p. 407)	債権などの定期キャッシュフローストリームのコンベクシティを返します。
	CUMIPMT 関数 (p. 425)	開始期間と終了期間の間でローンに対して支払われる累積利息を返します。
	CUMPRINC 関数 (p. 426)	開始期間と終了期間の間でローンに対して支払われる累積元本を返します。
	DUR 関数 (p. 454)	列挙キャッシュフローの修正デュレーションを返します。

カテゴリ	言語要素	説明
	DURP 関数 (p. 456)	債権などの定期キャッシュフローストリームの修正デュレーションを返します。
	EFFRATE 関数 (p. 459)	有効な年利を返します。
	FINANCE 関数 (p. 465)	減価償却、満期、未払い利息、正味現在価値、定期的預金、内部利益率などの財務計算を行います。
	FINANCE ACCRINT 関数 (p. 469)	定期的に利息を支払う証券の未払い利息を計算します。
	FINANCE ACCRINTM 関数 (p. 471)	満期に利息を支払う証券の未払い利息を計算します。
	FINANCE AMORDEGRC 関数 (p. 472)	減価償却係数を使用して各会計期間の減価償却を計算します。
	FINANCE AMORLINC 関数 (p. 474)	各会計期間の減価償却を計算します。
	FINANCE COUPDAYBS 関数 (p. 476)	クーポン期間の開始日から決済日までの日数を計算します。
	FINANCE COUPDAYS 関数 (p. 477)	決済日を含むクーポン期間の日数を計算します。
	FINANCE COUPDAYSNC 関数 (p. 479)	決済日から次のクーポン日までの日数を計算します。
	FINANCE COUPNCD 関数 (p. 480)	決済日後の次のクーポン日を計算します。
	FINANCE COUPNUM 関数 (p. 482)	決済日から満期日までの支払いクーポン数を計算します。
	FINANCE COUPPCD 関数 (p. 483)	決済日前の前のクーポン日を計算します。
	FINANCE CUMIPMT 関数 (p. 485)	2 つの期間の間に支払われる累積利息を計算します。
	FINANCE CUMPRINC 関数 (p. 486)	2 つの期間の間にローンで支払われる累積元本を計算します。
	FINANCE DB 関数 (p. 487)	定率法を使用して、指定期間の資産の減価償却を計算します。
	FINANCE DDB 関数 (p. 489)	倍額定率法またはその他の方法を指定して、指定期間の資産の減価償却を計算します。
	FINANCE DISC 関数 (p. 490)	証券の割引率を計算します。
	FINANCE DOLLARDE 関数 (p. 491)	分数で表すドル価格を小数で表すドル価格に変換します。

カテゴリ	言語要素	説明
	FINANCE DOLLARFR 関数 (p. 492)	小数で表すドル価格を分数で表すドル価格に変換します。
	FINANCE DURATION 関数 (p. 493)	定期的に利息を支払う証券の年度期間を計算します。
	FINANCE EFFECT 関数 (p. 495)	有効な年利を計算します。
	FINANCE FV 関数 (p. 495)	投資の将来価値を計算します。
	FINANCE FVSCHEDULE 関数 (p. 497)	一連の複利率を適用した後の当初元本の将来価値を計算します。
	FINANCE INTRATE 関数 (p. 498)	完全投資済み証券の利率を計算します。
	FINANCE IPMT 関数 (p. 499)	一定期間の投資の利払いを計算します。
	FINANCE IRR 関数 (p. 501)	一連のキャッシュフローの内部利益率を計算します。
	FINANCE ISPMT 関数 (p. 501)	投資の指定期間中に支払われる利息を計算します。
	FINANCE MDURATION 関数 (p. 502)	推定額面$100 の証券の修正マコーレーデュレーションを計算します。
	FINANCE MIRR 関数 (p. 504)	プラスおよびマイナスのキャッシュフローが異なる利率で資金調達される場合の内部利益率を計算します。
	FINANCE NOMINAL 関数 (p. 505)	年間の名目金利を計算します。
	FINANCE NPER 関数 (p. 506)	投資の期間数を計算します。
	FINANCE NPV 関数 (p. 507)	一連の定期的なキャッシュフローと割引率に基づいて投資の正味現在価値を計算します。
	FINANCE ODDFPRICE 関数 (p. 508)	端数の開始期間で額面$100 当たりの証券の価格を計算します。
	FINANCE ODDFYIELD 関数 (p. 510)	端数の開始期間で証券の利回りを計算します。
	FINANCE ODDLPRICE 関数 (p. 512)	端数の最終期間で額面$100 当たりの証券の価格を計算します。
	FINANCE ODDLYIELD 関数 (p. 514)	端数の最終期間で証券の利回りを計算します。
	FINANCE PMT 関数 (p. 516)	年金の定期的支払いを計算します。
	FINANCE PPMT 関数 (p. 517)	指定期間の投資の元本に対する支払いを計算します。

カテゴリ	言語要素	説明
	FINANCE PRICE 関数 (p. 519)	定期的に利息を支払う額面$100 当たりの証券の価格を計算します。
	FINANCE PRICEDISC 関数 (p. 521)	額面$100 当たりの割引証券の価格を計算します。
	FINANCE PRICEMAT 関数 (p. 522)	満期に利息を支払う額面$100 当たりの証券の価格を計算します。
	FINANCE PV 関数 (p. 524)	投資の現在価値を計算します。
	FINANCE RATE 関数 (p. 525)	年金期間当たりの利率を計算します。
	FINANCE RECEIVED 関数 (p. 526)	完全投資済み証券の満期受け取り額を計算します。
	FINANCE SLN 関数 (p. 528)	ある期間の資産の定額法による減価償却を計算します。
	FINANCE SYD 関数 (p. 529)	指定期間の資産の年次級数和法による減価償却を計算します。
	FINANCE TBILLEQ 関数 (p. 530)	米国財務省短期証券の債券換算利回りを計算します。
	FINANCE TBILLPRICE 関数 (p. 531)	額面$100 当たりの米国財務省短期証券の価格を計算します。
	FINANCE TBILLYIELD 関数 (p. 532)	米国財務省短期証券の利回りを計算します。
	FINANCE VDB 関数 (p. 533)	定率法を使用して、指定期間または一部期間の資産の減価償却を計算します。
	FINANCE XIRR 関数 (p. 534)	必ずしも定期的ではないキャッシュフロースケジュールの内部利益率を計算します。
	FINANCE XNPV 関数 (p. 535)	必ずしも定期的ではないキャッシュフロースケジュールの正味現在価値を計算します。
	FINANCE YIELD 関数 (p. 537)	定期的に利息を支払う証券の利回りを計算します。
	FINANCE YIELDDISC 関数 (p. 538)	割引証券(米国財務省短期証券など)の年間利回りを計算します。
	FINANCE YIELDMAT 関数 (p. 540)	満期に利息を支払う証券の年間利回りを計算します。
	GARKHCLPRC 関数 (p. 552)	Garman-Kohlhagen モデルに基づいて、株式のヨーロピアンオプションのコール価格を計算します。
	GARKHPTPRC 関数 (p. 555)	Garman-Kohlhagen モデルに基づいて、株式のヨーロピアンオプションのプット価格を計算します。

カテゴリ	言語要素	説明
	INTRR 関数 (p. 609)	内部利益率を小数値で返します。
	IPMT 関数 (p. 621)	将来の残高を達成するための、均等払いローンまたは定期預金に対する指定期間の利息の支払いを返します。
	IRR 関数 (p. 624)	内部利益率をパーセントで返します。
	MARGRCLPRC 関数 (p. 657)	Margrabe モデルに基づいて、株式のヨーロピアンオプションのコール価格を計算します。
	MARGRPTPRC 関数 (p. 660)	Margrabe モデルに基づいて、株式のヨーロピアンオプションのプット価格を計算します。
	MORT 関数 (p. 677)	割賦返済パラメーターを返します。
	NETPV 関数 (p. 680)	現在正味価値をパーセントで返します。
	NOMRATE 関数 (p. 684)	名目年利を返します。
	NPV 関数 (p. 704)	パーセントで表す利率を使用して現在正味価値を返します。
	PMT 関数 (p. 759)	均等払いローンまたは定期預金の将来残高に対する定期的支払いを返します。
	PVP 関数 (p. 790)	満期時の元本払い戻しで、定期的なキャッシュフローストリーム(債権など)の現在価値を返します。
	SAVING 関数 (p. 810)	定期預金の将来価値を返します。
	SAVINGS 関数 (p. 811)	変動金利を使用して定期預金の残高を返します。
	TIMEVALUE 関数 (p. 852)	変動金利を使用して、基準日の参照額に相当する額を返します。
	YIELDP 関数 (p. 888)	定期的なキャッシュフローストリーム(債権など)の最終利回りを返します。
三角関数	ARCOS 関数 (p. 302)	逆余弦をラジアンで返します。
	ARSIN 関数 (p. 304)	逆正弦をラジアンで返します。
	ATAN 関数 (p. 308)	逆正接をラジアンで返します。
	ATAN2 関数 (p. 309)	直角三角形の x 座標と y 座標の逆正接をラジアンで返します。
	COS 関数 (p. 409)	余弦をラジアンで返します。
	COSH 関数 (p. 410)	双曲線余弦をラジアンで返します。
	COT 関数 (p. 411)	正接をラジアンで返します。
	CSC 関数 (p. 422)	余割を返します。

カテゴリ	言語要素	説明
	DEGREES 関数 (p. 444)	角度の度数をラジアンで返します。
	RADIANS 関数 (p. 798)	数値の度数から変換されたラジアン数を返します。
	SEC 関数 (p. 823)	正割を返します。
	SIN 関数 (p. 826)	三角関数の正弦を返します。
	SINH 関数 (p. 827)	双曲線正弦を返します。
	TAN 関数 (p. 849)	正接を返します。
	TANH 関数 (p. 850)	双曲線正接を返します。
集計	AVG 関数 (p. 311)	列内のすべての値の平均を返します。
	COUNT 関数 (p. 413)	指定されたテーブルの SELECT ステートメントによって取得された行数を返します。
	CSS 関数 (p. 423)	式のすべての値の修正済み平方和を返します。
	KURTOSIS 関数 (p. 629)	式のすべての値の尖度を返します。
	MAX 関数 (p. 663)	列の最大値を返します。
	MIN 関数 (p. 668)	式の最小値を返します。
	NMISS 関数 (p. 682)	式内の null 値または SAS 欠損値の数を返します。
	PROBT 関数 (p. 787)	式の値の t 分布から確率を返します。
	RANGE 関数 (p. 799)	式の値の間の範囲を返します。
	SKEWNESS 関数 (p. 829)	式のすべての値の歪度を返します。
	STDDEV 関数 (p. 837)	式のすべての値の統計標準偏差を返します。
	STDERR 関数 (p. 838)	式のすべての値の統計標準誤差を返します。
	STUDENTS_T 関数 (p. 840)	式の値のスチューデントの t 分布を返します。
	SUM 関数 (p. 846)	式のすべての値の合計を返します。
	USS 関数 (p. 873)	式のすべての値の無修正平方和を返します。
	VARIANCE 関数 (p. 877)	式のすべての値のばらつきの尺度を返します。
数学	ABS 関数 (p. 281)	数値式の絶対値を返します。
	AIRY 関数 (p. 282)	Airy 関数の値を返します。
	BETA 関数 (p. 314)	beta 関数の値を返します。

カテゴリ	言語要素	説明
	CNONCT 関数 (p. 384)	カイ 2 乗分布の非心度パラメーターを返します。
	COALESCE 関数 (p. 386)	数値の引数のリストから最初の非 null または非欠損値を返します。
	CONSTANT 関数 (p. 400)	マシン定数および数学定数を計算します。
	DAIRY 関数 (p. 435)	AIRY 関数の導関数を返します。
	DEVIANCE 関数 (p. 447)	確率分布に基づくデビアンスを返します。
	DIGAMMA 関数 (p. 453)	ディガンマ関数の値を返します。
	E 関数 (p. 458)	自然対数 e を返します。
	ERF 関数 (p. 460)	(正規)誤差関数の値を返します。
	ERFC 関数 (p. 461)	相補(正規)誤差関数の値を返します。
	EXP 関数 (p. 462)	指定されたべき乗の e 定数の値を返します。
	FACT 関数 (p. 464)	階乗を計算します。
	FNONCT 関数 (p. 546)	F 分布の非心度パラメーターの値を返します。
	GAMMA 関数 (p. 551)	ガンマ関数の値を返します。
	GCD 関数 (p. 557)	整数の集合に対する最大公約数を返します。
	IBESSEL 関数 (p. 571)	変形ベッセル関数の値を返します。
	JBESSEL 関数 (p. 625)	ベッセル関数の値を返します。
	LCM 関数 (p. 632)	数の集合に対する最小公倍数を返します。
	LGAMMA 関数 (p. 637)	ガンマ関数の自然対数を返します。
	LOG 関数 (p. 638)	数値式の自然対数(底 e)を返します。
	LOG1PX 関数 (p. 639)	1 に引数を加えた値の対数を返します。
	LOG2 関数 (p. 640)	数値式の 2 を底とする対数を返します。
	LOG10 関数 (p. 641)	数値式の 10 を底とする対数を返します。
	LOGBETA 関数 (p. 642)	ベータ関数の対数を返します。
	LOGISTIC 関数 (p. 646)	引数のロジスティック変換を返します。
	MOD 関数 (p. 671)	第 1 引数を第 2 引数で除算したときの(最も期待しない浮動小数点の結果を避けるためにファジー化した)余りを返します。

カテゴリ	言語要素	説明
	MODZ 関数 (p. 673)	第 1 引数を第 2 引数で除算したときの余りを、ゼロファジーを使用して返します。
	PI 関数 (p. 758)	PI の定数値を浮動小数点値として返します。
	POWER 関数 (p. 762)	指定された累乗の数値式の値を返します。
	SIGN 関数 (p. 825)	数値式の符号を示す数字を返します。
	SQRT 関数 (p. 835)	値の平方根を返します。
	TNONCT 関数 (p. 856)	スチューデントの t 分布から非心度パラメーターの値を返します。
	TRIGAMMA 関数 (p. 865)	トリガンマ関数の値を返します。
	WHICHN 関数 (p. 885)	数値のリストから数値の最初の位置を返します。
スカラー	NMISS 関数 (p. 682)	式内の null 値または SAS 欠損値の数を返します。
双曲線	ARCOSH 関数 (p. 303)	逆双曲線余弦を返します。
	ARSINH 関数 (p. 305)	逆双曲線正弦を返します。
	ARTANH 関数 (p. 307)	逆双曲線正接を返します。
特殊	CAST 関数 (p. 332)	値をあるデータ型から別のデータ型に変換します。
	FMTINFO 関数 (p. 545)	SAS 出力形式または入力形式の情報を返します。
	IFNULL 関数 (p. 573)	最初の式の値をチェックし、null または SAS 欠損値の場合は、2 番目の式を返します。
	INPUTC 関数 (p. 577)	実行時に文字の入力形式を指定できるようにします。
	INPUTN 関数 (p. 579)	実行時に数値の入力形式を指定できるようにします。
	N 関数 (p. 679)	非 null または非欠損の数値の数を返します。
	PUT 関数 (p. 788)	指定した出力形式を使用して値を返します。
	SYSGET 関数 (p. 848)	指定された動作環境変数の値を返します。
	UUIDGEN 関数 (p. 875)	ユニバーサル一意識別子(UUID)の短い形式を返します。
日付と時間	CURRENT_DATE 関数 (p. 428)	タイムゾーンの現在の日付を返します。
	CURRENT_TIME 関数 (p. 429)	タイムゾーンの現在の時間を返します。
	CURRENT_TIME_GMT 関数 (p. 431)	現在の GMT 時間を返します。

カテゴリ	言語要素	説明
	CURRENT_TIMESTAMP 関数 (p. 432)	タイムゾーンの日時を返します。
	CURRENT_TIMESTAMP_GMT 関数 (p. 433)	現在の GMT 日時を返します。
	DATDIF 関数 (p. 436)	指定された日数計算規則に従って 2 つの日付間の差を計算した後に、その日付間の日数を返します。
	DATE 関数 (p. 439)	SAS 日付値として現在の日付を返します。
	DATEJUL 関数 (p. 440)	ユリウス暦の日付を SAS 日付値に変換します。
	DATEPART 関数 (p. 441)	日付を年、月、日として返します。
	DATETIME 関数 (p. 442)	SAS 日時値として現在の日時を返します。
	DAY 関数 (p. 443)	日付または日時値から月の日を数値で返します。
	DHMS 関数 (p. 451)	日、時、分、秒の値から SAS 日時値を返します。
	HMS 関数 (p. 565)	SAS 時間値(時、分および秒)を返します。
	HOLIDAY 関数 (p. 567)	指定した年の特定の祝日の SAS 日付値を返します。
	HOUR 関数 (p. 570)	時間または日時値から時間を返します。
	INTCINDEX 関数 (p. 582)	サイクルインデックスを返します。この関数には、日付、時間またはタイムスタンプの間隔と値を指定します。
	INTCK 関数 (p. 585)	DOUBLE としてエンコードされた 2 つの SAS 日付、時間、またはタイムスタンプ値の間にある特定の種類の間隔境界の数を返します。
	INTCYCLE 関数 (p. 591)	次に高い季節サイクルでの期間(日付、時間または日時の間隔)を返します。この関数には、期間(日付、時間または日時の間隔)を指定します。
	INTFIT 関数 (p. 595)	2 つの日付に基づく時間間隔を返します。
	INTGET 関数 (p. 596)	3 つの日付値または日時値に基づく時間間隔を返します。
	INTINDEX 関数 (p. 599)	季節インデックスを返します。この関数には、日付、時間またはタイムスタンプの間隔と値を指定します。
	INTNX 関数 (p. 603)	DOUBLE としてエンコードされた SAS 日付、時間または日時の値を増分し、DOUBLE としてエンコードされた SAS 日付、時間または日時の値を返します。
	INTSEAS 関数 (p. 611)	季節サイクルの長さを返します。この関数には、日付、時間または日時の間隔を指定します。
	INTSHIFT 関数 (p. 615)	ベース間隔に対応するシフト間隔を返します。

カテゴリ	言語要素	説明
	INTTEST 関数 (p. 617)	時間間隔が有効な場合は 1、時間間隔が無効な場合は 0 を返します。
	JULDATE 関数 (p. 626)	SAS 日付値からユリウス暦の日付を返します。
	JULDATE7 関数 (p. 628)	SAS 日付値から 7 桁のユリウス暦の日付を返します。
	MAKEDATE 関数 (p. 654)	日付を年、月、日として返します。
	MAKETIME 関数 (p. 655)	時間を時、分、秒で返します。
	MAKETIMESTAMP 関数 (p. 656)	タイムスタンプを返します。
	MDY 関数 (p. 664)	月、日および年の値から SAS 日付値を返します。
	MINUTE 関数 (p. 670)	時間または日時値から分を返します。
	MONTH 関数 (p. 676)	日付または日時値から月を数値で返します。
	NWKDOM 関数 (p. 706)	指定した月および年の n 番目に発生する曜日の日付を返します。
	QTR 関数 (p. 792)	SAS 日付値から年の四半期を返します。
	SECOND 関数 (p. 824)	時間または日時値から秒を返します。
	TIMEPART 関数 (p. 851)	時間を時、分、秒で返します。
	TODAY 関数 (p. 858)	数値の SAS 日付値として現在の日付を返します。
	WEEK 関数 (p. 880)	週番号の値を返します。
	WEEKDAY 関数 (p. 883)	SAS 日付値から、曜日に対応する整数を返します。
	YEAR 関数 (p. 887)	日付または日時値から年を返します。
	YRDIF 関数 (p. 890)	指定した日数計算規則に従って 2 つの日付の差を年単位で返します。人の年齢を返します。
	YYQ 関数 (p. 893)	年と四半期の値から SAS 日付値を返します。
ビットごとの論理演算	BAND 関数 (p. 313)	2 つの引数のビットごとの論理積を返します。
	BLSHIFT 関数 (p. 326)	2 つの引数を使ってビットを論理左シフトした値を返します。
	BNOT 関数 (p. 327)	引数のビットごとの論理否定を返します。
	BOR 関数 (p. 328)	2 つの引数のビットごとの論理和を返します。
	BRSHIFT 関数 (p. 329)	2 つの引数を使ってビットを論理右シフトした値を返します。

カテゴリ	言語要素	説明
	BXOR 関数 (p. 330)	2 つの引数のビットごとの排他的論理和を返します。
分位点	BETAINV 関数 (p. 315)	ベータ分布から分位点を返します。
	GAMINV 関数 (p. 550)	ガンマ分布から分位点を返します。
	PROBIT 関数 (p. 772)	標準正規分布の分位点を返します。
	QUANTILE 関数 (p. 793)	左側確率(CDF)を指定するときに分布から分位点を返します。
	SQUANTILE 関数 (p. 832)	右側確率(SDF)の指定した場合に分布から分位点を返します。
	TINV 関数 (p. 854)	t 分布から分位点を返します。
文字	ANYALNUM 関数 (p. 283)	文字列から英数字を検索し、その文字が見つかった最初の文字位置を返します。
	ANYALPHA 関数 (p. 285)	文字列からアルファベット文字を検索し、その文字が見つかった最初の文字位置を返します。
	ANYDIGIT 関数 (p. 287)	文字列から数字を検索し、最初に検索された数字の文字位置を返します。
	ANYFIRST 関数 (p. 288)	VALIDVARNAME=V7 の SAS 変数名の開始文字として有効な文字を文字列から検索し、最初に検索された文字の文字位置を返します。
	ANYLOWER 関数 (p. 291)	文字列から小文字を検索し、その文字が見つかった最初の文字位置を返します。
	ANYNAME 関数 (p. 292)	VALIDVARNAME=V7 の SAS 変数名として有効な文字を文字列から検索し、最初に検索された文字の文字位置を返します。
	ANYPUNCT 関数 (p. 294)	文字列から句読文字を検索し、その文字が見つかった最初の文字位置を返します。
	ANYSPACE 関数 (p. 296)	空白文字(空白、水平タブと垂直タブ、キャリッジリターン、ラインフィード、フォームフィード)を文字列で検索し、その文字が見つかった最初の文字位置を返します。
	ANYUPPER 関数 (p. 298)	文字列から大文字を検索し、その文字が見つかった最初の文字位置を返します。
	ANYXDIGIT 関数 (p. 300)	数字を表す 16 進法の文字を文字列から検索し、最初に検索された文字の文字位置を返します。
	BYTE 関数 (p. 331)	ASCII 照合順序で 1 文字を返します。
	CHARACTER_LENGTH 関数 (p. 382)	任意のデータ型の文字列の文字数を返します。

カテゴリ	言語要素	説明
	COALESCEC 関数 (p. 387)	文字引数のリストから最初の非 null または非欠損値を返します。
	COMPARE 関数 (p. 390)	2 つの文字列を比較し、異なる文字が検出された場合には最も左にある文字の位置を返し、異なる文字が検出されない場合には 0 を返します。
	COMPBL 関数 (p. 394)	文字列内の単語間にある複数の空白を取り除きます。
	COMPRESS 関数 (p. 396)	元の文字から指定した文字を削除した文字列を返します。
	COUNTC 関数 (p. 414)	文字のリストに表示される(または表示されない)文字列内の文字の個数を数えます。
	COUNTW 関数 (p. 418)	文字列中の単語数を数えます。
	CURRENT_LOCALE 関数 (p. 429)	現在のロケールの 5 文字の名前を返します。
	DEQUOTE 関数 (p. 445)	単一引用符で始まる文字列から一致する単一引用符を削除し、閉じ引用符の右側にあるすべての文字を削除します。
	INDEX 関数 (p. 574)	文字式から文字列を検索し、最初に検索された文字列の最初の文字の位置を返します。
	INDEXC 関数 (p. 576)	文字式から指定文字を検索し、そのいずれかの文字の最初の出現位置を返します。
	LEFT 関数 (p. 635)	文字式を左揃えにします。
	LOWCASE 関数 (p. 652)	文字式のすべての文字を小文字に変換します。
	NOTALNUM 関数 (p. 685)	文字列から英数字でない文字を検索し、その文字が見つかった最初の文字位置を返します。
	NOTALPHA 関数 (p. 687)	文字列からアルファベットでない文字を検索し、その文字が見つかった最初の文字位置を返します。
	NOTDIGIT 関数 (p. 689)	文字列から数字でない文字を検索し、その文字が見つかった最初の文字位置を返します。
	NOTFIRST 関数 (p. 691)	VALIDVARNAME=V7 の SAS 変数名として無効な開始文字を文字列から検索し、その文字が見つかった最初の文字位置を返します。
	NOTLOWER 関数 (p. 693)	文字列から小文字でない文字を検索し、その文字が見つかった最初の文字位置を返します。
	NOTNAME 関数 (p. 695)	VALIDVARNAME=V7 の SAS 変数名として無効な文字を文字列から検索し、その文字が見つかった最初の文字位置を返します。
	NOTPUNCT 関数 (p. 697)	文字列から句読文字でない文字を検索し、その文字が見つかった最初の文字位置を返します。

カテゴリ	言語要素	説明
	NOTSPACE 関数 (p. 699)	空白文字ではない文字(空白、水平タブと垂直タブ、キャリッジリターン、改行、およびフォームフィード)を文字列で検索し、その文字が見つかった最初の文字位置を返します。
	NOTUPPER 関数 (p. 701)	文字列から大文字でない文字を検索し、その文字が見つかった最初の文字位置を返します。
	NOTXDIGIT 関数 (p. 703)	文字列から 16 進文字でない文字を検索し、その文字が見つかった最初の文字位置を返します。
	OCTET_LENGTH 関数 (p. 708)	任意のデータ型の文字列のバイト数を返します。
	QUOTE 関数 (p. 797)	文字値に二重引用符を付加します。
	RANK 関数 (p. 801)	ASCII 照合順序で文字の位置を返します。
	REPEAT 関数 (p. 802)	文字式を繰り返します。
	REVERSE 関数 (p. 803)	文字式を逆にします。
	SCAN 関数 (p. 814)	文字式から n 番目の単語を返します。
	SUBSTR 関数 (p. 842)	引数から部分文字列を抽出します。
	SUBSTRING 関数 (p. 844)	文字列から部分文字列を抽出します。
	TRANSLATE 関数 (p. 859)	文字式内の特定の文字を置き換えます。
	TRANSTRN 関数 (p. 861)	文字列に含まれる特定の部分文字列をすべて 置き換えるか取り除きます。
	TRANWRD 関数 (p. 863)	文字列に含まれる特定の部分文字列をすべて 置き換えます。
	TRIM 関数 (p. 866)	文字列から先頭の文字、末尾の文字、またはその両方を削除します。
	UPCASE 関数 (p. 869)	引数内のすべての 文字を大文字に変換します。
	VERIFY 関数 (p. 878)	式に固有の最初の文字の位置を返します。
	WHICHC 関数 (p. 884)	文字列のリストから文字列の最初の位置を返します。

ディクショナリ

ABS 関数

数値式の絶対値を返します。

カテゴリ: 数学

返されるデータの種類: 式と同じデータ型

構文

ABS(*expression*)

必須引数

expression
数値に評価される有効な SQL 式を指定します。

種類 BIGINT、DOUBLE、FLOAT、INTEGER、REAL、SMALLINT、TINYINT

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

expression が null の場合、ABS 関数は null を返します。結果が引数のデータ型の範囲に収まらない数値の場合、ABS 関数は失敗します。

例

次のステートメントは、ABS 関数を示しています。

ステートメント	結果
select abs(-345);	345

ステートメント	結果
select abs((3 * 50) / 5)	30

AIRY 関数

Airy 関数の値を返します。

カテゴリ: 数学

返されるデータの種類: DOUBLE

構文

AIRY(*x*)

必須引数

x

数値の定数、変数または式を指定します。

データの種類 DOUBLE

詳細

AIRY 関数は、Airy 関数の値を返します。 (リファレンスのリストを参照。)これは、微分方程式

$$w^{(2)} - xw = 0$$

の解です(以下は条件)。

$$w(0) = \frac{1}{3^{\frac{2}{3}}\Gamma\left(\frac{2}{3}\right)}$$

および

$$w'(0) = -\frac{1}{3^{\frac{1}{3}}\Gamma\left(\frac{1}{3}\right)}$$

例

次のステートメントは、AIRY 関数を示しています。

ステートメント	結果
select airy(2.0);	0.034924
select airy(-2.0);	0.227407

ANYALNUM 関数

文字列から英数字を検索し、その文字が見つかった最初の文字位置を返します。

カテゴリ: 文字

返されるデータの種類: DOUBLE

構文

ANYALNUM('*expression*'[, *start*])

必須引数

expression
文字列に評価されるか強制変換できる有効な式を指定します。

データの種類 CHAR、NCHAR、VARCHAR、NVARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

オプション引数

start
数値に評価されるか強制変換できる有効な式を指定し、検索を開始する文字の位置と検索の方向を指定します。

データの種類 DOUBLE

詳細

ANYALNUM 関数の結果は、有効になっている変換テーブル(“TRANTAB= システムオプション” (*SAS 各国語サポート(NLS): リファレンスガイド*)を参照)に直接依存し、ENCODING および LOCALE システムオプションに間接的に依存します。

ANYALNUM 関数は、文字列で最初に出現する数字、大文字または小文字を検索します。このような文字が検出されると、ANYALNUM はその文字の文字列の位置を返します。このような文字が検出されないと、ANYALNUM は 0 の値を返します。

1 つの引数のみを使用する場合、ANYALNUM は文字列の先頭から検索を開始します。2 つの引数を使用する場合、第 2 引数 *start* の絶対値で検索の開始位置を指定します。検索方向は次のように決まります。

- *start* の値が正の場合、検索は右方向に進みます。
- *start* の値が負の場合、検索は左方向に進みます。
- *start* の値が文字列の負の長さよりも小さい場合、文字列の末尾から検索が開始されます。

ANYALNUM は、次のいずれかに該当する場合にゼロ値を返します。

- 検索文字が見つからない。
- *start* の値が文字列の長さよりも大きい。
- *start* の値が 0 になっている。

比較

ANYALNUM 関数は、文字列から英数字を検索します。NOTALNUM 関数は、文字列から英数字でない文字を検索します。

例

次のステートメントは、ANYALNUM 関数を示しています。

ステートメント	結果
select anyalnum('abc123, x');	1 (position of a)
select anyalnum('+abcd');	2 (position of a)
select anyalnum('789ab');	1 (position of 7)
select anyalnum('!&@start');	4 (position of s)
select anyalnum('!&@start1',-99);	9 (position of last alphanumeric in string)

関連項目

関数:

- “NOTALNUM 関数” (685 ページ)

ANYALPHA 関数

文字列からアルファベット文字を検索し、その文字が見つかった最初の文字位置を返します。

カテゴリ: 文字

返されるデータの種類: DOUBLE

構文

ANYALPHA('*expression*'[, *start*])

必須引数

expression
文字列に評価されるか強制変換できる有効な式を指定します。

データの種類 CHAR、NCHAR、VARCHAR、NVARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

オプション引数

start
数値に評価されるか強制変換できる有効な式を指定し、検索を開始する文字の位置と検索の方向を指定します。

データの種類 DOUBLE

詳細

ANYALPHA 関数の結果は、有効になっている変換テーブル(“TRANTAB= システムオプション” (*SAS 各国語サポート(NLS): リファレンスガイド*)を参照)に直接依存し、ENCODING および LOCALE システムオプションに間接的に依存します。

ANYALPHA 関数は、文字列で最初に出現する大文字または小文字を検索します。このような文字が検出されると、ANYALPHA はその文字の文字列の位置を返します。このような文字が検出されないと、ANYALPHA は 0 の値を返します。

1 つの引数のみを使用する場合、ANYALPHA は文字列の先頭から検索を開始します。2 つの引数を使用する場合、第 2 引数 *start* の絶対値で検索の開始位置を指定します。検索方向は次のように決まります。

- *start* の値が正の場合、検索は右方向に進みます。
- *start* の値が負の場合、検索は左方向に進みます。
- *start* の値が文字列の負の長さよりも小さい場合、文字列の末尾から検索が開始されます。

ANYALPHA は、次のいずれかに該当する場合にゼロ値を返します。

- 検索文字が見つからない。
- *start* の値が文字列の長さよりも大きい。
- *start* の値が 0 になっている。

比較

ANYALPHA 関数は、文字列からアルファベットを検索します。NOTALPHA 関数は、文字列からアルファベットでない文字を検索します。

例

次のステートメントは、ANYALPHA 関数を示しています。

ステートメント	結果
select anyalpha('123abc456');	4 (position of a)
select anyalpha('abcd1234');	1 (position of a)
select anyalpha('12345');	0 (no letters)
select anyalpha('789abc456',-99);	6 (position of last letter in string)

関連項目

関数:

- “NOTALPHA 関数” (687 ページ)

ANYDIGIT 関数

文字列から数字を検索し、最初に検索された数字の文字位置を返します。

カテゴリ: 文字

返されるデータの種類: DOUBLE

構文

ANYDIGIT('*expression*'[, *start*])

必須引数

expression
文字列に評価されるか強制変換できる有効な式を指定します。

データの種類 CHAR、NCHAR、VARCHAR、NVARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

オプション引数

start
数値に評価されるか強制変換できる有効な式を指定し、検索を開始する文字の位置と検索の方向を指定します。

データの種類 DOUBLE

詳細

ANYDIGIT 関数は、TRANTAB、ENCODING または LOCALE システムオプションに依存しません。

ANYDIGIT 関数は、文字列で最初に出現する数字を検索します。このような文字が検出されると、ANYDIGIT はその文字の文字列の位置を返します。このような文字が検出されないと、ANYDIGIT は 0 の値を返します。

1 つの引数のみを使用する場合、ANYDIGIT は文字列の先頭から検索を開始します。2 つの引数を使用する場合、第 2 引数 *start* の絶対値で検索の開始位置を指定します。検索方向は次のように決まります。

- *start* の値が正の場合、検索は右方向に進みます。

- *start* の値が負の場合、検索は左方向に進みます。
- *start* の値が文字列の負の長さよりも小さい場合、文字列の末尾から検索が開始されます。

ANYDIGIT は、次のいずれかに該当する場合にゼロ値を返します。

- 検索文字が見つからない。
- *start* の値が文字列の長さよりも大きい。
- *start* の値が 0 になっている。

比較

ANYDIGIT 関数は、文字列から数字を検索します。NOTDIGIT 関数は、文字列から数字でない文字を検索します。

例

次のステートメントは、ANYDIGIT 関数を示しています。

ステートメント	結果
select anydigit('123abc456');	1 (position of 1)
select anydigit('abcd456);	5 (position of 4)
select anydigit('abcde');	0 (no digits)
select anydigit('789ab',-99);	3 (position of last digit in string)

関連項目

関数:

- “NOTDIGIT 関数” (689 ページ)

ANYFIRST 関数

VALIDVARNAME=V7 の SAS 変数名の開始文字として有効な文字を文字列から検索し、最初に検索された文字の文字位置を返します。

カテゴリ: 文字

返されるデータの種類: DOUBLE

構文

ANYFIRST('*expression*'[, *start*])

必須引数

expression
文字列に評価されるか強制変換できる有効な式を指定します。

データの種類 CHAR、NCHAR、VARCHAR、NVARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

オプション引数

start
数値に評価されるか強制変換できる有効な式を指定し、検索を開始する文字の位置と検索の方向を指定します。

データの種類 DOUBLE

詳細

ANYFIRST 関数は、TRANTAB、ENCODING または LOCALE システムオプションに依存しません。

ANYFIRST 関数は、VALIDVARNAME=V7 の SAS 変数名の開始文字として有効な文字のうち最初に出現する文字を文字列から検索します。アンダースコア(_)および英語の大文字または小文字が有効な文字になります。このような文字が検出されると、ANYFIRST はその文字の文字列の位置を返します。このような文字が検出されないと、ANYFIRST は 0 の値を返します。

1 つの引数のみを使用する場合、ANYFIRST は文字列の先頭から検索を開始します。2 つの引数を使用する場合、第 2 引数 *start* の絶対値で検索の開始位置を指定します。検索方向は次のように決まります。

- *start* の値が正の場合、検索は右方向に進みます。
- *start* の値が負の場合、検索は左方向に進みます。
- *start* の値が文字列の負の長さよりも小さい場合、文字列の末尾から検索が開始されます。

ANYFIRST は、次のいずれかに該当する場合にゼロ値を返します。

- 検索文字が見つからない。

- *start* の値が文字列の長さよりも大きい。
- *start* の値が 0 になっている。

比較

ANYFIRST 関数は、VALIDVARNAME=V7 の SAS 変数名の開始文字として有効な文字のうち最初に出現する文字を文字列から検索します。NOTFIRST 関数は、VALIDVARNAME=V7 の SAS 変数名の開始文字として有効でない文字のうち最初に出現する文字を文字列から検索します。

例

次のステートメントは、ANYFIRST 関数を示しています。

ステートメント	結果
select anyfirst('123aBc4D6');	4 (position of a)
select anyfirst('Bac123, x');	1 (position of B)
select anyfirst('!?:;,');	0
select anyfirst('1,2,3,x');	7 (position of x)
select anyfirst('x,1,2,3');	1 (position of x)
select anyfirst('x,1,2,3',-99);	1 (position of last letter)
select anyfirst('&,x,y,z,20,!');	3 (position of x)
select anyfirst('&,x,y,z,20,!',-99);	7 (position of last letter in string)

関連項目

関数:

- “NOTFIRST 関数” (691 ページ)

ANYLOWER 関数

文字列から小文字を検索し、その文字が見つかった最初の文字位置を返します。

カテゴリ: 文字

返されるデータの種類: DOUBLE

構文

ANYLOWER('*expression*'[, *start*])

必須引数

expression
文字列に評価されるか強制変換できる有効な式を指定します。

データの種類 CHAR、NCHAR、VARCHAR、NVARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

オプション引数

start
数値に評価されるか強制変換できる有効な式を指定し、検索を開始する文字の位置と検索の方向を指定します。

データの種類 DOUBLE

詳細

ANYLOWER 関数の結果は、有効になっている変換テーブル(“TRANTAB= システムオプション” (*SAS 各国語サポート(NLS): リファレンスガイド*)を参照)に直接依存し、ENCODING および LOCALE システムオプションに間接的に依存します。

ANYLOWER 関数は、文字列で最初に出現する小文字を検索します。このような文字が検出されると、ANYLOWER はその文字の文字列の位置を返します。このような文字が検出されないと、ANYLOWER は 0 の値を返します。

1 つの引数のみを使用する場合、ANYLOWER は文字列の先頭から検索を開始します。2 つの引数を使用する場合、第 2 引数 *start* の絶対値で検索の開始位置を指定します。検索方向は次のように決まります。

- *start* の値が正の場合、検索は右方向に進みます。
- *start* の値が負の場合、検索は左方向に進みます。
- *start* の値が文字列の負の長さよりも小さい場合、文字列の末尾から検索が開始されます。

ANYLOWER は、次のいずれかに該当する場合にゼロ値を返します。

- 検索文字が見つからない。
- *start* の値が文字列の長さよりも大きい。
- *start* の値が 0 になっている。

比較

ANYLOWER 関数は、文字列から小文字を検索します。NOTLOWER 関数は、文字列から小文字でない文字を検索します。

例

次のステートメントは、ANYLOWER 関数を示しています。

ステートメント	結果
select anylower('123aBc4D6');	4 (position of a)
select anylower('123aBc4D6',-99);	6 (position of last lowercase letter)
select anylower('DAN BROWN');	0 (no lowercase letters)

関連項目

関数:

- “NOTLOWER 関数” (693 ページ)

ANYNAME 関数

VALIDVARNAME=V7 の SAS 変数名として有効な文字を文字列から検索し、最初に検索された文字の文字位置を返します。

カテゴリ: 文字

返されるデータの種類: DOUBLE

構文

ANYNAME('*expression*'[,*start*])

必須引数

expression
文字列に評価されるか強制変換できる有効な式を指定します。

データの種類 CHAR、NCHAR、VARCHAR、NVARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

オプション引数

start
数値に評価されるか強制変換できる有効な式を指定し、検索を開始する文字の位置と検索の方向を指定します。

データの種類 DOUBLE

詳細

ANYNAME 関数は、TRANTAB、ENCODING または LOCALE システムオプションに依存しません。

ANYNAME 関数は、VALIDVARNAME=V7 の SAS 変数名で有効な文字のうち最初に出現する文字を文字列から検索します。アンダースコア(_)、数字、英語の大文字または小文字が有効な文字になります。このような文字が検出されると、ANYNAME はその文字の文字列の位置を返します。このような文字が検出されないと、ANYNAME は 0 の値を返します。

1 つの引数のみを使用する場合、ANYNAME は文字列の先頭から検索を開始します。2 つの引数を使用する場合、第 2 引数 *start* の絶対値で検索の開始位置を指定します。検索方向は次のように決まります。

- *start* の値が正の場合、検索は右方向に進みます。
- *start* の値が負の場合、検索は左方向に進みます。
- *start* の値が文字列の負の長さよりも小さい場合、文字列の末尾から検索が開始されます。

ANYNAME は、次のいずれかに該当する場合にゼロ値を返します。

- 検索文字が見つからない。

- *start* の値が文字列の長さよりも大きい。
- *start* の値が 0 になっている。

比較

ANYNAME 関数は、VALIDVARNAME=V7 の SAS 変数名で有効な文字のうち最初に出現する文字を文字列から検索します。NOTNAME 関数は、VALIDVARNAME=V7 の SAS 変数名で有効でない文字のうち最初に出現する文字を文字列から検索します。

例

次のステートメントは、ANYNAME 関数を示しています。

ステートメント	結果
select anyname('<&abc!');	3 (position of a)
select anyname('&abc!');	2 (position of a)
select anyname('abc!>');	1 (position of a)
select anyname('789!>');	1 (position of 7)
select anyname('abc!>', -99);	3 (position of c)

関連項目

関数:

- “NOTNAME 関数” (695 ページ)

ANYPUNCT 関数

文字列から句読文字を検索し、その文字が見つかった最初の文字位置を返します。

カテゴリ:	文字
返されるデータの種類:	DOUBLE

構文

ANYPUNCT('*expression*'[, *start*])

必須引数

expression
文字列に評価されるか強制変換できる有効な式を指定します。

データの種類 CHAR、NCHAR、VARCHAR、NVARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

オプション引数

start
数値に評価されるか強制変換できる有効な式を指定し、検索を開始する文字の位置と検索の方向を指定します。

データの種類 DOUBLE

詳細

ANYPUNCT 関数の結果は、有効になっている変換テーブル(“TRANTAB= システムオプション” (*SAS 各国語サポート(NLS): リファレンスガイド*)を参照)に直接依存し、ENCODING および LOCALE システムオプションに間接的に依存します。

ANYPUNCT 関数は、文字列で最初に出現する句読文字を検索します。このような文字が検出されると、ANYPUNCT はその文字の文字列の位置を返します。このような文字が検出されないと、ANYPUNCT は 0 の値を返します。

1 つの引数のみを使用する場合、ANYPUNCT は文字列の先頭から検索を開始します。2 つの引数を使用する場合、第 2 引数 *start* の絶対値で検索の開始位置を指定します。検索方向は次のように決まります。

- *start* の値が正の場合、検索は右方向に進みます。
- *start* の値が負の場合、検索は左方向に進みます。
- *start* の値が文字列の負の長さよりも小さい場合、文字列の末尾から検索が開始されます。

ANYPUNCT は、次のいずれかに該当する場合にゼロ値を返します。

- 検索文字が見つからない。
- *start* の値が文字列の長さよりも大きい。
- *start* の値が 0 になっている。

比較

ANYPUNCT 関数は、文字列から句読文字を検索します。NOTPUNCT 関数は、文字列から句読文字でない文字を検索します。

例

次のステートメントは、ANYPUNCT 関数を示しています。

ステートメント	結果
select anypunct('abc123, x');	7 (position of comma)
select anypunct('!?:;,');	1 (position of exclamation mark)
select anypunct('!?:;,', -99);	5 (position of last punctuation mark in string)
select anypunct('987&efg');	4 (position of ampersand)

関連項目

関数:

- “NOTPUNCT 関数” (697 ページ)

ANYSPACE 関数

空白文字(空白、水平タブと垂直タブ、キャリッジリターン、ラインフィード、フォームフィード)を文字列で検索し、その文字が見つかった最初の文字位置を返します。

カテゴリ: 文字

返されるデータの種類: DOUBLE

構文

ANYSPACE('*expression*'[, *start*])

必須引数

expression
文字列に評価されるか強制変換できる有効な式を指定します。

データの種類 CHAR、NCHAR、VARCHAR、NVARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

オプション引数

start
数値に評価されるか強制変換できる有効な式を指定し、検索を開始する文字の位置と検索の方向を指定します。

データの種類 DOUBLE

詳細

ANYSPACE 関数の結果は、有効になっている変換テーブル(“TRANTAB= システムオプション” (*SAS 各国語サポート(NLS): リファレンスガイド*)を参照)に直接依存し、ENCODING および LOCALE システムオプションに間接的に依存します。

ANYSPACE 関数は、文字列で最初に出現する空白、水平タブ、垂直タブ、キャリッジリターン、ラインフィード、フォームフィードの文字を検索します。このような文字が検出されると、ANYSPACE はその文字の文字列の位置を返します。このような文字が検出されないと、ANYSPACE は 0 の値を返します。

1 つの引数のみを使用する場合、ANYSPACE は文字列の先頭から検索を開始します。2 つの引数を使用する場合、第 2 引数 *start* の絶対値で検索の開始位置を指定します。検索方向は次のように決まります。

- *start* の値が正の場合、検索は右方向に進みます。
- *start* の値が負の場合、検索は左方向に進みます。
- *start* の値が文字列の負の長さよりも小さい場合、文字列の末尾から検索が開始されます。

ANYSPACE は、次のいずれかに該当する場合にゼロ値を返します。

- 検索文字が見つからない。
- *start* の値が文字列の長さよりも大きい。
- *start* の値が 0 になっている。

比較

ANYSPACE 関数は、文字列で最初に出現する空白、水平タブ、垂直タブ、キャリッジリターン、ラインフィード、フォームフィードの文字を検索します。NOTSPACE

関数は、文字列で最初に出現する空白、水平タブ、垂直タブ、キャリッジリターン、ラインフィード、フォームフィードでない文字を検索します。

例

次のステートメントは、ANYSPACE 関数を示しています。

ステートメント	結果
select anyspace('abc123, x');	8 (position of space before x)
select anyspace('Ma ry Smith',-99);	6 (position of last space in string)

関連項目

関数:

- “NOTSPACE 関数” (699 ページ)

ANYUPPER 関数

文字列から大文字を検索し、その文字が見つかった最初の文字位置を返します。

カテゴリ: 文字

返されるデータの種類: DOUBLE

構文

ANYUPPER('*expression*'[, *start*])

必須引数

expression
文字列に評価されるか強制変換できる有効な式を指定します。

データの種類 CHAR、NCHAR、VARCHAR、NVARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

オプション引数

start
数値に評価されるか強制変換できる有効な式を指定し、検索を開始する文字の位置と検索の方向を指定します。

データの種類 DOUBLE

詳細

ANYUPPER 関数の結果は、有効になっている変換テーブル(“TRANTAB= システムオプション” (*SAS 各国語サポート(NLS): リファレンスガイド*)を参照)に直接依存し、ENCODING および LOCALE システムオプションに間接的に依存します。

ANYUPPER 関数は、文字列で最初に出現する大文字を検索します。このような文字が検出されると、ANYUPPER はその文字の文字列の位置を返します。このような文字が検出されないと、ANYUPPER は 0 の値を返します。

1 つの引数のみを使用する場合、ANYUPPER は文字列の先頭から検索を開始します。2 つの引数を使用する場合、第 2 引数 *start* の絶対値で検索の開始位置を指定します。検索方向は次のように決まります。

- *start* の値が正の場合、検索は右方向に進みます。
- *start* の値が負の場合、検索は左方向に進みます。
- *start* の値が文字列の負の長さよりも小さい場合、文字列の末尾から検索が開始されます。

ANYUPPER は、次のいずれかに該当する場合にゼロ値を返します。

- 検索文字が見つからない。
- *start* の値が文字列の長さよりも大きい。
- *start* の値が 0 になっている。

比較

ANYUPPER 関数は、文字列から大文字を検索します。NOTUPPER 関数は、文字列から大文字でない文字を検索します。

例

次のステートメントは、ANYUPPER 関数を示しています。

ステートメント	結果
select anyupper('123aBc4D6');	5 (position of first uppercase letter)

ステートメント	結果
select anyupper('123aBc4D6',-99);	8 (position of last uppercase letter in string)
select anyupper('abcde');	0 (no uppercase letters)

関連項目

関数:

- “NOTUPPER 関数” (701 ページ)

ANYXDIGIT 関数

数字を表す 16 進法の文字を文字列から検索し、最初に検索された文字の文字位置を返します。

カテゴリ: 文字

返されるデータの種類: DOUBLE

構文

ANYXDIGIT('*expression*'[,*start*])

必須引数

expression
文字列に評価されるか強制変換できる有効な式を指定します。

データの種類 CHAR、NCHAR、VARCHAR、NVARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

オプション引数

start
数値に評価されるか強制変換できる有効な式を指定し、検索を開始する文字の位置と検索の方向を指定します。

データの種類 DOUBLE

詳細

ANYXDIGIT 関数は、TRANTAB、ENCODING または LOCALE システムオプションに依存しません。

ANYXDIGIT 関数は、文字列で最初に出現する数字または A、B、C、D、E、F の大文字や小文字を検索します。このような文字が検出されると、ANYXDIGIT はその文字の文字列の位置を返します。このような文字が検出されないと、ANYXDIGIT は 0 の値を返します。

1 つの引数のみを使用する場合、ANYXDIGIT は文字列の先頭から検索を開始します。2 つの引数を使用する場合、第 2 引数 *start* の絶対値で検索の開始位置を指定します。検索方向は次のように決まります。

- *start* の値が正の場合、検索は右方向に進みます。
- *start* の値が負の場合、検索は左方向に進みます。
- *start* の値が文字列の負の長さよりも小さい場合、文字列の末尾から検索が開始されます。

ANYXDIGIT は、次のいずれかに該当する場合にゼロ値を返します。

- 検索文字が見つからない。
- *start* の値が文字列の長さよりも大きい。
- *start* の値が 0 になっている。

比較

ANYXDIGIT 関数は、文字列から 16 進文字を検索します。NOTXDIGIT 関数は、文字列から 16 進文字でない文字を検索します。

例

次のステートメントは、ANYXDIGIT 関数を示しています。

ステートメント	結果
select anyxdigit('123aBc4D6');	1 (position of 1)
select anyxdigit('abcd456');	1 (position of a)
select anyxdigit('789ab!',-99);	5 (position of b)
select anyxdigit('&789ab!');	2 (position of 7)

関連項目

関数:

- “NOTXDIGIT 関数” (703 ページ)

ARCOS 関数

逆余弦をラジアンで返します。

カテゴリ:	三角関数
別名:	ACOS
返されるデータの種類:	DOUBLE

構文

ARCOS(*expression*)

必須引数

expression
数値に評価される有効な SQL 式を指定します。

範囲	-1-1
データの種類	DOUBLE
参照項目:	“<sql-expression> 式” (918 ページ)
	“FedSQL 式” (46 ページ)

詳細

ARCOS 関数は、引数の逆余弦(逆コサイン)を返します。戻り値はラジアンで示されます。

例

次のステートメントは、ARCOS 関数を示しています。

ステートメント	結果
select arcos(1);	0
select arcos(0);	1.570796
select arcos(-0.5);	2.094395

関連項目

関数:

- “ARSIN 関数” (304 ページ)
- “COS 関数” (409 ページ)
- “SIN 関数” (826 ページ)

ARCOSH 関数

逆双曲線余弦を返します。

カテゴリ: 双曲線

返されるデータの種類: DOUBLE

構文

ARCOSH(*expression*)

必須引数

expression
数値に評価される有効な式を指定します。

範囲 *expression* >= 1

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

ARCOSH 関数は、逆双曲線余弦を計算します。ARCOSH 関数は、次の式で数学的に定義されます。ここで、*expression* >= 1 です。この式では、*expression* は *x* で表されます。

$$ARCOSH(x) = log\left(x + \sqrt{x^2 - 1}\right)$$

例

次のステートメントは、ARCOSH 関数を示しています。

ステートメント	結果
select arcosh(5);	2.292432
select arcosh(13);	3.256614

関連項目

関数:

- “ARSINH 関数” (305 ページ)
- “ARTANH 関数” (307 ページ)
- “COSH 関数” (410 ページ)
- “SINH 関数” (827 ページ)
- “TANH 関数” (850 ページ)

ARSIN 関数

逆正弦をラジアンで返します。

カテゴリ:	三角関数
別名:	ASIN
返されるデータの種類:	DOUBLE

構文

ARSIN(*expression*)

必須引数

expression
数値に評価される有効な SQL 式を指定します。

範囲 -1-1

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

例

次のステートメントは、ARSIN 関数を示しています。

ステートメント	結果
select arsin(0);	0
select arsin(1);	1.570796
select arsin(-0.5);	-0.5236

関連項目

関数:

- “ARCOS 関数” (302 ページ)
- “COS 関数” (409 ページ)
- “COSH 関数” (410 ページ)
- “SIN 関数” (826 ページ)

ARSINH 関数

逆双曲線正弦を返します。

カテゴリ:	双曲線
返されるデータの種類:	DOUBLE

構文

ARSINH(*expression*)

必須引数

expression
数値に評価される有効な式を指定します。

範囲	−∞‹ x ‹∞
データの種類	DOUBLE
参照項目:	“<sql-expression> 式” (918 ページ)
	“FedSQL 式” (46 ページ)

詳細

ARSINH 関数は、逆双曲線正弦を計算します。ARSINH 関数は、次の式で数学的に定義されます。ここで、$-\infty < x < \infty$です。

$$ARSINH(x) = log\left(x + \sqrt{x^2 + 1}\right)$$

無限の記号は、マシンで使用できる最大の倍精度浮動小数に置き換えられます。この式では、*expression* は *x* で表されます。

例

次のステートメントは、ARSINH 関数を示しています。

ステートメント	結果
select arsinh(5);	2.312438
select arsinh(-5);	-2.31244

関連項目

関数:

- “ARSINH 関数” (305 ページ)
- “ARTANH 関数” (307 ページ)
- “COSH 関数” (410 ページ)
- “SINH 関数” (827 ページ)
- “TANH 関数” (850 ページ)

ARTANH 関数

逆双曲線正接を返します。

カテゴリ:	双曲線
返されるデータの種類:	DOUBLE

構文

ARTANH(*expression*)

必須引数

expression
数値に評価される有効な式を指定します。

範囲	–1 < *expression* < 1
データの種類	DOUBLE
参照項目:	“<sql-expression> 式” (918 ページ)
	“FedSQL 式” (46 ページ)

詳細

ARTANH 関数は、逆双曲線正接を計算します。ARTANH 関数は、次の式で数学的に定義されます。ここで、–1 < *expression*< 1 です。この式では、*expression* は *x* で表されます。

$$ARTANH(x) = \frac{1}{2} log\left(\frac{1+x}{1-x}\right)$$

例

次のステートメントは、ARTANH 関数を示しています。

ステートメント	結果
select artanh(.5);	0.549306
select artanh(-.5);	-0.54931

関連項目

関数:

- “ARSINH 関数” (305 ページ)
- “ARTANH 関数” (307 ページ)
- “COSH 関数” (410 ページ)
- “SINH 関数” (827 ページ)
- “TANH 関数” (850 ページ)

ATAN 関数

逆正接をラジアンで返します。

カテゴリ: 三角関数

別名: ARTAN

返されるデータの種類: DOUBLE

構文

ATAN(*expression*)

必須引数

expression
数値に評価される有効な SQL 式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

ATAN 関数は、引数の 2 象限逆正接(逆タンジェント)を返します。戻り値は、正接が *x* の角度(ラジアン)です。値の範囲は- π /2から π /2までです。引数が欠損している場合、ATAN は欠損値を返します。

比較

ATAN 関数は ATAN2 関数と類似していますが、ATAN2 は 1 つの引数ではなく 2 つの引数の値から角度の逆正接を計算する点が異なります。

例

次のステートメントは、ATAN 関数を示しています。

ステートメント	結果
select atan(0);	0
select atan(1);	0.785398
select atan(-9.0);	-1.46014

関連項目

関数:

- “ATAN2 関数” (309 ページ)
- “COT 関数” (411 ページ)
- “TAN 関数” (849 ページ)
- “TANH 関数” (850 ページ)

ATAN2 関数

直角三角形の *x* 座標と *y* 座標の逆正接をラジアンで返します。

カテゴリ: 三角関数

返されるデータの種類: DOUBLE

構文

ATAN2(*expression-1*, *expression-2*)

必須引数

expression-1
数値に評価される有効な SQL 式を指定します。*expression-1* は、直角三角形の斜辺の端の x 座標を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

expression-2
数値に評価される有効な SQL 式を指定します。*expression-2* は、直角三角形の斜辺の端の y 座標を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

ATAN2 関数は、2 つの数値変数の逆正接(逆タンジェント)を返します。この関数の結果は、*expression-1*/*expression-2* の逆正接の計算結果と類似していますが、両方の引数の符号を使用して結果の象限を決定する点が異なります。ATAN2 のいずれかの引数が欠損している場合、ATAN2 は null または SAS 欠損値のいずれかを返します。

比較

ATAN2 関数は ATAN 関数と類似していますが、ATAN は 2 つの引数ではなく 1 つの引数の値から角度の逆正接を計算する点が異なります。

例

次のステートメントは、ATAN2 関数を示しています。

ステートメント	結果
select atan2(-1, 0.5);	-1.10715
select atan2(6,8);	0.643501
select atan2(5,-3);	2.111216

関連項目

- “FedSQL が null および SAS 欠損値を処理する方法” (21 ページ)

関数:

- “ATAN 関数” (308 ページ)
- “TAN 関数” (849 ページ)
- “TANH 関数” (850 ページ)

AVG 関数

列内のすべての値の平均を返します。

カテゴリ:	集計 記述統計
別名:	MEAN
返されるデータの種類:	DOUBLE

構文

AVG([DISTINCT] *expression*)

必須引数

expression
有効な SQL 式を指定します。

データの種類 BIGINT、DOUBLE、FLOAT、INTEGER、REAL、SMALLINT、TINYINT

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

オプション引数

DISTINCT
集計計算から重複行を削除します。

詳細

AVG 関数は、指定された列のすべての行の値を加算し、その結果を行数で割ります。null 値と SAS 欠損値は無視され、計算には含まれません。

集計関数を使用して、FROM 句にリストされているテーブル全体、または GROUP BY 句で指定されている各グループのデータの統計的要約を生成できます。GROUP BY 句は、指定された 1 つまたは複数の列でデータをグループ化します。GROUP BY 句を使用する場合、SELECT 句または HAVING 句の集計関数は、各グループのデータを要約する方法を FedSQL に指示します。FedSQL は、グループごとに個別に集計関数を計算します。GROUP BY を省略すると、テーブルまたはビュー内のすべての行が単一のグループと見なされます。

例

テーブル: DENSITIES (1260 ページ)

次のステートメントは、AVG 関数を示しています。

ステートメント	結果
select avg(density) from densities;	172.8324
select avg(population) from densities;	12277544

関連項目

関数:

- “SUM 関数” (846 ページ)

句:

- “SELECT 句” (980 ページ)

- “GROUP BY 句” (993 ページ)
- “HAVING 句” (995 ページ)

BAND 関数

2 つの引数のビットごとの論理積を返します。

カテゴリ: ビットごとの論理演算

返されるデータの種類: DOUBLE

構文

BAND(*expression-1*, *expression-2*)

必須引数

expression-1
数値に評価される有効な式を指定します。

範囲 0 から(2^{32})-1 まで(両端を含む)

データの種類 DOUBLE

注 この関数の入力として負の値が渡された場合、エラーメッセージはログに書き込まれず、欠損値が出力として返されます。

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

expression-2
数値に評価される有効な式を指定します。

範囲 0 から(2^{32})-1 まで(両端を含む)

データの種類 DOUBLE

例

次のステートメントは、BAND 関数を示しています。

ステートメント	結果
select band(9,11);	9
select band(15,5);	5

BETA 関数

beta 関数の値を返します。

カテゴリ: 数学

返されるデータの種類: DOUBLE

構文

BETA(*a*, *b*)

必須引数

a
第 1 形状パラメーターです。

範囲 *a* >0

データの種類 DOUBLE

b
第 2 形状パラメーターです。

範囲 *b* >0

データの種類 DOUBLE

詳細

BETA 関数は、次の式で数学的に求められます。

$$\beta(a,b) = \int_0^1 x^{a-1}(1-x)^{b-1}dx$$

注:

$$\beta(a,b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}$$

前述の式で、$\Gamma(.)$はガンマ関数です。

式を計算できない場合、BETA は欠損値を返します。

例

次のステートメントは、BETA 関数を示しています。

ステートメント	結果
select beta(5,3);	0.00952380952381
select beta(15,45);	1.6710294365E-15

関連項目

関数:

- “LOGBETA 関数” (642 ページ)

BETAINV 関数

ベータ分布から分位点を返します。

カテゴリ: 分位点

返されるデータの種類: DOUBLE

構文

BETAINV(*p*, *a*, *b*)

必須引数

p
数値の確率です。

範囲 0 ≤ *p* ≤ 1

データの種類 DOUBLE

注　負の数または 1 より大きい数がこの引数の入力として渡された場合、エラーメッセージはログに書き込まれず、欠損値が出力として返されます。

a

数値の形状パラメーターです。

範囲　*a* > 0

データの種類　DOUBLE

注　この引数の入力としてゼロまたは負の数が渡された場合、エラーメッセージはログに書き込まれず、欠損値が出力として返されます。

b

数値の形状パラメーターです。

範囲　*b* > 0

データの種類　DOUBLE

注　この引数の入力としてゼロまたは負の数が渡された場合、エラーメッセージはログに書き込まれず、欠損値が出力として返されます。

詳細

BETAINV 関数は、ベータ分布(形状パラメーターは *a* および *b*)の *p* 分位点を返します。ベータ分布のオブザベーションが返される分位点以下になる確率は *p* です。

注: BETAINV は、PROBBETA 関数の逆数です。

例

次のステートメントは、BETAINV 関数を示しています。

ステートメント	結果
select betainv(0.001,2,4);	0.0101017879

関連項目

関数:

- “PROBBETA 関数” (763 ページ)

BLACKCLPRC 関数

Black モデルに基づき、先物のヨーロピアンオプションのコール価格を計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

BLACKCLPRC(*E*, *t*, *F*, *r*, *sigma*)

必須引数

E
権利行使価格を指定する正の非欠損値。

要件 *E* および *F* は同じ単位で指定します。

データの種類 DOUBLE

t
満期までの時間を年単位で指定する非欠損値。

データの種類 DOUBLE

F
先物価格を指定する正の非欠損値。

要件 *F* および *E* は同じ単位で指定します。

データの種類 DOUBLE

r
連続複利の無リスク年金利を指定する正の非欠損値。

データの種類 DOUBLE

sigma
ボラティリティ (*r* の分散の平方根)を指定する正の非欠損分数。

データの種類 DOUBLE

詳細

BLACKCLPRC 関数は、Black モデルに基づき、先物のヨーロピアンオプションのコール価格を計算します。この関数は次の関係に基づきます。

$$\text{CALL} = \varepsilon^{-rt}(FN(d_1) - EN(d_2))$$

引数

F
先物価格を指定します。

N
累積正規密度関数を指定します。

E
オプションの権利行使価格を指定します。

r
無リスク金利を指定します。これは、連続複利計算で表される年率です。

t
有効期限までの時間を年単位で指定します。

$$d_1 = \frac{\left(\ln\left(\frac{F}{E}\right) + \left(\frac{\sigma^2}{2}\right)t\right)}{\sigma\sqrt{t}}$$

$$d_2 = d_1 - \sigma\sqrt{t}$$

前述の式には次の引数が適用されます。

σ
原資産のボラティリティを 指定します。

σ^2
利益率の分散を指定します。

t=0 となる特別な場合には、次の式が真です。

$$\text{CALL} = \max((F - E), 0)$$

価格設定の基本については、*“価格関数の使用” (SAS 関数と CALL ルーチン: リファレンス)*を参照してください。

比較

BLACKCLPRC 関数は、Black モデルに基づき、先物のヨーロピアンオプションのコール価格を計算します。BLACKPTPRC 関数は、Black モデルに基づき、先物のヨーロピアンオプションのプット価格を計算します。これらの関数はスカラー値を返します。

例

次のステートメントは、BLACKCLPRC 関数を示しています。

ステートメント	結果
	----+----1----+-—-2--
select blackclprc(50,.25,48,.05,.25);	1.55130142723117
select blackclprc(9,1/12,10,.05,.2);	1

関連項目

関数:

- “BLACKPTPRC 関数” (319 ページ)

BLACKPTPRC 関数

Black モデルに基づき、先物のヨーロピアンオプションのプット価格を計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

BLACKPTPRC(*E*, *t*, *F*, *r*, *sigma*)

必須引数

E

権利行使価格を指定する正の非欠損値。

要件 *E* および *F* は同じ単位で指定します。

データの種類 DOUBLE

t

満期までの時間を年単位で指定する非欠損値。

データの種類 DOUBLE

F
先物価格を指定する正の非欠損値。

要件　*F* および *E* は同じ単位で指定します。

データの種類　DOUBLE

r
連続複利の無リスク年金利を指定する正の非欠損値。

データの種類　DOUBLE

sigma
ボラティリティ (*r* の分散の平方根)を指定する正の非欠損分数。

データの種類　DOUBLE

詳細

BLACKPTPRC 関数は、Black モデルに基づき、先物のヨーロピアンオプションのプット価格を計算します。この関数は次の関係に基づきます。

$$\mathrm{PUT} = \mathrm{CALL} + \varepsilon^{-rt}(E - F)$$

引数

E
オプションの権利行使価格を指定します。

r
無リスク金利を指定します。これは、連続複利計算で表される年率です。

t
有効期限までの時間を年単位で指定します。

F
先物価格を指定します。

$$d_1 = \frac{\left(\ln\left(\frac{F}{E}\right) + \left(\frac{\sigma^2}{2}\right)t\right)}{\sigma\sqrt{t}}$$

$$d_2 = d_1 - \sigma\sqrt{t}$$

前述の式には次の引数が適用されます。

σ
原資産のボラティリティを 指定します。

σ^2
利益率の分散を指定します。

t=0 となる特別な場合には、次の式が真です。

$$\mathrm{PUT} = \max((E - F), 0)$$

価格設定の基本については、“価格関数の使用” (*SAS 関数と CALL ルーチン: リファレンス*)を参照してください。

比較

BLACKPTPRC 関数は、Black モデルに基づき、先物のヨーロピアンオプションのプット価格を計算します。BLACKCLPRC 関数は、Black モデルに基づき、先物のヨーロピアンオプションのコール価格を計算します。これらの関数はスカラー値を返します。

例

次のステートメントは、BLACKPTPRC 関数を示しています。

ステートメント	結果
	----+----1----+----2--
select blackptprc(298,.25,350,.06,.25);	1.85980563934967
select blackptprc(145,.5,170,.05,.2);	1.41234979911583

関連項目

関数:

- “BLACKCLPRC 関数” (317 ページ)

BLKSHCLPRC 関数

Black-Scholes モデルに基づき、株式のヨーロピアンオプションのコール価格を計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

BLKSHCLPRC(*E*, *t*, *S*, *r*, *sigma*)

必須引数

E
権利行使価格を指定する正の非欠損値。

要件　*E* および *S* は同じ単位で指定します。

データの種類　DOUBLE

t
満期までの時間を年単位で指定する非欠損値。

データの種類　DOUBLE

S
株価を指定する正の非欠損値。

要件　*S* および *E* は同じ単位で指定します。

データの種類　DOUBLE

r
連続複利の無リスク年金利を指定する正の非欠損値。

データの種類　DOUBLE

sigma
原資産のボラティリティを 指定する正の非欠損分数。

データの種類　DOUBLE

詳細

BLKSHCLPRC 関数は、Black-Scholes モデルに基づき、株式のヨーロピアンオプションのコール価格を計算します。この関数は次の関係に基づきます。

$$\text{CALL} = SN(d_1) - EN(d_2)\varepsilon^{-rt}$$

引数

S
株価を指定する正の非欠損値。

N
累積正規密度関数を指定します。

E
オプションの権利行使価格を指定する正の非欠損値。

$$d_1 = \frac{\left(\ln\left(\frac{S}{E}\right) + \left(r + \frac{\sigma^2}{2}\right)t\right)}{\sigma\sqrt{t}}$$

$$d_2 = d_1 - \sigma\sqrt{t}$$

前述の式には次の引数が適用されます。

t
有効期限までの時間を年単位で指定します。

r
無リスク金利を指定します。これは、連続複利計算で表される年率です。

σ
ボラティリティ (分散の平方根)を指定します。

σ^2
利益率の分散を指定します。

t=0 となる特別な場合には、次の式が真です。

$$\mathrm{CALL} = \max((S - E), 0)$$

価格設定の基本については、“価格関数の使用” (*SAS 関数と CALL ルーチン: リファレンス*)を参照してください。

比較

BLKSHCLPRC 関数は、Black-Scholes モデルに基づき、株式のヨーロピアンオプションのコール価格を計算します。BLKSHPTPRC 関数は、Black-Scholes モデルに基づき、株式のヨーロピアンオプションのプットを計算します。これらの関数はスカラー値を返します。

例

次のステートメントは、BLKSHCLPRC 関数を示しています。

ステートメント	結果
	----+----1----+----2--
select blkshclprc(50,.25,48,.05,.25);	1.79894201954463
select blkshclprc(9,1/12,10,.05,.2);	1

関連項目

関数:

- “BLKSHPTPRC 関数” (324 ページ)

BLKSHPTPRC 関数

Black-Scholes モデルに基づき、株式のヨーロピアンオプションのプット価格を計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

BLKSHPTPRC(*E*, *t*, *S*, *r*, *sigma*)

必須引数

E
権利行使価格を指定する正の非欠損値。

要件 *E* および *S* は同じ単位で指定します。

データの種類 DOUBLE

t
満期までの時間を年単位で指定する非欠損値。

データの種類 DOUBLE

S
株価を指定する正の非欠損値。

要件 *S* および *E* は同じ単位で指定します。

データの種類 DOUBLE

r
連続複利の無リスク年金利を指定する正の非欠損値。

データの種類 DOUBLE

sigma
原資産のボラティリティを 指定する正の非欠損分数。

データの種類 DOUBLE

詳細

BLKSHPTPRC 関数は、Black-Scholes モデルに基づき、株式のヨーロピアンオプションのプットを計算します。この関数は次の関係に基づきます。

$$\text{PUT} = \text{CALL} - S + E\varepsilon^{-rt}$$

引数

S
株価を指定する正の非欠損値。

E
オプションの権利行使価格を指定する正の非欠損値。

$$d_1 = \frac{\left(\ln\left(\frac{S}{E}\right) + \left(r + \frac{\sigma^2}{2}\right)t\right)}{\sigma\sqrt{t}}$$

$$d_2 = d_1 - \sigma\sqrt{t}$$

前述の式には次の引数が適用されます。

t
有効期限までの時間を年単位で指定します。

r
無リスク金利を指定します。これは、連続複利計算で表される年率です。

σ
ボラティリティ (分散の平方根)を指定します。

σ^2
利益率の分散を指定します。

t=0 となる特別な場合には、次の式が真です。

$$\text{PUT} = \max((E - S), 0)$$

価格設定の基本については、“価格関数の使用” (*SAS 関数と CALL ルーチン: リファレンス*)を参照してください。

比較

BLKSHPTPRC 関数は、Black-Scholes モデルに基づき、株式のヨーロピアンオプションのプットを計算します。BLKSHCLPRC 関数は、Black-Scholes モデルに基づき、株式のヨーロピアンオプションのコール価格を計算します。これらの関数はスカラー値を返します。

例

次のステートメントは、BLKSHPTPRC 関数を示しています。

ステートメント	結果
	----+----1----+----2--
select blkshptprc(230,.5,290,.04,.25);	1.56597442946068
select blkshptprc(350,.3,400,.05,.2);	1.64091943067592

関連項目

関数:

- “BLKSHCLPRC 関数” (321 ページ)

BLSHIFT 関数

2 つの引数を使ってビットを論理左シフトした値を返します。

カテゴリ: ビットごとの論理演算

返されるデータの種類: DOUBLE

構文

BLSHIFT(*expression-1*, *expression-2*)

必須引数

expression-1
数値に評価される有効な式を指定します。

範囲 0 から(2^{32})-1 まで(両端を含む)

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

expression-2
数値に評価される有効な式を指定します。

範囲 0 から 31(0 と 31 を含む)

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

例

次のステートメントは、BLSHIFT 関数を示しています。

ステートメント	結果
select blshift(7,2);	28

関連項目

関数:

- “BRSHIFT 関数” (329 ページ)

BNOT 関数

引数のビットごとの論理否定を返します。

カテゴリ: ビットごとの論理演算

返されるデータの種類: DOUBLE

構文

BNOT(*expression*)

必須引数

expression
数値に評価される有効な式を指定します。

範囲 0 から(2^{32})-1 まで(両端を含む)

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

例

次のステートメントは、BNOT 関数を示しています。

ステートメント	結果
select bnot(16);	4.295E9
select put(bnot(16), best16.);	4294967279

BOR 関数

2 つの引数のビットごとの論理和を返します。

カテゴリ: ビットごとの論理演算

返されるデータの種類: DOUBLE

構文

BOR(*expression-1, expression-2*)

必須引数

expression-1, expression-2
数値に評価される有効な式を指定します。

範囲 0 から(2^{32})-1 まで(両端を含む)

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

例

次のステートメントは、BOR 関数を示しています。

ステートメント	結果
select bor(4,8);	12

BRSHIFT 関数

2 つの引数を使ってビットを論理右シフトした値を返します。

カテゴリ: ビットごとの論理演算

返されるデータの種類: DOUBLE

構文

BRSHIFT(*expression-1*, *expression-2*)

必須引数

expression-1
数値に評価される有効な式を指定します。

範囲 0 から(2^{32})-1 まで(両端を含む)

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

expression-2
数値に評価される有効な式を指定します。

範囲 0 から 31(0 と 31 を含む)

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

例

次のステートメントは、BRSHIFT 関数を示しています。

ステートメント	結果
select brshift(64,2);	16

関連項目

関数:

- “BLSHIFT 関数” (326 ページ)

BXOR 関数

2 つの引数のビットごとの排他的論理和を返します。

カテゴリ: ビットごとの論理演算

返されるデータの種類: DOUBLE

構文

BXOR(*expression-1, expression-2*)

必須引数

expression-1, expression-2
数値に評価される有効な式を指定します。

範囲 0 から(2^{32})-1 まで(両端を含む)

データの種類 DOUBLE

注 この関数の入力として負の数が渡された場合、エラーメッセージはログに書き込まれず、欠損値が出力として返されます。

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

例

次のステートメントは、BXOR 関数を示しています。

ステートメント	結果
select bxor(128,64);	192

BYTE 関数

ASCII 照合順序で 1 文字を返します。

カテゴリ: 文字

返されるデータの種類: CHAR、NCHAR、NVARCHAR、VARCHAR

構文

BYTE(*n*)

必須引数

n
特定の ASCII 文字を表す整数を指定します。

範囲 0-255

データの種類 DOUBLE

詳細

ASCII 照合順序では、0 から 127 までの値に対応する文字が標準文字セットを表します。128 から 255 までの値に対応するその他の ASCII 文字は、特定の ASCII 動作環境で使用できますが、それらの文字が表す情報は動作環境によって異なります。

例

次のステートメントは、BYTE 関数を示しています。

ステートメント	結果
	ASCII
	----+----1----+----2
select byte(80);	P

関連項目

関数:

- “RANK 関数” (801 ページ)

CAST 関数

値をあるデータ型から別のデータ型に変換します。

カテゴリ: 特殊

別名: CAST キーワードのかわりに演算子"::"を使用できます。

返されるデータの種類: ターゲットのデータ型。

構文

CAST(*expression* AS *data-type* [(*length*)])

必須引数

expression
有効な SQL 式を指定します。

データの種類 式は、データソースでサポートされている任意の FedSQL データ型に解決できます。

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

付録 2, “データ型リファレンス” (1199 ページ)

data-type
ターゲットのデータ型です。サポートされているデータ型変換については、“詳細” (333 ページ)を参照してください。

制限事項 ターゲットのデータ型は、FedSQL とデータソースでサポートされている必要があります。

参照項目: “データ型” (14 ページ)

付録 2, “データ型リファレンス” (1199 ページ)

オプション引数

length
ターゲットのデータ型の長さを指定するオプションの整数です。

注 CHAR(*n*)にキャストする場合は、LENGTH 引数を使用します。長さ指定子なしで CHAR(*n*)型にキャストすると、型の長さは CHAR(1)と同等になります。長さは、少なくとも CAST が生成する文字数と同じでなければなりません。長さ指定子なしで VARCHAR(*n*)に変換すると、型は任意のサイズの文字列を受け入れます。

詳細

FedSQL は、データソースサポートの必要に応じて暗黙的な型変換を実行します。CAST 関数は、明示的な型変換を実行するために提供されています。

CAST 式は、式が許可されている場所であればどこでも使用できます。あるデータ型を別のデータ型に変換するだけでなく、CAST を使用してターゲット型の長さと精度を変更することもできます。

次の型変換がサポートされています。

- 文字型は、別の文字型、数値型、および DATE または TIMESTAMP に変換できます。
- 数値型は、CHAR、VARCHAR、または別の数値型に変換できます。切り捨てなしではターゲット型が非小数部分を表すことができない場合、例外が発生します。ターゲット数値がソース数値の小数部分(スケール)を表すことができない場合、ソースはターゲットに収まるように暗黙のうちに切り捨てられます。たとえば、123.4763 を INTEGER としてキャストすると、123 が返されます。
- DATE または TIME 値は、TIMESTAMP および DOUBLE に変換できます。TIMESTAMP も DOUBLE に変換できます。DATE が TIMESTAMP に変換される場合、結果の TIMESTAMP の TIME 部分は常に 00:00:00 です。TIME データ値が TIMESTAMP に変換される場合、DATE 部分は、CAST の実行時に CURRENT_DATE の値に設定されます。TIMESTAMP が DATE に変換される場合、TIME 部分は暗黙のうちに切り捨てられます。TIMESTAMP が TIME に変換される場合、DATE 部分は暗黙のうちに切り捨てられます。

比較

CAST 関数は、指定された入力変数のデータ型を永続的に変更します。PUT 関数は、それが指定されているクエリの出力に影響を与えます。

例

テーブル: “CustonLine” (1259 ページ)

テーブル: “Integers” (1265 ページ)

次のステートメントは、CAST 関数を示しています。

ステートメント	結果
select cast(begintime as DATE) from custonline;	01SEP2013 02OCT2013 15OCT2013 01NOV2013 01DEC2013 02JAN2013 16JAN2013 01FEB2013 01MAR2013 15MAR2013
select si::integer as int from integers;	32767
select bi::bigint * 2::bigint as bigbang from integers;	ERROR: Numeric value out of range
select cast ('2014-04-10' as DATE);	10APR2014
select cast ('2014-04-10 10:56:49' as TIMESTAMP);	10APR2014:10:56:49
select cast ('10:56:49' as TIME);	10:56:49
d が DBMS テーブルの DECIMAL 列の場合: select d, cast(d as int) as d2 from *dbms.table*;	d: 123.4763 d2: 123

CDF 関数

さまざまな連続確率分布および離散確率分布の左累積分布関数を計算します。

カテゴリ: 確率

注: QUANTILE 関数は、指定した分布から分位点を返します。QUANTILE 関数は、CDF 関数の逆数です。詳細については、“QUANTILE 関数” (793 ページ)を参照してください。この関数の入力として無効な数値が指定された場合、エラーメッセージはログに書き込まれず、欠損値が出力として返されます。

構文

CDF(*'distribution'*, *quantile* [, *parameter-1, ..., parameter-k*])

必須引数

distribution
分布を特定する文字定数、変数または式です。

有効な分布は次のとおりです。

分布	引数
Bernoulli	'BERNOULLI'
ベータ	'BETA'
二項	'BINOMIAL'
Cauchy	'CAUCHY'
カイ 2 乗	'CHISQUARE'
Conway-Maxwell-Poisson	'CONMAXPOI'
指数	'EXPONENTIAL'
F	'F'
ガンマ	'GAMMA'
一般化 Poisson	'GENPOISSON'
幾何	'GEOMETRIC'
超幾何	'HYPERGEOMETRIC'
Laplace	'LAPLACE'
ロジスティック	'LOGISTIC'
対数正規	'LOGNORMAL'

分布	引数
負の二項	'NEGBINOMIAL'
正規	'NORMAL'|'GAUSS'
正規混合	'NORMALMIX'
パレート	'PARETO'
Poisson	'POISSON'
T	'T'
Tweedie	'TWEEDIE'
一様	'UNIFORM'
Wald (逆ガウス)	'WALD'|'IGAUSS'
Weibull	'WEIBULL'

注 T、F および NORMALMIX を除き、最初の 4 文字で分布を最小限に識別できます。

quantile
確率変数の値を指定する数値定数、変数または式です。

データの種類 DOUBLE

オプション引数

parameter-1, ... , parameter-k
特定の分布に適した*形状*、*位置*または*尺度*パラメーターの値を指定する定数、変数または式です(省略可能)。

データの種類 DOUBLE

関連項目

関数:

- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)

- “PDF 関数” (712 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

CDF BERNOULLI 分布 関数

Bernoulli 累積確率分布の値を返します。

カテゴリ: 確率

返されるデータの種類: DOUBLE

注: QUANTILE 関数は、指定した分布から分位点を返します。QUANTILE 関数は、CDF 関数の逆数です。詳細については、“QUANTILE 関数” (793 ページ)を参照してください。

構文

CDF('BERNOULLI', *x*, *p*)

必須引数

x
確率変数を指定する数値定数、変数または式です。

データの種類 DOUBLE

p
成功確率を指定する数値定数、変数または式です。

範囲 $0 \leq p \leq 1$

データの種類 DOUBLE

詳細

Bernoulli 分布の CDF 関数は、Bernoulli 分布(成功確率は *p*)からオブザベーションが *x* 以下となる確率を返します。

$$CDF('BERN', x, p) = \begin{cases} 0 & x < 0 \\ 1 - p & 0 \leq x < 1 \\ 1 & x \geq 1 \end{cases}$$

注: この分布には、*位置*または*尺度*パラメーターはありません。

例

次のステートメントは、CDF Bernoulli 分布関数を示しています。

ステートメント	結果
select cdf('BERN', 0, .25);	0.75

関連項目

関数:

- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “PDF BERNOULLI 分布 関数” (715 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

CDF BETA 分布 関数

ベータ累積確率分布の値を返します。

カテゴリ: 確率

返されるデータの種類: DOUBLE

注: QUANTILE 関数は、指定した分布から分位点を返します。QUANTILE 関数は、CDF 関数の逆数です。詳細については、“QUANTILE 関数” (793 ページ)を参照してください。

構文

CDF('BETA', *x*, *a*, *b* [, *l*, *r*])

必須引数

x
確率変数を指定する数値定数、変数または式です。

データの種類 DOUBLE

a
形状パラメーターを指定する数値定数、変数または式です。

範囲 $a > 0$

データの種類 DOUBLE

b
形状パラメーターを指定する数値定数、変数または式です。

範囲 $b > 0$

データの種類 DOUBLE

オプション引数

l
左位置パラメーターを指定する数値定数、変数または式です。

デフォルト 0

データの種類 DOUBLE

r
右位置パラメーターを指定する数値定数、変数または式です。

デフォルト 1

範囲 $r > l$

データの種類 DOUBLE

詳細

ベータ分布の CDF 関数は、ベータ分布(形状パラメーターは *a* および *b*)からオブザベーションが *v* 以下となる確率を返します。次の式は、ベータ分布の CDF 関数を表しています。

$$CDF('BETA', x, a, b, l, r) = \begin{cases} 0 & x \le l \\ \dfrac{1}{\beta(a,b)} \displaystyle\int_{l}^{x} \dfrac{(v-l)^{a-1}(r-v)^{b-1}}{(r-l)^{a+b-1}} dv & l < x \le r \\ 1 & x > r \end{cases}$$

前述の式には次の関係が適用されます。

$$\beta(a,b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}$$

前述の式には次の関係が適用されます。

$$\Gamma(a) = \int_0^\infty x^{a-1} e^{-x} dx$$

例

次のステートメントは、CDF ベータ分布関数を示しています。

ステートメント	結果
select cdf('BETA', 0.2, 3, 4);	0.09888

関連項目

関数:

- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “PDF BETA 分布 関数” (716 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

CDF BINOMIAL 分布 関数

二項累積確率分布の値を返します。

カテゴリ:	確率
返されるデータの種類:	DOUBLE
注:	QUANTILE 関数は、指定した分布から分位点を返します。QUANTILE 関数は、CDF 関数の逆数です。詳細については、“QUANTILE 関数” (793 ページ)を参照してください。

構文

CDF('BINOMIAL', *m*, *p*, *n*)

必須引数

m
成功数を数える整数の確率変数です。

範囲 *m* = 0, 1, ...

データの種類 DOUBLE

p
成功確率パラメーターを指定する数値定数、変数または式です。

範囲 $0 \le p \le 1$

データの種類 DOUBLE

n
独立 Bernoulli 試行数を数える整数パラメーターを指定する数値定数、変数または式です。

範囲 *n* = 0, 1, ...

データの種類 DOUBLE

詳細

二項分布の CDF 関数は、二項分布(パラメーターは *p* および *n*)からオブザベーションが *m* 以下となる確率を返します。

$$CDF('BINOM', m, p, n) = \begin{cases} 0 & m < 0 \\ \sum_{j=0}^{m} \binom{n}{j} p^j (1-p)^{n-j} & 0 \le m \le n \\ 1 & m > n \end{cases}$$

注: 二項分布には、*位置*または*尺度*パラメーターはありません。

例

次のステートメントは、CDF 二項分布関数を示しています。

ステートメント	結果
select cdf('BINOM', 4, .5, 10);	0.376953

関連項目

関数:

- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “PDF BINOMIAL 分布 関数” (718 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

CDF CAUCHY 分布 関数

Cauchy 累積確率分布の値を返します。

カテゴリ: 確率

別名: PMF

返されるデータの種類: DOUBLE

注: QUANTILE 関数は、指定した分布から分位点を返します。QUANTILE 関数は、CDF 関数の逆数です。詳細については、“QUANTILE 関数” (793 ページ)を参照してください。

構文

CDF('CAUCHY', *x* [,*θ*] [, *λ*])

必須引数

x
確率変数を指定する数値定数、変数または式です。

データの種類 DOUBLE

オプション引数

θ
位置パラメーターを指定する数値定数、変数または式です。

デフォルト 0

データの種類 DOUBLE

λ
尺度パラメーターを指定する数値定数、変数または式です。

デフォルト 1

範囲 λ > 0

データの種類 DOUBLE

詳細

Cauchy 分布の CDF 関数は、Cauchy 分布(位置パラメーターは θ、尺度パラメーターは λ)からオブザベーションが *x* 以下となる確率を返します。

$$CDF('CAUCHY', x, \theta, \lambda) = \frac{1}{2} + \frac{1}{\pi} tan^{-1}\left(\frac{x-\theta}{\lambda}\right)$$

例

次のステートメントは、CDF Cauchy 分布関数を示しています。

ステートメント	結果
select cdf('CAUCHY', 2);	0.852416

関連項目

関数:

- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “PDF CAUCHY 分布 関数” (720 ページ)
- “QUANTILE 関数” (793 ページ)

- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

CDF カイ 2 乗分布 関数

カイ 2 乗累積確率分布の値を返します。

カテゴリ: 確率

返されるデータの種類: DOUBLE

注: QUANTILE 関数は、指定した分布から分位点を返します。QUANTILE 関数は、CDF 関数の逆数です。詳細については、“QUANTILE 関数” (793 ページ)を参照してください。

構文

CDF('CHISQUARE', *x*, *df* [, *nc*])

必須引数

x
確率変数を指定する数値定数、変数または式です。

データの種類 DOUBLE

df
自由度パラメーターを指定する数値定数、変数または式です。

範囲 $df > 0$

データの種類 DOUBLE

オプション引数

nc
省略可能な非心度パラメーターを指定する数値定数、変数または式です。

範囲 $nc \geq 0$

データの種類 DOUBLE

詳細

カイ 2 乗分布の CDF 関数は、カイ 2 乗分布(自由度は *df*、非心度パラメーターは *nc*)からオブザベーションが *x* 以下となる確率を返します。この関数では、整数以外の自由度を使用できます。*nc* が省略されているかゼロの場合、心度カイ 2 乗分布の値が返されます。次の式で、$\nu = df$として、そして$\lambda = nc$とします。次の式は、カイ 2 乗分布の CDF 関数を表しています。

$$CDF('CHISQ', x, \nu, \lambda) = \begin{cases} 0 & x < 0 \\ \sum_{j=0}^{\infty} e^{-\frac{\lambda}{2}} \frac{\left(\frac{\lambda}{2}\right)^j}{j!} P_c(x, \nu + 2j) & x \geq 0 \end{cases}$$

この式では、$P_c(.,.)$は、心度カイ 2 乗分布からの確率を示しています。

$$P_c(x, a) = P_g\left(\frac{x}{2}, \frac{a}{2}\right)$$

この式では、$P_g(y,b)$は、次の式によって得られるガンマ分布からの確率です。

$$P_g(y, b) = \frac{1}{\Gamma(b)} \int_0^y e^{-v} v^{b-1} dv$$

例

次のステートメントは、CDF カイ 2 乗分布関数を示しています。

ステートメント	結果
select cdf('CHISQ', 11.264, 11);	0.578581

関連項目

関数:

- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “PDF カイ 2 乗分布 関数” (721 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

CDF Conway-Maxwell-Poisson 分布 関数

Conway-Maxwell-Poisson 累積確率分布の値を返します。

カテゴリ: 確率

返されるデータの種類: DOUBLE

注: QUANTILE 関数は、指定した分布から分位点を返します。QUANTILE 関数は、CDF 関数の逆数です。詳細については、“QUANTILE 関数” (793 ページ)を参照してください。

構文

CDF('CONMAXPOI', *y*, *λ*,*ν*)

必須引数

y
カウントデータを表す負でない整数を指定する数値定数、変数または式です。

データの種類 DOUBLE

λ
Poisson 分布の場合と同様で、平均に似ています。

データの種類 DOUBLE

ν
ばらつきのパラメーターを指定する数値定数、変数または式です。

データの種類 DOUBLE

詳細

CDF 関数は、0 から y までの累積確率を返します。詳細については、PDF 関数の “Conway-Maxwell-Poisson”分布 (723 ページ)を参照してください。

例

次のステートメントは、CDF Conway-Maxwell-Poisson 分布関数を示しています。

ステートメント	結果
select cdf('CONMAXPOI', 5, 2.3, .4);	0.244541

関連項目

関数:

- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “PDF Conway-Maxwell-Poisson 分布 関数” (723 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

CDF 指数分布 関数

指数累積確率分布の値を返します。

カテゴリ: 確率

返されるデータの種類: DOUBLE

注: QUANTILE 関数は、指定した分布から分位点を返します。QUANTILE 関数は、CDF 関数の逆数です。詳細については、“QUANTILE 関数” (793 ページ)を参照してください。

構文

CDF('EXPONENTIAL', *x* [,*λ*])

必須引数

x
確率変数を指定する数値定数、変数または式です。

データの種類 DOUBLE

オプション引数

λ

尺度パラメーターを指定する数値定数、変数または式です。

デフォルト	1
範囲	λ > 0
データの種類	DOUBLE

詳細

指数分布の CDF 関数は、指数分布(尺度パラメーターは λ)からオブザベーションが *x* 以下となる確率を返します。

$$CDF('EXPO', x, \lambda) = \begin{cases} 0 & x < 0 \\ 1 - e^{-\frac{x}{\lambda}} & x \geq 0 \end{cases}$$

例

次のステートメントは、CDF 指数分布関数を示しています。

ステートメント	結果
select cdf('EXPO', 1);	0.632121

関連項目

関数:

- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “PDF EXPONENTIAL 分布 関数” (726 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

CDF F 分布 関数

F 累積確率分布の値を返します。

カテゴリ: 確率

返されるデータの種類: DOUBLE

注: QUANTILE 関数は、指定した分布から分位点を返します。QUANTILE 関数は、CDF 関数の逆数です。詳細については、“CDF F 分布 関数” (349 ページ)を参照してください。

構文

CDF('F', *x*, *ndf*, *ddf* [, *nc*])

必須引数

x
確率変数を指定する数値定数、変数または式です。

ndf
分子の自由度パラメーターを指定する数値定数、変数または式です。

範囲 *ndf* > 0

データの種類 DOUBLE

ddf
分母の自由度パラメーターを指定する数値定数、変数または式です。

範囲 *ddf* > 0

データの種類 DOUBLE

オプション引数

nc
非心度パラメーターを指定する数値定数、変数または式です。

範囲 *nc* ≥ 0

データの種類 DOUBLE

詳細

F 分布の CDF 関数は、*F* 分布(分子の自由度は *ndf*、分母の自由度は *ddf*、非心度パラメーターは *nc*)からオブザベーションが *x* 以下となる確率を返します。この関数では、*ndf* および *ddf* に対して整数以外の自由度を使用できます。*nc* が除外されているかゼロの場合、心度 *F* 分布の値が返されます。次の式で、$\nu_1 = ndf$として、$\nu_2 = ddf$として、そして$\lambda = nc$とします。次の式は、*F* 分布の CDF 関数を表しています。

$$CDF('F', x, \nu_1, \nu_2, \lambda) = \begin{cases} 0 & x < 0 \\ \sum_{j=0}^{\infty} e^{-\frac{\lambda}{2}} \frac{\left(\frac{\lambda}{2}\right)^j}{j!} P_F(x, \nu_1 + 2j, \nu_2) & x \geq 0 \end{cases}$$

この式では、$P_f(f,u_1,u_2)$は、次の式を使用した心度 *F* 分布からの確率です。

$$P_F(x, u_1, u_2) = P_B\left(\frac{u_1 x}{u_1 x + u_2}, \frac{u_1}{2}, \frac{u_2}{2}\right)$$

また、$P_B(x,a,b)$は、標準ベータ分布からの確率です。

注: *F* 分布には、*位置*または*尺度*パラメーターはありません。

例

次のステートメントは、CDF F 分布関数を示しています。

ステートメント	結果
select cdf('F', 3.32, 2, 3);	0.826393

関連項目

関数:

- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “PDF F 分布 関数” (727 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

CDF GAMMA 分布 関数

ガンマ累積確率分布の値を返します。

カテゴリ: 確率

返されるデータの種類: DOUBLE

注: QUANTILE 関数は、指定した分布から分位点を返します。QUANTILE 関数は、CDF 関数の逆数です。詳細については、“QUANTILE 関数” (793 ページ)を参照してください。

構文

CDF('GAMMA', *x*, *α* [, *λ*])

必須引数

x
確率変数を指定する数値定数、変数または式です。

データの種類 DOUBLE

α
形状パラメーターを指定する数値定数、変数または式です。

範囲 $\alpha > 0$

データの種類 DOUBLE

オプション引数

λ
尺度パラメーターを指定する数値定数、変数または式です。

デフォルト 1

範囲 $\lambda > 0$

データの種類 DOUBLE

詳細

ガンマ分布の CDF 関数は、ガンマ分布(形状パラメーターは *α*、尺度パラメーターは λ)からオブザベーションが *x* 以下となる確率を返します。

$$CDF('GAMMA', x, a, \lambda) = \begin{cases} 0 & x < 0 \\ \frac{1}{\lambda^a \Gamma(a)} \int_0^x v^{a-1} e^{-\frac{v}{\lambda}} dv & x \geq 0 \end{cases}$$

例

次のステートメントは、CDF ガンマ分布関数を示しています。

ステートメント	結果
select cdf('GAMMA', 1, 3);	0.080301

関連項目

関数:

- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “PDF GAMMA 分布 関数” (729 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

CDF 一般化 Poisson 分布 関数

一般化 Poisson 累積確率分布の値を返します。

カテゴリ:	確率
返されるデータの種類:	DOUBLE
注:	QUANTILE 関数は、指定した分布から分位点を返します。QUANTILE 関数は、CDF 関数の逆数です。詳細については、“QUANTILE 関数” (793 ページ)を参照してください。

構文

CDF('GENPOISSON', *x*, *θ*, *η*)

必須引数

x

確率変数を指定する数値定数、変数または式です。この引数は整数にする必要があります。

データの種類 DOUBLE

θ

形状パラメーターを指定する数値定数、変数または式です。

範囲 ≤5 かつ>0

データの種類 DOUBLE

η

形状パラメーターを指定する数値定数、変数または式です。

範囲 ≥0 かつ<0.95

データの種類 DOUBLE

ヒント η =0 の場合、平均と分散が θ の Poisson 分布になります。η>0 の場合、平均は$\theta \div (1-\eta)$および分散は$\theta \div (1-\eta)^3$になります。

詳細

一般化 Poisson 分布の確率質量関数の次のとおりです。

$$f(x;\theta,\eta) = \theta(\theta+\eta x)^{x-1}e^{-\theta-\eta x}/x!, \quad x = 0, 1, 2, ..., \qquad \theta > 0,\ 0 \le \eta < 1$$

η = 0 の場合、一般化 Poisson 分布は、形状パラメーターが θ の標準 Poisson 分布になります。

例

次のステートメントは、CDF 一般化 Poisson 分布関数を示しています。

ステートメント	結果
select cdf('GENPOISSON', 9, 1, .7);	0.906163

関連項目

関数:

- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “PDF 一般化 Poisson 分布 関数” (731 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

CDF GEOMETRIC 分布 関数

幾何累積確率分布の値を返します。

カテゴリ: 確率

返されるデータの種類: DOUBLE

注: QUANTILE 関数は、指定した分布から分位点を返します。QUANTILE 関数は、CDF 関数の逆数です。詳細については、“QUANTILE 関数” (793 ページ)を参照してください。

構文

CDF('GEOMETRIC', *m*, *p*)

必須引数

m
失敗数を指定する数値の確率変数です。

データの種類 DOUBLE

ヒント 10 進数の値は、整数から遠く離れている場合は切り捨てられます。

p
成功確率を指定する数値定数、変数または式です。

範囲 $0 \le p \le 1$

データの種類 DOUBLE

詳細

幾何分布の CDF 関数は、幾何分布(パラメーターは *p*)からオブザベーションが *m* 以下となる確率を返します。

$$CDF('GEOM', m, p) = \begin{cases} 0 & m < 0 \\ 1 - (1-p)^{(m+1)} & m \geq 0 \end{cases}$$

注: この分布には、*位置*または*尺度*パラメーターはありません。

例

次のステートメントは、CDF 幾何分布関数を示しています。

ステートメント	結果
select cdf('GEOMETRIC', 5, .35);	0.924581109375

関連項目

関数:

- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “PDF GEOMETRIC 分布 関数” (732 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

CDF HYPERGEOMETRIC 分布 関数

超幾何累積確率分布の値を返します。

カテゴリ:	確率
返されるデータの種類:	DOUBLE

注: QUANTILE 関数は、指定した分布から分位点を返します。QUANTILE 関数は、CDF 関数の逆数です。詳細については、“QUANTILE 関数” (793 ページ)を参照してください。

構文

CDF('HYPER', *x*, *N*, *R*, *n* [, *o*])

必須引数

x
確率変数を指定する数値定数、変数または式です。この引数は整数にする必要があります。

データの種類 DOUBLE

N
母集団サイズパラメーターを指定する数値定数、変数または式です。この引数は整数にする必要があります。

範囲 *N* = 1, 2, ...

データの種類 DOUBLE

R
対象カテゴリの項目数を指定する数値定数、変数または式です。この引数は整数にする必要があります。

範囲 *R* = 0, 1, ..., *N*

データの種類 DOUBLE

n
サンプルサイズパラメーターを指定する数値定数、変数または式です。この引数は整数にする必要があります。

範囲 *n* = 1, 2, ..., *N*

データの種類 DOUBLE

オプション引数

o
省略可能な数値のオッズ比パラメーターを指定する数値定数、変数または式です。

範囲 *o* > 0

データの種類 DOUBLE

詳細

超幾何分布の CDF 関数は、拡張超幾何分布(母集団サイズは *N*、項目数は *R*、サンプルサイズは *n*、オッズ比は *o*)からオブザベーションが *x* 以下となる確率を返します。*o* が除外されているか 1 の場合、一般超幾何分布の値が返されます。

$$CDF\ ('HYPER', x, N, R, n, o) = \begin{cases} 0 & x < max(0, R+n-N) \\ \dfrac{\sum_{i=0}^{x} \binom{R}{i}\binom{N-R}{n-i} o^i}{\sum_{j=max(0,R+n-N)}^{min(R,n)} \binom{R}{j}\binom{N-R}{n-j} o^j} & max(0, R+n-N) \le x \le min(R, n) \\ 1 & x > min(R, n) \end{cases}$$

例

次のステートメントは、CDF 超幾何分布関数を示しています。

ステートメント	結果
select cdf('HYPER', 2, 200, 50, 10);	0.523673

関連項目

関数:

- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “PDF 超幾何分布 関数” (734 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

CDF LAPLACE 分布 関数

Laplace 累積確率分布の値を返します。

カテゴリ: 確率

返されるデータの種類: DOUBLE

注: QUANTILE 関数は、指定した分布から分位点を返します。QUANTILE 関数は、CDF 関数の逆数です。詳細については、“QUANTILE 関数” (793 ページ)を参照してください。

構文

CDF('LAPLACE', *x* [,*θ*, *λ*])

必須引数

x
確率変数を指定する数値定数、変数または式です。

データの種類 DOUBLE

オプション引数

θ
位置パラメーターを指定する数値定数、変数または式です。

デフォルト 0

データの種類 DOUBLE

λ
尺度パラメーターを指定する数値定数、変数または式です。

デフォルト 1

範囲 λ > 0

データの種類 DOUBLE

詳細

Laplace 分布の CDF 関数は、Laplace 分布(位置パラメーターは θ、尺度パラメーターは λ)からオブザベーションが *x* 以下となる確率を返します。

$$CDF('LAPLACE', x, \theta, \lambda) = \begin{cases} \frac{1}{2}e^{\frac{(x-\theta)}{\lambda}} & x < \theta \\ 1 - \frac{1}{2}e^{\left(-\frac{(x-\theta)}{\lambda}\right)} & x \geq \theta \end{cases}$$

例

次のステートメントは、CDF Laplace 分布関数を示しています。

ステートメント	結果
select cdf('LAPLACE', 1);	0.81606

関連項目

関数:

- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “PDF LAPLACE 分布 関数” (736 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

CDF LOGISTIC 分布 関数

ロジスティック累積確率分布の値を返します。

カテゴリ: 確率

返されるデータの種類: DOUBLE

注: QUANTILE 関数は、指定した分布から分位点を返します。QUANTILE 関数は、CDF 関数の逆数です。詳細については、“QUANTILE 関数” (793 ページ)を参照してください。

構文

CDF('LOGISTIC', *x* [, *θ*, *λ*])

必須引数

x
確率変数を指定する数値定数、変数または式です。

データの種類 DOUBLE

オプション引数

θ
位置パラメーターを指定する数値定数、変数または式です。

デフォルト 0

データの種類 DOUBLE

λ
尺度パラメーターを指定する数値定数、変数または式です。

デフォルト 1

範囲 λ > 0

データの種類 DOUBLE

詳細

ロジスティック分布の CDF 関数は、ロジスティック分布(位置パラメーターは θ、尺度パラメーターは λ)からオブザベーションが *x* 以下となる確率を返します。

$$CDF('LOGISTIC', x, \theta, \lambda) = \frac{1}{1 + e^{\left(-\frac{x-\theta}{\lambda}\right)}}$$

例

次のステートメントは、CDF ロジスティック分布関数を示しています。

ステートメント	結果
select cdf('LOGISTIC', 1);	0.731059

関連項目

関数:

- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)

- “PDF LOGISTIC 分布 関数” (737 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

CDF LOGNORMAL 分布 関数

対数正規累積確率分布の値を返します。

カテゴリ: 確率

返されるデータの種類: DOUBLE

注: QUANTILE 関数は、指定した分布から分位点を返します。QUANTILE 関数は、CDF 関数の逆数です。詳細については、“QUANTILE 関数” (793 ページ)を参照してください。

構文

CDF('LOGNORMAL', *x* [, *θ*, *λ*])

必須引数

x
確率変数を指定する数値定数、変数または式です。

データの種類 DOUBLE

オプション引数

θ
対数尺度パラメーターを指定する数値定数、変数または式です。e(θ)は尺度パラメーターです。

デフォルト 0

データの種類 DOUBLE

λ
形状パラメーターを指定する数値定数、変数または式です。

デフォルト 1

範囲 λ > 0

データの種類 DOUBLE

詳細

対数正規分布の CDF 関数は、対数正規分布(対数尺度パラメーターは θ、形状パラメーターは λ)からオブザベーションが *x* 以下となる確率を返します。

$$CDF('LOGN', x, \theta, \lambda) = \begin{cases} 0 & x \le 0 \\ \frac{1}{\lambda\sqrt{2\,\pi}} \int_{-\infty}^{log(x)} e\left(-\frac{(v-\theta)^2}{2\lambda^2}\right) dv & x > 0 \end{cases}$$

例

次のステートメントは、CDF 対数正規分布関数を示しています。

ステートメント	結果
select cdf('LOGNORMAL', 1);	0.5

関連項目

関数:

- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “PDF LOGNORMAL 分布 関数” (739 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

CDF NEGBINOMIAL 分布 関数

負の二項累積確率分布の値を返します。

カテゴリ:	確率
返されるデータの種類:	DOUBLE
注:	QUANTILE 関数は、指定した分布から分位点を返します。QUANTILE 関数は、CDF 関数の逆数です。詳細については、“QUANTILE 関数” (793 ページ)を参照してください。

構文

CDF('NEGBINOMIAL', *m*, *p*, *n*)

必須引数

m
失敗数を数える確率変数を指定する、数値定数、変数または式です。この引数は正の整数にする必要があります。

範囲 *m* = 0, 1, ...

データの種類 DOUBLE

p
成功確率を指定する数値定数、変数または式です。

範囲 $0 \le p \le 1$

データの種類 DOUBLE

n
成功数を数える値を指定する数値定数、変数または式です。

範囲 $n > 0$

データの種類 DOUBLE

詳細

負の二項分布の CDF 関数は、負の二項分布(成功確率は *p*、成功数は *n*)からオブザベーションが *m* 以下となる確率を返します。

$$CDF('NEGB', m, p, n) = \begin{cases} 0 & m < 0 \\ p^n \sum_{j=0}^{m} \binom{n+j-1}{n-1} (1-p)^j & m \ge 0 \end{cases}$$

注: 負の二項分布には、*位置*または*尺度*パラメーターはありません。

例

次のステートメントは、CDF 負の二項分布関数を示しています。

ステートメント	結果
select cdf('NEGB', 1, .5, 2);	0.5

関連項目

関数:

- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “PDF NEGBINOMIAL 分布 関数” (740 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

CDF NORMAL 分布 関数

正規累積確率分布の値を返します。

カテゴリ: 確率

返されるデータの種類: DOUBLE

注: QUANTILE 関数は、指定した分布から分位点を返します。QUANTILE 関数は、CDF 関数の逆数です。詳細については、“QUANTILE 関数” (793 ページ)を参照してください。

構文

CDF('NORMAL', *x* [, *θ*, *λ*])

必須引数

x
確率変数を指定する数値定数、変数または式です。

データの種類 DOUBLE

オプション引数

θ
位置パラメーターを指定する数値定数、変数または式です。

デフォルト 0

データの種類 DOUBLE

λ
尺度パラメーターを指定する数値定数、変数または式です。

デフォルト 1

範囲 λ > 0

データの種類 DOUBLE

詳細

正規分布の CDF 関数は、正規分布(位置パラメーターは θ、尺度パラメーターは λ)からオブザベーションが *x* 以下となる確率を返します。

$$CDF('NORMAL', x, \theta, \lambda) = \frac{1}{\lambda\sqrt{2\,\pi}} \int_{-\infty}^{x} e\left(-\frac{(v-\theta)^2}{2\lambda^2}\right) dv$$

例

次のステートメントは、CDF 正規分布関数を示しています。

ステートメント	結果
select cdf('NORMAL', 1.96);	0.975002

関連項目

関数:

- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “PDF NORMAL 分布 関数” (742 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

CDF NORMALMIX 分布 関数

正規混合累積確率分布の値を返します。

カテゴリ: 確率

返されるデータの種類: DOUBLE

注: QUANTILE 関数は、指定した分布から分位点を返します。QUANTILE 関数は、CDF 関数の逆数です。詳細については、“QUANTILE 関数” (793 ページ)を参照してください。

構文

CDF('NORMALMIX', *x*, *n*, *p*, *m*, *s*)

必須引数

x
確率変数を指定する数値定数、変数または式です。

データの種類 DOUBLE

n
混合数を指定する数値定数、変数または式です。

範囲 *n* = 1, 2, ...

データの種類 DOUBLE

p
n 個の比率$p_1, p_2, \ldots, p_n$を指定する、数値の定数、変数または式です。ここで、$\sum_{i=1}^{i=n} p_i = 1$です。

範囲 *p* = 0, 1, ...

データの種類 DOUBLE

m
n 個の平均$m_1, m_2, \ldots, m_n$を指定する数値定数、変数または式です。

データの種類 DOUBLE

s
n 個の標準偏差$s_1, s_2, \ldots, s_n$を指定する数値定数、変数または式です。

範囲	$s > 0$
データの種類	DOUBLE

詳細

正規混合分布の CDF 関数は、正規混合分布のオブザベーションが *x* 以下となる確率を返します。

$$CDF('NORMALMIX', x, n, p, m, s) = \sum_{i=1}^{i=n} p_i \, CDF('NORMAL', x, m_i, s_i)$$

正規混合分布の重み付けは、負でないことが必要です。重み付けの合計が 1 に等しくない場合は相対的な重み付けとして扱われ、合計が 1 となるように調整されます。

注: 正規混合分布には、*位置*または*尺度*パラメーターはありません。

例

次のステートメントは、CDF 正規混合分布関数を示しています。

ステートメント	結果
select cdf('NORMALMIX',2.3, 3,.33, .33, .34, .5, 1.5, 2.5, .79, 1.6, 4.3);	0.718131

関連項目

関数:

- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “PDF NORMALMIX 分布 関数” (743 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

CDF PARETO 分布 関数

パレート累積確率分布の値を返します。

カテゴリ: 確率

返されるデータの種類: DOUBLE

注: QUANTILE 関数は、指定した分布から分位点を返します。QUANTILE 関数は、CDF 関数の逆数です。詳細については、“QUANTILE 関数” (793 ページ)を参照してください。

構文

CDF('PARETO', *x*, *α* [, *k*])

必須引数

x
確率変数を指定する数値定数、変数または式です。

データの種類 DOUBLE

α
形状パラメーターを指定する数値定数、変数または式です。

範囲 $\alpha > 0$

データの種類 DOUBLE

オプション引数

k
尺度パラメーターを指定する数値定数、変数または式です。

デフォルト 1

範囲 $k > 0$

データの種類 DOUBLE

詳細

パレート分布の CDF 関数は、パレート分布(形状パラメーターは *α*、尺度パラメーターは *k*)からオブザベーションが *x* 以下となる確率を返します。

$$CDF('PARETO', x, a, k) = \begin{cases} 0 & x < k \\ 1 - \left(\frac{k}{x}\right)^a & x \geq k \end{cases}$$

例

次のステートメントは、CDF パレート分布関数を示しています。

ステートメント	結果
select cdf('PARETO', 1 ,1);	0

関連項目

関数:

- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “PDF PARETO 分布 関数” (745 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

CDF POISSON 分布 関数

Poisson 累積確率分布の値を返します。

カテゴリ: 確率

返されるデータの種類: DOUBLE

注: QUANTILE 関数は、指定した分布から分位点を返します。QUANTILE 関数は、CDF 関数の逆数です。詳細については、“QUANTILE 関数” (793 ページ)を参照してください。

構文

CDF('POISSON', *n*, *m*)

必須引数

n
確率変数を指定する数値定数、変数または式です。この引数は整数にする必要があります。

範囲 *n* = 0, 1, ...

データの種類 DOUBLE

m
平均パラメーターを指定する数値定数、変数または式です。

範囲 *m* > 0

データの種類 DOUBLE

詳細

Poisson 分布の CDF 関数は、Poisson 分布(平均は *m*)からオブザベーションが *n* 以下となる確率を返します。

$$CDF('POISSON', n, m) = \begin{cases} 0 & n < 0 \\ \sum_{i=0}^{n} e^{-m}\frac{m^i}{i!} & n \geq 0 \end{cases}$$

注: Poisson 分布には、*位置*または*尺度*パラメーターはありません。

例

次のステートメントは、CDF Poisson 分布関数を示しています。

ステートメント	結果
select cdf('POISSON', 2, 1);	0.919699

関連項目

関数:

- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)

- “PDF POISSON 分布 関数” (747 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

CDF T 分布 関数

T 累積確率分布の値を返します。

カテゴリ: 確率

返されるデータの種類: DOUBLE

注: QUANTILE 関数は、指定した分布から分位点を返します。QUANTILE 関数は、CDF 関数の逆数です。詳細については、“QUANTILE 関数” (793 ページ)を参照してください。

構文

CDF('T', *t*, *df* [, *nc*])

必須引数

t
確率変数を指定する数値定数、変数または式です。

データの種類 DOUBLE

df
自由度を指定する数値定数、変数または式です。

範囲 *df* > 0

データの種類 DOUBLE

オプション引数

nc
省略可能な非心度パラメーターを指定する数値定数、変数または式です。

データの種類 DOUBLE

詳細

T 分布の CDF 関数は、*T* 分布(自由度は *df*、非心度パラメーターは *nc*)からオブザベーションが *x* 以下となる確率を返します。この関数では、整数以外の自由度を使用できます。*nc* が除外されているかゼロの場合、心度 *T* 分布の値が返されます。次の式で、$\nu = df$として、そして$\delta = nc$とします。

$$CDF('T', t, \nu, \delta) = \frac{1}{2^{(\nu/2-1)}\Gamma\left(\frac{\nu}{2}\right)} \int_0^{\infty} x^{\nu-1} e^{-\frac{1}{2}x^2} \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\frac{tx}{\sqrt{\nu}}} e^{-\frac{1}{2}(u-\delta)^2} du dx$$

注: *T* 分布には、*位置*または*尺度*パラメーターはありません。

例

次のステートメントは、CDF T 分布関数を示しています。

ステートメント	結果
select cdf('T', .9, 5);	0.795314

関連項目

関数:

- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “PDF T 分布 関数” (748 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

CDF TWEEDIE 分布 関数

Tweedie 累積確率分布の値を返します。

カテゴリ: 確率

返されるデータの種類: DOUBLE

注: QUANTILE 関数は、指定した分布から分位点を返します。QUANTILE 関数は、CDF 関数の逆数です。詳細については、“QUANTILE 関数” (793 ページ)を参照してください。

構文

CDF (‘TWEEDIE’, *y*, *p* [, *μ*, *φ*])

必須引数

y

確率変数を指定する数値定数、変数または式です。

範囲 *y* ≥0

データの種類 DOUBLE

注 この引数は必須です。

p>1 の場合、*y* は数値になります。*p*=1 の場合、*y* は整数になります。

p

累乗パラメーターを指定する数値定数、変数または式です。

範囲 *p* ≥1

データの種類 DOUBLE

注 この引数は必須です。

オプション引数

μ

平均パラメーターを指定する数値定数、変数または式です。

デフォルト 1

範囲 *μ*>0

データの種類 DOUBLE

φ

ばらつきのパラメーターを指定する数値定数、変数または式です。

デフォルト 1

範囲 *φ*>0

データの種類 DOUBLE

詳細

Tweedie 分布の CDF 関数は、次の式の関係がある分散と平均を使用した指数散布度モデルを返します。*variance*= $\phi \times \mu^p$

$$\int_0^y \frac{1}{y} \sum_{j=1}^{\infty} \left(\frac{y^{-j\alpha}(p-1)^{ja}}{\phi^{j(1-\alpha)(2-p)} j!\,\Gamma(-j\alpha)} \right) e\left(\frac{1}{\phi} \left(y \frac{\mu^{1-p}-1}{1-p} - \frac{\mu^{2-p}-1}{2-p} \right) \right) dy$$

前述の式には次の関係が適用されます。

$$\alpha = \frac{2-p}{1-p}$$

注: 計算された Tweedie 確率の精度は、パラメーター空間における位置に大きく依存します。通常、10 桁の精度を利用できますが、*p* の近似値が 2 の場合やファイの近似値が 0 の場合は、6 桁の精度に低下する可能性があります。

例

次のステートメントは、CDF Tweedie 分布関数を示しています。

ステートメント	結果
select cdf('TWEEDIE', .8, 5);	0.591763

関連項目

関数:

- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “PDF TWEEDIE 分布 関数” (750 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

CDF UNIFORM 分布 関数

一様累積確率分布の値を返します。

カテゴリ: 確率

返されるデータの種類: DOUBLE

注: QUANTILE 関数は、指定した分布から分位点を返します。QUANTILE 関数は、CDF 関数の逆数です。詳細については、“QUANTILE 関数” (793 ページ)を参照してください。

構文

CDF('UNIFORM', *x* [, *l*, *r*])

必須引数

x
確率変数を指定する数値定数、変数または式です。

データの種類 DOUBLE

オプション引数

l
左位置パラメーターを指定する数値定数、変数または式です。

デフォルト 0

データの種類 DOUBLE

r
右位置パラメーターを指定する数値定数、変数または式です。

デフォルト 1

範囲 *r* > *l*

データの種類 DOUBLE

詳細

一様分布の CDF 関数は、一様分布(左位置パラメーターは *l*、右位置パラメーターは *r*)からオブザベーションが *x* 以下となる確率を返します。

$$CDF('UNIFORM', x, l, r) = \begin{cases} 0 & x < l \\ \frac{x-l}{r-l} & l \le x < r \\ 1 & x \ge r \end{cases}$$

注: *l* と *r* のデフォルト値は、それぞれ 0 と 1 になります。

例

次のステートメントは、CDF 一様分布関数を示しています。

ステートメント	結果
select cdf('UNIFORM', 0.25);	0.25

関連項目

関数:

- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “PDF UNIFORM 分布 関数” (752 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

CDF WALD (逆ガウス)分布 関数

Wald (逆ガウスとも呼ばれる)累積確率分布から値を返します。

カテゴリ:	確率
返されるデータの種類:	DOUBLE
注:	QUANTILE 関数は、指定した分布から分位点を返します。QUANTILE 関数は、CDF 関数の逆数です。詳細については、“QUANTILE 関数” (793 ページ)を参照してください。

構文

CDF('WALD', *x*, *λ* [, *μ*])
CDF('IGAUSS', *x*, *λ* [, *μ*])

必須引数

x
確率変数を指定する数値定数、変数または式です。

データの種類 DOUBLE

λ
形状パラメーターを指定する数値定数、変数または式です。

範囲 λ > 0

データの種類 DOUBLE

オプション引数

μ
平均パラメーターを指定する数値定数、変数または式です。

デフォルト 1

範囲 μ > 0

データの種類 DOUBLE

詳細

Wald 分布の CDF 関数は、Wald 分布(形状パラメーターは λ)からオブザベーションが *x* 以下となる確率を返します。

$$Fx(x) = \Phi\left\{\sqrt{\frac{\lambda}{x}}\left(\frac{x}{\mu} - 1\right)\right\} + e^{2\lambda/\mu}\Phi\left\{-\sqrt{\frac{\lambda}{x}}\left(\frac{x}{\mu} + 1\right)\right\}$$

式では、Φ(.)は、正規累積分布関数です。x≤0 の場合、CDF は 0 になります。

例

次のステートメントは、CDF Wald 分布関数を示しています。

ステートメント	結果
select cdf('WALD', 1, 2);	0.627698

関連項目

関数:

- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “PDF Wald (逆ガウス)分布 関数” (754 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

CDF WEIBULL 分布 関数

Weibull 累積確率分布の値を返します。

カテゴリ: 確率

返されるデータの種類: DOUBLE

注: QUANTILE 関数は、指定した分布から分位点を返します。QUANTILE 関数は、CDF 関数の逆数です。詳細については、“QUANTILE 関数” (793 ページ)を参照してください。

構文

CDF('WEIBULL', *x*, *α* [, *λ*])

必須引数

x
確率変数を指定する数値定数、変数または式です。

データの種類 DOUBLE

α
形状パラメーターを指定する数値定数、変数または式です。

範囲 $\alpha > 0$

データの種類 DOUBLE

オプション引数

λ

尺度パラメーターを指定する数値定数、変数または式です。

デフォルト	1
範囲	λ > 0
データの種類	DOUBLE

詳細

Weibull 分布の CDF 関数は、Weibull 分布(形状パラメーターは *a*、尺度パラメーターは λ)からオブザベーションが *x* 以下となる確率を返します。

$$CDF('WEIBULL', x, a, \lambda) = \begin{cases} 0 & x < 0 \\ 1 - e^{-\left(\frac{x}{\lambda}\right)^a} & x \geq 0 \end{cases}$$

例

次のステートメントは、CDF Weibull 分布関数を示しています。

ステートメント	結果
select cdf('WEIBULL', 1, 2);	0.632121

関連項目

関数:

- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “PDF WEIBULL 分布 関数” (755 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

CEIL 関数

数値式以上の最小の整数を返します。

カテゴリ: 切り捨て

返されるデータの種類: DECIMAL、DOUBLE、NUMERIC

構文

CEIL(*expression*)

必須引数

expression
数値に評価される有効な式を指定します。

データの種類 DECIMAL、DOUBLE、NUMERIC

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

expression が null の場合、CEILING 関数は null を返します。結果が引数のデータ型の範囲に収まらない数値の場合、CEIL 関数は失敗します。

引数が DECIMAL の場合、結果は DECIMAL になります。それ以外の場合、引数は (まだ変換されていなければ) DOUBLE に変換され、結果は DOUBLE になります。

比較

CEILZ 関数とは異なり、CEIL 関数は結果をファジー処理します。引数が整数の 1E-12 内にある場合、CEIL 関数はその整数に等しくなるように結果をファジー処理します。CEILZ 関数は結果をファジー処理しません。そのため、CEILZ 関数では予期しない結果になる可能性があります。

例

次のステートメントは、CEIL 関数を示しています。

ステートメント	結果
select ceil(-2.4);	-2
select ceil(1+1.e-11);	2
select ceil(-1+1.e-11);	0
select ceil(1+1.e-13);	1

関連項目

関数:

- “CEILZ 関数” (381 ページ)
- “FLOOR 関数” (542 ページ)
- “FLOORZ 関数” (543 ページ)

CEILZ 関数

ゼロファジーを使用して、引数より大きいか等しい整数のうち最小の値を返します。

カテゴリ: 切り捨て

返されるデータの種類: DOUBLE

構文

CEILZ(*expression*)

必須引数

expression
数値に評価される有効な式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ).

比較

CEIL 関数とは異なり、CEILZ 関数はゼロファジーを使用します。引数が整数の 1E-12 内にある場合、CEIL 関数はその整数に等しくなるように結果をファジー処理します。CEILZ 関数は結果をファジー処理しません。そのため、CEILZ 関数では予期しない結果になる可能性があります。

例

次のステートメントは、CEILZ 関数を示しています。

ステートメント	結果
select ceilz(2.1);	3
select ceilz(3);	3
select ceilz(1+1.e-11);	2
select ceilz(223.456);	224
select ceilz(-223.456);	-223

関連項目

関数:

- “CEIL 関数” (380 ページ)
- “FLOOR 関数” (542 ページ)
- “FLOORZ 関数” (543 ページ)

CHARACTER_LENGTH 関数

任意のデータ型の文字列の文字数を返します。

カテゴリ: 文字

別名: CHAR_LENGTH

返されるデータの種類: BIGINT

注: CHARACTER_LENGTH 関数は、末尾の空白をカウントします。末尾の空白をカウントしない場合は、“TRIM 関数” (866 ページ)を使用して削除する必要があります。

構文

CHARACTER_LENGTH(*expression*)

必須引数

expression
任意の有効な式を指定します。

データの種類 すべてのデータ型が有効です。

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

比較

CHARACTER_LENGTH 関数は文字列の文字数を返しますが、マルチバイト文字は 1 文字としてカウントします。OCTET_LENGTH 関数は、マルチバイト文字を複数の文字としてカウントします。

例

次のステートメントは、CHARACTER_LENGTH 関数を示しています。

ステートメント	結果
select character_length('December');	8
select character_length('FedSQL ');	7

関連項目

関数:

- “OCTET_LENGTH 関数” (708 ページ)

CNONCT 関数

カイ 2 乗分布の非心度パラメーターを返します。

カテゴリ: 数学

返されるデータの種類: DOUBLE

構文

CNONCT(*x*, *df*, *probability*)

必須引数

x
数値の確率変数です。

範囲 $x \geq 0$

データの種類 DOUBLE

df
数値の自由度パラメーターです。

範囲 $df > 0$

データの種類 DOUBLE

probability
確率です。

範囲 $0 < probability < 1$

データの種類 DOUBLE

詳細

CNONCT 関数は、非心度カイ 2 乗分布(パラメーターは *x*、*df*、*nc*)から負でない非心度パラメーターを返します。*probability* が心度カイ 2 乗分布(パラメーターは *x* および *df*)からの確率よりも大きい場合、この問題の平方根は存在しません。この場合、欠損値が返されます。式の負でない平方根 *nc* を検索するには、ニュートン型のアルゴリズムを使用します。

$$P_c(x|\mathrm{df}, nc) - prob = 0$$

前述の式には次の関係が適用されます。

$$P_c(x|\mathrm{df},\mathrm{nc}) = \varepsilon^{\frac{-nc}{2}} \sum_{j=0}^{\infty} \frac{\left(\frac{nc}{2}\right)^j}{j!} P_g\left(\frac{x}{2}\middle|\frac{df}{2}+j\right)$$

前述の式には次の関係が適用されます。

$$P_g(x|a)$$

次の式によって得られるガンマ分布の確率です。

$$P_g(x|a) = \frac{1}{\Gamma(a)}\int_0^x t^{a-1}\varepsilon^{-t}dt$$

アルゴリズムで固定小数点を収束できない場合、欠損値が返されます。

例

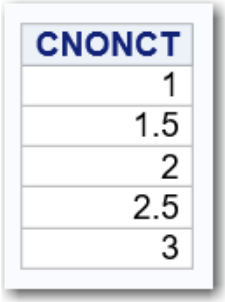
この例は、CNONCT 関数を示しています。

```
proc ds2;
data test /overwrite=yes;
  dcl double x df nc prob;
  method init();
    x=2;
    df=4;
    do nc=1 to 3 by .5;
      prob=probchi(x, df, nc);
      output;
    end;
  end;
enddata;
run;
quit;

proc fedsql;
 select * from test;
 select cnonct(x, df, prob) from test;
quit;
```

アウトプット 4.1　テーブル"Test"のコンテンツ

x	df	nc	prob
2	4	1	0.18611
2	4	1.5	0.155915
2	4	2	0.130477
2	4	2.5	0.109074
2	4	3	0.091092

アウトプット 4.2　CNONCT 関数の結果



CNONCT
1
1.5
2
2.5
3

関連項目

関数:

- “FNONCT 関数” (546 ページ)
- “TNONCT 関数” (856 ページ)

COALESCE 関数

数値の引数のリストから最初の非 null または非欠損値を返します。

カテゴリ:　数学

返されるデータの種類:　DOUBLE

構文

COALESCE(*expression*[, ...*expression*])

必須引数

expression
数値に評価される有効な式を指定します。

データの種類　DOUBLE

参照項目:　“<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ).

詳細

COALESCE では、1 つ以上の数値式を使用できます。COALESCE 関数は、リストされている順に各式の値を確認し、最初の非 null または非欠損値を返します。リストされている値が 1 つだけの場合、COALESCE 関数はその引数の値を返します。すべての式のすべての値が null または欠損値の場合、COALESCE 関数は、ANSI モードまたは SAS モードのどちらであるかに応じて、null または欠損値を返します。詳細については、“FedSQL が null および SAS 欠損値を処理する方法” (21 ページ)を参照してください。

比較

COALESCE 関数は数値式を検索するのに対し、COALESCEC 関数は文字式を検索します。

例

次のステートメントは、COALESCE 関数を示しています。

ステートメント	結果
select COALESCE(., .A, 33, 22, 44, .);	33

関連項目

関数:

- “COALESCEC 関数” (387 ページ)

COALESCEC 関数

文字引数のリストから最初の非 null または非欠損値を返します。

カテゴリ:	文字
返されるデータの種類:	CHAR、NCHAR、NVARCHAR、VARCHAR
参照項目:	“FedSQL での文字型を返す関数の使用” (260 ページ)

構文

COALESCEC(*expression*[, ...*expression*])

必須引数

expression
文字列に評価されるか強制変換できる有効な式を指定します。

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ).

詳細

COALESCEC では、1 つ以上の文字式を使用できます。COALESCEC 関数は、リストされている順に各式の値を確認し、最初の非 null または非欠損値を返します。リストされている値が 1 つだけの場合、COALESCEC 関数はその式の値を返します。すべての式のすべての値が null または欠損値の場合、COALESCEC 関数は、ANSI モードまたは SAS モードのどちらであるかに応じて、null または欠損値を返します。詳細については、“FedSQL が null および SAS 欠損値を処理する方法” (21 ページ)を参照してください。

比較

COALESCEC 関数は文字式を検索するのに対し、COALESCE 関数は数値式を検索します。

例

次のステートメントは、COALESCEC 関数を示しています。

ステートメント	結果
select coalescec('', 'Hello');	Hello
select coalescec('', 'Goodbye', 'Hello');	Goodbye

関連項目

関数:

- “COALESCE 関数” (386 ページ)

COMB 関数

n 個の要素を同時に *r* 個使用するときの組み合わせの数を計算します。

カテゴリ: 組み合わせ

返されるデータの種類: DOUBLE

構文

COMB(*n*, *r*)

必須引数

n
負でない整数で要素の合計数を表します。サンプルはこの中から選ばれます。

データの種類 DOUBLE

r
負でない整数で選ばれた要素の数を表します。

制限事項 $r \le n$

データの種類 DOUBLE

詳細

COMB 関数の数学的表現は、次の式によって得られます。

$$COMB\,(n, r) = \binom{n}{r} = \frac{n!}{r!\,(n-r)!}$$

前述の式で、$n \ge 0$、$r \ge 0$ および $n \ge r$ です。

式による計算ができない場合は欠損値が返されます。やや大きな値の場合、COMB 関数を計算できないことがあります。

例

次のステートメントは、COMB 関数を示しています。

ステートメント	結果
select comb(27, 2);	351

関連項目

関数:

- “FACT 関数” (464 ページ)

COMPARE 関数

2 つの文字列を比較し、異なる文字が検出された場合には最も左にある文字の位置を返し、異なる文字が検出されない場合には 0 を返します。

カテゴリ: 文字

返されるデータの種類: DOUBLE

構文

COMPARE(*string-1*, *string-2*[, *modifiers*])

必須引数

string-1
文字列を評価するか、文字列に強制変換できる文字定数、変数または式を指定します。

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

string-2
文字列を評価するか、文字列に強制変換できる文字定数、変数または式を指定します。

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

オプション引数

modifiers
COMPARE 関数のアクションを変更できる文字列を指定します。

次の 1 つ以上の文字を有効な修飾子として使用できます。

i
string-1 および *string-2* の大文字と小文字を区別しません。

l
値を比較する前に *string-1* および *string-2* の先頭の空白を削除します。

n
名前リテラルの引数から引用符を削除し、*string-1* および *string-2* の大文字と小文字を区別しません。名前リテラルは、引用符内の文字列として表される名前トークンで、大文字または小文字の *n* が後に続きます。名前リテラルにより、テーブルまたは変数名に使用できなかった特殊文字(空白文字を含む)を使用できるようになります。COMPARE で文字列が名前リテラルとして認識されるように、最初の文字を引用符にする必要があります。

:
(コロン) *string-1* と *string-2* の短い方の文字列の長さまたは 1 文字(どちらか大きい方)になるように長い方の文字列を切り捨てます。この修飾子を指定しない場合、長い方の文字列の長さと同じになるように短い方の文字列に空白が埋め込まれます。

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

ヒント COMPARE は、修飾子として使用される空白を無視します。

詳細

基本

COMPARE 関数では、修飾子の出現順序に意味があります。

- "LN"は各文字列から先頭の空白を削除した後、名前リテラルから引用符を削除します。
- "NL"は名前リテラルから引用符を削除した後、各文字列から先頭の空白を削除します。

COMPARE 関数では、*string–1* と *string–2* に違いがない場合はゼロ値を返します。引数が異なる場合、次のようになります。

- 並べ替え順序で *string–1* が *string–2* よりも前にある場合は結果の符合が負になり、*string–1* が *string–2* よりも後にある場合は正になります。

- 結果の大きさは、文字列が異なる最も左側の文字の位置と等しくなります。

DBCS の互換性

この関数に相当する DBCS 関数は、KCOMPARE です。COMPARE 関数と KCOMPARE 関数には若干の違いがあります。どちらの関数もさまざまな数の引数を受け入れますが、第 3 引数の使用法には互換性がありません。次の例は、構文の違いを示しています。

COMPARE(*string-1*, *string-2*[, *modifiers*])

KCOMPARE(*string-1*[, *position*[, *count*]], *string-2*)

詳細については、*"KCOMPARE 関数" (SAS 各国語サポート(NLS): リファレンスガイド)* を参照してください。

例

例 1: 2 つの文字列を比較するときの比較順序の理解

次のプログラムでは、COMPARE 関数を使用して 2 つの文字列を比較します。

```
proc fedsql;
create table test(string1 CHAR(8) having informat $char8. format $char8.,
                  string2 CHAR(8) having informat $char8. format $char8.,
                modifiers CHAR(8) having informat $char8. format $char8.);

 insert into test (string1, string2) values ('12345678', '12345678');
 insert into test (string1, string2) values ('123', 'abc');
 insert into test (string1, string2) values ('abc', 'abx');
 insert into test (string1, string2) values ('xyz', 'abcdef');
 insert into test (string1, string2) values ('aBc', 'abc');
 insert into test (string1, string2, modifiers) values ('aBc', 'AbC', 'i');
 insert into test (string1, string2, modifiers) values ('   abc  ', 'abc', '  ');
 insert into test (string1, string2, modifiers) values ('   abc  ', 'abc', '1');
 insert into test (string1, string2, modifiers) values (' abc    ', '   abx  ', '  ');
 insert into test (string1, string2, modifiers) values(' abc    ', '   abx  ', '1');

select string1, string2, modifiers, compare(string1, string2, modifiers)
 from test;
quit;
```

アウトプット 4.3　COMPARE 関数を使用して 2 つの文字列を比較した結果

STRING1	STRING2	MODIFIERS	COMPARE
12345678	12345678		0
123	abc		-1
abc	abx		-3
xyz	abcdef		1
aBc	abc		-2
aBc	AbC	i	0
abc	abc		-1
abc	abc	1	-1
abc	abx		2
abc	abx	1	2

例 2: COMPARE 関数を使用した文字列の切り捨て

次のプログラムでは、: (コロン)修飾子を使用して文字列を切り捨てています。

```
proc fedsql;
drop table test2 force;
create table test2 (pad1 double,
                    pad2 double,
                    truncate1 double,
                    truncate2 double,
                    blank double);

 insert into test2 (pad1, pad2, truncate1, truncate2, blank) values
(
    (select compare('abc','abc           ')),
    (select compare('abc','abcdef        ')),
    (select compare('abc','abcdef',':')),
    (select compare('abcdef','abc',':')),
    (select compare('','abc',         ':'))
 );

  select * from test2;
quit;
```

アウトプット 4.4　コロン修飾子を使用して文字列を切り捨てた結果

PAD1	PAD2	TRUNCATE1	TRUNCATE2	BLANK
0	-4	0	0	-1

COMPBL 関数

文字列内の単語間にある複数の空白を取り除きます。

カテゴリ: 文字

返されるデータの種類: CHAR、NCHAR、NVARCHAR、VARCHAR

構文

COMPBL(*character-expression*)

必須引数

character-expression
文字列を評価するか、文字列に強制変換できる有効な式であり、取り除く文字列を指定します。

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

文字列中に 2 つ以上の空白が連続して出現するたびに、COMPBL 関数はそれらを 1 つの空白に変換して、複数の空白を削除します。

比較

COMPRESS 関数は、特定の文字が文字列に出現するたびにそれを削除します。ソース文字列から削除する文字として空白を指定した場合、COMPRESS 関数は COMPBL 関数と同様の動作をします。ただし、COMPRESS 関数は、ソース文字列からすべての空白を削除します。COMPBL 関数は、複数の空白を 1 つの空白に圧縮するため、1 つの空白には影響しません。

例

次のステートメントは、COMPBL 関数を示しています。

ステートメント	結果
	----+----1----+----2--
select compbl('January Status');	January Status
select compbl('125 E. Main St.');	125 E. Main St.

関連項目

関数:

- “COMPRESS 関数” (396 ページ)

COMPOUND 関数

複利パラメーターを返します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

COMPOUND(*a*, *f*, *r*, *n*)

必須引数

a
初期の金額を指定します。

範囲 $a \geq 0$

データの種類 DOUBLE

f
将来の金額を指定します(*n* 期間の終了時)。

範囲 $f \geq 0$

データの種類 DOUBLE

r
期間単位の金利を分数で指定します。

範囲 $r \geq 0$

データの種類 DOUBLE

n
複利期間数を指定します。

範囲 $n \geq 0$

データの種類 DOUBLE

詳細

COMPOUND 関数は、複利計算の 4 つの引数のリストの欠損引数を返します。これらの引数には次の式の関係があります。

$$f = a(1 + r)^n$$

引数は 1 つを欠損値とする必要があります。他の 3 つの値から複利パラメーターが計算されます。結果を変換して丸めた数字にする調整は行われません。

n=0 のとき、

$$f = a$$

および

$$(1 + r)^n$$

は 1 です。

注: *r* を欠損値に選ぶと COMPOUND からエラーが返されます。

例

次のステートメントは、COMPOUND 関数を示しています。$2000 の投資で名目年利 9%、30 か月複利運用したときの累計額は次の式で表します。

```
select compound(2000,.,0.09/12,30);
```

返される値は 2502.5435276 です。

COMPRESS 関数

元の文字から指定した文字を削除した文字列を返します。

カテゴリ: 文字

返されるデータの種類: CHAR、NCHAR、NVARCHAR、VARCHAR

構文

COMPRESS(*character-expression*[, *character-list-expression*] [,*modifier(s)*])

必須引数

character-expression
文字式に評価され、指定された文字が削除される有効な式を指定します。

要件 文字のリテラル文字列は単一引用符で囲みます。

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

オプション引数

character-list-expression
文字リストを初期化する変数または任意の有効な式を指定します。

デフォルトでは、このリスト内の文字が *character-expression* から削除されます。

要件 文字のリテラル文字列は単一引用符で囲みます。

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

modifier
文字定数、変数または式を指定します。空白でない文字はそれぞれ COMPRESS 関数のアクションを変更します。空白は無視されます。修飾子では大文字と小文字は区別されません。

修飾子として使用できる文字は次のとおりです。

A
文字のリストにアルファベット文字を追加します。

C
文字のリストに制御文字を追加します。

D
文字のリストに数字を追加します。

F
アンダースコア文字および英文字を文字リストに追加します。

G
文字のリストにグラフィカル文字を追加します。グラフィカル文字は、現在のロケールの文字で、紙面に印刷するとイメージを生成します。

L
文字のリストに水平タブを追加します。

I
保持または削除する文字の大文字と小文字を区別しません。

K
リスト内の文字を削除するのではなく、保持します。

I
小文字を文字リストに追加します。

N
数字、アンダースコア文字および英文字を文字リストに追加します。

O
COMPRESS 関数が呼び出されるたびに第 2 引数および第 3 引数を実行するのではなく、一度だけ実行します。DATA ステップ(WHERE 句を除く)または SQL プロシジャで O 修飾子を使用すると、第 2 引数および第 3 引数が変更されないループで COMPRESS を呼び出すときに、より迅速に実行できます。

P
文字のリストに句読点を追加します。

S
文字のリストに空白文字(空白、水平タブ、垂直タブ、キャリッジリターン、ラインフィード、フォームフィード、NBSP ('A0'x - 10 進数の ASCII コードで 160))を追加します。

T
第 1 引数と第 2 引数から末尾の空白を取り除きます。

U
大文字を文字リストに追加します。

W
印刷可能文字を文字リストに追加します。

X
文字のリストに 16 進文字を追加します。

ヒント　修飾子が定数の場合、単一引用符で囲みます。一組の引用符で複数の定数を指定します。修飾子を変数または式として表すこともできます。

詳細

COMPRESS 関数は、第 2 引数内の文字に加えて、修飾子で指定される文字を含む文字を保持または削除する文字リストをコンパイルします。

引数の数に基づいて、COMPRESS 関数は次のように機能します。

引数の数	結果
第 1 引数のみ。*source*	すべての空白が削除されます。引数のすべてが空白の場合、結果は長さがゼロの文字列となります。固定長の文字変数に結果を割り当てると、その変数の値は定義した長さとなるよう空白が埋め込まれます。
2 つの引数。*source* および *chars*	第 2 引数に現れるすべての文字が結果から削除されます。
3 つの引数。*source*、*chars* および *modifier* (または *modifiers*)	K 修飾子(第 3 引数に指定)は、第 2 引数内の文字を結果に保持するか、削除するかを決定します。

これは、第 3 引数の効果を示す例です。D 修飾子は数字を指定します。次の関数呼び出しは、どちらも結果から数字を削除します。

```
compress(source, "1234567890");
compress(source, '', "d");
```

数字、プラス記号またはマイナス記号を削除するには、次の関数呼び出しを使用します。

```
COMPRESS(source, "1234567890+-");
compress(source, "+-", "d");
```

COMPRESS 関数は、第 2 引数で null または欠損(".")引数を許可します。null または欠損している引数は長さがゼロの文字列として扱われます。長さが 0 の文字列は空の文字列です。

```
compress(source, null);
compress(source, '');
```

例

これらの例は、COMPRESS 関数を使用して、文字列から空白、文字、小文字、スペース文字を削除する方法を示しています。最後の例は、COMPRESS を使用して文字列から文字を保持する方法を示しています。

ステートメント	結果
	----+----1----+----2----+----
select compress('AB C D ');	ABCD
select compress('AB C D ', 'A ');	BCD
select compress('123-4567-8901 e 234-5678-9012 i', 'aeiou');	123-4567-8901 234-5678-9012

ステートメント	結果
select compress('1 2 3 4 5', ' ','s');	12345
select compress('Math A English B Physics A', 'ABCD', 'k');	ABA

関連項目

関数:

- “COMPBL 関数” (394 ページ)
- “LEFT 関数” (635 ページ)
- “TRIM 関数” (866 ページ)

CONSTANT 関数

マシン定数および数学定数を計算します。

カテゴリ: 数学

返されるデータの種類: DOUBLE

構文

CONSTANT(*constant*[, *parameter*])

必須引数

constant
返す定数を特定する文字定数、変数または式を指定します。

有効な定数は次のとおりです。

定数	説明
'E'	自然数の底
'EULER'	Euler 定数
'PI'	PI

定数	説明
'EXACTINT' [,nbytes]	正確な整数
'BIG'	最大倍精度浮動小数点数
'LOGBIG' [,base]	*base* を底とする BIG の対数
'SQRTBIG'	BIG の平方根
'SMALL'	最小倍精度浮動小数点数
'LOGSMALL' [,base]	*base* を底とする SMALL の対数
'SQRTSMALL'	SMALL の平方根
'MACEPS'	マシン精度の定数
'LOGMACEPS' [,base]	*base* を底とする MACEPS の対数
'SQRTMACEPS'	MACEPS の平方根

オプション引数

parameter
数値パラメーターで、*constant* で指定された一部の定数と一緒にオプション引数として使用できます。この *parameter* を使用すると、CONSTANT 関数の機能が変更されます。

詳細

概要

注意
一部の動作環境では、ランタイムライブラリに制限があるために、ハードウェアが持つ浮動小数点数を最大限に活用できません。 そのような場合、CONSTANT 関数はラインタイムライブラリの制限内で使用できる値を返そうとします。たとえば、ランタイムライブラリで EXP(LOG(CONSTANT('BIG')))を計算できない場合は、CONSTANT('LOGBIG') で LOG(CONSTANT('BIG'))と同じ値が返されず、EXP(CONSTANT('LOGBIG'))を計算できる値が返されます。

自然数の底

CONSTANT('E')

自然数の底は次の式で表されます。

$$\lim_{x \to 0} (1+x)^{\frac{1}{x}} \approx 2.718281828459045$$

Euler 定数

CONSTANT('EULER')

Euler の定数は次の式で表されます。

$$\lim_{n \to \infty} \left\{ \sum_{j=1}^{j=n} \frac{1}{j} - \log(n) \right\} \approx 0.577215664901532860$$

PI

CONSTANT('PI')

PI(円周率)は、円の周の長さと直径の比です。この定数を計算する式は多数あります。この数列を表す式の 1 例は次のとおりです。

$$4 \sum_{j=0}^{j=\infty} \frac{(-1)^j}{2j+1} \approx 3.14159265358979323846$$

正確な整数

CONSTANT('EXACTINT'[, *nbytes*])

引数

nbytes
バイト数を表す数値です。

デフォルト 8

範囲 $2 \le nbytes \le 8$

正確な整数は、絶対値で *k* 以下のすべての整数を、長さが *nbytes* の SAS 数値変数で正確に表現できる最大整数 *k* です。この情報は、容量を節約するために SAS 数値変数をデフォルトの 8 バイトから小さいバイト数に調整するときにあらかじめ知っておくと役に立つ情報です。

最大倍精度浮動小数点数

CONSTANT('BIG')

使用しているコンピューターで表現できる最大倍精度浮動小数点数(8 バイト)を返します。

BIG の対数

CONSTANT('LOGBIG'[, *base*])

引数

base
対数の底となる数値です。

デフォルト　自然数の底、E。

制限事項　指定する *base* は 1+SQRTMACEPS の値より大きくする必要があります。

base を底として、使用しているコンピューターで表現できる最大倍精度浮動小数点数(8 バイト)の対数を返します。

CONSTANT('LOGBIG', *base*)以下に累乗された指定の *base* は、累乗演算(**)を使っても、オーバーフローを発生させず安全に累乗できます。

浮動小数点数は、CONSTANT('LOGBIG')以下の任意の浮動小数点数で累乗関数 EXP を使っても、オーバーフローを発生させず安全に累乗できます。

BIG の平方根

CONSTANT('SQRTBIG')

使用しているコンピューターで表現できる最大倍精度浮動小数点数(8 バイト)の平方根を返します。

浮動小数点数は、CONSTANT('SQRTBIG')以下の任意の浮動小数点数で、オーバーフローを発生させず安全に平方根を求めることができます。

最小倍精度浮動小数点数

CONSTANT('SMALL')

使用しているコンピューターで表現できる最小倍精度浮動小数点数(8 バイト)を返します。

SMALL の対数

CONSTANT('LOGSMALL'[, *base*])

引数

base
対数の底となる数値です。

デフォルト　自然数の底、E。

制限事項　指定する *base* は 1+SQRTMACEPS の値より大きくする必要があります。

base を底として、使用しているコンピューターで表現できる最小倍精度浮動小数点数(8 バイト)の対数を返します。

CONSTANT('LOGSMALL', *base*)以上に累乗された指定の *base* は、累乗演算(**)を使って安全に累乗できます。アンダーフローまたは 0 にはなりません。

浮動小数点数は、CONSTANT('LOGSMALL')以上の任意の浮動小数点数で累乗関数 EXP を使って安全に累乗できます。アンダーフローまたは 0 にはなりません。

SMALL の平方根

CONSTANT('SQRTSMALL')

コンピューターで表現できる最小倍精度浮動小数点数(8 バイト)の平方根を返します。

浮動小数点数は、CONSTANT('SQRTBIG')以上の任意の浮動小数点数で、安全に平方根を求めることができます。アンダーフローまたはゼロにはなりません。

マシンの精度

CONSTANT('MACEPS')

この場合、$1+\varepsilon > 1$のように、ある整数 *j* に対して最小倍精度浮動小数点数(8 バイト) $\varepsilon = 2^{-j}$が返されます。

この定数は有限精度の計算を行うときに重要です。

MACEPS の対数

CONSTANT('LOGMACEPS'[, *base*])

引数

base
対数の底となる数値です。

デフォルト　自然数の底、E。

制限事項　指定する *base* は 1+SQRTMACEPS の値より大きくする必要があります。

base を底とする CONSTANT('MACEPS')の対数を返します。

MACEPS の平方根

CONSTANT('SQRTMACEPS')

CONSTANT('MACEPS')の平方根を返します。

例

次の例では、CONSTANT 関数を使用して、さまざまな定数の値を返します。

```
create table test(constant char(10), output double);
  insert into test(constant, output) values ('E', (select constant('E')));
  insert into test(constant, output) values ('EULER',(select constant('EULER')));
  insert into test(constant, output) values ('PI',(select constant('PI')));
  insert into test(constant, output) values ('EXACTINT', (select constant('EXACTINT')));
  insert into test(constant, output) values ('BIG', (select constant('BIG')));
  insert into test(constant, output) values ('LOGBIG',(select constant('LOGBIG')));
  insert into test(constant, output) values ('SQRTBIG', (select constant('SQRTBIG')));
  insert into test(constant, output) values ('SMALL', (select constant('SMALL')));
```

```
insert into test(constant, output) values ('LOGSMALL', (select constant('LOGSMALL')));
insert into test(constant, output) values ('SQRTSMALL', (select constant('SQRTSMALL')));
insert into test(constant, output) values ('MACEPS', (select constant('MACEPS')));
insert into test(constant, output) values ('LOGMACEPS', (select constant('LOGMACEPS')));
insert into test(constant, output) values ('SQRTMACEPS',(select constant('SQRTMACEPS')));

select * from test;
```

アウトプット 4.5　指定された定数に対して返される値

CONSTANT	OUTPUT
E	2.718282
EULER	0.577216
PI	3.141593
EXACTINT	9.007E15
BIG	1.8E308
LOGBIG	709.7827
SQRTBIG	1.34E154
SMALL	223E-310
LOGSMALL	-708.396
SQRTSMALL	149E-156
MACEPS	2.22E-16
LOGMACEPS	-36.0437
SQRTMACEPS	1.49E-8

CONVX 関数

列挙キャッシュフローのコンベクシティを返します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

CONVX(*y*, *f*, *c(1)*, ..., *c(k)*)

必須引数

y
期間当たりの有効満期利回りを指定します。

範囲 $0 < y < 1$

データの種類 DOUBLE

注 この引数の入力としてゼロまたは負の数が渡された場合、エラーメッセージはログに書き込まれず、欠損値が出力として返されます。

f
期間当たりのキャッシュフローの頻度を指定します。

範囲 $f > 0$

データの種類 DOUBLE

注 この引数の入力として負の数が渡された場合、エラーメッセージはログに書き込まれず、欠損値が出力として返されます。

c(1), ... , c(k)
キャッシュフローのリストを指定します。

データの種類 DOUBLE

詳細

CONVX 関数は、次の式から値を返します。

$$C = \sum_{k=1}^{K} \frac{k(k+f)\dfrac{c(k)}{(1+y)^{\frac{k}{f}}}}{P\left((1+y)^2\right)f^2}$$

前述の式には次の関係が適用されます。

$$P = \sum_{k=1}^{K} \frac{c(k)}{(1+y)^{\frac{k}{f}}}$$

例

次のステートメントは、CONVX 関数を示しています。

ステートメント	結果
select convx(1./20,.1,.33,.44,	976.9411

ステートメント	結果
.55,.49,.50,.22,.4,.8,.01,.36,.2,.4);	

関連項目

関数:

- “CONVXP 関数” (407 ページ)

CONVXP 関数

債権などの定期キャッシュフローストリームのコンベクシティを 返します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

CONVXP(*A*, *c*, *n*, *K*, k_0, *y*)

必須引数

A
額面価格を指定します。

範囲 $A > 0$

A > 0

データの種類 DOUBLE

c
期間当たりの名目クーポンレートを小数で指定します。

範囲 0 ≤ *c* < 1

データの種類 DOUBLE

n
期間当たりのクーポン数を指定します。

範囲 *n* > 0

データの種類　DOUBLE

K
クーポンの残数を指定します。

範囲　*K* > 0

データの種類　DOUBLE

k_0
現在の日付から最初のクーポン日までの時間を指定します。期間数で表します。

範囲　$0 < k_0 \leq \frac{1}{n}$

$0 < k_0 \leq 1/n$

データの種類　DOUBLE

注　ゼロ、負の数、または 1/*n* より大きい分数がこの引数の入力として渡された場合、エラーメッセージはログに書き込まれず、欠損値が出力として返されます。

y
期間当たりの名目満期利回りを指定します。

範囲　*y* > 0

データの種類　DOUBLE

注　この引数の入力としてゼロまたは負の数が渡された場合、エラーメッセージはログに書き込まれず、欠損値が出力として返されます。

詳細

CONVXP 関数は、次の式から値を返します。

$$C = \frac{1}{n^2}\left(\frac{\sum_{k=1}^{K} t_k(t_k+1)\frac{c(k)}{\left(1+\frac{y}{n}\right)^{t_k}}}{P\left(1+\frac{y}{n}\right)^2}\right)$$

前述の式には次の関係が適用されます。

$$t_k = nk_0 + k - 1$$

$$c(k) = \frac{c}{n}A \quad for\ k = 1, \ldots, K-1$$

$$c(K) = \left(1 + \frac{c}{n}\right)A$$

前述の式には次の関係が適用されます。

$$P = \sum_{k=1}^{K} \frac{c(k)}{\left(1 + \frac{y}{n}\right)^{tk}}$$

例

次のステートメントは、CONVXP 関数を示しています。

ステートメント	結果
select convxp(1000,.01,4,14,.33/2,.08);	11.729

関連項目

関数:

- “CONVX 関数” (405 ページ)

COS 関数

余弦をラジアンで返します。

カテゴリ: 三角関数

返されるデータの種類: DOUBLE

構文

COS(*expression*)

必須引数

expression
数値に評価される有効な SQL 式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

例

次のステートメントは、COS 関数を示しています。

ステートメント	結果
select cos(0.5);	0.877583
select cos(0);	1
select cos(3.14159/3);	0.50003

関連項目

関数:

- “ARCOS 関数” (302 ページ)
- “COSH 関数” (410 ページ)
- “SIN 関数” (826 ページ)
- “TAN 関数” (849 ページ)

COSH 関数

双曲線余弦をラジアンで返します。

カテゴリ: 三角関数

返されるデータの種類: DOUBLE

構文

COSH(*expression*)

必須引数

expression
数値に評価される有効な SQL 式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

COSH 関数は、次の式によって得られる引数の双曲線余弦を返します。

$$e^{argument} + e^{-argument}/2$$

例

次のステートメントは、COSH 関数を示しています。

ステートメント	結果
select cosh(0);	1
select cosh(-5.0);	74.20995
select cosh(4.37);	39.52814
select cosh(0.5);	1.127626

関連項目

関数:

- “ARCOS 関数” (302 ページ)
- “COS 関数” (409 ページ)
- “SINH 関数” (827 ページ)
- “TANH 関数” (850 ページ)

COT 関数

正接をラジアンで返します。

カテゴリ: 三角関数

返されるデータの種類: DOUBLE

構文

COT(*expression*)

必須引数

expression
数値に評価される有効な SQL 式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

例

次のステートメントは、COT 関数を示しています。

ステートメント	結果
select cot(0);	.
select cot(0.5)	1.830488
select cot(5.79);	-1.86051
select cot(1);	0.642093

関連項目

関数:

- “ATAN 関数” (308 ページ)
- “COS 関数” (409 ページ)
- “TAN 関数” (849 ページ)

COUNT 関数

指定されたテーブルの SELECT ステートメントによって取得された行数を返します。

カテゴリ: 集計

別名: N

返されるデータの種類: BIGINT

構文

形式 1: **COUNT**([* | [DISTINCT]*expression*)]

必須引数

*
null 値または SAS 欠損値を含む行を含め、テーブルからすべての 行の数を返します。

expression
有効な SQL 式を指定します。

データの種類 すべてのデータ型が有効です。

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

オプション引数

DISTINCT
集計計算から重複行を削除します。

詳細

SELECT ステートメントで COUNT 関数を使用して、要求された行数をテーブルに返します。

次のリストは、さまざまなバージョンの COUNT 関数を使用して返される内容を示しています。

形式 1: COUNT(*expression*)
null 値を持たない *expression* の行数を返します。

形式 2: COUNT(*)
テーブル内の行数を返します。

形式 3: COUNT(DISTINCT *expression*)
一意の値を持つ *expression* の行数を返します。SAS 欠損値が結果に含まれます。null 値は結果に含まれません。

集計関数を使用して、FROM 句にリストされているテーブル全体、または GROUP BY 句で指定されている各グループのデータの統計的要約を生成できます。GROUP BY 句は、指定された 1 つまたは複数の列でデータをグループ化します。GROUP BY 句を使用する場合、SELECT 句または HAVING 句の集計関数は、各グループのデータを要約する方法を FedSQL に指示します。FedSQL は、グループごとに個別に集計関数を計算します。GROUP BY を省略すると、テーブルまたはビュー内のすべての行が単一のグループと見なされます。

例

テーブル: WORLDTEMPS (1277 ページ)

次のステートメントは、COUNT 関数を示しています。

ステートメント	結果
select count(AvgHigh) from worldtemps;	11
select count(*) from worldtemps;	12
select count(distinct AvgHigh) from worldtemps;	8

関連項目

SELECT ステートメント句:

- “SELECT 句” (980 ページ)
- “GROUP BY 句” (993 ページ)
- “HAVING 句” (995 ページ)

COUNTC 関数

文字のリストに表示される(または表示されない)文字列内の文字の個数を数えます。

カテゴリ:	文字
返されるデータの種類:	DOUBLE

構文

COUNTC(*string*, *charlist*[, *modifiers*])

必須引数

string
文字定数、変数または式を指定します。この中にある文字を数えます。

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

ヒント 文字のリテラル文字列を引用符で囲みます。

charlist
文字リストを初期化する文字定数、変数または式を指定します。COUNTC はこのリストの文字を数えます(ただし、V 修飾子を *modifiers* 引数で指定しない場合)。V 修飾子を指定すると、このリストにないすべての文字を数えます。他の修飾子を使うことでリストに文字をさらに追加できます。

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

ヒント 文字のリテラル文字列を引用符で囲みます。

修飾子を処理した後でリストに文字がない場合、COUNTC はゼロを返します。

オプション引数

modifiers
文字定数、変数または式を指定します。これらの空白以外の各文字で COUNTC 関数のアクションを変更します。空白は無視されます。A (アルファベット文字をリストに追加)、C (制御文字を追加)、D (数字を追加)の大文字と小文字を修飾子として使用できます。他の修飾子も使用することができます。詳細については、*SAS DS2 言語リファレンス*を参照してください。

文字定数、変数または式を指定します。これらの空白以外の各文字で COUNTC 関数のアクションを変更します。空白は無視されます。

修飾子として使用できる文字(大文字または小文字)は次のとおりです。

A
文字のリストにアルファベット文字を追加します。

B
左から右方向ではなく、右から左方向に *string* をスキャンします。

C
文字のリストに制御文字を追加します。

D
文字のリストに数字を追加します。

F
アンダースコアと 英字(つまり、VALIDVARNAME=V7 を使用して SAS 変数名を開始できる文字)を文字のリストに追加します。

G
文字のリストにグラフィカル文字を追加します。

H
文字のリストに水平タブを追加します。

I
大文字と小文字を区別しません。

L
小文字を文字リストに追加します。

N
文字のリストに数字、アンダースコアおよび英文字(VALIDVARNAME=V7 を使用した SAS 変数名内に表示可能な文字)を追加します。

O
この COUNTC インスタンスを最初に呼び出す 1 回のみ、*charlist* 引数および *modifier* 引数を処理します。以降の呼び出しにおける *charlist* または *modifier* の値の変更は、COUNTC で無視されることがあります。

P
文字のリストに句読点を追加します。

S
文字のリストに空白文字(空白、水平タブ、垂直タブ、キャリッジリターン、ラインフィード、フォームフィード)を追加します。

T
string および *chars* から末尾の空白を取り除きます。両方(またはすべて)の文字引数ではなく一方のみから末尾の空白を削除する場合は、COUNTC 関数に T 修飾子を使用するかわりに、TRIM 関数を使用します。

U
大文字を文字リストに追加します。

V
文字リストに現れない文字を数えます。この修飾子を指定しないと、COUNTC はこの文字リストに現れない文字を数えます。

W
印刷可能文字を文字リストに追加します。

X
文字のリストに 16 進文字を追加します。

データの種類　CHAR、VARCHAR

ヒント　*modifier* が定数の場合、引用符で囲みます。一組の引用符で複数の定数を指定します。

詳細

COUNTC 関数では、文字引数を null に指定できます。null 引数は長さがゼロの文字列として扱われます。文字リストに数える文字がない場合、COUNTC はゼロを返します。

注: CHAR データ型の文字列には、常に宣言された長さまで空白が埋め込まれることに注意してください。VARCHAR データ型の文字列では、宣言された長さではなく、実際の文字列の長さが返されます。

比較

COUNTC 関数は文字列中の個別の文字を数えるのに対し、COUNT 関数は文字列中の部分文字列の文字を数えます。

例

COUNTC 関数で文字列中の文字数を数えるときに、修飾子を使用する場合と使用しない場合の例を次に示します。

```
create table test(string char(24));
 insert into test (string) values ('Baboons Eat Bananas');

 select string,
   countc(string, 'a') as a,
   countc(string, 'b') as b,
   countc(string, 'b', 'i') as b_i,
      countc(string,'abc','i') as abc_i,
   countc(string, 'abc', 'iv') as abc_iv,
   countc(string, 'abc', 'ivt') as abc_ivt
   from test;
```

アウトプット 4.6　COUNTC 関数で修飾子を使用する場合としない場合の結果

STRING	A	B	B_I	ABC_I	ABC_IV	ABC_IVT
Baboons Eat Bananas	5	1	3	8	16	11

関連項目

関数:

- “COUNTW 関数” (418 ページ)

- “INDEXC 関数” (576 ページ)
- “VERIFY 関数” (878 ページ)

COUNTW 関数

文字列中の単語数を数えます。

カテゴリ: 文字

返されるデータの種類: DOUBLE

構文

COUNTW(*string*[, *chars*[, *modifiers*]])

必須引数

string
文字定数、変数または式を指定します。この中にある単語を数えます。

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

オプション引数

chars
文字のリストを初期化する任意の文字定数、変数または式を指定します。リスト中の文字は単語を区切るための区切り文字です(ただし *modifier* 引数で K 修飾子を使用しない場合)。K 修飾子を指定すると、このリストにないすべての文字が区切り文字となります。他の修飾子を使うことでリストに文字をさらに追加できます。

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

modifiers
文字定数、変数または式を指定します。これらの空白以外の各文字で COUNTW 関数のアクションを変更します。A (アルファベット文字をリストに追加)、C (制御文字を追加)、D (数字を追加)の大文字と小文字を修飾子として使用できます。他の修飾子も使用することができます。詳細については、*SAS DS2 言語リファレンス*を参照してください。

文字定数、変数または式を指定します。これらの空白以外の各文字で COUNTW 関数のアクションを変更します。

修飾子として使用できる文字(大文字または小文字)は次のとおりです。

A
文字のリストにアルファベット文字を追加します。

B
左から右方向ではなく、右から左方向に *string* をスキャンします。

C
文字のリストに制御文字を追加します。

D
文字のリストに数字を追加します。

F
アンダースコアと 英字(つまり、VALIDVARNAME=V7 を使用して SAS 変数名を開始できる文字)を文字のリストに追加します。

G
文字のリストにグラフィカル文字を追加します。

H
文字のリストに水平タブを追加します。

I
大文字と小文字を区別しません。

L
小文字を文字リストに追加します。

N
文字のリストに数字、アンダースコアおよび英文字(VALIDVARNAME=V7 を使用した SAS 変数名内に表示可能な文字)を追加します。

O
この COUNTC インスタンスを最初に呼び出す 1 回のみ、*charlist* 引数および *modifier* 引数を処理します。以降の呼び出しにおける *charlist* または *modifier* の値の変更は、COUNTC で無視されることがあります。

P
文字のリストに句読点を追加します。

Q
引用符で囲まれた部分文字列内の区切り文字を無視します。*string* の値に、対になっていない引用符が含まれている場合、左から右へのスキャンと、右から左へのスキャンとでは異なる単語が生成されます。

S
文字のリストに空白文字(空白、水平タブ、垂直タブ、キャリッジリターン、ラインフィード、フォームフィード)を追加します。

T
string および *chars* から末尾の空白を取り除きます。両方(またはすべて)の文字引数ではなく一方のみから末尾の空白を削除する場合は、COUNTC 関数に T 修飾子を使用するかわりに、TRIM 関数を使用します。

U
大文字を文字リストに追加します。

V
文字リストに現れない文字を数えます。この修飾子を指定しないと、COUNTC はこの文字リストに現れない文字を数えます。

W
印刷可能文字を文字リストに追加します。

X
文字のリストに 16 進文字を追加します。

データの種類　CHAR、VARCHAR

詳細

"単語"の定義

COUNTW 関数では、"単語"とは、次の特性のいずれかを持つ部分文字列を指します。

- 左側が区切り文字または文字列の先頭で境界設定されている
- 右側が区切り文字または文字列の末尾で境界設定されている
- 区切り文字を含まない。ただし、Q 修飾子を使用している場合で、引用符を含む部分文字列内に区切り文字が囲まれている場合を除きます。

注: "単語"の定義は SCAN 関数と COUNTW 関数の両方で共通しています。

区切り文字は、単語を区切るために指定できるいくつかの文字を示します。

ASCII 環境で COUNTW 関数を使用する

引数が 2 つしかない COUNTW 関数を使用する場合、デフォルトの区切り文字は次のとおりです。

- お使いのコンピューターが ASCII 文字を使用している場合、デフォルトの区切り文字は次のようになります。

  空白 ! $ % & ( ) * + , - ./ ; < ^ |

null 引数の使用

COUNTW 関数では、文字引数を null に指定できます。null 引数は長さがゼロの文字列として扱われます。数値引数は null にできません。

M 修飾子の使用

M 修飾子を使用しない場合、単語には少なくとも 1 つの文字が含まれている必要があります。M 修飾子を使用する場合、単語の長さをゼロとすることができます。この場合、単語数は文字列中の区切り文字数に 1 を加算した数です。Q 修飾子を使用するときは引用符に囲まれている文字列中の区切り文字は数えません。

例

次のプログラムでは、COUNTW 関数に M 修飾子および P 修飾子を使う方法を示します。

各文字列の **mp** の値の説明は次のとおりです。

- ピリオドは区切り文字であり、m 修飾子を使用すると、最後のピリオドは長さがゼロの後続の単語を参照しますが、それでも単語を参照します。つまり、ピリオドの前に 1 単語、ピリオドの後に 1 単語で、合計 2 つの単語があるということです。
- 区切り文字がないため、1 単語だけです。
- p 修飾子が区切り文字として句読点を追加するため、3 単語になります。
- p 修飾子が句読点を追加するため、/は区切り文字です。m 修飾子を使用すると、先頭の/が長さゼロの先頭の単語を参照して、合計 6 単語になります。
- 最初の\はエスケープ文字です 2 番目の\は区切り文字なので、6 単語があります。

```
proc fedsql;
drop table test;
create table test(string1 char(60) having informat $char60. format $char60.);
describe table test;

 insert into test values ('The quick brown fox jumps over the lazy dog.');
 insert into test values ('       Leading blanks');
 insert into test values ('2+2=4');
 insert into test values ('/unix/path/names/use/slashes');
 insert into test values ('\\Windows\\Path\\Names\\Use\\Backslashes');

select string1,
countw(string1) as basic,
countw(string1, ' ') as blank,
countw(string1, '.', 'mp') as mp from test;

quit;
```

注: Windows のパス名にはダブルバックスラッシュが使用されています。

アウトプット 4.7　M および P 修飾子で COUNTW 関数を使用した結果

STRING1	BASIC	BLANK	MP
The quick brown fox jumps over the lazy dog.	9	9	2
Leading blanks	2	2	1
2+2=4	2	1	3
/unix/path/names/use/slashes	5	1	6
\Windows\Path\Names\Use\Backslashes	1	1	6

関連項目

関数:

- “COUNTC 関数” (414 ページ)
- “SCAN 関数” (814 ページ)

CSC 関数

余割を返します。

カテゴリ: 三角関数

返されるデータの種類: DOUBLE

構文

CSC(*argument*)

必須引数

argument
ラジアンで表される数値定数、変数または式を指定します。

制限事項 *argument* は、0 または PI の倍数にはできません。

データの種類 DOUBLE

比較

CSC 関数と SIN 関数は次のような関係です。

csc(x)=1/sin(x)

例

次のステートメントは、CSC 関数を示しています。

ステートメント	結果
select csc(0.5);	2.08583

ステートメント	結果
select csc(1);	1.188395
csc(3.14159/3);	1.154701

注: csc(0)を使用すると、CSC 関数は欠損値を返します。これは正しい動作です。

関連項目

関数:

- “COS 関数” (409 ページ)
- “COT 関数” (411 ページ)
- “SEC 関数” (823 ページ)
- “SIN 関数” (826 ページ)
- “TAN 関数” (849 ページ)

CSS 関数

式のすべての値の修正済み平方和を返します。

カテゴリ: 集計
記述統計

返されるデータの種類: DOUBLE

構文

CSS([DISTINCT]*expression*)

必須引数

expression
有効な SQL 式を指定します。

データの種類 DOUBLE、FLOAT、REAL

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

オプション引数

DISTINCT
　集計計算から重複行を削除します。

詳細

修正済み平方和は、平均からの平方偏差(差分)和です。

null 値と SAS 欠損値は無視され、計算には含まれません。

集計関数を使用して、FROM 句にリストされているテーブル全体、または GROUP BY 句で指定されている各グループのデータの統計的要約を生成できます。GROUP BY 句は、指定された 1 つまたは複数の列でデータをグループ化します。GROUP BY 句を使用する場合、SELECT 句または HAVING 句の集計関数は、各グループのデータを要約する方法を FedSQL に指示します。FedSQL は、グループごとに個別に集計関数を計算します。GROUP BY を省略すると、テーブルまたはビュー内のすべての行が単一のグループと見なされます。

比較

USS 関数は、無修正平方和を返します。CSS 関数は、すべての値の修正済み平方和を返します。

例

テーブル: DENSITIES (1260 ページ)

次のステートメントは、CSS 関数を示しています。

ステートメント	結果
select css(density) from densities;	211483.1

関連項目

関数:

- “USS 関数” (873 ページ)

SELECT ステートメント句:

- “SELECT 句” (980 ページ)
- “GROUP BY 句” (993 ページ)
- “HAVING 句” (995 ページ)

CUMIPMT 関数

開始期間と終了期間の間でローンに対して支払われる累積利息を返します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

CUMIPMT(*rate*, *number-of-periods*, *principal-amount*[, *start-period*][, *end-period*][, *type*])

必須引数

rate
支払期間ごとの利率を指定します。

データの種類 DOUBLE

number-of-periods
支払期間の数を指定します。*number-of-periods* は正の整数にする必要があります。

データの種類 DOUBLE

principal-amount
ローンの元金を指定します。欠損値が指定されている場合、ゼロとみなされます。

データの種類 DOUBLE

オプション引数

start-period
計算の開始期間を指定します。

データの種類 DOUBLE

end-period
計算の終了期間を指定します。

データの種類 DOUBLE

type
期首と期末のどちらで支払がなされるのかを指定します。0 は期末の支払を表し、1 は期首の支払を表します。*type* が省略された場合、または欠損値が指定されている場合は、ゼロとみなされます。

データの種類 DOUBLE

例

- 名目年利 9%で$125,000 の 30 年ローンを組み、期末の月次払いとした場合に 2 年目に支払う累積利息は次のように計算します。

  ```
  select put(CUMIPMT(0.09/12, 360, 125000, 13, 24, 0), dollar10.2);
  ```

  計算によって$11,135.23 という値が返されます。
- 同じローンの最初の期間に支払われる利息は、次の方法で計算されます。

  ```
  select put(CUMIPMT(0.09/12, 360, 125000, 1, 1, 0), dollar10.2);
  ```

  計算によって$937.50 という値が返されます。

関連項目

関数:

- “CUMPRINC 関数” (426 ページ)

CUMPRINC 関数

開始期間と終了期間の間でローンに対して支払われる累積元本を返します。

カテゴリ:	財務
返されるデータの種類:	DOUBLE

構文

CUMPRINC(*rate*, *number-of-periods*, *principal-amount*[, *start-period*] [, *end-period*][, *type*])

必須引数

rate
支払期間ごとの利率を指定します。

データの種類 DOUBLE

number-of-periods
支払期間の数を指定します。

要件 *number-of-periods* は正の整数にする必要があります。

データの種類 DOUBLE

principal-amount
ローンの元金を指定します。

データの種類 DOUBLE

注 欠損値または null 値が指定されている場合、ゼロとみなされます。

オプション引数

start-period
計算の開始期間を指定します。

データの種類 DOUBLE

end-period
計算の終了期間を指定します。

データの種類 DOUBLE

type
期首と期末のどちらで支払がなされるのかを指定します。0 は期末の支払を表し、1 は期首の支払を表します。*type* が省略された場合、または欠損値が指定されている場合は、ゼロとみなされます。

データの種類 DOUBLE

例

- 名目年利 9%で$125,000 の 30 年ローンを組み、期末の月次払いとした場合に 2 年目に支払う累積元本は次のように計算します。

  ```
  select put(CUMPRINC(0.09/12, 360, 125000, 12, 24, 0), dollar10.2);
  ```

  計算によって$1008.23 という値が返されます。
- 同じローンで、期首支払の場合に 2 年目に支払う元本は次のように計算します。

  ```
  select put(CUMPRINC(0.09/12, 360, 125000, 12, 24, 1), dollar10.2);
  ```

  計算によって$1000.73 という値が返されます。

関連項目

関数:

- “CUMIPMT 関数” (425 ページ)

CURRENT_DATE 関数

タイムゾーンの現在の日付を返します。

カテゴリ:	日付と時間
返されるデータの種類:	DATE

構文

CURRENT_DATE

引数なし

CURRENT_DATE 関数には引数がありません。

詳細

CURRENT_DATE 関数は、現在の日付を *ddmmmyyyy* の形式で返します。

比較

CURRENT_DATE 関数は、タイムゾーンの現在の日付を返します。
CURRENT_TIMESTAMP_GMT()関数は、現在の GMT 日付を返します。

例

次のステートメントは、CURRENT_DATE 関数を示しています。

ステートメント	結果
select current_date;	05FEB2021

関連項目

関数:

- “CURRENT_TIME 関数” (429 ページ)
- “CURRENT_TIME_GMT 関数” (431 ページ)
- “CURRENT_TIMESTAMP 関数” (432 ページ)
- “CURRENT_TIMESTAMP_GMT 関数” (433 ページ)

CURRENT_LOCALE 関数

現在のロケールの 5 文字の名前を返します。

カテゴリ: 文字

返されるデータの種類: CHAR(5)

構文

CURRENT_LOCALE()

引数なし

CURRENT_LOCALE()関数には引数がありません。

例

次のステートメントは、CURRENT_LOCALE 関数を示しています。

ステートメント	結果
select current_locale();	en_US

CURRENT_TIME 関数

タイムゾーンの現在の時間を返します。

カテゴリ: 日付と時間

別名:	LOCALTIME
返されるデータの種類:	TIME

構文

CURRENT_TIME

引数なし

CURRENT_TIME 関数には引数がありません。

詳細

CURRENT_TIME 関数は、タイムゾーンの時間を *hh:mm:ss* の形式で返します。

比較

CURRENT_TIME 関数は、タイムゾーンの現在の時間を返します。
CURRENT_TIME_GMT 関数は、現在の時間を GMT で返します。

例

次のステートメントは、CURRENT_TIME 関数を示しています。

ステートメント	結果
select current_time;	11:00:49
select localtime;	11:00:49

関連項目

関数:

- “CURRENT_DATE 関数” (428 ページ)
- “CURRENT_TIME_GMT 関数” (431 ページ)
- “CURRENT_TIMESTAMP 関数” (432 ページ)
- “CURRENT_TIMESTAMP_GMT 関数” (433 ページ)

CURRENT_TIME_GMT 関数

現在の GMT 時間を返します。

カテゴリ: 日付と時間

返されるデータの種類: TIME

構文

CURRENT_TIME_GMT()

引数なし

CURRENT_TIME_GMT()関数には引数がありません。

詳細

CURRENT_TIME_GMT 関数は、現在の GMT 時間を *hh:mm:ss* の形式で返します。

比較

CURRENT_TIME_GMT 関数は、現在の GMT 時間を返します。CURRENT_TIME 関数は、タイムゾーンの現在の時間を返します。

例

次のステートメントは、CURRENT_TIME_GMT 関数を示しています。

ステートメント	結果
select current_time_gmt( );	15:00:49

関連項目

関数:

- “CURRENT_DATE 関数” (428 ページ)
- “CURRENT_TIME 関数” (429 ページ)
- “CURRENT_TIMESTAMP 関数” (432 ページ)
- “CURRENT_TIMESTAMP_GMT 関数” (433 ページ)

CURRENT_TIMESTAMP 関数

タイムゾーンの日時を返します。

カテゴリ:	日付と時間
別名:	LOCALTIMESTAMP
返されるデータの種類:	TIMESTAMP

構文

CURRENT_TIMESTAMP

引数なし

CURRENT_TIMESTAMP 関数には引数がありません。

詳細

CURRENT_TIMESTAMP 関数は、タイムゾーンの日時を *ddmmmyyyy:hh:mm:ss* の形式で返します。

例

次のステートメントは、CURRENT_TIMESTAMP 関数を示しています。

ステートメント	結果
select current_timestamp;	05FEB2014:11:06:05
select localtimestamp;	05FEB2014:11:06:05

関連項目

関数:

- “CURRENT_DATE 関数” (428 ページ)
- “CURRENT_TIME 関数” (429 ページ)
- “CURRENT_TIME_GMT 関数” (431 ページ)
- “CURRENT_TIMESTAMP_GMT 関数” (433 ページ)

CURRENT_TIMESTAMP_GMT 関数

現在の GMT 日時を返します。

カテゴリ:	日付と時間
返されるデータの種類:	TIMESTAMP

構文

CURRENT_TIMESTAMP_GMT()

引数なし

CURRENT_TIMESTAMP_GMT()関数には引数がありません。

詳細

CURRENT_TIMESTAMP_GMT 関数は、現在の GMT 日時を *ddmmmyyyy:hh:mm:ss* の形式で返します。

比較

CURRENT_TIMESTAMP_GMT 関数は、現在の GMT 日時を返します。
CURRENT_TIMESTAMP 関数は、タイムゾーンの現在の日時を返します。

例

次のステートメントは、CURRENT_TIMESTAMP_GMT 関数を示しています。

ステートメント	結果
select current_timestamp_gmt( );	24MAR2020:16:02:48

関連項目

関数:

- “CURRENT_DATE 関数” (428 ページ)
- “CURRENT_TIME 関数” (429 ページ)
- “CURRENT_TIMESTAMP 関数” (432 ページ)
- “CURRENT_TIME_GMT 関数” (431 ページ)

CV 関数

変動係数を返します。

カテゴリ: 記述統計

返されるデータの種類: DOUBLE

構文

CV(*expression-1*, *expression-2* [, ...*expression-n*])

必須引数

expression
数値に評価される有効な式を指定します。

要件 少なくとも 2 つの引数が必要です。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

例

次のステートメントは、CV 関数を示しています。

ステートメント	結果
select cv(5,9,3,6);	43.4782608695652
select cv(5,8,9,6,.);	26.0820265478651
select cv(8,9,6,.);	19.9242421519819

DAIRY 関数

AIRY 関数の導関数を返します。

カテゴリ: 数学

返されるデータの種類: DOUBLE

構文

DAIRY(*x*)

必須引数

x
数値の定数、変数または式を指定します。

詳細

DAIRY 関数は AIRY 関数の導関数の値を返します。 (リファレンスのリストを参照。)

例

次のプログラムは、DAIRY 関数を示しています。

ステートメント	結果
select dairy(2.0);	-0.05309
select dairy(-2.0);	0.618259

DATDIF 関数

指定された日数計算規則に従って 2 つの日付間の差を計算した後に、その日付間の日数を返します。

カテゴリ: 日付と時間

返されるデータの種類: DOUBLE

構文

DATDIF(*sdate*, *edate*, *basis*)

必須引数

sdate
開始日を識別する SAS 日付値を指定します。

データの種類 DATE

ヒント *sdate* が月末の場合、SAS はその日付を 1 か月が 30 日の月の最終日として処理します。

edate
終了日を識別する SAS 日付値を指定します。

データの種類 DATE

ヒント *edate* が月末の場合、SAS はその日付を 1 か月が 30 日の月の最終日として処理します。

basis
日数計算基準を表す文字列を指定します。*basis* の有効な値は、'30/360'、'ACT/ACT'、'ACT/360'、'ACT/365'です。たとえば、'ACT/365'は、各月に実際のカレンダー日数を使用し、すべての年の日数に実際の日数に関わらず 365 日を使用します。詳細については、*SAS DS2 言語リファレンス*を参照してください。

日数計算基準を表す文字列を指定します。*basis* の有効な値は次のとおりです。

'30/360'
その月または年の実際のカレンダー日数に関わらず、1 か月を 30 日、1 年を 360 日として指定します。

月の最終日に利息を支払う証券では、その利息を常にその月の最終日に支払うか、または常に各月の同じ日付に支払います。唯一の例外は、2 月 30 日など、その日がその月の有効な日ではない場合です。詳細については、"日数計算基準(30/360)の計算方法" (*SAS 関数と CALL ルーチン: リファレンス*)を参照してください。

別名 '360'

'ACT/ACT'
日付間の実際の日数を使用します。各月にその月の実際のカレンダー日数を使用し、各年にその年の実際のカレンダー日数を使用します。

別名 *'Actual'*

'ACT/360'
各月に実際のカレンダー日数を使用し、すべての年の日数に実際の日数に関わらず 360 日を使用します。

ヒント *ACT/360* は短期間の証券で使用されます。

'ACT/365'
各月に実際のカレンダー日数を使用し、すべての年の日数に実際の日数に関わらず 365 日を使用します。

ヒント *ACT/365* は短期間の証券で使用されます。

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

詳細

基本

DATDIF 関数は証券業界にとって特定の意味を持ち、その計算方法は実際の日数の数え方とは異なります。実際の日数を含む月と年を計算に使用できます。1 か月を 30 日または 1 年を 360 日として計算することもできます。標準の証券計算方法の詳細については、この関数の最後にあるリファレンスセクションを参照してください。

注: 月の日数を数える場合、DATDIF では*常に*開始日が含まれ、終了日は除外されます。

日数計算基準(30/360)の計算方法

2 つの日付間の日数を計算するには、次の式を使用します。

$$Number of days = [(Y2 - Y1) * 360] + [(M2 - M1) * 30] + (D2 - D1)$$

引数

Y2
後の日付の年を指定します。

Y1
前の日付の年を指定します。

M2
後の日付の月を指定します。

M1
前の日付の月を指定します。

D2
後の日付の日を指定します。

D1
前の日付の日を指定します。

すべての月は 30 日のみになるため、30 日以外の日数が含まれる月は調整する必要があります。この調整は、2 つの日付間の日数を計算する前に行います。

次の規則が適用されます。

- 証券が月末ルールを使用していて、D2 と D1 の両方が 2 月(うるう年以外は 28 日、うるう年は 29 日)の最終日の場合、D2 を 30 に変更します。
- 証券が月末ルールを使用していて、D1 が 2 月の最終日の場合、D1 を 30 に変更します。
- D2 の値が 31 で D1 の値が 30 または 31 の場合、D2 を 30 に変更します。
- D1 の値が 31 の場合、D1 を 30 に変更します。

例

次の例では、DATDIF は 2 つの日付間の実際の日数、および 1 か月を 30 日、1 年を 360 日とした場合の日数を返します。

この FedSQLSELECT ステートメントは、2 つの日付の間の実際の日数を返します。

```
select datdif(date'1978-10-16',date'1996-02-16', 'act/act');
```

このステートメントは、1 か月を 30 日、1 年を 360 日とした場合の日数を返します。

```
select datdif(date'1978-10-16',date'1996-02-16','30/360');
```

結果は次のとおりです。

```
6332
6240
```

参考文献

Securities Industry Association 1994. *Standard Securities Calculation Methods - Fixed Income Securities Formulas for Analytic Measures*. 2 巻 New York, NY: Securities Industry Association.

DATE 関数

SAS 日付値として現在の日付を返します。

カテゴリ:	日付と時間
別名:	TODAY
返されるデータの種類:	DOUBLE

構文

DATE( )

引数なし

DATE 関数には引数がありません。

比較

DATE 関数は引数を取りません。返される SAS 日付値は、1960 年 1 月 1 日から現在の日付までの日数です。

FedSQL が日付を処理する方法の詳細については、“FedSQL の日付と時間” (59 ページ)を参照してください。

例

次のステートメントは、DATE 関数を示しています。

ステートメント	結果
select date();	19611

関連項目

関数:

- “TODAY 関数” (858 ページ)

DATEJUL 関数

ユリウス暦の日付を SAS 日付値に変換します。

カテゴリ: 日付と時間

返されるデータの種類: DOUBLE

構文

DATEJUL(*julian-date*)

必須引数

julian-date
数値に評価され、ユリウス暦の日付を表す有効な式を指定します。ユリウス暦の日付は *yyddd* または *yyyyddd* 形式の日付です。このとき、*yy* と *yyyy* はそれぞれ年を表す 2 桁または 4 桁の整数、*ddd* は該当日がその年の何日目かを表す数です。*ddd* の値は 1 から 365(閏年の場合は 366)の間である必要があります。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

SAS 日付値は、1960 年 1 月 1 日から指定日付までの日数です。DATEJUL 関数は、1960 年 1 月 1 日から *julian-date* で指定されたユリウス暦の日付までの日数を返します。

FedSQL での日付の処理方法の詳細については、“FedSQL の日付と時間” (59 ページ)を参照してください。

例

次のステートメントは、DATEJUL 関数を示しています。

ステートメント	結果
select datejul(11365);	18992

関連項目

関数:

- “JULDATE 関数” (626 ページ)

DATEPART 関数

日付を年、月、日として返します。

カテゴリ:	日付と時間
返されるデータの種類:	DATE

構文

DATEPART(*ts*)

必須引数

ts
タイムスタンプを表す有効な式を指定します。

詳細

DATEPART 関数は、タイムスタンプ値を *ddmonyyyy* 形式の日付値に変換します。

例

次のステートメントは、DATEPART 関数を示しています。

ステートメント	結果
select datepart(timestamp '2013-09-24 14:46:58');	24SEP2013
select datepart(current_timestamp);	24MAR2020

関連項目

関数:

- “MAKEDATE 関数” (654 ページ)
- “MAKETIME 関数” (655 ページ)
- “MAKETIMESTAMP 関数” (656 ページ)
- “TIMEPART 関数” (851 ページ)

DATETIME 関数

SAS 日時値として現在の日時を返します。

カテゴリ:	日付と時間
返されるデータの種類:	DOUBLE

構文

DATETIME( )

比較

DATETIME 関数は引数を取りません。返される SAS 日時値は、1960 年 1 月 1 日から現在の日時までの秒数です。

例

次のステートメントは、DATETIME 関数を示しています。

```
data test(overwrite=yes);
  dcl double dt ;
  method run();
    dt=datetime();
    put dt;
  end;
enddata;
run;
```

SAS は次の出力をログに書き込みます。

```
1863786305.136
```

関連項目

概念:

- “FedSQL の日付と時間” (59 ページ)

関数:

- “DATE 関数” (439 ページ)

DAY 関数

日付または日時値から月の日を数値で返します。

カテゴリ: 日付と時間

返されるデータの種類: TINYINT

構文

DAY(*date* | *datetime*)

必須引数

date
日付値を表す有効な式を指定します。

データの種類 DATE

参照項目: “FedSQL 式” (46 ページ)

datetime
日時値を表す有効な式を指定します。

データの種類 TIMESTAMP

参照項目: “FedSQL 式” (46 ページ)

例

テーブル: CUSTONLINE (1259 ページ)

次のステートメントは、DAY 関数を示しています。

ステートメント	結果
select day(endtime) from custonline;	1 2 15 1 1 2 16 1 1 15
select day(current_date);	17

関連項目

- “FedSQL の日付と時間” (59 ページ)

関数:

- “HOUR 関数” (570 ページ)
- “MINUTE 関数” (670 ページ)
- “MONTH 関数” (676 ページ)
- “SECOND 関数” (824 ページ)
- “YEAR 関数” (887 ページ)

DEGREES 関数

角度の度数をラジアンで返します。

カテゴリ: 三角関数

返されるデータの種類: DOUBLE

構文

DEGREES(*expression*)

必須引数

expression
ラジアンで指定された角度に評価される有効な式を指定します。

データの種類 BIGINT、DOUBLE、FLOAT、INTEGER、REAL、SMALLINT、TINYINT。

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

expression が null 値の場合、DEGREES 関数は null 値を返します。結果が DOUBLE データ型の範囲に収まらない数値の場合、DEGREES 関数は失敗します。

例

次のステートメントは、DEGREES 関数を示しています。

ステートメント	結果
select degrees(2*pi());	360
select degrees(pi());	180
select degrees(pi()/2);	90
select degrees(pi()/4);	45

関連項目

関数:

- “RADIANS 関数” (798 ページ)

DEQUOTE 関数

単一引用符で始まる文字列から一致する単一引用符を削除し、閉じ引用符の右側にあるすべての文字を削除します。

カテゴリ: 文字

返されるデータの種類: CHAR、NCHAR、NVARCHAR、VARCHAR

構文

DEQUOTE(*expression*)

必須引数

expression
文字列に評価されるか強制変換できる有効な式を指定します。

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

DEQUOTE 関数で返される値は、*expression* の最初の文字または最初の 2 文字に依存します。

- *expression* の最初の文字が引用符でない場合、DEQUOTE は構文エラーを返します。
- *expression* の最初の文字が単一引用符の場合、DEQUOTE 関数は結果からその単一引用符を削除します。DEQUOTE は、次に *expression* を左から右にスキャンし、他の単一引用符または二重引用符を探します。

  ペアになっている単一引用符はすべて、1 つの単一引用符に置き換えられます。

  ペアになっている二重引用符はすべて保持されます。

  二重引用符が 2 文字目の場合、DEQUOTE は結果から二重引用符を削除します。DEQUOTE は、次に、*expression* を左から右にスキャンしていきます。一致する二重引用符が見つかった場合は、二重引用符の間のテキストが返されます。閉じ二重引用符の右側、つまり *expression* の末尾までのテキストは、結果から削除されます。

  expression でペアになっていない最初の単一引用符は閉じ単一引用符であり、削除されます。

  閉じかっこが単一引用符の後に続く場合、関数は引用符で囲まれていない文字列を返します。閉じ単一引用符の右側に文字が存在する場合、関数は構文エラーになり、エラーが SAS ログに出力されます。
- *expression* が二重引用符で囲まれている場合、DEQUOTE 関数は null または欠損値を返します。

注: *expression* が引用符に囲まれた定数の場合、それらの引用符は *expression* の値には含まれません。そのため、定数を示す引用符は DEQUOTE を使用して削除する必要はありません。

例

次のステートメントは、DEQUOTE 関数を示しています。

ステートメント	結果
select dequote(No quotation marks);	ERROR: Syntax error at or near "No"
select dequote(No 'leading' quotation marks);	ERROR: Syntax error at or near "No"
select dequote('Single matched quotation marks are removed');	Single matched quotation marks are removed
select dequote("Matched double quotation marks");	ERROR: Column "Matched double quotation marks" not found or cannot be accessed
select dequote('Paired ''single'' quotation marks are reduced to a single quotation mark');	Paired 'single' quotation marks are reduced to a single quotation mark
select dequote(' "Double quotation marks" within “single quotation marks“, with space before open quotation mark');	"Double quotation marks" within "single quotation marks", with space before open quotation mark
select dequote('"Double quotation marks within single quotation marks, without space before open quotation mark');	Double quotation marks within single quotation marks, without space before open quotation mark
select dequote('"Text after closing double quotation mark" is removed');	Text after closing double quotation mark
select dequote('No matching quotation mark);	ステートメントの実行は完了しません。 次の文字をサブミットして実行を完了します。 ');
select dequote('Identifiers after close quotation mark' results in a syntax error);	ERROR: Syntax error at or near "RESULTS"

DEVIANCE 関数

確率分布に基づくデビアンスを返します。

カテゴリ: 数学

返されるデータの種類: DOUBLE

構文

DEVIANCE(*distribution*, *variable*, *shape-parameter(s)*[, *ε*])

必須引数

distribution
分布を特定する文字定数、変数または式です。有効な分布は、'BERNOULLI', 'BINOMIAL'、'GAMMA'、'IGAUSS'、'NORMAL'、'POISSON'です。これらの各分布でサポートされているパラメーターの説明については、*SAS DS2 言語リファレンス*を参照してください。

分布を特定する文字定数、変数または式です。有効な分布については、次の表を参照してください。

分布	引数
Bernoulli (p. 449)	'BERNOULLI' | 'BERN'
二項 (p. 449)	'BINOMIAL' | 'BINO'
ガンマ (p. 450)	'GAMMA'
逆ガウス(Wald) (p. 450)	'IGAUSS' | 'WALD'
正規 (p. 451)	'NORMAL' | 'GAUSSIAN'
Poisson (p. 451)	'POISSON' | 'POIS'

variable
数値定数、変数または式です。

shape-parameter(s)
分布の形状の特性を示す 1 つ以上の分布固有の数値パラメーターです。

オプション引数

ε
正規分布以外のすべての分布に使用される任意の小さい数値です。

詳細

Bernoulli 分布

DEVIANCE('BERNOULLI', *variable*, *p*[, *ε*])

引数

variable
成功した場合は 1、失敗した場合は 0 の二値の数値の確率変数です。

p
$\varepsilon \le p \le 1-\varepsilon$ の成功率の数値です。

ε
限界 *p* に使用される任意の正の数値です。間隔 $0 \le p \le \varepsilon$ のすべての *p* 値は、*ε* で置換されます。間隔 $1-\varepsilon \le p \le 1$ のすべての *p* 値は、$1-\varepsilon$ で置換されます。

DEVIANCE 関数は、成功が 1 の確率変数値として定義された成功率 *p* の Bernoulli 分布からデビアンスを返します。式は次のとおりです。

$$\text{DEVIANCE}('BERN', variable, p, \varepsilon) = \begin{cases} -2\log(1-p) & x = 0 \\ -2\log(p) & x = 1 \\ . & otherwise \end{cases}$$

二項分布

DEVIANCE('BINO', *variable*, *μ*, *n*[, *ε*])

引数

variable
成功数が含まれる数値の確率変数です。

範囲　$0 \le variable \le 1$

μ
数値の平均パラメーターです。

範囲　$n\varepsilon \le \mu \le n(1-\varepsilon)$

n
Bernoulli 試行数の整数パラメーターです。

範囲　$n \ge 0$

ε
限界 *μ* に使用される任意の正の数値です。間隔 $0 \le \mu \le n\varepsilon$ のすべての *μ* 値は、$n\varepsilon$ で置換されます。間隔 $n(1-\varepsilon) \le \mu \le n$ のすべての *μ* 値は、$n(1-\varepsilon)$で置換されます。

DEVIANCE 関数は、成功率 *p*、独立した Bernoulli 試行数 *n* の二項分布からデビアンスを返します。正規分布の DEVIANCE 関数は次の式で表されます。*x* は確率変数です。

$$\text{DEVIANCE}('BINO', x, \mu, n) = \begin{cases} . & x < 0 \\ 2\left(x\log\left(\frac{x}{\mu}\right) + (n-x)\log\left(\frac{n-x}{n-\mu}\right)\right) & 0 \le x \le n \\ . & x > n \end{cases}$$

ガンマ分布

DEVIANCE('GAMMA', *variable*, *μ*[, *ε*])

引数

variable
数値の確率変数です。

範囲 *variable* ≥ *ε*

μ
数値の平均パラメーターです。

範囲 *μ* ≥*ε*

ε
限界 *variable* および *μ* に使用される任意の正の数値です。間隔 0 ≤ *variable* ≤ *ε* のすべての *variable* 値は、*ε* で置換されます。間隔 0 ≤ *μ* ≤ *ε* のすべての *μ* 値は、*ε* で置換されます。

DEVIANCE 関数は、平均パラメーター *μ* のガンマ分布からデビアンスを返します。ガンマ分布の DEVIANCE 関数は次の式で表されます。*x* は確率変数です。

$$\text{DEVIANCE}('GAMMA', x, \mu) = \begin{cases} . & x < 0 \\ 2\left(-\log\left(\frac{x}{\mu}\right) + \frac{x-\mu}{\mu}\right) & x \ge \varepsilon, \mu \ge \varepsilon \end{cases}$$

逆ガウス(Wald)分布

DEVIANCE('IGAUSS' | 'WALD', *variable*, *μ*[, *ε*])

引数

variable
数値の確率変数です。

範囲 *variable* ≥ *ε*

μ
数値の平均パラメーターです。

範囲 *μ* ≥*ε*

ε
限界 *variable* および *μ* に使用される任意の正の数値です。間隔 0 ≤ *variable* ≤ *ε* のすべての *variable* 値は、*ε* で置換されます。間隔 0 ≤ *μ* ≤ *ε* のすべての *μ* 値は、*ε* で置換されます。

DEVIANCE 関数は、平均パラメーター *μ* の逆ガウス分布からデビアンスを返します。逆ガウス分布の DEVIANCE 関数は次の式で表されます。*x* は確率変数です。

$$\text{DEVIANCE}('IGAUSS', x, \mu) = \begin{cases} . & x < 0 \\ \frac{(x-\mu)^2}{\mu^2 x} & x \geq \varepsilon, \mu \geq \varepsilon \end{cases}$$

正規分布

DEVIANCE('NORMAL' | 'GAUSSIAN', *variable*, *μ*)

引数

variable
数値の確率変数です。

μ
数値の平均パラメーターです。

DEVIANCE 関数は、平均パラメーター *μ* の正規分布からデビアンスを返します。正規分布の DEVIANCE 関数は次の式で表されます。*x* は確率変数です。

$$\text{DEVIANCE}('NORMAL', x, \mu) = (x-\mu)^2$$

Poisson 分布

DEVIANCE('POISSON', *variable*, *μ*[, *ε*])

引数

variable
数値の確率変数です。

範囲 *variable* ≥ 0

μ
数値の平均パラメーターです。

範囲 *μ* ≥*ε*

ε
限界 *μ* に使用される任意の正の数値です。間隔 0 ≤ *μ* ≤ *ε* のすべての *μ* 値は、*ε* で置換されます。

DEVIANCE 関数は、平均パラメーター *μ* の Poisson 分布からデビアンスを返します。Poisson 分布の DEVIANCE 関数は次の式で表されます。*x* は確率変数です。

$$\text{DEVIANCE}('POISSON', x, \mu) = \begin{cases} . & x < 0 \\ 2\left(x log\left(\frac{x}{\mu}\right) - (x-\mu)\right) & x \geq 0, \mu \geq \varepsilon \end{cases}$$

DHMS 関数

日、時、分、秒の値から SAS 日時値を返します。

カテゴリ: 日付と時間

返されるデータの種類: DOUBLE

構文

DHMS(*date*, *hour*, *minute*, *second*)

必須引数

date
SAS 日付値を表す有効な式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

hour
1 から 12 までの整数を表す数値式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

minute
1 から 59 までの整数を表す数値式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

second
1 から 59 までの整数を表す数値式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

DHMS 関数は、SAS 日時値を表す数値を返します。正または負の数値が返されます。

例

次のステートメントは、DHMS 関数を示しています。

ステートメント	結果
select dhms(mdy(08,10,2016),5,10,15);	1786425015
select put(dhms(date'2016-08-10', 5,10,15),datetime18.);	10AUG16:05:10:15

関連項目

概念:

- “FedSQL の日付と時間” (59 ページ)

関数:

- “HMS 関数” (565 ページ)

DIGAMMA 関数

ディガンマ関数の値を返します。

カテゴリ: 数学

返されるデータの種類: DOUBLE

構文

DIGAMMA(*expression*)

必須引数

expression
数値に評価される有効な式を指定します。

制限事項 ゼロおよび負の整数は無効です。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

DIGAMMA 関数は、次の式で求められる比率を返します。

$$\Psi(x) = \Gamma'(x)/\Gamma(x)$$

$\Gamma(.)$および$\Gamma'(.)$は、それぞれガンマ関数とその導関数を示します。*expression*>0 の場合、DIGAMMA 関数は lgamma 関数の導関数です。

例

次のステートメントは、DIGAMMA 関数を示しています。

ステートメント	結果
select digamma(1.0);	-0.577215665

DUR 関数

列挙キャッシュフローの修正デュレーションを返します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

DUR(*y*, *f*, *c(1), ..., c(k)*)

必須引数

y

期間当たりの有効満期利回りを指定します。分数で表します。

範囲 *y* > 0

データの種類 DOUBLE

f
期間当たりのキャッシュフローの頻度を指定します。

範囲 $f > 0$

データの種類 DOUBLE

c(1), ... ,c(k)
キャッシュフローのリストを指定します。

データの種類 DOUBLE

詳細

DUR 関数は、次の式から値を返します。

$$C = \sum_{k=1}^{K} \frac{k\left(\frac{c(k)}{(1+y)^{\frac{k}{f}}}\right)}{(P(1+y)f)}$$

前述の式には次の関係が適用されます。

$$P = \sum_{k=1}^{K} \frac{c(k)}{(1+y)^{\frac{k}{f}}}$$

例

次のステートメントは、DUR 関数を示しています。

ステートメント	結果
select dur(.05,1,.33,.44,.55, .49,.50,.22,.4,.8,.01,.36,.2,.4);	5.284025

関連項目

関数:

- “DURP 関数” (456 ページ)

DURP 関数

債権などの定期キャッシュフローストリームの修正デュレーションを返します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

DURP(*A*, *c*, *n*, *K*, k_0, *y*)

必須引数

A
額面価格を指定します。

範囲 $A > 0$

データの種類 DOUBLE

c
期間当たりの名目クーポンレートを分数で指定します。

範囲 $0 \le c < 1$

データの種類 DOUBLE

n
期間当たりのクーポン数を指定します。

範囲 $n > 0$ で整数

データの種類 DOUBLE

K
クーポンの残数を指定します。

範囲 $K > 0$ で整数

データの種類 DOUBLE

k_0
現在の日付から最初のクーポン日までの時間を指定します。期間数で表します。

範囲 $0 < k_0 \le 1/n$

データの種類 DOUBLE

y
期間当たりの名目有効満期利回りを指定します。分数で表します。

範囲 $y > 0$

データの種類 DOUBLE

詳細

DURP 関数は、次の式から値を返します。

$$D = \frac{1}{n} \frac{\sum_{k=1}^{K} t_k \frac{c(k)}{\left(1+\frac{y}{n}\right)^{t_k}}}{P\left(1+\frac{y}{n}\right)}$$

前述の式には次の関係が適用されます。

- $t_k = nk_0 + k - 1$
- $c(k) = \frac{c}{n}A \quad for\ k = 1, \ldots, K-1$
- $c(K) = \left(1+\frac{c}{n}\right)A$

前述の式には次の関係が適用されます。

$$P = \sum_{k=1}^{K} \frac{c(k)}{\left(1+\frac{y}{n}\right)^{t_k}}$$

例

次のステートメントは、DURP 関数を示しています。

ステートメント	結果
select durp(1000,1/100,4,14,.33/2,.10);	3.331707

関連項目

関数:

- “DUR 関数” (454 ページ)

E 関数

自然対数 e を返します。

カテゴリ: 数学

返されるデータの種類: DOUBLE

構文

E()

引数なし

E()関数には引数がありません。

詳細

E 関数は、自然対数 e、2.7182818 の値を返します。

比較

E 関数は引数を取らず、値 2.7182818 を返します。EXP 関数は引数を取り、e を引数で指定されたべき乗にします。

例

次のステートメントは、E 関数を示しています。

ステートメント	結果
select e();	2.718282

関連項目

関数:

- “EXP 関数” (462 ページ)
- “LOG 関数” (638 ページ)

EFFRATE 関数

有効な年利を返します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

EFFRATE(*compounding-interval*, *rate*)

必須引数

compounding-interval
SAS 間隔です。この値は、*rate* を複利計算する頻度を表します。

データの種類 CHAR

rate
数値です。*rate* は、各利息計算期間で複利計算する名目年利(パーセント)です。

データの種類 DOUBLE

詳細

EFFRATE 関数は、有効な年利を返します。この関数は、名目年利に対応する有効な年利を計算します。

EFFRATE 関数には次の詳細が適用されます。

- rate の値は–99 以上にする必要があります。
- 名目金利と利息計算期間を考えるにあたって、*compounding-interval* が 'CONTINUOUS'の場合、EFFRATE によって返される値は $e^{rate/100}–1$ です。

  compounding-interval が'CONTINUOUS'以外で、1 年に *m* 回の利息計算期間が含まれる場合、EFFRATE によって返される値は$(1+[rate/100\ m])^{m–1}$ です。
- *compounding-interval* で有効な値は次のとおりです。
  - 'CONTINUOUS'
  - 'DAY'

- 'SEMIMONTH'
- 'MONTH'
- 'QUARTER'
- 'SEMIYEAR'
- 'YEAR'

- 間隔が'DAY'の場合、*m*=365 です。

例

次のプログラムでは、有効な利率の計算方法を示します。

- このプログラムは、名目利率が 10%で利息が毎月複利計算される場合に対応する有効な利率を示します。

  ```
  select EFFRATE('MONTH', 10);
  ```

  有効な利率は 10.4713067441296 です。

- このプログラムは、名目利率が 10%で利息が四半期ごとに複利計算される場合に対応する有効な利率を示します。

  ```
  select EFFRATE('QUARTER', 10);
  ```

  有効な利率は 10.3812890624999 です。

ERF 関数

(正規)誤差関数の値を返します。

カテゴリ: 数学

返されるデータの種類: DOUBLE

構文

ERF(*expression*)

必須引数

expression
数値に評価される有効な式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

ERF 関数は、次の式で求められる積分を返します。

$$\mathrm{ERF}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \varepsilon^{-z^2} dz$$

ERF 関数を使用して、平均値が 0 で標準偏差が 1 の正規分布に従う確率変数の値が X よりも小さい確率(p)を求めることができます。たとえば、次のステートメントで求められる数量は PROBNORM(X)と同等です。

```
p=.5+.5*erf(x/sqrt(2));
```

例

次のステートメントは、ERF 関数を示しています。

ステートメント	結果
select erf(1.0);	0.8427007929
select erf(-1.0);	-0.842700793

関連項目

関数:

- “ERFC 関数” (461 ページ)
- “PROBNORM 関数” (786 ページ)

ERFC 関数

相補(正規)誤差関数の値を返します。

カテゴリ:	数学
返されるデータの種類:	DOUBLE

構文

ERFC(*expression*)

必須引数

expression
数値に評価される有効な式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

ERFC 関数は、ERF 関数(1 − ERF(*argument*))の補数を返します。

例

次のステートメントは、ERFC 関数を示しています。

ステートメント	結果
select erfc(1.0);	0.1572992071
select erfc(-1.0);	1.8427007929

関連項目

関数:

- “ERF 関数” (460 ページ)

EXP 関数

指定されたべき乗の e 定数の値を返します。

カテゴリ: 数学

返されるデータの種類: DOUBLE

構文

EXP(*expression*)

必須引数

expression
数値に評価される有効な SQL 式を指定します。

データの種類 BIGINT、DOUBLE、FLOAT、INTEGER、REAL、SMALLINT、TINTYINT

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

EXP 関数は、定数 *e* (およそ 2.71828 で与えられる)を引数で指定されたべき乗にします。この結果は、コンピューターの倍精度小数値の最大値によって制限されます。

比較

EXP 関数は引数を取り、*e* を引数で指定されたべき乗にします。E 関数は引数を取らず、値 2.7182818 を返します。

例

次のステートメントは、EXP 関数を示しています。

ステートメント	結果
select exp(1.0);	2.718282
select exp(0);	1

関連項目

関数:

- “E 関数” (458 ページ)
- “LOG 関数” (638 ページ)

FACT 関数

階乗を計算します。

カテゴリ: 数学

返されるデータの種類: DOUBLE

構文

FACT(*expression*)

必須引数

expression
数値に評価される有効な式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

FACT 関数の数学的表現は、次の式によって得られます。

$$FACT(n) = n!$$

この式では、$n \geq 0$ です。

式による計算ができない場合は欠損値が返されます。やや大きな値の場合、FACT 関数を計算できないことがあります。

例

次のステートメントは、FACT 関数を示しています。

ステートメント	結果
select fact(5);	120
select fact(-27);	.
select fact(7.3);	.

関連項目

関数:

- “COMB 関数” (389 ページ)

FINANCE 関数

減価償却、満期、未払い利息、正味現在価値、定期的預金、内部利益率などの財務計算を行います。

カテゴリ: 財務

構文

FINANCE(*string-identifier*, *parameter–1*, *parameter–2*, ...)

必須引数

string-identifier
文字定数、変数または式を指定します。次の表に、*string-identifier* の有効な値を示します。

string-identifier	説明
'ACCRINT'	定期的に利息を支払う証券の未払い利息を計算します。
'ACCRINTM'	満期に利息を支払う証券の未払い利息を計算します。
' AMORDEGRC'	減価償却係数を使用して各会計期間の減価償却を計算します。

string-identifier	説明
'AMORLINC'	各会計期間の減価償却を計算します。
'COUPDAYBS'	クーポン期間の開始日から決済日までの日数を計算します。
'COUPDAYS'	決済日を含むクーポン期間の日数を計算します。
'COUPDAYSNC'	決済日から次のクーポン日までの日数を計算します。
'COUPNCD'	決済日後の次のクーポン日を計算します。
'COUPNUM'	決済日から満期日までの支払いクーポン数を計算します。
'COUPPCD'	決済日前の前のクーポン日を計算します。
'CUMIPMT'	2 つの期間の間に支払われる累積利息を計算します。
'CUMPRINC'	2 つの期間の間にローンで支払われる累積元本を計算します。
'DB'	定率法を使用して、指定期間の資産の減価償却を計算します。
'DDB'	倍額定率法またはその他の方法を指定して、指定期間の資産の減価償却を計算します。
'DISC'	証券の割引率を計算します。
'DOLLARDE'	分数で表すドル価格を小数で表すドル価格に変換します。
'DOLLARFR'	小数で表すドル価格を分数で表すドル価格に変換します。
'DURATION'	定期的に利息を支払う証券の年度期間を計算します。
'EFFECT'	有効な年利を計算します。
'FV'	投資の将来価値を計算します。
'FVSCHEDULE'	一連の複利率を適用した後の当初元本の将来価値を計算します。
'INTRATE'	完全投資済み証券の利率を計算します。
'IPMT'	一定期間の投資の利払いを計算します。

string-identifier	説明
'IRR'	一連のキャッシュフローの内部利益率を計算します。
'ISPMT'	投資の指定期間中に支払われる利息を計算します。
'MDURATION'	推定額面$100 の証券の修正マコーレーデュレーションを計算します。
'MIRR'	プラスおよびマイナスのキャッシュフローが異なる利率で資金調達される場合の内部利益率を計算します。
'NOMINAL'	年間の名目金利を計算します。
'NPER'	投資の期間数を計算します。
'NPV'	一連の定期的なキャッシュフローと割引率に基づいて投資の正味現在価値を計算します。
'ODDFPRICE'	端数の開始期間で証券の額面$100 当たりの価格を計算します。
'ODDFYIELD'	端数の開始期間で証券の利回りを計算します。
'ODDLPRICE'	端数の最終期間で証券の額面$100 当たりの価格を計算します。
'ODDLYIELD'	端数の最終期間で証券の利回りを計算します。
'PMT'	年金の定期的支払いを計算します。
'PPMT'	一定期間の投資の元本に対する支払いを計算します。
'PRICE'	定期的に利息を支払う証券の額面$100 当たりの価格を計算します。
'PRICEDISC'	割引証券の額面$100 当たりの価格を計算します。
'PRICEMAT'	満期に利息を支払う証券の額面$100 当たりの価格を計算します。
'PV'	投資の現在価値を計算します。
'RATE'	年金期間当たりの利率を計算します。
'RECEIVED'	完全投資済み証券の満期受け取り額を計算します。
'SLN'	ある期間の資産の定額法による減価償却を計算します。

string-identifier	説明
'SYD'	指定期間の資産の年次級数和法による減価償却を計算します。
'TBILLEQ'	米国財務省短期証券の債券換算利回りを計算します。
'TBILLPRICE'	米国財務省短期証券の額面$100 当たりの価格を計算します。
'TBILLYIELD'	米国財務省短期証券の利回りを計算します。
'VDB'	定率法を使用して、指定期間または一部期間の資産の減価償却を計算します。
'XIRR'	必ずしも定期的ではないキャッシュフロースケジュールの内部利益率を計算します。
'XNPV'	必ずしも定期的ではないキャッシュフロースケジュールの正味現在価値を計算します。
'YIELD'	定期的に利息を支払う証券の利回りを計算します。
'YIELDDISC'	割引証券(米国財務省短期証券など)の年間利回りを計算します。
'YIELDMAT'	満期に利息を支払う証券の年間利回りを計算します。

parameter
各 *string-identifier* に関連付けられたパラメーターを指定します。

次のパラメーターを使用できます。

basis
使用する日数計算基準の種類を示す文字または数値の値を示すパラメーターです(省略可能)。

数値	文字列値	日数計算方法
0	"30/360"	US (NASD) 30/360
1	"ACTUAL"	実日数/実日数
2	"ACT/360"	実日数/360
3	"ACT/365"	実日数/365
4	"EU30/360"	ヨーロッパ 30/360

interest-rates
パーセントではなく数値で示す利率を指定します。

dates
財務関数のすべての日付が SAS 日付であることを指定します。

sign-of-cash-values
すべての引数に対し、貯蓄預金口座への入金やその他の出金など、支払う金額を負の数値で指定します。また、配当小切手やその他の入金など、受け取る金額を正の数値で指定します。

FINANCE ACCRINT 関数

定期的に利息を支払う証券の未払い利息を計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

FINANCE('ACCRINT', *issue*, *first-interest*, *settlement*, *rate*, *par-value*, *frequency*, [*basis*]);

必須引数

issue
証券の発行日を指定します。

要件 *issue* は SAS 日付です。

データの種類 DOUBLE

first-interest
証券の最初の利息日を指定します。

要件 *first-interest* は SAS 日付です。

データの種類 DOUBLE

settlement
決済日を指定します。

要件 *settlement* は SAS 日付です。

データの種類 DOUBLE

rate
利率を指定します。

要件　*rate* は、パーセンテージではなく数値として提供されます。

データの種類　DOUBLE

par-value
証券の額面価格を指定します。*par-value* を省略すると、**$1000** という値が使用されます。

データの種類　DOUBLE

frequency
年間のクーポン支払い回数を指定します。年 1 回支払いの場合は *frequency*=1、年 2 回支払いの場合は *frequency*=2、年 4 回支払いの場合は *frequency*=4 となります。

データの種類　DOUBLE

オプション引数

basis
使用する日数計算基準の種類を示す文字または数値としてオプションのパラメーターを指定します。

数値	文字列値	日数計算方法
0	"30/360"	US (NASD) 30/360
1	"ACTUAL"	実日数/実日数
2	"ACT/360"	実日数/360
3	"ACT/365"	実日数/365
4	"EU30/360"	ヨーロッパ 30/360

データの種類　DOUBLE

例: 未払い利息を計算する: ACCRINT

次の例では、定期的に利息を支払う証券の未払い利息を計算します。

ステートメント	結果
select finance('accrint', mdy(2, 28, 2016), mdy(8, 31, 2016), MDY(5, 1, 2016), 0.1, 1000, 2, 1);	17.12255

FINANCE ACCRINTM 関数

満期に利息を支払う証券の未払い利息を計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

FINANCE('ACCRINTM', *issue*, *settlement*, *rate*, *par-value*, [*basis*]);

必須引数

issue
証券の発行日を指定します。

要件 *issue* は SAS 日付です。

データの種類 DOUBLE

settlement
決済日を指定します。

要件 *settlement* は SAS 日付です。

データの種類 DOUBLE

rate
利率を指定します。

要件 *rate* は、パーセンテージではなく数値として提供されます。

データの種類 DOUBLE

par-value
証券の額面価格を指定します。*par-value* を省略すると、**$1000** という値が使用されます。

データの種類 DOUBLE

オプション引数

basis
使用する日数計算基準の種類を示す文字または数値としてオプションのパラメーターを指定します。

数値	文字列値	日数計算方法
0	"30/360"	US (NASD) 30/360
1	"ACTUAL"	実日数/実日数
2	"ACT/360"	実日数/360
3	"ACT/365"	実日数/365
4	"EU30/360"	ヨーロッパ 30/360

データの種類　DOUBLE

例: 未払い利息を計算する: ACCRINTM

次の例では、満期に利息を支払う証券の未払い利息を計算します。

ステートメント	結果
select finance('accrintm', mdy(2, 28, 2015), mdy(8, 31, 2015), 0.1, 1000, 0);	50.55556

FINANCE AMORDEGRC 関数

減価償却係数を使用して各会計期間の減価償却を計算します。

カテゴリ:　財務

返されるデータの種類:　DOUBLE

構文

FINANCE('AMORDEGRC', *cost*, *date-purchased*, *first-period*, *salvage*, *period*, *rate*, [*basis*]);

必須引数

cost
　資産の初期コストを指定します。

データの種類 DOUBLE

date-purchased
資産の購入日を指定します。

要件 *date-purchased* は SAS 日付です。

データの種類 DOUBLE

first-period
第 1 期間の最終日を指定します。

要件 *first-period* は SAS 日付です。

データの種類 DOUBLE

salvage
減価償却を終了した時点の価値(資産の救済価額とも呼ばれる)を指定します。

データの種類 DOUBLE

period
減価償却期間を指定します。

データの種類 DOUBLE

rate
減価償却率を指定します。

要件 *rate* は、パーセンテージではなく数値として提供されます。

データの種類 DOUBLE

オプション引数

basis
使用する日数計算基準の種類を示す文字または数値としてオプションのパラメーターを指定します。

数値	文字列値	日数計算方法
0	"30/360"	US (NASD) 30/360
1	"ACTUAL"	実日数/実日数
2	"ACT/360"	実日数/360
3	"ACT/365"	実日数/365
4	"EU30/360"	ヨーロッパ 30/360

> ヒント　FINANCE 関数の第 1 引数が AMORDEGR で、*basis* の値が 2 の場合、欠損値が返されます。

データの種類　DOUBLE

例: 減価償却を計算する: AMORDEGRC

次の例では、減価償却係数を使用して各会計期間の減価償却を計算します。

ステートメント	結果
select finance('amordegrc', 2400, mdy(8, 19, 2016), mdy(12, 31, 2016), 300, 1, 0.15, 1);	776

FINANCE AMORLINC 関数

各会計期間の減価償却を計算します。

カテゴリ:　財務

返されるデータの種類:　DOUBLE

構文

FINANCE('AMORLINC', *cost*, *date-purchased*, *first-period*, *salvage*, *period*, *rate*, [*basis*]);

必須引数

cost
資産の初期コストを指定します。

データの種類　DOUBLE

date-purchased
資産の購入日を指定します。

要件　*date-purchased* は SAS 日付です。

データの種類　DOUBLE

first-period
第 1 期間の最終日を指定します。

要件 *first-period* は SAS 日付です。

データの種類 DOUBLE

salvage
減価償却を終了した時点の価値(資産の救済価額とも呼ばれる)を指定します。

データの種類 DOUBLE

period
減価償却期間を指定します。

データの種類 DOUBLE

rate
減価償却率を指定します。

要件 *rate* は、パーセンテージではなく数値として提供されます。

データの種類 DOUBLE

オプション引数

basis
使用する日数計算基準の種類を示す文字または数値としてオプションのパラメーターを指定します。

数値	文字列値	日数計算方法
0	"30/360"	US (NASD) 30/360
1	"ACTUAL"	実日数/実日数
2	"ACT/360"	実日数/360
3	"ACT/365"	実日数/365
4	"EU30/360"	ヨーロッパ 30/360

ヒント FINANCE 関数の第 1 引数が AMORLINC で、*basis* の値が 2 の場合、欠損値が返されます。

データの種類 DOUBLE

例: 計算の説明: AMORLINC

次の例では、各会計期間の減価償却を計算します。

ステートメント	結果
select finance('AMORLINC', 2400, mdy(9, 30, 2016), mdy(12, 31, 2016), 245, 0, 0.115, 0);	69

FINANCE COUPDAYBS 関数

クーポン期間の開始日から決済日までの日数を計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

FINANCE('COUPDAYBS', *settlement*, *maturity*, *frequency*, [*basis*]);

必須引数

settlement
証券の決済日を指定します。証券決済日は、証券が買い手に対して売買される発行日の翌日になります。

要件 *settlement* は SAS 日付です。

データの種類 DOUBLE

maturity
証券の満期日を指定します。満期日は証券の有効期限が切れる日です。

要件 *maturity* は SAS 日付です。

データの種類 DOUBLE

frequency
年間のクーポン支払い回数を指定します。年 1 回支払いの場合は *frequency*=1、年 2 回支払いの場合は *frequency*=2、年 4 回支払いの場合は *frequency*=4 となります。

データの種類 DOUBLE

オプション引数

basis
使用する日数計算基準の種類を示す文字または数値としてオプションのパラメーターを指定します。

数値	文字列値	日数計算方法
0	"30/360"	US (NASD) 30/360
1	"ACTUAL"	実日数/実日数
2	"ACT/360"	実日数/360
3	"ACT/365"	実日数/365
4	"EU30/360"	ヨーロッパ 30/360

データの種類　DOUBLE

例: 計算の説明: COUPDAYBS

次の例では、クーポン期間の開始日から決済日までの日数を計算します。

ステートメント	結果
select finance('COUPDAYBS', mdy(12,30,2013), mdy(11,29,2016), 4, 2);	31

FINANCE COUPDAYS 関数

決済日を含むクーポン期間の日数を計算します。

カテゴリ:　財務

返されるデータの種類:　DOUBLE

構文

FINANCE('COUPDAYS', *settlement*, *maturity*, *frequency*, [*basis*]);

必須引数

settlement
決済日を指定します。

要件 *settlement* は SAS 日付です。

データの種類 DOUBLE

maturity
満期日を指定します。

要件 *maturity* は SAS 日付です。

データの種類 DOUBLE

frequency
年間のクーポン支払い回数を指定します。年 1 回支払いの場合は *frequency*=1、年 2 回支払いの場合は *frequency*=2、年 4 回支払いの場合は *frequency*=4 となります。

データの種類 DOUBLE

オプション引数

basis
使用する日数計算基準の種類を示す文字または数値としてオプションのパラメーターを指定します。

数値	文字列値	日数計算方法
0	"30/360"	US (NASD) 30/360
1	"ACTUAL"	実日数/実日数
2	"ACT/360"	実日数/360
3	"ACT/365"	実日数/365
4	"EU30/360"	ヨーロッパ 30/360

データの種類 DOUBLE

例: 計算の説明: COUPDAYS

次の例では、決済日を含むクーポン期間の日数を計算します。

ステートメント	結果
select finance('COUPDAYS', mdy(1,25,2015), mdy(11,15,2016), 2, 1);	181

FINANCE COUPDAYSNC 関数

決済日から次のクーポン日までの日数を計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

FINANCE('COUPDAYSNC', *settlement*, *maturity*, *frequency*, [*basis*]);

必須引数

settlement
決済日を指定します。

要件 *settlement* は SAS 日付です。

データの種類 DOUBLE

maturity
満期日を指定します。

要件 *maturity* は SAS 日付です。

データの種類 DOUBLE

frequency
年間のクーポン支払い回数を指定します。年 1 回支払いの場合は *frequency*=1、年 2 回支払いの場合は *frequency*=2、年 4 回支払いの場合は *frequency*=4 となります。

データの種類 DOUBLE

オプション引数

basis
使用する日数計算基準の種類を示す文字または数値としてオプションのパラメーターを指定します。

数値	文字列値	日数計算方法
0	"30/360"	US (NASD) 30/360
1	"ACTUAL"	実日数/実日数
2	"ACT/360"	実日数/360
3	"ACT/365"	実日数/365
4	"EU30/360"	ヨーロッパ 30/360

データの種類　DOUBLE

例: 計算の説明: COUPDAYSNC

次の例では、決済日から次のクーポン日までの日数を計算します。

ステートメント	結果
select FINANCE('COUPDAYSNC', mdy(1,25,2007), mdy(11,15,2008), 2, 1);	110

FINANCE COUPNCD 関数

決済日後の次のクーポン日を計算します。

カテゴリ:　財務

返されるデータの種類:　DOUBLE

構文

FINANCE('COUPNCD', *settlement*, *maturity*, *frequency*, [*basis*]);

必須引数

settlement
　決済日を指定します。

要件 *settlement* は SAS 日付です。

データの種類 DOUBLE

maturity
満期日を指定します。

要件 *maturity* は SAS 日付です。

データの種類 DOUBLE

frequency
年間のクーポン支払い回数を指定します。年 1 回支払いの場合は *frequency*=1、年 2 回支払いの場合は *frequency*=2、年 4 回支払いの場合は *frequency*=4 となります。

データの種類 DOUBLE

オプション引数

basis
使用する日数計算基準の種類を示す文字または数値としてオプションのパラメーターを指定します。

数値	文字列値	日数計算方法
0	"30/360"	US (NASD) 30/360
1	"ACTUAL"	実日数/実日数
2	"ACT/360"	実日数/360
3	"ACT/365"	実日数/365
4	"EU30/360"	ヨーロッパ 30/360

データの種類 DOUBLE

例: 計算の説明: COUPNCD

次の例では、決済日後の次のクーポン日を計算します。

ステートメント	結果
select FINANCE('COUPNCD', mdy(1, 25, 2007), mdy(11, 15, 2008), 2, 1);	17301

ステートメント	結果
select put(FINANCE('COUPNCD', mdy(1, 25, 2007), mdy(11, 15, 2008), 2, 1), date7.);	15MAY07

FINANCE COUPNUM 関数

決済日から満期日までの支払いクーポン数を計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

FINANCE('COUPNUM', *settlement*, *maturity*, *frequency*, [*basis*]);

必須引数

settlement
決済日を指定します。

要件 *settlement* は SAS 日付です。

データの種類 DOUBLE

maturity
満期日を指定します。

要件 *maturity* は SAS 日付です。

データの種類 DOUBLE

frequency
年間のクーポン支払い回数を指定します。年 1 回支払いの場合は *frequency*=1、年 2 回支払いの場合は *frequency*=2、年 4 回支払いの場合は *frequency*=4 となります。

データの種類 DOUBLE

オプション引数

basis
使用する日数計算基準の種類を示す文字または数値としてオプションのパラメーターを指定します。

数値	文字列値	日数計算方法
0	"30/360"	US (NASD) 30/360
1	"ACTUAL"	実日数/実日数
2	"ACT/360"	実日数/360
3	"ACT/365"	実日数/365
4	"EU30/360"	ヨーロッパ 30/360

データの種類　DOUBLE

例: 計算の説明: COUPNUM

次の例では、決済日から満期日までの支払いクーポン数を計算します。

ステートメント	結果
select FINANCE('COUPNUM', mdy(1, 25, 2015), mdy(11,15,2016), 2, 1);	4

FINANCE COUPPCD 関数

決済日前の前のクーポン日を計算します。

カテゴリ:　財務

返されるデータの種類:　DOUBLE

構文

FINANCE('COUPPCD', *settlement*, *maturity*, *frequency*, [*basis*]);

必須引数

settlement
　決済日を指定します。

要件 *settlement* は SAS 日付です。

データの種類 DOUBLE

maturity
満期日を指定します。

要件 *maturity* は SAS 日付です。

データの種類 DOUBLE

frequency
年間のクーポン支払い回数を指定します。年 1 回支払いの場合は *frequency*=1、年 2 回支払いの場合は *frequency*=2、年 4 回支払いの場合は *frequency*=4 となります。

データの種類 DOUBLE

オプション引数

basis
使用する日数計算基準の種類を示す文字または数値としてオプションのパラメーターを指定します。

数値	文字列値	日数計算方法
0	"30/360"	US (NASD) 30/360
1	"ACTUAL"	実日数/実日数
2	"ACT/360"	実日数/360
3	"ACT/365"	実日数/365
4	"EU30/360"	ヨーロッパ 30/360

データの種類 DOUBLE

例: 計算の説明: COUPPCD

次の例では、決済日前の前のクーポン日を計算します。

ステートメント	結果
select FINANCE('COUPPCD', mdy(1, 25, 2007), mdy(11, 15, 2008), 2, 1);	1720

ステートメント	結果
select put(FINANCE('COUPPCD', mdy(1, 25, 2007), mdy(11, 15, 2008), 2, 1), date7.);	15NOV06

FINANCE CUMIPMT 関数

2 つの期間の間に支払われる累積利息を計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

FINANCE('CUMIPMT', *rate*, *nper*, *pv*, *start-period*, *end-period*, [*type*]);

必須引数

rate
利率を指定します。

要件 *rate* は、パーセンテージではなく数値として提供されます。

データの種類 DOUBLE

nper
支払い期間の総数を指定します。

データの種類 DOUBLE

pv
現在価値、または一連の将来の支払いに相当する現時点での一括払い金額を指定します。

データの種類 DOUBLE

start-period
計算の開始期間を指定します。支払い期間は 1 から番号が付けられます。

データの種類 DOUBLE

end-period
計算の最終期間を指定します。

データの種類 DOUBLE

オプション引数

type
0 か 1 を指定し、支払い期日がいつであるかを示します。*type* を省略すると、0 とみなされます。

支払い期日が期末の場合は、*type* 引数を省略するか、0 に設定します。支払い期日が期首の場合は、*type* を 1 に設定します。

データの種類 DOUBLE

例: 計算の説明: CUMIPMT

次の例では、2 つの期間の間に支払われる累積利息を計算します。

ステートメント	結果
select FINANCE('CUMIPMT', 0.09, 30, 125000, 13, 24, 0);	-94054.8

FINANCE CUMPRINC 関数

2 つの期間の間にローンで支払われる累積元本を計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

FINANCE('CUMPRINC', *rate*, *nper*, *pv*, *start-period*, *end-period*, [*type*]);

必須引数

rate
利率を指定します。

要件 *rate* は、パーセンテージではなく数値として提供されます。

データの種類 DOUBLE

nper
支払い期間の総数を指定します。

データの種類 DOUBLE

pv
現在価値、または一連の将来の支払いに相当する現時点での一括払い金額を指定します。

データの種類 DOUBLE

start-period
計算の開始期間を指定します。支払い期間は 1 から番号が付けられます。

データの種類 DOUBLE

end-period
計算の最終期間を指定します。

データの種類 DOUBLE

オプション引数

type
0 か 1 を指定し、支払い期日がいつであるかを示します。*type* を省略すると、0 とみなされます。

支払い期日が期末の場合は、*type* 引数を省略するか、0 に設定します。支払い期日が期首の場合は、*type* を 1 に設定します。

データの種類 DOUBLE

例: 計算の説明: CUMPRINC

次の例では、2 つの期間の間に支払われる累積利息を計算します。

ステートメント	結果
select FINANCE('CUMPRINC', 0.09, 30, 125000, 13, 24, 0);	-51949.7

FINANCE DB 関数

定率法を使用して、指定期間の資産の減価償却を計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

FINANCE('DB', *cost*, *salvage*, *life*, *period*, [*month*]);

必須引数

cost
資産の初期コストを指定します。

データの種類 DOUBLE

salvage
減価償却を終了した時点の価値(資産の救済価額とも呼ばれる)を指定します。

データの種類 DOUBLE

life
資産が減価償却される期間数(資産の耐用年数とも呼ばれる)を指定します。

データの種類 DOUBLE

period
減価償却を計算する期間を指定します。*Period* には、*life* と同じ時間単位を使用する必要があります。

データの種類 DOUBLE

オプション引数

month
月数を指定します(month は数値引数で、省略可能です)。month を省略すると、デフォルト値 12 が使用されます。

データの種類 DOUBLE

例: 計算の説明: DB

次の例では、定率法を使用して、指定した期間の資産の減価償却を計算します。

ステートメント	結果
select FINANCE('DB', 1000000, 100000, 6, 2, 7);	259639.4

FINANCE DDB 関数

倍額定率法またはその他の方法を指定して、指定期間の資産の減価償却を計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

FINANCE('DDB', *cost*, *salvage*, *life*, *period*, [*factor*]);

必須引数

cost
資産の初期コストを指定します。

データの種類 DOUBLE

salvage
減価償却を終了した時点の価値(資産の救済価額とも呼ばれる)を指定します。

データの種類 DOUBLE

life
資産が減価償却される期間数(資産の耐用年数とも呼ばれる)を指定します。

データの種類 DOUBLE

period
減価償却を計算する期間を指定します。*Period* には、*life* と同じ時間単位を使用する必要があります。

データの種類 DOUBLE

オプション引数

factor
減価償却率を指定します。*factor* を省略すると、2 とみなされます(倍額定率法)。

データの種類 DOUBLE

例: 計算の説明: DDB

次の例では、倍額定率法またはその他の方法を指定して、指定した期間の資産の減価償却を計算します。

ステートメント	結果
select FINANCE('DDB', 2400, 300, 10*365, 1, .)	1.315068

FINANCE DISC 関数

証券の割引率を計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

FINANCE('DISC', *settlement*, *maturity*, *price*, *redemption*, [*basis*]);

必須引数

settlement
決済日を指定します。

要件 *settlement* は SAS 日付です。

データの種類 DOUBLE

maturity
満期日を指定します。

要件 *maturity* は SAS 日付です。

データの種類 DOUBLE

price
額面$100 当たりの証券の価格を指定します。

データの種類 DOUBLE

redemption
満期受け取り額を指定します。

データの種類 DOUBLE

オプション引数

basis
使用する日数計算基準の種類を示す文字または数値としてオプションのパラメーターを指定します。

数値	文字列値	日数計算方法
0	"30/360"	US (NASD) 30/360
1	"ACTUAL"	実日数/実日数
2	"ACT/360"	実日数/360
3	"ACT/365"	実日数/365
4	"EU30/360"	ヨーロッパ 30/360

データの種類 DOUBLE

例: 計算の説明: DISC

次の例では、証券の割引率を計算します。

ステートメント	結果
select FINANCE('DISC', mdy(1, 25, 2007), mdy(6, 15, 2007), 97.975, 100, 1);	0.05242

FINANCE DOLLARDE 関数

分数で表すドル価格を小数で表すドル価格に変換します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

FINANCE('DOLLARDE', *fractional-dollar*, *fraction*);

必須引数

fractional-dollar
小数で表す数値を指定します。

データの種類 DOUBLE

fraction
分母に使用する整数を指定します。

データの種類 DOUBLE

例: 計算の説明: DOLLARDE

次の例では、分数で表すドル価格を小数で表すドル価格に変換します。

ステートメント	結果
select FINANCE('DOLLARDE', 1.125, 16);	1.78125

FINANCE DOLLARFR 関数

小数で表すドル価格を分数で表すドル価格に変換します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

FINANCE('DOLLARFR', *decimal-dollar*, *fraction*);

必須引数

decimal-dollar
小数を指定します。

データの種類 DOUBLE

fraction
分母に使用する整数を指定します。

データの種類 DOUBLE

例: 計算の説明: DOLLARFR

次の例では、小数で表すドル価格を分数で表すドル価格に変換します。

ステートメント	結果
select FINANCE('DOLLARFR', 1.125, 16);	1.02

FINANCE DURATION 関数

定期的に利息を支払う証券の年度期間を計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

FINANCE('DURATION', *settlement*, *maturity*, *coupon*, *yield*, *frequency*, [*basis*]);

必須引数

settlement
決済日を指定します。

要件 *settlement* は SAS 日付です。

データの種類 DOUBLE

maturity
満期日を指定します。

要件 *maturity* は SAS 日付です。

データの種類 DOUBLE

coupon
証券の年間のクーポン率を指定します。

要件 *coupon* は、パーセンテージではなく数値として提供されます。

データの種類 DOUBLE

yield
証券の年間利回りを指定します。

データの種類 DOUBLE

frequency
年間のクーポン支払い回数を指定します。年 1 回支払いの場合は *frequency*=1、年 2 回支払いの場合は *frequency*=2、年 4 回支払いの場合は *frequency*=4 となります。

データの種類 DOUBLE

オプション引数

basis
使用する日数計算基準の種類を示す文字または数値としてオプションのパラメーターを指定します。

数値	文字列値	日数計算方法
0	"30/360"	US (NASD) 30/360
1	"ACTUAL"	実日数/実日数
2	"ACT/360"	実日数/360
3	"ACT/365"	実日数/365
4	"EU30/360"	ヨーロッパ 30/360

データの種類 DOUBLE

例: 計算の説明: DURATION

次の例では、定期的に利息を支払う証券の年度期間を計算します。

ステートメント	結果
select FINANCE('DURATION', mdy(1, 1, 2008), mdy(1, 1, 2016), 0.08, 0.09, 2, 1);	5.993775

FINANCE EFFECT 関数

有効な年利を計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

FINANCE('EFFECT', *nominal-rate*, *npery*);

必須引数

nominal-rate
名目金利を指定します。

データの種類 DOUBLE

npery
年間の複利計算期間数を指定します。

データの種類 DOUBLE

例: 計算の説明: EFFECT

次の例では、有効な年利を計算します。

ステートメント	結果
select FINANCE('EFFECT', 0.0525, 4);	0.053543

FINANCE FV 関数

投資の将来価値を計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

FINANCE('FV', *rate*, *nper*, [*payment*], [*present-value*], [*type*]);

必須引数

rate
利率を指定します。

要件 *rate* は、パーセンテージではなく数値として提供されます。

データの種類 DOUBLE

nper
支払い期間の総数を指定します。

データの種類 DOUBLE

オプション引数

payment
各期間に行われる支払いを指定します。年金の有効期間中に支払いが変更されることはありません。通常、*payment* には元本と利息が含まれますが、手数料や税金は含まれません。*payment* を省略する場合、*present-value* 引数を含める必要があります。

データの種類 DOUBLE

present-value
現在価値、または一連の将来の支払いに相当する現時点での一括払い金額を指定します。*present-value* を省略すると、0(ゼロ)とみなされ、*payment* 引数を含める必要があります。

データの種類 DOUBLE

type
0 か 1 を指定し、支払い期日がいつであるかを示します。*type* を省略すると、0 とみなされます。

支払い期日が期末の場合は、*type* 引数を省略するか、0 に設定します。支払い期日が期首の場合は、*type* を 1 に設定します。

データの種類 DOUBLE

例: 計算の説明: FV

次の例では、投資の将来価値を計算します。

ステートメント	結果
select finance('FV', 0.06/12, 10, -200, -500, 1);	2581.403

FINANCE FVSCHEDULE 関数

一連の複利率を適用した後の当初元本の将来価値を計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

FINANCE('FVSCHEDULE', *principal*, *schedule-1*, *schedule-2* ...);

必須引数

principal
現在価値を指定します。

データの種類 DOUBLE

schedule
適用する一連の利率を指定します。

要件 *schedule* 率は、パーセンテージではなく数値として提供されます。

データの種類 DOUBLE

例: 計算の説明: FVSCHEDULE

次の例では、一連の複利率を適用した後の当初元本の将来価値を計算します。

ステートメント	結果
select FINANCE('FVSCHEDULE', 1, 0.09, 0.11, 0.1);	1.33089

FINANCE INTRATE 関数

完全投資済み証券の利率を計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

FINANCE('INTRATE', *settlement*, *maturity*, *investment*, *redemption*, [*basis*]);

必須引数

settlement
決済日を指定します。

要件 *settlement* は SAS 日付です。

データの種類 DOUBLE

maturity
満期日を指定します。

要件 *maturity* は SAS 日付です。

データの種類 DOUBLE

investment
証券に投資される金額を指定します。

データの種類 DOUBLE

redemption
満期受け取り額を指定します。

データの種類 DOUBLE

オプション引数

basis
使用する日数計算基準の種類を示す文字または数値としてオプションのパラメーターを指定します。

数値	文字列値	日数計算方法
0	"30/360"	US (NASD) 30/360
1	"ACTUAL"	実日数/実日数
2	"ACT/360"	実日数/360
3	"ACT/365"	実日数/365
4	"EU30/360"	ヨーロッパ 30/360

データの種類 DOUBLE

例: 計算の説明: INTRATE

次の例では、完全投資済み証券の利率を計算します。

ステートメント	結果
select finance('intrate', mdy(2, 15, 2008), mdy(5, 15, 2008), 1000000, 1014420, 2);	0.05768

FINANCE IPMT 関数

一定期間の投資の利払いを計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

FINANCE('IPMT', *rate*, *period*, *nper*, *pv*, [*fv*], [*type*]);

必須引数

rate
利率を指定します。

要件　*rate* は、パーセンテージではなく数値として提供されます。

データの種類　DOUBLE

period
減価償却を計算する期間を指定します。*Period* には、*life* と同じ単位を使用する必要があります。

データの種類　DOUBLE

nper
支払い期間の総数を指定します。

データの種類　DOUBLE

pv
現在価値、または一連の将来の支払いに相当する現時点での一括払い金額を指定します。*pv* を省略すると、0(ゼロ)とみなされ、*fv* 引数を含める必要があります。

データの種類　DOUBLE

オプション引数

fv
将来価値、または最終支払い後に達する現金残高を指定します。*fv* を省略すると、0(ゼロ)とみなされます(たとえば、ローンの将来価値が 0)。

データの種類　DOUBLE

type
0 か 1 を指定し、支払い期日がいつであるかを示します。*type* を省略すると、0 とみなされます。

支払い期日が期末の場合は、*type* 引数を省略するか、0 に設定します。支払い期日が期首の場合は、*type* を 1 に設定します。

データの種類　DOUBLE

例: 計算の説明: IPMT

次の例では、指定期間の投資の利払いを計算します。

ステートメント	結果
select FINANCE('IPMT', 0.1/12, 2, 3, 100, ., .);	-0.55786

FINANCE IRR 関数

一連のキャッシュフローの内部利益率を計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

FINANCE('IRR', *value-1*, *value-2*, ..., *value-n*);

必須引数

value
内部利益率を計算する対象の数値を含む数値引数のリストを指定します。

データの種類 DOUBLE

例: 計算の説明: IRR

次の例では、一連のキャッシュフローの内部利益率を計算します。

ステートメント	結果
select FINANCE('IRR', -70000, 12000, 15000, 18000, 21000, 26000);	0.086631

FINANCE ISPMT 関数

投資の指定期間中に支払われる利息を計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

FINANCE('ISPMT', *interest-rate*, *period*, *number-payments*, *pv*);

必須引数

interest-rate
投資の利率を指定します。

要件 *rate* は、パーセンテージではなく数値として提供されます。

データの種類 DOUBLE

period
利率を計算する期間を指定します。*Period* は 1 から *number-payments* までの間の値にする必要があります。

データの種類 DOUBLE

number-payments
年金の支払い回数を指定します。

データの種類 DOUBLE

pv
ローン金額または支払いの現在価値を指定します。

データの種類 DOUBLE

例: 計算の説明: ISPMT

次の例では、年間 7.5%を得る$5000 の投資の 2 年間の利払いを計算します。利息の支払いは 8 か月目に計算されます。

ステートメント	結果
select FINANCE('ISPMT', 0.075/12, 8, 2*12, 5000);	-20.8333

FINANCE MDURATION 関数

推定額面$100 の証券の修正マコーレーデュレーションを計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

FINANCE('MDURATION', *settlement*, *maturity*, *coupon*, *yield*, *frequency*, [*basis*]);

必須引数

settlement
決済日を指定します。

要件 *settlement* は SAS 日付です。

データの種類 DOUBLE

maturity
満期日を指定します。

要件 *maturity* は SAS 日付です。

データの種類 DOUBLE

coupon
証券の年間のクーポン率を指定します。

要件 *coupon* は、パーセンテージではなく数値として提供されます。

データの種類 DOUBLE

yield
証券の年間利回りを指定します。

データの種類 DOUBLE

frequency
年間のクーポン支払い回数を指定します。年 1 回支払いの場合は *frequency*=1、年 2 回支払いの場合は *frequency*=2、年 4 回支払いの場合は *frequency*=4 となります。

データの種類 DOUBLE

オプション引数

basis
使用する日数計算基準の種類を示す文字または数値としてオプションのパラメーターを指定します。

数値	文字列値	日数計算方法
0	"30/360"	US (NASD) 30/360

数値	文字列値	日数計算方法
1	"ACTUAL"	実日数/実日数
2	"ACT/360"	実日数/360
3	"ACT/365"	実日数/365
4	"EU30/360"	ヨーロッパ 30/360

データの種類　DOUBLE

例: 計算の説明: MDURATION

次の例では、推定額面$100 の証券の修正マコーレーデュレーションを計算します。

ステートメント	結果
select FINANCE('MDURATION', mdy(1, 1, 2008), mdy(1, 1, 2016), 0.08, 0.09, 2, 1);	5.73567

FINANCE MIRR 関数

プラスおよびマイナスのキャッシュフローが異なる利率で資金調達される場合の内部利益率を計算します。

カテゴリ:　財務

返されるデータの種類:　DOUBLE

構文

FINANCE('MIRR', *value-1*, ..., *value-n*, *finance-rate*, *reinvest-rate*);

必須引数

value
数値を含む数値引数のリストを指定します。これらの数値は、定期的に発生する一連の支払い(負の値)と収入(正の値)を表します。*value* は、修正内部利益率を計

算するためには、正の値と負の値が少なくとも 1 つずつ含まれている必要があります。

データの種類 DOUBLE

finance-rate
キャッシュフローで使用する資金に対して支払う利率を指定します。

要件 *finance-rate* は、パーセンテージではなく数値として提供されます。

データの種類 DOUBLE

reinvest-rate
投資するときにキャッシュフローに対して受け取る利率を指定します。

要件 *reinvest-rate* は、パーセンテージではなく数値として提供されます。

データの種類 DOUBLE

例: 計算の説明: MIRR

次の例では、プラスおよびマイナスのキャッシュフローが異なる利率で資金調達される場合の内部利益率を計算します。

ステートメント	結果
select FINANCE('MIRR', -1000, 3000, 4000, 5000, 0.08, 0.10);	1.353142

FINANCE NOMINAL 関数

年間の名目金利を計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

FINANCE('NOMINAL', *effective-rate*, *npery*);

必須引数

effective-rate
有効な利率を指定します。

要件 *effective-rate* は、パーセンテージではなく数値として提供されます。

データの種類 DOUBLE

npery
年間の複利計算期間数を指定します。

データの種類 DOUBLE

例: 計算の説明: NOMINAL

次の例では、年間の名目金利を計算します。

ステートメント	結果
select FINANCE('NOMINAL', 0.08, 4);	0.077706

FINANCE NPER 関数

投資の期間数を計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

FINANCE('NPER', *rate*, *payment*, *pv*, [*fv*], [*type*]);

必須引数

rate
利率を指定します。

要件 *rate* は、パーセンテージではなく数値として提供されます。

データの種類 DOUBLE

payment
各期間に行われる支払いを指定します。年金の有効期間中に支払いが変更されることはありません。通常、*payment* には元本と利息が含まれますが、その他の手数料や税金は含まれません。*payment* を省略する場合、*pv* 引数を含める必要があります。

データの種類　DOUBLE

pv
現在価値、または一連の将来の支払いに相当する現時点での一括払い金額を指定します。*pv* を省略すると、0(ゼロ)とみなされ、*payment* 引数を含める必要があります。

データの種類　DOUBLE

オプション引数

fv
将来価値、または最終支払い後に達する現金残高を指定します。*fv* を省略すると、0(ゼロ)とみなされます(たとえば、ローンの将来価値が 0)。

データの種類　DOUBLE

type
0 か 1 を指定し、支払い期日がいつであるかを示します。*type* を省略すると、0 とみなされます。

支払い期日が期末の場合は、*type* 引数を省略するか、0 に設定します。支払い期日が期首の場合は、*type* を 1 に設定します。

データの種類　DOUBLE

例: 計算の説明: NPER

次の例では、投資の期間数を計算します。

ステートメント	結果
select FINANCE('NPER', 0.08, 200, 1000, 0, 0);	-4.37198

FINANCE NPV 関数

一連の定期的なキャッシュフローと割引率に基づいて投資の正味現在価値を計算します。

カテゴリ:　財務

返されるデータの種類: DOUBLE

構文

FINANCE('NPV', *rate*, *value-1*, [, ... *value-n*]);

必須引数

rate
利率を指定します。

要件 *rate* は、パーセンテージではなく数値として提供されます。

データの種類 DOUBLE

value
一連のキャッシュフローを表します。

データの種類 DOUBLE

例: 計算の説明: NPV

次の例では、一連の定期的なキャッシュフローと割引率に基づいて投資の正味現在価値を計算します。

ステートメント	結果
select FINANCE('NPV', 0.08, 200, 1000, 0);	1042.524

FINANCE ODDFPRICE 関数

端数の開始期間で額面$100 当たりの証券の価格を計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

FINANCE('ODDFPRICE', *settlement*, *maturity*, *issue*, *first-coupon*, *rate*, *yield*,

redemption, *frequency*, [*basis*]);

必須引数

settlement
決済日を指定します。

要件 *settlement* は SAS 日付です。

データの種類 DOUBLE

maturity
満期日を指定します。

要件 *maturity* は SAS 日付です。

データの種類 DOUBLE

issue
証券の発行日を指定します。

要件 *issue* は SAS 日付です。

データの種類 DOUBLE

first-coupon
証券の開始クーポン日を指定します。

要件 *first-coupon* は SAS 日付です。

データの種類 DOUBLE

rate
利率を指定します。

要件 *rate* は、パーセンテージではなく数値として提供されます。

データの種類 DOUBLE

yield
証券の年間利回りを指定します。

データの種類 DOUBLE

redemption
満期受け取り額を指定します。

データの種類 DOUBLE

frequency
年間のクーポン支払い回数を指定します。年 1 回支払いの場合は *frequency*=1、年 2 回支払いの場合は *frequency*=2、年 4 回支払いの場合は *frequency*=4 となります。

データの種類 DOUBLE

オプション引数

basis
使用する日数計算基準の種類を示す文字または数値としてオプションのパラメーターを指定します。

数値	文字列値	日数計算方法
0	"30/360"	US (NASD) 30/360
1	"ACTUAL"	実日数/実日数
2	"ACT/360"	実日数/360
3	"ACT/365"	実日数/365
4	"EU30/360"	ヨーロッパ 30/360

データの種類　DOUBLE

例: 計算の説明: ODDFPRICE

次の例では、端数の開始期間で額面$100 当たりの証券の価格を計算します。

ステートメント	結果
select FINANCE('ODDFPRICE', mdy(1, 15, 93), mdy(1, 1, 98), mdy(1, 1, 93), mdy(7, 1, 94), 0.07, 0.06, 100, 2, 0);	103.941

FINANCE ODDFYIELD 関数

端数の開始期間で証券の利回りを計算します。

カテゴリ:　財務

返されるデータの種類:　DOUBLE

構文

FINANCE('ODDFYIELD', *settlement*, *maturity*, *issue*, *first-coupon*, *rate*, *price*,

redemption, *frequency*, [*basis*]);

必須引数

settlement
決済日を指定します。

要件 *settlement* は SAS 日付です。

データの種類 DOUBLE

maturity
満期日を指定します。

要件 *maturity* は SAS 日付です。

データの種類 DOUBLE

issue
証券の発行日を指定します。

要件 *issue* は SAS 日付です。

データの種類 DOUBLE

first-coupon
証券の開始クーポン日を指定します。

要件 *first-coupon* は SAS 日付です。

データの種類 DOUBLE

rate
利率を指定します。

要件 *rate* は、パーセンテージではなく数値として提供されます。

データの種類 DOUBLE

price
額面$100 当たりの証券の価格を指定します。

データの種類 DOUBLE

redemption
満期受け取り額を指定します。

データの種類 DOUBLE

frequency
年間のクーポン支払い回数を指定します。年 1 回支払いの場合は *frequency*=1、年 2 回支払いの場合は *frequency*=2、年 4 回支払いの場合は *frequency*=4 となります。

データの種類 DOUBLE

オプション引数

basis
使用する日数計算基準の種類を示す文字または数値としてオプションのパラメーターを指定します。

数値	文字列値	日数計算方法
0	"30/360"	US (NASD) 30/360
1	"ACTUAL"	実日数/実日数
2	"ACT/360"	実日数/360
3	"ACT/365"	実日数/365
4	"EU30/360"	ヨーロッパ 30/360

データの種類 DOUBLE

例: 計算の説明: ODDFYIELD

次の例では、端数の開始期間で利回りの利息を計算します。

ステートメント	結果
select FINANCE('ODDFYIELD', mdy(1, 15, 93), mdy(1, 1, 98), mdy(1, 1, 93), mdy(7, 1, 94), 0.07, 103.94103984, 100, 2, 0);	0.06

FINANCE ODDLPRICE 関数

端数の最終期間で額面$100 当たりの証券の価格を計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

FINANCE('ODDLPRICE', *settlement*, *maturity*, *last-interest*, *rate*, *yield*,

redemption, *frequency*, [*basis*]);

必須引数

settlement
決済日を指定します。

要件 *settlement* は SAS 日付です。

データの種類 DOUBLE

maturity
満期日を指定します。

要件 *maturity* は SAS 日付です。

データの種類 DOUBLE

last-interest
証券の最終クーポン日を指定します。

要件 *last-interest* は SAS 日付です。

データの種類 DOUBLE

rate
利率を指定します。

要件 *rate* は、パーセンテージではなく数値として提供されます。

データの種類 DOUBLE

yield
証券の年間利回りを指定します。

データの種類 DOUBLE

redemption
満期受け取り額を指定します。

データの種類 DOUBLE

frequency
年間のクーポン支払い回数を指定します。年 1 回支払いの場合は *frequency*=1、年 2 回支払いの場合は *frequency*=2、年 4 回支払いの場合は *frequency*=4 となります。

データの種類 DOUBLE

オプション引数

basis
使用する日数計算基準の種類を示す文字または数値としてオプションのパラメーターを指定します。

数値	文字列値	日数計算方法
0	"30/360"	US (NASD) 30/360
1	"ACTUAL"	実日数/実日数
2	"ACT/360"	実日数/360
3	"ACT/365"	実日数/365
4	"EU30/360"	ヨーロッパ 30/360

データの種類　DOUBLE

例: 計算の説明: ODDLPRICE

次の例では、端数の最終期間で額面$100 当たりの証券の価格を計算します。

ステートメント	結果
select FINANCE('ODDLPRICE', mdy(2, 7, 2008), mdy(6, 15, 2008), mdy(10, 15, 2007), 0.0375, 0.0405, 100, 2, 0);	99.87829

FINANCE ODDLYIELD 関数

端数の最終期間で証券の利回りを計算します。

カテゴリ:　財務

返されるデータの種類:　DOUBLE

構文

FINANCE('ODDLYIELD', *settlement*, *maturity*, *last-interest*, *rate*, *price*, *redemption*, *frequency*, [*basis*]);

必須引数

settlement
決済日を指定します。

要件 *settlement* は SAS 日付です。

データの種類 DOUBLE

maturity
満期日を指定します。

要件 *maturity* は SAS 日付です。

データの種類 DOUBLE

last-interest
証券の最終クーポン日を指定します。

要件 *last-interest* は SAS 日付です。

データの種類 DOUBLE

rate
利率を指定します。

要件 *rate* は、パーセンテージではなく数値として提供されます。

データの種類 DOUBLE

price
額面$100 当たりの証券の価格を指定します。

データの種類 DOUBLE

redemption
満期受け取り額を指定します。

データの種類 DOUBLE

frequency
年間のクーポン支払い回数を指定します。年 1 回支払いの場合は *frequency*=1、年 2 回支払いの場合は *frequency*=2、年 4 回支払いの場合は *frequency*=4 となります。

データの種類 DOUBLE

オプション引数

basis
使用する日数計算基準の種類を示す文字または数値としてオプションのパラメーターを指定します。

数値	文字列値	日数計算方法
0	"30/360"	US (NASD) 30/360
1	"ACTUAL"	実日数/実日数
2	"ACT/360"	実日数/360
3	"ACT/365"	実日数/365
4	"EU30/360"	ヨーロッパ 30/360

データの種類　DOUBLE

例: 計算の説明: ODDLYIELD

次の例では、端数の最終期間で証券の利回りを計算します。

ステートメント	結果
select FINANCE('ODDLYIELD', mdy(2, 7, 2008), mdy(6, 15, 2008), mdy(10, 15, 2007), 0.0375, 99.878286015, 100, 2, 0);	0.0405

FINANCE PMT 関数

年金の定期的支払いを計算します。

カテゴリ:　財務

返されるデータの種類:　DOUBLE

構文

FINANCE('PMT', *rate*, *nper*, *pv*, [*fv*], [*type*]);

必須引数

rate
　利率を指定します。

要件 *rate* は、パーセンテージではなく数値として提供されます。

データの種類 DOUBLE

nper
支払い期間の総数を指定します。

データの種類 DOUBLE

pv
現在価値、または一連の将来の支払いに相当する現時点での一括払い金額を指定します。*pv* を省略すると、0(ゼロ)とみなされ、*fv* 引数を含める必要があります。

データの種類 DOUBLE

オプション引数

fv
将来価値、または最終支払い後に達する現金残高を指定します。*fv* を省略すると、0(ゼロ)とみなされます(たとえば、ローンの将来価値が 0)。

データの種類 DOUBLE

type
0 か 1 を指定し、支払い期日がいつであるかを示します。*type* を省略すると、0 とみなされます。

支払い期日が期末の場合は、*type* 引数を省略するか、0 に設定します。支払い期日が期首の場合は、*type* を 1 に設定します。

データの種類 DOUBLE

例: 計算の説明: PMT

次の例では、年金の定期的支払いを計算します。

ステートメント	結果
select FINANCE('PMT', 0.08, 5, 91, 3, 0);	-23.3029

FINANCE PPMT 関数

指定期間の投資の元本に対する支払いを計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

FINANCE('PPMT', *rate*, *period*, *nper*, *pv*, [*fv*], [*type*]);

必須引数

rate
利率を指定します。

要件 *rate* は、パーセンテージではなく数値として提供されます。

データの種類 DOUBLE

period
期間を指定します。

範囲: 1–*nper*

データの種類 DOUBLE

nper
支払期間の数を指定します。

データの種類 DOUBLE

pv
現在価値、または一連の将来の支払いに相当する現時点での一括払い金額を指定します。*pv* を省略すると、0(ゼロ)とみなされ、*fv* 引数を含める必要があります。

データの種類 DOUBLE

オプション引数

fv
将来価値、または最終支払い後に達する現金残高を指定します。*fv* を省略すると、0(ゼロ)とみなされます(たとえば、ローンの将来価値が 0)。

データの種類 DOUBLE

type
0 か 1 を指定し、支払い期日がいつであるかを示します。*type* を省略すると、0 とみなされます。

支払い期日が期末の場合は、*type* 引数を省略するか、0 に設定します。支払い期日が期首の場合は、*type* を 1 に設定します。

データの種類 DOUBLE

例: 計算の説明: PPMT

次の例では、指定期間の投資の元本に対する支払いを計算します。

ステートメント	結果
select FINANCE('PPMT', 0.08, 10, 10, 200000, 0, 0);	-27598.1

FINANCE PRICE 関数

定期的に利息を支払う額面$100 当たりの証券の価格を計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

FINANCE('PRICE', *settlement*, *maturity*, *rate*, *yield*, *redemption*, *frequency*, [*basis*]);

必須引数

settlement
決済日を指定します。

要件 *settlement* は SAS 日付です。

データの種類 DOUBLE

maturity
満期日を指定します。

要件 *maturity* は SAS 日付です。

データの種類 DOUBLE

rate
利率を指定します。

要件 *rate* は、パーセンテージではなく数値として提供されます。

データの種類 DOUBLE

yield
証券の年間利回りを指定します。

データの種類 DOUBLE

redemption
満期受け取り額を指定します。

データの種類 DOUBLE

frequency
年間のクーポン支払い回数を指定します。年 1 回支払いの場合は *frequency*=1、年 2 回支払いの場合は *frequency*=2、年 4 回支払いの場合は *frequency*=4 となります。

データの種類 DOUBLE

オプション引数

basis
使用する日数計算基準の種類を示す文字または数値としてオプションのパラメーターを指定します。

数値	文字列値	日数計算方法
0	"30/360"	US (NASD) 30/360
1	"ACTUAL"	実日数/実日数
2	"ACT/360"	実日数/360
3	"ACT/365"	実日数/365
4	"EU30/360"	ヨーロッパ 30/360

データの種類 DOUBLE

例: 計算の説明: PRICE

次の例では、定期的に利息を支払う額面$100 当たりの証券の価格を計算します。

ステートメント	結果
select FINANCE('PRICE', mdy(2,15,2008), mdy(11,15,2017), 0.0575, 0.065, 100, 2, 0);	94.63436

FINANCE PRICEDISC 関数

額面$100 当たりの割引証券の価格を計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

FINANCE('PRICEDISC', *settlement*, *maturity*, *discount*, *redemption*, [*basis*]);

必須引数

settlement
決済日を指定します。

要件 *settlement* は SAS 日付です。

データの種類 DOUBLE

maturity
満期日を指定します。

要件 *maturity* は SAS 日付です。

データの種類 DOUBLE

discount
証券の割引率を指定します。

要件 *discount* は、パーセンテージではなく数値として提供されます。

データの種類 DOUBLE

redemption
満期受け取り額を指定します。

データの種類 DOUBLE

オプション引数

basis
使用する日数計算基準の種類を示す文字または数値としてオプションのパラメーターを指定します。

数値	文字列値	日数計算方法
0	"30/360"	US (NASD) 30/360
1	"ACTUAL"	実日数/実日数
2	"ACT/360"	実日数/360
3	"ACT/365"	実日数/365
4	"EU30/360"	ヨーロッパ 30/360

データの種類　DOUBLE

例: 計算の説明: PRICEDISC

次の例では、額面$100 当たりの割引証券の価格を計算します。

ステートメント	結果
select FINANCE('PRICEDISC', mdy(2,15,2008), mdy(11,15,2017), 0.0525, 100, 0);	48.8125

FINANCE PRICEMAT 関数

満期に利息を支払う額面$100 当たりの証券の価格を計算します。

カテゴリ:　財務

返されるデータの種類:　DOUBLE

構文

FINANCE('PRICEMAT', *settlement*, *maturity*, *issue*, *rate*, *yield*, [*basis*]);

必須引数

settlement
　決済日を指定します。

要件 *settlement* は SAS 日付です。

データの種類 DOUBLE

maturity
満期日を指定します。

要件 *maturity* は SAS 日付です。

データの種類 DOUBLE

issue
証券の発行日を指定します。

要件 *issue* は SAS 日付です。

データの種類 DOUBLE

rate
利率を指定します。

要件 *rate* は、パーセンテージではなく数値として提供されます。

データの種類 DOUBLE

yield
証券の年間利回りを指定します。

データの種類 DOUBLE

オプション引数

basis
使用する日数計算基準の種類を示す文字または数値としてオプションのパラメーターを指定します。

数値	文字列値	日数計算方法
0	"30/360"	US (NASD) 30/360
1	"ACTUAL"	実日数/実日数
2	"ACT/360"	実日数/360
3	"ACT/365"	実日数/365
4	"EU30/360"	ヨーロッパ 30/360

データの種類 DOUBLE

例: 計算の説明: PRICEMAT

次の例では、満期に利息を支払う額面$100 当たりの証券の価格を計算します。

ステートメント	結果
select FINANCE('PRICEMAT', mdy(2,15,2008), mdy(4, 13, 2008), mdy(11, 11, 2007), 0.061, 0.061, 0);	99.9845

FINANCE PV 関数

投資の現在価値を計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

FINANCE('PV', *rate*, *nper*, *payment*, [*fv*], [*type*]);

必須引数

rate
利率を指定します。

要件 *rate* は、パーセンテージではなく数値として提供されます。

データの種類 DOUBLE

nper
支払い期間の総数を指定します。

データの種類 DOUBLE

payment
各期間に行われる支払いを指定します。年金の有効期間中に支払いが変更されることはありません。通常、*payment* には元本と利息が含まれますが、その他の手数料や税金は含まれません。

データの種類 DOUBLE

オプション引数

fv
将来価値、または最終支払い後に達する現金残高を指定します。*fv* を省略すると、0(ゼロ)とみなされます(たとえば、ローンの将来価値が 0)。

データの種類 DOUBLE

type
0 か 1 を指定し、支払い期日がいつであるかを示します。*type* を省略すると、0 とみなされます。

支払い期日が期末の場合は、*type* 引数を省略するか、0 に設定します。支払い期日が期首の場合は、*type* を 1 に設定します。

データの種類 DOUBLE

例: 計算の説明: PV

次の例では、投資の現在価値を計算します。

ステートメント	結果
select FINANCE('PV', 0.05, 10, 1000, 200, 0);	-7844.52

FINANCE RATE 関数

年金期間当たりの利率を計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

FINANCE('RATE', *nper*, *payment*, *pv*, [*fv*], [*type*]);

必須引数

nper
支払い期間の総数を指定します。

データの種類 DOUBLE

payment
各期間に行われる支払いを指定します。年金の有効期間中に支払いが変更されることはありません。通常、*payment* には元本と利息が含まれますが、その他の手数料や税金は含まれません。*payment* を省略する場合、*pv* 引数を含める必要があります。

データの種類　DOUBLE

pv
現在価値、または一連の将来の支払いに相当する現時点での一括払い金額を指定します。*pv* を省略すると、0(ゼロ)とみなされ、*fv* 引数を含める必要があります。

データの種類　DOUBLE

オプション引数

fv
将来価値、または最終支払い後に達する現金残高を指定します。*fv* を省略すると、0(ゼロ)とみなされます(たとえば、ローンの将来価値が 0)。

データの種類　DOUBLE

type
0 か 1 を指定し、支払い期日がいつであるかを示します。*type* を省略すると、0 とみなされます。

支払い期日が期末の場合は、*type* 引数を省略するか、0 に設定します。支払い期日が期首の場合は、*type* を 1 に設定します。

データの種類　DOUBLE

例: 計算の説明: RATE

次の例では、年金期間当たりの利率を計算します。

ステートメント	結果
select FINANCE('RATE', 4, -2481, 8000);	0.092148

FINANCE RECEIVED 関数

完全投資済み証券の満期受け取り額を計算します。

カテゴリ:　財務

返されるデータの種類:　DOUBLE

構文

FINANCE('RECEIVED', *settlement*, *maturity*, *investment*, *discount*, [*basis*]);

必須引数

settlement
決済日を指定します。

要件 *settlement* は SAS 日付です。

データの種類 DOUBLE

maturity
満期日を指定します。

要件 *maturity* は SAS 日付です。

データの種類 DOUBLE

investment
証券に投資される金額を指定します。

データの種類 DOUBLE

discount
証券の割引率を指定します。

要件 *discount* は、パーセンテージではなく数値として提供されます。

データの種類 DOUBLE

オプション引数

basis
使用する日数計算基準の種類を示す文字または数値としてオプションのパラメーターを指定します。

数値	文字列値	日数計算方法
0	"30/360"	US (NASD) 30/360
1	"ACTUAL"	実日数/実日数
2	"ACT/360"	実日数/360
3	"ACT/365"	実日数/365
4	"EU30/360"	ヨーロッパ 30/360

データの種類 DOUBLE

例: 計算の説明: RECEIVED

次の例では、完全投資済み証券の満期受け取り額を計算します。

ステートメント	結果
select FINANCE('RECEIVED', mdy(2,15,2008), mdy(5, 15, 2008), 1000000, 0.0575, 2);	1014585

FINANCE SLN 関数

ある期間の資産の定額法による減価償却を計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

FINANCE('SLN', *cost*, *salvage*, *life*);

必須引数

cost
資産の初期コストを指定します。

データの種類 DOUBLE

salvage
減価償却を終了した時点の価値(資産の救済価額とも呼ばれる)を指定します。

データの種類 DOUBLE

life
資産が減価償却される期間数(資産の耐用年数とも呼ばれる)を指定します。

データの種類 DOUBLE

例: 計算の説明: SLN

次の例では、ある期間の資産の定額法による減価償却を計算します。

ステートメント	結果
select FINANCE('SLN', 2000, 200, 11);	163.6364

FINANCE SYD 関数

指定期間の資産の年次級数和法による減価償却を計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

FINANCE('SYD', *cost*, *salvage*, *life*, *period*);

必須引数

cost
資産の初期コストを指定します。

データの種類 DOUBLE

salvage
減価償却を終了した時点の価値(資産の救済価額とも呼ばれる)を指定します。

データの種類 DOUBLE

life
資産が減価償却される期間数(資産の耐用年数とも呼ばれる)を指定します。

データの種類 DOUBLE

period
引数 *life* に使用するものと同じ時間単位で期間を指定します。

データの種類 DOUBLE

例: 計算の説明: SYD

次の例では、指定期間の資産の年次級数和法による減価償却を計算します。

ステートメント	結果
select FINANCE('SYD', 2000, 200, 11, 1);	300

FINANCE TBILLEQ 関数

米国財務省短期証券の債券換算利回りを計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

FINANCE('TBILLEQ', *settlement*, *maturity*, *discount*);

必須引数

settlement
決済日を指定します。

要件 *settlement* は SAS 日付です。

データの種類 DOUBLE

maturity
満期日を指定します。

要件 *maturity* は SAS 日付です。

データの種類 DOUBLE

discount
証券の割引率を指定します。

要件 *discount* は、パーセンテージではなく数値として提供されます。

データの種類 DOUBLE

例: 計算の説明: TBILLEQ

次の例では、米国財務省短期証券の債券換算利回りを計算します。

ステートメント	結果
select FINANCE('TBILLEQ', mdy(3, 31, 2008), mdy(6, 1, 2008), 0.0914);	0.094151

FINANCE TBILLPRICE 関数

額面$100 当たりの米国財務省短期証券の価格を計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

FINANCE('TBILLPRICE', *settlement*, *maturity*, *discount*);

必須引数

settlement
決済日を指定します。

要件 *settlement* は SAS 日付です。

データの種類 DOUBLE

maturity
満期日を指定します。

要件 *maturity* は SAS 日付です。

データの種類 DOUBLE

discount
証券の割引率を指定します。

要件 *discount* は、パーセンテージではなく数値として提供されます。

データの種類 DOUBLE

例: 計算の説明: TBILLPRICE

次の例では、額面$100 当たりの米国財務省短期証券の価格を計算します。

ステートメント	結果
select FINANCE('TBILLPRICE', mdy(3, 31, 2008), mdy(6, 1, 2008), 0.09);	98.45

FINANCE TBILLYIELD 関数

米国財務省短期証券の利回りを計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

FINANCE('TBILLYIELD', *settlement*, *maturity*, *price*);

必須引数

settlement
決済日を指定します。

要件 *settlement* は SAS 日付です。

データの種類 DOUBLE

maturity
満期日を指定します。

要件 *maturity* は SAS 日付です。

データの種類 DOUBLE

price
額面$100 当たりの証券の価格を指定します。

データの種類 DOUBLE

例: 計算の説明: TBILLYIELD

次の例では、米国財務省短期証券の利回りを計算します。

ステートメント	結果
select FINANCE('TBILLYIELD', mdy(3, 31, 2008), mdy(6, 1, 2008), 98);	0.118499

FINANCE VDB 関数

定率法を使用して、指定期間または一部期間の資産の減価償却を計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

FINANCE('VDB', *cost*, *salvage*, *life*, *start-period*, *end-period*, [*factor*], [*noswitch*]);

必須引数

cost
資産の初期コストを指定します。

データの種類 DOUBLE

salvage
減価償却を終了した時点の価値(資産の救済価額とも呼ばれる)を指定します。

データの種類 DOUBLE

life
資産が減価償却される期間数(資産の耐用年数とも呼ばれる)を指定します。

データの種類 DOUBLE

start-period
計算の開始期間を指定します。支払い期間は 1 から番号が付けられます。

データの種類 DOUBLE

end-period
計算の最終期間を指定します。

データの種類 DOUBLE

オプション引数

factor
減価償却率を指定します。*factor* を省略すると、2 とみなされます(倍額定率法)。

データの種類 DOUBLE

noswitch
減価償却費が定率法による計算よりも大きい場合、定額法による減価償却に切り替えるかどうかを決定する論理値を指定します。*noswitch* を省略すると、1 とみなされます。

データの種類 DOUBLE

例: 計算の説明: VDB

次の例では、定率法を使用して、指定期間または一部期間の資産の減価償却を計算します。

ステートメント	結果
select FINANCE('VDB', 2400, 300, 10, 0, 1, 1.5);	360

FINANCE XIRR 関数

必ずしも定期的ではないキャッシュフロースケジュールの 内部利益率を計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

FINANCE('XIRR', *values*, *dates*, [*guess*]);

必須引数

values
支払いのスケジュール(日付)に対応する一連のキャッシュフローを指定します。1 回目の支払いは任意で、投資開始時に発生するコストまたは支払いに対応しま

す。開始値がコストまたは支払いの場合、負の値にする必要があります。それ以降のすべての 支払いは、1 年 365 日に基づいて割引されます。一連の値には、正の値と負の値が少なくとも 1 つずつ含まれている必要があります。

データの種類 DOUBLE

dates
キャッシュフローの支払いに対応する支払い日のスケジュールを指定します。1 回目の支払い日は、支払いスケジュールの開始時を示します。その他のすべての日付は、この日付より後にする必要がありますが、どの順序で発生してもかまいません。

要件 *dates* は SAS 日付です。

データの種類 DOUBLE

オプション引数

guess
XIRR の結果に近い推測する数値を指定します(省略可能)。

データの種類 DOUBLE

例: 計算の説明: XIRR

次の例では、必ずしも定期的ではないキャッシュフロースケジュールの 内部利益率を計算します。

ステートメント	結果
select FINANCE('XIRR', -10000, 2750, 4250, 3250, 2750, mdy(1, 1, 2008), mdy(3, 1, 2008), mdy(10, 30, 2008), mdy(2, 15, 2009), mdy(4, 1, 2009), 0.1);	0.373363

FINANCE XNPV 関数

必ずしも定期的ではないキャッシュフロースケジュールの 正味現在価値を計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

FINANCE('XNPV', *rate*, *values*, *dates*);

必須引数

rate
利率を指定します。

要件 *rate* は、パーセンテージではなく数値として提供されます。

データの種類 DOUBLE

values
支払いのスケジュール(日付)に対応する一連のキャッシュフローを指定します。1 回目の支払いは任意で、投資開始時に発生するコストまたは支払いに対応します。開始値がコストまたは支払いの場合、負の値にする必要があります。それ以降のすべての 支払いは、1 年 365 日に基づいて割引されます。一連の値には、正の値と負の値が少なくとも 1 つずつ含まれている必要があります。

データの種類 DOUBLE

dates
キャッシュフローの支払いに対応する支払い日のスケジュールを指定します。1 回目の支払い日は、支払いスケジュールの開始時を示します。その他のすべての日付は、この日付より後にする必要がありますが、どの順序で発生してもかまいません。

要件 *dates* は SAS 日付です。

データの種類 DOUBLE

例: 計算の説明: XNPV

次の例では、必ずしも定期的ではないキャッシュフロースケジュールの 正味現在価値を計算します。

ステートメント	結果
select FINANCE('XNPV', .09, -10000, 2750, 4250, 3250, 2750, mdy(1, 1, 2008), mdy(3, 1, 2008), mdy(10, 30, 2008), mdy(2, 15, 2009), mdy(4, 1, 2009));	2086.648

FINANCE YIELD 関数

定期的に利息を支払う証券の利回りを計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

FINANCE('YIELD', *settlement*, *maturity*, *rate*, *price*, *redemption*, *frequency*, [*basis*]);

必須引数

settlement
決済日を指定します。

要件 *settlement* は SAS 日付です。

データの種類 DOUBLE

maturity
満期日を指定します。

要件 *maturity* は SAS 日付です。

データの種類 DOUBLE

rate
利率を指定します。

要件 *rate* は、パーセンテージではなく数値として提供されます。

データの種類 DOUBLE

price
額面$100 当たりの証券の価格を指定します。

データの種類 DOUBLE

redemption
満期受け取り額を指定します。

データの種類 DOUBLE

frequency
年間のクーポン支払い回数を指定します。年 1 回支払いの場合は *frequency*=1、年 2 回支払いの場合は *frequency*=2、年 4 回支払いの場合は *frequency*=4 となります。

データの種類　DOUBLE

オプション引数

basis
使用する日数計算基準の種類を示す文字または数値としてオプションのパラメーターを指定します。

数値	文字列値	日数計算方法
0	"30/360"	US (NASD) 30/360
1	"ACTUAL"	実日数/実日数
2	"ACT/360"	実日数/360
3	"ACT/365"	実日数/365
4	"EU30/360"	ヨーロッパ 30/360

データの種類　DOUBLE

例: 計算の説明: YIELD

次の例では、定期的に利息を支払う証券の利回りを計算します。

ステートメント	結果
select FINANCE('YIELD', mdy(2, 15, 2008), mdy(11, 15, 2016), 0.0575, 95.04287, 100, 2, 0);	0.065

FINANCE YIELDDISC 関数

割引証券(米国財務省短期証券など)の年間利回りを計算します。

カテゴリ:　財務

返されるデータの種類: DOUBLE

構文

FINANCE('YIELDDISC', *settlement*, *maturity*, *price*, *redemption*, [*basis*]);

必須引数

settlement
決済日を指定します。

要件 *settlement* は SAS 日付です。

データの種類 DOUBLE

maturity
満期日を指定します。

要件 *maturity* は SAS 日付です。

データの種類 DOUBLE

price
額面$100 当たりの証券の価格を指定します。

データの種類 DOUBLE

redemption
満期受け取り額を指定します。

データの種類 DOUBLE

オプション引数

basis
使用する日数計算基準の種類を示す文字または数値としてオプションのパラメーターを指定します。

数値	文字列値	日数計算方法
0	"30/360"	US (NASD) 30/360
1	"ACTUAL"	実日数/実日数
2	"ACT/360"	実日数/360
3	"ACT/365"	実日数/365
4	"EU30/360"	ヨーロッパ 30/360

データの種類 DOUBLE

例: 計算の説明: YIELDDISC

次の例では、割引証券(米国財務省短期証券など)の年間利回りを計算します。

ステートメント	結果
select FINANCE('YIELDDISC', mdy(2,15,2008), mdy(11, 15, 2016), 95.04287, 100, 0);	0.005961

FINANCE YIELDMAT 関数

満期に利息を支払う証券の年間利回りを計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

FINANCE('YIELDMAT', *settlement*, *maturity*, *issue*, *rate*, *price*, [*basis*]);

必須引数

settlement
決済日を指定します。

要件 *settlement* は SAS 日付です。

データの種類 DOUBLE

maturity
満期日を指定します。

要件 *maturity* は SAS 日付です。

データの種類 DOUBLE

issue
証券の発行日を指定します。

要件 *issue* は SAS 日付です。

データの種類 DOUBLE

rate
利率を指定します。

要件 *rate* は、パーセンテージではなく数値として提供されます。

データの種類 DOUBLE

price
額面$100 当たりの証券の価格を指定します。

データの種類 DOUBLE

オプション引数

basis
使用する日数計算基準の種類を示す文字または数値としてオプションのパラメーターを指定します。

数値	文字列値	日数計算方法
0	"30/360"	US (NASD) 30/360
1	"ACTUAL"	実日数/実日数
2	"ACT/360"	実日数/360
3	"ACT/365"	実日数/365
4	"EU30/360"	ヨーロッパ 30/360

データの種類 DOUBLE

例: 計算の説明: YIELDMAT

次の例では、満期に利息を支払う証券の年間利回りを計算します。

ステートメント	結果
select FINANCE('YIELDMAT', mdy(3, 15, 2008), mdy(11, 3, 2008), mdy(11, 8, 2007), 0.0625, 100.0123, 0);	0.060954

FLOOR 関数

数値式以下の最大の整数を返します。

カテゴリ: 切り捨て

返されるデータの種類: DECIMAL、DOUBLE、NUMERIC

構文

FLOOR(*expression*)

必須引数

expression
数値に評価される有効な式を指定します。

データの種類 DECIMAL、DOUBLE、NUMERIC

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

expression が整数の 1E-12 以内の場合、関数はその整数を返します。結果が DOUBLE の範囲に収まらない数値の場合、FLOOR 関数は失敗します。

引数が DECIMAL の場合、結果は DECIMAL になります。それ以外の場合、引数は (まだ変換されていなければ) DOUBLE に変換され、結果は DOUBLE になります。

比較

FLOOR 関数は、結果が整数の 1E-12 内にある場合はその整数を返すように、結果をファジー処理します。FLOORZ 関数はゼロファジーを使用します。そのため、FLOORZ 関数では予期しない結果になる可能性があります。

例

テーブル: DENSITIES (1260 ページ)

次のステートメントは、FLOOR 関数を示しています。

ステートメント	結果
select floor(1.95);	1
select floor(density) from densities;	67 306 30 323 20 383 31 309 6 247

関連項目

関数:

- “CEIL 関数” (380 ページ)
- “CEILZ 関数” (381 ページ)
- “FLOORZ 関数” (543 ページ)

FLOORZ 関数

ゼロファジーを使用して、引数より小さいか等しい整数のうち最大の値を返します。

カテゴリ: 切り捨て

返されるデータの種類: DOUBLE

構文

FLOORZ(*expression*)

必須引数

expression
数値に評価される有効な式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

比較

FLOOR 関数とは異なり、FLOORZ 関数はゼロファジーを使用します。引数が整数の 1E-12 内にある場合、FLOOR 関数はその整数に等しくなるように結果をファジー処理します。FLOORZ 関数は結果をファジー処理しません。そのため、FLOORZ 関数では予期しない結果になる可能性があります。

例

テーブル: DENSITIES (1260 ページ)

次のステートメントは、FLOORZ 関数を示しています。

ステートメント	結果
select floorz(-2.4);	-3
select floorz(-1.6);	-2
select floorz(density) from densities;	67 306 30 323 20 383 31 309 6 247

関連項目

関数:

- “CEIL 関数” (380 ページ)
- “CEILZ 関数” (381 ページ)
- “FLOOR 関数” (542 ページ)

FMTINFO 関数

SAS 出力形式または入力形式の情報を返します。

カテゴリ: 特殊

制限事項: この関数は、SAS で提供されている出力形式の情報を返します。FORMAT プロシジャで作成されるユーザー定義出力形式には使用できません。

構文

FMTINFO('*format-name*', '*information-type*');

必須引数

'*format-name*'
SAS 出力形式または入力形式の名前を指定します。

要件 *format-name* は、単一引用符で囲む必要があります。

information-type
返される情報の種類を指定します。*format-information* には、'DESC' (簡単な説明)、'MINW' (最小幅)、'MAXW' (最大幅)のいずれかの値を指定できます。追加の情報の種類が使用可能です。詳細については、*SAS DS2 言語リファレンス*を参照してください。

返される情報の種類を指定します。*format-information* には、次のいずれかの値を指定できます。

'CAT'
関数カテゴリを返します。

'TYPE'
format-name が出力形式、入力形式、またはその両方のいずれであるかを返します。

'DESC'
出力形式または入力形式の簡単な説明を返します。

'MIND'
出力形式または入力形式の小数点以下の最小桁数を返します。

'MAXD'
出力形式または入力形式の小数点以下の最大桁数を返します。

'DEFD'
出力形式または入力形式の小数点以下のデフォルト桁数を返します。

'MINW'
出力形式または入力形式の最小の幅の値を返します。

'MAXW'
出力形式または入力形式の最大の幅の値を返します。

'DEFW'
出力形式または入力形式のデフォルトの幅の値を返します。

制限事項 *information-type* 引数は 1 つだけ指定できます。

要件 *information-type* は、単一引用符で囲む必要があります。

詳細

FMTINFO 関数は、出力形式または入力形式の情報を返します。言語要素の種類、言語要素の説明、小数点以下の値の最小/最大/デフォルト値、幅の値の最小/最大/デフォルト値などの、出力形式または入力形式のカテゴリに関する情報を返すことができます。

FMTINFO 関数で複数の引数を指定することはできません。

FMTINFO 関数は、数値引数 MIND、MAXD、DEFD、MINW、MAXW、DEFW を含めて、すべてのデータ値に対して文字列を返します。

例

次の例では、DATE 出力形式についての情報を返します。

```
select fmtinfo('date','type') as ftype,
    fmtinfo('date','cat') as fcat,
    fmtinfo('date','desc') as fdesc,
    fmtinfo('date','mind') as fmind,
    fmtinfo('date','maxd') as fmaxd,
    fmtinfo('date','defd') as fdefd,
    fmtinfo('date','minw') as fminw,
    fmtinfo('date','maxw') as fmaxw,
    fmtinfo('date','defw') as fdefw;
```

アウトプット 4.8 DATE 出力形式での FMTINFO クエリの結果

FTYPE	FCAT	FDESC	FMIND	FMAXD	FDEFD	FMINW	FMAXW	FDEFW
BOTH	date	date value	0	8	0	5	11	7

FNONCT 関数

F 分布の非心度パラメーターの値を返します。

カテゴリ: 数学

返されるデータの種類: DOUBLE

構文

FNONCT(*x*, *ndf*, *ddf*, *probability*)

必須引数

x
数値の確率変数です。

範囲 $x \geq 0$

データの種類 DOUBLE

ndf
数値の分子の自由度パラメーターです。

範囲 $ndf > 0$

データの種類 DOUBLE

ddf
数値の分母の自由度パラメーターです。

範囲 $ddf > 0$

データの種類 DOUBLE

probability
確率です。

範囲 $0 < probability < 1$

データの種類 DOUBLE

詳細

FNONCT 関数は、非心度 *F* 分布(パラメーターは *x*、*ndf*、*ddf*、*nc*)から負でない非心度パラメーターを返します。*probability* が心度 *F* 分布(パラメーターは *x*、*ndf* および *ddf*)からの確率よりも大きい場合、この問題の平方根は存在しません。この場合、欠損値が返されます。式の負でない平方根 *nc* を検索するには、ニュートン型のアルゴリズムを使用します。

$$P_f(x|ndf, ddf, nc) - prob = 0$$

前述の式には次の関係が適用されます。

$$P_f(x|ndf, ddf, nc) = \varepsilon^{\frac{-nc}{2}} \sum_{j=0}^{\infty} \frac{\left(\frac{nc}{2}\right)^j}{j!} I_{\frac{(ndf)x}{ddf+(ndf)x}}\left(\frac{ddf}{2}+j, \frac{ddf}{2}\right)$$

この式では、$I(...)$は、次の式によって得られるベータ分布の確率です。

$$I_x(a,b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)} \int_0^x t^{a-1}(1-t)^{b-1} dt$$

アルゴリズムで固定小数点を収束できない場合、欠損値が返されます。

例

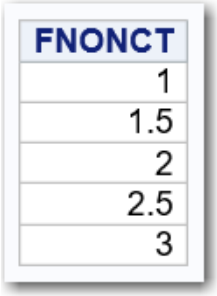
次の例では、*F* 分布から非心度パラメーターを計算します。

```
proc ds2;
data test /overwrite=yes;
  dcl double x df ddf nc prob;
  method init();
    x=2;
    df=4;
    ddf=5;
    do nc=1 to 3 by .5;
      prob=probf(x, df, ddf, nc);
      output;
    end;
  end;
enddata;
run;
quit;

proc print data=test;
run;

proc fedsql;
 select * from test;
 select fnonct(x, df, ddf, prob) from test;
quit;
```

アウトプット 4.9　テーブル Test のコンテンツ

x	df	ddf	nc	prob
2	4	5	1	0.692767
2	4	5	1.5	0.657008
2	4	5	2	0.622321
2	4	5	2.5	0.588776
2	4	5	3	0.556424

アウトプット 4.10　テーブル Test での FNONCT 関数の結果



FNONCT
1
1.5
2
2.5
3

関連項目

関数:

- “CNONCT 関数” (384 ページ)
- “TNONCT 関数” (856 ページ)

FUZZ 関数

引数が整数の 1E-12 以内の場合、最も近い整数を返します。

カテゴリ:　切り捨て

返されるデータの種類:　DOUBLE

構文

FUZZ(*expression*)

必須引数

expression
数値に評価される有効な式を指定します。

データの種類　DOUBLE

参照項目:　“<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

式が整数の 1E-12 以内の場合(整数と引数間の絶対差が 1E-12 よりも小さい場合)、FUZZ 関数は最も近い整数値を返します。それ以外の場合は、その式を返します。

例

次のステートメントは、FUZZ 関数を示しています。

ステートメント	結果
select fuzz(5.99999999);	5.99999999

GAMINV 関数

ガンマ分布から分位点を返します。

カテゴリ: 分位点

返されるデータの種類: DOUBLE

構文

GAMINV(*p*, *a*)

必須引数

p
数値の確率に評価される有効な式を指定します。

範囲 $0 \le p < 1$

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

a
数値の形状パラメーターに評価される有効な式を指定します。

範囲 $a > 0$

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

GAMINV 関数は、ガンマ分布(形状パラメーターは *a*)の *p*分位点を返します。ガンマ分布の行が返される分位点以下になる確率は *p* です。

注: GAMINV は、PROBGAM 関数の逆数です。

例

次のステートメントは、GAMINV 関数を示しています。

ステートメント	結果
select gaminv(0.5, 9);	8.6689511844
select gaminv(0.1, 2.1);	0.5841932369

関連項目

関数:

- “PROBGAM 関数” (769 ページ)

GAMMA 関数

ガンマ関数の値を返します。

カテゴリ: 数学

返されるデータの種類: DOUBLE

構文

GAMMA(*expression*)

必須引数

expression
数値に評価される有効な式を指定します。

制限事項 正でない整数は無効です。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

GAMMA 関数は、次の式で求められる積分を返します。

$$\mathrm{GAMMA}(x) = \int_0^{\infty} t^{x-1} e^{-t}\, dt.$$

正の整数の場合、GAMMA(x)は(x − 1)!になります。この関数は、一般的に$\Gamma(x)$で表されます。

例

次のステートメントは、GAMMA 関数を示しています。

ステートメント	結果
select gamma(6);	120

GARKHCLPRC 関数

Garman-Kohlhagen モデルに基づいて、株式のヨーロピアンオプションのコール価格を計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

GARKHCLPRC(*E*, *t*, *S*, R_d, R_f, *sigma*)

必須引数

E
権利行使価格を指定する正の非欠損値。

要件　*E* および *S* は同じ単位で指定します。

データの種類　DOUBLE

t
満期までの時間を指定する非欠損値です。

データの種類　DOUBLE

S
通貨の現物価格を指定する非欠損値の正の値です。

要件　*S* および *E* は同じ単位で指定します。

データの種類　DOUBLE

R_d
期間 *t* の国内の安全利率を指定する非欠損値の正の分数です。

要件　単位 *t* と同じ期間の R_d の値を指定します。

データの種類　DOUBLE

R_f
期間 *t* の海外の安全利率を指定する非欠損値の正の分数です。

要件　単位 *t* と同じ期間の R_f の値を指定します。

データの種類　DOUBLE

sigma
為替相場のボラティリティを 指定する非欠損値の正の分数です。

要件　単位 *t* と同じ期間の *sigma* の値を指定します。

データの種類　DOUBLE

詳細

GARKHCLPRC 関数は、Garman-Kohlhagen モデルに基づいて、株式のヨーロピアンオプションのコール価格を計算します。この関数は次の関係に基づきます。

$$\text{CALL} = SN(d_1)\left(\varepsilon^{-R_f t}\right) - EN(d_2)\left(\varepsilon^{-R_d t}\right)$$

引数

S
通貨の現物価格を指定します。

N
累積正規密度関数を指定します。

E
オプションの権利行使価格を指定します。

t
有効期限までの時間を指定します。

R_d
期間 *t* の国内の安全利率を指定します。

R_f
期間 *t* の海外の安全利率を指定します。

$$d_1 = \frac{\left(\ln\left(\frac{S}{E}\right) + \left(R_d - R_f + \frac{\sigma^2}{2}\right)t\right)}{\sigma\sqrt{t}}$$

$$d_2 = d_1 - \sigma\sqrt{t}$$

前述の式には次の引数が適用されます。

σ
原資産のボラティリティを指定します。

σ^2
利益率の分散を指定します。

t=0 となる特別な場合には、次の式が真です。

$$\text{CALL} = \max((S - E), 0)$$

価格設定の基本については、*“価格関数の使用” (SAS 関数と CALL ルーチン: リファレンス)*を参照してください。

比較

GARKHCLPRC 関数は、Garman-Kohlhagen モデルに基づいて、株式のヨーロピアンオプションのコール価格を計算します。GARKHPTPRC 関数は、Garman-Kohlhagen モデルに基づいて、株式のヨーロピアンオプションのプット価格を計算します。これらの関数はスカラー値を返します。

例

次のステートメントは、GARKHCLPRC 関数を示しています。

ステートメント	結果
	----+----1----+-—-2--
select garkhclprc(40, .5, 38, .06, .04, .2);	1.449425
select garkhclprc(19, .25, 20, .05, .03, .09);	1.130421

関連項目

関数:

- “GARKHPTPRC 関数” (555 ページ)

GARKHPTPRC 関数

Garman-Kohlhagen モデルに基づいて、株式のヨーロピアンオプションのプット価格を計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

GARKHPTPRC(*E*, *t*, *S*, R_d, R_f, *sigma*)

必須引数

E
権利行使価格を指定する正の非欠損値。

要件 *E* および *S* は同じ単位で指定します。

データの種類 DOUBLE

t
満期までの時間を年単位で指定する非欠損値。

データの種類 DOUBLE

S
通貨の現物価格を指定する非欠損値の正の値です。

要件 *S* および *E* は同じ単位で指定します。

データの種類 DOUBLE

R_d
期間 *t* の国内の安全利率を指定する非欠損値の正の分数です。

要件 単位 *t* と同じ期間の R_d の値を指定します。

データの種類 DOUBLE

R_f
期間 *t* の海外の安全利率を指定する非欠損値の正の分数です。

要件 単位 *t* と同じ期間の R_t の値を指定します。

データの種類 DOUBLE

sigma
為替相場のボラティリティを 指定する非欠損値の正の分数です。

データの種類 DOUBLE

詳細

GARKHPTPRC 関数は、Garman-Kohlhagen モデルに基づいて、株式のヨーロピアンオプションのプット価格を計算します。この関数は次の関係に基づきます。

$$\text{PUT} = \text{CALL} - S\left(\varepsilon^{-R_f t}\right) + E\left(\varepsilon^{-R_d t}\right)$$

引数

S
通貨の現物価格を指定します。

E
オプションの権利行使価格を指定します。

t
有効期限までの時間を年単位で指定します。

R_d
期間 *t* の国内の安全利率を指定します。

R_f
期間 *t* の海外の安全利率を指定します。

$$d_1 = \frac{\left(\ln\left(\frac{S}{E}\right) + \left(R_d - R_f + \frac{\sigma^2}{2}\right)t\right)}{\sigma\sqrt{t}}$$

$$d_2 = d_1 - \sigma\sqrt{t}$$

前述の式には次の引数が適用されます。

σ
原資産のボラティリティを指定します。

σ^2
利益率の分散を指定します。

t=0 となる特別な場合には、次の式が真です。

$$\text{PUT} = \max((E - S), 0)$$

価格設定の基本については、“価格関数の使用” (*SAS 関数と CALL ルーチン: リファレンス*)を参照してください。

比較

GARKHPTPRC 関数は、Garman-Kohlhagen モデルに基づいて、株式のヨーロピアンオプションのプット価格を計算します。GARKHCLPRC 関数は、Garman-Kohlhagen モデルに基づいて、株式のヨーロピアンオプションのコール価格を計算します。これらの関数はスカラー値を返します。

例

次のステートメントは、GARKHPTPRC 関数を示しています。

ステートメント	結果
	----+----1----+----2--
select garkhptprc(50, .7, 55, .05, .04, .2);	1.405088
select garkhptprc(32, .3, 33, .05, .03, .3);	1.564732

関連項目

関数:

- “GARKHCLPRC 関数” (552 ページ)

GCD 関数

整数の集合に対する最大公約数を返します。

カテゴリ: 数学

返されるデータの種類: DOUBLE

構文

GCD(*expression-1*, *expression-2* [, ...*expression-n*])

必須引数

expression
数値に評価される有効な式を指定します。

要件 少なくとも 2 つの引数が必要です。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

GCD (最大公約数)関数は、1 つ以上の整数の最大公約数を返します。たとえば、30 と 42 の最大公約数は 6 になります。最大公約数は、最高共通因子とも呼ばれます。

例

次のステートメントは、GCD 関数を示しています。

ステートメント	結果
select gcd(5,15);	5
select gcd(36,45);	9

関連項目

関数:

- “LCM 関数” (632 ページ)

GEOMEAN 関数

幾何平均を返します。

カテゴリ: 記述統計

返されるデータの種類: DOUBLE

構文

GEOMEAN(*expression* [, ...*expression*])

必須引数

expression
負でない数値に評価される有効な式です。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

いずれかの引数が負の場合、結果は null または欠損値になります。負の引数が無効であること示すメッセージがログに表示されます。いずれかの引数がゼロの場合、幾何平均はゼロになります。すべての引数が null または欠損値の場合、結果は null または欠損値になります。それ以外の場合、結果は非 null または非欠損値の幾何平均になります。

nを非 null または非欠損値を持つ引数の数とし、$x_1, x_2, \ldots, x_n$をそれらの引数の値とします。幾何平均は、次の値の積のn^{th}の平方根です。

$$\sqrt[n]{(x_1 * x_2 * \ldots * x_n)}$$

同様に、幾何平均はこの式で示されます。

$$\exp\left(\frac{(\log(x_1) + \log(x_2) + \ldots + \log(x_n))}{n}\right)$$

多くの場合、浮動小数点の計算では僅かな数値誤差が生じます。正確に計算すればゼロになる計算でも、浮動小数点の計算が使用されると僅かな非ゼロ値になる場合があります。そのため、GEOMEAN はほぼゼロの引数の値をファジー処理します。ある引数の値が最大の引数と比較して極端に小さい場合、前者の引数はゼロとして

扱われます。極端に小さい値を SAS でファジー処理しない場合、GEOMEANZ 関数を使用します。

比較

MEAN 関数は算術平均(平均)、HARMEAN 関数は調和平均、GEOMEAN 関数は非 null または非欠損値の幾何平均を返します。GEOMEANZ や GEOMEAN とは異なり、ほぼゼロの引数の値をファジー処理します。

例

次のステートメントは、GEOMEAN 関数を示しています。

ステートメント	結果
select geomean(1,2,2,4);	2
select geomean(.,2,4,8);	4

関連項目

関数:

- “GEOMEANZ 関数” (560 ページ)
- “HARMEAN 関数” (562 ページ)
- “HARMEANZ 関数” (564 ページ)
- “MEAN 関数” (666 ページ)

GEOMEANZ 関数

ゼロファジーを使用して、幾何平均を返します。

カテゴリ:	記述統計
返されるデータの種類:	DOUBLE

構文

GEOMEANZ(*expression* [, ...*expression*])

必須引数

expression
負でない数値に評価される有効な式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

いずれかの引数が負の場合、結果は null または欠損値になります。負の引数が無効であること示すメッセージがログに表示されます。いずれかの引数がゼロの場合、幾何平均はゼロになります。すべての引数が null または欠損値の場合、結果は null または欠損値になります。それ以外の場合、結果は非 null または非欠損値の幾何平均になります。

nを非 null または非欠損値を持つ引数の数とし、$x_1, x_2, \ldots, x_n$をそれらの引数の値とします。幾何平均は、次の値の積のn^{th}の平方根です。

$$\sqrt[n]{(x_1 * x_2 * \ldots * x_n)}$$

同様に、幾何平均はこの式で示されます。

$$\exp\left(\frac{(\log(x_1) + \log(x_2) + \ldots + \log(x_n))}{n}\right)$$

比較

MEAN 関数は算術平均(平均)、HARMEAN 関数は調和平均、GEOMEANZ 関数は非 null または非欠損値の幾何平均を返します。GEOMEAN や GEOMEANZ とは異なり、ほぼゼロの引数の値をファジー処理しません。

例

次のステートメントは、GEOMEANZ 関数を示しています。

ステートメント	結果
select geomeanz(1,2,2,4);	2
select geomeanz(.,2,4,8);	4

関連項目

関数:

- “GEOMEAN 関数” (559 ページ)
- “HARMEAN 関数” (562 ページ)
- “HARMEANZ 関数” (564 ページ)
- “MEAN 関数” (666 ページ)

HARMEAN 関数

調和平均を返します。

カテゴリ: 記述統計

返されるデータの種類: DOUBLE

構文

HARMEAN(*expression* [, ...*expression*])

必須引数

expression
負でない数値に評価される有効な式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

いずれかの引数が負の場合、結果は null または欠損値になります。負の引数が無効であること示すメッセージがログに表示されます。すべての引数が null または欠損値の場合、結果は null または欠損値になります。それ以外の場合、結果は非 null または非欠損値の調和平均になります。

いずれかの引数がゼロの場合、調和平均はゼロになります。それ以外の場合、調和平均は値の逆数の算術平均の逆数になります。

nを非 null または非欠損値を持つ引数の数とし、$x_1, x_2, \ldots, x_n$をそれらの引数の値とします。調和平均はこの式で示されます。

$$\frac{n}{\frac{1}{x_1} + \frac{1}{x_2} + \ldots + \frac{1}{x_n}}$$

多くの場合、浮動小数点の計算では僅かな数値誤差が生じます。正確に計算すればゼロになる計算でも、浮動小数点の計算が使用されると僅かな非ゼロ値になる場合があります。そのため、HARMEAN はほぼゼロの引数の値をファジー処理します。ある引数の値が最大の引数と比較して極端に小さい場合、前者の引数はゼロとして扱われます。極端に小さい値を SAS でファジー処理しない場合、HARMEANZ 関数を使用します。

比較

MEAN 関数は算術平均(平均)、GEOMEAN 関数は幾何平均、HARMEAN 関数は非 null または非欠損値の調和平均を返します。HARMEANZ や HARMEAN とは異なり、ほぼゼロの引数の値をファジー処理します。

例

次のステートメントは、HARMEAN 関数を示しています。

ステートメント	結果
select harmean(1,2,4,4);	2
select harmean(.,4,12,24);	8

関連項目

関数:

- “GEOMEAN 関数” (559 ページ)
- “GEOMEANZ 関数” (560 ページ)

- “HARMEANZ 関数” (564 ページ)
- “MEAN 関数” (666 ページ)

HARMEANZ 関数

ゼロファジーを使用して、調和平均を返します。

カテゴリ: 記述統計

返されるデータの種類: DOUBLE

構文

HARMEANZ(*expression* [, ...*expression*])

必須引数

expression
負でない数値に評価される有効な式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

いずれかの引数が負の場合、結果は null または値になります。負の引数が無効であること示すメッセージがログに表示されます。すべての引数が null または値の場合、結果は null または値になります。それ以外の場合、結果は非 null または非欠損値の調和平均になります。

いずれかの引数がゼロの場合、調和平均はゼロになります。それ以外の場合、調和平均は値の逆数の算術平均の逆数になります。

nを非 null または非欠損値を持つ引数の数とし、$x_1, x_2, \ldots, x_n$をそれらの引数の値とします。調和平均はこの式で示されます。

$$\frac{n}{\frac{1}{x_1} + \frac{1}{x_2} + \ldots + \frac{1}{x_n}}$$

比較

MEAN 関数は算術平均(平均)、GEOMEAN 関数は幾何平均、HARMEANZ 関数は非 null または非欠損値の調和平均を返します。HARMEAN や HARMEANZ とは異なり、ほぼゼロの引数の値をファジー処理しません。

例

次のステートメントは、HARMEANZ 関数を示しています。

ステートメント	結果
select harmeanz(1,2,4,4);	2
select harmeanz(.,4,12,24);	8

関連項目

関数:

- “GEOMEAN 関数” (559 ページ)
- “GEOMEANZ 関数” (560 ページ)
- “HARMEAN 関数” (562 ページ)
- “MEAN 関数” (666 ページ)

HMS 関数

SAS 時間値(時、分および秒)を返します。

カテゴリ: 日付と時間

返されるデータの種類: DOUBLE

構文

HMS(*hour*, *minute*, *second*)

必須引数

hour
数値式。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

minute
数値式。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

second
数値式。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

HMS 関数は、SAS 時間値を表す数値を返します。SAS 時間値は、現在日の午前 0 時からの秒数を表す数値です。

例

次のステートメントは、HMS 関数を示しています。

ステートメント	結果
select hms(12,45,10);	45910
select put(hms(12,45,10), time.);	12:45:10

関連項目

概念:

- “FedSQL の日付と時間” (59 ページ)

関数:

- “DHMS 関数” (451 ページ)
- “HOUR 関数” (570 ページ)
- “MINUTE 関数” (670 ページ)
- “SECOND 関数” (824 ページ)

HOLIDAY 関数

指定した年の特定の祝日の SAS 日付値を返します。

カテゴリ: 日付と時間

返されるデータの種類: DOUBLE

構文

HOLIDAY('*holiday*', *year*)

必須引数

'*holiday*'
次のいずれかの値を指定する文字定数、変数または式です。BOXING、CANADA、CANADAOBSERVED、CHRISTMAS、COLUMBUS、EASTER、FATHERS、HALLOWEEN、LABOR、MLK、MEMORIAL、MOTHERS、NEWYEAR、THANKSGIVING、THANKSGIVINGCANADA、USINDEPENDENCE、USPRESIDENTS、VALENTINES、VETERANS、VETERANSUSG、VETERANSUSPS、VICTORIA。*holiday* の値には、大文字または小文字を使用できます。詳細については、*SAS DS2 言語リファレンス*を参照してください。

次の表に記載されているいずれかの値を指定する文字定数、変数または式です。

holiday の値には、大文字または小文字を使用できます。

表 4.1 祝日の値とその説明

祝日の値	説明	祝日の日付
BOXING	ボクシングデー	12 月 26 日

祝日の値	説明	祝日の日付
CANADA	カナダ独立記念日	7 月 1 日
CANADAOBSERVED	カナダ独立記念日振替休日	7 月 1 日または 7 月 2 (7/1 が日曜日の場合)
CHRISTMAS	クリスマス	12 月 25 日
COLUMBUS	コロンブス記念日	10 月の第 2 月曜日
EASTER	復活の主日	毎年変わる
FATHERS	父の日	6 月の第 3 日曜日
HALLOWEEN	ハロウィーン	10 月 31 日
LABOR	労働祭	9 月の第 1 月曜日
MLK	マーティンルーサーキング牧師の誕生日	1986 以降の 1 月の第 3 月曜日
MEMORIAL	メモリアルデー	5 月の最終月曜日(1971 以降)
MOTHERS	母の日	5 月の第 2 日曜日
NEWYEAR	元日	1 月 1 日
THANKSGIVING	U.S. 感謝祭	11 月の第 4 木曜日
THANKSGIVINGCANADA	カナダ感謝祭	10 月の第 2 月曜日
USINDEPENDENCE	U.S. 独立記念日	7 月 4 日
USPRESIDENTS	大統領記念日の祝日	2 月の第 3 月曜日 (1971 以降)
VALENTINES	バレンタインデー	2 月 14 日
VETERANS	退役軍人の日	11 月 11 日
VETERANSUSG	退役軍人の日(米国政府の祝日)	月曜日から金曜日までにスケジュールされる米国政府の祝日
VETERANSUSPS	退役軍人の日(米国郵政公社の祝日)	月曜日から日曜日までにスケジュールされる

祝日の値	説明	祝日の日付
		米国政府の祝日(米国郵政公社)
VICTORIA	ビクトリアデー	5 月 24 日以前の直近の月曜日

データの種類 CHAR

year
4 桁の年を指定する数値定数、変数または式です。2 桁の年を使用する場合、YEARCUTOFF=システムオプションを指定する必要があります。

データの種類 DOUBLE

詳細

HOLIDAY 関数は、指定した年に発生する特定の祝日の日付を計算します。米国およびカナダの特定の一般的な祝日のみがこの関数で使用できるように定義されています。(有効な祝日のリストについては、*"祝日の値とその説明" (SAS 関数と CALL ルーチン: リファレンス)*を参照)。

多くの祝日の定義は時代とともに変わります。米国では、1971 年 2 月 11 日に出された大統領命令 11582 で、多くの米国連邦休日の遵守を定着させました。

多くの休日はそれらが設立された日付が確立していますが、現在の祝日の定義は過去および将来に無限に拡張されます。一貫した定義がこれまでにされていない休日もあります。

HOLIDAY 関数は、SAS 日付値を返します。SAS 日付値をカレンダー日付に変換するには、DATE9.形式などの有効な SAS 日付形式を使用します。

比較

HOLIDAY 関数と NWKDOM 関数が同じ結果を返す場合があります。たとえば、ステートメント holiday('thanksgiving', 2012);は、nwkdom(4, 5, 11, 2012);.と同じ値を返します。

また、HOLIDAY 関数と MDY 関数が同じ結果を返す場合もあります。たとえば、ステートメント holiday('christmas', 2012);は、mdy(12, 25, 2012);.と同じ値を返します。

例

次のステートメントは、HOLIDAY 関数を示しています。

ステートメント	結果
select put(holiday('thanksgiving', 2013), date.);	28NOV2013
select put(holiday('boxing', 2013), date.);	26DEC2013
select put(holiday('easter', 2013), date.);	31MAR2013
select put(holiday('canada', 2013), date.);	01JUL2013
select put(holiday('fathers', 2013), date.);	16JUN2013
select put(holiday('valentines', 2013), date.);	14FEB2013
select put(holiday('victoria', 2013), date.);	20MAY2013

関連項目

関数:

- “MDY 関数” (664 ページ)

HOUR 関数

時間または日時値から時間を返します。

カテゴリ: 日付と時間

返されるデータの種類: TINYINT

構文

HOUR(*time* | *datetime*)

オプション引数

time
時間値を表す有効な式を指定します。

データの種類 TIME

参照項目: “FedSQL 式” (46 ページ)

datetime
日時値を表す有効な式を指定します。

データの種類 TIMESTAMP

参照項目: “FedSQL 式” (46 ページ)

例

テーブル: CUSTONLINE (1259 ページ)

次のステートメントは、HOUR 関数を示しています。

ステートメント	結果
select hour(endtime) from custonline;	10 22 19 12 12 16 15 11 10 9
select hour(current_time);	14

関連項目

- “FedSQL の日付と時間” (59 ページ)

関数:

- “DAY 関数” (443 ページ)
- “MINUTE 関数” (670 ページ)
- “MONTH 関数” (676 ページ)
- “SECOND 関数” (824 ページ)
- “YEAR 関数” (887 ページ)

IBESSEL 関数

変形ベッセル関数の値を返します。

カテゴリ: 数学

返されるデータの種類: DOUBLE

構文

IBESSEL(*nu*, *x*, *kode*)

必須引数

nu
数値の定数、変数または式を指定します。

範囲 *nu* ≥ 0

データの種類 DOUBLE

x
数値の定数、変数または式を指定します。

範囲 *x* ≥ 0

データの種類 DOUBLE

kode
負でない整数を指定する数値定数、変数または式です。

データの種類 DOUBLE

詳細

IBESSEL 関数は、*x* で評価される順序 *nu* の変形ベッセル関数(Abramowitz, Stegun 1964; Amos, Daniel, Weston 1977)の値を返します。*kode* が 0 の場合、ベッセル関数が返されます。それ以外の場合、次の関数の値が返されます。

$$\varepsilon^{-x} I_{nu}(x)$$

例

次のステートメントは、IBESSEL 関数を示しています。

ステートメント	結果
select ibessel(2, 2, 0);	0.688948
select ibessel(2, 2, 1);	0.093239

関連項目

関数:

- “JBESSEL 関数” (625 ページ)

IFNULL 関数

最初の式の値をチェックし、null または SAS 欠損値の場合は、2 番目の式を返します。

カテゴリ: 特殊

返されるデータの種類: コンテキストに応じた任意のデータ型

構文

IFNULL(*expression-1*, *expression-2*)

必須引数

expression
任意の有効な式を指定します。

データの種類 すべてのデータ型が有効です。

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

IFNULL 関数は、null でない場合、最初の式を返します。IFNULL 関数を使用して、null または SAS 欠損値を別の値に置き換えることができます。

比較

NULLIF 式は、2 つの式を比較し、その 2 つの式が等しい場合は null 値を返します。IFNULL 関数は、最初の式の値をチェックし、null の場合は 2 番目の式を返し、それ以外の場合は最初の式を返します。

例

テーブル: WORLDCITYCOORDS (1276 ページ)

テーブル WORLDTEMPS では、最後の行の中国の都市に欠損値が含まれています。

ステートメント	結果
select ifnull(AvgHigh, 0) as "AvgHigh: 0 indicates null/missing" from worldtemps;	AvgHigh: 0 indicates null/missing 90 70 86 90 97 83 76 89 90 89 0 78
select ifnull(0,2);	0
select ifnull(1,2);	1
select ifnull(null, 2);	2

関連項目

式:

- “NULLIF 式” (917 ページ)

INDEX 関数

文字式から文字列を検索し、最初に検索された文字列の最初の文字の位置を返します。

カテゴリ:	文字
返されるデータの種類:	INTEGER

構文

INDEX(*target-expression*, *search-expression*)

必須引数

target-expression
文字列に評価されるか強制変換できる有効な式を指定します。

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

search-expression
文字列を評価するか文字列に強制変換できて、*target-expression* での検索に使用される有効な式を指定します。

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

ヒント 文字のリテラル文字列は単一引用符で囲みます。

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

INDEX 関数は、*target-expression* を左から右へ、*search-expression* で指定される文字列の最初の出現箇所を検索し、*target-expression* 内での文字列の最初の文字の位置を返します。*target-expression* 内に文字列が見つからない場合、INDEX は値 0 を返します。文字列が複数回出現する場合、INDEX は、最初の出現箇所の位置のみを返します。

注: *target-expression* または *search-expression* のいずれかの長さが 0 の場合、INDEX 関数は値 0 を返します。宣言されていない VARCHAR 変数の長さは 0 であることにも注意してください。

比較

VERIFY 関数は、*search-expression* 式を含まない *target-expression* の最初の文字の位置を返し、INDEX 関数は、*target-expression* に存在する *search-expression* の最初の出現位置を返します。

例

次のステートメントは、INDEX ステートメントを示しています。

ステートメント	結果
select index('aabb','ab');	2

関連項目

関数:

- “INDEXC 関数” (576 ページ)
- “VERIFY 関数” (878 ページ)

INDEXC 関数

文字式から指定文字を検索し、そのいずれかの文字の最初の出現位置を返します。

カテゴリ: 文字

返されるデータの種類: DOUBLE

構文

INDEXC(*target-expression*, *search-expression*[, ...*search-expression*])

必須引数

target-expression
文字列を評価するか、文字列に強制変換できる検索先の有効な式を指定します。

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

search-expression
target-expression で検索する文字を指定します。

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

ヒント　文字のリテラル文字列は単一引用符で囲みます。

参照項目:　“<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

INDEXC 関数は、検索式に存在する *target-expression* のいずれかの文字の最初の出現箇所を検索し、その文字の *target-expression* 内の位置を返します。検索は左から右に実行されます。*target-expression* で検索式内の文字が見つからない場合、INDEXC は値 0 を返します。

比較

INDEXC 関数が、検索式に含まれる個々の文字が最初に出現する箇所を検索するのに対し、INDEX 関数は、検索式がパターンとして最初に出現する箇所を検索します。

例

次のステートメントは、INDEXC 関数を示しています。

ステートメント	結果
select indexc('aabb','ab');	1

関連項目

関数:

- “INDEX 関数” (574 ページ)

INPUTC 関数

実行時に文字の入力形式を指定できるようにします。

カテゴリ:　特殊

返されるデータの種類:　CHAR、NCHAR、NVARCHAR、VARCHAR

参照項目: “FedSQL での文字型を返す関数の使用” (260 ページ)

構文

INPUTC(*source*, *informat*[, *w*])

必須引数

source
入力形式を適用する文字定数、変数または式を指定します。

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

informat
source に適用する文字の入力形式が含まれる文字定数、変数または式です。

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

オプション引数

w
入力形式に適用する数値幅に評価される有効な式を指定します。

操作 ここで指定した幅は、入力形式での幅の指定より優先されます。

データの種類 DOUBLE

詳細

INPUTC 関数が長さの割り当てられていない変数に値を返した場合、デフォルトでは変数の長さは最初の引数の長さによって決定されます。

比較

INPUTN 関数は、実行時に数値の入力形式を指定できるようにします。

例

次のステートメントは、INPUTC 関数を示しています。

ステートメント	結果
select inputc('positive', '$upcase15.');	POSITIVE

ステートメント	結果
select inputc('positive', '$upcase15.', 3);	POS

関連項目

関数:

- “INPUTN 関数” (579 ページ)
- “PUT 関数” (788 ページ)

INPUTN 関数

実行時に数値の入力形式を指定できるようにします。

カテゴリ: 特殊

返されるデータの種類: DOUBLE

構文

INPUTN(*source*, *informat*[, *w*[, *d*]])

必須引数

source
入力形式を適用する文字定数、変数または式を指定します。

データの種類 CHAR

informat
source に適用する数値の入力形式が含まれる文字定数、変数または式です。

データの種類 CHAR

オプション引数

w
入力形式に適用する幅を指定する数値定数、変数または式です。

操作 ここで指定した幅は、入力形式での幅の指定より優先されます。

データの種類 DOUBLE

d
使用する小数点以下の桁数を指定する数値定数、変数または式です。

操作　ここで指定した桁数は、入力形式での小数点以下の桁数の指定より優先されます。

データの種類　DOUBLE

比較

INPUTC 関数は、実行時に文字の入力形式を指定できるようにします。コンパイル時に入力形式を指定するため、PUT 関数を使用した方が高速になります。

例

次のステートメントは、INPUTN 関数を示しています。

ステートメント	結果
select inputn('20,000.00', 'comma10.2');	20000

関連項目

関数:

- “INPUTC 関数” (577 ページ)
- “PUT 関数” (788 ページ)

INT 関数

予期しない浮動小数点の結果を避けるためにファジー処理された整数を返します。

カテゴリ:　切り捨て

返されるデータの種類:　DOUBLE

構文

INT(*expression*)

必須引数

expression
数値に評価される任意の式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

FedSQL では、INT は予約語です。関数を使用するには、関数名を区切り ID として指定する必要があります。

INT 関数は、引数の整数部(小数点以下は切り捨て)を返します。引数の値が 1E-12 内の整数である場合、関数はその整数を返します。*expression* が正の値の場合、INT 関数の結果は FLOOR 関数の結果と同じです。*expression* が負の値の場合、INT 関数の結果は CEIL 関数の結果と同じです。

比較

INTZ 関数とは異なり、INT 関数は結果をファジー処理します。引数が整数の 1E-12 内にある場合、INT 関数はその整数に等しくなるように結果をファジー処理します。INTZ 関数は結果をファジー処理しません。そのため、INTZ 関数では予期しない結果になる可能性があります。

例

次のステートメントは、INT 関数を示しています。

ステートメント	結果
select "int"(2.6);	2
select "int"(-2.4);	-2
select "int"(1+1.3-11);	-8
select "int"(-1.6);	-1

関連項目

関数:

- “CEIL 関数” (380 ページ)
- “FLOOR 関数” (542 ページ)
- “INTZ 関数” (619 ページ)
- “MOD 関数” (671 ページ)
- “MODZ 関数” (673 ページ)
- “ROUND 関数” (805 ページ)

INTCINDEX 関数

サイクルインデックスを返します。この関数には、日付、時間またはタイムスタンプの間隔と値を指定します。

カテゴリ:	日付と時間
制限事項:	FedSQL は、カスタムの日付または時間間隔をサポートしていません。
返されるデータの種類:	DOUBLE

構文

INTCINDEX({*interval*[*multiple*][.*shift-index*]}, *date-time-value*)

必須引数

interval
文字列を評価するか、文字列に強制変換でき、WEEK、MONTH、QTR などの間隔名を含む文字定数、変数、または式を指定します。

データの種類	CHAR、NCHAR、NVARCHAR、VARCHAR
注	*interval* の可能な値は、“日付間隔と時間間隔について” (*SAS 出力形式と入力形式: リファレンス*)にリストされています。
ヒント	*Interval* は、大文字または小文字で表示できます。
例	YEAR は、年ベースの間隔を指定します。

date-time-value
指定した間隔の期間を表す日付、時間またはタイムスタンプの値を指定します。

データの種類 DOUBLE

オプション引数

multiple
乗数を指定します(省略可能)。基本タイプの間隔の期間に対する倍数と同等の間隔を設定します。

データの種類 DOUBLE

参照項目: 詳細については、"乗数とシフト間隔を使用した日時の増分" (*SAS 関数と CALL ルーチン: リファレンス*)を参照してください。

例 YEAR2 は、2 年、つまり 2 年間続く間隔タイプを指定します。

shift-index
シフトインデックスを指定します(省略可能)。間隔の開始時点を指定したサブ期間にシフトします。

制限事項 シフトインデックスは、間隔全体のサブ期間の数以下にする必要があります。たとえば、2 年間隔では 25 番目の月は存在しないため、YEAR2.24 は使用できますが YEAR2.25 はエラーになります。

デフォルトのシフト期間が間隔の種類と同じ場合、複数期間の間隔のみを任意のシフトインデックスでシフトできます。たとえば、MONTH の種類の間隔はデフォルトでは MONTH のサブ期間でシフトされるため、シフトインデックスで月間隔をシフトできません。ただし、各 MONTH2 間隔には 2 つの MONTH 間隔が含まれるため、シフトインデックスで 2 か月間隔をシフトできます。たとえば、間隔名 MONTH2.2 では、偶数月の 1 日目に開始する 2 か月の期間が指定されます。

データの種類 DOUBLE

参照項目: 詳細については、"乗数とシフト間隔を使用した日時の増分" (*SAS 関数と CALL ルーチン: リファレンス*)を参照してください。

例 YEAR.3 は、暦年の 3 月 1 日に開始し、翌年の 2 月に終了するようにシフトされた年次期間を指定します。

詳細

基本

INTCINDEX 関数は、季節サイクルのインデックスを返します。この関数には、間隔と DATE、TIME または TIMESTAMP の値を指定します。たとえば、間隔が MONTH の場合、データ内の各オブザベーションは特定の月に対応します。月単位のデータは、1 年間で周期的とみなされます。1 年には 12 か月あるため、季節サイクル(年)内の間隔(月)数は 12 です。WEEK は、DAY と等しい間隔の季節サイクルです。2013

年 9 月 1 日はその年の 35 番目の週で 6 番目の曜日なので、この例では 36 の値が返されます。

```
select intcindex('day', date'2013-09-01');
```

日付間隔と時間間隔の操作の詳細については、“日付間隔と時間間隔” (*SAS 関数と CALL ルーチン: リファレンス*)を参照してください。

INTCINDEX 関数は、小売業界によるカレンダーの間隔でも使用できます。これらの間隔は ISO 8601 に準拠します。これらの間隔のリストについては、*SAS プログラマガイド: エッセンシャル*の"Retail Calendar Intervals: ISO 8601 Compliant"を参照してください。

間隔

間隔は、基本的なものから複雑なものまであります。基本間隔は、SAS が経過時間内にカウントできる測定単位(DAY、MONTH、HOUR など)です。より複雑な間隔を指定するには、基本間隔名に乗数とシフトインデックスを組み合わせて使用できます。

間隔構文は次のとおりです。

interval[*multiple*][.*shift-index*]

詳細については、“必須引数” (582 ページ)を参照してください。

比較

INTCINDEX 関数はサイクルインデックスを返しますが、INTINDEX 関数は季節インデックスを返します。

この例では、INTCINDEX 関数は年間通算週を返します。

```
select intcindex('day', date'2013-04-04');
```

この例では、INTINDEX 関数は曜日を返します。

```
select intindex('day', date'2013-04-04');
```

この例では、INTCINDEX 関数は時刻を返します。

```
select intcindex('minute', timestamp'2012-09-01 00:00:00');
```

この例では、INTINDEX 関数は分を返します。

```
select intindex('minute', timestamp'2012-09-01 00:00:00');
```

例 intseas(intcycle('interval'));では、INTSEAS 関数は intcindex('interval', date);によって返される最大数を返します。

例

次のステートメントは、INTCINDEX 関数を示しています。

ステートメント	結果
select intcindex('dtqtr', timestamp'2013-05-23 05:03:01');	1
select intcindex('tenday', date'2013-12-13');	1
select intcindex('minute', time'23:13:02');	24
select intcindex('semimonth', timestamp'2013-05-05 10:54:03');	1

関連項目

関数:

- “INTCYCLE 関数” (591 ページ)
- “INTINDEX 関数” (599 ページ)
- “INTSEAS 関数” (611 ページ)

INTCK 関数

DOUBLE としてエンコードされた 2 つの SAS 日付、時間、またはタイムスタンプ値の間にある特定の種類の間隔境界の数を返します。

カテゴリ: 日付と時間

制限事項: FedSQL は、カスタムの日付または時間間隔をサポートしていません。

返されるデータの種類: DOUBLE

構文

INTCK({*interval*[*multiple*][.*shift-index*]}, *start-date*, *end-date*[, '*method*'])

INTCK(*start-date*, *end-date*[, '*method*'])

必須引数

interval
文字列を評価するか、文字列に強制変換でき、WEEK、MONTH、QTR などの間隔名を含む文字定数、変数、または式を指定します。

データの種類: CHAR、NCHAR、NVARCHAR、VARCHAR

注 *interval* の可能な値は、"日付間隔と時間間隔について" (*SAS 出力形式と入力形式: リファレンス*)にリストされています。

ヒント *Interval* は、大文字または小文字で表示できます。

例 YEAR は、年ベースの間隔を指定します。

start-date
SAS を開始する日付、時間、またはタイムスタンプ値を表す式を指定します。

データの種類 DOUBLE

end-date
SAS を終了する日付、時間、またはタイムスタンプ値を表す式を指定します。

データの種類 DOUBLE

オプション引数

multiple
乗数を指定します(省略可能)。基本タイプの間隔の期間に対する倍数と同等の間隔を設定します。

データの種類 DOUBLE

参照項目: 詳細については、"乗数とシフト間隔を使用した日時の増分" (*SAS 関数と CALL ルーチン: リファレンス*)を参照してください。

例 YEAR2 は、2 年、つまり 2 年間続く間隔タイプを指定します。

shift-index
シフトインデックスを指定します(省略可能)。間隔の開始時点を指定したサブ期間にシフトします。

制限事項 シフトインデックスは、間隔全体のサブ期間の数以下にする必要があります。たとえば、2 年間隔では 25 番目の月は存在しないため、YEAR2.24 は使用できますが YEAR2.25 はエラーになります。

デフォルトのシフト期間が間隔の種類と同じ場合、複数期間の間隔のみを任意のシフトインデックスでシフトできます。たとえば、MONTH の種類の間隔はデフォルトでは MONTH のサブ期間でシフトされるため、シフトインデックスで月間隔をシフトできません。ただし、各 MONTH2 間隔には 2 つの MONTH 間隔が含まれるため、シフトインデックスで 2 か月間隔をシフトできます。たとえば、間隔名 MONTH2.2 では、偶数月の 1 日目に開始する 2 か月の期間が指定されます。

データの種類 DOUBLE

参照項目: 詳細については、"乗数とシフト間隔を使用した日時の増分" (*SAS 関数と CALL ルーチン: リファレンス*)を参照してください。

例 YEAR.3 は、暦年の 3 月 1 日に開始し、翌年の 2 月に終了するようにシフトされた年次期間を指定します。

'*method*'
DISCRETE メソッドと CONTINUOUS メソッドのどちらを使用して間隔を数えるかを指定します。*method* は引用符で囲みます。*method* は、CONTINUOUS と DISCRETE (間隔境界をカウント)のいずれかの値になります。

DISCRETE メソッドと CONTINUOUS メソッドのどちらを使用して間隔を数えるかを指定します。

method は引用符で囲む必要があります。*Method* には次のいずれかの値を指定できます。

CONTINUOUS
連続した期間を測定単位に指定します。間隔は開始日に基づいてシフトされます。

たとえば、2013 年 1 月 15 日から 2013 年 2 月 15 日までの月単位の距離は 1 か月です。

別名 C または CONT

DISCRETE
連続しない期間を測定の単位に指定します。DISCRETE メソッドは、間隔の境界(月末など)を数えます。

デフォルトの DISCRETE メソッドは、時系列オブザベーションをビンに分類して処理する場合に役立ちます。たとえば、日単位のデータを月単位の時系列として処理するために月単位のデータに累積できます。

DISCRETE メソッドの場合、2000 年 1 月 31 日から 2000 年 2 月 1 日までの間隔を月単位で示すと、1 か月になります。

別名 D または DISC

デフォルト DISCRETE

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

詳細

間隔

間隔は、基本的なものから複雑なものまであります。基本間隔は、SAS が経過時間内にカウントできる測定単位(DAY、MONTH、HOUR など)です。より複雑な間隔を指定するには、基本間隔名に乗数とシフトインデックスを組み合わせて使用できます。

間隔構文は次のとおりです。

interval[*multiple*][.*shift-index*]

詳細については、“必須引数” (585 ページ)を参照してください。

カレンダー間隔の計算

個々の時間間隔内に含まれる値は、すべて同等とみなします。つまり、月間隔を指定した場合、2013 年 1 月 1 日と 2013 年 1 月 15 日は同等です。これら両方の日付は、2013 年 1 月 1 日に開始して 2013 年 1 月 31 日に終了する間隔を表します。この間隔を示すには、間隔の開始日(2013 年 1 月 1 日)または間隔の終了日(2013 年 1 月 31 日)を使用できます。これらの日付によって、月間隔内のすべての日付が表されます。

次の例では、*start-date* (2013 年 1 月 14 日)は、2013 年の第 1 四半期と同等です。

```
select intck('qtr', date'2013-01-14', date'2013-09-02)';
```

end-date (2013 年 9 月 2 日)は、2013 年の第 3 四半期と同等です。間隔数(*start-date* と *end-date* の間に間隔の開始点が含まれる回数)は 2 です。

デフォルトの DISCRETE メソッドを使用する INTCK 関数は、1 番目の日付と 2 番目の日付の間に次の間隔の開始点が含まれる回数を数えます。2 つの日付間に含まれる間隔数は計算しません。

- 次の例では、2 つの日付が同じ月内にあるため、0 が返されます。

  ```
  select intck('month', date'2013-01-01', date'2013-01-31');
  ```

- 次の例では、2 つの日付が 1 か月離れた異なる月にあるため、1 が返されます。

  ```
  select intck('month', date'2013-01-31', date'2013-02-01');
  ```

- 次の例では、最初の日付が 2 番目の日付よりも遅い離散間隔にあるため、-1 が返されます。(1 番目の日付が 2 番目の日付よりも後で、2 つの日付が同じ間隔内に存在しない場合、INTCK は常に負の値を返します)。

  ```
  select intck('month', date'2013-02-01', date'2013-01-31');
  ```

DISCRETE メソッドを使用する場合、WEEK 間隔は *start-date* と *end-date* の間に 7 日間がいくつ含まれるかではなく、2 つの日付間に存在する日曜日(週のデフォルトの開始曜日)の数で決定されます。*start-date* と *end-date* の間に 7 日間が含まれる数を数えるには、CONTINUOUS メソッドを使用します。

multiple 引数と *shift-index* 引数は両方とも任意で、デフォルトで 1 になっています。たとえば、YEAR、YEAR1、YEAR.1、YEAR1.1 は、すべて通常のカレンダー年を指定します。

日付間隔と時間間隔の操作の詳細については、*“日付間隔と時間間隔” (SAS 関数と CALL ルーチン: リファレンス)*を参照してください。

カテゴリ別の間隔

表 4.2　日付と時間の関数で使用される間隔

カテゴリ	間隔	定義	デフォルトの開始点	シフト期間	例	説明
日付	DAY	日間隔	毎日	日	DAY3	日曜日から始まる 3 日間隔

カテゴリ	間隔	定義	デフォルトの開始点	シフト期間	例	説明
	WEEK	7 日間の週間隔	毎週日曜日	日(1=日曜日 ...7=土曜日)	WEEK.7	土曜日を週の最初の日とする毎週
	WEEKDAY <*days*W>	金土日を含む日間隔	毎日	日	WEEKDAY1W	日曜日を週末日、6 日間を就業日とする週
		同じ日としてカウントされます(土曜日と日曜日を週末、5 日間を就業日とする週)。*days* は、週末日を数値で識別します(1=日曜日 ... 7=土曜日)。デフォルトでは、*days*=17 です。			WEEKDAY35W	火曜日と木曜日を週末日とし、5 日間を就業日とする週(W で 3 日目と 5 日目を週末日に指定)
	TENDAY	10 日間隔(米国自動車産業の慣例)	毎月 1 日、11 日、21 日	10 日間	TENDAY4.2	TENDAY 単位の 2 番目を開始時点とする 40 日単位
	SEMIMONTH	半月間隔	毎月 1 日と 16 日	半月間	SEMIMONTH2.2	月の 16 日目から翌月の 15 日目までの間隔
	MONTH	月間隔	毎月 1 日	月	MONTH2.2	翌年の 2 月-3 月、4 月-5 月、6 月-7 月、8 月-9 月、10 月-11 月、12 月-1 月
	QTR	四半期(3 か月)間隔	1 月 1 日 4 月 1 日	月	QTR3.2	4 月 1 日、7 月 1 日、10 月 1 日、1 月

カテゴリ	間隔	定義	デフォルトの開始点	シフト期間	例	説明
			7 月 1 日 10 月 1 日			1 日から 3 か月間隔
	SEMIYEAR	半年(6 か月)間隔	1 月 1 日 7 月 1 日	月	SEMIYEAR.3	3 月-8 月と 9 月-2 月の 6 か月間隔
	YEAR	年間隔	1 月 1 日	月		
日時	日付間隔のいずれかに DT を追加	関連する日付間隔に対応する間隔	1960 年 1 月 1 日の午前 0 時		DTMONTH DTWEEKDAY	
時間	SECOND	秒間隔	1 日の始まり(午前 0 時)	秒		
	MINUTE	分間隔	1 日の始まり(午前 0 時)	分		
	HOUR	時間隔	1 日の始まり(午前 0 時)	時		

小売カレンダーの間隔

小売業界では、1 年のカレンダーを 13 週間からなる 4 つの期間に分けてデータを計算することがよくあります。期間の形式は、4-4-5、4-5-4 または 5-4-4 のいずれかに基づきます。1 番目、2 番目、3 番目の数値には、それぞれ各期間の 1 か月目、2 か月目、3 か月目の週数を指定します。詳細については、*"ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の処理" (SAS 出力形式と入力形式: リファレンス)* を参照してください。

例

次のステートメントは、INTCK 関数を示しています。

ステートメント	結果
select intck('qtr', date'2013-01-10', date'2013-07-01');	2
select intck('year', date'2012-12-31',	1

ステートメント	結果
date'2013-01-01');	
select intck('year', date'2013-01-01', date'2013-12-31');	0
select intck('semiyear', date'2010-01-01', date'2013-01-01');	6
select intck('week2.2', date'2013-01-07', date'2013-04-01');	6
select intck('weekday7w', date'2013-01-01', date'2013-02-01');	27
select intck('year', date'2003-09-01', date'2013-09-01');	10

2 番目の例では、経過日数は 1 日のみでも INTCK は 1 の値を返します。これは、2012 年 12 月 31 日から 2013 年 1 月 1 日までの間に YEAR 間隔の開始点が含まれるためです。ただし、3 番目の例では、経過日数は 364 日でも 0 の値が返されます。この結果が発生するのは、2013 年 1 月 1 日から 2013 年 12 月 31 日の間に YEAR 間隔の開始点が含まれないためです。

4 番目の例では、2010 年 1 月 1 日から 2013 年 1 月 1 日までの期間に半年の間隔が 6 回含まれるため、SAS は 6 の値を返します。(終了日が 2012 年 12 月 31 日の場合、SAS は 5 回の間隔を数えます)。5 番目の例では、2013 年 1 月 7 日から 2013 年 4 月 1 日の間には第 1 日を月曜日とする 2 週間の間隔が 6 回含まれるため、SAS は 6 の値を返します。6 番目の例では、SAS は値 27 を返します。この例では 2013 年 1 月 1 日から 2013 年 2 月 1 日までの間に土曜日のみを週末として数えるため、この期間には 27 日の平日が含まれることになります。最後の例では、指定された日付から経過した年数(10)を指定しています。

関連項目

その他のリファレンス:

- “FedSQL の日付と時間” (59 ページ)

関数:

- “INTNX 関数” (603 ページ)

INTCYCLE 関数

次に高い季節サイクルでの期間(日付、時間または日時の間隔)を返します。この関数には、期間(日付、時間または日時の間隔)を指定します。

カテゴリ:	日付と時間
制限事項:	FedSQL は、カスタムの日付または時間間隔をサポートしていません。
返されるデータの種類:	CHAR、NCHAR、NVARCHAR、VARCHAR

構文

INTCYCLE({*interval*[*multiple*][.*shift-index*]}[, *seasonality*])

必須引数

interval
文字列を評価するか、文字列に強制変換でき、WEEK、MONTH、QTR などの間隔名を含む文字定数、変数、または式を指定します。

データの種類	CHAR、NCHAR、NVARCHAR、VARCHAR
注	*interval* の可能な値は、"日付間隔と時間間隔について" (*SAS 出力形式と入力形式: リファレンス*)にリストされています。
ヒント	*Interval* は、大文字または小文字で表示できます。
例	YEAR は、年ベースの間隔を指定します。

オプション引数

multiple
乗数を指定します(省略可能)。基本タイプの間隔の期間に対する倍数と同等の間隔を設定します。

データの種類	DOUBLE
参照項目:	詳細については、"乗数とシフト間隔を使用した日時の増分" (*SAS 関数と CALL ルーチン: リファレンス*)を参照してください。
例	YEAR2 は、2 年、つまり 2 年間続く間隔タイプを指定します。

shift-index
シフトインデックスを指定します(省略可能)。間隔の開始時点を指定したサブ期間にシフトします。

制限事項	シフトインデックスは、間隔全体のサブ期間の数以下にする必要があります。たとえば、2 年間隔では 25 番目の月は存在しないため、YEAR2.24 は使用できますが YEAR2.25 はエラーになります。 デフォルトのシフト期間が間隔の種類と同じ場合、複数期間の間隔のみを任意のシフトインデックスでシフトできます。たとえば、MONTH の種類の間隔はデフォルトでは MONTH のサブ期間でシフトされるため、シフトインデックスで月間隔をシフトできません。ただし、各 MONTH2 間隔

には 2 つの MONTH 間隔が含まれるため、シフトインデックスで 2 か月間隔をシフトできます。たとえば、間隔名 MONTH2.2 では、偶数月の 1 日目に開始する 2 か月の期間が指定されます。

データの種類 DOUBLE

参照項目: 詳細については、"乗数とシフト間隔を使用した日時の増分" (*SAS 関数と CALL ルーチン: リファレンス*)を参照してください。

例 YEAR.3 は、暦年の 3 月 1 日に開始し、翌年の 2 月に終了するようにシフトされた年次期間を指定します。

seasonality
数値を指定します。この引数では、日付と時間のサイクルをより柔軟に操作できます。1 年間の季節性を 52 週にするか 53 週にするかを指定できます。

数値を指定します。

この引数では、日付と時間のサイクルをより柔軟に操作できます。1 年間の季節性を 52 週にするか 53 週にするかを指定できます。

デフォルト 52

データの種類 DOUBLE、CHAR、NCHAR、NVARCHAR、VARCHAR

例 次の例では、*seasonality* 引数により
INTCYCLE('MONTH', 3);
関数呼び出しで値 QTR が返されます。この関数呼び出しは
INTCYCLE('MONTH');
seasonality 引数がないため、値 YEAR を返します。

詳細

基本

INTCYCLE 関数は、日付、時間または日時の間隔に応じて、季節サイクルの間隔を返します。たとえば、INTCYCLE('MONTH');では、1 月から 12 月までで 1 年サイクルを構成するため、値 YEAR を返します。INTCYCLE('DAY');では、日曜日から土曜日までで 1 週間サイクルを構成するため、値 WEEK を返します。

乗数とシフトインデックスの詳細については、"乗数とシフト間隔を使用した日時の増分" (*SAS 関数と CALL ルーチン: リファレンス*)を参照してください。間隔の計算方法については、"よく使用される時間間隔" (*SAS 関数と CALL ルーチン: リファレンス*)を参照してください。

日付間隔と時間間隔の操作の詳細については、"日付間隔と時間間隔" (*SAS 関数と CALL ルーチン: リファレンス*)を参照してください。

INTCYCLE 関数は、小売業界によるカレンダーの間隔でも使用できます。これらの間隔は ISO 8601 に準拠します。詳細については、"小売カレンダーの間隔: ISO 8601 準拠" (*SAS 関数と CALL ルーチン: リファレンス*)を参照してください。

間隔

間隔は、基本的なものから複雑なものまであります。基本間隔は、SAS が経過時間内にカウントできる測定単位(DAY、MONTH、HOUR など)です。より複雑な間隔を指定するには、基本間隔名に乗数とシフトインデックスを組み合わせて使用できます。

間隔構文は次のとおりです。

interval[*multiple*][.*shift-index*]

詳細については、“必須引数” (592 ページ)を参照してください。

季節性

季節性は、年内の異なる間隔での周期的変動を測定する時系列の概念です。季節性の指定では、季節が最も一般的な変動ソースです。たとえば、家庭暖房用の灯油は、一般的に他の季節よりも冬の売り上げの方が高くなります。多くの場合、日単位の時系列では曜日による定期的な変動(週末のレジャーで支出が増えるなど)が発生します。INTCYCLE 関数は季節性の概念を使用して、次に高い季節サイクルでの期間(日付、時間または日時の間隔)を返します。この関数には、期間(日付、時間または日時の間隔)を指定します。

例

次のステートメントは、INTCYCLE 関数を示しています。

ステートメント	結果
select intcycle('year');	YEAR
select intcycle('qtr');	YEAR
select intcycle('month', 3);	QTR
select intcycle('month');	YEAR
select intcycle('weekday');	WEEK
select intcycle('weekday', 5);	WEEK
select intcycle('day');	WEEK
select intcycle('day', 10);	TENDAY
intcycle('day', 10);	DTMINUTE

INTFIT 関数

2 つの日付に基づく時間間隔を返します。

カテゴリ: 日付と時間

制限事項: FedSQL は、カスタムの日付または時間間隔をサポートしていません。

返されるデータの種類: CHAR、NCHAR、NVARCHAR、VARCHAR

構文

INTFIT(*expresssion-1*, *expression-2*, '*type*')

必須引数

expression
文字列を評価するか、文字列に強制変換できて、SAS 日付または日時値を表す有効な式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

'*type*'
引数が SAS 日付値、日時値、行のいずれであるかを指定します。*type* の有効な値は、d (日付)、dt (日時)、および obs (行)です。

引数が SAS 日付値、日時値、行のいずれであるかを指定します。

type の有効な値は次のとおりです。

d
expression-1 と *expression-2* が日付値であることを指定します。

dt
expression-1 と *expression-2* が日時値であることを指定します。

obs
expression-1 と *expression-2* が行であることを指定します。

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

詳細

INTFIT 関数は、間隔内に配置された 2 つの日付値、日時値または行に基づいて、間隔内の最も可能性の高い時間間隔を返します。INTFIT では配置の値は SAME とみなされ、対応する間隔を増分したカレンダー日付と同じ日付を使用します。*alignment* 引数の詳細については、“INTNX 関数” (*SAS 関数と CALL ルーチン: リファレンス*)を参照してください。

INTFIT で行の引数を使用する場合、行番号を使用してそのサイクルを確認できます。次の例では、INTFIT の最初の 2 つの引数は行番号で、*type* 引数は obs です。調査員がデータを記録した 1 回目と 25 回目に Jason がジムを使用した場合、次のステートメントを使用して間隔を確認できます。interval=intfit(1, 25, 'obs');.この場合、間隔の値は 24.2 です。

INTFIT 関数は、小売業界によるカレンダーの間隔でも使用できます。これらの間隔は ISO 8601 に準拠します。詳細については、“ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の処理” (*SAS 出力形式と入力形式: リファレンス*)を参照してください。

例

次のステートメントは、INTFIT 関数を示しています。この例は、2 つの日付に基づく間隔を示しています。

ステートメント	結果
select intfit(date'2013-08-01', date'2013-09-01', 'd');	month

関連項目

関数:

- “INTCK 関数” (585 ページ)
- “INTNX 関数” (603 ページ)

INTGET 関数

3 つの日付値または日時値に基づく時間間隔を返します。

カテゴリ: 日付と時間

制限事項: FedSQL は、カスタムの日付または時間間隔をサポートしていません。

返されるデータの種類: CHAR、NCHAR、NVARCHAR、VARCHAR

構文

INTGET(*date-1*, *date-2*, *date-3*)

必須引数

date
SAS 日付または日時の値に評価される有効な式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

INTGET 関数の間隔

INTGET 関数は、3 つの日付値または日時値に基づく時間間隔を返します。この関数は最初の 2 つの日付間で可能なすべての間隔を最初に確認し、次に 2 番目と 3 番目の日付間で可能なすべての間隔を確認します。すべての間隔が同じ場合、INTGET はその間隔を返します。1 番目と 2 番目の日付間に異なる間隔があり、2 番目と 3 番目の日付間に異なる間隔がある場合、INTGET はそれらの間隔を比較します。他の間隔の倍数となる間隔がある場合、INTGET はその小さい方の間隔を返します。それ以外の場合、INTGET は欠損値を返します。INTGET は、配置の値が BEGIN の INTNX 関数によって生成された日付に適しています。

次の例では、INTGET は間隔 DAY2 を返します。

```
select intget(date'2000-03-01', date'2000-03-03', date'2000-03-09');
```

2000 年 3 月 1 日から 2000 年 3 月 3 日までの日数は 2 であるため、1 番目と 2 番目の日付間の間隔は DAY2 です。2000 年 3 月 3 日から 2000 年 3 月 9 日までの日数は 6 であるため、2 番目と 3 番目の日付間の間隔は DAY6 です。DAY6 は DAY2 の倍数です。INTGET はこの 2 つの間隔のうち小さい方を返します。

次の例では、INTGET は間隔 MONTH4 を返します。

```
select intget(date'2000-01-01', date'2000-05-01', date'2001-05-01');
```

2000 年 1 月 1 日から 2000 年 5 月 1 日までの月数は 4 であるため、最初の 2 つの日付間の間隔は MONTH4 です。2 番目と 3 番目の日付間の間隔は YEAR です。YEAR は MONTH4 の倍数(YEAR には 3 つの MONTH4 間隔が含まれる)であるため、INTGET は 2 つの間隔のうち小さい方の間隔を返します。

次の例では、INTGET は欠損値を返します。

```
select intget(date'2006-01-01', date'2006-04-01', date'2006-12-01');
```

最初の 2 つの日付間の間隔は MONTH3、2 番目と 3 番目の日付間の間隔は MONTH8 です。MONTH8 は MONTH3 の倍数ではないため、INTGET は欠損値を返します。

返される間隔は有効な SAS 間隔で、間隔とシフト間隔の倍数が含まれます。有効な SAS 間隔は、"日付間隔と時間間隔について" (*SAS 出力形式と入力形式: リファレンス*)にリストされています。

注: INTGET が一致する間隔を確認できない場合、関数は欠損値を返します。SAS ログにメッセージは書き込まれません。

小売カレンダーの間隔

INTGET 関数は、小売業界によるカレンダーの間隔でも使用できます。これらの間隔は ISO 8601 に準拠します。詳細については、"ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の処理" (*SAS 出力形式と入力形式: リファレンス*)を参照してください。

例

次のステートメントは、INTGET 関数を示しています。

ステートメント	結果
select intget(date'2013-01-01', date'2014-01-01', date'2014-05-01');	MONTH4
select intget(date'2012-02-29', date'2014-02-28', date'2016-02-29');	YEAR2.2
select intget(date'2013-02-01', date'2013-02-16', date'2013-03-01');	SEMIMONTH
select intget(date'2013-01-02', date'2014-02-02', date'2015-03-02');	MONTH13.13
select intget(date'2013-02-10', date'2013-02-19', date'2013-02-28');	DAY9.5
select intget(timestamp'2014-04-01 01:03:00.0000', timestamp'2014-04-01 01:04:00.0000', timestamp'2014-04-01 01:05:00.0000');	MINUTE

関連項目

関数:

- “INTFIT 関数” (595 ページ)
- “INTNX 関数” (603 ページ)

INTINDEX 関数

季節インデックスを返します。この関数には、日付、時間またはタイムスタンプの間隔と値を指定します。

カテゴリ:	日付と時間
制限事項:	FedSQL は、カスタムの日付または時間間隔をサポートしていません。
返されるデータの種類:	DOUBLE

構文

INTINDEX({*interval*[*multiple*][.*shift-index*]}, *date-value*[, *seasonality*])

必須引数

interval
文字列を評価するか、文字列に強制変換でき、WEEK、MONTH、QTR などの間隔名を含む文字定数、変数、または式を指定します。

データの種類	CHAR、NCHAR、NVARCHAR、VARCHAR
注	*interval* の可能な値は、“日付間隔と時間間隔について” (*SAS 出力形式と入力形式: リファレンス*)にリストされています。
ヒント	*Interval* は、大文字または小文字で表示できます。
例	YEAR は、年ベースの間隔を指定します。

date-value
指定した間隔の期間を表す日付、時間またはタイムスタンプの値を指定します。

データの種類	DOUBLE

オプション引数

multiple
乗数を指定します(省略可能)。基本タイプの間隔の期間に対する倍数と同等の間隔を設定します。

データの種類	DOUBLE

参照項目: 詳細については、“乗数とシフト間隔を使用した日時の増分” (*SAS 関数と CALL ルーチン: リファレンス*)を参照してください。

例 YEAR2 は、2 年、つまり 2 年間続く間隔タイプを指定します。

shift-index
シフトインデックスを指定します(省略可能)。間隔の開始時点を指定したサブ期間にシフトします。

制限事項 シフトインデックスは、間隔全体のサブ期間の数以下にする必要があります。たとえば、2 年間隔では 25 番目の月は存在しないため、YEAR2.24 は使用できますが YEAR2.25 はエラーになります。

デフォルトのシフト期間が間隔の種類と同じ場合、複数期間の間隔のみを任意のシフトインデックスでシフトできます。たとえば、MONTH の種類の間隔はデフォルトでは MONTH のサブ期間でシフトされるため、シフトインデックスで月間隔をシフトできません。ただし、各 MONTH2 間隔には 2 つの MONTH 間隔が含まれるため、シフトインデックスで 2 か月間隔をシフトできます。たとえば、間隔名 MONTH2.2 では、偶数月の 1 日目に開始する 2 か月の期間が指定されます。

データの種類 DOUBLE

参照項目: 詳細については、“乗数とシフト間隔を使用した日時の増分” (*SAS 関数と CALL ルーチン: リファレンス*)を参照してください。

例 YEAR.3 は、暦年の 3 月 1 日に開始し、翌年の 2 月に終了するようにシフトされた年次期間を指定します。

seasonality
数値またはサイクルを指定します。この引数では、日付と時間のサイクルをより柔軟に操作できます。1 年間の季節性を 52 週にするか 53 週にするかを指定できます。

数値またはサイクルを指定します。

この引数では、日付と時間のサイクルをより柔軟に操作できます。1 年間の季節性を 52 週にするか 53 週にするかを指定できます。

データの種類 DOUBLE、CHAR、NCHAR、NVARCHAR、VARCHAR

例 この例では、次の関数で同じ結果が生成されます。
INTINDEX('MONTH', sasdate, 3); INTINDEX('MONTH', sasdate, 'QTR');
最初の例の *seasonality* は数値(月数)で、2 番目の例の *seasonality* はサイクル(QTR)です。

詳細

基本

INTINDEX 関数は、季節インデックスを返します。この関数には、間隔と適切な日付、時間またはタイムスタンプの値を指定します。季節インデックスは、指定した間隔の季節サイクルでの日付、時間またはタイムスタンプの値の位置を表す数値です。1 年サイクルには 12 か月があり 12 月はその年の 12 番目の月であるため、この例では 12 の値が返されます。

```
select intindex('month', date'2012-12-01');
```

次の例では、両方のステートメントが 2013 年第 1 四半期内の日付を含むため、INTINDEX は同じ値(1)を返します。

```
select intindex('qtr', date'2013-01-01');
select intindex('qtr', date'2013-03-01');
```

次の例では、日単位のデータは週単位の周期に含まれ、2012 年 12 月 7 日は金曜日で 6 番目の曜日なので、6 の値を返します。

```
select intindex('day', date'2012-12-07');
```

間隔

間隔は、基本的なものから複雑なものまであります。基本間隔は、SAS が経過時間内にカウントできる測定単位(DAY、MONTH、HOUR など)です。より複雑な間隔を指定するには、基本間隔名に乗数とシフトインデックスを組み合わせて使用できます。

間隔構文は次のとおりです。

interval[*multiple*][.*shift-index*]

詳細については、“必須引数” (599 ページ)を参照してください。

Interval と *Date-Time-Value* の関係

```
sasdate=to_double(date'2012-12-07');
x=intindex('day', sasdate);
put x;
```

季節指標を正しく識別するには、間隔が日付、時間またはタイムスタンプの値に適合する必要があります。たとえば、 intindex('month', date'2012–12–01');は、年間隔には 12 か月があり 12 月はその年の 12 番目の月であるため、12 の値を返します。DAY 間隔には SAS 日付値が必要です。次の例では、日単位のデータは週単位の周期に含まれ、2012 年 12 月 7 日は金曜日で 6 番目の曜日なので、6 の値を返します。

```
select intget('day', date'2012-12-07');
```

DAY 間隔には SAS 日付値が必要です。

この例では、QTR 間隔が TIMESTAMP 値でなく SAS 日付値を日付に必要とするため、欠損値が返されます。

```
select intindex('qtr',timestamp'2013-01-01 00:00:00');
```

この例では、12 の値が返されます。DTMONTH 間隔には TIMESTAMP 値が必要です。

```
select intindex('dtmonth',timestamp'2013-12-01 00:00:00');
```

日付間隔と時間間隔の操作の詳細については、*“日付間隔と時間間隔” (SAS 関数と CALL ルーチン: リファレンス)*を参照してください。

小売カレンダーの間隔

INTINDEX 関数は、小売業界によるカレンダーの間隔でも使用できます。これらの間隔は ISO 8601 に準拠します。詳細については、*“小売カレンダーの間隔: ISO 8601 準拠” (SAS 関数と CALL ルーチン: リファレンス)*を参照してください。

季節性

季節性は、年内の異なる間隔での周期的変動を測定する時系列の概念です。季節性の指定では、季節が最も一般的な変動ソースです。たとえば、家庭暖房用の灯油は、一般的に他の季節よりも冬の売り上げの方が高くなります。多くの場合、日単位の時系列では曜日による定期的な変動(週末のレジャーで 支出が増えるなど)が発生します。INTINDEX 関数は、季節性の概念を使用して季節インデックスを返します。この関数には、日付、時間またはタイムスタンプの間隔と値を指定します。

比較

INTINDEX 関数は季節インデックスを返しますが、INTCINDEX 関数はサイクルインデックスを返します。

次の例では、2013 年 4 月 4 日は木曜日で週の 5 番目の曜日なので、INTINDEX 関数は 5 を返します。

```
select intindex('day', date'2013-04-04');
```

同じ日を使用しても、2013 年 4 月 4 日は年の 14 番目の週なので、INTCINDEX 関数は 14 を返します。

```
select intcindex('day', date'2013-04-04');
```

この例では、INTINDEX 関数は分を返します。

```
select intindex('minute', timestamp'2012-09-01 06:05:04');
```

同じ日付と時間を使用しても、INTCINDEX 関数は時刻を返します。

```
select intcindex('minute', timestamp'2012-09-01 06:05:04');
```

たとえば、intseas('interval');では、INTSEAS は intindex('interval', date);によって返される最大数を返します。

例

次のステートメントは、INTINDEX 関数を示しています。

ステートメント	結果
select intindex('qtr', date'2013-08-14');	3
select intindex('dtqtr', timestamp'2013-12-23 15:09:19');	4
select intindex('hour', time'09:05:15');	10
select intindex('month', date '2013-02-26');	2
select intindex('dtmonth', timestamp'2013-05-28 05:15:00');	5
select intindex('week', date '2013-09-09');	37
select intindex('tenday', date '2013-04-16');	11

関連項目

関数:

- “INTCINDEX 関数” (582 ページ)
- “INTCYCLE 関数” (591 ページ)
- “INTSEAS 関数” (611 ページ)

その他のリファレンス:

- *SAS/ETS User's Guide*

INTNX 関数

DOUBLE としてエンコードされた SAS 日付、時間または日時の値を増分し、DOUBLE としてエンコードされた SAS 日付、時間または日時の値を返します。

カテゴリ:	日付と時間
制限事項:	FedSQL は、カスタムの日付または時間間隔をサポートしていません。
返されるデータの種類:	DOUBLE

構文

INTNX({*interval*[*multiple*][*.shift-index*]}, *start-from*, *increment*[, '*alignment*'])

INTNX(*start-from*, *increment*[, '*alignment*'])

必須引数

interval
文字列を評価するか、文字列に強制変換でき、WEEK、MONTH、QTR などの間隔名を含む文字定数、変数、または式を指定します。

データの種類　CHAR、NCHAR、NVARCHAR、VARCHAR

注　*interval* の可能な値は、"日付間隔と時間間隔について" (*SAS 出力形式と入力形式: リファレンス*)にリストされています。

ヒント　*Interval* は、大文字または小文字で表示できます。

例　YEAR は、年ベースの間隔を指定します。

start-from
DOUBLE としてエンコードされた SAS 日付、時間または日時の値を表し、開始点を識別する式を指定します。

データの種類　DOUBLE

increment
日付に追加する日数を表す負、正またはゼロの整数を指定します。*increment* は、*start-from* の値をシフトする間隔数です。

データの種類　DOUBLE

オプション引数

multiple
乗数を指定します(省略可能)。基本タイプの間隔の期間に対する倍数と同等の間隔を設定します。

データの種類　DOUBLE

参照項目:　詳細については、"乗数とシフト間隔を使用した日時の増分" (*SAS 関数と CALL ルーチン: リファレンス*)を参照してください。

例　YEAR2 は、2 年、つまり 2 年間続く間隔タイプを指定します。

shift-index
シフトインデックスを指定します(省略可能)。間隔の開始時点を指定したサブ期間にシフトします。

制限事項　シフトインデックスは、間隔全体のサブ期間の数以下にする必要があります。たとえば、2 年間隔では 25 番目の月は存在しないため、YEAR2.24 は使用できますが YEAR2.25 はエラーになります。

デフォルトのシフト期間が間隔の種類と同じ場合、複数期間の間隔のみを任意のシフトインデックスでシフトできます。たとえば、MONTH の種類の間隔はデフォルトでは MONTH のサブ期間でシフトされるため、シフトインデックスで月間隔をシフトできません。ただし、各 MONTH2 間隔には 2 つの MONTH 間隔が含まれるため、シフトインデックスで 2 か月

間隔をシフトできます。たとえば、間隔名 MONTH2.2 では、偶数月の 1 日目に開始する 2 か月の期間が指定されます。

データの種類 DOUBLE

参照項目: 詳細については、“乗数とシフト間隔を使用した日時の増分” (*SAS 関数と CALL ルーチン: リファレンス*)を参照してください。

例 YEAR.3 は、暦年の 3 月 1 日に開始し、翌年の 2 月に終了するようにシフトされた年次期間を指定します。

'*alignment*'

間隔内の SAS 日付の位置を制御します。*alignment* は引用符で囲む必要があります。*alignment* には、BEGINNING、MIDDLE、END、SAME のいずれかの値を指定します。SAME では、返された日付を入力日付と同じ場所に配置するように指定されます。詳細については、SAS DS2 言語リファレンスを参照してください。

間隔内の SAS 日付の位置を制御します。*alignment* は引用符で囲む必要があります。*Alignment* には次のいずれかの値を指定できます。

BEGINNING

返された日付値または日時値を間隔の開始点に配置するように指定します。

別名 B

MIDDLE

返された日付値または日時値を間隔の中間点(開始位置と終了位置の値の平均)に配置するように指定します。

別名 M

END

返された日付値または日時値を間隔の終了点に配置するように指定します。

別名 E

SAME

返された日付を入力日付と同じ場所に配置するように指定します。

別名 S

SAMEDAY

参照項目: “SAME 配置” (606 ページ)

デフォルト BEGINNING

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

参照項目: “SAS 日付出力の間隔内の配置” (606 ページ)

詳細

基本

INTNX 関数は、DAY、WEEK、QTR、MINUTE などの間隔、または定義したカスタム間隔単位で日付、時間または日時の値を増分します。間隔は、開始日付、時間または日時の値、および指定した時間間隔に基づきます。

INTNX 関数は、*start-from* 引数で指定した間隔の開始日付、時間または日時の値に対する SAS 日付値を返します。(日付値をカレンダー日付に変換するには、DATE9. 形式などの有効な DS2 日付形式を使用します)。2011 年 10 月 17 日の週から 6 週間後の週の開始日を確認する方法の例を次に示します。

```
select put(intnx('week', date'2011-10-17', 6), date9.);
```

INTNX は値 27NOV2011 を返します。

日付間隔と時間間隔の操作の詳細については、“日付間隔と時間間隔” (*SAS 関数と CALL ルーチン: リファレンス*)を参照してください。

間隔

間隔は、基本的なものから複雑なものまであります。基本間隔は、SAS が経過時間内にカウントできる測定単位(DAY、MONTH、HOUR など)です。より複雑な間隔を指定するには、基本間隔名に乗数とシフトインデックスを組み合わせて使用できます。

間隔構文は次のとおりです。

interval[*multiple*][.*shift-index*]

詳細については、“必須引数” (604 ページ)を参照してください。

SAS 日付出力の間隔内の配置

通常、SAS 日付値は *interval* 引数で指定した時間間隔の開始点に配置されます。

任意の *alignment* 引数を使用して、返された値の配置を指定できます。値 BEGINNING、MIDDLE または END は、それぞれ間隔の開始点、中間点または終了点に日付を配置します。

SAME 配置

alignment 引数の SAME 値を使用した場合、INTNX は指定した間隔の増分を計算した後に同じカレンダー日付を返します。同じカレンダー日付は、間隔ではなく間隔のシフト期間に基づいて配置されます。有効なシフト期間を表示するには、“カテゴリ別の間隔” (588 ページ)を参照してください。

シフト期間のほとんどの値は、その対応する間隔と等しくなります。この例外は、間隔 WEEK、WEEKDAY、QTR、SEMIYEAR、YEAR、およびこれらに対応する DT です。WEEK および WEEKDAY 間隔には DAYS のシフト期間が含まれ、QTR、SEMIYEAR および YEAR 間隔には MONTH のシフト期間が含まれます。たとえば、YEAR で SAME 配置を使用した場合、この間隔のシフト期間である MONTH に基づいて同日に配置されます。YEAR 間隔の同日に配置されるわけではありません。倍

数の間隔を指定した場合、デフォルトのシフト間隔は倍数の間隔ではなく元の間隔に基づきます。

QTR、SEMIYEAR および YEAR 間隔に SAME 配置を使用する場合、計算結果日付は間隔の開始点(入力日付)からの月数と同じです。月の日は可能な限り同じ日に合わせられます。月によっては日数が異なるため、月の日を常に一致させることは不可能です。

シフト期間の詳細については、“カテゴリ別の間隔” (588 ページ)を参照してください。

配置の間隔

入力日付の配置に基づいて計算結果日付を配置するには、*alignment* 引数に SAME 値を使用します。

```
/*** returns 22MAR2011 ***/
select put(intnx('week', date'2011-03-15', 1, 'same'), date9.);

/*** returns 15MAR2016 ***/
select put(intnx('year', date'2011-03-15', 5, 'same'), date9.);
```

日付の調整

増分された間隔の日付が存在しない場合、INTNX 関数はその日付を自動的に調整します。次に例を示します。

```
/*** returns 15AUG2011 ***/
select put(intnx('month', date'2011-03-15', 5, 'same'), date9.);

/*** returns 28FEB2014 ***/
select put(intnx('year', date'2012-02-29', 2, 'same'), date9.);

/*** returns 30SEP2011 ***/
select put(intnx('month', date'2011-08-31', 1, 'same'), date9.);

/*** returns 01MAR2012 (the 1st day of the 3rd month of the year) ***/
select put(intnx('year', date'2011-03-01', 1, 'same'), date9.);

/*** returns 29FEB2012 (the 60th day of the year) ***/
select put(intnx('year', date'2011-03-01', 1, 'same', 'day'), date9.);
```

次の例では、INTNX 関数は値 01JAN2014 を返します。この値は、開始日(29FEB2012)の年から 2 年後の最初の日付です。

```
select put(intnx('year', date'2012-02-29', 2), date9.);
```

この例では、INTNX 関数は値 28FEB2014 を返します。この場合、開始日は 2012 年で、結果となる年は 2 年後(2014)、月は同月(2 月)、日付は 28 日が 2014 年 2 月 29 日に最も近いため、これらの値になります。

```
select put(intnx('year', date'2012-02-29', 2, 'same'), date9.);
```

小売カレンダーの間隔

小売業界では、1 年のカレンダーを 13 週間からなる 4 つの期間に分けてデータを計算することがよくあります。期間の形式は、4-4-5、4-5-4 または 5-4-4 のいずれか

に基づきます。1 番目、2 番目、3 番目の数値には、それぞれ各期間の 1 か月目、2 か月目、3 か月目の週数を指定します。詳細については、"ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の処理" (*SAS 出力形式と入力形式: リファレンス*) を参照してください。

例

例 1: INTNX 関数の使用

次のステートメントは、INTNX 関数を示しています。

ステートメント	結果
select intnx('year', date'2013-02-05', 3); select put(intnx('year', date'2013-02-05', 3), date7.);	20454 01Jan16
select intnx('month', date'2013-01-05', 0); select put(intnx('month', date'2013-01-05', 0), date7.);	19359 01JAN13
select intnx('semiyear', date'2013-01-01', 1); select put(intnx('semiyear', date'2013-01-01', 1), date7.);	19540 01JUL13
select intnx('month2', date'2012-08-01', -1); select put(intnx('month2', date'2012-08-01', -1), date7.);	19114 01MAY12
select intnx('semimonth2.2', date'2013-04-01', 4); select put(intnx('semimonth2.2', date'2013-04-01', 4), date7.);	19555 16JUL13

例 2: ALIGNMENT 引数の使用

任意の *alignment* 引数を使用して日付を進める例を次に示します。

ステートメント	結果
select intnx('month', date'2013-01-01', 5, 'beginning'); select put(intnx('month', date'2013-01-01', 5, 'beginning'), date7.);	19510 01JUN13
select intnx('month', date'2013-01-01', 5, 'middle'); select put(intnx('month', date'2013-01-01', 5, 'middle'), date7.);	19524 15JUN13
select intnx('month', date'2013-01-01', 5, 'end'); select put(intnx('month', date'2013-01-01', 5, 'end'), date7.);	19539 30JUN13
select intnx('month', date'2013-01-01', 5, 'sameday'); select put(intnx('month', date'2013-01-01', 5, 'sameday'), date7.);	19510 01JUN13

ステートメント	結果
select intnx('month', date'2013-03-15', 5, 'same'); select put(intnx('month', date'2013-03-15', 5, 'same'), date7.);	19585 15AUG13

関連項目

関数:

- “INTCK 関数” (585 ページ)

その他のリファレンス:

- “FedSQL の日付と時間” (59 ページ)

INTRR 関数

内部利益率を小数値で返します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

INTRR(*freq*, *c0, c1*[*..., cn*])

必須引数

freq
目的の内部利益率に関連付けられた指定ベース期間での支払い数を示す数値です。

範囲 *freq* > 0

データの種類 DOUBLE

ヒント *freq* = 0 は、連続複利計算を許可するフラグです。

c0, c1[***..., cn***]
任意の現金払いを示す数値です。

データの種類 DOUBLE

詳細

INTRR 関数は、現金払いセット *c0*, *c1*,..., *cn* の指定ベース期間での内部利益率を返します。2 つの連続した支払い間の時間間隔は同じとみなされます。引数 *freq* > 0 は、指定ベース期間で発生する支払い数を表します。各インスタンスから発行される情報の数は制限されます。

内部利益率は、一連の支払いの正味現在価値が 0 となる利率です。(“NPV 関数” (704 ページ)を参照してください。)次の式で表されます。

$$r = \begin{cases} \frac{1}{x^{freq}} - 1 & freq > 0 \\ -log_{\varepsilon}(x) & freq = 0 \end{cases}$$

この式では、*x* は多項式の実根です。

$$\sum_{i=0}^{n} c_i x^i = 0$$

重根の場合、1 つの実根が返され、返される内部利益率の非一意性に関する警告が発行されます。支払いの値によっては、方程式の根が存在しないことがあります。その場合、欠損値が返されます。

支払いの欠損値は 0 値として扱われます。*freq* > 0 の場合、計算される利益率は、指定ベース期間で有効な利率です。月払いでの四半期の内部利益率(ベース期間は 3 か月)を計算するには、*freq* を 3 に設定します。

freq が 0 の場合は連続複利計算とみなされ、ベース期間は 2 つの連続する支払い間の時間間隔です。計算される内部利益率は、ベース期間の名目利益率です。月単位の支払いを連続複利計算するには、*freq* を 0 に設定します。計算される内部利益率は、月単位の利率です。

比較

IRR 関数は INTRR と似ていますが、IRR 関数では内部利益率がパーセントという点で異なります。

例

当初の支払いが$400 で、次の 3 年間の予定支払い額がそれぞれ$100、$200、$300 の場合、年間内部利益率は次のように計算できます。

```
select intrr(1,-400,100,200,300);
```

```
0.19437709962747
```

SAS は次の出力をログに書き込みます。

```
0.1943770996
```

関連項目

関数:

- “IRR 関数” (624 ページ)

INTSEAS 関数

季節サイクルの長さを返します。この関数には、日付、時間または日時の間隔を指定します。

カテゴリ:	日付と時間
制限事項:	FedSQL は、カスタムの日付または時間間隔をサポートしていません。
返されるデータの種類:	DOUBLE

構文

INTSEAS({*interval*[*multiple*][.*shift-index*]}[, *seasonality*])

必須引数

interval
文字列を評価するか、文字列に強制変換でき、WEEK、MONTH、QTR などの間隔名を含む文字定数、変数、または式を指定します。

データの種類	CHAR、NCHAR、NVARCHAR、VARCHAR
注	*interval* の可能な値は、“日付間隔と時間間隔について” (*SAS 出力形式と入力形式: リファレンス*)にリストされています。
ヒント	*Interval* は、大文字または小文字で表示できます。
例	YEAR は、年ベースの間隔を指定します。

オプション引数

multiple
乗数を指定します(省略可能)。基本タイプの間隔の期間に対する倍数と同等の間隔を設定します。

データの種類	DOUBLE
参照項目:	詳細については、“乗数とシフト間隔を使用した日時の増分” (*SAS 関数と CALL ルーチン: リファレンス*)を参照してください。

例 YEAR2 は、2 年、つまり 2 年間続く間隔タイプを指定します。

shift-index
シフトインデックスを指定します(省略可能)。間隔の開始時点を指定したサブ期間にシフトします。

制限事項 シフトインデックスは、間隔全体のサブ期間の数以下にする必要があります。たとえば、2 年間隔では 25 番目の月は存在しないため、YEAR2.24 は使用できますが YEAR2.25 はエラーになります。

デフォルトのシフト期間が間隔の種類と同じ場合、複数期間の間隔のみを任意のシフトインデックスでシフトできます。たとえば、MONTH の種類の間隔はデフォルトでは MONTH のサブ期間でシフトされるため、シフトインデックスで月間隔をシフトできません。ただし、各 MONTH2 間隔には 2 つの MONTH 間隔が含まれるため、シフトインデックスで 2 か月間隔をシフトできます。たとえば、間隔名 MONTH2.2 では、偶数月の 1 日目に開始する 2 か月の期間が指定されます。

データの種類 DOUBLE

参照項目: 詳細については、"乗数とシフト間隔を使用した日時の増分" (*SAS 関数と CALL ルーチン: リファレンス*)を参照してください。

例 YEAR.3 は、暦年の 3 月 1 日に開始し、翌年の 2 月に終了するようにシフトされた年次期間を指定します。

seasonality
数値を指定します。この引数では、日付と時間のサイクルをより柔軟に操作できます。1 年間の季節性を 52 週にするか 53 週にするかを指定できます。

数値を指定します。

この引数では、日付と時間のサイクルをより柔軟に操作できます。1 年間の季節性を 52 週にするか 53 週にするかを指定できます。

デフォルト 52

データの種類 DOUBLE、CHAR、NCHAR、NVARCHAR、VARCHAR

例 次の例では、*seasonality* 引数により
INTSEAS('MONTH', 'qtr');
関数呼び出しで値 3 が返されます。この関数呼び出しは
INTSEAS('MONTH');
seasonality 引数がないため、値 12 を返します。

詳細

基本

INTSEAS 関数は、季節サイクル内の間隔数を返します。たとえば、時系列の間隔が月単位の場合、多くのプロシジャでオプション INTERVAL=MONTH が使用されます。この場合、データ内の各オブザベーションは特定の月に対応します。月単位のデータは、1 年間で周期的とみなされます。1 年には 12 か月あるため、季節サイクル(年)内の間隔(月)数は 12 です。

四半期のデータは、1 年間で周期的とみなされます。1 年には 4 つの四半期があるため、季節サイクル内の間隔数は 4 です。

周期性は常に 1 年とは限りません。たとえば、INTERVAL=DAY は 1 週間の周期とみなされます。1 週間には 7 日間あるため、季節サイクルの間隔数は 7 です。

日付間隔と時間間隔の操作の詳細については、“日付間隔と時間間隔” (*SAS 関数と CALL ルーチン: リファレンス*)を参照してください。

間隔

間隔は、基本的なものから複雑なものまであります。基本間隔は、SAS が経過時間内にカウントできる測定単位(DAY、MONTH、HOUR など)です。より複雑な間隔を指定するには、基本間隔名に乗数とシフトインデックスを組み合わせて使用できます。

間隔構文は次のとおりです。

interval[*multiple*][.*shift-index*]

詳細については、“必須引数” (611 ページ)を参照してください。

小売カレンダーの間隔

小売業界では、1 年のカレンダーを 13 週間からなる 4 つの期間に分けてデータを計算することがよくあります。期間の形式は、4-4-5、4-5-4 または 5-4-4 のいずれかに基づきます。1 番目、2 番目、3 番目の数値には、それぞれ各期間の 1 か月目、2 か月目、3 か月目の週数を指定します。詳細については、“ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の処理” (*SAS 出力形式と入力形式: リファレンス*)を参照してください。

季節性

季節性は、年内の異なる間隔での周期的変動を測定する時系列の概念です。季節性の指定では、季節が最も一般的な変動ソースです。たとえば、家庭暖房用の灯油は、一般的に他の季節よりも冬の売り上げの方が高くなります。多くの場合、日単位の時系列では曜日による定期的な変動(週末のレジャーで 支出が増えるなど)が発生します。INTSEAS 関数は、季節性の概念を使用して季節サイクルの長さを返します。この関数には、日付、時間または日時の間隔を指定します。

例

次のステートメントは、INTCYCLE 関数を示しています。

ステートメント	結果
select intseas('year');	1
select intseas('semiyear');	2
select intseas('quarter');	4
select intseas('month');	12
select intseas('semimonth');	24
select intseas('tenday');	36
select intseas('week');	52
select intseas('weekday');	5
select intseas('hour');	24
select intseas('minute');	60
select intseas('month2.2');	6
select intseas('week2');	26
select intseas('month4.3');	3
select intseas('day1');	7

関連項目

関数:

- “INTCYCLE 関数” (591 ページ)
- “INTINDEX 関数” (599 ページ)

その他のリファレンス:

- *SAS/ETS User's Guide*

INTSHIFT 関数

ベース間隔に対応するシフト間隔を返します。

カテゴリ: 日付と時間

制限事項: FedSQL は、カスタムの日付または時間間隔をサポートしていません。

返されるデータの種類: CHAR、NCHAR、NVARCHAR、VARCHAR

構文

INTSHIFT(*interval*[*multiple*][.*shift-index*])

必須引数

interval
文字列を評価するか、文字列に強制変換でき、WEEK、MONTH、QTR などの間隔名を含む文字定数、変数、または式を指定します。

データの種類: CHAR、NCHAR、NVARCHAR、VARCHAR

注: *interval* の可能な値は、"日付間隔と時間間隔について" (*SAS 出力形式と入力形式: リファレンス*)にリストされています。

ヒント: *Interval* は、大文字または小文字で表示できます。

例: YEAR は、年間隔を指定します。

オプション引数

multiple
乗数を指定します(省略可能)。基本タイプの間隔の期間に対する倍数と同等の間隔を設定します。

データの種類: DOUBLE

参照項目: 詳細については、"乗数とシフト間隔を使用した日時の増分" (*SAS 関数と CALL ルーチン: リファレンス*)を参照してください。

例: YEAR2 は 2 年の期間で構成されます。

shift-index
シフトインデックスを指定します(省略可能)。間隔の開始時点を指定したサブ期間にシフトします。

制限事項 シフトインデックスは、間隔全体のサブ期間の数以下にする必要があります。たとえば、2 年間隔では 25 番目の月は存在しないため、YEAR2.24 は使用できますが YEAR2.25 はエラーになります。

デフォルトのシフト期間が間隔の種類と同じ場合、複数期間の間隔のみを任意のシフトインデックスでシフトできます。たとえば、MONTH の種類の間隔はデフォルトでは MONTH のサブ期間でシフトされるため、シフトインデックスで月間隔をシフトできません。ただし、各 MONTH2 間隔には 2 つの MONTH 間隔が含まれるため、シフトインデックスで 2 か月間隔をシフトできます。たとえば、間隔名 MONTH2.2 では、偶数月の 1 日目に開始する 2 か月の期間が指定されます。

データの種類 DOUBLE

参照項目: 詳細については、"乗数とシフト間隔を使用した日時の増分" (*SAS 関数と CALL ルーチン: リファレンス*)を参照してください。

例 YEAR.3 は、暦年の 3 月 1 日に開始し、翌年の 2 月に終了するようにシフトされた年次期間を指定します。

詳細

基本

INTSHIFT 関数は、ベース間隔に対応するシフト間隔を返します。カスタム間隔の場合、返される値はベースカスタム間隔名です。INTSHIFT は、間隔と間隔シフトの倍数は無視します。

INTSHIFT 関数は、小売業界によるカレンダーの間隔でも使用できます。これらの間隔は ISO 8601 に準拠します。詳細については、"ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の処理" (*SAS 出力形式と入力形式: リファレンス*)を参照してください。

間隔

間隔は、基本的なものから複雑なものまであります。基本間隔は、SAS が経過時間内にカウントできる測定単位(DAY、MONTH、HOUR など)です。より複雑な間隔を指定するには、基本間隔名に乗数とシフトインデックスを組み合わせて使用できます。

間隔構文は次のとおりです。

interval[*multiple*][.*shift-index*]

詳細については、"必須引数" (615 ページ)を参照してください。

例

次のステートメントは、INTSHIFT 関数を示しています。

ステートメント	結果
select intshift('year');	MONTH
select intshift('dtyear');	DTMONTH
select intshift('minute');	DTMINUTE
select intshift('weekdays');	WEEKDAY
select intshift('weekday5.4');	WEEKDAY
select intshift('qtr');	MONTH
select intshift('dttenday');	DTTENDAY

INTTEST 関数

時間間隔が有効な場合は 1、時間間隔が無効な場合は 0 を返します。

カテゴリ: 日付と時間

制限事項: FedSQL は、カスタムの日付または時間間隔をサポートしていません。

返されるデータの種類: DOUBLE

構文

INTTEST(*interval*[*multiple*][.*shift-index*])

必須引数

interval
文字列を評価するか、文字列に強制変換でき、WEEK、MONTH、QTR などの間隔名を含む文字定数、変数、または式を指定します。

データの種類: CHAR、NCHAR、NVARCHAR、VARCHAR

注: *interval* の可能な値は、“日付間隔と時間間隔について” (*SAS 出力形式と入力形式: リファレンス*)にリストされています。

ヒント　*Interval* は、大文字または小文字で表示できます。

例　YEAR は、年ベースの間隔を指定します。

オプション引数

multiple
乗数を指定します(省略可能)。基本タイプの間隔の期間に対する倍数と同等の間隔を設定します。

データの種類　DOUBLE

参照項目:　詳細については、"乗数とシフト間隔を使用した日時の増分" (*SAS 関数と CALL ルーチン: リファレンス*)を参照してください。

例　YEAR2 は、2 年、つまり 2 年間続く間隔タイプを指定します。

shift-index
シフトインデックスを指定します(省略可能)。間隔の開始時点を指定したサブ期間にシフトします。

制限事項　シフトインデックスは、間隔全体のサブ期間の数以下にする必要があります。たとえば、2 年間隔では 25 番目の月は存在しないため、YEAR2.24 は使用できますが YEAR2.25 はエラーになります。

デフォルトのシフト期間が間隔の種類と同じ場合、複数期間の間隔のみを任意のシフトインデックスでシフトできます。たとえば、MONTH の種類の間隔はデフォルトでは MONTH のサブ期間でシフトされるため、シフトインデックスで月間隔をシフトできません。ただし、各 MONTH2 間隔には 2 つの MONTH 間隔が含まれるため、シフトインデックスで 2 か月間隔をシフトできます。たとえば、間隔名 MONTH2.2 では、偶数月の 1 日目に開始する 2 か月の期間が指定されます。

データの種類　DOUBLE

参照項目:　詳細については、"乗数とシフト間隔を使用した日時の増分" (*SAS 関数と CALL ルーチン: リファレンス*)を参照してください。

例　YEAR.3 は、暦年の 3 月 1 日に開始し、翌年の 2 月に終了するようにシフトされた年次期間を指定します。

詳細

基本

INTTEST 関数は、有効な間隔名を確認します。この関数は、*multiple* と *shift-index* の有効な値を確認するときに役立ちます。乗数とシフトインデックスの詳細につい

ては、*SAS プログラマガイド: エッセンシャル*の"Multiunit Intervals"を参照してください。

INTTEST 関数は、小売業界によるカレンダーの間隔でも使用できます。これらの間隔は ISO 8601 に準拠します。詳細については、“日付、時間、および間隔” (*SAS プログラマガイド: エッセンシャル*)を参照してください。

間隔

間隔は、基本的なものから複雑なものまであります。基本間隔は、SAS が経過時間内にカウントできる測定単位(DAY、MONTH、HOUR など)です。より複雑な間隔を指定するには、基本間隔名に乗数とシフトインデックスを組み合わせて使用できます。

間隔構文は次のとおりです。

interval[*multiple*][.*shift-index*]

詳細については、“必須引数” (617 ページ)を参照してください。

例

次の例では、SAS は *interval* 引数が有効な場合は 1、interval 引数が無効な場合は 0 を返します。

ステートメント	結果
select inttest('month');	1
select inttest('week6.13');	1
select inttest('tenday');	1
select inttest('twoweeks');	0
select inttest('hour2.2');	1

INTZ 関数

ゼロファジーを使用して、引数の整数部を返します。

カテゴリ: 切り捨て

返されるデータの種類: DOUBLE

構文

INTZ(*expression*)

必須引数

expression
数値に評価される有効な式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

次の規則が適用されます。

- 引数の値が整数のみの場合、INTZ はその整数を返します。
- 引数が整数ではない正の値の場合、INTZ は引数より小さい最大の整数を返します。
- 引数が整数ではない負の値の場合、INTZ は引数より大きい最小の整数を返します。

比較

INT 関数とは異なり、INTZ 関数はゼロファジーを使用します。引数が整数の 1E-12 内にある場合、INT 関数はその整数に等しくなるように結果をファジー処理します。INTZ 関数は結果をファジー処理しません。そのため、INTZ 関数では予期しない結果になる可能性があります。

例

次のステートメントは、INTZ 関数を示しています。

ステートメント	結果
select intz('2.1');	2
select intz(-2.4);	-2
select intz(1+1.e11);	1E11

ステートメント	結果
select intz(-1.6);	-1

関連項目

関数:

- “CEIL 関数” (380 ページ)
- “CEILZ 関数” (381 ページ)
- “FLOOR 関数” (542 ページ)
- “FLOORZ 関数” (543 ページ)
- “INT 関数” (580 ページ)
- “MOD 関数” (671 ページ)
- “MODZ 関数” (673 ページ)
- “ROUND 関数” (805 ページ)
- “ROUNDZ 関数” (808 ページ)

IPMT 関数

将来の残高を達成するための、均等払いローンまたは定期預金に対する指定期間の利息の支払いを返します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

IPMT(*rate*, *period*, *number-of-periods*, *principal-amount*[, *future-amount*][, *type*])

必須引数

rate
支払期間ごとの利率を指定します。

データの種類 DOUBLE

period
利息の支払いを計算するための支払い期間を指定します。

要件　*period* は、*number-of-periods* の値以下の正の整数で指定する必要があります。

データの種類　INTEGER

number-of-periods
支払期間の数を指定します。

要件　*number-of-periods* は正の整数にする必要があります。

データの種類　INTEGER

principal-amount
ローンの元金を指定します。

データの種類　DOUBLE

注　欠損値が指定されている場合、ゼロとみなされます。

オプション引数

future-amount
将来の残高を指定します。

データの種類　DOUBLE

注　*future-amount* は、指定した支払い期間数終了後のローンの残高、または定期預金の将来の残高を指定できます。

0 を指定した場合、*future-amount* が省略されたか欠損値が指定されたとみなされます。

type
期首と期末のどちらで支払がなされるのかを指定します。0 は期末の支払を表し、1 は期首の支払を表します。

データの種類　INTEGER

注　*type* が省略された場合、または欠損値が指定されている場合は、ゼロとみなされます。

例

$8,000 のローンで名目年利が 10%、月単位の期末払いが 36 の場合、初回の定期的支払いでの利息の支払いは次のとおりです。

```
select ipmt(0.1/12, 1, 36, 8000);
```

```
66.6666666666666
```

SAS は次の出力をログに書き込みます。

```
66.66667
```

同じローンに期首払いがある場合、利息の支払いは次のように計算されます。

- select ipmt(0.1/12, 1, 36, 8000, 0, 1);

  計算によって 0 という値が返されます。
- select ipmt(0.1, 3, 3, 8000);

  計算によって 292.4471 という値が返されます。
- select ipmt(0.09/12, 359, 360, 125000, 0, 1);

  計算によって 14.80757 という値が返されます。

IQR 関数

四分位範囲を返します。

カテゴリ:	記述統計
返されるデータの種類:	DOUBLE

構文

IQR(*expression* [, ...*expression*])

必須引数

expression
数値に評価される有効な式を指定します。

データの種類	DOUBLE
参照項目:	“<sql-expression> 式” (918 ページ)
	“FedSQL 式” (46 ページ)

詳細

すべての引数に null または欠損値がある場合、ANSI モードまたは SAS モードのどちらであるかに応じて、結果は null または欠損値になります。 詳細については、“FedSQL が null および SAS 欠損値を処理する方法” (21 ページ)を参照してください。

例

次のステートメントは、IQR 関数を示しています。

ステートメント	結果
select iqr(2,4,1,3,999999);	2

関連項目

関数:

- “MAD 関数” (653 ページ)
- “PCTL 関数” (711 ページ)

IRR 関数

内部利益率をパーセントで返します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

IRR(*freq*, *c1, c2* [*..., cn*])

必須引数

freq
目的の内部利益率に関連付けられた指定ベース期間での支払い数を示す数値です。

範囲 *freq* > 0。

データの種類 DOUBLE

ヒント *freq* = 0 は、連続複利計算を許可するフラグです。

c1, c2[*..., cn*]
任意の現金払いを示す数値です。

要件 最小で 2 つの現金払い値が必要です。

データの種類 DOUBLE

詳細

IRR 関数は、現金払いセット *c1*, *c2*, ..., *cn* の指定ベース期間での内部利益率を返します。2 つの連続した支払い間の時間間隔は同じとみなされます。引数 *freq* > 0 は、指定ベース期間で発生する支払い数を表します。各インスタンスから発行される情報の数は制限されます。

比較

IRR 関数は INTRR と似ていますが、IRR 関数では内部利益率がパーセントという点で異なります。

例

当初の支払いが$400 で、次の 3 年間の予定支払い額がそれぞれ$100、$200、$300 の場合、年間内部利益率はパーセントで次のプログラムで表されます。

```
select irr(1,-400,100,200,300);
```

返される値は 19.437709963 です。

関連項目

関数:

- “INTRR 関数” (609 ページ)

JBESSEL 関数

ベッセル関数の値を返します。

カテゴリ: 数学

返されるデータの種類: DOUBLE

構文

JBESSEL(*nu*, *x*)

必須引数

nu
数値の定数、変数または式を指定します。

範囲 $nu \geq 0$

x
数値の定数、変数または式を指定します。

範囲 $x \geq 0$

詳細

JBESSEL 関数は、*x* で評価された次数 *nu* のベッセル関数の値を返します(詳細は Abramowitz and Stegun 1964 および Amos, Daniel, and Weston 1977 を参照)。

例

次のステートメントは、JBESSEL 関数を示しています。

ステートメント	結果
select jbessel(2, 2);	0.352834

関連項目

関数:

- “IBESSEL 関数” (571 ページ)

JULDATE 関数

SAS 日付値からユリウス暦の日付を返します。

カテゴリ: 日付と時間

返されるデータの種類: DOUBLE

構文

JULDATE(*date*)

必須引数

date
SAS 日付値を表す有効な式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

SAS 日付値は、1960 年 1 月 1 日から特定の日付までの日数を表す数値です。JULDATE 関数は、SAS 日付値をユリウス暦の日付に変換します。*date* がシステムオプション YEARCUTOFF=で定義された 100 年の期間内にある場合、結果は 3 桁、4 桁または 5 桁です。5 桁の結果では、最初の 2 桁は年を表し、残りの 3 桁は年間通算日(1 から 365、またはうるう年の場合は 1 から 366)を表します。先頭の 0 は結果から削除されるため、ユリウス暦の日付の年は省略されたり(末尾が 00 の年の場合)、1 桁のみになったり(年の末尾が 01 から 09 の場合)することがあります。それ以外の場合、結果は 7 桁になります。最初の 4 桁は年を表し、残りの 3 桁は年間通算日を表します。

00-09 で終わる年については、Z5.出力形式を使用して 5 桁のユリウス暦の日付をフォーマットできます。

FedSQL が日付を処理する方法の詳細については、“FedSQL の日付と時間” (59 ページ)を参照してください。

比較

関数 JULDATE7 は JULDATE に似ていますが、JULDATE7 は常に 4 桁の年を返す点で異なります。JULDATE7 は 2 桁の年を考慮する必要がなく、2000 年問題に対応しています。

例

次のステートメントは、JULDATE 関数を示しています。

ステートメント	結果
select juldate(mdy(12,31,2013));	7365
select put(juldate(mdy(12,31,2013)),z5.);	07365
select juldate(mdy(9,1,1999));	99244
select juldate(mdy(7,1,1886));	1886182

関連項目

関数:

- “DATEJUL 関数” (440 ページ)
- “JULDATE7 関数” (628 ページ)

JULDATE7 関数

SAS 日付値から 7 桁のユリウス暦の日付を返します。

カテゴリ: 日付と時間

返されるデータの種類: DOUBLE

構文

JULDATE7(*date*)

必須引数

date
SAS 日付値を表す有効な式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

SAS 日付値は、1960 年 1 月 1 日から特定の日付までの日数を表す数値です。JULDATE7 関数は、SAS 日付値から 7 桁のユリウス暦の日付を返します。最初の 4 桁は年を表し、残りの 3 桁は年間通算日を表します。

FedSQL が日付を処理する方法の詳細については、“FedSQL の日付と時間” (59 ページ)を参照してください。

比較

関数 JULDATE7 は JULDATE に似ていますが、JULDATE7 は常に 4 桁の年を返す点で異なります。JULDATE7 は 2 桁の年を考慮する必要がなく、2000 年問題に対応しています。

例

次のステートメントは、JULDATE7 関数を示しています。

ステートメント	結果
select juldate7(mdy(12,31,2006));	2007365
select juldate7(mdy(12,31,2016));	2016366

関連項目

関数:

- “JULDATE 関数” (626 ページ)

KURTOSIS 関数

式のすべての値の尖度を返します。

カテゴリ:	集計 記述統計
返されるデータの種類:	DOUBLE

構文

KURTOSIS([DISTINCT] *expression*)

必須引数

expression
有効な SQL 式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

オプション引数

DISTINCT
集計計算から重複行を削除します。

詳細

*尖度*は、主に分布の裾の重さの尺度です。尖度の値が大きい場合は、分布の裾が重いことを示しています。

null 値と SAS 欠損値は無視され、計算には含まれません。

少なくとも 4 つの非 null または非欠損の引数が必要です。それがない場合は、関数から null 値が返されます。すべての非 null または非欠損引数の値が等しい場合、尖度は数学的に定義されておらず、KURTOSIS 関数は null 値を返します。

集計関数を使用して、FROM 句にリストされているテーブル全体、または GROUP BY 句で指定されている各グループのデータの統計的要約を生成できます。GROUP BY 句は、指定された 1 つまたは複数の列でデータをグループ化します。GROUP BY 句を使用する場合、SELECT 句または HAVING 句の集計関数は、各グループのデータを要約する方法を FedSQL に指示します。FedSQL は、グループごとに個別に集計関数を計算します。GROUP BY を省略すると、テーブルまたはビュー内のすべての行が単一のグループと見なされます。

例

テーブル: WORLDTEMPS (1277 ページ)

次のステートメントは、KURTOSIS 関数を示しています。

ステートメント	結果
select kurtosis(AvgLow) from worldtemps;	-0.87431

関連項目

関数:

- “STDDEV 関数” (837 ページ)

SELECT ステートメント句:

- “SELECT 句” (980 ページ)
- “GROUP BY 句” (993 ページ)
- “HAVING 句” (995 ページ)

LARGEST 関数

k 番目に大きい非 null または非欠損値を返します。

カテゴリ: 記述統計

返されるデータの種類: DOUBLE

構文

LARGEST(*k*, *expression* [, ...*expression*])

必須引数

k
数値に評価され、最大の戻り値を表す有効な式を指定します。たとえば、*k* が 2 の場合、LARGEST 関数は、式のリストから 2 番目に大きい値を返します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

expression
数値に評価される検索先の有効な式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

k が null または欠損値、0 未満、または値の数よりも大きい場合、結果は null または欠損値になります。そうでない場合は、*k* が非 null または非欠損値の数よりも大きいと、結果は null または欠損値になります。

例

次のステートメントは、LARGEST 関数を示しています。

ステートメント	結果
select largest(1, 456, 789, .Q, 123);	789
select largest(2, 456, 789, .Q, 123);	456
select largest(3, 456, 789, .Q, 123);	123
select largest(4, 456, 789, .Q, 123);	.

関連項目

関数:

- “ORDINAL 関数” (710 ページ)
- “PCTL 関数” (711 ページ)
- “SMALLEST 関数” (830 ページ)

LCM 関数

数の集合に対する最小公倍数を返します。

カテゴリ:	数学
返されるデータの種類:	DOUBLE

構文

LCM(*expression*-1, *expression*-2 [, ...*expression-n*])

必須引数

expression
整数に評価される有効な式を指定します。

要件	少なくとも 2 つの引数が必要です。
データの種類	DOUBLE
注	いずれかの式がゼロに評価されると、エラーが発生し、欠損値が返されます。
参照項目:	“<sql-expression> 式” (918 ページ)
	“FedSQL 式” (46 ページ)

詳細

最小公倍数とは、2 つ以上の数を均等に割り切れる最小の数です。

例

次のステートメントは、LCM 関数を示しています。

ステートメント	結果
select lcm(1,5,3,0);	.
select lcm(25,70,85,130);	77350
select lcm(33,78);	858

関連項目

関数:

- “GCD 関数” (557 ページ)

LCOMB 関数

COMB 関数の対数を計算します。これは、*n* 個のオブジェクトから一度に *r* 個を取り出した組み合わせ数の対数です。

カテゴリ:	組み合わせ
返されるデータの種類:	DOUBLE

構文

LCOMB(*n*, *r*)

必須引数

n
負でない整数で要素の合計数を表します。サンプルはこの中から選ばれます。

r
負でない整数で選ばれた要素の数を表します。

制限事項 $r \leq n$

比較

LCOMB 関数は、COMB 関数の対数を計算します。

例

次のステートメントは、LCOMB 関数を示しています。

ステートメント	結果
select lcomb(5000, 500);	1621.441
select lcomb(100, 10);	30.48232

関連項目

関数:

- “COMB 関数” (389 ページ)

LEFT 関数

文字式を左揃えにします。

カテゴリ: 文字

返されるデータの種類: CHAR、NCHAR、NVARCHAR、VARCHAR

構文

LEFT(*expression*)

必須引数

expression
文字列に評価されるか強制変換できる有効な式を指定します。

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

LEFT は、先頭の空白を値の末尾に移動した文字列を返します。

例

次のステートメントは、LEFT 関数を示しています。

ステートメント	結果
	----+----1----+----2--
select left(' END-OF-YEAR');	END-OF-YEAR

関連項目

関数:

- “COMPRESS 関数” (396 ページ)

LFACT 関数

FACT(階乗)関数の対数を計算します。

カテゴリ:	組み合わせ
返されるデータの種類:	DOUBLE

構文

LFACT(*n*)

必須引数

n
整数で要素の合計数を表します。サンプルはこの中から選ばれます。

詳細

LFACT 関数は、FACT 関数の対数を計算します。

例

次のステートメントは、LFACT 関数を示しています。

ステートメント	結果
select lfact(5000);	37591.14
select lfact(100);	363.7394

関連項目

関数:

- “FACT 関数” (464 ページ)

LGAMMA 関数

ガンマ関数の自然対数を返します。

カテゴリ:	数学
返されるデータの種類:	DOUBLE

構文

LGAMMA(*expression*)

必須引数

expression
数値に評価される有効な式を指定します。

要件	正数にする必要があります。
データの種類	DOUBLE
参照項目:	“<sql-expression> 式” (918 ページ) “FedSQL 式” (46 ページ)

例

次のステートメントは、LGAMMA 関数を示しています。

ステートメント	結果
select lgamma(2);	0
select lgamma(1.5);	-0.120782238

LOG 関数

数値式の自然対数(底 e)を返します。

カテゴリ: 数学

返されるデータの種類: DOUBLE

構文

LOG(*expression*)

必須引数

expression
数値に評価される有効な SQL 式を指定します。

データの種類 DOUBLE

注 この関数の入力としてゼロまたは負の数が渡された場合、エラーメッセージはログに書き込まれず、欠損値が出力として返されます。

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

例

次のステートメントは、LOG 関数を示しています。

ステートメント	結果
select log(1.0);	0
select log(10.0);	2.302585

関連項目

関数:

- “E 関数” (458 ページ)
- “LOG10 関数” (641 ページ)
- “LOG2 関数” (640 ページ)

LOG1PX 関数

1 に引数を加えた値の対数を返します。

カテゴリ: 数学

返されるデータの種類: DOUBLE

構文

LOG1PX(*x*)

必須引数

x
数値定数、変数または式を指定します。

データの種類 DOUBLE

詳細

LOG1PX 関数は、1 に引数を加えた値の対数を計算します。LOG1PX 関数は、次の式(–1 < x の場合)で数学的に定義されます。

$$LOG1PX(x) = log(1 + x)$$

x が 0 に近い場合、LOG1PX(x)は LOG(1+x)より精度が高くなることがあります。

例

次のステートメントは、LOG1PX 関数を示しています。最初の例では、1 に値 0.5 を加えた値の対数を計算します。2 番目では、LOG1PX 関数を使用して X の値を計算し、LOG 関数を使用して Y の値を計算します。PUT 関数は HEX16 出力形式を適用して、FedSQL が可能な完全な精度を示します。

ステートメント	結果
select log1px(0.5);	0.405465
select put(log1px(1.e-5), hex16.) as x, put(log(1+1.e-5), hex16.) as y;	x=3EE4F8AEA9AE7317 y=3EE4F8AEA9AF0A25

関連項目

関数:

- “LOG 関数” (638 ページ)

LOG2 関数

数値式の 2 を底とする対数を返します。

カテゴリ: 数学

返されるデータの種類: DOUBLE

構文

LOG2(*expression*)

必須引数

expression
数値に評価される有効な SQL 式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

例

次のステートメントは、LOG2 関数を示しています。

ステートメント	結果
select log2(8.0);	3
select log2(4);	2

関連項目

関数:

- “LOG 関数” (638 ページ)
- “LOG10 関数” (641 ページ)

LOG10 関数

数値式の 10 を底とする対数を返します。

カテゴリ: 数学

返されるデータの種類: DOUBLE

構文

LOG10(*expression*)

必須引数

expression
数値に評価される有効な SQL 式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

例

次のステートメントは、LOG10 関数を示しています。

ステートメント	結果
select log10(1.0);	0
select log10(10.0);	1
select log10(100.0);	2

関連項目

関数:

- “LOG 関数” (638 ページ)
- “LOG2 関数” (640 ページ)

LOGBETA 関数

ベータ関数の対数を返します。

カテゴリ: 数学

返されるデータの種類: DOUBLE

構文

LOGBETA(*a*, *b*)

必須引数

a
第 1 形状パラメーターです (*a*>0)。

データの種類 DOUBLE

b
第 2 形状パラメーターです (*b*>0)。

データの種類 DOUBLE

詳細

LOGBETA 関数は、式で数学的に求められます。

$$log(\beta(a,b)) = log(\Gamma(a)) + log(\Gamma(b)) - log(\Gamma(a+b))$$

この式で、$\Gamma(.)$はガンマ関数です。

式が計算できない場合、LOGBETA は欠損値を返します。

例

次のステートメントは、LOGBETA 関数を示しています。

ステートメント	結果
select logbeta(5,3);	-4.6539603501575

関連項目

関数:

- “BETA 関数” (314 ページ)

LOGCDF 関数

左累積分布関数の対数を返します。

カテゴリ: 確率

参照項目: “CDF 関数” (334 ページ)

構文

LOGCDF('*distribution*', *quantile* [, *parameter-1*, ..., *parameter-k*])

必須引数

distribution
分布を特定する文字定数、変数または式です。

注: 各 LOGCDF 分布関数の引数は、対応する CDF 分布関数の引数と同じです。

有効な分布は次のとおりです。

分布	引数
Bernoulli	'BERNOULLI'
ベータ	'BETA'
二項	'BINOMIAL'
Cauchy	'CAUCHY'
カイ 2 乗	'CHISQUARE'
Conway-Maxwell-Poisson	'CONMAXPOI'
指数	'EXPONENTIAL'
F	'F'
ガンマ	'GAMMA'
一般化 Poisson	'GENPOISSON'
幾何	'GEOMETRIC'
超幾何	'HYPERGEOMETRIC'
Laplace	'LAPLACE'
ロジスティック	'LOGISTIC'
対数正規	'LOGNORMAL'
負の二項	'NEGBINOMIAL'
正規	'NORMAL'|'GAUSS'
正規混合	'NORMALMIX'
パレート	'PARETO'
Poisson	'POISSON'
T	'T'
Tweedie	'TWEEDIE'
一様	'UNIFORM'

分布	引数
Wald (逆ガウス)	'WALD'|'IGAUSS'
Weibull	'WEIBULL'

注: T、F および NORMALMIX を除き、最初の 4 文字で分布を最小限に識別できます。

quantile
確率変数の値を指定する、数値変数、定数または式です。

データの種類 DOUBLE

オプション引数

parameter-1, ..., parameter-k
特定の分布に適した任意の *shape*、*location* または *scale* パラメーターです。

データの種類 DOUBLE

詳細

LOGCDF 関数は、さまざまな連続分布および離散分布から左累積分布関数の対数(左側の対数)を計算します。詳細については、前述の表の個々の分布を参照してください。

表に示された分布の詳細については、“CDF 関数” (334 ページ)を参照してください。

関連項目

関数:

- “CDF 関数” (334 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “PDF 関数” (712 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

LOGISTIC 関数

引数のロジスティック変換を返します。

カテゴリ: 数学

構文

LOGISTIC(*argument*)

必須引数

argument
数値の確率変数の値を指定する、数値変数、定数または式です。*argument* が欠損値または null 値の場合、LOGISTIC 関数は欠損値または null 値を返します。

詳細

LOGISTIC 関数は引数のロジスティック変換を返します。通常は、対数オッズ値を確率尺度の値に変換するために使用されます。この関数は、数学的に次の式で表現されます。

$$logistic=\frac{e^x}{1+e^x}$$

引数に欠損値が含まれている場合、LOGISTIC 関数は欠損値または null 値を返します。

例

次のステートメントは、LOGISTIC 関数を示しています。

ステートメント	結果
select LOGISTIC(.5);	0.622459
select LOGISTIC(7.3);	0.999325

LOGPDF 関数

さまざまな連続分布および離散分布から確率密度(質量)関数の対数を計算します。

カテゴリ: 確率

別名: LOGPMF

参照項目: “PDF 関数” (712 ページ)

構文

LOGPDF(*'distribution'*, *quantile*, *parameter-1, ..., parameter-k*)

必須引数

distribution
分布を特定する文字定数、変数または式です。

注: 各 LOGPDF 分布関数の引数は、対応する PDF 分布関数の引数と同じです。

有効な分布は次のとおりです。

分布	引数
Bernoulli	'BERNOULLI'
ベータ	'BETA'
二項	'BINOMIAL'
Cauchy	'CAUCHY'
カイ 2 乗	'CHISQUARE'
Conway-Maxwell-Poisson	'CONMAXPOI'
指数	'EXPONENTIAL'
F	'F'
ガンマ	'GAMMA'
一般化 Poisson	'GENPOISSON'

分布	引数
幾何	'GEOMETRIC'
超幾何	'HYPERGEOMETRIC'
Laplace	'LAPLACE'
ロジスティック	'LOGISTIC'
対数正規	'LOGNORMAL'
負の二項	'NEGBINOMIAL'
正規	'NORMAL'|'GAUSS'
正規混合	'NORMALMIX'
パレート	'PARETO'
Poisson	'POISSON'
T	'T'
Tweedie	'TWEEDIE'
一様	'UNIFORM'
Wald (逆ガウス)	'WALD'|'IGAUSS'
Weibull	'WEIBULL'

注: T、F および NORMALMIX を除き、最初の 4 文字で分布を最小限に識別できます。

quantile
確率変数の値を指定する、数値定数、変数または式です。

データの種類 DOUBLE

parameter-1, ..., parameter-k
特定の分布に適した任意の *shape*、*location* または *scale* パラメーターです。

データの種類 DOUBLE

詳細

LOGPDF 関数は、さまざまな連続分布および離散分布から確率密度(質量)関数の対数を計算します。詳細については、前述の表の個々の分布を参照してください。

表に示された分布の詳細については、“PDF 関数” (712 ページ)を参照してください。

関連項目

関数:

- “CDF 関数” (334 ページ)
- “LOGCDF 関数” (643 ページ)
- “LOGSDF 関数” (649 ページ)
- “PDF 関数” (712 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

LOGSDF 関数

生存関数の対数を返します。

カテゴリ: 確率

参照項目: “SDF 関数” (819 ページ)

構文

LOGSDF('*distribution*', *quantile*, *parameter-1, ..., parameter-k*)

必須引数

distribution
分布を特定する文字定数、変数または式です。

注: 各 LOGSDF 分布関数の引数は、対応する CDF 分布関数の引数と同じです。

有効な分布は次のとおりです。

分布	引数
Bernoulli	'BERNOULLI'
ベータ	'BETA'
二項	'BINOMIAL'
Cauchy	'CAUCHY'
カイ 2 乗	'CHISQUARE'
Conway-Maxwell-Poisson	'CONMAXPOI'
指数	'EXPONENTIAL'
F	'F'
ガンマ	'GAMMA'
一般化 Poisson	'GENPOISSON'
幾何	'GEOMETRIC'
超幾何	'HYPERGEOMETRIC'
Laplace	'LAPLACE'
ロジスティック	'LOGISTIC'
対数正規	'LOGNORMAL'
負の二項	'NEGBINOMIAL'
正規	'NORMAL'|'GAUSS'
正規混合	'NORMALMIX'
パレート	'PARETO'
Poisson	'POISSON'
T	'T'
Tweedie	'TWEEDIE'
一様	'UNIFORM'

分布	引数
Wald (逆ガウス)	'WALD'|'IGAUSS'
Weibull	'WEIBULL'

注: T、F および NORMALMIX を除き、最初の 4 文字で分布を最小限に識別できます。

quantile
確率変数の値を指定する、数値定数、変数または式です。

データの種類 DOUBLE

parameter-1, ..., parameter-k
特定の分布に適した任意の *shape*、*location* または *scale* パラメーターです。

データの種類 DOUBLE

詳細

LOGSDF 関数は、さまざまな連続分布および離散分布から生存関数の対数を計算します。詳細については、“SDF 関数” (819 ページ)を参照してください。

表に示された分布の詳細については、“CDF 関数” (334 ページ)を参照してください。

関連項目

関数:

- “CDF 関数” (334 ページ)
- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “PDF 関数” (712 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

LOWCASE 関数

文字式のすべての文字を小文字に変換します。

カテゴリ:	文字
別名:	LOWER
返されるデータの種類:	CHAR、NCHAR、NVARCHAR、VARCHAR

構文

LOWCASE(*expression*)

必須引数

expression
文字列に評価されるか強制変換できる有効な式を指定します。

要件	リテラル文字表現は、単一引用符で囲む必要があります。
データの種類	CHAR、NCHAR、NVARCHAR、VARCHAR
参照項目:	“<sql-expression> 式” (918 ページ)
	“FedSQL 式” (46 ページ)

詳細

LOWCASE 関数は、文字式をコピーし、すべての大文字を小文字に変換して、変更された値を結果として返します。

比較

UPCASE 関数は、引数内のすべての文字を大文字に変換します。LOWCASE 関数は、引数内のすべての文字を小文字に変換します。

例

テーブル: AFEWWORDS (1256 ページ)

次のステートメントは、LOWCASE (または LOWER)関数を示しています。

ステートメント	結果
select lowcase('INTRODUCTION');	introduction
select lower(word2) from afewwords;	where thing body

関連項目

関数:

- “UPCASE 関数” (869 ページ)

MAD 関数

中央値からの中央絶対偏差を返します。

カテゴリ: 記述統計

返されるデータの種類: DOUBLE

構文

MAD(*expression-1*[, ...*expression-n*])

必須引数

expression
中央値からの中央絶対偏差を計算する数値に評価される有効な式を指定します。

データの種類 DOUBLE

詳細

すべての引数が欠損値または null 値の場合、結果は欠損値または null 値になります。それ以外の場合は、結果は非欠損値または非 null 値の中央値からの中央絶対偏差になります。中央値の計算式は、UNIVARIATE プロシジャで使用するものと同じ

です。詳細については、Base SAS プロシジャガイド: 統計プロシジャを参照してください。

例

次のステートメントは、MAD 関数を示しています。

ステートメント	結果
select mad(2, 4, 1, 3, 5, 999999);	1.5

関連項目

関数:

- “IQR 関数” (623 ページ)
- “MEDIAN 関数” (667 ページ)
- “PCTL 関数” (711 ページ)

MAKEDATE 関数

日付を年、月、日として返します。

カテゴリ: 日付と時間

返されるデータの種類: DATE

構文

MAKEDATE(*y*, *m*, *d*)

必須引数

y
年を指定します。

m
月を指定します。

d
日を指定します。

例

次のステートメントは、MAKEDATE 関数を示しています。

ステートメント	結果
select makedate(2020, 10, 30);	30OCT2020
select makedate(y, m, d) from adate;	30OCT2013

関連項目

関数:

- “DATEPART 関数” (441 ページ)
- “MAKETIME 関数” (655 ページ)
- “MAKETIMESTAMP 関数” (656 ページ)
- “TIMEPART 関数” (851 ページ)

MAKETIME 関数

時間を時、分、秒で返します。

カテゴリ: 日付と時間

返されるデータの種類: TIME

構文

MAKETIME(*h*, *m*, *s*)

必須引数

h
時間を指定します。

m
分を指定します。

s
秒を指定します。

例

次のステートメントは、MAKETIME 関数を示しています。

ステートメント	結果
select maketime(15,39,10);	15:39:10
select maketime(h, mn, s) from adate;	15:39:10

関連項目

関数:

- “DATEPART 関数” (441 ページ)
- “MAKEDATE 関数” (654 ページ)
- “MAKETIMESTAMP 関数” (656 ページ)
- “TIMEPART 関数” (851 ページ)

MAKETIMESTAMP 関数

タイムスタンプを返します。

カテゴリ: 日付と時間

返されるデータの種類: TIMESTAMP

構文

MAKETIMESTAMP(*d, t*) | (*y, m, d, h, m, s*)

オプション引数

d,t

d
ANSI 日付値を指定します。

t
ANSI 時間値を指定します。

(*y, m, d, h, m, s*)

y
2 桁または 4 桁の年の値を指定します。

m
1 から 12 までの月の値を指定します。

d
1 から 31 までの日の値を指定します。

h
0 から 24 までの時間の値を指定します。

m
0 から 59 までの分の値を指定します。

s
0 から 59 までの秒の値を指定します。

例

次のステートメントは、MAKETIMESTAMP 関数を示しています。

ステートメント	結果
select maketimestamp(date '2020-03-24', time '10:39:10');	24MAR2020:10:39:10
select maketimestamp(2013,10,30,13,39,10);	30OCT2013:13:39:10
select maketimestamp(2020,3,2,2,1,1);	02MAR2020:02:01:01

関連項目

関数:

- “DATEPART 関数” (441 ページ)
- “MAKEDATE 関数” (654 ページ)
- “MAKETIME 関数” (655 ページ)
- “TIMEPART 関数” (851 ページ)

MARGRCLPRC 関数

Margrabe モデルに基づいて、株式のヨーロピアンオプションのコール価格を計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

MARGRCLPRC(*X_1*, *t*, *X_2*, *sigma1*, *sigma2*, *rho12*)

必須引数

X_1
第 1 資産の価格を指定する正の非欠損値です。

要件 同じ単位で X_1 と X_2 を指定します。

データの種類 DOUBLE

t
有効期限までの時間を年単位で指定する非欠損値です。

データの種類 DOUBLE

X_2
第 2 資産の価格を指定する正の非欠損値です。

要件 同じ単位で X_2 と X_1 を指定します。

データの種類 DOUBLE

sigma1
第 1 資産のボラティリティを指定する非欠損の正の分数です。

データの種類 DOUBLE

sigma2
第 2 資産のボラティリティを指定する非欠損の正の分数です。

データの種類 DOUBLE

rho12
第 1 資産と第 2 資産の間の相関を指定します。詳細については、*SAS DS2 言語リファレンス*を参照してください。

第 1 資産と第 2 資産の間の相関$\rho_{x_1x_2}$を指定します。

範囲 -1 から 1 まで

データの種類 DOUBLE

詳細

MARGRCLPRC 関数は、Margrabe モデルに基づいて、株式のヨーロピアンオプションのコール価格を計算します。この関数は次の関係に基づきます。

$$\text{CALL} = X_1 N(d_1) - X_2 N(d_2)$$

引数

X$_1$
第 1 資産の価格を指定します。

X$_2$
第 2 資産の価格を指定します。

N
累積正規密度関数を指定します。

$$d_1 = \frac{\left(\ln\left(\frac{N_1}{N_2}\right) + \left(\frac{\sigma^2}{2}\right)t\right)}{\sigma\sqrt{t}}$$

$$d_2 = d_1 - \sigma\sqrt{t}$$

$$\sigma^2 = \sigma_{x1}^2 + \sigma_{x2}^2 - 2\rho_{x1,x2}\sigma_{x1}\sigma_{x2}$$

前述の式には次の引数が適用されます。

t
有効期限までの時間を指定します。

σ_{x1}^2
第 1 資産の分散を指定します。

σ_{x2}^2
第 2 資産の分散を指定します。

σ_{x1}
第 1 資産のボラティリティを指定します。

σ_{x2}
第 2 資産のボラティリティを指定します。

$\rho_{x1,x2}$
第 1 資産と第 2 資産の間の相関を指定します。

t=0 となる特別な場合には、次の式が真です。

$$\text{CALL} = \max((X_1 - X_2), 0)$$

注: この関数は、2 つの資産から利益配当金がないことを想定しています。

価格設定の基本については、“価格関数の使用” (*SAS 関数と CALL ルーチン: リファレンス*)を参照してください。

比較

MARGRCLPRC 関数は、Margrabe モデルに基づいて、株式のヨーロピアンオプションのコール価格を計算します。MARGRCLPRC 関数は、Margrabe モデルに基づいて、株式のヨーロピアンオプションのプット価格を計算します。これらの関数はスカラー値を返します。

例

次のステートメントは、MARGRCLPRC 関数を示しています。

ステートメント	結果
select margrclprc(15, .5, 13, .06, .05, 1);	2
select margrclprc(2, .25, 1, .3, .2, 1);	1

関連項目

関数:

- “MARGRPTPRC 関数” (660 ページ)

MARGRPTPRC 関数

Margrabe モデルに基づいて、株式のヨーロピアンオプションのプット価格を計算します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

MARGRPTPRC(*X_1*, *t*, *X_2*, *sigma1*, *sigma2*, *rho12*)

必須引数

X_1
第 1 資産の価格を指定する正の非欠損値です。

要件 同じ単位で X_1 と X_2 を指定します。

データの種類 DOUBLE

t
有効期限までの時間を年単位で指定する非欠損値です。

データの種類 DOUBLE

X_2
第 2 資産の価格を指定する正の非欠損値です。

要件 同じ単位で X_2 と X_1 を指定します。

データの種類 DOUBLE

sigma1
第 1 資産のボラティリティを指定する非欠損の正の分数です。

データの種類 DOUBLE

sigma2
第 2 資産のボラティリティを指定する非欠損の正の分数です。

データの種類 DOUBLE

rho12
第 1 資産と第 2 資産の間の相関を指定します。詳細については、*SAS DS2 言語リファレンス*を参照してください。

第 1 資産と第 2 資産の間の相関$\rho_{x_1y_1}$を指定します。

範囲 -1 から 1 まで

データの種類 DOUBLE

詳細

MARGRCLPRC 関数は、Margrabe モデルに基づいて、株式のヨーロピアンオプションのプット価格を計算します。この関数は次の関係に基づきます。

$$\text{PUT} = X_2 N(pd_1) - X_1 N(pd_2)$$

引数

X_1
第 1 資産の価格を指定します。

X_2
第 2 資産の価格を指定します。

N
累積正規密度関数を指定します。

$$pd_1 = \frac{\left(\ln\left(\frac{N_1}{N_2}\right) + \left(\frac{\sigma^2}{2}\right)t\right)}{\sigma\sqrt{t}}$$

$$pd_2 = pd_1 - \sigma\sqrt{t}$$

$$\sigma^2 = \sigma_{x1}^2 + \sigma_{x2}^2 - 2\rho_{x_1,\,x_2}\sigma_{x1}\sigma_{x2}$$

前述の式には次の引数が適用されます。

t
有効期限までの時間を年単位で指定する非欠損値です。

σ_{x1}^2
第 1 資産の分散を指定します。

σ_{x2}^2
第 2 資産の分散を指定します。

σ_{x1}
第 1 資産のボラティリティを指定します。

σ_{x2}
第 2 資産のボラティリティを指定します。

$\rho_{x_1,\,x_2}$
第 1 資産と第 2 資産の間の相関を指定します。

対応する CALL 関係を表示するには、“MARGRCLPRC 関数” (657 ページ)を参照してください。

t=0 となる特別な場合には、次の式が真です。

$$\text{PUT} = \max((X_2 - X_1), 0)$$

注: この関数は、2 つの資産から利益配当金がないことを想定しています。

価格設定の基本については、“価格関数の使用” (*SAS 関数と CALL ルーチン: リファレンス*)を参照してください。

比較

MARGRCLPRC 関数は、Margrabe モデルに基づいて、株式のヨーロピアンオプションのプット価格を計算します。MARGRCLPRC 関数は、Margrabe モデルに基づいて、株式のヨーロピアンオプションのコール価格を計算します。これらの関数はスカラー値を返します。

例

次のステートメントは、MARGRPTPRC 関数を示しています。

ステートメント	結果
select margrptprc(2, .25, 3, .06, .2, 1);	1.0000000000973
select margrptprc(3, .25, 4, .05, .3, 1);	1.00157624907712

関連項目

関数:

- “MARGRCLPRC 関数” (657 ページ)

MAX 関数

列の最大値を返します。

カテゴリ: 集計
記述統計

返されるデータの種類: 式と同じデータ型

構文

MAX([DISTINCT] *expression*)

必須引数

expression
有効な SQL 式を指定します。

データの種類 すべてのデータ型がサポートされています。

参照項目: “<sql-expression> 式” (918 ページ)
“FedSQL 式” (46 ページ)

オプション引数

DISTINCT
集計計算から重複行を削除します。

詳細

MAX 関数は、null 値と SAS 欠損値を無視します。

集計関数を使用して、FROM 句にリストされているテーブル全体、または GROUP BY 句で指定されている各グループのデータの統計的要約を生成できます。GROUP BY 句は、指定された 1 つまたは複数の列でデータをグループ化します。GROUP BY 句を使用する場合、SELECT 句または HAVING 句の集計関数は、各グループのデータを要約する方法を FedSQL に指示します。FedSQL は、グループごとに個別に集計関数を計算します。GROUP BY を省略すると、テーブルまたはビュー内のすべての行が単一のグループと見なされます。

比較

MIN 関数は、列の最小値を返します。MAX 関数は、列の最大値を返します。

例

テーブル: DENSITIES (1260 ページ)

次のステートメントは、MAX 関数を示しています。

ステートメント	結果
select max(population) from densities;	34248705

関連項目

関数:

- “MIN 関数” (668 ページ)

SELECT ステートメント句:

- “SELECT 句” (980 ページ)
- “GROUP BY 句” (993 ページ)
- “HAVING 句” (995 ページ)

MDY 関数

月、日および年の値から SAS 日付値を返します。

カテゴリ: 日付と時間

返されるデータの種類: DOUBLE

構文

MDY(*month*, *day*, *year*)

必須引数

month
1 から 12 までの整数を表す数値式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

day
1 から 31 までの整数を表す数値式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

year
2 桁または 4 桁の年を表す数値式を指定します。YEARCUTOFF=システムオプションは、2 桁の日付の年の値を定義します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

例

次のステートメントは、MDY 関数を示しています。

ステートメント	結果
select put(mdy(8,27,12),date7.);	27AUG12
select put(mdy(8,27,90),date7.);	27AUG90

関連項目

概念:

- “FedSQL の日付と時間” (59 ページ)

関数:

- “DAY 関数” (443 ページ)
- “MONTH 関数” (676 ページ)
- “YEAR 関数” (887 ページ)

MEAN 関数

非 null または非欠損引数の算術平均(平均)を返します。

カテゴリ:	記述統計
返されるデータの種類:	DOUBLE

構文

MEAN(*expression-1*[, ...*expression-n*])

必須引数

expression
数値に評価される有効な式を指定します。

要件	少なくとも 1 つの非 null または非欠損の引数が必要です。それがない場合は、関数から null または欠損値が返されます。
データの種類	DOUBLE
参照項目:	“<sql-expression> 式” (918 ページ) “FedSQL 式” (46 ページ)

比較

GEOMEAN 関数が幾何平均を返し、HARMEAN 関数が調和平均を返すのに対し、MEAN 関数は算術平均(平均)を返します。

例

次のステートメントは、MEAN 関数を示しています。

ステートメント	結果
select mean(2,.,.,6);	4
select mean(1,2,3,2);	2

関連項目

関数:

- “GEOMEAN 関数” (559 ページ)
- “GEOMEANZ 関数” (560 ページ)
- “HARMEAN 関数” (562 ページ)
- “HARMEANZ 関数” (564 ページ)
- “MEDIAN 関数” (667 ページ)

MEDIAN 関数

中央値を返します。

カテゴリ: 記述統計

返されるデータの種類: DOUBLE

構文

MEDIAN(*expression-1*[, ...*expression-n*])

必須引数

expression
数値に評価される有効な式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

MEDIAN 関数は、非欠損値または非 null 値の中央値を返します。すべての引数が欠損値または null 値の場合、結果は欠損値または null 値になります。

注: MEDIAN 関数で使用される式は、Base SAS プロシジャガイド: 統計プロシジャに説明されている、PROC UNIVARIATE で使用される式と同じです。詳細については、SAS Elementary Statistics Procedures を参照してください。

比較

MEDIAN 関数が非欠損値または非 null 値の中央値を返すのに対し、MEAN 関数は算術平均(平均)を返します。

例

次のステートメントは、MEDIAN 関数を示しています。

ステートメント	結果
select median(2, 4, 1, 3);	2.5
select median(5, 8, 0, 3, 4);	4

関連項目

関数:

- “MEAN 関数” (666 ページ)

MIN 関数

式の最小値を返します。

カテゴリ: 集計
記述統計

返されるデータの種類: 式と同じデータ型

構文

MIN([DISTINCT] *expression*)

必須引数

expression
有効な SQL 式を指定します。

データの種類 すべてのデータ型がサポートされています。

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

オプション引数

DISTINCT
集計計算から重複行を削除します。

詳細

MIN 関数は、null 値と SAS 欠損値を無視します。

集計関数を使用して、FROM 句にリストされているテーブル全体、または GROUP BY 句で指定されている各グループのデータの統計的要約を生成できます。GROUP BY 句は、指定された 1 つまたは複数の列でデータをグループ化します。GROUP BY 句を使用する場合、SELECT 句または HAVING 句の集計関数は、各グループのデータを要約する方法を FedSQL に指示します。FedSQL は、グループごとに個別に集計関数を計算します。GROUP BY を省略すると、テーブルまたはビュー内のすべての行が単一のグループと見なされます。

比較

MAX 関数は、列の最大値を返します。MIN 関数は、列の最小値を返します。

例

テーブル: DENSITIES (1260 ページ)

次のステートメントは、MIN 関数を示しています。

ステートメント	結果
select min(density) from densities;	6.154657

関連項目

関数:

- “MAX 関数” (663 ページ)

SELECT ステートメント句:

- “SELECT 句” (980 ページ)
- “GROUP BY 句” (993 ページ)
- “HAVING 句” (995 ページ)

MINUTE 関数

時間または日時値から分を返します。

カテゴリ: 日付と時間

返されるデータの種類: TINYINT

構文

MINUTE(*time* | *datetime*)

オプション引数

time
時間値を表す有効な式を指定します。

データの種類 TIME

参照項目: “式と述語の概要” (897 ページ)

datetime
日時値を表す有効な式を指定します。

データの種類 TIMESTAMP

参照項目: “式と述語の概要” (897 ページ)

例

テーブル: CUSTONLINE (1259 ページ)

次のステートメントは、MINUTE 関数を示しています。

ステートメント	結果
select minute(endtime) from custonline;	5 21 4 25 47 6 5 15 35 6
select minute(current_time);	16
select minute(localtimestamp);	16

関連項目

- “FedSQL の日付と時間” (59 ページ)

関数:

- “DAY 関数” (443 ページ)
- “HOUR 関数” (570 ページ)
- “MONTH 関数” (676 ページ)
- “SECOND 関数” (824 ページ)
- “YEAR 関数” (887 ページ)

MOD 関数

第 1 引数を第 2 引数で除算したときの(最も期待しない浮動小数点の結果を避けるためにファジー化した)余りを返します。

カテゴリ:	数学
返されるデータの種類:	DOUBLE

構文

MOD(*dividend-expression*, *divisor-expression*)

必須引数

dividend-expression
数値に評価される有効な式である被除数を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

divisor-expression
数値に評価される有効な式である除数を指定します。

制限事項 *divisor-expression* を 0 にすることはできません

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

MOD 関数は、*dividend-expression* を *divisor-expression* で除算したときの余りを返します。結果がゼロ以外のとき、結果は第 1 引数と同じ符号になります。第 2 引数の符号は無視されます。

次の条件の両方を満たす場合、MOD 関数で実行される計算は厳密になります。

- 両方の引数が厳密な整数である。
- すべての整数が、厳密に 8 バイトの浮動小数点で表されるいずれかの引数より小さい。

前述の条件のいずれかを満たしていない場合、浮動小数点計算で小さな数値誤差が発生する可能性があります。この場合、次のようになります。

- 余りが 0 か第 2 引数の値にとても近いと、MOD はゼロを返す。
- 約 3 桁以上の精度で余りを計算できなければ、MOD は null 値または欠損値を返す。この場合、ログにエラーメッセージが書き込まれる。

比較

MOD 関数と MODZ 関数の比較を次に示します。

- MOD 関数は、ファジーと呼ばれる追加の計算を実行し、厳密なゼロを返す(ファジーを実行しなければ数値誤差のため結果はゼロにならない)。
- MODZ 関数はファジーを実行しない。
- 約 3 桁以上の精度で余りを計算できなければ、MOD 関数と MODZ 関数はどちらも null 値または欠損値を返します。

例

次のステートメントは、MOD 関数を示しています。

ステートメント	結果
select mod(10,3);	1
select mod(.35,-.1);	0.05
select mod(17,3);	2
select mod(.3,-.9);	0.3

関連項目

関数:

- “MODZ 関数” (673 ページ)
- “INTZ 関数” (619 ページ)

MODZ 関数

第 1 引数を第 2 引数で除算したときの余りを、ゼロファジーを使用して返します。

カテゴリ: 数学

返されるデータの種類: DOUBLE

構文

MODZ(*dividend-expression*, *divisor-expression*)

必須引数

dividend-expression
数値に評価される有効な式である被除数を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

divisor-expression
数値に評価される有効な式である除数を指定します。

制限事項 *divisor-expression* を 0 にすることはできません

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

MODZ 関数は、*dividend-expression* を *divisor-expression* で除算したときの余りを返します。結果がゼロ以外のとき、結果は第 1 引数と同じ符号になります。第 2 引数の符号は無視されます。

次の条件の両方を満たす場合、MODZ 関数で実行される計算は厳密になります。

- 両方の引数が厳密な整数である。
- すべての整数が、厳密に 8 バイトの浮動小数点で表されるいずれかの引数より小さい。

前述の条件のいずれかを満たしていない場合、浮動小数点計算で小さな数値誤差が発生する可能性があります。たとえば、厳密算術を使用して結果がゼロのとき、MODZ は正の極小値または第 2 引数より若干小さい値を返す場合があります。

比較

MODZ 関数と MOD 関数の比較を次に示します。

- MODZ 関数はファジーを実行しない。
- MOD 関数は、ファジーを実行し、厳密なゼロを返す(ファジーを実行しなければ数値誤差のため結果はゼロにならない)。
- 約 3 桁以上の精度で余りを計算できなければ、MODZ 関数と MOD 関数はどちらも null 値または欠損値を返します。

例

例 1

次のステートメントは、MOD 関数を示しています。

ステートメント	結果
select mod(10,3);	1
select mod(.35,-.1);	0.05
select mod(17,3);	2
select mod(.3,-.9);	0.3

例 2

次のステートメントは、MODZ 関数を示しています。

ステートメント	結果
select modz(10,3);	1
select modz(.35,-.1);	0.05
select modz(17,3);	2
select modz(.3,-.9);	0.3

関連項目

関数:

- “INTZ 関数” (619 ページ)
- “MOD 関数” (671 ページ)

MONTH 関数

日付または日時値から月を数値で返します。

カテゴリ: 日付と時間

返されるデータの種類: TINYINT

構文

MONTH(*date* | *datetime*)

オプション引数

date
日付値を表す有効な式を指定します。

データの種類 DATE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

datetime
日時値を表す有効な式を指定します。

データの種類 TIMESTAMP

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

例

テーブル: CUSTONLINE (1259 ページ)

次のステートメントは、MONTH 関数を示しています。

ステートメント	結果
select month(endtime) from custonline;	9 10 10 11 12 1

ステートメント	結果
	1 2 3 3
select month(current_time);	10

関連項目

- “FedSQL の日付と時間” (59 ページ)

関数:

- “DAY 関数” (443 ページ)
- “HOUR 関数” (570 ページ)
- “MINUTE 関数” (670 ページ)
- “SECOND 関数” (824 ページ)
- “YEAR 関数” (887 ページ)

MORT 関数

割賦返済パラメーターを返します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

MORT(*a*, *p*, *r*, *n*)

必須引数

a

初期の金額に評価される有効な式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

p
定期的支払いに評価される有効な式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

r
分数で表す定期的な利率に評価される有効な式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

n
複利期間数に評価される有効な式を指定します。

範囲 $n \geq 0$

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

結果の計算

MORT 関数は毎期間複利計算される固定利率の割賦返済計算から 4 つの引数のリスト内の欠損引数を返します。これらの引数には次の式の関係があります。

$$p = \frac{ar(1+r)^n}{(1+r)^n - 1}$$

引数は 1 つを欠損値とする必要があります。値は残りの 3 つから計算されます。結果を変換して丸めた数字にする調整は行われません。

結果を計算するときの制限事項

次の引数の組み合わせのいずれかに該当する場合、MORT 関数は無効な引数というメモを SAS ログに返し、_ERROR_を 1 に設定します。

- rate < –1 or n < 0
- principal <= 0 または payment <= 0 または n <= 0
- principal <= 0 または payment <= 0 または rate <= –1
- principal * rate > payment

- principal > payment * n

例

次のプログラムでは、$50,000 を毎月複利計算される年利 10%の 30 年で借入します。

```
select mort(50000, . , .10/12, 30*12);
```

返される値は 438.79(丸め値)です。第 2 引数は欠損に設定されています。これは定期的支払いが計算されることを示します。名目年利 10%は、月利 0.10/12 に変換されています。rate は分数(パーセントではない)で表す複利計算期間当たりの利率です。30 年は 360 か月に変換されます。

N 関数

非 null または非欠損の数値の数を返します。

カテゴリ:	特殊
返されるデータの種類:	INTEGER

構文

N(*expression* [, ...*expression*])

必須引数

expression
数値に評価される有効な式を指定します。

要件	少なくとも 1 つの引数が必要です。
データの種類	DECIMAL、DOUBLE、NUMERIC
参照項目:	“<sql-expression> 式” (918 ページ)
	“FedSQL 式” (46 ページ)

詳細

null 値は欠損値に変換され、欠損値として数えられます。

比較

N 関数は非 null 値と非欠損値を数えますが、NMISS 関数は欠損値を数えます。N 関数には数値引数が必要です。

例

次のステートメントは、N 関数を示しています。

ステートメント	結果
select n(1,0,.,2,5,.);	4
select n(1,2);	2

関連項目

関数:

- “NMISS 関数” (682 ページ).

NETPV 関数

現在正味価値をパーセントで返します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

NETPV(*r*, *freq*, *c0*, *c1*, ..., *cn*)

必須引数

r

指定したベース期間の利率を分数で表した数値。

範囲 *r* >= 0

データの種類 DOUBLE

freq
利率 *r* で指定したベース期間中の支払い回数を表す数値。

範囲 *freq* > 0

データの種類 DOUBLE

注 *freq* = 0 は、連続割引を許可するフラグです。

c0, c1 ,..., cn
時間 0,1,...n に発生する現金支出(支払い)または現金流入(収入)を表す数値のキャッシュフローです。これらのキャッシュフローは、等間隔の期間開始時の値であると想定されます。負の値は支払い、正の値は収入、値 0 はその時点でキャッシュフローがないことを表します。*c0* 引数と *c1* 引数は必須です。

データの種類 DOUBLE

詳細

NETPV 関数は、指定したベース期間中に利率 *r* を使用して、一連の現金支払い *c0*,*c1*, ...,*cn* に対する時間 0 の正味現在価値を返します。引数 *freq*>0 は、指定ベース期間で発生する支払い数を表します。

正味現在価値は次の式で求められます。

$$\mathrm{NETPV}(r, freq, c_0, c_1, ..., c_n) = \sum_{i=0}^{n} c_i x^i$$

前述の式には次の関係が適用されます。

$$x = \begin{cases} \frac{1}{(1+r)^{(1/freq)}} & freq > 0 \\ \varepsilon^{-r} & freq = 0 \end{cases}$$

支払いの欠損値は 0 値として扱われます。*freq*>0 のとき、利率 *r* は指定したベース期間の実効金利です。四半期金利 4%(ベース期間は 3 か月)の毎月現金支払いで計算するには、*freq* を 3、*r* を.04 に設定します。

freq が 0 の場合は、連続割引とみなされます。ベース期間は 2 つの連続する支払い間の時間間隔で、利率 *r* は名目金利です。

名目年利 11%、連続割引、毎月支払いで計算するには、*freq* を 0、*r* を.11/12 に設定します。

例

翌 6 年間にわたり$200、$300、$400 の年 2 回支払いおよび年間割引率 10%を返す初期投資$500 では、投資の現在正味価値は次のように表すことができます。

```
select netpv(.10,.5,-500,200,300,400);
```

返される値は 95.982864829 です。

関連項目

関数:

- “NPV 関数” (704 ページ)

NMISS 関数

式内の null 値または SAS 欠損値の数を返します。

カテゴリ: 集計
スカラー

返されるデータの種類: BIGINT

構文

Aggregate:

NMISS([DISTINCT] *expression*)

Scalar:

NMISS(*expression*[,...*expression*])

オプション引数

expression (aggregate)
評価結果が文字列になる SQL 式を指定します。

データの種類	CHAR
返されるデータの種類	BIGINT

expression (scalar)
評価結果が数値になる SQL 式を指定します。複数の式はカンマで区切る必要があります。

データの種類	DECIMAL、DOUBLE、NUMERIC
返されるデータの種類	INTEGER
参照項目:	“<sql-expression> 式” (918 ページ)
	“FedSQL 式” (46 ページ)

DISTINCT
(オプション)集計計算から重複行を削除します。

詳細

NMISS は、null または SAS 欠損値の数を返します。FedSQL を ANSI モードで実行すると、ANSI null 値が SAS 欠損値に変換されます。

集計関数として使用すると、NMISS は 1 つの文字列で見つかった null または欠損値の数を返します。集計関数を使用して、FROM 句にリストされているテーブル全体、または GROUP BY 句で指定されている各グループのデータの統計的要約を生成できます。GROUP BY 句は、指定された 1 つまたは複数の列でデータをグループ化します。GROUP BY 句を使用する場合、SELECT 句または HAVING 句の集計関数は、各グループのデータを要約する方法を FedSQL に指示します。FedSQL は、グループごとに個別に集計関数を計算します。GROUP BY を省略すると、テーブルまたはビュー内のすべての行が単一のグループと見なされます。

スカラー関数として使用すると、NMISS は、指定された式の中から null または欠損値の数を返します。

比較

NMISS 関数は、null または SAS 欠損値の数を返します。N 関数は、非 null および非欠損値の数を返します。

例

例 1: 集計演算

テーブル: WORLDTEMPS (1277 ページ)

次のステートメントは、NMISS を集計関数として使用する方法を示しています。

ステートメント	結果
select nmiss(AvgHigh) from worldtemps;	1

例 2: スカラー演算

次のステートメントは、NMISS をスカラー関数として使用する方法を示しています。

ステートメント	結果
select nmiss(1,2,.,3,.);	2

関連項目

- “FedSQL が null および SAS 欠損値を処理する方法” (21 ページ)

関数:

- “N 関数” (679 ページ)

SELECT ステートメント句:

- “SELECT 句” (980 ページ)
- “GROUP BY 句” (993 ページ)
- “HAVING 句” (995 ページ)

NOMRATE 関数

名目年利を返します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

NOMRATE(*compounding-interval*, *rate*)

必須引数

compounding-interval
SAS 間隔です。この値は、返される値が複利計算される頻度を表します。

データの種類 CHAR

rate
数値です。*rate* は各期間複利計算される実効年利(パーセント)です。

データの種類 DOUBLE

詳細

NOMRATE 関数は名目年利を返します。NOMRATE は実効年利に対応する名目年利を計算します。

次の詳細は NOMRATE 関数に適用されます。

- rate の値は–99 以上にする必要があります。
- 実効金利と複利計算間隔を検討するとき、*compounding-interval* が 'CONTINUOUS'であれば、NOMRATE によって返される値は $\log_e(1+[rate/100])$に等しくなります。

  compounding-interval が'CONTINUOUS'でなく、1 年に *m* 個の間隔が発生する場合、NOMRATE によって返される値は次に等しくなります。

  $$m\left((1+\frac{rate}{100})^{\frac{1}{m}}-1\right)$$

- *compounding-interval* で有効な値は次のとおりです。
  - 'CONTINUOUS'
  - 'DAY'
  - 'SEMIMONTH'
  - 'MONTH'
  - 'QUARTER'
  - 'SEMIYEAR'
  - 'YEAR'
- 間隔が'DAY'の場合、*m*=365 です。

例

- 毎月複利で実効金利が 10%の場合、対応する名目金利は次のように表すことができます。

  select NOMRATE('MONTH', 10);

  effective_rate1=9.56896851468451

- 四半期複利で実効金利が 10%の場合、対応する名目金利は次のように表すことができます。

  select NOMRATE('QUARTER', 10);

  effective_rate2=9.64547563377804

NOTALNUM 関数

文字列から英数字でない文字を検索し、その文字が見つかった最初の文字位置を返します。

カテゴリ:	文字
返されるデータの種類:	DOUBLE

構文

NOTALNUM('*expression*'[, *start*])

必須引数

expression
文字列に評価されるか強制変換できる有効な式を指定します。

データの種類 CHAR、NCHAR、VARCHAR、NVARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

オプション引数

start
数値に評価されるか強制変換できる有効な式を指定し、検索を開始する文字の位置と検索の方向を指定します。

データの種類 DOUBLE

詳細

NOTALNUM 関数の結果は、有効になっている変換テーブル(“TRANTAB= システムオプション” (*SAS 各国語サポート(NLS): リファレンスガイド*)を参照)に直接依存し、ENCODING および LOCALE システムオプションに間接的に依存します。

NOTALNUM 関数は、文字列から数字、大文字または小文字でない文字の最初の出現を検索します。対象の文字が検出されると、NOTALNUM は文字列内の該当文字の位置を返します。対象の文字が検出されない場合、NOTALNUM は値 0 を返します。

引数を 1 つのみ使用すると、NOTALNUM は文字列の先頭から検索を開始します。2 つの引数を使用する場合、第 2 引数 *start* の絶対値で検索の開始位置を指定します。検索方向は次のように決まります。

- *start* の値が正の場合、検索は右方向に進みます。
- *start* の値が負の場合、検索は左方向に進みます。
- *start* の値が文字列の負の長さよりも小さい場合、文字列の末尾から検索が開始されます。

次のいずれかの条件が満たされると、NOTALNUM は値 0 を返します。

- 検索文字が見つからない。
- *start* の値が文字列の長さよりも大きい。
- *start* の値が 0 になっている。

比較

NOTALNUM 関数は、文字列から英数字でない文字を検索します。ANYALNUM 関数は、文字列から英数字を検索します。

例

次のステートメントは、NOTALNUM 関数を示しています。

ステートメント	結果
select notalnum('abc123, x');	7 (position of comma)
select notalnum('+abcd');	1 (position of plus sign)
select notalnum('789ab');	0
select notalnum('myname@gmail.com');	7 (position of @)
select notalnum('531-66-1234');	4 (position of first dash)
select notalnum('531-66-1234', -99);	7 (position of last dash in string) */

関連項目

関数:

- “ANYALNUM 関数” (283 ページ)

NOTALPHA 関数

文字列からアルファベットでない文字を検索し、その文字が見つかった最初の文字位置を返します。

カテゴリ: 文字

返されるデータの種類: DOUBLE

構文

NOTALPHA('*expression*'[, *start*])

必須引数

expression
文字列に評価されるか強制変換できる有効な式を指定します。

データの種類 CHAR、NCHAR、VARCHAR、NVARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

オプション引数

start
数値に評価されるか強制変換できる有効な式を指定し、検索を開始する文字の位置と検索の方向を指定します。

データの種類 DOUBLE

詳細

NOTALPHA 関数の結果は、有効になっている変換テーブル(“TRANTAB= システムオプション” (*SAS 各国語サポート(NLS): リファレンスガイド*)を参照)に直接依存し、ENCODING および LOCALE システムオプションに間接的に依存します。

NOTALPHA 関数は、文字列から大文字または小文字でない文字の最初の出現を検索します。対象の文字が検出されると、NOTALPHA は文字列内の該当文字の位置を返します。対象の文字が検出されない場合、NOTALPHA は値 0 を返します。

引数を 1 つのみ使用すると、NOTALPHA は文字列の先頭から検索を開始します。2 つの引数を使用する場合、第 2 引数 *start* の絶対値で検索の開始位置を指定します。検索方向は次のように決まります。

- *start* の値が正の場合、検索は右方向に進みます。
- *start* の値が負の場合、検索は左方向に進みます。
- *start* の値が文字列の負の長さよりも小さい場合、文字列の末尾から検索が開始されます。

次のいずれかの条件が満たされると、NOTALPHA は値 0 を返します。

- 検索文字が見つからない。
- *start* の値が文字列の長さよりも大きい。
- *start* の値が 0 になっている。

比較

NOTALPHA 関数は、文字列からアルファベットでない文字を検索します。
ANYALPHA 関数は、文字列からアルファベットを検索します。

例

次のステートメントは、NOTALPHA 関数を示しています。

ステートメント	結果
select notalpha('123abc456');	1 (position of 1)
select notalpha('abcd1234');	5 (position of 1)
select notalpha('abcd');	0
select notalpha('789abc456',-99);	9 (position of 6)
select notalpha('abc$123');	4 (position of dollar sign)

関連項目

関数:

- “ANYALPHA 関数” (285 ページ)

NOTDIGIT 関数

文字列から数字でない文字を検索し、その文字が見つかった最初の文字位置を返します。

カテゴリ: 文字

返されるデータの種類: DOUBLE

構文

NOTDIGIT('*expression*'[, *start*])

必須引数

expression
文字列に評価されるか強制変換できる有効な式を指定します。

データの種類 CHAR、NCHAR、VARCHAR、NVARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

オプション引数

start
数値に評価されるか強制変換できる有効な式を指定し、検索を開始する文字の位置と検索の方向を指定します。

データの種類 DOUBLE

詳細

NOTDIGIT 関数の結果は、有効になっている変換テーブル(“TRANTAB= システムオプション” (*SAS 各国語サポート(NLS): リファレンスガイド*)を参照)に直接依存し、ENCODING および LOCALE システムオプションに間接的に依存します。

NOTDIGIT 関数は、文字列から数字でない文字の最初の出現を検索します。対象の文字が検出されると、NOTDIGIT は文字列内の該当文字の位置を返します。対象の文字が検出されない場合、NOTDIGIT は値 0 を返します。

引数を 1 つのみ使用すると、NOTDIGIT は文字列の先頭から検索を開始します。2 つの引数を使用する場合、第 2 引数 *start* の絶対値で検索の開始位置を指定します。検索方向は次のように決まります。

- *start* の値が正の場合、検索は右方向に進みます。
- *start* の値が負の場合、検索は左方向に進みます。
- *start* の値が文字列の負の長さよりも小さい場合、文字列の末尾から検索が開始されます。

次のいずれかの条件が満たされると、NOTDIGIT は値 0 を返します。

- 検索文字が見つからない。
- *start* の値が文字列の長さよりも大きい。
- *start* の値が 0 になっている。

比較

NOTDIGIT 関数は、文字列から数字でない文字を検索します。ANYDIGIT 関数は、文字列から数字を検索します。

例

注: この例では、DS2 ステートメントは PROC DS2 を使用して SAS に送信されます。

次のステートメントは、NOTDIGIT 関数を示しています。

ステートメント	結果
select notdigit('123abc456');	4 (position of a)
select notdigit('1234');	0
select notdigit('123abc456',-99);	6 (position of last non-digit in string)
select notdigit('$12,300');	1 (position of dollar sign)
select notdigit('$12,300', -99);	4 (position of comma)

関連項目

関数:

- “ANYDIGIT 関数” (287 ページ)

NOTFIRST 関数

VALIDVARNAME=V7 の SAS 変数名として無効な開始文字を文字列から検索し、その文字が見つかった最初の文字位置を返します。

カテゴリ: 文字

返されるデータの種類: DOUBLE

構文

NOTFIRST('*expression*'[, *start*])

必須引数

expression
文字列に評価されるか強制変換できる有効な式を指定します。

データの種類 CHAR、NCHAR、VARCHAR、NVARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

オプション引数

start
数値に評価されるか強制変換できる有効な式を指定し、検索を開始する文字の位置と検索の方向を指定します。

データの種類 DOUBLE

詳細

NOTFIRST 関数は、TRANTAB、ENCODING または LOCALE システムオプションに依存しません。

NOTFIRST 関数は、VALIDVARNAME=V7 の SAS 変数名の開始文字として有効でない文字のうち最初に出現する文字を文字列から検索します。これらの文字は、アンダースコア(_)、大文字または小文字の英文字以外の文字です。対象の文字が検出されると、NOTFIRST は文字列内の該当文字の位置を返します。対象の文字が検出されない場合、NOTFIRST は値 0 を返します。

引数を 1 つのみ使用すると、NOTFIRST は文字列の先頭から検索を開始します。2 つの引数を使用する場合、第 2 引数 *start* の絶対値で検索の開始位置を指定します。検索方向は次のように決まります。

- *start* の値が正の場合、検索は右方向に進みます。
- *start* の値が負の場合、検索は左方向に進みます。
- *start* の値が文字列の負の長さよりも小さい場合、文字列の末尾から検索が開始されます。

次のいずれかの条件が満たされると、NOTFIRST は値 0 を返します。

- 検索文字が見つからない。
- *start* の値が文字列の長さよりも大きい。
- *start* の値が 0 になっている。

比較

NOTFIRST 関数は、VALIDVARNAME=V7 の SAS 変数名の開始文字として有効でない文字のうち最初に出現する文字を文字列から検索します。ANYFIRST 関数は、VALIDVARNAME=V7 の SAS 変数名の開始文字として有効な文字のうち最初に出現する文字を文字列から検索します。

例

次のステートメントは、NOTFIRST 関数を示しています。

ステートメント	結果
select notfirst('123aBc4D6');	1 (position of 1)
select notfirst('123aBc4D6',-99);	9 (position of 6)
select notfirst('Bac123, x');	4 (position of 1)
select notfirst('x,&,y,z,20,!');	2 (position of ampersand)
select notfirst('x,&,y,z,20,!', -99);	12 (position of exclamation mark)

関連項目

関数:

- “ANYFIRST 関数” (288 ページ)

NOTLOWER 関数

文字列から小文字でない文字を検索し、その文字が見つかった最初の文字位置を返します。

カテゴリ: 文字

返されるデータの種類: DOUBLE

構文

NOTLOWER('*expression*'[, *start*])

必須引数

expression
文字列に評価されるか強制変換できる有効な式を指定します。

データの種類 CHAR、NCHAR、VARCHAR、NVARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

オプション引数

start
数値に評価されるか強制変換できる有効な式を指定し、検索を開始する文字の位置と検索の方向を指定します。

データの種類 DOUBLE

詳細

NOTLOWER 関数の結果は、有効になっている変換テーブル(“TRANTAB= システムオプション” (*SAS 各国語サポート(NLS): リファレンスガイド*)を参照)に直接依存し、ENCODING および LOCALE システムオプションに間接的に依存します。

NOTLOWER 関数は、文字列から小文字でない文字の最初の出現を検索します。対象の文字が検出されると、NOTLOWER は文字列内の該当文字の位置を返します。対象の文字が検出されない場合、NOTLOWER は値 0 を返します。

引数を 1 つのみ使用すると、NOTLOWER は文字列の先頭から検索を開始します。2 つの引数を使用する場合、第 2 引数 *start* の絶対値で検索の開始位置を指定します。検索方向は次のように決まります。

- *start* の値が正の場合、検索は右方向に進みます。
- *start* の値が負の場合、検索は左方向に進みます。
- *start* の値が文字列の負の長さよりも小さい場合、文字列の末尾から検索が開始されます。

次のいずれかの条件が満たされると、NOTLOWER は値 0 を返します。

- 検索文字が見つからない。
- *start* の値が文字列の長さよりも大きい。
- *start* の値が 0 になっている。

比較

NOTLOWER 関数は、文字列から小文字でない文字を検索します。ANYLOWER 関数は、文字列から小文字を検索します。

例

次のステートメントは、NOTLOWER 関数を示しています。

ステートメント	結果
select notlower('123aBc4D6');	1 (position of 1)

ステートメント	結果
select notlower('123aBc4D6',-99);	9 (position of 6)
select notlower('DAN BROWN');	1 (position of D)
select notlower('dan brown');	4 (position of space)
select notlower('mr. dan brown');	3 (position of period)

関連項目

関数:

- “ANYLOWER 関数” (291 ページ)

NOTNAME 関数

VALIDVARNAME=V7 の SAS 変数名として無効な文字を文字列から検索し、その文字が見つかった最初の文字位置を返します。

カテゴリ: 文字

返されるデータの種類: DOUBLE

構文

NOTNAME('*expression*'[, *start*])

必須引数

expression
文字列に評価されるか強制変換できる有効な式を指定します。

データの種類 CHAR、NCHAR、VARCHAR、NVARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

オプション引数

start
数値に評価されるか強制変換できる有効な式を指定し、検索を開始する文字の位置と検索の方向を指定します。

データの種類 DOUBLE

詳細

NOTNAME 関数は、TRANTAB、ENCODING または LOCALE システムオプションに依存しません。

NOTNAME 関数は、VALIDVARNAME=V7 の SAS 変数名で有効でない文字のうち最初に出現する文字を文字列から検索します。これらの文字は、アンダースコア(_)、数字、大文字または小文字の英文字以外の文字です。対象の文字が検出されると、NOTNAME は文字列内の該当文字の位置を返します。対象の文字が検出されない場合、NOTNAME は値 0 を返します。

引数を 1 つのみ使用すると、NOTNAME は文字列の先頭から検索を開始します。2 つの引数を使用する場合、第 2 引数 *start* の絶対値で検索の開始位置を指定します。検索方向は次のように決まります。

- *start* の値が正の場合、検索は右方向に進みます。
- *start* の値が負の場合、検索は左方向に進みます。
- *start* の値が文字列の負の長さよりも小さい場合、文字列の末尾から検索が開始されます。

次のいずれかの条件が満たされると、NOTNAME は値 0 を返します。

- 検索文字が見つからない。
- *start* の値が文字列の長さよりも大きい。
- *start* の値が 0 になっている。

比較

NOTNAME 関数は、VALIDVARNAME=V7 の SAS 変数名で有効でない文字のうち最初に出現する文字を文字列から検索します。ANYNAME 関数は、VALIDVARNAME=V7 の SAS 変数名で有効な文字のうち最初に出現する文字を文字列から検索します。

例

次のステートメントは、NOTNAME 関数を示しています。

ステートメント	結果
select notname('<&abc!');	1 (position of angle bracket)
select notname('&abc!');	1 (position of ampersand)
select notname('abc!>');	4 (position of exclamation mark)
select notname('789!>');	4 (position of exclamation mark)
select notname('abc!>', -99);	5 (position of angle bracket)

関連項目

関数:

- “ANYNAME 関数” (292 ページ)

NOTPUNCT 関数

文字列から句読文字でない文字を検索し、その文字が見つかった最初の文字位置を返します。

カテゴリ: 文字

返されるデータの種類: DOUBLE

構文

NOTPUNCT('*expression*'[, *start*])

必須引数

expression
文字列に評価されるか強制変換できる有効な式を指定します。

データの種類 CHAR、NCHAR、VARCHAR、NVARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

オプション引数

start
数値に評価されるか強制変換できる有効な式を指定し、検索を開始する文字の位置と検索の方向を指定します。

データの種類 DOUBLE

詳細

NOTPUNCT 関数の結果は、有効になっている変換テーブル(“TRANTAB= システムオプション” (*SAS 各国語サポート(NLS): リファレンスガイド*)を参照)に直接依存し、ENCODING および LOCALE システムオプションに間接的に依存します。

NOTPUNCT 関数は、文字列から句読文字でない文字の最初の出現を検索します。対象の文字が検出されると、NOTPUNCT は文字列内の該当文字の位置を返します。対象の文字が検出されない場合、NOTPUNCT は値 0 を返します。

引数を 1 つのみ使用すると、NOTPUNCT は文字列の先頭から検索を開始します。2 つの引数を使用する場合、第 2 引数 *start* の絶対値で検索の開始位置を指定します。検索方向は次のように決まります。

- *start* の値が正の場合、検索は右方向に進みます。
- *start* の値が負の場合、検索は左方向に進みます。
- *start* の値が文字列の負の長さよりも小さい場合、文字列の末尾から検索が開始されます。

次のいずれかの条件が満たされると、NOTPUNCT は値 0 を返します。

- 検索文字が見つからない。
- *start* の値が文字列の長さよりも大きい。
- *start* の値が 0 になっている。

比較

NOTPUNCT 関数は、文字列から句読文字でない文字を検索します。ANYPUNCT 関数は、文字列から句読文字を検索します。

例

次のステートメントは、NOTPUNCT 関数を示しています。

ステートメント	結果
select notpunct('abc123, x');	1 (position of a)

ステートメント	結果
select notpunct('!?:;,');	0
select notpunct('987&efg');	1 (position of 9)

関連項目

関数:

- “ANYPUNCT 関数” (294 ページ)

NOTSPACE 関数

空白文字ではない文字(空白、水平タブと垂直タブ、キャリッジリターン、改行、およびフォームフィード)を文字列で検索し、その文字が見つかった最初の文字位置を返します。

カテゴリ: 文字

返されるデータの種類: DOUBLE

構文

NOTSPACE('*expression*'[, *start*])

必須引数

expression
文字列に評価されるか強制変換できる有効な式を指定します。

データの種類 CHAR、NCHAR、VARCHAR、NVARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

オプション引数

start
数値に評価されるか強制変換できる有効な式を指定し、検索を開始する文字の位置と検索の方向を指定します。

データの種類 DOUBLE

詳細

NOTSPACE 関数の結果は、有効になっている変換テーブル(“TRANTAB= システムオプション” (*SAS 各国語サポート(NLS): リファレンスガイド*)を参照)に直接依存し、ENCODING および LOCALE システムオプションに間接的に依存します。

NOTSPACE 関数は、文字列から空白、水平タブ、垂直タブ、キャリッジリターン、ラインフィード、フォームフィードでない文字の最初の出現を検索します。対象の文字が検出されると、NOTSPACE は文字列内の該当文字の位置を返します。対象の文字が検出されない場合、NOTSPACE は値 0 を返します。

引数を 1 つのみ使用すると、NOTSPACE は文字列の先頭から検索を開始します。2 つの引数を使用する場合、第 2 引数 *start* の絶対値で検索の開始位置を指定します。検索方向は次のように決まります。

- *start* の値が正の場合、検索は右方向に進みます。
- *start* の値が負の場合、検索は左方向に進みます。
- *start* の値が文字列の負の長さよりも小さい場合、文字列の末尾から検索が開始されます。

次のいずれかの条件が満たされると、NOTSPACE は値 0 を返します。

- 検索文字が見つからない。
- *start* の値が文字列の長さよりも大きい。
- *start* の値が 0 になっている。

比較

NOTSPACE 関数は、文字列で最初に出現する空白、水平タブ、垂直タブ、キャリッジリターン、ラインフィード、フォームフィードでない文字を検索します。
ANYSPACE 関数は、文字列で最初に出現する空白、水平タブ、垂直タブ、キャリッジリターン、ラインフィード、フォームフィードの文字を検索します。

例

次のステートメントは、NOTSPACE 関数を示しています。

ステートメント	結果
select notspace('abc123, x');	1 (position of a)
select notspace('Ma ry Smith');	1 (position of M)
select notspace('Ma ry Smith',-99);	11 (position of h)

関連項目

関数:

- “ANYSPACE 関数” (296 ページ)

NOTUPPER 関数

文字列から大文字でない文字を検索し、その文字が見つかった最初の文字位置を返します。

カテゴリ: 文字

返されるデータの種類: DOUBLE

構文

NOTUPPER('*expression*'[, *start*])

必須引数

expression
文字列に評価されるか強制変換できる有効な式を指定します。

データの種類 CHAR、NCHAR、VARCHAR、NVARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

オプション引数

start
数値に評価されるか強制変換できる有効な式を指定し、検索を開始する文字の位置と検索の方向を指定します。

データの種類 DOUBLE

詳細

NOTUPPER 関数の結果は、有効になっている変換テーブル(“TRANTAB= システムオプション” (*SAS 各国語サポート(NLS): リファレンスガイド*)を参照)に直接依存し、ENCODING および LOCALE システムオプションに間接的に依存します。

NOTUPPER 関数は、文字列から大文字でない文字の最初の出現を検索します。対象の文字が検出されると、NOTUPPER は文字列内の該当文字の位置を返します。対象の文字が検出されない場合、NOTUPPER は値 0 を返します。

引数を 1 つのみ使用すると、NOTUPPER は文字列の先頭から検索を開始します。2 つの引数を使用する場合、第 2 引数 *start* の絶対値で検索の開始位置を指定します。検索方向は次のように決まります。

- *start* の値が正の場合、検索は右方向に進みます。
- *start* の値が負の場合、検索は左方向に進みます。
- *start* の値が文字列の負の長さよりも小さい場合、文字列の末尾から検索が開始されます。

次のいずれかの条件が満たされると、NOTUPPER は値 0 を返します。

- 検索文字が見つからない。
- *start* の値が文字列の長さよりも大きい。
- *start* の値が 0 になっている。

比較

NOTUPPER 関数は、文字列から大文字でない文字を検索します。ANYUPPER 関数は、文字列から大文字を検索します。

例

次のステートメントは、NOTUPPER 関数を示しています。

ステートメント	結果
select notupper('aBc4D');	1 (position of a)
select notupper('aBc4D', -99);	4 (position of 4)
select notupper('123aBc4D6');	1 (position of 1)
select notupper('123aBc4D6', 5);	6 (position of c)
select notupper('123aBc4D6', -5);	4 (position of a)
select notupper('ABCDE');	0

関連項目

関数:

- “ANYUPPER 関数” (298 ページ)

NOTXDIGIT 関数

文字列から 16 進文字でない文字を検索し、その文字が見つかった最初の文字位置を返します。

カテゴリ: 文字

返されるデータの種類: DOUBLE

構文

NOTXDIGIT('*expression*'[, *start*])

必須引数

expression
文字列に評価されるか強制変換できる有効な式を指定します。

データの種類 CHAR、NCHAR、VARCHAR、NVARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

オプション引数

start
数値に評価されるか強制変換できる有効な式を指定し、検索を開始する文字の位置と検索の方向を指定します。

データの種類 DOUBLE

詳細

NOTXDIGIT 関数は、文字列から数字、大文字または小文字の A、B、C、D、E、F でない文字の最初の出現を検索します。対象の文字が検出されると、NOTXDIGIT は文字列内の該当文字の位置を返します。対象の文字が検出されない場合、NOTXDIGIT は値 0 を返します。

引数を 1 つのみ使用すると、NOTXDIGIT は文字列の先頭から検索を開始します。2 つの引数を使用する場合、第 2 引数 *start* の絶対値で検索の開始位置を指定します。検索方向は次のように決まります。

- *start* の値が正の場合、検索は右方向に進みます。

- *start* の値が負の場合、検索は左方向に進みます。
- *start* の値が文字列の負の長さよりも小さい場合、文字列の末尾から検索が開始されます。

次のいずれかの条件が満たされると、NOTXDIGIT は値 0 を返します。

- 検索文字が見つからない。
- *start* の値が文字列の長さよりも大きい。
- *start* の値が 0 になっている。

比較

NOTXDIGIT 関数は、文字列から 16 進文字でない文字を検索します。ANYXDIGIT 関数は、文字列から 16 進文字を検索します。

例

次のステートメントは、NOTXDIGIT 関数を示しています。

ステートメント	結果
select notxdigit('123aBc4D6');	0 (no special characters)
select notxdigit('abcd456');	0 (no special characters)
select notxdigit('789ab!',-99);	6 (position of exclamation mark)
select notxdigit('&789ab!');	1 (position of ampersand)

関連項目

関数:

- “ANYXDIGIT 関数” (300 ページ)

NPV 関数

パーセントで表す利率を使用して現在正味価値を返します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

NPV(*r*, *freq*, *c0*, *c1*, ..., *cn*)

必須引数

r
指定したベース期間の利率をパーセントで表した数値。

データの種類 DOUBLE

freq
利率 *r* を使用して指定したベース期間中の支払い回数を表す数値。

範囲 *freq* > 0

データの種類 DOUBLE

注 *freq* = 0 は、連続割引を許可するフラグです。

c0, c1,..., cn
時間 0,1,...n に発生する現金支出(支払い)または現金流入(収入)を表す数値のキャッシュフローです。これらのキャッシュフローは、等間隔の期間開始時の値であると想定されます。負の値は支払い、正の値は収入、値 0 はその時点でキャッシュフローがないことを表します。*c0* 引数と *c1* 引数は必須です。

データの種類 DOUBLE

比較

NPV 関数は、*r* 引数をパーセントで指定する点を除き、NETPV と同じです。

例

次のステートメントは、NPV 関数を示しています。

ステートメント	結果
select npv(1,2,3,4,5,6,7,8);	32.50727639

関連項目

関数:

- “NETPV 関数” (680 ページ)

NWKDOM 関数

指定した月および年の *n* 番目に発生する曜日の日付を返します。

カテゴリ: 日付と時間

返されるデータの種類: DOUBLE

構文

NWKDOM(*n*, *weekday*, *month*, *year*)

必須引数

n
指定した日を含む月の週を数値で指定します。

範囲 1-5

データの種類 DOUBLE

ヒント *N*=5 は、指定した日がその月の最終週に発生することを示します。*n*=4 と *n*=5 の結果が同じになる場合もあります。

weekday
週の曜日に対応する数値を指定します。

範囲 1-7

データの種類 DOUBLE

ヒント 日曜日は、週の最初の日として見なされ、*weekday* の値は 1 になります。

month
年の月に対応する数値を指定します。

範囲 1-12

データの種類 DOUBLE

year
4 桁のカレンダー年を指定します。

データの種類 DOUBLE

詳細

NWKDOM 関数は、指定した月および年の *n* 番目の曜日の SAS 日付値を返します。DATE9.形式などの有効な SAS 日付形式を使用して、カレンダー日付を表示します。月の最後に発生する特定の曜日の場合、*n*=5 を指定できます。

n=5 と *n*=4 の結果が同じになる場合もあります。要求された曜日がその月に 4 回しか発生しない場合にこのような結果になります。たとえば、1 月の初日が日曜日の場合、日曜日、月曜日および火曜日は 5 回発生しますが、水曜日、木曜日、金曜日、土曜日は 4 回しか発生しません。この場合、水曜日、木曜日、金曜日、土曜日に対して *n*=5 または *n*=4 を指定すると、結果が同じになります。

その年がうるう年でない場合、2 月は 28 日間あり、週の各曜日は 4 回発生します。この場合、すべての曜日で *n*=5 と *n*=4 の結果が同じになります。

比較

NWKDOM 関数では、*weekday* の値は、日曜日から始まる週の曜日の数値に対応しています。この値は、WEEKDAY 関数で使用される値と同じで、日曜日=1 (以下同様)になります。*month* の値は、1 月から始まる年の月の数値に対応します。この値は、MONTH 関数で使用される値と同じで、1 月=1 (以下同様)になります。

NWKDOM 関数を使用して、HOLIDAY 関数で定義されていないイベントを計算できます。たとえば、大学の卒業が常に 6 月の第 1 土曜日にスケジュールされている場合、次のステートメントを使用してその日付を計算できます。UnivGrad = nwkdom(1, 7, 6, *year*);

例

例 1: 日付値を返す

次のステートメントでは、NWKDOM 関数を使用して、指定した月および年に発生する特定の曜日の日付を返す方法を示します。例では WEEKDATX.出力形式を適用して、出力 SAS 日付値を読み取り可能な形式で表示します。

ステートメント	結果
select put(nwkdom(3, 2, 5, 2012), weekdatx.);	Monday, 21 May 2012
select put(nwkdom(4, 4, 11, 2012), weekdatx.);	Wednesday, 28 November 2012

ステートメント	結果
select put(nwkdom(4, 7, 11, 2012), weekdatx.);	Saturday, 24 November 2012
select put(nwkdom(1, 1, 1, 2013), weekdatx.);	Sunday, 6 January 2013
select put(nwkdom(2, 3, 9, 2012), weekdatx.);	Tuesday, 11 September 2012
select put(nwkdom(5, 5, 12, 2012), weekdatx.);	Thursday, 27 December 2012

例 2: 日付を返す

次のステートメントは、NWKDOM 関数を使用して 5 月の最終月曜日の日付を返す方法を示しています。DATE9.出力形式は、日付値を表示するために適用されます。

ステートメント	結果
select put(nwkdom(5,2,5,2012), date9.);	28MAY2012

関連項目

関数:

- “HOLIDAY 関数” (567 ページ)
- “INTNX 関数” (603 ページ)
- “WEEKDAY 関数” (883 ページ)

OCTET_LENGTH 関数

任意のデータ型の文字列のバイト数を返します。

カテゴリ:	文字
返されるデータの種類:	BIGINT

構文

OCTET_LENGTH(*expression*)

必須引数

expression
任意の有効な式を指定します。

データの種類 すべてのデータ型が有効です。

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

OCTET_LENGTH 関数は、マルチバイト文字を複数の文字としてカウントします。たとえば、指定された文字列に 6 つの 2 バイト文字が含まれている場合、OCTET_LENGTH 関数は値 12 を返します。

文字列がビット列の場合、OCTET_LENGTH 関数は、そのビット数を保持するのに必要なバイト数を返します。

比較

CHARACTER_LENGTH 関数も文字列の文字数を返しますが、マルチバイト文字は 1 文字としてカウントします。OCTET_LENGTH 関数は、マルチバイト文字を複数の文字としてカウントします。

例

次のステートメントは、OCTET_LENGTH 関数を示しています。

ステートメント	結果
select octet_length('December');	8
select octet_length(B'0100011100');	2
select octet_length(x'FEA364BEA234');	6
select octet_length(' FedSQL ');	8

関連項目

関数:

- “CHARACTER_LENGTH 関数” (382 ページ)

ORDINAL 関数

値のリストを並べ替え、リスト内の位置に基づく値を返します。

カテゴリ: 記述統計

返されるデータの種類: DOUBLE

構文

ORDINAL(*position*, *expression-1, expression-2* [, ...*expression-n*])

必須引数

position
引数リストの要素数以下の整数を指定します。

要件 *position* は正数にする必要があります。

データの種類 DOUBLE

expression
数値に評価される有効な式を指定します。

要件 少なくとも 2 つの引数が必要です。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

ORDINAL 関数はリストを並べ替えて、*position* で指定されたリスト内の引数を返します。欠損値は下位に並べ替えられ、数値の前に配置されます。

比較

ORDINAL 関数は、null 値、欠損値、非欠損値、非 null 値の両方を数えますが、SMALLEST 関数は非 null 値および非欠損値のみを数えます。

例

次のステートメントは、ORDINAL 関数を示しています。

ステートメント	結果
select ordinal(4,1,2,3,-4,5,6,7);	3

PCTL 関数

パーセントに対応するパーセント点を返します。

カテゴリ: 記述統計

返されるデータの種類: DOUBLE

構文

PCTL[*n*](*percentage*, *expression*[, ...*expression*])

必須引数

percentage
計算するパーセント点を指定します。

データの種類 DOUBLE

ヒント *percentage* は、0≤*percentage*≤100 の数値です。

percentage は、0 ≤ *percentage* ≤ 100 の数値です。

オプション引数

n
計算するパーセント点の定義を指定する 1 から 5 までの数字です。

デフォルト 定義 5

データの種類 DOUBLE

参照項目: *Base SAS プロシジャガイド*の"Methods for Computing Quantile Statistics"の表にある QNTLDEF= (エイリアス PCTLDEF=)の説明。

expression
数値に評価される有効な式を指定します。その値はパーセント点計算で計算されます。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

PCTL 関数は、パーセントに対応する非 null 値または非欠損値のパーセント点を返します。*percentage* が null 値または欠損値、ゼロ未満、あるいは 100 を超える場合、PCTL 関数はエラーメッセージを生成します。

例

次のステートメントは、PCTL 関数を示しています。

説明	ステートメント	結果
下側四分位点	select pctl(25,2,4,1,3);	1.5
パーセント点の定義 2	select pctl2(25,2,4,1,3);	1
下側三分位点	select pctl(100/3,2,4,1,3);	2
パーセント点の定義 3	select pctl3(100/3,2,4,1,3);	2
中央値	select pctl(50,2,4,1,3);	2.5
上側三分位点	select pctl(200/3,2,4,1,3);	3
上側四分位点	select pctl(75,2,4,1,3);	3.5

PDF 関数

確率密度(質量)分布の値を返します。

カテゴリ: 確率

別名: PMF

データの種類: DOUBLE

構文

PDF(*'distribution'*, *quantile* [, *parameter-1, ..., parameter-k*])

必須引数

distribution
分布を特定する文字定数、変数または式です。有効な分布は次のとおりです。

分布	引数
Bernoulli	'BERNOULLI'
ベータ	'BETA'
二項	'BINOMIAL'
Cauchy	'CAUCHY'
カイ 2 乗	'CHISQUARE'
Conway-Maxwell-Poisson	'CONMAXPOI'
指数	'EXPONENTIAL'
F	'F'
ガンマ	'GAMMA'
一般化 Poisson	'GENPOISSON'
幾何	'GEOMETRIC'
超幾何	'HYPERGEOMETRIC'
Laplace	'LAPLACE'
ロジスティック	'LOGISTIC'
対数正規	'LOGNORMAL'
負の二項	'NEGBINOMIAL'
正規	'NORMAL'|'GAUSS'

分布	引数
正規混合	'NORMALMIX'
パレート	'PARETO'
Poisson	'POISSON'
T	'T'
Tweedie	'TWEEDIE'
一様	'UNIFORM'
Wald (逆ガウス)	'WALD'|'IGAUSS'
Weibull	'WEIBULL'

注 T、F および NORMALMIX を除き、最初の 4 文字で分布を識別できます。

quantile
確率変数の値を指定する数値定数、変数または式です。

データの種類 DOUBLE

parameter-1, ..., parameter-k
特定の分布に適した*形状*、*位置*または*尺度*パラメーターの値を指定する数値定数、変数または式です(省略可能)。

データの種類 DOUBLE

関連項目

関数:

- “CDF 関数” (334 ページ)
- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

参考文献

Shmueli, G., T. Minka,, S. Borle, and P. Boatwright. 2005. “A Useful Distribution for Fitting Discrete Data: Revival of the Conway-Maxwell-Poisson Distribution.” *Applied Statistics* : 54:127–142.

PDF BERNOULLI 分布 関数

Bernoulli 確率密度(質量)分布の値を返します。

カテゴリ:	確率
別名:	PMF
返されるデータの種類:	DOUBLE

構文

PDF ('BERNOULLI', *x*, *p*)

必須引数

x
確率変数を指定する数値定数、変数または式です。

データの種類	DOUBLE

p
成功確率を指定する数値定数、変数または式です。

範囲	0≤*p*≤1
データの種類	DOUBLE

詳細

Bernoulli 分布の PDF 関数は、*p* に等しい成功確率で確率密度関数を返します。この PDF 関数は、値 *x* で評価されます。

$$PDF('BERN', x, p) = \begin{cases} 0 & x < 0 \\ 1 - p & x = 0 \\ 0 & 0 < x < 1 \\ p & x = 1 \\ 0 & x > 1 \end{cases}$$

注: この分布には、*位置*または*尺度*パラメーターはありません。

例

次のステートメントは、PDF Bernoulli 分布関数を示しています。

ステートメント	結果
select pdf('BERN', 0, .25);	0.75

関連項目

関数:

- “CDF BERNOULLI 分布 関数” (337 ページ)
- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

PDF BETA 分布 関数

ベータ確率密度(質量)分布の値を返します。

カテゴリ:	確率
別名:	PMF
返されるデータの種類:	DOUBLE

構文

PDF ('BETA', *x*, *a*, *b* [, *l*, *r*])

必須引数

x
確率変数を指定する数値定数、変数または式です。

データの種類 DOUBLE

a
形状パラメーターを指定する数値定数、変数または式です。

範囲 $a > 0$

データの種類 DOUBLE

b
形状パラメーターを指定する数値定数、変数または式です。

範囲 $b > 0$

データの種類 DOUBLE

オプション引数

l
左位置パラメーターを指定する数値定数、変数または式です。

デフォルト 0

データの種類 DOUBLE

r
右位置パラメーターを指定する数値定数、変数または式です。

デフォルト 1

範囲 $r > l$

データの種類 DOUBLE

詳細

ベータ分布の PDF 関数は、形状パラメーター *a* と *b* で確率密度関数を返します。この PDF 関数は、値 *x* で評価されます。

$$PDF('BETA', x, a, b, l, r) = \begin{cases} 0 & x < l \\ \frac{1}{\beta(a,b)} \frac{(x-l)^{a-1}(r-x)^{b-1}}{(r-l)^{a+b-1}} & l \le x \le r \\ 0 & x > r \end{cases}$$

注: 数量$\frac{x-l}{r-l}$は、$\varepsilon \le \frac{x-l}{r-l} \le 1 - 2\varepsilon$へと強制されます。

例

次のステートメントは、PDF ベータ分布関数を示しています。

ステートメント	結果
select pdf('BETA', 0.2, 3, 4);	1.2288

関連項目

関数:

- “CDF BETA 分布 関数” (338 ページ)
- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

PDF BINOMIAL 分布 関数

二項確率密度(質量)分布の値を返します。

カテゴリ:	確率
別名:	PMF
返されるデータの種類:	DOUBLE

構文

PDF ('BINOMIAL', *m*, *p*, *n*)

必須引数

m
成功数を数える確率変数です。この引数は整数にする必要があります。

範囲 *m*=0, 1, ...

データの種類 DOUBLE

p

成功確率を指定する数値定数、変数または式です。

範囲 $0 \le p \le 1$

データの種類 DOUBLE

n

独立 Bernoulli 試行数を数えるパラメーターです。この引数は整数にする必要があります。

範囲 *n*=0, 1, ...

データの種類 DOUBLE

詳細

二項分布の PDF 関数は、パラメーター *p* と *n* で確率密度関数を返します。この PDF 関数は、値 *m* で評価されます。

$$PDF('BINOM', m, p, n) = \begin{cases} 0 & m < 0 \\ \binom{n}{m} p^m (1-p)^{n-m} & 0 \le m \le n \\ 0 & m > n \end{cases}$$

注: 二項分布には、*位置*または*尺度*パラメーターはありません。

例

次のステートメントは、PDF 二項分布関数を示しています。

ステートメント	結果
select pdf('BINOM', 4, .5, 10);	0.205078

関連項目

関数:

- “CDF BINOMIAL 分布 関数” (340 ページ)
- “LOGCDF 関数” (643 ページ)

- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

PDF CAUCHY 分布 関数

Cauchy 確率密度(質量)分布の値を返します。

カテゴリ:	確率
別名:	PMF
返されるデータの種類:	DOUBLE

構文

PDF ('CAUCHY', *x* [, *θ*, *λ*])

必須引数

x
確率変数を指定する数値定数、変数または式です。

データの種類 DOUBLE

オプション引数

θ
位置パラメーターを指定する数値定数、変数または式です。

デフォルト 0

データの種類 DOUBLE

λ
尺度パラメーターを指定する数値定数、変数または式です。

デフォルト 1

範囲 λ > 0

データの種類 DOUBLE

詳細

Cauchy 分布の PDF 関数は、位置パラメーターは θ、尺度パラメーターは λ の確率密度関数を返します。PDF 関数は、値 *x* で評価されます。

$$PDF('CAUCHY', x, \theta, \lambda) = \frac{1}{\pi}\left(\frac{\lambda}{\lambda^2 + (x-\theta)^2}\right)$$

例

次のステートメントは、PDF Cauchy 分布関数を示しています。

ステートメント	結果
select pdf('CAUCHY', 2);	0.063662

関連項目

関数:

- “CDF CAUCHY 分布 関数” (342 ページ)
- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

PDF カイ 2 乗分布 関数

カイ 2 乗確率密度(質量)分布の値を返します。

カテゴリ:	確率
別名:	PMF
返されるデータの種類:	DOUBLE

構文

PDF ('CHISQUARE', *x*, *df* [, *nc*])

必須引数

x
確率変数を指定する数値定数、変数または式です。

データの種類 DOUBLE

df
自由度を指定する数値定数、変数または式です。

範囲 *df* > 0

データの種類 DOUBLE

オプション引数

nc
省略可能な非心度パラメーターを指定する数値定数、変数または式です。

範囲 *nc* ≥ 0

データの種類 DOUBLE

詳細

カイ 2 乗分布の PDF 関数は、自由度が *df*、非心度パラメーターが *nc* のカイ 2 乗分布の確率密度関数を返します。PDF 関数は、値 *x* で評価されます。この関数では、整数以外の自由度を使用できます。*nc* が省略されているかゼロの場合、心度カイ 2 乗分布の値が返されます。

$$PDF('CHISQ', x, \nu, \lambda) = \begin{cases} 0 & x < 0 \\ \sum_{j=0}^{\infty} e^{-\frac{\lambda}{2}} \frac{\left(\frac{\lambda}{2}\right)^j}{j!} p_c(x, \nu + 2j) & x \geq 0 \end{cases}$$

この式では、$p_c(.,.)$は、心度カイ 2 乗分布の密度を示しています。

$$p_c(x, a) = \frac{1}{2} p_g\left(\frac{x}{2}, \frac{a}{2}\right)$$

この式では、$p_g(y,b)$はガンマ分布からの密度です。

$$p_g(y, b) = \frac{1}{\Gamma(b)} e^{-y} y^{b-1}$$

例

次のステートメントは、PDF カイ 2 乗分布関数を示しています。

ステートメント	結果
select pdf('CHISQ', 11.264, 11);	0.081686

関連項目

関数:

- “CDF カイ 2 乗分布 関数” (344 ページ)
- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

PDF Conway-Maxwell-Poisson 分布 関数

Conway-Maxwell-Poisson 確率密度(質量)分布の値を返します。

カテゴリ:	確率
別名:	PMF
返されるデータの種類:	DOUBLE

構文

PDF('CONMAXPOI',*y*,*λ*,*ν*)

必須引数

y
カウントを表す負でない整数を指定する数値定数、変数または式です。

データの種類 DOUBLE

λ

Poisson 平均パラメーターに似た位置パラメーターを指定する数値定数、変数または式です。

データの種類 DOUBLE

ν

ばらつきのパラメーターを指定する数値定数、変数または式です。

データの種類 DOUBLE

詳細

Conway-Maxwell-Poisson (CMP)分布は Poisson 分布の一般化です。これにより、過小分散データと過大分散データをモデル化できます。CMP 分布は、次の式に従って定義されます。

$$P\left(Y = y\,;\lambda,\nu\right) = \frac{1}{Z(\lambda,\nu)}\frac{\lambda^{y}}{\left(y!\right)^{\nu}} \qquad y = 0, 1, 2, \ldots$$

正規化因子は次の式で表されます。

$$Z\left(\lambda,\nu\right) = \sum_{n=0}^{\infty}\frac{\lambda^{n}}{\left(n!\right)^{\nu}}$$

λ および ν は負ではなく、同時にゼロにもなりません。

追加パラメーター ν は、分布の裾挙動のモデル化における柔軟性が考慮されています。ν=1 の場合、比は Poisson 分布の減衰率に等しくなります。ν<1 の場合、減衰率が減少し、Poisson 分布より裾が長いプロセス(過大分散データ)をモデル化できます。ν>1 の場合、減衰率が非線形方法で増加し、したがって、分布の裾が短くなります(過小分散データ)。

Conway-Maxwell-Poisson 分布には、いくつか特殊なケースがあります。λ<1 および ν→∞の場合、結果的に Conway-Maxwell-Poisson 分布が Bernoulli 分布になります。このケースでは、データが取り得る値は 0 および 1 です。これは極過小分散を表します。ν=1 の場合、等分散の特性を持つ Poisson 分布に戻ります。ν=0 および λ<1 の場合、正規化因子は収束性であり、次の幾何級数を形成します。

$$Z\left(\lambda,0\right) = \frac{1}{1-\lambda}$$

確率密度関数は、次の式で表されます。

$$P\left(Y = y\,;\lambda,\nu = 0\right) = \left(1-\lambda\right)\lambda^{y}$$

幾何分布は、著しい過大分散のケースを表します。

Conway-Maxwell-Poisson モデルの平均、分散およびばらつき

Conway-Maxwell-Poisson 分布の平均および分散は、次の式で定義されます。

$$E\left[Y\right] = \frac{\partial\, 1n\; Z}{\partial\, 1n\,\lambda}$$

$$V\left[Y\right] = \frac{\partial^2 \ln Z}{\partial^2 \ln \lambda}$$

Conway-Maxwell-Poisson 分布は、そのパラメーター λ および ν に関して、その積率の閉形式を持ちません。ただし、積率は近似できます。(Conway-Maxwell-Poisson 分布および離散データの詳細については、この関数の最後にあるリファレンスセクションを参照してください。)次の式で示すように、Z に対する漸近表現を使用すると、E(Y)および V(Y)を導き出せます。

$$E\left[Y\right] \approx \lambda^{1/\nu} + \frac{1}{2\nu} - \frac{1}{2}$$

$$V\left[Y\right] \approx \frac{1}{\nu}\lambda^{1/\nu}$$

Conway-Maxwell-Poisson モデルでは、対数展開を使用して無限級数の総和が評価されます。(Conway-Maxwell-Poisson 分布および離散データの詳細については、この関数の最後にあるリファレンスセクションを参照してください。)Conway-Maxwell-Poisson モデルについては、次のように平均および分散が計算されます。

$$E\left(Y\right) = \frac{1}{Z(\lambda,\nu)} \sum_{j=0}^{\infty} \frac{j\lambda^j}{(j!)^\nu}$$

$$V\left(Y\right) = \frac{1}{Z(\lambda,\nu)} \sum_{j=0}^{\infty} \frac{j^2\lambda^j}{(j!)^\nu} - E(Y)^2$$

ばらつきは、次のように定義されます。

$$D\left(Y\right) = \frac{V(Y)}{E(Y)}$$

例

次のステートメントは、PDF Conway-Maxwell-Poisson 分布関数を示しています。

ステートメント	結果
select pdf('CONMAXPOI', .2, 2.3, .4);	0.009773

関連項目

関数:

- “CDF Conway-Maxwell-Poisson 分布 関数” (346 ページ)
- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)

- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

PDF EXPONENTIAL 分布 関数

指数確率密度(質量)分布の値を返します。

カテゴリ:	確率
別名:	PMF
返されるデータの種類:	DOUBLE

構文

PDF ('EXPONENTIAL', *x* [, *λ*])

必須引数

x
確率変数を指定する数値定数、変数または式です。

データの種類	DOUBLE

オプション引数

λ
尺度パラメーターを指定する数値定数、変数または式です。

デフォルト	1
範囲	λ > 0
データの種類	DOUBLE

詳細

指数分布の PDF 関数は、指数分布(尺度パラメーターは λ)の確率密度関数を返します。PDF 関数は、値 *x* で評価されます。

$$PDF('EXPO', x, \lambda) = \begin{cases} 0 & x < 0 \\ \frac{1}{\lambda} exp\left(-\frac{x}{\lambda}\right) & x \geq 0 \end{cases}$$

例

次のステートメントは、PDF 指数分布関数を示しています。

ステートメント	結果
select pdf('EXPO', 1);	0.367879

関連項目

関数:

- “CDF 指数分布 関数” (347 ページ)
- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

PDF F 分布 関数

F 確率密度(質量)分布の値を返します。

カテゴリ: 確率

別名: PMF

返されるデータの種類: DOUBLE

構文

PDF ('F', *x*, *ndf*, *ddf* [, *nc*])

必須引数

x

確率変数を指定する数値定数、変数または式です。

データの種類 DOUBLE

ndf
分子の自由度を指定する数値定数、変数または式です。

範囲 *ndf*> 0

データの種類 DOUBLE

ddf
分母の自由度を指定する数値定数、変数または式です。

範囲 *ddf* > 0

データの種類 DOUBLE

オプション引数

nc
省略可能な非心度パラメーターを指定する数値定数、変数または式です。

範囲 *nc* ≥ 0

データの種類 DOUBLE

詳細

F 分布の PDF 関数は、*F* 分布(分子の自由度は *ndf*、分母の自由度は *ddf*、非心度パラメーターは *nc*)の確率密度関数を返します。PDF 関数は、値 *x* で評価されます。この PDF 関数では、*ndf* および *ddf* に対して整数以外の自由度を使用できます。*nc* が除外されているかゼロの場合、心度 *F* 分布の値が返されます。次の式で、$\nu_1 = ndf$として、$\nu_2 = ddf$として、そして$\lambda = nc$とします。次の式は、*F* 分布の PDF 関数を表しています。

$$PDF('F', x, \nu_1, \nu_2, \lambda) = \begin{cases} 0 & x < 0 \\ \sum_{j=0}^{\infty} e^{-\frac{\lambda}{2}} \frac{\left(\frac{\lambda}{2}\right)^j}{j!} p_f(f, \nu_1 + 2j, \nu_2) & x \geq 0 \end{cases}$$

この式では、$p_f(f,u_1,u_2)$は、次の式を使用した心度 *F* 分布の密度です。

$$p_f(f, u_1, u_2) = p_B\left(\frac{u_1 f}{uf + u_2}, \frac{u_1}{2}, \frac{u_2}{2}\right) \frac{u_1 u_2}{(u_2 + u_1 f)^2}$$

この式では、$p_B(x,a,b)$は、標準ベータ分布の密度です。

注: *F* 分布には、*位置*または*尺度*パラメーターはありません。

例

次のステートメントは、PDF F 分布関数を示しています。

ステートメント	結果
select pdf('F', 3.32, 2, 3);	0.054027

関連項目

関数:

- “CDF F 分布 関数” (349 ページ)
- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

PDF GAMMA 分布 関数

ガンマ確率密度(質量)分布の値を返します。

カテゴリ:	確率
別名:	PMF
返されるデータの種類:	DOUBLE

構文

PDF ('GAMMA', *x*, *α* [, *λ*])

必須引数

x
確率変数を指定する数値定数、変数または式です。

データの種類 DOUBLE

a
形状パラメーターを指定する数値定数、変数または式です。

範囲 $a > 0$

データの種類 DOUBLE

オプション引数

λ
尺度パラメーターを指定する数値定数、変数または式です。

デフォルト 1

範囲 $\lambda > 0$

データの種類 DOUBLE

詳細

ガンマ分布の PDF 関数は、ガンマ分布(形状パラメーターは *a*、尺度パラメーターは λ)の確率密度関数を返します。PDF 関数は、値 *x* で評価されます。

$$PDF('GAMMA', x, a, \lambda) = \begin{cases} 0 & x < 0 \\ \frac{1}{\lambda^a \Gamma(a)} x^{a-1} exp\left(-\frac{x}{\lambda}\right) & x \geq 0 \end{cases}$$

例

次のステートメントは、PDF ガンマ分布関数を示しています。

ステートメント	結果
select pdf('GAMMA', 1, 3);	0.18394

関連項目

関数:

- “CDF GAMMA 分布 関数” (351 ページ)
- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)

- “SDF 関数” (819 ページ)

PDF 一般化 Poisson 分布 関数

一般化 Poisson 確率密度(質量)分布の値を返します。

カテゴリ:	確率
別名:	PMF
返されるデータの種類:	DOUBLE

構文

PDF ('GENPOISSON', *x*, *θ*, *η*)

必須引数

x
確率変数を指定する数値定数、変数または式です。この引数は整数にする必要があります。

データの種類	DOUBLE

θ
形状パラメーターを指定する数値定数、変数または式です。

範囲	$<10^5$ かつ>0
データの種類	DOUBLE

η
形状パラメーターを指定する数値定数、変数または式です。

範囲	≥0 かつ<0.95
データの種類	DOUBLE
ヒント	η =0 の場合、平均と分散が θ の Poisson 分布になります。η>0 の場合、平均は$\theta \div (1-\eta)$および分散は$\theta \div (1-\eta)^3$になります。

詳細

一般化 Poisson 分布の確率質量関数の次のとおりです。

$$f(x;\theta,\eta) = \theta(\theta+\eta x)^{x-1} e^{-\theta-\eta x}/x!, \quad x = 0, 1, 2, ..., \qquad \theta > 0,\ 0 \le \eta < 1$$

例

次のステートメントは、PDF 一般化 Poisson 分布関数を示しています。

ステートメント	結果
select pdf('GENPOISSON', 9, 1, .7);	0.015013

関連項目

関数:

- “CDF 一般化 Poisson 分布 関数” (352 ページ)
- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

PDF GEOMETRIC 分布 関数

幾何確率密度(質量)分布の値を返します。

カテゴリ: 確率

別名: PMF

返されるデータの種類: DOUBLE

構文

PDF ('GEOMETRIC', *m*, *p*)

必須引数

m
初めて成功する前の失敗数を指定する数値定数、変数または式です。

範囲 $m \geq 0$

データの種類 DOUBLE

p

成功確率を指定する数値定数、変数または式です。

範囲 $0 \leq p \leq 1$

データの種類 DOUBLE

詳細

幾何分布の PDF 関数は、パラメーター *p* をもつ確率密度関数を返します。この PDF 関数は、値 *m* で評価されます。

$$PDF('GEOM', m, p) = \begin{cases} 0 & m < 0 \\ p(1-p)^m & m \geq 0 \end{cases}$$

注: この分布には、*位置*または*尺度*パラメーターはありません。

例

次のステートメントは、PDF 幾何分布関数を示しています。

ステートメント	結果
select pdf('GEOMETRIC', 5, .3);	0.050421

関連項目

関数:

- “CDF GEOMETRIC 分布 関数” (354 ページ)
- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

PDF 超幾何分布 関数

超幾何確率密度(質量)分布の値を返します。

カテゴリ:	確率
別名:	PMF
返されるデータの種類:	DOUBLE

構文

PDF ('HYPER', *x*, *N*, *R*, *n* [, *o*])

必須引数

x
確率変数を指定する数値定数、変数または式です。この引数は整数にする必要があります。

データの種類 DOUBLE

N
母集団サイズパラメーターを指定する数値定数、変数または式です。この引数は整数にする必要があります。

範囲 *N*=1, 2, ...

データの種類 DOUBLE

R
対象カテゴリの項目数を指定する数値定数、変数または式です。この引数は整数にする必要があります。

範囲 *R*=0, 1, ..., *N*

データの種類 DOUBLE

n
サンプルサイズパラメーターを 指定する数値定数、変数または式です。この引数は整数にする必要があります。

範囲 *n*=1, 2, ..., *N*

データの種類 DOUBLE

オプション引数

o

省略可能なオッズ比パラメーターを指定する数値定数、変数または式です。

範囲	*o* > 0
データの種類	DOUBLE

詳細

超幾何分布の PDF 関数は、拡張超幾何分布(母集団サイズは *N*、項目数は *R*、サンプルサイズは *n*、オッズ比は *o*)の確率密度関数を返します。この PDF 関数は、値 *x* で評価されます。*o* が除外されているか 1 の場合、一般超幾何分布の値が返されます。

$$PDF\ ('HYPER', x, N, R, n, o) =$$

$$\begin{cases} 0 & x < max(0, R+n-N) \\ \dfrac{\binom{R}{x}\binom{N-R}{n-x}o^x}{\sum_{j=max(0,R+n-N)}^{min(R,n)}\binom{R}{j}\binom{N-R}{n-j}o^j} & max(0, R+n-N) \le x \le min(R, n) \\ 0 & x > min(R, n) \end{cases}$$

例

次のステートメントは、PDF 超幾何分布関数を示しています。

ステートメント	結果
select pdf('HYPER', 2, 200, 50, 10);	0.286851

関連項目

関数:

- “CDF HYPERGEOMETRIC 分布 関数” (355 ページ)
- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

PDF LAPLACE 分布 関数

Laplace 確率密度(質量)分布の値を返します。

カテゴリ:	確率
別名:	PMF
返されるデータの種類:	DOUBLE

構文

PDF ('LAPLACE', *x* [, *θ*, *λ*])

必須引数

x
確率変数を指定する数値定数、変数または式です。

データの種類 DOUBLE

オプション引数

θ
位置パラメーターを指定する数値定数、変数または式です。

デフォルト 0

データの種類 DOUBLE

λ
尺度パラメーターを指定する数値定数、変数または式です。

デフォルト 1

範囲 λ > 0

データの種類 DOUBLE

詳細

Laplace 分布の PDF 関数は、Laplace 分布(位置パラメーターは θ、尺度パラメーターは λ)の確率密度関数を返します。PDF 関数は、値 *x* で評価されます。

$$PDF('LAPLACE', x, \theta, \lambda) = \frac{1}{2\lambda} exp\left(- \frac{|x - \theta|}{\lambda} \right)$$

例

次のステートメントは、PDF Laplace 分布関数を示しています。

ステートメント	結果
select pdf('LAPLACE', 1);	0.18394

関連項目

関数:

- “CDF LAPLACE 分布 関数” (357 ページ)
- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

PDF LOGISTIC 分布 関数

ロジスティック確率密度(質量)分布の値を返します。

カテゴリ: 確率

別名: PMF

返されるデータの種類: DOUBLE

構文

PDF ('LOGISTIC', *x* [, *θ*, *λ*])

必須引数

x

確率変数を指定する数値定数、変数または式です。

データの種類 DOUBLE

オプション引数

θ
位置パラメーターを指定する数値定数、変数または式です。

デフォルト　0

データの種類　DOUBLE

λ
尺度パラメーターを指定する数値定数、変数または式です。

デフォルト　1

範囲　λ > 0

データの種類　DOUBLE

詳細

ロジスティック分布の PDF 関数は、ロジスティック分布(位置パラメーターは θ、尺度パラメーターは λ)の確率密度関数を返します。PDF 関数は、値 *x* で評価されます。

$$PDF('LOGISTIC', x, \theta, \lambda) = \frac{exp\left(-\frac{x-\theta}{\lambda}\right)}{\lambda\left(1+exp\left(-\frac{x-\theta}{\lambda}\right)\right)^2}$$

例

次のステートメントは、PDF ロジスティック分布関数を示しています。

ステートメント	結果
select pdf('LOGISTIC', 1);	0.196612

関連項目

関数:

- “CDF LOGISTIC 分布 関数” (359 ページ)
- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “QUANTILE 関数” (793 ページ)

- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

PDF LOGNORMAL 分布 関数

対数正規確率密度(質量)分布の値を返します。

カテゴリ: 確率

別名: PMF

返されるデータの種類: DOUBLE

構文

PDF ('LOGNORMAL', *x* [, *θ*, *λ*])

必須引数

x
確率変数を指定する数値定数、変数または式です。

データの種類 DOUBLE

オプション引数

θ
対数尺度パラメーターを指定する数値定数、変数または式です。exp(θ)は尺度パラメーターです。

デフォルト 0

データの種類 DOUBLE

λ
形状パラメーターを指定する数値定数、変数または式です。

デフォルト 1

範囲 λ > 0

データの種類 DOUBLE

詳細

対数正規分布の PDF 関数は、対数正規分布(対数尺度パラメーターは θ、形状パラメーターは λ)の確率密度関数を返します。PDF 関数は、値 *x* で評価されます。

$$PDF('LOGN', x, \theta, \lambda) = \begin{cases} 0 & x \le 0 \\ \frac{1}{x\lambda\sqrt{2\,\pi}}\ exp\left(-\frac{(log(x)-\theta)^2}{2\lambda^2}\right) & x > 0 \end{cases}$$

例

次のステートメントは、PDF 対数正規分布関数を示しています。

ステートメント	結果
select pdf('LOGNORMAL', 1);	0.398942

関連項目

関数:

- “CDF LOGNORMAL 分布 関数” (361 ページ)
- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

PDF NEGBINOMIAL 分布 関数

負の二項確率密度(質量)分布の値を返します。

カテゴリ:	確率
別名:	PMF
返されるデータの種類:	DOUBLE

構文

PDF ('NEGBINOMIAL', *m*, *p*, *n*)

必須引数

m
失敗数を数える確率変数を指定する、数値定数、変数または式です。この引数は正の整数にする必要があります。

範囲 *m*=0, 1, ...

データの種類 DOUBLE

p
成功確率を指定する数値定数、変数または式です。

範囲 $0 \le p \le 1$

データの種類 DOUBLE

n
成功数を数える値を指定する数値定数、変数または式です。

範囲 *n*>0

データの種類 DOUBLE

詳細

負の二項分布の PDF 関数は、負の二項分布(成功確率は *p*、成功数は *n*)の確率密度関数を返します。この PDF 関数は、値 *m* で評価されます。

$$PDF('NEGB', m, p, n) = \begin{cases} 0 & m < 0 \\ \binom{n+m-1}{n-1} p^n (1-p)^m & m \ge 0 \end{cases}$$

注: 負の二項分布には、*位置*または*尺度*パラメーターはありません。

例

次のステートメントは、PDF 負の二項分布関数を示しています。

ステートメント	結果
select pdf('NEGB',1,.5,2);	0.25

関連項目

関数:

- “CDF NEGBINOMIAL 分布 関数” (362 ページ)
- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “SDF 関数” (819 ページ)

PDF NORMAL 分布 関数

正規確率密度(質量)分布の値を返します。

カテゴリ:	確率
別名:	PMF
返されるデータの種類:	DOUBLE

構文

PDF ('NORMAL', *x* [, *θ*, *λ*])

必須引数

x
確率変数を指定する数値定数、変数または式です。

データの種類 DOUBLE

オプション引数

θ
位置パラメーターを指定する数値定数、変数または式です。

デフォルト 0

データの種類 DOUBLE

λ
尺度パラメーターを指定する数値定数、変数または式です。

デフォルト	1
範囲	λ > 0
データの種類	DOUBLE

詳細

正規分布の PDF 関数は、正規分布(位置パラメーターは θ、尺度パラメーターは λ)の確率密度関数を返します。PDF 関数は、値 *x* で評価されます。

$$PDF('NORMAL', x, \theta, \lambda) = \frac{1}{\lambda\sqrt{2\,\pi}}\ exp\left(-\frac{(x-\theta)^2}{2\lambda^2}\right)$$

例

次のステートメントは、PDF 正規分布関数を示しています。

ステートメント	結果
select pdf('NORMAL', 1.96);	0.058441

関連項目

関数:

- “CDF NORMAL 分布 関数” (364 ページ)
- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

PDF NORMALMIX 分布 関数

正規混合確率密度(質量)分布の値を返します。

カテゴリ:	確率
別名:	PMF
返されるデータの種類:	DOUBLE

構文

PDF ('NORMALMIX', *x*, *n*, p_1, p_2, ... p_n, m_1, m_2, ... m_n, s_1, s_2, ... s_n)

必須引数

x
確率変数を指定する数値定数、変数または式です。

データの種類 DOUBLE

n
混合数を指定する数値定数、変数または式です。この引数は整数にする必要があります。

範囲 *n*=1, 2, ...

データの種類 DOUBLE

p_i
n 個の比率$p_1, p_2, \ldots, p_n$を指定する、数値の定数、変数または式のリストです。ここで、$\sum_{i=1}^{i=n} p_i = 1$です。

範囲 *p*=0, 1, ...

データの種類 DOUBLE

m_i
n 個の平均$m_1, m_2, \ldots, m_n$を指定する、数値の定数、変数または式のリストです。

s_i
n 個の標準偏差$s_1, s_2, \ldots, s_n$を指定する、数値の定数、変数または式のリストです。

範囲 *s* > 0

データの種類 DOUBLE

詳細

正規混合分布の PDF 関数は、正規混合分布のオブザベーションが *x* 以下となる確率を返します。

$$PDF('NORMALMIX', x, n, p, m, s) = \sum_{i=1}^{i=n} p_i \, PDF('NORMAL', x, m_i, s_i)$$

正規混合分布の重み付けは、負でないことが必要です。重み付けの合計が 1 に等しくない場合は相対的な重み付けとして扱われ、合計が 1 となるように調整されます。

注: 正規混合分布には、*位置*または*尺度*パラメーターはありません。

例

次のステートメントは、PDF 正規混合分布関数を示しています。

ステートメント	結果
select pdf('NORMALMIX',2.3, 3, .33, .33, .34,.5, 1.5, 2.5,.79, 1.6, 4.3);	0.116554

関連項目

関数:

- “CDF NORMALMIX 分布 関数” (366 ページ)
- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

PDF PARETO 分布 関数

パレート確率密度(質量)分布の値を返します。

カテゴリ:	確率
別名:	PMF
返されるデータの種類:	DOUBLE

構文

PDF ('PARETO', *x*, *a* [, *k*])

必須引数

x
数値の確率変数を指定する数値定数、変数または式です。

データの種類 DOUBLE

a
形状パラメーターを指定する数値定数、変数または式です。

範囲 $a > 0$

データの種類 DOUBLE

オプション引数

k
尺度パラメーターを指定する数値定数、変数または式です。

デフォルト 1

範囲 $k > 0$

データの種類 DOUBLE

詳細

パレート分布の PDF 関数は、パレート分布(形状パラメーターは *a*、尺度パラメーターは *k*)の確率密度関数を返します。PDF 関数は、値 *x* で評価されます。

$$PDF('PARETO', x, a, k) = \begin{cases} 0 & x < k \\ \frac{a}{k}\left(\frac{k}{x}\right)^{a+1} & x \geq k \end{cases}$$

例

次のステートメントは、PDF パレート分布関数を示しています。

ステートメント	結果
select pdf('PARETO', 1, 1);	1

関連項目

関数:

- “CDF PARETO 分布 関数” (368 ページ)
- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

PDF POISSON 分布 関数

Poisson 確率密度(質量)分布の値を返します。

カテゴリ:	確率
別名:	PMF
返されるデータの種類:	DOUBLE

構文

PDF ('POISSON', *n*, *m*)

必須引数

n
確率変数を指定する数値定数、変数または式です。この引数は整数にする必要があります。

範囲	*n*=0, 1, ...
データの種類	DOUBLE

m
平均パラメーターを指定する数値定数、変数または式です。

範囲	*m* > 0
データの種類	DOUBLE

詳細

Poisson 分布の PDF 関数は、平均が *m* の確率密度関数を返します。この PDF 関数は、値 *n* で評価されます。

$$PDF('POISSON', n, m) = \begin{cases} 0 & n < 0 \\ e^{-m}\frac{m^n}{n!} & n \geq 0 \end{cases}$$

注: Poisson 分布には、*位置*または*尺度*パラメーターはありません。

例

次のステートメントは、PDF Poisson 分布関数を示しています。

ステートメント	結果
select pdf('POISSON', 2, 1);	0.18394

関連項目

関数:

- “CDF POISSON 分布 関数” (369 ページ)
- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

PDF T 分布 関数

T 確率密度(質量)分布の値を返します。

カテゴリ: 確率

別名: PMF

返されるデータの種類: DOUBLE

構文

PDF ('T', *t*, *df* [, *nc*])

必須引数

t
確率変数を指定する数値定数、変数または式です。

データの種類 DOUBLE

df
自由度を指定する数値定数、変数または式です。

範囲 *df* > 0

データの種類 DOUBLE

オプション引数

nc
省略可能な非心度パラメーターを指定する数値定数、変数または式です。

データの種類 DOUBLE

詳細

T 分布の PDF 関数は、*T* 分布(自由度は *df*、非心度パラメーターは *nc*)の確率密度関数を返します。PDF 関数は、値 *x* で評価されます。この PDF 関数では、整数以外の自由度を使用できます。*nc* が除外されているかゼロの場合、心度 *T* 分布の値が返されます。この式で、$\nu = df$として、そして$\delta = nc$とします。

$$PDF('T', t, \nu, \delta) = \frac{1}{2^{\frac{\nu}{2}-1}\Gamma\left(\frac{\nu}{2}\right)} \int_0^{\infty} x^{\nu-1} e^{-\frac{1}{2}x^2} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{tx}{\sqrt{\nu}}-\delta\right)^2} \frac{x}{\sqrt{\nu}} dx$$

注: *T* 分布には、*位置*または*尺度*パラメーターはありません。

例

次のステートメントは、PDF T 分布関数を示しています。

ステートメント	結果
select pdf('T',.9,5);	0.24194

関連項目

関数:

- “CDF T 分布 関数” (371 ページ)
- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

PDF TWEEDIE 分布 関数

Tweedie 確率密度(質量)分布の値を返します。

カテゴリ:	確率
別名:	PMF
返されるデータの種類:	DOUBLE

構文

PDF ('TWEEDIE', *y*, *p* [, *μ*, *φ*])

必須引数

y

確率変数を指定する数値定数、変数または式です。

範囲	$y \geq 0$
データの種類	DOUBLE
注	この引数は必須です。

y>1 の場合、*y* は数値になります。*p*=1 の場合、*y* は整数になります。

p
累乗パラメーターを指定する数値定数、変数または式です。

範囲 $p \geq 1$

データの種類 DOUBLE

注 この引数は必須です。

オプション引数

μ
平均パラメーターを指定する数値定数、変数または式です。

デフォルト 1

範囲 $\mu > 0$

データの種類 DOUBLE

φ
ばらつきのパラメーターを指定する数値定数、変数または式です。

デフォルト 1

範囲 $\varphi > 0$

データの種類 DOUBLE

詳細

Tweedie 分布の PDF 関数は、*variance* の式の関係がある分散と平均を使用した指数散布度モデルを返します。$= \phi * \mu^p$

$$\frac{1}{y}\sum_{j=1}^{\infty}\left(\frac{y^{-j\alpha}(p-1)^{j\alpha}}{\phi^{j(1-\alpha)}(2-p)^{j}j!\Gamma(-j\alpha)}\right)\exp\left(\frac{1}{\phi}\left(y\frac{\mu^{1-p}-1}{1-p}-\frac{\mu^{2-p}-1}{2-p}\right)\right)$$

前述の式には次の関係が適用されます。

$$\alpha = \frac{2-p}{1-p}$$

注: 計算された Tweedie 確率の精度は、パラメーター空間における位置に大きく依存します。通常、10 桁の精度を利用できますが、次の場合を除きます。*p* の近似値が 2 の場合、またはϕの近似値が 0 の場合どちらのケースにしても、6 桁の精度に低下する可能性があります。

注: 数値データでの問題を避けるため、μ および Φ は定数 SQRTMACEPS より小さくしないようにします。

例

次のステートメントは、PDF Tweedie 分布関数を示しています。

ステートメント	結果
select pdf('TWEEDIE', .8, 5);	0.742291

関連項目

関数:

- “CDF TWEEDIE 分布 関数” (372 ページ)
- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

PDF UNIFORM 分布 関数

一様確率密度(質量)分布の値を返します。

カテゴリ:	確率
別名:	PMF
返されるデータの種類:	DOUBLE

構文

PDF ('UNIFORM', *x* [, *l*, *r*])

必須引数

x
確率変数を指定する数値定数、変数または式です。

データの種類 DOUBLE

オプション引数

l
左位置パラメーターを指定する数値定数、変数または式です。

デフォルト 0

データの種類 DOUBLE

r
右位置パラメーターを指定する数値定数、変数または式です。

デフォルト 1

範囲 $r > l$

データの種類 DOUBLE

詳細

一様分布の PDF 関数は、一様分布(左位置パラメーターは *l*、右位置パラメーターは *r*)の確率密度関数を返します。PDF 関数は、値 *x* で評価されます。

$$PDF('UNIFORM', x, l, r) = \begin{cases} 0 & x < l \\ \frac{1}{r-l} & l \le x \le r \\ 0 & x > r \end{cases}$$

例

次のステートメントは、PDF 一様分布関数を示しています。

ステートメント	結果
select pdf('UNIFORM', 0.25);	1

関連項目

関数:

- “CDF UNIFORM 分布 関数” (375 ページ)
- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

PDF Wald (逆ガウス)分布 関数

Wald (逆ガウスとも呼ばれる)確率密度(質量)分布から値を返します。

カテゴリ:	確率
別名:	PMF
返されるデータの種類:	DOUBLE

構文

PDF ('WALD', *x*, *λ* [, *μ*])
PDF ('IGAUSS', *x*, *λ* [, *μ*])

必須引数

x
確率変数を指定する数値定数、変数または式です。

λ
形状パラメーターを指定する数値定数、変数または式です。

範囲 $\lambda > 0$

オプション引数

μ
平均パラメーターを指定する数値定数、変数または式です。

デフォルト 1

範囲 μ > 0

詳細

Wald 分布の PDF 関数は、値 *x* で評価される Wald 分布(形状パラメーターは λ)の確率密度関数を返します。

$$fx(x) = \left[\frac{\lambda}{2\pi x^3}\right]^{1/2} \exp\left\{-\frac{\lambda}{2\mu^2 x}(x-\mu)^2\right\}, \qquad x > 0$$

例

次のステートメントは、PDF Wald 分布関数を示しています。

ステートメント	結果
select pdf('WALD', 1, 2);	0.56419

関連項目

関数:

- “CDF WALD (逆ガウス)分布 関数” (376 ページ)
- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

PDF WEIBULL 分布 関数

Weibull 確率密度(質量)分布の値を返します。

カテゴリ: 確率

別名: PMF

返されるデータの種類: DOUBLE

構文

PDF('WEIBULL', *x*, *α* [, *λ*])

必須引数

x
確率変数を指定する数値定数、変数または式です。

データの種類 DOUBLE

α
形状パラメーターを指定する数値定数、変数または式です。

範囲 $\alpha > 0$

データの種類 DOUBLE

オプション引数

λ
尺度パラメーターを指定する数値定数、変数または式です。

デフォルト 1

範囲 $\lambda > 0$

データの種類 DOUBLE

詳細

Weibull 分布の PDF 関数は、Weibull 分布(形状パラメーターは *α*、尺度パラメーターは λ)の確率密度関数を返します。PDF 関数は、値 *x* で評価されます。

$$PDF('WEIBULL', x, a, \lambda) = \begin{cases} 0 & x < 0 \\ exp\left(-\left(\frac{x}{\lambda}\right)^{a}\right)\frac{a}{\lambda}\left(\frac{x}{\lambda}\right)^{a-1} & x \geq 0 \end{cases}$$

例

次のステートメントは、PDF Weibull 分布関数を示しています。

ステートメント	結果
select pdf('WEIBULL', 1, 2);	0.735759

関連項目

関数:

- “CDF WEIBULL 分布 関数” (378 ページ)
- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “QUANTILE 関数” (793 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

PERM 関数

n 個の項目から一度に *r* 個の項目を抜き出す場合の順列数を計算します。

カテゴリ: 組み合わせ

返されるデータの種類: DOUBLE

構文

PERM(*n*[, *r*])

必須引数

n

要素の合計数を表す有効な式を指定します。サンプルはこの中から選ばれます。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

オプション引数

r

選択した要素の数を表す有効な式を指定します。

制限事項 $r \leq n$

データの種類 DOUBLE

注 *r* を省略すると、関数は *n* の階乗を返します。

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

PERM 関数の数学的表現は、次の式によって得られます。

$$PERM(n, r) = \frac{n!}{(n-r)!}$$

$n \geq 0$、$r \geq 0$ および $n \geq r$ となります。

式による計算ができない場合は欠損値が返されます。やや大きな値の場合、PERM 関数を計算できないことがあります。

例

次のステートメントは、PERM 関数を示しています。

ステートメント	結果
select perm(5, 1);	5
select perm(5);	120
select perm(5, 2);	20

PI 関数

PI の定数値を浮動小数点値として返します。

カテゴリ: 数学

返されるデータの種類: DOUBLE

構文

PI()

引数なし

この関数には引数がありません。

詳細

PI の値は 3.141593 です。

例

次のステートメントは、PI()関数を示しています。

ステートメント	結果
select pi();	3.141593

PMT 関数

均等払いローンまたは定期預金の将来残高に対する定期的支払いを返します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

PMT(*rate*, *number-of-periods*, *principal-amount*[, *future-amount*][, *type*])

必須引数

rate
支払期間ごとの利率を指定します。

データの種類 DOUBLE

number-of-periods
支払期間の数を指定します。

要件 *number-of-periods* は正の整数にする必要があります。

データの種類 DOUBLE

principal-amount
ローンの元金を指定します。null 値または欠損値が指定されている場合、ゼロとみなされます。

データの種類 DOUBLE

オプション引数

future-amount
将来の残高を指定します。*future-amount* は、指定した支払い期間数終了後のローンの残高、または定期預金の将来の残高を指定できます。0 を指定した場合、*future-amount* が省略されたか欠損値が指定されたとみなされます。

データの種類 DOUBLE

type
期首と期末のどちらで支払がなされるのかを指定します。0 は期末の支払を表し、1 は期首の支払を表します。*type* が省略された場合や、null 値または欠損値が指定された場合は、ゼロとみなされます。

データの種類 DOUBLE

例

- 名目年利が 8%で支払い期間が 10 か月の$10,000 のローンの月々の支払いは次のように計算されます。

  ```
  select PMT(0.08/12., 10, 10000, 0, 0);

  select PMT(0.08/12., 10, 10000);

  1037.03208935915
  ```

  これらの計算の戻り値は 1037.032 です。
- 同じローンに期首払いがある場合、支払いは次のように計算できます。

  ```
  select PMT(0.08/12., 10, 10000, 0, 1);

  1030.16432717796
  ```

  計算によって 1030.164 という値が返されます。

- 名目年利 12%の$5,000 のローンを 5 か月の支払いで返済する場合の支払い額は、次のように計算されます。

  select PMT(.01/12., 5, 5000);

  1002.50138831008

  計算によって 1002.501 という値が返されます。

- 名目年利 6%で 18 年以上積み立て、18 年終了時に$50,000 積み立てる毎月の定期預金の支払い額は、次のように計算されます。

  select PMT(0.06/12., 216, 0, 50000, 0);

  -129.081160867993

  計算によって-129.081 という値が返されます。

POISSON 関数

Poisson 分布の確率を返します。

カテゴリ: 確率

返されるデータの種類: DOUBLE

構文

POISSON(*m*, *n*)

必須引数

m

数値の平均パラメーターに評価される有効な式を指定します。

範囲 *m* ≥ 0

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

n

確率変数に評価される有効な式を指定します。

範囲 *n* ≥ 0

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

POISSON 関数は、平均が *m* の Poisson 分布のオブザベーションが *n* 以下となる確率を返します。オブザベーションが指定した値の *n* と等しくなる確率を計算するには、Poisson 分布からの 2 つの確率(*n* の確率と *n*-1 の確率)の差異を計算します。

例

次のステートメントは、POISSON 関数を示しています。

ステートメント	結果
select poisson(1, 2);	0.9196986029

POWER 関数

指定された累乗の数値式の値を返します。

カテゴリ: 数学

返されるデータの種類: DOUBLE

構文

POWER(*numeric-expression*, *power-numeric-expression*)

必須引数

numeric-expression
数値に評価される有効な SQL 式を指定します。

データの種類 BIGINT、DOUBLE、FLOAT、INTEGER、REAL、SMALLINT、TINYINT

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

power-numeric-expression
数値に評価される有効な SQL 式を指定します。この引数は累乗値です。

詳細

numeric-expression が null または欠損値の場合、POWER 関数は null または欠損値を返します。結果が DOUBLE の範囲に収まらない数値の場合、POWER 関数は失敗します。

例

テーブル: DENSITIES (1260 ページ)

次のステートメントは、POWER 関数を示しています。

ステートメント	結果
select power(5*3, 2);	225
select name from densities where squaremiles >= power(density,2);	Afghanistan Algeria Angola Argentina Australia

PROBBETA 関数

ベータ分布の確率を返します。

カテゴリ: 確率

返されるデータの種類: DOUBLE

構文

PROBBETA(*x*, *a*, *b*)

必須引数

x
数値の確率変数です。

範囲 $0 \le x \le 1$

データの種類 DOUBLE

a
数値の形状パラメーターです。

範囲 $a > 0$

データの種類 DOUBLE

b
数値の形状パラメーターです。

範囲 $b > 0$

データの種類 DOUBLE

詳細

PROBBETA 関数は、形状パラメーターが *a* と *b* のベータ分布のオブザベーションが *x* 以下となる確率を返します。

例

次のステートメントは、PROBBETA 関数を示しています。

ステートメント	結果
select probbeta(.2,3,4);	0.09888

PROBBNML 関数

二項分布の確率を返します。

カテゴリ: 確率

返されるデータの種類: DOUBLE

構文

PROBBNML(*p*, *n*, *m*)

必須引数

p
数値の成功率パラメーターです。

範囲 $0 \le p \le 1$

データの種類 DOUBLE

n
独立した Bernoulli 試行数のパラメーターです。この引数は整数にする必要があります。

範囲 $n > 0$

データの種類 DOUBLE

m
成功数の確率変数です。この引数は整数にする必要があります。

範囲 $0 \le m \le n$

データの種類 DOUBLE

詳細

PROBBNML 関数は、成功確率が *p*、試行数が *n*、成功数が *m* の二項分布のオブザベーションが *m* 以下となる確率を返します。オブザベーションが指定した値の *m* と等しくなる確率を計算するには、二項分布の 2 つの確率(*m* の成功と *m*-1 の成功)の差異を計算します。

例

次のステートメントは、PROBBNML 関数を示しています。

ステートメント	結果
select probbnml(0.5,10,4);	0.376953125

PROBBNRM 関数

2 変量正規分布の確率を返します。

カテゴリ: 確率

返されるデータの種類: DOUBLE

構文

PROBBNRM(*x*, *y*, *r*)

必須引数

x
数値の定数、変数または式を指定します。

データの種類 DOUBLE

y
数値の定数、変数または式を指定します。

データの種類 DOUBLE

r
数値相関係数です。

範囲 -1 ≤ *r* ≤ 1

データの種類 DOUBLE

詳細

PROBBNRM 関数は、平均が 0、分散が 1、相関係数が *r* の標準 2 変量正規分布のオブザベーション(X, Y)が(*x*, *y*)以下となる確率を返します。すなわち、X≤*x* かつ Y≤*y* となる確率を返します。次の等式は、*u* と *v* がそれぞれ確率変数 *x* と *y* を表す PROBBNRM 関数を示しています。

$$\mathrm{PROBBNRM}(x, y, r) = \frac{1}{2\pi\sqrt{1-r^2}} \int_{-\infty}^{x} \int_{-\infty}^{y} \exp\left[-\frac{u^2 - 2ruv + v^2}{2(1-r^2)} \right] dv\, du$$

例

次のステートメントは、PROBBNRM 関数を示しています。

ステートメント	結果
select probbnrm(.4, -.3, .2);	0.27831833451902

PROBCHI 関数

カイ 2 乗分布の確率を返します。

カテゴリ: 確率

返されるデータの種類: DOUBLE

構文

PROBCHI(*x*, *df*[, *nc*])

必須引数

x
数値の確率変数です。

範囲 $x \geq 0$

データの種類 DOUBLE

df
数値の自由度パラメーターです。

範囲 $df > 0$

データの種類 DOUBLE

オプション引数

nc
数値の非心度パラメーターです(省略可能)。

範囲 $nc \geq 0$

データの種類 DOUBLE

詳細

PROBCHI 関数は、自由度が *df*、非心度パラメーターが *nc* のカイ 2 乗分布のオブザベーションが *x* 以下となる確率を返します。この関数では、整数以外の自由度パラメーター *df* を使用できます。パラメーター *nc* (省略可能)が指定されていないかゼロの場合、心度カイ 2 乗分布の値が返されます。

例

次のステートメントは、PROBCHI 関数を示しています。

ステートメント	結果
select probchi(11.264,11);	0.5785813293173

PROBF 関数

F 分布の確率を返します。

カテゴリ: 確率

返されるデータの種類: DOUBLE

構文

PROBF(*x*, *ndf*, *ddf*[, *nc*])

必須引数

x
数値の確率変数です。

範囲 $x \geq 0$

データの種類 DOUBLE

ndf
数値の分子の自由度パラメーターです。

範囲 $ndf > 0$

データの種類 DOUBLE

ddf
数値の分母の自由度パラメーターです。

範囲 $ddf > 0$

データの種類 DOUBLE

オプション引数

nc
数値の非心度パラメーターです(省略可能)。

範囲 $nc \geq 0$

データの種類 DOUBLE

詳細

PROBF 関数は、分子の自由度が *ndf*、分母の自由度が *ddf*、非心度パラメーターが *nc* の *F* 分布のオブザベーションが *x* 以下となる確率を返します。PROBF 関数では、自由度パラメーター *ndf* と *ddf* に非整数を指定できます。パラメーター *nc* (省略可能)が指定されていないかゼロの場合、心度 *F* 分布の値が返されます。

F 検定統計の有意水準は次の式で求められます。

```
p=1-probf(x,ndf,ddf);
```

例

次のステートメントは、PROBF 関数を示しています。

ステートメント	結果
select probf(3.32,2,3);	0.8263933602

PROBGAM 関数

ガンマ分布の確率を返します。

カテゴリ: 確率

返されるデータの種類: DOUBLE

構文

PROBGAM(*x*, *a*)

必須引数

x
数値の確率変数です。

範囲 $x \geq 0$

データの種類 DOUBLE

a
数値の形状パラメーターです。

データの種類 DOUBLE

詳細

PROBGAM 関数は、形状パラメーターが *a* のガンマ分布のオブザベーションが *x* 以下となる確率を返します。

例

次のステートメントは、PROBGAM 関数を示しています。

ステートメント	結果
select probgam(1,3);	0.0803013971

PROBHYPR 関数

超幾何分布の確率を返します。

カテゴリ: 確率

返されるデータの種類: DOUBLE

構文

PROBHYPR(*N*, *K*, *n*, *x*[, *r*])

必須引数

N
母集団サイズパラメーターです。この引数は整数にする必要があります。

範囲 $N \geq 1$

データの種類 DOUBLE

K
対象カテゴリの項目数のパラメーターです。この引数は整数にする必要があります。

範囲 $0 \leq K \leq N$

データの種類 DOUBLE

n
サンプルサイズパラメーターです。この引数は整数にする必要があります。

範囲 $0 \leq n \leq N$

データの種類 DOUBLE

x
確率変数です。この引数は整数にする必要があります。

範囲 $max(0, K + n-N) \leq x \leq min(K,n)$

オプション引数

r
数値オッズ比パラメーターです。

範囲 $r \geq 0$

データの種類 DOUBLE

詳細

PROBHYPR 関数は、拡張超幾何分布(母集団サイズは *N*、項目数は *K*、サンプルサイズは *n*、オッズ比は *r*)からオブザベーションが *x* 以下となる確率を返します。パラメーター *r* (省略可能)が指定されていないか 1 に設定されている場合、一般超幾何分布の値が返されます。

例

次のステートメントは、PROBHYPR 関数を示しています。

ステートメント	結果
select probhypr(200,50,10,2);	0.5236734081

PROBIT 関数

標準正規分布の分位点を返します。

カテゴリ: 分位点

返されるデータの種類: DOUBLE

構文

PROBIT(*p*)

必須引数

p
数値の確率です。

範囲 $0 < p < 1$

データの種類 DOUBLE

詳細

PROBIT 関数は、標準正規分布の *p*番目の分位点を返します。標準正規分布のオブザベーションが、返される分位点以下となる確率は *p* です。

注意
結果は-8.222 から 7.941 の間の値に切り捨てられる場合があります。

注: PROBIT は PROBNORM 関数の逆数です。

例

次のステートメントは、PROBIT 関数を示しています。

ステートメント	結果
select probit(.025);	-1.959963985
select probit(1.e-7);	-5.199337582

PROBMC 関数

平均値の多重比較を行うために各種の分布から確率または分位点を返します。

カテゴリ: 確率

返されるデータの種類: DOUBLE

構文

PROBMC(*distribution*, *q*, *prob*, *df*, *nparms*[, *parameters*])

必須引数

distribution
文字列を評価するか、文字列に強制変換できる文字定数、変数、または式であり、分布を識別します。分布として、ANOM (平均の分析)、DUNNETT1、DUNNETT2、MAXMOD (最大モジュール)、PARTRANGE (パーティション分割範囲)、RANGE (Student 化範囲)、WILLIAMS が有効です。

分布を特定する文字定数、変数または式です。有効な分布は次のとおりです。

分布	引数
平均分析	ANOM
片側 Dunnett	DUNNETT1
両側 Dunnett	DUNNETT2
最大係数	MAXMOD
分割された範囲	PARTRANGE
スチューデント化された範囲	RANGE
Williams	WILLIAMS

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

q
分布からの分位点です。

制限事項 *q* または *prob* のいずれか一方を指定できます(両方指定することはできません)。

データの種類 DOUBLE

prob
分布からの左側確率です。

制限事項 *prob* または *q* のいずれか一方を指定できます(両方指定することはできません)。

データの種類 DOUBLE

df
自由度です。

注: 欠損値は無限値として解釈されます。

nparms
治療の数です。

データの種類 DOUBLE

注 DUNNETT1 と DUNNETT2 については、コントロールグループを計算に入れません。

オプション引数

parameters
標本サイズが等しくない場合の処理に指定する必要のある *nparms* パラメーターのセットです。*parameters* の意味は *distribution* の値によって異なります。*parameters* が指定されていない場合、サンプルサイズは揃っているものと見なされます。通常、帰無仮説のケースはサンプルサイズが揃っています。

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

詳細

概要

PROBMC 関数は、分散推定値に有限および無限の自由度を使用して、各種の分布の確率または分位点を返します。

prob 引数は確率変数が *q* 未満となる確率です。したがって、*p*-values は 1– *prob* のように計算できます。たとえば、有意水準 5%の臨界値を計算するには、*prob*= 0.95

と設定します。計算された確率の精度は *O*(10-8) (絶対誤差)、計算された分位点の精度は *O*(10^{-5})です。

注: 自由度が有限のときと、サンプルサイズが不揃いのときは、スチューデント化された範囲は計算されません。

注: Williams 検定はサンプルサイズが揃っているときにのみ計算されます。

式とパラメーター

ここに掲げる等式は、各種の分布と、それぞれの分布におけるさまざまな状況で確率 *prob* と分位点 *q* を関連付ける等式に使用する式を定義します。これらの式では、*v* を自由度 *df* とします。

$$d\mu_\nu(x) = \frac{\nu^{\frac{\nu}{2}}}{\Gamma\left(\frac{\nu}{2}\right)2^{\frac{\nu}{2}-1}} x^{\nu-1}\varepsilon^{-\frac{\nu x^2}{2}}dx$$

$$\phi(x) = \frac{1}{\sqrt{2\,\pi}}\varepsilon^{-\frac{x^2}{2}}$$

$$\Phi(x) = \int_{-\infty}^{x} \phi(u)du$$

平均の分析の計算

平均の分析(ANOM)は、*k* (ガウス)標本として編成されるデータに適用されます。ここでは、*i*番目の標本のサイズが n_i となります。$I = \sqrt{-1}$であるとします。分布関数[1, 2, 3, 4, 5]は、多次元 *k* 多変量$\mathbb{T}$ベクトルの最大絶対値、ν 自由度、関連付けられた相関行列$\rho_{ij} = -\alpha_i\alpha_j$の CDF です。この式は次のように記述することができます。

$$\begin{aligned} prob &= r(|t_1| < h, |t_2| < h, ..., |t_k| < h) \\ &= \int_0^\infty \left\{ \int_0^\infty \prod_{j=0}^{j=k} g(sh, y, \alpha_j)\phi(y)dy \right\} d\mu_\nu(s) \end{aligned}$$

前述の式には次の関係が適用されます。

$$g(sh, y, \alpha_j) = \Phi\left(\frac{sh - y\alpha_j}{\sqrt{1+\alpha_j^2}}\right) - \Phi\left(\frac{-sh - y\alpha_j}{\sqrt{1+\alpha_j^2}}\right)$$

この式では、$\Gamma(.)$、$\phi(.)$、$\Phi(.)$はそれぞれ、標準正規分布からのガンマ関数、密度、および CDF です。

$\nu = \infty$では、分布はこの式になります。

$$r(|t_1| < h, |t_2| < h, ..., |t_k| < h) = \int_0^\infty \prod_{j=0}^{j=k} g(h, y, \alpha_j)\phi(y)dy$$

前述の式には次の関係が適用されます。

$$g(h, y, \alpha_j) = \Phi\left(\frac{h - y\alpha_j}{\sqrt{1 + \alpha_j^2}}\right) - \Phi\left(\frac{-h - y\alpha_j}{\sqrt{1 + \alpha_j^2}}\right)$$

バランスの取れている場合、分布は次の式になります。

$$r(|t_1| < h, |t_2| < h, ..., |t_n| < h) = \int_0^\infty f(h, y, \rho)^n \phi(y) dy$$

前述の式には次の関係が適用されます。

$$f(h, y, \rho) = \Phi\left(\frac{h - y\sqrt{\rho}}{\sqrt{1 + \rho}}\right) - \Phi\left(\frac{-h - y\sqrt{\rho}}{\sqrt{1 + \rho}}\right)$$

$$\rho = \frac{1}{n - 1}$$

この分布の構文は次のとおりです。

x=*probmc*('*anom*', *q*, *p*, *nu*, *n*[, *alpha*$_1$, ..., *alpha*$_n$]);

引数

x
返される結果の数値です。

q
分位点を示す数値です。

p
確率を示す数値です。*p* と *q* は、いずれかが欠損値である必要があります。

nu
自由度を示す数値です。

n
サンプル数を示す数値です。

alpha$_i$, *i* = 1, ...,*k*
この分布の最初の式のアルファ値を示すオプションの数値です。 “平均の分析の計算” (775 ページ)を参照してください。

多対 1 の *t* 統計量: Dunnett の片側検定

- 自由度が有限の不揃いなケースで確率 *prob* と分位点 *q* を相関させます。*parameters* は $\lambda_1, ..., \lambda_k$、*nparms* の設定値は *k*、*df* の設定値は *v* です。式は次のとおりです。

$$prob = \int_0^\infty \int_{-\infty}^\infty \phi(y) \prod_{i=1}^k \Phi\left(\frac{\lambda_i y + qx}{\sqrt{1 - \lambda_i^2}}\right) dy\, du_\nu(x)$$

- 自由度が有限の揃っているケースで確率 *prob* と分位点 *q* を相関させます。*parameters* は渡されず、$\left(\lambda = \sqrt{\frac{1}{2}}\right)$*nparms* の値は *k* に設定され、*df* の値は *v* に設定されます。式は次のとおりです。

$$prob = \int_0^\infty \int_{-\infty}^\infty \phi(y)[\Phi(y + \sqrt{2qx})]^k dy\, du_\nu(x)$$

- 自由度が無限の不揃いなケースで確率 *prob* と分位点 *q* を相関させます。*parameters* は $\lambda_1, \ldots, \lambda_k$、*nparms* の設定値は *k*、*df* の設定値は欠損です。式は次のとおりです。

$$prob = \int_{-\infty}^{\infty} \phi(y) \prod_{i=1}^{k} \Phi\left(\frac{\lambda_i y + q}{\sqrt{1-\lambda_i^2}}\right) dy$$

- 自由度が無限の揃っているケースで確率 *prob* と分位点 *q* を相関させます。*parameters* は渡されず、$\left(\lambda = \sqrt{\frac{1}{2}}\right)$*nparms* の値は *k* に設定され、*df* の値は欠損値に設定されます。式は次のとおりです。

$$prob = \int_{-\infty}^{\infty} \phi(y)[\Phi(y + \sqrt{2q})]^k dy$$

多対 1 の *t* 統計量: Dunnett の両側検定

- 自由度が有限の不揃いなケースで確率 *prob* と分位点 *q* を相関させます。*parameters* は $\lambda_1, \ldots, \lambda_k$、*nparms* の設定値は *k*、*df* の設定値は *ν* です。式は次のとおりです。

$$prob = \int_{0}^{\infty} \int_{-\infty}^{\infty} \phi(y) \prod_{i=1}^{k} \left[\Phi\left(\frac{\lambda_i y + qx}{\sqrt{1-\lambda_i^2}}\right) - \Phi\left(\frac{\lambda_i y - qx}{\sqrt{1-\lambda_i^2}}\right)\right] dy\, du_\nu(x)$$

- 自由度が有限の揃っているケースで確率 *prob* と分位点 *q* を相関させます。*parameters* は渡されず、*nparms* の設定は *k*、*df* の設定値は *ν* です。式は次のとおりです。

$$prob = \int_{0}^{\infty} \int_{-\infty}^{\infty} \phi(y)[\Phi(y + \sqrt{2qx}) - \Phi(y - \sqrt{2qx})]^k dy\, du_\nu(x)$$

- 自由度が無限の不揃いなケースで確率 *prob* と分位点 *q* を相関させます。*parameters* は $\lambda_1, \ldots, \lambda_k$、*nparms* の設定値は *k*、*df* の設定値は欠損です。式は次のとおりです。

$$prob = \int_{-\infty}^{\infty} \phi(y) \prod_{i=1}^{k} \left[\Phi\left(\frac{\lambda_i y + q}{\sqrt{1-\lambda_i^2}}\right) - \Phi\left(\frac{\lambda_i y - q}{\sqrt{1-\lambda_i^2}}\right)\right] dy$$

- 自由度が無限の揃っているケースで確率 *prob* と分位点 *q* を相関させます。*parameters* は渡されず、*nparms* の設定は *k*、*df* の設定値は欠損です。式は次のとおりです。

$$prob = \int_{-\infty}^{\infty} \phi(y)[\Phi(y + \sqrt{2q}) - \Phi(y - \sqrt{2q})]^k dy$$

パーティション分割範囲の計算

RANGE は、*n* のグループ平均のスチューデント化範囲の分布で使用されます。PARTRANGE は、分割されたスチューデント化範囲の分布で使用されます。*n* グループを、大きさが $n_1 + \ldots + n_k = n$ の *k* 個のサブセットに分割します。このとき、分割された範囲はそれぞれのサブセットのスチューデント化範囲の最大です。スチューデント化係数はすべてのケースで 同一です。

$$prob = \int_{0}^{\infty} \prod_{i=1}^{i=k} \left(\int_{-\infty}^{\infty} k\phi(y)(\Phi(y) - \Phi(y - qx))^{k-1} dy \right)^{n_i} d\mu_{\nu}(x)$$

この分布の構文は次のとおりです。

x=*probmc*('*partrange*', *q*, *p*,*nu*, *k*, n_1 ,...,n_k);

引数

x
返される結果の数値です(確率または分位点)。

q
分位点を示す数値です。

p
確率を示す数値です。*p* と *q* は、いずれかが欠損値である必要があります。

nu
自由度を示す数値です。

k
グループ数を示す数値です。

n_i *i*=1,...,*k*
この分布の式の *n* 値を示すオプションの数値です。“パーティション分割範囲の計算” (777 ページ)を参照してください。

Student 化範囲

- 自由度が有限の揃っているケースで確率 *prob* と分位点 *q* を相関させます。*parameters* は渡されず、*nparms* の設定は *k*、*df* の設定値は *v* です。式は次のとおりです。

$$prob = \int_{0}^{\infty} \int_{-\infty}^{\infty} k\phi(y)[\Phi(y) - \Phi(y - qx)]^{k-1} dy \, du_{\nu}(x)$$

- 自由度が無限の不揃いなケースで確率 *prob* と分位点 *q* を相関させます。*parameters* は σ_1, ..., σ_k、*nparms* の設定値は *k*、*df* の設定値は欠損です。式は次のとおりです。

$$prob = \int_{-\infty}^{\infty} \sum_{j=1}^{k} \left\{ \prod_{i=1}^{k} \left[\Phi\left(\frac{y}{\sigma_i}\right) - \Phi\left(\frac{y-q}{\sigma_i}\right) \right] \right\} \phi\left(\frac{y}{\sigma_j}\right) \frac{1}{\sigma_j} dy$$

- 自由度が無限の揃っているケースで確率 *prob* と分位点 *q* を相関させます。*parameters* は渡されず、*nparms* の設定は *k*、*df* の設定値は欠損です。式は次のとおりです。

$$prob = \int_{-\infty}^{\infty} k\phi(y)[\Phi(y) - \Phi(y - q)]^{k-1} dy$$

Student 化最大モジュール

- 自由度が有限の不揃いなケースで確率 *prob* と分位点 *q* を相関させます。*parameters* は σ_1, ..., σ_k、*nparms* の設定値は *k*、*df* の設定値は *v* です。式は次のとおりです。

$$prob = \int_0^{\infty} \prod_{i=1}^{k} \left[2\Phi\left(\frac{qx}{\sigma_i}\right) - 1 \right] d\mu_\nu(x)$$

- 自由度が有限の揃っているケースで確率 *prob* と分位点 *q* を相関させます。*parameters* は渡されず、*nparms* の設定は *k*、*df* の設定値は *ν* です。式は次のとおりです。

$$prob = \int_0^{\infty} [2\Phi(qx) - 1]^k d\mu_\nu(x)$$

- 自由度が無限の不揃いなケースで確率 *prob* と分位点 *q* を相関させます。*parameters* は $\sigma_1, \ldots, \sigma_k$、*nparms* の設定値は *k*、*df* の設定値は欠損です。式は次のとおりです。

$$prob = \prod_{i=1}^{k} \left[2\Phi\left(\frac{q}{\sigma_i}\right) - 1 \right]$$

- 自由度が無限の揃っているケースで確率 *prob* と分位点 *q* を相関させます。*parameters* は渡されず、*nparms* の設定は *k*、*df* の設定値は欠損です。式は次のとおりです。

$$prob = [2\Phi(q) - 1]^k$$

Williams の検定

PROBMC は、Williams (1971、1972)によって定義されている分布の確率または分位点を計算します。""参考文献"" (*SAS 関数と CALL ルーチン: リファレンス*)を参照してください。Williams 検定の必要性は、処方治療の平均をコントロール平均と比較し、治療で有効な最小の処方量を判断するときに発生します。

注: Williams 検定はサンプルサイズが揃っているときにのみ計算されます。

$X_1, X_2, \ldots, X_k$ は同一の独立している N(0,1)の確率変数であるものとします。ここで、Y_k は次の式で与えられる平均値を表します。

$$Y_k = \frac{X_1 + X_2 + \ldots + X_k}{k}$$

次の値の分布を計算する必要があります。

$$(Y_k - Z)/S$$

引数

Y_k
前述の定義のとおりです。

Z
N(0,1)の独立した確率変数です。

S
$\frac{1}{2}\nu S^2$ が χ^2 変数です(自由度 ν)。

Williams (1971)の説明によれば、全体の計算は非常に冗長であり、3 段階で行われます。""参考文献"" (*SAS 関数と CALL ルーチン: リファレンス*)を参照してください。

1 Y_k の分布を計算します。分布の計算はこの演算の基本的な(高コストな)部分で、Y_k の密度と確率の両方を求めるのに使われます。U_i をこの式に定義するとします。

$$U_i = \frac{X_1 + X_2 + ... + X_i}{i},\ i = 1, 2, ..., k$$

$Y_k > d$ の確率の再帰式を書くことができます。d の値は任意の実数になります。

$$\begin{aligned}\Pr(Y_k > d) &= \Pr(U_1 > d) \\ &\quad +\Pr(U_2 > d, U_1 < d) \\ &\quad +\Pr(U_3 > d, U_2 < d, U_1 < d) \\ &\quad +\ldots \\ &\quad +\Pr(U_k > d, U_{k-1} < d, \ldots, U_1 < d) \\ &= \Pr(Y_{k-1} > d) + \Pr(X_k + (k-1)U_{k-1} > kd)\end{aligned}$$

この確率を計算するには、N(0,1)密度関数から開始します。

$$D(U_1 = x) = \phi(x)$$

そして、畳み込みを再帰的に計算します。

$$D(U_k = x, U_{k-1} < d, \ldots, U_1 < d) =$$
$$\int_{-\infty}^{d} D(U_{k-1} = y, U_{k-2} < d, \ldots, U_1 < d)(k-1)\phi(kx - (k-1)y)dy$$

この順次畳み込みから、前に示した$\Pr(Y_k < d)$の再帰方程式のすべての要素を計算することができます。

2 $Y_k - Z$ の分布を計算します。この計算には、確率を計算するための別の畳み込みが必要です。

$$\Pr((Y_k - Z) > d) = \int_{-\infty}^{\infty} \Pr(Y_k > \sqrt{2d} + y)\phi(y)dy$$

3 $(Y_k - Z)/S$ の分布を計算します。この計算には、確率を計算するための別の畳み込みが含まれます。

$$\Pr((Y_k - Z) > tS) = \int_{0}^{\infty} \Pr((Y_k - Z) > ty)d\mu_\nu(y)$$

ν = ∞のときは、第 3 段階は必要ありません。演算が複雑なため、この冗長なアルゴリズムは有限および無限の自由度 ν の両方で $k \le 15$ のとき、高速なアルゴリズムに置き換えられます。$k \ge 16$ のときは、冗長計算が実行されます。この計算は、アルゴリズムが複雑なため、きわめて高コストで非常に時間がかかります。

比較

SAS/STAT ソフトウェアの GLM プロシジャの MEANS ステートメントは次の検定を計算します。

- Dunnett の片側検定

- Dunnett の両側検定
- スチューデント化された範囲

例

例 1: 平均の分析の計算

次のステートメントは、PROBMC 関数を使用した平均の分析の計算を示しています。

```
select probmc('anom',.,0.9,.,20) as q1,
      probmc('anom',.,0.9,20,5,0.1,0.1,0.1,0.1,0.1) as q2,
      probmc('anom',.,0.9,20,5,0.5,0.5,0.5,0.5,0.5) as q3,
      probmc('anom',.,0.9,20,5,0.1,0.2,0.3,0.4,0.5) as q4;
 quit;
```

アウトプット 4.11　平均分析の計算結果

Q1	Q2	Q3	Q4
2.789506	2.454996	2.454996	2.453232

例 2: パーティション分割範囲の計算

次のステートメントは、PROBMC 関数を使用したパーティション分割範囲の計算を示しています。

```
select probmc('partrange',.,0.9,.,4,3,4,5,6) as q1,
     probmc('partrange',.,0.9,12,4,3,4,5,6) as q2;
quit;
```

アウトプット 4.12　パーティション分割範囲の計算結果

Q1	Q2
4.10224	4.788863

例 3: Williams の検定の計算

次のプログラムでは、8 つのブロックからなる乱塊法を使用し、7 つの水準である物質を検定しました。観測された処理の平均は次のとおりです。

処理	平均
X_0	10.4
X_1	9.9

処理	平均
X_2	10.0
X_3	10.6
X_4	11.4
X_5	11.9
X_6	11.7

自由度が (7 – 1)(8 – 1) = 42 である平均平方は、s^2 = 1.16 です。

平均化プロセスを経て最大尤度推定 M_i を求めます。

- $X_0 > X_1$ であるため、$X_{0,1} = (X_0 + X_1)/2$ = 10.15 となります。
- $X_{0,1} > X_2$ であるため、$X_{0,1,2} = (X_0 + X_1 + X_2)/3 = (2X_{0,1} + X_2)/3$ = 10.1 となります。
- $X_{0,1,2} < X_3 < X_4 < X_5$
- $X_5 > X_6$ であるため、$X_{5,6} = (X_5 + X_6)/2$ = 11.8 となります。

これで順序制約が満たされました。

対立仮説の最尤推定値は、次のようになります。

- $M_0 = M_1 = M_2 = X_{0,1,2}$ = 10.1
- $M_3 = X_3$ = 10.6
- $M_4 = X_4$ = 11.4
- $M_5 = M_6 = X_{5,6}$ = 11.8

$t = (11.8 - 10.4)/\sqrt{2s^2/8} = 2.60$を計算します。$k$ = 6、ν = 42、t = 2.60 の確率は、.9924467341 です。この物質に対する反応があるという強力な証拠を示しています。

```
prob=0.99244668715827
quant5=1.80656253603894
quant1=2.49090827298707
```

また、次の表に示すように、上の 5%および裾の 1%の分位点も計算できます。

ステートメント	結果
select probmc('williams',2.6,.,42,6) as prob;	0.992447
select probmc('williams',.,.95,42,6) as quant5;	1.806563
select probmc('williams',.,.99,42,6) as quant1;	2.490908

参考文献

Guirguis, G. H., and R. D. Tobias. 2004. “On the Computation of the Distribution for the Analysis of Means.” *Communications in Statistics: Simulation and Computation* 33: 861–887.

Nelson, P. R. 1981. “Numerical evaluation of an equi-correlated multivariate non-central t distribution.” *Communications in Statistics: Part B - Simulation and Computation* 10: 41–50.

Nelson, P.R. 1982a. “An Approximation for the Complex Normal Probability Integral.” *BIT* 22(1): 94–100.

Nelson, P. R. 1982b. “Exact critical points for the analysis of means.” *Communications in Statistics: Part A - Theory and Methods* 11: 699–709.

Nelson, P. R. 1988. “Application of the Analysis of Means.” *Proceedings of the SAS Users Group International Conference* 13: 225–230.

Nelson, P. R. 1991. “Numerical evaluation of multivariate normal integrals with correlations.” *The Frontiers of Statistical Scientific Theory and Industrial Applications* 2: 97–114.

Nelson, P. R. 1993. “Additional Uses for the Analysis of Means and Extended Tables of Critical Values.” *Technometrics* 35: 61–71.

PROBMED 関数

サンプル中央値の累積確率を計算します。

カテゴリ: 確率

返されるデータの種類: DOUBLE

構文

PROBMED(*n*, *x*)

必須引数

n
サンプルサイズを指定します。

データの種類 DOUBLE

x
関心対象です。つまり、PROBMED 関数は、中央値が *x* 以下である確率を計算します。

データの種類　DOUBLE

詳細

PROBMED 関数は、 *n 個*の独立した標準正規確率変数(平均 0、分散 1)のサンプルについて、サンプルの中央値が *x* 以下である確率を計算します。

n がサンプルサイズを表し、 $x_{(i)}$が *i* 次統計量を表すとします。次に、 *n* が奇数の場合、関数は次の計算を行います。

$$\Pr\left[X_{((n+1)/2)} \le x\right] = \mathrm{I}_{\Phi(x)}\left(\frac{n+1}{2}, \frac{n+1}{2}\right)$$

次の式では、前述の式を参照します。

$$\mathrm{I}_p(a,b) = \frac{\mathrm{I}}{\mathrm{B}(a,b)}\int_0^p t^{a-1}(\mathrm{I}-t)^{b-1}dt$$

式$\mathrm{B}(a,b) = \Gamma(a)\Gamma(b)/\Gamma(a+b)$の Γ(。)はガンマ関数です。*n* が偶数の場合、PROBMED 関数は次の計算を実行します。

$$\Pr[\frac{X_{(n/2)} + X_{((n/2)+1)}}{2} \le x] =$$

$$\frac{2}{\mathrm{B}(\frac{n}{2},\frac{n}{2})}\int_{-\infty}^{x}\{[\mathrm{I}-\Phi(u)]^{n/2} - [\mathrm{I}-\Phi(2x-u)]^{n/2}\}[\Phi(u)]^{(n/2)-1}\phi(u)du$$

この式では、$\mathrm{B}(n/2,\ n/2) = [\Gamma(n/2)]^2/\Gamma(n)$ 、および Φ(.)と φ(.)は、それぞれ、標準正規累積分布関数と密度関数です。

例

次のステートメントは、PROBMED 関数を示しています。

ステートメント	結果
select probmed(5,-0.1);	0.425638

参考文献

David, H.A. 1981. *Order Statistics*. 2nd 版 New York, , New York: John Wiley & Sons.

PROBNEGB 関数

負の二項分布の確率を返します。

カテゴリ: 確率

返されるデータの種類: DOUBLE

構文

PROBNEGB(*p*, *n*, *m*)

必須引数

p
数値の成功率パラメーターです。

範囲 $0 \leq p \leq 1$

データの種類 DOUBLE

n
成功数パラメーターです。この引数は整数にする必要があります。

範囲 $n \geq 1$

データの種類 DOUBLE

m
確率変数の失敗数です。この引数は正の整数にする必要があります。

範囲 $m \geq 0$

データの種類 DOUBLE

詳細

PROBNEGB 関数は、成功確率が *p*、成功数が *n* の負の二項分布のオブザベーションが *m* 以下となる確率を返します。

オブザベーションが指定した値の *m* と等しくなる確率を計算するには、負の二項分布からの 2 つの確率(*m* の確率と *m*-1 の確率)の差異を計算します。

例

次のステートメントは、PROBNEGB 関数を示しています。

ステートメント	結果
select probnegb(0.5,2,1);	0.5

PROBNORM 関数

標準正規分布の確率を返します。

カテゴリ: 確率

返されるデータの種類: DOUBLE

構文

PROBNORM(*x*)

必須引数

x
数値の確率変数です。

データの種類 DOUBLE

詳細

PROBNORM 関数は、標準正規分布からのオブザベーションが *x* 以下となる確率を返します。

注: PROBNORM は PROBIT の逆関数です。

例

次のステートメントは、PROBNORM 関数を示しています。

ステートメント	結果
select probnorm(1.96);	0.9750021049

PROBT 関数

式の値の t 分布から確率を返します。

カテゴリ: 集計
記述統計

返されるデータの種類: DOUBLE

構文

PROBT([DISTINCT] *expression*)

必須引数

expression
有効な SQL 式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

オプション引数

DISTINCT
集計計算から重複行を削除します。

詳細

PROBT 関数は、自由度が *n-1* で非心度パラメーター *nc* が 0 に等しいスチューデントの t 分布のオブザベーションが *expression* 以下となる確率を返します。

両側 t 検定の有意水準は、このコード行で求められます。

```
p=(1-probt(abs(x),df))*2;
```

集計関数を使用して、FROM 句にリストされているテーブル全体、または GROUP BY 句で指定されている各グループのデータの統計的要約を生成できます。GROUP

BY 句は、指定された 1 つまたは複数の列でデータをグループ化します。GROUP BY 句を使用する場合、SELECT 句または HAVING 句の集計関数は、各グループのデータを要約する方法を FedSQL に指示します。FedSQL は、グループごとに個別に集計関数を計算します。GROUP BY を省略すると、テーブルまたはビュー内のすべての行が単一のグループと見なされます。

比較

STUDENTS_T 関数は、スチューデントの t 分布を返します。PROBT 関数は、スチューデントの t 分布が特定の値以下である確率を返します。

例

テーブル: DENSITIES (1260 ページ)

次のステートメントは、PROBT 関数を示しています。

ステートメント	結果
select probt(density) from densities;	0.006068
select probt(population) from densities;	0.009722

関連項目

関数:

- “STUDENTS_T 関数” (840 ページ)

SELECT ステートメント句:

- “SELECT 句” (980 ページ)
- “GROUP BY 句” (993 ページ)
- “HAVING 句” (995 ページ)

PUT 関数

指定した出力形式を使用して値を返します。

カテゴリ: 特殊

返されるデータの種類: NVARCHAR VARBINARY BINARY

構文

PUT(*source*, *format*.)

必須引数

source
値を再フォーマットする変数または定数を識別します。

データの種類 format 引数でサポートされているデータ型

format.
ソースで指定された変数または定数に適用する SAS または FedSQL 出力形式が含まれます。

デフォルトの配置をオーバーライドするには、配置指定を出力形式に追加します。

-L
値を左揃えにする

-C
値を中央揃えにする

-R
値を右揃えにする

制限事項 出力形式は、*source* の値と同じ型である必要があります

詳細

PUT 関数は、それが指定されているクエリの出力に影響を与えます。指定された入力変数のデータ型を一時的に変更します。

PUT 関数の結果は常に文字列です。ソースが数値の場合、結果の文字列は右揃えになります。ソースが文字の場合、結果は左揃えになります。

PUT を使用して、定数をフォーマットし、格納されたデータを別の出力形式で出力します。数値を文字値に変換するには、PUT を使用します。詳細については、“PUT 関数で出力をフォーマットする方法” (82 ページ)を参照してください。

比較

CAST 関数は、指定された入力変数のデータ型を永続的に変更します。PUT 関数は、それが指定されているクエリの出力に影響を与えます。

例

テーブル: WORLDTEMPS (1277 ページ)

次のステートメントは、PUT 関数を示しています。

ステートメント	結果
select put(0.6666666667, fract8.);	2/3
select put(date(), date.);	19SEP13
select put(AvgLow, 4.1) from worldtemps;	45.0 33.0 17.0 68.0 56.0 57.0 28.0 51.0 75.0 36.0 33.0 25.0

関連項目

“PUT 関数で出力をフォーマットする方法” (82 ページ)

PVP 関数

満期時の元本払い戻しで、定期的なキャッシュフローストリーム(債権など)の現在価値を返します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

PVP(*A*, *c*, *n*, *K*, k_0, *y*)

必須引数

A
額面価格を指定します。

範囲 $A > 0$

$A > 0$

データの種類 DOUBLE

c

1 年当たりの名目クーポン率を分数で指定します。

範囲 $0 \le c < 1$

$0 \le c < 1$

データの種類 DOUBLE

n

1 年当たりのクーポン数を指定します。

範囲 $n > 0$

$n > 0$

データの種類 DOUBLE

K

クーポンの残数を指定します。

範囲 $K > 0$

$K > 0$

データの種類 DOUBLE

k_0

現在の日付から最初のクーポン日までの時間を指定します。年数で表します。

範囲 $0 < k_0 \le \frac{1}{n}$

$0 < k_0 \le 1/n$

データの種類 DOUBLE

y

1 年当たりの名目最終利回りを指定します。分数で表します。

範囲 $y > 0$

$y > 0$

データの種類 DOUBLE

詳細

PVP 関数は次の関係に基づきます。

$$P = \sum_{k=1}^{K} \frac{c(k)}{\left(1+\frac{y}{n}\right)^{t_k}}$$

前述の式には次の関係が適用されます。

- $t_k = nk_0 + k - 1$
- $c(k) = \frac{c}{n}A \quad for\ k = 1, \ldots, K-1$
- $c(K) = \left(1+\frac{c}{n}\right)A$

例

次のステートメントは、PVP 関数を示しています。

ステートメント	結果
select pvp(1000,.01,4,14,.33/2,.10);	743.1676

QTR 関数

SAS 日付値から年の四半期を返します。

カテゴリ: 日付と時間

返されるデータの種類: DOUBLE

構文

QTR(*date*)

必須引数

date
SAS 日付値を表す有効な式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

QTR 関数は、SAS 日付値から、日付値が含まれる四半期を示す値 1、2、3、4 を返します。

FedSQL での日付値と時間値の処理方法の詳細については、“FedSQL の日付と時間” (59 ページ)を参照してください。

例

次のステートメントは、QTR 関数を示しています。

ステートメント	結果
select qtr(16983);	3
select qtr(17075);	4

関連項目

関数:

- “YYQ 関数” (893 ページ)

QUANTILE 関数

左側確率(CDF)を指定するときに分布から分位点を返します。

カテゴリ: 分位点

返されるデータの種類: DOUBLE

参照項目: “CDF 関数” (334 ページ)

構文

QUANTILE(*'distribution'*, *probability*, *parameter-1, ..., parameter-k*)

必須引数

distribution
分布を特定する文字定数、変数または式です。

注: 各 QUANTILE 分布関数の引数は、対応する CDF 分布関数の引数と同じです。

有効な分布は次のとおりです。

分布	引数
Bernoulli	'BERNOULLI'
ベータ	'BETA'
二項	'BINOMIAL'
Cauchy	'CAUCHY'
カイ 2 乗	'CHISQUARE'
Conway-Maxwell-Poisson	'CONMAXPOI'
指数	'EXPONENTIAL'
F	'F'
ガンマ	'GAMMA'
一般化 Poisson	'GENPOISSON'
幾何	'GEOMETRIC'
超幾何	'HYPERGEOMETRIC'
Laplace	'LAPLACE'
ロジスティック	'LOGISTIC'
対数正規	'LOGNORMAL'
負の二項	'NEGBINOMIAL'
正規	'NORMAL'|'GAUSS'

分布	引数
正規混合	'NORMALMIX'
パレート	'PARETO'
Poisson	'POISSON'
T	'T'
Tweedie	'TWEEDIE'
一様	'UNIFORM'
Wald (逆ガウス)	'WALD'|'IGAUSS'
Weibull	'WEIBULL'

注: T、F および NORMALMIX を除き、最初の 4 文字で分布を最小限に識別できます。

probability
確率変数の値を指定する、数値定数、変数または式です。

parameter-1, ..., parameter-k
特定の分布に適した任意の *shape*、*location* または *scale* パラメーターです。

詳細

QUANTILE 関数は、指定された確率値に基づいて、指定された連続分布または離散分布から分位点を計算します。詳細については、前述の表に記載されている個々の分布を参照してください。

QUANTILE 関数の Conway-Maxwell-Poisson 分布は、CDF 値が *p* 以下の最大の整数であるカウント値 *y* を返します。QUANTILE 関数の Conway-Maxwell-Poisson 分布の構文は、次の形式です。

QUANTILE('CONMAXPOI',*p*,*λ*,*v*)

p
0 と 1 の間の実数(両端の値を含める)。

λ
Poisson 分布の場合と同様で、平均に似ています。

v
ばらつきのパラメーターです。

詳細については、PDF 関数の“Conway-Maxwell-Poisson” (723 ページ)を参照してください。

例

次のステートメントは、QUANTILE 関数を示しています。

ステートメント	結果
select quantile('BERN', .75, .25);	0
select quantile('BETA', 0.1, 3, 4);	0.200909
select quantile('BINOM', .4, .5, 10);	5
select quantile('CAUCHY', .85);	1.962611
select quantile('CHISQ', .6, 11);	11.52983
select quantile('CONMAXPOI', .2, 2.3, .4);	5
select quantile('EXPO', .6);	0.916291
select quantile('F', .8, 2, 3);	2.886027
select quantile('GAMMA', .4, 3);	2.285077
select quantile('GENPOISSON', .9, 1, .7);	9
select quantile('HYPER', .5, 200, 50, 10);	2
select quantile('LAPLACE', .8);	0.916291
select quantile('LOGISTIC', .7);	0.847298
select quantile('LOGNORMAL', .5);	1
select quantile('NEGB', .5, .5, 2);	1
select quantile('NORMAL', .975);	1.959964
select quantile('NORMALMIX', 0.5, 1, 0.2, 1.1,0.1);	1.1
select quantile('PARETO', .01, 1);	1.010101
select quantile('POISSON', .9, 1);	2
select quantile('T', .8, 5);	0.919544
select quantile('TWEEDIE', .8, 5);	1.26112
select quantile('UNIFORM', 0.25);	0.25
select quantile('WALD', .6, 2);	0.952621

ステートメント	結果
select quantile('WEIBULL', .6, 2);	0.957231

関連項目

関数:

- “CDF 関数” (334 ページ)
- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “PDF 関数” (712 ページ)
- “SDF 関数” (819 ページ)
- “SQUANTILE 関数” (832 ページ)

QUOTE 関数

文字値に二重引用符を付加します。

カテゴリ: 文字

返されるデータの種類: CHAR、NCHAR、NVARCHAR、VARCHAR

参照項目: “FedSQL での文字型を返す関数の使用” (260 ページ)

構文

QUOTE(*expression*)

必須引数

expression
文字列に評価されるか強制変換できる有効な式を指定します。

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

QUOTE 関数は、文字値にデフォルト文字である二重引用符を付加します。引数内から二重引用符が検出されると、さらに二重にして出力されます。

受け取る変数には、引数(末尾の空白も含む)、先頭および末尾の引用符、二重にされた埋め込み引用符を含む十分な長さが必要です。たとえば、引数が ABC でその末尾に 3 つの空白が続く場合、"ABC###"を保持するために受け取る変数の長さは 8 文字以上であることが必要です。(文字#は空白を表します)。受け取るフィールドが長すぎると、QUOTE 関数は空白の文字列を返し、無効な引数というメモを SAS ログに書き込みます。

二重引用符で囲まれた文字列は DS2 識別子であり、文字定数ではありません。二重引用符は識別子の一部になります。引用符で囲まれた識別子を使用して、出力テーブルに列名を作成することはできません。

例

次のステートメントは、QUOTE 関数を示しています。

ステートメント	結果
select quote('A"B');	"A""B"
select quote('A''B');	"A'B"
select quote('Paul''s Catering Service');	"Paul's Catering Service"

RADIANS 関数

数値の度数から変換されたラジアン数を返します。

カテゴリ: 三角関数

制限事項: RADIANS 関数で許可される最大指数は 308 です。

返されるデータの種類: DOUBLE

注: STRING データを DOUBLE に変換するときに指数表記の計算に使用するアルゴリズムが異なるため、Linux ホストと 64 ビット Windows ホストでは、入力値 1e-306、1e-307、および 1e-308 に対して異なる出力値が発生します。

構文

RADIANS(*expression*)

必須引数

expression
数値に評価される有効な SQL 式を指定します。

データの種類 BIGINT、DOUBLE、FLOAT、INTEGER、REAL、SMALLINT、TINYINT

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

例

次のステートメントは、RADIANS 関数を示しています。

ステートメント	結果
select radians(360);	6.283185

関連項目

関数:

- “DEGREES 関数” (444 ページ)

RANGE 関数

式の値の間の範囲を返します。

カテゴリ: 集計
記述統計

返されるデータの種類: 式と同じ型

構文

RANGE([DISTINCT] *expression*)

必須引数

expression
有効な SQL 式を指定します。

データの種類 BIGINT、DOUBLE、FLOAT、INTEGER、REAL、SMALLINT、TINYINT

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

オプション引数

DISTINCT
集計計算から重複行を削除します。

詳細

RANGE 関数は、指定された式の最大値と最小値の差を返します。RANGE 関数は、null 値と SAS 欠損値を無視します。

集計関数を使用して、FROM 句にリストされているテーブル全体、または GROUP BY 句で指定されている各グループのデータの統計的要約を生成できます。GROUP BY 句は、指定された 1 つまたは複数の列でデータをグループ化します。GROUP BY 句を使用する場合、SELECT 句または HAVING 句の集計関数は、各グループのデータを要約する方法を FedSQL に指示します。FedSQL は、グループごとに個別に集計関数を計算します。GROUP BY を省略すると、テーブルまたはビュー内のすべての行が単一のグループと見なされます。

例

テーブル: WORLDTEMPS (1277 ページ)

次のステートメントは、RANGE 関数を示しています。

ステートメント	結果
select range(AvgHigh) from worldtemps;	27

関連項目

SELECT ステートメント句:

- “SELECT 句” (980 ページ)
- “GROUP BY 句” (993 ページ)

- “HAVING 句” (995 ページ)

RANK 関数

ASCII 照合順序で文字の位置を返します。

カテゴリ: 文字

返されるデータの種類: DOUBLE

構文

RANK(*expression*)

必須引数

expression
文字列に評価されるか強制変換できる有効な式を指定します。

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

RANK 関数は、ASCII 照合順序で文字の位置を表す整数を返します。2 つ以上の文字が指定されている場合、RANK 関数は、最初の文字の ASCII 照合順序での位置を返します。

注: ASCII 127 より上の文字を指定して RANK 関数を使用するプログラムはいずれも移植性がありません。(16 進表記は**'7F'x** です。)プログラムに移植性がないのは、これらの文字が国別文字であり、国によって異なるためです。

例

次のステートメントは、RANK 関数を示しています。

ステートメント	結果	
	ASCII	EBCDIC
select "rank"('A');	65	193

注: RANK は FedSQL の予約語です。RANK 関数は、区切り ID として指定する必要があります。

関連項目

関数:

- “BYTE 関数” (331 ページ)

REPEAT 関数

文字式を繰り返します。

カテゴリ: 文字

返されるデータの種類: CHAR、NCHAR、NVARCHAR、VARCHAR

参照項目: “FedSQL での文字型を返す関数の使用” (260 ページ)

構文

REPEAT(*expression*, *n*)

必須引数

expression
文字列に評価されるか強制変換できる有効な式を指定します。

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

n
expression を繰り返す回数を指定します。

制限事項 *n* は、0 以上である必要があります。

データの種類 BIGINT、DOUBLE

詳細

REPEAT 関数は、*n* 回繰り返された第 1 引数で構成される文字値を返します。つまり、第 1 引数が *n*+1 回、結果に表示されます。

例

次のステートメントは、REPEAT 関数を示しています。

ステートメント	結果
select repeat('ONE',2);	ONEONEONE

REVERSE 関数

文字式を逆にします。

カテゴリ: 文字

返されるデータの種類: CHAR、NCHAR、NVARCHAR、VARCHAR

構文

REVERSE(*expression*)

必須引数

expression
文字列に評価されるか強制変換できる有効な式を指定します。

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

REVERSE 関数が返す文字値では、式の最後の文字が結果の最初の文字になり、式の最後から 2 番目の文字が結果の 2 番目の文字になります(以下同様です)。

注: 式の末尾の空白は結果の先頭の空白になります。

例

次のステートメントは、REVERSE 関数を示しています。

ステートメント	結果
	----+----1
select reverse('xyz ');	zyx

RMS 関数

2 乗平均平方根を返します。

カテゴリ: 記述統計

返されるデータの種類: DOUBLE

構文

RMS(*expression* [, ...*expression*])

必須引数

expression
数値に評価される有効な式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

2 乗平均平方根は、値の 2 乗の算術平均の平方根です。すべての引数が null または欠損値の場合、結果は null または欠損値になります。それ以外の場合、結果は非 null または非欠損値の 2 乗平均平方根になります。

nを非 null または非欠損値を持つ引数の数とし、$x_1, x_2, \ldots, x_n$をそれらの引数の値とします。2 乗平均平方根は次のように計算されます。

$$\sqrt{\frac{x_1^2 + x_2^2 + \ldots + x_n^2}{n}}$$

例

次のステートメントは、RMS 関数を示しています。

ステートメント	結果
select rms(1,7);	5
select rms(.,1,5,11);	7

ROUND 関数

第 1 引数を第 2 引数の最も近い倍数(第 2 引数が省略された場合は最も近い整数)に丸めます。

カテゴリ: 切り捨て

返されるデータの種類: DOUBLE

構文

ROUND(*expression* [, *rounding-unit*])

必須引数

expression
数値に評価される丸め対象の有効な式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

オプション引数

rounding-unit
丸め単位を示す正の数値式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

基本概念

ROUND 関数は、第 1 引数を第 2 引数の倍数の近似値に丸めます。結果は第 2 引数の正確な倍数でない場合があります。

2 進算術演算と 10 進算術演算の差異

コンピューターは、有限精度の 2 進算術演算を使用します。通常、厳密な 2 進表現がない数値を扱う場合、コンピューターによって生成される結果は 10 進算術演算で生成される結果とわずかに異なります。

たとえば、10 進値の 0.1 および 0.3 には厳密な 2 進表現はありません。10 進算術演算の 3*0.1 は厳密に 0.3 になりますが、2 進算術演算ではこの等式は成り立ちません。

丸めの影響

丸める値に最も近い丸め単位の厳密な倍数を見つけることが丸めの定義となります。たとえば、10 進算術演算の場合、0.33 を最も近い 0.1 の倍数に丸めると、3*0.1 つまり 0.3 になります。2 進算術演算では 0.3 が 0.1 の厳密な倍数ではないため、2 進算術演算の場合、0.33 を最も近い 0.1 の倍数に丸めると 3*0.1 になりますが、0.3 にはなりません。

ROUND 関数は、数学的に正しい厳密な結果でなくても 10 進算術演算に基づいて値を返します。ROUND(0.33,0.1)の例では、ROUND は 3*0.1 ではなく 0.3 を返します。

近似等式のテスト

近似等式をテストする一般的な方法として ROUND 関数を使用しないでください。最下位ビットのみが異なる 2 つの数値を丸める場合、一方の数値が切り下げられてもう一方の数値が切り上げられると、異なる値に丸められる可能性があります。近似等式のテストは、数値の計算方法によって異なります。両方の数値が高い相対精度で計算されている場合は、ABS および MAX 関数を使用して近似等式をテストできます。

期待される結果の生成

通常、ROUND(expression, rounding-unit)は、結果の有効桁数が 9 以下で、次のどの条件にも該当しない場合、10 進算術演算で期待される結果を生成します。

- 丸め単位が整数である。
- 丸め単位が 1e-15 以上の 10 のべき乗である。 [1]
- 10 進算術演算で期待される結果が小数第 4 位以下である。

次に例を示します。

```
select round(1234.56789,100);          Result: 1200
select round(1234.56789,10);           Result: 1230
select round(1234.56789,1);            Result: 1235
select round(1234.56789,.1);           Result: 1234.6
select round(1234.56789,.01);          Result: 1234.57
select round(1234.56789,.001);         Result: 1234.568
select round(1234.56789,.0001);        Result: 1234.5679
select round(1234.56789,.00001);       Result: 1234.56789
select round(1234.56789,.1111);        Result: 1234.5432
select round(1234.56789,.11111);       Result: 1234.54321
```

丸め単位が整数の逆数である場合

丸め単位が整数の逆数である場合 [2]ROUND 関数は整数で除算して結果を計算します。そのため、丸め単位の倍数ではなく 2 つの整数の比率を使用して ROUND の結果を比較できます。

特殊なケースの計算結果

ROUND 関数は、次の条件にすべて該当する場合に丸め単位で整数を乗算して結果を計算します。

- 丸め単位が整数でない。
- 丸め単位が 10 のべき乗でない。
- 丸め単位が整数の逆数でない。
- 10 進算術演算で期待される結果が小数第 4 位以下である。

次に例を示します。

```
select round(1234.56789,.11111) - 11111*.11111;
```

値 0 (ゼロ)を返します。

値が丸め単位の倍数の中間にある場合の計算結果

丸める値が丸め単位の 2 つの倍数のほぼ中間にある場合、ROUND 関数は絶対値を切り上げ、符合を元に戻します。

1. 丸め単位が 1 よりも小さい場合、ROUND は、丸め単位の逆数が 10 のべき乗との違いが最下位から第 3 または第 4 ビットまでであれば、丸め単位を 10 のべき乗として扱います。
2. ROUND は、丸め単位の逆数と整数の違いが最下位から第 3 または第 4 ビットまでであれば、丸め単位を整数の逆数として扱います。

例

次のステートメントは、ROUND 関数を示しています。

ステートメント	結果
select round(9.5,10);	10

関連項目

関数:

- “CEIL 関数” (380 ページ)
- “CEILZ 関数” (381 ページ)
- “FLOOR 関数” (542 ページ)
- “FLOORZ 関数” (543 ページ)

ROUNDZ 関数

第 1 引数を第 2 引数の最も近い倍数にゼロファジーを使用して丸めます。

カテゴリ: 切り捨て

返されるデータの種類: DOUBLE

構文

ROUNDZ(*expression* [, *rounding-unit*])

必須引数

expression
数値に評価される有効な式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

オプション引数

rounding-unit
数値式に評価され、丸め単位を示す有効な式を指定します。

デフォルト	1
要件	正の値のみが有効です。
データの種類	DOUBLE
参照項目:	“<sql-expression> 式” (918 ページ)
	“FedSQL 式” (46 ページ)

詳細

ROUNDZ 関数は、第 1 引数を第 2 引数の最も近い倍数に丸めます。

比較

ROUNDZ 関数は、ROUND 関数と同じですが、次の点が異なります。

- ROUNDZ は、第 1 引数が第 2 引数の最も近い 2 つの倍数のちょうど中間にある場合には偶数の倍数を返します。ROUND は、第 1 引数が最も近い 2 つの倍数のほぼ中間にある場合には絶対値が大きい方の倍数を返します。
- 丸め単位が 1 よりも小さい場合や整数の逆数でない場合、ROUNDZ によって返される結果は 10 進算術演算の結果と完全には一致しない可能性があります。ROUNDZ は、結果をファジー処理しません。ROUND は、結果が 10 進算術演算に一致するようにファジーと呼ばれる追加の計算を実行します。

例: ROUNDZ 関数の出力例

次のステートメントは、ROUNDZ 関数を示しています。

ステートメント	結果
select put(roundz(223.456, 1), 9.5);	223.00000
select put(roundz(223.456, .01), 9.5);	223.46000
select put(roundz(223.456, 100), 9.5);	200.00000
select put(roundz(223.456, .3), 9.5);	223.50000

関連項目

関数:

- “ROUND 関数” (805 ページ)

SAVING 関数

定期預金の将来価値を返します。

カテゴリ:	財務
返されるデータの種類:	DOUBLE

構文

SAVING(*f*, *p*, *r*, *n*)

必須引数

f
数値の将来総額です(*n* 期間の終了時)。

範囲	$f \geq 0$
データの種類	DOUBLE

p
数値の固定定期的支払いです。

範囲	$p \geq 0$
データの種類	DOUBLE

r
10 進で表される数値の定期利率です。

範囲	$r \geq 0$
データの種類	DOUBLE

n
整数の複利期間数です。

範囲	$n \geq 0$
データの種類	DOUBLE

詳細

SAVING 関数は、定期預金の 4 つの引数のリストの欠損引数を返します。これらの引数には次の式の関係があります。

$$f = \frac{p(1+r)\left((1+r)^n - 1\right)}{r}$$

引数は 1 つを欠損値とする必要があります。次に、残りの 3 つから計算されます。結果を変換して丸めた数字にする調整は行われません。

例

ある定期預金口座は 5%の名目年利が支払われ、毎月複利計算されます。毎月$100 を入金する場合、少なくとも累計が$12,000 になるために必要な支払い回数は、次のように表すことができます。

```
select saving(12000, 100, .05/12, .);
```

返される値は 97.18 か月です。第 4 引数には欠損値が設定されます。これは、支払い回数を計算することを示します。5%の名目年利は、0.05/12 の月利に変換されます。rate は分数(パーセントではない)で表す複利計算期間当たりの利率です。

関連項目

関数:

- “SAVINGS 関数” (811 ページ)

SAVINGS 関数

変動金利を使用して定期預金の残高を返します。

カテゴリ:	財務
返されるデータの種類:	DOUBLE

構文

SAVINGS(*base-date*, *initial-deposit-date*, *deposit-amount*, *deposit-number*, *deposit-interval*, *compounding-interval*, *date-1*, *rate-1*[, ...*date-n*, *rate-n*])

必須引数

base-date
返される値が基準日の預金残高になるように指定します。

要件 *base-date* は SAS 日付です。

データの種類 DOUBLE

initial-deposit-date
初回入金日を指定します。後続の入金間隔の開始時に次の入金が行われます。

要件 *initial-deposit-date* は SAS 日付です。

データの種類 DOUBLE

deposit-amount
各入金の値を指定します。すべての 入金は定数を前提としています。

データの種類 DOUBLE

deposit-number
入金の回数を指定します。

データの種類 DOUBLE

deposit-interval
入金の頻度を指定します。

要件 *deposit-interval* は SAS 間隔です。

データの種類 CHAR

compounding-interval
複利間隔を指定します。

要件 *compounding-interval* は SAS 間隔です。

データの種類 CHAR

date
rate が有効になるタイミングを指定します。各日付は利率とペアになります。

要件 *date* は SAS 日付です。

データの種類 DOUBLE

rate
date に開始する利率を数値のパーセンテージで指定します。各利率は日付とペアになります。

データの種類 DOUBLE

詳細

SAVINGS 関数には、次の詳細情報が適用されます。

- 利率の値は–99 から 120 までの値にする必要があります。
- *deposit-interval* は'CONTINUOUS'にすることはできません。
- 日付-利率ペアのリストは日付順にする必要はありません。
- 1 日に複数の利率が変更される場合、SAVINGS 関数は、その日付でリストされている最後の利率のみを適用します。
- 一部の期間には単利が適用されます。
- *initial-deposit-date* と *base-date* の両方の日付またはその前の日付となる有効な日付-利率ペアが必要になります。

例

- 年利 4%で年 4 回複利計算される口座に 2 年間毎月$300 を入金する場合、5 年後の口座残高は次のように表すことができます。

  ```
  select savings(date'2005-01-01', date'2000-01-01',
    300, 24, 'month', 'qtr', date'2000-01-01',4.00);
  ```

  次の行が SAS ログに書き込まれます。

  ```
  8458.794159
  ```

- 金利が毎年 1/4 ポイント上昇すると、口座の残高は次のように表すことができます。

  ```
  select savings(date'2005-01-01', date'2000-01-01',
    300, 24, 'month', 'qtr', date'2000-01-01', 4.00, date'2001-01-01', 4.25,
   date'2002-01-01', 4.50, date'2003-01-01', 4.75, date'2004-01-01', 5.0);
  ```

  次の行が SAS ログに書き込まれます。

  ```
  8665.5059376
  ```

- 1 年間の預金の後の残高を決定するために、次のプログラムは amount_base3 を目的の残高に設定します。

  ```
  select savings(date'2001-01-01', date'2000-01-01',
    300, 24, 'month', 'qtr', date'2000-01-01',4);
  ```

  次の行が SAS ログに書き込まれます。

  ```
  3978.69037121739
  ```

SAVINGS 関数は、基準日後の入金を無視するため、基準日後の入金は戻り値に影響しません。

関連項目

関数:

- “SAVING 関数” (810 ページ)

SCAN 関数

文字式から *n* 番目の単語を返します。

カテゴリ: 文字

返されるデータの種類: CHAR、NCHAR、NVARCHAR、VARCHAR

構文

SCAN(*expression*, *n* [, *delimiters*[, *modifier*]])

必須引数

expression
文字列に評価されるか強制変換できる有効な式を指定します。

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

n
SCAN で選択する文字式の単語番号を指定するゼロ以外の数値式です。次のルールが適用されます。*n* が正の場合、SCAN は文字列内の単語を左から右にカウントします。*n* が負の場合、SCAN は文字列内の単語を右から左にカウントします。*n* が *expression* の単語数より大きい場合、SCAN は空白値を返します。

SCAN で選択する文字式の単語番号を指定するゼロ以外の数値式です。次の規則が適用されます。

- *n* が正の場合、SCAN は文字列内の単語を左から右にカウントします。
- *n* が負の場合、SCAN は文字列内の単語を右から左にカウントします。
- *n* が *expression* の単語数より大きい場合、SCAN は空白値を返します。

オプション引数

delimiters
文字列を評価するか文字列に強制変換できて、SCAN で式の単語区切りとして使用される有効な式を指定します。

デフォルト

要件 *delimiter* が定数の場合は、*delimiter* を単一引用符で囲みます。

操作 ASCII のデフォルトの区切り文字は、空白 ! $ % & ( ) * + , - . / ; < | です。^文字のない環境では、SCAN はかわりに~文字を使用します。

修飾子を指定すると、*delimiter* の文字を変更できます。たとえば、*modifier* 引数で K 修飾子を指定すると、*delimiter* に含まれていないすべての文字が区切り文字として使用されます。

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

ヒント 他の修飾子を使用して、*delimiter* にさらに文字を追加できます。

参照項目: “ASCII 環境でデフォルトの区切り文字を使用する” (817 ページ)

“<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

modifier

文字定数、変数または式を指定します。これらの空白以外の各文字で SCAN 関数のアクションを変更します。空白は無視され、大文字と小文字は区別されません。次の文字を修飾子として使用します。

A

文字のリストにアルファベット文字を追加します。

B

count 引数の符号に関係なく、左から右ではなく、右から左に逆方向にスキャンします。

C

文字のリストに制御文字を追加します。

D

文字のリストに数字を追加します。

F

文字のリストにアンダースコアと 英字を追加します。

G

文字のリストにグラフィカル文字を追加します。グラフィカル文字は、紙面に印刷するとイメージになる文字です。

H

文字のリストに水平タブを追加します。

I

大文字か小文字かは無視します。

K

文字のリストに含まれていないすべての 文字を区切り文字として扱うようにします。つまり、K が指定されている場合、文字のリストにある文字は、区切り文字であるために省略されるのではなく、返された値に保持されます。K を指定しない場合、文字のリストに含まれているすべての 文字が区切り文字として扱われます。

L
小文字を文字リストに追加します。

M
複数の連続する区切り文字、および *string* 引数の先頭または末尾の区切り文字が、長さがゼロの単語を参照するように指定します。M 修飾子を指定しない場合、複数の連続する区切り文字は 1 つの区切り文字として扱われ、*string* 引数の先頭または末尾の区切り文字は無視されます。

N
文字のリストに数字、アンダースコア、および英字を追加します。

O
charlist および *modifier* 引数を 1 回だけ処理します。SCAN 関数の呼び出し時毎には処理されません。DATA ステップ(WHERE 句以外)または SQL プロシジャで O 修飾子を使用すると、*character-list* 引数および *modifier* 引数に変更がないループで SCAN が呼び出されるとき、より迅速に実行できます。O 修飾子は SAS コードの SCAN 関数の各インスタンスに個別に適用され、SCAN 関数のすべてのインスタンスで 同じ区切り文字および修飾子が使用されるようには*なりません*。

P
文字のリストに句読点を追加します。

Q
引用符で囲まれた部分文字列内の区切り文字を無視します。*string* 引数の値に、対になっていない引用符が含まれている場合、左から右へのスキャンと、右から左へのスキャンとでは異なる単語が生成されます。

R
SCAN が返す単語から先頭および末尾の空白を削除します。Q 修飾子と R 修飾子を指定すると、SCAN 関数は単語から先頭および末尾の空白をまず削除します。次に、SCAN は単語が引用符で始まる場合には単語から引用符の層も 1 つ削除します。

S
文字のリストに空白文字(空白、水平タブ、垂直タブ、キャリッジリターン、ラインフィード、フォームフィード)を追加します。

T
string 引数と *charlist* 引数から末尾の空白をトリムします。両方の文字引数ではなく 1 つの文字引数からのみ末尾の空白を削除する場合、T 修飾子を指定して SCAN 関数を使用するかわりに TRIM 関数を使用します。

U
大文字を文字リストに追加します。

W
印刷可能文字を文字リストに追加します。

X
文字のリストに 16 進文字を追加します。

制限事項　この引数は、SAS Viya プラットフォームでのみサポートされます。

ヒント *modifier* 引数が文字定数の場合は、引数を引用符で囲みます。一組の引用符で複数の修飾子を指定します。引数 *modifier* には、文字変数や文字式も指定できます。

詳細

"区切り文字"と"単語"の定義

*区切り文字*とは、単語を区切るために使用される複数の文字のどれかです。*delimiter* 引数と *modifier* 引数で区切り文字を指定できます。

Q 修飾子を指定すると、引用符で囲まれた部分文字列内部の区切り文字は無視されます。

SCAN 関数では、"単語"は、次の特徴をすべて備えた部分文字列を示します。

- 左側が区切り文字で囲まれているか、文字列の先頭である
- 右側が区切り文字で囲まれているか、文字列の末尾である
- 区切り文字が含まれていない

文字列の先頭または末尾に区切り文字がある場合、または文字列に 2 つ以上の連続する区切り文字が含まれている場合、単語の長さがゼロになることがあります。ただし、SCAN 関数では、M 修飾子を指定しなければ、長さがゼロの単語は無視されます。

注: ”単語”の定義は SCAN 関数と COUNTW 関数に共通します。

ASCII 環境でデフォルトの区切り文字を使用する

2 つの引数のみを指定して SCAN 関数を使用する場合、デフォルトの区切り文字は、コンピューターで使用している文字(ASCII または EBCDIC)によって異なります。

- お使いのコンピューターが ASCII 文字を使用している場合、デフォルトの区切り文字は次のとおりです。

  空白 ! $ % & ( ) * + , - ./ ; < ^ |

区切り文字とする文字を指定せずに *modifier* 引数を使用すると、使用される区切り文字は *modifier* 引数で定義される区切り文字のみになります。この場合、ASCII 環境のデフォルトの区切り文字のリストは使用されません。つまり、修飾子は、*delimiter* 引数によって明示的に指定された区切り文字のリストに追加されます。修飾子は、デフォルトの修飾子のリストには追加しません。

結果の長さ

式の最初の単語の前にある先頭の区切り文字は、SCAN に影響しません。連続する区切り文字が 2 つ以上ある場合、SCAN はそれらを 1 つとして扱います。

SCAN 関数によって返される単語の最小長は、M 修飾子が指定されているかどうかによって異なります。“M 修飾子ありで SCAN 関数を使用” (818 ページ)を参照してください。“M 修飾子なしで SCAN 関数を使用” (818 ページ)も参照してください。

M 修飾子ありで SCAN 関数を使用

M 修飾子を指定すると、文字列内の単語数は文字列内の区切り文字数に 1 を足した数になります。ただし、Q 修飾子を指定すると、引用符内の区切り文字は無視されます。

M 修飾子を指定すると、SCAN 関数は次のいずれかの条件に該当する場合に長さがゼロの単語を返します。

- 文字列の先頭が区切り文字であり、ユーザーが最初の単語を要求した場合
- 文字列の末尾が区切り文字であり、ユーザーが最後の単語を要求した場合
- 文字列が 2 つの連続する区切り文字を含んでおり、ユーザーがこれら 2 つの区切り文字間にある単語を要求した場合

M 修飾子なしで SCAN 関数を使用

M 修飾子を指定しない場合、文字列内の単語数は連続する非区切り文字の最大部分文字列数になります。ただし、Q 修飾子を指定すると、引用符内の区切り文字は無視されます。

M 修飾子を指定しないと、SCAN 関数は次のように動作します。

- 文字列の最初または最後の区切り文字は無視されます。
- 2 つ以上の連続する区切り文字を、単一の区切り文字のように扱います。

文字列に区切り文字以外の文字が含まれていない場合や、文字列の単語数よりも絶対値の大きい count を指定する場合、SCAN 関数は次のいずれかの項目を返します。

- 1 つの空白(DATA ステップから SCAN 関数を呼び出す場合)
- 長さがゼロの文字列(マクロプロセッサから SCAN 関数を呼び出す場合)

null 引数の使用

SCAN 関数では、文字引数を null に指定できます。null 引数は長さがゼロの文字列として扱われます。数値引数は null にできません。

SBCS および DBCS データの処理

SCAN 関数は、SBCS データを処理するように設計されていますが、DBCS データを処理することもできます。基準は次のとおりです。

- *expression* が VARCHAR として宣言されておらず、シングルバイトデータを処理している場合、SCAN は SBCS を処理します。
- *string* が VARCHAR として宣言されていて、マルチバイトデータを処理している場合、SCAN は DBCS を処理します。

例: 基本的な SCAN 関数の使用法

次のステートメントは、SCAN 関数を示しています。

ステートメント	結果
select scan('ABC.DEF(X=Y)', 3);	X=Y
select scan('ABC.DEF(X=Y)', -3);	ABC

SDF 関数

生存関数を返します。

カテゴリ: 確率

参照項目: “CDF 関数” (334 ページ)

構文

SDF(*'distribution'*, *quantile*, *parameter-1, ..., parameter-k*)

必須引数

distribution
分布を識別する文字列です。

注: 各 SDF 分布関数の引数は、対応する CDF 分布関数の引数と同じです。

有効な分布は次のとおりです。

分布	引数
Bernoulli	'BERNOULLI'
ベータ	'BETA'
二項	'BINOMIAL'
Cauchy	'CAUCHY'
カイ 2 乗	'CHISQUARE'
Conway-Maxwell-Poisson	'CONMAXPOI'
指数	'EXPONENTIAL'

分布	引数
F	'F'
ガンマ	'GAMMA'
一般化 Poisson	'GENPOISSON'
幾何	'GEOMETRIC'
超幾何	'HYPERGEOMETRIC'
Laplace	'LAPLACE'
ロジスティック	'LOGISTIC'
対数正規	'LOGNORMAL'
負の二項	'NEGBINOMIAL'
正規	'NORMAL'|'GAUSS'
正規混合	'NORMALMIX'
パレート	'PARETO'
Poisson	'POISSON'
T	'T'
Tweedie	'TWEEDIE'
一様	'UNIFORM'
Wald (逆ガウス)	'WALD'|'IGAUSS'
Weibull	'WEIBULL'

注 T、F および NORMALMIX を除き、最初の 4 文字で分布を最小限に識別できます。

quantile
確率変数の値を指定する、数値定数、変数または式です。

データの種類 DOUBLE

parameter-1, ..., parameter-k
特定の分布に適した任意の *shape*、*location* または *scale* パラメーターです。

データの種類 DOUBLE

詳細

SDF 関数は、さまざまな連続分布および離散分布の生存関数(上裾)を計算します。詳細については、前述の表にリストされている個々の分布を参照してください。

Conway-Maxwell-Poisson 分布の SDF 関数の形式は次のとおりです。

SDF('CONMAXPOI',*y*,*λ*,*ν*)

y
負でない整数でカウントデータを表します。

λ
Poisson 分布の場合と同様で、平均に似ています。

ν
ばらつきのパラメーターです。

SDF 関数は、カウント値が *y* より大きくなる確率を返します。

詳細については、PDF 関数の“Conway-Maxwell-Poisson” (723 ページ)を参照してください。

例

次のステートメントは、SDF 関数を示しています。

ステートメント	結果
select sdf('BERN', 0, .25);	0.25
select sdf('BETA', 0.2, 3, 4);	0.90112
select sdf('BINOM', 4, .5, 10);	0.623047
select sdf('CAUCHY', 2);	0.147584
select sdf('CHISQ', 11.264, 11);	0.421419
select sdf('CONMAXPOI', 12, 2.3, .4);	0.197051
select sdf('EXPO', 1);	0.367879
select sdf('F', 3.32, 2,3);	0.173607
select sdf('GAMMA', 1, 3);	0.919699
select sdf('GENPOISSON', .9, 1, .7);	0.632121

ステートメント	結果
select sdf('GEOMETRIC', 5, .3);	0.117649
select sdf('HYPER', 2, 200, 50, 10);	0.476327
select sdf('LAPLACE', 1);	0.18394
select sdf('LOGISTIC', 1);	0.268941
select sdf('LOGNORMAL', 1);	0.5
select sdf('NEGB', 1, .5, 2);	0.5
select sdf('NORMAL', 1.96);	0.024998
select sdf('NORMALMIX', 2.3, 3, .33, .33, .34, .5, 1.5, 2.5, .79, 1.6, 4.3);	0.281869
select sdf('PARETO', 1, 1);	1
select sdf('POISSON', 2, 1);	0.080301
select sdf('T', .9, 5);	0.204686
select sdf('TWEEDIE', .8, 5);	0.408237
select sdf('UNIFORM', 0.25);	0.75
select sdf('WALD', 1, 2);	0.372302
select sdf('WEIBULL', 1, 2);	0.367879

関連項目

関数:

- “CDF 関数” (334 ページ)
- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “PDF 関数” (712 ページ)

SEC 関数

正割を返します。

カテゴリ: 三角関数

返されるデータの種類: DOUBLE

構文

SEC(*expression*)

必須引数

expression
数値に評価される有効な式をラジアンで表して指定します。

制限事項 *expression* は、PI/2 の奇数倍にはできません。

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

比較

SEC 関数は COS 関数と関係があります。

sec(x)=1/cos(x)

例

次のステートメントは、SEC 関数を示しています。

ステートメント	結果
select sec(0.5);	1.139494
select sec(0);	1
select sec(3.14159/3);	1.999997

関連項目

関数:

- “COS 関数” (409 ページ)
- “COT 関数” (411 ページ)
- “CSC 関数” (422 ページ)
- “SIN 関数” (826 ページ)
- “TAN 関数” (849 ページ)

SECOND 関数

時間または日時値から秒を返します。

カテゴリ: 日付と時間

返されるデータの種類: DOUBLE

構文

SECOND(*time* | *datetime*)

オプション引数

time
時間値を表す有効な式を指定します。

データの種類 TIME

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

datetime
日時値を表す有効な式を指定します。

データの種類 TIMESTAMP

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

例

テーブル: CUSTONLINE (1259 ページ)

次のステートメントは、SECOND 関数を示しています。

ステートメント	結果
select second(endtime) from custonline;	1.253 9.421 55.746 9.398 45.221 15.766 56.288 33.955 27.908 20.475

関連項目

- “FedSQL の日付と時間” (59 ページ)

関数:

- “DAY 関数” (443 ページ)
- “HOUR 関数” (570 ページ)
- “MINUTE 関数” (670 ページ)
- “MONTH 関数” (676 ページ)
- “YEAR 関数” (887 ページ)

SIGN 関数

数値式の符号を示す数字を返します。

カテゴリ:	数学
返されるデータの種類:	TINYINT
注:	この関数は Teradata 14.0 以降ではサポートされていません。

構文

SIGN(*expression*)

必須引数

expression
数値に評価される有効な SQL 式を指定します。

データの種類 BIGINT、DOUBLE、FLOAT、INTEGER、REAL、SMALLINT、TINYINT

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

SIGN 関数は次の値を返します。

-1
expression < 0 の場合

0
expression = 0 の場合

1
expression > 0 の場合

例

次のステートメントは、SIGN 関数を示しています。

ステートメント	結果
select sign(-5);	-1
select sign(5);	1
select sign(0);	0

SIN 関数

三角関数の正弦を返します。

カテゴリ: 三角関数

返されるデータの種類: DOUBLE

構文

SIN(*expression*)

必須引数

expression
数値に評価される有効な SQL 式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

例

次のステートメントは、SIN 関数を示しています。

ステートメント	結果
select sin(25.6);	0.450441
select sin(5);	-0.95892

関連項目

関数:

- “ARCOS 関数” (302 ページ)
- “ARSIN 関数” (304 ページ)
- “COS 関数” (409 ページ)
- “SINH 関数” (827 ページ)

SINH 関数

双曲線正弦を返します。

カテゴリ: 三角関数

返されるデータの種類: DOUBLE

構文

SINH(*expression*)

必須引数

expression
数値に評価される有効な SQL 式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

SINH 関数は、次の式によって得られる引数の双曲線正弦を返します。

$$e^{argument} - e^{-argument}/2$$

例

次のステートメントは、SINH 関数を示しています。

ステートメント	結果
select sinh(0);	0
select sinh(1);	1.175201
select sinh(-1.0);	-1.1752

関連項目

関数:

- “COSH 関数” (410 ページ)

- “SIN 関数” (826 ページ)
- “TANH 関数” (850 ページ)

SKEWNESS 関数

式のすべての値の歪度を返します。

カテゴリ: 集計
記述統計

返されるデータの種類: DOUBLE

構文

SKEWNESS([DISTINCT] *expression*)

必須引数

expression
有効な SQL 式を指定します。

操作 計算を実行するには、列に少なくとも 3 つの有効な値が必要です。それがない場合は関数から null 値が返されます。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

オプション引数

DISTINCT
集計計算から重複行を削除します。

詳細

*歪度*は、平均からの偏差が一方の方向で他方の方向より大きくなる傾向の尺度です。歪度の正の値は、データが右に歪んでいることを示します。負の値は、データが左に歪んでいることを示します。

null 値と SAS 欠損値は無視され、計算には含まれません。

集計関数を使用して、FROM 句にリストされているテーブル全体、または GROUP BY 句で指定されている各グループのデータの統計的要約を生成できます。GROUP

BY 句は、指定された 1 つまたは複数の列でデータをグループ化します。GROUP BY 句を使用する場合、SELECT 句または HAVING 句の集計関数は、各グループのデータを要約する方法を FedSQL に指示します。FedSQL は、グループごとに個別に集計関数を計算します。GROUP BY を省略すると、テーブルまたはビュー内のすべての行が単一のグループと見なされます。

例

テーブル: WORLDTEMPS (1277 ページ)

次のステートメントは、SKEWNESS 関数を示しています。

ステートメント	結果
select skewness(AvgHigh) from worldtemps;	-0.69811

関連項目

関数:

- “STDDEV 関数” (837 ページ)

SELECT ステートメント句:

- “SELECT 句” (980 ページ)
- “GROUP BY 句” (993 ページ)
- “HAVING 句” (995 ページ)

SMALLEST 関数

k 番目に小さい非 null 値または非欠損値を返します。

カテゴリ:	記述統計
返されるデータの種類:	DOUBLE

構文

SMALLEST(*k*, *expression* [, ...*expression*])

必須引数

k
数値に評価されて返される有効な式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

expression
数値に評価されて処理される有効な式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

k が null または欠損値、0 未満、または値の数よりも大きい場合、結果は null または欠損値になります。

比較

SMALLEST 関数と ORDINAL 関数の違いは、SMALLEST は null 値と欠損値を無視しますが、ORDINAL は null 値と欠損値を数える点です。

例

次のステートメントは、SMALLEST 関数を示しています。

ステートメント	結果
select smallest(1, 456, 789, .Q, 123);	123
select smallest(2, 456, 789, .Q, 123);	456
select smallest(3, 456, 789, .Q, 123);	789
select smallest(4, 456, 789, .Q, 123);	.

関連項目

関数:

- “LARGEST 関数” (631 ページ)
- “ORDINAL 関数” (710 ページ)
- “PCTL 関数” (711 ページ)

SQUANTILE 関数

右側確率(SDF)の指定した場合に分布から分位点を返します。

カテゴリ: 分位点

返されるデータの種類: DOUBLE

参照項目: “SDF 関数” (819 ページ)

構文

SQUANTILE(*'distribution'*, *probability*, *parameter-1*, *...*, *parameter-k*)

必須引数

distribution
分布を特定する文字定数、変数または式です。

注: 各 SQUANTILE 分布関数の引数は、対応する CDF 分布関数の引数と同じです。

有効な分布は次のとおりです。

分布	引数
Bernoulli	'BERNOULLI'
ベータ	'BETA'
二項	'BINOMIAL'
Cauchy	'CAUCHY'
カイ 2 乗	'CHISQUARE'

分布	引数
Conway-Maxwell-Poisson	'CONMAXPOI'
指数	'EXPONENTIAL'
F	'F'
ガンマ	'GAMMA'
一般化 Poisson	'GENPOISSON'
幾何	'GEOMETRIC'
超幾何	'HYPERGEOMETRIC'
Laplace	'LAPLACE'
ロジスティック	'LOGISTIC'
対数正規	'LOGNORMAL'
負の二項	'NEGBINOMIAL'
正規	'NORMAL'|'GAUSS'
正規混合	'NORMALMIX'
パレート	'PARETO'
Poisson	'POISSON'
T	'T'
Tweedie	'TWEEDIE'
一様	'UNIFORM'
Wald (逆ガウス)	'WALD'|'IGAUSS'
Weibull	'WEIBULL'

注: T、F および NORMALMIX を除き、最初の 4 文字で分布を最小限に識別できます。

probability
確率変数の値を指定する、数値定数、変数または式です。

データの種類 DOUBLE

parameter-1, ..., parameter-k
特定の分布に対応する *shape*、*location* または *scale* パラメーターです(省略可能)。

データの種類 DOUBLE

詳細

SQUANTILE 関数は、指定された確率値に基づいて、指定された連続分布または離散分布から分位点を計算します。詳細については、前述の表に記載されている個々の分布を参照してください。

SQUANTILE 関数の Conway-Maxwell-Poisson 分布は、SDF 値が *p* 未満の最小の整数であるカウント値 *y* を返します。SQUANTILE 関数の Conway-Maxwell-Poisson 分布の構文は、次の形式です。

SQUANTILE('CONMAXPOI', *p*, *λ*, *ν*)

p
0 と 1 の間の実数(両端の値を含める)。

λ
Poisson 分布の場合と同様で、平均に似ています。

ν
ばらつきのパラメーターです。

詳細については、PDF 関数の“Conway-Maxwell-Poisson” (723 ページ)を参照してください。

例

次のステートメントは、SQUANTILE 関数を示しています。

ステートメント	結果
select squantile('logistic', 1.e-20);	46.0517
select squantile('conmaxpoi',.2,2.3,.4);	12

関連項目

関数:

- “CDF 関数” (334 ページ)

- “LOGCDF 関数” (643 ページ)
- “LOGPDF 関数” (647 ページ)
- “LOGSDF 関数” (649 ページ)
- “PDF 関数” (712 ページ)
- “SDF 関数” (819 ページ)
- “QUANTILE 関数” (793 ページ)

SQRT 関数

値の平方根を返します。

カテゴリ: 数学

返されるデータの種類: DOUBLE

構文

SQRT(*expression*)

必須引数

expression
負でない数値に評価される SQL の有効な式を指定します。

データの種類 DOUBLE

注 この関数の入力として負の数が渡された場合、エラーメッセージはログに書き込まれず、欠損値が出力として返されます。

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

例

次のステートメントは、SQRT 関数を示しています。

ステートメント	結果
select sqrt(36);	6

ステートメント	結果
select sqrt(25);	5
select sqrt(4.4);	2.097618

STD 関数

標準偏差を返します。

カテゴリ: 記述統計

返されるデータの種類: DOUBLE

構文

STD(*expression-1, expression-2* [, ...*expression-n*])

必須引数

expression
数値に評価される有効な式を指定します。

要件 少なくとも 2 つの非 null または非欠損の引数が必要です。それがない場合は、関数から null または欠損値が返されます。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ).

例

次のステートメントは、STD 関数を示しています。

ステートメント	結果
select std(2,6);	2.82842712474619
select std(2,6,.);	2.82842714274619

ステートメント	結果
select std(2,4,6,3,1);	1.92353840616714

STDDEV 関数

式のすべての値の統計標準偏差を返します。

カテゴリ: 集計
記述統計

別名: STD

返されるデータの種類: DOUBLE

構文

STDDEV([DISTINCT] *expression*)

必須引数

expression
有効な SQL 式を指定します。

操作 計算を実行するには、列に少なくとも 2 つの有効な値が必要です。それがない場合は関数から null 値が返されます。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

オプション引数

DISTINCT
(オプション)集計計算から重複行を削除します。

詳細

標準偏差は、分散の平方根として計算されます。

null 値と SAS 欠損値は無視され、計算には含まれません。

集計関数を使用して、FROM 句にリストされているテーブル全体、または GROUP BY 句で指定されている各グループのデータの統計的要約を生成できます。GROUP BY 句は、指定された 1 つまたは複数の列でデータをグループ化します。GROUP BY 句を使用する場合、SELECT 句または HAVING 句の集計関数は、各グループのデータを要約する方法を FedSQL に指示します。FedSQL は、グループごとに個別に集計関数を計算します。GROUP BY を省略すると、テーブルまたはビュー内のすべての行が単一のグループと見なされます。

例

テーブル: WORLDTEMPS (1277 ページ)

次のステートメントは、STDDEV 関数を示しています。

ステートメント	結果
select stddev(AvgHigh) from worldtemps;	7.811414

関連項目

関数:

- “STDERR 関数” (838 ページ)
- “VARIANCE 関数” (877 ページ)

SELECT ステートメント句:

- “SELECT 句” (980 ページ)
- “GROUP BY 句” (993 ページ)
- “HAVING 句” (995 ページ)

STDERR 関数

式のすべての値の統計標準誤差を返します。

カテゴリ:	集計 記述統計
返されるデータの種類:	DOUBLE

構文

STDERR([DISTINCT] *expression*)

必須引数

expression
有効な SQL 式を指定します。

操作	計算を実行するには、列に少なくとも 2 つの有効な値が必要です。それがない場合は関数から null 値が返されます。
データの種類	DOUBLE

オプション引数

DISTINCT
集計計算から重複行を削除します。

詳細

null 値と SAS 欠損値は無視され、計算には含まれません。

集計関数を使用して、FROM 句にリストされているテーブル全体、または GROUP BY 句で指定されている各グループのデータの統計的要約を生成できます。GROUP BY 句は、指定された 1 つまたは複数の列でデータをグループ化します。GROUP BY 句を使用する場合、SELECT 句または HAVING 句の集計関数は、各グループのデータを要約する方法を FedSQL に指示します。FedSQL は、グループごとに個別に集計関数を計算します。GROUP BY を省略すると、テーブルまたはビュー内のすべての行が単一のグループと見なされます。

例

テーブル: WORLDTEMPS (1277 ページ)

次のステートメントは、STDERR 関数を示しています。

ステートメント	結果
select stderr(AvgHigh) from worldtemps;	2.35523

関連項目

関数:

- “STDDEV 関数” (837 ページ)

SELECT ステートメント句:

- “SELECT 句” (980 ページ)
- “GROUP BY 句” (993 ページ)
- “HAVING 句” (995 ページ)

STUDENTS_T 関数

式の値のスチューデントの t 分布を返します。

カテゴリ: 集計
記述統計

別名: T

返されるデータの種類: DOUBLE

構文

STUDENTS_T([DISTINCT] *expression*)

必須引数

expression
有効な SQL 式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)
“FedSQL 式” (46 ページ)

オプション引数

DISTINCT
集計計算から重複行を削除します。

詳細

STUDENTS_T 関数は、自由度が *n-1* で中心分布(nc=0)のスチューデントの t 分布の確率を返します。

集計関数を使用して、FROM 句にリストされているテーブル全体、または GROUP BY 句で指定されている各グループのデータの統計的要約を生成できます。GROUP BY 句は、指定された 1 つまたは複数の列でデータをグループ化します。GROUP BY 句を使用する場合、SELECT 句または HAVING 句の集計関数は、各グループのデータを要約する方法を FedSQL に指示します。FedSQL は、グループごとに個別に集計関数を計算します。GROUP BY を省略すると、テーブルまたはビュー内のすべての行が単一のグループと見なされます。

比較

STUDENTS_T 関数は、スチューデントの t 分布を返します。PROBT 関数は、スチューデントの t 分布が特定の値以下である確率を返します。

例

テーブル: DENSITIES (1260 ページ)

次のステートメントは、STUDENTS_T 関数を示しています。

ステートメント	結果
select students_t(density) from densities;	3.565402
select students_t(population) from densities;	3.267438

関連項目

関数:

- “PROBT 関数” (787 ページ)

SELECT ステートメント句:

- “SELECT 句” (980 ページ)
- “GROUP BY 句” (993 ページ)
- “HAVING 句” (995 ページ)

SUBSTR 関数

引数から部分文字列を抽出します。

カテゴリ: 文字

返されるデータの種類: CHAR、NCHAR、NVARCHAR、VARCHAR

構文

SUBSTR(*character-expression*, *position-expression* [, *length-expression*])

必須引数

character-expression
文字定数、変数または式を指定します。

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

参照項目: “FedSQL 式” (46 ページ)

position-expression
開始文字位置を示す数値定数、変数または式を指定します。

要件 *position-expression* は、ゼロ以上である必要があります。

データの種類 BIGINT

参照項目: “FedSQL 式” (46 ページ)

オプション引数

length-expression
抽出する部分文字列の長さを示す数値定数、変数または式を指定します。

操作 *length-expression* を省略した場合、SUBSTR 関数は式の開始文字位置以降を抽出します。*length-expression* がゼロ、負の値または string 内で position より後に残った式の長さよりも長い場合、この関数は式の残りを抽出します。この関数は、また、引数が無効であることを示す WARNING もログに出力します。

character-expression または *position-expression* が空白または null の場合、関数は空白値を返し、SAS ログには何も出力しません。

データの種類 BIGINT

参照項目: “FedSQL 式” (46 ページ)

詳細

SUBSTR 関数は、式内の *character-expression* で指定された部分を抽出します。この部分は、*position-expression* で指定した文字位置から始まり、*length-expression* で指定した文字数になります。

比較

DATA ステップと同様の部分文字列抽出動作が必要な場合は、SUBSTR 関数を使用します。入力式が無効な場合、DATA ステップは_ERROR_を 1 に設定し、ログに注を出力します。FedSQL はエラーを設定したり、注を返したりしません。ANSI SQL 部分文字列抽出動作の FedSQL 実装が必要な場合は、SUBSTRING 関数を使用します。

例

テーブル: “AFewWords” (1256 ページ)

次のステートメントは、SUBSTR 関数の使用法を示しています。

ステートメント	結果
select substr(word2, 2, 3) from afewwords;	HER HIN ODY
select substr('chsh234960b3', 5);	234960b3
select substr('chsh234960b3', 5, 6);	234960
select substr('chsh234960b3', 5, 15);	234960b3
select substr('chsh234960b3', 5, -5);	234960b3
select substr('chsh234960b3', 2, 4);	hsh2
select substr('kidnap', .);	blank string
select substr(' ', 1);	blank string

関連項目

関数:

- “SUBSTRING 関数” (844 ページ)

SUBSTRING 関数

文字列から部分文字列を抽出します。

カテゴリ: 文字

返されるデータの種類: VARCHAR NVARCHAR

構文

SUBSTRING(*character-expression* FROM *position* [FOR *length*])

必須引数

character-expression
文字列に評価される任意の有効な式を指定します。*character-string* は、部分文字列を検索するソース文字列です。

データの種類 CHAR、VARCHAR、NVARCHAR

position
抽出された部分文字列の開始文字位置を表す式を指定します。

データの種類 INTEGER

オプション引数

length
抽出する部分文字列の長さを表す式を指定します。

データの種類 INTEGER

詳細

SUBSTRING 関数は、式内の *character-expression* で指定された部分を返します。この部分は、*position* で指定した文字位置から始まり、*length* で指定した文字数になります。

- 次の条件のいずれかに該当する場合、FedSQL は null 値を返します。
  - *position* がソース文字列の長さを超えている場合。たとえば、5 文字目で始まる部分文字列を要求したが、ソース文字列の長さが 4 文字しかない場合、null の結果が得られます。
  - *character-expression* が欠損値。
  - *position* が欠損値。
- *length* が負の値の場合、FedSQL はエラーを返します。
- FedSQL は、次の条件下で文字式の残りを抽出します。
  - *length* が、position の後の文字列内の式の長さよりも大きい。
  - *length* が省略されている。

比較

SUBSTRING は、ANSI SQL 部分文字列抽出動作の FedSQL 実装です。DATA ステップの構文抽出動作をエミュレートする場合は、SUBSTR を使用します。

例

テーブル: “AFewWords” (1256 ページ)

次のステートメントは、SUBSTRING 関数を示しています。

ステートメント	結果
select substring(word2 from 2 for 3) from afewwords; “AFewWords” (1256 ページ)を参照してください。	HER HIN ODY
select substring('chsh234960b3' from 5);	234960b3
select substring('chsh234960b3' from 5 for 6);	234960
select substring('chsh234960b3' from 5 for 15);	234960b3
select substring('chsh234960b3' from 5 for -5);	ERROR: A substring error occurred
select substring('kidnap' from .);	Blank string
select substring(' ' from 1);	Blank string

関連項目

関数:

- “SUBSTR 関数” (842 ページ)
- “TRIM 関数” (866 ページ)

SUM 関数

式のすべての値の合計を返します。

カテゴリ: 集計
記述統計

返されるデータの種類: 式と同じデータ型

構文

SUM([DISTINCT] *expression*)

必須引数

expression
有効な SQL 式を指定します。

要件 SUM 関数の結果は、データ型の範囲内にある必要があります。

データの種類 BIGINT、DOUBLE、FLOAT、INTEGER、REAL、SMALLINT、TINYINT

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

オプション引数

DISTINCT
集計計算から重複行を削除します。

詳細

null 値と SAS 欠損値は無視され、計算には含まれません。

集計関数を使用して、FROM 句にリストされているテーブル全体、または GROUP BY 句で指定されている各グループのデータの統計的要約を生成できます。GROUP

BY 句は、指定された 1 つまたは複数の列でデータをグループ化します。GROUP BY 句を使用する場合、SELECT 句または HAVING 句の集計関数は、各グループのデータを要約する方法を FedSQL に指示します。FedSQL は、グループごとに個別に集計関数を計算します。GROUP BY を省略すると、テーブルまたはビュー内のすべての行が単一のグループと見なされます。

例

テーブル: DENSITIES (1260 ページ)

次のステートメントは、SUM 関数を示しています。

ステートメント	結果
select sum(density) from densities;	1728.324
select sum(population) from densities;	1.2278E8

関連項目

SELECT ステートメント句:

- “SELECT 句” (980 ページ)
- “GROUP BY 句” (993 ページ)
- “HAVING 句” (995 ページ)

SUMABS 関数

非欠損引数の絶対値の合計を返します。

カテゴリ: 記述統計

返されるデータの種類: DOUBLE

構文

SUMABS(*value-1*[, ...*value-n*])

必須引数

value
数値に評価される有効な式を指定します。

データの種類 DOUBLE

詳細

すべての引数が null または欠損値の場合、結果は null または欠損値になります。それ以外の場合は、結果は非欠損値の絶対値の合計です。

例

次のステートメントは、SUMABS 関数を示しています。

ステートメント	結果
select sumabs(1,.,-2,0,3,.q,-4);	10
select sumabs(1,3,4,3,1);	12

SYSGET 関数

指定された動作環境変数の値を返します。

カテゴリ: 特殊

制限事項: この関数は、管理レベルの機能を持つユーザーに対してのみ CAS でサポートされます。

構文

SYSGET('*environment-variable*')

必須引数

environment-variable
Linux での環境変数名を値とする文字定数、変数または式です。

要件 この引数は、単一引用符で囲む必要があります。

詳細

概要情報

SYSGET 関数は、文字列として環境変数の値を返します。たとえば、次のステートメントは、Linux の HOME 環境変数の値を返します。

```
select sysget('HOME');
```

まだ長さが割り当てられていない変数に SYSGET 関数から値が返される場合、その変数にはデフォルトで長さ 200 が割り当てられます。

注: *environment-variable* 引数に入力する値の大文字と小文字の区別は、Linux 動作環境に保存されている変数の大文字と小文字の区別に一致している必要があります。

TAN 関数

正接を返します。

カテゴリ: 三角関数

返されるデータの種類: DOUBLE

構文

TAN(*expression*)

必須引数

expression
数値に評価される有効な SQL 式を指定します。

制限事項 *expression* は、π/2 の奇数倍にはできません。

要件 式は、ラジアン単位の値に評価される必要があります。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

例

次のステートメントは、TAN 関数を示しています。

ステートメント	結果
select tan(0.5);	0.546302
select tan(3.14159/3);	1.732047

関連項目

関数:

- “ATAN 関数” (308 ページ)
- “ATAN2 関数” (309 ページ)
- “COS 関数” (409 ページ)
- “COT 関数” (411 ページ)
- “SIN 関数” (826 ページ)
- “TANH 関数” (850 ページ)

TANH 関数

双曲線正接を返します。

カテゴリ: 三角関数

返されるデータの種類: DOUBLE

構文

TANH(*expression*)

必須引数

expression
数値に評価される有効な SQL 式を指定します。

制限事項 *expression* は、π/2 の奇数倍にはできません。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

TANH 関数は、次の式によって得られる引数の双曲線正接を返します。

$$\frac{e^{argument} - e^{-argument}}{e^{argument} + e^{-argument}}$$

例

次のステートメントは、TANH 関数の例です。

ステートメント	結果
select tanh(0);	0
select tanh(0.5);	0.462117
select tanh(-0.5);	-0.46212

関連項目

関数:

- “ATAN 関数” (308 ページ)
- “ATAN2 関数” (309 ページ)
- “COSH 関数” (410 ページ)
- “SINH 関数” (827 ページ)
- “TAN 関数” (849 ページ)

TIMEPART 関数

時間を時、分、秒で返します。

カテゴリ: 日付と時間

返されるデータの種類: TIME

構文

TIMEPART(*ts*)

必須引数

ts
タイムスタンプを指定します。

例

次のステートメントは、TIMEPART 関数を示しています。

ステートメント	結果
select timepart(timestamp '2013-10-30 15:39:10');	15:39:10

関連項目

関数:

- “DATEPART 関数” (441 ページ)
- “MAKEDATE 関数” (654 ページ)
- “MAKETIME 関数” (655 ページ)
- “MAKETIMESTAMP 関数” (656 ページ)

TIMEVALUE 関数

変動金利を使用して、基準日の参照額に相当する額を返します。

カテゴリ:	財務
返されるデータの種類:	DOUBLE

構文

TIMEVALUE(*base-date*, *reference-date*, *reference-amount*, *compounding-interval*, *date-1*, *rate-1*[, ...*date-n*, *rate-n*])

必須引数

base-date
base-date における *reference-amount* の時間値を指定します。

要件 *base-date* は SAS 日付です。

データの種類 DOUBLE

reference-date
reference-amount の日付を指定します。

要件 *reference-date* は SAS 日付です。

データの種類 DOUBLE

reference-amount
reference-date における金額を指定します。

データの種類 DOUBLE

compounding-interval
複利間隔を指定します。

要件 *compounding-interval* は SAS 間隔です。

データの種類 CHAR

date
rate が有効になるタイミングを指定します。各日付は利率とペアになります。

要件 *date* は SAS 日付です。

データの種類 DOUBLE

rate
date に開始する利率を数値のパーセンテージで指定します。各利率は日付とペアになります。

データの種類 DOUBLE

詳細

TIMEVALUE 関数には次の詳細が適用されます。

- 利率の値は–99 から 120 までの値にする必要があります。
- 日付-利率ペアのリストは、日付順に並べ替える必要はありません。
- 1 日に複数の利率が変更される場合、TIMEVALUE 関数は、その日付でリストされている最後の利率のみを適用します。
- 一部の期間には単利が適用されます。

- *reference-date* と *base-date* の両方の日付またはその前の日付となる有効な日付-利率ペアが必要になります。

例

- 名目金利が 10%で 1 年間毎月複利計算される$1,000 の投資の累計額は、次のように表すことができます。

  ```
  select timevalue(date'2001-01-01', date'2000-01-01', 1000, 'month',
  date'2000-01-01', 10);
  ```

  ```
  1104.71306744129
  ```

- 半年後に利率が 20%に上昇した場合の計算は次のとおりです。

  ```
  select timevalue(date'2001-01-01', date'2000-01-01', 1000, 'month',
  date'2000-01-01', 10, date'2000-07-01', 20);
  ```

  ```
  1160.63657783831
  ```

- 日付-利率ペアは、日付順に並べ替える必要はありません。この柔軟性があるため、amount_base2 と amount_base3 は同じ値になります。

  ```
  select timevalue(date'2001-01-01', date'2000-01-01', 1000, 'month',
  date'2000-07-01', 20, date'2000-01-01', 10);
  ```

  ```
  1160.63657783831
  ```

TINV 関数

t 分布から分位点を返します。

カテゴリ:	分位点
返されるデータの種類:	DOUBLE

構文

TINV(*p*, *df*[, *nc*])

必須引数

p
数値の確率に評価される有効な式を指定します。

範囲	0 < *p* < 1
データの種類	DOUBLE
参照項目:	“<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

df
数値の自由度パラメーターに評価される有効な式を指定します。

範囲 *df* > 0

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

オプション引数

nc
数値の非心度パラメーターに評価される有効な式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

TINV 関数は、自由度が *df*、非心度パラメーターが *nc* のスチューデントの *t* 分布から *p*番目の分位点を返します。*t* 分布のオブザベーションが、返される分位点以下となる確率は *p* です。

TINV では、整数以外の自由度パラメーター *df* を指定できます。このパラメーター *nc* (省略可能)が指定されていないか 0 の場合、心度 *t* 分布から分位点が返されます。

注意
nc の値が大きいとアルゴリズムが失敗する場合があります。 その場合、欠損値が返されます。

比較

TINV は PROBT 関数の逆数です。

例

次のステートメントは、TINV 関数を示しています。

ステートメント	結果
select tinv(.95, 2);	2.9199855804
select tinv(.95, 2.5, 3);	11.033833625

関連項目

関数:

- “PROBT 関数” (787 ページ)

TNONCT 関数

スチューデントの *t* 分布から非心度パラメーターの値を返します。

カテゴリ: 数学

返されるデータの種類: DOUBLE

構文

TNONCT(*x*, *df*, *prob*)

必須引数

x
数値の確率変数です。

データの種類 DOUBLE

df
数値の自由度パラメーターです。

範囲 $df > 0$

データの種類 DOUBLE

prob
確率です。

範囲 $0 < prob < 1$

データの種類 DOUBLE

詳細

TNONCT 関数は、パラメーターが *x*、*df* および *nc* の非心度 *t* 分布から、負でない非心度パラメーターを返します。式の平方根 *nc* を求めるためにニュートン型のアルゴリズムが使用されます。

$$P_t(x|df, nc) - prob = 0$$

前述の式には次の関係が適用されます。

$$P_t(x|df, nc) = \frac{1}{\Gamma\left(\frac{df}{2}\right)} \int_0^{\infty} v^{\frac{df}{2}-1} \varepsilon^{-v} \int_{-\infty}^{x\sqrt{\frac{2v}{df}}} \varepsilon^{-\frac{(u-nc)^2}{2}} du dv$$

アルゴリズムで固定小数点を収束できない場合、欠損値が返されます。

例

次の例では、*t* 分布から非心度パラメーターを計算します。

```
proc ds2;
data test /overwrite=yes;
  dcl double x df nc prob;
  method init();
    x=2;
    df=4;
    do nc=1 to 3 by .5;
      prob=probt(x, df, nc);
      output;
    end;
  end;
enddata;
run;
quit;

proc print data=test;
run;

proc fedsql;
 select tnonct(x, df, prob) from test;
quit;
```

アウトプット 4.13　テーブル Test のコンテンツ

Obs	x	df	nc	prob
1	2	4	1.0	0.76457
2	2	4	1.5	0.61893
3	2	4	2.0	0.45567
4	2	4	2.5	0.30115
5	2	4	3.0	0.17702

アウトプット 4.14　テーブル Test での TNONCT 関数の結果

TNONCT
1
1.5
2
2.5
3

関連項目

関数:

- “CNONCT 関数” (384 ページ)
- “FNONCT 関数” (546 ページ)

TODAY 関数

数値の SAS 日付値として現在の日付を返します。

カテゴリ:	日付と時間
返されるデータの種類:	DOUBLE

構文

TODAY( )

詳細

TODAY 関数は引数を取りません。SAS 日付値(1960 年 1 月 1 日からの日数)の形式で現在の日付を生成します。

FedSQL が日付を処理する方法の詳細については、“FedSQL の日付と時間” (59 ページ)を参照してください。

例

次のステートメントは、TODAY 関数を示しています。

ステートメント	結果
select today();	19612
select put(td,date.);	11SEP13

TRANSLATE 関数

文字式内の特定の文字を置き換えます。

カテゴリ: 文字

返されるデータの種類: CHAR、NCHAR、NVARCHAR、VARCHAR

構文

TRANSLATE(*expression*, *to-characters*, *from-characters*)

必須引数

expression
文字列に評価されるか強制変換できる有効な式を指定します。*expression* には元の文字値が含まれます。

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

参照項目: “FedSQL 式” (46 ページ)

to-characters
TRANSLATE で代替文字として使用する文字を指定します。

データの種類　CHAR、NCHAR、NVARCHAR、VARCHAR

from-characters
TRANSLATE で置換する文字を指定します。

データの種類　CHAR、NCHAR、NVARCHAR、VARCHAR

詳細

to-characters と *from-characters* の値は、個々の文字ごとに対応しています。たとえば、TRANSLATE では、*from-characters* の最初の文字が *to-characters* の最初の文字に変更されます。*to-characters* の文字数が *from-characters* の文字数より少ない場合、TRANSLATE は *from-characters* の余った文字を空白に変更します。*to-characters* の文字数が *from-characters* の文字数より多い場合、TRANSLATE は *to-characters* の余った文字を無視します。

比較

TRANSTRN 関数は、置換文字列にゼロの長さを使用できるという点で TRANWRD 関数と異なります。TRANWRD では、置換文字列の長さがゼロ場合はかわりに 1 つの空白を使用します。

TRANSLATE 関数は、ユーザーが指定したすべての文字を別の文字に変換します。TRANSLATE は 1 回の呼び出しで複数の文字をスキャンできます。ただし、このスキャンを実行すると、TRANSLATE は文字列に含まれる個々のすべての文字を検索します。つまり、ターゲット文字列内の文字がソース文字列内に見つかった場合、その文字は対応する置換文字列内の文字に置き換えられます。

TRANWRD 関数は、単語(または文字のパターン)をスキャンし、それらの単語を 2 番目の単語(または文字のパターン)で置き換える点で TRANSLATE とは異なります。

例

次のステートメントは、TRANSLATE 関数を使用してテキストを修正する方法を示しています。

注: "translate"いう言葉は、FedSQL の予約語です。関数名を区切る必要があります。

ステートメント	結果
select "translate" ('AABBAABABB', '12', 'AB');	1122112122
select "translate" ('XYZW', 'AB', 'VW');	XYZB

関連項目

関数:

- “TRANSTRN 関数” (861 ページ)
- “TRANWRD 関数” (863 ページ)

TRANSTRN 関数

文字列に含まれる特定の部分文字列をすべて置き換えるか取り除きます。

カテゴリ: 文字

返されるデータの種類: CHAR、NCHAR、NVARCHAR、VARCHAR

参照項目: “FedSQL での文字型を返す関数の使用” (260 ページ)

構文

TRANSTRN(*source-expression*, *target-expression*, *replacement-expression*)

必須引数

source-expression
文字列に評価されるか強制変換できて、変換する文字を含む有効な式を指定します。

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

target-expression
文字列に評価されるか強制変換できて、その文字が *source-expression* で検索される有効な式を指定します。

要件 *target-expression* の長さにはゼロより大きい値を指定する必要があります。

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

replacement-expression
文字列に評価されるか強制変換できて、*target-expression* を置換する有効な式を指定します。

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

TRANSTRN 関数は、文字列に含まれる特定の部分文字列をすべて置き換えるか取り除きます。TRANSTRN 関数では、*target-expression* 文字列と *replacement-expression* 文字列の末尾の空白は取り除きません。出現する *target* をすべて取り除くには、*replacement-expression* に TRIMN(' ')を指定します。

比較

TRANSTRN 関数は、置換文字列にゼロの長さを使用できるという点で TRANWRD 関数と異なります。TRANWRD では、置換文字列の長さがゼロ場合はかわりに 1 つの空白を使用します。

TRANSLATE 関数は、ユーザーが指定したすべての文字を別の文字に変換します。TRANSLATE は 1 回の呼び出しで複数の文字をスキャンできます。ただし、このスキャンを実行すると、TRANSLATE は文字列に含まれる個々のすべての文字を検索します。つまり、ターゲット文字列内の文字がソース文字列内に見つかった場合、その文字は対応する置換文字列内の文字に置き換えられます。

TRANSTRN 関数は、部分文字列をスキャンし、それらの部分文字列を 2 番目の部分文字列で置き換える点で TRANSLATE と異なります。

例

次のステートメントは、TRANSTRN 関数を使用して出現する単語を置換し、スペースを操作する方法を示しています。

ステートメント	結果
select transtrn('Mrs. Joan Smith', 'Mrs.', 'Ms.');	Ms. Joan Smith
select transtrn('Miss Alice Cooper', 'Miss', 'Ms.');	Ms. Alice Cooper
select transtrn('CATFISH', 'FISH', 'NIP');	CATNIP
select '*' || transtrn('abcxabc', 'abc',	*x*

ステートメント	結果
trimn(' ')) || '*';	
select '*' || transtrn('abcxabc', 'abc', ' ') || '*';	* x *

関連項目

関数:

- “TRANSLATE 関数” (*SAS 関数と CALL ルーチン: リファレンス*)
- “TRANWRD 関数” (863 ページ)

TRANWRD 関数

文字列に含まれる特定の部分文字列をすべて置き換えます。

カテゴリ: 文字

返されるデータの種類: CHAR、NCHAR、NVARCHAR、VARCHAR

参照項目: “FedSQL での文字型を返す関数の使用” (260 ページ)

構文

TRANWRD(*source-expression*, *target-expression*, *replacement-expression*)

必須引数

source-expression
置き換える文字を含む文字列に評価されるか強制変換できる有効な式を指定します。

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

target-expression
文字列に評価されるか強制変換できて、*source-expression* で検索される有効な式を指定します。

要件 *target-expression* の長さにはゼロより大きい値を指定する必要があります。

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

replacement-expression
文字列に評価されるか強制変換できて、*target-expression* を置換する有効な式を指定します。

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

TRANWRD 関数は、文字列に含まれる特定の部分文字列(または文字のパターン)の出現箇所をすべて置き換えます。TRANWRD 関数では、*target-expression* 文字列と *replacement-expression* 文字列の末尾の空白は取り除きません。

比較

TRANSTRN 関数は、置換文字列にゼロの長さを使用できるという点で TRANWRD 関数と異なります。TRANWRD では、置換文字列の長さがゼロ場合はかわりに 1 つの空白を使用します。

TRANSLATE 関数は、ユーザーが指定したすべての文字を別の文字に変換します。TRANSLATE は 1 回の呼び出しで複数の文字をスキャンできます。ただし、これを実行すると、TRANSLATE は文字列に含まれる個々のすべての文字を検索します。つまり、ターゲット文字列内の文字がソース文字列内に見つかった場合、その文字は対応する置換文字列内の文字に置き換えられます。

TRANWRD 関数は、部分文字列(または文字のパターン)をスキャンし、それらの部分文字列を 2 番目の部分文字列(または文字のパターン)で置き換える点で TRANSLATE とは異なります。

例

次のステートメントは、TRANWRD 関数を使用して、出現する部分文字列を置換する方法を示しています。

ステートメント	結果
select tranwrd('Whatever you so, so it with all your might. -P.T. Barnum','so','do');	Whatever you do, do it with all your might. -P.T. Barnum
select tranwrd('Believe and act as if it were impossible to pail. -Charles F. Kettering', 'pail', 'fail');	Believe and act as if it were impossible to fail. -Charles F. Kettering

関連項目

関数:

- “TRANSTRN 関数” (861 ページ)

TRIGAMMA 関数

トリガンマ関数の値を返します。

カテゴリ: 数学

返されるデータの種類: DOUBLE

構文

TRIGAMMA(*expression*)

必須引数

expression
数値に評価される有効な式を指定します。

制限事項 正でない整数は無効です。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

TRIGAMMA 関数は、ディガンマ関数の導関数を返します。*expression* > 0 の場合、TRIGAMMA 関数は lgamma 関数の第 2 導関数です。

例

次のステートメントは、TRIGAMMA 関数を示しています。

ステートメント	結果
select trigamma(3);	0.394934

関連項目

関数:

- “DIGAMMA 関数” (453 ページ)
- “LGAMMA 関数” (637 ページ)

TRIM 関数

文字列から先頭の文字、末尾の文字、またはその両方を削除します。

カテゴリ: 文字

返されるデータの種類: VARCHAR NVARCHAR

構文

TRIM([BOTH | LEADING | TRAILING] [*trim-character*] FROM *column*)

必須引数

column
列名に評価される有効な式です。

オプション引数

BOTH | LEADING | TRAILING
先頭の文字、末尾の文字、またはその両方を削除するかどうかを指定します。

デフォルト BOTH

'*trim-character*'
column から削除する 1 文字を指定します。リテラル文字は単一引用符で囲みます。*trim-character* が指定されていない場合、TRIM 関数は 1 文字だけでなく、すべての空白スペースを削除します。

デフォルト 空白

データの種類 CHAR、VARCHAR、NVARCHAR

詳細

TRIM 関数は、空白などの文字列を連結前に削除する場合に便利です。

例

テーブル: “AFewWords” (1256 ページ)

次の表のステートメントでは、TRIM 関数を説明します。

ステートメント	結果
select trim(word1) from afewwords;	*some/ *every* *no*
select trim(both '*' from word1) from afewwords;	some every no
select trim(leading '*' from word1) from afewwords;	some/ every* no*
select trim(trailing '*' from word1) from afewwords;	*some/ *every *no

関連項目

関数:

- “SUBSTRING 関数” (844 ページ)

TRUNC 関数

数値を指定した長さに切り捨てます。

カテゴリ: 切り捨て

返されるデータの種類: DOUBLE

構文

TRUNC(*expression*, *length-expression*)

必須引数

expression
数値に評価される有効な式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

length-expression
数値に評価される有効な式を指定します。

範囲 3-8

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

TRUNC 関数では、完全な長さの数値式(DOUBLE として保存)を *length-expression* で指定したより小さいバイト数に切り捨て、切り捨てられたバイトを 0 で埋め込みます。切り捨てとその後の展開は、最初に完全な長さより短い数を保存した結果を複製し、次にそれを読み込みます。

例

次のステートメントは、TRUNC 関数を示しています。

ステートメント	結果
select trunc(3.1,3);	3.099609375
select trunc(3.1,4);	3.0999984741
select trunc(3.1,5);	3.099999994
select trunc(3.1,6);	3.1
select trunc(3.1,7);	3.1
select trunc(3.1,8);	3.1

UPCASE 関数

引数内のすべての文字を大文字に変換します。

カテゴリ: 文字

別名: UPPER

返されるデータの種類: CHAR、NCHAR、NVARCHAR、VARCHAR

構文

UPCASE(*expression*)

必須引数

expression
文字列に評価されるか強制変換できる有効な式を指定します。

データの種類 CHAR、NCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

UPCASE 関数は、文字式をコピーし、すべての小文字を大文字に変換して、変更された値を結果として返します。

比較

LOWCASE 関数は、引数内のすべての文字を小文字に変換します。UPCASE 関数は、引数内のすべての文字を大文字に変換します。

例

テーブル: AFEWWORDS (1256 ページ)

次のステートメントは、UPCASE (または UPPER)関数を示しています。

ステートメント	結果
select upcase('John B. Smith');	JOHN B. SMITH
select upper(word1) from afewwords;	*SOME/ *EVERY* *NO*

関連項目

関数:

- “LOWCASE 関数” (652 ページ)

URLDECODE 関数

URL のエスケープ構文を使用してデコードされた文字列を返します。

カテゴリ: Web ツール

参照項目: “FedSQL での文字型を返す関数の使用” (260 ページ)

構文

URLDECODE(*expression*)

必須引数

expression
文字列に評価されるか強制変換できる有効な式を指定します。

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

詳細

基本

URL のエスケープ構文は、URL に使用されると意味を持つ可能性のある文字を非表示にするために使用されます。

URL のエスケープシーケンスは、次のいずれです。

- プラス記号。空白で置き換えられます。
- パーセント記号で始まり、その後に 2 つの 16 進文字が続く 3 つの文字のシーケンス。このパターンは、指定された 16 進数値を持つ単一の文字に置き換えられます。

expression は、SAS セッションエンコーディングまたは UTF-8 エンコーディングのいずれかを使用してデコードできます。

文字確認

URLDECODE および URLENCODE 関数では、エスケープシーケンスによって 生成されたバイトがエンコーディングに基づく有効な文字かどうかは確認されません。

返される変数の長さ

まだ長さが割り当てられていない変数に URLDECODE 関数から値が返される場合、その変数には引数の長さが割り当てられます。

例

次のステートメントは、URLDECODE 関数を示しています。

ステートメント	結果
select urldecode('abc+def');	abc def
select urldecode('why%3F');	why?
select urldecode('%41%42%43%23%31');	ABC#1

関連項目

関数:

- “URLENCODE 関数” (872 ページ)

URLENCODE 関数

URL のエスケープ構文を使用してエンコードされた文字列を返します。

カテゴリ: Web ツール

参照項目: “FedSQL での文字型を返す関数の使用” (260 ページ)

構文

URLENCODE(*expression*)

必須引数

expression
文字列に評価されるか強制変換できる有効な式を指定します。

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

詳細

基本

expression は、SAS セッションエンコーディングまたは UTF-8 エンコーディングのいずれかを使用してエンコードできます。

URLENCODE 関数は、URL に使用されると意味を持つ可能性のある文字をエンコードします。この関数は、次を除くすべての文字をエンコードします。

- すべての英数字
- ドル記号($)
- ハイフン(-)
- アンダースコア(_)
- アットマーク(@)
- ピリオド(.)
- 感嘆符(!)
- アスタリスク(*)
- 開始かっこ(( )と閉じかっこ( ))
- カンマ(,)

注: エンコードされた文字列は、元の文字列よりも長くなる可能性があります。この関数を使用する場合は、長さを追加することを検討する必要があります。

文字確認

URLDECODE および URLENCODE 関数では、エスケープシーケンスによって 生成されたバイトがエンコーディングに基づく有効な文字かどうかは確認されません。

例

次のステートメントは、URLENCODE 関数を示しています。

ステートメント	結果
select urlencode('abc def');	abc%20def

関連項目

関数:

- “URLDECODE 関数” (870 ページ)

USS 関数

式のすべての値の無修正平方和を返します。

カテゴリ: 集計
記述統計

返されるデータの種類: DOUBLE

構文

USS([DISTINCT] *expression*)

必須引数

expression
有効な SQL 式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

オプション引数

DISTINCT
集計計算から重複行を削除します。

詳細

null 値と SAS 欠損値は無視され、計算には含まれません。

集計関数を使用して、FROM 句にリストされているテーブル全体、または GROUP BY 句で指定されている各グループのデータの統計的要約を生成できます。GROUP BY 句は、指定された 1 つまたは複数の列でデータをグループ化します。GROUP BY 句を使用する場合、SELECT 句または HAVING 句の集計関数は、各グループのデータを要約する方法を FedSQL に指示します。FedSQL は、グループごとに個別に集計関数を計算します。GROUP BY を省略すると、テーブルまたはビュー内のすべての行が単一のグループと見なされます。

比較

CSS 関数は、すべての値の修正済み平方和を返します。USS 関数は、無修正平方和を返します。

例

テーブル: DENSITIES (1260 ページ)

次のステートメントは、USS 関数を示しています。

ステートメント	結果
select uss(density) from densities;	510190
select uss(population) from densities;	2.82E15

関連項目

関数:

- “CSS 関数” (423 ページ)

SELECT ステートメント句:

- “SELECT 句” (980 ページ)
- “GROUP BY 句” (993 ページ)
- “HAVING 句” (995 ページ)

UUIDGEN 関数

ユニバーサル一意識別子(UUID)の短い形式を返します。

カテゴリ:	特殊
返されるデータの種類:	CHAR、NCHAR、NVARCHAR、VARCHAR
参照項目:	“FedSQL での文字型を返す関数の使用” (260 ページ)

構文

UUIDGEN( )

引数なし

UUIDGEN 関数には引数がありません。

詳細

UUIDGEN 関数は、各呼び出しに対して UUID (固有値)を返します。デフォルトの結果は、次のように 36 文字になります。

5ab6fa40–426b-4375–bb22–2d0291f43319

例

次のステートメントは、UUIDGEN 関数を示しています。

ステートメント	結果
select uuidgen();	8F096323-F32A-9549-AAE7-7C8B9D520194

関連項目

その他のリファレンス:

- “ユニバーサル一意識別子” (*SAS プログラマガイド: エッセンシャル*)

VAR 関数

分散を返します。

カテゴリ:	記述統計
返されるデータの種類:	DOUBLE

構文

VAR(*expression-1*, *expression-2* [,...*expression-n*])

必須引数

expression
数値に評価される有効な式を指定します。引数リストは変数リストで構成することができます。

要件	少なくとも 2 つの非 null または非欠損の引数が必要です。それがない場合は、関数から null または欠損値が返されます。
データの種類	DOUBLE
参照項目:	“<sql-expression> 式” (918 ページ) “FedSQL 式” (46 ページ)

例

次のステートメントは、VAR 関数を示しています。

ステートメント	結果
select var(4,2,3.5,6);	2.7291666667
select var(4,6,.);	2

VARIANCE 関数

式のすべての値のばらつきの尺度を返します。

カテゴリ: 集計
記述統計

返されるデータの種類: DOUBLE

構文

VARIANCE([DISTINCT] *expression*)

必須引数

expression
有効な SQL 式を指定します。

操作 計算を実行するには、少なくとも 2 つの値が必要です。それがない場合は関数から null 値が返されます。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

オプション引数

DISTINCT
(オプション)集計計算から重複行を削除します。

詳細

null 値と SAS 欠損値は無視され、計算には含まれません。

集計関数を使用して、FROM 句にリストされているテーブル全体、または GROUP BY 句で指定されている各グループのデータの統計的要約を生成できます。GROUP BY 句は、指定された 1 つまたは複数の列でデータをグループ化します。GROUP BY 句を使用する場合、SELECT 句または HAVING 句の集計関数は、各グループのデータを要約する方法を FedSQL に指示します。FedSQL は、グループごとに個別に集計関数を計算します。GROUP BY を省略すると、テーブルまたはビュー内のすべての行が単一のグループと見なされます。

例

テーブル: DENSITIES (1260 ページ)

次のステートメントは、VARIANCE 関数を示しています。

ステートメント	結果
select variance(density) from densities;	23498.12
select variance(population) from densities;	1.412E14

関連項目

関数:

- “STDDEV 関数” (837 ページ)

SELECT ステートメント句:

- “SELECT 句” (980 ページ)
- “GROUP BY 句” (993 ページ)
- “HAVING 句” (995 ページ)

VERIFY 関数

式に固有の最初の文字の位置を返します。

カテゴリ: 文字

返されるデータの種類: DOUBLE

構文

VERIFY(*target-expression*, *search–expression*)

必須引数

target-expression
文字列に評価されるか強制変換できる検索先の有効な式を指定します。

要件 リテラル文字列は、単一引用符で囲む必要があります。

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

search-expression
文字列に評価されるか強制変換できる有効な式を指定します。

要件 リテラル文字列は、単一引用符で囲む必要があります。

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

VERIFY 関数は、*target-expression* における *search-expression* に存在しない最初の文字の位置を返します。*target-expression* に *search-expression* と異なる文字がない場合、VERIFY は 0 を返します。

比較

INDEX 関数は、*target-expression* に存在する *search-expression* の最初の出現位置を返し、VERIFY 関数は、*target-expression* における *search-expression* が含まれていない最初の文字の位置を返します。

例

次のステートメントは、VERIFY 関数を示しています。

ステートメント	結果
select verify('abc', 'ab');	3

関連項目

関数:

- “INDEX 関数” (574 ページ)

WEEK 関数

週番号の値を返します。

カテゴリ: 日付と時間

返されるデータの種類: DOUBLE

構文

WEEK([*sas-date*], ['*descriptor*'])

オプション引数

sas-date
SAS 日付値を指定します。*sas-date* 引数を指定しない場合、WEEK 関数は現在の日付の週番号の値を返します。

データの種類 DOUBLE

descriptor
ディスクリプターの値を指定します。次のディスクリプターを大文字または小文字で指定できます。

U
1 年のうちの週番号を指定します。月曜日が週の第 1 日とみなされます。週番号値は、0 から 53 の範囲の 10 進数として表されます。第 53 週に特別な意味はありません。week('31dec2013'd, 'u')の値は 53 です。U はデフォルト値です。

ヒント U および W ディスクリプターは似ていますが、U ディスクリプターが日曜日を週の第 1 日とみなすのに対し、W ディスクリプターは月曜日を週の第 1 日とみなします。

参照項目: “U ディスクリプター” (881 ページ)

V
1 から 53 の範囲の 10 進数として表される週番号値を指定します。月曜日が週の第 1 日とみなされ、年の第 1 週は、1 月 4 日とその年の最初の木曜日の両方を含む週です。1 月の最初の月曜日が 2 日、3 日または 4 日の場合、それより前の日は前年の最後の週に組み込まれます。

参照項目: “V ディスクリプター” (881 ページ)

W
1 年のうちの週番号を指定します。月曜日が週の第 1 日とみなされます。週番号値は、0 から 53 の範囲の 10 進数として表されます。第 53 週に特別な意味はありません。week('31dec2013'd, 'w')の値は 53 です。

ヒント U および W ディスクリプターは似ていますが、U ディスクリプターが日曜日を週の第 1 日とみなすのに対し、W ディスクリプターは月曜日を週の第 1 日とみなします。

参照項目: “W ディスクリプター” (882 ページ)

デフォルト U

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

注 U、V、W 以外のディスクリプターを指定した場合、引数が無効であることを示すメモが SAS ログに書き込まれます。この関数は NULL を返します。

詳細

基本

WEEK 関数は、SAS 日付値を読み取り、週番号を返します。WEEK 関数はロケールに依存せず、計算にはグレゴリオ暦のみを使用します。

U ディスクリプター

U ディスクリプターを使用する WEEK 関数は、SAS 日付値を読み取り、1 年のうちの週番号を返します。週番号値は、先頭に 0 を付けた 0 から 53 までの範囲の 10 進数として表され、最大値は 53 になります。第 0 週は、週の第 1 日が前年になるということを意味します。年の第 5 週は 05 と表されます。

月曜日が週の第 1 日とみなされます。たとえば、week('01jan2013'd, 'u')の値は 0 です。

V ディスクリプター

V ディスクリプターを使用する WEEK 関数は、SAS 日付値を読み取り、週番号を返します。週番号値は、01 から 53 の範囲の 10 進数として表されます。10 進数の先頭に 0 を使用し、最大値は 53 です。各週は月曜日から始まり、年の第 1 週は 1 月 4 日とその年の最初の木曜日の両方を含む週です。1 月の最初の月曜日が 2 日、3 日または 4 日の場合、それより前の日は前年の最後の週に組み込まれます。次の例では、01jan2014 と 31dec2013 は同じ週になります。この週の第 1 日(月曜日)は 30dec2013 です。したがって、week('01jan2014'd, 'v')と week('30dec2013'd, 'v')の両方が値 53 を返します。つまり、両方の日付が 2013 年の第 53 週に含まれるという意味です。

W ディスクリプター

W ディスクリプターを使用する WEEK 関数は、SAS 日付値を読み取り、1 年のうちの週番号を返します。週番号値は、先頭に 0 を付けた 0 から 53 までの範囲の 10 進数として表され、最大値は 53 になります。第 0 週は、週の第 1 日が前年になるということを意味します。年の第 5 週は 05 と表されます。

月曜日が週の第 1 日とみなされます。したがって、week('30dec2013'd, 'w')の値は 1 です。

ディスクリプターの比較

U はデフォルトのディスクリプターです。範囲は 0 から 53 で、週の第 1 日は日曜日です。V ディスクリプターの範囲は 1 から 53 で、週の第 1 日は月曜日です。W ディスクリプターの範囲は 0 から 53 で、週の第 1 日は月曜日です。

ディスクリプターおよび関連付けられた週を次のリストに示します。

- 第 0 週:

  U　現在のグレゴリオ暦年の第 1 週より前の日を示します。

  V　該当しません。

  W　現在のグレゴリオ暦年の第 1 週より前の日を示します。

- 第 1 週:

  U　グレゴリオ暦年の最初の日曜日から始まります。

  V　グレゴリオ暦による前年 12 月 29 日から当年 1 月 4 日までの間の月曜日から始まります。ISO 形式の最初の週は、グレゴリオ暦による前年と当年をまたぐことがあります。

  W　グレゴリオ暦年の最初の月曜日から始まります。

- 年末の週:

  U　1 年の最後の週(第 52 または 53)の日数を 7 日未満にできることを指定します。グレゴリオ暦による連続する 2 年をまたぐ日曜日から土曜日までの期間は、第 52 週と第 0 週、または第 53 週と第 0 週として指定されます。

  V　ISO 形式の年の最後の週(第 52 または 53)の日数が 7 日であることを指定します。ただし、ISO 形式の年の最後の週は、グレゴリオ暦の当年と翌年をまたぐことがあります。

  W　1 年の最後の週(第 52 または 53)の日数を 7 日未満にできることを指定します。グレゴリオ暦による連続する 2 年をまたぐ月曜日から日曜日までの期間は、第 52 と第 0 週、または第 53 週と第 0 週として指定されます。

例

次のステートメントは、2013 年 8 月 16 日の日付における U、V、W ディスクリプターの値を示しています。

ステートメント	結果
select week(date'2013-08-16', 'u');	32
select week(date'2013-08-16', 'v');	33
select week(date'2013-08-16', 'w');	32

関連項目

関数:

- “INTNX 関数” (603 ページ)

出力形式:

- “WEEKDATEw. 出力形式” (212 ページ)
- “WEEKDATXw. 出力形式” (214 ページ)
- “WEEKDAYw. 出力形式” (216 ページ)

WEEKDAY 関数

SAS 日付値から、曜日に対応する整数を返します。

カテゴリ: 日付と時間

返されるデータの種類: DOUBLE

構文

WEEKDAY(*expression*)

必須引数

expression
SAS 日付値を表す有効な式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

WEEKDAY 関数は、曜日を表す整数を生成します(1=日曜日、2=月曜日、...、7=土曜日)。

FedSQL での日付値と時間値の処理法については、“FedSQL の日付と時間” (59 ページ)を参照してください。

例

次のステートメントは、現在日が日曜日の場合の WEEKDAY 関数を示しています。

ステートメント	結果
select weekday(today());	1

WHICHC 関数

文字列のリストから文字列の最初の位置を返します。

カテゴリ: 文字

返されるデータの種類: DOUBLE

構文

WHICHC(*search-expression*, *expression-list-item-1*, *expression-list-item-2* [, ...*expression-list-item-n*])

必須引数

search-expression
文字列式のリストと比較される文字列に評価されるか強制変換できる有効な式を指定します。

要件 リテラル文字列は、単一引用符で囲む必要があります。

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

expression-list-item
文字列に評価されるか強制変換できて、文字列式のリストのメンバーである有効な式を指定します。

要件 リテラル文字列は、単一引用符で囲む必要があります。

リストには少なくとも 2 つの式が必要です。

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

WHICHC 関数は、文字式リストを左から右に検索して、検索式に一致する最初の式を探します。一致するものが見つかった場合、WHICHC は式リスト内のその位置を返します。いずれの式も検索式に一致しない場合、WHICHC は値 0 を返します。

例

次の例では、'Spain'がリストに 2 回出現します。WHICHC 関数は、リスト内の'Spain'の最初の位置を返します。

ステートメント	結果
select whichc('Spain', 'Denmark', 'Germany', 'Austria', 'Spain', 'China', 'Egypt', 'Spain', 'France');	4

関連項目

関数:

- “WHICHN 関数” (885 ページ)

WHICHN 関数

数値のリストから数値の最初の位置を返します。

カテゴリ: 数学

返されるデータの種類: DOUBLE

構文

WHICHN(*search-expression*, *expression-list-item-1*, *expression-list-item-2* [, ...*expression-list-item-n*])

必須引数

search-expression
数値に評価され、数値式のリストと比較される有効な式を指定します。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

expression-list-item
数値に評価され、リストの一部である有効な式を指定します。

要件 リストには少なくとも 2 つの式が必要です。

データの種類 DOUBLE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

WHICHN 関数は、数値式リストを左から右に検索して、検索式に一致する最初の式を探します。一致するものが見つかった場合、WHICHN は式リスト内のその位置を返します。いずれの式も検索式に一致しない場合、WHICHN は値 0 を返します。WHICHN 関数の引数には、任意の数値データ型を指定できます。

例

次の例では、4.5 がリストに 2 回出現します。WHICHN 関数は、リスト内の 4.5 の最初の位置を返します。

ステートメント	結果
select whichn(4.5,7.3, 8.6, 4.5, 4.5, 2.1, 6.4);	3

関連項目

関数:

- “WHICHC 関数” (884 ページ)

YEAR 関数

日付または日時値から年を返します。

カテゴリ: 日付と時間

返されるデータの種類: SMALLINT

構文

YEAR(*date* | *datetime*)

オプション引数

date
日付値を表す有効な式を指定します。

データの種類 DATE

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

datetime
日時値を表す有効な式を指定します。

データの種類 TIMESTAMP

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

例

テーブル: CUSTONLINE (1259 ページ)

次のステートメントは、YEAR 関数を示しています。

ステートメント	結果
select year(endtime) from custonline;	2012 2012 2012 2012 2012 2013 2013 2013 2013 2013
select year(current_time);	2013

関連項目

- “FedSQL の日付と時間” (59 ページ)

関数:

- “DAY 関数” (443 ページ)
- “HOUR 関数” (570 ページ)
- “MINUTE 関数” (670 ページ)
- “MONTH 関数” (676 ページ)
- “SECOND 関数” (824 ページ)

YIELDP 関数

定期的なキャッシュフローストリーム(債権など)の最終利回りを返します。

カテゴリ: 財務

返されるデータの種類: DOUBLE

構文

YIELDP(*A*, *c*, *n*, *K*, k_0, *p*)

必須引数

A
額面価格を指定します。

範囲 $A > 0$

$A > 0$

データの種類 DOUBLE

c

分数で表す、年間の名目クーポン率を指定します。

範囲 $0 \leq c < 1$

$0 \leq c < 1$

データの種類 DOUBLE

n

1 年当たりのクーポン数を指定します。

範囲 $n > 0$

$n > 0$

データの種類 DOUBLE

K

決済日から満期までの残りのクーポン数を指定します。

範囲 $K > 0$

$K > 0$

データの種類 DOUBLE

k_0

決算日から次のクーポンまでの時間を、1 年ごとの分数で指定します。

範囲 $0 < k_0 \leq \frac{1}{n}$

$0 < k_0 \leq 1/n$

データの種類 DOUBLE

p

未払い利息込みの価格を指定します。

範囲 $p > 0$

$p > 0$

データの種類 DOUBLE

詳細

YIELDP 関数は次の関係に基づきます。

$$P = \sum_{k=1}^{K} c(k) \frac{1}{\left(1 + \frac{y}{n}\right)^{t_k}}$$

前述の式には次の関係が適用されます。

- $t_k = nk_0 + k - 1$
- $c(k) = \frac{c}{n} A \quad for\ k = 1, \ldots, K-1$
- $c(K) = \left(1 + \frac{c}{n}\right) A$

YIELDP 関数は、*y* に対して解決します。

例

次の例では、YIELDP 関数は、額面$1000、年間クーポン率 0.01、年間クーポン数 4、残りクーポン数 14 の債権の最終利回りを返します。決算日から次のクーポン日までの時間は 0.165、未払い利息込みの価格は 800 です。返される値は 0.077503 です。

```
select yieldp(1000,.01,4,14,.165,800);
```

YRDIF 関数

指定した日数計算規則に従って 2 つの日付の差を年単位で返します。人の年齢を返します。

カテゴリ: 日付と時間

返されるデータの種類: DOUBLE

構文

YRDIF(*start-date*, *end-date*[, *basis*])

必須引数

start-date
開始日を識別する SAS 日付値を指定します。

データの種類 DOUBLE

end-date
終了日を識別する SAS 日付値を指定します。

データの種類 DOUBLE

オプション引数

basis
SAS で日付の差(人の年齢)が計算される方法を記述する文字定数、変数または式を指定します。有効な文字列は、'30/360' (30 日の月と 360 日の年)、'ACT/ACT'、'ACT/360',、'ACT/365'、'AGE'です。たとえば、'ACT/360'では、年数の計算に日付間の実際の日数を使用します。この値は、各年の実際の日数に関係なく、日数を 360 で除算して計算されます。詳細については、SAS DS2 言語リファレンスを参照してください。

SAS で日付の差(人の年齢)が計算される方法を記述する文字定数、変数または式を指定します。有効な文字列は次のとおりです。

'30/360'
年数の計算で 30 日の月と 360 日の年を指定します。各月または年の実際の日数に関係なく、各月は 30 日、各年は 360 日とみなされます。

別名 '360'

ヒント いずれかの日付が月末になる場合、その日は 30 日の月の最終日として扱われます。

'ACT/ACT'
年数の計算には、日付間の実際の日数を使用します。この値は、365 日の年の日数を 365 で除算して得た値と、366 日の年の日数を 366 で除算して得た値を足して計算されます。

別名 'Actual'

'ACT/360'
年数の計算には、日付間の実際の日数を使用します。この値は、各年の実際の日数に関係なく、日数を 360 で除算して計算されます。

'ACT/365'
年数の計算には、日付間の実際の日数を使用します。この値は、各年の実際の日数に関係なく、日数を 365 で除算して計算されます。

'AGE'
人の年齢が計算されることを指定します。

第 3 引数を指定しない場合、AGE が *basis* のデフォルト値になります。

データの種類 CHAR、NCHAR、NVARCHAR、VARCHAR

詳細

会計アプリケーションでの YRDIF の使用

基本

YRDIF 関数は、第 3 引数 *basis* が存在する場合に、確定利付証券の利息を計算するために使用できます。指定した日数計算規則に従って 2 つの日付の差を返します。

ACT/ACT 基準を使用する計算

ACT/ACT 基準を使用する YRDIF 計算では、365 日の年と 366 日の年の両方が考慮されます。たとえば、n365 が 365 日の年の開始日から終了日までの日数と等しく、n366 が 366 日の年の開始日から終了日までの日数と等しい場合、YRDIF 計算は YRDIF=n365/365.0+ n366/366.0 として計算されます。この計算は、金融関係の文献に記されている一般に理解されている *ACT/ACT* 日数計算基準に対応します。*basis* の値には、他に *30/360*、*ACT/360* および *ACT/365* があります。各基準には、特定の金融商品の利息の支払いを計算するために従う必要がある明確な意味があります。

人の年齢の計算

YRDIF 関数で人の年齢を計算できます。最初の 2 つの引数 *start-date* と *end-date* は必須です。*basis* の値が AGE の場合、YRDIF は年齢を計算します。年齢の計算では、うるう年が考慮されます。*basis* のその他の値は、人の年齢の計算には無効です。

例

例 1: basis に基づいた年の差の計算

次の例では、YRDIF は *basis* の各 ACT オプションに基づいて 2 つの日付の差(年数)を返します。

```
select yrdif(date'1998-10-16', date'2010-02-06', '30/360') as y30360,
  yrdif(date'1998-10-16', date'2010-02-06', 'ACT/ACT') as yactact,
  yrdif(date'1998-10-16', date'2010-02-06', 'ACT/360') as yact360,
  yrdif(date'1998-10-16', date'2010-02-06', 'ACT/365') as yact365;
quit;
```

アウトプット 4.15　ACT 計算からの出力

Y30360	YACTACT	YACT360	YACT365
11.30556	11.30959	11.475	11.31781

例 2: 人の年齢の計算

次のように、YRDIF 関数で *basis* 引数の AGE オプションを使用して、人の年齢を計算できます。

```
select yrdif(date'1998-10-16', date'2010-02-16', 'AGE');
```

SAS は値 11.33699 を返します。

関連項目

関数:

- “DATDIF 関数” (436 ページ)

参考文献

“Day count convention.” *Wikipedia*, 2010. URI: Day count convention. Accessed on May 1, 2013..

ISDA International Swaps and Derivatives Association, Inc “EMU and Market Conventions: Recent Developments.” 1998. ISDA. URI: EMU and Market Conventions: Recent Developments.

Mayle, Jan. 1994. *Standard Securities Calculation Methods – Fixed Income Securities Formulas for Analytic Measures*. 2 巻 New York, , New York: Securities Industry Association.

YYQ 関数

年と四半期の値から SAS 日付値を返します。

カテゴリ:	日付と時間
返されるデータの種類:	DOUBLE

構文

YYQ(*year*, *quarter*)

必須引数

year
年を表す 2 桁または 4 桁の整数に評価される有効な式を指定します。

操作　YEARCUTOFF=システムオプションは、2 桁の日付の年の値を定義します。

データの種類　DOUBLE

参照項目:　“<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

quarter
年の四半期(1、2、3、4)を指定します。

データの種類　DOUBLE

参照項目:　“<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

YYQ 関数は、指定した四半期の最初の日に対応する SAS 日付値を返します。*year* または *quarter* のいずれかが null または欠損している場合、または四半期の値が無効である場合、結果は null 値または欠損値になります。

例

次のステートメントは、YYQ 関数を示しています。

ステートメント	結果

select yyq(2006,3);
select put(yyq(2006,3), date7.);
select put(yyq(2006,3), date9.);
	16983
	01JUL06
	01JUL2006

select yyq(2006,4); select put(yyq(2006,4), date9.);	17075 01OCT2006

関連項目

概念:

- “FedSQL の日付と時間” (59 ページ)

関数:

- “QTR 関数” (792 ページ)

- “YEAR 関数” (887 ページ)

5

FedSQL の式と述語

式と述語の概要 **897**

ディクショナリ **898**
BETWEEN 述語 898
CASE 式 899
COALESCE 式 903
DISTINCT 述語 904
EXISTS 述語 906
IN 述語 907
IS FALSE 述語 908
IS MISSING 述語 909
IS NULL 述語 911
IS TRUE 述語 913
IS UNKNOWN 述語 914
LIKE 述語 915
NULLIF 式 917
<sql-expression> 式 918

式と述語の概要

*式*とは、FedSQL が評価して単一の値を返す記号と演算子の組み合わせです。式は、単一の定数または列のように単純な場合もあれば、演算子によって結合された複数の式のように複雑な場合もあります。

この章では、CASE、COALESCE、および NULLIF の 3 つの条件式と、他のすべてのタイプの式を定義する一般的な式である<sql-expression>に焦点を当てます。

*述語*は、true、false、または unknown のいずれかに評価される条件を指定します。これらは、WHERE 句や HAVING 句、および結合条件の FROM 句でよく使われます。

ディクショナリ

BETWEEN 述語

列値が値の範囲内にある行を選択します。

構文

expression [NOT] **BETWEEN** *expression* AND *expression*

引数

expression
有効な SQL 式を指定します。

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

BETWEEN 述語は、次の基準を使用して、選択する列値の範囲を指定します。

- SQL 式は、互換性のあるデータ型である必要があります。
- BETWEEN 条件は境界値を範囲として評価するため、小さい方の数量を先に指定する必要はありません。
- NOT 論理演算子を使用して、数値の範囲を除外することができます。たとえば、NOT を使用して、1-15 (両端を含む)の顧客番号を削除すると、最近獲得した顧客に関するデータを取得できます。

例

```
select * from invtry
  where invtry.name
    between 'A' and 'Mzzz';
```

関連項目

式:

- “<sql-expression> 式” (918 ページ)

CASE 式

検索条件と値の比較を満たす結果値を選択します。

構文

CASE [*case-expression*]
WHEN *when-expression* THEN *result-expression*
...
[WHEN *when-expression* THEN *result-expression*]
[ELSE *result-expression*]
END

引数

case-expression
有効な SQL 式を指定します。その評価結果のテーブル列の値が *when-expression* と比較されます。

参照項目: “<sql-expression> 式” (918 ページ)
“FedSQL 式” (46 ページ)

when-expression
有効な SQL 検索条件式または値式を指定します。

- *case-expression* が指定されていない場合、*when-expression* は、評価結果が true または false になる検索条件式です。
- *case-expression* が指定されている場合、*when-expression* は SQL 値式で、*case-expression* と比較され、評価結果は true または false になります。

参照項目: “<sql-expression> 式” (918 ページ)

result-expression
評価結果が値になる SQL 式を指定します。

参照項目: “<sql-expression> 式” (918 ページ)

詳細

CASE 式は、特定の条件が満たされた場合に値を選択します。*case-expression* 引数は、テーブルの各行に対して条件付きで評価される単一の値を返します。クエリまたは作成されるテーブル内のすべての行ではなく一部の行に対して CASE 式を実行するには、WHEN-THEN 句を使用します。オプションの ELSE 式は、THEN 式が実行されない場合に代替アクションを提供します。

case-expression を省略した場合、*when-expression* はブール(true または false)値として評価されます。*when-expression* がゼロと null 以外の結果を返す場合、WHEN 句は true です。 *case-expression* が指定されている場合は、*when-expression* と比較され、等しいかどうかが確認されます。*case-expression* が *when-expression* と等しい場合、WHEN 句は true です。

実行されている行に対して *when-expression* が true の場合、THEN に続く *result-expression* が実行されます。*when-expression* が false の場合、FedSQL は、すべてが評価されるまで、次の *when-expression* の評価を続けます。すべての *when-expression* が false の場合、FedSQL は ELSE 式を実行し、その結果が CASE 式の結果になります。ELSE 式が存在せず、すべての *when-expression* が false の場合、CASE 式の結果は null になります。

CASE 式は、SELECT 句の項目として、および SQL 式のいずれかのオペランドとして使用できます。

比較

COALESCE 式と NULLIF 式は、CASE 式のバリエーションです。

次の CASE 式と COALESCE 式は同等です。

```
case
  when value1 is not null
    then value1
  when value2 is not null
    then value2
  else value3
end
```

```
coalesce(value1, value2, value3)
```

次の CASE 式と NULLIF 式は同等です。

```
case
  when value1 = -1 then null
  else value1
end
```

```
nullif(value1, -1);
```

例

例 1: 検索条件を使用した CASE 式

テーブル: WORLDTEMPS (1277 ページ)

```
proc fedsql;
  select AvgLow,
    case
      when AvgLow < 32 then AvgLow + 2
      when ((AvgLow < 60) and (AvgLow > 32)) then AvgLow + 5
        when AvgLow > 60 then AvgLow + 10
        else AvgLow
    end
    as Adjusted from worldtemps;
quit;
```

SAS は次のテーブルを作成します。

アウトプット 5.1　検索条件を使用した CASE の使用(部分的な結果)

AvgLow	ADJUSTED
45	50
33	38
41	46
44	49
69	79
17	19
29	31
25	27
43	48
68	78
24	26
25	27
48	53

例 2: 値を使用した CASE 式

テーブル: WORLDTEMPS (1277 ページ)

```
proc fedsql;
  create table tempTable as select City, Country,
    case Country
      when 'Algeria' then 'Africa'
      when 'Nigeria' then 'Africa'
```

```
            when 'Netherlands' then 'Europe'
            when 'Spain' then 'Europe'
            when 'Switzerland' then 'Europe'
            when 'China' then 'Asia'
            when 'India' then 'Asia'
            when 'Venezuela' then 'South America'
            else null
          end
        as Continent from worldtemps;
        select * from tempTable where Continent is not null;
        drop table tempTable force;
      quit;
```

SAS は次のテーブルを作成します。

アウトプット 5.2　値を使用した CASE

City	Country	CONTINENT
Algiers	Algeria	Africa
Amsterdam	Netherlands	Europe
Beijing	China	Asia
Bombay	India	Asia
Calcutta	India	Asia
Caracas	Venezuela	South America
Geneva	Switzerland	Europe
Hong Kong	China	Asia
Lagos	Nigeria	Africa
Madrid	Spain	Europe
Shanghai	China	Asia
Zurich	Switzerland	Europe

関連項目

式:

- “COALESCE 式” (903 ページ)
- “NULLIF 式” (917 ページ)
- <search-condition> - “SELECT ステートメント” (977 ページ)

COALESCE 式

列のリストから最初の null 以外の値を返します。

制限事項: SAS データセット、SPD Engine データセット、および SPD Server テーブルは、null 値を空白文字列として処理します。

構文

COALESCE(*expression* [, ...*expression*])

引数

expression
有効な SQL 式を指定します。

参照項目: "<sql-expression> 式" (918 ページ)

"FedSQL 式" (46 ページ)

詳細

COALESCE は、同じデータ型の SQL 式を 1 つ以上受け入れます。COALESCE 式は、リストされている順に各 SQL 式の値をチェックし、最初の null 以外の値を返します。SQL 式が 1 つだけリストされている場合、COALESCE 式はその SQL 式の値を返します。すべての引数のすべての値が null の場合、COALESCE 式は null 値を返します。

一部の SQL DBMS では、COALESCE 式は IFNULL 式と呼ばれます。

注: クエリに多数の COALESCE 式が含まれている場合は、かわりに自然結合を使用する方が効率的です。詳細については、"自然結合" (991 ページ)を参照してください。

比較

COALESCE 式は、CASE 式のバリエーションです。たとえば、次の 2 つのコードセットは同等です。

```
coalesce(value1, value2, value3)
```

```
case
```

```
  when value1 is not null
    then value1
  when value2 is not null
    then value2
  else value3
end;
```

関連項目

式:

- “CASE 式” (899 ページ)

DISTINCT 述語

入力または出力として一意の行のみを選択するように指定します。

構文

形式 1: *aggregate-function* (**DISTINCT** *expression*);

形式 2: **SELECT DISTINCT** <select-list> FROM <table-expression>;

引数

aggregate-function
任意の集計関数を指定できます。

expression
有効な SQL 式を指定します。

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

SELECT <select-list> FROM <table-expression>
テーブルから行を取得するクエリです。

参照項目: SELECT ステートメントでの DISTINCT 述語の使用の詳細については、“SELECT 句” (980 ページ)を参照してください。

詳細

DISTINCT 述語の形式は両方とも、重複削除の形式を実行します。集計関数で DISTINCT キーワードを使用すると、FedSQL は集計計算への入力として一意の値のみを提供します。SELECT DISTINCT を使用すると、FedSQL はクエリの結果セット内の一意の行のみを返します。

注: 2 つの null 値は別個のものとは*見なされません*。

例

- select count(distinct avghigh) from worldtemps;
- select distinct c1.employee, firstname, salary
  from company as c1;

関連項目

ステートメント:

- “SELECT ステートメント” (977 ページ)

関数:

- “AVG 関数” (311 ページ)
- “COUNT 関数” (413 ページ)
- “CSS 関数” (423 ページ)
- “KURTOSIS 関数” (629 ページ)
- “MAX 関数” (663 ページ)
- “MIN 関数” (668 ページ)
- “NMISS 関数” (682 ページ)
- “PROBT 関数” (787 ページ)
- “RANGE 関数” (799 ページ)
- “SKEWNESS 関数” (829 ページ)
- “STD 関数” (836 ページ)
- “STDDEV 関数” (837 ページ)
- “STDERR 関数” (838 ページ)
- “STUDENTS_T 関数” (840 ページ)
- “SUM 関数” (846 ページ)
- “USS 関数” (873 ページ)

- “VARIANCE 関数” (877 ページ)

EXISTS 述語

サブクエリが 1 つ以上の行を返すかどうかをテストします。

構文

[NOT] **EXISTS** (*select-statement*)

引数

select-statement
SELECT ステートメントでサブクエリを指定します。

参照項目: “SELECT ステートメント” (977 ページ)

詳細

EXISTS 述語は、右オペランドがサブクエリである演算子です。EXISTS 述語の結果は、サブクエリが少なくとも 1 行に解決された場合に true となります。NOT EXISTS 述語の結果は、サブクエリがゼロ行と評価された場合に true となります。

例

次のクエリは、サブクエリの条件に基づいてテーブル PAYROLL をサブセット化します。STAFF.IDNUM の値がテーブル STAFF の値 CT と同じ行にある場合、テーブル PAYROLL の一致する IDNUM が出力に含まれます。したがって、クエリはテーブル PAYROLL から、CT に住むすべての従業員を返します。

```
proc fedsql;
  select *
    from payroll p
      where exists (select * from staff s
        where p.idnumber=s.idnum and state='CT');
quit;
```

関連項目

ステートメント:

- “SELECT ステートメント” (977 ページ)

IN 述語

セットメンバーシップをテストします。

構文

expression [NOT] **IN** (*select-statement* | *constant* [, ...*constant*])

引数

expression
有効な SQL 式を指定します。

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

select-statement
SELECT ステートメントでサブクエリを指定します。

参照項目: “SELECT ステートメント” (977 ページ)

constant
固定値を示す数値または引用符で囲まれた文字列(またはその他の特殊表記)を指定します。定数は *リテラル*とも呼ばれます。

詳細

IN 述語は、左側の SQL 式によって返される列値が、右側の(クエリ式によって返される定数または値の)セットのメンバーであるかどうかをテストします。左側のオペランドの値が、右側のオペランドによって定義された値のセットに含まれている場合、IN 条件は true です。

NOT IN 述語は、返された値を否定します。

例

テーブル: WORLDTEMPS (1277 ページ)

```
proc fedsql;
  select city, country
    from worldtemps
```

```
        where avghigh in (90, 97);
quit;
```

SAS は次のテーブルを作成します。

アウトプット 5.3　IN 述語の例の出力テーブル

CITY	COUNTRY
Algiers	Algeria
Bombay	India
Calcutta	India
Lagos	Nigeria

IS FALSE 述語

false 値をテストします。

構文

(*expression*) **IS** [NOT] **FALSE**

引数

expression
有効な SQL 式を指定します。

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

IS FALSE は、false 値をテストする述語です。IS FALSE は、WHERE 句、ON 句、および HAVING 句で使用されます。IS FALSE 述語は、SQL 式の結果が false の場合は true に、true の場合は false に解決されます。

比較

IS TRUE 述語は、true 値をテストします。

例

テーブル: WORLDCITYCOORDS (1276 ページ)

```
proc fedsql;
  select city, country
    from worldcitycoords
      where (country = 'Brazil') is false;
quit;
```

SAS は次のテーブルを作成します。

アウトプット 5.4　IS FALSE の例の出力テーブル(部分的)

City	Country
Buenos Aires	Argentina
Brisbane	Australia
Nassau	Bahamas
La Paz	Bolivia
Havre	Canada
Quebec	Canada
Winnipeg	Canada
Punta Arenas	Chile
Valparaiso	Chile
Cali	Colombia

関連項目

述語:

- “IS TRUE 述語” (913 ページ)
- “IS UNKNOWN 述語” (914 ページ)
- <search-condition> in the “SELECT ステートメント” (977 ページ)

IS MISSING 述語

SAS ネイティブデータストアの SAS 欠損値をテストします。

構文

expression **IS** [NOT] **MISSING**

引数

expression
有効な SQL 式を指定します。

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

IS MISSING は、SAS 欠損値をテストする述語です。IS MISSING は、WHERE 句、ON 句、および HAVING 句で使用されます。IS MISSING 述語は、SQL 式の結果が SAS 欠損値である場合は true に解決され、SAS 欠損値でない場合は false に解決されます。

IS MISSING 述部は、SAS ネイティブデータストアでのみ使用できます。DOUBLE および CHAR データ型のみが欠損値をサポートしています。

比較

IS NULL 述語は、null 値をテストします。

例

テーブル: WORLDCITYCOORDS (1276 ページ)

この例では、次のコードによってテーブル WorldCITYCOORDS からテーブル WorldCITYCOORDSMISS を作成し、Buenos Aires と Brisbane の City を欠損値に設定します。

```
proc fedsql;
   drop table worldcitycoordsmiss force;
   create table worldcitycoordsmiss as select * from worldcitycoords;
   update worldcitycoordsmiss set city=' '
   where city in ('Buenos Aires', 'Brisbane');
quit;
```

次では、テーブル WORLDCITYCOORDSMISS の欠損値を選択します。

```
proc fedsql;
  select *
    from worldcitycoordsmiss
    where city is missing;
```

```
quit;
```

SAS は次のテーブルを作成します。

アウトプット 5.5　IS MISSING の例の出力テーブル

City	Country	Latitude	Longitude
	Argentina	-34	-59
	Australia	-27	153

関連項目

述語:

- “IS NULL 述語” (911 ページ)
- <search-condition> - “SELECT ステートメント” (977 ページ)

IS NULL 述語

null 値をテストします。

構文

expression **IS** [NOT] **NULL**

引数

expression
有効な SQL 式を指定します。

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

IS NULL は、null 値をテストする述語です。IS NULL は、WHERE 句、ON 句、および HAVING 句で使用されます。IS NULL 述語は、SQL 式の結果が null の場合は true に解決され、null でない場合は false に解決されます。

比較

IS MISSING 述語は、SAS ネイティブデータストアの SAS 欠損値をテストします。

例

テーブル: WORLDCITYCOORDS (1276 ページ)

この例では、次のコードによってテーブル WorldCITYCOORDS からテーブル WorldCITYCOORDSNULL を作成し、Buenos Aires と Brisbane の City を null に設定します。

```
proc fedsql;
   drop table worldcitycoordsnull force;
   create table worldcitycoordsnull as select * from worldcitycoords;
   update worldcitycoordsnull set city=null
   where city in ('Buenos Aires', 'Brisbane');
quit;
```

次のコードは、null ではない City 値を選択します。

```
proc fedsql;
  select city, country
    from worldcitycoordsnull
      where city is not null;
quit;
```

SAS は次のテーブルを作成します。

アウトプット 5.6　IS NOT NULL の出力テーブル例(部分的)

City	Country
Nassau	Bahamas
La Paz	Bolivia
Porto Alegre	Brazil
Recife	Brazil
Rio de Janeiro	Brazil
Salvador	Brazil
Sao Paulo	Brazil
Havre	Canada
Quebec	Canada
Winnipeg	Canada
Punta Arenas	Chile
Valparaiso	Chile

関連項目

述語:

- “IS MISSING 述語” (909 ページ)
- <search-condition> in the “SELECT ステートメント” (977 ページ)

IS TRUE 述語

真の値をテストします。

構文

(*expression*) **IS** [NOT] **TRUE**

引数

expression
有効な SQL 式を指定します。

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

IS TRUE は、true 値をテストする述語です。IS TRUE は、WHERE 句、ON 句、および HAVING 句で使用されます。IS TRUE 述語は、SQL 式の結果が true の場合は true に、false の場合は false に解決されます。

比較

IS FALSE 述語は、false 値をテストします。

例

テーブル: WORLDCITYCOORDS (1276 ページ)

```
proc fedsql;
  select city, country
    from worldcitycoords
      where (country = 'Brazil') is true;
quit;
```

SAS は次のテーブルを作成します。

アウトプット 5.7　IS TRUE 出力例

City	Country
Porto Alegre	Brazil
Recife	Brazil
Rio de Janeiro	Brazil
Salvador	Brazil
Sao Paulo	Brazil

関連項目

述語:

- “IS FALSE 述語” (908 ページ)
- “IS UNKNOWN 述語” (914 ページ)

IS UNKNOWN 述語

unknown 値をテストします。

構文

expression **IS** [NOT] **UNKNOWN**

引数

expression
有効な SQL 式を指定します。

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

IS UNKNOWN は、unknown 値をテストする述語です。IS UNKNOWN は、WHERE 句、ON 句、および HAVING 句で使用されます。IS UNKNOWN 述語は、SQL 式の結果が unknown な場合は true に、有効な値の場合は false に解決されます。

関連項目

述語:

- “IS FALSE 述語” (908 ページ)
- “IS TRUE 述語” (913 ページ)
- <search-condition> - “SELECT ステートメント” (977 ページ)

LIKE 述語

一致パターンをテストします。

構文

expression [NOT] **LIKE** *expression*

引数

expression
文字列型またはバイナリ文字列型のいずれかである有効な SQL 式を指定します。

ヒント　構文の右側にある SQL 式、つまりパターンは、リテラルである可能性が最も高いです。

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

LIKE 述語の概要

LIKE 述語は、文字列をパターンマッチング指定と比較して行を選択します。左オペランドが右オペランドで指定されたパターンと一致する場合、true に解決され、一致した 1 つまたは複数の文字列が表示されます。

エスケープ文字はサポートされていません。

注: 行が返されない場合、結果は null 値になります。

検索のパターン

パターンには、次の 3 つのクラスの文字が含まれます。

アンダースコア(_)
　任意の 1 文字に一致します。

パーセント記号(%)
　任意の 0 文字以上のシーケンスに一致します。

その他の文字
　その文字に一致します。

これらのパターンは、一致させる文字の前、後、または両側に指定できます。LIKE 条件では、大文字と小文字が区別されます。

次のリストでは、Smith、Smooth、Smothers、Smart、Smuggle の値が使用されています。

'Sm%'
　Smith、Smooth、Smothers、Smart、Smuggle と一致します。

'%th'
　Smith, Smooth と一致します。

'S__gg%'
　Smuggle と一致します。

'S_o'
　3 文字の単語と一致するため、ここでは一致するものはありません。

'S_o%'
　Smooth, Smothers と一致します。

'S%th'
　Smith, Smooth と一致します。

'M'
　単一の大文字の m のみと一致するため、ここでは一致するものはありません。

大文字と小文字が混在する文字列の検索

大文字と小文字が混在する文字列を検索するには、LIKE 条件に入る前に、UPPER 関数を使用してすべての名前を大文字にします。

```
upper(name) like 'SM%';
```

注: %文字を使用する場合は、末尾の空白の影響に注意してください。値を一致させるためには、TRIM 関数による末尾の空白の削除が必要になる場合があります。

例

テーブル: DENSITIES (1260 ページ)

```
select name, population
  from densities
    where name like 'Al%';
```

関連項目

関数:

- “TRIM 関数” (866 ページ)

NULLIF 式

指定された 2 つの式が等しい場合は null 値を返します。それ以外の場合は、最初の式を返します。

制限事項: BASE ファイル形式では、状況によっては、null 値が DOUBLE 値として処理される場合と、空白文字列として処理される場合があります。詳細については、“FedSQL が null および SAS 欠損値を処理する方法” (21 ページ)を参照してください。

構文

NULLIF(*expression-1, expression-2*)

引数

expression
有効な SQL 式を指定します。

データの種類 すべてのデータ型が有効です。

参照項目: “<sql-expression> 式” (918 ページ)

“FedSQL 式” (46 ページ)

詳細

NULLIF 式は、2 つの SQL 式を比較し、等しい場合は null 値を返します。NULLIF 式を使用すると、欠損値や適用できない値を null 値に置き換えたり、null 値に対する SQL の動作を使用したりできます。

比較

NULLIF 式は、特殊な CASE 式の短縮構文です。たとえば、学生がテストを欠席した場合、GRADES テーブルに-1 が入力されます。この-1 を null 値に置き換えるには、次の CASE コードを使用します。

```
update grades
  set testscore =
```

```
  CASE
   when testscore = '-1' then null
   ELSE testscore
  END;
```

次のコードでは、より短い NULLIF 式を使用しています。

```
update grades
  set testscore = NULLIF(testscore, '-1');
```

IFNULL 関数は、2 つの SQL 式を比較し、最初の SQL 式が null 値の場合は 2 番目の SQL 式を返します。NULLIF 式は、2 つの SQL 式を比較し、その 2 つの SQL 式が等しい場合は null 値を返します。

例

テーブル: WORLDCITYCOORDS (1276 ページ)

```
missingLong= '.L';
update worldcitycoords
set longitude = nullif(missingLong, '.');
select city
  from worldcitycoords
    where Longitude='.L';
```

関連項目

式:

- “CASE 式” (899 ページ)
- “COALESCE 式” (903 ページ)

関数:

- “IFNULL 関数” (573 ページ)

<sql-expression> 式

記号と演算子または述語の組み合わせから単一の値を生成します。

構文

```
<sql-expression>::=
    constant
    | [alias] column
```

| *function*

| (*scalar-subquery*)

| (<sql-expression>)

| <sql-expression> {*operator* | *predicate*} <sql-expression>

引数

constant
単一の特定のデータ値を表す数値、引用符で囲まれた文字列、または日時値です。

alias
SELECT ステートメントの FROM 句で AS キーワードを使用してテーブルに割り当てられるエイリアスです。

column
列の名前です。

function
SAS または集計関数です。

参照項目: “FedSQL 関数” (249 ページ)

scalar-subquery
単一の値を返すサブクエリです。

operator
1 つ以上の式で実行されるアクションを指定する記号です。FedSQL 演算子のリストについては、“演算子と評価順序” (920 ページ)を参照してください。式には、CASE または COALESCE 式を含めることもできます。詳細については、“CASE 式” (899 ページ)または “COALESCE 式” (903 ページ)を参照してください。

FedSQL を使用するように PROC SQL プログラムを書き換える場合は、FedSQL 言語でサポートされる演算子が PROC SQL でサポートされる演算子とは異なることに注意してください。詳細については、“比較演算子” (1172 ページ)を参照してください。

predicate
true、false、または unknown を返す式です。

有効な述語は次のとおりです。

- “BETWEEN 述語” (898 ページ)
- “DISTINCT 述語” (904 ページ)
- “EXISTS 述語” (906 ページ)
- “IN 述語” (907 ページ)
- “IS FALSE 述語” (908 ページ)
- “IS MISSING 述語” (909 ページ)
- “IS NULL 述語” (911 ページ)
- “IS TRUE 述語” (913 ページ)
- “IS UNKNOWN 述語” (914 ページ)

- “LIKE 述語” (915 ページ)

詳細

<sql-expression>の概要

単純な式には、単一の定数、列名、または関数があります。複雑な式は、演算子または述語によって結合された 2 つ以上の単純な式です。

式の関数

式には、SAS 関数または集計関数を含めることができます。SAS 関数は、1 つ以上の引数に対して計算またはシステム操作を実行し、値を返します。集計関数は、FROM 句にリストされているテーブル全体、または GROUP BY 句で指定されている各グループのデータの統計的要約を生成します。GROUP BY を省略すると、テーブル内のすべての行が単一のグループと見なされます。集計関数は、テーブルの各行または列のすべての値を減らして 1 つの要約値または集計値にします。たとえば、列の合計(1 つの値)は、列のすべての値を加算した結果です。

式のサブクエリ

FedSQL では、単純な列値または定数を使用できる式の任意の場所で、スカラーサブクエリ(かっこ内に含まれる)を使用できます。この場合、サブクエリは単一の値(つまり、列が 1 つだけの 1 つの行)を返す必要があります。

演算子と評価順序

FedSQL でサポートされている演算子と述語を次の表に示します。これらは評価される順序でリストされています。

表 5.1　式、演算子および述語と評価の順序

グループ	式、演算子および述語	説明
0	( )	囲まれた式が最初に評価されるように強制する
1	CASE 式	参照: “CASE 式” (899 ページ)
2	**	累乗
	単項+、単項-	正数または負数を示す
3	*	乗算
	/	除算
4	+	加算

グループ	式、演算子および述語	説明
	-	減算
5	||	連結
6	[NOT] BETWEEN 述語	参照: “BETWEEN 述語” (898 ページ)
	DISTINCT 述語	参照: “DISTINCT 述語” (904 ページ)
	[NOT] EXISTS 述語	参照: “EXISTS 述語” (906 ページ)
	[NOT] IN 述語	参照: “IN 述語” (907 ページ)
	IS [NOT] TRUE 述語	参照: “IS TRUE 述語” (913 ページ)
	IS [NOT] FALSE 述語	参照: “IS FALSE 述語” (908 ページ)
	IS [NOT] MISSING 述語	参照: “IS MISSING 述語” (909 ページ)
	IS [NOT] NULL 述語	参照: “IS NULL 述語” (911 ページ)
	IS [NOT] UNKNOWN 述語	参照: “IS UNKNOWN 述語” (914 ページ)
	LIKE 述語	参照: “LIKE 述語” (915 ページ)
7	=	等しい
	^=、<>、!=	等しくない
	>	より大きい
	<	より小さい
	>=	以上
	<=	以下
8	AND	論理 AND を示す
9	OR	論理 OR を示す
10	NOT	論理 NOT を示す

SAS の欠損値と null 値は、常に照合順序の最小値として表示されます。

かっこを使用して、値をグループ化したり、数式をネストしたりできます。かっこは式を読みやすくし、演算子の評価順序を変更するためにも使用できます。かっこを使用した式の評価は、かっこの最も深いレベルから始まり、外側に進みます。たとえば、SAS は A+B*C を A+(B*C)として評価しますが、かっこを追加して、(A+B)*C と評価して別の結果を得ることができます。

関連項目

ステートメント:

- “SELECT ステートメント” (977 ページ)
- “FedSQL 式” (46 ページ)

6

FedSQL 入力形式

入力形式の定義 923
一般的な入力形式構文 923
FedSQL での入力形式の使用方法 924
FedSQL で入力形式を指定する方法 926
FedSQL 入力形式の検証 926
FedSQL 入力形式の例 926

入力形式の定義

*入力形式*は、値が列に読み込まれる方法を決定する命令です。たとえば、次の値にはドル記号とカンマが含まれています。

$1,000,000

数値 1000000 を変数に保存する前にドル記号($)とカンマ(,)を削除するには、この値を COMMA11.入力形式で読み取ります。

一般的な入力形式構文

FedSQL 入力形式の構文は次のとおりです。

[$]*informat*[*w*].[*d*]

引数

$
文字入力形式であることを表します。数値入力形式には使用されません。

informat
入力形式名です。入力形式には、SAS 入力形式、PROC FORMAT の INVALUE ステートメントを使用して定義されたユーザー定義の入力形式があります。ユーザー定義の入力形式の詳細については、 *Base SAS プロシジャガイド*の PROC FORMAT を参照してください。

w
入力形式の幅です。大部分の入力形式では、入力データのカラム数となります。

d
オプションの小数点以下の桁数です(数値入力形式)。入力データを 10 の *d* 乗で除算します。

注: 小数点以下 31 桁まで読み取れるように数値入力形式を指定している場合でも、多くのコンピューターで使用されている 8 バイトの浮動小数点数表記の制限により、小数点以下 12 桁を超える浮動小数点数では精度が低下することがあります。

入力形式は、名前の一部として必ずピリオド(.)を含みます。入力形式の *w* 値と *d* 値を省略すると、デフォルト値が使用されます。データに小数点が含まれる場合、*d* 値は無視され、実際に入力データにある小数点以下桁数が読み込まれます。

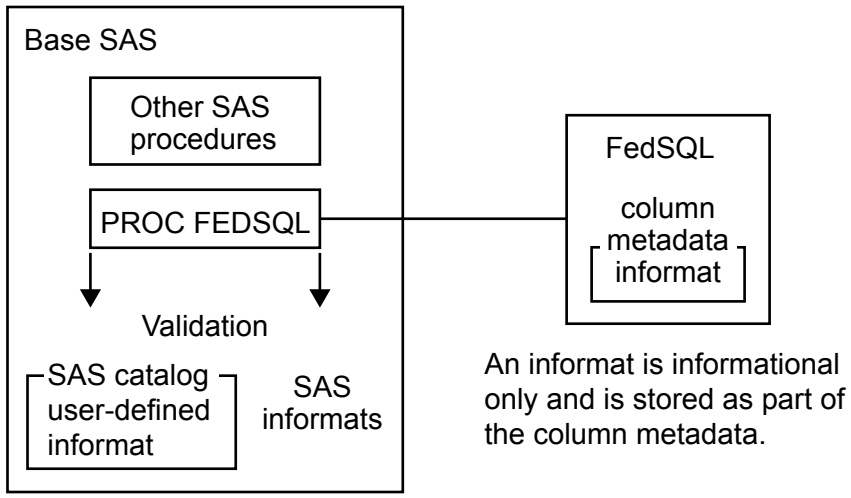
入力形式の動作と入力形式の完全なリストの詳細については、 *SAS 出力形式と入力形式: リファレンス*を参照してください。

FedSQL での入力形式の使用方法

入力形式は情報を提供するものであり、FedSQL 環境で実行時にデータを操作することはありません。クライアントアプリケーションは、テーブルがアプリケーションに返されるときに入力形式を使用して生データを変換する必要があります。

注: FedSQL 環境に保存されている入力形式が無効な場合、クライアントアプリケーションで無効な入力形式が使用されている場合にのみエラーが発生します。

図 6.1 入力形式の仕組みの図



ただし、FedSQL では、入力形式名を保存および取得できます。入力形式名は、列のメタデータ属性として入力形式を保存することによって、操作中に一時的に、または永続的に列に関連付けられます。その後、後続の操作のためにメタデータを取得できます。

FedSQL は、次のように SAS 入力形式をサポートします。

- SAS によって提供される入力形式とユーザー定義の入力形式の両方を列に関連付けることができます。SAS で独自の入力形式を作成する方法については、*Base SAS プロシジャガイド*の PROC FORMAT を参照してください。

  注: ユーザー定義の入力形式を作成してアクセスするには、SAS 入力形式の定義を保存する SAS カタログファイルにアクセスするために、Base SAS セッションが使用可能である必要があります。

- SAS データセット、SPD Engine データセット、および SPD Server テーブルのみが、列を含む入力形式の保存と取得をサポートします。
- 入力形式はすべてのデータ型に関連付けることができますが、すべてのデータ型は CHAR または DOUBLE のいずれかに変換されます。
- FedSQL CREATE TABLE の HAVING 句または FedSQL ALTER TABLE ステートメントの SET 句を使用して、SAS 入力形式を列に関連付けることができます。詳細については、"FedSQL で入力形式を指定する方法" (926 ページ)を参照してください。

SAS が提供する入力形式の詳細と完全なリストについては、*SAS 出力形式と入力形式: リファレンス*の入力形式のセクションを参照してください。

FedSQL で入力形式を指定する方法

FedSQL では、CREATE TABLE ステートメントの HAVING 句の属性として入力形式を指定します。たとえば、次のステートメントでは、列 X は IEEE8.2 出力形式および BITS5.2 入力形式で宣言されています。

```
create table a (x double having format ieee8.2 informat bits5.2 label 'foo');
```

FedSQL 入力形式の検証

入力形式は、実行時までデータソースによって 検証されたり、列に適用されたりすることはありません。

FedSQL を使用してテーブルを 作成する場合、検証は行われません。

列のメタデータが要求されると、検証なしで入力形式名が返されます。

FedSQL 入力形式の例

次の例では、テーブル Assets の列 Val の入力形式を指定する方法を示します。

```
proc fedsql;
  create table work.assets(
    prod char(10) having label 'Item' format $10.,
    val double having label 'Value' format dollar12. informat 12.
  );

  insert into work.assets values('Car', 11578);
  insert into work.assets values('Boat', 8575);
  insert into work.assets values('House', 320570);

  select * from work.assets;
quit;
```

7

FedSQL ステートメント

ステートメントの概要 **927**
カテゴリ別 FedSQL ステートメント **928**
ディクショナリ **929**
ALTER TABLE ステートメント 929
BEGIN ステートメント 935
COMMIT ステートメント 936
CREATE INDEX ステートメント 937
CREATE TABLE ステートメント 939
CREATE VIEW ステートメント 958
DELETE ステートメント 960
DESCRIBE TABLE ステートメント 961
DESCRIBE VIEW ステートメント 964
DROP INDEX ステートメント 965
DROP TABLE ステートメント 966
DROP VIEW ステートメント 968
EXECUTE ステートメント 970
INSERT ステートメント 972
ROLLBACK ステートメント 976
SELECT ステートメント 977
UPDATE ステートメント 1009

ステートメントの概要

FedSQL ステートメントは、キーワード、識別子、特殊文字、演算子を含めることができる一連のアイテムです。すべての FedSQL ステートメントはセミコロンで終わります。

ステートメントには、次の 5 つのカテゴリがあります。

データ定義
: テーブルとビューの作成、変更、および削除に使用されるステートメント。

データコントロール
: データベースとのトランザクションを制御するために使用されるステートメント。

データ操作
: データの追加、更新、および削除に使用されるステートメント。

データ取得
: データの取得、あるいはデータテーブルまたは SQL ビューに関する情報の取得に使用されるステートメント。

SQL パススルー機能
: DBMS に接続し、DBMS 固有の SQL ステートメントを DBMS に直接送信できるようにするステートメント。

一部のステートメントは、複数のカテゴリに含まれています。

カテゴリ別 FedSQL ステートメント

カテゴリ	言語要素	説明
FedSQL パススルー	EXECUTE ステートメント (p. 970)	FedSQL がサポートする DBMS に DBMS 固有のステートメントを送信します。
データコントロール	BEGIN ステートメント (p. 935)	複数のステートメントで構成されるトランザクションの先頭をマークします。
	COMMIT ステートメント (p. 936)	トランザクションの開始以降に実行された変更をデータベースの永続的な部分にします。
	ROLLBACK ステートメント (p. 976)	トランザクションの変更をトランザクションの先頭にロールバックします。
データ取得	DESCRIBE TABLE ステートメント (p. 961)	テーブルから SQL を取得し、結果セットを返します。
	DESCRIBE VIEW ステートメント (p. 964)	ビューから SQL を取得し、結果セットを返します。
	SELECT ステートメント (p. 977)	テーブルからデータの列と行を取得します。
データ操作	DELETE ステートメント (p. 960)	テーブルから行を削除します。
	INSERT ステートメント (p. 972)	指定されたテーブルに行を追加します。
	SELECT ステートメント (p. 977)	テーブルからデータの列と行を取得します。

カテゴリ	言語要素	説明
	UPDATE ステートメント (p. 1009)	テーブルの既存の行の列の値を変更します。
データ定義	ALTER TABLE ステートメント (p. 929)	テーブルの列を追加または削除し、列の定義を変更します。
	CREATE INDEX ステートメント (p. 937)	指定されたテーブルの列にインデックスを作成します。
	CREATE TABLE ステートメント (p. 939)	新しいテーブルを作成します。
	CREATE VIEW ステートメント (p. 958)	1 つ以上のテーブルまたは他のビューからデータの FedSQL ビューを作成します。
	DESCRIBE TABLE ステートメント (p. 961)	テーブルから SQL を取得し、結果セットを返します。
	DESCRIBE VIEW ステートメント (p. 964)	ビューから SQL を取得し、結果セットを返します。
	DROP INDEX ステートメント (p. 965)	指定されたインデックスをテーブルから削除します。
	DROP TABLE ステートメント (p. 966)	データベースからテーブルを削除します。
	DROP VIEW ステートメント (p. 968)	データベースからビューを削除します。
	SELECT ステートメント (p. 977)	テーブルからデータの列と行を取得します。

ディクショナリ

ALTER TABLE ステートメント

テーブルの列を追加または削除し、列の定義を変更します。

カテゴリ: データ定義

制限事項: このステートメントは、CAS で実行される FedSQL プログラムではサポートされていません。

制約構文は、データソースによって 定義されます。ここで提供される制約構文は、一般的な構文です。完全な制約構文については、データソースのドキュメントを参照してください。

データソース: SAS データセット、Amazon Redshift、DB2、Greenplum、Hive、Impala、JDBC、MDS、Microsoft SQL Server、MySQL、Netezza、ODBC、Oracle、PostgreSQL、SAP HANA、SAP IQ、SingleStore、Snowflake、Spark、Teradata、Vertica、Yellowbrick

注: 構文規則の中かっこは、構文のグループ化を示します。中かっこの前のエスケープ文字(\ )は、構文で中かっこが必要であることを示します。テーブルオプションは中かっこ({ })で囲む必要があります。

構文

ALTER TABLE *table* [\{OPTIONS *SAS-table-option=value* [, ...*SAS-table-option=value*]\ }]
ADD COLUMN<column-definition>[,...<column-definition>]
| ADD CONSTRAINT<table-constraint>
| ALTER[COLUMN]<column-definition>[,...<column-definition>]
| DROP COLUMN *column*[,...*column*] [FORCE]
| DROP CONSTRAINT*constraint*[,...*constraint*]
| RENAME TO*table*
| RENAME*column* TO*column*
;

<column-definition>::=

*column**data-type* [<column-constraint>] [SET DEFAULT*value* | DROP DEFAULT]

<column-constraint>::=
CONSTRAINT *constraint*

{CHECK (*search-condition*)
| PRIMARY KEY
| UNIQUE
| NOT NULL }

<table-constraint>::=
CONSTRAINT *constraint*

{ CHECK (*search-condition*)
| PRIMARY KEY (*column*[, ...*column*])
| UNIQUE (*column*[, ...*column*])
| <referential-constraint >}
[<constraint-check-time>]

<referential-constraint>::=

FOREIGN KEY (*referencing-column* [, ... *referencing-column*])
REFERENCES *referenced-table* (*referenced-column* [, ...*referenced-column*])
[<referential-trigger-action>]

<referential-trigger-action>::=

{[ON UPDATE<referential-action>[ON DELETE <referential-action>]

| ON DELETE<referential-action>[ON UPDATE <referential-action>]] }

<referential-action>::=

CASCADE | SET NULL | SET DEFAULT | RESTRICT | NO ACTION

<constraint-check-time>::=

{ DEFERRABLE[INITIALLY DEFERRED | INITIALLY IMMEDIATE]
| [INITIALLY DEFERRED | INITIAILLY IMMEDIATE] DEFERRABLE
| INITIALLY DEFERRED
| NOT DEFERRABLE[INITIALLY IMMEDIATE]
| [INITIALLY IMMEDIATE] NOT DEFERRABLE }

引数

table
変更するテーブルを指定します。*table* は、次のいずれかの形式で指定できます。

- *catalog*.*schema*.*table-name*
- *schema*.*table-name*
- *catalog*.*table-name*
- *table-name*

catalog
SQL カタログの ANSI SQL 規格の実装です。カタログは、論理的に関連するスキーマをグループ化するデータコンテナーオブジェクトです。これは、データ編成階層の第 1 レベル(最上位)のグループ化メカニズムです。スキーマとともに使用され、名前を修飾する方法を提供します。カタログは、SAS メタデータリポジトリ内のメタデータオブジェクトです。

schema
SQL スキーマの ANSI SQL 規格の実装です。スキーマは、テーブルやビューなどのファイルや、ストアドプロシジャなどのデータソースでサポートされるその他のオブジェクトをグループ化するデータコンテナーオブジェクトです。スキーマは、カタログとともに使用されて、名前を修飾する方法を提供するグループ化オブジェクトを提供します。

table-name
テーブルの名前です。

制限事項　別のトランザクションまたはステートメントで参照されている間、MDS テーブルを変更することはできません。要求が失敗した場合は、他のユーザーがテーブルを使用していないか、切断されていることを確認してください。

要件　テーブルの命名規則は、データソースに基づきます。複数のデータソースが関係する場合、テーブル名の最大長は、すべてのデータソースと FedSQL でサポートされる最大長によって決まります。

{OPTIONS *SAS-table-option=value* [,...*SAS-table-option=value*]}
テーブルに適用する 1 つ以上のテーブルオプションとそれぞれの値を指定します。

要件　OPTIONS 引数とすべてのテーブルオプションは 中かっこ({ })で囲む必要があります。

参照項目:　“FedSQL ステートメントテーブルオプション” (1011 ページ)

column
テーブル内の列の名前を指定します。

要件　テーブルの各列は一意である必要があります。

ADD COLUMN <column-definition> [,...<column-definition>]
テーブルに列を追加することを指定します。

データソース　Amazon Redshift、DB2、Greenplum、Hive、Impala、JDBC、MDS、Microsoft SQL Server、MySQL、Netezza、ODBC、Oracle、PostgreSQL、SAP HANA、SAP IQ、SingleStore、Snowflake、Spark、Teradata、Vertica、Yellowbrick

参照項目:　CREATE TABLE ステートメントの column (932 ページ)および “<column-definition>引数” (942 ページ)

例　“列の追加と削除” (933 ページ)

ADD CONSTRAINT <table-constraint>
1 つ以上の列に一貫性制約を追加することを指定します。

データソース　SAS データセット、Amazon Redshift、DB2、Greenplum、Microsoft SQL Server、MySQL、ODBC、Oracle、PostgreSQL、SAP HANA、SAP IQ、SingleStore、Snowflake、Teradata、Vertica、Yellowbrick

注　制約の検証と適用は、データソースによって 実行されます。一貫性制約を追加または削除する前に、データソースの要件をよく理解しておく必要があります。

参照項目:　CREATE TABLE ステートメントの“<column-constraint>および<table-constraint>引数” (952 ページ)

例　“制約の追加と削除” (934 ページ)

ALTER [COLUMN] <column-definition>[,...<column-definition>]
1 つ以上の列の定義を変更することを指定します。

データソース　Amazon Redshift、DB2、Greenplum、MDS、Microsoft SQL Server、MySQL、Netezza、ODBC、Oracle、PostgreSQL、SAP IQ、SingleStore、Snowflake、Teradata、Vertica、Yellowbrick

参照項目:　CREATE TABLE ステートメントの“<column-definition>引数” (942 ページ)

例　“列定義の変更” (934 ページ)

DROP COLUMN *column* , [...*column*] [FORCE]
指定された列をテーブルから削除することを指定します。FORCE を指定すると、エラー処理なしで列がテーブルから削除されます。FORCE キーワードは、

エラー処理を行わずに列を削除してもテーブルに 悪影響が及ばないことが確実な場合にのみ使用してください。

データソース Amazon Redshift、DB2、Greenplum、Hive、JDBC、MDS、Microsoft SQL Server、MySQL、Netezza、ODBC、Oracle、PostgreSQL、SAP HANA、SAP IQ、SingleStore、Snowflake、Teradata、Vertica、Yellowbrick

参照項目: “*column*” (932 ページ)

DROP CONSTRAINT *constraint* [, ...*constraint*]
一貫性制約の削除を指定します。

constraint
削除する制約の名前を指定します。

データソース SAS データセット、Amazon Redshift、DB2、Greenplum、Microsoft SQL Server、ODBC、Oracle、PostgreSQL、SAP HANA、SAP IQ、Snowflake、Teradata、Vertica、Yellowbrick

RENAME
指定されたテーブルまたは列の名前を新しい名前に変更します。新しい名前は TO キーワードの後に続きます。

データソース Amazon Redshift、DB2、Greenplum、MDS、Microsoft SQL Server、Netezza、ODBC、Oracle、PostgreSQL、SAP HANA、SAP IQ、Snowflake、Teradata、Vertica、Yellowbrick

<column-definition>引数

CREATE TABLE ステートメントの“<column-definition>引数” (942 ページ)を参照してください。

<column-constraint>および<table-constraint>引数

CREATE TABLE ステートメントの“<column-constraint>および<table-constraint>引数” (952 ページ)を参照してください。

詳細

列の追加と削除

テーブルに列を追加するには、ALTER TABLE ステートメントを ADD COLUMN 句とともに使用します。ADD COLUMN 句を使用して、列に名前を付け、列のデータ型、デフォルト値、列のラベルなどの列属性を定義します。

```
alter table personal_info add column em_num char(16);
```

列を削除するには、列名のみを指定する必要があります。

```
alter table personal_info drop em_num;
```

制約の追加と削除

次のように、一貫性制約をサポートするすべてのデータソースで、ADD COLUMN 句を使用して追加された列に制約を定義できます。

```
alter table mytest add y char(5) constraint c2 unique;
```

例外は、SAP HANA と Vertica です。これらのデータソースの場合、最初に列を追加してから、ADD CONSTRAINT で制約を追加する必要があります。さらに、Vertica では、ALTER TABLE ステートメントで UNIQUE 制約を追加するときに、制約名を指定する必要があります。

ADD CONSTRAINT 句を使用すると、テーブルの制約を追加したり、1 つ以上の既存の列の制約を追加または変更したりできます。次のコードは、ALTER TABLE を使用して主キー制約をテーブルに追加します。

```
alter table customers add constraint pkey primary key(custid);
```

次のコードは、CHECK 制約を既存の列に追加します。

```
alter table mytest add constraint chk check (col1 < 5);
```

DROP CONSTRAINT 句を使用して制約を削除します。次のコードは、SAS データセットから制約を削除する例です。このコードは、テーブルの作成時に定義された主キー制約を削除します。テーブルの作成時に定義されると、SAS データセットの主キーがデフォルト名_pk0001_で作成されます。

```
alter table customers drop constraint _pk0001_;
```

制約を追加または変更するときは、データソースがサポートする制約構文を知っている必要があります。制約の検証は、データソースによって実行されます。

列定義の変更

列定義を変更するには、ALTER 句を指定した ALTER TABLE ステートメントを使用します。

```
alter table personal_info alter country set default 'UK';
```

この例では、列からデフォルト値を削除します。

```
alter table personal_info alter country drop default;
```

この例では、列の名前を変更します。

```
alter table sales rename column y to customers;
```

比較

ALTER TABLE ステートメントを使用して、作成後にテーブルを変更します。テーブルを作成するには、CREATE TABLE ステートメントを使用します。

関連項目

- “保存出力形式の保存、変更、削除、使用方法” (81 ページ)

ステートメント:

- “CREATE TABLE ステートメント” (939 ページ)
- “DROP INDEX ステートメント” (965 ページ)

BEGIN ステートメント

複数のステートメントで構成されるトランザクションの先頭をマークします。

カテゴリ: データコントロール

制限事項: このステートメントは、CAS で実行される FedSQL プログラムではサポートされていません。

BEGIN ステートメントは、自動コミット機能がオフの場合にのみ効果があります。

データソース: Greenplum、MDS、Microsoft SQL Server、MySQL、ODBC、Oracle、SAP HANA、SAP IQ、SingleStore、Snowflake、Teradata、Vertica

構文

BEGIN [TRANSACTION];

詳細

BEGIN ステートメントは、データが論理的および物理的に一貫しているポイントをマークします。トランザクション中にエラーが発生した場合、ユーザーは、データに対するすべての変更を、一貫性のある最後の既知の状態にロールバックできます。

FedSQL 処理を使用する場合、このステートメントは実際には必要ありません。デフォルトでは、自動コミット機能がオンになっています。つまり、各要求がサブミットされた直後にすべての更新がコミットされ、ロールバックは不可能です。トランザクションは、更新ごとに効果的に開始されます。BEGIN ステートメントは、複数のステートメントで構成されるトランザクションの開始をマークできるようにするために提供されています。

自動コミット機能をオフにする方法については、サーバー管理のドキュメントを参照してください。たとえば、FedSQL ドライバーへの適切な接続オプションについては、*SAS Federation Server: 管理者ガイド*を参照してください。FEDSQL プロシジャについては、*Base SAS プロシジャガイド*を参照してください。

関連項目

ステートメント:

- “COMMIT ステートメント” (936 ページ)

■ “ROLLBACK ステートメント” (976 ページ)

COMMIT ステートメント

トランザクションの開始以降に実行された変更をデータベースの永続的な部分にします。

カテゴリ: データコントロール

制限事項: このステートメントは、CAS で実行される FedSQL プログラムではサポートされていません。
COMMIT ステートメントは、自動コミット機能がオフの場合にのみ効果があります。

データソース: Greenplum、MDS、Microsoft SQL Server、MySQL、ODBC、Oracle、SAP HANA、SAP IQ、SingleStore、Snowflake、Teradata、Vertica

構文

COMMIT [TRANSACTION];

詳細

プログラムがトランザクション内のすべてのステートメントを完了したら、COMMIT または ROLLBACK を使用してトランザクションを明示的に終了する必要があります。COMMIT ステートメントを使用して、データベースへの変更を永続的にします。

COMMIT ステートメントの実行後は、変更をデータベースにロールバックすることはできません。

注: COMMIT ステートメントは、自動コミット機能がオフの場合にのみ効果があります。ほとんどのデータソースでは、自動コミット機能がデフォルトでオンになっています。自動コミット機能をオフにする方法については、サーバー管理のドキュメントを参照してください。たとえば、FedSQL ドライバーへの適切な接続オプションについては、*SAS Federation Server: 管理者ガイド*を参照してください。FEDSQL プロシジャについては、*Base SAS プロシジャガイド*を参照してください。

比較

ROLLBACK ステートメントを使用すると、トランザクションによって行われたコミットされていないすべての 変更が、トランザクションの開始点または BEGIN ステートメントによってマークされたポイントまでロールバックされます。COMMIT ス

テートメントは、トランザクションの開始以降に実行されたすべてのデータ変更を取得し、それらをデータベースの永続的な部分にします。

関連項目

ステートメント:

- “BEGIN ステートメント” (935 ページ)
- “ROLLBACK ステートメント” (976 ページ)

CREATE INDEX ステートメント

指定されたテーブルの列にインデックスを作成します。

カテゴリ: データ定義

制限事項: このステートメントは、CAS で実行される FedSQL プログラムではサポートされていません。

データソース: SAS データセット、SPD Engine データセット、SPD Server テーブル、Amazon Redshift、DB2、Greenplum、JDBC、MDS、Microsoft SQL Server、MySQL、ODBC、Oracle、PostgreSQL、SAP HANA、SAP IQ、SingleStore、Teradata

構文

CREATE [UNIQUE] **INDEX** *index* ON *table*
(*column* [ASCENDING | DESCENDING][, ...*column*]);

引数

UNIQUE
テーブルに一意のインデックスを作成します。

index
インデックスの名前を指定します。

制限事項 *SAS データセットの場合*: 1 つの列のみにインデックスを作成する場合、*index* は *column* と同じである必要があります。複数の列にインデックスを作成する場合、*index* はテーブル内のどの列とも同じにすることはできません。

table
インデックス付けする列を含むテーブルの名前を指定します。

column
インデックスが適用される列の名前を指定します。複合インデックスを作成するには、2 つ以上の列名を指定します。

ヒント　2 つ以上の列を 1 つの単位として検索する場合、または特定の列のみを含むクエリがある場合は、複合インデックスを使用します。

ASCENDING
行は、最小値から最大値の順に並べ替えられます。これがデフォルト値です。

別名　ASC

注　このオプションは、SAS データセット、SPD Engine データセット、および SPD Server テーブルには無効です。

DESCENDING
行は、最大値から最小値の順に並べ替えられます。

別名　DESC

注　このオプションは、SAS データセット、SPD Engine データセット、および SPD Server テーブルには無効です。

詳細

テーブルインデックスの概要

index には、テーブルの列の値と、そのテーブル内の行へのインデックス値によるアクセスを可能にする指示のシステムの両方が格納されます。列または列のセットにインデックスを定義すると、特定の状況下で、SAS がテーブル内の行をより迅速かつ効率的に検索できるようになります。インデックスを使用すると、SAS は次のクラスのクエリをより効率的に実行できます。

- インデックス付けされた列との比較。
- 内部サブクエリの列がインデックス付けされている IN サブクエリ。
- 相関サブクエリ。相関参照と比較される列にインデックスが付けられます。
- 結合クエリ。結合式は等値比較であり、結合式内のすべての列は、結合されるテーブルの 1 つでインデックス付けされます。

SAS は、変更が FedSQL から発生したか、他のソースから発生したかに関係なく、テーブルに対するすべての変更のインデックスを維持します。したがって、列の定義を変更したり値を更新したりしても、同じインデックスが引き続き定義されます。ただし、テーブル内のインデックス付きの列が削除されると、その列のインデックスも削除されます。

単純インデックスまたは複合インデックスを作成できます。*単純インデックス*は、テーブルの 1 つの列に対して作成されます。単純インデックスには、その列と同じ名前が必要です。*複合インデックス*は、2 つ以上の列に対して定義される 1 つのインデックス名です。列は任意の順序で指定でき、異なるデータ型を持つことができます。複合インデックス名は、テーブル内のどの列の名前とも一致できません。複合インデックスを削除すると、その複合インデックスで指定されたすべての列のインデックスが削除されます。

テーブルにインデックスを作成できるのは、テーブルの所有者だけです。

インデックスを作成するときは、データソースがサポートする構文を知っている必要があります。インデックスの検証は、データソースによって実行されます。FedSQL は DBCREATE_INDEX_OPTS=テーブルオプションを提供し、ODBC データベースのデータソース固有のインデックスパラメーターを指定できるようにします。

UNIQUE キーワード

UNIQUE キーワードを使用すると、SAS は、複数の行が同じインデックス値を持つ原因となるテーブルへの変更を拒否します。一意のインデックスは、1 つの列または列の複合グループ内のデータが、テーブル内のすべての行で一意のままであることを保証します。各行に一意のインデックス値がある場合、null または SAS 欠損値を含む列に対して一意のインデックスを定義できます。

関連項目

ステートメント:

- “DROP INDEX ステートメント” (965 ページ)

テーブルオプション:

- “DBCREATE_INDEX_OPTS= テーブルオプション” (1085 ページ)

CREATE TABLE ステートメント

新しいテーブルを作成します。

カテゴリ: データ定義

制限事項: 既存のテーブルを上書きすることはできません。DROP TABLE ステートメントを使用してテーブルを削除し、CREATE TABLE ステートメントを使用してテーブルを再作成する必要があります。

CAS で実行される FedSQL プログラムでは、CREATE TABLE ステートメントの完全な機能はサポートされていません。CAS 情報については、“CREATE TABLE ステートメント” (*SAS Viya プラットフォーム: SAS Cloud Analytic Services の FedSQL プログラミング*)を参照してください。.

制約構文は、データソースによって定義されます。ここで提供される制約構文は、一般的な構文です。完全な制約構文については、データソースのドキュメントを参照してください。

データソース: SAS データセット、SPD Engine データセット、SPD Server テーブル、Amazon Redshift、DB2、Google BigQuery、Greenplum、Hive、Impala、JDBC、MDS、Microsoft SQL Server、MongoDB、MySQL、Netezza、ODBC、Oracle、PostgreSQL、Salesforce、SAP HANA、SAP IQ、SingleStore、Snowflake、Spark、Teradata、Vertica、Yellowbrick

注:

構文規則の中かっこは、構文のグループ化を示します。中かっこの前のエスケープ文字(\ )は、構文で中かっこが必要であることを示します。テーブルオプションは中かっこ({ })で囲む必要があります。

MongoDB テーブルはリレーショナルテーブルではなく、リレーショナルテーブルとは大きな違いがあります。MongoDB でテーブルを作成する前に、*SAS/ACCESS for Nonrelational Databases: リファレンス*で MongoDB のドキュメントを参照してください。

テーブルを作成するとき、Salesforce は Name 列を自動的に生成します。Name 列を指定すると、その列は無視されます。Name という名前の列を追加する必要がある場合は、Salesforce 接続で USE_NATIVE_NAMES=オプションを指定し、列名を Name_c のように修飾します。Salesforce テーブルの列は、定義された順序で作成されない場合があります。結果の順序は、その後のテーブルでの FedSQL ステートメントの実行でも保持されることに注意してください。

構文

CREATE TABLE {*table* | _NULL_ }
[\{OPTIONS *SAS-table-option=value* [*...SAS-table-option=value*] \}]
(<column-definition>[, ...<column-definition> | <table-constraint>])
| AS*query-expression*
;

<column-definition>::=

column data-type [<column-constraint>] [DEFAULT *value*]
[HAVING [FORMAT *format*][INFORMAT *informat*][LABEL '*label*']]

<column-constraint>::=

CONSTRAINT *constraint*
{CHECK (*search-condition*)
| PRIMARY KEY
| UNIQUE
| NOT NULL}

<table-constraint>::=

CONSTRAINT *constraint*
{CHECK (*search-condition*)
| PRIMARY KEY (*column*[, ...*column*])
| UNIQUE (*column*[, ...*column*])
| <referential-constraint>}
[<constraint-check-time>]

<referential-constraint>::=

FOREIGN KEY (*referencing-column* [, ...*referencing-column*])
REFERENCES *referenced-table* (*referenced-column* [, ...*referenced-column*])
[<referential-trigger-action>]

<referential-trigger-action>::=

{ON UPDATE<referential-action>[ON DELETE <referential-action>]
| ON DELETE<referential-action>[ON UPDATE <referential-action>]}

<referential-action>::=

CASCADE | SET NULL | SET DEFAULT | RESTRICT | NO ACTION

<constraint-check-time>::=
{DEFERRABLE[INITIALLY DEFERRED | INITIALLY IMMEDIATE]
| [INITIALLY DEFERRED | INITIALLY IMMEDIATE] DEFERRABLE
| INITIALLY DEFERRED
| NOT DEFERRABLE[INITIALLY IMMEDIATE]
| [INITIALLY IMMEDIATE] NOT DEFERRABLE}

引数

table
テーブルの名前を指定します。*table* は、次のいずれかの形式をとります。

- *catalog*.*schema*.*table-name*
- *schema*.*table-name*
- *catalog*.*table-name*
- *table-name*

catalog
SQL カタログの ANSI SQL 規格の実装です。カタログは、論理的に関連するスキーマをグループ化するデータコンテナーオブジェクトです。カタログは、データ編成階層の第 1 レベル(最上位)のグループ化メカニズムです。スキーマとともに使用され、名前を修飾する方法を提供します。

schema
SQL スキーマの ANSI SQL 規格の実装です。スキーマは、テーブルやビューなどのファイルや、ストアドプロシジャなどのデータソースでサポートされるその他のオブジェクトをグループ化するデータコンテナーオブジェクトです。スキーマは、カタログとともに使用されて、名前を修飾する方法を提供するグループ化オブジェクトを提供します。

table-name
テーブルの名前です。

制限事項
Hive のテーブル名はアンダースコア(_)で始めないでください。HDFS では、_で始まる名前は隠しシステムファイルです。Hive テーブルのテーブル名には、. (ピリオド)、: (コロン)、/(スラッシュ)、\ (バックスラッシュ)を含めないでください。

要件
テーブルの命名規則は、データソースに基づきます。複数のデータソースが関係する場合、テーブル名の最大長は、すべてのデータソースと FedSQL でサポートされる最大長によって決まります。たとえば、データソースが最大 32 文字の SAS データセットと、最大 64 文字の MySQL である場合、テーブル名の最大長は 32 文字になります。詳細については、データソースのドキュメントを参照してください。

操作
スキーマをサポートしていないデータソースの場合は、*catalog*.*table-name* の形式を使用します。

注
3 レベルのテーブル名は、Hive 0.14 以降でサポートされています。以前の Hive リリースでテーブルを 作成またはアクセスする場合は、2 レベル形式の *catalog*.*table-name* を使用します。

NULL
AS *query-expression* 句を使用してテーブルを作成するパフォーマンスをテストすることを指定します。FedSQL は、テーブルが保存されるかのようにテーブルを内部的に作成し、通常の進行状況メッセージを書き込みます。テーブルの作成は成功したようです。クエリ式が完了すると、テーブルは破棄されます。

操作 SELECT *select-list* FROM _NULL_ステートメントはエラーを返します。

{OPTIONS *SAS-table-option=value* [...*SAS-table-option=value*]}
テーブルに適用する 1 つ以上のテーブルオプションとそれぞれの値を指定します。

要件 OPTIONS 引数とすべてのテーブルオプションは 中かっこ({ })で囲む必要があります。

参照項目: “FedSQL ステートメントテーブルオプション” (1011 ページ)

AS *query-expression*
クエリ式を使用して既存のテーブルから行を選択することにより、既存のテーブルから新しいテーブルを作成することを指定します。出力形式やラベルなどの列属性は、既存のテーブルから新しいテーブルにコピーされます。

query-expression
新しいテーブルの作成に使用する既存のテーブルから情報を取得する SELECT ステートメントを指定します。

要件 SELECT ステートメントで使用される列の数は、テーブル内の列の数と等しい必要があります。

参照項目: “クエリ式からのテーブルの作成と入力” (956 ページ)

“クエリ式とサブクエリ” (46 ページ)

“SELECT ステートメント” (977 ページ)

データソース SAS データセット、SPD Engine データセット、SPD Server テーブル、DB2、Greenplum、Hive、JDBC、MDS、Microsoft SQL Server、MySQL、Netezza、ODBC、Oracle、SAP HANA、SAP IQ、SingleStore、Snowflake、Spark、Teradata、Vertica

<column-definition>引数

column
列の名前を指定します。列名は、次のいずれかの形式を使用した、ローカルまたはスキーマ修飾名のいずれかです。

- *schema*.*column-name*
- *column-name*

schema
SQL スキーマの ANSI SQL 規格の実装です。スキーマは、テーブルやビューなどのファイルや、ストアドプロシジャなどのデータソースでサポートされるその他のオブジェクトをグループ化するデータコンテナーオブジェクトで

す。スキーマは、カタログとともに使用されて、名前を修飾する方法を提供するグループ化オブジェクトを提供します。

column-name
列の名前です。

制限事項 SAS データセット、SPD Engine データセット、および SPD Server テーブルの場合、列名は 32 文字を超えることはできません。

Hive では、列名に. (ピリオド)または: (コロン)を使用することはできません。

Hive では、列名に. (ピリオド)または: (コロン)を使用することはできません。

要件 複数のデータソースが関係する場合、列名の最大長は、すべてのデータソースと FedSQL でサポートされる最大長によって決まります。たとえば、データソースが最大 32 文字の SAS データセットと、最大 64 文字の DBMS である場合、列名の最大長は 32 文字になります。テーブルの各列は一意である必要があります。

参照項目: 列名の要件については、データソースのドキュメントを参照してください。

data-type
列が格納できるデータの型を指定します。列定義が SAS データセット、SPD Engine データセット、または SPD Server テーブル用であり、データ型が CHAR または DOUBLE 以外の場合、FedSQL は文字データ型を CHAR に、数値データ型を DOUBLE に変換します。FedSQL は、次のテーブルデータ型をサポートします。

BIGINT
大きな符号付きの正確な整数を格納します。

範囲 -9,223,372,036,854,775,808-9,233,372,036,854,775,807

データソース Amazon Redshift、DB2、Greenplum、Hive、Impala、JDBC、MDS、Microsoft SQL Server、MongoDB、MySQL、Netezza、ODBC、Oracle、PostgreSQL、SAP HANA、SAP IQ、SingleStore、Snowflake、Spark、Teradata、Vertica、Yellowbrick

ストレージサイズ 8 バイト

BINARY(*n*)
可変長バイナリデータを格納します。

n
バイナリデータの格納に使用できる最大バイト数を指定します。バイナリデータを格納するために使用されるバイト数は、バイナリデータを表すために必要なバイト数(最大 *n* バイト)です。

データソース　DB2、Hive、JDBC、MDS、Microsoft SQL Server、MySQL、ODBC、SAP HANA、SAP IQ、SingleStore、Snowflake、Spark、Vertica

ストレージサイズ　最大 *n* バイト

CHAR(*n*) [CHARACTER SET "*character-set-identifier*"]
固定長の文字列を格納します。

n
文字列を格納するために使用されるマルチバイト文字の数。格納されるバイト数は、*n* 文字までの実際のマルチバイト文字数です。文字列が *n* 文字未満の場合、値の右側がスペースで埋められます。

CHARACTER SET "*character-set-identifier*"
CHAR データ型の文字セットエンコーディング情報を指定します。

制限事項　SAS データセット、SPD Engine データセット、SPD Server テーブル、Hive、Impala、JDBC、MongoDB、Salesforce、Spark は、CHARACTER SET "*character-set-identifier*"の使用をサポートしていません。Hive、Impala、MongoDB、Salesforce、Spark の文字列は UTF-8 として格納されます。SAS データセットの作成時に文字セット識別子を指定すると、望ましくない結果が生じます。“ENCODING= テーブルオプション” (1090 ページ)を使用して、SAS データセットのエンコーディング値を指定できます。

別名　CHARACTER(*n*)

要件　*n* を指定する必要があります。

Salesforce では、*n* は 255 文字以下にする必要があります。255 より大きい値を指定すると、列は VARCHAR(*n*)として作成されます。

データソース　SAS データセット、SPD Engine データセット、SPD Server テーブル、Amazon Redshift、DB2、Greenplum、Hive、Impala、JDBC、MDS、Microsoft SQL Server、MongoDB、MySQL、Netezza、ODBC、Oracle、PostgreSQL、Salesforce、SAP HANA、SAP IQ、SingleStore、Snowflake、Spark、Teradata、Vertica、Yellowbrick

ストレージサイズ　*n* 文字

DATE
yyyy-mm-dd の形式で日付値を格納します。

日付要素	説明	有効な値
yyyy	4 桁の年	0001-9999
mm	2 桁の月	01-12

日付要素	説明	有効な値
dd	2 桁の日	00-31

要件 入力値は、DATE 定数として指定する必要があります。詳細については、“FedSQL の日付、時間、および日時の定数” (60 ページ)を参照してください。

データソース SAS データセット、SPD Engine データセット、SPD Server テーブル、Amazon Redshift、DB2、Greenplum、Hive、Impala、JDBC、MDS、Microsoft SQL Server、MySQL、Netezza、ODBC、Oracle、PostgreSQL、SAP HANA、SAP IQ、Salesforce、SingleStore、Snowflake、Spark、Teradata、Vertica、Yellowbrick

ストレージサイズ 8 バイト

DECIMAL
符号付き固定小数点 10 進数を格納します。

別名 NUMERIC

データソース Amazon Redshift、DB2、Greenplum、Hive、JDBC、MDS、Microsoft SQL Server、MySQL、Netezza、ODBC、Oracle、PostgreSQL、Salesforce、SAP HANA、SAP IQ、SingleStore、Snowflake、Spark、Teradata、Vertica、Yellowbrick

ストレージサイズ 8 バイト

DOUBLE
符号付き近似浮動小数点数を格納します。

別名 DOUBLE PRECISION

データソース SAS データセット、SPD Engine データセット、SPD Server テーブル、Amazon Redshift、DB2、Greenplum、Hive、Impala、JDBC、MDS、Microsoft SQL Server、MongoDB、MySQL、Netezza、ODBC、Oracle、PostgreSQL、SAP HANA、SAP IQ、Salesforce、SingleStore、Snowflake、Teradata、Vertica、Yellowbrick

ストレージサイズ 8 バイト

FLOAT(*p*)
符号付き近似の単精度または倍精度の浮動小数点数が保存されます。ユーザー指定の精度によって、データ型が単精度数または倍精度数のどちらを保存するかが決まります。指定された精度が 25 以上の場合、値は倍精度数(DOUBLE)として保存されます。指定された精度が 25 未満の場合、値は単精度数(REAL)として保存されます。

p
浮動小数点数の最大桁数を指定します。

データソース　SAS データセット、SPD Engine データセット、SPD Server テーブル、Amazon Redshift、DB2、Greenplum、Hive、Impala、Microsoft SQL Server、MySQL、ODBC、Oracle、PostgreSQL、SAP HANA、SAP IQ、SingleStore、Snowflake、Spark、Teradata、Vertica、Yellowbrick

ストレージサイズ　4 バイト

ヒント　*p* が指定されていない場合、DOUBLE が使用されます。

INTEGER
正確な整数を格納します。

範囲　-2,147,483,648-2,147,483,647

データソース　Amazon Redshift、DB2、Greenplum、Hive、Impala、JDBC、MDS、Microsoft SQL Server、MongoDB、MySQL、Netezza、ODBC、Oracle、PostgreSQL、SAP HANA、SAP IQ、Salesforce、SingleStore、Snowflake、Spark、Teradata、Vertica、Yellowbrick

ストレージサイズ　4 バイト

NCHAR(*n*)
Unicode 各国語文字セットを使用して、固定長の文字列を格納します。

n
文字列を格納するために使用されるマルチバイト文字の最大数を指定します。文字列が *n* 文字未満の場合、値の右側がスペースで 埋められます。

要件　*n* を指定する必要があります。

データソース　Amazon Redshift、JDBC、MDS、Microsoft SQL Server、Netezza、ODBC、Oracle、Salesforce、SAP HANA

ストレージサイズ　最大 *n* 個のマルチバイト文字。オペレーティングシステムに応じて、Unicode 文字は 1 文字あたり 2 バイトまたは 4 バイトを使用し、すべての 国際文字をサポートします。

NUMERIC
符号付き固定小数点 10 進数を格納します。

別名　DECIMAL

データソース　Amazon Redshift、DB2、Greenplum、Hive、JDBC、MDS、Microsoft SQL Server、MySQL、Netezza、ODBC、Oracle、PostgreSQL、Salesforce、SAP IQ、SingleStore、Snowflake、Spark、Teradata、Vertica、Yellowbrick

ストレージサイズ　8 バイト

NVARCHAR(*n*)
Unicode 各国語文字セットを使用して、可変長マルチバイト文字列を格納します。

n
文字列の格納に使用できるマルチバイト文字の最大数を指定します。バイト数は、指定されたマルチバイト文字の実際の数であり、最大 *n* 文字です。

要件	*n* を指定する必要があります。
データソース	Amazon Redshift、DB2、JDBC、MDS、Microsoft SQL Server、ODBC、Oracle、Netezza、Salesforce、SAP HANA、Vertica
ストレージサイズ	マルチバイト文字の数。プラットフォームに応じて、Unicode 文字は 1 文字あたり 2 バイトまたは 4 バイトを使用し、すべての国際文字をサポートできます。

REAL
符号付き近似単精度浮動小数点数が保存されます。

データソース	SAS データセット、SPD Engine データセット、SPD Server テーブル、Amazon Redshift、DB2、Greenplum、Hive、Impala、JDBC、Microsoft SQL Server、MySQL、Netezza、ODBC、Oracle、PostgreSQL、Salesforce、SAP HANA、SAP IQ、Snowflake、Spark、Vertica、Yellowbrick
ストレージサイズ	4 バイト

SMALLINT
小さな符号付きの正確な整数を格納します。

範囲	-32,768-32,767
データソース	SAS データセット、SPD Engine データセット、SPD Server テーブル、Amazon Redshift、DB2、Greenplum、Hive、Impala、JDBC、Microsoft SQL Server、MySQL、Netezza、ODBC、Oracle、PostgreSQL、Salesforce、SAP HANA、SAP IQ、SingleStore、Snowflake、Spark、Teradata、Vertica、Yellowbrick
ストレージサイズ	2 バイト

TIME(*p*)
秒の精度で時間値を格納します。

時間値の形式は、*hh*:*mm*:*ss*[.*nnnnnn*]です。

時間要素	説明	有効な値
hh	2 桁の時間	00-23
mm	2 桁の分	00-59

時間要素	説明	有効な値
ss	2 桁の秒	00-61
nnnnnn	秒の小数部を示す最大 6 桁	0-999999

p 秒の小数部の精度に使用する 0-6 桁を指定します。

要件 入力値は、TIME 定数として指定する必要があります。詳細については、“FedSQL の日付、時間、および日時の定数” (60 ページ)を参照してください。

データソース SAS データセット、SPD Engine データセット、SPD Server テーブル、Amazon Redshift、DB2、Greenplum、Hive、Impala、JDBC、Microsoft SQL Server、MDS、MySQL、Netezza、ODBC、Oracle、PostgreSQL、SAP HANA、SAP IQ、Salesforce、SingleStore、Snowflake、Spark、Teradata、Vertica、Yellowbrick

ストレージサイズ 8 バイト

TIMESTAMP(*p*)
日付と時間の両方の値を秒の精度で格納します。

日付と時間の値の形式は、*yyyy-mm-dd hh*:*mm*:*ss*[.*nnnnnn*]です。

時間要素	説明	有効な値
日付コンポーネント		
yyyy	4 桁の年	0001-9999
mm	2 桁の月	01-12
dd	2 桁の日	01-31
時間コンポーネント		
hh	2 桁の時間	00-23
mm	2 桁の分	00-59
ss	2 桁の秒	00-61
nnnnnn	秒の小数部を示す最大 6 桁	0-999999

p 秒の小数部の精度に使用する 0-6 桁を指定します。

別名 DATETIME

要件 入力値は、TIMESTAMP 定数として指定する必要があります。詳細については、“FedSQL の日付、時間、および日時の定数” (60 ページ)を参照してください。

データソース SAS データセット、SPD Engine データセット、SPD Server テーブル、Amazon Redshift、DB2、Greenplum、Hive、Impala、JDBC、MDS、Microsoft SQL Server、MongoDB、MySQL、Netezza、ODBC、Oracle、PostgreSQL、SAP HANA、SAP IQ、Salesforce、SingleStore、Snowflake、Spark、Teradata、Vertica、Yellowbrick

ストレージサイズ 8 バイト

TINYINT
非常に小さな符号付きの正確な整数を格納します。

範囲 -128-127

データソース Hive、Impala、JDBC、Microsoft SQL Server、MySQL、Netezza、ODBC、Oracle、Salesforce、SAP HANA、SAP IQ、SingleStore、Spark、Teradata、Vertica

ストレージサイズ 1 バイト

VARBINARY(*n*)
可変長バイナリデータを格納します。

n
バイナリデータの格納に使用できる最大バイト数を指定します。バイナリデータを格納するために使用されるバイト数は、バイナリデータを表すために必要なバイト数(最大 *n* バイト)です。

範囲

データソース DB2、Hive、JDBC、MDS、Microsoft SQL Server、ODBC、Oracle、SAP HANA、SAP IQ、Snowflake、Spark、Teradata、Vertica

ストレージサイズ 最大 *n* バイト

VARCHAR(*n*) [CHARACTER SET "*character-set-identifier*"]
可変長の文字列を格納します。

n
文字列を格納するために使用されるマルチバイト文字の数を指定します。格納されるバイト数は、*n* 文字までの実際のマルチバイト文字数です。

[CHARACTER SET "*character-set-identifier*"]
VARCHAR データ型の文字セットエンコーディング情報を指定します。

デフォルト　デフォルトのエンコーディングは、オペレーティングシステムとロケールによって異なります。

制限事項　Hive Spark、CHARACTER SET "*character-set-identifier*"の使用をサポートしていません。VARCHAR 列は UTF-8 として格納されます。

Hive、JDBC、MongoDB、Salesforce、Spark は、CHARACTER SET "*character-set-identifier*"の使用をサポートしていません。Hive、JDBC、MongoDB、Salesforce、Spark の VARCHAR 列は UTF-8 として格納されます。VARCHAR 列の Hive サポートの詳細については、表 A2.8 (1211 ページ)を参照してください。

データソース　Amazon Redshift、DB2、Greenplum、Hive、Impala、JDBC、MDS、Microsoft SQL Server、MongoDB、MySQL、Netezza、ODBC、Oracle、PostgreSQL、SAP HANA、SAP IQ、Salesforce、SingleStore、Snowflake、Spark、Teradata、Vertica、Yellowbrick

ストレージサイズ　最大 *n* バイト

<column-constraint>
列に一貫性制約を設定することを指定します。

データソース　SAS データセット、DB2、MDS、Microsoft SQL Server、MongoDB、MySQL、ODBC、Oracle、Salesforce、SAP HANA、SAP IQ、SingleStore、Snowflake、Teradata、Vertica

参照項目:　“<column-constraint>および<table-constraint>引数” (952 ページ)

“一貫性制約” (957 ページ)

DEFAULT *value*
テーブルに追加される行ごとに、その列に他の値が入力されていない場合に想定される列の値を指定します。

value
デフォルト値を指定します。

要件　*value* は、指定されたデータ型の範囲内にある必要があります。

データソース　DB2、Microsoft SQL Server、MySQL、ODBC、Oracle、SAP HANA、SingleStore、Teradata

HAVING [FORMAT*format*] [INFORMAT*informat*] [LABEL '*label*']
出力形式、入力形式、またはラベルを列に関連付けるために使用される句を指定します。

FORMAT *format*
列のメタデータとして格納される SAS データ出力形式を指定します。出力形式は、実行時まで列データに適用されません。

format
有効な SAS データ出力形式を指定します。出力形式が FedSQL または DS2 出力形式以外の出力形式である場合は、Base SAS に対して有効な出力形式である必要があります。

参照項目: “FedSQL 出力形式” (77 ページ)

“DS2 出力形式” (*SAS DS2 言語リファレンス*)

“出力形式のディクショナリ” (*SAS 出力形式と入力形式: リファレンス*)

制限事項: PROC FORMAT によって作成された出力形式は、列のメタデータとして格納できません。

注: テーブルを作成するときに、SAS データ出力形式を列に関連付けることができます。出力形式は、PUT 関数などでの実行時まで検証されず、列に適用されません。出力形式が Base SAS 環境の列に適用される場合、任意の出力形式を列に格納できます。出力形式が FedSQL 環境で列に適用される場合、その出力形式は有効な FedSQL または DS2 出力形式である必要があります。

INFORMAT *informat*
列のメタデータとして格納される SAS データ入力形式を指定します。入力形式は列データに適用されません。入力形式の検証は行われません。これは情報提供のみを目的としています。

informat
SAS データ入力形式を指定します。

参照項目: 6 章, “FedSQL 入力形式” (923 ページ)

“入力形式のディクショナリ” (*SAS 出力形式と入力形式: リファレンス*)

LABEL '*text-string*'
クエリ式の結果で列名のかわりに表示される代替の列見出しとして使用するテキスト文字列を指定します。

範囲: 1 - 255 文字

制限事項: ラベルは CREATE TABLE ステートメントでのみ作成でき、クエリ式操作では使用できません。

要件: *text-string* は、二重引用符または単一引用符で囲む必要があります。*text-string* に単一引用符が含まれる場合は、*text-string* を二重引用符で囲みます。

データソース: SAS データセット、SPD Engine データセット、SPD Server テーブル

<column-constraint>および<table-constraint>引数

CONSTRAINT
列制約またはテーブル制約を開始します。

データソース SAS データセット、DB2、MDS、Microsoft SQL Server、MongoDB、MySQL、ODBC、Oracle、SAP HANA、SingleStore、Snowflake、Teradata、Vertica

constraint
制約を識別する名前を指定します。

参照項目: 制約名の要件については、データソースのドキュメントを参照してください。

CHECK(*search-condition*)
列またはテーブルの値の条件を指定します。*search-condition* は、ブール値に解決される有効な FedSQL 式です。*search-condition* が偽の場合、テーブルは変更されません。

制限事項 SAS データセットの場合、検索条件に FedSQL 関数を含めることはできません。

CHECK 制約は、Vertica 7.2 以降でサポートされています。

データソース SAS データセット、Amazon Redshift、DB2、Greenplum、Microsoft SQL Server、MySQL、ODBC、Oracle、PostgreSQL、SAP HANA、SAP IQ、SingleStore、Teradata、Vertica、Yellowbrick

参照項目: “FedSQL 式” (46 ページ)

PRIMARY KEY
テーブルの各行について、列の値を使用してテーブルのそれぞれの行を一意に識別できることを指定します。変わらない値です。主キーに null 値を設定することはできません。

データソース SAS データセット、Amazon Redshift、DB2、Greenplum、Microsoft SQL Server、MongoDB、MySQL、ODBC、Oracle、PostgreSQL、SAP HANA、SAP IQ、SingleStore、Snowflake、Teradata、Vertica

注 MongoDB での PRIMARY KEY 制約の使用方法については、*SAS/ACCESS for Nonrelational Databases: リファレンス*を参照してください。

PRIMARY KEY (*column* [,...*column*])
テーブル内の各行について、指定されたすべての列の値を使用して、テーブル内のそれぞれの行を一意に識別することを指定します。主キーは null 値を持つことはできず、列の各行で一意である必要があります。変わらない値です。

データソース SAS データセット、Amazon Redshift、DB2、MDS、Microsoft SQL Server、MySQL、ODBC、Oracle、PostgreSQL、SAP HANA、SAP IQ、SingleStore、Snowflake、Teradata、Vertica、Yellowbrick

UNIQUE
列内の 2 つの行が同じ値を持つことができないことを指定します。NOT NULL を指定しない限り、null 値が許可されます。

注 Vertica の場合、UNIQUE 制約が適用される方法については、ベンダーのドキュメントを参照してください。

UNIQUE(*column* [,...*column*])
指定された列の 2 つの行が同じ値を持つことができないことを指定します。NOT NULL を指定しない限り、null 値が許可されます。

データソース SAS データセット、Amazon Redshift、DB2、Greenplum、MDS、Microsoft SQL Server、MySQL、ODBC、Oracle、PostgreSQL、SAP HANA、SAP IQ、SingleStore、Snowflake、Teradata、Vertica、Yellowbrick

NOT NULL
指定された列のどの行でも null 値が無効であることを指定します。

データソース SAS データセット、Amazon Redshift、DB2、Greenplum、Hive、MDS、Microsoft SQL Server、MySQL、ODBC、Oracle、PostgreSQL、Salesforce、SAP HANA、SAP IQ、SingleStore、Snowflake、Teradata、Vertica、Yellowbrick

FOREIGN KEY (*referencing-column* [,...*referencing column*]) REFERENCES *referenced-table* (*referenced-column* [,...*referenced-column*] [<referential-trigger-action>])
table 内の列の値を使用して別のテーブル内の列に関連付ける句を指定します。外部キーには、参照先テーブル内の行を一意に識別するのに十分な数の列が常に含まれている必要があります。外部キー制約は、複数のテーブル内の関連データの一貫性を確保するのに役立ちます。

FOREIGN KEY
別のテーブルの列に関連する *table* の 1 つ以上の列を識別する句を開始します。

データソース Amazon Redshift、DB2、Greenplum、Microsoft SQL Server、MongoDB、MySQL、ODBC、Oracle、PostgreSQL、SAP HANA、SAP IQ、Snowflake、Teradata、Vertica

注 MongoDB では、親テーブルとの関係を定義するために FOREIGN KEY 制約が使用されます。参照トリガー、アクション、およびチェック時間は、MongoDB ではサポートされていません。MongoDB での参照関係の使用方法の詳細については、*SAS/ACCESS for Nonrelational Databases: リファレンス*を参照してください。

referencing-columm
別のテーブルの列に関連する *table* の列を指定します。

REFERENCES
別のテーブルの列を識別する句を開始します。

referenced-table
列が *table* の列に関連するテーブルを指定します。

referenced-column
table 内の列に関連する、参照先テーブル内の 1 つ以上の列を指定します。

<referential-trigger-action>
table 内の列が更新または削除されたときに、参照先テーブルで実行するアクションを指定します。

ON UPDATE <referential-action>
table 内の列が更新されたときに実行するアクションの句を指定します。

ON DELETE <referential-action>
table 内の列が削除されたときに実行するアクションの句を指定します。

<referential-action>
table 内の列が更新または削除されたときに実行するアクションを指定します。

CASCADE
ON UPDATE は、参照先テーブルの更新された値を参照する *table* 内のすべての行を更新することを指定します。ON DELETE は、参照先テーブルで削除された行を参照する *table* 内のすべての行を削除します。

SET NULL
参照先の列を null 値に設定することを指定します。

SET DEFAULT
参照先の列をデフォルト値に設定することを指定します。

RESTRICT
FedSQL ステートメントの処理中にいつでも参照制約が満たされない場合、アクションは実行されず、更新または削除は失敗することを指定します。

NO ACTION
FedSQL ステートメントの最後で参照制約が満たされていない場合、アクションは実行されず、更新または削除は失敗することを指定します。

<constraint-check-time>
制約をいつチェックするかを指定します。

DEFERRABLE
NOT DEFERRABLE
トランザクションの完了後に制約違反チェックを実行できるか、または各 FedSQL ステートメントの最後にチェックする必要があるかを指定します。DEFERRABLE が指定されている場合、トランザクションの最後に制約チェックを行うことができます。NOT DEFERRABLE が指定されている場合、FedSQL ステートメントの終了時に制約をチェックする必要があります。DEFERRABLE の指定は、トランザクションを完了するために複数のステートメントが必要な場合に役立ちます。

デフォルト　DEFERRABLE

INITIALLY DEFERRED
デフォルトでは、制約違反チェックが各トランザクションの開始時に延期され、トランザクションの終了まで行われないことを指定します。

INITIALLY IMMEDIATE
各 FedSQL ステートメントの最後に制約違反チェックを行うことを指定します。

詳細

CREATE TABLE ステートメントの概要

CREATE TABLE ステートメントを使用すると、テーブルの列を定義するか、クエリ式を使用して既存のテーブルから列を選択することにより、テーブルを作成できます。SAS は、SAS データソースをサポートするために CREATE TABLE ステートメントに拡張機能を提供します。

SAS データソースをサポートするための CREATE TABLE ステートメントの SAS 拡張機能

FedSQL を使用した CREATE TABLE ステートメントの SAS 拡張機能を使用すると、出力形式と入を列に割り当てたり、列にラベルを追加したりできます。

SAS 出力形式
HAVING FORMAT 句を使用して、列データを SAS フォーマット値として格納および取得するように指定できます。データ型が CHAR または DOUBLE のいずれでもない場合、CHAR または DOUBLE のいずれかに変換されます。

たとえば、日付を *ddmmmyy* の出力形式で格納するには、*dd* が 2 桁の年、*mmm* が 3 桁の月、*yy* が 2 桁の年である場合、列定義でデータ型として DOUBLE を指定し、FORMAT='DATE7.'を指定します。2012 年 11 月 1 日は 01Nov12 として格納されます。列の定義は次のようになります。

```
saleenddate double not null having format date7.;
```

これに対して、DATE データ型を使用して日付情報を格納する列には、日付が *yyyy-mm-dd* の出力形式で格納されます。2012 年 11 月 1 日は 2012-11-01 として格納されます。

SAS 入力形式
SAS およびユーザー定義の入力形式は、HAVING INFORMAT 句を使用して、列データとともに格納および取得できます。入力形式はデータに適用されません。これは情報提供のみを目的としています。クライアントアプリケーションは、入力形式をデータに適用する役割を果たします。入力形式はすべてのデータ型に関連付けることができますが、すべてのデータ型は CHAR または DOUBLE のいずれかに変換されます。

```
saledates double having informat anydtdte14.;
```

SAS 入力形式の詳細については、6 章, “FedSQL 入力形式” (923 ページ)を参照してください。

列ラベル
ラベルは、列名のかわりにクエリ式の結果に表示される説明的な引用符付きのテキスト文字列です。HAVING 句で LABEL 引数を使用して、ラベルを指定します。

```
saleenddate double not null having label 'Last Day of Sale';
```

クエリ式の操作ではラベルを使用できません。

テーブルの列の定義

テーブルの列を定義するときは、列名とデータ型が必須です。他のすべての列定義引数はオプションです。列の命名規則については、データソースのドキュメントを参照してください。FedSQL 予約語は列名として使用できません。FedSQL 予約語のリストについては、“FedSQL 予約語” (72 ページ)を参照してください。

各列定義をカンマで区切ることにより、指定されていない数の列をテーブルに追加できます。列定義の完全なグループをかっこで囲みます。次の CREATE TABLE ステートメントは Customers テーブルを作成します。

```
create table customers
  ( custid double unique,
    name char(16),
    address char(64),
    city char(16),
    state char(2),
    country char(16),
    phone char(16),
    initorder date having label 'Initial Order'
  );
```

各列定義は列に名前を付け、データ型を割り当てます。CUSTID 列には、テーブル行を一意に識別するために使用される UNIQUE 一貫性制約が含まれています。INITORDER 列は DATE データ型です。コードを使用して SAS データセット、SPD Engine データセット、または SPD Server テーブルを作成する場合は、次の結果に注意してください。列が表示されるとき、列名 initorder のかわりにラベル Initial Order が使用されます。他のデータソースの場合、ラベルは無視されます。

INSERT ステートメントを使用して、テーブルに行を追加できます。INSERT ステートメントを使用すると、CREATE TABLE ステートメントで定義されたテーブル列ごとに値を指定できます。詳細については、“INSERT ステートメント” (972 ページ)を参照してください。

クエリ式からのテーブルの作成と入力

クエリ式を使用してテーブルを作成する場合、テーブルの作成時にテーブルに行を追加します。SELECT ステートメントを使用して既存のテーブルからデータを取得し、新しいテーブルを作成します。CREATE TABLE ステートメントの列定義の列数は、SELECT ステートメントによって返される列数と等しくなければなりません。CREATE TABLE ステートメントで列名が指定されていない場合、SELECT ステートメントによって返される列とデフォルト値が新しいテーブルで使用されます。

この CREATE TABLE ステートメントは、CorpData テーブルの 3 つの列のみに基づく新しいテーブルを作成します。

```
create table spainEmails
as select name, emailid, lastPurchaseDate from corpdata where country='Spain';
```

次の CREATETABLE ステートメントは、CorpData テーブルからすべての 列を選択します。

```
create table spain
as select * from corpdata where country='Spain';
```

次の例のように、クエリ式に_NULL_のテーブル名を使用して、テーブルを作成する前にクエリ式のパフォーマンスをテストできます。

```
create table _null_ as select * from corpdata where country='Spain';
```

FedSQL は通常の処理を実行してテーブルを 作成し、テーブルが作成されたように見えます。クエリが完了すると、テーブルは破棄されます。

データソース

FedSQL 言語を使用してデータ型を定義する場合、データを格納するには、そのデータソースのデータストレージで定義されたデータ型が使用できる必要があります。FedSQL はいくつかのデータ型をサポートしていますが、特定のテーブルに定義できるデータ型はデータソースによって 異なります。各データソースが必ずしもすべての FedSQL データ型をサポートしているわけではありません。

さらに、データソースは標準 SQL データ型のバリエーションをサポートします。つまり、指定したデータ型がデフォルトで別のデータ型になる可能性があり、基になるデータソースで別の属性を持つ可能性もあります。これは、データソースが特定のデータ型をネイティブにサポートしていないが、同様のデータ型のデータ値をデータ損失なしで変換できる場合に行われます。たとえば、INTEGER データ型をサポートするために、SAS データセットはデータ型定義をデフォルトで DOUBLE である SAS 数値にマッピングします。データソース依存の属性など、各データ型のデータソース実装の詳細については、付録 2, “データ型リファレンス” (1199 ページ)を参照してください。

一貫性制約

一貫性制約は、データの有効性と整合性を維持するために指定できる一連のデータ検証規則です。CREATE TABLE ステートメントで指定された一貫性制約は、データソースに渡されます。トランザクションがテーブルを変更すると、データソースは一貫性制約を適用します。

列制約は、1 つの列に対して定義される制約です。テーブル制約は、2 つ以上の列に対する制約を定義します。

次のステートメントは、一貫性制約を使用して SAS データセットを作成します。

- 製品 ID を主キーとして使用して Products テーブルを作成します。2 つの製品 ID で同じ製品を識別することはできず、製品を null 値にすることはできません。トランザクションの最後に制約をチェックします。

  ```
  create table products (prodid double primary key,
    product char(8) unique not null initially deferred);
  ```

- 合計が米ドルで表示され、国が含まれる Sales テーブルを作成します。

  ```
  create table sales (prodid double,
    custid double  primary key,
    totals double having format dollar10.,
    country char(30) not null);
  ```

- 顧客のクレジットカード情報用の SAS データセットを作成します。顧客は、名前と顧客番号によって一意に識別されます。

  ```
  create table custcredit (name char(30),
    custNum double,
    ccType char(15) having label 'Credit Card Type',
    ccNum char(20) having label 'Credit Card Number',
    ccExp date having label 'Expiration Date',
    CONSTRAINT ccconst primary key(name, CustNum));
  ```

FedSQL は、SAS データセットの DATE データ型を検出すると、DATE から DOUBLE への型変換を行い、DATE9.出力形式を列に割り当てます。

制約の詳細については、データソースのドキュメントを参照してください。

関連項目

ステートメント

- “ALTER TABLE ステートメント” (929 ページ)
- “DROP TABLE ステートメント” (966 ページ)

CREATE VIEW ステートメント

1 つ以上のテーブルまたは他のビューからデータの FedSQL ビューを作成します。

カテゴリ: データ定義

制限事項: このステートメントは、CAS で実行される FedSQL プログラムではサポートされていません。

データソース: SAS データセット、SPD Server テーブル、Amazon Redshift、DB2、Google BigQuery、Greenplum、Hive、Impala、JDBC、MDS、Microsoft SQL Server、MySQL、Netezza、ODBC、Oracle、PostgreSQL、SAP HANA、SingleStore、Snowflake、Spark、Teradata、Vertica、Yellowbrick

構文

CREATE VIEW *view* [SECURITY *security-type*] AS *query-expression*;

引数

SECURITY *security-type*
ビューに適用されるセキュリティの種類を指定します。*security-type* は、次の値のいずれかです。

DEFINER
ビューがスキーマ所有者の資格情報で実行されることを指定します。

INVOKER
ビューが呼び出し元のユーザーの資格情報で実行されることを指定します。

デフォルト: SECURITY を使用しない場合、ビューのセキュリティはデフォルトで INVOKER になります。

制限事項: セキュリティは、SAS Federation Server でのみ使用できます。SECURITY *security-type* 引数は、SAS Federation Server に向けられていない FedSQL 要求では無視されます。SAS Federation Server で使用で

きるセキュリティの詳細については、SAS Federation Server のドキュメントを参照してください。

view
作成するビューの名前を指定します。

要件 ビュー名は、同じデータベース内の他のビュー、テーブル、またはインデックスとは異なる必要があります。

query-expression
ビューの作成に使用される既存のテーブルから情報を取得する SELECT ステートメントを指定します。

参照項目: “クエリ式からのテーブルの作成と入力” (956 ページ)、“クエリ式とサブクエリ” (46 ページ)、および “SELECT ステートメント” (977 ページ)

詳細

CREATE VIEW ステートメントの概要

FedSQL CREATE VIEW ステートメントで作成されるビューは、DBMS ビューでも SAS ビューでもありません。CREATE VIEW ステートメントの出力は、FedSQL 言語でのみ表示可能なネイティブテーブルです。

ビュー内の基になるテーブルまたはビューが削除された場合、SAS はビューの実行時にエラーメッセージを返します。基になるテーブルまたはビューが変更された場合、ビューの再作成が必要になる場合があります。

ビューへのアクセス

ビューを作成するには、CREATE VIEW 権限が必要です。ビューの所有者は、他のユーザーに権限を付与する機能を含め、ビューに対するすべての権限を持っています。ビューを実行するには、ユーザーはビューで参照されているテーブルとビューに対する SELECT 権限を持っている必要があります。

権限は現在、SAS Federation Server でのみサポートされています。

SELECT ステートメントの制限事項

SELECT ステートメントに ORDER BY 句を含めることはできません。

FedSQL ビューと SAS

SAS プロシジャ CONTENTS、PRINT、および SQL は、FedSQL ビューのデータにアクセスできません。ただし、これらのプロシジャは、FedSQL ビューの構造に関する情報を出力します。

関連項目

ステートメント:

- “CREATE TABLE ステートメント” (939 ページ)
- “DESCRIBE VIEW ステートメント” (964 ページ)
- “DROP VIEW ステートメント” (968 ページ)
- “SELECT ステートメント” (977 ページ)

DELETE ステートメント

テーブルから行を削除します。

カテゴリ: データ操作

制限事項: このステートメントは、CAS で実行される FedSQL プログラムではサポートされていません。

データソース: SAS データセット、SPD Engine データセット、SPD Server テーブル、Amazon Redshift、DB2、Google BigQuery、Greenplum、JDBC、MDS、Microsoft SQL Server、MongoDB、MySQL、Netezza、ODBC、Oracle、PostgreSQL、SAP HANA、SAP IQ、SingleStore、Snowflake、Teradata、Vertica、Yellowbrick

構文

DELETE FROM *table* [WHERE <search-condition>];

引数

table
行を削除するテーブルを指定します。

制限事項 テーブルに対して行レベルのアクセス許可が有効になっている場合、テーブルから行を削除することはできません。行レベルのセキュリティは、SAS Federation Server でのみ使用できます。詳細については、SAS Federation Server のドキュメントを参照してください。

ヒント Microsoft Access テーブルとの互換性のために、テーブル名の後に'.*'を使用できます。

WHERE <search-condition>
削除される行数を制限するために使用される有効な WHERE 句を指定します。

注 MongoDB サブクエリでは、単一変数演算子のみが許可されます。FedSQL は、サブクエリを内部クエリとして使用して、MongoDB 子テーブルまたはメタデータテーブルのデータを削除することをサポートしていません。

ヒント　WHERE ステートメントでパラメーター配列を使用できます。

参照項目:　“SELECT ステートメント” (977 ページ)の WHERE 句

詳細

DELETE ステートメントは、指定されたテーブルから WHERE 句を満たす行を削除します。WHERE 句を使用しない場合、すべての行が削除されます。

比較

DROP TABLE ステートメントは、テーブル構造を含め、データベースからテーブルを完全に削除します。DELETE ステートメントは、テーブル内の行(データ)のみを削除します。

関連項目

ステートメント:

- “DROP TABLE ステートメント” (966 ページ)
- “INSERT ステートメント” (972 ページ)
- “UPDATE ステートメント” (1009 ページ)

DESCRIBE TABLE ステートメント

テーブルから SQL を取得し、結果セットを返します。

カテゴリ:　データ定義
データ取得

制限事項:　このステートメントは、CAS で実行される FedSQL プログラムではサポートされていません。

データソース:　すべて

構文

DESCRIBE TABLE [*catalog*.][*schema*.] *table*;

引数

catalog
テーブルを含むカタログを指定します。

schema
テーブルを含むスキーマを指定します。

table
テーブルの名前を指定します。

詳細

DESCRIBE TABLE ステートメントは、テーブルが最初に作成された方法(別のプログラミング言語など)に関係なく、指定されたテーブルの CREATE TABLE ステートメントを SAS ログに書き込みます。DESCRIBE TABLE ステートメントによって返される列定義は、列のネイティブデータ型を示します。テーブルの列にインデックスが定義されている場合、それらのインデックスの CREATE INDEX ステートメントが SAS ログに書き込まれます。CREATE TABLE ステートメントには、指定されたテーブルに対して定義されている一貫性制約も含まれます。SAS データセットの場合、主キーの一貫性制約は CREATE UNIQUE INDEX ステートメントによって記述されます。主キー制約を参照する外部キー変数は、主キー制約の DESCRIBE TABLE 出力の一部として表示されません。

FedSQL 言語は、サポートするデータソースで使用されるほとんどのデータ型をサポートしますが、すべてではありません。各データソースで FedSQL がサポートするデータ型のリストについては、付録 2, “データ型リファレンス” (1199 ページ)を参照してください。FedSQL が直接サポートしていないデータ型のテーブルを作成する必要がある場合は、EXECUTE ステートメントを使用して要求を行います。

DESCRIBE TABLE ステートメントは、FedSQL DICTIONARY テーブルではサポートされていません。

例

例 1: SAS データセットについて返される情報

SAS DATA ステップで作成された SAS データセットを考えてみます。

```
data myfiles.employees;
  input IdNum $4. +2 LName $11. FName $11. JobCode $3.
      +1 Salary 5. +1 Phone $12.;
```

DESCRIBE TABLE ステートメントは、次の情報を SAS ログに書き込みます。

```
describe table myfiles.employees;
```

```
CREATE TABLE MYFILES."MYFILES".EMPLOYEES (
     "IdNum"     CHARACTER(4),
     "LName"     CHARACTER(11),
     "FName"     CHARACTER(11),
     "JobCode"     CHARACTER(3),
     "Salary"     DOUBLE,
     "Phone"     CHARACTER(12)
);
```

例 2: サポートされているデータ型について返される情報

次のステートメントで FedSQL を使用して作成された Oracle テーブルについて考えてみます。

CREATE TABLE ORACLE.FOO1 (COL1 INTEGER, COL2 VARHCHAR(10);

Oracle は INTEGER データ型をサポートしていません。FedSQL 言語は、Oracle にデータを書き込むときに、INTEGER データ型を Oracle の NUMBER データ型に変換します。

DESCRIBE TABLE oracle.foo1;

```
CREATE TABLE "ORACLE"."SCOTT".FOO1 (
     COL1     NUMBER,
     COL2     VARCHAR2(10))
```

例 3: FedSQL のサポートが制限されているデータ型について返される情報

DB2 ドライバーをネイティブに使用して、次のステートメントで作成された DB2 テーブルを考えてみましょう。

CREATE TABLE dlfoo1 (col1 CLOB);

FedSQL では、CLOB データ型のサポートが制限されています。現在、CLOB データ型の列の作成は許可されていませんが、CLOB 列の読み取りはある程度サポートされています。DESCRIBE TABLE ステートメントは、次の情報を SAS ログに書き込みます。

DESCRIBE TABLE dlfoo1;

```
CREATE TABLE "DB2"."DBITEST".DLFOO1 (
     COL1     CLOB
);
```

例 4: Teradata について返される情報

FedSQL を使用して作成された Teradata テーブルについて考えてみましょう。

create table x.customer (item_name char(30), cost decimal(10,2), delivered date);

DESCRIBE TABLE ステートメントは、テーブル内の列を記述する CREATE TABLE ステートメントを返します。インデックスが明示的に定義されていませんが、CREATE INDEX ステートメントも返します。インデックスは FedSQL の抽象化です。テーブルの再作成には使用しないでください。

describe table x.customer;

```
CREATE TABLE X."dmvdq1".CUSTOMER (
     ITEM_NAME     CHAR(60) DEFAULT NULL,
     COST     DECIMAL DEFAULT NULL,
     DELIVERED     DATE DEFAULT NULL
);
CREATE INDEX NONAME1 ON X."dmvdq1".CUSTOMER(ITEM_NAME);
```

関連項目

ステートメント:

- “CREATE TABLE ステートメント” (939 ページ)

DESCRIBE VIEW ステートメント

ビューから SQL を取得し、結果セットを返します。

カテゴリ: データ定義
データ取得

制限事項: このステートメントは、CAS で実行される FedSQL プログラムではサポートされていません。
このステートメントは、SAS DATA ステップまたは PROC SQL で作成された SAS データビューではサポートされていません。

データソース: ビューをサポートする SQL ベースのデータソース。

構文

DESCRIBE VIEW ["*catalog*".]["*schema*".]"*view*"

DESCRIBE VIEW XML ["*catalog*".]["*schema*".]"*view*"

引数

catalog
ビューを含むカタログを指定します。

schema
ビューを含むスキーマを指定します。

view
ビューの名前を指定します。

詳細

DESCRIBE VIEW を使用して、FedSQL または DBMS ビューから標準 SQL テキストを取得します。ビューから SQL を XML として取得するには、DESCRIBE VIEW XML を使用します。これにより、クライアントに SQL パーサーを実装する必要がなくなります。

セキュリティで保護されたビューで DESCRIBE VIEW を実行するには、ALTER VIEW 権限が必要です。

関連項目

ステートメント:

- “CREATE VIEW ステートメント” (958 ページ)

DROP INDEX ステートメント

指定されたインデックスをテーブルから削除します。

カテゴリ: データ定義

制限事項: このステートメントは、CAS で実行される FedSQL プログラムではサポートされていません。

データソース: SAS データセット、SPD Engine データセット、SPD Server テーブル、Amazon Redshift、DB2、Greenplum、JDBC、MDS、Microsoft SQL Server、MySQL、ODBC、Oracle、PostgreSQL、SAP HANA、SAP IQ、SingleStore、Teradata

構文

DROP INDEX *index* [FROM *table*] [FORCE];

引数

index
削除するインデックスの名前を指定します。

FROM *table*
インデックスが存在するテーブルの名前を指定します。

要件 SAS データセット、SPD Engine データセット、SPD Server テーブル、MDS テーブルの場合、インデックスが存在するテーブルの名前を指定するには、FROM *table* 構文を含める必要があります。

すべてのデータソースについて、SAS セキュリティが有効になっているデータソース名(DSN)を使用してデータソースに接続した場合は、FROM *table* 構文を含める必要があります。

FORCE
エラー処理なしでインデックスを削除するように指定します。FORCE キーワードは、エラー処理を行わずにインデックスを削除してもテーブルに 悪影響が及ばないことが確実な場合にのみ使用してください。

詳細

複合インデックスを削除すると、そのインデックスで指定されたすべての列のインデックスが削除されます。

注: DROP INDEX ステートメントは、それぞれ CREATE TABLE または ALTER TABLE ステートメントの PRIMARY KEY または UNIQUE 制約を使用して作成されたインデックスには適用されません。これらのインデックスを削除するには、まず ALTER TABLE ステートメントを使用して制約を削除する必要があります。

関連項目

ステートメント:

- “ALTER TABLE ステートメント” (929 ページ)
- “CREATE INDEX ステートメント” (937 ページ)

DROP TABLE ステートメント

データベースからテーブルを削除します。

カテゴリ: データ定義

制限事項: DROP TABLE ステートメントは、CAS では異なる動作をします。CAS 情報については、“DROP TABLE ステートメント” (*SAS Viya プラットフォーム: SAS Cloud Analytic Services の FedSQL プログラミング*)を参照してください。

データソース: SAS データセット、SPD Engine データセット、SPD Server テーブル、Amazon Redshift、DB2、Google BigQuery、Greenplum、Hive、Impala、JDBC、MDS、Microsoft SQL Server、MongoDB、MySQL、Netezza、ODBC、Oracle、PostgreSQL、Salesforce、SAP HANA、SAP IQ、SingleStore、Snowflake、Spark、Teradata、Vertica、Yellowbrick

構文

DROP TABLE *table* [FORCE];

引数

table
削除するテーブルの名前を指定します。

制限事項
別のトランザクションまたはステートメントで参照されている間、MDS テーブルを変更または削除することはできません。要求が失敗した場合は、他のユーザーがテーブルを使用していないか、切断されていることを確認してください。

子テーブルを持つ MongoDB テーブルは削除できません。子テーブルを持つ MongoDB テーブルを削除するには、最初に子テーブルを削除する必要があります。

FORCE
エラー処理なしでテーブルを削除するように指定します。

詳細

DROP TABLE ステートメントは、テーブル定義と、テーブルのすべてのデータ 、メタデータ、およびインデックスを削除します。

DELETE ステートメントを使用してテーブル内のすべての行を削除する場合、テーブルは DROP TABLE ステートメントで削除されるまで存在し続けます。

DROP TABLE ステートメントを使用して、外部キー制約によって参照されているテーブルを削除することはできません。最初に外部キー制約を削除してから、テーブルを削除する必要があります。

インデックス付きの列を含むテーブルを削除すると、すべてのインデックスが 自動的に削除されます。複合インデックスを削除すると、そのインデックスで指定されたすべての列のインデックスが削除されます。

CREATE TABLE ステートメントと同じサブミットで DROP TABLE ステートメントを指定する場合は、DROP TABLE ステートメントで FORCE オプションを指定します。指定されたテーブルが見つからない場合、DROP TABLE ステートメントはメッセージをログに書き込みます。これらのメッセージは、CREATE TABLE ステートメントに干渉する可能性があります。FORCE オプションを使用すると、メッセージが表示されても CREATE TABLE 要求を成功させることができます。

Teradata ユーザー: 付録 7, “使用上の注意” (1297 ページ)を参照してください。

比較

DELETE ステートメントは、テーブル内の行(データ)のみを削除します。DROP TABLE ステートメントは、テーブル構造を含め、データベースからテーブルを 完全に削除します。

例

例 1: テーブルを削除する

```
proc fedsql;
  drop table mytable;
quit;
```

例 2: テーブルの削除とテーブルの作成を同時に行う

```
proc fedsql;
  drop table mytable force;
  create table mytable (col1 integer, col2 char(5));
quit;
```

関連項目

ステートメント:

- “CREATE TABLE ステートメント” (939 ページ)
- “DELETE ステートメント” (960 ページ)

DROP VIEW ステートメント

データベースからビューを削除します。

カテゴリ: データ定義

制限事項: このステートメントは、CAS で実行される FedSQL プログラムではサポートされていません。

データソース: SAS データセット、SPD Server テーブル、Amazon Redshift、DB2、Google BigQuery、Greenplum、Hive、Impala、JDBC、MDS、Microsoft SQL Server、MySQL、Netezza、ODBC、Oracle、PostgreSQL、SAP HANA、SingleStore、Snowflake、Spark、Teradata、Vertica、Yellowbrick

構文

DROP VIEW *view* [FORCE];

引数

view
削除するビューの名前を指定します。

FORCE
エラー処理なしでビューを削除するように指定します。

詳細

ビューは仮想テーブルと見なすことができます。ビューは、1 つ以上のテーブルに対してクエリ式を実行することによって形成されます。ビューを削除しても、データベース内のデータは変更されません。ビューに関連付けられているメタデータのみが削除されます。

削除されたテーブルのビューは、DROP VIEW ステートメントを使用して明示的に削除できます。

CREATE VIEW ステートメントと同じサブミットで DROP VIEW ステートメントを指定する場合は、DROP VIEW ステートメントで FORCE オプションを指定します。指定されたテーブルが見つからない場合、DROP VIEW ステートメントはメッセージをログに書き込みます。これらのメッセージは、CREATE VIEW ステートメントに干渉する可能性があります。FORCE オプションを使用すると、メッセージが表示されても CREATE VIEW 要求を成功させることができます。

比較

DROP TABLE ステートメントは、データベースからテーブルを削除します。DROP VIEW ステートメントは、データベースからビューを削除します。

関連項目

ステートメント:

- “CREATE VIEW ステートメント” (958 ページ)
- “DESCRIBE VIEW ステートメント” (964 ページ)
- “DROP TABLE ステートメント” (966 ページ)

EXECUTE ステートメント

FedSQL がサポートする DBMS に DBMS 固有のステートメントを送信します。

カテゴリ: FedSQL パススルー

制限事項: このステートメントは、CAS で実行される FedSQL プログラムではサポートされていません。

データソース: Amazon Redshift、DB2、Google BigQuery、Greenplum、Hive、JDBC、Microsoft SQL Server、MongoDB、MySQL、Netezza、ODBC、Oracle、PostgreSQL、Salesforce、SAP HANA、SAP IQ、SingleStore、Snowflake、Spark、Teradata、Vertica、Yellowbrick

参照項目: “FedSQL 明示的パススルー機能” (70 ページ)
DBMS の SQL ドキュメント

構文

EXECUTE(*native-syntax*) **BY** *catalog*;

引数

native-syntax
カタログのドライバーで実行できるネイティブクエリを指定します。EXECUTE ステートメントは、結果セットを生成するステートメントだけでなく、結果セットを生成しないステートメントも受け入れます。

制限事項 PROC FEDSQL は、埋め込まれていないセミコロンを含むネイティブ構文をサポートしていません。ただし、FedSQL 言語プロセッサを使用してこれらのステートメントを実行することはできます。

注 SQL ステートメントは、データソースによっては大文字と小文字が区別される場合があり、入力したとおりにデータソースに渡されます。

catalog
既存の FedSQL 接続のカタログの名前を指定します。

詳細

一般的な使用

EXECUTE ステートメントは、結果セットを生成しないネイティブ SQL ステートメントを DBMS にサブミットするために FedSQL が提供する唯一の方法です。ネイティブ SQL は、DDL、UPDATE、SELECT、バルクロードのステートメントで使用できます。

MongoDB で使用する

MongoDB は SQL ベースのデータソースではありません。MongoDB からデータを読み書きするために、SAS/ACCESS ソフトウェアはリレーショナルスキーマを生成します。デフォルトでは、このスキーマはメモリ内に存在しますが、スキーマを格納できます。EXECUTE ステートメントを使用して REFRESH コマンドをサブミットし、格納されている MongoDB スキーマを生成して新しいスキーマで上書きすることができます。

コレクションや列の追加や削除など、データベースの変更が SAS 以外の方法で行われた場合にのみ、このコマンドを使用して格納されたスキーマを更新します。CREATE TABLE ステートメントで新しいコレクションを追加するか、DROP TABLE ステートメントでコレクションを削除する場合は、このコマンドを使用しないでください。データベース接続で SCHEMA_COLLECTION=オプションが設定されている場合、SAS で行われた構造変更は即座にスキーマに保存されます。

REFRESH コマンドは、次のように SAS セッションからサブミットされます。

```
proc fedsql;
  execute (refresh) by libref;
quit;
```

REFRESH コマンドの実行は管理タスクであり、システム管理者のみが実行する必要があります。REFRESH は、現在の MongoDB データベースコレクションのスキャンを強制し、それに応じて現在のスキーマ(メモリ内または名前で保存されたもの)を更新します。これにより、格納されたスキーマに直接加えられた可能性のある手動の変更が上書きされるため、このコマンドは注意して使用する必要があります。

MongoDB スキーマの詳細については、*SAS/ACCESS for Nonrelational Databases: リファレンス*の"MongoDB データのスキーマの操作"を参照してください。

また、FedSQL 明示的パススルーを使用して、JSON スタイルの MongoDB コマンドをサブミットし、MongoDB コレクションのクエリ、作成、および更新を行うこともできます。サポートされている構文については、*SAS/ACCESS for Nonrelational Databases: リファレンス*の MongoDB に関する情報を参照してください。

例

例 1: EXECUTE を使用して Saphana でテーブルを作成および削除する

```
execute(create table t_blob (col1 blob)) by saphana;
execute(drop table t_blob) by saphana;
```

例 2: EXECUTE を使用して Oracle で結果セットを生成する

```
execute(select i, rank() over (order by j) rank from table_a )
   by oracle;
```

関連項目

概念:

- “FedSQL 明示的パススルー機能” (70 ページ)

ステートメント:

- “ALTER TABLE ステートメント” (929 ページ)
- “BEGIN ステートメント” (935 ページ)
- “COMMIT ステートメント” (936 ページ)
- “CREATE INDEX ステートメント” (937 ページ)
- “CREATE TABLE ステートメント” (939 ページ)
- “DELETE ステートメント” (960 ページ)
- “DROP INDEX ステートメント” (965 ページ)
- “DROP TABLE ステートメント” (966 ページ)
- “INSERT ステートメント” (972 ページ)
- “ROLLBACK ステートメント” (976 ページ)
- “SELECT ステートメント” (977 ページ)
- “UPDATE ステートメント” (1009 ページ)

INSERT ステートメント

指定されたテーブルに行を追加します。

カテゴリ:	データ操作
制限事項:	このステートメントは、CAS で実行される FedSQL プログラムではサポートされていません。
要件	Salesforce では、列名を使用するために挿入が必要です。
データソース:	SAS データセット、SPD Engine データセット、SPD Server テーブル、Amazon Redshift、DB2、Google BigQuery、Greenplum、Hive、Impala、JDBC、MDS、MongoDB、MySQL、Netezza、ODBC、Oracle、PostgreSQL、Salesforce、SAP HANA、SAP IQ、SingleStore、Snowflake、Spark、Teradata、Vertica、Yellowbrick
注:	MongoDB テーブルはリレーショナルテーブルではなく 、リレーショナルテーブルとは大きな違いがあります。MongoDB にデータを挿入する前に、*SAS/ACCESS for Nonrelational Databases: リファレンス*で MongoDB に関するドキュメントを参照してください。

構文

形式 1: **INSERT** INTO *table*

{VALUES (*value* | NULL [, ...*value* | NULL])};

形式 2: **INSERT** INTO *table*

{ (*column* [, ...*column*]) VALUES (*value* | NULL [, ...*value* | NULL]) };

形式 3: **INSERT** INTO *table*

{ (*column* [, ...*column*]) [*query-expression*] };

引数

table
行を挿入するテーブルの名前を指定します。

制限事項 テーブルに対して行レベルのアクセス許可が有効になっている場合、テーブルに行を挿入することはできません。行レベルのセキュリティは、SAS Federation Server でのみ使用できます。行レベルのセキュリティの詳細については、SAS Federation Server のドキュメントを参照してください。

value
テーブルに挿入するデータ値を指定します。

制限事項 テーブルの各列、または列リスト(*column*)の各列に 1 つの *value* が必要です。

要件 列が列リスト(*column*)で指定されている場合、データ値リストは列リストと同じ順序にする必要があります。

ヒント INSERT ステートメントでパラメーター配列を使用できます。

column
データを挿入する列を指定します。

注 構造 ID 値を含む MongoDB テーブルで INSERT を実行する場合、構造 ID 値を JSON 文字列として表す必要があります。詳細については、*SAS/ACCESS for Nonrelational Databases: リファレンス*の"MongoDB での構造 ID 値の操作"を参照してください。

ヒント 複数の列を指定できます。列リストとデータ値リストの順序が互いに一致している限り、列はテーブルに表示される順序と同じである必要はありません。

<query-expression>
行を返す任意の有効なクエリ式を指定します。各行の列数は、挿入するアイテムの数と同じです。

参照項目: “SELECT ステートメント” (977 ページ)

詳細

形式 1: 列名のない INSERT ステートメント

列名を指定せずに VALUES 句を使用するこの形式の INSERT ステートメントを使用して、値のリストをテーブルに挿入できます。テーブルの各列に値を指定します。VALUES 句ごとに 1 行が挿入されます。VALUES 句の値の順序は、テーブルの列の順序と一致する必要があります。

次のコードフラグメントは、INSERT ステートメントを VALUES 句とともに使用して、“CREATE TABLE ステートメント” (939 ページ)で作成された Customers テーブルにデータ値を挿入する方法を示しています。Customers テーブルには、列 CustId、Name、Address、City、State、Country、PhoneNumber、InitOrder があります。

```
insert into customers values (1,'Peter Frank', '300 Rock Lane', 'Boulder', 'CO',
'United States', '3039564321', date '2012-01-14');
insert into customers values (2,'Jim Stewart', '1500 Lapis Lane', 'Little Rock',
'AR', 'United States', '8705553978', date '2012-03-20');
insert into customers values (3,'Janet Chien', '75 Jujitsu', 'Nagasaki', '', 'Japan',
'01181956879932', date '2012-06-07');
insert into customers values (4,'Qing Ziao', '10111 Karaje', 'Tokyo', '', 'Japan',
'0118136774351', date '2012-10-12');
insert into customers values (5,'Humberto Sertu', '876 Avenida Blanca', 'Buenos Aires',
'', 'Argentina','01154118435029', date '2012-12-15');
```

これらのステートメントは、“CREATE TABLE ステートメント” (939 ページ).で作成された Sales テーブルにデータ値を追加します。Sales テーブルには、列 ProdID、CustID、Totals、Country があります。

```
insert into sales values (3234, 1, 189400, 'United States');
insert into sales values (1424, 3, 555789, 'Japan');
insert into sales values (3421, 4, 781183, 'Japan');
insert into sales values (3421, 2, 2789654, 'United States');
insert into sales values (3975, 5, 899453, 'Argentina');
```

文字値は引用符で囲まれていることに注意してください。

形式 2: 列名を含む INSERT ステートメント

特定の列名を持つ VALUES 句を使用するこの形式の INSERT ステートメントは、テーブルに値のリストを挿入するためにも使用できます。列名のリストで指定された列に値を指定します。列名はテーブルに表示される順序と同じである必要はありませんが、VALUES 句の値の順序は INSERT 列リストの列名の順序と一致する必要があります。VALUES 句ごとに 1 行が挿入されます。

次のコードフラグメントでは、列名を使用して、“CREATE TABLE ステートメント” (939 ページ).で作成された Products テーブルに値を追加します。

```
insert into products (prodid, product) values (1424, 'Rice');
insert into products (prodid, product) values (3421, 'Corn');
insert into products (prodid, product) values (3234, 'Wheat');
insert into products (prodid, product) values (3485, 'Oat');
```

列リストを使用すると、指定した列の値を含む行を追加できます。列リストを指定しない場合は、テーブル内のすべての列の値を VALUES 句で指定する必要があります。

形式 3: クエリ式を含む INSERT ステートメント

この形式の INSERT ステートメントは、別のテーブルから選択された行を挿入するクエリ式を使用します。クエリ式によって返される値の順序は、列名のリストで指定された列の順序と一致する必要があります。クエリ式によって返されるすべての行がテーブルに挿入されます。この形式の INSERT ステートメントを使用する場合は、次の制限に注意する必要があります。

- 列リストを指定しない場合は、テーブル内のすべての列の値をクエリ式で指定する必要があります。
- 列リストですべての列を指定しない場合、リストされていない各列に null 値が挿入されます。
- 追加する列のデータ型がテーブル内の列のデータ型と一致しない場合、自動型変換が試行されます。

クエリ式を含む INSERT ステートメントの例を次に示します。

```
insert into coldtemps select * from worldtemps
 where AvgLow >=40;
```

この例では、テーブル WorldTemps をクエリし、クエリが返した行を新しいテーブル ColdTemps に挿入します。ColdTemps テーブルには、WorldTemps テーブルと同じ列名とデータ型があります。テーブル ColdTemps に挿入される行には、平均最低気温が 40 度以下のテーブル WorldTemps の国が含まれます。テーブル WorldTemps の詳細については、“WorldTemps” (1277 ページ)を参照してください。

日付、時間、およびタイムスタンプ値の挿入

日付、時間、およびタイムスタンプ値の挿入については、“FedSQL の日付と時間” (59 ページ)を参照してください。

比較

DELETE ステートメントを使用すると、テーブルから行を削除できます。UPDATE ステートメントを使用すると、テーブル内の行を変更できます。INSERT ステートメントを使用すると、新しい行をテーブルに挿入できます。

関連項目

ステートメント:

- “DELETE ステートメント” (960 ページ)
- “SELECT ステートメント” (977 ページ)
- “UPDATE ステートメント” (1009 ページ)

ROLLBACK ステートメント

トランザクションの変更をトランザクションの先頭にロールバックします。

カテゴリ: データコントロール

制限事項: このステートメントは、CAS で実行される FedSQL プログラムではサポートされていません。

ROLLBACK ステートメントは、自動コミット機能がオフの場合にのみ効果があります。

データソース: Greenplum、MDS、Microsoft SQL Server、MySQL、ODBC、Oracle、SAP HANA、SAP IQ、SingleStore、Snowflake、Teradata、Vertica

構文

ROLLBACK [TRANSACTION];

詳細

プログラムがトランザクション内のすべてのステートメントを完了したら、COMMIT または ROLLBACK を使用してトランザクションを明示的に終了する必要があります。ROLLBACK ステートメントを使用して、トランザクションの開始以降に行われた変更をロールバックまたは元に戻します。

COMMIT ステートメントの実行後は、変更をデータベースにロールバックすることはできません。

ROLLBACK ステートメントは、自動コミット機能がオフの場合にのみ効果があります。ほとんどのデータソースでは、自動コミット機能がデフォルトでオンになっています。自動コミット機能をオフにする方法については、サーバー管理のドキュメントを参照してください。たとえば、FedSQL ドライバーへの適切な接続オプションについては、*SAS Federation Server: 管理者ガイド*を参照してください。FEDSQL プロシジャについては、*Base SAS プロシジャガイド*を参照してください。

比較

COMMIT ステートメントは、トランザクションの開始以降に実行されたすべての変更を取得し、それらをデータベースの永続的な部分にします。ROLLBACK ステートメントを使用すると、トランザクションによって行われたすべての 変更が、トランザクションの開始時にロールバックされます。

関連項目

ステートメント:

- “BEGIN ステートメント” (935 ページ)
- “COMMIT ステートメント” (936 ページ)

SELECT ステートメント

テーブルからデータの列と行を取得します。

カテゴリ: データ定義
データ操作
データ取得

制限事項: CAS で実行される FedSQL プログラムでは、SELECT ステートメントの完全な機能はサポートされていません。CAS 情報については、“SELECT ステートメント” (*SAS Viya プラットフォーム: SAS Cloud Analytic Services の FedSQL プログラミング*)を参照してください。

データソース: SAS データセット、SPD Engine データセット、SPD Server テーブル、Amazon Redshift、DB2、Google BigQuery、Greenplum、Hive、Impala、JDBC、MDS、Microsoft SQL Server、MongoDB、MySQL、Netezza、ODBC、Oracle、PostgreSQL、Salesforce、SAP、SAP HANA、SAP IQ、SingleStore、Snowflake、Spark、Teradata、Vertica、Yellowbrick

注: MongoDB テーブルはリレーショナルテーブルではなく 、リレーショナルテーブルとは大きな違いがあります。MongoDB でテーブルを結合する前に、*SAS/ACCESS for Nonrelational Databases: リファレンス*で MongoDB に関する情報を参照してください。

構文

The main clauses of the SELECT statement can be summarized as follows.

```
SELECT <select-list>
    FROM <table-specification>
    [WHERE <search-condition>]
    [GROUP BY <grouping-column>]
    [HAVING <search-condition>]
    [ORDER BY <sort-specification>]
    [LIMIT {count | ALL}]
    [OFFSET number]
;
```

The detailed syntax of the SELECT statement is as follows.

<query-expression>
[ORDER BY <sort-specification> [, ...<sort-specification>]];

<query-expression>::=
{<query-specification> | <query-expression>}
{UNION | EXCEPT | INTERSECT} {[ALL][CORRESPONDING [BY (column[,... column])]]} {<query-specification> | <query-expression>}

<query-specification>::=
SELECT [ALL | DISTINCT] <select-list> <table-expression>

<select-list>::=
*
| *column* [AS *column-alias*]
| *expression* [AS *column-alias*]
| *table*.*
| *table-alias*.*
| *table.column* [AS *column-alias*]

<table-expression>::=
FROM <table-specification> [, ...<table-specification>]
[WHERE <search-condition>]
[GROUP BY <grouping-column> [, ...<grouping-column>]]
[HAVING <search-condition>]

<table-specification>::=
table [[AS]*alias*]
| CONNECTION TO *catalog* (<native-syntax>) [[AS]*table-alias*]
| (<query-specification>) [AS] *alias*
| <joined-table>

<joined-table>::=
<cross-join>
| <qualified-join>
| <natural-join>
<cross-join>::=
<table-specification> CROSS JOIN <table-specification>
<qualified-join>::=
<table-specification> [<join-type>] JOIN <table-specification> <join-specification>
<natural-join>::=
<table-specification> NATURAL [<join-type>] JOIN <table-specification>
<join-type>::=
INNER
| LEFT [OUTER]

```
        | RIGHT [OUTER]
        | FULL [OUTER]
    <join-specification>::=
        ON <search-condition>
        | USING (column [, ...column])
<search-condition>::=
    {
        [NOT] {<sql-expression> | (<search-condition>)}
            [{AND | OR} [NOT] {<sql-expression> | (<search-condition>)}]
     }
        [,... {[NOT] {<sql-expression> | (<search-condition>)}
          [{AND | OR} [NOT] {<sql-expression> | (<search-condition>)}]}]
<sql-expression>::=
    expression {operator | predicate} expression
<sort-specification>::=
    {order-by-expression [ASC | DESC]} [, ...order-by-expression [ASC | DESC]]
<grouping-column>::=
    column [, ...column]
    | column-position-number
    | <sql-expression>
```

引数

構文引数の説明については、次のセクションを参照してください。

- “SELECT 句” (980 ページ)
- “FROM 句” (982 ページ)
- “WHERE 句” (992 ページ)
- “GROUP BY 句” (993 ページ)
- “HAVING 句” (995 ページ)
- “ORDER BY 句” (996 ページ)
- “LIMIT 句” (1000 ページ)
- “OFFSET 句” (1001 ページ)
- “UNION 演算子” (1001 ページ)
- “EXCEPT 演算子” (1003 ページ)
- “INTERSECT 演算子” (1005 ページ)

詳細

概要

SELECT ステートメントは 2 つの方法で使用できます。

- 単独で実行できる単一行の SELECT ステートメントは、1 行のみを返します。次に例を示します。

  ```
  select 42;
  select 42 as x;
  ```

  最初のコードフラグメントは、値 42 を含む単一列を返します。この列の名前は“column”です。2 番目のコードフラグメントも同様の列を返します。ただし、列の名前は“x”です。

- クエリ指定は SELECT キーワード(SELECT 句と呼ばれる)で始まり、単独で使用することはできません。1 つ以上のテーブルから列の値を読み取り、テーブルから返されるデータの条件を定義できるようにします。別の SQL ステートメントの一部として使用する必要があり、複数の行を返すことができます。クエリ指定により、仮想テーブルが作成されます。次に例を示します。

  ```
  select column(s)
  from table(s)
  where condition(s);
  ```

SELECT ステートメントの句の順序が重要です。オプションの句は省略できますが、使用する場合は適切な順序で記述する必要があります。SELECT ステートメントは、SELECT ステートメント内でも指定できます(サブクエリと呼ばれます)。ORDER BY、OFFSET、および LIMIT 句は、SELECT ステートメントの最も外側の SELECT でのみ使用できます。

ビューは、テーブルを指定できる場所ならどこでも SELECT ステートメントで指定できます。FedSQL は、ネイティブの DBMS ビューと FedSQL ビューをサポートしています。PROC SQL ビューはサポートしていません。

注: FedSQL クエリで参照できるテーブルの数に制限はありません。ただし、テーブル参照の数が多いクエリは、パフォーマンスの問題を引き起こす可能性があります。

SELECT 句

説明

仮想結果セットに表示される列をリストします。

構文

SELECT [ALL | DISTINCT] <select-list>

<select-list>::=

*

| *column* [AS *column-alias*]

| <query-specification>

| <sql-expression> [AS *column-alias*]

| *table*.*

| *table-alias*.*

| *table.column* [AS *column-alias*]

引数

ALL
結果セット内の重複行を含め、すべての行が含まれます。

DISTINCT
結果セット内の重複行を削除します。

<select-list>
結果セット用に選択する列を指定します。

*****
FROM 句にリストされているテーブル内のすべての列を選択します。

column-alias
列に一時的な代替名を割り当てます。

***column* [AS*column-alias*]**
単一の列を選択します。[AS *column-alias*]を指定すると、列エイリアスを列に割り当てます。

<query-specification>
埋め込み SELECT サブクエリを指定します。

参照項目: “サブクエリの概要” (51 ページ)

<sql-expression> [AS*column-alias*]
式から列名を導出します。

参照項目: “<sql-expression> 式” (918 ページ)

table*.
テーブル内のすべての列を選択します。

table-alias*.
テーブル内のすべての列を選択します。

参照項目: “テーブルエイリアス” (984 ページ)

***table.column* [AS*column-alias*]**
指定されたテーブルから単一の列を選択します。

アスタリスク(*)表記

アスタリスク(*)は、FROM 句にリストされている 1 つまたは複数のテーブルのすべての列を表します。アスタリスクの前にテーブル名が付いていない場合は、FROM 句のすべてのテーブルのすべての 列が含まれます。テーブル名が付いている場合(たとえば、*table*.*または *table-alias*.*)、そのテーブルのみからすべての 列が含まれます。

列エイリアス

列エイリアスは、列の一時的な代替名です。エイリアスは SELECT 句で指定され、結果セットの列に名前を付けたり名前を変更したりすることで、よりわかりやすく、読みやすくします。エイリアスは、算術式または要約関数の結果である列に名前を付けるためによく使用されます。エイリアスは 1 語のみです。

列エイリアスを他の列名と区別するには、キーワード AS が必要です。

列エイリアスはオプションであり、SELECT 句の各列名にエイリアスを含めることができます。列にエイリアスを割り当てた後、エイリアスを使用して他の句でその列を参照できます。

FROM 句

説明

(オプション)ソーステーブルを指定します。

構文

FROM <table-specification> [, ...<table-specification>]

<table-specification>::=

table [AS*table-alias*]

| <CONNECTION TO *catalog*> (<native-syntax>) [AS*alias*]

| (<query-specification>) [AS*alias*]

| <joined-table>

<joined-table>::=

<cross-join>

| <qualified-join>

| <natural-join>

<cross-join>::=

<table-specification> CROSS JOIN <table-specification>

<qualified-join>::=

<table-specification> [<join-type>] JOIN <table-specification> <join-specification>

<natural-join>::=

<table-specification> NATURAL [<join-type>] JOIN <table-specification>

<join-type>::=

INNER

| LEFT [OUTER]

| RIGHT [OUTER]

| FULL [OUTER]

<join-specification>::=

ON <search-condition>

| USING (*column* [, ...*column*])

引数

CONNECTION TO *catalog* (<native-syntax>) [[AS]*alias*]
SQL パススルー機能を使用して DBMS カタログからのデータを指定します。DBMS が理解できる SQL 構文は、FedSQL でその構文が無効な場合でも使用できます。詳細については、“FedSQL 明示的パススルー機能” (70 ページ)を参照してください。

CROSS JOIN
2 つのテーブルのデカルト積である結合を定義します。

参照項目: “クロス結合” (987 ページ)

JOIN
検索条件または特定の列を使用してデータをフィルタリングできるようにする結合を定義します。

参照項目: “限定結合” (987 ページ)

NATURAL JOIN
同じ名前と同じ型を共有する列の値が等しい 2 つのテーブルから行を選択する結合を定義します。

参照項目: “自然結合” (991 ページ)

(<query-specification>) [AS*alias*]
インラインビューとして機能する埋め込み SELECT サブクエリを指定します。*alias* は、インラインビューの一時的な名前を定義し、必須です。インラインビューを使用すると、プログラミングの手間が省けます。ビューを作成して別のクエリで参照するのではなく、FROM 句でビューをインラインで指定できます。

参照項目: “サブクエリの概要” (51 ページ)

table
テーブルの名前を指定します。名前は、“*table*” (941 ページ)で説明した任意の形式にすることができます。

table-alias
table の一時的な代替名を指定します。AS キーワードはオプションです。

INNER
1 番目のテーブルから、2 番目のテーブルの行と一致するサブセットのみが返されるように指定します。不一致行は両方のテーブルから破棄されます。

LEFT [OUTER]
一致する行と、1 番目のテーブルのうち 2 番目のテーブルのどの行とも一致しない行が返されるように指定します。

RIGHT [OUTER]
一致する行と、2 番目のテーブルのうち 1 番目のテーブルのどの行とも一致しない行が返されるように指定します。

FULL [OUTER]
1 番目と 2 番目のテーブルの一致行と不一致行がすべて返されるように指定します。

column
列の名前を指定します。

ON <search-condition>
あるテーブルと別のテーブルの行をマッチングするために使用される条件結合を指定します。検索条件が満たされると、一致する行が結果セットに追加されます。

参照項目: “<search-condition>” (1006 ページ)

USING (*column*[,...*column*])
内部結合または外部結合で使用する列を指定します。

参照項目: “ON 句と USING 句” (991 ページ)

概要

FROM 句を使用すると、ソーステーブルを指定できます。テーブル名を指定するか、クエリを DBMS にサブミットするか、埋め込み SELECT サブクエリを指定するか、結合を指定することにより、テーブルを参照できます。

テーブルエイリアス

テーブルエイリアスは、テーブルの一時的な代替名です。テーブルエイリアスは、1 つのテーブルの列を他のテーブルの列と区別するために結合で使用され、テーブル名を省略してクエリを読みやすくすることができます。列名が一致するテーブルを結合する場合は、テーブル名またはエイリアスを列名の前に付ける必要があります。*再帰結合*(テーブルのそれ自体との結合)の列名には、列がテーブルのどのコピーからのものであるかを区別するために、テーブルエイリアスを前に付ける必要があります。

テーブルエイリアスは FROM 句で定義されます。AS キーワードは、テーブルエイリアスとテーブル名を区別するために使用されます。次に例を示します。

```
select b.* from x as b;
```

列名でテーブルエイリアスを指定し、FROM 句でテーブルエイリアスを定義しない場合、FedSQL はエラーメッセージを返します。

テーブルエイリアスにエイリアスを指定することはできません。

結合されたテーブル

FROM 句に複数のテーブル指定がリストされている場合、それらは 1 つのテーブルを形成するように処理されます。結果セットには、各寄与テーブルからのデータが含まれています。これらのクエリは*結合*と呼ばれます。結合によって元のテーブルが変更されることはありません。

概念的には、2 つのテーブルが指定されている場合、テーブル A の各行がテーブル B のすべての行とマッチングされ、内部テーブルまたは中間テーブルが生成されます。中間テーブルの行数(*デカルト*)は、各ソーステーブルの行数の積に等しくなりま

す。中間テーブルはクエリの残り部分への入力になります。クエリでは、その行の一部を WHERE、ON、または USING 句で削除したり、関数で要約したりできます。

FedSQL 結合操作の概要については、“結合操作” (30 ページ)を参照してください。

返される行の指定

WHERE、ON、および USING 句には、結果セットでデカルト積の行を保持または削除する条件が含まれます。WHERE は、内部結合から行を選択するために使用されます。ON は、内部結合または外部結合から行を選択するために使用されます。USING は、結合に含める特定の列を選択するために使用されます。条件は、“結合されたテーブル” (984 ページ)で説明されている中間テーブルの各テーブルの各行に対して評価されます。その行の式の結果が真(非ゼロ、非欠損値、または null 値)である場合、その行は一致していると見なされます。

単純結合

結合の最も基本的なタイプは、SELECT ステートメントの FROM 句で複数のテーブルをカンマで区切った単純なリストです。次のクエリは、2 つのテーブル GrainProducts と Sales を結合します:

```
/* FedSQL code for simple join */
proc fedsql;
  select * from grainproducts, sales;
quit;
```

アウトプット 7.1　単純結合 - GrainProducts と Sales テーブル

PRODID	PRODUCT	PRODID	CUSTID	TOTALS	COUNTRY
1424	Rice	3234	1	$189,400	United States
3421	Corn	3234	1	$189,400	United States
3234	Wheat	3234	1	$189,400	United States
3485	Oat	3234	1	$189,400	United States
1424	Rice	1424	3	$555,789	Japan
3421	Corn	1424	3	$555,789	Japan
3234	Wheat	1424	3	$555,789	Japan
3485	Oat	1424	3	$555,789	Japan
1424	Rice	3421	4	$781,183	Japan
3421	Corn	3421	4	$781,183	Japan
3234	Wheat	3421	4	$781,183	Japan
3485	Oat	3421	4	$781,183	Japan
1424	Rice	3421	2	$2,789,654	United States
3421	Corn	3421	2	$2,789,654	United States
3234	Wheat	3421	2	$2,789,654	United States
3485	Oat	3421	2	$2,789,654	United States
1424	Rice	3975	5	$899,453	Argentina
3421	Corn	3975	5	$899,453	Argentina
3234	Wheat	3975	5	$899,453	Argentina
3485	Oat	3975	5	$899,453	Argentina

この方法でテーブルを結合すると、テーブルの *デカルト*積が返されます。最初のテーブルの各行は、2 番目のテーブルのすべての 行と結合されます。結果セットの行数は、最初のテーブルの行数に 2 番目のテーブルの行数を掛けた値に等しくなります。

単純結合のデカルト積は、意味のない大きなテーブルになる可能性があります。WHERE 句を使用して単純結合をサブセット化できます。このタイプの単純結合は、*等価結合*と呼ばれます。次のクエリは、ID 列をマッチングすることで前のテーブルをサブセット化し、アウトプット 7.64 (987 ページ)に示すテーブルを作成します。

```
/* FedSQL code for equijoin */
  select * from grainproducts, sales
    where grainproducts.prodid=sales.prodid;
```

等価結合では、比較は等価比較でなければなりません。AND 演算子を使用すると、複数の一致基準(ここには示されていません)を指定できます。複数の一致基準が指定されている場合、すべての等価性テストを満たす行のみが返されます。

```
proc fedsql;
  select * from grainproducts, sales
    where grainproducts.prodid=sales.prodid;
quit;
```

アウトプット 7.2　等価結合 - GrainProducts と Sales テーブル

PRODID	PRODUCT	PRODID	CUSTID	TOTALS	COUNTRY
3234	Wheat	3234	1	$189,400	United States
1424	Rice	1424	3	$555,789	Japan
3421	Corn	3421	4	$781,183	Japan
3421	Corn	3421	2	$2,789,654	United States

クロス結合

クロス結合は単純結合と同じように機能します。2 つのテーブルの積を返します。デカルト積と同様に、クロス結合の出力は WHERE 句によって制限できます。

次のクエリは同じ結果を生成します。

```
select * from grainproducts, sales;
```

```
select * from grainproducts cross join sales;
```

注: クロス結合では ON 句を使用しないでください。ON 句を使用すると、クロス結合が失敗します。ただし、WHERE 句を使用して出力をサブセット化できます。

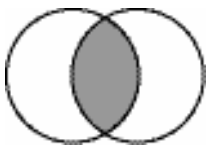
限定結合

限定結合によって、結果セットに表示される行を制御する簡単な方法が提供されます。ON または USING 句を使用して、結果セットをさらにサブセット化することもできます。

限定結合には、内部結合と外部結合の 2 つのタイプがあります。

内部結合

図 7.1　内部結合図



*内部結合*は、あるテーブル内の行のうち、別のテーブル内に 1 つ以上の一致する行があるすべての行の結果セットを返します。および GrainProducts と Sales テーブルを使用して、次のクエリは 2 つのテーブルの製品 ID 列を一致させ、アウトプット 7.65 (988 ページ)に示す結果セットを作成します。

```
proc fedsql;
```

```
    select *
      from grainproducts inner join sales
        on grainproducts.prodid=sales.prodid;
quit;
```

アウトプット 7.3　内部結合 - GrainProducts と Sales テーブル

PRODID	PRODUCT	PRODID	CUSTID	TOTALS	COUNTRY
3234	Wheat	3234	1	$189,400	United States
1424	Rice	1424	3	$555,789	Japan
3421	Corn	3421	4	$781,183	Japan
3421	Corn	3421	2	$2,789,654	United States

WHERE 句のかわりに ON または USING 句を使用して、テーブルを結合する列を指定できます。ただし、引き続き WHERE 句を使用してクエリ結果をサブセット化できます。

ON 句または USING 句を使用した内部結合は、FROM 句でテーブルをリストし、WHERE 句で結合列を指定(等価結合)するのと同じ機能を提供できることに注意してください。たとえば、これら 2 つのコードセットは、内部結合構造を使用します。

```
select *
  from grainproducts inner join sales
    on grainproducts.prodid=sales.prodid;
```

```
select *
  from grainproducts inner join sales
    using (prodid);
```

このコードは、前のコードと同じ出力を生成しますが、内部結合構造を使用します。

```
select *
  from grainproducts, sales
    where grainproducts.prodid=sales.prodid;
```

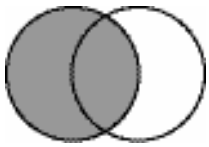
外部結合

*外部結合*は、結合内の他のテーブルのどの行とも一致しない 1 つのテーブルの行で拡張された内部結合です。結果セットには、結合のソーステーブルと一致する行と一致しない行が含まれます。一致しない行には、不一致テーブルの列に null または欠損値があります。WHERE 句のかわりに ON または USING 句を使用して、テーブルを結合する列を指定できます。ただし、引き続き WHERE 句を使用してクエリ結果をサブセット化できます。

外部結合には、左結合、右結合、および完全結合の 3 つのタイプがあります。

左外部結合

図 7.2　左外部結合の図



左外部結合は、一致する行と、FROM 句にリストされた 2 番目のテーブルのどの行にも一致しない FROM 句にリストされた最初のテーブルの行をリストします。GrainProducts と Sales テーブルを使用して、GrainProducts テーブルと Sales テーブルの一致行と、GrainProducts テーブルの不一致行を含むテーブルを作成します。GrainProducts テーブルの不一致行の Sales テーブルデータに欠損値が表示されていることに注意してください。

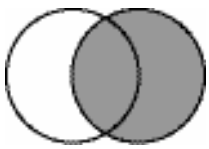
```
proc fedsql;
  select *
    from grainproducts left outer join sales
    on grainproducts.prodid=sales.prodid;
quit;
```

アウトプット 7.4　左外部結合 - GrainProducts と Sales テーブル

PRODID	PRODUCT	PRODID	CUSTID	TOTALS	COUNTRY
1424	Rice	1424	3	$555,789	Japan
3234	Wheat	3234	1	$189,400	United States
3421	Corn	3421	4	$781,183	Japan
3421	Corn	3421	2	$2,789,654	United States
3485	Oat	.	.	.	

右外部結合

図 7.3　右外部結合の図



右外部結合は、一致する行と、FROM 句にリストされた最初のテーブルのどの行にも一致しない FROM 句にリストされた 2 番目のテーブルの行をリストします。GrainProducts と Sales テーブルを使用して、GrainProducts テーブルと Sales テーブルの一致行と、Sales テーブルの不一致行を含むテーブルを作成します。Sales テーブルの不一致行の GrainProducts テーブルデータに欠損値が表示されていることに注意してください。

```
proc fedsql;
  select *
```

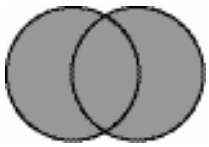
```
    from grainproducts right outer join sales
    on grainproducts.prodid=sales.prodid;
quit;
```

アウトプット 7.5　右外部結合 - GrainProducts と Sales テーブル

PRODID	PRODUCT	PRODID	CUSTID	TOTALS	COUNTRY
3234	Wheat	3234	1	$189,400	United States
1424	Rice	1424	3	$555,789	Japan
3421	Corn	3421	4	$781,183	Japan
3421	Corn	3421	2	$2,789,654	United States
.		3975	5	$899,453	Argentina

完全外部結合

図 7.4　完全外部結合の図



完全外部結合は、左外部結合と右外部結合を組み合わせたものです。結果セットには、左右のテーブルの一致行と不一致行の両方が含まれます。GrainProducts と Sales を使用して、GrainProducts テーブルと Sales テーブルの一致行と、GrainProducts テーブルと Sales テーブルの不一致行を含むテーブルを作成します。 不一致行のデータには欠損値が表示されていることに注意してください。

```
proc fedsql;
  select *
    from grainproducts full outer join sales
      on grainproducts.prodid=sales.prodid;
quit;
```

アウトプット 7.6　完全外部結合 - GrainProducts と Sales テーブル

PRODID	PRODUCT	PRODID	CUSTID	TOTALS	COUNTRY
1424	Rice	1424	3	$555,789	Japan
3234	Wheat	3234	1	$189,400	United States
3421	Corn	3421	4	$781,183	Japan
3421	Corn	3421	2	$2,789,654	United States
3485	Oat	.	.	.	
.		3975	5	$899,453	Argentina

ON 句と USING 句

結合の基になる条件を指定する式で ON 句を使用できます。条件式には任意の述語を含めることができますが、列名と比較演算子が最もよく使用されます。内部結合を含む ON 句は、WHERE 句と同等です。外部結合(左、右、または完全)を含む ON 句は、WHERE 句とは異なります。外部結合を含む ON 句は、行をフィルタリングしてから、null または欠損値を持つ不一致行を含めます。

USING 句を使用して、結果セットに含める 2 つの列のいずれかを指定できます。

ON 句と USING 句の違いは、ON 句では任意の条件式を使用できることです。USING 句は、常に列名が等しいことを意味します。たとえば、この ON 句は結果テーブルから **United States** を除外します。

```
proc fedsql;
  select *
    from grainproducts inner join sales
    on sales.country <> 'United States'
      AND grainproducts.prodid=sales.prodid;
quit;
```

アウトプット 7.7　内部結合 - 米国外の GrainProducts と Sales

PRODID	PRODUCT	PRODID	CUSTID	TOTALS	COUNTRY
1424	Rice	1424	3	$555,789	Japan
3421	Corn	3421	4	$781,183	Japan

自然結合

*自然結合*では、同じ名前と同じ型を共有する列の値が等しい 2 つのテーブルから行が選択されます。2 つの列の名前が同じで型が異なる場合、エラーが発生します。内部、左、右、完全のいずれかの自然結合を実行できます。自然結合を指定するときに *join-type* を省略すると、INNER が暗黙的に指定されます。類似列が見つからない場合は、クロス結合が実行されます。WHERE 句を使用して出力を制限できます。

GrainProducts と Sales テーブルを使用して、自然左外部結合を実行します。

```
proc fedsql;
  select *
    from grainproducts natural left outer join sales;
quit;
```

アウトプット 7.8　自然左外部結合 - GrainProducts と Sales テーブル

PRODID	PRODUCT	CUSTID	TOTALS	COUNTRY
1424	Rice	3	$555,789	Japan
3234	Wheat	1	$189,400	United States
3421	Corn	4	$781,183	Japan
3421	Corn	2	$2,789,654	United States
3485	Oat	.	.	

prodid 列が結果セットに 1 回だけ表示されることに注意してください。

注: 自然結合では ON 句を使用しないでください。ON 句を使用すると、自然結合が失敗します。自然結合を使用すると、ON 句が暗黙的に指定され、類似列がすべて一致します。

WHERE 句

説明

指定された検索条件に基づいて結果セットをサブセット化します。

構文

WHERE <search-condition>

引数

<search-condition>
　WHERE 句によって返される行の条件を指定します。

参照項目:　“<search-condition>” (1006 ページ)

詳細

WHERE 句には、結果セットに含めるために選択される行を指定する検索条件(オペランドまたは述語で区切られた 1 つ以上の式)が必要です。条件が満たされると(つまり、条件が true に解決されると)、その行が結果セットに表示されます。それ以外の場合、行は表示されません。

注: 1 つの列のみを指定する集計関数は使用できません。たとえば、次のコードは使用できません。

```
where max(inventory1)>10000;
```

ただし、この WHERE 句は使用できます。

```
where max(inventory1, inventory2)>10000;
```

注: 列に REAL または DOUBLE 値が含まれている場合は、WHERE 句で=および<>演算子を使用しないでください。REAL 値と DOUBLE 値は近似的な数値データ型であり、WHERE 句で=および<>演算子を使用すると、不正確な結果が生じる可能性があります。REAL 列と DOUBLE 列は、>または<演算子での比較に制限する必要があります。

GROUP BY 句

説明

データをグループ化して要約する方法を指定します。

構文

GROUP BY <grouping-column> [, ...<grouping-column>]

<grouping-column>::=

column [, ...*column*]

| *column-position-number*

| <sql-expression>

引数

column
列の名前または列エイリアスを指定します。

column-position-number
列の位置を表す非負の整数を指定します。

<sql-expression>
有効な SQL 式を指定します。

参照項目: "<sql-expression> 式" (918 ページ)

詳細

GROUP BY 句は、指定された 1 つまたは複数の列でデータをグループ化します。

グループ化する 1 つまたは複数の列で一部の行に欠損値または null 値が含まれている場合、SAS は、グループ化列で欠損値または null 値を持つすべての行を収集して 1 つのグループにします。

複数のグループ化列を指定して、より詳細なレポートを取得することができます。複数のグループ化列が指定されている場合、最初の列が主要なグループ化を決定します。

GROUP BY 句の列名のかわりに整数を使用できます。たとえば、グループ化列が 2 の場合、結果は 2 番目の列の値によってグループ化されます。整数のかわりに浮動小数点値(例: 2.3)を使用する場合、FedSQL が小数部分を無視することに注意してください。

式によって返される値によって出力をグループ化できます。たとえば、X が数値変数の場合、次の出力は X の値によってグループ化されます。

```
select x, sum(y)
  from table1
  group by x;
```

同様に、Y が文字変数の場合、次の出力は Y の値によってグループ化されます。

```
select sum(x), y
  from table1
  group by y;
```

GROUP BY 句を使用する場合、SELECT 句または HAVING 句で集計関数を使用して、各グループのデータを要約する方法を SAS に指示することもできます。集計関数なしで GROUP BY 句を使用すると、SAS は GROUP BY 句を ORDER BY 句であるかのように扱います。

ORDER BY 句を使用して、結果セットに表示される行の順序を指定できます。ORDER BY 句を指定しない場合、GROUP BY 句によって返されるグループは特定の順序ではありません。

注: SPD Engine データセットの場合、ユーティリティファイルは、余分なスペースを必要とする特定の操作に使用されます。GROUP BY 句にはユーティリティファイルが必要であり、SPDEUTILLOC=システムオプションはそのユーティリティファイルにスペースを割り当てます。詳細については、*SAS Scalable Performance Data Engine: リファレンス*を参照してください。

注: FedSQL GROUP BY 構文は、要約統計量の再マージをサポートしていません。GROUP BY クエリの選択リストは、グループ化列と集計関数のみを参照できます。GROUP BY 句で指定されていない列を SELECT リストで指定すると、次のエラーが発生します: column name must be grouped or used in an aggregate function.

ヒント　FedSQL では、サブクエリを使用して、分析の結果をデータの他の列と統合できます。サブクエリで分析を指定し、サブクエリの結果に基づいて残りの行を選択します。

サブクエリを使用してデータを再マージする例を次に示します。この要求は、サブクエリを使用して、Payroll テーブル内の各性別の最年長の従業員を特定し、外部クエリでサブクエリを参照します。サブクエリ(t)は、SELECT リストと結合条件の両方で参照されることに注意してください。

```
proc fedsql;
  select
  p.IdNumber, p.Gender, p.Jobcode, p.Salary, p.Birth, p.Hired
    from
    (select Gender, min(Birth) as min_birth
      from payroll
      group by Gender
      ) as t
    inner join payroll as p on p.Gender=t.gender and p.Birth=t.min_birth;
quit;
```

詳細については、“FedSQL で GROUP BY 再マージクエリを書き換える” (1180 ページ)を参照してください。

HAVING 句

説明

指定された検索条件に基づいてグループ化されたデータをサブセット化します。

構文

HAVING <search-condition>

引数

<search-condition>
HAVING 句によって返される行の条件を指定します。

参照項目: “<search-condition>” (1006 ページ)

詳細

HAVING 句には、結果セットに含めるために選択される行を指定する検索条件(オペランドまたは述語で区切られた 1 つ以上の式)が必要です。HAVING 句は、クエリ内の各グループに対して true または false として評価します。HAVING 句を GROUP BY 句とともに使用すると、グループ化されたデータをフィルタリングできます。HAVING 句は、WHERE 句が個々の行に影響を与えるのと同様の方法でグループに影響を与えます。

HAVING 句を含むクエリには、通常、GROUP BY 句または集計関数、あるいはその両方が含まれます。GROUP BY 句なしで HAVING 句を使用すると、SAS は HAVING 句を WHERE 句であるかのように扱います。

表 7.1 HAVING 句と WHERE 句の違い

HAVING 句の属性	WHERE 句の属性
通常、テーブルの行のグループを含めるかまたは除外するための条件を指定するために使用されます。	テーブルの個々の行を含めるかまたは除外するための条件を指定するために使用されます。
GROUP BY 句とともに使用する場合は、クエリで GROUP BY 句の後に置く必要があります。	GROUP BY 句とともに使用する場合は、クエリで GROUP BY 句の前に置く必要があります。
GROUP BY 句の影響を受けます。GROUP BY 句がない場合、HAVING 句は WHERE 句のように扱われます。	GROUP BY 句の影響を受けません。

HAVING 句の属性	WHERE 句の属性
GROUPBY 句とすべての 集計関数の後に処理されます。	GROUP BY 句がある場合はその前、およびすべての 集計関数の前に処理されます。

ORDER BY 句

説明

結果セットで行が返される順序を指定します。

構文

ORDER BY <sort-specification> [, ...<sort-specification>];

<sort-specification>::=

{*order-by-expression* [ASC | DESC]} [, ...*order-by-expression* [ASC | DESC]]

| {*order-by-expression* [COLLATE *collating-sequence-options*]}
[, ...*order-by-expression* [COLLATE *collating-sequence-options*]]

引数

order-by-expression
並べ替える列を指定します。並べ替え列は、次のいずれかになります。

column
列の名前または列エイリアスを指定します。

column-position-number
列の位置を表す非負の整数を指定します。

<sql-expression>
有効な SQL 式を指定します。

参照項目: “<sql-expression> 式” (918 ページ)

ASC
データを昇順に並べます。これがデフォルトの順序です。ASC または DESC が指定されていない場合、データは昇順で並べられます。

DESC
データを降順に並べます。

COLLATE *collating-sequence-options*
指定された言語の規則に従って文字を並べ替える言語照合を指定します。規則とデフォルトの照合順序オプションは、現在のロケール設定で指定されている言語に基づいています。この実装は International Components for Unicode (ICU)ライブラリによって 提供され、Unicode Collation Algorithms (UCA)とほぼ互換性のある結果を生成します。

collating-sequence-options 引数には、次のいずれかの値を指定できます。

DANISH | FINNISH | ITALIAN | NORWEGIAN | POLISH | SPANISH | SWEDISH
指定された言語に従って文字を並べ替えます。

LINGUISTIC [*collating-rules*]
collating-rules には次のいずれかの値を指定できます。

ALTERNATE_HANDLING=SHIFTED
スペース、句読点、記号などの可変文字の処理を制御します。このオプションが指定されていない場合(デフォルト値 NON_IGNORABLE を使用)、これらの可変文字間の違いは、文字間の違いと同じ重要性を持ちます。ALTERNATE_HANDLING オプションが指定されている場合、これらの可変文字はあまり重要ではありません。

デフォルト NON_IGNORABLE

ヒント SHIFTED 値は、多くの場合、Quaternary に設定された STRENGTH=と組み合わせて使用されます。このような場合、文字列を比較するときにスペース、句読点、および記号が考慮されますが、文字列の他のすべての側面(基本文字、アクセント、および大文字と小文字)が同一である場合に限ります。

CASE_FIRST=
大文字と小文字の順序を指定します。この引数は、TERTIARY、QUATERNARY、または IDENTICAL レベルでのみ有効です。次の表に、CASE_FIRST 引数の値と情報を示します。

表 7.2 CASE_FIRST=値

値	説明
UPPER	最初に大文字を並べ替え、次に小文字を並べ替えます。
LOWER	最初に小文字を並べ替え、次に大文字を並べ替えます。

COLLATION=
次の表に、使用可能な COLLATION=値を示します。照合順序値を選択しない場合、ユーザーのロケールのデフォルトの照合順序が選択されます。

表 7.3 COLLATION=値

値	説明
BIG5HAN	ラテン語のピンイン順序を指定し、中国語、日本語、および韓国語の文字の big5 文字セット順序を指定します。
DIRECT	ヒンディー語バリアントを指定します。

値	説明
GB2312HAN	ラテン語のピンイン順序を指定し、中国語、日本語、および韓国語の文字の gb2312han 文字セット順序を指定します。
PHONEBOOK	文字の順序付けの電話帳スタイルを指定します。ドイツ語のみの PHONEBOOK を選択します。
PINYIN	ピンインへの文字単位の音訳に基づいて、中国語、日本語、および韓国語の文字の順序を指定します。この順序は通常、簡体字中国語で使用されます。
POSIX	ポータブルオペレーティングシステムインターフェイスです。このオプションは、文字の"C"ロケール順序を指定します。
STROKE	文字のアルファベット以外の書体順序を指定します。中国語、日本語、韓国語、ベトナム語のいずれかで STROKE を選択します。この順序は通常、繁体字中国語で使用されます。
TRADITIONAL	文字の順序付けの伝統的なスタイルを指定します。たとえば、スペイン語では TRADITIONAL を選択します。

LOCALE= *locale_name*
ロケール名を POSIX 名(ja_JP など)の形式で指定します。詳細については、*SAS 各国語サポート(NLS): リファレンスガイド*を参照してください。

NUMERIC_COLLATION=
数値を表すために使用される文字ではなく、数値によってテキスト内の整数値を並べ替えます。

表 7.4 NUMERIC_COLLATION=値

値	説明
ON	数値で数字を順序付けします。たとえば、"8 Main St." は"45 Main St."の前に並べ替えられます。
OFF	文字値で数字を順序付けします。たとえば、"45 Main St."は"8 Main St."の前に並べ替えられます。

デフォルト OFF

STRENGTH=
強度の値は、照合レベルに関連しています。照合レベルの値は 5 つあります。次の表に、5 つのレベルに関する情報を示します。強度のデフォルト値はロケールに関連しています。

表 7.5 STRENGTH=値

値	照合の種類	説明
PRIMAR または 1	PRIMARY は、基本文字間の違いを指定します(たとえば、"a" < "b")。	最強の違いです。たとえば、辞書は基本文字によってさまざまなセクションに分割されます。
SECONDARY または 2	文字のアクセントは二次の違いと見なされます(たとえば、"as" < "às" < "at")。	文字列のどこかに一次の違いがある場合、二次の違いは無視されます。言語によっては、文字間のその他の違いも二次の違いと見なすことができます。
TERTIARY または 3	文字の大文字と小文字の違いは、三次レベルで区別されます(たとえば、"ao" < "Ao" < "aò")。	文字列のどこかに一次または二次の違いがある場合、三次の違いは無視されます。もう一つの例は、大小のかなの違いです。
QUATERNARY または 4	句読点がレベル 1-3 で無視される場合、追加のレベルを使用して、句読点のある単語とない単語を区別できます(たとえば、"ab" < "a-b" < "aB")。	句読点を無視する必要がある場合、または日本語テキストを処理する場合は、四次レベルを使用する必要があります。この違いは、一次、二次、または三次の違いがある場合は無視されます。
IDENTICAL または 5	他のすべてのレベルが等しい場合、同一のレベルがタイブレーカーとして使用されます。レベル 1-4 で違いがない場合に備えて、各文字列の正規化形式 D (NFD)形式の Unicode コードポイント値がこのレベルで比較されます。	このレベルは慎重に使用する必要があります。これは、2 つの文字列間のコードポイント値の違いのみが非常にまれにしか発生しないためです。たとえば、このレベルではヘブライ語のカンティレーションマークのみが識別されます。

別名 LEVEL=

ヒント *collating-rules* はかっこで囲む必要があります。複数の照合規則を指定できます。

参照項目: 付録 8, “ICU ライセンス” (1299 ページ)

*SAS 各国語サポート(NLS): リファレンスガイド*の言語照合に関するセクション

Unicode Collation Algorithm (UCA)仕様については、*http://www.unicode.org* を参照してください。

詳細

ORDER BY 句は、クエリで指定された順序に従ってクエリ式の結果を並べ替えます。この句を使用すると、デフォルトの順序は、最小値から最大値へと昇順になります。

ORDER BY 句を省略した場合、クエリされたテーブルで行が検出される順序など、出力行の特定の順序は保証されません。ORDER BY 句がない場合、出力行の順序は、FedSQL の内部処理、SAS のデフォルト照合順序、および動作環境によって決定されます。したがって、結果セットを特定の順序で表示する場合は、ORDER BY 句を使用します。

複数の *order-by-expression* が(カンマで区切って)指定されている場合、最初の指定によって主要並べ替え順序が決定されます。

ORDER BY 句の列名のかわりに整数を使用できます。たとえば、*order-by-expression* が 2 の場合、結果は 2 列目の値によって順序指定されます。整数のかわりに浮動小数点値(例: 2.3)を使用すると、FedSQL がエラーメッセージを発行することに注意してください。

ORDER BY 句では、クエリの SELECT 句に列が含まれているかどうかに関係なく、クエリ式の FROM 句で指定されたテーブルの任意の列を指定できます。ただし、SELECT DISTINCT が指定されている場合、または SELECT ステートメントに UNION セット演算子が含まれている場合は、クエリの SELECT 句に並べ替え列を含める必要があります。

注: SAS の欠損値または null 値は、可能な限り最も低い値として扱われます。

注: SPD Engine データセットの場合、ユーティリティファイルは、余分なスペースを必要とする特定の操作に使用されます。ORDER BY 句にはユーティリティファイルが必要であり、Base SAS UTILLOC システムオプションはそのユーティリティファイルにスペースを割り当てます。詳細については、*SAS Scalable Performance Data Engine: リファレンス*を参照してください。

LIMIT 句

説明

SELECT ステートメントが返す行数を指定します。

構文

LIMIT {*count* | ALL}

引数

count
SELECT ステートメントが返す行数を指定します。

ヒント *count* は、整数か、または整数値に解決される任意の単純な式になります。

ALL
すべての行が返されるように指定します。

詳細

LIMIT 句は、単独で使用することも、OFFSET 句と組み合わせて使用することもできます。OFFSET 句は、SELECT ステートメントが行を返し始める前にスキップする行の数を指定します。

注: LIMIT 句を使用する場合は、ORDER BY 句を使用して順序指定をすることをお勧めします。そうしないと、クエリの行の予測不可能なサブセットを取得する可能性があります。

OFFSET 句

説明

SELECT ステートメントが行を返し始める前にスキップする行の数を指定します。

構文

OFFSET *number*

引数

number
スキップする行数を指定します。

ヒント *number* は、整数か、または整数値に解決される任意の単純な式になります。

詳細

OFFSET 句は、単独で使用することも、LIMIT 句と組み合わせて使用することもできます。OFFSET 句は、SELECT ステートメントが行を返し始める前にスキップする行の数を指定します。

注: OFFSET 句を使用する場合は、ORDER BY 句を使用して順序指定をすることをお勧めします。そうしないと、クエリの行の予測不可能なサブセットが取得されます。

UNION 演算子

説明

2 つのクエリの結果を単一の結果セットに結合します。

構文

{<query-specification> | <query-expression>}
UNION
{
[ALL] [CORRESPONDING [BY (column[, ...column])]]
}
{<query-specification> | <query-expression>}

引数

<query-specification> | <query-expression>
仮想結果セットを生成する 1 つ以上の SELECT ステートメントを指定します。

参照項目: “SELECT ステートメント” (977 ページ)

“サブクエリの概要” (51 ページ)

UNION
複数の結果セットを結合して単一の結果セットとして返すように指定します。

ALL
(オプション)重複を含むすべての行を結果セットに含めるように指定します。

CORRESPONDING
(オプション)結果セットの列を列名または列リストで結合するように指定します。

BY (column[,... column])
(オプション)これらの列のみが結果セットに含まれるように指定します。列は列名で識別されます。

要件 列名は両方の入力テーブルに存在する必要があります。

詳細

UNION セット演算子は、2 つのクエリの結果として得られる一意の行をすべて 含む結果セットを生成します。つまり、結果セットには、最初のクエリと 2 番目のクエリ(および UNION 演算子で区切られたオプションの後続のクエリ)によって生成された行が含まれます。UNION セット演算子は、結果セットから重複する行を自動的に削除します。オプションの ALL キーワードを使用すると、重複する行を保持し、実行を 1 ステップ減らすことができるため、クエリのパフォーマンスが向上します。一意の行だけでなく、クエリの結果のすべての行を表示する場合に使用します。

列は、列名に関係なく、デフォルトでテーブル内の位置によって追加されます。ただし、対応する列のデータ型は、一致するか、または長さが唯一の差異である対応するデータ型である必要があります。一致しない場合、和集合は発生しません。つまり、CHAR は CHAR または VARCHAR とマージでき、INTEGER は BIGINT、INTEGER、SMALLINT、TINYINT とマージできます。ただし、CHAR も INTEGER も DOUBLE とマージすることはできません。

長さの異なる列をマージする場合は、長い方の列の長さが使用されます。たとえば、ある列が CHAR(4)で、対応する列が CHAR(10)の場合、CHAR(10)が使用されます。

オプションの CORRESPONDING キーワードを使用すると、FedSQL は位置ではなく列名で列をマッチングします。名前のマッチングでは大文字と小文字は区別されません。出力形式は、最初のクエリで作成されたものでない限り無視されます。列名が不一致の列はすべて無視されます。マッチングする列が、列名は同じであるが、入力テーブルの異なる位置に存在する場合は、CORRESPONDING キーワードを使用します。

列リストのない CORRESPONDING は、同じ名前を持つすべての列を含む結果セットを形成します。BY キーワードは、返される列を、列リストで名前が指定されている列に制限します。元のクエリで SELECT *が指定されている場合でも、列リストで指定されていない列は削除されます。BY キーワードを使用して、両方のクエリの 1 つ以上の列のみを結果セットに含めるように指定します。

結果セットの列の名前は、最初のクエリ式またはクエリ指定で列(式など)に名前がない場合を除いて、最初のクエリ式またはクエリ指定の列の名前になります。このとき、使用される名前は、SELECT ステートメントの結果セットがクライアントに返されるかどうか、または SELECT ステートメントが CREATE TABLE AS ステートメントの一部であるかどうかによって異なります。SELECT ステートメントの結果セットがクライアントに返されるとき、結果テーブル内の名前のない列の名前は“column”になります。CREATE TABLE AS ステートメントの名前のない列には、生成された名前が付けられます。生成される名前は、_expr_1、_expr_2 などです。

関連項目

“クエリ式とサブクエリ” (46 ページ)

EXCEPT 演算子

説明

2 つのクエリの結果を、最初のクエリに固有の行のみを含む単一の結果セットに結合します。

構文

```
{<query-specification> | <query-expression>}
    EXCEPT
        {
            [ALL][CORRESPONDING[BY (column [, ...column])]]
          }
{<query-specification> | <query-expression>}
```

引数

<query-specification> | <query-expression>
仮想結果セットを生成する 1 つ以上の SELECT ステートメントを指定します。

参照項目: “SELECT ステートメント” (977 ページ)

“サブクエリの概要” (51 ページ)

EXCEPT
複数の結果セットを結合し、最初の結果セットに含まれ、2 番目の結果セットには含まれない行のみを含めることを指定します。

ALL
(オプション)重複を含むすべての行を結果セットに含めるように指定します。

CORRESPONDING
(オプション)結果セットの列を列名または列リストで結合するように指定します。

BY (*column*[,...*column*])
(オプション)これらの列のみが結果セットに含まれるように指定します。列は列名で識別されます。

要件　列名は両方の入力テーブルに存在する必要があります。

詳細

EXCEPT セット演算子は、2 番目のクエリの結果にも存在する行を除いて、最初のクエリの結果に存在する行を含む結果セットを生成します。EXCEPT 演算子を使用して、最初のクエリに固有の結果を識別します。最初のクエリの中間結果に、2 番目のクエリの中間結果にない行が少なくとも 1 つある場合、その行(最初のクエリの行)が結果セットに含まれます。

EXCEPT セット演算子は、結果セットから重複する行を自動的に削除します。オプションの ALL キーワードを使用すると、重複する行を保持し、実行を 1 ステップ減らすことができるため、クエリのパフォーマンスが向上します。一意の行だけでなく、クエリの結果のすべての行を表示する場合に使用します。

列は、列名に関係なく、デフォルトで入力テーブル内の位置に基づいて評価されます。ただし、対応する列のデータ型は、一致するか、または長さが唯一の差異である対応するデータ型である必要があります。そうでない場合、比較は行われません。つまり、CHAR は CHAR または VARCHAR に対して評価でき、INTEGER は BIGINT、INTEGER、SMALLINT、TINYINT のいずれかに対して評価できます。ただし、CHAR または INTEGER は DOUBLE に対して評価できません。

長さの異なる列をマージする場合は、長い方の列の長さが使用されます。

オプションの CORRESPONDING キーワードを使用すると、FedSQL は位置ではなく列名で列をマッチングします。名前のマッチングでは大文字と小文字は区別されません。列名は、最初のクエリで返された順序で表示されます。出力形式は、最初のクエリで作成されたものでない限り無視されます。名前が不一致の列はすべて無視されます。マッチングする列が、列名は同じであるが、入力テーブルの異なる位置に存在する場合は、CORRESPONDING キーワードを使用します。

列リストのない CORRESPONDING は、同じ名前を持つすべての列を含む結果セットを形成します。BY キーワードは、返される列を、列リストで名前が指定されている列に制限します。元のクエリで SELECT *が指定されている場合でも、列リストで指定されていない列は削除されます。BY キーワードを使用すると、両方のクエリから 1 つ以上の列のみを結果セットで返すことができます。

結果セットの列の名前は、最初の query-expression または query-specification で列(式など)に名前がない場合を除いて、最初の query-expression または query-

specification の列の名前になります。このような場合、結果セット内のその列の名前は"column"です。

関連項目

“クエリ式とサブクエリ” (46 ページ)

INTERSECT 演算子

説明

2 つのクエリの結果を、クエリに共通の行のみを含む単一の結果セットに結合します。

構文

{<query-specification> | <query-expression>}
    INTERSECT
        {
            [ALL][CORRESPONDING [BY (*column* [, ...*column*)]]]
          }
{<query-specification> | <query-expression>}

引数

<query-specification> | <query-expression>
仮想テーブルを生成する 1 つ以上の SELECT ステートメントを指定します。

参照項目: “SELECT ステートメント” (977 ページ)

“サブクエリの概要” (51 ページ)

INTERSECT
複数の結果セットを結合し、結果セットに共通の行のみを含めることを指定します。

ALL
(オプション)重複を含むすべての行を結果セットに含めるように指定します。

CORRESPONDING
(オプション)結果セットの列を列名または列リストで結合するように指定します。

BY (*column*[,...*column*])
(オプション)これらの列のみが結果セットに含まれるように指定します。列は列名で識別されます。

要件 列名は両方の入力テーブルに存在する必要があります。

詳細

INTERSECT 演算子は、2 つのクエリに共通の行を含む結果セットを生成します。

INTERSECT セット演算子は、結果セットから重複する行を自動的に削除します。オプションの ALL キーワードを使用すると、重複する行を保持し、実行を 1 ステップ減らして、クエリのパフォーマンスを向上させることができます。一意の行だけでなく、クエリの結果のすべての行を表示する場合に使用します。

列は、列名に関係なく、デフォルトで入力テーブル内の位置に基づいて評価されます。ただし、対応する列のデータ型は、一致するか、または長さが唯一の差異である対応するデータ型である必要があります。そうでない場合、比較は行われません。つまり、CHAR は CHAR または VARCHAR に対して評価でき、INTEGER は BIGINT、INTEGER、SMALLINT、TINYINT のいずれかに対して評価できます。ただし、CHAR または INTEGER は DOUBLE に対して評価できません。

長さの異なる列をマージする場合は、長い方の列の長さが使用されます。

オプションの CORRESPONDING キーワードを使用すると、FedSQL は位置ではなく列名で列をマッチングします。名前のマッチングでは大文字と小文字は区別されません。列名は、最初のクエリで返された順序で表示されます。出力形式は、最初のクエリで作成されたものでない限り無視されます。名前が不一致の列はすべて無視されます。マッチングする列が、列名は同じであるが、入力テーブルの異なる位置に存在する場合は、CORRESPONDING キーワードを使用します。

列リストのない CORRESPONDING は、同じ名前を持つすべての列を含む結果セットを形成します。BY キーワードは、返される列を、列リストで名前が指定されている列に制限します。元のクエリで SELECT *が指定されている場合でも、列リストで指定されていない列は削除されます。BY キーワードを使用すると、両方のクエリから 1 つ以上の列のみを結果セットで返すことができます。

結果セットの列の名前は、最初の query-expression または query-specification で列(式など)に名前がない場合を除いて、最初の query-expression または query-specification の列の名前になります。このような場合、結果セット内のその列の名前は"column"です。

関連項目

“クエリ式とサブクエリ” (46 ページ)

<search-condition>

説明

結果セットに含めるために選択される行を指定する、1 つ以上の演算子と述部の組み合わせです。

構文

<search-condition>::=

{

[NOT] {<sql-expression> | (<search-condition>)}

```
        [{AND | OR} [NOT] {<sql-expression> | (<search-condition>)}]
    }
      [, ...{[NOT] {<sql-expression> | (<search-condition>)}
        [{AND | OR} [NOT] {<sql-expression> | (<search-condition>)}]}]
```

<sql-expression>::=

expression {*operator* | *predicate*} *expression*

引数

NOT

ブール条件を否定します。この表は、NOT 演算子を使用して true 値と false 値を比較したときの結果の概要を示しています。

表 7.6 NOT 演算子の真理表

NOT	結果
True	False
False	True
Unknown	Unknown

AND

両方の条件を満たすオブザベーションを見つけることにより、2 つの条件を組み合わせます。この表は、AND 演算子を使用して TRUE 値と FALSE 値を比較したときの結果の概要を示しています。

表 7.7 AND 演算子の真理表

AND	*True*	*False*	*Unknown*
True	True	False	Unknown
False	False	False	False
Unknown	Unknown	False	Unknown

OR

いずれかまたは両方を満たすオブザベーションを見つけることにより、2 つの条件を組み合わせます。この表は、OR 演算子を使用して TRUE 値と FALSE 値を比較したときの結果の概要を示しています。

表 7.8 OR 演算子の真理表

OR	*True*	*False*	*Unknown*
True	True	True	True

OR	*True*	*False*	*Unknown*
False	True	False	Unknown
Unknown	True	Unknown	Unknown

<sql-expression>
有効な SQL 式を指定します。

参照項目: “<sql-expression> 式” (918 ページ)

詳細

検索条件では、SELECT ステートメントの結果セットで返される行が指定されます。SELECT ステートメント内で、検索条件は WHERE 句、HAVING 句、および結合を含む ON 句で使用されます。

論理演算子の優先順位は、NOT、AND、OR の順になりますが、かっこを使用して順序をオーバーライドすることができます。かっこ内のすべてが最初に評価されて単一の値が生成されてから、その値をかっこ外の演算子で使用できるようになります。

REAL 値が検索条件に含まれる場合、精度の問題が発生する可能性があります。これらの問題を回避するには、条件内の値のキャストを定義または作成することをお勧めします。つまり、元のクエリで値が使用されている場合は常に、この構文でキャストを強制します。

CAST (*value* AS *type)*

ここではいくつかの例を示します。

```
select * from test where x = cast (1.0e20 as real);
select cast (1.0e20 as real) from test;
select cast (col1 as real) from test;
```

注: 検索条件は、“UPDATE ステートメント” (1009 ページ)と“DELETE ステートメント” (960 ページ)でも使用されます。UPDATE ステートメントの場合、検索指定は更新される行を指定します。DELETE ステートメントの場合、検索指定は削除する行を指定します。

関連項目

概念:

- “FedSQL 明示的パススルー機能” (70 ページ)

ステートメント:

- “DELETE ステートメント” (960 ページ)
- “UPDATE ステートメント” (1009 ページ)

UPDATE ステートメント

テーブルの既存の行の列の値を変更します。

カテゴリ: データ操作

制限事項: このステートメントは、CAS で実行される FedSQL プログラムではサポートされていません。

データソース: SAS データセット、SPD Engine データセット、SPD Server テーブル、Amazon Redshift、DB2、Google BigQuery、Greenplum、JDBC、MDS、Microsoft SQL Server、MongoDB、MySQL、Netezza、ODBC、Oracle、PostgreSQL、SAP HANA、SAP IQ、SingleStore、Snowflake、Teradata、Vertica、Yellowbrick

構文

UPDATE *table*

{

SET *column*=<sql-expression> [, ...*column*=<sql-expression>]

| *column*=*value-expression* [, ...*column*=*value-expression*]

}

[WHERE <sql-expression> | *value-expression*];

引数

table
テーブル名を指定します。

制限事項 Teradata テーブルには、UPDATE ステートメントで更新される主キーが定義されている必要があります。

column
列名を指定します。

WHERE <*sql-expression*>
有効な SQL 式を指定します。

注 MongoDB サブクエリでは、単一変数演算子のみが許可されます。FedSQL は、サブクエリを内部クエリとして使用して、MongoDB 子テーブルまたはメタデータテーブルのデータを更新することをサポートしていません。

参照項目: “<sql-expression> 式” (918 ページ)

value-expression
有効な値式を指定します。

ヒント　INSERT ステートメントでパラメーター配列を使用できます。

詳細

UPDATE ステートメントは、テーブル内の既存の行のすべてまたは一部の値を変更します。

SET 句は、変更する列とそれらに割り当てる値を指定します。SET でない列は、以前の値を保持します。SET 句では、等号の左側の列参照は、等号の右側の式の一部として表示することもできます。SET 句は、式の列の現在の値を使用します。たとえば、この式を使用して、従業員に$1,000 のホリデーボーナスを与えることができます。

```
update payroll set salary = salary + 1000;
```

WHERE 句を使用すると、更新する行を選択できます。WHERE 句を省略すると、すべての行が更新されます。WHERE 句を使用すると、WHERE 句の条件を満たす行のみが更新されます。

注: テーブルに対して行レベルのアクセス許可が有効になっている場合、テーブル内の行を更新することはできません。行レベルのセキュリティは、SAS Federation Server でのみ使用できます。行レベルのセキュリティの詳細については、SAS Federation Server のドキュメントを参照してください。

比較

DELETE ステートメントを使用すると、テーブルから行を削除できます。INSERT ステートメントを使用すると、新しい行をテーブルに挿入できます。UPDATE ステートメントを使用すると、テーブル内の行を変更できます。

関連項目

ステートメント:

- “DELETE ステートメント” (960 ページ)
- “INSERT ステートメント” (972 ページ)
- “SELECT ステートメント” (977 ページ)の WHERE 句

8

FedSQL ステートメントテーブルオプション

ステートメントテーブルオプションの概要 **1014**
FedSQL ステートメントテーブルオプションについて 1014
制限 1014

テーブルオプションと他のタイプのオプションとの相互作用 **1014**

FedSQL ステートメントテーブルオプションの構文 **1015**

BULKLOAD テーブルオプションについて **1016**

データソース別の FedSQL ステートメントテーブルオプション **1016**

ディクショナリ **1032**
ALTER= テーブルオプション 1032
ASYNCINDEX= テーブルオプション 1033
BL_ACCOUNTNAME= テーブルオプション 1034
BL_ALLOW_READ_ACCESS= テーブルオプション 1035
BL_APPLICATIONID= テーブルオプション 1036
BL_AWS_CONFIG_FILE= テーブルオプション 1037
BL_AWS_CREDENTIALS_FILE= テーブルオプション 1037
BL_AZURE_SAS= テーブルオプション 1038
BL_AZURE_TENANTID= テーブルオプション 1039
BL_BUCKET= テーブルオプション 1040
BL_COMPRESS= テーブルオプション 1041
BL_COPY_LOCATION= テーブルオプション 1041
BL_CPU_PARALLELISM= テーブルオプション 1042
BL_DATA_BUFFER_SIZE= テーブルオプション 1043
BL_DEFAULT_DIR= テーブルオプション 1045
BL_DELETE_DATAFILE= テーブルオプション 1046
BL_DELIMITER= テーブルオプション 1047
BL_DISK_PARALLELISM= テーブルオプション 1047
BL_ERRORS= テーブルオプション 1048
BL_EXCEPTION= テーブルオプション 1049
BL_FILESYSTEM= テーブルオプション 1050
BL_FOLDER= テーブルオプション 1051
BL_INDEXING_MODE= テーブルオプション 1052

BL_INTERNAL_STAGE= テーブルオプション ... 1053
BL_KEY= テーブルオプション ... 1054
BL_LOAD= テーブルオプション ... 1055
BL_LOAD_REPLACE= テーブルオプション ... 1055
BL_LOG= テーブルオプション ... 1056
BL_LOGFILE= テーブルオプション ... 1057
BL_NUM_DATAFILES= テーブルオプション ... 1058
BL_OPTIONS= テーブルオプション ... 1059
BL_PARALLEL= テーブルオプション ... 1060
BL_PORT_MAX= テーブルオプション ... 1061
BL_PORT_MIN= テーブルオプション ... 1061
BL_RECOVERABLE= テーブルオプション ... 1062
BL_REGION= テーブルオプション ... 1063
BL_REMOTE_FILE= テーブルオプション ... 1064
BL_SECRET= テーブルオプション ... 1065
BL_SKIP= テーブルオプション ... 1066
BL_SKIP_INDEX_MAINTENANCE= テーブルオプション ... 1066
BL_SKIP_UNUSABLE_INDEXES= テーブルオプション ... 1067
BL_TIMEOUT= テーブルオプション ... 1068
BL_TOKEN= テーブルオプション ... 1069
BL_USE_ESCAPE= テーブルオプション ... 1070
BL_WARNING_COUNT= テーブルオプション ... 1071
BUFNO= テーブルオプション ... 1072
BUFSIZE= テーブルオプション ... 1073
BULKLOAD= テーブルオプション ... 1074
BULKOPTS= テーブルオプション ... 1079
COMPRESS= テーブルオプション ... 1082
CONFIG= テーブルオプション ... 1084
DATAFILE= テーブルオプション ... 1085
DBCREATE_INDEX_OPTS= テーブルオプション ... 1085
DBCREATE_TABLE_OPTS= テーブルオプション ... 1086
ENCODING= テーブルオプション ... 1090
ENCRYPT= テーブルオプション ... 1091
ENCRYPTKEY= テーブルオプション ... 1094
ENDOBS= テーブルオプション ... 1097
FETCH_NUMERIC_TYPE= テーブルオプション ... 1098
GP_DISTRIBUTED_BY= テーブルオプション ... 1099
IDXNAME= テーブルオプション ... 1100
IDXWHERE= テーブルオプション ... 1102
IOBLOCKSIZE= テーブルオプション ... 1104
KMS_PATH= テーブルオプション ... 1105
LABEL= テーブルオプション ... 1105
LOCATION= テーブルオプション ... 1107
LOCKTABLE= テーブルオプション ... 1108
ORHINTS= テーブルオプション ... 1109
ORNUMERIC= テーブルオプション ... 1110
PADCOMPRESS= テーブルオプション ... 1111
PARTSIZE= テーブルオプション ... 1112
PASSWORD= テーブルオプション ... 1113
PICKLIST= テーブルオプション ... 1113
POINTOBS= テーブルオプション ... 1114
POST_TABLE_OPTS= テーブルオプション ... 1116
PRE_TABLE_OPTS= テーブルオプション ... 1117
PW= テーブルオプション ... 1118

READ=テーブルオプション テーブルオプション ... 1119
READMODE= テーブルオプション ... 1120
REUSE= テーブルオプション ... 1120
RS_AWS_CONFIG= テーブルオプション ... 1121
RS_BUCKET= テーブルオプション ... 1122
RS_CREDENTIALS_FILE= テーブルオプション ... 1123
RS_DEFAULT_DIR= テーブルオプション ... 1123
RS_DELETE_DATAFILE= テーブルオプション ... 1124
RS_KEY= テーブルオプション ... 1125
RS_REGION= テーブルオプション ... 1125
RS_SECRET= テーブルオプション ... 1126
RS_TOKEN= テーブルオプション ... 1127
SCANSTRINGCOLUMNS= テーブルオプション ... 1127
STARTOBS= テーブルオプション ... 1128
TABLE_TYPE= テーブルオプション ... 1129
TD_BUFFER_MODE= テーブルオプション ... 1130
TD_CHECKPOINT= テーブルオプション ... 1131
TD_DATA_ENCRYPTION= テーブルオプション ... 1132
TD_DROP_ERROR_TABLE= テーブルオプション ... 1133
TD_DROP_LOG_TABLE= テーブルオプション ... 1133
TD_DROP_WORK_TABLE= テーブルオプション ... 1134
TD_ERROR_LIMIT= テーブルオプション ... 1135
TD_ERROR_TABLE_1= テーブルオプション ... 1136
TD_ERROR_TABLE_2= テーブルオプション ... 1137
TD_LOG_MECH_TYPE= テーブルオプション ... 1138
TD_LOG_MECH_DATA= テーブルオプション ... 1139
TD_LOG_TABLE= テーブルオプション ... 1140
TD_LOGDB= テーブルオプション ... 1141
TD_MAX_SESSIONS= テーブルオプション ... 1142
TD_MIN_SESSIONS= テーブルオプション ... 1143
TD_NOTIFY_LEVEL= テーブルオプション ... 1143
TD_NOTIFY_METHOD= テーブルオプション ... 1144
TD_NOTIFY_STRING= テーブルオプション ... 1145
TD_PACK= テーブルオプション ... 1146
TD_PACK_MAXIMUM= テーブルオプション ... 1147
TD_PAUSE_ACQ= テーブルオプション ... 1147
TD_SESSION_QUERY_BAND= テーブルオプション ... 1148
TD_TENACITY_HOURS= テーブルオプション ... 1149
TD_TENACITY_SLEEP= テーブルオプション ... 1149
TD_TPT_OPER= テーブルオプション ... 1150
TD_TRACE_LEVEL=, TD_TRACE_LEVEL_INF= テーブルオプション ... 1152
TD_TRACE_OUTPUT= テーブルオプション ... 1153
TD_UNICODE_PASSTHRU= テーブルオプション ... 1154
TD_WORK_TABLE= テーブルオプション ... 1155
TD_WORKING_DB= テーブルオプション ... 1156
THREADNUM= テーブルオプション ... 1157
TYPE= テーブルオプション ... 1158
UNIQUESAVE= テーブルオプション ... 1158
USER= テーブルオプション ... 1159
WHERENOINDEX= テーブルオプション ... 1160
WRITE=テーブルオプション テーブルオプション ... 1161

ステートメントテーブルオプションの概要

FedSQL ステートメントテーブルオプションについて

FedSQL ステートメントテーブルオプションは、一緒に記述されるテーブルにのみ適用されるアクションを指定します。これにより、SAS ファイルのパスワードの指定や列の名前変更などの操作を実行できます。

CREATE TABLE、ALTER TABLE、または SELECT ステートメントなど、テーブル名を指定する FedSQL ステートメントでテーブルオプションを指定できます。

制限

FedSQL ステートメントオプションの可用性と動作は、データソース固有のものです。

テーブルオプションと他のタイプのオプションとの相互作用

多くのタイプのオプションは同じ名前を共有し、同じ機能を持っています。たとえば、COMPRESS=テーブルオプションを接続文字列オプションおよび FedSQL ステートメントテーブルオプションとして指定することで、SAS データセットを圧縮できます。

同じ機能のオプションが複数タイプ指定されている場合、ソフトウェアは次の優先順位に従います。

1 FedSQL ステートメントテーブルオプション

2 データソース接続文字列オプション

たとえば、接続文字列に COMPRESS=NO が含まれ、FedSQL ステートメントにテーブルオプション COMPRESS=YES が含まれている場合、テーブルオプションは、指定されたテーブルの接続文字列の指定をオーバーライドします。

Base SAS セッションで FedSQL ステートメントをサブミットする場合、次の優先順位が適用されます。

1 FedSQL ステートメントオプション

2 LIBNAME ステートメントオプション

3 データソース接続文字列オプション

4 システムオプション

つまり、FedSQL ステートメントテーブルオプションは、データソース接続文字列オプションをオーバーライドする LIBNAME ステートメントオプションをオーバーライドします。

FedSQL ステートメントテーブルオプションの構文

FedSQL ステートメントのテーブルオプションは、テーブル名の直後に、中かっこ(つまり、{ })で囲み、キーワード OPTIONS を含めて指定します。複数のテーブルオプションを指定するには、スペースまたはカンマで区切ります。

注意
テーブルオプションの構文を指定する際、左中かっこ{と OPTIONS キーワードの間にスペースを 入れることはできません。スペースがあると 構文エラーになります。
たとえば、次のステートメントは正しいです。

```
create table temp {options dbcreate_table_opts='primary index(b)'}
(a int, b int);
```

ただし、次のステートメントは正しくなく、エラーメッセージが表示されます。

```
create table temp { options dbcreate_table_opts='primary index(b)'}
(a int, b int);
```

{OPTIONS *option-1=value* [... *option-n=value*]}

これらの例は、FedSQL ステートメントのテーブルオプションを示しています。

```
create table salary {options encrypt=yes read=green};

select * from salary {options read=green};
```

BULKLOAD テーブルオプションについて

バルクロードに関連するテーブルオプションは、ドライバーがネイティブ SQL 言語を使用する場合とは対照的に、FedSQL 言語が呼び出しを処理する場合にのみ使用できます。FedSQL 言語プロセッサは、ネイティブ SQL よりも使用され、テーブルを作成したり、複数のデータソースからデータを挿入したりする要求を処理します。テーブルオプションを適用するには、それらをサポートするデータソースをクエリの CREATE TABLE または INSERT INTO 部分で指定する必要があります。FedSQL 言語プロセッサは、要求で SAS 関数が指定されている場合に、単一データソースの CREATE TABLE および INSERT ステートメントを処理するためにも使用されます。

バルクロードは、BULKLOAD=テーブルオプションで開始されます。データソース固有のバルクロードテーブルオプションは、BULKOPTS=テーブルオプションで指定されます。

データソース別の FedSQL ステートメントテーブルオプション

次の表は、FedSQL プログラムでサポートされているテーブルオプションを、それをサポートするデータソース別にリストにしたものです。オプションは、データソースごとにアルファベット順にリストされています。

注: 2026.08 以降、Amazon Redshift データソースの BULKOPTS= オプションは、他のデータソースとの一貫性を保つために、該当する BL テーブルオプションを受け入れます。下位互換性のために、RS テーブルオプションは引き続きサポートされています。各 RS テーブルオプションは、同等の BL オプションのエイリアスになりました。Amazon Redshift の場合、次の表に、該当する BL テーブルオプションを示します。各オプションの RS エイリアスは、オプションの説明にリストされています。

表 8.1 サポートされているテーブルオプションのデータソース別リスト

データソース	言語要素	カテゴリ	説明
Amazon Redshift	BULKLOAD=テーブルオプション	バルクロード	SAS が DBMS 機能を使用して、DBMS テーブルにデータを挿入するかどうかを決定します
	BULKOPTS=テーブルオプション	バルクロード	バルクロードオプションのコンテナーを提供します。
	DBCREATE_TABLE_OPTS=テーブルオプション	テーブルコントロール	データベース固有のオプションを CREATE TABLE ステートメントに追加できるようにします。
	RS_AWS_CONFIG=テーブルオプション	バルクロード	Amazon Web Services (AWS)構成ファイルの場所を指定します。
	RS_BUCKET=テーブルオプション	バルクロード	データをバルクで書き込むときに使用するクラウドプラットフォームバケットの名前を指定します。
	RS_CREDENTIALS_FILE=テーブルオプション	バルクロード	データをバルクロードするときに使用する AWS 資格情報ファイルを指定します。
	RS_DEFAULT_DIR=テーブルオプション	バルクロード	中間バルクロードファイルが作成される場所を指定します。
	RS_DELETE_DATAFILE= テーブルオプション	バルクロード	データファイルのみを削除するか、バルクロードプロセスによって作成されたすべてのファイルを削除するかを指定します。
	RS_KEY=テーブルオプション	バルクロード	キーベースのアクセスコントロールで使用される Amazon Web Services アクセスキーを指定します。一時トークン資格情報を使用している場合、これが一時アクセスキー ID になります。
	RS_REGION=テーブルオプション	バルクロード	Amazon S3 データの AWS リージョンを指定します。
	RS_SECRET=テーブルオプション	バルクロード	データのバルクロードに使用されるシークレットアクセスキーまたは一時シークレットアクセスキーを指定します。

データソース	言語要素	カテゴリ	説明
	RS_TOKEN=テーブルオプション	バルクロード	BL_KEY=および BL_SECRET=テーブルオプションで指定する一時資格情報に関連付けられる一時トークンを指定します。
DB2	BL_ALLOW_READ_ACCESS= テーブルオプション	バルクロード	バルクロード中に元のテーブルデータが引き続きリーダーに表示されることを指定します。
	BL_COPY_LOCATION= テーブルオプション	バルクロード	DB2 がロードされたデータのコピーを保存するディレクトリを指定します。
	BL_CPU_PARALLELISM= テーブルオプション	バルクロード	テーブルオブジェクトを構築するときに使用するプロセスまたはスレッドの数を指定します。
		バルクロード	データ転送用のバッファーとして使用するバルクロードユーティリティに割り当てるメモリの合計量を指定します。
	BL_DISK_PARALLELISM= テーブルオプション	バルクロード	データをディスクに書き込むときに使用するプロセスまたはスレッドの数を指定します。
	BL_EXCEPTION= テーブルオプション	バルクロード	エラーのある行がコピーされる例外テーブルを指定します。
	BL_INDEXING_MODE= テーブルオプション	バルクロード	DB2 ロードユーティリティがインデックスメンテナンスに使用するスキームを指定します。
	BL_LOAD_REPLACE= テーブルオプション	バルクロード	バルクロード中に DB2 が行を追加するか置換するかを指定します。
	BL_LOG= テーブルオプション	バルクロード	バルクロードの統計量やエラー情報などの情報を含むログファイルを識別します。
	BL_OPTIONS= テーブルオプション	バルクロード	オプションを DBMS バルクロード機能に渡し、データのロードと処理の方法に影響を与えます。
	BL_PORT_MAX= テーブルオプション	バルクロード	同時アップロードに使用できる最大のポート番号を設定します。

データソース	言語要素	カテゴリ	説明
	BL_PORT_MIN= テーブルオプション	バルクロード	同時アップロードに使用できる最小のポート番号を設定します。
	BL_RECOVERABLE= テーブルオプション	バルクロード	LOAD プロセスが回復可能かどうかを指定します。
	BL_REMOTE_FILE= テーブルオプション	バルクロード	DB2 LOAD 一時ファイルの基本ファイル名と場所を指定します。
	BL_WARNING_COUNT= テーブルオプション	バルクロード	ロードが失敗する前に許可する行警告の最大数を指定します。
	BULKLOAD= テーブルオプション	バルクロード	SAS が DBMS 機能を使用して DBMS テーブルにデータを挿入するかどうかを決定します。
	BULKOPTS= テーブルオプション	バルクロード	バルクロードオプションのコンテナー。このオプションは、BULKLOAD=YES の後に付ける必要があります。
	DBCREATE_TABLE_OPTS= テーブルオプション	テーブルコントロール	データベース固有のオプションを CREATE TABLE ステートメントに追加できるようにします。
Google BigQuery	BULKLOAD= テーブルオプション	バルクロード	SAS が DBMS 機能を使用して DBMS テーブルにデータを挿入するかどうかを決定します。
	BULKOPTS= テーブルオプション	バルクロード	バルクロードオプションのコンテナー。このオプションは、BULKLOAD=YES の後に付ける必要があります。
	DBCREATE_TABLE_OPTS=テーブルオプション	テーブルコントロール	データベース固有のオプションを CREATE TABLE ステートメントに追加できるようにします。
	FETCH_NUMERIC_TYPE= テーブルオプション	テーブルコントロール	Google BigQuery からの NUMERIC データの処理方法を指定します。
	KMS_PATH=テーブルオプション	テーブルアクセス	プロジェクト内のキーファイルリソースの場所を指定します。
	LOCATION=テーブルオプション	テーブルアクセス	データセットの場所を指定します。

データソース	言語要素	カテゴリ	説明
	READMODE=テーブルオプション	テーブルアクセス	Google BigQuery から SAS にデータを移動するときに使用する方法を指定します。
	SCANSTRINGCOLUMNS=テーブルオプション	テーブルアクセス	データベーステーブルまたはクエリの文字列の最大長を決定するかどうかを指定します。
Greenplum	DBCREATE_TABLE_OPTS=テーブルオプション	テーブルコントロール	データベース固有のオプションを CREATE TABLE ステートメントに追加できるようにします。
	GP_DISTRIBUTED_BY=テーブルオプション	テーブルコントロール	作成するテーブルの分散キーを指定します。
Hive	DBCREATE_TABLE_OPTS=テーブルオプション	テーブルコントロール	データベース固有のオプションを CREATE TABLE ステートメントの後に配置できるようにします。
	POST_TABLE_OPTS=テーブルオプション	テーブルコントロール	CREATE TABLE ステートメントのテーブル名の後にデータベース固有のオプションを配置できるようにします。
	PRE_TABLE_OPTS=テーブルオプション	テーブルコントロール	CREATE TABLE ステートメントのテーブル名の前にデータベース固有のオプションを配置できるようにします。
Impala	BL_AWS_CONFIG_FILE=テーブルオプション [2]	バルクロード	AWS 構成ファイルの場所を指定します。
	BL_AWS_CREDENTIALS_FILE=テーブルオプション [2]	バルクロード	認証情報ファイルへのパスを指定します。
	BL_BUCKET=テーブルオプション [2]	バルクロード	データファイルをバルクで転送するときに使用するバケット名を指定します。
	BL_DEFAULT_DIR=テーブルオプション [2]	バルクロード	バルクロード機能が中間ファイルを作成する場所を指定します。
	BL_DELETE_DATAFILE=テーブルオプション [2]	バルクロード	バルクロード機能によって作成された中間ファイルを削除するかどうかを指定します。
	BL_KEY=テーブルオプション [2]	バルクロード	キーベースのアクセスコントロールで使用される Amazon Web

データソース	言語要素	カテゴリ	説明
			Services アクセスキーを指定します。
	BL_REGION=テーブルオプション [2]	バルクロード	データを取得する Amazon Web Services(AWS)リージョンを指定します。
	BL_SECRET=テーブルオプション [2]	バルクロード	データのバルクロードに使用されるシークレットアクセスキーを指定します。
	BL_TOKEN=テーブルオプション [2]	バルクロード	BL_KEY=および BL_SECRET=オプションで指定する一時資格情報に関連付けられる一時トークンを指定します。
	BULKLOAD=テーブルオプション	バルクロード	SAS が DBMS 機能を使用して DBMS テーブルにデータを挿入するかどうかを決定します。
	BULKOPTS=テーブルオプション	バルクロード	バルクロードオプションのコンテナーを提供します。
	CONFIG=テーブルオプション (1084 ページ)[3]	バルクロード	Hadoop 構成パス解決のファイル名またはパス名を指定します。
	DATAFILE=テーブルオプション	バルクロード	一時 HDFS ファイルの代替名と場所を指定します。
	PASSWORD=テーブルオプション	バルクロード	HDFS ユーザーのパスワードを指定します。
	PICKLIST=テーブルオプション (1113 ページ)[3]	バルクロード	バルクロード操作に使用するピックリストを指定します。
	USER=テーブルオプション	バルクロード	HDFS ユーザー名を指定します。
MDS	BULKLOAD=テーブルオプション	バルクロード	SAS が DBMS 機能を使用して DBMS テーブルにデータを挿入するかどうかを決定します。
Microsoft SQL Server	BULKLOAD=テーブルオプション	バルクロード	SAS が DBMS 機能を使用して DBMS テーブルにデータを挿入するかどうかを決定します。
	DBCREATE_INDEX_OPTS=テーブルオプション	テーブルコントロール	データベース固有のオプションを CREATE INDEX ステートメントに追加できるようにします。

データソース	言語要素	カテゴリ	説明
	DBCREATE_TABLE_OPTS=テーブルオプション	テーブルコントロール	データベース固有のオプションを CREATE TABLE ステートメントに追加できるようにします。
MySQL	BULKLOAD=テーブルオプション	バルクロード	SAS が DBMS 機能を使用して DBMS テーブルにデータを挿入するかどうかを決定します。
	DBCREATE_TABLE_OPTS=テーブルオプション	テーブルコントロール	データベース固有のオプションを CREATE TABLE ステートメントに追加できるようにします。
Netezza	DBCREATE_TABLE_OPTS=テーブルオプション	テーブルコントロール	データベース固有のオプションを CREATE TABLE ステートメントに追加できるようにします。
ODBC	DBCREATE_INDEX_OPTS=テーブルオプション	インデックスコントロール	データベース固有のオプションを CREATE INDEX ステートメントに追加できるようにします。
	DBCREATE_TABLE_OPTS=テーブルオプション	テーブルコントロール	データベース固有のオプションを CREATE TABLE ステートメントに追加できるようにします。
Oracle	BL_DEFAULT_DIR=テーブルオプション	バルクロード	バルクロードが中間ファイルを作成する場所を指定します。
	BL_ERRORS=テーブルオプション	バルクロード	示された数のエラーを受信した後、ロードを停止する必要があることを指定します。
	BL_LOAD=テーブルオプション	バルクロード	指定された行数がロードされた後、ロードを停止する必要があることを指定します。
	BL_LOGFILE=テーブルオプション	バルクロード	バルクロードログファイルのファイル名を指定します。
	BL_PARALLEL=テーブルオプション	バルクロード	並列バルクロードを実行するかどうかを指定します。
	BL_RECOVERABLE=テーブルオプション	バルクロード	LOAD プロセスが回復可能かどうかを判別します。
	BL_SKIP=テーブルオプション	バルクロード	バルクロードを開始する前に、指定された行数をスキップするように指定します。

データソース	言語要素	カテゴリ	説明
	BL_SKIP_INDEX_MAINTENANCE=テーブルオプション	バルクロード	バルクロードでインデックスメンテナンスを実行するかどうかを指定します。
	BL_SKIP_UNUSABLE_INDEXES=テーブルオプション	バルクロード	使用できない状態のインデックスエントリをスキップして、バルクロードを続行するかどうかを指定します。
	BULKLOAD=テーブルオプション	バルクロード	SAS が DBMS 機能を使用して DBMS テーブルにデータを挿入するかどうかを決定します。
	BULKOPTS=テーブルオプション [1]	バルクロード	バルクロードオプションのコンテナー。このオプションは、BULKLOAD=YES の後に付ける必要があります。
	DBCREATE_TABLE_OPTS=テーブルオプション	テーブルコントロール	データベース固有のオプションを CREATE TABLE ステートメントに追加できるようにします。
	ORHINTS=テーブルオプション	データコントロール	FedSQL から Oracle に渡す Oracle ヒントを指定します。
	ORNUMERIC=テーブルオプション	テーブルコントロール	Oracle NUMBER 列から読み取られた数値または Oracle NUMBER 列に挿入された数値の処理方法を指定します。
PostgreSQL	BULKLOAD=テーブルオプション	バルクロード	SAS が DBMS 機能を使用して DBMS テーブルにデータを挿入するかどうかを決定します。
SAP HANA	DBCREATE_TABLE_OPTS=テーブルオプション	テーブルコントロール	データベース固有のオプションを CREATE TABLE ステートメントに追加できるようにします。
	TABLE_TYPE=テーブルオプション	テーブルアクセス	SAP HANA でテーブルを作成するときに FedSQL が使用するテーブルストレージの種類を指定します。
SAS データセット	ALTER=テーブルオプション	テーブルコントロール	SAS データセットに ALTER パスワードを割り当て、ユーザーがファイルを置換または削除できないようにしたり、読み取り保護または書き込み保護されたファイルへのアクセスを有効にしたりします。

データソース	言語要素	カテゴリ	説明
	BUFNO=テーブルオプション	テーブルコントロール	SAS データセットを処理するために割り当てられるバッファーの数を指定します。
	BUFSIZE=テーブルオプション	テーブルコントロール	出力 SAS データセットの永久バッファーページのサイズを指定します。
	COMPRESS=テーブルオプション	テーブルコントロール	新しい出力データセットで行を圧縮する方法を指定します。
	ENCODING=テーブルオプション	テーブルコントロール	SAS データセットの作成時に使用するエンコーディングを指定します。
	ENCRYPT=テーブルオプション	テーブルコントロール	出力 SAS データセットを暗号化するかどうかを指定します。
	ENCRYPTKEY=テーブルオプション	テーブルコントロール	AES 暗号化のキー値を指定します。
	IDXNAME=テーブルオプション	インデックス使用のユーザーコントロール	WHERE 句の条件に一致する特定のインデックスを使用するように SAS に指示します。
	IDXWHERE=テーブルオプション	インデックス使用のユーザーコントロール	SAS が WHERE 句の条件に一致するためにインデックス検索と順次検索のどちらを使用するかを指定します。
	LABEL=テーブルオプション	オブザベーションコントロール	SAS データセットのラベルを指定します。
	LOCKTABLE=テーブルオプション	テーブルコントロール	テーブルに共有ロックまたは排他ロックをかけます。
	POINTOBS=テーブルオプション	テーブルコントロール	SAS が圧縮データセットを作成し、そのオブザベーションがランダムアクセス可能か、順次アクセス可能かを指定します。
	PW=テーブルオプション	テーブルコントロール	READ、WRITE、および ALTER パスワードを SAS ファイルに割り当て、パスワードで保護されたファイルへのアクセスを有効にします。
	READ=テーブルオプション	テーブルコントロール	READ パスワードを SAS ファイルに割り当てて、ユーザーがパスワード

データソース	言語要素	カテゴリ	説明
			を入力しない限り、ファイルを読み取れないようにします。
	REUSE=テーブルオプション	テーブルコントロール	圧縮された SAS データセットの空き領域に新しい行を書き込むことができるかどうかを指定します。
	TYPE=テーブルオプション	テーブルコントロール	特別に構造化された SAS データセットのデータセットタイプを指定します。
	WRITE=テーブルオプション	テーブルコントロール	SAS ファイルに WRITE パスワードを割り当てて、ユーザーがファイルに書き込めないようにするか、書き込み保護されたファイルにアクセスできるようにします。
SingleStore	BULKLOAD=テーブルオプション	バルクロード	SAS が DBMS 機能を使用して DBMS テーブルにデータを挿入するかどうかを決定します。
	DBCREATE_TABLE_OPTS=テーブルオプション	テーブルコントロール	データベース固有のオプションを CREATE TABLE ステートメントに追加できるようにします。
Snowflake	BULKLOAD=テーブルオプション	バルクロード	SAS が DBMS 機能を使用して DBMS テーブルにデータを挿入するかどうかを決定します。
	BULKOPTS=テーブルオプション	バルクロード	バルクロードオプションのコンテナーを提供します。
	BL_ACCOUNTNAME=テーブルオプション	バルクロード	外部ストレージシステムで使用するアカウントの名前を指定します。
	BL_APPLICATIONID=テーブルオプション	バルクロード	外部ストレージシステムにアクセスするためのアプリケーション ID (GUID 文字列)を指定します。
	BL_AZURE_SAS=テーブルオプション	バルクロード	Azure 共有アクセス署名トークン値を指定します。
	BL_AZURE_TENANTID=テーブルオプション	バルクロード	Microsoft Azure ストレージシステムに接続するためのテナント ID を指定します。

データソース	言語要素	カテゴリ	説明
	BL_COMPRESS=テーブルオプション		gzip 形式を使用してデータファイルを圧縮するかどうかを指定します。
	BL_DEFAULT_DIR=テーブルオプション	バルクロード	バルクロードが中間ファイルを作成する場所を指定します。
	BL_DELETE_DATAFILE=テーブルオプション	バルクロード	データファイルのみを削除するか、バルクロード機能によって作成されたすべてのファイルを削除するかを指定します。
	BL_DELIMITER=テーブルオプション	バルクロード	バルクロード中にデータの列を区切るためのデフォルトの区切り文字のオーバーライドを指定します。
	BL_FILESYSTEM=テーブルオプション	バルクロード	外部ストレージシステム内のファイルシステムの名前を指定します。
	BL_FOLDER=テーブルオプション	バルクロード	外部ストレージシステム内のデータのフォルダーまたはファイルパスを指定します。
	BL_INTERNAL_STAGE=テーブルオプション	バルクロード	内部ステージおよびパスを指定します。
	BL_NUM_DATAFILES=テーブルオプション	バルクロード	ソースデータを含むように作成するデータファイルの数を指定します。
	BL_TIMEOUT=テーブルオプション	バルクロード	ストレージタスクの最大完了時間を秒単位で指定します。
	BL_USE_ESCAPE=テーブルオプション	バルクロード	バルクロードまたはバルク取得中に、事前定義された特殊文字のセットをエスケープするかどうかを指定します。
SPD Engine データセット	ALTER=テーブルオプション	テーブルコントロール	SPD Engine データセットに ALTER パスワードを割り当て、ユーザーがファイルを置換または削除できないようにしたり、読み取りまたは書き込み保護されたファイルへのアクセスを有効にしたりします。
	ASYNCINDEX=テーブルオプション	インデックス使用のユーザーコントロール	SPD Engine データセットに複数のインデックスを作成するときに、インデックスを並行して作成するように指定します。

データソース	言語要素	カテゴリ	説明
	COMPRESS=テーブルオプション	テーブルコントロール	作成時にディスク上の SPD Engine データセットを圧縮するように指定します。
	ENCRYPT=テーブルオプション	テーブルコントロール	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。
	ENCRYPTKEY=テーブルオプション	テーブルコントロール	AES 暗号化のキー値を指定します。
	ENDOBS=テーブルオプション	オブザベーションコントロール	処理するオブザベーションのユーザー定義範囲内の終了オブザベーション番号を指定します。
	IDXWHERE=テーブルオプション	インデックス使用のユーザーコントロール	SPD Engine で WHERE 式を処理するときにインデックスを使用するように指定します。
	IOBLOCKSIZE=テーブルオプション	テーブルコントロール	I/O 操作で使用され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。
	LABEL=テーブルオプション	オブザベーションコントロール	SPD Engine データセットのラベルを指定します。
	PADCOMPRESS=テーブルオプション	テーブルコントロール	OUTPUT または UPDATE 用に開かれた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=テーブルオプション	テーブルコントロール	SPD Engine データセット内のデータコンポーネントパーティションのサイズを指定します。
	PW=テーブルオプション	テーブルコントロール	READ、WRITE、および ALTER パスワードを SAS ファイルに割り当て、パスワードで保護されたファイルへのアクセスを有効にします。
	READ=テーブルオプション	テーブルコントロール	READ パスワードを SAS ファイルに割り当てて、ユーザーがパスワードを入力しない限り、ファイルを読み取れないようにします。
	STARTOBS=テーブルオプション	オブザベーションコントロール	処理するオブザベーションのユーザー定義範囲内の開始オブザベーション番号を指定します。

データソース	言語要素	カテゴリ	説明
	THREADNUM=テーブルオプション	テーブルコントロール	SPD Engine データセットを処理するために SPD Engin が起動できる I/O スレッドの最大数を指定します。
	TYPE=テーブルオプション	テーブルコントロール	特別に構造化された SPD Engine データセットのデータセットタイプを指定します。
	UNIQUESAVE=テーブルオプション	インデックス使用のユーザーコントロール	一意のインデックスを持つ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、一意でないキー値を持つオブザベーション(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。
	WHERENOINDEX=テーブルオプション	インデックス使用のユーザーコントロール	WHERE 式の評価を行うときに除外するインデックスのリストを指定します。
	WRITE=テーブルオプション	テーブルコントロール	SAS ファイルに WRITE パスワードを割り当てて、ユーザーがファイルに書き込めないようにするか、書き込み保護されたファイルにアクセスできるようにします。
SPD Server テーブル	COMPRESS=テーブルオプション	テーブルコントロール	作成時にディスク上の SPD Server テーブルを圧縮するように指定します。
	ENCRYPT=テーブルオプション	テーブルアクセス	出力 SPD Server テーブルを暗号化するかどうかを指定します。
	ENCRYPTKEY=テーブルオプション	テーブルアクセス	AES 暗号化のキーを指定します。
	ENDOBS=テーブルオプション	オブザベーションコントロール	処理するオブザベーションのユーザー定義範囲内の終了オブザベーション番号を指定します。
	IOBLOCKSIZE=テーブルオプション	テーブルコントロール	I/O 操作で使用される行ブロックのサイズをバイト単位で指定します。
	PARTSIZE=テーブルオプション	テーブルコントロール	SPD Server テーブル内のデータコンポーネントパーティションのサイズを指定します。

データソース	言語要素	カテゴリ	説明
	PW=テーブルオプション	テーブルアクセス	SAS Proprietary 暗号化によって保護されている SPD Server テーブルにアクセスできるようにします。
	STARTOBS=テーブルオプション	オブザベーションコントロール	処理するオブザベーションのユーザー定義範囲内の開始オブザベーション番号を指定します。
Spark	DBCREATE_TABLE_OPTS=テーブルオプション	テーブルコントロール	データベース固有のオプションを CREATE TABLE ステートメントの後に配置できるようにします。
	POST_TABLE_OPTS=テーブルオプション	テーブルコントロール	CREATE TABLE ステートメントのテーブル名の後にデータベース固有のオプションを配置できるようにします。
	PRE_TABLE_OPTS=テーブルオプション	テーブルコントロール	CREATE TABLE ステートメントのテーブル名の前にデータベース固有のオプションを配置できるようにします。
Teradata	BULKLOAD=テーブルオプション	バルクロード	SAS が DBMS 機能を使用して DBMS テーブルにデータを挿入するかどうかを決定します。
	BULKOPTS=テーブルオプション	バルクロード	バルクロードオプションのコンテナー。このオプションは、BULKLOAD=YES の後に付ける必要があります。
	DBCREATE_TABLE_OPTS=テーブルオプション	テーブルコントロール	データベース固有のオプションを CREATE TABLE ステートメントに追加できるようにします。
	TD_BUFFER_MODE=テーブルオプション	バルクロード	LOAD メソッドを使用するかどうかを指定します。
	TD_CHECKPOINT=テーブルオプション	バルクロード	TPT 操作がデータベースにチェックポイントまたはセーブポイントを発行するタイミングを指定します。
	TD_DATA_ENCRYPTION=テーブルオプション	バルクロード	データ暗号化をアクティブにします。
	TD_DROP_ERROR_TABLE=テーブルオプション	バルクロード	ジョブが正常に完了したかどうかに関係なく、ジョブの終了時にログテーブルを削除します。

データソース	言語要素	カテゴリ	説明
	TD_DROP_LOG_TABLE=テーブルオプション	バルクロード	ジョブが正常に完了したかどうかに関係なく、ジョブの終了時にエラーテーブルを削除します。
	TD_DROP_WORK_TABLE=テーブルオプション	バルクロード	ジョブが正常に完了したかどうかに関係なく、ジョブの終了時に作業テーブルを削除します。
	TD_ERROR_LIMIT=テーブルオプション	バルクロード	エラーテーブルに保存できるレコードの最大数を指定します。
	TD_ERROR_TABLE_1=テーブルオプション	バルクロード	最初のエラーテーブルの名前を指定します。
	TD_ERROR_TABLE_2=テーブルオプション	バルクロード	2 番目のエラーテーブルの名前を指定します。
	TD_LOG_MECH_TYPE=テーブルオプション	バルクロード	バルクロードのログオンメカニズムを指定します。
	TD_LOG_MECH_DATA=テーブルオプション	バルクロード	ログオンメカニズムの追加データを指定します。
	TD_LOG_TABLE=テーブルオプション	バルクロード	再起動ログテーブルの名前を指定します。
	TD_LOGDB=テーブルオプション	バルクロード	TPT ユーティリティテーブルが 作成されるデータベースを指定します。
	TD_MAX_SESSIONS=テーブルオプション	バルクロード	TPT がジョブに対して取得できるログオンセッションの最大数を指定します。
	TD_MIN_SESSIONS=テーブルオプション	バルクロード	ジョブが開始する前に TPT が取得するセッションの最小数を指定します。
	TD_NOTIFY_LEVEL=テーブルオプション	バルクロード	ログイベントが記録されるレベルを指定します。
	TD_NOTIFY_METHOD=テーブルオプション	バルクロード	イベントを報告する方法を指定します。
	TD_NOTIFY_STRING=テーブルオプション	バルクロード	システムログに送信されるすべてのメッセージの前にある文字列を定義します。

データソース	言語要素	カテゴリ	説明
	TD_PACK=テーブルオプション	バルクロード	マルチステートメント要求にパックするステートメントの数を指定します。
	TD_PACK_MAXIMUM=テーブルオプション	バルクロード	Stream オペレーターが、現在の Stream ジョブの可能な最大パック係数を決定できるようにします。
	TD_PAUSE_ACQ=テーブルオプション	バルクロード	ロードジョブの取得フェーズと適用フェーズの間に一時停止を強制します。
	TD_SESSION_QUERY_BAND=テーブルオプション	バルクロード	TPT セッションで使用するために、ユーザー指定の名前=値ペアの文字列を渡します。
	TD_TENACITY_HOURS=テーブルオプション	バルクロード	TPT オペレーターが Teradata データベースへのログオンを継続して試行する時間を指定します。
	TD_TENACITY_SLEEP=テーブルオプション	バルクロード	Teradata データベースへのログオンを再試行する前に、TPT オペレーターが一時停止する時間を指定します。
	TD_TPT_OPER=テーブルオプション	バルクロード	Teradata Parallel Transporter が使用するロードオペレーターを指定します。
	TD_TRACE_LEVEL=、TD_TRACE_LEVEL_INF=テーブルオプション	バルクロード	ドライバートレースのトレースレベルを指定します。TD_TRACE_LEVEL は、プライマリトレースレベルを設定します。TD_TRACE_LEVEL_INF は、セカンダリトレースレベルを設定します。
	TD_TRACE_OUTPUT=テーブルオプション	バルクロード	トレースメッセージに使用される外部ファイルの名前を指定します。
	TD_UNICODE_PASSTHRU=テーブルオプション	バルクロード	パススルー文字(PTC)を Teradata にインポートおよび Teradata からエクスポートできるようにします。
	TD_WORKING_DB=テーブルオプション	バルクロード	テーブルが作成されるデータベースを指定します。

データソース	言語要素	カテゴリ	説明
	TD_WORK_TABLE=テーブルオプション	バルクロード	TPT 作業テーブルの名前を指定します。

1 2025.05 以前のリリースまで、Oracle 向けのこのオプションは BL_BULKLOPTS=でした。2025.06 以降、Oracle 向けのこのオプションは BULKOPTS=です。
2 このオプションのサポートは 2024.09 から開始されます。
3 このオプションは、2026.02 以前のリリースで使用できます。これは 2026.01 で非推奨となり、2026.03 で削除されました。

ディクショナリ

ALTER= テーブルオプション

データセットに ALTER パスワードを割り当て、ユーザーがファイルを置換または削除できないようにしたり、読み取り保護または書き込み保護されたファイルへのアクセスを有効にしたりします。

カテゴリ: テーブルコントロール

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

データソース: SAS データセット、SPD Engine データセット

注: この操作の後にログをチェックして、パスワード値が表示されていないことを確認してください。詳細については、“ログからパスワードと暗号化キーの値を消去する” (*SAS プログラマガイド: エッセンシャル*)を参照してください。

構文

ALTER=*alter-password*

引数

alter-password
パスワードを指定します。

制限事項 *alter-password* は有効な SAS 名である必要があります。

詳細

ALTER=テーブルオプションは、SAS データセットまたは SPD Engine データセットにのみ適用されます。このオプションを使用して、パスワードを割り当てたり、読み取り保護、書き込み保護、または変更保護されたファイルにアクセスしたりできます。ALTER パスワードで保護されているデータセットを置換すると、新しいデータセットには ALTER パスワードが継承されます。

注: SAS パスワードは、SAS システム以外の SAS ファイルへのアクセスを制御しません。SAS 外の SAS ファイルへのアクセスを制御するには、オペレーティングシステムが提供するユーティリティとファイルシステムのセキュリティ 制御を使用する必要があります。

ASYNCINDEX= テーブルオプション

SPD Engine データセットに複数のインデックスを作成するときに、インデックスを並行して作成するように指定します。

該当要素: CREATE INDEX ステートメント

カテゴリ: インデックス使用のユーザーコントロール

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

データソース: SPD Engine データセット

構文

ASYNCINDEX= YES | NO

引数

YES
インデックスを並行して(非同期に)作成します。

NO
一度に 1 つのインデックスを(同期的に)作成します。これがデフォルト値です。

詳細

SPD Engine は、データセットの複数のインデックスを同時に作成できます。SPD Engine は、作成されたインデックスごとに 1 つのスレッドを生成し、スレッドを同時に処理します。インデックスを並行して作成する方が、一度に 1 つのインデックスを作成するよりもはるかに高速ですが、このオプションのデフォルトは NO です。並列作成には、追加のユーティリティワークスペースと 追加のメモリが必要ですが、

使用できない場合があります。リソースが不足しているためにインデックスの作成に失敗した場合は、次のいずれかを実行できます。

- SAS システムオプションを MEMSIZE=0 に設定する
- 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す

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。インデックスを並行して作成するように指定する場合は、SPDEUTILLOC=システムオプションを使用して十分な大きさの領域を指定します。

BL_ACCOUNTNAME= テーブルオプション

外部ストレージシステムで使用するアカウントの名前を指定します。

カテゴリ: バルクロード

デフォルト: なし

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件: このオプションを指定するには、BULKLOAD=YES または BULKUNLOAD=YES も設定する必要があります。
“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。

データソース: Snowflake

構文

BL_ACCOUNTNAME= *account-name*

引数

account-name
外部ストレージシステムで使用するアカウントの名前を指定します。

詳細

このオプションは、Azure Data Lake Storage (ADLS)の場所にある Snowflake にデータをバルクロードするために使用されます。

外部データの場所は、このオプション、BL_FILESYSTEM=テーブルオプション、およびオプションで BL_FOLDER=テーブルオプションを使用して生成されます。これらの値を組み合わせて、Azure Data Lake Storage Gen2 の場所にある外部ストレージの場所への URL を定義します。

https://<account-name>.<network-storage-host-name>/<file-system-name>/<file-path-folder>

HTTPS パスでは、*account-name* はステージアカウントです。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)

BL_ALLOW_READ_ACCESS= テーブルオプション

バルクロード中に元のテーブルデータが引き続きリーダーに表示されることを指定します。

カテゴリ: バルクロード

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件: このオプションを指定するには、BULKLOAD=YES または BULKUNLOAD=YES も設定する必要があります。
“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。

データソース: DB2

構文

BL_ALLOW_READ_ACCESS= YES | NO

引数

YES
バルクロードの進行中に、テーブル内の元の(変更されていない)データが引き続きリーダーに表示されることを指定します。

NO
バルクロードの進行中は、リーダーがテーブル内の元のデータを表示できないことを指定します。これがデフォルト値です。

詳細

このオプションの使用方法の詳細については、*IBM DB2 Universal Database Data Movement Utilities Guide and Reference* の SQLU_ALLOW_READ_ACCESS パラメーターを参照してください。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)

BL_APPLICATIONID= テーブルオプション

外部ストレージシステムにアクセスするためのアプリケーション ID (GUID 文字列)を指定します。

カテゴリ: バルクロード

デフォルト: なし

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件: このオプションを指定するには、BULKLOAD=YES または BULKUNLOAD=YES も設定する必要があります。

“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。

認証に使用するユーザー ID は、ADLS アカウント構成で共同作成者とストレージ BLOB データ共同作成者の役割が割り当てられている必要があります。

データソース: Snowflake

例: bl_applicationid="b1fc955d5c-e0e2-45b3-a3cc-a1cf54120f"

構文

BL_APPLICATIONID= "*string*"

引数

string
外部ストレージシステムにアクセスするための Microsoft Entra アプリケーション ID (GUID 文字列)を指定します。

詳細

このオプションは、 ADLS の場所にある Snowflake にデータをバルクロードするために使用されます。

BL_APPLICATIONID=オプションが指定されている場合、特定のユーザーではなく、Microsoft Entra アプリケーション ID がデータへのアクセス権限を付与されたプリンシパルになります。Microsoft Entra アプリケーション ID の資格情報を持つすべての人がデータにアクセスできます。

関連項目

テーブルオプション:

- “BL_ACCOUNTNAME= テーブルオプション” (1034 ページ)
- “BL_AZURE_SAS= テーブルオプション” (1038 ページ)
- “BL_AZURE_TENANTID= テーブルオプション” (1039 ページ)
- “BL_FILESYSTEM= テーブルオプション” (1050 ページ)
- “BULKLOAD= テーブルオプション” (1074 ページ)

BL_AWS_CONFIG_FILE= テーブルオプション

AWS 構成ファイルの場所を指定します。

カテゴリ: バルクロード

デフォルト: **~/.aws/config**

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件: このオプションを指定するには、BULKLOAD=YES または BULKUNLOAD=YES も設定する必要があります。
“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。

データソース: Impala

注: Impala のサポートは 2025.09 から始まります。

構文

BL_AWS_CONFIG_FILE=*path-and-file-name*

必須引数

path-and-file-name
AWS 構成ファイルの場所を指定します。デフォルトでは、この場所は**~/.aws/config** です。

BL_AWS_CREDENTIALS_FILE= テーブルオプション

認証情報ファイルへのパスを指定します。

カテゴリ: バルクロード

デフォルト: **~/.aws/credentials**

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件: このオプションを指定するには、BULKLOAD=YES または BULKUNLOAD=YES も設定する必要があります。
“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。

データソース: Amazon Redshift, Impala

注: Impala のサポートは 2025.09 から始まります。

構文

BL_AWS_CREDENTIALS_FILE=*path-and-file-name*

必須引数

path-and-file-name
AWS 認証情報ファイルの場所を指定します。

BL_AZURE_SAS= テーブルオプション

Azure 共有アクセス署名トークン値を指定します。

カテゴリ: バルクロード

デフォルト: なし

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件: このオプションを指定するには、BULKLOAD=YES または BULKUNLOAD=YES も設定する必要があります。
“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。

データソース: Snowflake

構文

BL_AZURE_SAS= *value*

引数

value
Azure 共有アクセス署名トークン値を指定します。

詳細

このオプションは、 ADLS 接続のステージアプリケーション署名を提供します。

関連項目

テーブルオプション:

- “BL_ACCOUNTNAME= テーブルオプション” (1034 ページ)
- “BL_APPLICATIONID= テーブルオプション” (1036 ページ)
- “BL_AZURE_TENANTID= テーブルオプション” (1039 ページ)
- “BL_FILESYSTEM= テーブルオプション” (1050 ページ)
- “BULKLOAD= テーブルオプション” (1074 ページ)

BL_AZURE_TENANTID= テーブルオプション

Microsoft Azure ストレージシステムに接続するためのテナント ID を指定します。

カテゴリ: バルクロード

デフォルト: なし

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件: このオプションを指定するには、BULKLOAD=YES または BULKUNLOAD=YES も設定する必要があります。

“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。

認証に使用するユーザー ID は、ADLS アカウント構成で共同作成者とストレージ BLOB データ共同作成者の役割が割り当てられている必要があります。

データソース: Snowflake

重要: ユーザーが初めてデータにアクセスしようとするときには、ユーザーの操作が必要です。詳細については、AZURETENANTID=システムオプションの"デバイスコード認証の使用"を参照してください。

構文

BL_AZURE_TENANTID= *tenant-id*

引数

tenant-id
Microsoft Azure Data Lake Storage Gen2 に接続するためのテナント ID (GUID)を指定します。

長さ	値の最大長は 64 バイトです。
制限事項	名前にスペースまたは英数字以外の文字が含まれている場合は、名前を単一引用符または二重引用符で囲みます。通常、この値にはハイフン(-)文字が含まれます。

詳細

AZURETENANTID=テーブルオプションは、Microsoft Azure Data Lake Storage Gen2 のデバイスコード認証で使用されます。このテーブルオプションは、AZURETENANTID=システムオプションの機能を FedSQL に提供します。

関連項目

テーブルオプション:

- “BL_ACCOUNTNAME= テーブルオプション” (1034 ページ)
- “BL_APPLICATIONID= テーブルオプション” (1036 ページ)
- “BL_AZURE_SAS= テーブルオプション” (1038 ページ)
- “BL_FILESYSTEM= テーブルオプション” (1050 ページ)
- “BULKLOAD= テーブルオプション” (1074 ページ)

BL_BUCKET= テーブルオプション

データファイルをバルクで転送するときに使用するバケット名を指定します。

カテゴリ:	バルクロード
デフォルト:	なし
制限事項:	このテーブルオプションは、CAS サーバーではサポートされていません。
要件:	Amazon S3 データを Impala にバルクロードするには、バケット名を指定する必要があります。 指定するバケット名はすでに存在する必要があります。 このオプションを指定するには、BULKLOAD=YES または BULKUNLOAD=YES も設定する必要があります。 “BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。
データソース:	Impala
注:	Impala のサポートは 2025.09 から始まります。
参照項目:	Amazon S3 バケットについては、Amazon Web Services S3 の情報を参照してください。

構文

BL_BUCKET='*<bucket-name>*'

BL_COMPRESS= テーブルオプション

gzip 形式を使用してデータファイルを圧縮するかどうかを指定します。

カテゴリ: バルクロード

別名: RS_COMPRESS (Amazon Redshift のみ)

デフォルト: NO

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件: このオプションを指定するには、BULKLOAD=YES または BULKUNLOAD=YES も設定する必要があります。
“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。

データソース: Amazon Redshift, Snowflake

構文

BL_COMPRESS= YES | NO

詳細

低速なネットワークまたは非常に大きなデータファイルの場合、
BL_COMPRESS=YES を使用するとパフォーマンスが向上することがあります。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)

BL_COPY_LOCATION= テーブルオプション

DB2 がロードされたデータのコピーを保存するディレクトリを指定します。

カテゴリ: バルクロード

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件: このオプションを指定するには、BULKLOAD=YES または BULKUNLOAD=YES も設定する必要があります。

“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。

データソース: DB2

構文

BL_COPY_LOCATION= *pathname*

引数

pathname
ロードされたデータがコピーされるパスを指定します。

関連項目

テーブルオプション:

- “BL_RECOVERABLE= テーブルオプション” (1062 ページ)
- “BULKLOAD= テーブルオプション” (1074 ページ)

BL_CPU_PARALLELISM= テーブルオプション

テーブルオブジェクトを構築するときに使用するプロセスまたはスレッドの数を指定します。

カテゴリ: バルクロード

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件: このオプションを指定するには、BULKLOAD=YES または BULKUNLOAD=YES も設定する必要があります。

“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。

データソース: DB2

構文

BL_CPU_PARALLELISM= *number-of-processes-or-threads*

引数

number-of-processes-or-threads
テーブルオブジェクトを構築するときに、ロードユーティリティがデータレコードを解析、変換、およびフォーマットするために使用するプロセスまたはスレッドの数を指定します。

詳細

このオプションは、パーティション内並列処理を活用し、ロードパフォーマンスを大幅に向上させます。ソースデータのレコード順序が保持されるため、事前に並べ替えられたデータをロードする場合に特に便利です。

許可される最大数は 30 です。値が 0 であるか指定されていない場合、ロードユーティリティはインテリジェントなデフォルトを選択します。このデフォルトは、実行時にシステムで使用可能な CPU の数に基づいています。指定された値をサポートするのに十分なメモリがない場合、ユーティリティは値を調整します。

BL_CPU_PARALLELISM が 1 より大きい場合、フラッシュ操作は非同期になり、ローダーが CPU を利用できるようになります。テーブルに LOB または LONG VARCHAR データが含まれる場合、並列処理はサポートされません。システム CPU の数や指定値に関係なく、値は 1 に設定されます。

このパラメーターの使用は対称型マルチプロセッサ(SMP)ハードウェアに限定されませんが、SMP 以外の環境で使用しても、パフォーマンス上の利点を認識できない場合があります。

BL_CPU_PARALLELISM=の使用方法の詳細については、*IBM DB2 Universal Database Data Movement Utilities Guide and Reference* の CPU_PARALLELISM パラメーターを参照してください。

関連項目

テーブルオプション:

- “BL_DATA_BUFFER_SIZE= テーブルオプション” (1043 ページ)
- “BL_DISK_PARALLELISM= テーブルオプション” (1047 ページ)
- “BULKLOAD= テーブルオプション” (1074 ページ)

BL_DATA_BUFFER_SIZE= テーブルオプション

データ転送用のバッファーとして使用するバルクロードユーティリティに割り当てるメモリの合計量を指定します。

カテゴリ: バルクロード

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件: このオプションを指定するには、BULKLOAD=YES または BULKUNLOAD=YES も設定する必要があります。
“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。

データソース: DB2

構文

BL_DATA_BUFFER_SIZE= *buffer-size*

引数

buffer-size
バルクロードユーティリティがユーティリティ 内でデータを転送するためのバッファー領域として使用するために割り当てられるメモリの合計量(4KB ページ単位)を指定します。この設定では、使用可能な並列度は考慮されません。

詳細

アルゴリズムの最小値より小さい値を指定すると、必要最小限のリソースが使用され、警告は返されません。このメモリはユーティリティヒープから 直接割り振られ、そのサイズは util_heap_sz データベース構成パラメーターを介して変更できます。値を指定しない場合、ユーティリティは実行時にインテリジェントなデフォルトを計算します。この計算は、ローダーのインスタンス化時にユーティリティヒープで使用可能な空き領域のパーセンテージと、テーブルのいくつかの特性に基づいています。

バッファーのサイズは数エクステントにすることをお勧めします。*エクステント*は DB2 内のデータの移動単位であり、エクステントサイズは 1 つ以上の 4KB ページにすることができます。

DATA BUFFER パラメーターは、入力および出力の待機時間を短縮するため、ラージオブジェクト(LOB)を操作する場合に役立ちます。データバッファーは、ユーティリティヒープから 割り当てられます。システムで使用可能なストレージの量に応じて、DB2 ユーティリティで 使用するためにより多くのメモリを割り当てることを検討する必要があります。それに応じて、データベース構成パラメーター util_heap_sz を変更できます。ユーティリティヒープサイズ 構成パラメーターのデフォルト値は、5,000 4KB ページです。ロードは、ユーティリティヒープのメモリを使用するいくつかのユーティリティの 1 つであるため、ロードユーティリティで使用できるのは、このパラメーターで 定義されたページの 50%以下にすることをお勧めします。

このオプションの使用方法の詳細については、*IBM DB2 Universal Database Data Movement Utilities Guide and Reference* の DATA BUFFER パラメーターを参照してください。

関連項目

テーブルオプション:

- “BL_CPU_PARALLELISM= テーブルオプション” (1042 ページ)
- “BL_DISK_PARALLELISM= テーブルオプション” (1047 ページ)
- “BULKLOAD= テーブルオプション” (1074 ページ)

BL_DEFAULT_DIR= テーブルオプション

バルクロード機能が中間ファイルを作成する場所を指定します。

カテゴリ:	バルクロード
デフォルト:	UTILLOC=システムオプションで指定される一時ファイルディレクトリ。
制限事項:	このテーブルオプションは、CAS サーバーではサポートされていません。
要件:	このオプションを指定するには、BULKLOAD=YES または BULKUNLOAD=YES も設定する必要があります。 “BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。
データソース:	Impala, Oracle, Snowflake
注:	Impala のサポートは 2025.09 から始まります。

構文

BL_DEFAULT_DIR= *host-specific-directory-path*

引数

host-specific-directory-path
中間バルクロードファイルが作成されるホスト固有のディレクトリパスを指定します。

注 ディレクトリパスはスラッシュで終わる必要があります。例: BL_DEFAULT_DIR='/temp/'

詳細

このオプションに指定した値は、ファイル名の先頭に追加されます。すべてのプラットフォームに対応するために、ファイルとディレクトリの区切り文字を含む完全なホスト固有のディレクトリパスを指定してください。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)

BL_DELETE_DATAFILE= テーブルオプション

バルクロード機能によって作成された中間ファイルを削除するかどうかを指定します。

カテゴリ:	バルクロード
デフォルト:	YES
制限事項:	このテーブルオプションは、CAS サーバーではサポートされていません。
要件	このオプションを指定するには、BULKLOAD=YES または BULKUNLOAD=YES も設定する必要があります。
データソース:	Google BigQuery, Impala, Snowflake
注:	Impala のサポートは 2025.09 から始まります。

構文

BL_DELETE_DATAFILE= YES | NO

引数

YES
SAS/ACCESS エンジンが DBMS バルクロード機能用に作成するすべての(データ、コントロール、およびログ)ファイルを削除します。

NO
ファイルを保持します。

詳細

このオプションは“BULKOPTS= テーブルオプション”内で指定する必要があります。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)

BL_DELIMITER= テーブルオプション

バルクロード中にデータの列を区切るためのデフォルトの区切り文字をオーバーライドします。

カテゴリ: バルクロード

別名: RS_DELIMITER (Amazon Redshift のみ)

デフォルト: ベル文字(ASCII 0x07 または 8 進数\007)

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件: このオプションを指定するには、BULKLOAD=YES または BULKUNLOAD=YES も設定する必要があります。
“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。

データソース: Amazon Redshift, Snowflake

注: Snowflake のサポートは、2024.11 から開始されます。

構文

BL_DELIMITER= '*single-ascii-character* | *octal-representation*'

詳細

CSV は、Snowflake へのバルクロードに使用されるファイル形式です。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)

BL_DISK_PARALLELISM= テーブルオプション

データをディスクに書き込むときに使用するプロセスまたはスレッドの数を指定します。

カテゴリ: バルクロード

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件: このオプションを指定するには、BULKLOAD=YES または BULKUNLOAD=YES も設定する必要があります。
“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。

データソース: DB2

構文

BL_DISK_PARALLELISM= *number-of-processes-or-threads*

引数

number-of-processes-or-threads
テーブルスペースコンテナーにデータレコードを 書き込むためにロードユーティリティーが使用するプロセスまたはスレッドの数を指定します。

詳細

このオプションは、データをロードするときに使用可能なコンテナーを活用し、ロードパフォーマンスを大幅に向上させます。

許可される最大数は、ロードユーティリティが実際に使用する BL_CPU_PARALLELISM 値の 4 倍、または 50 のうち大きい方です。デフォルトでは、BL_DISK_PARALLELISM は、この値が許可されている最大数を超えている場合を除き、ロードされるテーブルのオブジェクトを含むすべてのテーブルスペースのテーブルスペースコンテナーの 合計に等しくなります。

値を指定しない場合、ユーティリティは、テーブルスペースコンテナーの 数とテーブルの特性に基づいたインテリジェントなデフォルトを選択します。

このオプションの使用方法の詳細については、*IBM DB2 Universal Database Data Movement Utilities Guide and Reference* の DISK_PARALLELISM パラメーターを参照してください。

関連項目

テーブルオプション:

- “BL_CPU_PARALLELISM= テーブルオプション” (1042 ページ)
- “BULKLOAD= テーブルオプション” (1074 ページ)

BL_ERRORS= テーブルオプション

示された数のエラーを受信した後、ロードを停止する必要があることを指定します。

カテゴリ: バルクロード

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件: このオプションを指定するには、BULKLOAD=YES または BULKUNLOAD=YES も設定する必要があります。
“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。

データソース: Oracle

構文

BL_ERRORS= *number*

引数

number
ロードが停止する前に受信するエラーの数を指定します。デフォルトは 1000000 エラーです。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)

BL_EXCEPTION= テーブルオプション

エラーのある行がコピーされる例外テーブルを指定します。

カテゴリ: バルクロード

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件: このオプションを指定するには、BULKLOAD=YES または BULKUNLOAD=YES も設定する必要があります。
“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。

データソース: DB2

構文

BL_EXCEPTION= *exception-table-name*

引数

exception-table-name
エラーのある行がコピーされる例外テーブルを指定します。

詳細

一意のインデックスまたは主キーインデックスに違反している行はすべてコピーされます。DATALINK 例外も例外テーブルに取り込まれます。修飾されていないテーブル名を指定すると、テーブルは CURRENT SCHEMA で修飾されます。例外テーブルに書き込まれる情報は、ダンプファイルには書き込まれません。パーティションデータベース環境では、ロードテーブルが定義されているパーティションに対して例外テーブルを定義する必要があります。ただし、ダンプファイルには、無効であるか構文エラーが含まれているためにロードできない行が含まれています。

このオプションの使用方法の詳細については、*IBM DB2 Universal Database Data Movement Utilities Guide and Reference* の EXCEPTION パラメーターを参照してください。ロード例外テーブルの詳細については、*IBM DB2 Universal Database Data Movement Utilities Guide and Reference* および *IBM DB2 Universal Database SQL Reference, Volume 1* のロード例外テーブルのトピックを参照してください。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)

BL_FILESYSTEM= テーブルオプション

外部ストレージシステム内のファイルシステムの名前を指定します。

カテゴリ:	バルクロード
デフォルト:	なし
制限事項:	このテーブルオプションは、CAS サーバーではサポートされていません。
要件:	このテーブルオプションは、Azure で Snowflake へのバルクロードを有効にするために必要です。 このオプションを指定するには、BULKLOAD=YES または BULKUNLOAD=YES も設定する必要があります。 “BULKOPTS= テーブルオプション” (1079 ページ) 内で指定する必要があります。
データソース:	Snowflake
注:	このオプションのサポートは 2024.11 から開始されます。

構文

BL_FILESYSTEM= *file-system-name*

引数

file-system-name
ファイルの書き込み先となる ADLS 内のファイルシステムの名前を指定します。

詳細

このオプションは、 ADLS の場所にある Snowflake にデータをバルクロードするために使用されます。

外部データの場所は、BL_ACCOUNTNAME=オプション、BL_APPLICATIONID=オプション、およびオプションで BL_FOLDER=テーブルオプションを使用して生成されます。これらの値を組み合わせて、Azure Data Lake Storage Gen2 の場所にアクセスするための URL を定義します。

https://<account-name>.<network-storage-host-name>/<file-system-name>/<file-path-folder>

HTTPS パスでは、*file-system-name* は BLOB コンテナーです。

関連項目

テーブルオプション:

- “BL_ACCOUNTNAME= テーブルオプション” (1034 ページ)
- “BL_APPLICATIONID= テーブルオプション” (1036 ページ)
- “BL_AZURE_SAS= テーブルオプション” (1038 ページ)
- “BL_AZURE_TENANTID= テーブルオプション” (1039 ページ)
- “BL_FOLDER= テーブルオプション” (1051 ページ)
- “BULKLOAD= テーブルオプション” (1074 ページ)

BL_FOLDER= テーブルオプション

外部ストレージシステム内のデータのフォルダーまたはファイルパスを指定します。

カテゴリ: バルクロード

デフォルト: なし

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件: このオプションを指定するには、BULKLOAD=YES または BULKUNLOAD=YES も設定する必要があります。
“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。

データソース: Snowflake

注: このオプションのサポートは 2024.11 から開始されます。

構文

BL_FOLDER= *file-path-and-folder*

引数

file-path-and-folder
外部ストレージシステム内のデータのフォルダーまたはファイルパスを指定します。場所のパスはコンテナー名で始まります。

詳細

このオプションは、 ADLS の場所にある Snowflake にデータをバルクロードするために使用されます。このオプションを指定すると、Azure Data Lake Storage Gen2 の場所へのアクセスに使用される生成された URL に含まれます。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)

BL_INDEXING_MODE= テーブルオプション

DB2 ロードユーティリティがインデックスメンテナンスに 使用するスキームを指定します。

カテゴリ: バルクロード

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件: このオプションを指定するには、BULKLOAD=YES または BULKUNLOAD=YES も設定する必要があります。
“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。

データソース: DB2

構文

BL_INDEXING_MODE= AUTOSELECT | REBUILD | INCREMENTAL | DEFERRED

引数

AUTOSELECT
ロードユーティリティが REBUILD または INCREMENTAL モードを自動的に決定することを指定します。

REBUILD
すべてのインデックスが 再構築されることを指定します。

INCREMENTAL
インデックスが新しいデータで拡張されることを指定します。

DEFERRED
ロードユーティリティがインデックスの 作成を試行しないことを指定します。インデックスは更新が必要としてマークされます。

詳細

このオプションの値の使用方法の詳細については、*IBM DB2 Universal Database Data Movement Utilities Guide and Reference* を参照してください。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)

BL_INTERNAL_STAGE= テーブルオプション

Snowflake へのバルクロードの内部ステージとパスを指定します。

カテゴリ:	バルクロード
別名:	BL_STAGE=
デフォルト:	なし
制限事項:	このテーブルオプションは、CAS サーバーではサポートされていません。
要件:	このテーブルオプションは、Snowflake への内部ステージバルクロードを有効にするために必要です。 このオプションを指定するには、BULKLOAD=YES または BULKUNLOAD=YES も設定する必要があります。

“BULKOPTS= テーブルオプション” (1079 ページ) 内で指定する必要があります。

データソース: Snowflake

注: このオプションのサポートは 2024.11 から開始されます。

構文

BL_INTERNAL_STAGE=*stage-and-path*

必須引数

stage-and-path
Snowflake の内部ステージとパスを指定します。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)

BL_KEY= テーブルオプション

キーベースのアクセスコントロールで使用される Amazon Web Services アクセスキーを指定します。

カテゴリ: バルクロード

デフォルト: なし

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件: このオプションを指定するには、BULKLOAD=YES または BULKUNLOAD=YES も設定する必要があります。
“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。

データソース: Impala

注: Amazon Redshift のサポートは、2026.08 で追加されました。
Impala のサポートは 2025.09 から始まります。

ヒント: 一時トークン資格情報を使用している場合、これが一時アクセスキー ID になります。

構文

BL_KEY=*key-value*

必須引数

key-value
AWS 環境へのアクセスに使用する AWS キーを指定します。

BL_LOAD= テーブルオプション

指定された行数がロードされた後、ロードを停止する必要があることを指定します。

カテゴリ: バルクロード

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件: このオプションを指定するには、BULKLOAD=YES または BULKUNLOAD=YES も設定する必要があります。
“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。

データソース: Oracle

構文

BL_LOAD= *number-of-rows*

引数

number-of-rows
ロードする行数を指定します。テーブルの最初の行がロードされます。デフォルトでは、すべての行をロードします。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)

BL_LOAD_REPLACE= テーブルオプション

バルクロード中に DB2 が行を追加するか置換するかを指定します。

カテゴリ: バルクロード

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件: このオプションを指定するには、BULKLOAD=YES または BULKUNLOAD=YES も設定する必要があります。

“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。

データソース: DB2

構文

BL_LOAD_REPLACE= NO | YES

引数

NO
CLI LOAD インターフェイスが新しいデータ行を DB2 テーブルに追加することを指定します。これがデフォルト値です。

YES
CLI LOAD インターフェイスがテーブル内の既存のデータを置き換えることを指定します。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)

BL_LOG= テーブルオプション

バルクロードの統計量やエラー情報などの情報を含むログファイルを識別します。

カテゴリ: バルクロード

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件: このオプションを指定するには、BULKLOAD=YES または BULKUNLOAD=YES も設定する必要があります。
“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。

データソース: DB2、Oracle

構文

BL_LOG= "*path-and-log-file-name*"

引数

path-and-log-file-name
ロードプロセスに関する情報が書き込まれるファイルです。デフォルトのパスとログファイル名は DBMS 固有です。

詳細

DBMS バルクロード機能が呼び出されると、ログファイルが作成されます。ログファイルの内容は DBMS 固有です。BL_接頭辞は、このログファイルを SAS ログによって作成されたものと区別します。BL_LOG=テーブルオプションが既存のログと同じパスとファイル名で指定されている場合、既存のログは新しいログに置き換えられます。

BL_LOG=テーブルオプションが指定されていない場合、ログファイルは操作の成功後に自動的に削除されます。

例

BL_LOG=テーブルオプションは、BULKOPTS=テーブルオプション内で指定されます。

```
bulkload=yes; bulkopts=(bl_log="/tmp/bulkload.log");
```

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)

BL_LOGFILE= テーブルオプション

バルクロードログファイルのファイル名を指定します。

カテゴリ:	バルクロード
制限事項:	このテーブルオプションは、CAS サーバーではサポートされていません。
要件:	このオプションを指定するには、BULKLOAD=YES または BULKUNLOAD=YES も設定する必要があります。 “BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。
データソース:	Oracle

構文

BL_LOGFILE= '*log-file-name*'

引数

log-file-name
バルクロードログファイルの名前を指定します。デフォルトは、テンプレート BL_*TablenameUniquenumber* を持つ生成されたファイル名です。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)

BL_NUM_DATAFILES= テーブルオプション

ソースデータを含むように作成するデータファイルの数を指定します。

カテゴリ:	バルクロード
別名:	BL_NUM_DATA_FILES
デフォルト:	1
制限事項:	このテーブルオプションは、CAS サーバーではサポートされていません。
要件:	このオプションを指定するには、BULKLOAD=YES または BULKUNLOAD=YES も設定する必要があります。 “BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。
データソース:	Snowflake
注:	このオプションのサポートは 2024.11 から開始されます。

構文

BL_NUM_DATAFILES= *value*

引数

value
ソースデータを保持するために作成するファイルの数を指定します。この値には 1 以上 100 以下の整数を指定できます。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)

BL_OPTIONS= テーブルオプション

オプションを DBMS バルクロード機能に渡し、データのロードと処理の方法に影響を与えます。

カテゴリ:	バルクロード
別名:	RS_OPTIONS (Amazon Redshift のみ)
制限事項:	このテーブルオプションは、CAS サーバーではサポートされていません。
要件:	このオプションを指定するには、BULKLOAD=YES または BULKUNLOAD=YES も設定する必要があります。 “BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。
データソース:	Amazon Redshift, DB2
注:	Amazon Redshift のサポートは、2026.08 で追加されました。

構文

BL_OPTIONS= '*option* [, ...*option*] '

引数

option
有効な DB2 オプションを指定します。デフォルトでは、オプションは渡されません。

詳細

BL_OPTIONS=テーブルオプションを使用すると、DBMS バルクロード機能が呼び出されたときにオプションを渡すことができるため、データのロードおよび処理方法に影響を与えます。複数のオプションをカンマで区切り、オプションの文字列全体を単一引用符で囲む必要があります。

このオプションは、DB2 ファイルタイプ修飾子を DB2 LOAD または IMPORT コマンドに渡し、データのロードおよび処理方法に影響を与えます。すべての DB2 ファイルタイプ修飾子がすべての状況に適しているわけではありません。.IXF ファイルで 1 つ以上の DB2 ファイルタイプ修飾子を指定できます。ファイルタイプ修飾子のリストについては、*DB2 Data Movement Utilities Guide and Reference* の LOAD および IMPORT ユーティリティの説明を参照してください。

例

このオプションは、BULKOPTS=テーブルオプション内で指定されます。

```
bulkload=yes; bulkopts=(bl_options='option1, option2');
```

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)
- “BULKOPTS= テーブルオプション” (1079 ページ)

BL_PARALLEL= テーブルオプション

並列バルクロードを実行するかどうかを指定します。

カテゴリ: バルクロード

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件: このオプションを指定するには、BULKLOAD=YES または BULKUNLOAD=YES も設定する必要があります。
“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。

データソース: Oracle

構文

BL_PARALLEL= YES | NO

引数

YES
並列ロードを実行する必要があることを指定します。

NO
並列ロードを実行しないことを指定します。これがデフォルトの動作です。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)

BL_PORT_MAX= テーブルオプション

同時アップロードに使用できる最大のポート番号を設定します。

カテゴリ: バルクロード

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件: このオプションを指定するには、BULKLOAD=YES または BULKUNLOAD=YES も設定する必要があります。
“BULKOPTS= テーブルオプション” (1079 ページ) 内で指定する必要があります。

データソース: DB2

構文

BL_PORT_MAX= *integer*

引数

integer
同時アップロードに使用できる最大のポート番号を表す正の整数を指定します。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)

BL_PORT_MIN= テーブルオプション

同時アップロードに使用できる最小のポート番号を設定します。

カテゴリ: バルクロード

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件: このオプションを指定するには、BULKLOAD=YES または BULKUNLOAD=YES も設定する必要があります。
“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。

データソース: DB2

構文

BL_PORT_MIN= *integer*

引数

integer
同時アップロードに使用できる最小のポート番号を表す正の整数を指定します。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)

BL_RECOVERABLE= テーブルオプション

LOAD プロセスが回復可能かどうかを指定します。

カテゴリ: バルクロード

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件: このオプションを指定するには、BULKLOAD=YES または BULKUNLOAD=YES も設定する必要があります。
“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。

データソース: DB2、Oracle

構文

BL_RECOVERABLE= YES | NO

引数

YES
LOAD プロセスが回復可能であることを指定します。DB2 の場合、YES は、BL_COPY_LOCATION=がデータのコピー場所を指定する必要があることも指定します。

NO
LOAD プロセスが回復不可能であることを指定します。

詳細

DB2: デフォルトは NO です。

Oracle: デフォルトは YES です。直接ロードのパフォーマンスを向上させるには、このオプションを NO に設定します。NO を指定すると、制御ファイル内の LOAD キーワードの前に UNRECOVERABLE キーワードが追加されます。

注意
回復不可能なロードでは、ロードされたデータが REDO ログファイルに記録されないことに注意してください。したがって、ロードされたテーブルのメディアリカバリは無効になります。 Oracle で UNRECOVERABLE パラメーターを使用した場合の影響の詳細については、Oracle ユーティリティのドキュメントを参照してください。

関連項目

テーブルオプション:

- “BL_COPY_LOCATION= テーブルオプション” (1041 ページ)
- “BULKLOAD= テーブルオプション” (1074 ページ)

BL_REGION= テーブルオプション

データを取得する Amazon Web Services(AWS)リージョンを指定します。

カテゴリ:	バルクロード
デフォルト:	なし
制限事項:	このテーブルオプションは、CAS サーバーではサポートされていません。
要件:	このオプションを指定するには、BULKLOAD=YES または BULKUNLOAD=YES も設定する必要があります。 “BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。
データソース:	Impala
注:	Amazon Redshift のサポートは、2026.08 で追加されました。 Impala のサポートは 2025.09 から始まります。

構文

BL_REGION=*region*

必須引数

region
S3 データのロード元の AWS リージョンを指定します。S3 ツールを使用してデータをロードするときは、リージョンを指定する必要があります。AWS リージョンの詳細については、 COPY Parameter Reference を参照してください。

BL_REMOTE_FILE= テーブルオプション

DB2 LOAD 一時ファイルの基本ファイル名と場所を指定します。

カテゴリ: バルクロード

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件: このオプションを指定するには、BULKLOAD=YES または BULKUNLOAD=YES も設定する必要があります。
“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。

データソース: DB2

構文

BL_REMOTE_FILE= *pathname-and-base-filename*

引数

pathname-and-base-filename
ロード操作中に一時ファイルを作成するために DB2 が拡張子(.log、.msg、.dat ファイルなど)を追加する完全なパス名と基本ファイル名を指定します。デフォルトでは、BL_<*table*>_<*unique-ID*>が基本ファイル名の形式です。

table
テーブル名を指定します。

unique-ID
特定のテーブルの 2 つ以上の同時バルクロードが発生した場合に衝突を防ぐ番号を指定します。テーブルドライバーは、この番号を生成します。

詳細

このオプションを指定すると、(IMPORT コマンドのかわりに) DB2 LOAD コマンドが使用されます。

pathname には、単一の DB2 サーバーインスタンスによって排他的にアクセスされ、インスタンス所有者が読み取りおよび書き込みアクセス許可を持つ DB2 サーバー上の場所を指定します。各 LOAD コマンドが一意の *pathname-and-base-filename* 値に関連付けられていることを確認してください。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)

BL_SECRET= テーブルオプション

データのバルクロードに使用されるシークレットアクセスキーを指定します。

カテゴリ: バルクロード

デフォルト: なし

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件: このオプションを指定するには、BULKLOAD=YES または BULKUNLOAD=YES も設定する必要があります。
“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。

データソース: Impala

注: Impala のサポートは 2025.09 から始まります。

構文

BL_SECRET=*secret-access-key*

必須引数

secret-access-key
データソースにアクセスするための秘密アクセスキー値を指定します。

ヒント セキュリティを強化するには、PROC PWENCODE を使用してキー値をエンコードします。

詳細

PROC PWENCODE を使用して秘密をエンコードすると、SAS/ACCESS は値をデータソースに渡す前にデコードします。

この値は、Amazon Web Services(AWS)の秘密アクセスキーまたは一時的な秘密アクセスキーです。AWS シークレットアクセスキーは、BL_KEY= LIBNAME オプションまたはデータセットオプションで指定したキー ID に関連付けられます。一時的なトークン認証情報を使用している場合、この値は一時的な秘密アクセスキーです。

BL_SKIP= テーブルオプション

バルクロードを開始する前に、指定された行数をスキップするように指定します。

カテゴリ: バルクロード

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件: このオプションを指定するには、BULKLOAD=YES または BULKUNLOAD=YES も設定する必要があります。
“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。

データソース: Oracle

構文

BL_SKIP= *number-of-rows*

引数

number-of-rows
ロードを開始する前にスキップする行数を指定します。デフォルトは 0 です。これは、レコードがスキップされないことを意味します。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)

BL_SKIP_INDEX_MAINTENANCE= テーブルオプション

バルクロードでインデックスメンテナンスを実行するかどうかを指定します。

カテゴリ: バルクロード

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件: このオプションを指定するには、BULKLOAD=YES または BULKUNLOAD=YES も設定する必要があります。
“BULKOPTS= テーブルオプション” (1079 ページ) 内で指定する必要があります。

データソース: Oracle

構文

BL_SKIP_INDEX_MAINTENANCE= YES | NO

引数

YES
ロード時にインデックスメンテナンスを停止するように指定します。これにより、インデックスキーが追加されたはずのインデックスパーティションがインデックス使用禁止状態とマークされます。インデックスセグメントが、インデックスを作成するデータと矛盾しています。ロードの影響を受けないインデックスセグメントは、ロード前のインデックス使用禁止状態を保持します。

NO
ロード時にインデックスメンテナンスを実行するように指定します。これがデフォルト値です。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)

BL_SKIP_UNUSABLE_INDEXES= テーブルオプション

使用できない状態のインデックスエントリをスキップして、バルクロードを続行するかどうかを指定します。

カテゴリ: バルクロード

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件: このオプションを指定するには、BULKLOAD=YES または BULKUNLOAD=YES も設定する必要があります。
“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。

データソース: Oracle

構文

BL_SKIP_UNUSABLE_INDEXES= YES | NO

引数

YES
使用できない状態のインデックスエントリをスキップすることを指定します。これがデフォルト値です。

NO
使用できない状態のインデックスエントリをスキップしないことを指定します。

詳細

使用できない状態のインデックスが検出された場合、デフォルトではスキップされ、ロード操作が続行されます。これにより、SQL*Loader は、ロードを開始する前に、使用できない状態にあるインデックスを含むテーブルをロードできます。ロード時に使用できない状態でないインデックスは、SQL*Loader によってメンテナンスされます。ロード時に使用できない状態にあるインデックスはメンテナンスされず、ロードの完了時に使用できない状態のままになります。

このバルクロードオプションが指定されていない場合、デフォルト値が Oracle データベース構成パラメーター SKIP_UNUSABLE_INDEXES で指定されます。この値は、初期化パラメーターファイルで指定されます。BL_SKIP_UNUSABLE_INDEXES バルクロードテーブルオプションは、初期化パラメーターファイルの SKIP_UNUSABLE_INDEXES 構成パラメーターの値をオーバーライドします。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)

BL_TIMEOUT= テーブルオプション

ストレージタスクの最大完了時間を秒単位で指定します。

カテゴリ:	バルクロード
デフォルト:	60 秒
制限事項:	このテーブルオプションは、CAS サーバーではサポートされていません。
要件:	このオプションを指定するには、BULKLOAD=YES または BULKUNLOAD=YES も設定する必要があります。 “BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。
データソース:	Snowflake
注:	このオプションのサポートは 2024.11 から開始されます。

構文

BL_TIMEOUT= *value*

引数

value
ストレージタスクの最大完了時間を秒数で指定します。タスクが指定された時間内に完了しない場合、ストレージシステムはエラーを返すことがあります。

詳細

このオプションは、 ADLS の場所にある Snowflake へのデータのバルクロードをサポートします。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)

BL_TOKEN= テーブルオプション

BL_KEY=および BL_SECRET=オプションで指定する一時資格情報に関連付けられる一時トークンを指定します。

カテゴリ:	バルクロード
デフォルト:	なし
制限事項:	このテーブルオプションは、CAS サーバーではサポートされていません。
要件:	このオプションを指定するには、BULKLOAD=YES または BULKUNLOAD=YES も設定する必要があります。 “BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。
データソース:	Impala
注:	Impala のサポートは 2025.09 から始まります。 トークンが長すぎるというメモを受け取った場合、は無視しても問題ありません。

構文

BL_TOKEN=*temporary-token*

必須引数

temporary-token
BL_KEY=オプションと BL_SECRET=LIBNAME オプションで提供する一時資格情報に関連付けられる一時トークンを指定します。

詳細

Amazon Web Services 環境へのアクセスに使用する一時認証情報を取得すると、AWS キー、AWS シークレットキー、および AWS トークンが得られます。これらの認証情報は、限られた期間のみ有効です。

BL_USE_ESCAPE= テーブルオプション

バルクロードまたはバルク取得中に、事前定義された特殊文字のセットをエスケープするかどうかを指定します。

カテゴリ: バルクロード

別名: RS_ESCAPE (Amazon Redshift のみ)

デフォルト: NO

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件: このオプションを指定するには、BULKLOAD=YES または BULKUNLOAD=YES も設定する必要があります。
“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。

データソース: Amazon Redshift, Snowflake

注: ユーザー定義の区切り文字は、BL_DELIMITER テーブルオプションで指定されます。BL_DELIMITER=の値を指定した場合、その値もエスケープ文字として扱われます。
Snowflake のサポートは、2024.11 から開始されます。

構文

BL_USE_ESCAPE= YES | NO

引数

YES
バックスラッシュ('\')、改行('\n')、復帰('\r')、またはユーザー定義の区切り文字の前にバックスラッシュが挿入されることを指定します。バックスラッシュは、バルクロードまたはデータ取得に使用されるデータファイルでこれらの出現を保持します。NULL 文字(0x00)の出現は、すべてスペース (' ')で置換されます。

NO
バルクロードまたはデータ取得に使用されるデータファイル内で、バックスラッシュ('\')、改行('\n')、復帰('\r')、またはユーザー定義の区切り文字がエスケープされないことを指定します。

詳細

このオプションを使用すると、エスケープ文字のセットの前にエスケープ文字を追加してから、最終的なデータセットで再度削除する必要があるため、データのバルクロードの時間が長くなる可能性があります。

Snowflake のバルクロード操作では、CSV ファイルを使用します。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)

BL_WARNING_COUNT= テーブルオプション

ロードが失敗する前に許可する行警告の最大数を指定します。

カテゴリ:	バルクロード
制限事項:	このテーブルオプションは、CAS サーバーではサポートされていません。
要件:	このオプションを指定するには、BULKLOAD=YES または BULKUNLOAD=YES も設定する必要があります。 “BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。
データソース:	DB2

構文

BL_WARNING_COUNT= *number-of-warnings*

引数

number-of-warnings
ロードが失敗する前に許可する行警告の最大数を指定します。

詳細

このオプションを指定するには、最初に BULKLOAD=YES を設定し、BL_REMOTE_FILE=の値も指定する必要があります。

このオプションを使用して、警告を生成する行の最大数を制限します。行が警告を生成した理由については、ログファイルを参照してください。

関連項目

テーブルオプション:

- “BL_REMOTE_FILE= テーブルオプション” (1064 ページ)
- “BULKLOAD= テーブルオプション” (1074 ページ)

BUFNO= テーブルオプション

SAS データセットを処理するために割り当てられるバッファーの数を指定します。

カテゴリ: テーブルコントロール

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

データソース: SAS データセット

構文

BUFNO=*n* | *n*K | *hex*X | MIN | MAX

引数

n | *n*K
バッファーの数を 1 (バイト)または 1,024 (KB)の倍数で指定します。たとえば、値 **8** は 8 個のバッファーを指定し、値 **1K** は 1024 個のバッファーを指定します。

要件 *K* は大文字でなければなりません。

*hex*X
バッファーの数を 16 進値で指定します。数値(0-9)で始まり、その後に X が続く値を指定する必要があります。たとえば、値 **2dX** は、バッファーの数を 45 バッファーに設定します。

要件 *X* は大文字でなければなりません。

MIN
バッファーの最小数を 0 に設定します。これにより、SAS は動作環境に最適な最小値を使用します。これがデフォルト値です。

MAX
バッファーの数を、動作環境で可能な最大数(最大の 4 バイトの符号付き整数まで)に設定します。最大の 4 バイトの符号付き整数は 2^{31}-1、つまり約 20 億です。

詳細

バッファー数は、データセットの永続的な属性ではありません。現在の操作に対してのみ有効です。

BUFNO=テーブルオプションは、入力、出力、または更新用に開かれた SAS データセットに適用されます。

バッファーの数を増やすと、特定の SAS データセットに必要な入出力(I/O)操作の数が制限されるため、実行時間が短縮されます。ただし、実行時間の向上は、メモリ消費量の増加という犠牲を伴います。

小さなデータセットの I/O 操作を減らし、実行時間を短縮するには、処理するデータのページごとに 1 つのバッファーを割り当てます。この手法は、処理中に同じオブザベーションを数回読み取る場合に最も効果的です。

BUFSIZE= テーブルオプション

出力 SAS データセットの永久バッファーページのサイズを指定します。

カテゴリ: テーブルコントロール

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。
出力データセットでのみ使用します。

データソース: SAS データセット

構文

BUFSIZE=*n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

引数

n | *n*K | *n*M | *n*G
ページサイズを 1 (バイト)、1,024 (KB)、1,048,576 (MB)、1,073,741,824 (GB)のいずれかの倍数で指定します。たとえば、値 **8** はページサイズ 8 バイトを指定し、値 **4K** はページサイズ 4096 バイトを指定します。

要件 *K*、*M*、および *G* は大文字にする必要があります。

*hex*X
ページサイズを 16 進値で指定します。数値(0-9)で始まり、その後に X が続く値を指定する必要があります。たとえば、値 **2dX** は、ページサイズを 45 バイトに設定します。

要件 *X* は大文字でなければなりません。

MIN
バッファーの最小数を 0 に設定します。これにより、SAS は動作環境に最適な最小値を使用します。

MAX
ページサイズを、動作環境で可能な最大数(最大の 4 バイトの符号付き整数まで)に設定します。最大の 4 バイトの符号付き整数は 2^{31}-1、つまり約 20 億バイトです。

詳細

ページサイズは、1 回の I/O 操作で 1 つのバッファーに転送できるデータの量です。ページサイズは、データセットの永続的な属性であり、データセットの処理時に使用されます。

ページサイズを大きくすると、SAS が記憶媒体から読み書きする回数が減るため、実行時間が短縮されます。ただし、実行時間の向上は、メモリ消費量の増加という代償を伴います。

ページサイズを変更するには、データセットをコピーし、新しいページを指定するか、SAS のデフォルトを使用します。ページサイズを動作環境のデフォルト値にリセットするには、BUFSIZE=0 を指定します。

動作環境の情報: BUFSIZE=テーブルオプションのデフォルト値は、動作環境によって決まり、順次アクセスを最適化するように設定されています。直接(ランダム)アクセスのパフォーマンスを向上させるには、BUFSIZE=の値を変更する必要があります。

BULKLOAD= テーブルオプション

SAS が DBMS 機能を使用して DBMS テーブルにデータを挿入するかどうかを決定します。

カテゴリ: バルクロード

デフォルト: NO

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件 ターゲットデータソースに適切な SAS/ACCESS ソフトウェアを構成する必要があります。

操作: “BULKOPTS= テーブルオプション” (1079 ページ)と組み合わせて使用されることがよくあります。

データソース: Amazon Redshift、DB2、Google BigQuery、Impala、 MDS、Microsoft SQL Server、MySQL、PostgreSQL、Oracle、SingleStore、Snowflake、Teradata

注: Amazon Redshift および Microsoft SQL Server でのバルクロードのサポートは、2024.07 から開始されます。

MySQL および SingleStore でのバルクロードのサポートは、2024.10 から開始されます。

Snowflake でのバルクロードのサポートは、2024.11 から開始されます。

PostgreSQL でのバルクロードのサポートは、2024.12 から開始されます。

参照項目: バルクロードテーブルオプションについて

構文

BULKLOAD= YES | NO

引数

YES
DBMS テーブルに行を挿入または追加するために、DBMS 固有のバルクロード機能を呼び出します。

NO
DBMS 固有のバルクロード機能を呼び出しません。

詳細

概要

バルクロードを使用するのが、DBMS テーブルに行を挿入する最も速い方法です。SAS/ACCESS 接続のバルクロードを有効にするには、BULKLOAD=YES を指定します。

LIBNAME ステートメントで BULKLOAD=YES が指定されている場合、一部の SAS/ACCESS エンジンでは、PROC FEDSQL のバルクロードが有効になります。これらのエンジンは、プロシジャ用に生成する接続文字列で内部接続オプションを渡します。これらのデータソースでは、BULKLOAD=テーブルオプションを使用して、LIBNAME ステートメントの設定をオーバーライドできます。

その他のデータソースでは、BULKLOAD=テーブルオプションがバルクロードを有効にする方法です。これらのデータソースのほとんどでは、BULKLOAD=テーブルオプションを YES に設定するだけでバルクロードが有効になります。ただし、一部のデータソースでは、BULKLOAD=YES に加えて、データソース固有のテーブルオプションを指定する必要があります。データソース固有のオプションは、BULKOPTS=テーブルオプション(2025.05 以前のリリースまで Oracle 向けのこのオプションは BL_BULKLOPTS=)で指定されます。 BULKOPTS=は、データソース固有のオプションのコンテナーとして機能します。どのデータソースにバルクロードのためのデータソース固有のテーブルオプションがあるかについては、“データソース別の FedSQL ステートメントテーブルオプション” (1016 ページ)を参照してください。

BULKLOAD=NO の場合、単純な複数行挿入 SQL スキームを使用してデータ行が挿入されます。

使用上の注意

Amazon Redshift

バルクロード機能は、Amazon Simple Storage Service (S3)ツールを使用してデータを Amazon Redshift データベースに移動します。バルクロードを有効にするには、BULKLOAD=テーブルオプションを YES に設定する必要があります。バルクロードのデータソース固有のオプションが必須であり、BULKOPTS=オプションで指定する必要があります。オプションの使用方法については、"Bulk Loading and Bulk Unloading for Amazon Redshift" (*SAS/ACCESS for Relational Databases: Reference*)を参照してください。テーブルオプションの指定方法の例については、Amazon Redshift の BULKOPTS=の例 (1080 ページ)を参照してください。

Google BigQuery

バルクロードは、BIGQUERY エンジンの LIBNAME ステートメントで、または BULKLOAD=YES テーブルオプションを指定することによって有効にできます。

データソース固有のテーブルオプションは、Google BigQuery へのバルクロードで使用できますが、必須ではありません。LIBNAME ステートメントのデータソース固有のオプションは、接続オプションとして内部的に FEDSQL プロシジャに渡されます。BULKLOAD=テーブルオプションを使用してバルクロードを有効にする場合は、BULKOPTS=内でデータソース固有のテーブルオプションを指定する必要があります。

詳細については、"Bulk Loading and Bulk Unloading for Google BigQuery" (*SAS/ACCESS for Relational Databases: Reference*)と Google BigQuery の BULKOPTS=の例 (1080 ページ)を参照してください。

Impala

Impala でバルクロードを有効にするには、BULKLOAD=テーブルオプションを YES に設定し、適切なシステムオプションを設定する必要があります。Impala サーバーへのバルクロードは、次の 2 つの方法で実行できます。Hadoop への WebHDFS インターフェイスを使用してデータを HDFS にプッシュする方法と、Hadoop JAR ファイルの必要なセットを構成する方法です。どちらのアプローチでも、SAS が必要とする Hadoop 構成ファイルを 1 つの場所に配置し、クライアントマシンで使用できるようにする必要があります。WebHDFS を使用するには、SAS_HADOOP_RESTFUL=環境変数も 1 に設定する必要があります。Java を使用するには、Hadoop JAR の場所をクライアントマシンに認識させ、SAS_HADOOP_RESTFUL=環境変数が 1 (または TRUE または YES)に設定されていないことを確認する必要があります。

BULKLOAD=YES を指定すると、2 つの CREATE TABLE ステートメントが Impala サーバーに発行されます。1 つはターゲットの Impala テーブルを作成します。もう 1 つは一時テーブルを作成します。SAS は WebHDFS を使用して、テーブルデータを HDFS の/tmp ディレクトリにアップロードします。結果のファイルは、UTF-8 区切りテキストファイルです。SAS は LOAD DATA ステートメントを発行して、データファイルを/tmp ディレクトリから一時テーブルに移動します。次に、テキストデ

ータを一時テーブルからターゲットテーブルにコピーして 変換する INSERT INTO ステートメントを発行します。その後、一時テーブルは HDFS から削除されます。

Impala のバルクロードオプションの指定方法の例については、Impala の BULKOPTS=の例 (1081 ページ)を参照してください。

Microsoft SQL Server

テーブルオプション BULKLOAD=YES は、SQL Server データベースのバルクロードを有効にします。このオプションは、基になる DataDirect ドライバーで属性 SQL_ATTR_BULK_LOAD_ENABLED=1 を設定します。Microsoft SQL Server へのバルクロードには、データソース固有のテーブルオプションはありません。

MDS

テーブルオプション BULKLOAD=YES は、MDS のバルクロードを有効にします。BULKLOAD=テーブルオプションが YES に設定されている場合、新しく挿入された行はすぐにコミットされ、既存のトランザクションで表示されるようになります。BULKLOAD=テーブルオプションが NO に設定されるか、設定されていない場合、新しく挿入された行は、既存のトランザクションがコミットまたはロールバックされるまで表示されません。MDS へのバルクロードには、データソース固有のテーブルオプションはありません。

MySQL

バルクロードは、MYSQL エンジンの LIBNAME ステートメントで、または BULKLOAD=テーブルオプションで YES を指定することによって有効にできます。MySQL へのバルクロードには、データソース固有のテーブルオプションはありません。

バルクロード用の中間ファイルは、UTILLOC=システムオプションで指定された場所に作成されます。

PostgreSQL

バルクロードは、POSTGRES エンジンの LIBNAME ステートメントで、または BULKLOAD=テーブルオプションで YES を指定することによって有効にできます。PostgreSQL でバルクロードを有効にするために必要なデータソース固有のテーブルオプションはありません。

SingleStore

バルクロードは、SSTORE エンジンの LIBNAME ステートメントで、または BULKLOAD=テーブルオプションで YES を指定することによって有効にできます。SingleStore へのバルクロードには、データソース固有のテーブルオプションはありません。

バルクロード用の中間ファイルは、UTILLOC=システムオプションで指定された場所に作成されます。

Snowflake

バルクロードは、Snowflake 内部ステージおよび ADLS の Snowflake で使用できます。バルクロードは、SNOW エンジンの LIBNAME ステートメントで BULKLOAD=YES を指定するか、BULKLOAD=テーブルオプションを使用して有効にできます。

バルクロードのために Snowflake に接続するには、データソース固有のオプションが必要です。LIBNAME ステートメントのデータソース固有のオプションは、接続オプションとして内部的に FEDSQL プロシジャに渡されます。テーブルオプションを使用してバルクロードを有効にする場合は、BULKOPTS=テーブルオプションで適切なデータソース固有のテーブルオプションを指定する必要があります。詳細については、"Authentication for Bulk Loading to and Bulk Unloading from Azure" (*SAS/ACCESS for Relational Databases: Reference*)を参照してください。バルクロードテーブルオプションの指定方法の例については、Azure ADLS での Snowflake の BULKOPTS=の例 (1081 ページ)と the Snowflake のステージバルクロードの BULKOPTS=の例 (1081 ページ)を参照してください。

Teradata

BULKLOAD=YES を指定すると、Teradata Parallel Transporter (TPT) API プロトコルドライバーが呼び出されます。デフォルトの TPT オペレーターは Stream オペレーターです。別の TPT オペレーターを指定するには、TD_TPT_OPER=テーブルオプションを使用します。Teradata へのバルクロードには、いくつかのデータソース固有のテーブルオプションが使用できます。テーブルオプションの指定方法の例については、Teradata の BULKOPTS=の例 (1081 ページ)を参照してください。オプションの使用方法については、SAS/ACCESS のドキュメントを参照してください。

例: バルクロードの有効化

多くのデータソースでは、BULKLOAD=YES テーブルオプションを指定するだけでバルクロードを有効にできます。バルクプロセスによって作成された中間ファイルは、デフォルトで削除されます。

```
create table accesslib.myexample{options BULKLOAD=YES}
as select * from otherlib.other-table;

insert into accesslib.mytable{options BULKLOAD=YES}
select * from otherlib.othertable;
```

関連項目

概念:

- "データソース別の FedSQL ステートメントテーブルオプション" (1016 ページ)

テーブルオプション:

- "BULKOPTS= テーブルオプション" (1079 ページ)

BULKOPTS= テーブルオプション

バルクロードオプションのコンテナーを提供します。

カテゴリ:	バルクロード
制限事項:	このテーブルオプションは、CAS サーバーではサポートされていません。
要件	BULKOPTS=を使用するには、BULKLOAD=YES または BULKUNLOAD=YES を設定する必要があります。
データソース:	Amazon Redshift、DB2、Google BigQuery、Impala、Oracle、Snowflake、Teradata
注:	2025.05 以前のリリースでは、Oracle のこのオプションは BL_BULKLOPTS=という名前です。2025.06 以降、Oracle 向けのこのオプションは BULKOPTS=です。

構文

BULKOPTS= (*option*[...*option*])

引数

option[...option]
スペースで区切られた 1 つ以上のバルクロードテーブルオプションを指定します。

詳細

BULKOPTS=テーブルオプションは、バルクロードのデータソース固有のオプションを指定するためのコンテナーオプションです。すべてのデータソースに、バルクロードのためのデータソース固有のテーブルオプションがあるわけではありません。各データソースにこのコンテナーで指定できるオプションのリストについては、“データソース別の FedSQL ステートメントテーブルオプション” (1016 ページ)を参照してください。

バルクロードのデータソース固有のオプションの一部はオプションです。ただし、一部のデータソースでは、バルクロードを有効にするためにデータソース固有のテーブルオプションが必要です(Amazon Redshift や Snowflake など)。必要なオプションにより、バルクロードのステージやホスティングプラットフォームへの認証が指定されします。

例

例 1: Amazon Redshift

次の例では、BULKLOAD=YES および BULKOPTS= オプションを使用して、Amazon Redshift S3 バケットのバルクロードを有効にします。BL_BUCKET= と BL_REGION= は必須オプションです。その他は認証オプションです。つまり、次に示すオプションを使用して認証することも、BL_KEY=、BL_SECRET=、BL_TOKEN= を使用して認証することもできます。

```
BULKLOAD=YES BULKOPTS=(BL_AWS_CONFIG_FILE='path-and-file-name'
BL_AWS_CREDENTIALS_FILE='path-and-file-name' BL_REGION='region-value'
BL_BUCKET='bucket-name')
```

注: 2026.08 以降、Amazon Redshift データソースの BULKOPTS= オプションは、前の例に示すように、該当する BL テーブルオプションを受け入れます。下位互換性のために、RS テーブルオプションは引き続きサポートされています。Amazon Redshift のテーブルオプションの“データソース別の FedSQL ステートメントテーブルオプション” (1016 ページ)を参照してください。

例 2: DB2

この例では、バルクロードオプションを DB2 にサブミットします。

```
BULKLOAD=YES BULKOPTS=(BL_LOG='/tmp/bulkload.log'  BL_LOAD_REPLACE=yes
    BL_OPTIONS='ERRORS=999, LOAD=2000');
```

例 3: Google BigQuery

Google BigQuery ローダークライアントアプリケーションは、クライアントから Google BigQuery データベースにデータを移動するために使用されます。有効なバルクロードテーブルオプションは、*SAS/ACCESS for Relational Databases: リファレンス*の“Google BigQuery でのデータのバルクロード”に記載されている LIBNAME オプションと同じです。これらのバルクロードオプションは、中間ファイルの処理のプリファレンスを設定します。

```
BULKLOAD=YES BULKOPTS=(BL_DELETE_DATA_FILE=NO;BL_DEFAULT_DIR='/u/myid/temp')
```

例 4: Oracle

この例では、バルクロードオプションを Oracle にサブミットします。

```
BULKLOAD=YES BULKOPTS=(BL_DEFAULT_DIR='/tmp/mylogs' BL_LOGFILE='novemberdatalog'
BL_ERRORS=999)
```

注: Oracle では、2025.05 以前のリリースまでオプション BL_BULKLOPTS=を使用してバルクロードオプションを指定します。2025.06 以降、オプション BULKOPTS=を使用します。

例 5: Impala

この例では、BULKOPTS=オプションを使用して、HDFS 認証オプションを Impala にサブミットします。バルクロード要求は、WebHDFS を使用してデータを HDFS にロードします。要求を成功させるには、SAS_HADOOP_RESTFUL=環境変数も 1 に設定する必要があることに注意してください。

```
BULKLOAD=YES BULKOPTS=(USER=HDFS PASS="hdfs-pwd"
CONFIG="my-config-path")
```

例 6: ADLS の Snowflake

Snowflake にバルクロードする場合は、バルクロードステージを設定する必要があります。ADLS へのバルクロードの外部ステージを設定するには、BL_ACCOUNTNAME=、BL_FILESYSTEM=、BL_APPLICATIONID=、BL_AZURE_TENANT_ID=、BL_AZURE_SAS=テーブルオプションが必要です。これらのオプションは、外部ステージのバルクロードに必要です。

```
create table accesslib.mytable {options BULKLOAD=YES BULKOPTS=(BL_DELIMITER=","
BL_FOLDER="folder-name" BL_DEFAULT_DIR="directory-path" BL_ACCOUNTNAME="account-name"
BL_FILESYSTEM="filesystem-name" BL_APPLICATIONID="application-id"
BL_AZURE_TENANTID="tenant-id" BL_AZURE_SAS="value")}
as select * from otherlib.othertable;
```

例 7: Snowflake のステージバルクロード

これらのテーブルオプションは、Snowflake の内部ステージバルクロードを有効にします。BL_INTERNAL_STAGE=は必須オプションです。複数のファイルをアップロードする場合は、BL_NUM_DATA_FILES=が必須です。

```
create table accesslib.mytable {options BULKLOAD=YES BULKOPTS=(BL_DELETE_DATA_FILE=NO;
BL_NUM_DATA_FILES=2;BL_INTERNAL_STAGE='MY_INTERNAL_STAGE/user-name')}
as select * from otherlib.othertable;
```

例 8: Teradata

次の例では、BULKLOAD=YES および BULKOPTS=オプションを使用して、Teradata へのバルクロードオプションを有効にします。BULKLOAD=YES の場合、Teradata Parallel Transporter (TPT)を使用して Teradata テーブルにデータをロードします。TPT ジョブは、デフォルトで Stream オペレーターを使用します。次の例では、TD_TPT_OPER=テーブルオプションで、かわりに Load オペレーターを使用することなどを指定しています。

```
BULKLOAD=YES BULKOPTS=(TD_TPT_OPER=LOAD TD_ERROR_LIMIT=1
TD_TRACE_OUTPUT=tpttrace TD_TRACE_LEVEL=TD_OPER_ALL
TD_MAX_SESSIONS=8 TD_TRACE_LEVEL_INF=TD_OPER_ALL)
```

関連項目

テーブルオプション:

- “TD_TPT_OPER= テーブルオプション” (1150 ページ)
- “CONFIG= テーブルオプション” (1084 ページ)

COMPRESS= テーブルオプション

新しい出力データセットまたはテーブルで行を圧縮する方法を指定します。

カテゴリ: テーブルコントロール

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。
出力テーブルのみで使用します。

データソース: SAS データセット、SPD Engine データセット、SPD Server テーブル

構文

COMPRESS= NO | YES | CHAR | BINARY

引数

CHAR | YES
新しく作成されたテーブルの行が圧縮されることを指定します(可変長レコード)。SAS データセットは、ランレングスエンコーディング(RLE)を使用して圧縮されています。RLE は、繰り返される連続する文字(空白を含む)を 2 バイトまたは 3 バイトの表現に減らすことによって、行を圧縮します。SPD Engine データセットは、ランレングス圧縮アルゴリズム SPDSRLC2 を使用して圧縮されます。SPD Server テーブルは、ランレングス圧縮アルゴリズム SPDSRLLC を使用して圧縮されます。

別名 ON (SPD Engine データセットのみ)

ヒント この圧縮アルゴリズムを文字データに使用します。

BINARY
新しく作成されたデータセットまたはテーブルの行が、ロスデータ圧縮(RDC)を使用して SAS によって圧縮されることを指定します。RDC は、ランレングスエンコーディングとスライディングウィンドウ圧縮を組み合わせてファイルを圧縮します。SPD Engine データセットは、SPDSRDC を使用して SPD Engine によって圧縮されます。

ヒント この方法は、中規模から大規模(数百バイト以上)のバイナリデータ(数値変数)ブロックを圧縮する場合に非常に効果的です。圧縮機能は一度に 1 つのレコードに対して動作するため、効果的な圧縮にはレコード長が数百バイト以上である必要があります。

NO
新しく作成されたテーブルの行が非圧縮であることを指定します(固定長レコード)。

詳細

テーブルの圧縮は、各行を表すために必要なバイト数を減らす処理です。テーブルを圧縮する利点としては、テーブルのストレージ要件が削減され、処理中にデータの読み書きに必要な I/O 操作が少なくなることが挙げられます。ただし、圧縮されたテーブルを読み取るには、より多くの CPU リソースが必要になります(各行の圧縮を解除するオーバーヘッドのため)。また、結果のファイルサイズが減少せずに増加する場合もあります。

個々のテーブルを圧縮するには、COMPRESS=テーブルオプションを使用します。出力テーブルのみ、つまり CREATE TABLE ステートメントのテーブル名に対してオプションを指定します。

注: SPD Engine データセットと SPD Server テーブルでは、暗号化と圧縮は相互に排他的です。圧縮された SPD Engine データセットまたは SPD Server テーブルの暗号化を指定することはできません。暗号化された SPD Engine データセットまたは SPD Server テーブルは圧縮できません。

テーブルが圧縮されると、設定はテーブルの永続的な属性になります。つまり、設定を変更するには、テーブルを再作成する必要があります。つまり、テーブルを圧縮解除するには、DROP TABLE ステートメントを使用してテーブルを削除し、CREATE TABLE ステートメントと COMPRESS=NO オプションを使用してテーブルを再作成する必要があります。

SPD Engine データセットまたは SPD Server テーブルを圧縮するために COMPRESS=を指定すると、テーブルドライバーは、作成時にテーブルをブロックごとに圧縮します。圧縮ブロックのサイズを指定するには、“IOBLOCKSIZE= テーブルオプション” (1104 ページ)を使用します。

SPD Engine テーブルドライバーは、SPD Engine データセットを作成または更新するときに、PADCOMPRESS=テーブルオプションもサポートします。PADCOMPRESS=オプションを使用すると、新しく圧縮されたブロックにパディングを追加できます。“PADCOMPRESS= テーブルオプション” (1111 ページ)を参照してください。

比較

COMPRESS=テーブルオプションは、COMPRESS=接続文字列オプションをオーバーライドします。

SAS data sets only: 圧縮テーブルを作成する場合、スペースを追跡して再利用するために REUSE=YES テーブルオプションを指定することもできます。REUSE=YES を指定すると、他の行が更新または削除されたときに解放されたスペースに新しい行が挿入されます。デフォルトの REUSE=NO が有効な場合、新しい行が既存のテーブルに追加されます。

関連項目

テーブルオプション:

- “ENCRYPT= テーブルオプション” (1091 ページ)
- “POINTOBS= テーブルオプション” (1114 ページ)
- “IOBLOCKSIZE= テーブルオプション” (1104 ページ)
- “PADCOMPRESS= テーブルオプション” (1111 ページ)
- “REUSE= テーブルオプション” (1120 ページ)

CONFIG= テーブルオプション

Hadoop 構成パス解決のファイル名またはパス名を指定します。

該当要素:	2026.02 以前のリリース
カテゴリ:	バルクロード
別名:	CFG=、HD_CONFIG=、CONFIGDIR=、CFGDIR=、HD_CONFIGDIR=
制限事項:	このオプションは、2026.03 以降のリリースでは使用できません。 このテーブルオプションは、CAS サーバーではサポートされていません。
要件	“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。
データソース:	Impala
注:	このオプションは、2026.01 で非推奨となり、2026.03 で削除されました。

構文

CONFIG="*configuration-path*"

引数

"*configuration-path*"
マシン上の SAS が必要とする Hadoop 構成情報を含む単一のファイルまたはディレクトリの名前を指定します。このオプションが指定されていない場合、SAS は SAS_HADOOP_CONFIG_PATH 環境変数の値を検索します。

詳細

SAS_HADOOP_CONFIG_PATH 環境変数は、複数のディレクトリ(PATH 変数と同様に定義)をリストして、構成情報を検索できます。特定の構成ファイルを使用する場合、または特定のディレクトリで構成情報を検索する場合は、CONFIG=オプションを使用します。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)

DATAFILE= テーブルオプション

一時 HDFS ファイルの代替名と場所を指定します。

カテゴリ: バルクロード

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件 “BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。

データソース: Impala

構文

DATAFILE= "*filename*"

引数

"*filename*"
ファイルへの完全なパス名を指定します。このオプションが指定されていない場合、一時ファイルのデフォルトのパス名は"/tmp/bl_*tablename_unixtimeval*.dat"です。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)

DBCREATE_INDEX_OPTS= テーブルオプション

DBMS 固有のオプションを CREATE INDEX ステートメントに追加できるようにします。

カテゴリ: インデックスコントロール

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

データソース: Microsoft SQL Server、ODBC

構文

DBCREATE_INDEX_OPTS= '*DBMS-SQL-clauses*'

引数

DBMS-SQL-clauses
SQL CREATE INDEX ステートメントの末尾に追加できる 1 つ以上の DBMS 固有の句を指定します。

詳細

このオプションを使用すると、DBMS 固有の句を CREATE INDEX ステートメントの末尾に追加できます。インターフェイスは CREATE INDEX ステートメントとその句を DBMS に渡し、DBMS はステートメントを実行して DBMS インデックスを作成します。

例

次の例では、DBCREATE_INDEX_OPTS= “as 'compact' with deferred rebuild”オプションの値を CREATE INDEX ステートメントに追加して、Hive インデックスが作成されます。

```
create index "COL1_1A03" on "TKTS002_1A03"{options DBCREATE_INDEX_OPTS="as 'compact'
with deferred rebuild"}("COL1")
```

インデックスを作成するために、次の CREATE INDEX ステートメントが DBMS に渡されます。

```
create index 'COL1_1A03' on table 'TKTS002_1A03'('COL1') as 'compact' with
deferred rebuild
```

DBCREATE_TABLE_OPTS= テーブルオプション

生成された SQL コードでデータベース固有のオプションを CREATE TABLE ステートメントの後に配置できるようにします。

カテゴリ:	テーブルコントロール
別名:	POST_STMT_OPTS= (Hive のみ)
制限事項:	このテーブルオプションは、CAS サーバーではサポートされていません。
データソース:	Amazon Redshift、DB2、Google BigQuery、Greenplum、Hive、Microsoft SQL Server、MySQL、Netezza、Oracle、SAP HANA、SingleStore、Snowflake、Spark、Teradata

構文

DBCREATE_TABLE_OPTS= '*DBMS-SQL-clauses*'

引数

DBMS-SQL-options
SQL CREATE TABLE ステートメントの末尾に追加できる 1 つ以上の DBMS 固有のオプションを指定します。

詳細

このオプションを使用すると、DBMS 固有のオプションを CREATE TABLE ステートメントの末尾に追加できます。インターフェイスは CREATE TABLE ステートメントとそのオプションを DBMS に渡し、DBMS はステートメントを実行して DBMS テーブルを作成します。

FedSQL ステートメントオプションの可用性と動作は、データソース固有のものです。DBCREATE_TABLE_OPTS=オプションが CREATE TABLE 以外のステートメントで使用されている場合、それらのオプションは無視される可能性があります。

例

例 1: Amazon Redshift

Amazon Redshift テーブルを作成するときに DBCREATE_TABLE_OPTS=がどのように使用されるかを示す例を次に示します。CREATE TABLE ステートメントに追加された DBCREATE_TABLE_OPTS='BACKUP NO'オプションの値を使用して、Redshift テーブルが作成されます。

```
CREATE TABLE temp{options dbcreate_table_opts='backup no'}
  (EMPNUM   CHAR(3) NOT NULL UNIQUE);
```

テーブルを作成するために、次の CREATE TABLE ステートメントが DBMS に渡されます。

```
CREATE TABLE "test"."DBCREATE" ("EMPNUM" char (3)  NOT NULL ,  UNIQUE ("EMPNUM") ) backup no
```

例 2: Google BigQuery

次の例では、CREATE TABLE ステートメントに"table created by SAS"という説明を追加して、Google BigQuery テーブル TEMP が作成されます。

```
create table a."my_dataset".temp {option dbcreate_table_opts='options
(description="table created by SAS")'}(col1 char(8));
```

次の CREATE TABLE ステートメントが DBMS に送信されます。

```
create table `my_project`.`my_dataset`.`TEMP` (`COL1` STRING)
 options  (description="table created by SAS")
```

例 3: Greenplum

次の例では、CREATE TABLE ステートメントに追加された DBCREATE_TABLE_OPTS='DISTRIBUTED BY ("B")'オプションの値を使用して、Greenplum テーブル TEMP が作成されます。

```
create table temp {options dbcreate_table_opts='distributed by ("b")'}
(a int, b int);
```

テーブルを作成するために、次の CREATE TABLE ステートメントが DBMS に渡されます。

```
create table temp (a int, b int) distributed by ("b")
```

例 4: Hive

次の例では、CREATE TABLE ステートメントに追加された DBCREATE_TABLE_OPTS='PARTITIONED BY ("col1")'オプションの値を使用して、Hive テーブル TEMP が作成されます。

```
create table temp {options DBCREATE_TABLE_OPTS='PARTITIONED BY (col1
int)'} (col1 int, col2 int);
```

テーブルを作成するために、次の CREATE TABLE ステートメントが DBMS に渡されます。

```
create table `temp` (`COL2` int) PARTITIONED BY (col1 int)
```

Hive では、DBCREATE_TABLE_OPTS=を関連するテーブルオプション PRE_TABLE_OPTS=および POST_TABLE_OPTS=と組み合わせて使用できます。使用方法については、“PRE_TABLE_OPTS= テーブルオプション” (1117 ページ)および “POST_TABLE_OPTS= テーブルオプション” (1116 ページ)を参照してください。DBCREATE_TABLE_OPTS=のかわりにエイリアス POST_STMT_OPTS=を使用することもできます。同じステートメントで POST_STMT_OPTS=と DBCREATE_TABLE_OPTS=を指定しないでください。同じステートメントで両方の名前を指定すると、エラーになります。

例 5: MySQL

次の例では、CREATE TABLE ステートメントに追加された DBCREATE_TABLE_OPTS='PARTITIONING KEY (x) USING HASHING'オプションの値を使用して、MySQL テーブル TEMP が作成されます。

```
create table temp {option dbcreate_table_opts='partitioning key (x) using hashing'}
  (x double);
```

テーブルを作成するために、次の CREATE TABLE ステートメントが DBMS に渡されます。

```
create table temp (x double) partitioning key (x) using hashing;
```

例 6: Netezza

次の例では、CREATE TABLE ステートメントに追加された DBCREATE_TABLE_OPTS='DISTRIBUTE ON (b)'オプションの値を使用して、Netezza テーブル TEMP が作成されます。

```
create table temp {options dbcreate_table_opts='distribute on (b)'}
```

```
(a int, b int);
```

テーブルを作成するために、次の CREATE TABLE ステートメントが DBMS に渡されます。

```
create table temp (A int, B int) distribute on (b)
```

例 7: Oracle

次の例では、CREATE TABLE ステートメントに追加された DBCREATE_TABLE_OPTS='NOLOGGING'オプションの値を使用して、Oracle テーブル TEMP が作成されます。

```
create table temp{options dbcreate_table_opts='nologging'}
(a int, b int)
```

テーブルを作成するために、次の CREATE TABLE ステートメントが DBMS に渡されます。

```
create table temp ("a" number (10) , "b" number (10) ) nologging
```

例 8: SAP HANA

次の例では、CREATE TABLE ステートメントに追加された DBCREATE_TABLE_OPTS='PARTITION BY RANGE'オプションの値を使用して、SAP HANA テーブル TEMP が作成されます。

```
create table temp {options table_type=column dbcreate_table_opts="partition by range (x)
(partition 1 <= values < 10, partition others)"}
(x int, y int);
```

テーブルを作成するために、次の CREATE TABLE ステートメントが DBMS に渡されます。

```
create column table temp (x int, y int) partition by range (x)
(partition 1 <= values < 10, partition others)
```

例 9: SingleStore

次の例では、CREATE TABLE ステートメントに追加された DBCREATE_TABLE_OPTS='COMPRESSION=SPARSE'オプションの値を使用して、SingleStore テーブル TEMP が作成されます。

```
create table temp {option dbcreate_table_opts='compression=sparse'}(x int);
```

テーブルを作成するために、次の CREATE TABLE ステートメントが DBMS に渡されます。

```
create table temp (x int) compression=sparse
```

例 10: Spark

次の例では、CREATE TABLE ステートメントに追加された DBCREATE_TABLE_OPTS='PARTITIONED BY ("B")'オプションの値を使用して、Spark テーブル TEMP が作成されます。

```
create table temp {options dbcreate_table_opts='PARTITIONED by (b int)'}
 (a int, b int);
```

テーブルを作成するために、次の CREATE TABLE ステートメントが DBMS に渡されます。

```
create table temp (a int) PARTITIONED by (b int)
```

例 11: Teradata

次の例では、CREATE TABLE ステートメントに追加された DBCREATE_TABLE_OPTS='PRIMARY INDEX (B)'オプションの値を使用して、Teradata テーブル TEMP が作成されます。

```
create table temp {options dbcreate_table_opts='primary index(b)'}
(a int, b int);
```

テーブルを作成するために、次の CREATE TABLE ステートメントが DBMS に渡されます。

```
create table temp (a int, b int) primary index(b)
```

関連項目

テーブルオプション:

- “POST_TABLE_OPTS= テーブルオプション” (1116 ページ)
- “PRE_TABLE_OPTS= テーブルオプション” (1117 ページ)
- “TABLE_TYPE= テーブルオプション” (1129 ページ)

ENCODING= テーブルオプション

SAS データセットの作成時に使用するエンコーディングを指定します。

カテゴリ: テーブルコントロール

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

データソース: SAS データセット

構文

ENCODING=*encoding-value*

引数

encoding-value
エンコーディング値を指定します。

参照項目: *SAS 各国語サポート(NLS): リファレンスガイド*の"NLS のエンコーディング"

詳細

SAS データセットは、デフォルトでセッションエンコーディングを使用して作成されます。ENCODING=テーブルオプションを使用して、新しいテーブルに別のエンコーディングを指定できます。ENCODING=は、入力に対してのみサポートされています。SELECT ステートメントで ENCODING=を指定して、既存のテーブルのエンコーディングを一時的に変更することはできません。同様に、既存のテーブルのエンコーディングを UPDATE ステートメントで指定して永続的に変更することはできません。

列を定義する前に、CREATE TABLE ステートメントで ENCODING=テーブルオプションを指定します。次に例を示します。

```
create table test {options encoding=ebcdic1145} (x char(8));
```

ENCODING=テーブルオプションは、CREATE TABLE AS ではサポートされていません。CREATE TABLE AS でテーブルオプションを指定すると、エラーが発生します。要求されたテーブルは作成されますが、セッションエンコーディングが使用されます。ただし、CREATE TABLE AS を使用して、エンコーディングが指定されている既存のテーブルを、エンコーディングが指定されていない新しいテーブルに挿入すると、新しいテーブルには挿入されたテーブルのエンコーディングが適用されます。たとえば、次の要求は、テーブル Test で使用されるエンコーディング ebcdic1145 スペイン(EBCDIC)でテーブル Test2 を作成します。

```
create table test2 as select * from test;
```

ENCRYPT= テーブルオプション

出力 SAS データセットを暗号化するかどうかを指定します。

カテゴリ: テーブルコントロール

制限事項: 出力テーブルのみで使用します。
このテーブルオプションは、CAS サーバーではサポートされていません。
SPD Engine データセットでは、このテーブルオプションを COMPRESS=と一緒に使用することはできません。

データソース: SAS データセット、SPD Engine データセット、SPD Server テーブル

構文

ENCRYPT= AES | AES2 | NO | YES

引数

AES or AES2
高度暗号化標準(AES)アルゴリズムを使用してファイルを暗号化します。ENCRYPT=AES または ENCRYPT=AES2 を使用する場合は、ENCRYPTKEY=テーブルオプションを指定する必要があります。詳細については、“ENCRYPTKEY= テーブルオプション” (1094 ページ)を参照してください。

制限事項 SPD Engine と SPD Server は AES2 暗号化をサポートしていません。

注意 **ENCRYPT=AES または ENCRYPT=AES2 を使用している場合は、すべての ENCRYPTKEY=値を記録してください。** ENCRYPTKEY=値を記録するのを忘れると、データが失われます。SAS は、ENCRYPTKEY=値の回復を支援することはできません。次の注がログに書き込まれます。

Note: If you lose or forget the ENCRYPTKEY= value, there will be no way to open the file or recover the data.

NO
データセットを暗号化しません。

YES
SASProprietary アルゴリズムを使用してデータセットを暗号化します。この暗号化法では、データセットに保存されているパスワードを使用します。少なくとも、ENCRYPT=YES を指定すると同時に、READ=または PW=テーブルオプションを指定する必要があります。詳細については、“例: ENCRYPT=YES オプションの使用” (1093 ページ)を参照してください。

注 暗号化法ではパスワードを使用するため、テーブルを再作成しないと暗号化ファイルの*いずれの*パスワードも変更することはできません。

注意 **ENCRYPT=YES を使用する場合は、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、リセットすることはできません。これは、時間とリソースを大量に消費するプロセスです。

詳細

ENCRYPT=YES を使用する場合、次のルールが適用されます。

- データファイルが暗号化されている場合、SPD Server テーブルを除いて、関連するすべてのインデックスも暗号化されます。SPD Server はテーブルインデックスまたはメタデータを暗号化しません。テーブル行データのみが暗号化されます。
- 暗号化されたデータファイルをコピーするには、出力エンジンが暗号化をサポートしている必要があります。それ以外の場合、データファイルはコピーされません。
- SAS Proprietary の暗号化されたデータファイルで PROC CPORT を使用することはできません。

ENCRYPT=AES または ENCRYPT=AES2 を使用する場合、次のルールが適用されます。

- テーブルを暗号化するときは、ENCRYPTKEY=テーブルオプションを使用する必要があります。
- テーブルを読み取るために復号化を有効にするには、ENCRYPTKEY=テーブルオプションを使用する必要があります。
- AES 暗号化で暗号化された SAS データセットは、AES 暗号化をサポートするエンジンによってのみ復号化できます。
- テーブルを再作成せずに、AES 暗号化テーブルの ENCRYPTKEY=値を変更することはできません。
- 参照整合性制約のある SAS データファイルでは、主キーデータファイルとすべての外部キーデータファイルが同じ暗号化キーを使用している限り、AES 暗号化を使用できます。

AES2 暗号化を使用するデフォルトの Base SAS データセットをコピーして新しい SPD Engine データセットまたは SPD Server テーブルを作成すると、暗号化は AES に変換されます。警告がログに書き込まれます。

SAS が提供する暗号化の詳細については、*SAS Viya プラットフォームの暗号化: 保存データ*を参照してください。

例: ENCRYPT=YES オプションの使用

この例では、SAS Proprietary アルゴリズムを使用して暗号化されたデータセットを作成します。

```
create table myfiles.salary {option encrypt=yes read=green}
  (name char(15),
   yrsal double,
   bonuspct double);
insert into myfiles.salary {option read=green} values ('Muriel', 34567, 3.2);
insert into myfiles.salary {option read=green} values ('Bjorn', 74644, 2.5);
insert into myfiles.salary {option read=green} values ('Freda', 38755, 4.1);
insert into myfiles.salary {option read=green} values ('Benny', 29855, 3.5);
insert into myfiles.salary {option read=green} values ('Agnetha', 70998, 4.1);
```

データセットからデータを取得するには、読み取りパスワードを指定する必要があります。

```
select * from myfiles.salary {option read=green};
```

関連項目

テーブルオプション:

- “ENCRYPTKEY= テーブルオプション” (1094 ページ)
- “PW= テーブルオプション” (1118 ページ)
- “READ=テーブルオプション テーブルオプション” (1119 ページ)

ENCRYPTKEY= テーブルオプション

AES および AES2 暗号化のキー値を指定します。

カテゴリ: テーブルコントロール

制限事項: AES および AES2 暗号化データファイルでのみ使用してください。
このテーブルオプションは、CAS サーバーではサポートされていません。

データソース: SAS データセット、SPD Engine データセット、SPD Server テーブル

注: この操作の後にログをチェックして、暗号化キー値が表示されていないことを確認してください。詳細については、“ログからパスワードと暗号化キーの値を消去する” (*SAS プログラマガイド: エッセンシャル*)を参照してください。

構文

ENCRYPTKEY= [' | "]*key-value* [' | "]

構文の説明

key-value
暗号化キー値を割り当てます。ENCRYPT=AES または ENCRYPT=AES2 を使用する場合は、ENCRYPTKEY=テーブルオプションを指定する必要があります。キー値の長さは最大 64 バイトです。

注 ENCRYPTKEY=値は、引用符付きまたは引用符なしで作成できます。詳細については、“ENCRYPTKEY=テーブルオプションでの引用符の使用” (1095 ページ)を参照してください。

ENCRYPTKEY=キー値が DBCS 文字を使用する場合、64 バイトの制限は、UTF-8 エンコーディングにトランスコードされた後の文字列に適用されます。次の DATA ステップを使用して、DBCS のキー値の長さをバイト単位で計算できます。

```
data _null_;
   key=length(unicodec("key-value","UTF8"));
   put "key length=" key;
run;
```

参照整合性制約を持つ SAS データファイルに暗号化キー値を割り当てる場合、主キーテーブルと、相互に参照するすべての外部キーテーブルは同じ暗号化キーを使用する必要があります。詳細については、*SAS プログラマガイド: エッセンシャル*の暗号化に関するトピックと、*SAS V9 LIBNAME エンジン: リファレンス*の一貫性制約に関するトピックを参照してください。

詳細

基本

注意
キー値を記録します。 ENCRYPTKEY=キー値を忘れると、データが失われます。キー値はデータセットに保存されていないため、SAS は ENCRYPTKEY=キー値の回復を支援できません。次の警告がログに書き込まれます。

```
WARNING: If you lose or forget the ENCRYPTKEY= value, there
will be not be any way to open the file or recover the data.
```

ENCRYPTKEY=テーブルオプションは、ファイルを削除または置換から保護しません。暗号化されたデータセットは、ENCRYPTKEY=キー値を指定せずに、次のシナリオのいずれかを使用して削除できます。

- PROC DATASETS の KILL オプション
- PROC SQL の DROP ステートメント
- PROC DATASETS の DELETE ステートメントまたは DELETE プロシジャ

ENCRYPTKEY=オプションは、ファイルのコンテンツへのアクセスのみを防止します。ファイルを削除または置換から保護するには、ファイルに ALTER=パスワードも含まれている必要があります。

FedSQL を使用して AES 暗号化されたデータファイルのデータを挿入、取得、変更するには、データファイルにアクセスするすべての FedSQL ステートメントで ENCRYPTKEY=オプションを指定する必要があります。暗号化されたデータファイルの削除を防ぐためにパスワードが指定されている場合は、そのパスワードも指定する必要があります。

次の DATASETS プロシジャーステートメントでは、保護されたファイルを操作するときに ENCRYPTKEY=キー値を指定する必要があります: AGE、AUDIT、APPEND、CHANGE、CONTENTS、MODIFY、REBUILD、REPAIR ステートメント。

ENCRYPTKEY=キー値としてマクロ変数を使用することができます。ENCRYPTKEY=キー値にマクロ変数を指定する場合は、マクロ変数を二重引用符で囲む必要があります。二重引用符を使用しないと、予期しない結果が発生する可能性があります。次の例では、マクロ変数を定義し、マクロ変数を ENCRYPTKEY=キー値として使用します。

```
%let secret=myvalue;
create table myschema.dsname {options encrypt=aes encryptkey="&secret"};
```

データファイルを再作成せずに、AES 暗号化データファイルのキー値を変更することはできません。

ENCRYPTKEY=テーブルオプションでの引用符の使用

暗号化キーの値が引用符なしで作成される場合、値は文字で始まる必要があり、英数字とアンダースコアのみを含めることができます。キー値に空白を含めることはできません。文字は大文字、小文字、または大文字と小文字の混合にすることができます。許可される例を次に示します。

```
encryptkey=G_rEeN1
```

引用符で囲まれていないキー値の評価では、大文字と小文字は区別されません。次のキー値を使用して、前述のキーで作成された SAS データファイルを読み取ることができます。

encryptkey=G_REEN1

単一引用符で囲んで指定する暗号化キー値には、英数字、特殊文字、DBCS 文字を含めることができます。空白は許可されますが、キー値をすべて空白で構成することはできません。文字列の評価では、大文字と小文字が区別されます。有効なキー値の例を次に示します。

encryptkey='1Gr*#E7e.N 1'

引用符で囲まれた暗号化キー値を使用して暗号化されたデータファイルにアクセスするために指定される暗号化キー値は、データファイルの作成に使用された暗号化キー値と正確に一致する必要があります。

二重引用符で囲んで指定される暗号化キー値には、単一引用符で囲んで指定される暗号化キー値と同じ要件が適用されます。違いは、マクロ解決に対して暗号化キー値が有効になっていることです。

例: ENCRYPTKEY=オプションの使用

ENCRYPTKEY=テーブルオプションを使用して SAS データセットのコンテンツを暗号化および復号化する FedSQL ステートメントの例を次に示します。

```
libname x 'location';
proc fedsql;
title 'Current Compensation';
  create table x.salary {options encrypt=aes encryptkey='1Gr*#E7e.N 1'}
    (name char(15),
     yrsal double having format dollar12.2,
     bonuspct double);
  insert into x.salary {option encryptkey='1Gr*#E7e.N 1'} values ('Muriel', 34567, 3.2);
  insert into x.salary {option encryptkey='1Gr*#E7e.N 1'} values ('Bjorn', 74644, 2.5);
  insert into x.salary {option encryptkey='1Gr*#E7e.N 1'} values ('Freda', 38755, 4.1);
  insert into x.salary {option encryptkey='1Gr*#E7e.N 1'} values ('Benny', 29855, 3.5);
  insert into x.salary {option encryptkey='1Gr*#E7e.N 1'} values ('Agnetha', 70998, 4.1);
  select * from x.salary {option encryptkey='1Gr*#E7e.N 1'};
quit;

proc fedsql;
title 'Adjusted Compensation';
  update x.salary {option encryptkey='1Gr*#E7e.N 1'}  set yrsal = yrsal + 5000;
  update x.salary {option encryptkey='1Gr*#E7e.N 1'}  set bonuspct = 5.0;
  delete from x.salary {option encryptkey='1Gr*#E7e.N 1'} where name = 'Benny';
  select * from x.salary{option encryptkey='1Gr*#E7e.N 1'} ;
quit;

proc fedsql;
  drop table x.salary{option encryptkey='1Gr*#E7e.N 1'} ;
quit;
```

ステートメントからの出力は次のとおりです。

Current Compensation

NAME	YRSAL	BONUSPCT
Muriel	$34,567.00	3.2
Bjorn	$74,644.00	2.5
Freda	$38,755.00	4.1
Benny	$29,855.00	3.5
Agnetha	$70,998.00	4.1

Adjusted Compensation

NAME	YRSAL	BONUSPCT
Muriel	$39,567.00	5
Bjorn	$79,644.00	5
Freda	$43,755.00	5
Agnetha	$75,998.00	5

関連項目

テーブルオプション:

- “ENCRYPT= テーブルオプション” (1091 ページ)

ENDOBS= テーブルオプション

処理するオブザベーションのユーザー定義範囲内の終了オブザベーション番号を指定します。

該当要素: SELECT ステートメント

カテゴリ: オブザベーションコントロール

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。
入力データセットでのみ使用します。

操作: STARTOBS=テーブルオプションと組み合わせて使用します。

データソース: SPD Engine データセット、SPD Server テーブル

構文

ENDOBS= *n*

引数

n
終了オブザベーションの番号です。

詳細

STARTOBS=または ENDOBS=オプションでオブザベーションの範囲を指定しない限り、ソフトウェアはデータセット全体のすべてのオブザベーションを 処理します。ENDOBS=オプションを指定せずに STARTOBS=オプションを使用すると、ENDOBS=の暗黙の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=の値は STARTOBS=の値より大きくなければなりません。

SAS ソフトウェアオプションの FIRSTOBS=および OBS=とは対照的に、STARTOBS=および ENDOBS= SPD Server オプションは、テーブル入力操作に加えて、WHERE 句の処理に使用できます。ENDOBS=が WHERE 式で使用されている場合、ENDOBS=値は、返されるオブザベーションの数ではなく、処理する最後のオブザベーションを表します。

例

次のコードは、ENDOBS=テーブルオプションが FedSQL SELECT ステートメントでどのように指定されているかを示しています。

```
select * from spdslib.table1(startobs=3 endobs=4);
```

関連項目

テーブルオプション:

- “STARTOBS= テーブルオプション” (1128 ページ)

FETCH_NUMERIC_TYPE= テーブルオプション

Google BigQuery からの NUMERIC データの処理方法を指定します。

カテゴリ: テーブルコントロール

デフォルト: FLOAT64

データソース: Google BigQuery

注: このオプションのサポートは 2024.09 から開始されます。

構文

FETCH_NUMERIC_TYPE=FLOAT64 | NUMERIC

必須引数

FLOAT64
Google BigQuery から取得された NUMERIC(p,s)および BIGNUMERIC (p,s)値が DOUBLE として扱われることを指定します。(FedSQL DOUBLE データ型は、Google BigQuery FLOAT64 データ型にマッピングされます。)

NUMERIC
Google BigQuery から取得された NUMERIC(p,s)および BIGNUMERIC (p,s)値が NUMERIC として扱われることを指定します。

詳細

デフォルト値を使用すると、読み取りパフォーマンスが向上します。このテーブルオプションを使用すると、NUMERIC 動作に戻すことができます。

データ定義は影響を受けません。

GP_DISTRIBUTED_BY= テーブルオプション

作成するテーブルの分散キーを指定します。

カテゴリ: テーブルコントロール

別名: DISTRIBUTED_BY=

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

データソース: Greenplum

構文

GP_DISTRIBUTED_BY= 'DISTRIBUTED BY (*column*[, ...*column*])
| DISTRIBUTED RANDOMLY'

引数

DISTRIBUTED BY (*column*, [***...column***])
分散キーとして使用する 1 つ以上の DBMS 列名を指定します。

DISTRIBUTED RANDOMLY
テーブルの行をデータベースセグメントに分散するために使用する列または列のセットを決定することを指定します。これは、ラウンドロビン分散として知られています。

詳細

DISTRIBUTED BY は、分散キーとして宣言された 1 つ以上の列でハッシュ分散を使用します。最も均一なデータ分散の場合、分散キーはテーブルの主キーまたは一意の列(または列のセット)にする必要があります。それが不可能な場合は、DISTRIBUTED RANDOMLY を選択して、データをラウンドロビンでセグメントインスタンスに送信します。値が指定されていない場合は、主キー(テーブルにある場合)またはテーブルの最初の適格な列を分散キーとして使用して、ハッシュ分散が選択されます。

DISTRIBUTED_BY は、前述のように、または DBCREATE_TABLE_OPTS=テーブルオプション内でサブミットできます。DBCREATE_TABLE_OPTS=での指定方法の例を次に示します。

```
dbcreate_table_opts='distributed by ("b")'
```

関連項目

テーブルオプション:

- “DBCREATE_TABLE_OPTS= テーブルオプション” (1086 ページ)

IDXNAME= テーブルオプション

WHERE 句の条件に一致する特定のインデックスを使用するように SAS に指示します。

カテゴリ: SAS インデックス使用のユーザーコントロール

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。
入力データセットでのみ使用します。
IDXWHERE=テーブルオプションでは使用できません

データソース: SAS データセット

構文

IDXNAME= *index-name*

引数

index-name
SAS データセットの単純インデックスまたは複合インデックスの名前(最大 32 文字)を指定します。SAS は、指定されたインデックスが最適かどうか、または順次検索の方がリソース効率が高いかどうかを判断しようとしません。

操作 指定は、データセットの永続的な属性ではなく、データセットの現在の使用に対してのみ有効です。

詳細

デフォルトでは、インデックス付き SAS データセットの WHERE 句の条件を満たすために、SAS は WHERE 句の最適化に使用できる 0 個以上の候補インデックスを識別します。候補インデックスのリストから、SAS は最高のパフォーマンスを提供すると判断したものを選択するか、データの順次パスがより効率的であると予想される場合はすべてのインデックスを 拒否します。

SAS が選択するインデックスが常に最良の最適化を提供するとは限らないため、IDXNAME=テーブルオプションを指定して、候補インデックスの 1 つを使用するよう SAS に指示できます。SAS が候補インデックスとして識別しないインデックスを指定すると、IDXNAME=テーブルオプションは要求を処理しません。つまり、IDXNAME=では、誤った結果を生成するインデックスを指定することはできません。

比較

IDXWHERE=テーブルオプションを使用すると、インデックスを使用するかどうかに関する SAS の決定をオーバーライドできます。

例

この例では、IDXNAME=テーブルオプションを使用して、特定のインデックスを使用して WHERE 句を最適化するよう SAS に指示します。その後、SAS は、データセットの順次検索がリソース効率を高める可能性を無視し、指定されたインデックスが最適かどうかを判断しようとしません。(EMPNUM インデックスは NOMISS オプションを使用して作成されていないことに注意してください。)

```
create table mydata.empnew
  as select * from mydata.employee {option idxname=empnum}
  where empnum < 2000;
```

関連項目

テーブルオプション

- “IDXWHERE= テーブルオプション” (1102 ページ)

IDXWHERE= テーブルオプション

SAS が WHERE 句の条件に一致するためにインデックス検索と順次検索のどちらを使用するかを指定します。

カテゴリ: SAS インデックス使用のユーザーコントロール

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。
入力データセットでのみ使用します。
SAS データセット: IDXNAME=では使用できません。
SPD Engine データセット: IDXWHERE=NO は WHERENOINDEX=では使用できません。

データソース: SAS データセット、SPD Engine データセット

構文

IDXWHERE= YES | NO

引数

YES
最適なインデックスを選択して WHERE 句を最適化し、データセットの順次検索の方がリソース効率が高い可能性を無視するよう SAS に指示します。これがデフォルト値です。

NO
すべてのインデックスを 無視し、データセットの順次検索で WHERE 句の条件を満たすように SAS に指示します。

注 IDXWHERE=テーブルオプションを使用して、BY ステートメントを処理するためのインデックスの使用をオーバーライドすることはできません。

IDXWHERE=NO が使用されている場合、WHERENOINDEX=テーブルオプションは使用できません。

詳細

デフォルトでは、インデックス付きデータセットの WHERE 句の条件を満たすために、インデックスを使用するか、データセットを順次読み取るかがソフトウェアによって決定されます。ソフトウェアは相対的な効率を推定し、より効率的な方法を選択します。

IDXWHERE=テーブルオプションを指定して、ソフトウェアの決定をオーバーライドする必要がある場合があります。決定は、最良の結果が得られない場合がある一般的な規則に基づいているためです。つまり、IDXWHERE=テーブルオプションを指定することで、処理方法を決定することができます。

注: 指定は、データセットの永続的な属性ではなく、データセットの現在の使用に対してのみ有効です。

比較

IDXNAME=テーブルオプション(SAS データセットでのみサポート)を使用すると、SAS に特定のインデックスを使用するように指示できます。

WHERENOINDEX=テーブルオプションを使用すると、WHERE 式の評価を行うときに除外するインデックスのリストを指定できます。

例

例 1: インデックス使用の指定

この例では、IDXWHERE=テーブルオプションを使用して、WHERE 句を最適化するのに最適なインデックスを決定するよう SAS に指示します。次に、SAS は、データセットの順次検索の方がリソース効率が高い可能性を無視します。

```
create table mydata.empnew
  as select * from mydata.employee {option idxwhere=yes}
  where empnum < 2000;
```

例 2: インデックス使用なしの指定

この例では、IDXWHERE=テーブルオプションを使用して、インデックスを無視し、データセットの順次検索で WHERE 句の条件を満たすように SAS に指示します。

```
create table mydata.empnew
  as select * from mydata.employee {option idxwhere=no}
  where empnum < 2000;
```

関連項目

テーブルオプション:

- “IDXNAME= テーブルオプション” (1100 ページ)
- “WHERENOINDEX= テーブルオプション” (1160 ページ)

IOBLOCKSIZE= テーブルオプション

I/O 操作で使用されるブロック内の行数を指定します。

カテゴリ: テーブルコントロール

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

データソース: SPD Engine データセット、SPD Server テーブル

構文

IOBLOCKSIZE= *n*

引数

n
ブロックのサイズをバイト単位で指定します。デフォルト値は、データソースでサポートされる最小のブロックサイズです。

詳細

ソフトウェアは、テーブル内の行をブロック単位で読み取り、格納します。IOBLOCKSIZE=は、圧縮または暗号化されたテーブルで役立ちます。ソフトウェアは、非圧縮テーブルまたは非暗号化テーブルでは IOBLOCKSIZE=を使用しません。

圧縮または暗号化するテーブルの場合、IOBLOCKSIZE=の指定によって、ブロックに含める行数が決まります。この指定は、ブロック圧縮と、ディスクとの間のデータ I/O に適用されます。IOBLOCKSIZE=値は、ディスク上のテーブルの編成に影響します。

圧縮または暗号化を使用する場合は、データへのアクセス方法(順次またはランダム)を補足する IOBLOCKSIZE=値を指定します。フルテーブルスキャンを必要とする順次アクセスまたは操作では、大きなブロックサイズが優先されます。対照的に、ランダムアクセスではブロックサイズが小さい方が優先されます。SPD Engine では、大きなブロックサイズは 131,072 バイトになります。許可される最小値は 32,768 バイトです。SPD Server の場合、大きなブロックサイズは 64,000 バイトです。許可される最小値は 8,000 バイトです。

関連項目

テーブルオプション:

- “COMPRESS= テーブルオプション” (1082 ページ)
- “ENCRYPT= テーブルオプション” (1091 ページ)
- “PADCOMPRESS= テーブルオプション” (1111 ページ)

KMS_PATH= テーブルオプション

プロジェクト内のキーファイルリソースの 場所を指定します。

カテゴリ: テーブルアクセス

別名: KMS_FILE=

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

データソース: Google BigQuery

例: kms_path="/u/BigQuery/projFiles/filename"

構文

KMS_PATH="*path-and-filename*"

引数

"*path-and-filename*"
プロジェクト内のキーファイルリソースへのパスを指定します。

関連項目

テーブルオプション:

- “LOCATION= テーブルオプション” (1107 ページ)

LABEL= テーブルオプション

テーブルのラベルを指定します。

カテゴリ: テーブルコントロール

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

データソース: SAS データセット、SPD Engine データセット

構文

LABEL= '*label* '

引数

'*label*'
最大 256 文字のテキスト文字列を指定します。

要件 SAS データセットおよび SPD Engine データセットの場合、テキストに単一引用符が含まれている場合は、ラベルを二重引用符で囲みます。または、ラベルテキストで 2 つの単一引用符を使用し、文字列を単一引用符で囲みます。データセットからラベルを削除するには、引用符で囲まれた空白に等しいラベルを割り当てます。

詳細

LABEL=テーブルオプションで指定されたラベルは、テーブルのメタデータの一部として保存されます。ただし、この情報は FedSQL では使用されません。つまり、一度保存されたラベルは表示できません。ラベルは SAS CONTENTS プロシジャで表示できます。

LABEL=テーブルオプションは、入力テーブルと出力テーブルの両方で使用できます。入力テーブルで LABEL=テーブルオプションを使用すると、操作中にファイルにラベルが割り当てられます。これを出力テーブルに指定すると、ラベルはそのファイルの永続的な部分になります。

テーブルに割り当てられたラベルは、APPEND プロシジャや MODIFY ステートメントを使用する場合など、テーブルをインプレースで 更新しても、そのテーブルに関連付けられたままになります。ただし、以前にラベルが割り当てられたテーブルを使用して、CREATE TABLE ステートメントで新しいテーブルを作成すると、ラベルが失われます。たとえば、次の CREATE TABLE ステートメントで新しい出力テーブル ONE を作成すると、以前にテーブル ONE に割り当てられていたラベルが失われます。

```
create table one as select * from one;
```

例

これらの例は、SAS データセットにラベルを割り当てます。

```
create table tst {option label='1976 W2 Info, Hourly'} (col1 char(4), col2 double);
create table tst1 {option label="Hillside's Daily Account"} as select * from tst;
create table tst2 {option label='Peter''s List'}(col1 char(20), col2 char(20));
```

LOCATION= テーブルオプション

データセットの場所を指定します。

カテゴリ: テーブルアクセス

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

データソース: Google BigQuery

注: このオプションのサポートは、SAS/ACCESS の 2019 年 11 月リリースで追加されました。

構文

LOCATION='*value*'

引数

'*value*'
データセットの場所を指定します。有効な値については、https://cloud.google.com/bigquery/docs/locations を参照してください。

詳細

LOCATION=が指定されていない場合、BigQuery が場所を決定します。

例

例 1: 米国のロケールを指定

この設定では、米国の場所を指定します。

```
LOCATION='US'
```

例 2: 欧州連合のロケールを指定

この設定では、欧州連合の場所を指定します。

```
LOCATION='EU'
```

関連項目

テーブルオプション:

- “KMS_PATH= テーブルオプション” (1105 ページ)

LOCKTABLE= テーブルオプション

テーブルに共有ロックまたは排他ロックをかけます。

カテゴリ: テーブルコントロール

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

データソース: SAS データセット

構文

LOCKTABLE= EXCLUSIVE | SHARE

引数

SHARE
テーブルを共有モードでロックし、他のユーザーまたはプロセスがテーブルからデータを読み取ることを許可しますが、ユーザーがデータを更新することはできません。

EXCLUSIVE
テーブルを排他的にロックし、開いているテーブルに他のユーザーがアクセスできないようにします。

詳細

テーブルをロックできるのは、自分が所有者であるか、必要な権限が付与されている場合のみです。

PROC FEDSQL を介して BASE テーブルドライバーにアクセスする 場合、LOCKTABLE オプションのデフォルト値は EXCLUSIVE です。ただし、SAS Federation Server を介して BASE テーブルドライバーにアクセスする 場合、または SAS Federation Server LIBNAME エンジンを使用してローカルでプログラムを実行する場合、LOCKTABLE オプションのデフォルト値は SHARE です。

例

```
select x from test {options locktable=share} where x=2;
```

ORHINTS= テーブルオプション

FedSQL から Oracle に渡す Oracle ヒントを指定します。

該当要素: DELETE ステートメント、INSERT ステートメント、SELECT ステートメント、UPDATE ステートメント

カテゴリ: データコントロール

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

データソース: Oracle

構文

ORHINTS='/* *Oracle-hint* */'

引数

Oracle-hint
FedSQL 言語プロセッサがクエリの一部として DBMS に渡す Oracle ヒントを指定します。ヒントを指定しない場合、FedSQL は Oracle にヒントを送信しません。詳細については、*SAS Federation Server: 管理者ガイド*を参照してください。

詳細

ORHINTS=テーブルオプションは、DRIVER_TRACE=および DRIVER_TRACEFILE=データソース接続オプションと組み合わせて使用されます。PROC FEDSQL および PROC DS2 は、データソース接続オプションの使用をサポートしていません。

例: ORHINTS=オプションの使用

次の例は、ORHINTS=を指定する方法を示しています。

```
select * from test{options orhints='/* dummy hint */'} ;

update test{options orhints='/* dummy hint */'}  set c1=1;

insert into test{options orhints='/* dummy hint */'}  values (100);

delete from test{options orhints='/* dummy hint */'} ;
```

ORNUMERIC= テーブルオプション

Oracle NUMBER 列から読み取られた数値または Oracle NUMBER 列に挿入された数値の処理方法を指定します。

カテゴリ: テーブルコントロール

デフォルト: 2025.08 以降:BULKLOAD=YES が有効な場合は NO です。それ以外の場合は、YES です。
以前のリリースの場合: YES

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

データソース: Oracle

データの種類: DECIMAL、NUMERIC

構文

ORNUMERIC=YES | NO

引数

NO
数値が TKTS_DOUBLE 値として扱われることを示します。精度が 14 桁を超えない場合があります。

YES
明示的な精度を持つ非整数値が TKTS_NUMERIC 値として扱われることを示します。

詳細

このオプションは、接続オプションとテーブルオプションの両方として指定できます。接続オプションとテーブルオプションの両方を指定すると、テーブルオプションの値が接続オプションをオーバーライドします。2025.07 以前のリリースでは、このオプションのデフォルトは YES であるため、精度またはスケールのある NUMBER 列は TKTS_NUMERIC として記述されます。2025.08 以降では、BULKLOAD=YES が指定されている場合、このオプションのデフォルトは NO になります。それ以外の場合は、デフォルトで YES になります。

PADCOMPRESS= テーブルオプション

OUTPUT または UPDATE 用に開かれた SPD Engine 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。

カテゴリ: テーブルコントロール

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

データソース: SPD Engine データセット

構文

PADCOMPRESS= *n*

引数

n
追加するバイト数を指定します。デフォルトの数は 0 (ゼロ)です。

詳細

圧縮された SPD Engine データセットは、ディスク上の領域のブロックを占有します。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=テーブルオプションから取得されます。データセットが更新されると、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場合があります。さらに更新す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 操作の数が増加します。

データセットへの更新が多数予想される特定の状況でブロックパディングを 増やすことにより、断片化を最小限に抑えることができます。ただし、データセットを更新しない場合、パディングを追加すると領域が浪費される可能性があります。

すべての圧縮ブロックをパディングするコストと、一部の圧縮ブロックが断片化する可能性があるコストを比較検討する必要があります。

データセットを作成または更新するときに PADCOMPRESS=テーブルオプションを指定すると、ブロックがディスクに書き戻されるときに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に保持されません。

関連項目

テーブルオプション:

- “COMPRESS= テーブルオプション” (1082 ページ)
- “IOBLOCKSIZE= テーブルオプション” (1104 ページ)

PARTSIZE= テーブルオプション

テーブルパーティションのサイズを 指定します。

該当要素: CREATE TABLE ステートメント

カテゴリ: テーブルコントロール

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

データソース: SPD Engine データセット、SPD Server テーブル

構文

PARTSIZE= *n*

引数

n
パーティションのサイズを指定します。

要件 サイズは、MB、GB、および TB で指定できます。*n* が M、G、または T なしで指定された場合、デフォルトは MB です。たとえば、PARTSIZE=128 は PARTSIZE=128M と同じです。SPD Engine データセットの場合、デフォルト値は 128MB です。最大値は 8,796,093,022,207MB です。SPD Server テーブルの場合、デフォルト値は MINPARTSIZE=サーバーオプションの設定です。MINPARTSIZE=サーバーオプションが設定されていない場合、デフォルトは 16MB です。

SPD Engine データセットを作成するときに、行の長さが 65K を超えると、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ことがあります。これらの数値を一致させるには、32 の倍数と行の長さである PARTSIZE=を指定します。

詳細

詳細については、*SAS Scalable Performance Data Engine: リファレンス*の "PARTSIZE=データセットオプション"および *SAS Scalable Performance Data Server: ユーザーガイド*の"PARTSIZE=テーブルオプション"を適宜参照してください。

PASSWORD= テーブルオプション

HDFS ユーザーのパスワードを指定します。

カテゴリ: バルクロード

別名: PASS=、PWD=

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件 “BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。

操作: “USER= テーブルオプション” (1159 ページ)と組み合わせて使用します。

データソース: Impala

構文

PASSWORD="*value*"

引数

"*value*"
HDFS ユーザーのパスワードを指定します。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)
- “USER= テーブルオプション” (1159 ページ)

PICKLIST= テーブルオプション

バルクロード操作に使用するピックリストを指定します。

該当要素: 2026.02 以前のリリース

カテゴリ: バルクロード

制限事項: このオプションは、2026.03 以降のリリースでは使用できません。
このテーブルオプションは、CAS サーバーではサポートされていません。

要件 “BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。

データソース: Impala

注: このオプションは、2026.01 で非推奨となり、2026.03 で削除されました。

構文

PICKLIST="*filename*"

引数

"*filename*"
Hadoop ピックリストを含むテキストファイルへの完全なパス名を指定します。

デフォルト このテーブルオプションを省略した場合、ファイル"hadoopbasics/hadoopwrapr.txt"が使用されます。

詳細

"hadoopbasics/hadoopwrapr.txt"のデフォルトピックリストをオーバーライドする場合は、PICKLIST=オプションを使用します。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)

POINTOBS= テーブルオプション

SAS が圧縮データセットを作成し、そのオブザベーションがランダムアクセス可能か、順次アクセス可能かを指定します。

カテゴリ: テーブルコントロール

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。
出力テーブルのみで使用します。

操作: COMPRESS=テーブルオプションで使用

データソース: SAS データセット

構文

POINTOBS= YES | NO

構文の説明

YES
FedSQL 言語は、オブザベーション番号によってランダムにアクセスされる 可能性のある圧縮データセットを生成します。これがデフォルトです。

オブザベーション番号でデータに直接アクセスする例を次に示します。

- DATA ステップの MODIFY および SET ステートメントの POINT=オプション
- PROC FSEDIT を使用して特定のオブザベーション番号に直接移動

> ヒント　POINTOBS=YES を指定しても、データセットから情報を取得する効率には影響しません。圧縮データセットを作成するとき、および情報を更新または追加するときに、CPU 使用率が約 10%増加します。

NO
オブザベーション番号によって圧縮されたデータセット内のオブザベーションにランダムにアクセスする機能を抑制します。

> ヒント　POINTOBS=NO を指定することは、圧縮データセット内の番号によってオブザベーションを 直接指す機能が重要でないアプリケーションに適しています。オブザベーション番号でデータにアクセスする必要がない場合は、POINTOBS=NO を指定することで、パフォーマンスを約 10%向上させることができます。
>
> - 圧縮データセットの作成時
> - オブザベーションの更新または追加時

詳細

REUSE=YES は POINTOBS=YES よりも優先されることに注意してください。次に例を示します。

```
create table test{options compress=yes pointobs=yes reuse=yes};
```

このコード例では、POINTOBS=NO のデータセットが生成されます。圧縮を使用する場合は POINTOBS=YES がデフォルトであるため、REUSE=YES によって POINTOBS=が NO に変更されます。

関連項目

テーブルオプション:

- “COMPRESS= テーブルオプション” (1082 ページ)
- “REUSE= テーブルオプション” (1120 ページ)

POST_TABLE_OPTS= テーブルオプション

CREATE TABLE ステートメントのテーブル名の後にデータベース固有のオプションを配置できるようにします。

カテゴリ: テーブルコントロール

デフォルト: なし

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

データソース: Hive、Spark

参照項目: “DBCREATE_TABLE_OPTS= テーブルオプション” (1086 ページ)
“PRE_TABLE_OPTS= テーブルオプション” (1117 ページ)

構文

POST_TABLE_OPTS="*DBMS-SQL-options*"

必須引数

"*DBMS-SQL-options*"
CREATE TABLE ステートメントのテーブル名の後に配置する追加のデータベース固有のオプションを指定します。DBMS 値を単一引用符または二重引用符で囲むことができます。

詳細

POST_TABLE_OPTS=テーブルオプションを単独で使用することも、関連するテーブルオプション PRE_TABLE_OPTS=および POST_STMT_OPTS=と一緒に使用することもできます。POST_STMT_OPTS=は、DBCREATE_TABLE_OPTS=のエイリアスです。たとえば、次のテンプレートに従ってデータベースオプションを指定できます。

```
create table dblib.test{OPTIONS post_table_opts="external"} col1(int);

create table dblib.test{options pre_table_opts="external"
                        post_table_opts="stored as orc"} col1(int);


create table dblib.test{options pre_table_opts="external"
                        post_table_opts="stored as"
                        post_stmt_options="orc"} col1(int);
```

3 つのテーブルオプションをすべて 一緒に指定すると、次のように処理されます。

```
CREATE [PRE_TABLE_OPTIONS] TABLE test (col1 int) [POST_TABLE_OPTS]
[POST_STMT_OPTS/DBCREATE_TABLE_OPTS]
```

PRE_TABLE_OPTS= テーブルオプション

CREATE TABLE ステートメントのテーブル名の前にデータベース固有のオプションを配置できるようにします。

カテゴリ: テーブルコントロール

デフォルト: なし

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

データソース: Hive、Spark

参照項目: “DBCREATE_TABLE_OPTS= テーブルオプション” (1086 ページ)
“POST_TABLE_OPTS= テーブルオプション” (1116 ページ)

構文

PRE_TABLE_OPTS="*DBMS-SQL-options*"

必須引数

"*DBMS-SQL-options*"
CREATE TABLE ステートメントのテーブル名の前に配置する追加のデータベース固有のオプションを指定します。DBMS 値を単一引用符または二重引用符で囲むことができます。

詳細

PRE_TABLE_OPTS=テーブルオプションを単独で使用することも、関連するテーブルオプション POST_TABLE_OPTS=および POST_STMT_OPTS=と一緒に使用することもできます。POST_STMT_OPTS=は、DBCREATE_TABLE_OPTS=のエイリアスです。たとえば、次のテンプレートに従ってデータベースオプションを指定できます。

```
create table dblib.test{OPTIONS pre_table_opts="external"} col1(int);

create table dblib.test{OPTIONS pre_table_opts="external"
                        post_table_opts="stored as orc"} col1(int);


create table dblib.test{OPTIONS pre_table_opts="external"
                        post_table_opts="stored as"
                        post_stmt_opts="orc"} col1(int);
```

3 つのテーブルオプションをすべて 一緒に指定すると、次のように処理されます。

```
CREATE [PRE_TABLE_OPTIONS] TABLE test (col1 int) [POST_TABLE_OPTS]
[POST_STMT_OPTS/DBCREATE_TABLE_OPTS]
```

PW= テーブルオプション

READ、WRITE、および ALTER パスワードを SAS データセットまたは SPD Engine データセットに割り当て、パスワードで保護されたファイルへのアクセスを有効にします。暗号化された SPD Server テーブルにアクセスするためのキー値を指定します。

カテゴリ: テーブルコントロール

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

データソース: SAS データセット、SPD Engine データセット、SPD Server テーブル

注: この操作の後にログをチェックして、パスワード値が表示されていないことを確認してください。詳細については、*SAS プログラマガイド: エッセンシャル*の"パスワードと暗号化キーの値を消去する"を参照してください。

構文

PW= *password* | *key-value*

引数

password
有効な SAS 名である必要があります。

key-value
有効な SAS 名である必要があります。

詳細

PW=テーブルオプションは、SAS データセット、SPD Engine データセット、および SPD Server テーブルに適用されます。SAS データセットおよび SPD Engine データセットの場合、このオプションを使用して、データセットにパスワードを割り当てたり、パスワードで保護されたデータセットにアクセスしたりできます。パスワードで保護されているデータセットが置き換えられると、新しいデータセットはパスワードを継承します。

注: SAS パスワードは、SAS システム以外の SAS ファイルへのアクセスを制御しません。SAS 外の SAS ファイルへのアクセスを制御するには、オペレーティングシステムが提供するユーティリティとファイルシステムのセキュリティ 制御を使用する必要があります。

SPD Server テーブルの場合、PW=オプションを使用して、すでに暗号化されている SPD Server テーブルにアクセスできます。PW=オプションを使用すると、SAS 独自の暗号化アルゴリズムによって保護されているテーブルファイルにアクセスできます。

例

例 1: PW=によるパスワードまたは暗号化キーの割り当て

```
create table myfiles.mytable {options pw=green}(column1 double);
```

例 2: PW=によるパスワードまたは暗号化キーの指定

```
select * from myfiles.mytable {option pw=green};
```

READ=テーブルオプション テーブルオプション

READ パスワードを SAS ファイルに割り当てて、ユーザーがパスワードを入力しない限り、ファイルを読み取れないようにします。

カテゴリ: テーブルコントロール

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

データソース: SAS データセット、SPD Engine データセット

注: この操作の後にログをチェックして、パスワード値が表示されていないことを確認してください。詳細については、*"ログからパスワードと暗号化キーの値を消去する" (SAS プログラマガイド: エッセンシャル)*を参照してください。

構文

READ= *read-password*

引数

read-password
有効な SAS 名である必要があります。

詳細

READ=オプションは、SAS データセットまたは SPD Engine データセットにのみ適用されます。このオプションを使用して、パスワードを割り当てたり、読み取り保護されたファイルにアクセスしたりできます。

注: SAS パスワードは、SAS 以外の SAS ファイルへのアクセスを制御しません。SAS 外の SAS ファイルへのアクセスを制御するには、オペレーティングシステムが提供するユーティリティとファイルシステムのセキュリティ 制御を使用する必要があります。

READMODE= テーブルオプション

Google BigQuery から SAS にデータを移動するときに使用する方法を指定します。

カテゴリ: テーブルアクセス

デフォルト: STANDARD

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

データソース: Google BigQuery

構文

READMODE=STANDARD | STORAGE

オプション引数

STANDARD
SAS/ACCESS 方法を使用して、データを SAS に移動します。

STORAGE
Storage API for Google を使用して、データを SAS に移動します。

詳細

READMODE=STORAGE は、READMODE=STANDARD より高速です。

REUSE= テーブルオプション

圧縮された SAS データセットの空き領域に新しい行を書き込むことができるかどうかを指定します。

カテゴリ: テーブルコントロール

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。
出力データセットでのみ使用します。

データソース: SAS データセット

構文

REUSE= NO | YES

引数

NO
圧縮された SAS データセット内の領域を追跡および再利用しません。新しい行が既存のデータセットに追加されます。NO を指定すると、データセット内の多くの行を削除または更新する場合、データストレージの効率が低下します。

YES
圧縮された SAS データセット内の領域を追跡して再利用します。新しい行は、他の行が更新または削除されたときに解放された領域に挿入されます。

圧縮データセットの最後に行を追加する操作を使用する場合は、REUSE=NO を使用します。REUSE=YES を指定すると、必ずしもファイルの最後ではなく、ファイル内に領域があればどこでも新しい行が追加されます。

詳細

デフォルトでは、新しい行は既存の圧縮データセットに追加されます。他の行を削除または更新して空き領域を追跡および再利用する場合は、圧縮された SAS データセットを作成するときに REUSE=テーブルオプションを使用します。

REUSE=テーブルオプションは、COMPRESS=YES オプションを指定して新しいデータセットを作成する場合にのみ意味があります。既存の SAS データセットにアクセスしているときに REUSE=テーブルオプションを使用しても効果はありません。

関連項目

テーブルオプション:

- “COMPRESS= テーブルオプション” (1082 ページ)
- “POINTOBS= テーブルオプション” (1114 ページ)

RS_AWS_CONFIG= テーブルオプション

Amazon Web Services (AWS)構成ファイルの場所を指定します。

カテゴリ:	バルクロード
デフォルト:	~/.aws/config
制限事項:	このテーブルオプションは、CAS サーバーではサポートされていません。
要件:	このオプションを指定するには、最初に BULKLOAD=YES を指定する必要があります。 このオプションは“BULKOPTS= テーブルオプション” (1079 ページ)内で指定されます。
データソース:	Amazon Redshift

注: このオプションは、2024.07 からサポートされています。

参照項目: BL_AWS_CONFIG_FILE=LIBNAME ステートメントオプション

構文

RS_AWS_CONFIG='*path-and-file-name*'

必須引数

path-and-file-name
AWS 構成ファイルの完全ディレクトリパスと名前です。

例: Amazon Redshift の例

```
create table x.mytable {OPTIONS BULKLOAD=YES BULKOPTS=(RS_AWS_CONFIG='/my/.aws/config'
RS_CREDENTIALS_FILE='/my/.aws/creds' RS_BUCKET=mybucket RS_REGION='us-east-1'
RS_DEFAULT_DIR='/temp/' RS_KEY='99999' RS_SECRET="12345")}
 as select * from rstfile.othertable_1000000;
```

RS_BUCKET= テーブルオプション

データをバルクロードするときに使用する AWS バケットの名前を指定します。

カテゴリ: バルクロード

デフォルト: なし

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件: このオプションを指定するには、最初に BULKLOAD=YES を指定する必要があります。
このオプションは“BULKOPTS= テーブルオプション” (1079 ページ)内で指定されます。

データソース: Amazon Redshift

注: このオプションは、2024.07 からサポートされています。

構文

RS_BUCKET='*bucket-name*'

必須引数

bucket_name
Amazon S3 バケットを指定します。

RS_CREDENTIALS_FILE= テーブルオプション

データをバルクロードするときに使用する AWS 資格情報ファイルを指定します。

カテゴリ: バルクロード

デフォルト: ~/.aws/credentials

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件: このオプションを指定するには、最初に BULKLOAD=YES を指定する必要があります。
このオプションは“BULKOPTS= テーブルオプション” (1079 ページ)内で指定されます。

データソース: Amazon Redshift

注: このオプションは、2024.07 からサポートされています。

構文

RS_CREDENTIALS_FILE='*path-and-file-name*'

必須引数

path-and-file-name
AWS 資格情報ファイルの完全ディレクトリパスと名前を指定します。

RS_DEFAULT_DIR= テーブルオプション

中間バルクロードファイルが作成される場所を指定します。

カテゴリ: バルクロード

デフォルト: なし

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件: このオプションを指定するには、最初に BULKLOAD=YES を指定する必要があります。
このオプションは“BULKOPTS= テーブルオプション” (1079 ページ)内で指定されます。

データソース: Amazon Redshift

注: このオプションは、2024.07 からサポートされています。

構文

RS_DEFAULT_DIR='*directory-path*'

必須引数

directory-path
中間バルクロードファイルが作成されるホスト固有のディレクトリパスを指定します。

注 ディレクトリパスはスラッシュで終わる必要があります。例: BL_DEFAULT_DIR='/temp/'

RS_DELETE_DATAFILE= テーブルオプション

データファイルを削除するか、バルクロード機能によって作成されたすべてのファイルを削除するかを指定します。

カテゴリ: バルクロード

デフォルト: TRUE

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件 このオプションは“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。

データソース: Amazon Redshift

注: このオプションは、2024.07 からサポートされています。

構文

RS_DELETE_DATAFILE=TRUE | FALSE

必須引数

TRUE
バルクロード機能によって作成されたすべてのファイルを削除するように指定します。

FALSE
データファイルのみを削除するように指定します。

RS_KEY= テーブルオプション

キーベースのアクセスコントロールで使用される Amazon Web Services アクセスキーを指定します。一時トークン資格情報を使用している場合、これが一時アクセスキー ID になります。

カテゴリ: バルクロード

デフォルト: なし

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件: このオプションを指定するには、最初に BULKLOAD=YES を指定する必要があります。

このオプションは“BULKOPTS= テーブルオプション” (1079 ページ)内で指定されます。

データソース: Amazon Redshift

注: このオプションは、2024.07 からサポートされています。

構文

RS_KEY='*key-value*'

必須引数

key-value
AWS 環境へのアクセスに使用する AWS キーを指定します。

詳細

アクセスキーの管理の詳細については、Amazon Web Services (AWS)のドキュメントを参照してください。

RS_REGION= テーブルオプション

Amazon S3 データの AWS リージョンを指定します。

カテゴリ: バルクロード

デフォルト: なし

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件: このオプションを指定するには、最初に BULKLOAD=YES を指定する必要があります。

このオプションは“BULKOPTS= テーブルオプション” (1079 ページ)内で指定されます。

データソース: Amazon Redshift

注: このオプションは、2024.07 からサポートされています。

構文

RS_REGION='*region*'

必須引数

region
S3 データのロード元の AWS リージョンを指定します。S3 ツールを使用してデータをロードするときは、リージョンを指定する必要があります。

RS_SECRET= テーブルオプション

データのバルクロードに使用されるシークレットアクセスキーを指定します。

カテゴリ: バルクロード

デフォルト: なし

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件: このオプションを指定するには、最初に BULKLOAD=YES を指定する必要があります。
このオプションは“BULKOPTS= テーブルオプション” (1079 ページ)内で指定されます。

データソース: Amazon Redshift

注: このオプションは、2024.07 からサポートされています。

構文

RS_SECRET='*secret-access-key*'

必須引数

secret-access-key
RS_KEY=オプションで指定された AWS キー ID に関連付けられているシークレットアクセスキーを指定します。

ヒント セキュリティを強化するには、PROC PWENCODE を使用してキー値をエンコードします。

詳細

シークレットアクセスキーが PROC PWENCODE でエンコードされると、その値はデータソースに渡す前に SAS/ACCESS によってデコードされます。

RS_TOKEN= テーブルオプション

RS_KEY=および RS_SECRET=テーブルオプションで指定する一時資格情報に関連付けられる一時トークンを指定します。

カテゴリ: バルクロード

デフォルト: なし

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件: このオプションを指定するには、最初に BULKLOAD=YES を指定する必要があります。
このオプションは“BULKOPTS= テーブルオプション” (1079 ページ)内で指定されます。

データソース: Amazon Redshift

注: このオプションは、2024.07 からサポートされています。

構文

RS_TOKEN='*token*'

必須引数

token
RS_KEY=および RS_SECRET=テーブルオプションで指定された一時資格情報に関連付けられる一時トークンを指定します。

SCANSTRINGCOLUMNS= テーブルオプション

データベーステーブルまたはクエリの文字列の最大長を決定するかどうかを指定します。

カテゴリ: テーブルアクセス

別名: SCAN_STRINGS=
SCAN_STRING_COLUMNS=
SCANSTRINGCOLUMNS=

デフォルト: NO

データソース: Google BigQuery

構文

SCANSTRINGCOLUMNS=NO | YES

必須引数

NO
データをロードする前に文字列がスキャンされないように指定します。

YES
データをロードする前に、文字列をスキャンして実際の最大長を判別することを指定します。

詳細

Google BigQuery の文字データは、固定長ではない STRING 列に格納されます。MAX_CHAR_LEN= LIBNAME オプションは、BigQuery テーブルの列の最大長を指定するために使用されます。MAX_CHAR_LEN=に値が指定されていない場合、デフォルト値の 16,384 文字が使用されます。デフォルト設定は、切り捨てを避けるために大きくなっています。

SCANSTRINGCOLUMNS=テーブルオプションを使用すると、テーブルの MAX_CHAR_LEN= LIBNAME オプションの設定をオーバーライドできます。SCANSTRINGCOLUMNS=は、選択した列の実際の最大長を取得します。このオプションを使用すると、結果のテーブルのサイズを縮小し、ロードプロセスを高速化できます。このオプションは任意のテーブルに指定できますが、多くの文字列を持つ大きなテーブルで最適なパフォーマンスの向上が得られます。

関連項目

"SCANSTRINGCOLUMNS= LIBNAME Statement Option" (*SAS/ACCESS for Relational Databases: Reference*)

STARTOBS= テーブルオプション

処理するユーザー定義の行範囲の開始行番号を指定します。

該当要素: SELECT ステートメント

カテゴリ: オブザベーションコントロール

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。
入力データセットでのみ使用します。

データソース: SPD Engine データセット、SPD Server テーブル

構文

STARTOBS= *n*

引数

n
開始行の番号です。

詳細

STARTOBS=および ENDOBS=オプションで行の範囲を指定しない限り、ソフトウェアはデータセット全体を処理します。ENDOBS=オプションを STARTOBS=オプションなしで使用すると、STARTOBS=の暗黙の値は 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=の値は、ENDOBS=の値より小さくなければなりません。

SAS ソフトウェアオプションの FIRSTOBS=および OBS=とは対照的に、STARTOBS=および ENDOBS=オプションは、テーブル入力操作に加えて、WHERE 句の処理に使用できます。STARTOBS=が WHERE 式で使用されている場合、STARTOBS=値は WHERE 式を適用する最初の行を表します。

例

次のコードは、STARTOBS=テーブルオプションが FedSQL SELECT ステートメントでどのように指定されているかを示しています。

```
select * from spdslib.table1(startobs=3 endobs=4);
```

関連項目

テーブルオプション:

- “ENDOBS= テーブルオプション” (1097 ページ)

TABLE_TYPE= テーブルオプション

SAP HANA でテーブルを作成するときに FedSQL が使用するテーブルストレージの種類を指定します。

該当要素: CREATE TABLE ステートメント

カテゴリ: データアクセス

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

操作: GLOBAL と GLOBAL TEMPORARY は同じ動作をする
LOCAL と LOCAL TEMPORARY は同じ動作をする

データソース: SAP HANA

構文

TABLE_TYPE= [COLUMN | GLOBAL | GLOBAL TEMPORARY | LOCAL | LOCAL TEMPORARY | ROW]

引数

COLUMN
SAP HANA で列ベースのストレージを使用してテーブルを作成します。

GLOBAL | GLOBAL TEMPORARY
SAP HANA にグローバルな一時テーブルを作成します。テーブルはグローバルに使用できます。ただし、データは現在のセッションでのみ表示されます。

LOCAL | LOCAL TEMPORARY
SAP HANA にローカルの一時テーブルを作成します。テーブルの定義とデータは、現在のセッションでのみ表示されます。

ROW
SAP HANA で行ベースのストレージを使用してテーブルを作成します。

詳細

SAP HANA TABLE_TYPE=オプションは、接続オプションおよびテーブルオプションとして使用できます。どちらも指定されていない場合、SAP HANA で作成されたテーブルは、行ストアまたは列ストアの SAP HANA デフォルトに従います。

関連項目

テーブルオプション:

- “DBCREATE_TABLE_OPTS= テーブルオプション” (1086 ページ)

TD_BUFFER_MODE= テーブルオプション

LOAD メソッドを使用するかどうかを指定します。

カテゴリ: バルクロード

デフォルト: NO

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件 “BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。

データソース: Teradata

構文

TD_BUFFER_MODE= YES | NO

引数

YES
バルクロード機能を有効にします。バルクロード機能を機能させるには、このオプションを YES に設定する必要があります。

NO
バルクロード機能を無効にします。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)

TD_CHECKPOINT= テーブルオプション

TPT 操作がデータベースにチェックポイントまたはセーブポイントを 発行するタイミングを指定します。

カテゴリ: バルクロード

デフォルト: 0 (ゼロ)

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件 “BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。

データソース: Teradata

構文

TD_CHECKPOINT= *number*

引数

number
TPT 操作がデータベースにチェックポイントまたはセーブポイントを 発行するまでの行数を指定します。デフォルト値 0 は、チェックポイントが取得されないことを意味します。ジョブの終了時にすべての行が保存されます。

TD_DATA_ENCRYPTION= テーブルオプション

データ暗号化をアクティブにします。

カテゴリ:	バルクロード
別名:	TPT_DATA_ENCRYPTION=
デフォルト:	NO
制限事項:	このテーブルオプションは、CAS サーバーではサポートされていません。
要件	“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。
データソース:	Teradata

構文

TD_DATA_ENCRYPTION= YES | NO

引数

YES
すべての SQL 要求、応答、およびデータを暗号化します。

別名 ON

NO
暗号化は行われません。

別名 OFF

詳細

TD_DATA_ENCRYPTION=セッション属性は、Teradata TPT (Teradata Parallel Transporter)が LOAD、UPDATE、および STREAM 操作のために作成するすべての接続の暗号化を制御します。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)

TD_DROP_ERROR_TABLE= テーブルオプション

ジョブが正常に完了したかどうかに関係なく、ジョブの終了時にログテーブルを削除します。

カテゴリ:	バルクロード
デフォルト:	NO
制限事項:	このテーブルオプションは、CAS サーバーではサポートされていません。
要件	“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。
データソース:	Teradata

構文

TD_DROP_ERROR_TABLE= YES | NO

引数

YES
ジョブが正常に完了したかどうかに関係なく、ジョブの終了時にエラーテーブルを削除するように指定します。

NO
ジョブが完了した後、エラーを含むエラーテーブルを保持します。正常終了後に空のエラーテーブルは削除されます。

関連項目

テーブルオプション:

- BULKLOAD=テーブルオプション (1074 ページ)

TD_DROP_LOG_TABLE= テーブルオプション

ジョブが正常に完了したかどうかに関係なく、ジョブの終了時にログテーブルを削除します。

カテゴリ: バルクロード

デフォルト: NO

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件 “BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。

データソース: Teradata

構文

TD_DROP_LOG_TABLE= YES | NO

引数

YES
ジョブが正常に完了したかどうかに関係なく、ジョブの終了時にログテーブルを削除するように指定します。

NO
正常に完了しなかったジョブのログテーブルを保持します。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)

TD_DROP_WORK_TABLE= テーブルオプション

ジョブが正常に完了したかどうかに関係なく、ジョブの終了時に作業テーブルを削除します。

カテゴリ: バルクロード

デフォルト: NO

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件 “BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。

データソース: Teradata

構文

TD_DROP_WORK_TABLE= YES | NO

引数

YES
: ジョブが正常に完了したかどうかに関係なく、ジョブの終了時に作業テーブルを削除するように指定します。

NO
: 正常に完了しなかったジョブの作業テーブルを保持します。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)

TD_ERROR_LIMIT= テーブルオプション

エラーテーブルに保存できるレコードの最大数を指定します。

カテゴリ:	バルクロード
デフォルト:	Unlimited
制限事項:	このテーブルオプションは、CAS サーバーではサポートされていません。
要件	“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。
データソース:	Teradata

構文

TD_ERROR_LIMIT= *number-of-records*

引数

number-of-records
: Load、Stream、または Update オペレータージョブが終了する前に、エラーテーブルに格納できるレコードの最大数を指定します。レコード数は 0 より大きくなければなりません。

詳細

ErrorLimit の指定は、オペレータージョブの各インスタンスに適用されます。無効な値を指定すると、ジョブが終了します。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)

TD_ERROR_TABLE_1= テーブルオプション

最初のエラーテーブルの名前を指定します。

カテゴリ: バルクロード

デフォルト: *ttname*_ET。ここで、*ttname* はターゲットテーブルの名前です。

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件 “BULKLOAD= テーブルオプション” (1074 ページ)内で指定する必要があります。

データソース: Teradata

構文

TD_ERROR_TABLE_1= *name-of-table*

引数

name-of-table
エラーテーブルの名前を指定します。27 文字を超えるテーブル名は、3 文字の接尾辞に対応するために切り捨てられます。したがって、切り捨てられないテーブルの名前を指定することをお勧めします。切り捨てが発生すると、メッセージがログに書き込まれます。

制限事項 一時停止したジョブを再開する場合を除き、既存のテーブルの名前は使用できません。

詳細

ErrorTable1 には、Load、Stream、または Update オペレータージョブの取得フェーズ中に次の理由で拒否されたレコードが含まれています。

- データ変換エラー
- 制約違反
- アクセスモジュールプロセッサ(AMP)構成の変更

ユーティリティテーブルの 別の場所を指定するために TD_LOGDB=テーブルオプションを使用しない限り、エラーテーブルはデフォルトのユーザー(ログオン)データベースに作成され、オプションでスキーマで修飾されます。

エラーを含むエラーテーブルは、ジョブの終了時に保持されます。保持しない場合は、TD_DROP_ERROR_TABLE=テーブルオプションを指定します。

エラーテーブルの形式とエラー修正の手順については、*Teradata Parallel Transporter リファレンス*を参照してください。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)
- “TD_DROP_ERROR_TABLE= テーブルオプション” (1133 ページ)
- “TD_LOGDB= テーブルオプション” (1141 ページ)

TD_ERROR_TABLE_2= テーブルオプション

2 番目のエラーテーブルの名前を指定します。

カテゴリ:	バルクロード
デフォルト:	*ttname*_UV。ここで、*ttname* はターゲットテーブルの名前です。
制限事項:	このテーブルオプションは、CAS サーバーではサポートされていません。
要件	“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。
データソース:	Teradata

構文

TD_ERROR_TABLE_2= *name-of-table*

引数

name-of-table
エラーテーブルの名前を指定します。27 文字を超えるテーブル名は、3 文字の接尾辞に対応するために切り捨てられます。したがって、切り捨てられないテーブルの名前を指定することをお勧めします。切り捨てが発生すると、メッセージがログに書き込まれます。

制限事項 このオプションは、Load および Update オペレーター用に提供されています。これは Stream オペレーターによって無視されます。

一時停止したジョブを再開する場合を除き、既存のテーブルの名前は使用できません。

詳細

ErrorTable2 には、一意のプライマリインデックスの制約に違反したレコードが含まれています。このタイプのエラーは、Load または Update オペレータージョブの適用フェーズ中に発生します。

ユーティリティテーブルの 別の場所を指定するために TD_LOGDB=テーブルオプションを使用しない限り、エラーテーブルはデフォルトのユーザー(ログオン)データベースに作成され、オプションでスキーマで修飾されます。

エラーテーブルの形式とエラー修正の手順については、*Teradata Parallel Transporter リファレンス*を参照してください。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)
- “TD_LOGDB= テーブルオプション” (1141 ページ)

TD_LOG_MECH_TYPE= テーブルオプション

バルクロードのログオンメカニズムを指定します。

カテゴリ:	バルクロード
デフォルト:	LDAP
制限事項:	このテーブルオプションは、CAS サーバーではサポートされていません。
要件	“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。
データソース:	Teradata

構文

TD_LOG_MECH_TYPE= *mechanism*

引数

mechanism
ログオン認証メカニズムを指定します。現在、有効な値は LDAP です。

要件 ドライバーには 8 文字の入力値が必要です。サブミットするには、次のように 4 つのスペースを値に埋め込みます。

```
TD_LOG_MECH_TYPE="LDAP    "
```

詳細

ユーザーが UID 接続オプションを使用して LDAP 経由で接続した場合、このオプションを使用する理由はありません。ユーザーが接続文字列で UID の LDAP 認証を指定しなかった場合は、TD_LOG_MECH_TYPE=および TD_LOG_MECH_DATA=を使用して、TPT の LDAP 認証を許可します。このオプションを TD_LOG_MECH_DATA=とともに使用して、UID オプションで指定された LDAP ID を別の LDAP ID でオーバーライドすることもできます。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)
- “TD_LOG_MECH_DATA= テーブルオプション” (1139 ページ)

TD_LOG_MECH_DATA= テーブルオプション

ログオンメカニズムの追加データを指定します。

カテゴリ:	バルクロード
デフォルト:	なし
制限事項:	このテーブルオプションは、CAS サーバーではサポートされていません。
要件	“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。
データソース:	Teradata

構文

TD_LOG_MECH_DATA="*string*"

引数

string
TD_LOG_MECH_TYPE=テーブルオプションと組み合わせて使用し、認証用の資格情報を指定します。現在、LDAP 資格情報のみが受け入れられます。資格情報は次の形式である必要があります。

```
TD_LOG_MECH_DATA="authcid=value  password=value  realm=value"
```

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)
- “TD_LOG_MECH_TYPE= テーブルオプション” (1138 ページ)

TD_LOG_TABLE= テーブルオプション

再起動ログテーブルの名前を指定します。

カテゴリ:	バルクロード
デフォルト:	*ttname*_RS。ここで、*ttname* はターゲットテーブルの名前です。
制限事項:	このテーブルオプションは、CAS サーバーではサポートされていません。
要件	“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。
データソース:	Teradata

構文

TD_LOG_TABLE= *name-of-table*

引数

name-of-table
Load、Stream、および Update オペレーターの再起動ログテーブルの名前を指定します。再起動ログテーブルには、再起動情報が含まれます。30 文字を超えるテーブル名は、3 文字の接尾辞に対応するために切り捨てられます。したがって、切り捨てられないテーブルの名前を指定することをお勧めします。切り捨てが発生すると、メッセージがログに書き込まれます。

詳細

再起動ログテーブルは、TD_LOGDB=テーブルオプションを使用して別の場所を指定しない限り、ユーザーのデフォルト(ログオン)データベースに作成されます。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)
- “TD_DROP_LOG_TABLE= テーブルオプション” (1133 ページ)
- “TD_LOGDB= テーブルオプション” (1141 ページ)
- “TD_WORKING_DB= テーブルオプション” (1156 ページ)

TD_LOGDB= テーブルオプション

TPT ユーティリティテーブルが 作成されるデータベースを指定します。

カテゴリ:	バルクロード
制限事項:	このテーブルオプションは、CAS サーバーではサポートされていません。
要件	“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。
データソース:	Teradata

構文

TD_LOGDB= *database-name*

引数

database-name
ユーティリティテーブルを 作成するデータベースの名前を指定します。このオプションが指定されていない場合、ユーティリティテーブルは 指定された SCHEMA に作成されます。SCHEMA が指定されていない場合、テーブルはデフォルトのユーザー(ログオン)データベースに作成されます。ユーティリティテーブルには、ErrorTable1、ErrorTable2、再起動ログテーブル、および作業テーブルが含まれます。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)

TD_MAX_SESSIONS= テーブルオプション

TPT がジョブに対して取得できるログオンセッションの最大数を指定します。

カテゴリ: バルクロード

デフォルト: 4 (セッション)

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件 “BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。

データソース: Teradata

構文

TD_MAX_SESSIONS= *integer*

引数

integer
Teradata Parallel Transporter がオペレータージョブに対して取得できるログオンセッションの最大数を指定します。最大値は、使用可能な AMP の数を超えることはできません。詳細については、Teradata のドキュメントを参照してください。

詳細

TD_MAX_SESSIONS=値はゼロより大きくなければなりません。1 未満の値を指定すると、ジョブが終了します。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)

TD_MIN_SESSIONS= テーブルオプション

ジョブが開始する前に TPT が取得するセッションの最小数を指定します。

カテゴリ: バルクロード

デフォルト: 1 (セッション)

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件 “BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。

データソース: Teradata

構文

TD_MIN_SESSIONS= *integer*

引数

integer
TPT がジョブを開始するために必要なセッションの最小数を指定します。この数は、Load、Stream、および Update オペレーターに適用されます。

詳細

TD_MIN_SESSIONS=値は、ゼロより大きく、セッションの最大数以下でなければなりません。1 未満の値を指定すると、アクティブなオペレーターが終了します。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)

TD_NOTIFY_LEVEL= テーブルオプション

ログイベントが記録されるレベルを指定します。

カテゴリ: バルクロード

デフォルト: OFF

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件 “BULKOPTS= テーブルオプション” (1079 ページ)で指定する必要があります。

データソース: Teradata

構文

TD_NOTIFY_LEVEL= *level*

引数

level
イベントが報告されるレベルを指定します。有効な設定は次のいずれかです。

OFF	イベントの通知は提供されません(デフォルト値)。
LOW	初期化、CLIv2/DBS エラー、終了イベントのみ。
MEDIUM	Err1 と Err2 を除くすべてのイベント。
HIGH	すべてのイベント。

詳細

TD_NOTIFY_LEVEL=を設定する場合は、TD_NOTIFY_METHOD=を設定して、イベントの報告方法を示します。

関連項目

テーブルオプション:

- “TD_NOTIFY_METHOD= テーブルオプション” (1144 ページ)

TD_NOTIFY_METHOD= テーブルオプション

イベントを報告する方法を指定します。

カテゴリ: バルクロード

デフォルト: NONE

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件 “BULKOPTS= テーブルオプション” (1079 ページ)で指定する必要があります。

データソース: Teradata

構文

TD_NOTIFY_METHOD= *method*

引数

method
イベントの報告に使用する通知方法を指定します。有効な設定は次のいずれかです。

NONE	イベントの通知は提供されません(デフォルト値)。
MSG	イベントをログに送信します。イベントの宛先は、/etc/syslog.conf ファイルで指定されます。
EXIT	ユーザー定義の通知出口ルーチンにイベントを送信します。

詳細

TD_NOTIFY_METHOD=を設定する場合は、TD_NOTIFY_LEVEL=を設定して、報告するイベントの種類を示します。

関連項目

テーブルオプション:

- “TD_NOTIFY_LEVEL= テーブルオプション” (1143 ページ)
- “TD_NOTIFY_STRING= テーブルオプション” (1145 ページ)

TD_NOTIFY_STRING= テーブルオプション

システムログに送信されるすべてのメッセージの 前にある文字列を定義します。

カテゴリ:	バルクロード
制限事項:	このテーブルオプションは、CAS サーバーではサポートされていません。
要件	“BULKOPTS= テーブルオプション” (1079 ページ)で指定する必要があります。
データソース:	Teradata

構文

TD_NOTIFY_STRING= *string*

引数

string
システムログに送信されるすべてのメッセージの前に付けられるオプションのユーザー指定の文字列です。この文字列は、ユーザー定義の通知出口ルーチンにも送信されます。NotifyMethod が MSG の場合、最大長は 16 バイトです。NotifyMethod が EXIT の場合、最大長は 80 バイトです。

関連項目

テーブルオプション:

- “TD_NOTIFY_METHOD= テーブルオプション” (1144 ページ)

TD_PACK= テーブルオプション

マルチステートメント要求にパックするステートメントの数を指定します。

カテゴリ:	バルクロード
デフォルト:	20 (ステートメント)
制限事項:	このテーブルオプションは、CAS サーバーではサポートされていません。
要件	“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。
データソース:	Teradata

構文

TD_PACK= *number*

引数

number
マルチステートメント要求にパックするステートメントの数を指定します。許可されるステートメントの最大数は 600 です。

制限事項 このオプションは、Load および Update オペレーターによって無視されます。

関連項目

テーブルオプション:

- “TD_PACK_MAXIMUM= テーブルオプション” (1147 ページ)

TD_PACK_MAXIMUM= テーブルオプション

Stream オペレーターが現在の Stream ジョブのパック係数を決定できるようにします。

カテゴリ: バルクロード

デフォルト: TD_PACK=テーブルオプションと TD_PACK_MAXIMUM=NO が指定されていない限り、YES。

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件 “BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。

データソース: Teradata

注: デフォルト値 YES のサポートは、2024.08 から開始されます。

構文

TD_PACK_MAXIMUM= YES | NO

引数

YES
Stream オペレーターをトリガーして、現在の Stream ジョブの最大パック係数を動的に決定します。

制限事項 このオプションは、Load および Update オペレーターによって無視されます。

NO
マルチステートメント要求の TD_PACK=オプションで指定された数のステートメントをロードします。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)

TD_PAUSE_ACQ= テーブルオプション

ロードジョブの取得フェーズと適用フェーズの間に一時停止を強制します。

カテゴリ: バルクロード

デフォルト: NO

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件 “BULKOPTS= テーブルオプション” (1079 ページ)で指定する必要があります。

データソース: Teradata

構文

TD_PAUSE_ACQ= YES | NO

引数

YES
取得フェーズの後にロードジョブを一時停止し、適用フェーズをスキップするように指定します。

制限事項 このオプションは、Load および Update オペレーター用に提供されています。これは Stream オペレーターによって無視されます。

NO
取得フェーズで Teradata データベースに送信されたすべての行を、AMP 上の最終的な宛先に分散させます。

TD_SESSION_QUERY_BAND= テーブルオプション

TPT セッションで使用するために、ユーザー指定の名前=値ペアの文字列を渡します。

カテゴリ: バルクロード

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件 “BULKOPTS= テーブルオプション” (1079 ページ)で指定する必要があります。

データソース: Teradata

構文

TD_SESSION_QUERY_BAND= *string*

引数

string
ジョブまたはプロセス用に生成された各 TPT セッションで使用する 1 つ以上のクエリバンド式を指定します。バンドは名前=値のペアとして指定されます。

詳細

バンドはデータベースに渡され、サーバーで負荷分散に使用されます。これらは、Teradata Multi-System Manager または TASM (Teradata Active System Management)によるトリガーメカニズムとしても 使用されます。

TD_TENACITY_HOURS= テーブルオプション

TPT オペレーターが Teradata データベースへのログオンを継続して試行する時間を指定します。

カテゴリ:	バルクロード
デフォルト:	0 (ゼロ)
制限事項:	このテーブルオプションは、CAS サーバーではサポートされていません。
要件	“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。
データソース:	Teradata

構文

TD_TENACITY_HOURS= *integer*

引数

integer
サーバーユーティリティスロットの 最大数が他のジョブによってすでに 占有されている場合に、TPT オペレーターが Teradata データベースへのログオンを試みる時間数を指定します。デフォルト値は 0 (ゼロ)です。これは、オペレーターが再試行せず、ジョブが失敗することを意味します。Load および Update オペレーターを使用する場合は、値を設定することをお勧めします。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)

TD_TENACITY_SLEEP= テーブルオプション

Teradata データベースへのログオンを再試行する前に、TPT オペレーターが一時停止する時間を指定します。

カテゴリ: バルクロード

デフォルト: 6 (分)

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件 “BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。

データソース: Teradata

構文

TD_TENACITY_SLEEP= *integer*

引数

integer
ジョブを再試行する前に TPT オペレーターが一時停止する時間(分)を指定します。このオプションは、TD_TENACITY_HOURS=が値に設定されている場合にのみ有効になります。

詳細

TD_TENACITY_SLEEP=値はゼロより大きくなければなりません。1 未満の値を指定すると、TPT オペレーターはエラーメッセージで応答し、ジョブを終了します。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)

TD_TPT_OPER= テーブルオプション

Teradata Parallel Transporter で使用されるロードオペレーターを指定します。

カテゴリ: バルクロード

デフォルト: STREAM

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件 “BULKOPTS= テーブルオプション” (1079 ページ)で指定する必要があります。

データソース: Teradata

構文

TD_TPT_OPER= *operator*

引数

operator
Teradata Parallel Transporter (TPT)がデータをロードするために使用するロードオペレーターを指定します。有効な値は、LOAD、STREAM、または UPDATE です。

LOAD	空のテーブルに挿入する Load オペレーターを指定します。これはロードオペレーターの中で最も高速です。
STREAM	Stream オペレーターを指定します。空の Teradata テーブルに行を挿入したり、既存のテーブルに追加したりするために使用できます。
UPDATE	既存のテーブルに挿入する Update オペレーターを指定します。Update オペレーターは、Stream オペレーターよりも高速です。

詳細

BULKLOAD=YES を指定すると、Teradata Parallel Transporter (TPT)を使用してデータがロードされます。TPT ジョブは、オペレーターを使用して実行されます。これらは、特定の抽出、ロード、および更新プロセスを実行する個別のオブジェクト指向モジュールです。

Update オペレーターを使用するときに最高のパフォーマンスを得るには、ロードの前に、すべての一意のセカンダリインデックス、外部キー参照、および結合インデックスをターゲットテーブルにドロップすることをお勧めします。

Teradata サーバーは、最大 16 個の同時ユーティリティスロットをサポートします。Stream オペレーターはユーティリティスロットを 占有しません。Load および Update オペレーターはスロットを占有します。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)
- “TD_PAUSE_ACQ= テーブルオプション” (1147 ページ)

TD_TRACE_LEVEL=, TD_TRACE_LEVEL_INF= テーブルオプション

ドライバートレースのトレースレベルを指定します。TD_TRACE_LEVEL は、プライマリトレースレベルを設定します。TD_TRACE_LEVEL_INF は、セカンダリトレースレベルを設定します。

カテゴリ:	バルクロード
デフォルト:	TD_OFF
制限事項:	このテーブルオプションは、CAS サーバーではサポートされていません。
要件	“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。
データソース:	Teradata

構文

TD_TRACE_LEVEL= *trace-level*

引数

trace-level
ドライバーの各インスタンスによって外部ログファイルに書き込まれる診断メッセージの種類を指定します。診断トレース機能は、問題の追跡と診断に役立つ、より詳細な情報(バージョン番号を含む)をログファイルに提供します。次のトレースレベルを使用できます。

TD_OFF	ドライバートレースを無効にします。これは、ドライバートレースとインフラストラクチャトレースの両方のデフォルト設定です。このデフォルトが変更されない限り、外部ログファイルは生成されません。デフォルト値は、トレースを無効にすることと同じです。
TD_OPER	ドライバー固有のアクティビティのトレース 機能をアクティブにします。PauseAcq 属性に値が指定されていないということは、ドライバーの更新ジョブが取得フェーズと適用フェーズの両方を一時停止せずに実行することを意味します。これにより、取得フェーズで Teradata データベースに送信されたすべての行を、AMP 上の最終的な宛先に分散させます。
TD_OPER_CLI	CLIv2 関連のアクティビティ (Teradata データベースとの対話)のトレース機能をアクティブにします。

TD_OPER_NOTIFY	通知機能に関連するアクティビティのトレース機能をアクティブにします。
TD_OPER_OPCOMMON	opcommon ライブラリに関連するアクティビティのトレース機能をアクティブにします。
TD_OPER_ALL	すべてのトレースアクティビティのトレース 機能をアクティブにします。

詳細

トレースレベルが TD_OFF 以外の値に設定されている場合、ドライバーのインスタンスごとに外部ログファイルが作成されます。

インフラストラクチャトレースのトレースレベルは、Teradata サポートから指示された場合にのみ使用してください。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)

TD_TRACE_OUTPUT= テーブルオプション

トレースメッセージに使用される外部ファイルの名前を指定します。

カテゴリ:	バルクロード
デフォルト:	なし
制限事項:	このテーブルオプションは、CAS サーバーではサポートされていません。
要件	“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。
データソース:	Teradata

構文

TD_TRACE_OUTPUT= *filename*

引数

filename
トレースに使用する外部ファイルの名前を指定します。指定した名前のファイルがすでに存在する場合、既存のファイルは上書きされます。新しいファイル名は、ドライバーの名前とタイムスタンプを使用して作成されます。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)

TD_UNICODE_PASSTHRU= テーブルオプション

パススルー文字(PTC)を Teradata にインポートおよび Teradata からエクスポートできるようにします。

カテゴリ:	バルクロード
別名:	TPT_UNICODE_PASSTHRU=
デフォルト:	NO
制限事項:	このテーブルオプションは、CAS サーバーではサポートされていません。 この属性には、Teradata データベースのバージョン 16.10 以降が必要です。
要件	“BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。
データソース:	Teradata

構文

TD_UNICODE_PASSTHRU= YES | NO

引数

YES
PTC のインポートとエクスポートが許可されます。

NO
PTC のインポートとエクスポートは許可されません。これがデフォルト設定です。

詳細

TD_UNICODE_PASSTHRU=セッション属性は、Teradata TPT (Teradata Parallel Transporter)が LOAD、UPDATE、および STREAM 操作のために作成するすべての接続に対する PTC の使用を制御します。PTC は、可能なすべての Unicode 文字のセットから、Teradata でサポートされている 6.0 BMP 文字と非文字のセットを除いたものです。詳細については、Teradata データベース管理ドキュメントの"Loading Unicode Data with Unicode Pass Through"を参照してください。

TD_UNICODE_PASSTHRU=テーブルオプションは、TPT_UNICODE_PASSTHRU=接続オプションの設定をオーバーライドします。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)

TD_WORK_TABLE= テーブルオプション

TPT 作業テーブルの名前を指定します。

カテゴリ:	バルクロード
デフォルト:	*ttname*_WT。ここで、*ttname* はターゲットテーブルの名前です。
制限事項:	このテーブルオプションは、CAS サーバーではサポートされていません。
要件	“BULKOPTS= テーブルオプション” (1079 ページ)で指定する必要があります。
データソース:	Teradata

構文

TD_WORK_TABLE= *name-of-table*

引数

name-of-table
TPT 作業テーブルの名前を指定します。27 文字を超えるテーブル名は、3 文字の接尾辞に対応するために切り捨てられます。したがって、切り捨てられないテーブルの名前を指定することをお勧めします。切り捨てが発生すると、メッセージがログに書き込まれます。

制限事項 このオプションは、Update オペレーター用に提供されています。Load および Stream オペレーターでは無視されます。

詳細

ユーティリティテーブルの 別の場所を指定するために TD_LOGDB=テーブルオプションを使用しない限り、作業テーブルはデフォルトのユーザー(ログオン)データベースに作成され、オプションでスキーマで修飾されます。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)
- “TD_DROP_WORK_TABLE= テーブルオプション” (1134 ページ)
- “TD_LOGDB= テーブルオプション” (1141 ページ)

TD_WORKING_DB= テーブルオプション

挿入テーブルが作成されるデータベースを指定します。

カテゴリ: バルクロード

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件 “BULKOPTS= テーブルオプション” (1079 ページ)で指定する必要があります。

データソース: Teradata

構文

TD_WORKING_DB= *database-name*

引数

database-name
データベース名を指定します。このオプションが指定されていない場合、テーブルは指定された SCHEMA に作成されます。SCHEMA が指定されていない場合、テーブルはデフォルトのユーザー(ログオン)データベースに作成されます。

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)

THREADNUM= テーブルオプション

SPD Engine データセットを処理するために SPD Engin が起動できる I/O スレッドの最大数を指定します。

カテゴリ: テーブルコントロール

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

データソース: SPD Engine データセット

構文

THREADNUM= *n*

引数

n
スレッド数を指定します。SPDEMAXTHREADS=システムオプションが設定されている場合、デフォルトはその値です。それ以外の場合、デフォルトはコンピューターの CPU 数の 2 倍です。

詳細

THREADNUM=を使用すると、SPD Engine データセットを処理するために SPD Engine が生成する 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 処理のいずれかに適用されます。

- WHERE 式の処理
- 並列インデックス作成
- スレッド対応アプリケーションによって要求された I/O

THREADNUM=を調整すると、システム管理者は SPD Engine が任意のプロセスに使用できる 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、プロセスが最大で 4 つの CPU に制限されるため、他のユーザーやアプリケーションのスループットが向上します。

THREADNUM=が 1 より大きい場合、並列処理が発生する可能性があります。したがって、物理的な順序が出力で保持されない場合があります。THREADNUM=1 の設定は、並列処理が発生しないことを意味します。

このオプションを使用して、WHERE 式の評価のスケーラビリティを調べることもできます。

構成可能なシステムオプションである SPDEMAXTHREADS=は、システムリソースの消費に上限を課すため、THREADNUM=値を制約します。

TYPE= テーブルオプション

特別に構造化された SAS データセットのデータセットタイプを指定します。

カテゴリ: テーブルコントロール

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

データソース: SAS データセット、SPD Engine データセット

構文

TYPE= *data-set-type*

引数

data-set-type
データセットの特殊なタイプを指定します。

詳細

TYPE=テーブルオプションを使用して、適切な形式で特殊なデータセットを作成するか、データセットの特殊なタイプを識別します。

ほとんどのデータセットには指定されたタイプがありません。ただし、一部の SAS/STAT プロシジャで使用される特別に構造化されたデータセットがいくつかあります。これらの SAS データセットには特殊な列と行が含まれており、通常は SAS 統計プロシジャによって 作成されます。特殊なデータセットのほとんどは SAS/STAT ソフトウェアで使用されるため、*SAS/STAT ユーザーガイド*で説明されています。

他の値は他の SAS ソフトウェア製品で使用可能であり、適切なドキュメントで説明されています。

UNIQUESAVE= テーブルオプション

一意のインデックスを持つデータセットにオブザベーションを挿入するときに、一意でないキー値を持つオブザベーション(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: INSERT ステートメント

カテゴリ: SAS インデックス使用のユーザーコントロール

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

操作: SPDSUSDS=自動マクロ変数と組み合わせて使用

データソース: SPD Engine データセット

構文

UNIQUESAVE= YES | NO

引数

YES
拒否されたオブザベーションを、マクロ変数 SPDSUSDS=への参照によってアクセスできる別のシステム作成テーブルに書き込みます。

NO
拒否されたオブザベーションを別のテーブルに書き込みません(つまり、一意でないキー値を無視します)。

詳細

一意のインデックスを持つデータセットにオブザベーションが挿入されると、拒否されたオブザベーションは無視されます。UNIQUESAVE=YES を指定すると、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、その名前はマクロ変数 SPDSUSDS に保存されます。データセット名のかわりに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することができます。

USER= テーブルオプション

HDFS ユーザー名を指定します。

カテゴリ: バルクロード

別名: USERID=、UID=

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

要件 “BULKOPTS= テーブルオプション” (1079 ページ)内で指定する必要があります。

操作: (オプション)“PASSWORD= テーブルオプション” (1113 ページ)とともに使用します。

データソース: Impala

構文

USER=*HDFS-user*

引数

HDFS-user
HDFS ユーザーを表す名前を指定します。名前にスペースが含まれていない限り、値を囲む引用符はオプションです。

例

BULKLOAD=YES BULKOPTS=(USER=HADOOP PWD="*hdfs-password*" CFG="*config-path*")

関連項目

テーブルオプション:

- “BULKLOAD= テーブルオプション” (1074 ページ)
- “PASSWORD= テーブルオプション” (1113 ページ)

WHERENOINDEX= テーブルオプション

WHERE 式の評価を行うときに除外するインデックスのリストを指定します。

カテゴリ: SAS インデックス使用のユーザーコントロール

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。IDXWHERE=NO では使用できません。

データソース: SPD Engine データセット

構文

WHERENOINDEX= (*name1 name2*...)

引数

(*name1 name2*...)
使用から除外するインデックス名のリストを指定します。

関連項目

テーブルオプション:

- “IDXWHERE= テーブルオプション” (1102 ページ)

WRITE=テーブルオプション テーブルオプション

SAS ファイルに WRITE パスワードを割り当てて、ユーザーがファイルに書き込めないようにするか、書き込み保護されたファイルにアクセスできるようにします。

カテゴリ: テーブルコントロール

制限事項: このテーブルオプションは、CAS サーバーではサポートされていません。

データソース: SAS データセット、SPD Engine データセット

注: この操作の後にログをチェックして、パスワード値が表示されていないことを確認してください。詳細については、*"ログからパスワードと暗号化キーの値を消去する" (SAS プログラマガイド: エッセンシャル)*を参照してください。

構文

WRITE= *write-password*

引数

write-password
有効な SAS 名である必要があります。

詳細

このオプションを使用して、パスワードを割り当てたり、書き込み保護されたファイルにアクセスしたりできます。

注: SAS パスワードは、SAS 以外の SAS ファイルへのアクセスを制御しません。SAS 外の SAS ファイルへのアクセスを制御するには、オペレーティングシステムが提供するユーティリティとファイルシステムのセキュリティ 制御を使用する必要があります。

9

FedSQL とシステムオプション

システムオプションの FedSQL サポート **1163**

システムオプションの FedSQL サポート

FedSQL は、オプションが指定されてから変更されるまで、SAS プログラム全体または対話型 SAS セッションの処理に影響を与えるオプションを提供しません。
FedSQL は、特定の SAS 呼び出しオプションを除いて、SAS システムオプションもサポートしません。これらの SAS システムオプションは、該当する場合、このドキュメントで説明されています。

10

FedSQL バージョン PROC SQL

言語の概要 **1165**
SQL プロシジャと FedSQL の高レベルの相違点 **1166**
ステートメントの比較 **1168**
引用要件 **1170**
比較演算子 **1172**
DICTIONARY テーブル **1174**
PROC SQL DICTIONARY クエリ 1174
FedSQL DICTIONARY クエリ 1174
FedSQL での一般的な PROC SQL タスクの実行 **1175**
概要 1175
FedSQL で既存のテーブルを上書きする 1175
テーブル作成時の SAS 出力形式、入力形式、およびラベルの定義 1176
テーブルに値を挿入する 1177
クエリ式で SAS 出力形式とラベルを指定する 1178
FedSQL を使用してタイムゾーン変換を実行する 1179
FedSQL で GROUP BY 再マージクエリを書き換える 1180
CALCULATED キーワードを指定する PROC SQL クエリを書き換える 1184
SQL 式で複数の引数を指定する 1185
明示的パススルーを使用する 1187
FedSQL で SAS マクロを使用する 1188
暗黙的 SQL パススルー: 比較 1189

言語の概要

SAS では、SQL プロシジャと FedSQL という 2 つの SQL 言語が提供されています。SQL プロシジャは、Base SAS と緊密に統合された SQL 言語を好むユーザーに適しています。FedSQL は、ANSI SQL データ型と ANSI SQL 規則のサポートという形で、ANSI SQL 規格にさらに準拠した言語を必要とするユーザー向けに提供されています。この付録では、FedSQL プログラムを作成する際に知っておく必要がある言語間の相違点を指摘します。

SQL プロシジャと FedSQL の高レベルの相違点

SQL プロシジャは、ANSI SQL:1992 の SAS 独自の実装です。SQL プロシジャは Base SAS と統合されており、SAS システムオプション、データセットオプション、出力形式、入力形式、SAS 関数などの SAS 言語要素をサポートしています。SQL プロシジャは、デフォルトで SAS クライアントで要求を処理します。SAS/ACCESS ソフトウェアで使用する場合、プロシジャは文書化された SAS 関数の暗黙的パススルーをサポートし、明示的パススルーをサポートして、ユーザーがネイティブ SQL ステートメントをサードパーティのデータソースにサブミットできるようにします。

FedSQL は、ANSI SQL:1999 コア規格の SAS 独自の実装であり、ANSI 1999 コア準拠機能と独自の拡張機能を備えています。FedSQL 言語は、ANSI SQL 規格に基づくベンダーニュートラルな SQL 言語として設計されました。FedSQL は、基盤となるドライバーを使用して、FedSQL 要求をターゲットデータソースのネイティブ言語に内部的に変換します。FedSQL 言語は、暗黙的パススルーの条件を満たす要求に対して、完全なクエリの暗黙的パススルーと部分的パススルーをサポートしています。暗黙的パススルーの要件を満たさない要求は、FedSQL によって処理されます。明示的 SQL パススルーもサポートされています。

FedSQL は、SAS 言語要素に対して限定的なサポートを提供します。FedSQL は、ほとんどの SAS および ANSI SQL 関数をサポートしています。SAS ストレージエンジンでテーブルを作成する際に、SAS 出力形式、入力形式、およびラベルをサポートします。次の SAS 言語要素はサポートされていません。

- SAS システムオプション(ドキュメントに記載されている場合を除く)
- SASHELP などの連結ライブラリ
- SAS データセットオプション。(FedSQL には独自のテーブルオプションがあり、SAS DS2 言語でもサポートされています。)

次の表では、2 つの言語のその他の機能を比較しています。

表 10.1　SAS SQL 言語と SAS FedSQL 言語の高レベルの比較

機能	SAS SQL 言語	SAS FedSQL 言語
ユーザーインターフェイス	Base SAS の SQL プロシジャ	Base SAS の FEDSQL プロシジャ SAS Cloud Analytic Services の fedSql.execDirect CAS アクション SAS Federation Server

機能	SAS SQL 言語	SAS FedSQL 言語
データ型	SAS 文字と数値、および VARCHAR	ANSI SQL データ型。リストについては、FedSQL データ型 (14 ページ)を参照してください。
null 値と欠損値の処理	SAS 欠損モード	ANSI null セマンティクスまたは SAS 欠損モード (SAS タイプのみ)
日時の処理	出力形式付きの SAS 数値として保存された SAS 日時値。[1]	ANSI 日時データ型(SAS データセットへの書き込み時に出力形式付きの DOUBLE に変換されます)[2]
処理	非パーティション分割データセットに対してシングルスレッド処理を使用します。	CAS の外部でも、必要に応じてマルチスレッドアルゴリズムを使用します。
SAS Cloud Analytic Services サポート	CAS エンジンを介して CAS テーブルにアクセスします。テーブルは SAS Compute Server で処理されます。	適切なインターフェイスまたはオプションが使用されている場合、FedSQL ステートメントは、fedSql.execDirect アクションの CAS で実行されます。[3]
他の言語との統合	Base SAS	Base SAS との部分的な統合。 FedSQL は、SAS DS2 言語パッケージメソッドを関数として呼び出すことができます。 FedSQL ステートメントは、SAS DS2 プログラム内に埋め込むことができます。 fedSql.execDirect アクションを使用すると、CASL、Python、Lua、R、Java プログラムから FedSQL ステートメントをサブミットできます。
データソースサポート	適切なソフトウェアがインストールされている場合、SAS/ACCESS ソフトウェアを介してすべての SAS デ	適切なソフトウェアがインストールされている場合、SAS/ACCESS ソフトウェアを介してすべての SAS デ

機能	SAS SQL 言語	SAS FedSQL 言語
	ータソースとサードパーティ DBMS を使用できます。	ータソースとほとんどのサードパーティ DBMS を使用できます。使用可能なサードパーティのデータソースは、インターフェイスによって異なります。

1 PROC SQL は、TIMEZONE=システムオプションからタイムゾーン情報を取得します。
2 FedSQL は、オペレーティングシステムからタイムゾーン情報を取得します。
3 FEDSQL プロシジャでサブミットされた FedSQL ステートメントは、デフォルトで Base SAS エンジンで処理されます。ただし、CAS セッションを確立し、SESSREF=プロシジャオプションで CAS セッション名(または SESSUUID=オプションでセッション ID)を指定すると、FedSQL ステートメントは fedSql.execDirect アクションに渡され、CAS で処理されます。

DS2 プログラムで FedSQL を使用する方法の詳細については、付録 3, “FedSQL と DS2 の使用” (1253 ページ)を参照してください。

ステートメントの比較

表 10.2 言語に共通するステートメント

ステートメント	説明
ALTER TABLE	テーブルの列を追加または削除し、列の定義を変更します。
CREATE INDEX	指定されたテーブルの列にインデックスを作成します。
CREATE TABLE	新しいテーブルを作成します。 **注:** PROC SQL は、新しいテーブルと同じ名前を持つ既存のテーブルを自動的に上書きします。対照的に、FedSQL は名前の一意性を保持します。FedSQL で同じ名前の新しいテーブルを作成するには、DROP TABLE ステートメントで既存のテーブルを明示的に削除する必要があります。
CREATE VIEW	1 つ以上のテーブルまたは他のビューからデータのビューを作成します。PROC SQL は SAS ビューを作成します。PROC FEDSQL は、FedSQL ビューを作成します。 **注:** PROC SQL には、FedSQL にはないオプションの句 REMEMBERING および USING LIBNAME が含まれています。
DELETE	テーブルから行を削除します。

ステートメント	説明
DESCRIBE TABLE	テーブルを記述した SQL を取得し、結果セットを返します。
DESCRIBE VIEW	ビューを記述した SQL を取得し、結果セットを返します。
DROP INDEX	指定されたインデックスをテーブルから削除します。
DROP TABLE	データベースからテーブルを削除します。
DROP VIEW	データベースからビューを削除します。
EXECUTE	ネイティブの非クエリ SQL ステートメントを DBMS に送信します。
INSERT	指定されたテーブルに行を追加します。
SELECT	テーブルからデータの列と行を取得します。どちらの言語も、SELECT FROM 句の CONNECTION TO コンポーネントでのネイティブ SQL クエリのサブミットをサポートしています。 **注:** FROM 句は、FedSQL ではオプションです。この構文は、定数値を使用して関数の効果をテストするのに役立ちます。 **注:** INTO *macro-variable-specification* 句は、FedSQL ではサポートされていません。
UPDATE	テーブルの既存の行の列の値を変更します。

FedSQL は、SQL プロシジャがサポートしていないトランザクションデータベース用に次のステートメントを提供します。

表 10.3 FedSQL で提供される追加ステートメント

ステートメント	説明
BEGIN	複数のステートメントで構成されるトランザクションの先頭をマークします。
COMMIT	トランザクションの開始以降に実行された変更をデータベースの永続的な部分にします。
ROLLBACK	トランザクションの変更をトランザクションの先頭にロールバックします。

PROC SQL は、FedSQL がサポートしていない次のステートメントを提供します。

表 10.4　PROC SQL で提供される追加ステートメント

ステートメント	説明
CONNECT	SAS/ACCESS ソフトウェアがサポートする DBMS との接続を確立します。[1]
DESCRIBE TABLE CONSTRAINTS	テーブルで定義されている一貫性制約を取得し、結果セットを返します。
DISCONNECT	SAS/ACCESS インターフェイスがサポートする DBMS との接続を終了します。[1]
RESET	プロシジャを再起動せずに PROC SQL オプションをリセットします。これらのオプションを使用すると、プロシジャの実行と出力を制御できます。[2]
VALIDATE	式を実行せずに、クエリ式の構文とセマンティクスの正確性をチェックします。[3]

1 FedSQL は、EXECUTE ステートメントまたは SELECT ステートメントで FROM CONNECTION TO 句が指定されている場合でも、libref を介して DBMS に接続します。

2 FedSQL インターフェイスには CNTL=オプションがあり、CAS での FedSQL クエリプランニングとクエリ実行のパラメーターを制御できます。CNTL=は PROC FEDSQL プロシジャオプションであり、fedSql.execDirect アクションのパラメーターです。

3 PROC FEDSQL には NOEXEC および_METHOD プロシジャオプションがあり、クエリを実行せずにライブラリのクエリプランを表示できます。

このセクションで FedSQL について説明されているすべてのステートメントが CAS でサポートされているわけではありません。CAS での FedSQL ステートメントのサポートについては、*SAS Viya プラットフォーム: SAS Cloud Analytic Services の FedSQL プログラミング*を参照してください。

引用要件

PROC SQL には柔軟な引用要件があります。

FedSQL は、標準識別子と区切り識別子の ANSI SQL: 1999 コア規格をサポートしています。

- 二重引用符で囲まれた文字列は、テーブル、列、またはカタログ名として解釈されます。
- 単一引用符で囲んだ文字列はリテラルまたは文字定数です。

詳細については、“識別子” (18 ページ)および“リテラル” (20 ページ)を参照してください。

表 10.5 FedSQL と PROC SQL の引用の違いの例

タスク	FedSQL	PROC SQL
列名または列エイリアスで大文字と小文字またはスペースを保持する	二重引用符を使用します。 例: "ListPrice" "List Price" as "ListPrice" as "Data Entry Errors for &col" **注:** 区切り識別子を使用すると、FedSQL は大文字と小文字を区別します。大文字と小文字の区別は強制されません。大文字と小文字のみが異なる値が同じと見なされるか、異なると見なされるかは、ターゲットデータソースによって決まります。	識別子の大文字と小文字はデフォルトで保持されます。引用符を使用すると、エラーが返されます。PROC SQL が PRESERVE_COL_NAMES=および PRESERVE_MEM_NAMES=オプションをサポートする SAS/ACCESS エンジンと通信している場合を除き、識別子は SAS 名の規則に従う必要があります。 これらのオプションが YES に設定されている場合、データソースは独自の大文字と小文字の区別規則を適用します。 エイリアスにスペースまたはアンダースコア(_)以外の特殊文字が含まれる場合は、二重引用符を使用します。 例: ListPrice as ListPrice as "Data Entry Errors for &col"
LABEL 修飾子の値を指定する	単一引用符を使用します。FedSQL はラベル値をリテラルのように扱います。 例: label 'Continent'	ラベル値は、単一引用符または二重引用符で囲んで指定できます。 例: label="Continent" label='Continent'
リテラル	単一引用符を使用します。 例: where col1='count me'	リテラルは、単一引用符または二重引用符で囲んで指定できます。 例: where col1='count me' where col1="count me"
予約語をエスケープする	ベストプラクティスとして、予約語を識別子として使用しないでください。	SAS データセットオプション RENAME=を使用して、クエリ中に予約語を使用する列の

	予約語を使用する必要がある場合は、二重引用符で囲みます。 例: "CASE" “FedSQL 予約語” (72 ページ)を参照してください。	名前を変更します。別の方法として、予約語を二重引用符で囲み("CASE")、PROC SQL プロシジャオプション DQUOTE=ANSI を設定します。

比較演算子

PROC SQL は、ほとんどの比較演算子の記号とニーモニックをサポートしています。ニーモニックは、記号のかわりに使用できる名前です。FedSQL では、ほとんどの比較演算子で記号を使用する必要があります。例外は、論理 AND および OR 演算子です。これらの演算子にはニーモニックが必要です。アンパサンド(&)および単一バー(|)記号は、AND および OR 演算子では使用できません。

表 10.6 PROC SQL と FedSQL の構文の違いの例

説明	PROC SQL 演算子	FedSQL 演算子
等しい	=、eq	=
等しくない	¬=、^=、< >、ne	^=、!=、<>[1]
より大きい	>、gt	>
より小さい	<、lt	<
以上	>=、ge	>=
以下	<=、le	<=
論理 AND	&、AND	AND
論理 OR	|、OR	OR

1 PROC SQL での^=演算子の使用によっては、FedSQL ではそのままでは機能しません。

PROC SQL とは異なり、FedSQL は SELECT リストのブール結果をサポートしません。例として、次のコードは PROC SQL で正常に実行されます。

```
/* Input data set */
data temp;
  eps = constant('maceps');
  x = 1;
  x0 = x +(eps/2);
```

```
  x1 = x + eps;
  x2 = x + eps * 2;
run;

/* PROC SQL test */
proc sql;
  select *
      , (x^=x0) as z0
      , (x^=x1) as z1
      , (x^=x2) as z2
  from TEMP;
quit;
```

eps	x	x0	x1	x2	z0	z1	z2
2.22E-16	1	1	1	1	0	1	1

PROC FEDSQL で実行すると、同じコードがエラーになります。

```
117  proc fedsql;
118       select *
119           , (x^=x0) as z0
120           , (x^=x1) as z1
121           , (x^=x2) as z2
122       from TEMP;
ERROR: Boolean results are not supported in select lists.
123  quit;
```

FedSQL を使用する場合は、等しくない演算子を WHERE 句または CASE 式で指定する必要があります。次に例を示します。

```
proc fedsql;
  select *
    from TEMP
    where (x^=x0) OR
      (x^=x1) OR
      (x^=x2);
quit;
```

eps	x	x0	x1	x2
2.22E-16	1	1	1	1

```
proc fedsql;
  select *,
    case when x ^= x0 then 1 else 0 end as z0,
    case when x ^= x1 then 1 else 0 end as z1,
    case when x ^= x2 then 1 else 0 end as z2
  from TEMP;
quit;
```

eps	x	x0	x1	x2	Z0	Z1	Z2
2.22E-16	1	1	1	1	0	1	1

DICTIONARY テーブル

PROC SQL と FedSQL の DICTIONARY テーブルには多くの違いがあります。このセクションでは、主な違いについて説明します。

PROC SQL DICTIONARY クエリ

DICTIONARY テーブルに対する PROC SQL クエリは、SAS LIBNAME エンジンによってレポートされるメタデータに依存します。リレーショナル DBMS への LIBNAME エンジン接続は、単一のスキーマに制限されています。したがって、PROC SQL は、接続されたスキーマ内のテーブルと列の情報のみをレポートします。たとえば、次のクエリはスキーマ Student の情報を返します。

```
libname my_ora oracle user=student password=Metadata0 schema=student;

proc sql;
  select *   from dictionary.columns
    where libname = 'MY_ORA';
quit;
```

PROC SQL 構文では、キーワード LIBNAME=を使用して libref を識別します。キーワード MEMNAME=を使用してテーブル名を指定します。

PROC SQL クエリの結果では、すべてのデータ型が CHAR または NUM のいずれかで、すべての変数名が最大 32 文字の長さであるため、サードパーティの DBMS で作業する場合は制限される可能性があります。

FedSQL DICTIONARY クエリ

FedSQL は、libref が参照されるときに、割り当てられた libref から内部的に呼び出されるスレッドドライバーを使用してデータにアクセスします。FedSQL 接続は、カタログレベルで行われます。次のようなカタログクエリは、呼び出し元のユーザーが読み取りアクセス権を持つすべてのテーブルの 情報をレポートするため、お勧めできません。これには SYSTEM テーブルが含まれ、他のスキーマのテーブルを含めることができます。

注: このサンプルコードでは、FedSQL ステートメントは PROC FEDSQL を使用して SAS に送信されます。

```
proc fedsql;
  select *
    from dictionary.columns
    where table_cat = 'MY_ORA';
```

```
quit;
```

かわりに、少なくとも FedSQL DICTIONARY クエリでカタログとスキーマ名を指定することをお勧めします。

```
proc fedsql;
  select *
    from dictionary.columns
    where table_cat = 'MY_ORA'
      and table_schema = 'STUDENT';
quit;
```

キーワード TABLE_CAT= (または CATALOG=)を使用して、カタログ名を指定します。スキーマ名を指定するには、キーワード TABLE_SCHEMA= (または SCHEMA=)を使用します。テーブルの名前を指定するには、キーワード TABLE_NAME= (または TABLE=)を使用します。

FedSQL DICTIONARY クエリの結果には、特に、FedSQL データ型とターゲットデータベースで使用されるデータ型の両方が含まれます。変数名は、ターゲットデータベースで許可されている長さです。

通常、FedSQL DICTIONARY クエリは、PROC SQL DICTIONARY クエリよりも遅くなります。これは、DICTIONARY テーブルのすべての 文字列が、ANSI SQL データベースで許可されているより大きな識別子文字列サイズに対応するために 1024 バイト長であるためです。

FedSQL での一般的な PROC SQL タスクの実行

概要

このセクションでは、非 ANSI 機能に依存する一般的な PROC SQL タスクを FedSQL で実行する方法について説明します。例で使用されるテーブルを作成するには、付録 4, “例で使用されているテーブル” (1255 ページ)を参照してください。

注: この例では、FedSQL ステートメントは PROC FEDSQL を使用して SAS に送信されます。

FedSQL で既存のテーブルを上書きする

FedSQL でテーブルを作成する場合、デフォルトでは既存のテーブルを上書きできません。最初に DROP TABLE ステートメントでテーブルを破棄する必要がありま

す。CREATE TABLE ステートメントと同じプロシジャ要求で既存のテーブルを削除するには、DROP TABLE ステートメントで FORCE オプションを次のように指定します。

```
proc fedsql;
  drop table test force;
  create table test(col1 char(5), col2 int);
  insert into test  values ('high', 5);
quit;
```

FORCE オプションは、FORCE が省略された場合に CREATE TABLE ステートメントが失敗する原因となる完了メッセージを抑制します。

CAS では、CREATE TABLE ステートメントで REPLACE=テーブルオプションを指定することにより、既存のセッションテーブルを上書きできます。次に例を示します。

```
cas mysess;
proc fedsql sessref=mysess;
  create table test2 {options replace=true} as
  select * from test1;
quit;
```

テーブル作成時の SAS 出力形式、入力形式、およびラベルの定義

注: このトピックの情報は、SAS エンジンでの FedSQL の使用に適用されます。CAS の FedSQL は、既存の入力テーブルのみからの FedSQL 出力テーブルの作成をサポートします。

FedSQL は、CREATE TABLE ステートメントの HAVING 句のデータ定義に対して、SAS FORMAT、INFORMAT、および LABEL 修飾子をサポートしています。FedSQL での修飾子の指定方法の例を次に示します。

```
proc fedsql;
  create table mytable
   (
    Name char(35) having format $35. informat $35. label 'Name',
   Capital char(35) having format $35. informat $35. label 'Capital',
   Population double having format comma15. informat comma15. label 'Population',
   Area double having format comma10. informat comma10. label 'Area',
   Continent char(30) having format $30. informat $30. label 'Continent',
   UNDate double having format year4. label 'UNDate'
 );
quit;
```

比較のために、PROC SQL での修飾子の指定方法の例を次に示します。

```
proc sql;
  create table mytableA
   (
    Name char(35) format=$35. informat=$35. label="Name",
    Capital char(35) format=$35. informat=$35. label="Capital",
    Population num format=comma15. informat=comma15. label="Population",
```

```
    Area num format=comma10. informat=comma10. label="Area",
    Continent char(30) format=$30. informat=$30. label="Continent",
    UNDate num format=year4. label="UNDate"
 );
quit;
```

FedSQL の修飾子については、次の点に注意してください。

- HAVING 句は、指定される最初の修飾子の前に置く必要があります。
- 値と修飾子は、等号のかわりに空白値で区切られます。
- LABEL 値は、単一引用符で囲む必要があります。
- 出力形式または入力形式を指定すると、データ型が CHAR でも DOUBLE でもない場合は、CHAR または DOUBLE に変換されます。
- FedSQL は、フォーマットカタログが使用可能なデータソースの表示用にデータが選択されると、出力形式を適用します。入力形式はデータに適用されません。これは情報提供のみを目的としています。クライアントアプリケーションは、入力形式をデータに適用する必要があります。

テーブルに値を挿入する

SQL プロシジャと FedSQL はどちらも、INSERT ステートメントで VALUES 句とクエリ式をサポートしています。ただし、FedSQL INSERT ステートメントには単一の VALUES 句を含めることができます。FedSQL は、スタックされた VALUES 句をサポートしていません。次のような要求は、FedSQL で構文エラーを返します。

```
proc sql;
  insert into mytableA
   values ('Afghanistan', 'Kabul', 17070323, 251825, 'Asia', 1946)
   values ('Albania', 'Tirane', 3407400, 11100, 'Europe', 1955)
   values ('Algeria', 'Algiers', 28171132, 919595, 'Africa', 1962)
   values ('Andorra', 'Andorra la Vella', 64634, 200, 'Europe', 1993);
quit;
```

FedSQL で同じ結果を得るには、複数の INSERT ステートメントをサブミットする必要があります。

```
proc fedsql;
  insert into mytableA values ('Afghanistan', 'Kabul', 17070323,
   251825, 'Asia', 1946);
  insert into mytableA values ('Albania', 'Tirane', 3407400, 11100,
   'Europe', 1955);
  insert into mytableA values ('Algeria', 'Algiers', 28171132,
   919595, 'Africa', 1962);
  insert into mytableA values ('Andorra', 'Andorra la Vella', 64634,
   200, 'Europe', 1993);
quit;
```

FedSQL でクエリ式を使用してテーブル値を挿入することもできます。次の例を考えてみましょう。

```
proc fedsql;
  drop table newcountries force;
```

```
create table newcountries
 (
  Name char(35) having format $35. informat $35. label 'Name',
  Capital char(35) having format $35. informat $35. label 'Capital',
  Continent char(30) having format $30. informat $30. label 'Continent',
  Population double having format comma15. informat comma15. label 'Population',
  Area double having format comma10. informat comma10. label 'Area',
  UNDate double having format year4. label 'UNDate'
 );
title "World's Largest Countries";
insert into newcountries
 select * from countries
 where population >= 130000000;
select * from newcountries;
quit;
```

この例では、Newcountries という名前の新しいテーブルを定義し、Countries (1256 ページ)テーブルに対する INSERT ステートメントでクエリ式を送信して、Newcountries テーブルにデータを入力します。

World's Largest Countries

Name	Capital	Population	Area	Continent	UNDate
Brazil	Brasilia	160,310,357	3,286,500	South America	1945
China	Beijing	1,202,215,077	3,696,100	Asia	1945
India	New Delhi	929,009,120	1,222,600	Asia	1945
Indonesia	Jakarta	202,393,859	741,100	Asia	1950
Russia	Moscow	151,089,979	6,592,800	Europe	1945
United States	Washington	263,294,808	3,787,318	North America	1945

テーブル Newcountries は、FedSQL CREATE TABLE ステートメントと AS 式を使用して作成することもできます。次に例を示します。

```
proc fedsql;
title "World's Largest Countries";
  create table newcountries2 as
  select * from countries
  where population >= 130000000;
  select * from newcountries;
quit;
```

ただし、FedSQL はクエリ式での FORMAT および LABEL 修飾子の使用をサポートしていません。

クエリ式で SAS 出力形式とラベルを指定する

注: このトピックの情報は、SAS エンジンおよび CAS での FedSQL の使用に適用されます。

SELECT ステートメントで列をフォーマットするには、PUT 関数を使用して一時出力形式を適用します。PUT 関数で適用される出力形式は、すべての FedSQL データソースでサポートされています。PUT 関数でサポートされている出力形式のリストについては、“PUT 関数でサポートされている出力形式(カテゴリ別)” (89 ページ)を参照してください。

SELECT ステートメントで列にラベルを付けるには、ラベルのかわりに列エイリアスを指定します。

次の例は、Payroll テーブルのクエリ式で形式とエイリアスを指定する方法を示しています:

```
proc fedsql;
  title 'Bonus Information';
  select IdNumber as "IdNumber",
    put(Salary,dollar8.) as "Salary",
    put(Salary*.025, dollar8.) as "Bonus"
    from payroll limit 10;
quit;
```

列に関数を適用し、その列にエイリアスが指定されていない場合、保存される列名は、リテラル"column" (PUT 関数の場合)または関数の名前になります。列エイリアスを使用すると、より意味のある名前を指定できます。

> ヒント　計算列に引用符で囲まれたエイリアスを使用する場合は、他の列にも引用符で囲まれたエイリアスを定義することを検討してください。
> FedSQL は、標準識別子を大文字で保存します。区切り識別子は指定されたとおりに保存されます。列エイリアスは、引用符で囲まない限り、大文字で表示されます。列エイリアスを使用すると、出力の列名の大文字と小文字が一致します。

注:

- FedSQL では、列エイリアスの前に AS キーワードを付ける必要があります。
- 列エイリアスで引用符を使用する場合は、二重引用符にする必要があります。

FedSQL を使用してタイムゾーン変換を実行する

日付と時間の処理が異なるため、FedSQL は、日付と時間に関連するすべての SAS 関数と出力形式をサポートしているわけではありません。特に、FedSQL は、タイムゾーンオフセットを計算する SAS 関数をサポートしていません。

GMTOFF()関数を使用して、タイムゾーン値を取得できます。次に例を示します。

```
proc fedsql;
  select gmtoff();
quit;
```

または、算術式を使用して、次のようにタイムゾーンオフセットを計算することもできます。

```
proc fedsql;
```

```
    select current_time - current_time_gmt() as offset;
quit;
```

サポートされている SAS の日付と時間の関数と出力形式については、“日付、時間、および日時関数” (62 ページ)および“SAS データセット、SPD Engine データセット、および SPD Server テーブルの日付、時間、および日時出力形式” (63 ページ)を参照してください。

FedSQL で GROUP BY 再マージクエリを書き換える

注: このトピックの情報は、SAS エンジンおよび CAS での FedSQL の使用に適用されます。

FedSQL は、PROCSQL 再マージ機能をサポートしていません。再マージクエリは統計分析を実行し、分析結果をデータの他の列と統合します。再マージは ANSI の概念ではありません。PROC SQL 要求に次の条件のいずれかが存在する場合は常に、再マージが発生します。

- SELECT 句が、集計関数に含まれていない列名、または GROUP BY 句で指定されていない他の列名を参照しています。
- ORDER BY 句が、SELECT 句で参照されていない列名を参照しています。

再マージクエリを認識するもう 1 つの方法は、PROC SQL が次のメッセージをログに書き込むことです: NOTE: The query requires remerging summary statistics back with the original data.

GROUP BY クエリでは、一連のグループ化に基づいてデータから極端な行のみを返したい場合、再マージは便利な機能です。たとえば、1 つ以上の地域で最も近い(または最も遠い)アイテム、1 つ以上のグループで最年長(または最年少)の人物、または 1 つ以上のグループで最も人気のある(または最も人気のない)アイテムのみを 説明する行がある場合があります。

再マージを使用する PROC SQL クエリの例を次に示します。クエリは、Payroll テーブル内の各性別の最年長従業員を識別します。クエリの極端な条件は、最も早い誕生日です。グループ化基準は性別です。

```
proc sql;
  title 'Oldest Employee of Each Gender';
  select *
    from payroll
    group by gender
    having birth=min(birth);
quit;
```

このクエリは、集計関数にも GROUP BY ステートメントにも含まれていない列名を (SELECT *ステートメントによって)参照することに注意してください。

図 10.1 テーブル Payroll に対する PROC SQL 再マージクエリの結果

IdNumber	Gender	Jobcode	Salary	Birth	Hired
1935	F	NA2	51081	28MAR54	16OCT81
1118	M	PT3	11379	16JAN44	18DEC80

出力テーブルには、Payroll テーブルの最年長の女性と最年長の男性について、テーブル Payroll のすべての列の値が含まれます。最年長の女性は 1954 年 3 月 28 日に生まれました。最年長の男性は 1944 年 1 月 16 日に生まれました。

FedSQL を使用して GROUP BY 再マージ要求を作成するには、次の操作を行います。

1 クエリの極値とグループを識別します。

2 サブクエリの結果に基づいて行の残りを選択し、GROUP BY 句で再マージが処理されることを意図していた要件を逆にします。

前述の例を FedSQL で書き換えるには、次の操作を行います。

1 テーブル Payroll で最も早い生年月日を性別ごとに識別します。

```
proc fedsql;
  title 'Earliest Birthdate by Gender';
  select Gender, min(Birth) from payroll
  group by Gender;
quit;
```

これがサブクエリです。

図 10.2 性別ごとに最も早い生年月日を識別するための FedSQL クエリの結果

Gender	MIN
F	28MAR1954
M	16JAN1944

2 サブクエリを指定し、サブクエリによって返された行の元のデータから他の列値を取得します。

```
proc fedsql;
  title 'Oldest Employee of Each Gender';
  select 2
   p.IdNumber, p.Gender, p.Jobcode, p.Salary, p.Birth, p.Hired
   from
   (
    select Gender, min(Birth) as min_birth 1
    from payroll
    group by Gender
    ) as t
  inner join payroll as p on p.Gender=t.gender and p.Birth=t.min_birth; 3
quit;
```

1 サブクエリが最初に実行されます。統計分析を実行し、グループ化を作成します。列 Birth は分析列です。列 Gender はグループ化列です。

2 外部クエリは、出力に含める列の名前を明示的に指定します。

3 グループ化列である Gender は、グループ化列として指定されるだけでなく、内部クエリ、外部クエリ、および結合条件としても含まれます。分析列 Birth は、外部クエリで指定され、結合条件としても指定されています。

クエリの結果は次のとおりです。FedSQL クエリは、PROC SQL 再マージクエリと同じ結果を返します。

図 10.3 FedSQL サブクエリと自己結合の結果

IdNumber	Gender	Jobcode	Salary	Birth	Hired
1935	F	NA2	51081	28MAR54	16OCT81
1118	M	PT3	11379	16JAN44	18DEC80

2 つのテーブルを含む再マージクエリの例を次に示します。Stores テーブルには、特定の地域の店舗に関する情報が含まれています。Houses テーブルには、同じ地域の住宅に関する情報が含まれています。PROC SQL クエリは、SELECT ステートメントから計算列を HAVING 句のグループ化条件として使用して、各住宅に最も近い店舗を識別します。

```
proc sql;
  title 'Each House and the Closest Store';
  select house, store, sqrt((abs(s.x-h.x)**2)+(abs(h.y-s.y)**2)) as dist
  from stores s, houses h
  group by house
  having dist=min(dist);
quit;
```

図 10.4 Stores テーブルと Houses テーブルに対する PROC SQL 再マージクエリの結果

House	Store	dist
house1	store1	4
house2	store2	2
house2	store3	2
house3	store3	2.236068
house4	store4	2

FedSQL でクエリを記述するには、次の操作を行います。

1 サブクエリを作成します。このクエリでは、極端な条件は最寄りの店舗までの距離です。グループ化は住宅ごとです。次のコードは、住宅と店舗の間の距離を計算し、Min_Dist 列を定義して結果を保存し、結果を住宅別にグループ化します。

```
proc fedsql;
  title 'House and Minimum Distance';
  select house, min(sqrt((abs(s.x-h.x)**2)+(abs(h.y-s.y)**2)))
    as min_dist
  from stores s, houses h
  group by house;
```

```
quit;
```

図 10.5 Houses テーブルと Stores テーブルに対する FedSQL サブクエリの結果

House	MIN_DIST
house1	4
house2	2
house3	2.236068
house4	2

2 サブクエリを実行し、サブクエリによって返された行の 2 つのテーブルから他の列値を取得します。

```
proc fedsql;
title 'Each House and the Closest Store';
select 2
   h.house, s.store, t.min_dist as "Dist"
from
 houses h
cross join
 stores s
inner join
 /* minimum distance from each house to any store */
 ( 1
   select house, min(sqrt((abs(s.x-h.x)**2)+(abs(h.y-s.y)**2)))
     as min_dist
   from stores s, houses h
   group by house 3
 ) as t
on h.house = t.house
where
 sqrt((abs(s.x-h.x)**2)+(abs(h.y-s.y)**2)) = t.min_dist;
quit;
```

1 ここでも、サブクエリが最初に実行されます。統計分析を実行し、グループ化を作成します。計算列 Min_Dist は分析列です。House はグループ化列です。

2 外部クエリは、Houses テーブルの House 列、Stores テーブルの Store 列、およびサブクエリの Min_Dist 列を表示するように指定しています。

3 グループ化列 House は、GROUP BY ステートメントに加えて、内部クエリ、外部クエリ、および結合条件として指定されます。分析列 Min_Dist は、外部クエリと WHERE 句で指定されます。WHERE 句は、距離計算の要件を満たす行(各住宅に最も近い店舗を識別する行)の値のみを取得するように指定します。

図 10.6　Stores テーブルと Houses テーブルに対する FedSQL サブクエリと結合の結果

House	Store	dist
house1	store1	4
house2	store2	2
house2	store3	2
house3	store3	2.236068
house4	store4	2

CALCULATED キーワードを指定する PROC SQL クエリを書き換える

注: このトピックの情報は、SAS エンジンおよび CAS での FedSQL の使用に適用されます。

FedSQL は、クエリ式で CALCULATED キーワードをサポートしていません。CALCULATED キーワードは ANSI の概念ではありません。CALCULATED キーワードは、インラインビューに似ています。ただし、列の一時的なエイリアスを定義するかわりに、キーワードは前の SQL 式へのショートカットを定義します。これを使用して、計算式に関連付けられている列エイリアスを参照できます。EmpSales テーブルのデータから四半期の売上を計算するときに CALCULATED キーワードを指定する次の例について考えてみます。

```
proc sql;
  title 'Total First Quarter Sales';
  select sum(January)  as JanTotal,
       sum(February) as FebTotal,
       sum(March)    as MarTotal,
       sum(calculated JanTotal, calculated FebTotal,
calculated MarTotal) as GrandTotal format=dollar10.
from EmpSales;
quit;
```

前述のクエリでは、次のように CALCULATED キーワードを使用して前のステートメントを参照しています。

- CALCULATED キーワードの最初のインスタンスは式 select sum(January)を参照する
- 2 番目のインスタンスは式 select sum(February)を参照する
- 3 番目のインスタンスは式 select sum(March)を参照する
- sum(January) + sum(February) + sum(March) = GrandTotal の計算を可能にする

図 10.7 CALCULATED キーワードを使用した SQL クエリの結果

JanTotal	FebTotal	MarTotal	GrandTotal
6800	7500	8100	$22,400

FedSQL を使用して前述のクエリを記述するには、計算式を繰り返します。

```
proc fedsql;
  title 'Total First Quarter Sales';
  select sum(January)  as JanTotal,
         sum(February) as FebTotal,
         sum(March)    as MarTotal,
         put(sum(January) + sum(February) + sum(March), dollar10.)
           as GrandTotal
  from EmpSales;
quit;
```

JANTOTAL	FEBTOTAL	MARTOTAL	GRANDTOTAL
6800	7500	8100	$22,400

ただし、FedSQL 要求を記述するよりよい方法は、サブクエリとして使用することです。

```
proc fedsql;
  select
    t.JanTotal,
    t.FebTotal,
    t.MarTotal,
    put(sum(t.JanTotal, t.FebTotal, t.MarTotal), dollar10.)
      as "GrandTotal"
    from
      (select sum(January) as "JanTotal",
        sum(February) as "FebTotal",
        sum(March) as "MarTotal"
        from EmpSales)
      as t;
quit;
```

要求をサブクエリとして記述する場合、式を繰り返すかわりに列エイリアスを使用して各式を参照できます。サブクエリを使用するとより効率的で、SQL 式が複雑な場合にも安全に使用できます。

SQL 式で複数の引数を指定する

注: このトピックの情報は、SAS ライブラリおよび CAS での FedSQL の使用に適用されます。

PROC SQL の CASE 式の WHEN 引数に複数の条件が指定されている場合、条件は暗黙の AND 演算子で連結されます。次の PROC SQL の例について考えてみます。こ

の例では、WorldCityCoords (1276 ページ)テーブルをクエリし、世界の都市の気候帯を記述子でラベル付けしています。

```
proc sql outobs=12;
  title 'Climate Zones of World Cities';
  select City, Country, Latitude,
    case
      when Latitude gt 67 then 'North Frigid'
      when 67 ge Latitude ge 23 then 'North Temperate'
      when 23 gt Latitude gt -23 then 'Torrid'
      when -23 ge Latitude ge -67 then 'South Temperate'
      else 'South Frigid'
    end as ClimateZone
    from worldcitycoords
    order by City;
quit;
```

City	Country	Latitude	ClimateZone
Abadan	Iran	30	North Temperate
Amman	Jordan	32	North Temperate
Asmara	Ethiopia	15	Torrid
Baghdad	Iraq	33	North Temperate
Brazzaville	Congo	-4	Torrid
Brisbane	Australia	-27	South Temperate
Bucharest	Romania	44	North Temperate
Buenos Aires	Argentina	-34	South Temperate
Cairo	Egypt	30	North Temperate
Cali	Colombia	3	Torrid
Cape Town	South Africa	-34	South Temperate
Copenhagen	Denmark	56	North Temperate

PROC SQL は、次に示すように、コードの強調表示されたセクションを処理します。

- 67 が Latitude より大きいか等しく(かつ) Latitude が 23 より大きいか等しい場合は、'North Temperate'を指定します。
- 23 が Latitude より大きく(かつ) Latitude が-23 より大きい場合は、'Torrid'を指定します。
- -23 が Latitude より大きいか等しく(かつ) Latitude が-67 より大きいか等しい場合は、'South Temperate'を指定します。
- それ以外の場合は、"South Frigid"を指定します。

対照的に、FedSQL の CASE 式の WHEN 引数に複数の条件が指定されている場合、コードの強調表示されたセクションは次のように処理されます。

- 条件は等しくなければなりません。
- 演算子は、条件間で行われる比較を指定する必要があります。

次に例を示します。

```
proc fedsql;
```

```
  title 'Climate Zones of World Cities';
  select City, Country, Latitude,
    case
      when Latitude > 67 then 'North Frigid'
      when (67 >= Latitude) and (Latitude >= 23) then 'North Temperate'
      when (23 > Latitude) and (Latitude > -23) then 'Torrid'
      when (-23 >= Latitude) and (Latitude >= -67) then 'South Temperate'
      else 'South Frigid'
    end as "ClimateZone"
  from worldcitycoords
  order by City limit 12;
quit;
```

PROC SQL のニーモニック演算子は、FedSQL で実行するために記号表現に変更されることに注意してください。FedSQL では、AND、OR、および NOT を除くすべての比較演算子に記号を使用する必要があります。

明示的パススルーを使用する

SQL プロシジャと FedSQL の両方を使用すると、明示的 SQL パススルーを使用して、DBMS 固有の SQL ステートメントを DBMS にできます。非クエリ DBMS SQL ステートメントは、EXECUTE ステートメントを使用して DBMS にサブミットされます。クエリは、SELECT ステートメントの FROM 句の CONNECTION TO コンポーネントでサブミットできます。

PROC SQL では、明示的パススルーを要求するステートメントは、CONNECT および DISCONNECT ステートメントと組み合わせて使用されます。次に例を示します。

```
proc sql;
  connect to oracle as ora2 (user=user-id password=password);
  execute (create table new (a NUMBER, b TIMESTAMP, c VARCHAR2(15)))
    by ora2;
  execute(insert into new values(12345, date'2003-11-22', 'John Doe'))
    by ora2;
  select * from connection to ora2 (select a, b, c from new);
  disconnect from ora2;
quit;
```

FedSQL では、CONNECT および DISCONNECT ステートメントを指定する必要はありません。データソース接続は、libref (または CAS の caslib)で確立されます。次に例を示します。

```
libname ora2 oracle ...;
proc fedsql;
  execute (create table new2 (a NUMBER, b TIMESTAMP, c VARCHAR2(15)))
    by ora2;
  execute(insert into new2 values(12345, date'2003-11-22', 'John Doe'))
    by ora2;
  select * from connection to ora2(select a, b, c from new2);
quit;
```

注: 現在、CAS の FedSQL では、SELECT from CONNECTION TO 明示的パススルー構文のみがサポートされています。

ただし、FedSQL では、明示的パススルーを使用してネイティブステートメントをサブミットするよりも、FedSQL ステートメントを使用することで、DBMS 内のテーブルをより簡単に作成、データ入力、およびクエリできます。FedSQL はベンダーニュートラルな SQL ダイアレクトであり、データソースに固有の SQL ダイアレクトでクエリをサブミットすることなく、さまざまなデータソースからのデータにアクセスすることができます。DBMS データを書き込むための ANSI 準拠のデータ型を提供し、DBMS データを読み取るときには、非 ANSI データ型を ANSI 型に内部的にマッピングします。SAS とサードパーティのデータ型マッピングの詳細については、付録 2, “データ型リファレンス” (1199 ページ)を参照してください。

たとえば、次の FedSQL ステートメントは、FedSQL でサポートされている任意の DBMS にテーブルを作成できます。

```
proc fedsql;
  create table ora2.new3(a decimal(10,2), b timestamp, c varchar(15));
  insert into ora2.new3 values (12345, timestamp(2003-11-22 02:25:00),
    'John Doe');
  select * from ora2.new3;
quit;
```

データ型の変換は、FedSQL によって内部的に処理されます。さらに、FedSQL はデフォルトで、サードパーティデータソースの完全および部分的な暗黙的パススルーをサポートします。

注: 部分的な FedSQL パススルーは、CAS の FedSQL ではまだ使用できません。

FedSQL で SAS マクロを使用する

注: このトピックの情報は、SAS エンジンでの FedSQL の使用に適用されます。

PROC SQL で使用されるマクロは、FedSQL セマンティクスを満たすようにマクロ定義を変更することにより、PROC FEDSQL から実行できます。PROC FEDSQL で使用するために変更された PROC SQL ドキュメントからの次の例について考えてみましょう。例では、テーブル Survey には、食事と運動に関するアンケートのデータが含まれています。SAS では、欠損値に対して特別な表記を使用できます。テーブル Survey の EDUC 列で、次の点に注意してください。

- .x 表記は、回答者が無効な回答をしたことを示します。
- .n は、回答者が質問に回答しなかったことを示します。
- 欠損値として使用されるピリオド(.)は、データ入力エラーを示します。

マクロは、各値を持つ行の数をカウントし、テーブルに情報を返します。

PROC SQL コードは次のとおりです。

```
%macro countm(col);
   count(&col) "Valid Responses for &col",
nmiss(&col) "Missing or NOT VALID Responses for &col",
count(case
        when &col=.n  then "count me"
```

```
        end) "Coded as NO ANSWER for &col",
  count(case
        when &col=.x  then "count me"
        end) "Coded as NOT VALID answers for &col",
  count(case
        when &col=.  then "count me"
        end) "Data Entry Errors for &col"
%mend;

proc sql;
  title 'Counts for Each Type of Missing Response';
  select count(*)  "Total No. of Rows",
    %countm(educ)
    from survey;
quit;
```

FEDSQL 用に変更されたマクロ定義とクエリは次のとおりです。

```
%macro countm_fedsql(col);
 count(&col) as 1 "Valid Responses for &col",
 nmiss(&col) as "Missing or NOT VALID Responses for &col",
 count(case
        when &col=.n  then 'count me' 2
        end) as "Coded as NO ANSWER for &col",
 count(case
        when &col=.x  then 'count me'
        end) as "Coded as NOT VALID answers for &col",
 count(case
        when &col=.  then 'count me'
        end) as "Data Entry Errors for &col"
%mend;

proc fedsql;
  title 'Counts for Each Type of Missing Response';
  select count(*) as "Total No. of Rows", 3
    %countm_fedsql(educ)
    from survey;
quit;
```

1 FedSQL では、列式とそのエイリアスの間にキーワード AS が必要です。大文字と小文字を区別するには、エイリアスを二重引用符で囲む必要があります。

2 FedSQL では、文字定数'count me'を二重引用符ではなく単一引用符で囲む必要があります。

3 FedSQL は、ラベルを適用するかわりに、置換列名を列エイリアスとして指定します。

暗黙的 SQL パススルー: 比較

PROC SQL と FedSQL はどちらも、暗黙的 SQL パススルーを介して処理する要求を DBMS に渡すことができます。

PROC SQL は、IPASSTHRU ステートメントオプションが設定されている場合、暗黙的パススルーをサポートします。IPASSTHRU はデフォルトで設定されています。PROC SQL ステートメントオプション、SAS システムオプション、および SAS/ACCESS LIBNAME ステートメントオプションを使用して、次のようなパススループロセスの側面を制御できます。

- 暗黙的 SQL パススルーが失敗したときに PROC SQL がクエリを再試行する回数(SQLIPONEATTEMPT ステートメントオプションおよびシステムオプション)
- 出力形式とオブザベーションの処理方法(SQLREDUCEPUTVALUES=および SQLREDUCEPUTOBS= SQL ステートメントオプションとシステムオプション)
- 処理のために DBMS に渡される関数(SQL_FUNCTIONS=)
- SAS/ACCESS エンジンが CREATE TABLE、INSERT、および DELETE FROM TABLE ステートメント要求に対してデータベース固有のクエリを生成するかどうか、または PROC SQL ステートメントが使用されるかどうか(DBIDIRECTEXEC=システムオプション)

FedSQL クエリは、次のルールに基づいて暗黙的パススルーについて自動的に評価されます。

- FedSQL は、クエリに次のいずれかが含まれていない限り、基になるデータソースに単一ソースクエリを渡します。
  - PUT などの SAS 固有の関数
  - SKEWNESS、STUDENTS_T、NMISS、KURTOSIS、CSS、USS、PROBT などの特定の集計統計量
  - SIN、COS、ATAN、TAN などの数学関数
  - データソースがその領域で ANSI 準拠でない場合、IP を妨げる可能性がある ANSI 準拠の FedSQL 構文

  完全なクエリの単一ソースパススルーが不可能な場合、述語(WHERE 句など)が渡されて、ローカル処理のために FedSQL に転送する必要がある行がサブセット化されます。
- FedSQL は、複数のデータソースを含むクエリを処理するときに、クエリを複数のクエリに分割し、これらの個々のクエリをそれぞれのドライバーに渡すことで、暗黙的パススルーを実現します。クエリは、単一ソースの暗黙的パススルーの要件を満たす必要があります。さらに、複数ソース IP には次の追加の制限があります。
  - 1 つのカンマ結合に最大 10 個のテーブルを含めることができます。
  - セット操作(UNION、INTERSECT、および EXCEPT など)の各側のテーブルは、複数ソース IP が正しく実行されるように同じデータソースを共有する必要があります。次に例を示します。

```
proc fedsql;
  /* join of two Teradata tables */
  select t.col1, t.col2, t1.col1, t1.col4
    from tera.employees t,  tera.employees t1
    where t.col1=t1.col1
  Union
  /* join of two Oracle tables */
  select b.col1, b.col2, b1.col1, b1.col4
    from ora.test1 b, ora.test2 b1
    where b.col1=b1.col1;
```

```
quit;
```

セット操作のいずれかの側のテーブルが複数のデータソースからのものである場合、部分的な暗黙的パススルーは発生しない可能性があります。

- クエリに相関サブクエリが含まれている場合、複数ソース IP は試行されません。
- 暗黙的パススルーを使用してマージおよび結合するには、データベーステーブルが同じカタログ内にある必要があります。異なるカタログに存在するデータベーステーブルのマージと結合は、クライアントで処理されます。複数のスキーマ(ただしカタログは 1 つだけ)への参照を含むクエリは、データソースに渡すことができます。

注: CAS では、FedSQL 暗黙的パススルーも複数ソースの FedSQL 暗黙的パススルーもサポートされていません。

FedSQL は、暗黙的 SQL パススルーがいつ使用されるかを示すメッセージをログに書き込みます。

3 部

付録

付録 1
トラブルシューティング: FedSQL エラーの解決 1195

付録 2
データ型リファレンス 1199

付録 3
FedSQL と DS2 の使用 1253

付録 4
例で使用されているテーブル 1255

付録 5
DICTIONARY テーブルの説明 1279

付録 6
FedSQL と ANSI 規格 1293

付録 7
使用上の注意 1297

付録 8
ICU ライセンス 1299

付録 1

トラブルシューティング: FedSQL エラーの解決

FedSQL エラーの解決 **1195**
エラーの種類 1195
FedSQL エラーの解決方法 1196
SAS ライブラリにサブミットされた FedSQL 要求に対するテクニカルサポートを提供するための追加情報 1197

FedSQL エラーの解決

エラーの種類

FedSQL は、要求を完了できない場合、SAS ログにエラーを書き込みます。ほとんどのエラーは、次のように分類できます。

使用法エラー
: 不正または無効な要求が原因で、使用法エラーが発生します。FedSQL の使用法エラーは通常、構文、セマンティック、または論理エラーが原因です。コンパイル時に構文エラーとセマンティックエラーが検出され、FedSQL ステートメントのコンパイルが妨げられます。正常にコンパイルされた FedSQL ステートメントには、実行時に表面化する論理エラーが含まれている可能性があります。

リソースエラー
: リソースサービス要求を満たせない場合、リソースエラーが発生します。さまざまな条件により、リソースサービス要求が失敗します。たとえば、リソースにアクセスできない、リソースのしきい値に達している、リソースにアクセスする権限がないなどの可能性があります。

FedSQL エラーの解決方法

エラーを解決するには、次の操作を行います。

1 SAS ログでエラーメッセージを確認します。これらのメッセージは、問題の原因を理解したり、エラーの解決に役立つアイデアを提供したりするのに役立つ場合があります。

問題が何であるかを示す例を次に示します。この例では、使用法エラーが発生しました。メッセージは、コンパイルフェーズで問題が発生し、構文エラーまたはアクセス違反のいずれかであったことを示しています。DRIP という単語で問題が発生しました。この例では、コードを確認したところ、テーブルオプションが必要な場所でこの単語が検出されたことが示されました。ドキュメントを簡単に確認すると、FedSQL が DRIP テーブルオプションをサポートしていないことがわかります。DROP=テーブルオプションはサポートしています。

```
ERROR: Compilation error.
ERROR: Syntax error or access violation DRIP
```

2 SAS ログで、まず一番最初のエラーメッセージに注目します。場合によっては、1 つの問題が連鎖的な失敗につながり、最初のエラーを解決すると、後続のエラーも解決されることがあります。

3 エラーメッセージがリソースエラーを示している場合は、システム管理者に問い合わせて、システムリソースを監視してください。システムリソースの例には、メモリ使用量とディスク容量が含まれます。システムリソースが少ない場合は、リソースを増やす方法を探すか、アプリケーションがそれらをより効率的に使用できる方法を検討してください。

リソースエラーを示す例を次に示します。このメッセージは、メモリのしきい値に達したことを示しています。

```
ERROR: Memory allocation failed.
```

4 前の 3 つのステップでもエラーを解決できない場合は、SAS テクニカルサポートにお問い合わせください。提供する必要がある情報のリストを次に示します: Four tips to remember when you contact SAS Technical Support。

SAS ログに内部エラーコードを含むエラーメッセージが含まれている場合は、そのコードを SAS テクニカルサポートの担当者に提供してください。内部エラーコードは、SAS が問題の原因をより迅速に特定するのに役立つ 16 進数です。内部エラーコードを含むエラーメッセージの例を次に示します。

```
ERROR: FedSQL internal error code 0xECCC8C8BF7C049FD. Refer to FedSQL
 documentation for resolving errors.
```

SAS ライブラリにサブミットされた FedSQL 要求に対するテクニカルサポートを提供するための追加情報

FedSQL は、SAS テクニカルサポートの追加情報を収集するために使用できる_DIAG オプションをサポートしています。内部エラーが発生した場合、SAS テクニカルサポートの担当者がこの情報を要求します。_DIAG オプションによって返されるメッセージは FedSQL 開発者向けであり、意味をなさないように見えるでしょう。

失敗した FedSQL 要求に_DIAG オプションを追加します。次のコードは、オプションを FedSQL プロシジャに追加する方法を示しています。プロシジャオプションとして_DIAG オプションを指定しています。

```
proc fedsql _diag;
...FedSQL statements...
quit;
```

テクニカルサポートに提供すると役立つその他の情報には、次のようなものがあります。

- 失敗した要求に含まれるテーブルの数
- 各入力テーブルの列と行の数
- 使用されるデータ型
- 要求に含まれるデータソース
- 1 つまたは複数のデータソース接続を確立するために使用される LIBNAME ステートメントまたは接続文字列

付録 2

データ型リファレンス

SAS データセットのデータ型 1200
SPD Engine データセットのデータ型 1201
SPD Server テーブルのデータ型 1202
Amazon Redshift のデータ型 1204
UNIX および PC ホストでの DB2 のデータ型 1206
Google BigQuery のデータ型 1207
Greenplum のデータ型 1209
Hive のデータ型 1210
Impala のデータ型 1213
JDBC のデータ型 1214
MDS のデータ型 1216
Microsoft SQL Server のデータ型 1218
MongoDB のデータ型 1220
MySQL のデータ型 1222
Netezza のデータ型 1224
ODBC のデータ型 1225
Oracle のデータ型 1227
PostgreSQL のデータ型 1229
Salesforce から読み取るためのデータ型 1231
Salesforce に書き込むためのデータ型 1233
SAP のデータ型 1235
SAP HANA のデータ型 1237
SAP IQ のデータ型 1238
SingleStore のデータ型 1240
Snowflake のデータ型 1242

Spark のデータ型 1244
Teradata のデータ型 1246
Vertica のデータ型 1248
Yellowbrick のデータ型 1251

SAS データセットのデータ型

次の表に、SAS データセットのデータ型サポートを示します。

BINARY および VARBINARY データ型は、データ定義ではサポートされていません。

注: この表の情報は、CAS で処理されるデータには適用されません。CAS にロードされたデータは、CAS データ型に変換されます。CAS データ型の変換については、*SAS Cloud Analytic Services: ユーザーガイド*の SAS Data Connector for SAS Data Sets のドキュメントを参照してください。

表 A7.1　FedSQL データ型と SAS データセットで使用されるデータ型のマッピング

データ型定義キーワード [1]	SAS データセットのデータ型	説明	返されるデータ型
BIGINT[2]	DOUBLE	64 ビットの倍精度浮動小数点数。 **注:** 精度が低下する可能性があります。	DOUBLE
CHAR(*n*)	CHAR(*n*)	固定長の文字列。 **注:** ANSI SQL null 値を含めることはできません。	CHAR(*n*)
DATE[3]	DOUBLE	64 ビットの倍精度浮動小数点数。デフォルトでは、DATE9 SAS 出力形式が適用されます。	DOUBLE
DECIMAL|NUMERIC(*p*,*s*)[2]	DOUBLE	64 ビットの倍精度浮動小数点数。	DOUBLE
DOUBLE[2]	DOUBLE	64 ビットの倍精度浮動小数点数。	DOUBLE
FLOAT(*p*)[2]	DOUBLE	64 ビットの倍精度浮動小数点数。	DOUBLE
INTEGER[2]	DOUBLE	64 ビットの倍精度浮動小数点数。	DOUBLE
NCHAR(*n*)	CHAR(*n*)	固定長の文字列。デフォルトでは、エンコーディングを Unicode UTF-8 に設定します。[4]	CHAR(*n*)

データ型定義キーワード [1]	SAS データセットのデータ型	説明	返されるデータ型
NVARCHAR(*n*)	CHAR(*n*)	固定長の文字列。デフォルトでは、エンコーディングを Unicode UTF-8 に設定します。[4]	CHAR(*n*)
REAL[2]	DOUBLE	64 ビットの倍精度浮動小数点数。	DOUBLE
SMALLINT[2]	DOUBLE	64 ビットの倍精度浮動小数点数。	DOUBLE
TIME(*p*)[3]	DOUBLE	64 ビットの倍精度浮動小数点数。デフォルトでは、TIME8 SAS 出力形式が適用されます。	DOUBLE
TIMESTAMP(*p*)[3]	DOUBLE	64 ビットの倍精度浮動小数点数。デフォルトでは、DATETIME19.2 SAS 出力形式が適用されます。	DOUBLE
TINYINT[2]	DOUBLE	64 ビットの倍精度浮動小数点数。	DOUBLE
VARCHAR(*n*)	CHAR(*n*)	固定長の文字列。 **注:** ANSI SQL null 値を含めることはできません。	CHAR(*n*)

1 データ型のマッピング方法を制御する CT_PRESERVE=接続引数は、データ型を定義できるかどうかに影響する可能性があります。値 FORCE (デフォルト)および FORCE_COL_SIZE は、データ型を定義できるかどうかには影響しません。要求されたデータ型がデータソースにネイティブでない場合、または指定された精度またはスケールがデータソースの範囲内にない場合、値 STRICT および SAFE はエラーになる可能性があります。

2 日付と時間の SAS 出力形式を数値データ型に適用しないでください。日付と時間の値には、DATE、TIME、または TIMESTAMP データ型を使用します。

3 値は倍精度浮動小数点数として保存されるため、算術式で値を使用できます。

4 UTF-8 は MBCS エンコーディングです。動作環境に応じて、UTF-8 文字の幅は 1 バイトから 4 バイトまでさまざまです。保存するマルチバイト文字の最大数である *n* の値に、動作環境の最大長を掛けます。UTF-8 から Wlatin2 などにトランスコーディングする場合、可変長(バイト単位)では値を保持するのに十分でない可能性があり、その結果、文字データが切り捨てられることに注意してください。

SPD Engine データセットのデータ型

次の表に、SPD Engine データセットのデータ型サポートを示します。

BINARY、DECIMAL、NUMERIC、NCHAR、NVARCHAR、VARBINARY データ型は、データ型定義ではサポートされていません。

注: この表の情報は、CAS で処理されるデータには適用されません。CAS にロードされたデータは、CAS データ型に変換されます。CAS データ型の変換については、*SAS Cloud Analytic Services: ユーザーガイド*の SAS Data Connector for SPD Engine Files のドキュメントを参照してください。

表 A7.2　FedSQL データ型と SPD Engine データセットで使用されるデータ型のマッピング

データ型定義キーワード	SPD データセットのデータ型	説明	返されるデータ型
BIGINT[1]	DOUBLE	64 ビットの倍精度浮動小数点数。 **注:** 精度が低下する可能性があります。	DOUBLE
CHAR(*n*)	CHAR(*n*)	固定長の文字列。 **注:** ANSI SQL null 値を含めることはできません。	CHAR(*n*)
DATE[2]	DOUBLE	64 ビットの倍精度浮動小数点数。デフォルトでは、DATE9 SAS 出力形式が適用されます。	DOUBLE
DOUBLE[1]	DOUBLE	64 ビットの倍精度浮動小数点数。	DOUBLE
FLOAT(*p*)[1]	DOUBLE	64 ビットの倍精度浮動小数点数。	DOUBLE
INTEGER[1]	DOUBLE	64 ビットの倍精度浮動小数点数。	DOUBLE
REAL[1]	DOUBLE	64 ビットの倍精度浮動小数点数。	DOUBLE
SMALLINT[1]	DOUBLE	64 ビットの倍精度浮動小数点数。	DOUBLE
TIME(*p*)[2]	DOUBLE	64 ビットの倍精度浮動小数点数。デフォルトでは、TIME8 SAS 出力形式が適用されます。	DOUBLE
TIMESTAMP(*p*)[2]	DOUBLE	64 ビットの倍精度浮動小数点数。デフォルトでは、DATETIME19.2 SAS 出力形式が適用されます。	DOUBLE
TINYINT[1]	DOUBLE	64 ビットの倍精度浮動小数点数。	DOUBLE
VARCHAR(*n*)	CHAR(*n*)	固定長の文字列。 **注:** ANSI SQL null 値を含めることはできません。	CHAR(*n*)

1 日付と時間の SAS 出力形式を数値データ型に適用しないでください。日付と時間の値には、DATE、TIME、または TIMESTAMP データ型を使用します。
2 値は倍精度浮動小数点数として保存されるため、算術式で値を使用できます。

SPD Server テーブルのデータ型

次の表に、SPD Server テーブルのデータ型サポートを示します。

BINARY、DECIMAL、NUMERIC、NCHAR、NVARCHAR、VARBINARY データ型は、データ型定義ではサポートされていません。

注: このデータソースは CAS サーバーではサポートされていません。

表 A7.3 FedSQL データ型と SPD Server テーブルで使用されるデータ型のマッピング

データ型定義キーワード	SPD データセットのデータ型	説明	返されるデータ型
BIGINT[1]	DOUBLE	64 ビットの倍精度浮動小数点数。 **注:** 精度が低下する可能性があります。	DOUBLE
CHAR(*n*)	CHAR(*n*)	固定長の文字列。 **注:** ANSI SQL null 値を含めることはできません。	CHAR(*n*)
DATE[2]	DOUBLE	64 ビットの倍精度浮動小数点数。デフォルトでは、DATE9 SAS 出力形式が適用されます。	DOUBLE
DOUBLE[1]	DOUBLE	64 ビットの倍精度浮動小数点数。	DOUBLE
FLOAT(*p*)[1]	DOUBLE	64 ビットの倍精度浮動小数点数。	DOUBLE
INTEGER[1]	DOUBLE	64 ビットの倍精度浮動小数点数。	DOUBLE
REAL[1]	DOUBLE	64 ビットの倍精度浮動小数点数。	DOUBLE
SMALLINT[1]	DOUBLE	64 ビットの倍精度浮動小数点数。	DOUBLE
TIME(*p*)[2]	DOUBLE	64 ビットの倍精度浮動小数点数。デフォルトでは、TIME8 SAS 出力形式が適用されます。	DOUBLE
TIMESTAMP(*p*)[2]	DOUBLE	64 ビットの倍精度浮動小数点数。デ	DOUBLE

データ型定義キーワード	SPD データセットのデータ型	説明	返されるデータ型
		フォルトでは、DATETIME19.2 SAS 出力形式が適用されます。	
TINYINT[1]	DOUBLE	64 ビットの倍精度浮動小数点数。	DOUBLE
VARCHAR(*n*)	CHAR(*n*)	固定長の文字列。 **注:** ANSI SQL null 値を含めることはできません。	CHAR(*n*)

1 日付と時間の SAS 出力形式を数値データ型に適用しないでください。日付と時間の値には、DATE、TIME、または TIMESTAMP データ型を使用します。
2 値は倍精度浮動小数点数として保存されるため、算術式で値を使用できます。

Amazon Redshift のデータ型

次の表に、Amazon Redshift データベースのデータ型サポートを示します。

BINARY、TINYINT、および VARBINARY データ型は、データ型定義ではサポートされていません。

Redshift データ型に関するデータソース固有の情報については、Amazon Redshift データベースのドキュメントを参照してください。

注: この表の情報は、CAS で処理されるデータには適用されません。CAS にロードされたデータは、CAS データ型に変換されます。CAS データ型の変換については、*SAS Cloud Analytic Services: ユーザーガイド*の SAS Data Connector for Amazon Redshift のドキュメントを参照してください。

表 A7.4 FedSQL データ型と Amazon Redshift で使用されるデータ型のマッピング

データ型定義キーワード [1]	Amazon Redshift データ型	説明	返されるデータ型
BIGINT	BIGINT	符号付き 8 バイト整数。	BIGINT
[2]	BOOLEAN	論理ブール(true/false)値。	[2]

データ型定義キーワード [1]	Amazon Redshift データ型	説明	返されるデータ型
CHAR(*n*)	CHAR	固定長の文字列。	CHAR(*n*)
DATE	DATE	日付値。	DATE
DECIMAL|NUMERIC(*p*,*s*)	DECIMAL、NUMERIC	符号付き固定小数点 10 進数。	DECIMAL|NUMERIC(*p*,*s*)[4]
DOUBLE	DOUBLE PRECISION	符号付き倍精度浮動小数点数。	DOUBLE
FLOAT(*p*)	FLOAT	符号付きの単精度または倍精度の浮動小数点数。	FLOAT(*p*)
INTEGER	INTEGER	通常の符号付きの正確な整数。	INTEGER
NCHAR	NCHAR	固定長の Unicode 文字列。	CHAR
NVARCHAR	NVARCHAR	可変長の Unicode 文字列。	CHAR
REAL	REAL	符号付き単精度浮動小数点数。	REAL
SMALLINT	SMALLINT	小さな符号付きの正確な整数。	SMALLINT
TIME(*p*)	[3]	時間値。	TIMESTAMP
TIMESTAMP(*p*)	TIMESTAMP	日時値。	TIMESTAMP(*p*)
[2]	TIMESTAMPZ	タイムゾーン付き日時値。	[2]
VARCHAR(*n*)	VARCHAR	可変長の文字列。	CHAR

1 データ型のマッピング方法を制御する CT_PRESERVE=接続引数は、データ型を定義できるかどうかに影響する可能性があります。値 FORCE (デフォルト)および FORCE_COL_SIZE は、データ型を定義できるかどうかには影響しません。要求されたデータ型がデータソースにネイティブでない場合、または指定された精度またはスケールがデータソースの範囲内にない場合、値 STRICT および SAFE はエラーになる可能性があります。

2 Amazon Redshift データ型は定義できません。データが取得されると、ネイティブデータ型が同様のデータ型にマッピングされます。

3 Amazon Redshift は、TIME(*p*)データ型をサポートしていません。TIME 型の列を定義すると、かわりに TIMESTAMP 型の列が作成されます。

4 FedSQL では、DECIMAL データ型は、データソースに完全に渡すことができる要求でサポートされています。データソースに完全に渡すことができない要求の DECIMAL 列は、DOUBLE として処理されるため、精度に影響を与える可能性があります。詳細については、“データ型” (14 ページ)を参照してください。

UNIX および PC ホストでの DB2 のデータ型

次の表に、UNIX および PC ホストでの DB2 データベースのデータ型サポートを示します。

NCHAR、NVARCHAR、および TINYINT データ型は、データ型定義ではサポートされていません。

DB2 データベースのデータ型に関するデータソース固有の情報については、DB2 データベースのドキュメントを参照してください。

注: この表の情報は、CAS で処理されるデータには適用されません。CAS にロードされたデータは、CAS データ型に変換されます。CAS データ型の変換については、*SAS Cloud Analytic Services: ユーザーガイド*の SAS Data Connector for DB2 のドキュメントを参照してください。

表 A7.5 FedSQL データ型と UNIX および PC ホストでの DB2 で使用されるデータ型のマッピング

データ型定義キーワード [1]	DB2 データ型	説明	返されるデータ型
BIGINT	BIGINT	大きな符号付きの正確な整数。	BIGINT
BINARY(*n*)	CHAR(*n*) FOR BIT DATA	固定長のバイナリ文字列。	BINARY(*n*)
[2]	BLOB(*n*[K|M|G])	可変長のバイナリラージオブジェクト文字列。	[2]
CHAR(*n*)	CHAR(*n*)	固定長の文字列。	CHAR(*n*)
[2]	CLOB(*n*[K|M|G])	可変長文字のラージオブジェクト文字列。	[2]
DATE	DATE	日付値。	DATE
[2]	DBCLOB(*n*[K|M|G])	可変長のダブルバイト文字ラージオブジェクト。	[2]
DECIMAL|NUMERIC(*p*,*s*)	DECIMAL|NUMERIC(*p*,*s*)	符号付き固定小数点 10 進数。	DECIMAL|NUMERIC(*p*,*s*)[3]

データ型定義キーワード [1]	DB2 データ型	説明	返されるデータ型
DOUBLE	DOUBLE	符号付き倍精度浮動小数点数。	DOUBLE
FLOAT(*p*)	FLOAT(*p*)	符号付きの単精度または倍精度の浮動小数点数。	FLOAT(*p*)
[2]	GRAPHIC(*n*)	固定長のグラフィック文字列。	[2]
INTEGER	INTEGER	通常の符号付きの正確な整数。	INTEGER
[2]	LONG VARCHAR [FOR BIT DATA]	可変長文字またはバイナリ文字列。	[2]
[2]	LONG VARGRAPHIC(*n*)	可変長のグラフィック文字列。	[2]
REAL	REAL	符号付き単精度浮動小数点数。	REAL
SMALLINT	SMALLINT	小さな符号付きの正確な整数。	SMALLINT
TIME(*p*)	TIME(*p*)	時間値。	TIME(*p*)
TIMESTAMP(*p*)	TIMESTAMP(*p*)	日時値。	TIMESTAMP(*p*)
VARBINARY(*n*)	VARCHAR(*n*) FOR BIT DATA	可変長のバイナリ文字列。	VARBINARY(*n*)
VARCHAR(*n*)	VARCHAR(*n*)	可変長の文字列。	VARCHAR(*n*)
[2]	VARGRAPHIC(*n*)	可変長のグラフィック文字列。	[2]

1 データ型のマッピング方法を制御する CT_PRESERVE=接続引数は、データ型を定義できるかどうかに影響する可能性があります。値 FORCE (デフォルト)および FORCE_COL_SIZE は、データ型を定義できるかどうかには影響しません。要求されたデータ型がデータソースにネイティブでない場合、または指定された精度またはスケールがデータソースの範囲内にない場合、値 STRICT および SAFE はエラーになる可能性があります。

2 DB2 データ型は定義できません。データが取得されると、ネイティブデータ型が同様のデータ型にマッピングされます。

3 FedSQL では、DECIMAL データ型は、データソースに完全に渡すことができる要求でサポートされています。データソースに完全に渡すことができない要求の DECIMAL 列は、DOUBLE として処理されるため、精度に影響を与える可能性があります。詳細については、“データ型” (14 ページ)を参照してください。

Google BigQuery のデータ型

次の表に、Google BigQuery プロジェクトのデータ型サポートを示します。

Google BigQuery の ARRAY、STRUCT、および GEOGRAPHY データ型はサポートされていません。

Google BigQuery のデータ型に関するデータソース固有の情報については、Google BigQuery のドキュメントを参照してください。

注: この表の情報は、CAS で処理されるデータには適用されません。CAS にロードされたデータは、CAS データ型に変換されます。CAS データ型の変換については、*SAS Cloud Analytic Services: ユーザーガイド*の SAS Data Connector for Google BigQuery のドキュメントを参照してください。

表 A7.6 FedSQL データ型と Google BigQuery で使用されるデータ型のマッピング

データ型定義キーワード	BigQuery データ型	BigQuery の説明	返されるデータ型
BIGINT	INT64	大きな符号付きの正確な整数で、範囲は-9,223,372,036,854,775,808 から 9,223,372,036,854,775,807 までです。	BIGINT
[1]	BIGNUMERIC(*p,s*)	倍精度浮動小数点数。	DOUBLE[4, 5]
BINARY	BYTES	可変長のバイナリ文字列。	VARBINARY[2]
[1]	BOOL	True または False として表されるブール値(大文字と小文字を区別しない)。	[1]
CHAR(*n*)	STRING	可変長の Unicode 文字列。	VARCHAR[3]
DATE	DATE	0001-01-01 から 9999-12-31 までの日付値。	DATE
DOUBLE	FLOAT64	倍精度浮動小数点数。	DOUBLE
FLOAT	FLOAT64	倍精度浮動小数点数。	DOUBLE
INTEGER	INT64	通常の符号付きの正確な整数。	BIGINT
NCHAR	STRING	固定長の Unicode 文字列。	VARCHAR
NUMERIC(*p,s*)	NUMERIC(*p,s*)	倍精度浮動小数点数。	DOUBLE[4, 5]
NVARCHAR	STRING	可変長の Unicode 文字列。	VARCHAR
SMALLINT	INT64	小さな符号付きの正確な整数。	BIGINT
TIME(*p*)	TIME	00:00:00 から 23:59:59.999999 までの範囲の時間、分、および秒の時間値。	TIME(*p*)
[1]	TIMESTAMP	日時値。	[1]

データ型定義キーワード	BigQuery データ型	BigQuery の説明	返されるデータ型
TIMESTAMP(*p*)	DATETIME	タイムゾーンや夏時間などの規則に依存しない、マイクロ秒精度の日時値。サポートされている値は、0001-01-01 00:00:00 から 9999-12-31 23:59:59.999999 UTC です。	TIMESTAMP(p)
TINYINT	INT64	非常に小さな符号付きの正確な整数。	BIGINT
VARBINARY(*n*)	BYTES	可変長のバイナリ文字列。	VARBINARY[2]
VARCHAR(*n*)	STRING	可変長の Unicode 文字列。	VARCHAR[3]

1 BigQuery データ型は定義できません。データが取得されると、ネイティブデータ型が同様のデータ型にマッピングされます。

2 長さは、MAX_BINARY_LEN= LIBNAME オプションによって決定されます。MAX_BINARY_LEN=が設定されていない場合、デフォルトの長さは 2,000 バイトです。

3 MAX_CHAR_LEN= LIBNAME オプションまたは SCANSTRINGCOLUMNS テーブルオプションが設定されていない限り、デフォルトの長さは 16,384 文字です。

4 2024.09 以降、Google BigQuery NUMERIC(p,s)および BIGNUMERIC(p,s)列は DOUBLE として扱われ、BigQuery FLOAT64 型にマッピングされます。FETCH_NUMERIC_TYPE=テーブルオプションは、これらの型を標準の NUMERIC(p,s)型として扱うように要求できるように提供されています。

5 FETCH_NUMERIC_TYPE=NUMERIC が設定されている場合、NUMERIC(p,s)および BIGNUMERIC(p,s)型は、データソースに完全に渡すことができる要求でサポートされています。データソースに完全に渡すことができない要求の NUMERIC(p,s)および BIGNUMERIC(p,s)列は、DOUBLE として処理されます。詳細については、“データ型” (14 ページ)を参照してください。

Greenplum のデータ型

次の表に、Greenplum データベースのデータ型サポートを示します。

BINARY、NCHAR、NVARCHAR、TINYINT データ型は、データ型定義ではサポートされていません。

Greenplum データ型に関するデータソース固有の情報については、Greenplum データベースのドキュメントを参照してください。

注: この表の情報は、CAS で処理されるデータには適用されません。CAS にロードされたデータは、CAS データ型に変換されます。CAS データ型の変換については、*SAS Cloud Analytic Services: ユーザーガイド*の SAS Data Connector for Greenplum のドキュメントを参照してください。

表 A7.7　FedSQL データ型と Greenplum で使用されるデータ型のマッピング

データ型定義キーワード [1]	Greenplum データ型	説明	返されるデータ型
BIGINT	INT8	大きな符号付きの正確な整数。	BIGINT
CHAR(*n*)	CHAR(*n*)	固定長の文字列。	CHAR(*n*)
DATE	DATE	日付値。	DATE
DECIMAL|NUMERIC(*p*,*s*)	DECIMAL(*p*,*s*)	符号付き固定小数点 10 進数。	DECIMAL|NUMERIC(*p*,*s*)[3]
DOUBLE	DOUBLE	浮動小数点数。	DOUBLE
FLOAT(*p*)	FLOAT8(*p*)	浮動小数点数。	FLOAT(*p*)
INTEGER	INTEGER	通常の符号付きの正確な整数。	INTEGER
REAL	REAL	浮動小数点数。	REAL
SMALLINT	INT8	小さな符号付きの正確な整数。	SMALLINT
TIME(*p*)	TIME(*p*)	時間、分、および秒の時間値。	TIME(*p*)[2]
TIMESTAMP(*p*)	TIMESTAMP(*p*)	日時値。	TIMESTAMP(*p*)
VARBINARY(*n*)	BYTEA	可変長のバイナリ文字列。	VARBINARY(*n*)
VARCHAR(*n*)	VARCHAR(*n*)	可変長の文字列。	VARCHAR(*n*)

1 データ型のマッピング方法を制御する CT_PRESERVE=接続引数は、データ型を定義できるかどうかに影響する可能性があります。値 FORCE (デフォルト)および FORCE_COL_SIZE は、データ型を定義できるかどうかには影響しません。要求されたデータ型がデータソースにネイティブでない場合、または指定された精度またはスケールがデータソースの範囲内にない場合、値 STRICT および SAFE はエラーになる可能性があります。

2 Greenplum サーバーとの通信に使用される ODBC スタイルのインターフェイスにより、サーバーからクライアントへのデータ転送で秒の端数が失われます。

3 FedSQL では、DECIMAL データ型は、データソースに完全に渡すことができる要求でサポートされています。データソースに完全に渡すことができない要求の DECIMAL 列は、DOUBLE として処理されるため、精度に影響を与える可能性があります。詳細については、“データ型” (14 ページ)を参照してください。

Hive のデータ型

次の表に、Hive のデータ型サポートを示します。

NCHAR および NVARCHAR データ型は、データ定義には使用できません。

Hive 複合データ型は読み取ることができますが、データ定義には使用できません。

Hive データ型に関するデータソース固有の情報については、Hive データベースのドキュメントを参照してください。

注: この表の情報は、CAS で処理されるデータには適用されません。CAS にロードされたデータは、CAS データ型に変換されます。CAS データ型の変換については、*SAS Cloud Analytic Services: ユーザーガイド*の SAS Data Connector for Hadoop のドキュメントを参照してください。

表 A7.8 FedSQL データ型と Hive で使用されるデータ型のマッピング

データ型定義キーワード	Hive データ型	説明	返されるデータ型
[1, 4]	ARRAY<*data-type*>	整数の配列(インデックス可能なリスト)。	STRING[8]
BIGINT	BIGINT	-9,223,372,036,854,775,807 から 9,223,372,036,854,775,807 までの符号付き 8 バイト整数。	BIGINT
BINARY(*n*)	BINARY	最大 32K の可変長のバイナリ文字列。	BINARY
[1]	BOOLEAN	テキストの true または false の値。	TINYINT
CHAR(*n*)	CHAR(n)	最大 255 文字の文字列。 255 文字を超える値で CHAR を指定すると、列はかわりに VARCHAR(n)として作成されます。	CHAR
DATE	DATE[2]	ANSI SQL の日付型。	DATE
DECIMAL/ NUMERIC(*p*,*s*)	DECIMAL	38 桁の精度の固定小数点 10 進数。	DECIMAL(*p*,*s*)[3]
DOUBLE	DOUBLE	8 バイトの倍精度浮動小数点数。	DOUBLE
FLOAT(*p*)	FLOAT	4 バイトの単精度浮動小数点数。	DOUBLE
INTEGER	INTEGER	10 桁の精度で通常サイズの符号付きの正確な整数。整数の範囲は-2,147,483,647 から 2,147,483,647 までです。	INTEGER
[1, 4]	MAP<*primitive-type*, *data-type*>	キーと値のペアの連想配列。	STRING[8]
REAL	DOUBLE	64 ビットの倍精度浮動小数点数。	DOUBLE

データ型定義キーワード	Hive データ型	説明	返されるデータ型
SMALLINT	SMALLINT	-32,767 から 32,767 までの符号付き 2 バイト整数。	SMALLINT
[1, 4]	STRING	可変長の文字列。	VARCHAR(n)[6]
[1, 4]	STRUCT<*col-name*: *data-type*>	確立された列要素とデータ型を持つ構造。列要素とデータ型は、ダブルドット(:)表記を使用してマッピングされます。	STRING[8]
TIME(*p*)	[7]	時間値。	STRING
TIMESTAMP(*p*)	TIMESTAMP	オプションのナノ秒精度の UNIX タイムスタンプ。	TIMESTAMP[(*p*)]
TINYINT	TINYINT	-127 から 127 までの符号付き 1 バイト整数。	TINYINT
[1, 4]	UNION<*data-type*, *data-type-n*>	指定されたいくつかのデータ型の 1 つを保持できる型。	STRING[8]
VARCHAR(*n*)[5]	VARCHAR(*n*)	可変長の文字列。	STRING
VARBINARY	BINARY	最大 32K の可変長のバイナリ文字列。	BINARY

1 Hive データ型は定義できません。データが取得されると、ネイティブデータ型が同様のデータ型にマッピングされます。

2 サポートされている日付値は、1582 年 10 月 15 日から 9999 年 12 月 31 日までです。1582 より前の年を含む日付値はエラーを返します。9999 より後の日付値は null 値として読み戻されます。

3 FedSQL では、DECIMAL データ型は、データソースに完全に渡すことができる要求でサポートされています。データソースに完全に渡すことができない要求の DECIMAL 列は、DOUBLE として処理されるため、精度に影響を与える可能性があります。詳細については、“データ型” (14 ページ)を参照してください。

4 文字列および Hive 複合型は VARCHAR(*n*)としてロードされ、その長さは DBMAX_TEXT=データソース接続オプションによって決定されます。

5 65,535 文字を超える値を持つ VARCHAR 列は、STRING として作成されます。

6 SASFMT TableProperties は、STRING 列を読み取るときに適用されます。

7 Hive は TIME(p)データ型をサポートしていません。データが Hive から読み取られると、SAS TIME8.出力形式を指定する SASFMT TableProperties が定義されている STRING 列が、TIME(*p*)データ型に変換されます。TIME 型がデータ定義に使用される場合、SASFMT TableProperties を使用して STRING 列にマッピングされます。秒の端数は保持されません。SASFMT TableProperties の詳細については、*SAS/ACCESS for Relational Databases: リファレンス*の"SAS Table Properties for Hive and HADOOP"を参照してください。

8 複合型 ARRAY、MAP、STRUCT、UNION は、基になる複合型の STRING 表現として読み取られます。ARRAY 値は、角かっこで囲まれて読み戻されます(例: [1、2、4])。STRUCT と MAP の値は、中かっこで囲まれて読み戻されます。例: {"firstname":"robert","nickname":"bob"}。

Impala のデータ型

次の表に、Impala のデータ型サポートを示します。

BINARY、DECIMAL(*p*,*s*)/NUMERIC、NCHAR、NVARCHAR、VARBINARY データ型は、データ定義には使用できません。

Impala データ型に関するデータソース固有の情報については、ベンダーのドキュメントを参照してください。

注: この表の情報は、CAS で処理されるデータには適用されません。CAS にロードされたデータは、CAS データ型に変換されます。CAS データ型の変換については、*SAS Cloud Analytic Services: ユーザーガイド*の SAS Data Connector for Impala のドキュメントを参照してください。

表 A7.9 FedSQL データ型と Impala で使用されるデータ型のマッピング

データ型定義キーワード	Impala データ型	説明	返されるデータ型
BIGINT	BIGINT	-9,223,372,036,854,775,807 から 9,223,372,036,854,775,807 までの符号付き 8 バイト整数。	BIGINT
CHAR(*n*)	CHAR[1]	最大 255 文字の固定長文字列。	CHAR
DATE	[2]	ANSI SQL の日付型。	TIMESTAMP
DOUBLE	DOUBLE	8 バイトの倍精度浮動小数点数。	DOUBLE
FLOAT(*p*)	FLOAT	4 バイトの単精度浮動小数点数。	DOUBLE
INTEGER	INT	-2,147,483,647 から 2,147,483,647 までの符号付き 4 バイト整数。	INTEGER
REAL	DOUBLE	8 バイトの倍精度浮動小数点数。	DOUBLE

データ型定義キーワード	Impala データ型	説明	返されるデータ型
SMALLINT	SMALLINT	-32,767 から 32,767 までの符号付き 2 バイト整数。	SMALLINT
TIME	[2]	時間値。	TIMESTAMP
TIMESTAMP	TIMESTAMP	オプションのナノ秒精度の UNIX タイムスタンプ。	TIMESTAMP
TINYINT	TINYINT	-127 から 127 までの符号付き 1 バイト整数。	TINYINT
VARCHAR(*n*)	VARCHAR[1]	可変長の文字列。	VARCHAR

1 このデータ型のサポートは、CDH 5.2 以降で使用できます。CHAR および VARCHAR 型をサポートしない以前の CDH バージョンを使用する環境では、CHAR または VARCHAR として定義された列は STRING にマッピングされます。

2 Impala はこのデータ型をサポートしていません。DATE 列または TIME 列が定義されると、TIMESTAMP 型の列として作成されます。

JDBC のデータ型

次の表に、JDBC のデータ型サポートを示します。

JDBC データ型に関するデータソース固有の情報については、データベースのドキュメントを参照してください。

注: この表の情報は、CAS で処理されるデータには適用されません。CAS にロードされたデータは、CAS データ型に変換されます。CAS データ型の変換については、*SAS Cloud Analytic Services: ユーザーガイド*の SAS Data Connector for JDBC のドキュメントを参照してください。

表 A7.10 FedSQL データ型と JDBC でサポートされるデータ型のマッピング

データ型定義キーワード	JDBC SQL 識別子	説明	返されるデータ型
BIGINT	SQL_BIGINT	大きな符号付きの正確な整数。	BIGINT
BINARY(*n*)	SQL_BINARY	固定長のバイナリ文字列。	BINARY(*n*)

データ型定義キーワード	JDBC SQL 識別子	説明	返されるデータ型
[1]	SQL_BIT	シングルビットのバイナリ値。	[1]
CHAR(*n*)[2]	SQL_CHAR	固定長の文字列。	CHAR(*n*)
DATE	SQL_TYPE_DATE	日付値。	DATE
DECIMAL|NUMERIC(*p*,*s*)	SQL_DECIMAL|SQL_NUMERIC	符号付き固定小数点 10 進数。	DECIMAL|NUMERIC(*p*,*s*)[3]
DOUBLE	SQL_DOUBLE	符号付き倍精度浮動小数点数。	DOUBLE
FLOAT(*p*)	SQL_FLOAT	符号付き近似浮動小数点数。	FLOAT(*p*)
[1]	SQL_GUID	グローバル一意識別子。	[1]
INTEGER	SQL_INTEGER	通常の符号付きの正確な整数。	INTEGER
[1]	SQL_INTERVAL	2 つの年、月、日、日付、時間の間隔。	[1]
[1]	SQL_LONGVARBINARY	可変長のバイナリ文字列。	[1]
[1]	SQL_LONGVARCHAR	可変長の Unicode 文字列。	[1]
NCHAR(*n*)	SQL_WCHAR	固定長の Unicode 文字列。	NCHAR(*n*)
NVARCHAR(*n*)	SQL_WVARCHAR	可変長の Unicode 文字列。	NVARCHAR(*n*)
REAL	SQL_REAL	符号付き単精度浮動小数点数。	REAL
SMALLINT	SQL_SMALLINT	小さな符号付きの正確な整数。	SMALLINT
TIME(*p*)	SQL_TYPE_TIME	時間値。	TIME(*p*)
TIMESTAMP(*p*)	SQL_TYPE_TIMESTAMP	日時値。	TIMESTAMP(*p*)

データ型定義キーワード	JDBC SQL 識別子	説明	返されるデータ型
TINYINT	SQL_TINYINT	非常に小さな符号付きの正確な整数。	TINYINT
VARBINARY(*n*)	SQL_VARBINARY	可変長のバイナリ文字列。	VARBINARY(*n*)
VARCHAR(*n*)[2]	SQL_VARCHAR	可変長の文字列。	VARCHAR(*n*)
[1]	SQL_WLONGVARCHAR	可変長の Unicode 文字列。	[1]

1 JDBC SQL データ型は定義できません。データが取得されると、ネイティブデータ型が同様のデータ型にマッピングされます。

2 CHAR(n)または VARCHAR(n)データ型を使用してマルチバイトデータを保存する場合は、CLIENT_ENCODING=接続オプションでエンコーディングを指定する必要があります。または、エンコーディングを設定する必要がないように、かわりにマルチバイトデータに NCHAR または NVARCHAR データ型を使用します。

3 FedSQL では、DECIMAL データ型は、データソースに完全に渡すことができる要求でサポートされています。データソースに完全に渡すことができない要求の DECIMAL 列は、DOUBLE として処理されるため、精度に影響を与える可能性があります。詳細については、“データ型” (14 ページ)を参照してください。

MDS のデータ型

次の表に、インメモリデータベースのデータ型サポートを示します。

注: このデータソースは CAS サーバーではサポートされていません。

表 A7.11　FedSQL データ型と MDS テーブルで使用されるデータ型のマッピング

データ型定義キーワード [2]	MDS データ型	説明	返されるデータ型
BIGINT	BIGINT	64 ビットの符号付き整数。	BIGINT
BINARY(*n*)	BINARY(*n*)	固定長のバイナリ文字列。	BINARY(*n*)
CHAR(*n*)	CHAR(*n*)	固定長の文字列。	CHAR(*n*)
DATE	DATE	日付値。	DATE

データ型定義キーワード [2]	MDS データ型	説明	返されるデータ型
DECIMAL | NUMERIC(*p*,*s*)	NUMERIC(*p*,*s*)	精度とスケールが固定された正確な整数。	DECIMAL/ NUMERIC(*p*,*s*)[3]
DOUBLE	DOUBLE	8 バイトの IEEE 浮動小数点値。 **注:** ANSI SQL null 値をサポートします。	DOUBLE
FLOAT(*p*)	DOUBLE	8 バイトの IEEE 浮動小数点値。 **注:** ANSI SQL null 値をサポートします。	DOUBLE
INTEGER	INTEGER	32 ビットの符号付き整数。	INTEGER
NCHAR(*n*)	NCHAR(*n*)	固定長の Unicode 文字列。	NCHAR(*n*)
NVARCHAR(*n*)	NVARCHAR(*n*)	可変長の Unicode 文字列。	NVARCHAR(*n*)
REAL	DOUBLE	8 バイトの IEEE 浮動小数点値。 **注:** ANSI SQL null 値をサポートします。	DOUBLE
SMALLINT	INTEGER	32 ビットの符号付き整数。	INTEGER
TIME(*p*)	TIME(*p*)	時間値。	TIME(*p*)
TIMESTAMP(*p*)	TIMESTAMP(*p*)	日時値。	TIMESTAMP(*p*)
TINYINT	INTEGER	32 ビットの符号付き整数。	INTEGER
[1]	UBIGINT	64 ビットの符号なし整数。	BIGINT
[1]	UINTEGER	32 ビットの符号なし整数。	INTEGER
VARCHAR(*n*)	VARCHAR(n)	可変長の文字列。	VARCHAR(*n*)

データ型定義キーワード [2]	MDS データ型	説明	返されるデータ型
VARBINARY(n)	VARBINARY(n)	可変長のバイナリ文字列。	VARBINARY(n)

1 MDS SQL データ型は定義できません。データが取得されると、ネイティブデータ型が同様のデータ型にマッピングされます。

2 データ型のマッピング方法を制御する CT_PRESERVE=接続引数は、データ型を定義できるかどうかに影響する可能性があります。値 FORCE (デフォルト)および FORCE_COL_SIZE は、データ型を定義できるかどうかには影響しません。要求されたデータ型がデータソースにネイティブでない場合、または指定された精度またはスケールがデータソースの範囲内にない場合、値 STRICT および SAFE はエラーになる可能性があります。

3 FedSQL では、DECIMAL データ型は、データソースに完全に渡すことができる要求でサポートされています。データソースに完全に渡すことができない要求の DECIMAL 列は、DOUBLE として処理されるため、精度に影響を与える可能性があります。詳細については、“データ型” (14 ページ)を参照してください。

Microsoft SQL Server のデータ型

次の表に、Microsoft SQL Server データベースのデータ型サポートを示します。

Microsoft SQL Server データ型に関するデータソース固有の情報については、Microsoft SQL Server データベースのドキュメントを参照してください。

注: この表の情報は、CAS で処理されるデータには適用されません。CAS にロードされたデータは、CAS データ型に変換されます。CAS データ型の変換については、*SAS Cloud Analytic Services: ユーザーガイド*の SAS Data Connector for Microsoft SQL Server のドキュメントを参照してください。

表 A7.12 FedSQL データ型と Microsoft SQL Server で使用されるデータ型のマッピング

データ型定義キーワード [1]	SQL Server SQL 識別子	説明	返されるデータ型
BIGINT	BIGINT	大きな符号付きの正確な整数。	BIGINT
BINARY(*n*)	BINARY(*n*)	固定長のバイナリ文字列。	BINARY(*n*)
[2]	BIT	シングルビットのバイナリ値。	[2]
CHAR(*n*)	CHAR(*n*)	固定長の文字列。	CHAR(*n*)
DATE	DATE	日付値。	DATE

データ型定義キーワード [1]	SQL Server SQL 識別子	説明	返されるデータ型
[2]	DATETIME	24 時間表記に基づく秒の端数付き日時値。	TIMESTAMP(*p*)
[2]	DATETIME2(*p*)	より大きな日付範囲、より大きなデフォルトの小数精度、およびオプションのユーザー指定の精度を持つ拡張 DATETIME 値。	TIMESTAMP(*p*)
[2]	DATETIMEOFFSET(*p*)	タイムゾーンを認識した日時値。	[2]
DECIMAL|NUMERIC(*p*,*s*)	DECIMAL|NUMERIC(*p*,*s*)	固定精度とスケールの符号付き数値。	DECIMAL|NUMERIC(*p*,*s*)[3]
DOUBLE	DOUBLE	符号付き倍精度浮動小数点数。	DOUBLE
FLOAT(*p*)	FLOAT(*n*)	符号付き近似浮動小数点数。	FLOAT(*p*)
[2]	IMAGE	可変長のバイナリ値。	[2]
INTEGER	INTEGER	通常の符号付きの正確な整数。	INTEGER
[2]	MONEY	8 バイトの通貨値。	DECIMAL(19,4)[3]
NCHAR(*n*)	NCHAR	固定長の Unicode 文字列。	NCHAR(*n*)
NVARCHAR(*n*)	NVARCHAR	可変長の Unicode 文字列。	NVARCHAR(*n*)
REAL	REAL	符号付き単精度浮動小数点数。	REAL
[2]	SMALLDATETIME	秒の端数のない日時値。	TIMESTAMP(*p*)
SMALLINT	SMALLINT	小さな符号付きの正確な整数。	SMALLINT
[2]	SMALLMONEY	4 バイトの通貨値。	DECIMAL(10,4)[3]
TIME(*p*)	TIME(*p*)	時間値。	TIME(*p*)

データ型定義キーワード [1]	SQL Server SQL 識別子	説明	返されるデータ型
TIMESTAMP(*p*)	BINARY(8)	日時値。	TIMESTAMP(*p*)
TINYINT	TINYINT	非常に小さな符号付きの正確な整数。	TINYINT
[2]	UNIQUEIDENTIFIER	グローバル一意識別子。	CHAR(36)
VARBINARY(*n*)	VARBINARY	可変長のバイナリ文字列。	VARBINARY(*n*)
[2]	VARBINARY(MAX)	可変長のバイナリ文字列。最大 2GB。	[2]
VARCHAR(*n*)	VARCHAR(*n*)、TEXT	可変長の文字列。[2]	VARCHAR(*n*)
[2]	VARCHAR(MAX)	可変長の Unicode 文字列。最大 2GB。	[2]

1 データ型のマッピング方法を制御する CT_PRESERVE=接続引数は、データ型を定義できるかどうかに影響する可能性があります。値 FORCE (デフォルト)および FORCE_COL_SIZE は、データ型を定義できるかどうかには影響しません。要求されたデータ型がデータソースにネイティブでない場合、または指定された精度またはスケールがデータソースの範囲内にない場合、値 STRICT および SAFE はエラーになる可能性があります。

2 Microsoft SQL Server データ型は定義できません。データが取得されると、ネイティブデータ型が同様のデータ型にマッピングされます。

3 FedSQL では、DECIMAL データ型は、データソースに完全に渡すことができる要求でサポートされています。データソースに完全に渡すことができない要求の DECIMAL 列は、DOUBLE として処理されるため、精度に影響を与える可能性があります。詳細については、“データ型” (14 ページ)を参照してください。

MongoDB のデータ型

次の表に、MongoDB 非リレーショナルデータベースのデータ型サポートを示します。

MongoDB の Timestamp 型、Min key 型、Max key 型は MongoDB の内部型です。したがって、SAS によって読み取られることはありません。

FedSQL の BINARY、DECIMAL、FLOAT、NCHAR、NVARCHAR、REAL、SMALLINT、TINYINT、VARBINARY データ型は、データ定義ではサポートされていません。

MongoDB データ型に関するデータソース固有の情報については、MongoDB のドキュメントを参照してください。

注: この表の情報は、CAS で処理されるデータには適用されません。CAS にロードされたデータは、CAS データ型に変換されます。CAS データ型の変換については、

*SAS Cloud Analytic Services: ユーザーガイド*の SAS Data Connector for MongoDB のドキュメントを参照してください。

表 A7.13 FedSQL データ型と MongoDB で使用されるデータ型のマッピング

データ型定義キーワード	MongoDB データ型	説明	返されるデータ型
[1]	Array	配列。	VARCHAR(n)
BIGINT	int	通常の 64 ビットの正確な整数。	BIGINT
[2]	BinData	バイナリ値。	VARCHAR[3]
[2]	Bool	ブール値。	INTEGER
CHAR(*n*)	String	固定長の UTF-8 文字列。	CHAR(*n*)
	Date	日時値。	DATE
[2]	Decimal	Decimal128 値。	DOUBLE
DOUBLE	Double	符号付き倍精度浮動小数点数。	DOUBLE
INTEGER	Int	通常の 32 ビットの正確な整数。	INTEGER
[2]	Javascript	Javascript 文字列。	[1]
[2]	Long	大きな符号付きの正確な整数。	BIGINT
[2]	Null	null 値。	VARCHAR
[1]	Object	JSON オブジェクト。	VARCHAR(n)
[2]	ObjectId[4]	オブジェクト識別子。	VARCHAR
[2]	Regex	正規表現文字列。	VARCHAR
TIMESTAMP	Date	日時値。	TIMESTAMP

データ型定義キーワード	MongoDB データ型	説明	返されるデータ型
VARCHAR	String	可変長の UTF-8 文字列。	VARCHAR、CHAR(*n*)[5]

1 配列とオブジェクトの型は、別々の子テーブルとしてモデル化されます。詳細については、*SAS/ACCESS for Nonrelational Databases: リファレンス*の MongoDB のドキュメントを参照してください。Javascript 型は VARCHAR にマッピングされます。

2 MongoDB データ型は定義できません。データが取得されると、ネイティブデータ型が同様のデータ型にマッピングされます。

3 バイナリデータの 16 進文字列表現が含まれます。

4 MongoDB テーブルで ObjectId フィールドを定義または複製することはできません。ただし、MongoDB コレクションにすでに存在する ObjectId フィールドに値を挿入および更新できます。

5 CHAR(n)を使用するようにデフォルトの MongoDB スキーマをカスタマイズできます。詳細については、*SAS/ACCESS for Nonrelational Databases: リファレンス*の MongoDB のドキュメントを参照してください。

MySQL のデータ型

次の表に、MySQL データベースのデータ型サポートを示します。

NCHAR、NVARCHAR、REAL、VARBINARY データ型は、データ型定義ではサポートされていません。

MySQL データ型に関するデータソース固有の情報については、MySQL データベースのドキュメントを参照してください。

注: この表の情報は、CAS で処理されるデータには適用されません。CAS にロードされたデータは、CAS データ型に変換されます。CAS データ型の変換については、*SAS Cloud Analytic Services: ユーザーガイド*の SAS Data Connector for MySQL のドキュメントを参照してください。

表 A7.14 FedSQL データ型と MySQL で使用されるデータ型のマッピング

データ型定義キーワード [1]	MySQL データ型	説明	返されるデータ型
BIGINT	BIGINT	大きな符号付きの正確な整数。	BIGINT
BINARY(*n*)	BINARY(*n*)	固定長のバイナリ文字列。	BINARY(*n*)
[2]	BLOB	バイナリラージオブジェクト文字列。	[2]
CHAR(*n*)	CHAR(*n*)[3]	固定長の文字列。	CHAR(*n*)
DATE	DATE	日付値。	DATE

データ型定義キーワード [1]	MySQL データ型	説明	返されるデータ型
[2]	DATETIME	日時値。	[2]
DECIMAL|NUMERIC(*p*,*s*)	DECIMAL(*p*,*s*)	符号付き固定小数点 10 進数。	DECIMAL(*p*,*s*)[4]
DOUBLE	DOUBLE	符号付き倍精度浮動小数点数。	DOUBLE
[2]	ENUM(*values*)	許可された値のリストからの文字値。	[2]
FLOAT(*p*)	FLOAT(*p*)	符号付きの単精度または倍精度の浮動小数点数。	FLOAT(*p*)
INTEGER	INT	通常の符号付きの正確な整数。	INTEGER
[2]	JSON	JSON 形式の可変長文字列。	VARCHAR
[2]	LONGBLOB	可変長のバイナリラージオブジェクト文字列。	[2]
[2]	LONGTEXT	可変長のテキスト文字列。	[2]
[2]	MEDIUMBLOB	可変長のバイナリラージオブジェクト文字列。	[2]
[2]	MEDIUMINT	通常の符号付きの正確な整数。	[2]
[2]	MEDIUMTEXT	可変長の文字列。	[2]
[2]	SET(*values*)	許可された値のリストからの文字値。	[2]
SMALLINT	SMALLINT	小さな符号付きの正確な整数。	SMALLINT
[2]	TEXT	テキスト文字列。	[2]
TIME(*p*)	TIME(*p*)	時間値。	TIME(*p*)
TIMESTAMP(*p*)	DATETIME(*p*)	日時値。	TIMESTAMP(*p*)
[2]	TINYBLOB	可変長のバイナリラージオブジェクト文字列。	[2]
TINYINT	TINYINT	非常に小さな符号付きの正確な整数。	TINYINT
[2]	TINYTEXT	可変長のテキスト文字列。	[2]

データ型定義キーワード [1]	MySQL データ型	説明	返されるデータ型
VARCHAR(*n*)	VARCHAR(*n*)	可変長の文字列。	VARCHAR(*n*)

1 データ型のマッピング方法を制御する CT_PRESERVE=接続引数は、データ型を定義できるかどうかに影響する可能性があります。値 FORCE (デフォルト)および FORCE_COL_SIZE は、データ型を定義できるかどうかには影響しません。要求されたデータ型がデータソースにネイティブでない場合、または指定された精度またはスケールがデータソースの範囲内にない場合、値 STRICT および SAFE はエラーになる可能性があります。

2 MySQL データ型は定義できません。データが取得されると、ネイティブデータ型が同様のデータ型にマッピングされます。

3 DS2 は、固定文字値の空白スペースを処理するために ANSI SQL 規則を使用します。MySQL 8 は、デフォルトで末尾の空白スペースを重要な値として扱います。この違いは、DS2 で FedSQL ステートメントを使用するときに実行される文字比較に干渉します。DS2 を使用して MySQL 8 CHAR()データに対して FedSQL ステートメントを発行する場合、データが正しく読み取られるように、FedSQL ステートメントで TRIM 関数を使用する必要があります。DS2 で FedSQL ステートメントがサポートされている場所については、“DS2 と FedSQL の使用” (*SAS DS2 プログラマガイド*)を参照してください。また、“例: MySQL 8 CHAR()データから末尾のスペースをフィルタリングする” (*SAS DS2 言語リファレンス*)も参照してください。MySQL 8 は、比較のデフォルト照合順序として utf8mb4_0900_ai_ci を使用します。

4 FedSQL では、DECIMAL データ型は、データソースに完全に渡すことができる要求でサポートされています。データソースに完全に渡すことができない要求の DECIMAL 列は、DOUBLE として処理されるため、精度に影響を与える可能性があります。詳細については、“データ型” (14 ページ)を参照してください。

Netezza のデータ型

次の表に、Netezza データベースのデータ型サポートを示します。

BINARY および VARBINARY データ型は、データ型定義ではサポートされていません。

Netezza データ型に関するデータソース固有の情報については、Netezza データベースのドキュメントを参照してください。

注: この表の情報は、CAS で処理されるデータには適用されません。CAS にロードされたデータは、CAS データ型に変換されます。CAS データ型の変換については、*SAS Cloud Analytic Services: ユーザーガイド*の SAS Data Connector for Netezza のドキュメントを参照してください。

表 A7.15 FedSQL データ型と Netezza で使用されるデータ型のマッピング

データ型定義キーワード [1]	Netezza データ型	説明	返されるデータ型
BIGINT	BIGINT	大きな符号付きの正確な整数。	BIGINT
CHAR(*n*)	CHAR(*n*)	固定長の文字列データ。	CHAR(*n*)
DATE	DATE	日付値。	DATE
DECIMAL|NUMERIC(*p*,*s*)	DECIMAL(*p*,*s*)	固定小数点 10 進数。	DECIMAL(*p*,*s*)[3]

データ型定義キーワード [1]	Netezza データ型	説明	返されるデータ型
DOUBLE	DOUBLE	浮動小数点数。	DOUBLE
FLOAT(*p*)	FLOAT(*p*)	64 ビットの倍精度浮動小数点数。	FLOAT(*p*)
INTEGER	INTEGER	大きな整数。	INTEGER
NCHAR(*n*)	NCHAR(*n*)	固定長の Unicode 文字列。	NCHAR(*n*)
NVARCHAR(*n*)	NVARCHAR(*n*)	可変長の Unicode 文字列。	NVARCHAR(*n*)
REAL	REAL	浮動小数点数。	REAL
SMALLINT	SMALLINT	小さな整数。	SMALLINT
TIME(*p*)	TIME(*p*)	時間値。	TIME(*p*)[2]
TIMESTAMP(*p*)	TIMESTAMP(*p*)	日時値。	TIMESTAMP(*p*)
TINYINT	BYTEINT	極小整数。	TINYINT
VARCHAR(*n*)	VARCHAR(*n*)	可変長の文字列データ。	VARCHAR(*n*)

1 データ型のマッピング方法を制御する CT_PRESERVE=接続引数は、データ型を定義できるかどうかに影響する可能性があります。値 FORCE (デフォルト)および FORCE_COL_SIZE は、データ型を定義できるかどうかには影響しません。要求されたデータ型がデータソースにネイティブでない場合、または指定された精度またはスケールがデータソースの範囲内にない場合、値 STRICT および SAFE はエラーになる可能性があります。

2 Netezza サーバーとの通信に使用される ODBC スタイルのインターフェイスにより、サーバーからクライアントへのデータ転送で秒の端数が失われます。

3 FedSQL では、DECIMAL データ型は、データソースに完全に渡すことができる要求でサポートされています。データソースに完全に渡すことができない要求の DECIMAL 列は、DOUBLE として処理されるため、精度に影響を与える可能性があります。詳細については、“データ型” (14 ページ)を参照してください。

ODBC のデータ型

次の表に、ODBC に準拠するデータソースのデータ型サポートを示します。

ODBC SQL データ型に関するデータソース固有の情報については、特定の ODBC データソースのドキュメントを参照してください。

注: この表の情報は、CAS で処理されるデータには適用されません。CAS にロードされたデータは、CAS データ型に変換されます。CAS データ型の変換については、*SAS Cloud Analytic Services: ユーザーガイド*の SAS Data Connector for ODBC のドキュメントを参照してください。

表 A7.16　FedSQL データ型と ODBC でサポートされるデータ型のマッピング

データ型定義キーワード [1]	ODBC SQL 識別子	説明	返されるデータ型
BIGINT	SQL_BIGINT	大きな符号付きの正確な整数。	BIGINT
BINARY(*n*)	SQL_BINARY	固定長のバイナリ文字列。	BINARY(*n*)
[2]	SQL_BIT	シングルビットのバイナリ値。	[2]
CHAR(*n*)[3]	SQL_CHAR	固定長の文字列。	CHAR(*n*)
DATE	SQL_TYPE_DATE	日付値。	DATE
DECIMAL|NUMERIC(*p*,*s*)	SQL_DECIMAL|SQL_NUMERIC	符号付き固定小数点 10 進数。	DECIMAL|NUMERIC(*p*,*s*)[4]
DOUBLE	SQL_DOUBLE	符号付き倍精度浮動小数点数。	DOUBLE
FLOAT(*p*)	SQL_FLOAT	符号付き近似浮動小数点数。	FLOAT(*p*)
[2]	SQL_GUID	グローバル一意識別子。	[2]
INTEGER	SQL_INTEGER	通常の符号付きの正確な整数。	INTEGER
[2]	SQL_INTERVAL	2 つの年、月、日、日付、時間の間隔。	[2]
[2]	SQL_LONGVARBINARY	可変長のバイナリ文字列。	[2]
[2]	SQL_LONGVARCHAR	可変長の Unicode 文字列。	[2]
NCHAR(*n*)	SQL_WCHAR	固定長の Unicode 文字列。	NCHAR(*n*)
NVARCHAR(*n*)	SQL_WVARCHAR	可変長の Unicode 文字列。	NVARCHAR(*n*)
REAL	SQL_REAL	符号付き単精度浮動小数点数。	REAL

データ型定義キーワード [1]	ODBC SQL 識別子	説明	返されるデータ型
SMALLINT	SQL_SMALLINT	小さな符号付きの正確な整数。	SMALLINT
TIME(*p*)	SQL_TYPE_TIME	時間値。	TIME(*p*)
TIMESTAMP(*p*)	SQL_TYPE_TIMESTAMP	日時値。	TIMESTAMP(*p*)
TINYINT	SQL_TINYINT	非常に小さな符号付きの正確な整数。	TINYINT
VARBINARY(*n*)	SQL_VARBINARY	可変長のバイナリ文字列。	VARBINARY(*n*)
VARCHAR(*n*)[3]	SQL_VARCHAR	可変長の文字列。	VARCHAR(*n*)
[2]	SQL_WLONGVARCHAR	可変長の Unicode 文字列。	[2]

1 データ型のマッピング方法を制御する CT_PRESERVE=接続引数は、データ型を定義できるかどうかに影響する可能性があります。値 FORCE (デフォルト)および FORCE_COL_SIZE は、データ型を定義できるかどうかには影響しません。要求されたデータ型がデータソースにネイティブでない場合、または指定された精度またはスケールがデータソースの範囲内にない場合、値 STRICT および SAFE はエラーになる可能性があります。

2 ODBC SQL データ型は定義できません。データが取得されると、ネイティブデータ型が同様のデータ型にマッピングされます。

3 CHAR(*n*)または VARCHAR(*n*)データ型を使用してマルチバイトデータを DB2、Greenplum、または Oracle データベースに保存する場合は、CLIENT_ENCODING=接続オプションでエンコーディングを指定する必要があります。または、Oracle の場合のみ、エンコーディングを設定する必要がないように、かわりにマルチバイトデータに NCHAR または NVARCHAR データ型を使用します。

4 FedSQL では、DECIMAL データ型は、データソースに完全に渡すことができる要求でサポートされています。データソースに完全に渡すことができない要求の DECIMAL 列は、DOUBLE として処理されるため、精度に影響を与える可能性があります。詳細については、“データ型” (14 ページ)を参照してください。

Oracle のデータ型

次の表に、Oracle データベースのデータ型サポートを示します。

Oracle データ型に関するデータソース固有の情報については、Oracle データベースのドキュメントを参照してください。

注: この表の情報は、CAS で処理されるデータには適用されません。CAS にロードされたデータは、CAS データ型に変換されます。CAS データ型の変換については、*SAS Cloud Analytic Services: ユーザーガイド*の SAS Data Connector for Oracle のドキュメントを参照してください。

表 A7.17　FedSQL データ型と Oracle で使用されるデータ型のマッピング

データ型定義キーワード [1]	Oracle データ型	説明	返されるデータ型
BIGINT	NUMBER	大きな符号付きの正確な整数。	BIGINT
BINARY(*n*)	RAW(*n*)	固定長または可変長のバイナリ文字列。	BINARY(*n*)
CHAR(*n*)	CHAR(*n*)	固定長の文字列。	CHAR(*n*)
DATE	DATE	日付値。	TIMESTAMP(*p*)[3]
DECIMAL|NUMERIC(*p*,*s*)	NUMBER(*p*,*s*)	符号付き固定小数点 10 進数。	DOUBLE[4]
DOUBLE	BINARY_DOUBLE	符号付き倍精度浮動小数点数。	DOUBLE
FLOAT(*p*)	FLOAT(p)	符号付き倍精度浮動小数点数。	DOUBLE
INTEGER	NUMBER(*p*,*s*)	通常の符号付きの正確な整数。	INTEGER
[2]	LONG	可変長の文字列。	[2]
NCHAR(*n*)	NCHAR(*n*)	固定長の Unicode 文字列。	NCHAR(*n*)
NVARCHAR(*n*)	NVARCHAR(*n*)	可変長の Unicode 文字列。	NVARCHAR(*n*)
REAL	FLOAT	符号付き単精度浮動小数点数。	REAL
SMALLINT	NUMBER	小さな符号付きの正確な整数。	SMALLINT
TIME(*p*)	TIME(*p*)	時間値。	TIMESTAMP(*p*)[3]
TIMESTAMP(*p*)	TIMESTAMP(*p*)	日時値。	TIMESTAMP(*p*)
TINYINT	NUMBER	非常に小さな符号付きの正確な整数。	TINYINT
VARBINARY(*n*)	RAW(*n*)	可変長のバイナリ文字列。	VARBINARY(*n*)
VARCHAR(*n*)	VARCHAR2(*n*)[5]	可変長の文字列。	VARCHAR(*n*)

1 データ型のマッピング方法を制御する CT_PRESERVE=接続引数は、データ型を定義できるかどうかに影響する可能性があります。値 FORCE (デフォルト)および FORCE_COL_SIZE は、データ型を定義できるかどうかには影響しません。要求され

たデータ型がデータソースにネイティブでない場合、または指定された精度またはスケールがデータソースの範囲内にない場合、値 STRICT および SAFE はエラーになる可能性があります。

2 Oracle データ型は定義できません。データが取得されると、ネイティブデータ型が同様のデータ型にマッピングされます。

3 DATE および TIME データ型によって返されるタイムスタンプは、PUT 関数で DATEPART 関数を使用することにより、日時値に変更できます。

4 ORNUMERIC=接続引数およびテーブルオプションは、Oracle NUMBER 列から読み取られた数値または Oracle NUMBER 列に挿入された数値の処理方法を決定します。デフォルトである ORNUMERIC=YES は、明示的な精度を持つ非整数値が NUMERIC 値として扱われることを示します。BULKLOAD=YES の場合、ORNUMERIC のデフォルト値は NO です。

5 Oracle バージョンが 12c で、Oracle MAX_STRING_SIZE=パラメーターが EXTENDED に設定されている場合、VARCHAR2(*n*)型は最大 32,767 バイトでサポートされます。

PostgreSQL のデータ型

次の表に、PostgreSQL データベースのデータ型サポートを示します。

BINARY、NCHAR、NVARCHAR、TINYINT、VARBINARY データ型は、データ型定義ではサポートされていません。

PostgreSQL データ型に関するデータソース固有の情報については、PostgreSQL データベースのドキュメントを参照してください。

注: この表の情報は、CAS で処理されるデータには適用されません。CAS にロードされたデータは、CAS データ型に変換されます。CAS データ型の変換については、*SAS Cloud Analytic Services: ユーザーガイド*の SAS Data Connector for PostgreSQL のドキュメントを参照してください。

表 A7.18 FedSQL データ型と PostgreSQL で使用されるデータ型のマッピング

データ型定義キーワード [1]	PostgreSQL データ型	説明	返されるデータ型
BIGINT	BIGINT	大きな符号付きの正確な整数または符号付き 8 バイト整数。	BIGINT
[2]	BIGSERIAL	自動増分 8 バイト整数。	[2]
[2]	BIT(*n*)	固定長のビット文字列。	[2]
[2]	BIT VARYING(*n*)	可変長のビット文字列。	[2]
[2]	BOOLEAN	論理ブール(true/false)値。	[2]
[2]	BOX	平面上の長方形のボックス。	[2]
[2]	BYTEA	バイト配列。	[2]
CHAR(*n*)	CHAR(*n*)	固定長の文字列。	CHAR(*n*)

データ型定義キーワード [1]	PostgreSQL データ型	説明	返されるデータ型
[2]	CIDR	IPv4 または IPv6 ネットワークアドレス。	[2]
[2]	CIRCLE	平面上の円。	[2]
DATE	DATE	日付値。	DATE
DECIMAL|NUMERIC(*p*,*s*)	NUMERIC(*p*,*s*)	符号付き固定小数点 10 進数。	DECIMAL|NUMERIC(*p*,*s*)[3]
DOUBLE	DOUBLE PRECISION	符号付き倍精度浮動小数点数。	DOUBLE
FLOAT(*p*)	FLOAT(*p*)	符号付きの単精度または倍精度の浮動小数点数。	FLOAT(*p*)
[2]	INET	IPv4 または IPv6 ホストアドレス。	[2]
INTEGER	INTEGER	通常の符号付きの正確な整数。	INTEGER
[2]	INTERVAL	期間。	[2]
[2]	LINE	平面上の無限の線。	[2]
[2]	LSEG	平面上の線分。	[2]
[2]	MACADDR	メディアアクセスコントロールアドレス。	[2]
[2]	MONEY	通貨額。	[2]
[2]	PATH	平面上の幾何学的パス。	[2]
[2]	POINT	平面上の幾何学的な点。	[2]
[2]	POLYGON	平面上の閉じた幾何学的パス。	[2]
REAL	REAL	符号付き単精度浮動小数点数。	REAL
[2]	SERIAL	自動増分 4 バイト整数。	[2]
SMALLINT	SMALLINT	小さな符号付きの正確な整数。	SMALLINT
[2]	SMALL SERIAL	自動増分 2 バイト整数。	[2]
[2]	TEXT	可変長の文字列。	[2]

データ型定義キーワード [1]	PostgreSQL データ型	説明	返されるデータ型
TIME(*p*)	TIME(*p*)	時間値。	TIME(*p*)
TIMESTAMP(*p*)	TIMESTAMP(*p*)	日時値。	TIMESTAMP(*p*)
[2]	TSQUERY	テキスト検索クエリ。	[2]
[2]	TSVECTOR	テキスト検索ドキュメント。	[2]
[2]	TXID_SNAPSHOT	ユーザーレベルのトランザクション ID スナップショット。	[2]
[2]	UUID	ユニバーサル一意識別子。	[2]
VARCHAR(*n*)	CHARACTER VARYING(*n*)	可変長の文字列。	VARCHAR(*n*)
[2]	XML	XML データ。	[2]

1 データ型のマッピング方法を制御する CT_PRESERVE=接続引数は、データ型を定義できるかどうかに影響する可能性があります。値 FORCE (デフォルト)および FORCE_COL_SIZE は、データ型を定義できるかどうかには影響しません。要求されたデータ型がデータソースにネイティブでない場合、または指定された精度またはスケールがデータソースの範囲内にない場合、値 STRICT および SAFE はエラーになる可能性があります。

2 PostgreSQL データ型は定義できません。データが取得されると、ネイティブデータ型が同様のデータ型にマッピングされます。

3 FedSQL では、DECIMAL データ型は、データソースに完全に渡すことができる要求でサポートされています。データソースに完全に渡すことができない要求の DECIMAL 列は、DOUBLE として処理されるため、精度に影響を与える可能性があります。詳細については、“データ型” (14 ページ)を参照してください。

Salesforce から読み取るためのデータ型

次の表に、Salesforce から読み取るためのデータ型のサポートを示します。SAS/ACCESS ソフトウェアは、Salesforce SOAP API を使用して Salesforce から読み取ります。Salesforce SOAP API は、プリミティブ型を返します。

Salesforce の計算型と複合型はサポートされていません。複合型の例は、住所と位置です。

詳細については、Salesforce SOAP API のドキュメントを参照してください。

注: この表の情報は、CAS で処理されるデータには適用されません。CAS にロードされたデータは、CAS データ型に変換されます。CAS データ型の変換については、*SAS Cloud Analytic Services: ユーザーガイド*の SAS Data Connector for Salesforce のドキュメントを参照してください。

表 A7.19　Salesforce から読み取るときのデータ型マッピング

Salesforce フィールドまたはプリミティブ型	説明	FedSQL で返されるデータ型
anyType	コンテキストに基づいてインターフェイス全体で異なる方法で使用できる多相データ型。	VARCHAR(*n*)
base64	attachment レコード、document レコード、および scontrol レコードを含む base-64 バイナリデータ。Body/Binary フィールドにはデータが含まれています。BodyLength フィールドは、データの長さを定義します。	VARCHAR(*n*)[1]
boolean	ブール値。	TINYINT
byte	ビットのセット。	TINYINT
combobox	ユーザーがリストにない値を定義できるようにする列挙値のセット。	VARCHAR(*n*)
currency	通貨値。	DOUBLE
DataCategoryGroupReference	Article オブジェクトと Question オブジェクトのデータカテゴリグループまたはカテゴリの一意の名前への参照。	VARCHAR(*n*)
date	日付値。	DATE[2]
dateTime	タイムスタンプ。	TIMESTAMP[2]
double	符号付き倍精度浮動小数点数。	DOUBLE[3]
email	メールアドレス。	VARCHAR(*n*)
encryptedstring	暗号化されたテキスト文字列。文字列の最大長は 175 文字です。	VARCHAR(*n*)
ID	オブジェクトの主キーフィールド。	VARCHAR(*n*)
int	サイズがフィールド設定によって異なる、通常の符号付きの正確な整数。	BIGINT
JunctionIdList	基になるジャンクションエンティティの 多対多の関係を表す、参照される ID 値の文字列配列。	VARCHAR(*n*)
masterrecord	マージされ、元のレコードが削除されるレコードの識別子。	VARCHAR(*n*)

Salesforce フィールドまたはプリミティブ型	説明	FedSQL で返されるデータ型
multipicklist	複数の値を選択できる列挙値のセット。選択された値は、セミコロンで区切られた値の文字列として返されます。	VARCHAR(*n*)
percent	パーセンテージ値。	DOUBLE
phone	電話番号。	CHAR(40)
picklist	1 つの値を選択できる列挙値のセット。	VARCHAR(*n*)
reference	別のオブジェクトの相互参照。SQL の外部キーに類似しています。	VARCHAR(*n*)
String	フィールド設定によって最大サイズが異なるテキスト文字列。	CHAR(*n*)
textarea	複数行のテキストフィールドとして表示される文字列。	VARCHAR(*n*)
Time	時間値。	TIME[2]
url	URL (Uniform Resource Locator)値。	VARCHAR(*n*)

1 base64 フィールドは、VARCHAR 型として base64 形式のままです。

2 最も早い有効な日付は、1700-01-01T00:00:00Z GMT、または 1700 年 1 月 1 日の午前 0 時直後です。最も遅い有効な日付は 4000-12-31T00:00:00Z GMT、または 4000 年 12 月 31 日の午前 0 時直後です。これらの値は、タイムゾーンによってオフセットされます。たとえば、太平洋タイムゾーンでは、最も早い有効な日付は 1699-12-31T16:00:00、または 1699 年 12 月 31 日の午後 4 時です。

3 “Salesforce に書き込むためのデータ型” (1233 ページ)で説明されている数値型の精度とスケールに一致する double 型は、その表に示すようにマッピングされます。たとえば、精度が 10 でスケールが 0 の double 列は、INTEGER にマッピングされます。

Salesforce に書き込むためのデータ型

次の表に、Salesforce に書き込むためのデータ型のサポートを示します。

SAS は、Salesforce の Winter'19 リリース以降をサポートしています。

BINARY、FLOAT、および VARBINARY データ型は、データ定義ではサポートされていません。

詳細については、Salesforce メタデータ API のドキュメントを参照してください。

注: この表の情報は、CAS で処理されるデータには適用されません。CAS にロードされたデータは、CAS データ型に変換されます。CAS データ型の変換については、

*SAS Cloud Analytic Services: ユーザーガイド*の SAS Data Connector for Salesforce のドキュメントを参照してください。

表 A7.20　Salesforce に書き込むときのデータ型マッピング

データ型定義キーワード	Salesforce カスタムフィールド型	説明	AS 式を使用して CREATE TABLE で返されるデータ型
BIGINT	Number	精度 18、スケール 0 の数値。	BIGINT
CHAR(*n*)[1]	Text	255 文字以下の文字列。	CHAR(*n*)
DATE	Date	日付値。[2]	DATE
DECIMAL|NUMERIC(*p*,*s*)	Number	精度とスケールのある数値。	DOUBLE
DOUBLE	Number	精度 18、スケール 4 の数値。	DOUBLE
INTEGER	Number	精度 10、スケール 0 の数値。	INTEGER
NCHAR(*n*)	Text	固定長の Unicode 文字列。	CHAR(*n*)
NVARCHAR(*n*)	Text	可変長の Unicode 文字列。	VARCHAR(*n*)
REAL	Number	精度 18、スケール 4 の数値。	DOUBLE
SMALLINT	Number	精度 5、スケール 0 の数値。	SMALLINT
TIME	Time	時間値。[2]	TIME
TIMESTAMP	Datetime	タイムスタンプ。[2]	TIMESTAMP
TINYINT	Number	精度 3、スケール 0 の数値。	TINYINT
VARCHAR(*n*)[3]	LongTextArea	255 文字より大きく 32K 文字未満の文字列。	VARCHAR(*n*)

1 *n* が 255 文字を超える CHAR(*n*)値は、VARCHAR(*n*)として作成されます。

2 最も早い有効な日付は、1700-01-01T00:00:00Z GMT、または 1700 年 1 月 1 日の午前 0 時直後です。最も遅い有効な日付は 4000-12-31T00:00:00Z GMT、または 4000 年 12 月 31 日の午前 0 時直後です。これらの値は、タイムゾーンによってオフセットされます。たとえば、太平洋タイムゾーンでは、最も早い有効な日付は 1699-12-31T16:00:00、または 1699 年 12 月 31 日の午後 4 時です。

3 *n* が 255 文字以下の VARCHAR(*n*)値は、CHAR(*n*)として作成されます。

SAP のデータ型

次の表に、SAP システムのデータ型サポートを示します。

SAP システムの場合、列定義でサポートされているデータ型はありません。ネイティブ ABAP SAP データ型は、データ取得のみを目的として同様のデータ型にマッピングされます。

ABAP SAP データ型に関するデータソース固有の情報については、SAP システムデータベースのドキュメントを参照してください。

注: このデータソースは CAS サーバーではサポートされていません。

表 A7.21　SAP データ型と FedSQL データ型のマッピング

ABAP SAP データ型	説明	データ取得に使用される FedSQL データ型
ACCP	転記期間。	非 Unicode SAP システムの場合は CHAR(*n*)、Unicode SAP システムの場合は NCHAR(*n*)
CHAR	固定長の文字列。	非 Unicode SAP システムの場合は CHAR(*n*)、Unicode SAP システムの場合は NCHAR(*n*)
CLNT	クライアントフィールド。	非 Unicode SAP システムの場合は CHAR(*n*)、Unicode SAP システムの場合は NCHAR(*n*)
CUKY	通貨キー。この型のフィールドは、CURR 型のフィールドによって参照されます。	非 Unicode SAP システムの場合は CHAR(*n*)、Unicode SAP システムの場合は NCHAR(*n*)
CURR	通貨フィールド。DEC フィールドに対応します。フィールドは、CUKY 型のフィールドを指します。	CHAR(*n*)
DATS	日付値。	DATE
DEC	符号付き固定小数点 10 進数。	CHAR(*n*)
FLTP	浮動小数点数。	DOUBLE

ABAP SAP データ型	説明	データ取得に使用される FedSQL データ型
INT1	非常に小さな符号付きの正確な整数。	TINYINT
INT2	小さな符号付きの正確な整数。	SMALLINT
INT4	通常の符号付きの正確な整数。	INTEGER
INT8	大きな符号付きの正確な整数。	BIGINT
LANG	特殊機能用の独自のフィールド形式を持つ言語キー。 変換 Exit ISOLA は、表示される値をデータベースの値に変換しますが、その逆が当てはまります。	非 Unicode SAP システムの場合は CHAR(*n*)、Unicode SAP システムの場合は NCHAR(*n*)
LCHR	固定長の文字列。	非 Unicode SAP システムの場合は VARCHAR(*n*)、Unicode SAP システムの場合は NVARCHAR(*n*)
LRAW	解釈されない可変長のバイト文字列。	VARBINARY(*n*)
NUMC	テキスト文字列。	非 Unicode SAP システムの場合は CHAR(*n*)、Unicode SAP システムの場合は NCHAR(*n*)
PREC	QUAN フィールドの精度。	CHAR(*n*)
QUAN	DEC フィールドに対応する数量。	CHAR(*n*)
RAW	解釈されないバイト文字列。	BINARY
TIMS	時間値。	TIME(*p*)
UNIT	QUAN データ型によって参照される単位キー。	非 Unicode SAP システムの場合は CHAR(*n*)、Unicode SAP システムの場合は NCHAR(*n*)
VARC	可変長の文字列データ。SAP リリース 3.0 以降、このデータ型のフィールドの作成はサポートされなくなりました。SELECT ステートメントの WHERE 条件を除いて、このデータ型の既存のフィールドを使用できます。	非 Unicode SAS システムの場合は VARCHAR(*n*)、Unicode SAP システムの場合は NVARCHAR(*n*)

SAP HANA のデータ型

次の表に、SAP HANA データベースのデータ型サポートを示します。

SAP HANA データ型に関するデータソース固有の情報については、SAP HANA データベースのドキュメントを参照してください。

注: この表の情報は、CAS で処理されるデータには適用されません。CAS にロードされたデータは、CAS データ型に変換されます。CAS データ型の変換については、*SAS Cloud Analytic Services: ユーザーガイド*の SAS Data Connector for SAP HANA のドキュメントを参照してください。

表 A7.22 FedSQL データ型と SAP HANA で使用されるデータ型のマッピング

データ型定義キーワード [1]	SAP HANA データ型	説明	返されるデータ型
[2]	ALPHANUM(*n*)	可変長の文字列。	NVARCHAR(*n*)
BIGINT	BIGINT	64 ビット整数。	BIGINT
BINARY(*n*)	BINARY(*n*)	固定長のバイナリ値。	BINARY(*n*)
[2]	BLOB	可変長のバイナリラージオブジェクト文字列。	[2]
CHAR(*n*)	CHAR(*n*)	可変長の文字列。	CHAR(*n*)
[2]	CLOB	可変長文字のラージオブジェクト文字列。	[2]
DATE	DATE	年、月、および日の値。	DATE
DECIMAL|NUMERIC(**p**,*s*)	DECIMAL(*p*,*s*)	符号付きの正確な固定小数点 10 進数。	DECIMAL(*p*,*s*)[3]
DOUBLE	DOUBLE	倍精度浮動小数点数。	DOUBLE
FLOAT(*p*)	FLOAT(*n*)	浮動小数点数。	FLOAT(*p*)
INTEGER	INTEGER	32 ビット整数。	INTEGER
NCHAR(*n*)	NCHAR(*n*)	固定長の Unicode 文字列。	NCHAR(*n*)

データ型定義キーワード [1]	SAP HANA データ型	説明	返されるデータ型
[2]	NCLOB	固定長文字のラージオブジェクト文字列。	[2]
NVARCHAR(*n*)	NVARCHAR(*n*)	可変長 Unicode 文字のラージオブジェクト文字列。	NVARCHAR(*n*)
REAL	REAL	浮動小数点数。	REAL
[2]	SECONDATE	日時値。	[2]
[2]	SHORTTEXT(*n*)	可変長の文字列。	NVARCHAR(*n*)
[2]	SMALLDECIMAL(*p*,*s*)	浮動小数点 10 進数。	DECIMAL(*p*,*s*)[3]
SMALLINT	SMALLINT	16 ビット整数。	SMALLINT
[2]	TEXT	可変長 Unicode 文字のラージオブジェクト文字列。	[2]
TIME(*p*)	TIME(*p*)	時間値。	TIME(*p*)
TIMESTAMP(*p*)	TIMESTAMP(*p*)	日時値。	TIMESTAMP(**p**)
TINYINT	TINYINT	符号なし 8 ビット整数。	TINYINT
VARBINARY(*n*)	VARBINARY(*n*)	可変長のバイナリ文字列。	VARBINARY(*n*)
VARCHAR(*n*)	VARCHAR(*n*)	可変長の文字列。	VARCHAR(*n*)

1 データ型のマッピング方法を制御する CT_PRESERVE=接続引数は、データ型を定義できるかどうかに影響する可能性があります。値 FORCE (デフォルト)および FORCE_COL_SIZE は、データ型を定義できるかどうかには影響しません。要求されたデータ型がデータソースにネイティブでない場合、または指定された精度またはスケールがデータソースの範囲内にない場合、値 STRICT および SAFE はエラーになる可能性があります。

2 SAP HANA データ型は定義できません。データが取得されると、ネイティブデータ型が同様のデータ型にマッピングされます。

3 FedSQL では、DECIMAL データ型は、データソースに完全に渡すことができる要求でサポートされています。データソースに完全に渡すことができない要求の DECIMAL 列は、DOUBLE として処理されるため、精度に影響を与える可能性があります。詳細については、“データ型” (14 ページ)を参照してください。

SAP IQ のデータ型

次の表に、SAP IQ データベースのデータ型サポートを示します。

SAP IQ データベースのデータ型に関するデータソース固有の情報については、SAP IQ データベースのドキュメントを参照してください。

注: このデータソースは CAS サーバーではサポートされていません。

表 A7.23 FedSQL データ型と SAP IQ で使用されるデータ型のマッピング

データ型定義キーワード [1]	SAP IQ データ型	説明	返されるデータ型
BIGINT	BIGINT	64 ビット整数。	BIGINT
BINARY(*n*)	BINARY(*n*)	固定長のバイナリ文字列。	BINARY(*n*)
[3]	BIT	値 0 または 1 のみを保存する整数。	[3]
CHAR(*n*)	CHAR(*n*)	可変長の文字列。	VARCHAR(*n*)
DATE	DATE	日付値。	DATE
DECIMAL|NUMERIC(*p*,*s*)	DECIMAL(*p*,*s*)	符号付きの正確な固定小数点 10 進数。	DECIMAL(*p*,*s*)[4]
DOUBLE	DOUBLE	倍精度浮動小数点数。	DOUBLE
FLOAT(*p*)	FLOAT(*p*)	浮動小数点数。	FLOAT(*p*)
INTEGER	INTEGER	32 ビット整数。	INTEGER
[2]	LONG BINARY	可変長のバイナリ文字列。	VARBINARY(*n*)
REAL	REAL	浮動小数点数。	REAL
SMALLINT	SMALLINT	16 ビット整数。	SMALLINT
TIME(*p*)	TIME(*p*)	時間値。	TIME(*p*)
TIMESTAMP(*p*)	TIMESTAMP(*p*)	日時値。	TIMESTAMP(*p*)
TINYINT	TINYINT	符号なし 8 ビット整数。	TINYINT
VARBINARY(*n*)	VARBINARY(*n*)	可変長のバイナリ文字列。	VARBINARY(*n*)
VARCHAR(*n*)	CHAR(*n*)	可変長の文字列。	VARCHAR(*n*)

1 データ型のマッピング方法を制御する CT_PRESERVE=接続引数は、データ型を定義できるかどうかに影響する可能性があります。値 FORCE (デフォルト)および FORCE_COL_SIZE は、データ型を定義できるかどうかには影響しません。要求され

たデータ型がデータソースにネイティブでない場合、または指定された精度またはスケールがデータソースの範囲内にない場合、値 STRICT および SAFE はエラーになる可能性があります。

2 SAP IQ データ型は定義できません。データが取得されると、ネイティブデータ型が同様のデータ型にマッピングされます。

3 SAP IQ データ型を定義または取得できません。

4 DECIMAL データ型は、データソースに完全に渡すことができる要求でサポートされています。データソースに完全に渡すことができない要求の DECIMAL 列は、DOUBLE として処理されるため、精度に影響を与える可能性があります。

SingleStore のデータ型

次の表に、SingleStore データベースのデータ型サポートを示します。

NCHAR、NVARCHAR、REAL、VARBINARY データ型は、データ型定義ではサポートされていません。

SingleStore データ型に関するデータソース固有の情報については、SingleStore データベースのドキュメントを参照してください。

注: この表の情報は、CAS で処理されるデータには適用されません。CAS にロードされたデータは、CAS データ型に変換されます。CAS データ型の変換については、*SAS Cloud Analytic Services: ユーザーガイド*の SAS Data Connector for SingleStore のドキュメントを参照してください。

表 A7.24 FedSQL データ型と SingleStore で使用されるデータ型のマッピング

データ型定義キーワード [1]	SingleStore データ型	説明	返されるデータ型
BIGINT	BIGINT	大きな符号付きの正確な整数。	BIGINT
BINARY(*n*)	BINARY(*n*)	固定長のバイナリ文字列。	BINARY(*n*)
[2]	BLOB	バイナリラージオブジェクト文字列。	[2]
CHAR(*n*)	CHAR(*n*)[3]	固定長の文字列。	CHAR(*n*)
DATE	DATE	日付値。	DATE
[2]	DATETIME	日時値。	[2]
DECIMAL|NUMERIC(*p*,*s*)	DECIMAL(*p*,*s*)	符号付き固定小数点 10 進数。	DECIMAL(*p*,*s*)[4]

データ型定義キーワード [1]	SingleStore データ型	説明	返されるデータ型
DOUBLE	DOUBLE	符号付き倍精度浮動小数点数。	DOUBLE
[2]	ENUM(*values*)	許可された値のリストからの文字値。	[2]
FLOAT(*p*)	FLOAT(*p*)	符号付きの単精度または倍精度の浮動小数点数。	FLOAT(*p*)
INTEGER	INT	通常の符号付きの正確な整数。	INTEGER
[2]	JSON	JSON 形式の可変長文字列。	VARCHAR
[2]	LONGBLOB	可変長のバイナリラージオブジェクト文字列。	[2]
[2]	LONGTEXT	可変長のテキスト文字列。	[2]
[2]	MEDIUMBLOB	可変長のバイナリラージオブジェクト文字列。	[2]
[2]	MEDIUMINT	通常の符号付きの正確な整数。	[2]
[2]	MEDIUMTEXT	可変長の文字列。	[2]
[2]	SET(*values*)	許可された値のリストからの文字値。	[2]
SMALLINT	SMALLINT	小さな符号付きの正確な整数。	SMALLINT
[2]	TEXT	テキスト文字列。	[2]
TIME(*p*)	TIME(*p*)	時間値。	TIME(*p*)
TIMESTAMP(*p*)	DATETIME(*p*)	日時値。	TIMESTAMP(*p*)
[2]	TINYBLOB	可変長のバイナリラージオブジェクト文字列。	[2]

データ型定義キーワード [1]	SingleStore データ型	説明	返されるデータ型
TINYINT	TINYINT	非常に小さな符号付きの正確な整数。	TINYINT
[2]	TINYTEXT	可変長のテキスト文字列。	[2]
VARCHAR(*n*)	VARCHAR(*n*)	可変長の文字列。	VARCHAR(*n*)

1 データ型のマッピング方法を制御する CT_PRESERVE=接続引数は、データ型を定義できるかどうかに影響する可能性があります。値 FORCE (デフォルト)および FORCE_COL_SIZE は、データ型を定義できるかどうかには影響しません。要求されたデータ型がデータソースにネイティブでない場合、または指定された精度またはスケールがデータソースの範囲内にない場合、値 STRICT および SAFE はエラーになる可能性があります。

2 SingleStore データ型は定義できません。データが取得されると、ネイティブデータ型が同様のデータ型にマッピングされます。

3 DS2 は、固定文字値の空白スペースを処理するために ANSI SQL 規則を使用します。MySQL 8 は、デフォルトで末尾の空白スペースを重要な値として扱います。この違いは、DS2 で FedSQL ステートメントを使用するときに実行される文字比較に干渉します。DS2 を使用して MySQL 8 CHAR()データに対して FedSQL ステートメントを発行する場合、データが正しく読み取られるように、FedSQL ステートメントで TRIM 関数を使用する必要があります。DS2 で FedSQL ステートメントがサポートされている場所については、“DS2 と FedSQL の使用” (*SAS DS2 プログラマガイド*)を参照してください。また、“例: MySQL 8 CHAR()データから末尾のスペースをフィルタリングする” (*SAS DS2 言語リファレンス*)も参照してください。MySQL 8 は、比較のデフォルト照合順序として utf8mb4_0900_ai_ci を使用します。

4 FedSQL では、DECIMAL データ型は、データソースに完全に渡すことができる要求でサポートされています。データソースに完全に渡すことができない要求の DECIMAL 列は、DOUBLE として処理されるため、精度に影響を与える可能性があります。詳細については、“データ型” (14 ページ)を参照してください。

Snowflake のデータ型

次の表に、Snowflake のデータ型サポートを示します。

FedSQL の NCHAR、NVARCHAR、および TINYINT データ型は、データ型定義ではサポートされていません。

Snowflake データ型に関するデータソース固有の情報については、Snowflake のドキュメントを参照してください。

注: この表の情報は、CAS で処理されるデータには適用されません。CAS にロードされたデータは、CAS データ型に変換されます。CAS データ型の変換については、*SAS Cloud Analytic Services: ユーザーガイド*の SAS Data Connector for Snowflake のドキュメントを参照してください。

表 A7.25　FedSQL データ型と Snowflake で使用されるデータ型のマッピング

データ型定義キーワード	Snowflake データ型	説明	返されるデータ型
[1]	ARRAY	値が VARIANT 型の値である、任意のサイズの密または疎な配列。	VARCHAR
BIGINT	BIGINT	64 ビット整数。	BIGINT
BINARY(*n*)	BINARY(*n*)	固定長のバイナリ文字列。	BINARY
[1]	BOOLEAN	true、false、または不明(null)の値。	BINARY
CHAR(*n*)	VARCHAR(1)	固定長の文字列。	VARCHAR
DATE	DATE	日付値。	DATE
DECIMAL|NUMERIC(*p*,*s*)	NUMBER(*p*,*s*)	符号付きの正確な固定小数点 10 進数。	DECIMAL(*p*,*s*)[2]
DOUBLE	FLOAT	倍精度浮動小数点数。	DOUBLE
FLOAT(*p*)	FLOAT、FLOAT4、FLOAT8	浮動小数点数。	FLOAT(*p*)
INTEGER	精度とスケールのない NUMBER。	32 ビット整数。	INTEGER
[1]	OBJECT	キーと値のペアのコレクション。キーは空でない文字列であり、値は VARIANT 型の値です。	VARCHAR
REAL	FLOAT	浮動小数点数。	FLOAT
SMALLINT	精度とスケールのない NUMBER。	16 ビット整数。	SMALLINT
TIME(*p*)	TIME(*p*)	時間値。	TIME(*p*)
TIMESTAMP(*p*)	TIMESTAMP、TIMESTAMP_LTZ、TIMESTAMP_NTZ、TIMESTAMP_TZ	日時値。	TIMESTAMP(*p*)

データ型定義キーワード	Snowflake データ型	説明	返されるデータ型
VARBINARY(*n*)	VARBINARY	可変長のバイナリ文字列。	BINARY
VARCHAR(*n*)	VARCHAR(*n*)	可変長の文字列。デフォルト(および最大)の長さは 16,777,216 です。	VARCHAR
[1]	VARIANT	OBJECT や ARRAY を含む他の Snowflake データ型の値を最大 16MB まで保存できるユニバーサルデータ型。	VARCHAR

1 Snowflake データ型は定義できません。データが取得されると、ネイティブデータ型が同様のデータ型にマッピングされます。

2 FedSQL では、DECIMAL データ型は、データソースに完全に渡すことができる要求でサポートされています。データソースに完全に渡すことができない要求の DECIMAL 列は、DOUBLE として処理されるため、精度に影響を与える可能性があります。詳細については、“データ型” (14 ページ)を参照してください。

Spark のデータ型

次の表に、Apache Spark のデータ型サポートを示します。

NCHAR および NVARCHAR データ型は、データ定義には使用できません。

Spark SQL データ型に関するデータソース固有の情報については、Spark SQL のドキュメントを参照してください。

注: この表の情報は、CAS で処理されるデータには適用されません。CAS にロードされたデータは、CAS データ型に変換されます。CAS データ型の変換については、*SAS Cloud Analytic Services: ユーザーガイド*の SAS Data Connector for Spark のドキュメントを参照してください。

表 A7.26 FedSQL データ型と Apache Spark で使用されるデータ型のマッピング

データ型定義キーワード	Spark データ型	説明	返されるデータ型
[1]	ARRAY<*data-type*>	整数の配列(インデックス可能なリスト)。	STRING[5, 7]
BIGINT	BIGINT	-9,223,372,036,854,775,808 から	BIGINT

データ型定義キーワード	Spark データ型	説明	返されるデータ型
		9,223,372,036,854,775,807 までの符号付き 8 バイト整数。	
BINARY(*n*)	BINARY	最大 32K の可変長のバイナリ文字列。	BINARY
[1]	BOOLEAN	テキストの true または false の値。	TINYINT
CHAR(*n*)	CHAR(*n*)[9]	最大 255 文字の文字列。[3]	CHAR
DATE	DATE[2]	ANSI SQL の日付値。	DATE
DECIMAL/ NUMERIC(*p*,*s*)	DECIMAL	38 桁の精度の固定小数点 10 進数。	DECIMAL(*p*,*s*)[4]
DOUBLE	DOUBLE	8 バイトの倍精度浮動小数点数。	DOUBLE
FLOAT(*p*)	FLOAT	4 バイトの単精度浮動小数点数。	DOUBLE
INTEGER	INTEGER	符号付き 4 バイト整数。	INTEGER
[1]	MAP<*primitive-type*, *data-type*>	キーと値のペアの連想配列。	STRING[5, 7]
REAL	DOUBLE	64 ビットの倍精度浮動小数点数。	DOUBLE
SMALLINT	SMALLINT	-32,768 から 32,767 までの符号付き 2 バイト整数。	SMALLINT
[1]	STRING	可変長の文字列。	VARCHAR(*n*)[5, 6]
[1]	STRUCT<*col-name*: *data-type*>	確立された列要素とデータ型を持つ構造。列要素とデータ型は、ダブルドット(:)表記を使用してマッピングされます。	STRING[5, 7]
TIME(*p*)	[8]	時間値。	STRING

データ型定義キーワード	Spark データ型	説明	返されるデータ型
TIMESTAMP(*p*)	TIMESTAMP	オプションのナノ秒精度の UNIX タイムスタンプ。	TIMESTAMP[(*p*)]
[1]	TIMESTAMP_NTZ[10]	タイムゾーンなしのタイムスタンプ。	TIMESTAMP
TINYINT	TINYINT	-128 から 127 までの符号付き 1 バイト整数。	TINYINT
VARBINARY	BINARY	最大 32K の可変長のバイナリ文字列。	BINARY
VARCHAR(*n*)	VARCHAR(*n*)[9]	可変長の文字列。	VARCHAR(*n*)[5]

1 Spark データ型は定義できません。データが取得されると、ネイティブデータ型が同様のデータ型にマッピングされます。

2 サポートされている日付値は、0001 年 1 月 1 日から 9999 年 12 月 31 日までです。9999 より後の日付値は null 値として読み戻されます。

3 255 文字を超える値で CHAR を指定すると、列はかわりに VARCHAR(*n*)として作成されます。

4 FedSQL では、DECIMAL データ型は、データソースに完全に渡すことができる要求でサポートされています。データソースに完全に渡すことができない要求の DECIMAL 列は、DOUBLE として処理されるため、精度に影響を与える可能性があります。詳細については、“データ型” (14 ページ)を参照してください。

5 VARCHAR(*n*)および Hive 複合型の最大長は、DBMAX_TEXT=データソース接続オプションによって決定されます。

6 SASFMT TableProperties は、STRING 列を読み取るときに適用されます。

7 複合型 ARRAY、MAP、STRUCT は、基になる複合型の STRING 表現として読み取られます。ARRAY 値は、角かっこで囲まれて読み戻されます(例: [1、2、4])。STRUCT と MAP の値は、中かっこで囲まれて読み戻されます。例: {"firstname":"robert","nickname":"bob"}。

8 Spark は TIME(*p*)データ型をサポートしていません。データが Spark から読み取られると、SAS TIME8.出力形式を指定する SASFMT TableProperties が定義されている STRING 列が、TIME(*p*)データ型に変換されます。TIME 型がデータ定義に使用される場合、SASFMT TableProperties を使用して STRING 列にマッピングされます。秒の端数は保持されません。SASFMT TableProperties の詳細については、*SAS/ACCESS for Relational Databases: リファレンス*の"SAS Table Properties for Spark and Hadoop"を参照してください。

9 Apache Spark 2.0.0 では、Spark テーブルで許可される列の最大数は https://issues.apache.org/jira/browse/SPARK-18016 によって制限される場合があります。

10 TIMESTAMP_NTZ 型は Databricks でのみサポートされ、Databricks Runtime 13.3 LTS 以上を使用している場合にサポートされます。

Teradata のデータ型

次の表に、Teradata データベースのデータ型サポートを示します。

NCHAR、NVARCHAR、および REAL データ型は、データ型定義ではサポートされていません。

Teradata データ型に関するデータソース固有の情報については、Teradata データベースのドキュメントを参照してください。

注: この表の情報は、CAS で処理されるデータには適用されません。CAS にロードされたデータは、CAS データ型に変換されます。CAS データ型の変換については、*SAS Cloud Analytic Services: ユーザーガイド*の SAS Data Connector for Teradata のドキュメントを参照してください。

表 A7.27 FedSQL データ型と Teradata で使用されるデータ型のマッピング

データ型定義キーワード [1]	Teradata データ型	説明	返されるデータ型
BIGINT	BIGINT	大きな符号付きの正確な整数。	BIGINT
BINARY(*n*)	BYTE(*n*)	固定長のバイナリ文字列。	BINARY(*n*)
[2]	BLOB	ラージバイナリオブジェクト。	[2]
CHAR(*n*)	CHAR(*n*)	固定長の文字列。	CHAR(*n*)
[2]	CLOB	ラージ文字オブジェクト。	[2]
DATE	DATE	日付値。	DATE
DECIMAL(*p*,*s*)	DECIMAL(*p*,*s*)	符号付き固定小数点 10 進数。	DECIMAL(*p*,*s*)[3]
NUMERIC(*p*,*s*)	NUMBER(*p*,*s*)	符号付き可変長小数点 10 進数。	NUMERIC(*p*,*s*)[3]
DOUBLE	FLOAT	符号付き倍精度浮動小数点数。	DOUBLE
FLOAT(*p*)	FLOAT(*p*)	符号付き倍精度浮動小数点数。	FLOAT(*p*)
INTEGER	INTEGER	通常の符号付きの正確な整数。	INTEGER
[2]	LONG VARCHAR	可変長の文字列。	[2]
SMALLINT	SMALLINT	小さな符号付きの正確な整数。	SMALLINT
TIME(*p*)	TIME(*p*)	時間値。	TIME(*p*)
TIMESTAMP(*p*)	TIMESTAMP(*p*)	日時値。	TIMESTAMP(*p*)
TINYINT	BYTEINT	非常に小さな符号付きの正確な整数。	TINYINT
VARBINARY(*n*)	VARBYTE(*n*)	可変長のバイナリ文字列。	VARBINARY(*n*)
VARCHAR(*n*)	VARCHAR(*n*)	可変長の文字列。	VARCHAR(*n*)

1 データ型のマッピング方法を制御する CT_PRESERVE=接続引数は、データ型を定義できるかどうかに影響する可能性があります。値 FORCE (デフォルト)および FORCE_COL_SIZE は、データ型を定義できるかどうかには影響しません。要求され

たデータ型がデータソースにネイティブでない場合、または指定された精度またはスケールがデータソースの範囲内にない場合、値 STRICT および SAFE はエラーになる可能性があります。

2 Teradata データ型は定義できません。データが取得されると、ネイティブデータ型が同様のデータ型にマッピングされます。

3 FedSQL では、DECIMAL および NUMERIC データ型は、データソースに完全に渡すことができる要求でサポートされています。データソースに完全に渡すことができない要求の DECIMAL および NUMERIC 列は、DOUBLE として処理されるため、精度に影響を与える可能性があります。詳細については、“データ型” (14 ページ)を参照してください。

Vertica のデータ型

次の表に、Vertica データベースのデータ型サポートを示します。

NCHAR および NVARCHAR データ型は、データ定義ではサポートされていません。

Vertica データ型に関するデータソース固有の情報については、Vertica データベースのドキュメントを参照してください。

注: この表の情報は、CAS で処理されるデータには適用されません。CAS にロードされたデータは、CAS データ型に変換されます。CAS データ型の変換については、*SAS Cloud Analytic Services: ユーザーガイド*の SAS Data Connector for Vertica のドキュメントを参照してください。

表 A7.28 FedSQL データ型と Vertica で使用されるデータ型のマッピング

データ型定義キーワード [1]	Vertica SQL 識別子	説明	返されるデータ型
BIGINT	BIGINT	符号付き 64 ビット整数。8 バイトのストレージが必要です。	BIGINT
BINARY(*n*)	BINARY(*n*)	固定長のバイナリ文字列。	BINARY(*n*)
[2]	BOOLEAN	論理ブール(true/false)値。	[2]
[2]	BYTEA	可変長のバイナリ文字列。	[2]
CHAR(*n*)	CHAR(*n*)	固定長の文字列。	CHAR(*n*)
DATE	DATE	日付値。	DATE

データ型定義キーワード[1]	Vertica SQL 識別子	説明	返されるデータ型
[2]	DATETIME	タイムゾーン付きまたはタイムゾーンなしの日時値。	TIMESTAMP(*p*)
DECIMAL|NUMERIC(*p*,*s*)	DECIMAL|NUMERIC(*p*,*s*)	符号付きの固定精度およびスケール数。デフォルトの精度とスケール: 37, 15。	DECIMAL|NUMERIC(*p*,*s*)[3]
DOUBLE	DOUBLE PRECISION	符号付き 64 ビット IEEE 浮動小数点数。8 バイトのストレージが必要です。	DOUBLE
FLOAT(*p*)	FLOAT(*n*)、FLOAT8	符号付き 64 ビット IEEE 浮動小数点数。8 バイトのストレージが必要です。	DOUBLE
INTEGER	INTEGER	符号付き 64 ビット整数。8 バイトのストレージが必要です。	BIGINT
[2]	INTERVAL	期間。	[2]
[2]	INTERVAL DAY TO SECOND	期間。	[2]
[2]	INTERVAL YEAR TO MONTH	期間。	[2]
[2]	LONG VARCHAR	空間データなどの可変長の生のバイト値。	[2]
[2]	LONG BINARY	長い可変長のバイナリ文字列。	[2]
[2]	MONEY(*p*,*s*)	符号付きの固定の精度とスケールの金銭または通貨の値。デフォルトの精度とスケール: 18, 4。	[2]
[2]	NUMBER(*p*,*s*)	符号付きの固定精度およびスケール数。デフォルトの精度とスケール: 38,0。	[2]

データ型定義キーワード [1]	Vertica SQL 識別子	説明	返されるデータ型
[2]	RAW	可変長のバイナリ文字列。	[2]
REAL	REAL	符号付き 64 ビット IEEE 浮動小数点数。8 バイトのストレージが必要です。	DOUBLE
[2]	SMALLDATETIME	タイムゾーン付きまたはタイムゾーンなしの日時値。	TIMESTAMP(*p*)
SMALLINT	SMALLINT	符号付き 64 ビット整数。8 バイトのストレージが必要です。	BIGINT
TIME(*p*)	TIME(*p*)	タイムゾーンのない時間値。	TIME(*p*)
[2]	TIMETZ	タイムゾーン付き時間値。	TIME(*p*)
TIMESTAMP(*p*)	TIMESTAMP	タイムゾーンのない日時値。	TIMESTAMP(*p*)
[2]	TIMESTAMPTZ	タイムゾーン付き日時値。	TIMESTAMP(*p*)
TINYINT	TINYINT	符号付き 64 ビット整数。8 バイトのストレージが必要です。	BIGINT
VARBINARY(*n*)	VARBINARY	可変長のバイナリ文字列。	VARBINARY(*n*)
VARCHAR(*n*)	VARCHAR(*n*)、TEXT	可変長の文字列。	VARCHAR(*n*)

1 データ型のマッピング方法を制御する CT_PRESERVE=接続引数は、データ型を定義できるかどうかに影響する可能性があります。値 FORCE (デフォルト)および FORCE_COL_SIZE は、データ型を定義できるかどうかには影響しません。要求されたデータ型がデータソースにネイティブでない場合、または指定された精度またはスケールがデータソースの範囲内にない場合、値 STRICT および SAFE はエラーになる可能性があります。

2 Vertica データ型は定義できません。データが取得されると、ネイティブデータ型が同様のデータ型にマッピングされます。

3 FedSQL では、DECIMAL データ型は、データソースに完全に渡すことができる要求でサポートされています。データソースに完全に渡すことができない要求の DECIMAL 列は、DOUBLE として処理されるため、精度に影響を与える可能性があります。詳細については、“データ型” (14 ページ)を参照してください。

Yellowbrick のデータ型

次の表に、Yellowbrick データベースのデータ型サポートを示します。

BINARY、NCHAR、NVARCHAR、TINYINT、VARBINARY データ型は、データ型定義ではサポートされていません。

Yellowbrick データ型に関するデータソース固有の情報については、Yellowbrick データベースのドキュメントを参照してください。

注: この表の情報は、CAS で処理されるデータには適用されません。CAS にロードされたデータは、CAS データ型に変換されます。CAS データ型の変換については、*SAS Cloud Analytic Services: ユーザーガイド*の SAS Data Connector for Yellowbrick のドキュメントを参照してください。

表 A7.29 FedSQL データ型と Yellowbrick で使用されるデータ型のマッピング

データ型定義キーワード [1]	Yellowbrick データ型	説明	返されるデータ型
BIGINT	BIGINT	大きな符号付きの正確な整数または符号付き 8 バイト整数。	BIGINT
[2]	BOOLEAN	論理ブール(true/false)値。	[2]
CHAR(*n*)	CHAR(*n*)	固定長の文字列。	CHAR(*n*)
DATE	DATE	日付値。	DATE
DECIMAL|NUMERIC(*p*,*s*)	NUMERIC(*p*,*s*)	符号付き固定小数点 10 進数。	DECIMAL|NUMERIC(*p*,*s*)[3]
DOUBLE	DOUBLE PRECISION	符号付き倍精度浮動小数点数。	DOUBLE
FLOAT(*p*)	DOUBLE PRECISION、FLOAT、FLOAT8	符号付きの単精度または倍精度の浮動小数点数。	FLOAT
INTEGER	INTEGER	通常の符号付きの正確な整数。	INTEGER
[2]	MACADDR、MACADDR8	メディアアクセスコントロールアドレス。	[2]

データ型定義キーワード [1]	Yellowbrick データ型	説明	返されるデータ型
REAL	REAL、FLOAT4	符号付き単精度浮動小数点数。	REAL
SMALLINT	SMALLINT	小さな符号付きの正確な整数。	SMALLINT
TIME(*p*)	TIME(*p*)	時間値。	TIME(*p*)
TIMESTAMP(*p*)	TIMESTAMP(*p*)	日時値。	TIMESTAMP(*p*)
[2]	UUID	ユニバーサル一意識別子。	[2]
VARCHAR(*n*)	CHARACTER VARYING(*n*)	可変長の文字列。	VARCHAR(*n*)

1 データ型のマッピング方法を制御する CT_PRESERVE=接続引数は、データ型を定義できるかどうかに影響する可能性があります。値 FORCE (デフォルト)および FORCE_COL_SIZE は、データ型を定義できるかどうかには影響しません。要求されたデータ型がデータソースにネイティブでない場合、または指定された精度またはスケールがデータソースの範囲内にない場合、値 STRICT および SAFE はエラーになる可能性があります。

2 Yellowbrick データ型は定義できません。データが取得されると、ネイティブデータ型が同様のデータ型にマッピングされます。

3 FedSQL では、DECIMAL データ型は、データソースに完全に渡すことができる要求でサポートされています。データソースに完全に渡すことができない要求の DECIMAL 列は、DOUBLE として処理されるため、精度に影響を与える可能性があります。詳細については、“データ型” (14 ページ)を参照してください。

付録 3

FedSQL と DS2 の使用

DS2 から FedSQL ステートメントを動的に実行する **1253**

DS2 から FedSQL ステートメントを動的に実行する

注: DS2 アクションまたは PROC DS2 の SET ステートメントを除いて、DS2 プログラム内での FedSQL クエリの使用は、現在 CAS サーバーではサポートされていません。

DS2 プログラム内から FedSQL ステートメントを埋め込んで実行できます。次の場合に FedSQL を DS2 で使用できます。

- DS2 パッケージメソッド式を FedSQL SELECT ステートメントの関数として呼び出すことができます。

  詳細については、“式での DS2 パッケージの使用” (258 ページ)を参照してください。

- SQLSTMT パッケージを使用して、FedSQL ステートメントを生成、準備、および実行し、実行時にテーブルから行を作成、選択、変更、挿入、削除できます。

  SQLSTMT パッケージは、複数回実行される FedSQL ステートメント、パラメーターを含むステートメント、または結果セットを生成するステートメントで使用することを目的としています。詳細については、“SQLSTMT パッケージの使用” (*SAS DS2 プログラマガイド*)を参照してください。

- また、SQLEXEC 関数を使用して、FedSQL ステートメントを生成、準備、および実行し、実行時にテーブルから行を作成、選択、変更、挿入、削除できます。

  SQLEXEC 関数は、1 回だけ実行され、パラメーターを持たず、結果セットを生成しないステートメントで使用することを目的としています。

詳細については、“SQLEXEC 関数” (*SAS DS2 言語リファレンス*)を参照してください。

- DECLARE PACKAGE ステートメントまたは DATASET メソッドで FedSQL SELECT ステートメントを使用して、実行時にデータをハッシュインスタンスにロードできます。

  詳細については、“FedSQL クエリとハッシュインスタンスを使用して実行時に動的に行を取得する” (*SAS DS2 プログラマガイド*)、“DATASET メソッド” (*SAS DS2 言語リファレンス*)および“DECLARE PACKAGE ステートメント: ハッシュパッケージ” (*SAS DS2 言語リファレンス*)を参照してください。

- SET ステートメントを使用して、FedSQL SELECT ステートメントでデータを読み込むことができます。

  詳細については、“FedSQL が埋め込まれた SET ステートメント” (*SAS DS2 プログラマガイド*)および“SET ステートメント” (*SAS DS2 言語リファレンス*)を参照してください。

付録 4

例で使用されているテーブル

概要 1256
AFewWords 1256
Countries 1256
Customers 1258
CustonLine 1259
Densities 1260
Depts 1261
Employees 1262
EmpSales 1263
GrainProducts 1264
Houses 1265
Integers 1265
Numbers1 1266
Numbers2 1267
Numbers3 1267
Payroll 1268
Products 1270
Sales 1271
Staff 1272
Stores 1274
Survey 1275
WorldCityCoords 1276
WorldTemps 1277

概要

このセクションでは、本書で提供される例で使用されるテーブルを作成するためのコードを提供します。各テーブルについては、参照のためにテーブルのコンテンツのすべてまたは一部が表示されます。

AFewWords

AFewWords テーブルは、“LOWCASE 関数” (652 ページ)、 “SUBSTRING 関数” (844 ページ)、 “TRIM 関数” (866 ページ)、“UPCASE 関数” (869 ページ)で使用されます。

注: このサンプルコードでは、FedSQL ステートメントは PROC FEDSQL を使用して SAS に送信されます。

```
proc fedsql;
  create table afewwords(Word1 char(10), Word2 char(10));
  insert into afewwords values ('*some/', 'WHERE');
  insert into afewwords values ('*every*', 'THING');
  insert into afewwords values ('*no*', 'BODY');
quit;
```

アウトプット A9.1　テーブル AFewWords のコンテンツ

WORD1	WORD2
*some/	WHERE
every	THING
no	BODY

Countries

Countries テーブルは、“テーブルに値を挿入する” (1177 ページ)セクションの例で使用されています。ランダムに選択された国の人口と、各国が国連に参加した日付が含まれています。

```
data countries;
```

```
  infile datalines delimiter='#' linesize=80;
  length Name $35 Capital $35 Continent $30;
  input Name
    Capital
    Population
    Area
    Continent
    UNDate date9.;
  format Name Capital $35. Continent $30. Population Area best8.
    UNDate date9.;
  label Capital="Capital" Population="Population" Continent="Continent";
datalines;
Argentina#Buenos Aires#34248705#1073518#South America#24OCT1945
Bangladesh#Dhaka#1.2639E8#57300#Asia#01MAY1974
Brazil#Brasilia#1.6031E8#3286500#South America#24OCT1945
China#Beijing#1.2022E9#3696100#Asia#24OCT1945
Colombia#Bogota#35930188#440800#South America#24OCT1945
Congo, Democratic Republic of#Kinshasa#43106529#905400#Africa#12OCT1960
Egypt#Cairo#59912259#385200#Africa#24OCT1945
England#London#49293170#50400#Europe#01JAN1945
Ethiopia#Addis Ababa#59291170#437800#Africa#24OCT1945
France#Paris#58412558#210000#Europe#24OCT1945
Germany#Berlin#81890690#137700#Europe#01JAN1973
India#New Delhi#9.2901E8#1222600#Asia#24OCT1945
Indonesia#Jakarta#2.0239E8#741100#Asia#01NOV1950
Iran#Tehran#66261493#632500#Asia#24OCT1945
Italy#Rome#58713508#116300#Europe#01MAR1955
Japan#Tokyo#1.2635E8#145900#Asia#01JUN1956
Korea, South#Seoul#45529277#38300#Asia#01JAN1991
Mexico#Mexico City#93114708#756100#North America#24OCT1945
Myanmar#Yangon#44715298#261200#Asia#01OCT1948
Nigeria#Abuja#99062003#356700#Africa#31AUG1960
Pakistan#Islamabad#1.2306E8#339700#Asia#10NOV1947
Philippines#Manila#70500039#115900#Asia#24OCT1945
Poland#Warsaw#39037645#120700#Europe#24OCT1945
Russia#Moscow#1.5109E8#6592800#Europe#24OCT1945
South Africa#Cape Town#44365873#473300#Africa#24OCT1945
Spain#Madrid#39692061#194900#Europe#15DEC1955
Thailand#Bangkok#60099089#198100#Asia#13OCT1946
Turkey#Ankara#62769263#300948#Europe#24OCT1945
Ukraine#Kiev#52360233#233100#Europe#24OCT1945
United States#Washington#2.6329E8#3787318#North America#24OCT1945
Vietnam#Hanoi#73827657#127200#Asia#01JAN1977
;
run;
```

アウトプット A9.2　テーブル Countries のコンテンツ(部分的)

Name	Capital	Continent	Population	Area	UNDate
Argentina	Buenos Aires	South America	34248705	1073518	24OCT1945
Bangladesh	Dhaka	Asia	1.2639E8	57300	01MAY1974
Brazil	Brasilia	South America	1.6031E8	3286500	24OCT1945
China	Beijing	Asia	1.2022E9	3696100	24OCT1945
Colombia	Bogota	South America	35930188	440800	24OCT1945
Congo, Democratic Republic of	Kinshasa	Africa	43106529	905400	12OCT1960
Egypt	Cairo	Africa	59912259	385200	24OCT1945
England	London	Europe	49293170	50400	01JAN1945
Ethiopia	Addis Ababa	Africa	59291170	437800	24OCT1945
France	Paris	Europe	58412558	210000	24OCT1945
Germany	Berlin	Europe	81890690	137700	01JAN1973
India	New Delhi	Asia	9.2901E8	1222600	24OCT1945
Indonesia	Jakarta	Asia	2.0239E8	741100	01NOV1950
Iran	Tehran	Asia	66261493	632500	24OCT1945
Italy	Rome	Europe	58713508	116300	01MAR1955
Japan	Tokyo	Asia	1.2635E8	145900	01JUN1956
Korea, South	Seoul	Asia	45529277	38300	01JAN1991
Mexico	Mexico City	North America	93114708	756100	24OCT1945
Myanmar	Yangon	Asia	44715298	261200	01OCT1948
Nigeria	Abuja	Africa	99062003	356700	31AUG1960

Customers

Customers テーブルは、“結合操作” (30 ページ)、“CREATE TABLE ステートメント” (939 ページ)、および“INSERT ステートメント” (972 ページ)で使用されます。

注: このサンプルコードでは、FedSQL ステートメントは PROC FEDSQL を使用して SAS に送信されます。

```
proc fedsql;
  create table customers
    ( custid double primary key,
    name char(16),
    address char(64),
```

```
  city char(16),
  state char(2),
  country char(16),
  phone char(16),
  initorder date
  );

 insert into customers values (1,'Peter Frank', '300 Rock Lane', 'Boulder', 'CO',
 'United States', '3039564321', date '2012-01-14');
 insert into customers values (2,'Jim Stewart', '1500 Lapis Lane', 'Little Rock',
 'AR', 'United States', '8705553978', date '2012-03-20');
 insert into customers values (3,'Janet Chien', '75 Jujitsu', 'Nagasaki', '',
 'Japan', '01181956879932', date '2012-06-07');
 insert into customers values (4,'Qing Ziao', '10111 Karaje', 'Tokyo', '',
 'Japan', '0118136774351', date '2012-10-12');
 insert into customers values (5,'Humberto Sertu', '876 Avenida Blanca', 'Buenos Aires',
 '', 'Argentina','01154118435029', date '2012-12-15');
quit;
```

アウトプット A9.3　テーブル Customers のコンテンツ

CUSTID	NAME	ADDRESS	CITY	STATE	COUNTRY	PHONE	INITORDER
1	Peter Frank	300 Rock Lane	Boulder	CO	United States	3039564321	14JAN2012
2	Jim Stewart	1500 Lapis Lane	Little Rock	AR	United States	8705553978	20MAR2012
3	Janet Chien	75 Jujitsu	Nagasaki		Japan	01181956879932	07JUN2012
4	Qing Ziao	10111 Karaje	Tokyo		Japan	0118136774351	12OCT2012
5	Humberto Sertu	876 Avenida Blanca	Buenos Aires		Argentina	01154118435029	15DEC2012

CustonLine

CustonLine テーブルは、“CAST 関数” (332 ページ)、“DAY 関数” (443 ページ)、“HOUR 関数” (570 ページ)、“MINUTE 関数” (670 ページ)、“MONTH 関数” (676 ページ)、“SECOND 関数” (824 ページ)、“YEAR 関数” (887 ページ)で使用されます。

注: このサンプルコードでは、FedSQL ステートメントは PROC FEDSQL を使用して SAS に送信されます。

```
proc fedsql;
  create table custonline(custnum varchar(15) having label 'Customer Number',
     begintime timestamp having label 'Begin Time',
     endtime timestamp having label 'End Time' format datetime23.3);

  insert into custonline values ('US-C-37533944', timestamp'2013-09-01 10:00:00',
  timestamp'2013-09-01 10:05:01.253');
  insert into custonline values ('GB-W-33944332', timestamp'2013-10-02 22:15:33',
  timestamp'2013-10-02 22:21:09.421');
  insert into custonline values ('SP-M-29443992', timestamp'2013-10-15 18:44:25',
```

```
    timestamp'2013-10-15 19:04:55.746');
    insert into custonline values ('US-A-37144324', timestamp'2013-11-01 12:03:59',
    timestamp'2013-11-01 12:25:09.398');
    insert into custonline values ('FR-P-98384488', timestamp'2013-12-01 12:15:34',
    timestamp'2013-12-01 12:47:45.221');
    insert into custonline values ('GB-L-24995559', timestamp'2013-01-02 15:43:24',
    timestamp'2013-01-02 16:06:15.766');
    insert into custonline values ('FR-L-42339887', timestamp'2013-01-16 14:55:00',
    timestamp'2013-01-16 15:05:56.288');
    insert into custonline values ('GB-P-87559899', timestamp'2013-02-01 11:02:44',
    timestamp'2013-02-01 11:15:33.955');
    insert into custonline values ('SP-N-44333958', timestamp'2013-03-01 10:14:33',
    timestamp'2013-03-01 10:35:27.908');
    insert into custonline values ('GB-R-24994990', timestamp'2013-03-15 09:06:00',
    timestamp'2013-03-15 09:06:20.475');
quit;
```

アウトプット A9.4　テーブル CustonLine のコンテンツ

Customer Number	Begin Time	End Time
US-C-37533944	01SEP13:10:00:00.00	01SEP2013:10:05:01.253
GB-W-33944332	02OCT13:22:15:33.00	02OCT2013:22:21:09.421
SP-M-29443992	15OCT13:18:44:25.00	15OCT2013:19:04:55.746
US-A-37144324	01NOV13:12:03:59.00	01NOV2013:12:25:09.398
FR-P-98384488	01DEC13:12:15:34.00	01DEC2013:12:47:45.221
GB-L-24995559	02JAN13:15:43:24.00	02JAN2013:16:06:15.766
FR-L-42339887	16JAN13:14:55:00.00	16JAN2013:15:05:56.288
GB-P-87559899	01FEB13:11:02:44.00	01FEB2013:11:15:33.955
SP-N-44333958	01MAR13:10:14:33.00	01MAR2013:10:35:27.908
GB-R-24994990	15MAR13:09:06:00.00	15MAR2013:09:06:20.475

Densities

Densities テーブルは、“スカラーサブクエリの例” (53 ページ)、および“AVG 関数” (311 ページ)、“CSS 関数” (423 ページ)、“FLOOR 関数” (542 ページ)、“MAX 関数” (663 ページ)、“MIN 関数” (668 ページ)、“POWER 関数” (762 ページ)、“PROBT 関数” (787 ページ)、“STUDENTS_T 関数” (840 ページ)、“SUM 関数” (846 ページ)、“USS 関数” (873 ページ)、“VARIANCE 関数” (877 ページ)で使用されます。このテーブルは、“LIKE 述語” (915 ページ)でも使用されます。

注: このサンプルコードでは、FedSQL ステートメントは PROC FEDSQL を使用して SAS に送信されます。

```
proc fedsql;
  create table densities(Name char(20),
    Population double having format comma12.,
    SquareMiles double having format comma12.,
    Density double);

  insert into densities values ('Afghanistan', 17070323, 251825, 67.79);
  insert into densities values ('Albania', 3407400, 11100, 306.97);
  insert into densities values ('Algeria', 28171132, 919595, 30.63);
  insert into densities values ('Andorra', 64634, 200, 323.17);
  insert into densities values ('Angola', 9901050, 481300, 20.57);
  insert into densities values ('Antigua and Bar', 65644, 171, 383.88);
  insert into densities values ('Argentina', 34248705, 1073518, 31.90);
  insert into densities values ('Armenia', 3556864, 11500, 309.29);
  insert into densities values ('Australia', 18255944, 2966200, 6.15);
  insert into densities values ('Austria', 8033746, 32400, 247.96);
quit;
```

アウトプット A9.5　テーブル Densities のコンテンツ

NAME	POPULATION	SQUAREMILES	DENSITY
Afghanistan	17,070,323	251,825	67.79
Albania	3,407,400	11,100	306.97
Algeria	28,171,132	919,595	30.63
Andorra	64,634	200	323.17
Angola	9,901,050	481,300	20.57
Antigua and Bar	65,644	171	383.88
Argentina	34,248,705	1,073,518	31.9
Armenia	3,556,864	11,500	309.29
Australia	18,255,944	2,966,200	6.15
Austria	8,033,746	32,400	247.96

Depts

Depts テーブルは、“相関サブクエリの例” (52 ページ)および“スカラーサブクエリの例” (53 ページ)で使用されます。

注: このサンプルコードでは、FedSQL ステートメントは PROC FEDSQL を使用して SAS に送信されます。

```
proc fedsql;
  create table depts(deptno double,
    deptname char(30) not null,
```

```
    manager char(30)
  );

  insert into depts values (10, 'Sales', 'Jim Barnes');
  insert into depts values (20, 'Research', 'Clifford James');
  insert into depts values (30, 'Accounting', 'Barbara Sandman');
  insert into depts values (40, 'Operations', 'William Baylor');
quit;
```

アウトプット A9.6　テーブル Depts のコンテンツ

DEPTNO	DEPTNAME	MANAGER
10	Sales	Jim Barnes
20	Research	Clifford James
30	Accounting	Barbara Sandman
40	Operations	William Baylor

Employees

Employees テーブルは、“相関サブクエリの例” (52 ページ)および“スカラーサブクエリの例” (53 ページ)で使用されます。

注: このサンプルコードでは、FedSQL ステートメントは PROC FEDSQL を使用して SAS に送信されます。

```
proc fedsql;
  create table employees
    (empid double,
     dept double,
     emp_name char(30),
     pos char(50),
     hire_date date
    );

  insert into employees values (1, 10, 'Jim Barnes', 'Manager',
  date '2000-11-26');
  insert into employees values (2, 20, 'Clifford James', 'Manager',
  date '2000-11-26');
  insert into employees values (3, 30, 'Barbara Sandman', 'Manager',
  date '2000-11-26');
  insert into employees values (4, 40, 'William Baylor', 'Manager',
  date '2000-11-26');
  insert into employees values (5, 20, 'Greg Welty', 'Developer',
  date '2004-11-26');
  insert into employees values (6, 20, 'Penny Jackson', 'Developer',
  date '2004-11-26');
  insert into employees values (7, 10, 'Edward Murray', 'Sales Associate',
```

```
  date '2004-11-26');
  insert into employees values (8, 10, 'Ronald Thomas', 'Sales Associate',
  date '2004-11-26');
  insert into employees values (9, 30, 'Elsie Marks', 'Executive Assistant',
  date '2005-02-11');
  insert into employees values (10, 40, 'Bruno Kramer', 'Grounds support technician',
  date '2005-11-02');
quit;
```

アウトプット A9.7　テーブル Employees のコンテンツ

EMPID	DEPT	EMP_NAME	POS	HIRE_DATE
1	10	Jim Barnes	Manager	26NOV2000
2	20	Clifford James	Manager	26NOV2000
3	30	Barbara Sandman	Manager	26NOV2000
4	40	William Baylor	Manager	26NOV2000
5	20	Greg Welty	Developer	26NOV2004
6	20	Penny Jackson	Developer	26NOV2004
7	10	Edward Murray	Sales Associate	26NOV2004
8	10	Ronald Thomas	Sales Associate	26NOV2004
9	30	Elsie Marks	Executive Assistant	11FEB2005
10	40	Bruno Kramer	Grounds support technician	02NOV2005

EmpSales

EmpSales テーブルは、“CALCULATED キーワードを指定する PROC SQL クエリを書き換える” (1184 ページ)で使用されます。

```
data EmpSales;
input Salesperson $ January February March;
datalines;
Smith 1000 650 800
Johnson 0 900 900
Reed 1200 700 850
Davis 1050 900 1000
Thompson 750 850 1000
Peterson 900 600 500
Jones 800 900 1200
Murphy 700 800 700
Garcia 400 1200 1150
;
```

アウトプット A9.8　テーブル EmpSales のコンテンツ

Salesperson	January	February	March
Smith	1000	650	800
Johnson	0	900	900
Reed	1200	700	850
Davis	1050	900	1000
Thompson	750	850	1000
Peterson	900	600	500
Jones	800	900	1200
Murphy	700	800	700
Garcia	400	1200	1150

GrainProducts

GrainProducts テーブルは、“SELECT ステートメント” (977 ページ)で使用されます。

注: このサンプルコードでは、FedSQL ステートメントは PROC FEDSQL を使用して SAS に送信されます。

```
proc fedsql;
  create table grainproducts(prodid double primary key, product char(8));

  insert into grainproducts values (1424, 'Rice');
  insert into grainproducts values (3421, 'Corn');
  insert into grainproducts values (3234, 'Wheat');
  insert into grainproducts values (3485, 'Oat');
quit;
```

アウトプット A9.9　テーブル GrainProducts のコンテンツ

PRODID	PRODUCT
1424	Rice
3421	Corn
3234	Wheat
3485	Oat

Houses

Houses テーブルは、“FedSQL で GROUP BY 再マージクエリを書き換える” (1180 ページ)で使用されます。

```
data houses;
  input House $ x y;
  datalines;
house1 1 1
house2 3 3
house3 2 3
house4 7 7
;
```

アウトプット A9.10　テーブル Houses のコンテンツ

House	x	y
house1	1	1
house2	3	3
house3	2	3
house4	7	7

Integers

Integers テーブルは、“CAST 関数” (332 ページ)で使用されます。

注: このサンプルコードでは、FedSQL ステートメントは PROC FEDSQL を使用して SAS に送信されます。

```
proc fedsql;
  create table integers
   (bi      bigint having format comma25.0,
    ii      integer,
    si      smallint,
    ti      tinyint);

  insert into integers values(9223372036854775800,2147483647,32767,127);
quit;
```

アウトプット A9.11　テーブル Integers のコンテンツ

BI	II	SI	TI
9,223,372,036,854,775,808	2.1475E9	32767	127

注: 指定された BIGINT 値は、テーブル Integers に挿入される前に DOUBLE に変換されます。変換の結果として精度が失われるため、テーブル Integers に挿入される値は、指定された値とは異なります。

Numbers1

Numbers1 テーブルは、“クエリ式とサブクエリ” (46 ページ)で使用されます。

注: このサンプルコードでは、FedSQL ステートメントは PROC FEDSQL を使用して SAS に送信されます。

```
proc fedsql;
  create table numbers1(x char(5), y double, z double);
  insert into numbers1 values ('one', 1, 10);
  insert into numbers1 values ('three', 3, 30);
  insert into numbers1 values ('four', 4, 40);
  insert into numbers1 values ('four', 4, 40);
  insert into numbers1 values ('five', 5, 50);
  insert into numbers1 values('five', 5, 50);
quit;
```

アウトプット A9.12　テーブル Numbers1 のコンテンツ

X	Y	Z
one	1	10
three	3	30
four	4	40
four	4	40
five	5	50
five	5	50

Numbers2

Numbers2 テーブルは、“クエリ式とサブクエリ” (46 ページ)で使用されます。

注: このサンプルコードでは、FedSQL ステートメントは PROC FEDSQL を使用して SAS に送信されます。

```
proc fedsql;
  create table numbers2(x varchar(5), y double, z double);
  insert into numbers2 values ('four', 4, 40);
  insert into numbers2 values ('five', 5, 50);
  insert into numbers2 values('five', 5, 50);
  insert into numbers2 values ('one', 1, 10);
  insert into numbers2 values ('two', 2, 20);
quit;
```

アウトプット A9.13　テーブル Numbers2 のコンテンツ

X	Y	Z
four	4	40
five	5	50
five	5	50
one	1	10
two	2	20

Numbers3

Numbers3 テーブルは、“クエリ式とサブクエリ” (46 ページ)で使用されます。

注: このサンプルコードでは、FedSQL ステートメントは PROC FEDSQL を使用して SAS に送信されます。

```
proc fedsql;
  create table numbers3(z double, y double, x varchar(6));
  insert into numbers3 values (40, 4, 'four');
  insert into numbers3 values(60, 6, 'six');
  insert into numbers3 values(70, 7, 'seven');
quit;
```

アウトプット A9.14　テーブル Numbers3 のコンテンツ

Z	Y	X
40	4	four
60	6	six
70	7	seven

Payroll

Payroll テーブルは、“FedSQL で GROUP BY 再マージクエリを書き換える” (1180 ページ)で使用されます。

```
data payroll;
  infile datalines delimiter=',' linesize=80;
  length IdNumber $4 Gender $1 Jobcode $3;
  input IdNumber
    Gender
    Jobcode
    Salary
    Birth
    Hired @@;
  informat IdNumber $4. Gender $1. Jobcode $3. Salary Best8.
    Birth Hired date7.;
  format Birth date7. Hired date7.;
datalines;
1919,M,TA2,34376,12SEP60,04JUN87,1653,F,ME2,35108,15OCT64,09AUG90
1400,M,ME1,29769,05NOV67,16OCT90,1350,F,FA3,32886,31AUG65,29JUL90
1401,M,TA3,38822,13DEC50,17NOV85,1499,M,ME3,43025,26APR54,07JUN80
1101,M,SCP,18723,06JUN62,01OCT90,1333,M,PT2,88606,30MAR61,10FEB81
1402,M,TA2,32615,17JAN63,02DEC90,1479,F,TA3,38785,22DEC68,05OCT89
1403,M,ME1,28072,28JAN69,21DEC91,1739,M,PT1,66517,25DEC64,27JAN91
1658,M,SCP,17943,08APR67,28FEB92,1428,F,PT1,68767,04APR60,16NOV91
1782,M,ME2,35345,04DEC70,22FEB92,1244,M,ME2,36925,31AUG63,17JAN88
1383,M,BCK,25823,25JAN68,20OCT92,1574,M,FA2,28572,27APR60,20DEC92
1789,M,SCP,18326,25JAN57,11APR78,1404,M,PT2,91376,24FEB53,01JAN80
1437,F,FA3,33104,20SEP60,31AUG84,1639,F,TA3,40260,26JUN57,28JAN84
1269,M,NA1,41690,03MAY72,28NOV92,1065,M,ME2,35090,26JAN44,07JAN87
1876,M,TA3,39675,20MAY58,27APR85,1037,F,TA1,28558,10APR64,13SEP92
1129,F,ME2,34929,08DEC61,17AUG91,1988,M,FA3,32217,30NOV59,18SEP84
1405,M,SCP,18056,05MAR66,26JAN92,1430,F,TA2,32925,28FEB62,27APR87
1983,F,FA3,33419,28FEB62,27APR87,1134,F,TA2,33462,05MAR69,21DEC88
1118,M,PT3,111379,16JAN44,18DEC80,1438,F,TA3,39223,15MAR65,18NOV87
1125,F,FA2,28888,08NOV68,11DEC87,1475,F,FA2,27787,15DEC61,13JUL90
1117,M,TA3,39771,05JUN63,13AUG92,1935,F,NA2,51081,28MAR54,16OCT81
1124,F,FA1,23177,10JUL58,01OCT90,1422,F,FA1,22454,04JUN64,06APR91
1616,F,TA2,34137,01MAR70,04JUN93,1406,M,ME2,35185,08MAR61,17FEB87
1120,M,ME1,28619,11SEP72,07OCT93,1094,M,FA1,22268,02APR70,17APR91
1389,M,BCK,25028,15JUL59,18AUG90,1905,M,PT1,65111,16APR72,29MAY92
1407,M,PT1,68096,23MAR69,18MAR90,1114,F,TA2,32928,18SEP69,27JUN87
```

```
1410,M,PT2,84685,03MAY67,07NOV86,1439,F,PT1,70736,06MAR64,10SEP90
1409,M,ME3,41551,19APR50,22OCT81,1408,M,TA2,34138,29MAR60,14OCT87
1121,M,ME1,29112,26SEP71,07DEC91,1991,F,TA1,27645,07MAY72,12DEC92
1102,M,TA2,34542,01OCT59,15APR91,1356,M,ME2,36869,26SEP57,22FEB83
1545,M,PT1,66130,12AUG59,29MAY90,1292,F,ME2,36691,28OCT64,02JUL89
1440,F,ME2,35757,27SEP62,09APR91,1368,M,FA2,27808,11JUN61,03NOV84
1369,M,TA2,33705,28DEC61,13MAR87,1411,M,FA2,27265,27MAY61,01DEC89
1113,F,FA1,22367,15JAN68,17OCT91,1704,M,BCK,25465,30AUG66,28JUN87
1900,M,ME2,35105,25MAY62,27OCT87,1126,F,TA3,40899,28MAY63,21NOV80
1677,M,BCK,26007,05NOV63,27MAR89,1441,F,FA2,27158,19NOV69,23MAR91
1421,M,TA2,33155,08JAN59,28FEB90,1119,M,TA1,26924,20JUN62,06SEP88
1834,M,BCK,26896,08FEB72,02JUL92,1777,M,PT3,109630,23SEP51,21JUN81
1663,M,BCK,26452,11JAN67,11AUG91,1106,M,PT2,89632,06NOV57,16AUG84
1103,F,FA1,23738,16FEB68,23JUL92,1477,M,FA2,28566,21MAR64,07MAR88
1476,F,TA2,34803,30MAY66,17MAR87,1379,M,ME3,42264,08AUG61,10JUN84
1104,M,SCP,17946,25APR63,10JUN91,1009,M,TA1,28880,02MAR59,26MAR92
1412,M,ME1,27799,18JUN56,05DEC91,1115,F,FA3,32699,22AUG60,29FEB80
1128,F,TA2,32777,23MAY65,20OCT90,1442,F,PT2,84536,05SEP66,12APR88
1417,M,NA2,52270,27JUN64,07MAR89,1478,M,PT2,84203,09AUG59,24OCT90
1673,M,BCK,25477,27FEB70,15JUL91,1839,F,NA1,43433,29NOV70,03JUL93
1347,M,TA3,40079,21SEP67,06SEP84,1423,F,ME2,35773,14MAY68,19AUG90
1200,F,ME1,27816,10JAN71,14AUG92,1970,F,FA1,22615,25SEP64,12MAR91
1521,M,ME3,41526,12APR63,13JUL88,1354,F,SCP,18335,29MAY71,16JUN92
1424,F,FA2,28978,04AUG69,11DEC89,1132,F,FA1,22413,30MAY72,22OCT93
1845,M,BCK,25996,20NOV59,22MAR80,1556,M,PT1,71349,22JUN64,11DEC91
1413,M,FA2,27435,16SEP65,02JAN90,1123,F,TA1,28407,31OCT72,05DEC92
1907,M,TA2,33329,15NOV60,06JUL87,1436,F,TA2,34475,11JUN64,12MAR87
1385,M,ME3,43900,16JAN62,01APR86,1432,F,ME2,35327,03NOV61,10FEB85
1111,M,NA1,40586,14JUL73,31OCT92,1116,F,FA1,22862,28SEP69,21MAR91
1352,M,NA2,53798,02DEC60,16OCT86,1555,F,FA2,27499,16MAR68,04JUL92
1038,F,TA1,26533,09NOV69,23NOV91,1420,M,ME3,43071,19FEB65,22JUL87
1561,M,TA2,34514,30NOV63,07OCT87,1434,F,FA2,28622,11JUL62,28OCT90
1414,M,FA1,23644,24MAR72,12APR92,1112,M,TA1,26905,29NOV64,07DEC92
1390,M,FA2,27761,19FEB65,23JUN91,1332,M,NA1,42178,17SEP70,04JUN91
1890,M,PT2,91908,20JUL51,25NOV79,1429,F,TA1,27939,28FEB60,07AUG92
1107,M,PT2,89977,09JUN54,10FEB79,1908,F,TA2,32995,10DEC69,23APR90
1830,F,PT2,84471,27MAY57,29JAN83,1882,M,ME3,41538,10JUL57,21NOV78
1050,M,ME2,35167,14JUL63,24AUG86,1425,F,FA1,23979,28DEC71,28FEB93
1928,M,PT2,89858,16SEP54,13JUL90,1480,F,TA3,39583,03SEP57,25MAR81
1100,M,BCK,25004,01DEC60,07MAY88,1995,F,ME1,28810,24AUG73,19SEP93
1135,F,FA2,27321,20SEP60,31MAR90,1415,M,FA2,28278,09MAR58,12FEB88
1076,M,PT1,66558,14OCT55,03OCT91,1426,F,TA2,32991,05DEC66,25JUN90
1564,F,SCP,18833,12APR62,01JUL92,1221,F,FA2,27896,22SEP67,04OCT91
1133,M,TA1,27701,13JUL66,12FEB92,1435,F,TA3,38808,12MAY59,08FEB80
1418,M,ME1,28005,29MAR57,06JAN92,1017,M,TA3,40858,28DEC57,16OCT81
1443,F,NA1,42274,17NOV68,29AUG91,1131,F,TA2,32575,26DEC71,19APR91
1427,F,TA2,34046,31OCT70,30JAN90,1036,F,TA3,39392,19MAY65,23OCT84
1130,F,FA1,23916,16MAY71,05JUN92,1127,F,TA2,33011,09NOV64,07DEC86
1433,F,FA3,32982,08JUL66,17JAN87,1431,F,FA3,33230,09JUN64,05APR88
1122,F,FA2,27956,01MAY63,27NOV88,1105,M,ME2,34805,01MAR62,13AUG90
;
run;
```

アウトプット A9.15　テーブル Payroll のコンテンツ(部分的)

IdNumber	Gender	Jobcode	Salary	Birth	Hired
1919	M	TA2	34376	12SEP60	04JUN87
1653	F	ME2	35108	15OCT64	09AUG90
1400	M	ME1	29769	05NOV67	16OCT90
1350	F	FA3	32886	31AUG65	29JUL90
1401	M	TA3	38822	13DEC50	17NOV85
1499	M	ME3	43025	26APR54	07JUN80
1101	M	SCP	18723	06JUN62	01OCT90
1333	M	PT2	88606	30MAR61	10FEB81
1402	M	TA2	32615	17JAN63	02DEC90
1479	F	TA3	38785	22DEC68	05OCT89
1403	M	ME1	28072	28JAN69	21DEC91
1739	M	PT1	66517	25DEC64	27JAN91
1658	M	SCP	17943	08APR67	28FEB92
1428	F	PT1	68767	04APR60	16NOV91
1782	M	ME2	35345	04DEC70	22FEB92
1244	M	ME2	36925	31AUG63	17JAN88

Products

Products テーブルは、“結合操作” (30 ページ)、“CREATE TABLE ステートメント” (939 ページ)、および“INSERT ステートメント” (972 ページ)で使用されます。

注: このサンプルコードでは、FedSQL ステートメントは PROC FEDSQL を使用して SAS に送信されます。

```
proc fedsql;
  create table products(prodid double primary key, product char(20));

  insert into products values (3234, 'Rice');
  insert into products values (1424, 'Corn');
  insert into products values (3421, 'Wheat');
  insert into products values (3422, 'Oat');
  insert into products values (3975, 'Barley');
quit;
```

アウトプット A9.16　テーブル Products のコンテンツ

PRODID	PRODUCT
3234	Rice
1424	Corn
3421	Wheat
3422	Oat
3975	Barley

Sales

Sales テーブルは、“結合操作” (30 ページ)、“CREATE TABLE ステートメント” (939 ページ)、および“INSERT ステートメント” (972 ページ)で使用されます。

注: このサンプルコードでは、FedSQL ステートメントは PROC FEDSQL を使用して SAS に送信されます。

```
proc fedsql;
  create table sales
    (prodid double,
     custid double primary key,
     totals double having format dollar12.,
     country char(30) not null
    );

  insert into sales values (3234, 1, 189400, 'United States');
  insert into sales values (1424, 3, 555789, 'Japan');
  insert into sales values (3421, 4, 781183, 'Japan');
  insert into sales values (3421, 2, 2789654, 'United States');
  insert into sales values (3975, 5, 899453, 'Argentina');
quit;
```

アウトプット A9.17　テーブル Sales のコンテンツ

PRODID	CUSTID	TOTALS	COUNTRY
3234	1	$189,400	United States
1424	3	$555,789	Japan
3421	4	$781,183	Japan
3421	2	$2,789,654	United States
3975	5	$899,453	Argentina

Staff

Staff テーブルは、“EXISTS 述語” (906 ページ)で使用されます。

```
data staff;
  infile datalines delimiter=',' flowover linesize=80;
  length idnum $4 lname $15 fname $15 city $15 state $2 hphone $12;
  input idnum $ lname $ fname $ city $ state $ hphone $ @@;
datalines;
1919,ADAMS,GERALD,STAMFORD,CT,203/781-1255,1653,ALIBRANDI,MARIA,BRIDGEPORT
CT,203/675-7715,1400,ALHERTANI,ABDULLAH,NEW YORK,NY,212/586-0808,1350,ALVAREZ
MERCEDES,NEW
YORK,NY,718/383-1549,1401,ALVAREZ,CARLOS,PATERSON,NJ,201/732-8787
1499,BAREFOOT,JOSEPH,PRINCETON,NJ,201/812-5665,1101,BAUCOM,WALTER,NEW
YORK,NY
212/586-8060,1333,BANADYGA,JUSTIN,STAMFORD,CT,203/781-1777,1402,BLALOCK,RALPH
NEW YORK,NY,718/384-2849,1479,BALLETTI,MARIE,NEW YORK,NY,718/384-8816,1403
BOWDEN,EARL,BRIDGEPORT,CT,203/675-3434,1739,BRANCACCIO,JOSEPH,NEW YORK
NY,212/587-1247,1658,BREUHAUS,JEREMY,NEW YORK,NY,212/587-3622,1428,BRADY
CHRISTINE,STAMFORD,CT,203/781-1212,1782,BREWCZAK,JAKOB,STAMFORD,CT,203/781-00
19
1244,BUCCI,ANTHONY,NEW YORK,NY,718/383-3334,1383,BURNETTE,THOMAS,NEW
YORK,NY
718/384-3569,1574,CAHILL,MARSHALL,NEW YORK,NY,718/383-2338,1789,CARAWAY,DAVIS
NEW YORK,NY,212/587-9000,1404,COHEN,LEE,NEW YORK,NY,718/384-2946,1437,CARTER
DOROTHY,BRIDGEPORT,CT,203/675-4117,1639,CARTER-COHEN,KAREN,STAMFORD,CT
203/781-8839,1269,CASTON,FRANKLIN,STAMFORD,CT,203/781-3335,1065,COPAS
FREDERICO,NEW YORK,NY,718/384-5618,1876,CHIN,JACK,NEW YORK,NY,212/588-5634
1037,CHOW,JANE,STAMFORD,CT,203/781-8868,1129,COUNIHAN,BRENDA,NEW YORK,NY
718/383-2313,1988,COOPER,ANTHONY,NEW YORK,NY,212/587-1228,1405,DACKO,JASON
PATERSON,NJ,201/732-2323,1430,DABROWSKI,SANDRA,BRIDGEPORT,CT,203/675-1647
1983,DEAN,SHARON,NEW YORK,NY,718/384-1647,1134,DELGADO,MARIA,STAMFORD,CT
203/781-1528,1118,DENNIS,ROGER,NEW YORK,NY,718/383-1122,1438,DABBOUSSI
KAMILLA,STAMFORD,CT,203/781-2229,1125,DUNLAP,DONNA,NEW YORK,NY
718/383-2094,1475,ELGES,MARGARETE,NEW YORK,NY,718/383-2828,1117,EDGERTON
JOSHUA,NEW YORK,NY,212/588-1239,1935,FERNANDEZ,KATRINA,BRIDGEPORT,CT
203/675-2962,1124,FIELDS,DIANA,WHITE PLAINS,NY,914/455-2998,1422,FUJIHARA
KYOKO,PRINCETON,NJ,201/812-0902,1616,FUENTAS,CARLA,NEW YORK,NY,718/384-3329
1406,FOSTER,GERALD,BRIDGEPORT,CT,203/675-6363,1120,GARCIA,JACK,NEW YORK,NY
718/384-4930,1094,GOMEZ,ALAN,BRIDGEPORT,CT,203/675-7181,1389,GOLDSTEIN,LEVI
NEW YORK,NY,718/384-9326,1905,GRAHAM,ALVIN,NEW YORK,NY,212/586-8815,1407
GREGORSKI,DANIEL,MT. VERNON,NY,914/468-1616,1114,GREENWALD,JANICE,NEW YORK
NY,212/588-1092,1410,HARRIS,CHARLES,STAMFORD,CT,203/781-0937,1439,HASENHAUER
CHRISTINA,BRIDGEPORT,CT,203/675-4987,1409,HAVELKA,RAYMOND,STAMFORD,CT
203/781-9697,1408,HENDERSON,WILLIAM,PRINCETON,NJ,201/812-4789,1121,HERNANDEZ
ROBERTO,NEW YORK,NY,718/384-3313,1991,HOWARD,GRETCHEN,BRIDGEPORT,CT
203/675-0007,1102,HERMANN,JOACHIM,WHITE PLAINS,NY,914/455-0976,1356,HOWARD
MICHAEL,NEW YORK,NY,212/586-8411,1545,HERRERO,CLYDE,STAMFORD,CT,203/781-1119
1292,HUNTER,HELEN,BRIDGEPORT,CT,203/675-4830,1440,JACKSON,LAURA,STAMFORD
CT,203/781-0088,1368,JEPSEN,RONALD,STAMFORD,CT,203/781-8413,1369,JONSON
ANTHONY,NEW YORK,NY,212/587-5385,1411,JOHNSEN,JACK,PATERSON,NJ,201/732-3678
```

1113,JOHNSON,LESLIE,NEW YORK,NY,718/383-3003,1704,JONES,NATHAN,NEW YORK,NY
718/384-0049,1900,KING,WILLIAM,NEW YORK,NY,718/383-3698,1126,KIMANI,ANNE
NEW YORK,NY,212/586-1229,1677,KRAMER,JACKSON,BRIDGEPORT,CT,203/675-7432
1441,LAWRENCE,KATHY,PRINCETON,NJ,201/812-3337,1421,LEE,RUSSELL,MT. VERNON
NY,914/468-9143,1119,LI,JEFF,NEW YORK,NY,212/586-2344,1834,LEBLANC,RUSSELL
NEW YORK,NY,718/384-0040,1777,LUFKIN,ROY,NEW YORK,NY,718/383-4413,1663
MARKS,JOHN,NEW YORK,NY,212/587-7742,1106,MARSHBURN,JASPER,STAMFORD,CT
203/781-1457,1103,MCDANIEL,RONDA,NEW YORK,NY,212/586-0013,1477,MEYERS
PRESTON,BRIDGEPORT,CT,203/675-8125,1476,MONROE,JOYCE,STAMFORD,CT
203/781-2837,1379,MORGAN,ALFRED,STAMFORD,CT,203/781-2216,1104
MORGAN,CHRISTOPHER,NEW YORK,NY,718/383-9740,1009,MORGAN,GEORGE
NEW YORK,NY,212/586-7753,1412,MURPHEY,JOHN,PRINCETON,NJ,201/812-4414
1115,MURPHY,ALICE,NEW YORK,NY,718/384-1982,1128,NELSON,FELICIA
BRIDGEPORT,CT,203/675-1166,1442,NEWKIRK,SANDRA,PRINCETON,NJ
201/812-3331,1417,NEWKIRK,WILLIAM,PATERSON,NJ,201/732-6611,1478
NEWTON,JAMES,NEW YORK,NY,212/587-5549,1673,NICHOLLS,HENRY,STAMFORD
CT,203/781-7770,1839,NORRIS,DIANE,NEW YORK,NY,718/384-1767,1347
O'NEAL,BRYAN,NEW YORK,NY,718/384-0230,1423,OSWALD,LESLIE,MT. VERNON
NY,914/468-9171,1200,OVERMAN,MICHELLE,STAMFORD,CT,203/781-1835
1970,PARKER,ANNE,NEW YORK,NY,718/383-3895,1521,PARKER,JAY,NEW YORK
NY,212/587-7603,1354,PARKER,MARY,WHITE PLAINS,NY,914/455-2337
1424,PATTERSON,RENEE,NEW YORK,NY,212/587-8991,1132,PEARCE,CAROL
NEW YORK,NY,718/384-1986,1845,PEARSON,JAMES,NEW YORK,NY
718/384-2311,1556,PENNINGTON,MICHAEL,NEW YORK,NY,718/383-5681
1413,PETERS,RANDALL,PRINCETON,NJ,201/812-2478,1123,PETERSON,SUZANNE
NEW YORK,NY,718/383-0077,1907,PHELPS,WILLIAM,STAMFORD,CT,203/781-1118
1436,PORTER,SUSAN,NEW YORK,NY,718/383-5777,1385,RAYNOR,MILTON
BRIDGEPORT,CT,203/675-2846,1432,REED,MARILYN,MT. VERNON,NY,914/468-5454
1111,RHODES,JEREMY,PRINCETON,NJ,201/812-1837,1116,RICHARDS,CASEY
NEW YORK,NY,212/587-1224,1352,RIVERS,SIMON,NEW YORK,NY,718/383-3345
1555,RODRIGUEZ,JULIA,BRIDGEPORT,CT,203/675-2401,1038,RODRIGUEZ,MARIA
BRIDGEPORT,CT,203/675-2048,1420,ROUSE,JEREMY,PATERSON,NJ,201/732-9834
1561,SANDERS,RAYMOND,NEW YORK,NY,212/588-6615,1434,SANDERSON,EDITH
STAMFORD,CT,203/781-1333,1414,SANDERSON,NATHAN,BRIDGEPORT,CT,203/675-1715
1112,SANYERS,RANDY,NEW YORK,NY,718/384-4895,1390,SMART,JONATHAN,NEW YORK
NY,718/383-1141,1332,STEPHENSON,ADAM,BRIDGEPORT,CT,203/675-1497
1890,STEPHENSON,ROBERT,NEW YORK,NY,718/384-9874,1429,THOMPSON,ALICE
STAMFORD,CT,203/781-3857,1107,THOMPSON,WAYNE,NEW YORK,NY,718/384-3785
1908,TRENTON,MELISSA,NEW YORK,NY,212/586-6262,1830,TRIPP,KATHY,BRIDGEPORT
CT,203/675-2479,1882,TUCKER,ALAN,NEW YORK,NY,718/384-0216,1050,TUTTLE
THOMAS,WHITE PLAINS,NY,914/455-2119,1425,UNDERWOOD,JENNY,STAMFORD,CT
203/781-0978,1928,UPCHURCH,LARRY,WHITE PLAINS,NY,914/455-5009,1480,UPDIKE
THERESA,NEW YORK,NY,212/587-8729,1100,VANDEUSEN,RICHARD,NEW YORK,NY
212/586-2531,1995,VARNER,ELIZABETH,NEW YORK,NY,718/384-7113,1135,VEGA
ANNA,NEW YORK,NY,718/384-5913,1415,VEGA,FRANKLIN,NEW YORK,NY,718/384-2823
1076,VENTER,RANDALL,NEW YORK,NY,718/383-2321,1426,VICK,THERESA,PRINCETON,NJ
201/812-2424,1564,WALTERS,ANNE,NEW YORK,NY,212/587-3257,1221,WALTERS,DIANE
NEW YORK,NY,718/384-1918,1133,WANG,CHIN,NEW YORK,NY,212/587-1956,1435,WARD
ELAINE,NEW YORK,NY,718/383-4987,1418,WATSON,BERNARD,NEW YORK,NY,718/383-1298
1017,WELCH,DARIUS,NEW YORK,NY,212/586-5535,1443,WELLS,AGNES,STAMFORD,CT
203/781-5546,1131,WELLS,NADINE,NEW YORK,NY,718/383-1045,1427,WHALEY
CAROLYN,MT. VERNON,NY,914/468-4528,1036,WONG,LESLIE,NEW YORK,NY,212/587-2570
1130,WOOD,DEBORAH,NEW YORK,NY,212/587-0013,1127,WOOD,SANDRA,NEW YORK,NY
212/587-2881,1433,YANCEY,ROBIN,PRINCETON,NJ,201/812-1874,1431,YOUNG,DEBORAH
STAMFORD,CT,203/781-2987,1122,YOUNG,JOANN,NEW
YORK,NY,718/384-2021,1105,YOUNG

```
LAWRENCE,NEW YORK,NY,718/384-0008
;
run;
```

アウトプット A9.18　テーブル Staff のコンテンツ(部分的)

idnum	lname	fname	city	state	hphone
1919	ADAMS	GERALD	STAMFORD	CT	203/781-1255
1653	ALIBRANDI	MARIA	BRIDGEPORT	CT	203/675-7715
1400	ALHERTANI	ABDULLAH	NEW YORK	NY	212/586-0808
1350	ALVAREZ	MERCEDES	NEW YORK	NY	718/383-1549
1401	ALVAREZ	CARLOS	PATERSON	NJ	201/732-8787
1499	BAREFOOT	JOSEPH	PRINCETON	NJ	201/812-5665
1101	BAUCOM	WALTER	NEW YORK	NY	212/586-8060
1333	BANADYGA	JUSTIN	STAMFORD	CT	203/781-1777
1402	BLALOCK	RALPH	NEW YORK	NY	718/384-2849
1479	BALLETTI	MARIE	NEW YORK	NY	718/384-8816
1403	BOWDEN	EARL	BRIDGEPORT	CT	203/675-3434
1739	BRANCACCIO	JOSEPH	NEW YORK	NY	212/587-1247
1658	BREUHAUS	JEREMY	NEW YORK	NY	212/587-3622
1428	BRADY	CHRISTINE	STAMFORD	CT	203/781-1212
1782	BREWCZAK	JAKOB	STAMFORD	CT	203/781-0019
1244	BUCCI	ANTHONY	NEW YORK	NY	718/383-3334
1383	BURNETTE	THOMAS	NEW YORK	NY	718/384-3569
1574	CAHILL	MARSHALL	NEW YORK	NY	718/383-2338
1789	CARAWAY	DAVIS	NEW YORK	NY	212/587-9000

Stores

Stores テーブルは、“FedSQL で GROUP BY 再マージクエリを書き換える” (1180 ページ)で使用されます。

```
data stores;
  input Store $ x y;
  datalines;
store1 5 1
store2 5 3
store3 3 5
store4 7 5
;
```

アウトプット A9.19　テーブル Stores のコンテンツ

Store	x	y
store1	5	1
store2	5	3
store3	3	5
store4	7	5

Survey

Survey テーブルは、“FedSQL で SAS マクロを使用する” (1188 ページ)で使用されます。

```
data survey;
 input id $ diet $ exer $ hours xwk educ;
 datalines;
1001 yes yes 1 3 1
1002 no  yes 1 4 2
1003 no  no  . . .n
1004 yes yes 2 3 .x
1005 no  yes 2 3 .x
1006 yes yes 2 4 .x
1007 no  yes .5 3 .
1008 no  no  . . .
;
```

アウトプット A9.20　テーブル Survey のコンテンツ

id	diet	exer	hours	xwk	educ
1001	yes	yes	1.0	3	1
1002	no	yes	1.0	4	2
1003	no	no	.	.	N
1004	yes	yes	2.0	3	X
1005	no	yes	2.0	3	X
1006	yes	yes	2.0	4	X
1007	no	yes	0.5	3	.
1008	no	no	.	.	.

WorldCityCoords

FedSQL ドキュメント内のいくつかの例では、WorldCityCoords テーブルを使用しています。

注: このサンプルコードでは、FedSQL ステートメントは PROC FEDSQL を使用して SAS に送信されます。

```
data WorldCityCoords;
  infile datalines delimiter='#' flowover linesize=80;
  length City Country $20;
  input City $
    Country $
    Latitude
    Longitude @@;
datalines;
Buenos Aires#Argentina#-34#-59#Brisbane#Australia#-27#153
Nassau#Bahamas#26#-77#La Paz#Bolivia#-16#-69
Porto Alegre#Brazil#-30#-51#Recife#Brazil#-9#-35
Rio de Janeiro#Brazil#-23#-43#Salvador#Brazil#-13#-38
Sao Paulo#Brazil#-23#-46#Havre#Canada#48#-110
Quebec#Canada#47#-71#Winnipeg#Canada#50#-98
Punta Arenas#Chile#-53#-71#Valparaiso#Chile#-33#-71
Cali#Colombia#3#-76#Brazzaville#Congo#-4#15
Guantanamo Bay#Cuba#20#-76#Copenhagen#Denmark#56#12
Cairo#Egypt#30#31#Guayaquil#Ecuador#-21#-80
Asmara#Ethiopia#15#39#Strasbourg#France#48#8
Munich#Germany#49#11#Makassar#Indonesia#-6#119
Abadan#Iran#30#48#Baghdad#Iraq#33#44
Dublin#Ireland#53#-6#Shannon#Ireland#53#-9
Tel Aviv#Israel#33#35#Naples#Italy#41#14
Amman#Jordan#32#36#Pyongyang#Korea, North#39#126
Monrovia#Liberia#6#-11#Kuala Lumpur#Malaysia#4#102
Manila#Philippines#14#121#Krakow#Poland#51#20
Bucharest#Romania#44#27#Dhahran#Saudi Arabia#26#51
Dakar#Senegal#15#-17#Singapore#Singapore#1#104
Mogadiscio#Somalia#2#49#Cape Town#South Africa#-34#18
Pretoria#South Africa#-26#28#Stockholm#Sweden#59#19
Port of Spain#Trinidad and Tobago#11#-61#Izmir#Turkey#38#27
Edinburgh#Scotland#56#-3
;
run;
```

アウトプット A9.21　テーブル WorldCityCoords のコンテンツ(部分的)

City	Country	Latitude	Longitude
Buenos Aires	Argentina	-34	-59
Brisbane	Australia	-27	153
Nassau	Bahamas	26	-77
La Paz	Bolivia	-16	-69
Porto Alegre	Brazil	-30	-51
Recife	Brazil	-9	-35
Rio de Janeiro	Brazil	-23	-43
Salvador	Brazil	-13	-38
Sao Paulo	Brazil	-23	-46
Havre	Canada	48	-110
Quebec	Canada	47	-71
Winnipeg	Canada	50	-98
Punta Arenas	Chile	-53	-71
Valparaiso	Chile	-33	-71
Cali	Colombia	3	-76
Brazzaville	Congo	-4	15
Guantanamo	Cuba	20	-76

WorldTemps

WorldTemps テーブルは、“CASE 式” (899 ページ)、“COUNT 関数” (413 ページ)、“IFNULL 関数” (573 ページ)、“KURTOSIS 関数” (629 ページ)、“NMISS 関数” (682 ページ)、“PUT 関数” (788 ページ)、“RANGE 関数” (799 ページ)、“SKEWNESS 関数” (829 ページ)、“STDDEV 関数” (837 ページ)、“STDERR 関数” (838 ページ)、“IN 述語” (907 ページ)で使用されます。

注: このサンプルコードでは、FedSQL ステートメントは PROC FEDSQL を使用して SAS に送信されます。

```
data WorldTemps;
  infile datalines delimiter='#' flowover linesize=80;
  length City $18 Country $15;
  input City
    Country
    AvgHigh
    AvgLow @@;
datalines;
Algiers#Algeria#90#45#Amsterdam#Netherlands#70#33#Athens#Greece#89#41
Auckland#New Zealand#75#44#Bangkok#Thailand#95#69#Beijing#China#86#17
Belgrade#Yugoslavia#80#29#Berlin#Germany#75#25#Bogota#Colombia#69#43
Bombay#India#90#68#Bucharest#Romania#83#24#Budapest#Hungary#80#25
```

```
Buenos Aires#Argentina#87#48#Cairo#Egypt#95#48#Calcutta#India#97#56
Cape Town#South
Africa#70#48#Caracas#Venezuela#83#57#Copenhagen#Denmark#73#30
Dublin#Ireland#68#36#Geneva#Switzerland#76#28#Glasgow#Scotland#65#35
Hamilton#Bermuda#85#59#Havana#Cuba#87#66#Helsinki#Finland#71#18
Hong Kong#China#89#51#Istanbul#Turkey#79#38#Jerusalem#Israel#87#42
Kingston#Jamaica#89#67#Lagos#Nigeria#90#75#Lima#Peru#81#60
Lisbon#Portugal#78#46#London#England#73#36#Madrid#Spain#89#36
Manila#Philippines#89#75#Mexico City#Mexico#77#42#Montreal#Canada#77#8
Moscow#Russia#76#9#Nairobi#Kenya#78#51#Nassau#Bahamas#88#65
Oslo#Norway#73#19#Paris#France#76#32#Prague#Czech Republic#74#27
Quebec#Canada#76#5#Reykjavik#Iceland#57#25#Rio de Janeiro#Brazil#85#64
Rome#Italy#87#39#San Juan#Puerto Rico#86#70#San Salvador#El Salvador#87#37
Sao Paulo#Brazil#81#53#Seoul#Korea, South#84#17#Shanghai#China#89#33
Singapore#Singapore#89#73#Stockholm#Sweden#70#23#Sydney#Australia#79#44
Taipei#Taiwan#91#52#Tokyo#Japan#85#33#Toronto#Canada#80#17
Vienna#Austria#76#28#Zurich#Switzerland#78#25
;
run;
```

アウトプット A9.22　テーブル WorldTemps のコンテンツ(部分的)

City	Country	AvgHigh	AvgLow
Algiers	Algeria	90	45
Amsterdam	Netherlands	70	33
Athens	Greece	89	41
Auckland	New Zealand	75	44
Bangkok	Thailand	95	69
Beijing	China	86	17
Belgrade	Yugoslavia	80	29
Berlin	Germany	75	25
Bogota	Colombia	69	43
Bombay	India	90	68
Bucharest	Romania	83	24
Budapest	Hungary	80	25
Buenos Aires	Argentina	87	48
Cairo	Egypt	95	48
Calcutta	India	97	56
Cape Town	South Africa	70	48

付録 5

DICTIONARY テーブルの説明

DICTIONARY.CATALOGS 1279
DICTIONARY.COLUMNS 1280
DICTIONARY.COLUMN_STATISTICS 1283
DICTIONARY.STATISTICS 1285
DICTIONARY.TABLES 1290

DICTIONARY.CATALOGS

次の表は、DICTIONARY.CATALOGS テーブルの結果テーブルに表示される列のリストです。

列名	列番号	データ型	説明
CATALOG	1	NVARCHAR	カタログ名
DRIVER	2	NVARCHAR	データソース名
DESCRIPTION	3	NVARCHAR	カタログの説明

DICTIONARY.COLUMNS

次の表は、DICTIONARY.COLUMNS テーブルの結果テーブルに表示される列のリストです。

注: Null は、null または欠損値のいずれかを示します。

列名	列番号	データ型	説明
TABLE_CAT	1	NVARCHAR[1]	カタログ名
TABLE_SCHEM	2	NVARCHAR	スキーマ名 スキーマがデータソースに適用できない場合は、Null が返されます。ドライバーが別の DBMS からデータを取得する場合など、ドライバーが一部のテーブルのスキーマをサポートし、他のテーブルのスキーマをサポートしない場合、スキーマを持たないテーブルには空の文字列("")が返されます。
TABLE_NAME	3	NVARCHAR[1]	テーブル名
COLUMN_NAME	4	NVARCHAR[1]	列名 ドライバーは、名前のない列に対して空の文字列を返します。
DATA_TYPE	5	符号付き INTEGER[1]	TKTS データ型。 これは、SAS FedSQL データ型またはリモートテーブルドライバー固有のデータ型です。日時および間隔データ型の場合、この列は、DATETIME や INTERVAL などの簡潔でないデータ型ではなく、DATE や INTERVAL_YEAR_TO_MONTH などの簡潔なデータ型を返します。 ドライバー固有の TKTS データ型については、リモートテーブルドライバーのドキュメントを参照してください。
TYPE_NAME	6	NVARCHAR[1]	データソース依存の共通データ型名(CHAR、NVARCHAR、BIGINT など)。
COLUMN_SIZE	7	符号付き BIGINT[1]	DATA_TYPE が CHAR または VARCHAR の場合、この列には列の最大長(文字数)が含まれます。日時データ型の場合、これは値を文字に変換し

列名	列番号	データ型	説明
			たときに値を表示するために必要な合計文字数です。数値データ型の場合、これは、NUM_PREC_RADIX 列に従って、列で許可されている合計桁数または合計ビット数のいずれかです。間隔データ型の場合、これは間隔リテラルの文字表現の文字数です。
BUFFER_LENGTH	8	符号付き BIGINT[1]	転送されたデータの長さ(バイト数)。数値データの場合、このサイズは、データソースに保存されているデータのサイズとは異なる場合があります。この値は、文字データの COLUMN_SIZE 列とは異なる場合があります。
DECIMAL_DIGITS	9	符号付き INTEGER	小数点以下の有効桁数の合計。 DATA_TYPE が TIME および TIMESTAMP の場合、この列には秒の端数部分の桁数が含まれます。他のデータ型の場合、これはデータソースの列の 10 進数です。時間部分を含む間隔データ型の場合、この列には小数点以下の桁数(秒の端数)が含まれます。時間部分を含まない間隔データ型の場合、この列は 0 です。DECIMAL_DIGITS が適用されないデータ型の場合は、Null が返されます。
NUM_PREC_RADIX	10	符号付き INTEGER	数値データ型の場合、10 または 2 が返されます。 返される値が 10 の場合、COLUMN_SIZE および DECIMAL_DIGITS の値には、列に許可されている 10 進数の桁数が含まれます。たとえば、DECIMAL(12,5)列は、NUM_PREC_RADIX に対して 10、COLUMN_SIZE に対して 12、DECIMAL_DIGITS に対して 5 を返します。FLOAT 列は、NUM_PREC_RADIX の場合は 10、COLUMN_SIZE の場合は 15、DECIMAL_DIGITS の場合は null を返す可能性があります。 返される値が 2 の場合、COLUMN_SIZE および DECIMAL_DIGITS の値は、列で許可されるビット数を示します。たとえば、FLOAT 列は、RADIX に対して 2、COLUMN_SIZE に対して 53、DECIMAL_DIGITS に対して null を返す可能性があります。 NUM_PREC_RADIX が適用されないデータ型の場合は Null が返されます。
NULLABLE	11	符号付き INTEGER 非 null	列が null 値を受け入れない場合は TKTS_NO_NULLS。 列が null 値を受け入れる場合は TKTS _NULLABLE。

列名	列番号	データ型	説明
			列が null 値を受け入れるかどうかがわからない場合は TKTS_NULLABLE_UNKNOWN。 この列に返される値は、IS_NULLABLE 列に返される値とは異なります。NULLABLE 列は、列が null 値を受け入れることができることを確実に示しますが、列が null 値を受け入れないことを確実に示すことはできません。IS_NULLABLE 列は、列が null 値を受け入れることができないことを確実に示しますが、列が null 値を受け入れることを確実に示すことはできません。
REMARKS	12	NVARCHAR	列の説明。FedSQL ビューの場合、REMARKS 列の値は FedSQL.VIEW です。
COLUMN_DEF	13	NVARCHAR	列のデフォルト値。この列の値は、引用符で囲まれている場合、文字列として解釈される必要があります。 null がデフォルト値として指定された場合、この列は引用符で囲まれていない単語 NULL になります。切り捨てなしではデフォルト値を表現できない場合、この列には引用符で囲まれていない TRUNCATED が含まれます。デフォルト値が指定されていない場合、この列は null になります。 COLUMN_DEF の値は、値 TRUNCATED が含まれている場合を除き、新しい列定義の生成に使用できます。
TKTS_DATA_TYPE	14	符号付き INTEGER[1]	TKTS_DESC_TYPE レコードフィールドに表示されるデータ型。 これは、SAS FedSQL データ型またはリモートテーブルドライバー固有のデータ型です。この列は、日時および間隔データ型を除いて、DATA_TYPE 列と同じです。この列は、日時および間隔データ型の簡潔なデータ型(DATE または YEAR_TO_MONTH など)ではなく、簡潔でないデータ型(DATETIME または INTERVAL など)を返します。この列が DATETIME または INTERVAL を返す場合、特定のデータ型は TKTS_DATETIME_SUB 列から判別できます。 ドライバー固有の TKTS データ型については、リモートテーブルドライバーのドキュメントを参照してください。
TKTS_DATETIME_SUB	15	符号付き INTEGER	日時および間隔データ型のサブタイプコード。 他のデータ型の場合、この列は null を返します。

列名	列番号	データ型	説明
CHAR_OCTET_LENGTH	16	符号付き BIGINT	文字データ型またはバイナリデータ型の列のバイト単位の最大長。 他のすべてのデータ型の場合、この列は null を返します。
ORDINAL_POSITION	17	符号付き INTEGER[1]	テーブル内の列の順序。テーブルの最初の列は番号 1 です。
IS_NULLABLE	18	NVARCHAR[1]	列に null が含まれていない場合は NO。 列に null が含まれる可能性がある場合は YES。 null 値の許容が不明な場合、この列は長さ 0 の文字列を返します。 null 値の許容を判断するには、ISO 規則に従います。ISO SQL 準拠の DBMS は、空の文字列を返すことはできません。 この列に返される値は、NULLABLE 列に返される値とは異なります。

1 値を null にすることはできません。

DICTIONARY.COLUMN_STATISTICS

次の表は、DICTIONARY.COLUMN_STATISTICS テーブルの結果テーブルに表示される列のリストです。

注: Null は、null または欠損値のいずれかを示します。

注: 最初の 4 列は結果テーブルに表示されません。

列名	列番号	データ型	説明
TABLE_CAT	1	NVARCHAR[1]	カタログ名
TABLE_SCHEM	2	NVARCHAR	スキーマ名 スキーマがデータソースに適用できない場合は、Null が返されます。ドライバーが別の DBMS からデータを取得する場合など、ドライバーが一部のテーブルのスキーマをサポートし、他のテーブルのスキーマをサポートしない

列名	列番号	データ型	説明
			場合、スキーマを持たないテーブルには空の文字列("")が返されます。
TABLE_NAME	3	NVARCHAR[1]	テーブル名
COLUMN_NAME	4	NVARCHAR[1]	列名 ドライバーは、名前のない列に対して空の文字列を返します。
TYPE	5	符号なし INTEGER[1]	返される情報の種類: TKTS_COL_STAT は、CARDINALITY および HISTOGRAM 列で指定された列の統計量を示します。
CARDINALITY	6	符号なし INTEGER	テーブル内の列または列セットのカーディナリティ。 これは、列または列セット内の重複しない非 null 値の数です。データソースから値を取得できない場合は、Null が返されます。
NULL_FRAC	7	DOUBLE	null である列のエントリの割合(10 進数で表される)。 列セットのいずれかの列が null の場合、そのエントリは null と見なされます。
MOST_COMMON_VALS	8	NVARCHAR	この列に共通の値が含まれます。 共通の値がない場合は Null。 列数が 1 より大きい場合は Null。
MOST_COMMON_FREQS	9	DOUBLE	MOST_COMMON_VALS で返される共通値の頻度。 これは、その特定の列値の出現回数をテーブル内の行の総数で割ったものである可能性があります。 列数が 1 より大きい場合は Null。
HISTOGRAM_ENTRY	10	符号なし INTEGER	HISTOGRAM_BOUNDS 値のエントリ。 列数が 1 より大きい場合は Null。
HISTOGRAM_BOUNDS	11	NVARCHAR	列の値をほぼ等しい母集団のグループに分割する値のリストの 1 つのエントリ。 最も一般的な値(MOST_COMMON_VALS によって返される)は、ヒストグラムの計算から除外されます。列のデータ型に小なり(<)演算子がない場合、または MOST_COMMON_VALS によって

列名	列番号	データ型	説明
			返される値が列の母集団全体を占める場合、HISTOGRAM_BOUNDS は返されません。 列数が 1 より大きい場合は Null。
CORRELATION	12	DOUBLE	行の物理的な順序付けと列の値の論理的な順序付けとの間の統計的相関。 値の範囲は-1 から+1 です。列のデータ型に小なり(<)演算子がない場合、この値は返されません。 列数が 1 より大きい場合は Null。

1 値を null または欠損値にすることはできません

DICTIONARY.STATISTICS

次の表は、DICTIONARY.STATISTICS テーブルの結果テーブルに表示される列のリストです。

注: Null は、null または欠損値のいずれかを示します。

列名	列番号	データ型	説明
TABLE_CAT	1	NVARCHAR[1]	TKTS カタログ名。
TABLE_SCHEM	2	NVARCHAR	統計量またはインデックスが適用されるテーブルのスキーマ名。 データソースに適用できない場合は Null。ドライバーが別の DBMS からデータを取得する場合など、ドライバーが一部のテーブルのスキーマをサポートし、他のテーブルのスキーマをサポートしない場合、スキーマを持たないテーブルには空の文字列("")が返されます。
TABLE_NAME	3	NVARCHAR 非 null	統計量またはインデックスが適用されるテーブルの名前。
NON_UNIQUE	4	符号付き INTEGER	インデックスが重複値を禁止するかどうかを示します。 インデックス値が一意でない場合は TKTS_TRUE。

列名	列番号	データ型	説明
			インデックス値が一意である必要がある場合は TKTS_FALSE。 TYPE が TKTS_TABLE_STAT の場合は Null が返されます。
INDEX_QUALIFIER	5	NVARCHAR	DROP INDEX を実行するインデックス名を修飾するために使用される識別子。 インデックス修飾子がデータソースでサポートされていない場合、または TYPE が TKTS_TABLE_STAT の場合は、Null が返されます。この列に非 null 値が返された場合は、DROP INDEX ステートメントでインデックス名を修飾するために使用する必要があります。それ以外の場合は、TABLE_SCHEM を使用してインデックス名を修飾する必要があります。
INDEX_NAME	6	NVARCHAR	インデックス名。 TYPE が TKTS_TABLE_STAT の場合は Null が返されます。
TYPE	7	符号付き INTEGER[1]	返される情報の種類: TKTS_TABLE_STAT は、CARDINALITY または PAGES 列のテーブルの統計量を示します。 TKTS_INDEX_BTREE は、B ツリーインデックスを示します。 TKTS_INDEX_CLUSTERED は、クラスター化インデックスを示します。 TKTS_INDEX_CONTENT は、コンテンツインデックスを示します。TKTS_INDEX_HASHED は、ハッシュされたインデックスを示します。 TKTS_INDEX_OTHER は、別のタイプのインデックスを示します。
ORDINAL_POSITION	8	符号付き INTEGER	1 から始まるインデックス内の列シーケンス番号。 TYPE が TKTS_TABLE_STAT の場合は Null が返されます。
COLUMN_NAME	9	NVARCHAR	列名。 列が SALARY + BENEFITS などの式に基づいている場合は、その式が返されます。式を判別できない場合は、空の文字列が返されます。TYPE が TKTS_TABLE_STAT の場合は Null が返されます。

列名	列番号	データ型	説明
ASC_OR_DESC	10	CHAR(1)	列の並べ替え順序。昇順の場合は A、降順の場合は D です。 列の並べ替え順序がデータソースでサポートされていない場合、または TYPE が TKTS_TABLE_STAT の場合は、Null が返されます。
CARDINALITY	11	符号付き BIGINT	テーブルまたはインデックスのカーディナリティ。 TYPE が TKTS_TABLE_STAT の場合はテーブル内の行数。 TYPE が TKTS_TABLE_STAT でない場合はインデックス内の一意の値の数。 データソースから値を取得できない場合は Null。
PAGES	12	符号付き INTEGER	インデックスまたはテーブルを保存するために使用されるページ数。 TYPE が TKTS_TABLE_STAT の場合はテーブルのページ数。 TYPE が TKTS_TABLE_STAT でない場合はインデックスのページ数。 値がデータソースから取得できない場合、またはデータソースに適用できない場合は Null。
FILTER_CONDITION	13	NVARCHAR	インデックスがフィルタリングされたインデックスの場合、これはフィルター条件(SALARY > 30000 など)です。 フィルター条件を特定できない場合、値は空の文字列になります。 インデックスがフィルタリングされたインデックスでない場合、インデックスがフィルタリングされたインデックスであるかどうかを判断できない場合、または TYPE が TKTS_TABLE_STAT である場合は Null。
AVG_FANOUT[2]	14	DOUBLE	インデックスの内部ノードの平均ファンアウト。TKTS_INDEX_* TYPE レコードに有効です。 不明または使用できない場合は Null。
MAX_FANOUT[2]	15	符号なし INTEGER	インデックスの内部ノードの最大ファンアウト。TKTS_INDEX_* TYPE レコードに有効です。 不明または使用できない場合は Null。

列名	列番号	データ型	説明
MIN_FANOUT[2]	16	符号なし INTEGER	インデックスの内部ノードの最小ファンアウト。TKTS_INDEX_* TYPE レコードに有効です。 不明または使用できない場合は Null。
INDEX_LEVELS[2]	17	符号なし INTEGER	インデックスのレベル数。TKTS_INDEX_* TYPE レコードに有効です。 不明または使用できない場合は Null。
LEAF_LEVEL_BLOCKS[2]	18	符号なし INTEGER	インデックスのリーフレベルのブロック数。TKTS_INDEX_* TYPE レコードに有効です。 不明または使用できない場合は Null。
LOCAL[2]	19	NVARCHAR	テーブルがローカルであるか、ネットワーク経由でアクセスする必要があるか。TKTS_TABLE_STAT TYPE レコードに有効です。 値は YES、NO、または不明な場合は Null。
PARTITION_SCHEME[2]	20	符号なし INTEGER	テーブルがパーティション分割されているかどうか。パーティション分割されている場合は、ラウンドロビン、ハッシュ、または範囲パーティション分割スキームを使用しているか。TKTS_TABLE_STAT TYPE レコードに有効です。 TKTS_PARTITION_NONE は、テーブルがパーティション分割されていないことを示します。 TKTS_PARTITION_ROUND_ROBIN は、ラウンドロビンパーティション分割スキームを示します。TKTS_PARTITION_HASH は、ハッシュパーティション分割スキームを示します。 TKTS_PARTITION_RANGE は、範囲パーティション分割スキームを示します。不明または使用できない場合は Null。
FRAGMENTATION[2]	21	符号なし INTEGER	データが分散されている場合、垂直断片化、水平断片化、派生断片化、ハイブリッド断片化のいずれを使用しているかを示します。TKTS_TABLE_STAT TYPE レコードに有効です。 TKTS_FRAG_NONE は、データが分散されていないことを示します。TKTS_FRAG_VERTICAL は、垂直断片化を示します。 TKTS_FRAG_HORIZONTAL は、水平断片化を示します。TKTS_FRAG_DERIVED は派生断片化を示します。TKTS_FRAG_HYBRID は、ハイブリッド断片化を使用します。不明または使用できない場合は Null。

列名	列番号	データ型	説明
LABEL[2]	22	NVARCHAR (最大 256 文字)	テーブルに関連付けられたラベル。TKTS_TABLE_STAT TYPE レコードに有効です。 不明または使用できない場合は Null。
APPLICATION_TYPE[2]	23	NVARCHAR (最大 8 文字)	テーブルに関連付けられた SAS アプリケーション固有の型。この型フィールドの値は、一部の SAS アプリケーションによって作成、保存、および取得されます。値の意味は、それを作成したアプリケーションに固有です。TKTS_TABLE_STAT TYPE レコードに有効です。 不明または使用できない場合は Null。
PROTECTED[2]	24	符号付き INTEGER	テーブルがパスワードで保護されているかどうかを示します。TKTS_TABLE_STAT TYPE レコードに有効です。 TKTS_PROTECTION_NONE は、テーブルがパスワードで保護されていないことを示します。 TKTS_PROTECTION_READ は、テーブルが読み取り保護されていることを示します。 TKTS_PROTECTION_WRITE は、テーブルが書き込み保護されていることを示します。 TKTS_PROTECTION_WRITE は、テーブルが変更保護されていることを示します。 TKTS_PROTECTION_ENCRYPTED は、テーブルが暗号化されていることを示します。 不明または使用できない場合は Null。
COMPRESS[2]	25	NVARCHAR (最大 8 文字)	テーブル内のレコードが圧縮されているかどうかを示します。TKTS_TABLE_STAT TYPE レコードに有効です。 NO レは、レコードが圧縮されていないことを示します。 YES|CHAR は、レコードの圧縮に使用される RLE (Run Length Encoding)を示します。 BINARY は、レコードの圧縮に使用される RDC (Ross Data Compression)を示します。 不明または使用できない場合は Null。
CHAR_CEI[2]	26	符号なし INTEGER	テーブルに使用されるエンコーディング値。TKTS_TABLE_STAT TYPE レコードに有効です。 不明または使用できない場合は Null。

列名	列番号	データ型	説明
DELETED_ROWS[2]	27	符号付き BIGINT	テーブル内の削除された行の数。TKTS_TABLE_STAT TYPE レコードに有効です。 不明または使用できない場合は Null。
DATE_CREATED[2]	28	DOUBLE	テーブルが作成された日付。値は、SAS によって定義された DATETIME です。TKTS_TABLE_STAT TYPE レコードに有効です。 不明または使用できない場合は Null。
DATE_MODIFIED[2]	29	DOUBLE	テーブルが最後に変更された日付。値は、SAS によって定義された DATETIME です。TKTS_TABLE_STAT TYPE レコードに有効です。 不明または使用できない場合は Null。
TABLE_ATTRIBUTES[2]	30	符号付き INTEGER	テーブルに適用される属性。この列は、特定の属性を伝えるためのフラグのセットを表します。フラグの値 0 は、偽または不明な状態を示します。TKTS_TABLE_STAT TYPE レコードに有効です。 不明または使用できない場合は Null。
ROW_LENGTH[2]	31	符号付き BIGINT	ディスク上のテーブルの行の長さ。TKTS_TABLE_STAT TYPE レコードに有効です。 不明または使用できない場合は Null。

1 値を null または欠損値にすることはできません
2 オプション - ODBC で定義されていません

DICTIONARY.TABLES

次の表は、DICTIONARY.TABLES テーブルの結果テーブルに表示される列のリストです。

注: Null は、null または欠損値のいずれかを示します。

列名	列番号	データ型	説明
TABLE_CAT	1	NVARCHAR 非 null	カタログ名

列名	列番号	データ型	説明
TABLE_SCHEM	2	NVARCHAR	スキーマ名。データソースに適用できない場合は null。ドライバーが別の DBMS からデータを取得する場合など、ドライバーが一部のテーブルのスキーマをサポートし、他のテーブルのスキーマをサポートしない場合、スキーマを持たないテーブルには空の文字列("")が返されます。
TABLE_NAME	3	NVARCHAR 非 null	テーブル名
TABLE_TYPE	4	NVARCHAR 非 null	テーブルタイプ名。TABLE、VIEW、SYSTEM TABLE、GLOBAL TEMPORARY、LOCAL TEMPORARY、ALIAS、SYNONYM のうちのいずれか、またはデータソース固有のタイプ名。ALIAS と SYNONYM の意味は、ドライバー固有です。
REMARKS	5	NVARCHAR	テーブルの説明
NATIVE_CAT	6	NVARCHAR	ネイティブデータストアのカタログ。データソースに適用できない場合は null。ドライバーが別の DBMS からデータを取得する場合など、ドライバーが一部のテーブルのカタログをサポートし、他のテーブルのカタログをサポートしない場合、カタログを持たないテーブルには空の文字列("")が返されます。
SECURITY	7	NVARCHAR	ビューが実行されるセキュリティ環境。有効な値は、"DEFINER"、"INVOKER"、または null です。"DEFINER"の値は、ビューを含むスキーマの所有者の権限でビューが実行されることを示します。 "INVOKER"の値は、ビューが起動者の権限で実行されることを示します。セキュリティ環境が不明な場合、またはビューがネイティブビューである場合は、null が返されます。
SERVICE	8	NVARCHAR	ビューの管理を担当するサービス。有効な値は、"FEDSQL"、"DRIVER"、または null です。"FEDSQL"の値は、ビューが適切なドライバーへの FEDSQL 接続を介して管理されることを示します。"DRIVER"の値(または null 値)は、DBMS ドライバーによって管理されるネイティブビューを示します。

付録 6

FedSQL と ANSI 規格

準拠 **1293**

FedSQL の拡張 **1294**
予約語 1294
列修飾子 1294
関数 1294
テーブルオプション 1295
SAS 欠損値と null 値 1295
関数としての DS2 パッケージメソッド 1295
一貫性制約 1295
FedSQL DICTIONARY テーブル 1296

FedSQL の制限事項 **1296**
ユーザー権限の付与 1296
Base SAS、SPD Engine、および SPD Server の識別子と命名規則 1296

準拠

FedSQL は、SQL の 1999 ANSI 規格にコア準拠しています。[1]

1. 国際標準化機構(ISO): ドキュメント ISO/IEC 9075–1:1999。また、米国規格協会(ANSI)ドキュメント ANSI 9075–1:1999 としても入手できます。

FedSQL の拡張

予約語

FedSQL の予約語の完全なリストについては、“FedSQL 予約語” (72 ページ)を参照してください。

列修飾子

FedSQL は、CREATE TABLE ステートメントの HAVING 句で FORMAT=、INFORMAT=、および LABEL=修飾子をサポートします。これらの修飾子は、出力データの表示およびラベル付けの出力形式を制御します。 詳細については、“保存出力形式の保存、変更、削除、使用方法” (81 ページ)、 “FedSQL で入力形式を指定する方法” (926 ページ)、および“HAVING 句” (995 ページ)を参照してください。

FedSQL は、SAS FORMAT プロシジャで作成された出力形式の保存をサポートしていません。

FedSQL SELECT ステートメントは、既存のテーブルの出力形式とラベルを保持し、PUT 関数で一時出力形式を指定できるようにします。詳細については、“PUT 関数で出力をフォーマットする方法” (82 ページ)を参照してください。

関数

集計関数

FedSQL は、SQL の ANSI 規格で必要とされるよりも多くの集計関数をサポートしています。 FedSQL がサポートする集計関数の完全なリストについては、“集計関数” (260 ページ)を参照してください。

Base SAS DATA ステップ関数

FedSQL は、Base SAS DATA ステップで使用できる関数の多くをサポートしています。サポートされていない関数には、変数情報関数およびデータの配列を操作する

関数が含まれます。他の FedSQL データソースは、独自の関数セットをサポートしています。

FedSQL がサポートする関数の完全なリストについては、“FedSQL 関数の概要” (256 ページ)を参照してください。

テーブルオプション

テーブルオプションは、パスワードの割り当てや指定など、テーブルに対する操作を実行できるようにするアクションを指定します。FedSQL のテーブルオプションは、Base SAS データセットオプションとほぼ同じ機能を実行します。詳細については、“ステートメントテーブルオプションの概要” (1014 ページ)を参照してください。

SAS 欠損値と null 値

FedSQL は、SAS 欠損値と null 値の両方をサポートしています。存在しないデータは、SAS データセット、SPD Engine データセット、および SPD Server テーブルの SAS 欠損値によって表されます。他のすべてのデータソースでは、存在しないデータは ANSI SQL null 値で表されます。詳細については、“FedSQL が null および SAS 欠損値を処理する方法” (21 ページ)を参照してください。

関数としての DS2 パッケージメソッド

DS2 パッケージメソッドを FedSQL SELECT ステートメントの関数として呼び出すことができます。詳細については、“式での DS2 パッケージの使用” (258 ページ)を参照してください。

一貫性制約

一貫性制約は、データの有効性と整合性を維持するために指定できる一連のデータ検証規則です。CREATE TABLE ステートメントで指定された一貫性制約は、データソースに渡されます。トランザクションがテーブルを変更すると、データソースは一貫性制約を適用します。詳細については、“CREATE TABLE ステートメント” (939 ページ)を参照してください。

FedSQL DICTIONARY テーブル

FedSQL は、Base SAS で使用できる DICTIONARY テーブルと同様の DICTIONARY テーブルをサポートしています。FedSQL DICTIONARY テーブルは、列、テーブル、およびカタログに関する情報と、単一のテーブルとそれに関連付けられたインデックスに関する統計量を含む読み取り専用のテーブルです。詳細については、“DICTIONARY テーブル” (67 ページ)を参照してください。

FedSQL の制限事項

ユーザー権限の付与

GRANT ステートメント、PRIVILEGES キーワード、および SQL の権限識別子機能はサポートされていません。かわりに、動作環境固有のセキュリティ手段を使用することをお勧めします。

Base SAS、SPD Engine、および SPD Server の識別子と命名規則

SAS データセット、SPD Engine データセット、および SPD Server テーブルのテーブル、列、およびインデックスの名前は、32 文字に制限されています。 詳細については、“識別子” (18 ページ)を参照してください。

付録 7

使用上の注意

使用上の注意 **1297**
Teradata: Teradata での FedSQL のパフォーマンスの向上 1297

使用上の注意

Teradata: Teradata での FedSQL のパフォーマンスの向上

システムカタログテーブルにアクセスする FedSQL クエリ(たとえば、UPDATE ステートメント、DROP TABLE ステートメントなど)は、システムテーブルの結合に依存して、Teradata データベースから結果セットを取得します。多数のデータベースオブジェクトを作成および削除する環境では、これらのシステムテーブルに関する統計量が最新でない場合、パフォーマンスが低下します。次のコマンドを定期的なデータベースメンテナンスプロセスに追加して、システムテーブルの統計量を最新の状態に保つことができます。

```
drop stats on dbc.tvm;
drop stats on dbc.owners;
drop stats on dbc.dbase;
drop stats on dbc.accessrights;
drop stats on dbc.tvfields;

collect stats on dbc.tvm column (tvmId);
collect stats on dbc.tvm INDEX ( DatabaseId ,TVMNameI );
collect stats on dbc.tvm column (DatabaseId );
collect stats on dbc.owners INDEX (ownerId);
collect stats on dbc.dbase INDEX ( DatabaseId );
collect stats on dbc.dbase column(JournalId);
collect stats on dbc.accessrights INDEX ( UserId ,DatabaseId );
```

```
collect stats on dbc.accessrights INDEX ( TVMId );
collect stats on dbc.accessrights column ( UserId ,TVMId);
collect stats on dbc.accessrights column (DatabaseId );
collect stats on dbc.tvfields column (DatabaseId);
collect stats on dbc.tvfields column (FieldId);
collect stats on dbc.tvfields column (tableId);


/* help stats displays what's there, this is optional */

help stats dbc.tvm;
help stats dbc.owners;
help stats dbc.dbase;
help stats dbc.accessrights;
help stats dbc.tvfields;



/* routine recollect, if stats have not been dropped */

collect stats on dbc.tvm ;

collect stats on dbc.owners ;
collect stats on dbc.dbase ;
collect stats on dbc.accessrights ;

collect stats on dbc.tvfields ;
```

付録 8

ICU ライセンス

UNICODE ライセンス V3 . . . . . **1299**

Third-Party Software Licenses: ICU 1.8.1–ICU 57 and ICU4J 1.3.1–ICU4J 57 . . . . . **1300**
1.Unicode Data Files and Software 1300
2.Chinese/Japanese Word Break Dictionary Data (cjdict.txt) 1301
3.Lao Word Break Dictionary Data (laodict.txt) 1305
4.Burmese Word Break Dictionary Data (burmesedict.txt) 1305
3.Time Zone Database 1306

Unicode, Inc. License Agreement - Data Files and Software: ICU 63 and Later . **1307**

UNICODE ライセンス V3

Copyright © 1991-2024 Unicode, Inc.

Copyright (c) 1995-2015 International Business Machines Corporation and others

NOTICE TO USER: Carefully read the following legal agreement.BY DOWNLOADING, INSTALLING, COPYING OR OTHERWISE USING DATA FILES, AND/OR SOFTWARE, YOU UNEQUIVOCALLY ACCEPT, AND AGREE TO BE BOUND BY, ALL OF THE TERMS AND CONDITIONS OF THIS AGREEMENT.IF YOU DO NOT AGREE, DO NOT DOWNLOAD, INSTALL, COPY, DISTRIBUTE OR USE THE DATA FILES OR SOFTWARE.

Permission is hereby granted, free of charge, to any person obtaining a copy of data files and any associated documentation (the "Data Files") or software and any associated documentation (the "Software") to deal in the Data Files or Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Data Files or Software, and to permit persons to whom the Data Files or Software are furnished to do so, provided that either (a) this copyright and permission notice appear with all copies of the Data Files or Software, or (b) this copyright and permission notice appear in associated Documentation.

THE DATA FILES AND SOFTWARE ARE PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS.

IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THE DATA FILES OR SOFTWARE.

Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in these Data Files or Software without prior written authorization of the copyright holder.

Third-Party Software Licenses: ICU 1.8.1–ICU 57 and ICU4J 1.3.1–ICU4J 57

This section contains third-party software notices and/or additional terms for licensed third-party software components included within ICU libraries

1.Unicode Data Files and Software

COPYRIGHT AND PERMISSION NOTICE

Copyright © 1991-2015 Unicode, Inc. All rights reserved.Distributed under the Terms of Use in http://www.unicode.org/copyright.html.

Permission is hereby granted, free of charge, to any person obtaining a copy of the Unicode data files and any associated documentation (the "Data Files") or Unicode software and any associated documentation (the "Software") to deal in the Data Files or Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Data Files or Software, and to permit persons to whom the Data Files or Software are furnished to do so, provided that (a) the above copyright notice(s) and this permission notice appear with all copies of the Data Files or Software, (b) both the above copyright notice(s) and this permission notice appear in associated documentation, and (c) there is clear notice in each modified Data File or in the Software as well as in the documentation associated with the Data File(s) or Software that the data or software has been modified.

THE DATA FILES AND SOFTWARE ARE PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS.IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THE DATA FILES OR SOFTWARE.

Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in these Data Files or Software without prior written authorization of the copyright holder.

2.Chinese/Japanese Word Break Dictionary Data (cjdict.txt)

```
#  The Google Chrome software developed by Google is licensed under
#  the BSD license. Other software included in this distribution is provided
#  under other licenses, as set forth below.
#
#   The BSD License
#   http://opensource.org/licenses/bsd-license.php
#   Copyright (C) 2006-2008, Google Inc.
#
#   All rights reserved.
#
#   Redistribution and use in source and binary forms, with or without
#  modification, are permitted provided that the following conditions are met:
#
#   Redistributions of source code must retain the above copyright notice, this
#  list of conditions and the following disclaimer.
#   Redistributions in binary form must reproduce the above copyright notice,
#  this list of conditions and the following disclaimer in the documentation
#  and/or other materials provided with the distribution.
#   Neither the name of Google Inc. nor the names of its contributors may be
#  used to endorse or promote products derived from this software without
#  specific prior written permission.
#
#
#  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
#  AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
#  IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
#  ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
#  LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
#  CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
#  SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
#  INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
#  CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
```

```
#  ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF
#  THE POSSIBILITY OF SUCH DAMAGE.
#
#
#   The word list in cjdict.txt are generated by combining three word lists
#  listed below with further processing for compound word breaking. The
#  frequency is generated with an iterative training against Google
#  web corpora.
#
#   * Libtabe (Chinese)
#     - https://sourceforge.net/project/?group_id=1519
#     - Its license terms and conditions are shown below.
#
#   * IPADIC (Japanese)
#     - http://chasen.aist-nara.ac.jp/chasen/distribution.html
#     - Its license terms and conditions are shown below.
#
#   ---------COPYING.libtabe ---- BEGIN--------------------
#
#   /*
#    * Copyrighy (c) 1999 TaBE Project.
#    * Copyright (c) 1999 Pai-Hsiang Hsiao.
#    * All rights reserved.
#    *
#    * Redistribution and use in source and binary forms, with or without
#    * modification, are permitted provided that the following conditions
#    * are met:
#    *
#    * . Redistributions of source code must retain the above copyright
#    *   notice, this list of conditions and the following disclaimer.
#    * . Redistributions in binary form must reproduce the above copyright
#    *   notice, this list of conditions and the following disclaimer in
#    *   the documentation and/or other materials provided with the
#    *   distribution.
#    * . Neither the name of the TaBE Project nor the names of its
#    *   contributors may be used to endorse or promote products derived
#    *   from this software without specific prior written permission.
#    *
#    * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
#    * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
#    * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
#    * FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
#    * REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
#    * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
#    * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
#    * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
#    * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
#    * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
#    * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
#    * OF THE POSSIBILITY OF SUCH DAMAGE.
#    */
#
#   /*
#    * Copyright (c) 1999 Computer Systems and Communication Lab,
#    *                    Institute of Information Science, Academia Sinica.
#    * All rights reserved.
```

```
#    *
#    * Redistribution and use in source and binary forms, with or without
#    * modification, are permitted provided that the following conditions
#    * are met:
#    *
#    * . Redistributions of source code must retain the above copyright
#    *   notice, this list of conditions and the following disclaimer.
#    * . Redistributions in binary form must reproduce the above copyright
#    *   notice, this list of conditions and the following disclaimer in
#    *   the documentation and/or other materials provided with the
#    *   distribution.
#    * . Neither the name of the Computer Systems and Communication Lab
#    *   nor the names of its contributors may be used to endorse or
#    *   promote products derived from this software without specific
#    *   prior written permission.
#    *
#    * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
#    * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
#    * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
#    * FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
#    * REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
#    * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
#    * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
#    * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
#    * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
#    * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
#    * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
#    * OF THE POSSIBILITY OF SUCH DAMAGE.
#    */
#
#    Copyright 1996 Chih-Hao Tsai @ Beckman Institute, University of Illinois
#    c-tsai4@uiuc.edu  http://casper.beckman.uiuc.edu/~c-tsai4
#
#    ----------------COPYING.libtabe-----END----------------------------------
#
#
#    ----------------COPYING.ipadic-----BEGIN---------------------------------
#
#    Copyright 2000, 2001, 2002, 2003 Nara Institute of Science
#    and Technology.  All Rights Reserved.
#
#    Use, reproduction, and distribution of this software is permitted.
#    Any copy of this software, whether in its original form or modified,
#    must include both the above copyright notice and the following
#    paragraphs.
#
#    Nara Institute of Science and Technology (NAIST),
#    the copyright holders, disclaims all warranties with regard to this
#    software, including all implied warranties of merchantability and
#    fitness, in no event shall NAIST be liable for
#    any special, indirect or consequential damages or any damages
#    whatsoever resulting from loss of use, data or profits, whether in an
#    action of contract, negligence or other tortuous action, arising out
#    of or in connection with the use or performance of this software.
#
#    A large portion of the dictionary entries
```

```
#   originate from ICOT Free Software.  The following conditions for ICOT
#   Free Software applies to the current dictionary as well.
#
#   Each User may also freely distribute the Program, whether in its
#   original form or modified, to any third party or parties, PROVIDED
#   that the provisions of Section 3 ("NO WARRANTY") will ALWAYS appear
#   on, or be attached to, the Program, which is distributed substantially
#   in the same form as set out herein and that such intended
#   distribution, if actually made, will neither violate or otherwise
#   contravene any of the laws and regulations of the countries having
#   jurisdiction over the User or the intended distribution itself.
#
#   NO WARRANTY
#
#   The program was produced on an experimental basis in the course of the
#   research and development conducted during the project and is provided
#   to users as so produced on an experimental basis.  Accordingly, the
#   program is provided without any warranty whatsoever, whether express,
#   implied, statutory or otherwise.  The term "warranty" used herein
#   includes, but is not limited to, any warranty of the quality,
#   performance, merchantability and fitness for a particular purpose of
#   the program and the nonexistence of any infringement or violation of
#   any right of any third party.
#
#   Each user of the program will agree and understand, and be deemed to
#   have agreed and understood, that there is no warranty whatsoever for
#   the program and, accordingly, the entire risk arising from or
#   otherwise connected with the program is assumed by the user.
#
#   Therefore, neither ICOT, the copyright holder, or any other
#   organization that participated in or was otherwise related to the
#   development of the program and their respective officials, directors,
#   officers and other employees shall be held liable for any and all
#   damages, including, without limitation, general, special, incidental
#   and consequential damages, arising out of or otherwise in connection
#   with the use or inability to use the program or any product, material
#   or result produced or otherwise obtained by using the program,
#   regardless of whether they have been advised of, or otherwise had
#   knowledge of, the possibility of such damages at any time during the
#   project or thereafter.  Each user will be deemed to have agreed to the
#   foregoing by his or her commencement of use of the program.  The term
#   "use" as used herein includes, but is not limited to, the use,
#   modification, copying and distribution of the program and the
#   production of secondary products from the program.
#
#   In the case where the program, whether in its original form or
#   modified, was distributed or delivered to or received by a user from
#   any person, organization or entity other than ICOT, unless it makes or
#   grants independently of ICOT any specific warranty to the user in
#   writing, such person, organization or entity, will also be exempted
#   from and not be held liable to the user for any such damages as noted
#   above as far as the program is concerned.
#
#   ----------COPYING.ipadic-----END----------------------------------
```

3.Lao Word Break Dictionary Data (laodict.txt)

```
#  Copyright (c) 2013 International Business Machines Corporation
#  and others. All Rights Reserved.
#
#  Project:    http://code.google.com/p/lao-dictionary/
#  Dictionary: http://lao-dictionary.googlecode.com/git/Lao-Dictionary.txt
#  License:    http://lao-dictionary.googlecode.com/git/Lao-Dictionary-LICENSE.txt
#              (copied below)
#
#  This file is derived from the above dictionary, with slight modifications.
#  --------------------------------------------------------------------------------
#  Copyright (C) 2013 Brian Eugene Wilson, Robert Martin Campbell.
#  All rights reserved.
#
#  Redistribution and use in source and binary forms, with or without modification,
#  are permitted provided that the following conditions are met:
#
#     Redistributions of source code must retain the above copyright notice, this
#     list of conditions and the following disclaimer. Redistributions in binary
#     form must reproduce the above copyright notice, this list of conditions and
#     the following disclaimer in the documentation and/or other materials
#     provided with the distribution.
#
#  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
#  ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
#  WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
#  DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
#  ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
#  (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
#  LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
#  ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
#  (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
#  SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
#  --------------------------------------------------------------------------------
```

4.Burmese Word Break Dictionary Data (burmesedict.txt)

```
#  Copyright (c) 2014 International Business Machines Corporation
#  and others. All Rights Reserved.
#
#  This list is part of a project hosted at:
#    github.com/kanyawtech/myanmar-karen-word-lists
#
#  --------------------------------------------------------------------------------
#  Copyright (C) 2013 LeRoy Benjamin Sharon
```

```
#    All rights reserved.
#
#    Redistribution and use in source and binary forms, with or without
#    modification, are permitted provided that the following conditions
# are met: Redistributions of source code must retain the above
#    copyright notice, this list of conditions and the following
#    disclaimer.  Redistributions in binary form must reproduce the
# above copyright notice, this list of conditions and the following
#    disclaimer in the documentation and/or other materials provided
# with the distribution
#
#       Neither the name Myanmar Karen Word Lists, nor the names of its
#       contributors may be used to endorse or promote products derived
#       from this software without specific prior written permission.
#
#    THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
#    ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
#    WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
#    DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
#    ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
#    (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
#    LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
#    ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
#    (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
#    SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
#    ------------------------------------------------------------------------------
```

3.Time Zone Database

ICU uses the public domain data and code derived from Time Zone Database for its time zone support.The ownership of the TZ database is explained in BCP 175: Procedure for Maintaining the Time Zone Database section 7.

7. Database Ownership

> The TZ database itself is not an IETF Contribution or an IETF document. Rather it is a pre-existing and regularly updated work that is in the public domain, and is intended to remain in the public domain. Therefore, BCPs 78 [RFC5378] and 79 [RFC3979] do not apply to the TZ Database or contributions that individuals make to it. Should any claims be made and substantiated against the TZ Database, the organization that is providing the IANA Considerations defined in this RFC, under the memorandum of understanding with the IETF, currently ICANN, may act in accordance with all competent court orders. No ownership claims will be made by ICANN or the IETF Trust on the database or the code. Any person making a contribution to the database or code waives all rights to future claims in that contribution or in the TZ Database.

Unicode, Inc. License Agreement - Data Files and Software: ICU 63 and Later

See Terms of Use for definitions of Unicode Inc.'s Data Files and Software.

NOTICE TO USER: Carefully read the following legal agreement.BY DOWNLOADING, INSTALLING, COPYING OR OTHERWISE USING UNICODE INC.'S DATA FILES ("DATA FILES"), AND/OR SOFTWARE ("SOFTWARE"), YOU UNEQUIVOCALLY ACCEPT, AND AGREE TO BE BOUND BY, ALL OF THE TERMS AND CONDITIONS OF THIS AGREEMENT.IF YOU DO NOT AGREE, DO NOT DOWNLOAD, INSTALL, COPY, DISTRIBUTE OR USE THE DATA FILES OR SOFTWARE.

COPYRIGHT AND PERMISSION NOTICE

Copyright © 1991-2020 Unicode, Inc. All rights reserved.Distributed under the Terms of Use in http://www.unicode.org/copyright.html.

Permission is hereby granted, free of charge, to any person obtaining a copy of the Unicode data files and any associated documentation (the "Data Files") or Unicode software and any associated documentation (the "Software") to deal in the Data Files or Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Data Files or Software, and to permit persons to whom the Data Files or Software are furnished to do so, provided that (a) the above copyright notice(s) and this permission notice appear with all copies of the Data Files or Software, or (b) this copyright and permission notice appear in associated Documentation.

THE DATA FILES AND SOFTWARE ARE PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS.IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THE DATA FILES OR SOFTWARE.

Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in these Data Files or Software without prior written authorization of the copyright holder.