IBM® DB2 Universal Database™

SQL リファレンス
第 1 巻

バージョン 8.2

SC88-9155-01
(英文原典：SC09-4844-01)

IBM® DB2 Universal Database™

SQL リファレンス
第 1 巻

バージョン 8.2

SC88-9155-01
(英文原典：SC09-4844-01)

ご注意！

本書および本書で紹介する製品をご使用になる前に、『*特記事項*』に記載されている情報をお読みください。

本マニュアルに関するご意見やご感想は、次の URL からお送りください。今後の参考にさせていただきます。

`http://www.ibm.com/jp/manuals/main/mail.html`

なお、日本 IBM 発行のマニュアルはインターネット経由でもご購入いただけます。詳しくは

`http://www.ibm.com/jp/manuals/` の「ご注文について」をご覧ください。

(URL は、変更になる場合があります)

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原　典： SC09–4844–01
IBM® DB2 Universal Database™
SQL Reference Volume 1
Version 8.2

発　行： 日本アイ・ビー・エム株式会社

担　当： ナショナル・ランゲージ・サポート

第1刷 2004.8

目次

本書について

2 つの巻に分かれた SQL リファレンスでは、 DB2 Universal Database バージョン 8 によって使用される SQL 言語が定義されていますが、それぞれの内容は以下のとおりです。

- リレーショナル・データベースの概念、言語エレメント、関数、および照会の形式 (第 1 巻)。
- SQL ステートメントの構文と意味に関する解説 (第 2 巻)。

本書の対象読者

本書は、構造化照会言語 (SQL) を使用してデータベースにアクセスする方々を対象にしています。対象読者は主にプログラマーとデータベース管理者の方々ですが、コマンド行プロセッサー (CLP) を介してデータベースにアクセスする方々にも役に立ちます。

本書は学習用ではなく、参照資料です。読者がアプリケーション・プログラムを作成することを前提にしており、したがって、データベース・マネージャーの機能を詳細に説明しています。

本書の構成

本書に記載されている内容は、次のような項目に大別されます。

- 1 ページの『第 1 章 概念』では、リレーショナル・データベースと SQL の基本概念について説明します。
- 61 ページの『第 2 章 言語エレメント』では、多くの SQL ステートメントに共通する SQL および言語エレメントの基本的な構文を説明します。
- 243 ページの『第 3 章 関数』では、SQL の列およびスカラー関数の構文図、意味の説明、規則、および使用例について説明します。
- 473 ページの『第 4 章 照会』では、様々な形式の照会について説明します。
- 527 ページの『付録 A. SQL の制限値』には、SQL の制約事項を一覧で示しています。
- 533 ページの『付録 B. SQLCA (SQL 連絡域)』では、SQLCA 構造について説明します。
- 539 ページの『付録 C. SQLDA (SQL 記述子域)』では、SQLDA 構造について説明します。
- 549 ページの『付録 D. カタログ・ビュー』では、データベース・カタログ・ビューについて説明します。
- 671 ページの『付録 E. フェデレーテッド・システム』では、フェデレーテッド・システムのオプションとタイプ割り当てについて説明します。
- 727 ページの『付録 F. SAMPLE データベース』では、サンプル表について説明します。

- 743 ページの『付録 G. 予約済みスキーマ名と予約語』では、 IBM SQL および ISO/ANSI SQL99 標準規格の予約スキーマ名と予約語について示します。
- 747 ページの『付録 H. トリガーと制約の相互作用』では、トリガーと参照制約の相互作用について説明します。
- 751 ページの『付録 I. Explain 表』では、Explain 表について説明します。
- 779 ページの『付録 J. EXPLAIN レジスター値』では、 CURRENT EXPLAIN MODE および CURRENT EXPLAIN SNAPSHOT 特殊レジスターの値の相互作用、また PREP および BIND コマンドとの相互作用を説明します。
- 787 ページの『付録 K. 例外表』では、SET INTEGRITY ステートメントで使用する、ユーザー作成の表に関する情報を示します。
- 791 ページの『付録 L. ルーチンで使用可能な SQL ステートメント』には、さまざまな SQL データ・アクセス・コンテキストをもったルーチンにおいて実行できる SQL ステートメントの一覧が示されています。
- 795 ページの『付録 M. コンパイル済みステートメントから呼び出される CALL』は、コンパイル済みステートメントから呼び出せる CALL ステートメントを説明しています。
- 801 ページの『付録 N. 日本語および中国語 (繁体字) の拡張 UNIX コード (EUC) の考慮事項』は、拡張 UNIX コード (EUC) 文字セットの使用時の考慮事項の一覧を示しています。
- 809 ページの『付録 O. DATALINK のバッカス正規形式 (BNF) 仕様』では、DATALINK のバッカス正規形式 (BNF) 仕様について説明します。

第 2 巻の概略

SQL リファレンスの第 2 巻では、SQL ステートメントの構文と意味に関する解説が述べられています。以下に、第 2 巻の各章について簡単に説明してあります。

- 『SQL ステートメント』では、すべての SQL ステートメントの構文図、意味の説明、規則、および例について説明します。
- 『SQL 制御ステートメント』では、SQL プロシージャー・ステートメントの構文図、意味の説明、規則、および例について説明します。

構文図の見方

本書全体を通して、次のように定義された構造図を使用して構文を説明しています。

構文図は、左から右、上から下に、線に沿って読みます。

記号 は、構文図の始まりを示します。

記号 は、構文が次の行に続くことを示します。

記号 は、構文が前の行から続いていることを示します。

記号 は、構文図の終わりを示します。

構文ブロックは、記号 で始まり、記号 で終わります。

必須項目は、横線 (メインパス) 上に示されます。

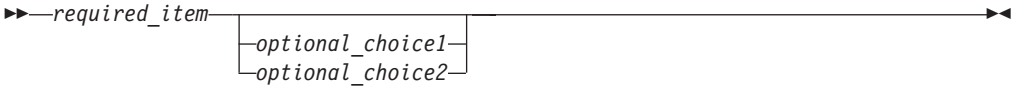
オプション項目は、メインパスの下に示されます。

オプション項目をメインパスの上に示すこともありますが、それは構文図を見やすくするためであり、実行には関係しません。

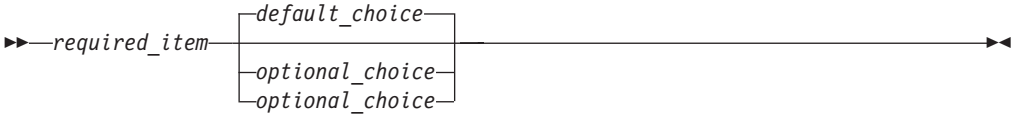
複数の項目からの選択が可能な場合、それらの項目を縦に並べて (スタックに) 示しています。

項目から 1 つを選択しなければならない 場合、スタックの項目の 1 つはメインパス上に示されます。

項目から 1 つをオプションで選択できる場合、スタック全体がメインパスよりも下に示されます。



項目の 1 つがデフォルト値の場合、その項目はメインパスより上に示され、残りの選択項目はメインパスよりも下に示されます。



メインパスの上に、左へ戻る矢印がある場合には、項目を繰り返して指定できることを示しています。このような場合、繰り返す項目相互の間は、1 つ以上のブランクで区切らなければなりません。

繰り返しの矢印にコンマが示されている場合は、繰り返し項目をコンマで区切らなければなりません。

スタックの上部の反復の矢印の記号は、そのスタックの中から複数の項目を選択できること、または 1 つの選択項目を繰り返して選択できることを示します。

キーワードは英大文字で示してあります (例: FROM)。示されているとおりに入力する必要があります。変数は英小文字で示しています (例: column-name)。このような変数は、構文にユーザーが指定する名前や値を示しています。

句読点、括弧、算術演算子、その他の記号が示されている場合には、それらを構文の一部として入力する必要があります。

1 つの変数が、構文を構成する大きいフラグメントを表すことがあります。たとえば次の図で、変数 parameter-block は、 **parameter-block** というラベルの構文フラグメント全体を表します。

parameter-block:

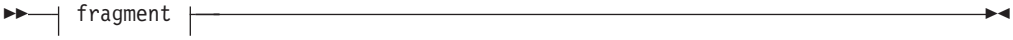
「黒丸」(●) ではさまれて隣接しているセグメントは、任意の順序で指定することができます。

上記の図は、項目 2 と 項目 3 をどのような順序で指定しても構わないことを示しています。以下はいずれも有効です。

```
required_item item1 item2 item3 item4
required_item item1 item3 item2 item4
```

共通構文エレメント

以下の項では、構文図で使用されるいくつかの構文フラグメントについて説明します。フラグメントは、以下のように参照されます。

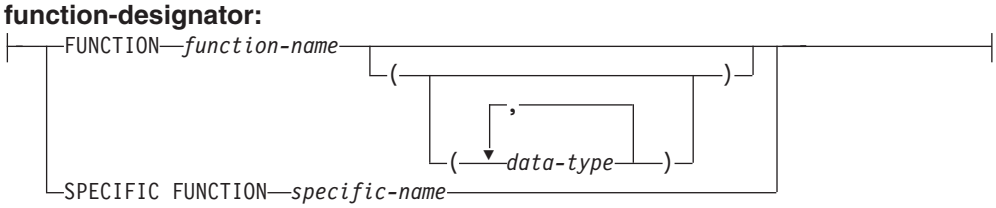


関数識別子

関数識別子は、単一の関数を固有指定します。関数識別子は通常、関数の DDL ステートメント (DROP や ALTER など) で使用されます。

構文:

function-designator:



説明:

FUNCTION *function-name*
特定の関数を指定します。スキーマ内の名前 *function-name* (関数名) の関数イン

スタンスが 1 つだけ存在している場合にのみ有効です。指定する関数には、任意の数のパラメーターを定義できます。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターは、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/BIND オプションによって、修飾子のないオブジェクト名の修飾子が暗黙指定されます。指定したスキーマまたは暗黙のスキーマにこの名前の関数が存在しない場合は、エラー (SQLSTATE 42704) になります。指定したスキーマまたは暗黙のスキーマに、この関数のインスタンスが複数存在する場合は、エラー (SQLSTATE 42725) になります。

FUNCTION *function-name (data-type,...)*

関数を固有に指定する関数シグニチャーを指定します。関数解決のアルゴリズムは使用されません。

function-name

関数の名前を指定します。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターは、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/BIND オプションによって、修飾子のないオブジェクト名の修飾子が暗黙指定されます。

(data-type,...)

値は、CREATE FUNCTION ステートメントの (対応する位置に) 指定されたデータ型に一致していなければなりません。データ型の数、およびデータ型を論理的に連結した値を使用して、特定の関数インスタンスが識別されます。

データ型が修飾なしの場合は、SQL パスでスキーマを検索してタイプ名が決定されます。 REFERENCE タイプに指定するデータ型名にも同様の規則が当てはまります。

パラメーター化データ型の長さ、精度、または位取りを指定する必要はありません。空の括弧をコーディングすることによって、一致するデータ型の検索時にそれらの属性を無視するように指定することができます。

パラメーター値が異なるデータ型 (REAL または DOUBLE) を示しているため、 FLOAT() を使用することはできません (SQLSTATE 42601)。

長さ、精度、または位取りをコーディングする場合、その値は、CREATE FUNCTION ステートメントで指定された値と完全に一致していなければなりません。

0 < *n* < 25 は REAL を意味し、24 < *n* < 54 は DOUBLE を意味するので、 FLOAT(*n*) のタイプは、*n* に定義された値と一致している必要はありません。タイプが REAL か DOUBLE かによって、生じる一致は異なってきます。

指定したスキーマまたは暗黙のスキーマに、指定したシグニチャーを持つ関数がない場合は、エラー (SQLSTATE 42883) になります。

SPECIFIC FUNCTION *specific-name*

関数の作成時に指定された名前、またはデフォルト値として使用された名前を使用して、特定のユーザー定義関数を指定します。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターは、修飾子のないオブジェクト名の修

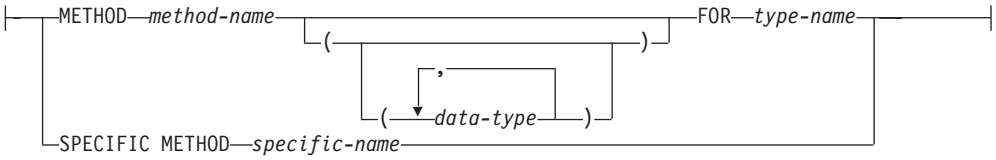
飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/BIND オプションによって、修飾子のないオブジェクト名の修飾子が暗黙指定されます。 *specific-name* (特定名) は、指定したスキーマまたは暗黙のスキーマの特定関数のインスタンスを指定していなければなりません。そうでない場合、エラー (SQLSTATE 42704) になります。

メソッド識別子

メソッド識別子は、単一のメソッドを固有指定します。メソッド識別子は通常、メソッドの DDL ステートメント (DROP や ALTER など) で使用されます。

構文:

method-designator:



説明:

METHOD *method-name*
特定のメソッドを指定します。これは、タイプ *type-name* に対する名前 *method-name* のメソッド・インスタンスが 1 つだけ存在している場合にのみ有効です。指定するメソッドには、任意の数のパラメーターを定義できます。タイプに、指定された名前のメソッドが存在しない場合は、エラーが戻されます (SQLSTATE 42704)。タイプに、そのメソッドのインスタンスが複数存在する場合も、エラーが戻されます (SQLSTATE 42725)。

METHOD *method-name (data-type,...)*
メソッドを固有識別できるメソッド・シグニチャーを指定します。メソッド解決のアルゴリズムは使用されません。

method-name
タイプ *type-name* のメソッドの名前を指定します。

(data-type,...)
値は、CREATE TYPE ステートメントの (対応する位置に) 指定されたデータ型に一致していなければなりません。データ型の数、およびデータ型を論理的に連結した値を使用して、特定のメソッドが識別されます。

データ型が修飾なしの場合は、SQL パスでスキーマを検索してタイプ名が決定されます。 REFERENCE タイプに指定するデータ型名にも同様の規則が当てはまります。

パラメーター化データ型の長さ、精度、または位取りを指定する必要はありません。空の括弧をコーディングすることによって、一致するデータ型の検索時にそれらの属性を無視するように指定することができます。

パラメーター値が異なるデータ型 (REAL または DOUBLE) を示しているため、 FLOAT() を使用することはできません (SQLSTATE 42601)。

長さ、精度、または位取りをコーディングする場合、その値は、CREATE TYPE ステートメントで指定された値と完全に一致していなければなりません。

0 < *n* < 25 は REAL を意味し、24 < *n* < 54 は DOUBLE を意味するので、 FLOAT(*n*) のタイプは、*n* に定義された値と一致している必要はありません。タイプが REAL か DOUBLE かによって、生じる一致は異なってきます。

指定したスキーマまたは暗黙のスキーマに、そのタイプに対して指定したシグニチャーを持つメソッドがない場合は、エラー (SQLSTATE 42883) になります。

FOR *type-name*
指定されたメソッドを関連付けるタイプを指定します。ここで指定される名前は、カタログにすでに記述されているタイプを示すものでなければなりません (SQLSTATE 42704)。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターは、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/BIND オプションによって、修飾子のないオブジェクト名の修飾子が暗黙指定されます。

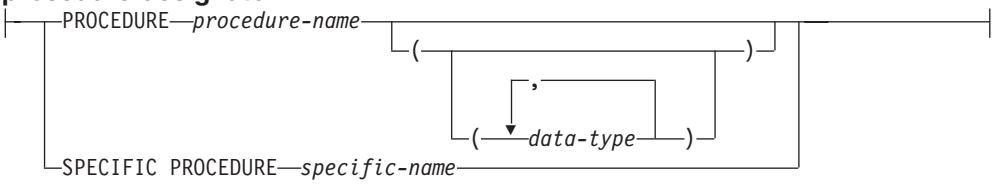
SPECIFIC METHOD *specific-name*
メソッドの作成時に指定された名前、またはデフォルト値として使用された名前を使用して、特定のメソッドを指定します。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターは、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/BIND オプションによって、修飾子のないオブジェクト名の修飾子が暗黙指定されます。 *specific-name* (特定名) は、指定したスキーマまたは暗黙のスキーマの特定メソッドのインスタンスを指定していなければなりません。そうでない場合、エラー (SQLSTATE 42704) になります。

プロシージャー識別子

プロシージャー識別子は、単一のプロシージャーを固有指定します。プロシージャー識別子は通常、プロシージャーの DDL ステートメント (DROP や ALTER など) で使用されます。

構文:

procedure-designator:



説明:

PROCEDURE *procedure-name*
特定のプロシージャーを指定します。スキーマ内の名前 *procedure-name* (プロシージャー名) のプロシージャー・インスタンスが 1 つだけ存在している場合にのみ有効です。指定するプロシージャーには、任意の数のパラメーターを定義できます。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターは、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/BIND オプションによって、修飾

子のないオブジェクト名の修飾子が暗黙指定されます。指定したスキーマまたは暗黙のスキーマに該当する名前のプロシージャーが存在しない場合は、エラーが戻されます (SQLSTATE 42704)。指定したスキーマまたは暗黙のスキーマに、このプロシージャーのインスタンスが複数存在する場合は、エラー (SQLSTATE 42725) になります。

PROCEDURE *procedure-name (data-type,...)*
プロシージャーを固有識別するプロシージャー・シグニチャーを指定します。プロシージャー解決のアルゴリズムは使用されません。

procedure-name
プロシージャーの名前を指定します。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターは、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/BIND オプションによって、修飾子のないオブジェクト名の修飾子が暗黙指定されます。

(data-type,...)
値は、CREATE PROCEDURE ステートメントの (対応する位置に) 指定されたデータ型に一致していなければなりません。データ型の数、およびデータ型を論理的に連結した値を使用して、特定のプロシージャーが識別されます。

データ型が修飾なしの場合は、SQL パスでスキーマを検索してタイプ名が決定されます。 REFERENCE タイプに指定するデータ型名にも同様の規則が当てはまります。

パラメーター化データ型の長さ、精度、または位取りを指定する必要はありません。空の括弧をコーディングすることによって、一致するデータ型の検索時にそれらの属性を無視するように指定することができます。

データ型 FLOAT と空の括弧の組み合わせ ″FLOAT ()″ は、指定することができません (SQLSTATE 42601)。これは、パラメーター値によってデータ型が REAL または DOUBLE に変わるためです。

長さ、精度、または位取りをコーディングする場合、その値は、CREATE PROCEDURE ステートメントで指定された値と完全に一致していなければなりません。

$0 < n < 25$ は REAL を意味し、$24 < n < 54$ は DOUBLE を意味するので、 FLOAT(*n*) のタイプは、*n* に定義された値と一致している必要はありません。タイプが REAL か DOUBLE かによって、生じる一致は異なってきます。

指定したスキーマまたは暗黙のスキーマに、指定したシグニチャーを持つプロシージャーがない場合は、エラー (SQLSTATE 42883) になります。

SPECIFIC PROCEDURE *specific-name*
プロシージャーの作成時に指定された名前、またはデフォルト値として使用された名前を使用して、特定のプロシージャーを指定します。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターは、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/BIND オプションによって、修飾子のないオブジェクト名の修飾子が暗黙指定されます。 *specific-name* に指定される名前は、指定したスキー

マまたは暗黙のスキーマに内の特定プロシージャーのインスタンスを識別するものでなければなりません。それ以外の名前が指定された場合は、エラーが戻されます (SQLSTATE 42704)。

本書の表記規則

この項では、本書全体で使用する表記規則について説明します。

エラー条件

本書の本文中では、エラーに対応する SQLSTATE を括弧で囲んでエラー条件が示されています。たとえば、

```
A duplicate signature raises an SQL error (SQLSTATE 42723)
```

強調表記の規則

本書では、以下の規則を使用しています。

太字	コマンド、キーワード、およびその名前がシステムによって事前定義されているその他の項目を示します。
イタリック	次のいずれかを示します。 • ユーザーが指定する名前または値 (変数) • 一般的な強調 • 新しい用語の紹介 • 他の情報源への参照
モノスペース	次のいずれかを示します。 • ファイルおよびディレクトリー • コマンド・プロンプトまたはウィンドウで入力するよう指示される情報 • 特定のデータ値の例 • システムが表示するテキストの例 • システム・メッセージの例

参考資料

アプリケーションの作成には、以下の資料が役立ちます。

- *管理ガイド*
  - ローカル・アクセスやクライアント/サーバー環境でのアクセス対象のデータベースの設計、実現、および保守を行うのに必要な情報が記載されています。
- *アプリケーション開発ガイド*
  - アプリケーションの開発プロセスについて述べ、組み込み SQL と API を使用してデータベースにアクセスするアプリケーション・プログラムをコーディング、コンパイル、および実行する方法について説明しています。
- *DB2 Universal Database for iSeries SQL 解説書*
  - iSeries (AS/400) 上で DB2 Query Manager および SQL Development Kit によってサポートされる構造化照会言語 (SQL) を定義しています。システム管理、データベース管理、アプリケーション・プログラミング、および操作に関

するタスクについての参照情報が記載されています。 DB2 を実行している iSeries (AS/400) システムで使用される SQL ステートメントごとに、構文、使用上の注意、キーワード、および例が記述されています。

- *DB2 Universal Database for z/OS and OS/390 SQL Reference*
  - DB2 for z/OS and OS/390 で使用される構造化照会言語 (SQL) を定義しています。 DB2 を実行している z/OS および OS/390 システムにおける照会の形式、SQL ステートメント、 SQL プロシージャー・ステートメント、DB2 の制限、SQLCA、 SQLDA、カタログ表、および SQL 予約語が記載されています。
- *DB2 Spatial Extender ユーザーズ・ガイド*
  - 本書は、地理情報システム (GIS) を作成して使うアプリケーションを作成する方法を説明しています。 GIS を作成して使用するには、データベースにリソースを提供し、そのデータを照会して、区域内の位置、距離、および分散などの情報を入手する必要があります。
- *IBM SQL リファレンス*
  - 本書は、データベース製品の IBM 全般に共通するすべての SQL エレメントについて説明しています。 IBM のデータベースを使用して可搬プログラムを作成するのに役立つ制約事項と規則が示されています。この資料は、さまざまな規格や製品 (SQL92E、XPG4-SQL、IBM-SQL および IBM リレーショナル・データベースの製品) の間での SQL の拡張機能や非互換性について、リストの形で示されています。
- *American National Standard X3.135-1992, Database Language SQL*
  - SQL の ANSI 規格の定義について説明しています。
- *ISO/IEC 9075:1992, Database Language SQL*
  - SQL の 1992 ISO 規格の定義について説明しています。
- *ISO/IEC 9075-2:1999, Database Language SQL - Part 2: Foundation (SQL/Foundation)*
  - SQL の 1999 ISO 規格の定義についてその大部分を網羅しています。
- *ISO/IEC 9075-4:1999, Database Language SQL - Part 4: Persistent Stored Modules (SQL/PSM)*
  - SQL プロシージャー制御ステートメントの 1999 ISO 規格の定義について説明しています。
- *ISO/IEC 9075-5:1999, Database Language SQL - Part 4: Host Language Bindings (SQL/Bindings)*
  - ホスト言語バインディングと動的 SQL の 1999 ISO 規格の定義について説明しています。

第 1 章 概念

この章では、構造化照会言語 (SQL) の使用時に理解している必要のある高レベルの概念を示しています。本書においてこの後に記載されている参照マテリアルでは、それに関してさらに詳しく解説しています。

リレーショナル・データベース

リレーショナル・データベース とは、表の集まりとして扱われるデータベースであり、データのリレーショナル・モデルに従って操作することができます。リレーショナル・データベースには、データの保管、管理、およびアクセスに使用される一連のオブジェクトが収められます。そのようなオブジェクトの例として、表、ビュー、索引、関数、トリガー、パッケージなどがあります。

パーティション・リレーショナル・データベースとは、複数のパーティション (ノードともいう) にまたがってデータが管理されるリレーショナル・データベースです。このように複数のパーティションにまたがってデータを分離しても、大部分の SQL ステートメントのユーザーには影響ありません。ただし、データ定義言語 (DDL) ステートメントにはパーティション情報を考慮に入れるものもあります (たとえば、CREATE DATABASE PARTITION GROUP)。 (データ定義言語 (DLL) とは、データベース中のデータの関係を記述するのに使用する SQL ステートメントのサブセットです。)

フェデレーテッド・データベースとは、データが複数のデータ・ソース (別個のリレーショナル・データベースなど) に格納されるリレーショナル・データベースのことです。データは、すべて 1 つの大規模なデータベース内にあるかのように扱うことができ、従来の SQL 照会でアクセスすることができます。データに対する変更は、該当するデータ・ソースへ明示的に送られます。

構造化照会言語 (SQL)

SQL は、リレーショナル・データベースのデータの定義と操作を行うための標準化された言語です。データのリレーショナル・モデルに従って、データベースは表の集まりとして扱うことができ、関係は表の中の各値で表され、データは 1 つまたは複数の基本表から派生する結果表を指定することによって検索されます。

SQL ステートメントは、データベース・マネージャーによって実行されます。データベース・マネージャーの機能の 1 つは、結果表の仕様を、データ検索を最適化する一連の内部命令に変換することです。この変換は、準備処理およびバインドの 2 つのフェーズで行われます。

実行可能な SQL ステートメントはすべて、その実行に先立って準備しておく必要があります。その準備の結果は、ステートメントの実行可能形式または操作可能形式です。 SQL ステートメントを準備する方式とその操作可能形式の持続の程度の違いによって、静的 SQL と動的 SQL とがあります。

特権、権限レベル、およびデータベース権限

特権 は、ユーザーがデータベース・リソースを作成したりデータベース・リソースにアクセスしたりすることを許可するためのものです。*権限レベル* によって、特権のグループ分けの方法、およびより高いレベルのデータベース・マネージャーの保守とユーティリティー操作の権利が得られます。データベース*権限* は、ユーザーがデータベース・レベルのアクティビティーを実行できるようにします。特権、権限レベル、およびデータベース権限を組み合わせて使用することで、データベース・マネージャーとそのデータベース・オブジェクトへのアクセスを制御できます。ユーザーは、必要な特権、権限レベル、またはデータベース権限が与えられているオブジェクトに対してのみアクセスできます。 DB2® Universal Database (DB2 UDB) は、認証されるユーザーに関してこれらの許可検査を実行します。

データベース・マネージャーでは、特定のタスクを実行するのに必要なデータベース機能を使用するために、各ユーザーが特定の許可を暗黙または明示的に与えられていなければなりません。*明示的* な権限あるいは特権は、ユーザーに対して付与されます (データベース・カタログでは GRANTEETYPE が U)。*暗黙* の権限あるいは特権は、各ユーザーが所属するグループに対して付与されます (データベース・カタログでは GRANTEETYPE が G)。たとえば、表を作成するには、表作成のための許可がユーザーに必要です。表を変更するには、表の変更を許可されていなければなりません。

図 1 は、権限とその制御の範囲 (データベース、データベース・マネージャー) の間の関係を示します。

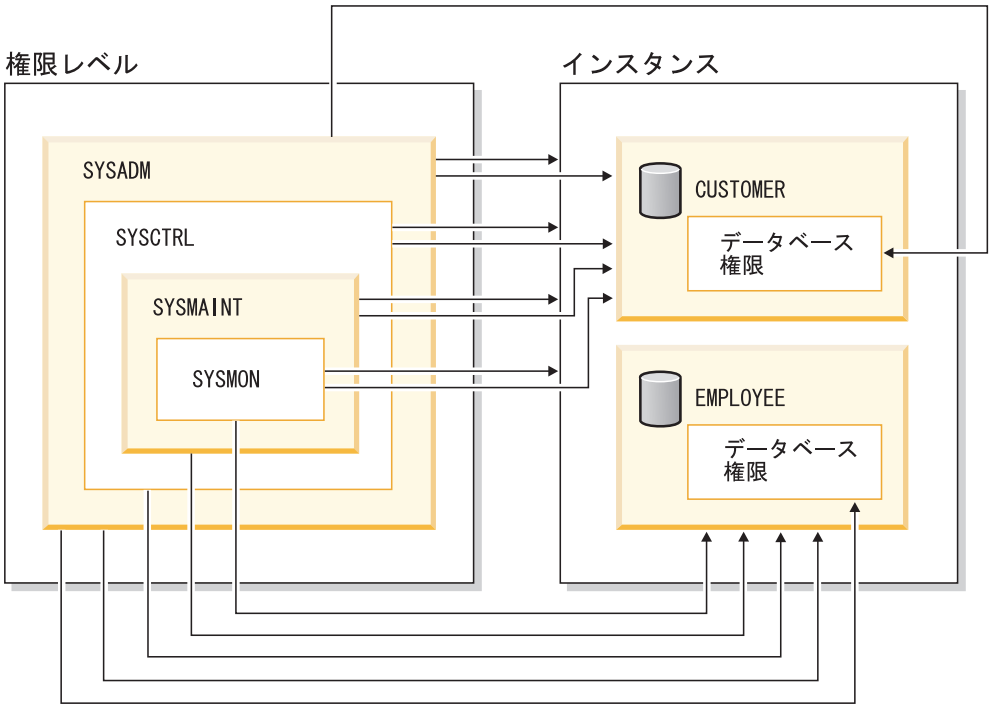


図 1. 権限の階層

各ユーザーまたはグループは、以下のような 1 つまたは複数の権限または特権を持つことができます。

- 管理権限:
  - SYSADM (システム管理者)

    SYSADM 権限レベルは、データベース・マネージャーによって作成および保守されるすべてのリソースに対する制御を与えます。システム管理者は DBADM、SYSCTRL、SYSMAINT、および SYSMON 権限をすべて所有し、DBADM 権限を付与または取り消す権限を持っています。

    SYSADM 権限を持つユーザーは、データベース・マネージャーの制御、およびデータの保護と保全性を担当します。 SYSADM 権限を持っている場合、データベース内のすべてのオブジェクトに対する暗黙特権が与えられ、どのユーザーがデータベース・マネージャーにアクセスできるか、およびそのアクセスの程度を制御することができます。 SYSADM 権限についての詳細は、 「システム管理権限 (SYSADM)」を参照してください。
  - DBADM (データベース管理者)

    DBADM データベース権限は、1 つのデータベースに対する管理権限を与えます。このデータベース管理者は、オブジェクトの作成、データベース・コマンドの発行、および表データへのアクセスに必要な権限を所有します。また、データベース管理者は、 CONTROL や個々の特権を付与または取り消すことができます。 DBADM 権限についての詳細は、 「データベース管理権限 (DBADM)」を参照してください。
- システム制御権限:
  - SYSCTRL (システム制御)

    SYSCTRL 権限レベルは、システム・リソースに影響を与える操作に対する制御を可能にします。たとえばは、SYSCTRL 権限を持つユーザーは、データベースの作成、更新、停止、またはドロップを行うことができます。さらに、このユーザーはインスタンスの停止を行うことができますが、表データへのアクセスはできません。 SYSCTRL 権限を持つユーザーには、SYSMON もまた与えられます。 SYSCTRL 権限についての詳細は、 「システム制御権限 (SYSCTRL)」を参照してください。
  - SYSMAINT (システム保守)

    SYSMAINT 権限レベルは、インスタンスに関連したすべてのデータベースに対する保守操作を実行するのに必要な権限を与えます。 SYSMAINT 権限を持つユーザーは、データベースの更新と構成、データベースまたは表スペースのバックアップ、既存のデータベースのリストア、およびデータベースのモニターを行うことができます。 SYSCTRL と同様に、SYSMAINT は表データへのアクセス権限を与えません。 SYSMAINT 権限を持つユーザーには、SYSMON 権限もまた与えられます。 SYSMAINT 権限についての詳細は、 「システム保守権限 (SYSMAINT)」を参照してください。
- SYSMON (システム・モニター権限)

  SYSMON 権限レベルは、データベース・システム・モニターの使用に必要な権限を与えます。 SYSMON 権限についての詳細は、 「システム・モニター権限 (SYSMON)」を参照してください。
- データベース権限

表やルーチンの作成、表へのデータのロードなどのアクティビティーを実行するには、特定のデータベース権限が必要です。詳しくは、 「データベース権限」を参照してください。

- 特権:

  データベース・オブジェクトに対するアクティビティー (たとえば索引の作成やドロップ) を実行するには、特権が必要です。特権は、ユーザーが実行できるタスクを厳密に定義します。たとえば、あるユーザーに対して、表に索引を作成する特権を与える一方、同じ表に対するトリガーを作成する特権を与えないことが可能です。

  – CONTROL 特権

    オブジェクトに対する CONTROL 特権を持っているユーザーは、そのデータベース・オブジェクトにアクセスでき、そのオブジェクトに対する他のユーザーの特権を付与または取り消すことができます。

    注: CONTROL 特権は、表、ビュー、ニックネーム、索引、およびパッケージにのみ適用されます。

    他のユーザーがそのオブジェクトに対する CONTROL 特権を要求した場合、SYSADM または DBADM 権限を持つユーザーが、そのオブジェクトに対する CONTROL 特権を付与する必要があります。 CONTROL 特権は、オブジェクト所有者から取り消されることがありません。

    場合によっては、オブジェクトの作成者がそのオブジェクトに対する CONTROL 特権を自動的に取得します。詳しくは、 「オブジェクト作成、所有権、および特権」を参照してください。

  – ユーザーが特定オブジェクトに対して特定のタスクを実行できるようにするために、個別特権を与えることができます。

    管理権限 (SYSADM または DBADM) を持つユーザー、または CONTROL 特権を持つユーザーは、他のユーザーの特権を付与または取り消すことができます。

    個別特権およびデータベース権限は特定の機能の実行を許可しますが、同じ特権または権限を他のユーザーに与えることはできません。 GRANT ステートメントで WITH GRANT OPTION を使用すれば、表、ビュー、スキーマ、パッケージ、ルーチン、シーケンスに関する特権を他のユーザーに対して GRANT できる権利を、他のユーザーに拡張して与えることができます。ただし、WITH GRANT OPTION によって GRANT する特権を与えられた人は、いったん GRANT された特権を REVOKE することはできません。特権を取り消すためには、SYSADM 権限、DBADM 権限、または CONTROL 特権を持っていなければなりません。

    さらに、PUBLIC に対して特権を付与することもできます。 PUBLIC 特権は、個々のユーザーにすでに特権が与えられているかどうかにかかわらず、すべてのユーザー (許可名) に適用されます。これには、将来のすべてのユーザーも含まれます。

– 暗黙特権。これは、パッケージを実行する特権を持つユーザーに与えられるものです。ユーザーがアプリケーションを実行できる場合でも、パッケージ内で使用されるデータ・オブジェクトに対する明示特権が必要であるとは限りません。

1 つのユーザーまたはグループに対して、個々の特権または権限をいくつか組み合わせて許可することもできます。特権をオブジェクトに関連付ける場合、そのオブジェクトはすでに存在していなければなりません。たとえば、表がそれ以前に作成されているのでなければ、その表についての SELECT 特権をユーザーに与えることはできません。

注: ある許可名が権限と特権を与えられ、しかもその許可名で作成されたユーザーがいない場合には、注意が必要です。後で、その許可名を使用してユーザーが作成され、その許可名に関連するすべての権限と特権を自動的に受け取る可能性があります。

すでに付与された特権を取り消すには、REVOKE ステートメントを使用します。DB2 UDB では、1 つの許可名から特権を取り消すと、すべての許可名によって付与された特権が取り消されます。

ある許可名から特権を取り消しても、その許可名によって特権を付与された他の許可名からその同じ特権が取り消されることはありません。たとえば、ユーザー CLAIRE が SELECT WITH GRANT OPTION をユーザー RICK に与えた後、RICK が SELECT を BOBBY および CHRIS に与えたとします。もし CLAIRE が SELECT 特権を RICK から取り消しても、 BOBBY と CHRIS は引き続き選択特権を保持します。

関連概念:

- 「*管理ガイド: インプリメンテーション*」の『システム管理権限 (SYSADM)』
- 「*管理ガイド: インプリメンテーション*」の『システム制御権限 (SYSCTRL)』
- 「*管理ガイド: インプリメンテーション*」の『システム保守権限 (SYSMAINT)』
- 「*管理ガイド: インプリメンテーション*」の『データベース管理権限 (DBADM)』
- 「*管理ガイド: インプリメンテーション*」の『LOAD 権限』
- 「*管理ガイド: インプリメンテーション*」の『データベース権限』
- 「*管理ガイド: インプリメンテーション*」の『スキーマの特権』
- 「*管理ガイド: インプリメンテーション*」の『表スペース特権』
- 「*管理ガイド: インプリメンテーション*」の『表およびビューの特権』
- 「*管理ガイド: インプリメンテーション*」の『パッケージの特権』
- 「*管理ガイド: インプリメンテーション*」の『索引の特権』
- 「*管理ガイド: インプリメンテーション*」の『シーケンス特権』
- 「*管理ガイド: インプリメンテーション*」の『データベース・オブジェクトへのアクセスの制御』
- 「*管理ガイド: インプリメンテーション*」の『パッケージ経由の間接特権』
- 「*管理ガイド: インプリメンテーション*」の『ルーチン特権』

- 「*管理ガイド: インプリメンテーション*」の『オブジェクト作成、所有権、および特権』
- 「*管理ガイド: インプリメンテーション*」の『システム・モニター権限 (SYSMON)』

スキーマ

スキーマ とは、名前を持つオブジェクトの集合のことです。スキーマは、データベースのオブジェクトを論理的に区分します。表、ビュー、ニックネーム、トリガー、関数、パッケージ、および他のオブジェクトをスキーマに入れることができます。

スキーマはデータベースのオブジェクトでもあります。現行ユーザーを指定するか、またはスキーマ所有者と記録された指定の許可 ID を指定した CREATE SCHEMA ステートメントを使用して、スキーマは明示的に作成されます。また、ユーザーが IMPLICIT_SCHEMA データベース権限を持っている場合には、他のオブジェクトを作成する際に暗黙的に作成することもできます。

スキーマ名 は、 2 つの部分から成るオブジェクト名の高位の部分として使用されます。オブジェクトを作成する際にスキーマを使用して固有に修飾すると、オブジェクトはこのスキーマに割り当てられます。オブジェクトを作成する際にスキーマ名を指定しないと、デフォルトのスキーマ名が使用されます。

たとえば、DBADM 権限を有するユーザーが、ユーザー A に対して C と呼ばれるスキーマを作成するとします。

```
CREATE SCHEMA C AUTHORIZATION A
```

次にユーザー A は、以下のステートメントを出して、スキーマ C 内に X という名前の表を作成することができます (ただし、ユーザー A が CREATETAB データベース権限をもつことを前提とします)。

```
CREATE TABLE C.X (COL1 INT)
```

予約済みのスキーマ名があります。たとえば、組み込み関数は SYSIBM スキーマに属しますし、プリインストールされたユーザー定義関数は SYSFUN スキーマに属します。

データベースが作成される場合に、すべてのユーザーが IMPLICIT_SCHEMA 権限を持ちます。これにより、すべてのユーザーが、まだ存在しない任意のスキーマにオブジェクトを作成することができます。暗黙的に作成されたスキーマに対して、どのようなユーザーも、そのスキーマに他のオブジェクトを作成することができます。別名、特殊タイプ、関数、およびトリガーの作成は、暗黙的に作成されるスキーマにまで拡張されます。暗黙的に作成されるスキーマについてのデフォルトの特権には、旧バージョンとの上位互換性があります。

IMPLICIT_SCHEMA 権限が PUBLIC から取り消される場合、スキーマは、CREATE SCHEMA ステートメントを使用して明示的に作成されるか、または IMPLICIT_SCHEMA 権限が与えられているユーザー (たとえば、 DBADM 権限のあるユーザー) によって暗黙的に作成されます。 PUBLIC から

IMPLICIT_SCHEMA 権限を取り消すことは、スキーマ名の使用に対する制御が増す一方で、既存のアプリケーションがオブジェクトの作成を試みる時に許可エラーが生じる可能性があります。

スキーマには特権もあるので、スキーマ所有者がその特権を使用すれば、どのようなユーザーがスキーマ中のオブジェクトを作成、変更、およびドロップする権限をもつかを制御することができます。当初、スキーマの所有者にはスキーマに関するこれらのすべての特権が与えられ、それらの特権を他のユーザーに付与することもできます。暗黙的に作成されたスキーマはシステムによって所有され、当初、そのようなスキーマにオブジェクトを作成する権限がすべてのユーザーに与えられます。 SYSADM または DBADM 権限を有するユーザーは、任意のスキーマでユーザーが保持する特権を変更することができます。したがって、任意のスキーマ (暗黙的に作成されたスキーマであっても) のオブジェクトを作成、変更、およびドロップするためのアクセスを制御することができます。

関連概念:

- 「*管理ガイド: インプリメンテーション*」の『スキーマの特権』

表

表は、データベース・マネージャーが管理する論理的な構造です。表は列と行で構成されます。表の内部で行の順序を並べ替える必要はありません (行の順序はアプリケーション・プログラムが決定します)。 1 つの列と 1 つの行が交差する箇所は、*値* と呼ばれる特定のデータ項目です。*列* は、同じタイプあるいはそのいずれかのサブタイプの値の集まりです。*行* は、その *n* 番目の値が表の *n* 番目の列の値であるように配置された値の並びです。

基本表 は、CREATE TABLE ステートメントを使用して作成され、永続的なユーザー・データを保持するのに使用されます。*結果表* は、照会の要求に応じるためにデータベース・マネージャーが 1 つ以上の基本表から選択または生成した行の集まりです。

マテリアライズ照会表 は、照会によって定義される表です。この照会は、サマリー表のデータを判別するのにも使用されます。マテリアライズ照会表を使用すると、照会のパフォーマンスが向上します。マテリアライズ照会表を使用して照会の一部を解決することができると判別した場合、データベース・マネージャーは、マテリアライズ照会表を使用するようその照会を書き換えることがあります。この決定は、CURRENT REFRESH AGE および CURRENT QUERY OPTIMIZATION 特殊レジスターなどの、データベース構成の設定値に基づいてなされます。

表では、列ごとにデータ型を個別に定義する (つまり、型をユーザー定義の構造化タイプの属性に基づいたものにする) ことができます。このような表は、*タイプ表* と呼ばれます。ユーザー定義の構造化タイプは、タイプ階層の一部にすることができます。*サブタイプ* は、*スーパータイプ* から属性を継承します。同様に、タイプ表は表階層の一部にすることができます。*副表* は、*スーパー表* から列を継承します。*サブタイプ* という用語は、タイプ階層において 1 つのユーザー定義の構造化タイプおよびその下にあるすべてのユーザー定義の構造化タイプを指して用いられることに注意してください。構造化タイプ T の*厳密な意味でのサブタイプ* とは、タイプ階層で T の下にある構造化タイプのことです。同様に、*副表* という用語

は、表階層において 1 つのタイプ表およびその下にあるすべてのタイプ表を指して用いられます。表 T の*厳密な意味での副表* とは、表階層において T の下にある表のことです。

宣言済み一時表 は、DECLARE GLOBAL TEMPORARY TABLE ステートメントで作成され、 1 つのアプリケーションのために一時データを保持するときに使います。この表は、アプリケーションがデータベースから切断されるときに、暗黙的にドロップされます。

ビュー

ビュー は 1 つ以上の表のデータを見るためのもう 1 つの方法を提供します。結果表の名前付き指定です。その指定は、SQL ステートメントでビューが参照されるたびに実行される SELECT ステートメントです。ビューには基本表と同じく列と行があります。ビューはすべてデータ検索において基本表と同じように使用することができます。挿入、更新、または削除の操作にビューを使用できるかどうかは、その定義によって異なります。

ビューを使用して、機密データへのアクセスを制御することができます。ビューを使うと、複数のユーザーが同じ表の異なった表示を見ることができます。たとえば、幾人かのユーザーが従業員データの表にアクセスするという場合があります。管理者は自分の部門の従業員のデータは見ることができますが、他の部門の従業員のデータは見ることができません。求人課員はすべての従業員の雇用日付を見ることができますが給料は見えません。経理課員は給料を見ることはできますが雇用日付は見えません。こうしたユーザーはそれぞれ基本表から派生したビューで作業します。それぞれのビューは表のように見えて、名前を持っています。

ビューの列が基本表の列から直接に派生している場合、その列は基本表の列に適用されるあらゆる制約を継承します。たとえば、ビューにその基本表の外部キーが入っている場合、そのビューを使用する挿入および更新操作は基本表と同じ参照制約に従います。また、ビューの基本表が親表である場合、そのビューを使用する削除および更新操作は、基本表の削除および更新操作と同じ規則に従います。

ビューでは、列ごとにデータ型を結果表から派生させる (つまり、型をユーザー定義の構造化タイプの属性に基づいたものにする) ことができます。このようなビューを、*タイプ・ビュー* といいます。タイプ表と同様に、タイプ・ビューはビュー階層の一部にすることができます。サブビューは、*スーパービュー* から列を継承します。*サブビュー* という用語は、ビュー階層において 1 つのタイプ・ビューおよびその下にあるすべてのタイプ・ビューを指して用いられます。ビュー V の*厳密な意味でのサブビュー* とは、タイプ・ビュー階層で V の下にあるビューのことです。

ビューは作動不能になることがあります (基本表がドロップされた場合など)。その場合、それ以後 SQL 操作で使用することはできません。

別名

別名 とは、表またはビューの代替名のことです。既存の表またはビューが参照できる 場合には、表またはビューを参照するのに使用することができます。別名はどのコンテキストでも使用できるわけではありません。たとえば、チェック制約の検査条件の中では使用できません。別名で宣言済み一時表を参照することはできません。

表やビューと同様に、別名は作成やドロップが可能であり、コメントを付けることもできます。ただし、表とは異なり、チェーニング と呼ばれるプロセスの中で相互に参照し合うことが可能です。別名は広く参照される名前であり、このためそれらの使用には特別な許可や特権などは必要ありません。しかしながら、別名で参照した表またはビューへのアクセスには、それらのオブジェクトに関連した許可が必要です。

データベース別名やネットワーク別名などその他の種類の別名があります。フェデレーテッド・システム上に置かれたデータ表やビューを参照するニックネーム用の別名を作成することもできます。

索引

索引 とは、基本表の行へのポインターに順序を付けた集合のことです。それぞれの索引は、1 つ以上の表列のデータ値に基づいています。索引は、表のデータとは別個の 1 つのオブジェクトです。索引を作成すると、データベース・マネージャーによって自動的にこのオブジェクトが作成されて保守されます。

索引は、以下の目的でデータベース・マネージャーが使用します。

- パフォーマンスの改善。ほとんどの場合、索引を使った方がデータへのアクセスが速くなります。ビューには索引は作成できませんが、ビューの元になっている表に索引を作成されていると、ビューの操作のパフォーマンスが向上する可能性があります。
- ユニーク性の確保。ユニーク索引をもつ表では、複数の行のキーを同じにすることができません。

キー

キー は、特定の行または行の集まりの識別やアクセスのために使用する列の集合です。キーは、表、索引、または参照制約の記述において指定されます。同じ列が複数のキーを構成することも可能です。

2 つ以上の列から成るキーは、*複合キー* と呼ばれます。複合キーのある表では、複合キー内の列の順序は表内の列の順序の制約を受けません。複合キーの *値* は、複合値を示します。したがって、「外部キーの値は主キーの値に等しくなければならない」という規則の場合、外部キーの値の各構成要素が、主キーの値の構成要素のうちのそれぞれ対応するものと等しくなければならないことを意味します。

ユニーク・キー は、キーのどのような 2 つの値も等しい値であってはならないという制約のあるキーです。ユニーク・キーの列の内容を NULL 値にすることはでき

ません。この制約は、データ値を変更する任意の操作 (たとえば、 INSERT や UPDATE など) の実行の過程でデータベース・マネージャーによって課せられます。制約を課すために使用されるメカニズムは、*ユニーク索引* と呼ばれます。つまり、ユニーク・キーはユニーク索引のキーであるということです。このような索引は UNIQUE 属性があるとも言われます。

主キー は、ユニーク・キーの特殊なケースです。表は、複数の主キーを持つことはできません。

外部キー とは、参照制約の定義で指定されているキーです。

区分化キー とは、パーティション・データベースの表の定義の一部であるキーです。区分化キーは、データの行が保管されるパーティションを判別するのに使用されます。区分化キーが定義される場合、ユニーク・キーおよび主キーには、区分化キーと同じ列が入っていなければなりませんが、それに加えて他の列が入っている場合もあります。表は、複数の区分化キーを持つことはできません。

制約

制約 は、データベース・マネージャーが実施する規則です。

制約には次の 4 つのタイプがあります。

- *ユニーク制約* は、表の中の 1 つまたは複数の列での重複値を禁止する規則です。ユニーク制約では、ユニーク・キーと主キーがサポートされています。たとえば、絶対に 2 つの製造業者に同じ製造業者 ID が与えられないようにするために、製造業者表の製造業者 ID にユニーク制約を定義することができます。
- *参照制約* は、 1 つ以上の表の中の 1 つ以上の列での値に関する論理規則です。たとえば、表集合は企業の製造業者に関する情報を共有します。場合によっては、製造業者の名前が変わることもあります。表の製造業者の ID が、製造業者情報の製造業者 ID と一致していなければならないことを示す、参照制約を定義できます。これで、場合によっては製造業者情報の消失に至るような、挿入、更新、削除操作が防げます。
- *表チェック制約* は、特定の表に追加されるデータに制約を設定します。たとえば、表チェック制約では、個人情報の入った表でデータが追加または更新される場合に、必ず従業員の給与レベルが $20,000 より低くならないようにできます。
- *情報制約*とは、SQL コンパイラーで使用できる規則ですが、データベース・マネージャーでは実施されません。

参照制約および表チェック制約は、オンまたはオフに切り替えることができます。たとえば、大量のデータがデータベースにロードされる場合、制約の実施をオフにする方が一般的に得策です。

ユニーク制約

ユニーク制約 は、キーの値が表内でユニークな場合にのみ、その値が有効になるという規則です。ユニーク制約はオプションであり、 PRIMARY KEY 文節または UNIQUE 文節を使用して、 CREATE TABLE または ALTER TABLE ステートメントで定義できます。ユニーク制約で指定される列は、 NOT NULL と定義されて

いなければなりません。データベース・マネージャーはユニーク索引を使用して、ユニーク制約の列への変更の際にキーのユニーク性を有効化します。

1 つの表で、任意の数のユニーク制約を定義できます。ただし、複数のユニーク制約を主キーと定義することはできません。 1 つの表で、同じ列のセット上に複数のユニーク制約を持つことはできません。

参照制約の外部キーによって参照されるユニーク制約を、*親キー* といいます。

CREATE TABLE ステートメントにユニーク制約が含まれている場合、データベース・マネージャーによってユニーク索引が自動的に作成され、一次索引またはユニークかつ system-requied である索引と指定されます。

ユニーク制約が ALTER TABLE ステートメントで定義されており、索引が同じ列にある場合、その索引はユニークおよびシステム必須と指定されます。そのような索引が存在しない場合、データベース・マネージャーによってユニーク索引が自動的に作成されて、一次索引またはユニークかつ system-required である索引と指定されます。

ユニーク制約の定義とユニーク索引の作成には、違いがあるので注意してください。いずれもユニーク性を有効化しますが、ユニーク索引では NULL 可能列を使用することができるので、一般的に親キーとして使用することはできません。

参照制約

参照保全 とは、すべての外部キーのすべての値が有効であるデータベースの状態です。*外部キー* は、親表の行の 1 つ以上の主キーまたはユニーク・キー値と値が一致していなければならない、表内の列または列のセットです。*参照制約* は、以下の条件のいずれかが当てはまる場合にのみ、外部キーの値が有効になる規則です。

- それらが親キーの値となっている。
- 外部キーの何らかの部分が NULL である。

親キーの入った表を参照制約の*親表* といい、外部キーの入った表を表の*従属* といいます。

参照制約はオプションであり、 CREATE TABLE ステートメントまたは ALTER TABLE ステートメントで定義できます。参照制約は、
INSERT、UPDATE、DELETE、 ALTER TABLE、ADD CONSTRAINT、および SET INTEGRITY ステートメントの実行中に、データベース・マネージャーによって実施されます。

RESTRICT の削除または更新規則が指定されている参照制約は、他のすべての参照制約の前に実施されます。 NO ACTION の削除または更新規則が指定されている参照制約は、ほとんどの場合、 RESTRICT のように動作します。

参照制約、チェック制約、およびトリガーは同時に使用できることに注意してください。

参照保全規則には、以下の概念および用語が関係します。

親キー 親キーとは、参照制約の主キーまたはユニーク・キーのことです。

親行 1 つ以上の従属行を持つ行。

親表 参照制約の親キーの入った表。 1 つの表を、任意の数の参照制約における親にすることができます。参照制約の中の親である表が、参照制約の中の従属になることもできます。

従属表 定義において 1 つ以上の参照制約の入った表。表を、任意の数の参照制約において従属表にすることができます。参照制約の中の従属である表が、参照制約の中の親になることもできます。

下層表 ある表が表 T の下層であるとは、それが T の従属であるか、または T の従属の下層であるということです。

従属行 1 つ以上の親行を持つ行。

下層行 ある行が行 r の下層であるとは、それが r の従属であるか、または r の従属の下層であるということです。

参照循環
セット内の各表がそれ自身の下層であるような、参照制約のセット。

自己参照表
同じ参照制約内の親および従属である表。この制約を、*自己参照制約* といいます。

自己参照行
それ自身の親である行。

挿入規則

参照制約の挿入規則とは、外部キーの非 NULL の挿入値が、親表の親キーの何らかの値に一致しなければならないというものです。複合外部キーの値は、値のいずれかの部分が NULL であれば、NULL になります。これは、外部キーが指定された場合は暗黙の規則になります。

更新規則

参照制約の更新規則は、参照制約の定義時に指定されます。選択項目には NO ACTION と RESTRICT があります。更新規則は、親の行または従属表の行の更新時に適用されます。

親行の場合、親キーの列内の値の更新時に以下の規則が適用されます。

- 従属表の行がキーの元の値と一致し、しかも更新規則が RESTRICT である場合には、その更新は拒否されます。
- 更新ステートメントの完了時に (トリガー後を除く)、従属表の行に対応する親キーがなく、しかも更新規則が NO ACTION である場合には、その更新は拒否されます。

従属行の場合、外部キーを指定した際の暗黙的な更新規則は NO ACTION です。NO ACTION は、更新ステートメントの完了時に、外部キーのヌル以外の更新値が親表の親キーの何らかの値に一致していなければならないことを意味します。

複合外部キーの場合、値がヌルであるのは、その構成要素のどれかがヌルである場合です。

削除規則

参照制約の削除規則は、参照制約の定義時に指定されます。選択項目としては、NO ACTION、RESTRICT、CASCADE、または SET NULL があります。 SET NULL を指定できるのは、外部キーのいずれかの列で NULL 値が可能な場合だけです。

参照制約の削除規則は、親表の行が削除されるときに適用されます。より正確には、親表の行が削除または伝搬された削除操作 (以下で定義する) の対象であり、その行に参照制約の従属表がある場合に、その規則が適用されます。例として、P が親表、D が従属表、そして p が削除または伝搬削除操作の対象である親行を指す場合を考えます。削除規則は以下のように機能します。

- RESTRICT または NO ACTION では、エラーが発生し、行が削除されません。
- CASCADE では、削除操作が表 D の従属に伝搬されます。
- SET NULL では、表 D 内の p の各従属の外部キーの各 NULL 可能列が、NULL に設定されます。

表が親となっている参照制約にはそれぞれ独自の削除規則があり、適用可能な削除規則のすべてを使用して削除操作の結果が判別されます。そのため、行に RESTRICT または NO ACTION の削除規則が指定されている参照制約の従属がある場合、または RESTRICT または NO ACTION の削除規則が指定されている参照制約の従属である、何らかの下層への削除カスケードがある場合、その行は削除できません。

親表 P からの行の削除には他の表も関係しており、以下の表の行に影響する場合があります。

- 表 D が P の従属であり、削除規則が RESTRICT または NO ACTION である場合、 D がその操作に関係していても、操作によって影響は受けません。
- D が P の従属であり、削除規則が SET NULL である場合、その操作に D が関係し、その操作中に D の行が更新される場合があります。
- D が P の従属であり、削除規則が CASCADE である場合、その操作に D が関係し、その操作中に D の行が削除される場合があります。

  D の行が削除される場合、 P での削除操作を D への伝搬といいます。 D が親表でもある場合、今度は、ここリストされているアクションが D の従属にも適用されます。

P での削除操作に関係する可能性のあるすべての表を、 P への*関連した削除* といいます。そのため、表が P の従属である場合、または P から削除操作のカスケードが行われる表の従属である場合、表は表 P への関連した削除になります。

関連した削除リレーションには以下の制約事項があります。

- 複数の表の参照サイクル中で表が自身に対して関連して削除されるには、そのサイクルに RESTRICT または SET NULL の削除規則が入っていてはなりません。
- 表は、CASCADE 関係 (自己参照、または別の表を参照) 内の従属表であって、しかも RESTRICT または SET NULL の削除規則に対する自己参照関係をもっていてはなりません。
- 表が、オーバーラップしている外部キーをもつ関係である複数の関係を通して別の表に関連して削除される場合は、それらのどの関係でも削除規則が同じでなければならず、しかもいずれの削除規則も SET NULL であってはなりません。

- 表が複数の関係をとおして別の表に関連して削除される場合に、それらの関係のうちのいずれかに SET NULL の削除規則が指定されているときは、その関係の外部キー定義には、区分化キーや MDC キー列が入っていてはなりません。
- 2 つの表が CASCADE 関係をとおして同じ表に関連して削除されている場合に、関連した削除パスが削除規則 RESTRICT または SET NULL で終わっていれば、この 2 つの表を相互に関連して削除してはなりません。

表チェック制約

表チェック制約 は、表の各行にある 1 つまたは複数の列で許可される値を指定するための規則です。制約はオプションであり、 CREATE TABLE または ALTER TABLE ステートメントを使用して定義できます。表チェック制約の指定は、制限付きフォームの検索条件を使って行われます。制限の 1 つは、表 T での表チェック制約の列名は、表 T の列を示していなければならないというものです。

1 つの表に、任意の数の表チェック制約を定義することができます。表チェック制約は、挿入または更新される各行へ検索条件を適用することで有効化されます。いずれかの行で検索条件の結果が false の場合、エラー検索が発生します。

既存のデータがある表で、 ALTER TABLE ステートメントに 1 つ以上の表チェック制約が定義されている場合、 ALTER TABLE ステートメントが完了する前に、既存のデータが新しい条件と比べてチェックされます。 SET INTEGRITY ステートメントを使用して、表をチェック・ペンディング 状態にすることができます。これにより、データの検査を行わずに、 ALTER TABLE ステートメントを続行できます。

情報制約

*情報制約*は、データへのアクセス・パスを改善するために SQL コンパイラーで使用できる規則です。情報制約はデータベース・マネージャーによって実施される制約ではなく、またデータの詳細検査に使用されることもありません。これは、照会のパフォーマンスの改善のために使用されます。

データベース・マネージャーが制約を実施するかどうかや、照会の最適化にその制約を使用するかどうかの決定のための制約属性を指定する参照制約または表チェック制約を定義するには、 CREATE TABLE または ALTER TABLE ステートメントを使用します。

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『SET INTEGRITY ステートメント』
- 747 ページの『付録 H. トリガーと制約の相互作用』

分離レベル

アプリケーション・プロセスに関連する*分離レベル* は、並行して実行している他のアプリケーション・プロセスからそのアプリケーション・プロセスを分離する度合いを定義します。したがって、アプリケーション・プロセスの分離レベルは、以下を指定します。

- アプリケーションによって読み取られ更新される行を、並行して実行される他のアプリケーション・プロセスから使用できる度合い。
- 並行して実行される他のアプリケーション・プロセスの更新活動によってアプリケーションが影響を受ける度合い。

静的 SQL ステートメントの分離レベルは、パッケージの属性として指定され、そのパッケージを使用するアプリケーション・プロセスに適用されます。分離レベルは、プログラム準備処理で指定されます。動的 SQL ステートメントの場合、デフォルトの分離レベルは、ステートメントを作成するパッケージに指定された分離レベルです。 SET CURRENT ISOLATION ステートメントを使用すると、セッション内で発行される動的 SQL に対して代替分離レベルを指定できます。これによって、並行アプリケーション・プロセスによるデータ・アクセスは、ロックの種類に応じて制限または禁止されます。 (宣言済み一時表とその行は、宣言したアプリケーションしかアクセスできないので、ロックされることはありません。)

データベース・マネージャーでは、大きく分けて次の 3 つのロック・カテゴリーがサポートされています。

共用 並行アプリケーション・プロセスの操作を、読み取り専用のデータ操作のみに制限します。

更新 並行アプリケーション・プロセスは行の更新を宣言したのではない限り、データへの読み取り専用操作に限定されます。データベース・マネージャーは、行を現在見ているプロセスがそれを更新する可能性があると想定します。

排他 同時アプリケーション・プロセスがどのような形であれ、そのデータにアクセスできないようにします。読み取りはできてもデータの変更はできない*非コミット読み取り* の分離レベルのアプリケーション・プロセスにはあてはまりません。

ロッキングは基本表の行について行われます。しかし、データベース・マネージャーが複数の行ロックを単一の表ロックに置き換えることがあります。これを*ロック・エスカレーション* といいます。アプリケーション・プロセスでは、最低限必要なロック・レベルが確保されます。

DB2® Universal Database のデータベース・マネージャーは 4 つの分離レベルをサポートしています。分離レベルとは関係なく、データベース・マネージャーは、挿入、更新、または削除の対象となる行のすべてに排他ロックをかけます。このため、どの分離レベルでも、このアプリケーション・プロセスが 1 作業単位中に変更する行は、その作業単位が完了するまで他のアプリケーション・プロセスから変更されることは決してありません。分離レベルには、以下のものがあります。

- 反復可能読み取り (RR)

  このレベルでは、以下の点が確実に守られます。

  – 作業単位の中で読み取られた行は、その作業単位が完了するまで、他のアプリケーション・プロセスから変更されません。行は、対応する OPEN ステートメントと同じ作業単位で読み取られます。 CLOSE ステートメントにオプションの WITH RELEASE 文節を使用することの意義は、カーソルが再オープンされた場合に、反復不能読み取りおよび幻像読み取りの防止は、以前アクセスした行には適用されないことにあります。

– 他のアプリケーション・プロセスによって変更される行は、そのアプリケーション・プロセスがコミットするまで読み取ることができません。

反復可能読み取りレベルでは、幻像読み取り行の出現は許されません (「読み取り固定」の説明を参照)。

RR レベルで実行しているアプリケーション・プロセスは、排他ロック以外に、参照するすべての行に対して共有ロックを獲得します。さらに、アプリケーション・プロセスが並行アプリケーション・プロセスの影響から完全に分離されるようにロッキングが実行されます。

- 読み取り固定 (RS)

  反復可能読み取りレベルと同様に、読み取り固定レベルでは、以下の点が確実に守られます。

  – 作業単位の中で読み取られた行は、その作業単位が完了するまで、他のアプリケーション・プロセスから変更されません。行は、対応する OPEN ステートメントと同じ作業単位で読み取られます。 CLOSE ステートメントにオプションの WITH RELEASE 文節を使用することの意義は、カーソルが再オープンされた場合に、反復不能読み取りおよび幻像読み取りの防止は、以前アクセスした行には適用されないことにあります。

  – 他のアプリケーション・プロセスによって変更される行は、そのアプリケーション・プロセスがコミットするまで読み取ることができません。

  反復可能読み取りとは異なり読み取り固定では、アプリケーション・プロセスは、他の並行アプリケーション・プロセスの影響から完全には分離されません。RS レベルでは、同じ照会を複数回発行するアプリケーション・プロセスで行が追加されていくという場合があります。これらの行は、データベースに新しい情報を付加する他のアプリケーション・プロセスによって作成されたものです。このような付加行は*幻像読み取り行* と呼ばれます。

  幻像読み取り行は、たとえば次のような状況で発生します。

  1. アプリケーションのプロセス P1 が、検索条件を満たす一連の *n* 個の行を読み取る。
  2. 次にアプリケーション・プロセス P2 が、検索条件を満たす 1 つまたは複数の行を挿入し、それらの挿入行をコミットする。
  3. P1 が同じ検索条件で一連の行を再度読み取り、元の行と P2 によって挿入された行を両方とも獲得する。

  RS 分離レベルで実行しているアプリケーション・プロセスは、排他ロックに加えて、条件に合うすべての行に対して少なくとも共用ロックを獲得します。

- カーソル固定 (CS)

  反復可能読み取りレベルと同様に、カーソル固定レベルでは、他のアプリケーション・プロセスによって変更された行は、そのアプリケーション・プロセスによってコミットされるまではまったく読み取り不能になります。

  反復可能読み取りとは異なりカーソル固定では、すべての更新可能なカーソルの現在行が、他のアプリケーション・プロセスによって変更されないことだけが確

実になります。このため、作業単位の中で読み取られた行が、他のアプリケーション・プロセスによって変更される可能性があります。

CS 分離レベルで実行しているアプリケーション・プロセスは、排他ロックに加えて、すべてのカーソルの現在行に対して少なくとも共用ロックを獲得します。

- 非コミット読み取り (UR)

  SELECT INTO、読み取り専用カーソルによる FETCH、INSERT で使用される全選択、 UPDATE での行の全選択、またはスカラーの全選択なら、非コミット読み取りレベルでは以下のことが可能です。

  – 作業単位の中で読み取られた行は、すべて他のアプリケーション・プロセスから変更できます。
  – 他のアプリケーション・プロセスで変更された行は、そのアプリケーション・プロセスで変更をコミットしていなくても、すべて読み取ることができます。

  これ以外の操作では、CS レベルの規則が適用されます。

分離レベルの比較

次の表は、分離レベルについて要約しています。

	UR	CS	RS	RR
アプリケーションは、他のアプリケーション・プロセスによって行われた非コミットの変更を見ることができるか ?	可	不可	不可	不可
アプリケーションは、他のアプリケーション・プロセスによって行われた非コミットの変更を更新できるか ?	不可	不可	不可	不可
ステートメントをもう一度実行した場合、他のアプリケーション・プロセスによる影響があるか ? *下記の現象 P3 (幻像) を参照。*	可	可	可	不可
「更新された」行を他のアプリケーション・プロセスで更新することができるか ? 下記の注 1 を参照。	不可	不可	不可	不可
「更新された」行を、UR 以外の分離レベルの他のアプリケーション・プロセスで読み取れるか ?	不可	不可	不可	不可
「更新された」行を UR の分離レベルの他のアプリケーション・プロセスで読み取れるか ?	可	可	可	可
「アクセスされた」行を、他のアプリケーション・プロセスによって更新することが可能か ? *下記の現象 P2 (反復不能読み取り) を参照。*	可	可	不可	不可
「アクセスされた」行を他のアプリケーション・プロセスが読み取ることができるか ?	可	可	可	可
「現在」行を、他のアプリケーション・プロセスによって更新または削除することが可能か ? *下記の現象 P1 (ダーティー読み取り) を参照。*	下記の注 2 を参照。	下記の注 2 を参照。	不可	不可

	UR	CS	RS	RR

注:

1. アプリケーションが表に対する読み取りと書き込みの両方を行う場合、分離レベルはアプリケーションのための保護を提供しません。たとえば、アプリケーションは表でカーソルをオープンし、それからその同じ表に挿入、更新、または削除の操作を実行します。オープン・カーソルでもっと行を取り出してゆくにつれて、アプリケーションが矛盾するデータを見つける場合があります。
2. カーソルが更新可能でない場合、CS では、現在行を他のアプリケーション・プロセスによって更新または削除できる場合もあります。たとえば、バッファリングによってクライアントの現在の行が、サーバーの実際の現在行の値と違うということが引き起こされる場合があります。

現象の例:

P1 *ダーティー読み取り*。作業単位 UW1 が行を変更するとします。 UW1 が COMMIT を実行する前に、作業単位 UW2 がその行を読み取るとします。次に UW1 が ROLLBACK を実行したとすると、 UW2 は実行しない行を読み取ったことになります。

P2 *反復不能読み取り*。作業単位 UW1 が行を読み取るとします。作業単位 UW2 がその行を変更してから、COMMIT を実行するとします。 UW1 がもう一度その行を読み取ると、値が修正されていることがあります。

P3 *幻像*。作業単位 UW1 が、ある検索条件を満たしている *n* 個の行を読み取るとします。次に作業単位 UW2 が、その検索条件を満たしている 1 つまたは複数の行を挿入して、 COMMIT を実行します。 UW1 が同じ検索条件でもう一度最初の読み取りを実行すると、元の行のほかに挿入された行が追加されていることになります。

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『DECLARE CURSOR ステートメント』

照会と表式

照会 は、 (一時的な) 結果表を指定するための特定の SQL ステートメントからなるコンポーネントです。

表式 は、単純な照会から一時的な結果表を作成します。文節を使うと、その結果表がさらに詳細なものになります。たとえば、表式を照会として使用して、複数の部門からすべての管理者を選択し、さらに管理者が 15 年以上の実務経験があり、ニューヨーク支社に配属されていなければならないことを指定することができます。

共通表式 は、複雑な照会内の一時ビューのようなものです。それは照会内のほかの場所から参照することができ、ビューの代わりに使用できます。複雑な照会の中で特定の共通表式を使用する場合、それぞれが同じ一時ビューを共用することになります。

1 つの照会の中で 1 つの共通表式を再帰的に使用することにより、航空座席予約システム、部品表 (BOM) 生成プログラム、ネットワーク計画などのアプリケーションのサポートのために利用できます。

関連資料:

- 516 ページの『Select-statement』
- 473 ページの『SQL 照会』

アプリケーションのプロセス、並行性、およびリカバリー

すべての SQL プログラムは、アプリケーション・プロセス またはエージェントの一部として実行されます。アプリケーション・プロセスには、1 つ以上のプログラムの実行が関係しており、データベース・マネージャーがリソースを割り当てたりロックしたりする場合の単位となります。異なるいくつかのプログラムの実行、または同じプログラムの複数の異なる実行には、異なる複数のアプリケーション・プロセスが関係しています。

同時に複数のアプリケーション・プロセスが同じデータへのアクセスを要求することがあります。このような状況でデータ保全性を維持するためのメカニズムとして *ロッキング* があります。これは、たとえば 2 つのアプリケーション・プロセスが同時にデータの同じ行を更新するのを防ぐ、などの処理を行います。

データベース・マネージャーは、あるアプリケーション・プログラムが行った変更でまだコミットされていないものが、誤って他のプロセスに認識されることのないよう、ロックを獲得します。プロセスが終了すると、データベース・マネージャーは、アプリケーション・プロセスのためにデータベース・マネージャーが獲得し保持していたロックをすべて解放します。もっと早い時期にロックを解放するには、アプリケーション・プロセス自体で明示的に要求する必要があります。この操作は *コミット* と呼ばれ、これにより作業単位中に獲得していたロックが解放され、作業単位中に加えられた変更がデータベースにコミットされます。

データベース・マネージャーには、アプリケーション・プロセスが行った変更で、まだコミットされていないものを取り消す手段が用意されています。これは、アプリケーション・プロセス側に障害が発生したとき、またはデッドロックやロック・タイムアウト状態などで必要になります。アプリケーション・プロセスで、自分の行ったデータベースへの変更を取り消すように明示的に要求することができます。これは*ロールバック* 操作を使って行います。

作業単位 とは、アプリケーション・プロセス内の、リカバリー可能な一連の操作のことです。作業単位は、アプリケーション・プロセスが開始されたときと、アプリケーションの終了以外の理由で直前の作業単位が終了したときに、開始します。作業単位は、コミット操作、ロールバック操作、またはアプリケーション・プロセスの終了によって終了します。コミットまたはロールバック操作は、それによって終了する作業単位の中で行われたデータベースへの変更内容にしか影響しません。

このような変更がコミットされないまま残っている間は、他のアプリケーション・プロセスはそれらの変更を認識することはできませんし、変更をバックアウトすることも可能です。ただし、分離レベルが非コミット読み取り (UR) である場合にはこの限りではありません。データベースの変更内容がコミットされると、他のアプリケーション・プロセスからその変更内容にアクセスできるようになり、ロールバックによってバックアウトすることはできなくなります。

DB2® コール・レベル・インターフェース (CLI) および組み込み SQL を使用すると、*並行トランザクション* と呼ばれる接続モードを使用できます。これは、それぞ

れが独立したトランザクションである複数の接続をサポートします。1 つのアプリケーションが同じデータベースに対して複数同時接続を行うことができます。

データベース・マネージャーがアプリケーション・プロセスのために獲得したロックは、作業単位が終了するまで保持されます。ただし、分離レベルがカーソル固定 (CS、カーソルが行から行に移動されるとロックは解放される) か非コミット読み取り (UR、ロックは取得されない) の場合はこの限りではありません。

アプリケーション・プロセスが自分自身のロックのために操作できなくなるということは決してありません。しかしながら、アプリケーションが並行してトランザクションを使用する場合、一方のトランザクションによるロックのために、他方のトランザクションの運用が影響を受ける可能性があります。

作業単位の開始と終了によって、アプリケーション・プロセス内の整合点が定義されます。たとえば、銀行業務のトランザクションで、ある口座から別の口座へ資金を振り込むことがあります。このようなトランザクションでは、その資金を第 1 の口座から減算してから、第 2 の口座に加算する、ということが必要になります。減算のステップの直後の段階では、データに矛盾が生じています。資金が第 2 の口座に加算して初めて、整合性が取り戻されるわけです。両方のステップが完了したときに、コミット操作を実行して作業単位を終了させれば、他のアプリケーション・プロセスが変更内容を利用できるようになります。 1 つの作業単位が終わる前に障害が発生すると、データベース・マネージャーはコミットされていない変更内容をロールバックし、その作業単位の開始時点でのデータ整合性をリストアします。

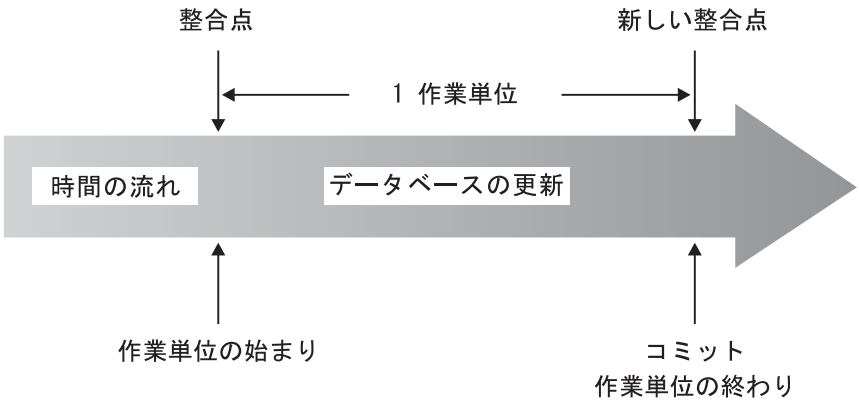


図 2. COMMIT ステートメントの作業単位

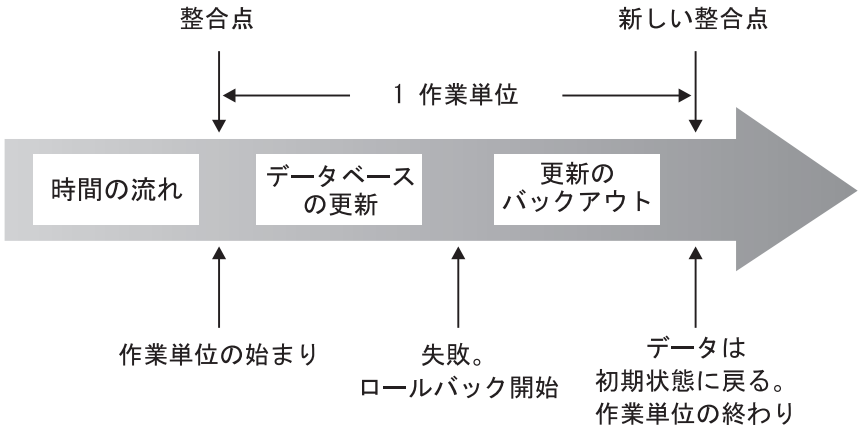


図 3. ROLLBACK ステートメントの作業単位

関連概念:

• 14 ページの『分離レベル』

DB2 コール・レベル・インターフェース (CLI) と Open Database Connectivity (ODBC)

DB2 コール・レベル・インターフェースは、アプリケーション・プログラムに動的 SQL ステートメントを処理する機能を提供するアプリケーション・プログラミング・インターフェースです。 CLI プログラムは Open database connectivity Software Developer's Kit (Microsoft® または他のベンダーから入手可能) を使用してコンパイルすることもできます。これにより、ODBC データ・ソースへアクセスできるようになります。組み込み SQL と違い、このインターフェースではプリコンパイルは必要ありません。種々のデータベースに対して実行できますが、それぞれのためにコンパイルする必要はありません。アプリケーションはランタイムにプロシージャー呼び出しを使用して、データベースへの接続、SQL ステートメントの発行、およびデータや状況情報の入手を行うことができます。

DB2 CLI インターフェースには、組み込み SQL では使用できない多くの機能があります。たとえば、次のような場合です。

- CLI では、データベース・システム・カタログ情報を照会する DB2 ファミリーを通じて一貫した方法をサポートする関数呼び出しが用意されています。これにより、特定のデータベース・サーバーに合わせてカタログ照会を作成する必要が少なくなります。
- CLI には、カーソルを使った、次のようなスクロール機能があります。
  - 1 行以上のフォワード・スクロール
  - 1 行以上のリバース・スクロール
  - 最初の行からの 1 行以上のフォワード・スクロール
  - 最後の行からの 1 行以上のリバース・スクロール
  - カーソルの直前保管位置からのスクロール
- CLI を使用して作成されたアプリケーション・プログラムから呼び出されるストアード・プロシージャーからは、結果セットをプログラムに戻すことができます。

JDBC (Java Database Connectivity) と組み込み SQL for Java (SQLJ) プログラム

DB2® Universal Database は Java database connectivity (JDBC) と組み込み SQL for Java (SQLJ) という 2 つの標準の Java プログラミング API を実装します。どちらを使用しても、 DB2 にアクセスする Java アプリケーションおよびアプレットを作成することができます。

- JDBC 呼び出しは Java 固有のメソッドによって DB2 CLI 呼び出しに変換されます。Universal JDBC Driver の場合は例外でCLI には変換されません。 JDBC は、DB2 CLI を介した DB2 クライアントから DB2 サーバーへの流れを要求します。 JDBC で静的 SQL を使用することはできません。
- SQLJ アプリケーションは、データベースへの接続や SQL エラーの処理といったタスクの基盤として JDBC を使用しますが、 SQLJ ソース・ファイルに組み込み静的 SQL ステートメントを組み入れることもできます。 SQLJ 変換プログラムを使って SQLJ ソース・ファイルを変換してからでないと、生成される Java ソース・コードをコンパイルすることはできません。

パッケージ

パッケージ とは、プログラム準備中に作成される、単一のソース・ファイル内にあるすべてのセクションが入ったオブジェクトのことです。セクション とは、コンパイル済み形式の SQL ステートメントのことです。セクションは必ず 1 つのステートメントに対応しますが、すべてのステートメントにセクションがあるわけではありません。静的 SQL 用に作成されたセクションは、バインド形式または操作可能形式の SQL ステートメントに相当します。動的 SQL 用に作成されたセクションは、ランタイムに使用されるプレースホルダー制御構造に相当します。

カタログ・ビュー

データベース・マネージャーは、その制御下のデータに関する情報の組み込まれた一連の基本表とビューを管理しています。このような基本表とビューを、まとめて カタログ と呼びます。カタログには、表、ビュー、索引、パッケージ、関数などの、データベース・オブジェクトの論理および物理構造についての情報が収められています。また統計情報も収められています。データベース・マネージャーはカタログ内の記述が常に正確であるようにします。

カタログ・ビューは、データベースの他のビューとよく似たものです。カタログ・ビューのデータを参照するには、 SQL ステートメントを使用することができます。カタログの特定の値を変更する場合には、更新可能なカタログ・ビューの集合を使用できます。

関連資料:

- 549 ページの『システム・カタログ・ビュー』

文字変換

ストリング は、文字を表す一連のバイトです。ストリング内のすべての文字は共通のコード化表現を持っています。場合によっては、このような文字を別のコード化表現に変換しなければならないことがあります。これは*文字変換* という処理です。文字変換が必要な場合は自動的に実行されますが、正常に終了すればその実行はアプリケーションからは認識されません。

文字変換は、SQL ステートメントがリモートで実行される場合に発生する可能性があります。たとえば、送信システムと受信システムでコード化表現が異なるかもしれない以下のシナリオを考えてみてください。

- ホスト変数の値が、アプリケーション・リクエスターからアプリケーション・サーバーに送信された。
- 結果列の値が、アプリケーション・サーバーからアプリケーション・リクエスターに送信された。

以下が文字変換を説明する際に使用する用語のリストです。

文字セット
定義済みの文字の集まり。たとえば、いくつかのコード・ページには次の文字セットが出現します。

- A から Z の 26 個の文字 (アクセント記号なし)
- a から z の 26 個の文字 (アクセント記号なし)
- 0 から 9 の数字
- . , : ; ? ( ) ' " / - _ & + % * = < >

コード・ページ
コード・ポイントに対する一連の文字の割り当て。たとえば、コード・ページ 850 の ASCII エンコード・スキーマでは、 "A" にはコード・ポイント X'41' が割り当てられ、 "B" にはコード・ポイント X'42' が割り当てられています。 1 つのコード・ページの中では、それぞれのコード・ポイントはただ 1 つの特定の意味をもちます。コード・ページはデータベースの 1 つの属性です。アプリケーション・プログラムがデータベースに接続している場合、データベース・マネージャーがそのアプリケーションのコード・ページを判別します。

コード・ポイント
文字を表すユニークなビット・パターン。

エンコード・スキーム
文字データを表現するために使用する規則の集まり。たとえば、次のとおりです。

- 1 バイト ASCII
- 1 バイト EBCDIC
- 2 バイト ASCII
- 1 バイト / 2 バイト混合 ASCII

以下の図は、典型的な文字セットが、 2 つの異なるコード・ページの異なるコード・ポイントにどのようにマップされるかを示しています。エンコード・スキーマ

が同じでも多くの異なるコード・ページがあり、同じコード・ポイントであってもコード・ページが異なれば異なる文字を表す場合があります。さらに、文字ストリングの中の 1 バイトは、 1 バイト文字セット (SBCS) の文字を表すとは限りません。文字ストリングは、混合およびビット・データにも使用されます。*混合データ*は、1 バイト文字、2 バイト文字、またはマルチバイト文字の混合です。ビット・データ (FOR BIT DATA か BLOB、またはバイナリー・ストリングと定義されている列) は、どの文字セットにも関連していません。

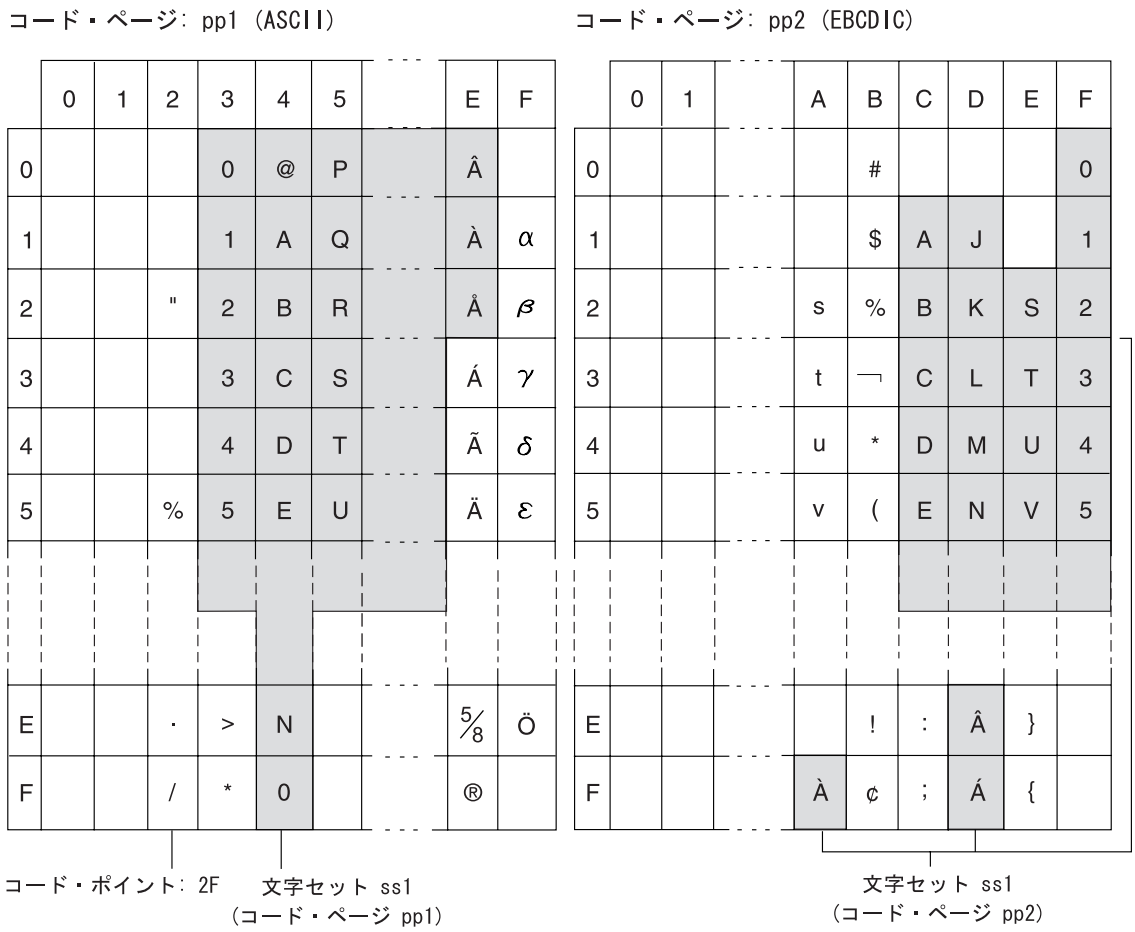


図 4. 別種のコード・ページにおける文字セットのマッピング

データベース・マネージャーは、アプリケーションがデータベースにバインドされるときに、すべての文字ストリングのコード・ページ属性を判別します。可能なコード・ページ属性には以下のものがあります。

データベース・コード・ページ
データベース・コード・ページは、データベース構成ファイルに保管されています。値はデータベースの作成時に指定され、その後の変更は不可能です。

アプリケーション・コード・ページ
このコード・ページの下でアプリケーションが実行されます。これはアプリケーションがバインドされたときのコード・ページと同じであるとは限りません。

セクション・コード・ページ
このコード・ページの下で SQL ステートメントが実行されます。通常、セクション・コード・ページはデータベース・コード・ページです。ただし、次の場合には Unicode コード・ページ (UTF-8) が使用されます。

- 非 Unicode データベース内で Unicode エンコード・スキームを使って作成された表をステートメントが参照する場合。
- 非 Unicode データベース内で PARAMETER CCSID UNICODE を使って定義された表関数をステートメントが参照する場合。

コード・ページ 0
これは、FOR BIT DATA の値または BLOB の値の入った式から派生したストリングを表すものです。

文字ストリング・コード・ページ属性は次のとおりです。

- 列は、データベース・コード・ページ、Unicode コード・ページ (UTF-8)、またはコード・ページ 0 (FOR BIT DATA または BLOB と定義されている場合) のいずれでもかまいません。
- 定数と特殊レジスター (たとえば USER、CURRENT SERVER、SERVER) は、セクション・データベース・コード・ページです。 SQL ステートメントがデータベースにバインドされるときに、必要があれば定数はアプリケーション・コード・ページからデータベース・コード・ページに変換されてから、セクション・コード・ページに変換されます。
- 入力ホスト変数は、アプリケーション・コード・ページです。バージョン 8 以降は、入力ホスト変数内のストリング・データは必要に応じて使用前にアプリケーション・コード・ページからセクション・コード・ページに変換されます。ホスト変数がビット・データとして解釈される状況で使用されると例外が発生します。たとえば、ホスト変数が FOR BIT DATA と定義された列に割り当てられる場合です。

スカラー操作、集合操作、または連結のように、ストリング・オブジェクトを結合する操作のコード・ページ属性は、一連の規則を使用して判別されます。ランタイムには、コード・ページ属性を使用して、ストリングのコード・ページ変換の要件が判別されます。

関連資料:

- 110 ページの『割り当てと比較』
- 131 ページの『ストリング変換の規則』

イベント・モニター

イベント・モニターを使用して、指定されたイベントの発生時に、データベースおよび接続されたアプリケーションに関する情報を収集します。イベントは、接続、デッドロック、ステートメント、トランザクションなどの、データベース・アクティビティーの遷移を表します。モニターしたいイベント (1 つ以上) のタイプごとにイベント・モニターを定義することができます。例えば、デッドロック・イベント・モニターは、デッドロックが発生するのを待機します。発生すると、関係するアプリケーションおよび競合するロックに関する情報を収集します。

注: デフォルトでは、どのデータベースにも DB2DETAILDEADLOCK という名前のイベント・モニターが定義されています。このモニターは DEADLOCKS WITH DETAILS を追跡します。 DB2DETAILDEADLOCK イベント・モニターは、データベースの開始時に自動的に開始されます。

スナップショット・モニターは一般に、予防的な保守および問題分析のために使用されますが、イベント・モニターは、現時点の問題について管理者に警告し、また今にも起こりそうな問題を追跡するするために使用されます。

イベント・モニターを作成するには、CREATE EVENT MONITOR SQL ステートメントを使用します。イベント・モニターは、それらがアクティブなときにだけイベント・データを収集します。イベント・モニターを活動化または非活動化するには、 SET EVENT MONITOR STATE SQL ステートメントを使用します。イベント・モニターの状況 (アクティブか非アクティブか) は、 SQL 関数 EVENT_MON_STATE によって判別することができます。

CREATE EVENT MONITOR SQL ステートメントを実行すると、それが作成するイベント・モニターの定義が、以下のデータベース・システム・カタログ表に保管されます。

- SYSCAT.EVENTMONITORS: データベースについて定義されたイベント・モニター
- SYSCAT.EVENTS: データベースについてモニターされるイベント
- SYSCAT.EVENTTABLES: 表イベント・モニターのためのターゲット表

それぞれのイベント・モニターには、モニター・エレメント内のインスタンスのデータの、独自の専用論理ビューがあります。特定のイベント・モニターが非活動化された後、再活動化されると、これらのカウンターのビューがリセットされます。リセットは、新たに活動化されたイベント・モニターだけで行われます。他のすべてのイベント・モニターは、引き続きカウンター値の独自のビューを使用し続けます (追加があればそのカウンター値に追加します)。

イベント・モニターの出力は、SQL 表、ファイル、または名前付きパイプに送ることができます。

関連概念:

- 「システム・モニター *ガイドおよびリファレンス*」の『データベース・システム・モニター』

関連タスク:

- 「システム・モニター *ガイドおよびリファレンス*」の『データベース・システム・イベントからの情報の収集』
- 「システム・モニター *ガイドおよびリファレンス*」の『イベント・モニターの作成』

関連資料:

- 「システム・モニター *ガイドおよびリファレンス*」の『イベント・モニターの出力例』
- 「システム・モニター *ガイドおよびリファレンス*」の『イベント・タイプ』

トリガー

トリガー は、指定した表に対する挿入、更新、または削除操作への応答として実行されるアクションのセットを定義します。このような SQL 操作が実行されるとき、トリガーが*起動*されるといいます。

トリガーはオプションであり、 CREATE TRIGGER ステートメントを使用して定義されます。

データ保全性規則を実施するために、参照制約およびチェック制約とともにトリガーを使用できます。また、トリガーを使用して、他の表への更新を行ったり、挿入または更新される行の値を自動的に生成またはトランスフォームできます。あるいは、関数を呼び出してタスク (アラートを発するなど) を実行することもできます。

トリガーは、*移り変わる* ビジネスルールを定義および実施するための便利な機構です。この規則は、さまざまな状態のデータ (たとえば、増加率が必ず 10 % 以下である給与など) を扱う規則です。

トリガーを使用すると、ビジネス規則を実施する論理をデータベース内に置くことができます。つまり、アプリケーションがそれらの規則の実施を担当しないということです。すべての表に対してロジックを一カ所に集中すれば、ロジックの変更時にアプリケーション・プログラムへの変更が必要ないため、簡単に保守を行えるようになります。

トリガーの作成する際に、以下を指定します。

- *サブジェクト表*。これは、トリガーが定義される表を指定します。
- *トリガー・イベント*。これは、サブジェクト表を変更する特定の SQL 操作を定義します。イベントには、挿入、更新、または削除操作があります。
- *トリガー起動タイミング*。これは、トリガー・イベントが発生する前か後のどちらで、トリガーを活動化するかを指定します。

トリガーを活動化するステートメントには、*影響を受ける行のセット* が組み入れられます。これらは、挿入、更新、または削除されるサブジェクト表の行です。*トリガー細分性* では、トリガーのアクションの実行をステートメントで 1 回か、または影響を受ける行ごとに 1 回かを指定します。

トリガー・アクション は、オプションの検索条件、およびトリガーが起動されると必ず実行される SQL ステートメントの集合で構成されます。 SQL ステートメントが実行されるのは、検索条件が true と評価された場合だけです。トリガー起動タイミングがトリガー・イベントの前の場合、トリガー・アクションに、 SELECT ステートメント、set 変数、または signal SQLstate を組み入れることができます。トリガー起動タイミングがトリガー・イベントの後の場合、トリガー・アクションに、SLECT、INSERT、UPDATE、DELETE ステートメント、または signal SQLstate を組み入れることができます。

トリガー・アクションでは、*遷移変数* を使用して、影響を受ける行のセット内の値を参照できます。遷移変数は、サブジェクト表の列の名前を使用します。この名前は、参照が古い値 (更新前) か新しい値 (更新後) かを示す、指定された名前によっ

て修飾されます。挿入または更新トリガーの前に、 SET Variable ステートメントを使用して新しい値を変更することもできます。

影響を受ける行のセット内の値を参照する別の方法は、*遷移表* を使用することです。遷移表では、サブジェクト表の列の名前も使用しますが、名前を指定することにより、影響を受ける行の完全なセットを表として扱うことができます。遷移表を使用できるのはトリガーの後だけであり、古い値と新しい値に、それぞれ別個の遷移表を定義できます。

表、イベント、起動時間の組み合わせで、複数のトリガーを指定できます。トリガーが活動化される順序は、作成された順序と同じです。そのため、一番あとに作成されたトリガーが、最後に活動化されます。

トリガーの活動化では、*トリガー・カスケード* が行われる場合があります。これは、 SQL ステートメントを実行するあるトリガーを活動化することにより、その SQL ステートメントによって、他のトリガーが活動化されるか、または同じトリガーが再度活動化された結果です。トリガー・アクションによって、削除に対する参照保全ルールのアプリケーションの結果である更新が行われることもあります、これにより、今度は、追加トリガーの活動化が行われる場合があります。トリガー・カスケードでは、トリガーおよび参照保全の削除規則のチェーンが活動化され、単一の INSERT、UPDATE、または DELETE ステートメントの結果として、データベースへの大幅な変更が行われる場合があります。

関連資料:

• 747 ページの『付録 H. トリガーと制約の相互作用』

表スペースおよびその他のストレージ構造

ストレージ構造には、データベース・オブジェクトが入れられます。基本となるストレージ構造は*表スペース* です。表スペースには、表、索引、ラージ・オブジェクト、および LONG データ型で定義されたデータが入っています。表スペースには、次の 2 つの種類があります。

データベース管理スペース (DMS)
データベース・マネージャーによって管理される表スペース。

システム管理スペース (SMS)
オペレーティング・システムによって管理される表スペース。

表スペースはすべて、いくつかのコンテナーで構成されます。*コンテナー* は、オブジェクトが保管されている場所を記述します。ファイル・システムのサブディレクトリーは 1 つのコンテナーの例です。

表スペースのコンテナーから読み取られたデータは、バッファー・プールと呼ばれるメモリーの領域に置かれます。*バッファー・プール* は特定の表スペースに関連付けられるため、どのデータがデータのバッファリングに同一のメモリー領域を共有するかは、制御できます。

パーティション・データベースでは、データは様々なデータベース・パーティションに分散されます。厳密にどのパーティションが組み込まれるかは、表スペースに割り当てられているデータベース・パーティション・グループによって決まりま

す。データベース・パーティション・グループ とは、データベースの一部と定義される 1 つまたは複数のパーティションのグループです。表スペースには、データベース・パーティション・グループの各パーティションごとに 1 つまたは複数のコンテナーが組み入れられます。特定のデータの行がどのパーティションに保管されるかは、各データベース・パーティション・グループに関連付けられている区分化マップ を使用して、データベース・マネージャーによって決定されます。区分化マップは、4096 のパーティション番号が配列されたものです。表の行ごとにパーティション関数によって生成された区分化マップの索引は、行が保管されるパーティションを判別するために区分化マップへの索引として使用されます。例として、次の図は、区分化キー値 (c1、c2、c3) の行が区分化マップの索引 2 にマップされ、この索引 2 がパーティション p5 を参照する様子を示しています。

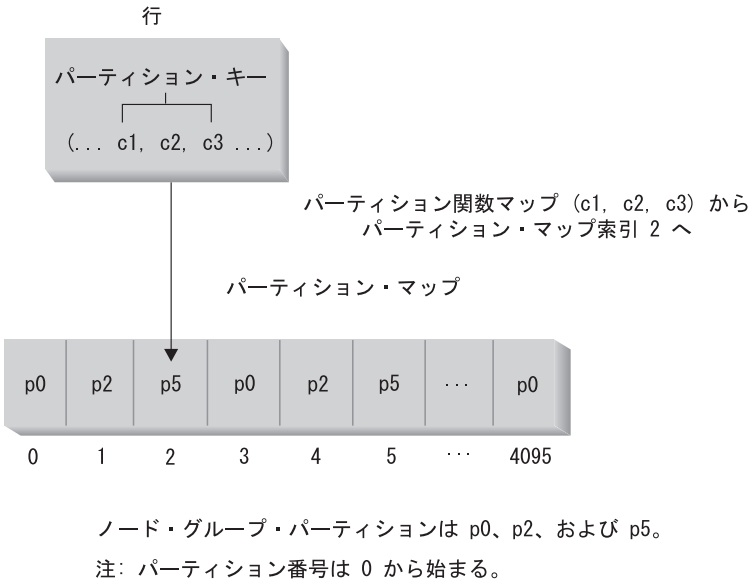


図 5. データ分散

区分化マップを変更することができ、それにより、区分化キーまたは実際のデータを変更することなくデータ分散を変更することができます。新しい区分化マップは、 REDISTRIBUTE DATABASE PARTITION GROUP コマンドまたは sqludrdt アプリケーション・プログラミング・インターフェース (API) の一部として指定されます。これらは、この区分化マップを使用してデータベース。データベース・パーティション・グループの表を再分散します。

DB2® データ・リンク・ファイル・マネージャーは、追加の保管能力をサポートする機能を備えています。通常のユーザー表には、外部ファイルに保管されているデータへのリンクを登録する列 (DATALINK データ型で定義されたもの) を組み込むことができます。DATALINK の値は、外部ファイル・サーバーに保管されているデータ・ファイルを指示します。

関連概念:

- 30 ページの『複数のパーティションにまたがるデータ・パーティション』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE BUFFERPOOL ステートメント』

- 「*SQL リファレンス 第 2 巻*」の『CREATE DATABASE PARTITION GROUP ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『CREATE TABLESPACE ステートメント』

複数のパーティションにまたがるデータ・パーティション

DB2® では、パーティション・データベースの複数のパーティション (ノード) にまたがってデータを柔軟に拡散させることができます。ユーザーは、区分化キーを宣言することによってデータをパーティション化する方法を選択することができ、また、データを保管するデータベース・パーティション・グループおよび表スペースを選択することによって、いくつのどのパーティションに表データを分散できるかを判別することができます。さらに、区分化マップ (更新可能) は、区分化キー値のパーティションへのマッピングを指定します。これにより、大きな表では複数のパーティション・データベースにまたがってワークロードを柔軟に均等化することができる一方で、アプリケーション設計者の選択しだいで 1 つまたは少数のパーティションに小さな表を保管することもできます。保管されるデータのローカル索引を各ローカル・パーティションに備えれば、ハイパフォーマンスのローカル・データ・アクセスの手段となります。

パーティション・データベースは、パーティション・ストレージ・モデルをサポートしています。このモデルでは、区分化キーを使用して一連のデータベース・パーティションに表データをパーティション分割します。索引データも、それに対応する表とともにパーティション化され、各パーティションにローカル保管されます。

パーティションを使用してデータベース・データを保管するには、事前にパーティションをデータベース・マネージャーに対して定義しておく必要があります。パーティションは、db2nodes.cfg というファイルに定義されます。

パーティション・データベース・パーティション・グループの表スペースの表の区分化キーは、 CREATE TABLE ステートメント (または ALTER TABLE ステートメント) に指定されます。指定されていない場合、デフォルト解釈によって、表の区分化キーは、主キーの最初の列から作成されます。主キーが定義されていない場合、デフォルトの区分化キーは、 long または LOB データ型以外のデータ型を持つ表に定義されている最初の列になります。パーティション分割された表には、データ型が long でも LOB でもない列が少なくとも 1 つは必要になります。明示的に指定されている場合、単一パーティション・データベース・パーティション・グループの表スペースの表は、区分化キーのみを持ちます。

ハッシュ・パーティション は、以下のようにパーティションに行を入れるのに使用されます。

1. ハッシュ・アルゴリズム (パーティション関数) が区分化キーのすべての列に適用され、その結果として区分化マップの索引の値が生成されます。
2. 区分化マップのその索引の値にあるパーティション番号は、行が保管されるパーティションを識別します。

DB2 は、*部分非クラスター化* をサポートします。これは、システム内のパーティションのサブセット (つまりデータベース・パーティション・グループ) 全体で表を

パーティション分割できることを意味しています。システム内のすべてのパーティションにわたって表をパーティション分割する必要はありません。

DB2 は、結合や副照会でアクセスされているデータが同じデータベース・パーティション・グループ内の同じパーティションにある場合に、それを認識する能力を備えています。これを、*表コロケーション* といいます。同一の区分化キー値を使用してコロケーションされている表の行は、同一のパーティションに置かれます。 DB2 は、データが保管されているパーティションでの結合処理や副照会処理の実行を選択できます。これによって、大幅なパフォーマンスの改善が得られる場合もあります。

連結する表は、以下の条件を満たしている必要があります。

- 同一のデータベース・パーティション・グループにあり、再分散されていない。(再分散されると、データベース・パーティション・グループ内の表は別の区分化マップを使用する可能性があります。このような表はコロケーションできません。)
- 表の列の区分化キーが同数でなければなりません。
- 区分化キーの対応する列に、パーティションの面で互換性がなければなりません。
- 表が、同一パーティションに定義されている単一のパーティション・データベース・パーティション・グループになければなりません。

関連資料:

- 133 ページの『パーティションの互換性データ型』

分散リレーショナル・データベース

分散リレーショナル・データベース は、別々のものであるが相互に接続されている複数のコンピューター・システムに散在する一連の表およびその他のオブジェクトで構成されています。コンピューター・システムごとにリレーショナル・データベース・マネージャーがあり、それぞれの環境で表を管理しています。データベース・マネージャーは、相互に情報をやり取りして協同で作業することにより、あるデータベース・マネージャーが別のコンピューター・システム上で SQL ステートメントを実行できるようになっています。

分散リレーショナル・データベースは、正式なリクエスター/サーバー・プロトコルと機能に基づいて構築されます。*アプリケーション・リクエスター* は、接続の両端のうち、アプリケーション側をサポートするものです。アプリケーション・リクエスターは、アプリケーションからのデータベース要求を分散データベース・ネットワークに適した通信プロトコルに変換します。このような要求は、接続のもう一方の側のデータベース・サーバー によって受信され、処理されます。アプリケーション・リクエスターとデータベース・サーバーは連携して通信とロケーションに関する考慮事項に取り組んで、アプリケーションがローカル・データベースにアクセスしているのと変わりなく稼働できるようにします。

アプリケーション・プロセスが表やビューを参照する SQL ステートメントを実行するためには、その前にデータベース・マネージャーのアプリケーション・サーバ

ーに接続する必要があります。アプリケーション・プロセスとそのサーバーとの接続を確立するには、 CONNECT ステートメントが使われます。

CONNECT ステートメントには、次の 2 つのタイプがあります。

- CONNECT (タイプ 1) では、作業単位 (リモート作業単位) セマンティクスごとに 1 つのデータベースがサポートされます。
- CONNECT (タイプ 2) では、作業単位 (アプリケーション制御の分散作業単位) セマンティクスごとに複数のデータベースがサポートされます。

DB2® コール・レベル・インターフェース (CLI) および組み込み SQL は*並行トランザクション* と呼ばれる接続モードをサポートします。これによりそれぞれが独立したトランザクションである複数の接続が可能になります。 1 つのアプリケーションが同じデータベースに対して複数の接続を並行して行うことができます。

アプリケーション・サーバーは、プロセスが開始される環境に対してローカルな位置にあっても、リモートの位置にあってもかまいません。アプリケーション・サーバーは、分散リレーショナル・データベースを使用していない環境でも存在しています。この環境には、 CONNECT ステートメントに指定されるアプリケーション・サーバーを記述するローカル・ディレクトリーが組み込まれています。

アプリケーション・サーバーは表やビューを参照するバインドした形式の静的 SQL ステートメントを実行します。このバインドされたステートメントはデータベース・マネージャーがバインド操作でそれ以前に作成したパッケージから取り出されます。

ほとんどの部分で、アプリケーション・サーバーに接続しているアプリケーションは、そのアプリケーション・サーバーのデータベース・マネージャーでサポートされているステートメントや文節を使用できます。このことは、一部のステートメントや文節をサポートしない データベース・マネージャーのアプリケーション・リクエスターによってアプリケーションが実行される場合でも当てはまります。

リモート作業単位

リモート作業単位機能 は、 SQL ステートメントをリモートで作成および実行するためのものです。コンピューター・システム A のアプリケーション・プロセスは、コンピューター・システム B のアプリケーション・プロセスのアプリケーション・サーバーに接続し、 1 つ以上の作業単位の中で、 B のオブジェクトを参照する任意の数の静的または動的 SQL ステートメントを実行できます。 B での作業単位が終了後、このアプリケーション・プロセスはコンピューター・システム C のアプリケーション・サーバーに接続する、というようなことが可能です。

以下の制限付きで、ほとんどの SQL ステートメントをリモートで準備、実行することができます。

- 単一の SQL ステートメントで参照されるオブジェクトは、すべて同一のアプリケーション・サーバーによって管理される必要があります。
- ある作業単位内の SQL ステートメントは、すべて同一のアプリケーション・サーバーによって実行される必要があります。

ある一時点で見ると、アプリケーション・プロセスは以下の 4 つの*接続状態* のいずれかにあります。

- 接続可能/接続済み

  アプリケーション・プロセスがアプリケーション・サーバーに接続されていて、CONNECT ステートメントの実行が可能です。

  暗黙の接続が可能な場合は以下のようになります。

  – 接続可能/未接続状態で CONNECT TO ステートメントまたはオペランドなしの CONNECT ステートメントが正常に実行されると、アプリケーション・プロセスは接続可能/接続済み状態になります。
  – また、CONNECT RESET、DISCONNECT、SET CONNECTION、または RELEASE ステートメント以外の SQL ステートメントが発行された場合は、アプリケーション・プロセスは暗黙接続可能状態からこの接続可能/接続済み状態になることがあります。

  暗黙の接続が可能か否かに関係なく、以下の場合にはこの接続可能/接続済み状態になります。

  – 接続可能/未接続状態から CONNECT TO ステートメントが正常に実行された場合。
  – 接続不可/接続済み状態から、COMMIT または ROLLBACK ステートメントが正常に発行されるか、または強制的なロールバックが発生した場合。

- 接続不可/接続済み

  アプリケーション・プロセスはアプリケーション・サーバーに接続されていますが、アプリケーション・サーバーを変更する CONNECT TO ステートメントを正常に実行することはできません。アプリケーション・プロセスが、CONNECT TO、オペランドなしの CONNECT、 CONNECT RESET、 DISCONNECT、 SET CONNECTION、 RELEASE、 COMMIT、 ROLLBACK 以外の SQL ステートメントを実行すると、接続可能/接続済み状態からこの状態になります。

- 接続可能/未接続

  アプリケーション・プロセスはアプリケーション・サーバーに接続していません。 CONNECT TO だけが実行できる SQL ステートメントで、それ以外を実行するとエラー (SQLSTATE 08003) になります。

  暗黙的接続が使用可能かどうかに関係なく、アプリケーション・プロセスは、CONNECT TO ステートメントの発行時にエラーが発生するか、または接続の消失とロールバックを引き起こすエラーが作業単位の中で発生した場合に、この接続可能/未接続状態になります。アプリケーション・プロセスが接続可能状態でないために発生したエラーや、サーバー名がローカル・ディレクトリーのリストにないために発生したエラーでは、この状態にはなりません。

  暗黙の接続が使用不可の場合は、以下のようになります。

  – アプリケーション・プロセスは最初にはこの状態にあります。
  – CONNECT RESET および DISCONNECT ステートメントが実行されると、この接続可能/未接続状態になります。

- 暗黙に接続可能 (暗黙接続が利用可能の場合)

暗黙の接続が可能な場合は、これがアプリケーション・プロセスの初期状態です。 CONNECT RESET ステートメントを使うと、この状態になります。接続不可/接続済み状態で COMMIT または ROLLBACK ステートメントを発行した後、接続可能/接続済み状態で DISCONNECT ステートメントを発行することによっても、この暗黙接続可能状態にすることができます。

暗黙接続の有効性は、インストール・オプション、環境変数と認証設定値によって決まります。

連続して複数の CONNECT ステートメントを実行することはエラーではありません。これは、CONNECT 自体ではアプリケーション・プロセスの接続可能状態は終了しないためです。しかし、連続的な CONNECT RESET ステートメントを実行するのはエラーです。CONNECT TO、CONNECT RESET、オペランドなしの CONNECT、 SET CONNECTION、RELEASE、COMMIT、ROLLBACK 以外の SQL ステートメントを実行した後も、 CONNECT TO ステートメントを実行すると、エラーになります。このエラーを回避するには、CONNECT TO ステートメントの前に、CONNECT RESET、 DISCONNECT (COMMIT または ROLLBACK ステートメントの後)、COMMIT、または ROLLBACK ステートメントを実行する必要があります。

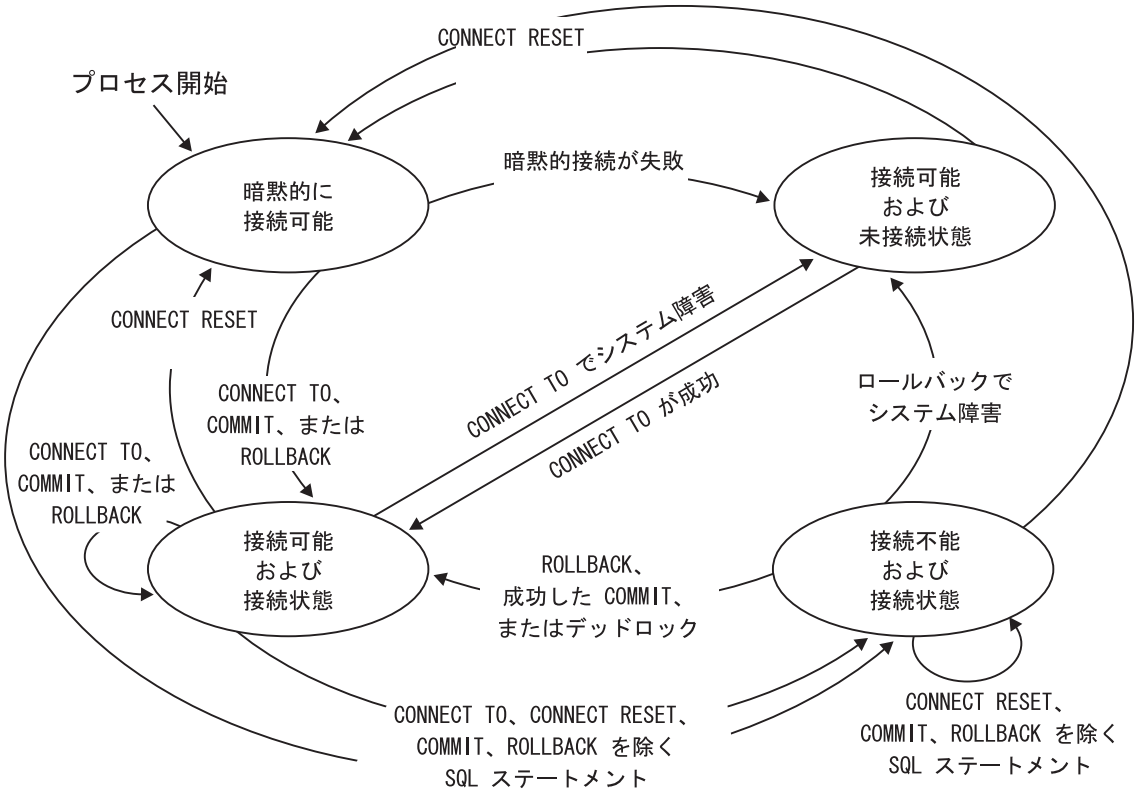


図 6. 暗黙的接続が使用可能な場合の接続状態の遷移

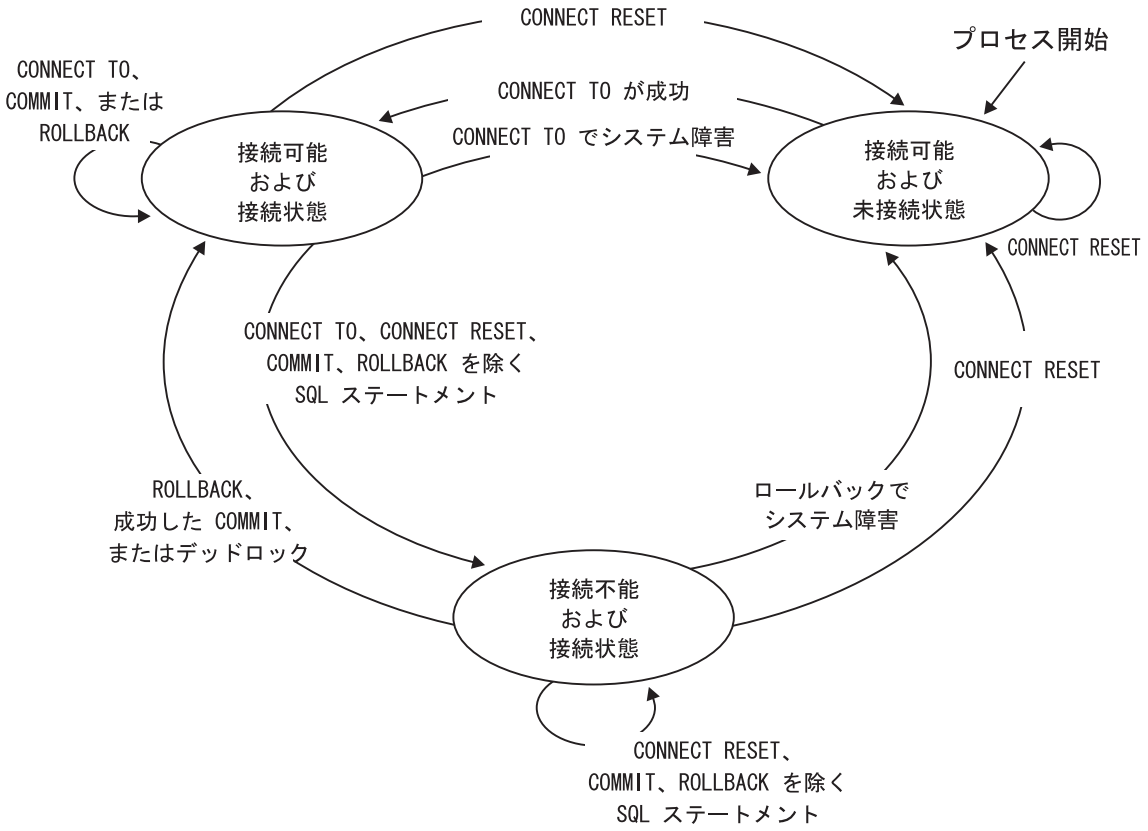


図 7. 暗黙の接続が使用不可な場合の接続状態の遷移

アプリケーション制御の分散作業単位機能

*アプリケーション制御の分散作業単位機能*を使えば、 SQL ステートメントをリモートで作成し、実行することができます。コンピューター・システム A のアプリケーション・プロセスは、 CONNECT または SET CONNECTION ステートメントを発行することにより、コンピューター・システム B のアプリケーション・サーバーに接続できます。作業単位の終了までの間に、アプリケーション・プロセスは、 B のオブジェクトを参照する任意の数の静的または動的 SQL ステートメントを実行することができます。単一の SQL ステートメントで参照されるオブジェクトは、すべて同一のアプリケーション・サーバーによって管理される必要があります。しかし、リモート作業単位機能とは異なり、複数のアプリケーション・サーバーが同じ作業単位に加わることができます。コミットまたはロールバック操作で、作業単位が終了します。

アプリケーション制御の分散作業単位では、タイプ 2 の接続を使用します。タイプ 2 の接続は、指定されたアプリケーション・サーバーにアプリケーション・プロセスを接続し、アプリケーション制御の分散作業単位のための規則を確立します。

タイプ 2 のアプリケーション・プロセスは、以下の状態になっています。

- 常に接続可能です
- 接続済み状態または未接続状態のいずれかです
- ゼロ個以上の接続があります

アプリケーション・プロセスの各接続は、その接続のアプリケーション・サーバーのデータベース別名によって、固有識別されます。

個々の接続は、常に以下の接続状態のいずれか 1 つになっています。

- 現行で保持
- 現行で解放ペンディング
- 休止で保持
- 休止で解放ペンディング

タイプ 2 のアプリケーション・プロセスは当初、未接続状態であり、接続はありません。初期の接続は、現行で保持状態です。

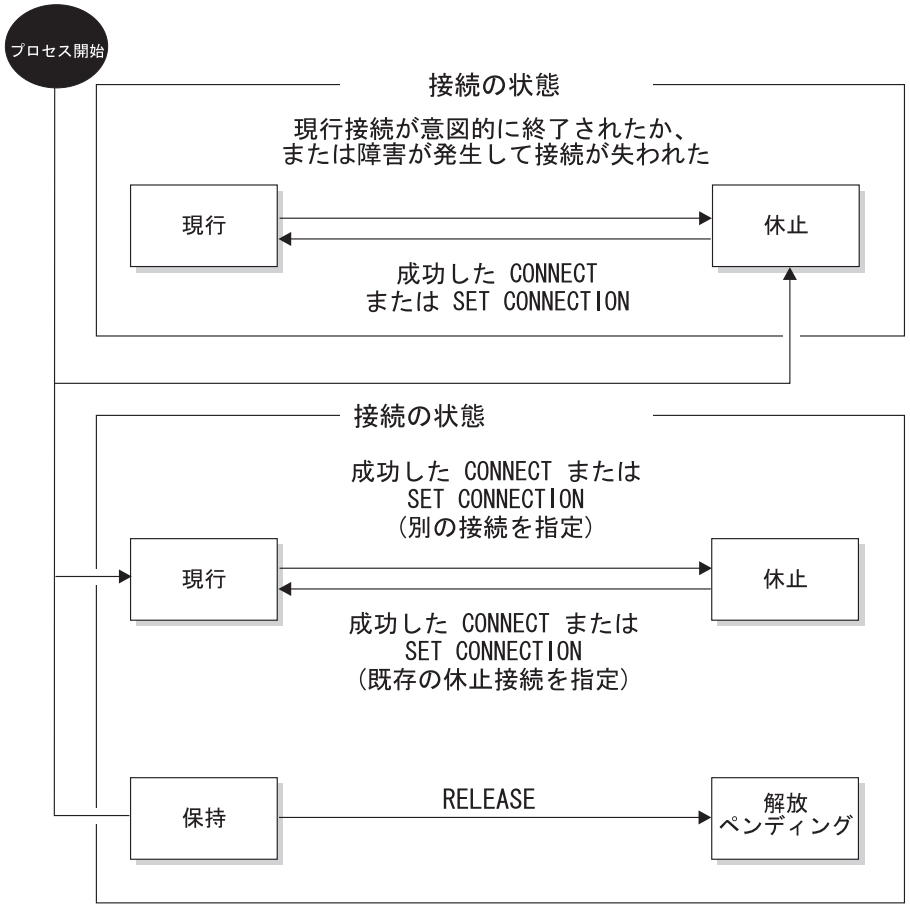


図 8. アプリケーション制御の分散作業単位の接続状態の遷移

アプリケーション・プロセスの接続状態

CONNECT ステートメントの実行には、次の規則が適用されます。

- コンテキストは、 1 つのアプリケーション・サーバーに対して同時に 2 つ以上の接続を持つことはできません。
- アプリケーション・プロセスが SET CONNECTION ステートメントを実行する場合、指定する位置名は、そのアプリケーション・プロセスの接続の集合に入っている既存の接続でなければなりません。

- アプリケーション・プロセスが CONNECT ステートメントを実行する場合、SQLRULES(STD) オプションが有効なら、指定するサーバー名は、そのアプリケーション・プロセスの接続の集合に入っている既存の接続であってはなりません。 SQLRULES については、38 ページの『分散作業単位のセマンティクスを規制するオプション』 を参照してください。

アプリケーション・プロセスに現行の接続がある場合、アプリケーション・プロセスは*接続済み* 状態です。 CURRENT SERVER 特殊レジスターに、現行の接続のアプリケーション・サーバーの名前が入れられています。アプリケーション・プロセスは、アプリケーション・サーバーによって管理されるオブジェクトを参照する SQL ステートメントを実行できます。

未接続状態のアプリケーション・プロセスは、 CONNECT または SET CONNECTION ステートメントが正常に実行されたときに、接続済み状態になります。接続がないのに SQL ステートメントが発行された場合、 DB2DBDFT 環境変数にデフォルトのデータベースが設定されているなら、暗黙の接続が行われます。

アプリケーション・プロセスに現行の接続がない場合、アプリケーション・プロセスは*未接続* 状態です。実行できる SQL ステートメントは、CONNECT、DISCONNECT ALL、(データベースを指定した) DISCONNECT、 SET CONNECTION、RELEASE、COMMIT、ROLLBACK、および local SET ステートメントだけです。

接続済み状態 のアプリケーション・プロセスが*未接続状態* になるのは、現行の接続が意図的に終了された場合、または SQL ステートメントが正常に実行されずアプリケーション・サーバーでのロールバック操作を引き起こし接続が失われてしまう場合です。接続が解放ペンディングのときに、 DISCONNECT ステートメントか COMMIT ステートメントが正常に実行されると、接続は意図的に終了させられます。 (DISCONNECT プリコンパイラー・オプションが AUTOMATIC に設定されていると、すべての接続は終了します。 CONDITIONAL に設定されている場合は、オープンされている WITH HOLD カーソルのない接続がすべて終了します。)

接続状態

アプリケーション・プロセスが CONNECT ステートメントを実行し、サーバー名はアプリケーション・リクエスターで分かっているが、そのサーバー名がそのアプリケーション・プロセスの既存の接続の集合にない場合は、以下のようになります。

- 現行の接続は*休止接続状態* になります。また、
- そのサーバー名が接続の集合に追加されます。また、
- 新しい接続は*現行接続状態* かつ*保持接続状態* になります。

サーバー名がすでにアプリケーション・プロセスの既存の接続の集合に入っている場合に、アプリケーションを SQLRULES(STD) オプション付きでプリコンパイルすると、エラー (SQLSTATE 08002) になります。

保持状態および解放ペンディング RELEASE ステートメントは、接続が保持状態か解放ペンディングかを制御します。*解放ペンディング* とは、次の正常なコミット操作で切断される状態のことです。 (ロールバックは接続に影響しません。) *held* 状態は、次のコミット操作で切断が起きない ことを意味します。

すべての接続は、最初は保持状態であり、 RELEASE ステートメントを使うと解放ペンディングになります。接続が、いったん解放ペンディングになると、保持状態に戻ることはありません。 ROLLBACK ステートメントが発行されるか、またはコミット操作の異常によりロールバック操作が発生した場合、接続は作業単位の境界を超えて解放ペンディングのまま残ります。

接続に対して解放が明示されていなくても、コミット操作が DISCONNECT プリコンパイラー・オプションの条件を満たしていれば、コミット操作によって切断可能です。

現在の状態および休止状態 これは、接続が保持状態または解放ペンディングのどちらであるかに関係なく、現行状態もしくは休止状態になることもあります。*現行* 状態にある接続とは、この状態にある間に SQL ステートメントの実行に使用される接続のことです。*休止* 状態にある接続とは、現行でない接続のことです。

休止状態の接続で実行できる SQL ステートメントは、COMMIT、ROLLBACK、DISCONNECT、RELEASE のみです。 SET CONNECTION および CONNECT ステートメントは指定したサーバーの接続状態を現行状態に変え、既存の接続は休止状態になるか休止状態のままになります。どの時点でも、現行状態にある接続は 1 つだけです。同じ作業単位の中で休止接続が現行状態に変わると、すべてのロック、カーソル、作成済みステートメントの状態は最後にその接続が現行であったときの状態と同じです。

接続の終了時

接続が終了すると、アプリケーション・プロセスが接続によって獲得していたすべてのリソース、および接続を確立し維持するために使用されたすべてのリソースが割り振り解除されます。たとえば、アプリケーション・プロセスが RELEASE ステートメントを実行すると、その次のコミット操作で接続が終了するときにオープンしているカーソルがすべてクローズされます。

接続は、通信の障害によっても終了することがあります。この接続が現行状態にあると、アプリケーション・プロセスは未接続状態になります。

アプリケーション・プロセスの接続はすべてそのプロセスが終了するときに終了します。

分散作業単位のセマンティクスを規制するオプション

タイプ 2 接続管理のセマンティクスは、プリコンパイラーのオプション群によって決定されます。これらのオプションについて以下に要約します。デフォルト値は太字に下線を付けたテキストで示します。

- CONNECT (**1** | 2)。 CONNECT ステートメントが、タイプ 1 とタイプ 2 のどちらとして処理されるかを指定します。
- SQLRULES (**DB2** | STD)。タイプ 2 の CONNECT が、 CONNECT による休止接続への切り替えを認める DB2 規則に従って処理されるのか、それともその切り替えを認めない SQL92 標準規則に従って処理されるのかを指定します。
- DISCONNECT (**EXPLICIT** | CONDITIONAL | AUTOMATIC)。以下のようにコミット操作の発生時に切断されるデータベース接続を指定します。
  - SQL の RELEASE ステートメントによって解放するよう明示的に指定されているデータベース接続 (EXPLICIT)

  - オープンされている WITH HOLD カーソルのないもの、および解放のためにマークされたもの (CONDITIONAL)
  - すべての接続 (AUTOMATIC)
- SYNCPOINT (**ONEPHASE** | TWOPHASE | NONE)。複数のデータベース接続にまたがって COMMIT または ROLLBACK を調整する仕方を指定します。
  - 作業単位の中で更新を行えるのは 1 つのデータベースに対してだけです。他のデータベースはすべて読み取り専用です (ONEPHASE)。他のデータベースに対して更新しようとすると、エラー (SQLSTATE 25000) になります。
  - ランタイムにトランザクション・マネージャー (TM) を使って、このプロトコルをサポートするデータベースの間で 2 フェーズ COMMIT を調整します (TWOPHASE)。
  - 2 フェーズ COMMIT を実行する TM を使用せず、単一更新・複数読み取りを強制しません (NONE)。 COMMIT または ROLLBACK ステートメントが実行されると、個々の COMMIT または ROLLBACK がすべてのデータベースに通知されます。 1 つ以上の ROLLBACK が失敗すると、エラー (SQLSTATE 58005) になります。 1 つ以上の COMMIT が失敗すると、別のエラー (SQLSTATE 40003) になります。

ランタイムに上記のいずれかのオプションをオーバーライドするには、 SET CLIENT コマンドまたは sqlesetc アプリケーション・プログラミング・インターフェース (API) を使用してください。その現在の設定値は QUERY CLIENT コマンドまたは sqleqryc API を使用して取得できます。これらは SQL ステートメントではなく、さまざまなホスト言語およびコマンド行プロセッサー (CLP) で定義されている API であることに注意してください。

データ表記の考慮事項

システムが異なると、データを表現する方式も異なります。データをあるシステムから別のシステムへ移動する場合、データ変換が必要なことがあります。 DRDA® をサポートする製品は、データを受け取る側のシステムで必要な変換を自動的に実行します。数値データの変換を実行するには、データ型と、そのデータ型が送り側システムでどのように表現されるかという情報がシステムに必要です。文字ストリングの変換のためにはさらに情報が必要です。ストリングの変換は、データのコード・ページと、そのデータに対して実行する操作の両方に応じて変わります。文字変換は、IBM® Character Data Representation Architecture (CDRA) に従って実行されます。文字変換の詳細については、 *Character Data Representation Architecture: Reference & Registry* (SC09-2190-00) マニュアルを参照してください。

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CONNECT (タイプ 1) ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『CONNECT (タイプ 2) ステートメント』

DB2 フェデレーテッド・システム

フェデレーテッド・システム

DB2® フェデレーテッド・システム は、特別なタイプの分散データベース管理システム (DBMS) です。フェデレーテッド・システムは、フェデレーテッド・サーバーとして働く DB2 インスタンス、フェデレーテッド・データベースとして働くデータベース、1 つまたは複数のデータ・ソース、およびデータベースとデータ・ソースにアクセスするクライアント (ユーザーおよびアプリケーション) からなります。フェデレーテッド・システムを使用すると、1 つの SQL ステートメントで複数のデータ・ソースに分散要求を送信することができます。たとえば、DB2 Universal Database™ 表、Oracle 表、およびXML タグ付きファイル内にあるデータを、1 つの SQL ステートメントに結合することができます。以下の図は、フェデレーテッド・システムのコンポーネントと、アクセス対象のデータ・ソースを示しています。

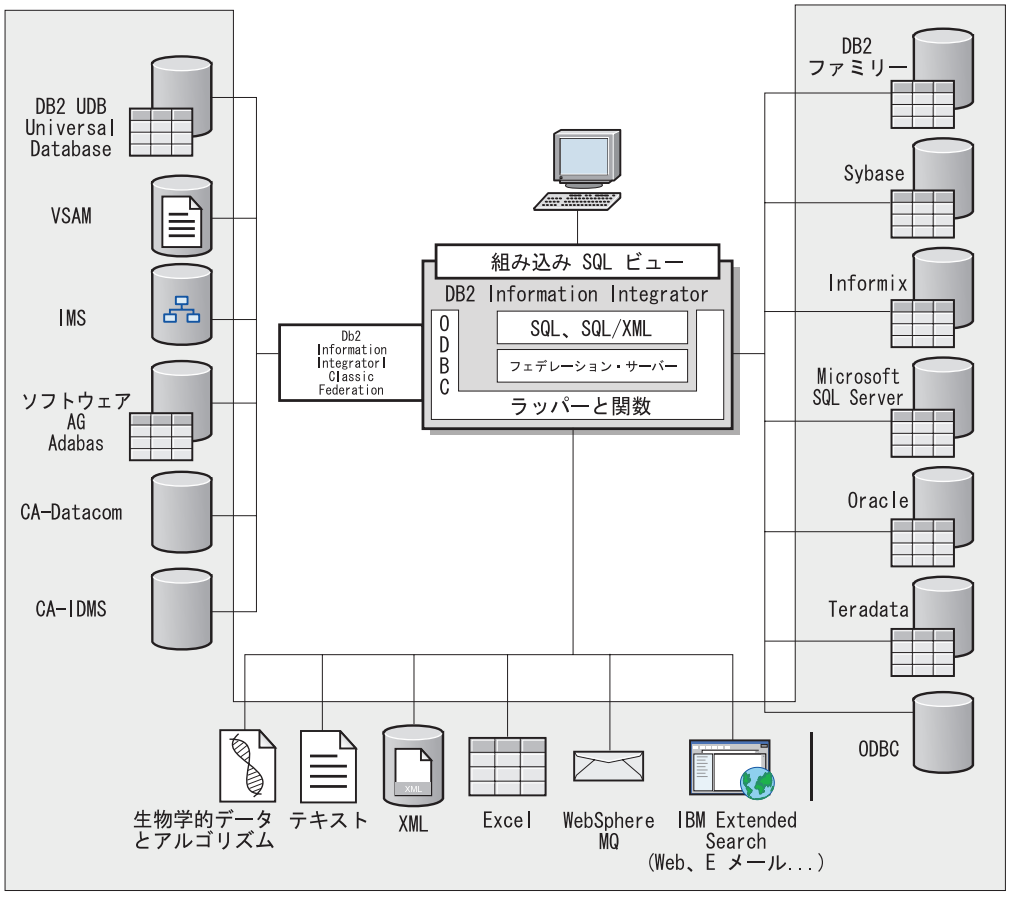


図 9. フェデレーテッド・システムのコンポーネント

DB2 フェデレーテッド・システムの機能では、以下が可能になります。

- ローカル表とリモートのデータ・ソースのデータを、すべてのデータがローカルにある場合と変わらずに結合する。
- フェデレーテッド・データベースにデータが保管されている場合と変わらずに、リレーショナル・データ・ソース内のデータを更新する。

- リレーショナル・データ・ソースを対象にデータを複製する。
- 処理用の要求をデータ・ソースに送信して、データ・ソースの処理能力を活用する。
- データ・ソース側での SQL の制約に対処するため、分散要求の一部をフェデレーテッド・サーバー側で処理する。

フェデレーテッド・サーバー

フェデレーテッド・システム内の DB2® サーバーは、フェデレーテッド・サーバーと呼ばれます。任意の数の DB2 インスタンスを、フェデレーテッド・サーバーとして機能するように構成することができます。既存の DB2 インスタンスをフェデレーテッド・サーバーとして使用することができますが、フェデレーテッド・システム専用に新しく作成することもできます。

フェデレーテッド・システムを管理する DB2 インスタンスをサーバー と呼びますが、その理由は、エンド・ユーザーおよびクライアント・アプリケーションからの要求に応答するからです。フェデレーテッド・サーバーが受信した要求の一部をデータ・ソースに送信して処理させることはよくあります。プッシュダウン 操作は、リモート側で処理される操作です。フェデレーテッド・システムを管理する DB2 インスタンスをフェデレーテッド・サーバー と呼びますが、このインスタンスは、要求をデータ・ソースにプッシュダウンする場合はクライアントとして働きます。

他のすべてのアプリケーション・サーバーと同様に、フェデレーテッド・サーバーはデータベース・マネージャー・インスタンスです。アプリケーション・プロセスは、フェデレーテッド・サーバー内のデータベースに接続して要求をサブミットします。ただし、次の 2 つの主要な機能により、その他のアプリケーション・サーバーとは区別されます。

- フェデレーテッド・サーバーは、部分的または全面的にデータ・ソース向けである要求を受信するように構成されています。フェデレーテッド・サーバーは、これらの要求をデータ・ソースに配布します。
- その他のアプリケーション・サーバーと同様に、フェデレーテッド・サーバーは DRDA® 通信プロトコル (TCP/IP を介して) を使用して、 DB2 ファミリーのインスタンスと通信します。ただし、他のアプリケーション・サーバーとは違って、フェデレーテッド・サーバーはデータ・ソースのネイティブ・クライアントを使用してデータ・ソースにアクセスします。たとえば、フェデレーテッド・サーバーは、Sybase Open Client を使って Sybase データ・ソースにアクセスし、Microsoft® SQL Server ODBC ドライバーを使って Microsoft SQL Server データ・ソースにアクセスします。

関連概念:

- 41 ページの『データ・ソースとは何か』

データ・ソースとは何か

フェデレーテッド・システムではデータ・ソース は、リレーショナル DBMS インスタンス (Oracle や Sybase など) または非リレーショナル・データ・ソース (BLAST 検索アルゴリズムは XML タグ付きファイル) のどちらでもかまいません。他のデータ・ソースへのアクセスの足がかりになるデータ・ソースもありま

す。たとえば、Extended Search データ・ソースを経由して、Lotus® Notes データベース、Microsoft® Access、Microsoft Index Server、Web 検索エンジン、Lightweight Directory Access Protocol (LDAP) ディレクトリーなどのデータ・ソースにアクセスすることができます。

データ・ソースへのアクセスに使用する方式つまりプロトコルは、データ・ソースのタイプによって異なります。たとえば、DB2® for z/OS™ and OS/390® のデータ・ソースにアクセスするには DRDA® を使用し、Documentum データ・ソースにアクセスするには Documentum Client API/Library を使用します。

データ・ソースは準自立的です。たとえば、フェデレーテッド・サーバーは、Oracle アプリケーションがデータ・ソースにアクセスしている同じ時に、それらの Oracle データ・ソースに照会を送信することができます。 DB2 フェデレーテッド・システムは、保全性およびロッキング制約を超えて、他のデータ・ソースへのアクセスを独占したり、制限することはありません。

関連概念:

• 44 ページの『フェデレーテッド・データベース』

関連資料:

• 42 ページの『サポートされているデータ・ソース』

サポートされているデータ・ソース

フェデレーテッド・システムを使ってアクセスできるデータ・ソースは多数あります。以下の表は、サポートされているデータ・ソースを一覧で示しています。

表 1. サポートされているデータ・ソースのバージョンおよびアクセス方式

データ・ソース	サポートされるバージョン	アクセス方式
DB2 Universal Database™ for Linux、UNIX、および Windows®	7.2、8.1、8.2	DRDA®
DB2 Universal Database for z/OS™ and OS/390®	6.1、7.1 (以下の APAR を適用済みのもの) • PQ62695 • PQ55393 • PQ56616 • PQ54605 • PQ46183 • PQ62139 8.1	DRDA

表 1. サポートされているデータ・ソースのバージョンおよびアクセス方式 (続き)

データ・ソース	サポートされるバージョン	アクセス方式
DB2 Universal Database for iSeries™	5.1 • 次のような APAR を適用済み – SE06003 – SE06872 – II13348 • 次のような PTF を適用済み – SI05990 SI05991 PTF SI0735 を適用済みの 5.2	DRDA
DB2 Server for VM および DB2 Server for VSE	7.1 またはそれ以上 (スキーマ機能用の APAR を適用済みのもの)	DRDA
Informix™	7.31、8.32、8.4、9.3、9.4	Informix Client SDK V2.7 またはそれ以上
ODBC	3.x	データ・ソース用の ODBC ドライバー (Redbrick へのアクセス用の Redbrick ODBC ドライバーなど)
OLE DB	2.7、2.8	OLE DB 2.0 以上
Oracle	8.0.6、8.1.6、8.1.7、9.0、9.1、9.2、9i、10g	Oracle net client または NET8 クライアント・ソフトウェア
Microsoft SQL Server	7.0。このリリース上の 2000 SP3 以上の Service Pack	Windows では、Microsoft SQL Server Client ODBC 3.0 以上のドライバー。 UNIX では、DataDirect Direct Technologies (旧称 MERANT) Connect ODBC 3.7 以上のドライバー
Sybase	11.9.2、12.x	Sybase Open Client ctlib インターフェース
Teradata	V2R3、V2R4、V2R5	Teradata Call-Level Interface バージョン 2 (CLIv2) リリース 04.06 以上
BLAST	2.2.3 以上の 2.2 フィックスパックがサポートされる	BLAST デーモン (ラッパーに付属)
BioRS	v5.0.14	なし

表 1. サポートされているデータ・ソースのバージョンおよびアクセス方式 (続き)

データ・ソース	サポートされるバージョン	アクセス方式
Documentum	3.x、4.x	Documentum Client library/APL3.1.7a またはそれ以上
Entrez (PubMed および GenBank データ・ソース)	1.0	なし
HMMER	2.2g、2.3	HMMER デーモン (ラッパーに付属)
IBM Lotus Extended Search	4.0.1、4.0.2	Extended Search Client Library (ラッパーに付属)
Microsoft Excel	97、2000、2002、2003	Excel 97、2000、2002、または 2003 (フェデレーテッド・サーバー上にインストール済み)
PeopleSoft	8.x	PeopleSoft 用の IBM WebSphere Business Integration Adapter v2.3.1、2.4
SAP	3.x、4.x	mySAP.com 用の IBM WebSphere Integration Adapter v2.3.1、2.4
Siebel	7、7.5、2000	Siebel eBusiness Applications 用 IBM WebSphere Business Integration Adapter v2.3.1, 2.4
表構造ファイル		なし
KEGG 用のユーザー定義関数	サポートされる	
Life Sciences 用のユーザー定義関数	サポートされる	
Web サービス	SOAP 1.0.、1.1、WSDL 1.0、1.1 の仕様	HTTP
XML	1.0 仕様	なし

関連概念:

- 41 ページの『データ・ソースとは何か』

フェデレーテッド・データベース

エンド・ユーザーとクライアント・アプリケーションからは、データ・ソースは DB2® 内の 1 つの集合データベースと見えます。ユーザーとアプリケーションは、フェデレーテッド・サーバーが管理するフェデレーテッド・データベース とやり取りを行います。フェデレーテッド・データベースにはシステム・カタログが収められています。フェデレーテッド・データベースのシステム・カタログには、データ・ソースとその特性を示す項目が入っています。フェデレーテッド・サーバー

は、フェデレーテッド・データベース・システム・カタログに保管された情報および、データ・ソース・ラッパーを検討した上で、 SQL ステートメントを処理する最善策を決めます。

フェデレーテッド・システムは、データ・ソースがフェデレーテッド・データベース内の通常のリレーショナルの表またはビューであるものとして SQL ステートメントを処理します。その結果、以下のようになります。

- フェデレーテッド・システムは、非リレーショナル・フォーマットのデータにリレーショナル・データを結合することができます。これは、データ・ソースが異なる SQL ダイアレクトを使用していたり、あるいは SQL をまったくサポートしていなくても、あてはまります。
- フェデレーテッド・データベースの特性とデータ・ソースの特性に差異がある場合、次のように、フェデレーテッド・データベースの特性が優先されます。
  - フェデレーテッド・サーバーで使われるコード・ページが、データ・ソースで使われるコード・ページとは異なると仮定します。データ・ソースの文字データは、フェデレーテッド・ユーザーに戻されるときは、フェデレーテッド・データベースで使われているコード・ページに基づいて変換されます。
  - フェデレーテッド・サーバーで使われる照合シーケンスが、データ・ソースで使われる照合シーケンスとは異なると仮定します。この場合、文字データのソート操作はすべて、データ・ソースではなくフェデレーテッド・サーバーで実行されます。

関連概念:

- 46 ページの『SQL コンパイラー』
- 45 ページの『フェデレーテッド・データベースのシステム・カタログ』

フェデレーテッド・データベースのシステム・カタログ

フェデレーテッド・データベース・システム・カタログには、フェデレーテッド・データベース内のオブジェクトの情報および、データ・ソース側のオブジェクトの情報が入っています。フェデレーテッド・データベース内のカタログは*グローバル・カタログ* と呼ばれますが、その理由は、フェデレーテッド・システム全体についての情報が入っているためです。 DB2® 照会オプティマイザーは、グローバル・カタログ内の情報およびデータ・ソース・ラッパーを使用して、SQL ステートメントを処理する最善の方法を計画します。グローバル・カタログに保管される情報には、リモートとローカルの情報、たとえば列名、列のデータ型、列のデフォルト値、および索引情報などがあります。

*リモート・カタログ情報*は、データ・ソースが使用する情報または名前です。*ローカル・カタログ情報*は、フェデレーテッド・データベースが使用する情報または名前です。たとえば、*EMPNO* という名前の列を持つリモート表があるとします。グローバル・カタログには、このリモートの列名が *EMPNO* として保管されます。別の名前を指定しないかぎり、ローカルの列名は *EMPNO* として保管されます。ローカルの列名を *Employee_Number* に変更することができます。この列を組み入れられた照会をサブミットするユーザーは、照会の中で *EMPNO* ではなく *Employee_Number* を使用します。データ・ソースの列のローカル名を変更するには、ALTER NICKNAME ステートメントを使用します。

リレーショナル・データ・ソースの場合、グローバル・カタログに保管される情報はリモートとローカルの両方の情報で構成されます。

非リレーショナル・データ・ソースの場合、グローバル・カタログに保管される情報はデータ・ソースによって異なります。

グローバル・カタログに保管されたデータ・ソース表の情報を見るには、フェデレーテッド・データベース内の SYSCAT.TABLES、SYSCAT.TABOPTIONS、SYSCAT.INDEXES、SYSCAT.COLUMNS、および SYSCAT.COLOPTIONS カタログ・ビューを照会してください。

グローバル・カタログには、データ・ソースについてのその他の情報も入っています。たとえば、フェデレーテッド・サーバーがデータ・ソースに接続したり、フェデレーテッド・ユーザー権限をデータ・ソースのユーザー権限にマップしたりするのに使用する情報がグローバル・カタログに入っています。グローバル・カタログには、サーバー・オプションなどの、明示的に設定するデータ・ソースに関する属性が収められます。

関連概念:

- 46 ページの『SQL コンパイラー』

関連資料:

- 「フェデレーテッド・システム・*ガイド*」の『フェデレーテッド情報を含むグローバル・カタログ表内のビュー』

SQL コンパイラー

データ・ソースからデータを入手するため、ユーザーおよびアプリケーションは DB2® SQL の照会をフェデレーテッド・データベースにサブミットします。照会をサブミットすると、 DB2 SQL コンパイラーはグローバル・カタログ内の情報およびデータ・ソース・ラッパーを検討し、照会の処理に役立てます。これには、データ・ソースへの接続についての情報、サーバー属性、マッピング、索引情報、および処理統計も入ります。

関連概念:

- 49 ページの『ラッパーおよびラッパー・モジュール』
- 46 ページの『照会オプティマイザー』

照会オプティマイザー

SQL コンパイラー処理の一環として、*照会オプティマイザー* は照会を分析します。コンパイラーは、照会を処理するための代替ストラテジー (アクセス・プランと呼ばれる) を作成します。アクセス・プランでは、次のように照会が処理される必要があります。

- データ・ソースによって処理される。
- フェデレーテッド・サーバーによって処理される。
- 一部がデータ・ソースによって処理され、一部がフェデレーテッド・サーバーによって処理される。

DB2® は、主にデータ・ソースの能力およびデータに関する情報を基にアクセス・プランを評価します。この情報はラッパーとグローバル・カタログにあります。DB2 UDB は照会を*照会フラグメント* と呼ばれるセグメントに分解します。通常、照会フラグメントをデータ・ソースにプッシュダウン した方が、より効率的です (データ・ソースがフラグメントを処理できる場合)。しかし、照会オプティマイザーでは次のような他の要素にも配慮されます。

- 処理する必要のあるデータの量。
- データ・ソースの処理速度。
- フラグメントが戻すデータの量。
- 通信の帯域幅。
- 同じ照会結果を表す使用可能なマテリアライズ照会表がフェデレーテッド・サーバー上にあるかどうか。

照会オプティマイザーは、照会フラグメントを処理するためのローカルとリモートのアクセス・プランを、リソースの費用に基づいて生成します。それから、DB2 UDB は最小のリソース費用で照会を処理すると思われるプランを選択します。

何らかのフラグメントをデータ・ソースで処理する場合、DB2 UDB はそれらのフラグメントをデータ・ソースにサブミットします。データ・ソースがフラグメントを処理した後、結果が取り出されて DB2 UDB に戻されます。 DB2 UDB が処理の一部を実行した場合、自身の処理結果とデータ・ソースから取り出した結果を組み合わせます。それから、DB2 UDB はすべての結果をクライアントに戻します。

関連概念:

- 46 ページの『SQL コンパイラー』
- 47 ページの『適合』
- 「フェデレーテッド・システム・ガイド」の『照会処理のチューニング』

適合

DB2® フェデレーテッド・サーバーは、データ・ソースが照会フラグメントを処理できない場合、またはデータ・ソースで処理するよりもフェデレーテッド・サーバーで処理した方が速い場合は、照会フラグメントをプッシュダウンしません。たとえば、データ・ソースの SQL ダイアレクトが、 GROUP BY 文節の CUBE グループ化をサポートしないとします。この場合、CUBE 化を収容していて、そのデータ・ソース内の表を参照する照会は、フェデレーテッド・サーバーにサブミットされます。 DB2 Information Integrator は CUBE グループ化をデータ・ソースにプッシュダウンせず、 CUBE そのものを処理します。データ・ソースではサポートされない SQL を処理する DB2 Information Integrator の機能を、*補償* と呼びます。

フェデレーテッド・サーバーは、データ・ソース側の機能不足を次の 2 つの方法で補います。

- 照会で使用される DB2 関数と同等の、1 つまたは複数の操作を使用することをデータ・ソースに求めることができます。たとえば、データ・ソースはコタンジェント (COT(x)) 関数をサポートしないが、タンジェント (TAN(x)) 関数はサポー

トするとします。この場合 DB2 Information Integrator は、コタンジェント (COT(x)) 関数と同等の計算 (1/TAN(x)) の実行をデータ・ソースに要求することができます。

- データの集合をフェデレーテッド・サーバーに戻し、関数をローカルで実行することができます。

リレーショナル・データ・ソースの場合、どのタイプの RDBMS も、国際 SQL 標準の 1 つのサブセットをサポートします。さらに、ある種の RDBMS は、この標準を超えた SQL 構文をサポートします。 *SQL ダイアレクト* とは、あるタイプの RDBMS がサポートする SQL 全体を指す用語です。ある SQL 構成が DB2 の SQL ダイアレクトにあり、データ・ソース側のダイアレクトにはない場合、フェデレーテッド・サーバーはデータ・ソースに代わってこの構成をインプリメントすることができます。

DB2 Information Integrator は、SQL ダイアレクトどうしの間の相違を補正することができます。この機能の一例に、共通表式の文節があります。 DB2 SQL には、共通表式という文節が組み込まれています。この文節で、ある名前を指定すると、この名前を使用して、全選択内のすべての FROM 文節が結果セットを参照できます。フェデレーテッド・サーバーは、データ・ソースで使用される SQL ダイアレクトに共通表式が入っていない場合でも、データ・ソース用に共通表式を処理します。

補正をすることによってフェデレーテッド・サーバーは、データ・ソースの照会における DB2 SQL ダイアレクトを完全にサポートすることができます。 SQL サポートが少ないかまたはこのサポートのないデータ・ソースでも、適合を活用することができます。フェデレーテッド・システムには、パススルー・セッションを除き、 DB2 SQL ダイアレクトを使用する必要があります。

関連概念:

- 48 ページの『パススルー・セッション』

パススルー・セッション

パススルー と呼ばれる特別なモードを使用すると、 SQL ステートメントを直接、データ・ソースにサブミットすることができます。この場合 SQL ステートメントは、データ・ソースで使用されている SQL ダイアレクトを使用してサブミットします。パススルー・セッションは、 DB2® SQL/API ではできない操作を実行したい場合に使用してください。たとえば、プロシージャーを作成する、索引を作成する、またはデータ・ソースに固有のダイアレクトで照会を実行する場合に、パススルー・セッションを使用します。

現在、パススルーをサポートするデータ・ソースは、 SQL を使用したパススルーをサポートします。将来的には、データ・ソースが SQL 以外のデータ・ソース言語を使用するパススルーをサポートすることもありえます。

同様に、パススルー・セッションを使用して、SQL がサポートしていないアクション (たとえば、ある種の管理用タスク) を実行することもできます。ただし、パススルー・セッションを使用してすべての管理用タスクを実行することはできません。

たとえば、データ・ソース側で表を作成したりドロップすることはできますが、リモート・データベースを開始または停止することはできません。

パススルー・セッションでは、静的 SQL と動的 SQL の両方を使用することができます。

フェデレーテッド・サーバーは、パススルー・セッションを管理するために次の SQL ステートメントを提供しています。

SET PASSTHRU
: パススルー・セッションを開く。別の SET PASSTHRU ステートメントを発行して新しいパススルー・セッションを開始すると、現行のパススルー・セッションは終了します。

SET PASSTHRU RESET
: 現行のパススルー・セッションを終了する。

GRANT (Server Privileges)
: ユーザー、グループ、許可 ID のリスト、または PUBLIC に対して、特定のデータ・ソースにパススルー・セッションを開始する権限を付与する。

REVOKE (Server Privileges)
: パススルー・セッションを開始する権限を取り消す。

パススルー・セッションには以下の制約事項があります。

- データ・ソースの SQL ダイアレクトまたは言語コマンドを使用しなければならず、 DB2 SQL ダイアレクトは使用できません。その結果、ニックネームを照会するのではなく、データ・ソースのオブジェクトを直接照会することになります。
- パススルー・セッションで UPDATE または DELETE 操作を実行する場合、WHERE CURRENT OF CURSOR 条件は使用できません。
- パススルー・セッションでは LOB はサポートされません。

関連概念:

- 49 ページの『ラッパーおよびラッパー・モジュール』
- 「フェデレーテッド・システム・ガイド」の『パススルーによりデータ・ソースを直接照会する』

ラッパーおよびラッパー・モジュール

ラッパー とは、フェデレーテッド・サーバーがデータ・ソースとやりとりするためのメカニズムです。フェデレーテッド・サーバーは、ライブラリーに保管されたルーチン (ラッパー・モジュール と呼ばれる) を使用してラッパーをインプリメントします。これらのルーチンは、データ・ソースに接続し、そこからデータを繰り返し検索するための接続といった操作を、フェデレーテッド・サーバーが実行できるようにします。通常、DB2® フェデレーテッド・インスタンスの所有者は、CREATE WRAPPER ステートメントを使用して、ラッパーをフェデレーテッド・データベースに登録します。 DB2_FENCED ラッパー・オプションを使用して、fenced またはトラステッドのどちらかでラッパーを登録することができます。

アクセスするデータ・ソースのタイプごとに、ラッパーを 1 つ作成します。たとえば、3 つの DB2 for z/OS™ データベース表、 1 つの DB2 for iSeries™ の表、2 つの Informix® の表、および 1 つの Informix ビューにアクセスしたいとします。作成する必要があるのは、DB2 データ・ソース・オブジェクト用に 1 つのラッパーと、 Informix データ・ソース・オブジェクト用に 1 つのラッパーです。フェデレーテッド・データベースにこれらのラッパーが登録されると、それらのラッパーを使用して、これらのデータ・ソースのその他のオブジェクトにアクセスすることができます。たとえば、DRDA® ラッパーをすべての DB2 ファミリーのデータ・ソース・オブジェクト (DB2 for Linux、DB2 for UNIX®、および DB2 for Windows®、DB2 for z/OS and OS/390®、DB2 for iSeries、および DB2 Server for VM and VSE) に使用することができます。

サーバー定義とニックネームを使用して、各データ・ソース・オブジェクトを特定して識別 (名前、ロケーションなど) します。

ラッパーは多くの作業を行います。 そのいくつかは次のようなものです。

- データ・ソースに接続します。ラッパーは、データ・ソースの標準の接続 API を使用します。
- データ・ソースに照会をサブミットします。
  - SQL をサポートするデータ・ソースの場合、照会は SQL でサブミットされます。
  - SQL をサポートしないデータ・ソースの場合、照会は、ソースに固有の照会言語に、または一連のソース API 呼び出しに変換されます。
- データ・ソースから結果セットを受信します。ラッパーは、データ・ソースの標準 API を使用して、結果セットを受信します。
- データ・ソースのデフォルトのデータ型マッピングについてのフェデレーテッド・サーバーの照会に応答します。ラッパーには、データ・ソース・オブジェクトにニックネームを作成する時に使用される、デフォルトのタイプ・マッピングが入っています。リレーショナル・ラッパーの場合、ユーザーが作成するデータ型マッピングは、デフォルトのデータ型マッピングをオーバーライドします。ユーザー定義のデータ型マッピングは、グローバル・カタログに保管されます。
- データ・ソースのデフォルトの関数マッピングについてのフェデレーテッド・サーバーの照会に応答します。ラッパーには、DB2 関数がデータ・ソースの関数と対応付けられるかどうか、またどのように関数が対応付けられるかを、フェデレーテッド・サーバーが判断する際に必要となる情報が入っています。この情報は、データ・ソースが照会操作を実行できるかどうかを判断するために、 SQL コンパイラーにより使用されます。リレーショナル・ラッパーの場合、ユーザーが作成する関数マッピングは、デフォルトの関数タイプ・マッピングをオーバーライドします。ユーザー定義の関数マッピングは、グローバル・カタログに保管されます。

ラッパー・オプション は、ラッパーを構成するためと、 DB2 Information Integrator がどのようにラッパーを使用するかを定義するために使用されます。

関連概念:

- 52 ページの『サーバー定義およびサーバー・オプション』

関連タスク:

- 「*IBM DB2 Information Integrator ラッパー開発者向けガイド*」の『トラステッドおよび fenced モードのプロセス環境』
- 「フェデレーテッド・システム・ガイド」の『ラッパーの変更』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『ALTER WRAPPER ステートメント』
- 700 ページの『フェデレーテッド・システムのラッパー・オプション』
- 51 ページの『デフォルトのラッパー名』

デフォルトのラッパー名

サポート対象のデータ・ソースごとに、ラッパーがあります。いくつかのラッパーにはデフォルト名があります。デフォルト名を使用してラッパーを作成すると、フェデレーテッド・サーバーはそのラッパーに関連するデータ・ソース・ライブラリーを自動的に選択します。

表 2. データ・ソースごとのデフォルト・ラッパー名

データ・ソース	デフォルトのラッパー名
DB2 Universal Database™ for Linux、UNIX、および Windows®	DRDA
DB2 Universal Database for z/OS and OS/390®	DRDA
DB2 Universal Database for iSeries	DRDA
DB2 Server for VM および DB2 Server for VSE	DRDA
Informix	INFORMIX
Microsoft® SQL Server	MSSQLODBC3
ODBC	ODBC
OLE DB	OLEDB
Oracle	NET8
Sybase	CTLIB
Teradata	TERADATA
BLAST	なし
BioRS	なし
Documentum	なし
Entrez	なし
Extended Search	なし
HMMER	なし
Microsoft Excel	なし
表構造ファイル	なし
Web サービス	なし
WebSphere Business Integration	なし
XML	なし

関連概念:

- 49 ページの『ラッパーおよびラッパー・モジュール』

サーバー定義およびサーバー・オプション

データ・ソース用のラッパーを作成した後、フェデレーテッド・インスタンスの所有者はデータ・ソースをフェデレーテッド・データベースに定義します。インスタンス所有者は、データ・ソースを識別するための名前を指定し、またデータ・ソースに関するその他の情報も指定します。この情報は次のような内容からなります。

- データ・ソースのタイプとバージョン。
- データ・ソースのデータベース名 (RDBMS のみ)。
- データ・ソースに固有のメタデータ。

たとえば、DB2® ファミリーのデータ・ソースは複数のデータベースを持つことができます。定義には、どのデータベースにフェデレーテッド・サーバーが接続できるかを指定する必要があります。対照的に、Oracle データ・ソースが持つデータベースは 1 つなので、フェデレーテッド・サーバーは名前を知らなくてもそのデータベースに接続することができます。 Oracle データ・ソースの場合は、フェデレーテッド・サーバー定義にデータベース名は組み込まれません。

インスタンス所有者がフェデレーテッド・サーバーに提供する、名前およびその他の情報をまとめてサーバー定義 と呼びます。データ・ソースはデータに対する要求に応答し、それ自体がサーバーです。

サーバー定義は、CREATE SERVER および ALTER SERVER ステートメントを使用して作成および変更します。

サーバー定義内の情報のあるものは、サーバー・オプション として保管されます。サーバー定義の作成時に、サーバーに関して指定できるオプションを理解することが重要です。ラッパーを構成するサーバー・オプションもあれば、 DB2 Information Integrator がラッパーをどのように使用するかに影響するサーバー・オプションもあります。

サーバー・オプションを、データ・ソースへの次の接続に持ち越されるように設定することも、または 1 回の接続の間ずっと設定したままにすることもできます。

関連概念:

- 52 ページの『ユーザー・マッピング』

関連資料:

- 682 ページの『フェデレーテッド・システムのサーバー・オプション』

ユーザー・マッピング

フェデレーテッド・サーバーがデータ・ソースに要求をプッシュダウンする必要がある場合、サーバーは最初にデータ・ソースに接続を確立する必要があります。

大半のデータ・ソースの場合、フェデレーテッド・サーバーは有効なユーザー ID とパスワードを使用して、そのデータ・ソースに接続を行います。データ・ソースへの接続にユーザー ID とパスワードが必要な場合、フェデレーテッド・サーバーの許可 ID と、データ・ソースのユーザー ID およびパスワードの間の関連を定義

することができます。この関連の作成は、フェデレーテッド・システムを使用して分散要求を送信するユーザー ID ごとに行うことができます。この関連付けは、ユーザー・マッピング と呼ばれます。

フェデレーテッド・データベースへの接続に使用するユーザー ID とパスワードが、リモート・データ・ソースへのアクセスに使用するものと同じであれば、ユーザー・マッピングを作成する必要のない場合もあります。

関連タスク:

- 「*IBM DB2 Information Integrator データ・ソース構成ガイド*」の『データ・ソースのユーザー・マッピングの登録』

ニックネームとデータ・ソース・オブジェクト

サーバー定義およびユーザー・マッピングを作成した後、フェデレーテッド・インスタンスの所有者はニックネームを作成します。ニックネーム とは、アクセスしたいデータ・ソース側にあるオブジェクトを参照するために使用される ID です。ニックネームが示すオブジェクトは、データ・ソース・オブジェクト と呼ばれます。

別名が代替名であるのとは異なり、ニックネームは、データ・ソース・オブジェクトの代替名ではありません。ニックネームは、フェデレーテッド・サーバーがこれらのオブジェクトを参照するためのポインターです。 ニックネームは通常、特定のニックネーム列オプションとニックネーム・オプションを指定した CREATE NICKNAME ステートメントを使用して定義します。

エンド・ユーザーまたはクライアント・アプリケーションが分散要求をフェデレーテッド・サーバーにサブミットする場合、その要求でデータ・ソースを指定する必要はありません。つまり要求では、データ・ソース・オブジェクトはそのニックネームで参照されます。ニックネームはデータ・ソース側の特定のオブジェクトにマップされます。このマッピング (対応付け) によって、ニックネームをデータ・ソース名で修飾する必要がなくなります。 エンド・ユーザーまたはクライアント・アプリケーションは、データ・ソース・オブジェクトのロケーションを意識する必要はありません。

たとえば、 *NFX1.PERSON* という Informix® データベース表を表すニックネーム *DEPT* を定義したとします。 SELECT * FROM *DEPT* というステートメントをフェデレーテッド・サーバーから使用できます。 しかし、フェデレーテッド・サーバー上に *NFX1.PERSON* というローカル表がないかぎり、フェデレーテッド・サーバーから、ステートメント SELECT * FROM *NFX1.PERSON* を使用することはできません (パススルー・セッションは除く)。

データ・ソース・オブジェクトにニックネームを作成すると、オブジェクトについてのメタデータがグローバル・カタログに追加されます。照会オプティマイザーは、このメタデータおよび、ラッパー内の情報を使用して、データ・ソース・オブジェクトへのアクセスを容易にします。たとえば、索引を持つ表にニックネームを作成すると、グローバル・カタログにはその索引についての情報が入ります。ラッパーには、DB2® のデータ型とデータ・ソースのデータ型間のマッピングが入っています。

現在、一部の DB2 UDB ユーティリティー操作は、ニックネームで実行することはできません。

ニックネーム内に相互ロードするのに、Cross Loader ユーティリティーを使用することはできません。

関連概念:

- 55 ページの『ニックネーム列オプション』

関連資料:

- 675 ページの『フェデレーテッド・システムのニックネーム列』
- 「フェデレーテッド・システム・ガイド」の『フェデレーテッド・システムのニックネーム・オプション』
- 54 ページの『有効なデータ・ソース・オブジェクト』

有効なデータ・ソース・オブジェクト

アクセスしたいデータ・ソースのオブジェクトは、ニックネームで特定されます。以下の表は、フェデレーテッド・システムでニックネームを作成できるオブジェクトのタイプを一覧で示しています。

表 3. 有効なデータ・ソース・オブジェクト

データ・ソース	有効なオブジェクト
DB2 for Linux、DB2 for UNIX、および DB2 for Windows	ニックネーム、マテリアライズ照会表、表、ビュー
DB2 for z/OS and OS/390	表、ビュー
DB2 for iSeries	表、ビュー
DB2 for VM and VSE	表、ビュー
Informix	表、ビュー、同義語
Microsoft SQL Server	表、ビュー
ODBC	表、ビュー
Oracle	表、ビュー、同義語
Sybase	表、ビュー
Teradata	表、ビュー
BLAST	BLAST 検索アルゴリズム用に索引が付けられた FASTA ファイル
BioRS	BioRS databanks
Documentum	Documentum Docbase 内のオブジェクトおよび登録された表
Entrez	Entrez データベース
Extended Search	Lotus Notes データベース、Microsoft Access、Microsoft Index Server、Web 検索エンジン、および LDAP ディレクトリーなどのデータ・ソースのファイル

表 3. 有効なデータ・ソース・オブジェクト (続き)

データ・ソース	有効なオブジェクト
HMMER	HMMER の hmmpfam または hmmsearch プログラムで検索できる HMM データベース・ファイル (PFAM などの Hierarchical Markov Model のライブラリー)
Microsoft Excel	.xls ファイル (ワークブック内の最初のシートだけがアクセスされる)
表構造ファイル	特定のフォーマットに適合したテキスト・ファイル
Websphere Business Integration アダプター	SAP 内の BAPI にマップされる Websphere Business Integration ビジネス・オブジェクト、 Siebel 内のビジネス・コンポーネント、および PeopleSoft 内のコンポーネント・インターフェース
Web サービス	Web サービス記述言語ファイル内の操作
XML タグ・ファイル	XML 文書内の項目のセット

関連概念:

- 53 ページの『ニックネームとデータ・ソース・オブジェクト』
- 55 ページの『ニックネーム列オプション』

ニックネーム列オプション

ニックネームが付けられたオブジェクトについての追加のメタデータ情報を、グローバル・カタログに入れることができます。このメタデータは、データ・ソース・オブジェクトの特定の列の値を記述します。このメタデータを、*ニックネーム列オプション* と呼ばれるパラメーターに割り当てます。ニックネーム列オプションは、この列内のデータを通常とは異なるやり方で扱うようにラッパーに指示します。SQL コンパイラーと照会オプティマイザーは、メタデータを使用して、データにアクセスするためのよりよいプランを作成します。

ニックネーム列オプションは、ラッパーにその他の情報を提供するためにも使用されます。たとえば XML データ・ソースの場合、ニックネーム列オプションは、ラッパーが XML 文書から列を解析する時に使用する XPath 式をラッパーに指示するために使用されます。

フェデレーションの場合、ニックネームによって参照されるデータ・ソース・オブジェクトは、 DB2® サーバーではローカル DB2 表であるものとして扱われます。したがって、ニックネームを作成する任意のデータ・ソース・オブジェクトに対して、ニックネーム列オプションをセットすることができます。一部のニックネーム列オプションは特定のタイプのデータ・ソース用に作られていて、そのようなデータ・ソースにのみ適用することができます。

仮に、データ・ソースが、フェデレーテッド・データベースの照合シーケンスとは異なる照合シーケンスを持つとします。フェデレーテッド・サーバーは通常、データ・ソース側で文字データの入った列をソートすることはありません。データはフェデレーテッド・データベースに戻され、ローカルでソートが行われます。しかし

ここで、列が文字データ型 (CHAR または VARCHAR) であり、数字 ('0'、'1'、...、'9') だけが入っているとします。このことは、NUMERIC_STRING ニックネーム列オプションを 'Y' にすれば示すことができます。これにより、DB2 照会オプティマイザーは、データ・ソース側でソートを実行するというオプションが得られます。ソートをリモート側で実行できれば、データをフェデレーテッド・サーバーにポーティングし、ソートをローカルで実行するというオーバーヘッドが避けられます。

ALTER NICKNAME ステートメントを使って、リレーショナル・ニックネーム用のニックネーム列オプションを定義することができます。 CREATE NICKNAME および ALTER NICKNAME ステートメントを使って、非リレーショナル・ニックネーム用のニックネーム列オプションを定義することができます。

関連概念:

- 56 ページの『データ型マッピング』

関連タスク:

- 「フェデレーテッド・システム・ガイド」の『ニックネームを使用した操作』

関連資料:

- 675 ページの『フェデレーテッド・システムのニックネーム列』

データ型マッピング

フェデレーテッド・サーバーがデータ・ソースからデータを検索するには、データ・ソース側のデータ型が、対応する DB2® のデータ型と対応付けられて (マッピングされて) いなければなりません。以下に、デフォルトのデータ型マッピングの例をいくつか示します。

- Oracle タイプ FLOAT は、DB2 タイプ DOUBLE にマップされます。
- Oracle タイプ DATE は、DB2 タイプ TIMESTAMP にマップされます。
- DB2 for z/OS™ のタイプ DATE は、DB2 タイプ DATE にマップされます。

ほとんどのデータ・ソースの場合、ラッパー内にデフォルトのタイプ・マッピングがあります。 DB2 データ・ソース用のデフォルトのタイプ・マッピングは、DRDA® ラッパー内にあります。 Informix® 用のデフォルトのタイプ・マッピングは INFORMIX ラッパーにあり、他も同様です。

一部の非リレーショナルのデータ・ソースの場合、 CREATE NICKNAME ステートメントでデータ型情報を指定する必要があります。ニックネームの作成時に、データ・ソース・オブジェクトの列ごとに、対応する DB2 for Linux、 DB2 for UNIX®、および DB2 for Windows® のデータ型を指定する必要があります。それぞれの列は、データ・ソース・オブジェクト内の特定のフィールドまたは列にマップされている必要があります。

リレーショナル・データ・ソースの場合、デフォルトのデータ型マッピングをオーバーライドできます。たとえば、デフォルトでは Informix INTEGER データ型は DB2 INTEGER データ型にマップされます。このデフォルト・マッピングをオーバーライドして、Informix の INTEGER データ型を DB2 DECIMAL(10,0) データ型にマップしてもかまいません。

新規のタイプ・マッピングの作成や、デフォルトのタイプ・マッピングの修正は、ニックネームの作成の前に行わなければなりません。そうしないと、タイプ・マッピングより前に作成されたニックネームには、新しいマッピングが反映されません。

関連概念:

- 「フェデレーテッド・システム・ガイド」の『フェデレーテッド・サーバーでのデータ・タイプ・マッピング』

関数マッピング

フェデレーテッド・サーバーがデータ・ソース関数を認識するには、その関数が既存の DB2® for Linux、DB2 for UNIX®、および DB2 for Windows® 内の既存の対応関数にマップされていなければなりません。 DB2 Information Integrator は、既存の組み込みデータ・ソース関数とそれに対応する組み込み DB2 Information Integrator 関数間のデフォルトのマッピングを備えています。ほとんどのデータ・ソースの場合、ラッパー内にデフォルトの関数マッピングがあります。 DB2 for z/OS™ および OS/390® の関数へのデフォルトの関数マッピングは、 DRDA® ラッパー内にあります。 Sybase 関数へのデフォルトの関数マッピングは CTLIB ラッパー内にある、というように続きます。

リレーショナル・データ・ソースの場合、フェデレーテッド・サーバーが認識しないデータ・ソース関数を使用するために、関数マッピングを作成することができます。作成するマッピングは、データ・ソース関数と、フェデレーテッド・データベースにある対応する DB2 関数を結び付けるものです。通常、関数マッピングが使用されるのは、新しい組み込み関数または新しいユーザー定義関数がデータ・ソース側で使用可能になった時です。さらに、DB2 側に対応する関数がない場合にも関数マッピングが使用されます。この場合、関数テンプレートも作成しなければなりません。

関連概念:

- 「フェデレーテッド・システム・ガイド」の『フェデレーテッド・システムでの関数マッピング』
- 57 ページの『索引の指定』

索引の指定

データ・ソース表にニックネームを作成すると、データ・ソース表が持つすべての索引についての情報がグローバル・カタログに追加されます。照会オプティマイザーはこの情報を使用して、分散要求の処理を速くします。データ・ソース索引についてのカタログ情報は、メタデータの集まりであり、*索引の指定* と呼ばれます。フェデレーテッド・サーバーは、次のものにニックネームが作成された場合、「索引の指定」を作成しません。

- 索引を持たない表。
- ビュー。これは通常、リモート・カタログに保管される索引情報はありません。
- フェデレーテッド・サーバーが索引情報を入手できるリモート・カタログを持たない、データ・ソース・オブジェクト。

あるいは、ある表に、ニックネームの作成時にあった索引に加えて、新しい索引が作られたとします。索引情報は、ニックネームの作成時にグローバル・カタログに入るため、フェデレーテッド・サーバーは新しい索引を認知しません。同様に、ビューにニックネームを作成すると、ビューの基になる表 (およびその索引) は、フェデレーテッド・サーバーで認知されません。このような場合に、必要な索引情報をグローバル・カタログに入れることができます。索引を持たない表に「索引の指定」を作成することができます。「索引の指定」は、データを速く見つけるには表内のどの列をサーチすべきかを照会オプティマイザーに示します。

関連概念:

- 「フェデレーテッド・システム・ガイド」の『フェデレーテッド・システムでの索引の指定』

照合シーケンス

データベース内で文字データがソートされる順序は、データベースで定義されているデータの構造と照合シーケンスによって決まります。

データベース内のデータはすべて大文字であって、数字や特殊文字は入っていないと仮定します。データ・ソースまたはフェデレーテッド・データベースのどちらでデータがソートされても、データのソートは同じ出力を生じるはずです。各データベースで使用する照合シーケンスによってソート結果が影響を受けてはなりません。同様に、データベース内のデータがすべて小文字または数字である場合も、ソートが実際に実行される場所に関係なく、データのソートでは同じ結果を生じるはずです。

データが以下の構造のいずれかで構成されているとします。

- 文字と数字の組み合わせ。
- 大文字と小文字の両方。
- @、#、€ などの特殊文字。

このようなデータをソートした場合に、使用される照合シーケンスがフェデレーテッド・データベースとデータ・ソースで異なっていると、別々の出力が生成されます。

一般用語としての*照合シーケンス* とは、ある特定の文字を別の文字より上位、下位、または同列のいずれにソートするかを決めるために定義された文字データ順序のことです。

照合シーケンスでソート順序が決定される方法

照合シーケンスによって、コード化文字セット内の文字のソート順序が決められます。*文字セット* とは、コンピューター・システムやプログラム言語で使用される文字の集まりのことです。*コード化* 文字セットでは、 0 から 255 (またはそれに相当する 16 進数) までの範囲の別々の数字に各文字が割り当てられます。そのような数字をコード・ポイント と呼び、セット内の文字への数字の割り当てをまとめて コード・ページ と呼びます。

コード・ポイントは、文字に割り当てられる以外に、ソート順序における文字位置にマップすることもできます。さらに、技術用語としての照合シーケンスとは、セ

ット内の文字のソート順序位置に対する文字セットのコード・ポイントのマッピングの集まりのことです。文字位置は番号で表されますが、その番号を文字の *重み* と呼びます。 *ID シーケンス* という最も単純な照合シーケンスでは、重みはコード・ポイントと同じになります。

データベース ALPHA は、EBCDIC コード・ページのデフォルト照合シーケンスを使用していて、データベース BETA は、ASCII コード・ページのデフォルト照合シーケンスを使用していると仮定します。この 2 つのデータベースでの文字ストリングのソート順序は、以下の例に示されているとおり異なります。

```
SELECT.....
  ORDER BY COL2

EBCDIC-Based Sort           ASCII-Based Sort

COL2                      COL2
----                         ----
V1G                          7AB
Y2W                          V1G
7AB                          Y2W
```

同様に、データベースでの文字比較は、そのデータベースで定義されている照合シーケンスによって異なります。この例では、データベース ALPHA は EBCDIC コード・ページのデフォルト照合シーケンスを使用しています。またデータベース BETA は、ASCII コード・ページのデフォルト照合シーケンスを使用しています。この 2 つのデータベースでの文字比較では、以下の例に示されているとおり、それぞれ異なる結果を生じます。

```
SELECT.....
  WHERE COL2 > 'TT3'

EBCDIC-Based Results     ASCII-Based Results

COL2                     COL2
----                     ----
TW4                      TW4
X82                      X82
39G
```

照会の最適化のためのローカル照合シーケンスの設定

管理者は、データ・ソースの照合シーケンスに一致する特定の照合シーケンスをもったフェデレーテッド・データベースを作成することができます。この場合、各データ・ソース・サーバー定義ごとに、COLLATING_SEQUENCE サーバー・オプションを Y に設定します。この設定によって、フェデレーテッド・データベースとデータ・ソースの照合シーケンスの一致がフェデレーテッド・データベースに指示されます。

フェデレーテッド・データベースの照合シーケンスは、CREATE DATABASE API の一環として設定します。この API では、以下のシーケンスのうちの 1 つを指定することができます。

- ID シーケンス。
- システム・シーケンス (データベースをサポートするオペレーティング・システムで使用されるシーケンス)。
- カスタマイズ・シーケンス (DB2 UDB 提供またはユーザー自身が定義する事前定義のシーケンス)。

データ・ソースは DB2 for z/OS and OS/390 であると仮定します。 ORDER BY 文節に定義されたソートが、 EBCDIC コード・ページをベースとする照合シーケンスでインプリメントされます。 ORDER BY 文節に従ってソートされた DB2 for z/OS and OS/390 データを取り出すには、該当する EBCDIC コード・ページをベースとする事前定義の照合シーケンスを使うようにフェデレーテッド・データベースを構成します。

関連概念:

- 「フェデレーテッド・システム・ガイド」の『プッシュダウンの可能性に影響を与えるサーバー特性』
- 「*IBM DB2 Information Integrator データ・ソース構成ガイド*」の『フェデレーテッド・システムの照合シーケンス』

関連タスク:

- 「*IBM DB2 Information Integrator データ・ソース構成ガイド*」の『フェデレーテッド・データベースの作成』

関連資料:

- 「*管理ガイド: プランニング*」の『各国語バージョン』
- 「*IBM DB2 Information Integrator データ・ソース構成ガイド*」の『フェデレーテッド・データベースの各国語に関する考慮事項』

第 2 章 言語エレメント

この章では、多くの SQL ステートメントに共通の言語エレメントについて説明します。

文字

SQL 言語のキーワードと演算子で使う基本的な記号は、すべての IBM 文字セットの一部である 1 バイト文字です。言語の文字は、英字、数字、または特殊文字に分類されます。

文字 とは、26 個の大文字 (A〜Z) および 26 個の小文字 (a〜z)、さらに 3 個の文字 ($、#、および @) のいずれかです。これらの特殊文字は、ホスト・データベース製品との互換性を保つために備えられています。たとえば、コード・ページ 850 では、$ は X'24'、# は X'23'、 @ は X'40' にあります。英字には、拡張文字セットのアルファベット文字も入っています。拡張文字セットには、追加のアルファベット文字が入っています。たとえば、分音符号 (´ は分音符号の一例です) の付いたアルファベット文字です。使用可能な文字は、使用するコード・ページによって異なります。

数字 は、0 から 9 のいずれかの文字です。

特殊文字 は、以下のいずれかの文字です。

文字	説明	文字	説明
	スペースまたはブランク	-	負符号
"	引用符または二重引用符	.	ピリオド
%	パーセント	/	スラッシュ
&	アンパーサンド	:	コロン
'	アポストロフィまたは単一引用符	;	セミコロン
(	左括弧	<	不等号 (より小さい)
)	右括弧	=	等号

文字	説明	文字	説明
*	アスタリスク	>	不等号 (より大きい)
+	正符号	?	疑問符
,	コンマ	_	下線
|	縦バー[1]	^	脱字記号
!	感嘆符	[	左大括弧
{	左中括弧	]	右大括弧
}	右中括弧		

[1] 縦バー (|) 文字を使用すると、 IBM リレーショナル製品どうしの間のコード・ポータビリティーが損なわれることがあります。 || 演算子ではなく、CONCAT 演算子を使用してください。

マルチバイト文字はすべて文字として扱われます。ただし、特殊文字である 2 バイト・ブランクは例外です。

トークン

トークンは、SQL の基本構成単位です。*トークン* は、1 つまたは複数の一連の文字です。 ブランク文字を使用してもかまわないストリング定数または、区切り ID の場合を除いて、トークンにブランク文字を使用することはできません。

トークンは、通常トークンと区切り文字に分類されます。

- *通常トークン* とは、数値定数、通常 ID、ホスト ID、またはキーワードです。

  例

  ```
  1       .1       +2       SELECT       E       3
  ```

- *区切りトークン* とは、ストリング定数、区切り ID、演算子記号、または構文図に示される特殊文字です。パラメーター・マーカーとして機能する場合は疑問符も区切りトークンです。

  例

  ```
  ,       'string'       "fld1"       =       .
  ```

スペース: スペースは、1 つまたは複数の一連のブランク文字です。ストリング定数と区切り ID 以外のトークンには、スペースを使用することはできません。トークンの後にはスペースを続けることができます。すべての通常トークンの後には、スペースか、または構文で許されているなら区切りトークンを付ける必要があります。

コメント: 静的 SQL ステートメントには、ホスト言語のコメントまたは SQL コメントが入っている場合があります。どちらのタイプのコメントも、スペースを指定できるロケーションであればどこにでも指定可能ですが、区切りトークンの中または EXEC と SQL キーワードの間には指定できません。 SQL のコメントは、2 つの連続するハイフン (--) で始まり、行末で終わります。

大文字小文字の区別: どのトークンにも小文字を使用することができますが、通常トークンでの小文字は大文字に変換されます。ただし、ID の大文字と小文字を区別する C 言語のホスト変数は例外です。区切りトークンが大文字に変換されることはありません。したがって、次のステートメントは、

```
select * from EMPLOYEE where lastname = 'Smith';
```

大文字に変換した後は、以下のステートメントと同等になります。

```
SELECT * FROM EMPLOYEE WHERE LASTNAME = 'Smith';
```

マルチバイトの英文字は大文字変換されません。 1 バイト文字 (a から z) は大文字に変換されます。

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『SQL ステートメントの呼び出し方法』
- 「*SQL リファレンス 第 2 巻*」の『PREPARE ステートメント』

ID

ID とは、名前の形成に使用されるトークンです。SQL ステートメントの ID は、SQL ID かホスト ID のいずれかです。

- SQL ID

  SQL ID には、通常 ID と区切り ID の 2 つのタイプがあります。

  – *通常 ID* は、英字の後にゼロ個またはそれ以上の文字が続いている ID です。各文字は、英大文字、数字、または下線文字です。通常 ID は予約語と同一であってはなりません。

    例

    ```
    WKLYSAL     WKLY_SAL
    ```

  – *区切り ID* は、 1 文字以上の文字を引用符で囲んだ ID です。区切り ID の中で 1 つの引用符を表す場合には、 2 つの連続した引用符を使用します。この方法で、ID に小文字を使用することができます。

    例

    ```
    "WKLY_SAL"     "WKLY SAL"     "UNION"     "wkly_sal"
    ```

  2 バイトのコード・ページで生成されているのに、使用するのはマルチバイトのコードのアプリケーションやデータベースであるという ID の文字変換の場合、次のような特別な配慮が必要な場合があります。つまり、このような ID の場合、文字変換後に ID の長さ制限を超えてしまう可能性があります。

- ホスト ID

  ホスト *ID* は、ホスト・プログラムで宣言されている名前です。ホスト ID の規則は、ホスト言語の規則に従います。ホスト ID は長さが 255 文字以下でなければならず、また、 SQL または DB2 で始まっていてはなりません (大文字小文字のどちらでも)。

命名規則と暗黙オブジェクト名の修飾

オブジェクトを命名する規則は、オブジェクト・タイプによって異なります。データベース・オブジェクト名は、1 つの ID で構成されていても、または 2 つの ID で構成されるスキーマ修飾オブジェクトでもかまいません。スキーマ修飾オブジェクト名は、スキーマ名なしで指定することができますが、その場合、スキーマ名は暗黙名になります。

動的 SQL ステートメントでは、スキーマ修飾オブジェクト名は、修飾子のないオブジェクト名参照の修飾子として CURRENT SCHEMA 特殊レジスター値を暗黙的に使用します。デフォルトでは、この値は現行の許可 ID に設定されています。バインド、定義、または起動の動作を備えたパッケージ内に動的 SQL ステートメントが入っている場合、 CURRENT SCHEMA 特殊レジスターは修飾には使用されません。バインド動作のパッケージでは、非修飾オブジェクト参照子の暗黙修飾の値として、パッケージのデフォルト修飾子が使用されます。定義動作のパッケージでは、ルーチン定義者の許可 ID が、そのルーチン内の非修飾オブジェクト参照子の暗黙修飾の値として使用されます。起動動作のパッケージでは、ルーチンの起動時に有効になっているステートメント許可 ID が、そのルーチン内の動的 SQL ステ

ートメント内の非修飾オブジェクト参照子の暗黙修飾の値として使用されます。詳しくは、 71 ページの『ランタイムにおける動的 SQL の特性』を参照してください。

静的 SQL ステートメントでは、 QUALIFIER プリコンパイル/BIND オプションにより、非修飾のデータベース・オブジェクト名の修飾子が暗黙指定されます。デフォルトでは、この値はパッケージの許可 ID に設定されています。

以下のオブジェクト名は、SQL プロシージャーのコンテキスト内で使用される場合、それらの名前が区切られていたとしても、通常 ID で許可されている文字しか使用できません。

- condition-name
- label
- parameter-name
- procedure-name
- SQL-variable-name
- statement-name

構文図では、異なる種類の名前には異なる用語を使用しています。以下のリストでそのような用語を定義します。

alias-name
別名を指定するスキーマ修飾名。

attribute-name
構造化データ型の属性を指定する ID。

authorization-name
ユーザーまたはグループを指定する ID。

- 有効な文字は、A から Z、a から z、0 から 9、#、@、 $、および _ です。
- 名前は、文字 'SYS'、'IBM'、または 'SQL' で始めることはできません。
- 名前は、ADMINS、GUESTS、LOCAL、PUBLIC、または USERS であってはなりません。
- 区切り許可 ID に小文字を使用することはできません。

bufferpool-name
バッファー・プールを指定する ID。

column-name
表またはビューの列を指定する修飾子付きまたは修飾子のない名前。修飾子は、表名、ビュー名、ニックネーム、または相関名です。

condition-name
SQL プロシージャーの条件を指定する ID。

constraint-name
参照制約、主キー制約、ユニーク制約、または表チェック制約を指定する ID。

correlation-name
結果表を指定する ID。

cursor-name
SQL カーソルを指定する ID。ホストとの互換性を保つため、この名前にハイフン文字を使用することもできます。

data-source-name
データ・ソースを指定する ID。この ID は、3 つの部分に分かれたリモート・オブジェクト名の最初の部分を成します。

descriptor-name
コロンの後に、SQL 記述子域 (SQLDA) を指定するホスト ID を付けたもの。ホスト ID の詳細は、79 ページの『ホスト変数の参照』を参照してください。記述子名には標識変数は使用されません。

distinct-type-name
特殊タイプを指定する修飾子付きまたは修飾子のない名前。 SQL ステートメントにおける修飾子のない特殊タイプ名は、コンテキストに応じてデータベース・マネージャーによって暗黙のうちに修飾されます。

event-monitor-name
イベント・モニターを指定する ID。

function-mapping-name
関数マッピングを指定する ID。

function-name
関数を指定する、修飾子付きまたは修飾子のない名前。 SQL ステートメントにおける修飾子のない関数名は、コンテキストに応じてデータベース・マネージャーによって暗黙のうちに修飾されます。

group-name
構造化タイプ用に定義した変換グループを指定する、修飾子のない ID。

host-variable
ホスト変数を指定するトークンを連結したもの。79 ページの『ホスト変数の参照』に説明されているように、ホスト変数内には少なくとも 1 つのホスト ID が使用されます。

index-name
索引または索引指定を指定するスキーマによって修飾された名前。

label
SQL プロシージャーのラベルを指定する ID。

method-name
メソッドを指定する ID。メソッドのスキーマ・コンテキストは、そのメソッドのサブジェクト・タイプ (または、サブジェクト・タイプのスーパータイプ) のスキーマによって決まります。

nickname
フェデレーテッド・サーバーが表またはビューに参照することを指定する、スキーマによって修飾された名前。

db-partition-group-name
データベース・パーティション・グループを指定する ID。

package-name
パッケージを指定するスキーマ修飾名。空ストリングではないパッケージ ID がパッケージに付いている場合、 `schema-id.package-id.version-id` のフォームのバージョン ID もそのパッケージ名の末尾に付いている必要があります。

parameter-name
プロシージャー、ユーザー定義関数、メソッド、または索引拡張機能で参照できるパラメーターを指定する ID。

procedure-name
プロシージャーを指定する修飾子付きまたは修飾子のない名前。 SQL ステートメントにおける修飾子のないプロシージャー名は、コンテキストに応じてデータベース・マネージャーによって暗黙のうちに修飾されます。

remote-authorization-name
データ・ソースのユーザーを指定する ID。許可名に関する規則は、データ・ソースごとに異なります。

remote-function-name
データ・ソース・データベースに登録されている関数を指定する名前。

remote-object-name
データ・ソース表またはビューを指定するとともに、その表またはビューが置かれているデータ・ソースを識別する 3 つの部分に分かれた名前。この名前は、データ・ソース名、リモート・スキーマ名、およびリモート表名の各部分で構成されます。

remote-schema-name
データ・ソース表またはビューが属するスキーマを指定する名前。この名前は、3 つの部分に分かれたリモート・オブジェクト名の 2 番目の部分を成します。

remote-table-name
データ・ソースにある表またはビューを指定する名前。この名前は、3 つの部分に分かれたリモート・オブジェクト名の 3 番目の部分を成します。

remote-type-name
データ・ソース・データベースがサポートするデータ型。組み込まれたタイプの場合は、長形式を使用しないでください (たとえば、 CHARACTER ではなく CHAR を使用してください)。

savepoint-name
セーブポイントを指定する ID。

schema-name
SQL オブジェクトを論理的にグループ化するための ID。オブジェクト名の修飾子として使用されるスキーマ名は、以下のものから暗黙的に決定されます。

- CURRENT SCHEMA 特殊レジスターの値
- QUALIFIER プリコンパイル/BIND オプションの値
- CURRENT PATH 特殊レジスターを使用する解決アルゴリズムに基づいて
- 同一の SQL ステートメントにある別のオブジェクトのスキーマ名に基づいて

混乱を避けるために、スキーマとして名前 SESSION を使わないようお勧めします。ただし、

宣言済みのグローバル一時表のスキーマは除きます (この場合は、スキーマ名 SESSION を使用する必要があります)。

server-name アプリケーション・サーバーを指定する ID。フェデレーテッド・システムでは、サーバー名によってデータ・ソースのローカル名も指定します。

specific-name ユニーク名を指定する、修飾子付きまたは修飾子のない名前。 SQL ステートメントにおける修飾子のないユニーク名は、コンテキストに応じてデータベース・マネージャーによって暗黙のうちに修飾されます。

SQL-variable-name SQL プロシージャー・ステートメントのローカル変数名。 SQL 変数名は、ホスト変数名が許可されている他の SQL ステートメントで使うこともできます。この名前に対しては、SQL 変数を宣言したコンパウンド・ステートメントのラベルで修飾できます。

statement-name 作成済み SQL ステートメントを指定する ID。

supertype-name タイプのスーパータイプを指定する修飾子付きまたは修飾子のない名前。 SQL ステートメントにおける修飾子のないスーパータイプ名は、コンテキストに応じてデータベース・マネージャーによって暗黙のうちに修飾されます。

table-name 表を指定するスキーマによって修飾された名前。

tablespace-name 表スペースを指定する ID。

trigger-name トリガーを指定するスキーマによって修飾された名前。

type-mapping-name データ型マッピングを指定する ID。

type-name 型を指定する修飾子付きまたは修飾子のない名前。 SQL ステートメントにおける修飾子のないタイプ名は、コンテキストに応じてデータベース・マネージャーによって暗黙のうちに修飾されます。

typed-table-name タイプ表を指定するスキーマによって修飾された名前。

typed-view-name タイプ・ビューを指定するスキーマによって修飾された名前。

view-name ビューを指定するスキーマによって修飾された名前。

wrapper-name ラッパーを指定する ID。

別名

表の別名は、表またはビューの代替名と見なすことができます。このため、SQL ステートメントの中で表またはビューは、名前か表別名のどちらかで参照できます。

別名は、表名またはビュー名を使用できるロケーションであればどこででも使用できます。別名は、オブジェクトが存在していなくても作成することができます (ただし、オブジェクトを参照するステートメントがコンパイルされる時点では存在している必要があります)。別名チェーンの中に循環参照または反復参照がない限り、他の別名を別名によって参照することができます。別名で参照できるのは、同じデータベース内の表、ビュー、別名だけです。 CREATE TABLE や CREATE VIEW ステートメントのような、新しい表名またはビュー名が指定されるのが当然のロケーションでは別名を使用できません。たとえば、 PERSONNEL という別名を作成した後で CREATE TABLE PERSONNEL... のような使い方をするとエラーが戻されます。

構文図や SQL ステートメントの説明では、別名によって表またはビューを参照するというオプションは明示的には示されません。

修飾子なしの新しい別名に、既存の表、ビュー、または別名と同じ完全修飾名を付けることはできません。

SQL ステートメントで別名を使用する効果は、テキスト置換の効果に似ています。別名は SQL ステートメントのコンパイル時には定義されている必要があり、ステートメントのコンパイル時には修飾子付きの基本表名またはビュー名に置き換えられます。たとえば、PBIRD.SALES が DSPN014.DIST4_SALES_148 の別名である場合に、コンパイル時には、

```
SELECT * FROM PBIRD.SALES
```

は、実際には次のようになります。

```
SELECT * FROM DSPN014.DIST4_SALES_148
```

フェデレーテッド・システムでは、上記のような使用法と制限は、表の別名だけでなく、ニックネームが表す別名にも適用されます。 したがって、ニックネームの代わりにニックネームの別名を SQL ステートメントで使用することも可能です。別名を参照するステートメントをコンパイルする前にニックネームを作成する場合、まだ存在していないニックネームに別名をあらかじめ作成しておくこともできます。あるニックネームの別名に、そのニックネームの他の別名を参照させることもできます。

他のリレーショナル・データベース管理システムのもとで実行するアプリケーションの構文を使えるようにするために、 CREATE ALIAS ステートメントと DROP ALIAS ステートメントにおいては、 ALIAS の代わりに SYNONYM を使用できるようになっています。

許可 ID と許可名

許可 ID とは、データベース・マネージャーとアプリケーション・プロセスとの間、またはデータベース・マネージャーとプログラム作成処理との間の接続が確立されるときに、データベース・マネージャーが獲得する文字ストリングのことです。これは、特権の集合を指定するものです。ユーザーやユーザー・グループを指す場合もありますが、この特性はデータベース・マネージャーからは制御されません。

許可 ID は、データベース・マネージャーにより以下の目的で使用されます。

- SQL ステートメントの許可検査
- QUALIFIER プリコンパイル/BIND オプションと CURRENT SCHEMA 特殊レジスターのデフォルト値。許可 ID は、デフォルトの CURRENT PATH 特殊レジスターと FUNCPATH プリコンパイル/BIND オプションにも入っています。

許可 ID はすべての SQL ステートメントに適用されます。静的 SQL ステートメントに適用される許可 ID は、プログラムのバインディングの過程で使用される許可 ID です。動的 SQL ステートメントに適用される許可 ID は、次のように、バインド時に指定した DYNAMICRULES オプションによってと、その動的 SQL ステートメントを発行したパッケージの現在のランタイム環境によって決まります。

- バインド動作をもつパッケージの場合に使用される許可 ID は、パッケージ所有者の許可 ID になります。
- 定義動作をもつパッケージの場合に使用される許可 ID は、それに対応するルーチンの定義者の許可 ID になります。
- 実行動作をもつパッケージの場合に使用される許可 ID は、パッケージを実行するユーザーの許可 ID になります。
- 起動動作をもつパッケージの場合に使用される許可 ID は、ルーチンの起動の時点で有効になっている許可 ID になります。これはランタイム許可 ID と呼ばれます。

詳しくは、 71 ページの『ランタイムにおける動的 SQL の特性』を参照してください。

SQL ステートメントで指定される*許可名* を、そのステートメントの許可 ID と混同してはなりません。 許可名は、種々の SQL ステートメントで使用される ID です。 許可名は、スキーマの所有者を指定するために CREATE SCHEMA ステートメントで使用されます。許可名は、付与または取り消しの操作の対象を指定するために GRANT および REVOKE ステートメントで使用されます。 *X* に特権を付与すると、それ以降、その特権を必要とするステートメントでは、 *X* (またはグループ *X* のメンバー) が許可 ID になるということです。

例:

- ユーザー ID が SMITH であり、またデータベース・マネージャーがアプリケーション・プロセスとの接続を確立したときに獲得した許可 ID も SMITH であるとします。以下のステートメントは対話式に実行されます。

  ```
  GRANT SELECT ON TDEPT TO KEENE
  ```

  SMITH はこのステートメントの許可 ID です。したがって、動的 SQL ステートメントでの CURRENT SCHEMA 特殊レジスターのデフォルト値は SMITH になり、静的 SQL でのデフォルトの QUALIFIER プリコンパイル/BIND オプションも SMITH になります。このため、このステートメントを実行できる権限は、SMITH につき合わせて検査され、 SMITH が、 64 ページの『命名規則と暗黙オブジェクト名の修飾』で説明されている修飾規則に基づく *table-name* 暗黙修飾子となります。

  KEENE はこのステートメントで指定された許可名です。 KEENE には SMITH.TDEPT に対する SELECT 特権が付与されます。

- SMITH が管理権限を持っており、セッション中に SET SCHEMA ステートメントが発行されない、以下の動的 SQL ステートメントの許可 ID であるとします。

  ```
  DROP TABLE TDEPT
  ```

  これは、SMITH.TDEPT 表を削除します。

  ```
  DROP TABLE SMITH.TDEPT
  ```

  これは、SMITH.TDEPT 表を削除します。

  ```
  DROP TABLE KEENE.TDEPT
  ```

  これは、KEENE.TDEPT 表を削除します。 KEENE.TDEPT と SMITH.TDEPT は別の表であることに注意してください。

  ```
  CREATE SCHEMA PAYROLL AUTHORIZATION KEENE
  ```

  KEENE は、PAYROLL と呼ばれるスキーマを作成するステートメントで指定されている許可名です。 KEENE は、スキーマ PAYROLL の所有者であり、CREATEIN、 ALTERIN、および DROPIN 特権が与えられ、このような特権を他のユーザーに付与することができます。

ランタイムにおける動的 SQL の特性

BIND オプション DYNAMICRULES によって、動的 SQL ステートメントの処理時の許可検査に使用される許可 ID が決まります。さらにこのオプションは、修飾されるオブジェクト参照子に使用される暗黙修飾子などの他の動的 SQL 属性や、特定の SQL ステートメントを動的に呼び出せるかどうかも制御します。

許可 ID とその他の動的 SQL 属性の一連の値を動的 SQL ステートメント動作と呼びます。使用可能な 4 つの動作は、実行、バインド、定義、および起動です。以下の表で明らかなとおり、 DYNAMICRULES BIND オプションの値とランタイム環境の組み合わせで、使用する動作が決まります。実行動作を意味する DYNAMICRULES RUN がデフォルトです。

表 4. DYNAMICRULES とランタイム環境による動的 SQL ステートメントの動作の決定

	動的 SQL ステートメントの動作	
DYNAMICRULES 値	**スタンドアロン・プログラム環境**	**ルーチン環境**
BIND	バインド動作	バインド動作
RUN	実行動作	実行動作
DEFINEBIND	バインド動作	定義動作
DEFINERUN	実行動作	定義動作
INVOKEBIND	バインド動作	起動動作
INVOKERUN	実行動作	起動動作

実行動作

DB2 では、動的 SQL ステートメントの許可検査に使われる値と、動的 SQL ステートメント内の非修飾オブジェクト参照子の暗黙修飾に使用される値用の初期値に使われる値として、パッケージを実行するユーザーの許可 ID (最初に DB2 に接続した ID) が使用されます。

バインド動作

ランタイムには DB2 では、静的 SQL に対して適

用されるすべての規則が許可と修飾に対して適用されます。その場合、動的 SQL ステートメントの許可検査に使われる値と、動的 SQL ステートメント内の非修飾オブジェクト参照子の暗黙修飾用のパッケージ・デフォルト修飾子として、パッケージ所有者の許可 ID が使用されます。

定義動作

定義動作が適用されるのは、ルーチン・コンテキストで実行されるパッケージ内に動的 SQL ステートメントがあって、しかもそのパッケージが DYNAMICRULES DEFINEBIND または DYNAMICRULES DEFINERUN でバインドされていた場合のみです。その場合、動的 SQL ステートメントの許可検査に使われる値と、ルーチン内の動的 SQL ステートメント内の非修飾オブジェクト参照子の暗黙修飾に使われる値として、ルーチン定義者 (ルーチンのパッケージ・バインド・プログラムではなく) の許可 ID が DB2 で使用されます。

起動動作

起動動作が適用されるのは、ルーチン・コンテキストで実行されるパッケージ内に動的 SQL ステートメントがあって、しかもそのパッケージが DYNAMICRULES INVOKEBIND または DYNAMICRULES INVOKERUN でバインドされていた場合のみです。その場合、動的 SQL の許可検査に使われる値と、ルーチン内の動的 SQL ステートメント内の非修飾オブジェクト参照子の暗黙修飾に使われる値として、ルーチンの起動時点で有効なステートメント許可 ID が DB2 で使用されます。これを以下の表に要約してあります

起動環境	使用 ID
任意の静的 SQL	ルーチンを呼び出す SQL が所属するパッケージの所有者の暗黙または明示的な値。
ビューまたはトリガーの定義での使用	ビューまたはトリガーの定義者。
バインド動作パッケージに属する動的 SQL	ルーチンを呼び出す SQL が所属するパッケージの所有者の暗黙または明示的な値。
実行動作パッケージに属する動的 SQL	DB2 への初期接続を確立するのに使用する ID。
定義動作パッケージに属する動的 SQL	ルーチンを呼び出す SQL が所属するパッケージを使用するルーチンの定義者。
起動動作パッケージに属する動的 SQL	ルーチンを呼び出す現在の許可 ID

実行動作が適用されない場合の制限付きステートメント

バインド、定義、または起動の動作が有効になっている場合、動的 SQL ステートメント
GRANT、REVOKE、ALTER、CREATE、DROP、COMMENT、RENAME、SET INTEGRITY、SET EVENT MONITOR STATE と、ニックネームを参照する照会は使用できません。

DYNAMICRULES オプションを使用するときは、以下の点を考慮してください

バインド、定義、または起動の動作パッケージから実行される動的 SQL ステートメント内の非修飾オブジェクト参照子を修飾するのに、 CURRENT SCHEMA 特殊レジスターを使用することはできません。これは、 CURRENT SCHEMA 特殊レジスターを変更するために SET CURRENT SCHEMA ステートメントを発行した後にもあてはまります。レジスター値は変更されますが、使用されることはありません。

単一の接続中に複数のパッケージが参照されると、それらのパッケージで準備されたすべての動的 SQL ステートメントは、個々のパッケージとその使用場所である環境用に DYNAMICRULES オプションで指定されている行動をとります。

パッケージがバインド動作をとるときは、パッケージのバインド・プログラムには、そのパッケージのユーザーには付与されるべきでない許可が付与されないように気を付けることが大切です。動的ステートメントは、パッケージ所有者の許可 ID を使用するからです。同様に、パッケージが定義動作をとるときは、ルーチンの定義者には、パッケージのユーザーには付与されるべきでない許可が付与されてはなりません。

許可 ID とステートメントの準備

BIND 時に VALIDATE BIND を指定する場合、バインド時に、表やビューを扱うときに必要な特権も存在していなければなりません。そのような特権や参照オブジェクトが存在しない場合に、SQLERROR NOPACKAGE が有効になっていると、バインド操作は失敗します。 SQLERROR CONTINUE オプションが指定されていると、バインド操作は正常に完了し、エラーを生じたステートメントにはすべてフラグが付けられます。そのようなステートメントを実行しようとすると、エラーが生じます。

VALIDATE RUN オプションでパッケージがバインドされると、通常のバインド処理はすべて完了しますが、アプリケーションで参照される表やビューを使うのに必要な特権は、この時点では存在していなくてもかまいません。必要な特権がバインドの実行時に存在しない場合に、アプリケーション内でステートメントを初めて実行すると、必ず増分バインドが実行されます。このときには、ステートメントに必要なすべての特権が存在していなければなりません。必要な特権が存在しない場合、ステートメントの実行は失敗します。

ランタイムの許可検査は、パッケージ所有者の許可 ID を使って行われます。

列名

列名 の意味はコンテキストによって異なります。 列名は以下の目的に使用できます。

- 列の名前を宣言する (CREATE TABLE ステートメントなどで)。
- 列を識別する (CREATE INDEX ステートメントなどで)。
- 列の値を指定する (以下に示すようなコンテキストで)。

  - 列関数においては、列名によって、その関数が適用されるグループまたは中間結果表の列のすべての値が指定されます。たとえば、MAX(SALARY) は、あるグループの SALARY 列の中のすべての値に関数 MAX が適用されることを表します。
  - GROUP BY または ORDER BY 文節の中では、その文節が適用される中間結果表の中のすべての値を、列名によって指定します。たとえば、ORDER BY DEPT と指定すると、 DEPT 列の値によって中間結果表が順序付けられます。
  - 式、検索条件、またはスカラー関数においては、列名によって、その構成項目が適用されるそれぞれの行またはグループの値が指定されます。たとえば、検索条件 CODE = 20 が何らかの行に適用される場合、列名 CODE によって指定される値は、その行の CODE 列の値です。
- FROM 文節における *table-reference* の *correlation-clause* のように、列の名前を一時的に変更する。

修飾列名

列名の修飾子としては、表名、ビュー名、ニックネーム、別名、または相関名が使用されます。

列名が修飾されるかどうかは、コンテキストによって異なります。

- COMMENT ON ステートメントの形式に応じ、単一の列名に修飾の必要な場合があります。複数の列名は修飾してはなりません。
- 列名が列の値を指定している場合、修飾することができます。
- UPDATE ステートメントの割り当て文節では、ユーザーのオプションで修飾することができます。
- 上記以外のコンテキストでは、列名を修飾してはなりません。

修飾子がオプションである場合、修飾子には 2 つの目的があります。これについては、76 ページの『あいまいさを避けるための列名修飾子』および 78 ページの『相関参照内の列名修飾子』で説明されています。

相関名

相関名 は、照会の FROM 文節、および UPDATE または DELETE ステートメントの第 1 文節で指定されます。たとえば、FROM X.MYTABLE Z という文節では、Z が X.MYTABLE の相関名として確立されます。

```
FROM X.MYTABLE Z
```

X.MYTABLE の相関名として Z が定義されると、その SELECT ステートメントにおいては Z だけが X.MYTABLE インスタンスの列を参照する修飾子として使用できます。

相関名は、それが定義されているコンテキスト内でのみ、表、ビュー、ニックネーム、別名、ネストされた表の式または表関数に関連付けられます。したがって、別の目的に使用するために、異なるステートメントの中や同じステートメントの異なる文節の中で同じ相関名を定義できます。

修飾子としての相関名は、あいまいさの回避や、相関参照の確立に利用できます。また、表、ビュー、ニックネーム、別名の単なる短縮名として利用することもでき

ます。ネストされた表の式または表関数の場合、相関名は結果表を識別するのに必要になります。例では、X.MYTABLE と何度も入力するのを避けるためだけに使用されています。

相関名を表名、ビュー名、ニックネーム、または別名に対して指定する場合、その表、ビュー、ニックネーム、または別名のそのインスタンスでの列への修飾子付き参照は、その表名、ビュー名、ニックネーム、別名ではなく、相関名を使用しなければなりません。たとえば、以下の例の EMPLOYEE.PROJECT への参照は、EMPLOYEE に対する相関名がすでに指定されているため誤りです。

例:

```
FROM EMPLOYEE E
  WHERE EMPLOYEE.PROJECT='ABC'      * incorrect*
```

PROJECT に対する修飾子付き参照では、以下のように、相関名 “E” を代わりに使用する必要があります。

```
FROM EMPLOYEE E
  WHERE E.PROJECT='ABC'
```

FROM 文節で指定する名前は、*直接的* か*間接的* のどちらかです。相関名が指定されていない場合の表名、ビュー名、ニックネーム、または別名は FROM 文節の直接的な名前です。相関名は常に直接的な名前です。たとえば、以下の FROM 文節では、EMPLOYEE には相関名が指定され、 DEPARTMENT には指定されていません。このため、DEPARTMENT は直接的な名前、EMPLOYEE は間接的な名前になります。

```
FROM EMPLOYEE E, DEPARTMENT
```

FROM 文節の直接的な表名、ビュー名、ニックネーム、または別名は、その FROM 文節での直接的なその他の表名、ビュー名、ニックネーム、または FROM 文節の相関名のどれかと同じになる場合があります。これにより列名参照があいまいとなり、エラー (SQLSTATE 42702) となる可能性があります。

以下に示す最初の 2 つの FROM 文節は、直接的な名前である EMPLOYEE をそれぞれ 2 回以上参照するような参照を行っていないので正しい FROM 文節です。

1. 次の FROM 文節が与えられているものとします。

   ```
   FROM EMPLOYEE E1, EMPLOYEE
   ```

   EMPLOYEE.PROJECT のような修飾子付き参照は、 FROM 文節での EMPLOYEE の 2 番目のインスタンスの列を指すことになります。 EMPLOYEE の 1 番目のインスタンスに対する修飾子付き参照では、相関名 “E1” を使用する (E1.PROJECT) 必要があります。
2. 次の FROM 文節が与えられているものとします。

   ```
   FROM EMPLOYEE, EMPLOYEE E2
   ```

   EMPLOYEE.PROJECT のような修飾子付き参照は、 FROM 文節での EMPLOYEE の 1 番目のインスタンスの列を指すことになります。 EMPLOYEE の 2 番目のインスタンスに対する修飾子付き参照では、相関名 “E2” を使用する (E2.PROJECT) 必要があります。
3. 次の FROM 文節が与えられているものとします。

```
FROM EMPLOYEE, EMPLOYEE
```

この文節では、直接的な 2 つの表名 (EMPLOYEE と EMPLOYEE) が同じになっています。これ自体は可能ですが、特定の列名への参照があいまいになってしまいます (SQLSTATE 42702)。

4. 次のステートメントが与えられているものとします。

```
SELECT *
  FROM EMPLOYEE E1, EMPLOYEE E2            * incorrect *
  WHERE EMPLOYEE.PROJECT = 'ABC'
```

修飾子付き参照 EMPLOYEE.PROJECT は誤りです。これは、FROM 文節の EMPLOYEE の 2 つのインスタンスの両方に相関名があるためです。そうするのではなく、PROJECT を参照するときは、どちらかの相関名 (E1.PROJECT または E2.PROJECT) で修飾する必要があります。

5. 次の FROM 文節が与えられているものとします。

```
FROM EMPLOYEE, X.EMPLOYEE
```

EMPLOYEE の 2 番目のインスタンスの列を参照するときは、 X.EMPLOYEE を使用する (X.EMPLOYEE.PROJECT) 必要があります。 X が、動的 SQL では CURRENT SCHEMA 特殊レジスター値、静的 SQL では QUALIFIER プリコンパイル/BIND オプションである場合、そのような参照はあいまいなので列を参照することはできません。

FROM 文節で相関名を使用することにより、結果表の列に関連付けられる列名のリストを指定することもできます。相関名の場合と同じように、このようにリストされた列名は、照会時に列の参照に使用する必要がある列の*直接的な* 名前になります。列名のリストを指定する場合、基礎表の列名は*間接的な名前* になります。

次の FROM 文節が与えられているものとします。

```
FROM DEPARTMENT D (NUM,NAME,MGR,ANUM,LOC)
```

D.NUM などの修飾子の付いた参照は、 DEPTNO として表に定義されている DEPARTMENT 表の最初の列を表します。この FROM 文節を使用した D.DEPTNO の参照は、列名 DEPTNO が間接的な列名であるため誤りです。

あいまいさを避けるための列名修飾子

関数、GROUP BY 文節、ORDER BY 文節、式、または検索条件のコンテキストでは、列名は、何らかの表、ビュー、ニックネーム、ネストされた表の式あるいは表関数の列の値を指します。列を収容する可能性のある表、ビュー、ニックネーム、ネストされた表の式および表関数は、そのコンテキストのオブジェクト表 と呼ばれます。複数の表が同じ名前の列を備えている場合があります。列名を修飾する理由の 1 つは、列がどの表のものかを指定することにあります。列名の修飾子は、SQL プロシージャーにおいて、列名と SQL ステートメントで使われる SQL 変数名を区別するときにも役立ちます。

ネストされた表の式または表関数は、 FROM 文節で先行する *table-references* をオブジェクト表と見なします。後続の *table-references* はオブジェクト表とは見なされません。

表指定子: 特定のオブジェクト表を指定する修飾子は、*表指定子* と呼ばれます。オブジェクト表を指定する文節では、そのオブジェクト表に対する表指定子も設定します。以下の例は、SELECT 文節の式のオブジェクト表を、直後の FROM 文節で指定しています。

```
SELECT CORZ.COLA, OWNY.MYTABLE.COLA
  FROM OWNX.MYTABLE CORZ, OWNY.MYTABLE
```

FROM 文節の表指定子は次のように設定されます。

- 表、ビュー、ニックネーム、別名、ネストされた表の式または表関数の後に続く名前は、相関名でもあり表指定子でもあります。したがって、CORZ は表指定子です。選択リストの中で、最初の列名を修飾するために CORZ が使用されています。
- 直接的な表名、ビュー名、ニックネーム、または別名は、表指定子です。したがって、OWNY.MYTABLE は表指定子です。選択リストの中で、第 2 の列名を修飾するために OWNY.MYTABLE が使用されています。

列へのあいまいな参照が生じる可能性を避けるため、各表指定子は、特定の FROM 文節の中ではユニークでなければなりません。

未定義またはあいまいな参照の回避: 列名が列の値を参照する場合、その名前の付いた列がただ 1 つのオブジェクト表の中になければなりません。以下の状態はエラーと見なされます。

- 指定された名前の列の入ったオブジェクト表がない。この参照は未定義になります。
- 列名が表指定子によって修飾されているが、指定された表に指定された名前の列が入っていない。この参照も未定義になります。
- 名前が修飾なしで、2 つ以上のオブジェクト表の中にその名前の列がある。この参照はあいまいです。
- 列名が表指定子で修飾されているが、その指定されている表が FROM 文節の中でユニークでなく、指定されている表のどちらのオカレンスにもその列がある。この参照はあいまいです。
- 列名は、TABLE キーワードが先行しないネストされた表の式、もしくは右外部結合または全外部結合の右側のオペランドである表関数またはネストされた表の式にある。列名は、ネストされた表の式の全選択内の *table-reference* の列を指しません。この参照は未定義になります。

ユニークに定義された表指定子で列名を修飾することによって、あいまいな参照を避けてください。列が名前の異なる複数のオブジェクト表の中に入っている場合、その表名を指定子として使用することができます。また、相関名の後に続いて列名のリストを使用してオブジェクト表のいずれかの列にユニーク名を指定することによって、表指定子を使用しなくてもあいまいな参照を避けることができます。

列を直接的な表名形式の表指定子で修飾するとき、直接的な表名は修飾の付いた形式でも付かない形式でも使用できます。しかし、表名、ビュー名、またはニックネームと、表指定子を完全に修飾した後は、使用される修飾子と表が同じものでなければなりません。

1. たとえば、ステートメントの許可 ID が CORPDATA とすると、以下のステートメントは有効です。

```
SELECT CORPDATA.EMPLOYEE.WORKDEPT
                    FROM EMPLOYEE
```

は有効なステートメントです。

2. ステートメントの許可 ID が REGION の場合、以下は無効です。

```
SELECT CORPDATA.EMPLOYEE.WORKDEPT
  FROM EMPLOYEE                          * 誤り *
```

これは、EMPLOYEE が表 REGION.EMPLOYEE を表しているのに対し、WORKDEPT の修飾子が別の表 CORPDATA.EMPLOYEE を表しているためです。

相関参照内の列名修飾子

全選択 とは、種々の SQL ステートメントのコンポーネントとして使用される照会の 1 つの形式です。任意のステートメントの検索条件で使用される全選択は、*副照会* と呼ばれます。ステートメントで式として単一値を検索するのに使用される全選択は、*スカラー全選択* または*スカラー副照会* と呼ばれます。照会の FROM 文節で使用される全選択は、*ネストされた表の式* と呼びます。検索条件、スカラー副照会、およびネストされた表の式の副照会を、このトピックのこれ以降の部分では副照会と呼びます。

副照会にはそれ自身の副照会を収めることができます。その副照会の中に、さらに副照会が収められていてもかまいません。したがって、SQL ステートメントに副照会の階層が入ることになる場合があります。副照会を収容する階層のエレメントは、そこに収容された副照会よりも高いレベルとされます。

階層のあらゆるエレメントには、1 つ以上の表指定子が入っています。副照会は、階層中の自分のレベルで指定されている表の列だけでなく、階層中のそれより前のレベルで指定されている表の列から階層の最上位で識別される表の列まで参照できます。上位のレベルで指定される表の列への参照は、*相関参照* と呼ばれます。

既存の SQL 標準規格との互換性のため、修飾子付きと修飾子なしのどちらの列名も相関参照として認められています。ただし、副照会で使用されるすべての列参照を修飾することをお勧めします。そうしないと、同一の列名により予期しない結果が生じることがあります。たとえば、ある階層の表が相関参照として同じ列名を収めるように変更され、ステートメントが再度作成処理された場合、新たな参照は変更された表に対して適用されます。

副照会内の列名が修飾されているときは、修飾されているその列名が出現するのと同じ副照会から探索が始まり、修飾子に一致する表指定子が見つかるまで、階層の上位へ向かって階層の各レベルの探索が続けられます。該当するものが見つかると、その表に指定の列があるかどうかが調べられます。列名を収容しているレベルより高いレベルで表が見つかった場合、これは表指定子が見つかったレベルに対する相関参照となります。ネストされた表の式の全選択より上の階層を探索するためには、ネストされた表の式の前にオプションの TABLE キーワードを指定しなければなりません。

副照会に収容されている列名が修飾されていないときは、その列名が出現するのと同じ副照会から始めて、階層の各レベルで参照されている表が探索され、一致する列名が見つかるまで、階層の上位へ向かって探索が続けられます。列名を収容して

いるレベルより高いレベルの表で列が見つかった場合は、その列を収めた表が見つかったレベルに対する相関参照となります。列名が、特定のレベルの 2 つ以上の表で見つかった場合は、参照はあいまいになり、エラーと見なされます。

以下の例の T は、どの場合も、列 C の入った表指定子を参照しています。列名 T.C は、以下の条件がすべて満たされているときにのみ相関参照となります (この T は暗黙の修飾子か明示的な修飾子のいずれかを表します)。

- T.C は副照会の式で使用される。
- T が、その副照会の from 文節で使用されている表を指していない。
- T が、副照会を収容している上位の階層レベルで使用されている表を示している。

同じ表、ビュー、またはニックネームが、多くのレベルで指定されていることがあるため、表指定子としてはユニークな相関名を使用するようお勧めします。 T が 2 つ以上のレベルで表の指定に使用される場合 (T は表名自体か重複の相関名)、 T.C は、T.C の入った副照会を最も直接的に収容するように T が使用されているレベルを参照することになります。上位レベルへの相関が必要な場合、ユニークな相関名を使用する必要があります。

相関参照 T.C は、2 つの検索条件が、検索条件 1 が副照会で、検索条件 2 が上位のレベルでそれぞれ適用されている T の行またはグループでの C の値を識別します。条件 2 が WHERE 文節で使用される場合、副照会は条件 2 が適用される行ごとに評価されます。条件 2 が HAVING 文節で使用される場合、副照会は条件 2 が適用されるグループごとに評価されます。

たとえば、次のステートメントにおいて、(最後の行の) 相関参照 X.WORKDEPT は、最初の FROM 文節のレベルにある表 EMPLOYEE の WORKDEPT の値を指します。 (この文節は X を EMPLOYEE の相関名として設定します。) このステートメントは、その部署の平均給与を下回る社員のリストを作成するものです。

```
SELECT EMPNO, LASTNAME, WORKDEPT
  FROM EMPLOYEE X
  WHERE SALARY < (SELECT AVG(SALARY)
                    FROM EMPLOYEE
                  WHERE WORKDEPT = X.WORKDEPT)
```

次の例は、THIS を相関名として使用しています。このステートメントは、社員のいない部門の行を削除します。

```
DELETE FROM DEPARTMENT THIS
   WHERE NOT EXISTS(SELECT *
                    FROM EMPLOYEE
                    WHERE WORKDEPT = THIS.DEPTNO)
```

ホスト変数の参照

ホスト変数 とは、以下のいずれかです。

- C の変数、C++ の変数、COBOL のデータ項目、FORTRAN の変数、または Java の変数など、ホスト言語の変数

または

- SQL 拡張機能を使って宣言された変数から SQL のプリコンパイラーによって生成されたホスト言語構成

これらは、SQL ステートメントで参照されています。ホスト変数はホスト言語のステートメントによって直接定義されるか、または SQL 拡張機能を使って間接的に定義されます。

SQL ステートメント内のホスト変数は、ホスト変数宣言規則に従ってプログラム内に記述されたホスト変数を識別する必要があります。

SQL ステートメントで使用されるホスト変数はすべて、 REXX を除くすべてのホスト言語の SQL DECLARE セクションで宣言する必要があります。 SQL DECLARE セクションで宣言されている変数と同じ名前の変数を、 SQL DECLARE セクションの外部で宣言することはできません。 SQL DECLARE セクションは、BEGIN DECLARE SECTION で始まり、 END DECLARE SECTION で終わります。

メタ変数の *host-variable* (ホスト変数) が構文図の中で使われる場合、それはホスト変数への参照を示します。 VALUES INTO 文節または、FETCH か SELECT INTO ステートメントの INTO 文節のホスト変数は、行の中の列の値または式の値が割り当てられるホスト変数を識別するものです。その他のコンテキストでのホスト変数は、アプリケーション・プログラムからデータベース・マネージャーに渡される値を指定します。

動的 SQL におけるホスト変数

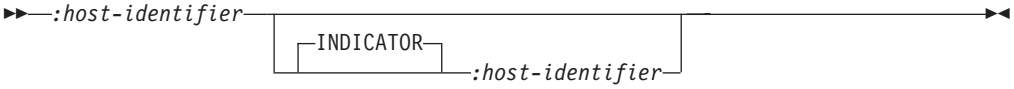
動的 SQL ステートメントにおいては、ホスト変数の代わりにパラメーター・マーカーが使用されます。 パラメーター・マーカーは疑問符 (?) で表されます。これは、動的 SQL ステートメントにおいてアプリケーションが値を提供する位置、すなわち、ステートメント・ストリングが静的 SQL ステートメントであるとすれば、ホスト変数が来ることになる位置を示します。以下に、ホスト変数を使った静的 SQL ステートメントの例を示します。

```
INSERT INTO DEPARTMENT
  VALUES (:hv_deptno, :hv_deptname, :hv_mgrno, :hv_admrdept)
```

次に、パラメーター・マーカーを使った動的 SQL ステートメントの例を示します。

```
INSERT INTO DEPARTMENT VALUES (?, ?, ?, ?)
```

構文図におけるメタ変数 *host-variable* (ホスト変数) は、一般に以下のように展開されます。



各 *host-identifier* (ホスト ID) は、ソース・プログラムの中で宣言される必要があります。 2 番目のホスト ID で指定される変数は、データ型が短整数のものでなければなりません。

最初の host-identifier (ホスト ID) は、*メイン変数* を指定します。演算に応じて、このホスト ID はデータベース・マネージャーに値を提供したり、またはデータベース・マネージャーから提供される値を受け取ったりします。入力ホスト変数は、ランタイム・アプリケーション・コード・ページの値を提供します。出力ホスト変数には、データが出力アプリケーション変数にコピーされるときに、必要に応じて

ランタイム・アプリケーション・コード・ページに変換される値が提供されます。指定されるホスト変数は、同じプログラム内で入力変数と出力変数の両方として機能できます。

2 番目の host-identifier (ホスト ID) は、その*標識変数* を示します。標識変数の目的は以下のとおりです。

- NULL 値を指定する。標識変数の負の値は、NULL 値を指定するものとなります。 -2 の値は、結果を求める際に数値変換または演算式のエラーが発生したことを示します。
- 切り捨てられたストリングの元の長さを記録する (値のソースがラージ・オブジェクト・タイプでない場合)。
- ホスト変数に割り当てたときに時刻が切り捨てられた場合、その時刻の秒の部分を記録します。

たとえば、:HV1:HV2 を使用して挿入値または更新値を指定する場合に、 HV2 が負であると、指定される値は NULL 値になります。 HV2 が負でない場合、指定される値は HV1 の値です。

同様に、:HV1:HV2 が VALUES INTO 文節、または FETCH あるいは SELECT INTO ステートメントに指定され、しかも戻された値が NULL 値である場合には、HV1 は変更されず、HV2 は負の値に設定されます。 DFT_SQLMATHWARN を yes にしてデータベースが構成されている場合 (または静的 SQL ステートメントのバインドの過程である場合)、 HV2 を -2 にすることができます。 HV2 が -2 である場合、HV1 の数値タイプへの変換エラー、または HV1 の値を判別するために使用される演算式の評価エラーにより、 HV1 の値を戻すことができません。 DB2 Universal Database のバージョン 5 より前のクライアント・バージョンを使用してデータベースにアクセスする場合、 HV2 は算術例外に対して -1 になります。戻された値が NULL 値でない場合は、その値が HV1 に割り当てられ、 HV2 はゼロに設定されます (ただし、 HV1 への割り当てに非 LOB ストリングのストリング切り捨てが必要になる場合を除きます。この場合 HV2 はストリングの元の長さに設定されます)。割り当て時に時刻の秒の部分の切り捨てが必要な場合、HV2 は秒数に設定されます。

2 番目のホスト ID が省略されている場合は、ホスト変数は標識変数を持たないことになります。ホスト変数参照 :HV1 によって指定される値は、常に HV1 の値であり、変数に NULL 値を割り当てることはできません。したがって、この形式は、対応する列で NULL 値を使えない場合以外は、INTO 文節では使用しないでください。この形式が使用された場合に、列に NULL 値が入っていると、データベース・マネージャーはランタイムにエラーを生成します。

ホスト変数を参照する SQL ステートメントは、対象のホスト変数の宣言の範囲内にある必要があります。カーソルの SELECT ステートメントで参照されるホスト変数の場合、この規則は DECLARE CURSOR ステートメントではなく、OPEN ステートメントに適用されます。

例: プロジェクト (PROJNO) 'IF1000' で、PROJECT 表を使用して、ホスト変数 PNAME (VARCHAR(26)) はプロジェクト名 (PROJNAME) に、ホスト変数 STAFF (dec(5,2)) はスタッフ配置の平均レベル (PRSTAFF) に、ホスト変数 MAJPROJ (char(6)) は主要プロジェクト (MAJPROJ) に設定します。 PRSTAFF と MAJPROJ

列は NULL 値である可能性があるため、標識変数 STAFF_IND (短精度整数) と MAJPROJ_IND (短精度整数) を使用します。

```
SELECT PROJNAME, PRSTAFF, MAJPROJ
  INTO :PNAME, :STAFF :STAFF_IND, :MAJPROJ :MAJPROJ_IND
  FROM PROJECT
  WHERE PROJNO = 'IF1000'
```

MBCS の考慮事項: ホスト変数名にマルチバイト文字を使用できるかどうかは、ホスト言語によって決まります。

BLOB、CLOB、および DBCLOB のホスト変数の参照

通常の BLOB、CLOB、および DBCLOB の変数、 LOB のロケーター変数 (『ロケーター変数の参照』を参照)、および LOB ファイル参照変数 (83 ページの『BLOB、CLOB、および DBCLOB ファイル参照変数の参照』を参照) は、すべてのホスト言語の中で定義可能です。 LOB が可能なロケーションでは、構文図の *host-variable* (ホスト変数) という用語は、通常のホスト変数、ロケーター変数、またはファイル参照変数を指します。これらはネイティブのデータ型ではないため、SQL 拡張機能が使用され、それぞれの変数を表現するのに必要なホスト言語構成をプリコンパイラーが生成します。 REXX の場合、LOB はストリングにマップされます。

ラージ・オブジェクト値全体を保持できるほど大きい変数を定義できる場合もあります。このような場合で、サーバーからのデータ転送を据え置いてもパフォーマンス上のメリットが期待できない場合は、ロケーターを使用する必要はありません。しかし、ホスト言語やスペースの制限により、ラージ・オブジェクト全体を一度に一時記憶に保管するのが難しい場合がよくありますし、パフォーマンス上のメリットを考え合わせた上で、ラージ・オブジェクトはロケーターによって参照し、一度にラージ・オブジェクトの一部分だけを保持するホスト変数にオブジェクトの一部を選択して割り当て、そこで更新するという方法を採用することもできるかもしれません。

他のすべてのホスト変数と同様に、ラージ・オブジェクトのロケーター変数にも標識変数を対応させることができます。ラージ・オブジェクトのロケーター・ホスト変数に対応する標識変数は、他のデータ型の標識変数と同じように動作します。データベースから NULL 値が戻されると、標識変数が設定され、ロケーター・ホスト変数は変更されません。つまり、ロケーターが NULL 値を指すことはないということです。

ロケーター変数の参照

ロケーター変数 は、アプリケーション・サーバーで LOB 値を表すロケーターの入ったホスト変数です。

SQL ステートメントにおけるロケーター変数は、ロケーター変数の宣言規則に従ってプログラムに記述されたロケーター変数を識別したものでなければなりません。これは常に SQL ステートメントによって間接的に行われます。

構文図で locator-variable (ロケーター変数) の語が使用される場合、それはロケーター変数への参照を表します。メタ変数 *locator-variable* (ロケーター変数) は、*host-variable* (ホスト変数) の場合と同じく、 *host-identifier* (ホスト ID) を組み込めるように拡張されています。

ロケーターに対応する標識変数が NULL 値のときは、参照される LOB の値は NULL 値です。

現時点で何の値も表していないロケーター変数が参照されると、エラー (SQLSTATE 0F001) になります。

トランザクションのコミット時、またはトランザクションの終了時に、そのトランザクションが獲得していたロケーターはすべて解放されます。

BLOB、CLOB、および DBCLOB ファイル参照変数の参照

BLOB、CLOB、および DBCLOB のファイル参照変数は、 LOB の直接のファイル入出力に使用されるもので、すべてのホスト言語で定義可能です。 これらはネイティブのデータ型ではないため、 SQL 拡張機能が使用され、それぞれの変数を表現するのに必要なホスト言語構成をプリコンパイラーが生成します。 REXX の場合、LOB はストリングにマップされます。

LOB ロケーターが LOB バイトを収容するのではなく LOB バイトを表すのと同じように、ファイル参照変数はファイルを収容するのではなくファイルを表します。データベースの照会、更新、および挿入では、ファイル参照変数を使用して 1 つの列値を保管したり検索したりすることができます。

ファイル参照変数には以下の特性があります。

データ型
BLOB、CLOB、または DBCLOB。この特性は、変数の宣言時に指定されます。

方向
これはアプリケーション・プログラムによってランタイムに指定される必要があります (ファイル・オプション値の一部として)。方向は以下のどちらかです。

- 入力 (EXECUTE ステートメント、OPEN ステートメント、 UPDATE ステートメント、INSERT ステートメント、または DELETE ステートメントでのデータのソースとして使用されます)。
- 出力 (FETCH ステートメントまたは SELECT INTO ステートメントのデータの宛先として使用されます)。

ファイル名
これはアプリケーション・プログラムによってランタイムに指定される必要があります。以下のいずれかです。

- ファイルの絶対パス名 (こちらをお勧めします)。
- 相対ファイル名。相対ファイル名を指定した場合、それはクライアント・プロセスの現行パスに追加されます。

アプリケーション内では、ファイルは 1 つのファイル参照変数でのみ参照する必要があります。

ファイル名の長さ
これはアプリケーション・プログラムによってランタイムに指定される必要があります。ファイル名の長さをバイト単位で表したものです。

ファイル・オプション

アプリケーションがファイル参照変数を使用するには、事前にいくつかのオプションの中の 1 つをその変数に割り当てる必要があります。オプションの設定は、ファイル参照変数構造の中のフィールドの INTEGER 値によって行います。ファイル参照変数ごとに、以下の値の 1 つを指定する必要があります。

- 入力 (クライアントからサーバーへ)

  SQL_FILE_READ
  これは、オープン、読み取り、クローズの対象となる通常のファイルです。 (オプションは、COBOL では SQL-FILE-READ、FORTRAN では sql_file_read、REXX では READ です。)

- 出力 (サーバーからクライアントへ)

  SQL_FILE_CREATE
  新規ファイルを作成します。該当のファイルがすでに存在していると、エラーが戻されます。 (オプションは、COBOL では SQL-FILE-CREATE、FORTRAN では sql_file_create、および REXX では CREATE です。)

  SQL_FILE_OVERWRITE (上書き)
  指定した名前のファイルが存在している場合には上書きされ、存在していない場合には新たなファイルが作成されます。(オプションは、COBOL では SQL-FILE-OVERWRITE、FORTRAN では sql_file_overwrite、および REXX では OVERWRITE です。)

  SQL_FILE_APPEND
  指定した名前のファイルが存在している場合には出力がそれに追加されます。存在していない場合には新たなファイルが作成されます。(オプションは、COBOL では SQL-FILE-APPEND、FORTRAN では sql_file_append、および REXX では APPEND です。)

データ長
これは入力では使用されません。出力のとき、ファイルに書き込まれる新規データの長さがこのデータ長に設定されます。このデータ長はバイト単位です。

他のすべてのホスト変数と同様に、ファイル参照変数にも標識変数を対応させることができます。

出力ファイル参照変数の例 (C の場合): 宣言部が以下のようにコーディングされているとします。

```
EXEC SQL BEGIN DECLARE SECTION
   SQL TYPE IS CLOB_FILE  hv_text_file;
   char  hv_patent_title[64];
EXEC SQL  END DECLARE SECTION
```

これをプリプロセスした後は以下のようになります。

```
EXEC SQL BEGIN DECLARE SECTION
   /* SQL TYPE IS CLOB_FILE  hv_text_file; */
   struct {
       unsigned long  name_length; //  File Name Length
       unsigned long  data_length; //  Data Length
       unsigned long  file_options; // File Options
       char           name[255];   // File Name
   } hv_text_file;
   char  hv_patent_title[64];
EXEC SQL END DECLARE SECTION
```

その後、以下のコードを使って、データベースの CLOB の列から選択し、:hv_text_file で参照される新規ファイルに書き込むことができます。

```
strcpy(hv_text_file.name, "/u/gainer/papers/sigmod.94");
hv_text_file.name_length = strlen("/u/gainer/papers/sigmod.94");
hv_text_file.file_options = SQL_FILE_CREATE;

EXEC SQL SELECT content INTO :hv_text_file from papers
     WHERE TITLE = 'The Relational Theory behind Juggling';
```

入力ファイル参照変数の例 (C の場合): 前出の例と同じ宣言部を考えます。以下のコードは、:hv_text_file によって参照される通常ファイルからのデータを CLOB の列へ挿入するものです。

```
strcpy(hv_text_file.name, "/u/gainer/patents/chips.13");
hv_text_file.name_length = strlen("/u/gainer/patents/chips.13");
hv_text_file.file_options = SQL_FILE_READ:
strcpy(:hv_patent_title, "A Method for Pipelining Chip Consumption");

EXEC SQL INSERT INTO patents( title, text )
         VALUES(:hv_patent_title, :hv_text_file);
```

構造化タイプ・ホスト変数の参照

構造化タイプ変数は、FORTRAN、REXX、および Java を除く、すべてのホスト言語で定義できます。これらはネイティブのデータ型ではないため、 SQL 拡張機能が使用され、それぞれの変数を表現するのに必要なホスト言語構成をプリコンパイラーが生成します。

他のすべてのホスト変数と同様に、構造化タイプ変数にも標識変数を対応させることができます。構造化タイプ・ホスト変数に対応する標識変数は、他のデータ型の

標識変数と同じように動作します。データベースから NULL 値が戻されると、標識変数が設定され、構造化タイプ・ホスト変数は変更されません。

構造化タイプ用の実際のホスト変数は、組み込みデータ型として定義されます。構造化タイプと関連した組み込みデータ型は、以下のようなアクセスが可能でなければなりません。

- プリコンパイル・コマンドで指定した TRANSFORM GROUP オプションによって定義したとおりの、構造化タイプの FROM SQL トランスフォーム関数の結果に基づくアクセス。
- プリコンパイル・コマンドで指定した TRANSFORM GROUP オプションによって定義したとおりの、構造化タイプの TO SQL トランスフォーム関数のパラメーターへのアクセス。

ホスト変数の代わりにパラメーター・マーカーを使用している場合、 SQLDA に適切なパラメーター・タイプの特性を指定する必要があります。この場合、SQLDA には SQLVAR 構造のセットが「2 つ」必要です。また、副次 SQLVAR の SQLDATATYPE_NAME フィールドには、構造化タイプのスキーマおよびタイプ名を入れなければなりません。 SQLDA 構造でスキーマを省略すると、エラーが発生します (SQLSTATE 07002)。

例: C プログラムで、(組み込みタイプの BLOB(1048576) を使用して、タイプ POLYGON の) ホスト変数 *hv_poly* と *hv_point* を定義します。

```
EXEC SQL BEGIN DECLARE SECTION;
      static SQL
         TYPE IS POLYGON AS BLOB(1M)
         hv_poly, hv_point;
EXEC SQL END DECLARE SECTION;
```

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE ALIAS ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『PREPARE ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『SET SCHEMA ステートメント』
- 527 ページの『付録 A. SQL の制限値』
- 539 ページの『付録 C. SQLDA (SQL 記述子域)』
- 743 ページの『付録 G. 予約済みスキーマ名と予約語』
- 801 ページの『付録 N. 日本語および中国語 (繁体字) の拡張 UNIX コード (EUC) の考慮事項』
- 473 ページの『SQL 照会』
- 94 ページの『ラージ・オブジェクト (LOB)』

データ型

データ型

SQL で扱うことのできる一番小さいデータの単位は*値* です。値の解釈方法は、値の出所 (ソース) のデータ型によって異なります。ソースには、以下のものがあります。

- 定数
- 列
- ホスト変数
- 関数
- 式
- 特殊レジスター

DB2 は、いくつかの組み込みデータ型をサポートします。また、ユーザー定義のデータ型もサポートします。 図 10 は、サポートされる組み込みデータ型を示しています。

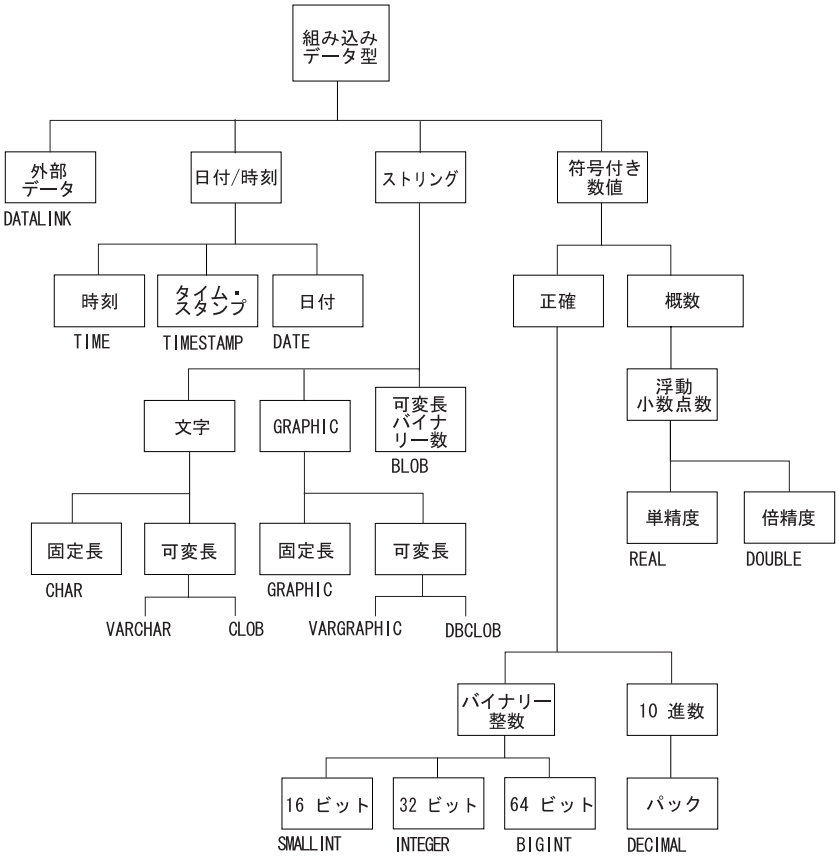


図 10. DB2 組み込みデータ型

すべてのデータ型には、NULL 値が入っています。 NULL 値とは、すべての非 NULL 値と区別されていて、それによって (非 NULL) 値がないことを指し示すた

めの特殊値のことです。すべてのデータ型には NULL 値が含まれますが、 NOT NULL として定義されている列に NULL 値を入れることはできません。

関連資料:

- 102 ページの『ユーザー定義タイプ』

数値

すべての数値には、符号と精度があります。数値の値がゼロのときは、*符号* は正であると見なされます。*精度* とは、符号を除いたビット数または桁数です。

CREATE TABLE ステートメントの説明の中のデータ型の項を参照してください。

短整数 (SMALLINT)

短整数 は、精度が 5 桁の 2 バイト整数です。短整数の範囲は -32 768 から 32 767 です。

長精度整数 (INTEGER)

長精度整数 は、精度が 10 桁の 4 バイトの整数です。長精度整数の範囲は -2 147 483 648 から +2 147 483 647 です。

64 ビット整数 (BIGINT)

64 ビット整数 は、精度が 19 桁の 8 バイトの整数です。 64 ビット整数の範囲は、
-9 223 372 036 854 775 808 から +9 223 372 036 854 775 807 です。

単精度浮動小数点 (REAL)

単精度浮動小数点 数は、実数の 32 ビット近似値です。 この数はゼロ、または -3.402E+38 から -1.175E-37、もしくは 1.175E-37 ～ 3.402E+38 の範囲にすることができます。

倍精度浮動小数点 (DOUBLE または FLOAT)

倍精度浮動小数点 数は、実数の 64 ビットの近似値です。 値は、ゼロ、または -1.79769E+308 から -2.225E-307、または 2.225E-307 ～ 1.79769E+308 の範囲です。

10 進数 (DECIMAL または NUMERIC)

10 進数値 は、暗黙的な小数点を持つパック 10 進数です。小数点の位置は、その数値の精度と位取りによって決定されます。 数値の小数部分の桁数である位取りが、負になったり精度数よりも大きくなったりすることはありません。最大精度は 31 桁です。

10 進数の列の値は、すべて同じ精度と位取りの値です。 10 進数の変数または 10 進数の列の数値の範囲は、 *-n* から *+n* です (絶対値 *n* は、適切な精度および 10 進数で表現できる最も大きな数値)。最大範囲は -10**31+1 から 10**31-1 です。

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE TABLE ステートメント』
- 539 ページの『付録 C. SQLDA (SQL 記述子域)』

文字ストリング

文字ストリング は、一連のバイトです。ストリングの長さは、その一連のバイトのバイト数です。長さがゼロの場合、その値は空ストリング と呼ばれます。この値を NULL 値と混同しないようにしてください。

固定長文字ストリング (CHAR)

固定長ストリングの列の値は、すべて同じ長さです。この長さは、その列の長さ属性によって決定されます。長さ属性は、1 以上 254 以下でなければなりません。

可変長文字ストリング

可変長文字ストリングには次の 3 つのタイプがあります。

- VARCHAR 値は、最大 32 672 バイトまでの長さにすることができます。
- LONG VARCHAR 値は、最大 32 700 バイトまでの長さにすることができます。
- CLOB (文字ラージ・オブジェクト) 値は、最大 2 ギガバイト (2 147 483 647 バイト) までの長さにすることができます。 CLOB は、(単一文字セットで記述された文書などの) ラージ SBCS、または混合 (SBCS および MBCS) 文字ベース・データを保管するのに使用されます。したがって、それに関連する SBCS または混合コード・ページがあります。

結果が LONG VARCHAR または CLOB データ型となる式、および構造型列に対して、特殊な制限が適用されます。

- DISTINCT 文節が先行している SELECT リスト
- GROUP BY 文節
- ORDER BY 文節
- UNION ALL 以外のセット演算子の副選択
- 基本述部、多値比較述部、BETWEEN 述部、または IN 述部
- 列関数
- VARGRAPHIC、TRANSLATE、および日付/時刻スカラー関数
- LIKE 述部のパターン・オペランドまたは POSSTR 関数の探索ストリング・オペランド
- 日付/時刻値のストリング表記。

VARCHAR を引き数として取る SYSFUN スキーマの関数は、 4 000 バイトよりも長い VARCHAR を引き数として受け入れません。しかし、そのような関数の多くには、 CLOB(1M) を受け入れるための代替シグニチャーが用意されています。そのような代替シグニチャーが用意されている関数の場合は、ユーザーが 4 000 バイトよりも長い VARCHAR ストリングを明示的に CLOB へキャストし、結果が戻されたら任意の長さの VARCHAR へ再びキャストし直すという手順を取ります。

C の NULL 終了文字ストリングは、プリコンパイル・オプションの標準レベルに応じて、異なった方式で処理されます。

それぞれの文字ストリングは、さらに次のいずれかと定義されます。

ビット・データ
コード・ページに対応していないデータ。

1 バイト文字セット (SBCS) データ
それぞれの文字が 1 バイトで表現されるデータ。

混合データ
1 バイト文字セットとマルチバイト文字セット (MBCS) の文字の混合を納めたデータ。

GRAPHIC ストリング

GRAPHIC ストリング は、 2 バイト文字データを表す一連のバイトです。ストリングの長さは、その一連のバイトの 2 バイト文字の数です。長さがゼロの場合、その値は空ストリング と呼ばれます。この値を NULL 値と混同しないようにしてください。

GRAPHIC ストリングの値に 2 バイト文字コード・ポイント以外の値が入っていないかどうかを調べる妥当性検査は行われません。 (この規則の例外は、WCHARTYPE CONVERT オプションを指定してプリコンパイルされたアプリケーションです。このオプションを指定した場合、妥当性検査が行われます。) データベース・マネージャーは、 2 バイト文字データが GRAPHIC データ・フィールドに入っていることを想定しています。データベース・マネージャーは、GRAPHIC ストリング値の長さが偶数バイトであることを検査します。

C の NULL 終了 GRAPHIC ストリングは、プリコンパイル・オプションの標準レベルに応じて、異なった方式で処理されます。このデータ型は表内に作成することはできません。データをデータベースに挿入するときやデータベースから検索するときにのみ使用可能です。

固定長 GRAPHIC ストリング (GRAPHIC)

固定長 GRAPHIC ストリングの列の値は、すべて同じ長さです。この長さは、その列の長さ属性によって決定されます。長さ属性は、1 以上 127 以下でなければなりません。

可変長 GRAPHIC ストリング

可変長 GRAPHIC ストリングには次の 3 つのタイプがあります。

- VARGRAPHIC 値は、最長 16 336 個の 2 バイト文字とすることができます。
- LONG VARGRAPHIC 値は、最長 16 350 個の 2 バイト文字とすることができます。
- DBCLOB (2 バイト文字ラージ・オブジェクト) 値は、最長 1 073 741 823 個の 2 バイト文字とすることができます。 DBCLOB は、(単一文字セットで記述された文書のような) 大規模な DBCS 文字ベースのデータの保管に使用されます。したがって、 DBCLOB にはそれに関連する DBCS コード・ページがあります。

最大長が 127 バイトを超える可変長 GRAPHIC ストリングが結果となる式には、特別な制限が適用されます。この制限は、 90 ページの『可変長文字ストリング』で指定されているものと同じです。

バイナリー・ストリング

バイナリー・ストリング は、一連のバイトです。通常はテキスト・データの入った文字ストリングとは異なり、バイナリー・ストリングは従来型ではないデータ、たとえば画像、音声、混合メディアなどを保持します。 FOR BIT DATA サブタイプの文字ストリングも似たような目的で使用されることがありますが、この 2 つのデータ型は互換ではありません。 BLOB スカラー関数を使用すると、 FOR BIT DATA 文字ストリングをバイナリー・ストリングにキャスト (タイプ変換) することができます。バイナリー・ストリングはコード・ページに対応していません。文字ストリングと同じ制限があります (詳細については、 90 ページの『可変長文字ストリング』を参照)。

バイナリー・ラージ・オブジェクト (BLOB)

バイナリー・ラージ・オブジェクト *(BLOB)* は、最長 2 ギガバイト (2 147 483 647 バイト) の可変長ストリングです。 BLOB は、ユーザー定義タイプおよびユーザー定義関数で活用するために構造化データを保持します。 FOR BIT DATA 文字ストリングと同じように、 BLOB ストリングに対応するコード・ページはありません。

ラージ・オブジェクト (LOB)

ラージ・オブジェクト および総称頭字語である LOB は、 BLOB、CLOB、または DBCLOB のデータ型を参照するときに使用されます。 LOB 値は、 90 ページの『可変長文字ストリング』で記述されている LONG VARCHAR 値に適用される制限に従います。このような制限は、LOB ストリングの長さ属性が 254 バイト以下であっても適用されます。

LOB 値は非常に大きいので、この値をデータベース・サーバーからクライアント・アプリケーション・プログラムのホスト変数に転送するには多くの時間がかかります。アプリケーションが一度に処理するのは通常は LOB 値の全体ではなく小さな部分だけなので、アプリケーションはラージ・オブジェクト・ロケーターを使用して LOB を参照できます。

ラージ・オブジェクト・ロケーター つまり LOB ロケーターは、データベース・サーバーの単一 LOB 値を表す値を伴うホスト変数です。

アプリケーション・プログラムは LOB ロケーターに LOB 値を選択できます。その後、アプリケーション・プログラムは LOB ロケーターを使用して、そのロケーター値を入力として指定することによって、その LOB 値に対するデータベース操作 (スカラー関数 SUBSTR、 CONCAT、VALUE、LENGTH の適用、割り当ての実行、 LIKE または POSSTR による LOB の探索、LOB に対するユーザー定義関数の適用など) を要求することができます。出力結果 (クライアントのホスト変数に割り当てられるデータ) は、多くの場合、入力 LOB 値の小さいサブセットとなります。

LOB ロケーターは、基本値以外のものを表現する場合もあり、 LOB 式に対応する値を表現することができます。たとえば、LOB ロケーターで、次の式に対応する値を表現できます。

```
SUBSTR( <lob 1> CONCAT <lob 2> CONCAT <lob 3>, <start>, <length> )
```

そのホスト変数に NULL 値が選択されている場合、標識変数は値が NULL 値であることを示す -1 に設定されます。しかし、LOB ロケーターの場合は、標識変数の意味が少し違います。ロケーター・ホスト変数自体は NULL 値にすることができないので、標識変数の負の値は、その LOB ロケーターが表す LOB 値が NULL 値であることを示します。標識変数の値により、NULL 値情報はクライアントにとってローカルに保持されます。サーバー側では有効なロケーターによって NULL 値を追跡しません。

LOB ロケーターが表すのは 1 つの値であって、データベースの行やロケーションを表すわけではない、ということは重要です。 値がロケーターに選択されると、ロケーターが参照する値に影響を及ぼすような操作を、元の行や表に対して実行することはできません。ロケーターに対応する値は、トランザクションが終了するか、ロケーターが明示的に解放されるか、どちらかが行われるまで有効です。ロケーターでは、この機能を実現するために、追加でデータのコピーなどを行ったりはしません。その代わりに、ロケーター・メカニズムに基本 LOB 値の内容が保管されます。 LOB 値 (または、上記のように式) のマテリアライズは、 LOB 値が実際に何らかの位置を割り当てられるまで延期されます。すなわち、ホスト変数の形式でユーザー・バッファーを割り当てられるか、もしくはデータベースの別のレコードを割り当てられるまでです。

LOB ロケーターは、トランザクションの中で LOB 値を参照するための唯一のメカニズムです。 LOB ロケーターはそれが作成されたトランザクションを超えて存続することはありません。これはデータベース・タイプではなく、データベースに保管されることはありません。したがって、ビューやチェック制約には加わりません。しかし、LOB ロケーターは LOB タイプのクライアント側の表現なので、FETCH、OPEN、または EXECUTE ステートメントで使用される SQLDA 構造の中で記述されるよう、 LOB ロケーターの SQLTYPE が用意されています。

日付/時刻の値

日付/時刻のデータ型には、DATE、TIME、および TIMESTAMP などがあります。日付/時刻の値は、特定の算術演算およびストリング操作で使用することができ、特定のストリングとは互換性がありますが、これはストリングでも数字でもありません。

日付

日付 (date) は、年、月、日の 3 つの部分からなる値です。年の部分の範囲は 0001 から 9999 です。月の部分の範囲は 1 から 12 です。日の部分の範囲は 1 から *x* です (*x* は月によって異なります)。

日付の内部表示は 4 バイトのストリングです。各バイトは、2 桁のパック 10 進数からなります。最初の 2 バイトは年、3 番目のバイトは月、最後のバイトは日です。

DATE 列の長さは、SQLDA の項で説明するように、10 バイトです。これは、日付の値を文字ストリングで表記するために適した長さになっています。

時刻

時刻 (time) は、時、分、秒の 3 つの部分からなる値であり、 24 時間制の時刻を表します。時の部分の範囲は 0 から 24。それ以外の部分の範囲は 0 から 59 です。時が 24 の場合、分と秒の指定はゼロになります。

時刻の内部表示は 3 バイトのストリングです。各バイトは、2 桁のパック 10 進数からなります。最初のバイトは時、2 番目のバイトは分、最後のバイトは秒です。

TIME 列の長さは、SQLDA の項で説明するように、8 バイトです。これは、時刻の値を文字ストリングで表記するために適した長さになっています。

タイム・スタンプ

タイム・スタンプ *(timestamp)* は、 7 つの部分 (年、月、日、時、分、秒、マイクロ秒) から成っており、時刻が小数部分でマイクロ秒を指定する以外は、上記の定義と同様に日時を示します。

タイム・スタンプの内部表示は 10 バイトのストリングです。各バイトは、2 桁のパック 10 進数からなります。最初の 4 バイトは日付、次の 3 バイトは時刻、最後の 3 バイトはマイクロ秒です。

SQLDA に記述されている TIMESTAMP 列の長さは 26 バイトです。これは、値の文字ストリング表示に適した長さです。

日付/時刻の値のストリング表記

データ型が DATE、TIME、または TIMESTAMP の値は、ユーザーが意識することのない内部形式で表されます。ただし、日付、時刻、およびタイム・スタンプの値は、ストリングで表すこともできます。データ型が DATE、TIME、または TIMESTAMP である定数や変数がないため、この表示方法は便利です。日付/時刻の値を取り出すには、この値をストリング変数に割り当てる必要があります。 CHAR または GRAPHIC 関数 (Unicode データベース用のみ) を使用すると、日付/時刻値をストリング表記に変更することができます。通常、ストリング表記は、プログラ

ムがプリコンパイルされるときか、またはデータベースにバインドされるときに、DATETIME オプションの指定によってオーバーライドされるのでない限り、アプリケーションのテリトリー・コードに関連する日付/時刻の値のデフォルトの形式になります。

ラージ・オブジェクト・ストリング、LONG VARCHAR 値、または LONG VARGRAPHIC は、その長さに関係なく、日付/時刻値を表すために使用することはできません (SQLSTATE 42884)。

日付/時刻の値の有効なストリング表記が内部の日付/時刻の値の操作に使用される場合、ストリング表記が日付、時刻、またはタイム・スタンプの値の内部形式に変換されてから、操作が実行されます。

日付、時刻、およびタイム・スタンプのストリングでは、文字と数字しか使用することができません。

日付ストリング: 日付のストリング表示は、数字で始まり、長さが 8 バイト以上のストリングです。 末尾のブランクを付けることができます。月と日の部分の先行ゼロは省略可能です。

日付を示す有効なストリング・フォーマットを、以下の表に示します。各フォーマットは、名前および関連する省略形によって識別されます。

表 5. 日付のストリング表記フォーマット

フォーマット名	省略形	日付 フォーマット	例
国際標準化機構	ISO	yyyy-mm-dd	1991-10-27
IBM USA 標準規格	USA	mm/dd/yyyy	10/27/1991
IBM 欧州標準規格	EUR	dd.mm.yyyy	27.10.1991
日本工業規格西暦	JIS	yyyy-mm-dd	1991-10-27
地域別定義	LOC	アプリケーションのテリトリー・コードに依存します。	—

時刻ストリング: 時刻のストリング表記は、数字で始まり、長さが 4 バイト以上のストリングです。 末尾のブランクを付けることができます。時刻の時部分の先行ゼロは省略可能であり、秒は完全に省略することができます。 秒が省略されている場合は、0 秒が指定されたと見なされます。したがって、13:30 は 13:30:00 に等しくなります。

時刻を示す有効なストリング・フォーマットを、以下の表に示します。各フォーマットは、名前および関連する省略形によって識別されます。

表 6. 時刻のストリング表記フォーマット

フォーマット名	省略形	時刻 フォーマット	例
国際標準化機構 [2]	ISO	hh.mm.ss	13.30.05

表 6. 時刻のストリング表記フォーマット (続き)

フォーマット名	省略形	時刻 フォーマット	例
IBM USA 標準規格	USA	hh:mm AM または PM	1:30 PM
IBM 欧州標準規格	EUR	hh.mm.ss	13.30.05
日本工業規格西暦	JIS	hh:mm:ss	13:30:05
地域別定義	LOC	アプリケーションのテリトリー・コードに依存します。	—

注:

1. ISO、EUR、および JIS フォーマットでは、.ss (もしくは :ss) は省略可能です。
2. 国際標準化機構は、時刻フォーマットを日本工業規格 (西暦) フォーマットと同じフォーマットに変更しました。このため、アプリケーションで現行の国際標準化機構フォーマットが必要な場合は、 JIS を使用してください。
3. USA 時刻ストリング・フォーマットでは、分の指定を省略できます。その場合、00 分と見なされます。したがって、1 PM は 1:00 PM に等しくなります。
4. USA 時刻フォーマットでは、時を 13 以上にすることはできず、 00:00 AM という特殊な場合を除いて、0 にすることはできません。 AM および PM の前にはスペースが 1 個入れられます。 24 時間制の JIS フォーマットを使用した場合、 USA フォーマットと 24 時間制との対応は次のようになります。

   12:01 AM から 12:59 AM は、00:01:00 から 00:59:00 に対応します。

   01:00 AM から 11:59 AM は、01:00:00 から 11:59:00 に対応します。

   12:00 PM (正午) から 11:59 PM は、12:00:00 から 23:59:00 に対応します。

   12:00 AM (深夜) は 24:00:00 に対応し、00:00 AM (深夜) は 00:00:00 に対応します。

タイム・スタンプ・ストリング: タイム・スタンプのストリング表記は、数字で始まり、長さが 16 バイト以上のストリングです。 タイム・スタンプの完全なストリング表示は、 *yyyy-mm-dd-hh.mm.ss.nnnnnn* という形式です。末尾のブランクを付けることができます。タイム・スタンプの月、日、および時の部分の先行ゼロは省略でき、マイクロ秒は切り捨てたり、完全に省略したりできます。マイクロ秒の部分で後続ゼロが省略されている場合、抜けている数字は 0 が指定されたと見なされます。したがって、 1991-3-2-8.30.00 は 1991-03-02-08.30.00.000000 に等しくなります。

また SQL ステートメントは、タイム・スタンプの ODBC ストリング表示を入力値としてのみサポートします。タイム・スタンプの ODBC ストリング表示の形式は、*yyyy-mm-dd hh:mm:ss.nnnnnn* です。

DATALINK 値

DATALINK 値はカプセル化された値であり、データベース以外のロケーションに保管されているファイルへのデータベースからの論理参照を収めています。このカプセル化された値の属性は以下のとおりです。

link type
現在サポートされているリンクのタイプは 'URL' です。

data location
DB2 内でリンクで参照されているファイルのロケーション。これは、URL 形式になります。 この URL に許可されているスキーム名は、以下のとおりです。

- HTTP
- FILE
- UNC

URL の他の部分は、以下のとおりです。

- HTTP、FILE、および UNC スキームのファイル・サーバー名
- ファイル・サーバー内の絶対ファイル・パス名

comment
データ位置属性を含む 最大 200 バイトの記述情報。これは、アプリケーションの特有の使用 (データのロケーションを詳細に識別するか、または別の方法で識別するなど) のためのものです。

URL によるデータ・ロケーション属性を解析しているときに、先行および末尾ブランク文字は切り詰められます。さらに、スキーム名 ('http'、'file'、'unc') およびホストは、大文字小文字を区別しないで、常に英大文字でデータベース内に格納されます。データベースから DATALINK 値を取り出すと、 DATALINK 列が READ PERMISSION DB または WRITE PERMISSION ADMIN で定義される場合には、URL 属性内にアクセス・トークンが組み込まれます。トークンは動的に生成され、データベースに保管される DATALINK 値に永続的に組み込まれているわけではありません。

DATALINK 値に指定されているのがコメント属性だけで、データ・ロケーション属性が空である場合もあります。そのような値が列に保管されていても、当然そのような列にリンクされているファイルはありません。 DATALINK 値のコメントおよびデータ・ロケーション属性の全長は、今のところ、200 バイトまでに制限されています。

ファイルへの DATALINK 参照と、 LOB ファイル参照変数との違いを理解することは大切です。類似点は、その両方にファイルの表記が入っているということです。しかし、以下の点で異なっています。

- DATALINK はデータベースに保存されるので、リンクおよびそのリンクされたファイルにあるデータは両方とも、データベース内のデータの自然な拡張と見なすことができます。
- ファイル参照変数はクライアント上に一時的に存在するものなので、ホスト・プログラムのバッファーの代わりと見なすことができます。

組み込みスカラー関数は、DATALINK 値 (DLVALUE、DLNEWCOPY、DLPREVIOUSCOPY、および DLREPLACECONTENT) の作成と、DATALINK 値 (DLCOMMENT、DLLINKTYPE、DLURLCOMPLETE、DLURLPATH、DLURLPATHONLY、DLURLSCHEME、DLURLSERVER、DLURLCOMPLETEONLY、DLURLCOMPLETEWRITE、および DLURLPATHWRITE) からのカプセル化された値の抽出を目的として提供されています。

関連資料:

- 64 ページの『ID』
- 809 ページの『付録 O. DATALINK のバッカス正規形式 (BNF) 仕様』

XML 値

XML データ型は、XML の内部表記で、このデータ型を入力として受け入れる関数への入力としてのみ使用できます。 XML は、データベース内に保管したり、またはアプリケーションに戻したりすることのできない一時データ型です。

XML データ型の有効な値には、以下のものが含まれます。

- エレメント
- 一群のエレメント
- エレメントのテキスト内容
- 空の XML 値

現在、唯一のサポートされる操作は、 CLOB 値として保管されるストリングに XML 値をシリアライズ (XML2CLOB 関数を使用して) することです。

ユーザー定義タイプ

ユーザー定義のデータ型には次の 3 つのタイプがあります。

- 特殊タイプ
- 構造化タイプ
- 参照タイプ

これらのそれぞれのタイプについて、次の項で説明します。

特殊タイプ

特殊タイプ とは、内部表記を既存のタイプ (その 『ソース』・タイプ) と共用するユーザー定義のデータ型です。しかし、特殊タイプはほとんどの操作で、非互換の別個のタイプと見なされます。たとえば、ピクチャー・タイプ、テキスト・タイプ、音声タイプを定義しようとする場合、これらのタイプのセマンティクスはどれも異なりますが、内部表記としては組み込みデータ型 BLOB を使用します。

次に、AUDIO という名前の特殊タイプを作成する例を示します。

```
CREATE DISTINCT TYPE AUDIO AS BLOB (1M)
```

AUDIO は組み込みデータ型の BLOB と内部表記は同じですが、別個のタイプと見なされます。これにより、AUDIO 用に特別に関数を設定できるようになり、そのような関数は他のどのデータ型 (ピクチャー、テキストなど) の値にも決して適用されないようになります。

特殊タイプは、修飾子付き ID を持っています。 CREATE DISTINCT TYPE、DROP DISTINCT TYPE、または COMMENT ON DISTINCT TYPE ステートメント以外で特殊タイプ名が使用されるとき、スキーマ名によってそれが修飾されていない場合は、 SQL パスを順に調べて、特殊タイプの一致する最初のスキーマが探索されます。

特殊タイプを使うと、そのインスタンスに対しては、明示的に特殊タイプに基づいて定義された関数や演算子しか適用されないようになるため、強力なタイプ識別機能が実現されます。そのため、特殊タイプはそのソース・タイプの関数や演算子を自動的には獲得しません。そのようなものは無意味である可能性があるためです。(たとえば、AUDIO タイプの LENGTH 関数は、そのオブジェクトの長さをバイト単位ではなく秒単位で戻します。)

LONG VARCHAR、LONG VARGRAPHIC、LOB の各タイプ、または DATALINK をソースとする特殊タイプは、ソース・タイプと同じ制限に従います。

しかし、ソース・タイプの特定の関数と演算子が特殊タイプに適用されるように明示的に指定することは可能です。これは、特殊タイプのソース・タイプに対して定義された関数をソースとするユーザー定義関数を定義することによって行うことができます。ソース・タイプとして LONG VARCHAR、LONG VARGRAPHIC、BLOB、CLOB、 DBCLOB、または DATALINK を使用しているもの以外のユーザー定義特殊タイプについては、自動的に比較演算子が生成されます。さらに、ソース・タイプから特殊タイプへ、また特殊タイプからソース・タイプへのキャストをサポートする関数も生成されます。

構造化タイプ

構造化タイプ とは、データベースに定義されている構造を持つユーザー定義のデータ型のことです。これには、名前が付けられている一連の*属性* が入っており、それぞれにデータ型があります。構造化タイプには、一連のメソッド仕様も組み込まれています。

構造化タイプは表、ビュー、または列のタイプとして使用することができます。表またはビューのタイプとして使用する場合、その表またはビューは、それぞれ*タイプ表* または*タイプ・ビュー* となります。タイプ表およびタイプ・ビューの場合、構造化タイプの属性の名前およびデータ型は、タイプ表またはタイプ・ビューの列の名前およびデータ型になります。タイプ表またはタイプ・ビューの行は、構造化タイプのインスタンスの表示と考えることができます。列のデータ型として使用する場合、その列には該当構造化タイプの値 (または、下記のように、そのタイプにサブタイプがあれば、その値) が入ります。構造列オブジェクトの属性を取り出して処理するときには、メソッドを使います。

用語: *スーパータイプ* とは、*サブタイプ* という、他の構造化タイプが定義されている構造化タイプのことです。サブタイプはスーパータイプのすべての属性およびメソッドを継承し、さらに属性およびメソッドを定義することもできます。共通のスーパータイプに関連する構造化タイプのセットは*タイプ階層* と呼ばれ、それより上位のスーパータイプを持たないタイプをそのタイプ階層の*ルート・タイプ* と呼びます。

サブタイプという用語は、タイプ階層において 1 つのユーザー定義の構造化タイプおよびその下にあるすべてのユーザー定義の構造化タイプを指して用いられます。したがって、階層内における構造化タイプ T のサブタイプは、 T と、T の下にあるすべての構造化タイプになります。構造化タイプ T の*厳密な意味でのサブタイプ*とは、タイプ階層で T の下にある構造化タイプのことです。

タイプ階層内での再帰的タイプ定義に対しては、いくつかの制限があります。このため、許可されている特定タイプの再帰的定義を参照するために、簡単な方法を考える必要があります。以下の定義が使われます。

- *直接的な使用*: 以下のいずれか 1 つが当てはまる場合のみ、タイプ **A** は、他のタイプ **B** を直接に使用します。
  1. タイプ **A** に、タイプ **B** の属性がある場合
  2. タイプ **B** が、**A** のサブタイプ、または **A** のスーパータイプである場合
- *間接的な使用*: 以下のいずれかが当てはまる場合、タイプ **A** は、タイプ **B** を間接的に使用します。
  1. タイプ **A** がタイプ **B** を直接に使う場合
  2. タイプ **A** が何らかのタイプ **C** を直接に使用し、タイプ **C** がタイプ **B** を間接的に使う場合

いずれかの属性タイプがそれ自体を直接にまたは間接的に使うように、タイプを定義することはできません。そのような構成を作成する必要がある場合、参照を属性として使うことを考慮してください。たとえば、構造化タイプ属性では、「管理職」が属性タイプ「従業員」である場合に、「管理職」の属性を持つ「従業員」のインスタンスというものはあり得ません。しかし、REF (従業員) のタイプを持つ「管理職」の属性はあり得ます。

他の特定のオブジェクトが、あるタイプを直接または間接的に使っている場合、そのタイプをドロップすることはできません。たとえば、表またはビューの列が、タイプを直接または間接的に使っている場合、タイプをドロップすることはできません。

参照タイプ

参照タイプ は構造化タイプと対になっているタイプです。特殊タイプに似て、参照タイプは組み込みデータ型の 1 つと共通の表記を使用するスカラー・タイプです。この同じ表記はタイプ階層のすべてのタイプで共用されます。参照タイプ表記は、タイプ階層のルート・タイプの作成時に定義されます。参照タイプを使用する場合、構造化タイプはタイプのパラメーターとして指定されます。このパラメーターを、参照の*ターゲット・タイプ* といいます。

参照のターゲットは、通常、タイプ表またはタイプ・ビューの行です。参照タイプを使用する場合、*有効範囲* を定義することができます。有効範囲は、参照値のターゲット行がある表 (*ターゲット表* と呼ばれる) またはビュー (*ターゲット・ビュー* と呼ばれる) を指定します。ターゲット表またはターゲット・ビューは、参照タイプのターゲット・タイプと同じタイプでなければなりません。効力範囲を持つ参照タイプのインスタンスは、タイプ表またはタイプ・ビューの行 (*ターゲット行* と呼ばれる) を固有識別します。

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『DROP ステートメント』
- 154 ページの『CURRENT PATH』
- 90 ページの『文字ストリング』
- 110 ページの『割り当てと比較』

データ型のプロモーション

データ型は、関連するいくつかのデータ型からなるグループに分類されます。そのようなグループの中では、あるデータ型を他のデータ型より優先すると見なす優先順位が存在します。この優先順位を使用すると、あるデータ型を、優先順位がそれより上のデータ型にプロモートすること (*プロモーション*) が可能になります。たとえば、CHAR データ型は VARCHAR に、 INTEGER は DOUBLE-PRECISION にプロモートできますが、 CLOB を VARCHAR にプロモートすることはできません。

データ型のプロモーションは、以下の場合に使用されます。

- 関数解決を実行する場合
- ユーザー定義タイプをキャストする場合
- ユーザー定義タイプを組み込みデータ型に割り当てる場合

次の表 7 に、各データ型の優先順位順のリストを示します。特定のデータ型のプロモート先として可能なデータ型を調べたいとき、この表を使うことができます。この表に示されているとおり、最適の選択は、別のデータ型へプロモートすることではなく、常に同じデータ型です。

表 7. データ型の優先順位表

データ型	データ型優先順位リスト (高いものから順に)
CHAR	CHAR、VARCHAR、LONG VARCHAR、CLOB
VARCHAR	VARCHAR、LONG VARCHAR、CLOB
LONG VARCHAR	LONG VARCHAR、CLOB
GRAPHIC	GRAPHIC、VARGRAPHIC、LONG VARGRAPHIC、DBCLOB
VARGRAPHIC	VARGRAPHIC、LONG VARGRAPHIC、DBCLOB
LONG VARGRAPHIC	LONG VARGRAPHIC、DBCLOB
BLOB	BLOB
CLOB	CLOB
DBCLOB	DBCLOB
SMALLINT	SMALLINT、INTEGER、BIGINT、decimal、real、double
INTEGER	INTEGER、BIGINT、decimal、real、double
BIGINT	BIGINT、decimal、real、double
decimal	decimal、real、double
real	real、double
double	double
DATE	DATE
TIME	TIME
TIMESTAMP	TIMESTAMP
DATALINK	DATALINK
udt	udt (同じ名前) または udt のスーパータイプ
REF(T)	REF(S) (S が T のスーパータイプの場合)

表 7. データ型の優先順位表 (続き)

データ型	データ型優先順位リスト (高いものから順に)

注:

1. 上記の小文字で示したタイプは、以下のように定義されます。
   - decimal = DECIMAL(p,s) または NUMERIC(p,s)
   - real = REAL または FLOAT(*n*)。ここで、*n* は 24 を超えない値。
   - double = DOUBLE、DOUBLE-PRECISION、FLOAT、または FLOAT(*n*)。ここで、*n* は 25 以上。
   - udt = ユーザー定義タイプ

   リストの中のデータ型の短形式および長形式の同義語は、リストの中の同義語と同じであると見なされます。
2. Unicode データベースの場合、以下は、等価のデータ型と見なされます。
   - CHAR および GRAPHIC
   - VARCHAR および VARGRAPHIC
   - LONG VARCHAR および LONG VARGRAPHIC
   - CLOB および DBCLOB

関連資料:

- 166 ページの『関数』
- 107 ページの『データ型間のキャスト』
- 110 ページの『割り当てと比較』

データ型間のキャスト

特定のデータ型の値を別のデータ型へキャストしたり、またはデータ型は同じでも長さ、精度、または位取りが異なるデータ型へキャスト する必要がよく生じます。データ型のプロモーションは、あるデータ型から別のデータ型へのプロモーションにおいて、値を新しいデータ型へキャストすることが必要になる 1 つの例です。別のデータ型へキャストできるデータ型は、ソース・データ型から宛先データ型へ*キャスト可能* であるといいます。

データ型間でのキャストは、 CAST 仕様を使用して明示的に行うことができますが、ユーザー定義タイプに関与した割り当ての過程で暗黙的に行われる場合もあります。また、ソース関数から派生するユーザー定義関数を作成するときは、ソース関数のパラメーターのデータ型が、作成しようとしている関数のデータ型にキャスト可能なものでなければなりません。

組み込みデータ型の間でサポートされているキャストを、 109 ページの表 8 に示します。第 1 列がキャスト・オペランドのデータ型 (ソース・データ型) を表し、上部に横方向に示したデータ型が CAST 指定の目的データ型を表します。

Unicode データベースでは、文字または GRAPHIC ストリングが別のデータ型にキャストされるときに切り捨てが行われる場合、非ブランク文字が切り捨てられる場合に警告が戻されます。この切り捨て動作は、非ブランク文字が切り捨てられる場合にエラーが起こるときの、ターゲットへの文字または GRAPHIC ストリングの割り当てとは異なります。

特殊タイプに関する以下のキャストがサポートされています。

- 特殊タイプ *DT* から、そのソース・データ型 *S* へのキャスト
- 特殊タイプ *DT* のソース・データ型 *S* から、特殊タイプ *DT* へのキャスト
- 特殊タイプ *DT* から、それと同じソース・データ型 *DT* へのキャスト
- データ型 *A* から、特殊タイプ *DT* へのキャスト。ただし、 *A* は特殊タイプ *DT* のソース・データ型 *S* へプロモート可能なもの
- INTEGER から、ソース・データ型が SMALLINT である特殊タイプ *DT* へのキャスト
- DOUBLE から、ソース・データ型が REAL である特殊タイプ *DT* へのキャスト
- VARCHAR から、ソース・データ型が CHAR である特殊タイプ *DT* へのキャスト
- VARGRAPHIC から、ソース・データ型が GRAPHIC である特殊タイプ *DT* へのキャスト
- Unicode データベースの場合、VARCHAR または VARGRAPHIC から、ソース・データ型が CHAR または GRAPHIC である特殊タイプ *DT* へのキャスト

文字タイプへのキャストを実行する際に、FOR BIT DATA を指定することはできません。

構造化タイプの値を何か別のものにキャストすることはできません。 *ST* のスーパータイプに対するすべてのメソッドは、 *ST* に当てはまるので、構造化タイプ *ST*

を、そのスーパータイプのいずれかにキャストすべきではありません。必要な操作が *ST* のサブタイプだけに当てはまる場合、サブタイプ処理式を使用して、*ST* をサブタイプの 1 つとして扱います。

キャストに関与したユーザー定義データ型がスキーマ名によって修飾されていない場合、 *SQL パス* が、ユーザー定義データ型を組み入れられた最初のスキーマをその名前で検出するために使用されます。

参照タイプに関して、以下のキャストがサポートされています。

- 参照タイプ *RT* から、表記データ型 *S* へのキャスト
- 参照タイプ *RT* の表記データ型 *S* から、参照タイプ *RT* へのキャスト
- ターゲット・タイプが *T* である参照タイプ *RT* から、ターゲット・タイプが *S* である参照タイプ *RS* へのキャスト (*S* は *T* のスーパータイプ)
- データ型 *A* から、参照タイプ *RT* へのキャスト (ただし *A* は、参照タイプ *RT* の表記データ型 *S* へプロモート可能なもの)

キャストに関与した参照データ型のターゲット・タイプがスキーマ名によって修飾されていない場合、 *SQL パス* が、ユーザー定義データ型を組み入れられた最初のスキーマをその名前で検出するために使用されます。

表 8. 組み込みデータ型間のサポートされるキャスト

ソース・データ・タイプ ↓ / ターゲット・データ・タイプ →	SMALLINT	INTEGER	BIGINT	DECIMAL	REAL	DOUBLE	CHAR	VARCHAR	LONG VARCHAR	CLOB	GRAPHIC	VARGRAPHIC	LONG VARGRAPHIC	DBCLOB	DATE	TIME	TIMESTAMP	BLOB
SMALLINT	Y	Y	Y	Y	Y	Y	Y	-	-	-	-	-	-	-	-	-	-	-
INTEGER	Y	Y	Y	Y	Y	Y	Y	-	-	-	-	-	-	-	-	-	-	-
BIGINT	Y	Y	Y	Y	Y	Y	Y	-	-	-	-	-	-	-	-	-	-	-
DECIMAL	Y	Y	Y	Y	Y	Y	Y	-	-	-	-	-	-	-	-	-	-	-
REAL	Y	Y	Y	Y	Y	Y	-	-	-	-	-	-	-	-	-	-	-	-
DOUBLE	Y	Y	Y	Y	Y	Y	-	-	-	-	-	-	-	-	-	-	-	-
CHAR	Y	Y	Y	Y	-	-	Y	Y	Y	Y	Y[1]	Y[1]	-	-	Y	Y	Y	Y
VARCHAR	Y	Y	Y	Y	-	-	Y	Y	Y	Y	Y[1]	Y[1]	-	-	Y	Y	Y	Y
LONG VARCHAR	-	-	-	-	-	-	Y	Y	Y	Y	-	-	Y[1]	Y[1]	-	-	-	Y
CLOB	-	-	-	-	-	-	Y	Y	Y	Y	-	-	-	Y[1]	-	-	-	Y
GRAPHIC	-	-	-	-	-	-	Y[1]	Y[1]	-	-	Y	Y	Y	Y	Y[1]	Y[1]	Y[1]	Y
VARGRAPHIC	-	-	-	-	-	-	Y[1]	Y[1]	-	-	Y	Y	Y	Y	Y[1]	Y[1]	Y[1]	Y
LONG VARGRAPHIC	-	-	-	-	-	-	-	-	Y[1]	Y[1]	Y	Y	Y	Y	-	-	-	Y
DBCLOB	-	-	-	-	-	-	-	-	-	Y[1]	Y	Y	Y	Y	-	-	-	Y
DATE	-	Y	Y	Y	-	-	Y	Y	-	-	Y[1]	Y[1]	-	-	Y	-	-	-
TIME	-	Y	Y	Y	-	-	Y	Y	-	-	Y[1]	Y[1]	-	-	-	Y	-	-
TIMESTAMP	-	-	Y	Y	-	-	Y	Y	-	-	Y[1]	Y[1]	-	-	Y	Y	Y	-
BLOB	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	Y

注

- ユーザー定義タイプおよび参照タイプに関してサポートされているキャストについては、この表の前にある説明を参照してください。
- DATALINK タイプにキャストできるのは DATALINK タイプのみです。
- 構造化タイプの値を何か別のものにキャストすることはできません。

[1] キャストは、Unicode データベースの場合にのみサポートされます。

関連資料:

- 184 ページの『式』
- 「*SQL リファレンス 第 2 巻*」の『CREATE FUNCTION ステートメント』
- 154 ページの『CURRENT PATH』
- 105 ページの『データ型のプロモーション』
- 110 ページの『割り当てと比較』

割り当てと比較

SQL の基本的な演算は、割り当てと比較です。割り当て演算は、INSERT、UPDATE、 FETCH、 SELECT INTO、 VALUES INTO および SET 遷移変数ステートメントの実行時に行われます。関数の引き数も、関数の呼び出し時に割り当てられます。比較演算は、MAX、 MIN、 DISTINCT、 GROUP BY、 ORDER BY のような述部およびその他の言語エレメントを組み入れられたステートメントの実行時に行われます。

両方の演算に適用される 1 つの基本的な規則は、関係するオペランドのデータ型は互換でなければならないということです。この互換性規則は集合演算にも適用されます。

割り当て操作の別の基本的な規則は、 NULL 値を入れることができない列や、関連する標識変数がないホスト変数に、 NULL 値を割り当てることができないという規則です。

文字と GRAPHIC データの両方に関与した割り当ておよび比較は、ストリングの 1 つがリテラルであるときにのみサポートされます。

以下は、割り当ておよび比較演算のためのデータ型の互換性を示す互換性マトリックスです。

表 9. 割り当てと比較におけるデータ型の互換性

オペランド	バイナリー整数	10 進数	浮動小数点数	文字ストリング	GRAPHIC ストリング	DATE	TIME	TIMESTAMP	バイナリー・ストリング	UDT
バイナリー整数	Yes	Yes	Yes	No	No	No	No	No	No	[2]
10 進数	Yes	Yes	Yes	No	No	No	No	No	No	[2]
浮動小数点数	Yes	Yes	Yes	No	No	No	No	No	No	[2]
文字ストリング	No	No	No	Yes	Yes [6,7]	[1]	[1]	[1]	No [3]	[2]
GRAPHIC ストリング	No	No	No	Yes [6,7]	Yes	[1]	[1]	[1]	No	[2]
DATE	No	No	No	[1]	[1]	Yes	No	No	No	[2]
TIME	No	No	No	[1]	[1]	No	Yes	No	No	[2]
TIMESTAMP	No	No	No	[1]	[1]	No	No	Yes	No	[2]
バイナリー・ストリング	No	No	No	No [3]	No	No	No	No	Yes	[2]
UDT	[2]	[2]	[2]	[2]	[2]	[2]	[2]	[2]	[2]	Yes

表 9. 割り当てと比較におけるデータ型の互換性 (続き)

オペランド	バイナリー整数	10 進数	浮動小数点数	文字ストリング	GRAPHIC ストリング	DATE	TIME	TIME STAMP	バイナリー・ストリング	UDT

[1] DATETIME とストリングの互換性は、割り当てと比較に限定されます。

- DATETIME 値は、ストリング列とストリング変数に割り当てることができます。
- 日付の有効なストリング表記は、DATE 列に割り当てるか、または DATE と比較できます。
- 時刻の有効なストリング表記は、TIME 列に割り当てるか、または TIME と比較できます。
- タイム・スタンプの有効なストリング表記は、 TIMESTAMP 列に割り当てるか、または TIMESTAMP と比較できます。

(GRAPHIC ストリング・サポートは、Unicode データベースの場合にのみ使用可能です。)

[2] ユーザー定義特殊タイプの値は、同じユーザー定義特殊タイプで定義された値とのみ比較できます。一般に、特殊タイプの値とそのソース・データ型との間では割り当てがサポートされます。ユーザー定義構造化タイプは比較することができません。また、同じ構造化タイプまたはそのスーパータイプのいずれかのオペランドにのみ、割り当てることができます。さらに詳しい情報については、 119 ページの『ユーザー定義タイプの割り当て』を参照してください。

[3] これは、 FOR BIT DATA 属性で定義された文字ストリングもバイナリー・ストリングと互換性がないことを意味します。

[4] DATALINK オペランドは、別の DATALINK オペランドにのみ割り当てることができます。列が NO LINK CONTROL と定義されている場合、またはファイルは存在しているがまだファイル・リンク制御されていない場合、DATALINK 値はその列にしか割り当てることができません。

[5] 参照タイプの割り当ておよび比較については、 119 ページの『参照タイプの割り当て』 および 124 ページの『参照タイプの比較』 を参照してください。

[6] Unicode データベースの場合にのみサポートされます。

[7] ビット・データと GRAPHIC ストリングは互換性がありません。

数値割り当て

数値割り当ての基本的な規則は、 10 進数または整数の整数部分は決して切り捨てられることがないということです。宛先数値の位取りが割り当て元の数値の位取りより小さい場合は、 10 進数の小数部の超過桁が切り捨てられます。

10 進数または整数から浮動小数点への割り当て: 浮動小数点数は、実数の近似値です。したがって、10 進数または整数が浮動小数点の列または変数に割り当てられた場合、その結果が元の数値と異なってくる可能性があります。

浮動小数点または 10 進数の整数への割り当て: 浮動小数点または 10 進数が、整数の列または整数変数に割り当てられた場合、その数の小数部分が失われます。

10 進数から 10 進数への割り当て: 10 進数が 10 進数の列または変数に割り当てられる場合、その数値は、必要に応じて宛先の精度および位取りに変換されます。必要な数だけ先行ゼロが追加または除去されます。また、数値の小数部分では、必要な数の後続ゼロが追加されるか、または必要な数だけ後続ブランクが除去されます。

整数から 10 進数への割り当て: 整数を 10 進数の列または変数に割り当てるときは、まず数値が一時的な 10 進数へ変換された後、必要であれば、宛先の精度と位取りに変換されます。一時的な 10 進数の精度と位取りは、短整数では 5,0、長精度整数では 11,0、 64 ビット整数では 19,0 です。

浮動小数点数から 10 進数への割り当て: 浮動小数点数を 10 進数に割り当てるときは、まず数値が精度 31 の一時的な 10 進数へ変換された後、必要なら、宛先の精度と位取りまで切り捨てられます。この変換では、数値は精度 31 の 10 進数に (浮動小数点数算術演算を使って) 丸められます。その結果、$0.5*10^{-31}$ 未満の数値は 0 になります。位取りは、有効数字を消失させることなく数値全体を表現できるような最大可能値になります。

ストリング割り当て

割り当てには以下の 2 つのタイプがあります。

- 値を列またはルーチン・パラメーターに割り当てる場合の*ストレージ割り当て*
- 値をホスト変数に割り当てる場合の*検索割り当て*

ストリング割り当てに関する規則は、割り当てのタイプによって異なります。

ストレージ割り当て: 基本的な規則は、列またはルーチン・パラメーターに割り当てられるストリングの長さが、列またはルーチン・パラメーターの長さ属性を超えてはならないということです。ストリングの長さが列またはルーチン・パラメーターの長さ属性を超えた場合は、以下の処置が取られます。

- ストリングは、宛先の列またはルーチン・パラメーターの長さ属性に適合するように、 (長ストリングを除くすべてのストリング・タイプから) 後続ブランクが切り捨てられた上で割り当てられます。
- 以下の場合にはエラー (SQLSTATE 22001) になります。
  - LONG ストリング以外のストリングからブランク以外の文字が切り捨てられるとき
  - LONG ストリングから任意の文字 (またはバイト) が切り捨てられるとき

ストリングが固定長列に割り当てられる際に、ストリングの長さがターゲットの長さ属性よりも短い場合、ストリングの右端に必要な数の 1 バイト、2 バイト、または UCS-2 のブランクが埋め込まれます。埋め込み文字は、FOR BIT DATA 属性で定義されている列の場合も含めて、常にブランクです。 (UCS-2 は、いくつかの SPACE 文字を異なるプロパティーで定義します。 Unicode データベースの場合、データベース・マネージャーは、常に、 UCS-2 ブランクとして位置 x'0020' にある ASCII SPACE を使用します。 EUC データベースの場合、位置 x'3000' にある IDEOGRAPHIC SPACE は、埋め込み GRAPHIC ストリングに使用されます。)

検索割り当て: ホスト変数に割り当てられるストリングの長さは、ホスト変数の長さ属性より長くてもかまいません。ストリングがホスト変数に割り当てられるときに、ストリングの長さが変数の長さ属性より長ければ、ストリングの右側から文字 (またはバイト) が必要な数だけ切り捨てられます。この場合には警告 (SQLSTATE 01004) が戻され、 SQLCA の SQLWARN1 フィールドに値 'W' が割り当てられます。

さらに、標識変数があってその値のソースが LOB でない場合は、その標識変数はストリングの元の長さに設定されます。

文字ストリングが固定長変数に割り当てられる際に、ストリングの長さがターゲットの長さ属性よりも短い場合、ストリングの右端に必要な数の 1 バイト、2 バイト、または UCS-2 のブランクが埋め込まれます。埋め込み文字は、FOR BIT DATA 属性で定義されているストリングの場合も含めて、常にブランクです。(UCS-2 は、いくつかの SPACE 文字を異なるプロパティーで定義します。 Unicode データベースの場合、データベース・マネージャーは、常に、 UCS-2 ブランクとして位置 x'0020' にある ASCII SPACE を使用します。 EUC データベースの場合、位置 x'3000' にある IDEOGRAPHIC SPACE は、埋め込み GRAPHIC ストリングに使用されます。)

C の NUL 終止符ホスト変数の検索割り当ては、 PREP または BIND コマンドに指定されたオプションに基づいて処理されます。

ストリング割り当てに関する変換規則: 列またはホスト変数に割り当てられる文字ストリングまたは GRAPHIC ストリングは、必要であれば、まず割り当て先のコード・ページに変換されます。文字変換が必要になるのは、以下の条件がすべて真の場合だけです。

- コード・ページが異なる。
- ストリングが NULL でも空でもない。
- どちらのストリングのコード・ページ値も 0 (FOR BIT DATA) でない。

Unicode データベースの場合、 GRAPHIC 列に文字ストリングを割り当てることができ、文字カラムに GRAPHIC ストリングを割り当てることができます。

文字ストリング割り当てに関する MBCS の考慮事項: 1 バイト文字とマルチバイト文字の両方を入れることのできる文字ストリングを割り当てる場合には、いくつかの考慮事項があります。このような考慮事項は、FOR BIT DATA と定義されているものを含めて、すべての文字ストリングに適用されます。

- ブランクの埋め込みは、常に単一バイトのブランク文字 (X'20') を使用して行われます。
- ブランクの切り捨ては、常に単一バイトのブランク文字 (X'20') に基づいて行われます。切り捨てに関しては、2 バイトのブランク文字はその他の文字と同様に扱われます。
- 文字ストリングをホスト変数に割り当てる場合に、割り当て先のホスト変数にソース・ストリング全体を収めるだけの長さがないなら、 MBCS 文字のフラグメント化が発生します。 MBCS 文字がこのようにフラグメント化される場合は、MBCS 文字フラグメントの各バイトがターゲットで単一バイトのブランク文字 (X'20') に設定されます。それ以外のバイトに関してはソースからの移動は行われないで、 SQLWARN1 が 'W' に設定されて切り捨ての発生を示します。 MBCS 文字のフラグメント化に関するこの処理は、文字ストリングが FOR BIT DATA と定義されている場合にも同じであることに注意してください。

GRAPHIC ストリング割り当てに関する DBCS の考慮事項: GRAPHIC ストリング割り当ては、文字ストリングに似た方法で処理されます。非 Unicode データベースの場合、GRAPHIC ストリング・データ型が互換であるのは他の GRAPHIC ストリング・データ型とだけであり、数値、文字ストリング、日付/時刻データ型とは互換ではありません。 Unicode データベースの場合、GRAPHIC ストリングのデータ型は、文字ストリングのデータ型と互換性があります。

GRAPHIC ストリング値が GRAPHIC ストリング列に割り当てられる場合、その値の長さがその列の長さを超えてはなりません。

GRAPHIC ストリング値 (「ソース」・ストリング) を固定長 GRAPHIC ストリング・データ型
(「割り当て先」、列またはホスト変数) に割り当てる場合に、ソース・ストリングの長さが割り当て先より短いなら、割り当て先には、ソース・ストリングのコピーの右端に、値の長さが割り当て先の長さに等しくなるために必要な数の 2 バイト・ブランク文字を埋め込んだものが入れられます。

GRAPHIC ストリング値を GRAPHIC ストリングのホスト変数に割り当てる場合に、ソース・ストリングの長さがホスト変数の長さよりも長いなら、ホスト変数には、ソース・ストリングのコピーの右端から、値の長さがホスト変数の長さと等しくなるために必要な数の 2 バイト・ブランク文字を切り捨てたものが入れられます。 (このシナリオでは、切り捨てにおいて 2 バイト文字の二分化を考慮する必要はありません。二分化が発生したなら、それはソース値または割り当て先のホスト変数のどちらかで GRAPHIC ストリング・データ型の定義に異常があるということです。) SQLCA の警告フラグ SQLWARN1 が 'W' に設定されます。標識変数が指定されていれば、標識変数にはソース・ストリングの元の長さ (2 バイト文字の文字数) が入れられます。しかし、DBCLOB の場合は、標識変数に元の長さは入れられません。

C の NUL 終止符ホスト変数 (wchar_t を使って宣言されたもの) の検索割り当ては、 PREP または BIND コマンドに指定されたオプションに基づいて処理されます。

日時割り当て

日付/時刻の割り当てに関する基本的な規則は、DATE、TIME、または TIMESTAMP 値は、データ型の一致する列 (DATE、 TIME、 TIMESTAMP のいずれか)、または、固定長または可変長のストリング列変数かストリング列にしか割り当てできないということです。 LONG VARCHAR、CLOB、LONG VARGRAPHIC、DBCLOB または BLOB の変数または列に割り当てることはできません。

日付/時刻値をストリング変数またはストリング列に割り当てるときは、ストリング表記に自動的に変換されます。日付、時刻、タイム・スタンプのどの部分からも先行ゼロが省略されることはありません。割り当て先で必要な長さは、ストリング表記のフォーマットによって異なります。割り当て先の長さが必要よりも長く、割り当て先が固定長ストリングである場合は、割り当て先の右端にブランクが埋め込まれます。割り当て先の長さが必要よりも短い場合は、関係する日付/時刻値のタイプと割り当て先のタイプによって結果が異なります。

宛先がホスト変数である場合、以下の規則が適用されます。

- **DATE の場合:** 変数の長さが 10 文字未満であればエラーが起こります。
- **TIME の場合:** USA フォーマットでは 8 文字未満の長さの変数を使用できません。その他のフォーマットでは 5 文字未満にすることはできません。

ISO または JIS フォーマットを使用しホスト変数の長さが 8 文字未満の場合、時刻の 2 番目の部分は結果から省略され、標識変数があれば、その変数に割り当てられます。 SQLCA の SQLWARN1 フィールドに、省略処理を示す値が設定されます。

- **TIMESTAMP の場合:** ホスト変数が 19 文字未満であればエラーになります。長さが 19 文字以上 26 文字未満の場合、値のマイクロ秒部分の末尾桁が省略されます。 SQLCA の SQLWARN1 フィールドに、省略処理を示す値が設定されます。

DATALINK 割り当て

値を DATALINK 列に割り当てると、値のリンケージ属性が指定されていないか、あるいはその列に NO LINK CONTROL が定義されていない限り、ファイルへのリンクが確立されます。リンクされている値がその列にすでにある場合、そのファイルへのリンクは解除されます。リンクされている値がすでにある場合に、NULL 値を割り当てることによっても、その元の値に関連付けられているファイルへのリンクを解除することができます。

すでに列にあるデータの位置と同じ位置をアプリケーションが指定している場合、そのリンクが保存されます。このようになる理由には、以下のいくつかがあります。

- コメントが変更されるため。
- 表がデータ・リンク調整不能 (DRNP) 状態の場合、その表でのリンクは列と同じリンケージ属性を指定することにより、元に戻すことができます。
- 列が WRITE PERMISSION ADMIN で定義され、ファイル内容が変更される場合、同じデータ位置を持つ DLURLNEWCOPY 関数を使用して構成された DATALINK 値を提供することにより、リンクの新規バージョンを確立できます。
- 列が WRITE PERMISSION ADMIN で定義され、ファイル内容が変更されますが、変更をバックアウトする必要がある場合、同じデータ位置を持つ DLURLPREVIOUSCOPY 関数を使用して作成された DATALINK 値を提供することにより、リンクの既存バージョンを元に戻すことができます。
- 参照ファイルの内容は、DLURLREPLACECONTENT スカラー関数で指定された別のファイルで置き換えられます。

DATALINK 値は、以下のいずれかの方法で列に割り当てることができます。

- DLVALUE スカラー関数を使用して、新しい DATALINK 値を作成し、その値を列に割り当てることができます。値に入っているのがコメントだけか、あるいは URL がまったく同じでない場合には、この割り当てを行うことによりファイルがリンクされます。
- CLI 関数 SQLBuildDataLink を使用して、 CLI パラメーターで DATALINK 値を構成することができます。その後、この値を列に割り当てることができます。値に入っているのがコメントだけか、あるいは URL がまったく同じでない場合には、この割り当てを行うことによりファイルがリンクされます。
- DLURLNEWCOPY スカラー関数を使用して、DATALINK 値を作成し、その値を列に割り当てることができます。作成された DATALINK 値によって参照されるデータ位置は、列内にすでに存在するデータ位置と同じでなければなりません。UPDATE ステートメントを使用して割り当てを行うと、ファイルへのリンクが再確立されます。列が RECOVERY YES に定義される場合、ファイルのバックア

ップがとられます。このタイプの割り当ては、ファイルが更新されたことをデータベースに通知するために使用されます。したがって、データベースは、新規ファイルに気付き、ファイルへの新規リンクを再確立します。

- DLURLPREVIOUSCOPY スカラー関数を使用して、DATALINK 値を作成し、その値を列に割り当てることができます。作成された DATALINK 値によって参照されるデータ位置は、列内にすでに存在するデータ位置と同じでなければなりません。 UPDATE ステートメントを使用して割り当てを行うと、リンクが元に戻されます。また、列が RECOVERY YES に定義される場合、アーカイブから以前のバージョンにファイルをリストアします。このタイプの割り当ては、以前にコミットされたバージョン以降に行われたファイルへの変更をバックアウトするために使用されます。
- DLURLREPLACECONTENT スカラー関数を使用して、新しい DATALINK 値を作成し、その値を列に割り当てることができます。割り当てを行うと、ファイルのみにリンクされるだけでなく、 DLURLREPLACECONTENT スカラー関数に指定された別のファイルの内容に置き換えられます。

値を DATALINK 列に割り当てる場合、以下のエラー条件では SQLSTATE 428D1 が戻されます。

- データ位置 (URL) フォーマットが無効です (理由コード 21)
- ファイル・サーバーがこのデータベースに登録されていません (理由コード 22)
- 無効なリンク・タイプが指定されています (理由コード 23)
- コメントまたは URL の長さが無効です (理由コード 27)

  URL パラメーターまたは関数結果のサイズは入力と出力のいずれでも同じで、DATALINK 列の長さに束縛されていることに注意してください。ただし、URL 値にアクセス・トークンが付加されて返される場合があります。このことが起こりうる状態では、アクセス・トークンと、DATALINK 列の長さのための十分の記憶スペースが出力位置になければなりません。したがって、入力で提供される (完全に展開された書式の) 注釈と URL の実際の長さには、出力ストレージ・スペースに合わせるための制限が加えられます。制限された長さを超えると、このエラーが生じます。
- 入力データ位置には、有効な書き込みトークンが入っていません (理由コード 32)。

  割り当てでは、有効な書き込みトークンがデータ位置に組み込まれている必要があります。これは、列が WRITE PERMISSION ADMIN REQUIRING TOKEN FOR UPDATE に定義されているときにのみ適用され、 DLURLNEWCOPY または DLURLPREVIOUSCOPY スカラー関数によって DATALINK 値が作成されます。一方、ユーザーは、DATALINK 列が WRITE PERMISSION ADMIN NOT REQUIRING TOKEN FOR UPDATE に定義されている書き込みトークンを提供するためのオプションを持っています。ただし、トークンが有効でない場合、同じエラーが起こります。

  このエラーは、値 '1' が 2 番目の引き数に指定されている DLURLNEWCOPY または DLURLPREVIOUSCOPY スカラー関数を使用して DATALINK 値を作成しますが、値に有効な書き込みトークンが入っていないときにも起こることがあります。

- DLURLPREVIOUSCOPY スカラー関数によって作成された DATALINK 値は、WRITE PERMISSION ADMIN および RECOVERY YES に定義された DATALINK 列にのみ割り当てることができます (理由コード 33)。
- DLURLNEWCOPY または DLURLPREVIOUSCOPY スカラー関数によって作成された DATALINK 値が、列内にすでに存在する値と一致しません (理由コード 34)。
- DLURLNEWCOPY または DLURLPREVIOUSCOPY スカラー関数によって作成された DATALINK 値は、新規値を割り当てるために INSERT ステートメント内で使用できません (理由コード 35)。
- DLURLNEWCOPY スカラー関数によって作成された DATALINK 値は、 WRITE PERMISSION BLOCKED で定義された DATALINK 列に割り当てることはできません (理由コード 39)。
- DLURLNEWCOPY または DLURLPREVIOUSCOPY スカラー関数によって作成された同じ DATALINK 値は、同じトランザクション内で何度も割り当てることはできません (理由コード 41)。
- DLURLREPLACECONTENT スカラー関数によって作成された DATALINK 値は、 2 番目の引き数が空ストリングまたは NULL 値である場合にのみ、 NO LINK CONTROL で定義された DATALINK 列に割り当てることができます (理由コード 42)。
- DLREPLACECONTENT スカラー関数に指定された置換ファイルのリンク解除操作がコミットされていません (理由コード 43)。
- DLREPLACECONTENT スカラー関数に指定された置換ファイルは、別の置換プロセスで使用されています (理由コード 44)。
- DATALINK 参照ファイルは、別の操作で置換ファイルとして使用されています (理由コード 45)。
- DLREPLACECONTENT スカラー関数に指定された置換ファイルのフォーマットが有効ではありません (理由コード 46)。
- DLREPLACECONTENT スカラー関数に指定された置換ファイル値が、ディレクトリーまたはシンボリック・リンクになることができません (理由コード 47)。
- DLREPLACECONTENT スカラー関数に指定された置換ファイルは、データベースにリンクされています (理由コード 48)。
- DLREPLACECONTENT スカラー関数に指定された置換ファイルを、データ・リンク・ファイル・マネージャーが見つけることができません (理由コード 49)。
- 書き込みトークンがデータ位置値に入っている DLURLNEWCOPY スカラー関数によって作成された DATALINK 値は、 WRITE PERMISSION ADMIN をもつ DATALINK 列にのみ割り当てることができます (理由コード 50)。

割り当てによりリンクの作成も行う場合、以下のエラーが発生する場合があります。

- ファイル・サーバーは現在使用できません (SQLSTATE 57050)。
- ファイルがありません (SQLSTATE 428D1、理由コード 24)。
- ファイルはすでにほかの列にリンクされています (SQLSTATE 428D1、理由コード 25)。

ほかのデータベースへのリンクでも、このエラーは発生することに注意してください。

- 参照されているファイルにアクセスできず、リンクすることができません (理由コード 26)。
- データ位置に組み込まれている書き込みトークンが、ファイルをオープンするために使用された書き込みトークンと一致していません (SQLSTATE 428D1、理由コード 36)。
- DATALINK 値の参照ファイルが更新進行中の状態にあります (SQLSTATE 428D1、理由コード 37)。
- DATALINK 値の参照ファイルの以前のアーカイブ・コピーが使用不可です (SQLSTATE 428D1、理由コード 40)。

さらに、割り当てにより既存のリンクを除去する場合、以下のエラーが発生する場合があります。

- ファイル・サーバーは現在使用できません (SQLSTATE 57050)。
- 参照保全制御が指定されているファイルが、データ・リンク・ファイル・マネージャーに従った正しい状態ではありません (SQLSTATE 58004)。
- DATALINK 値の参照ファイルが更新進行中の状態にあります (SQLSTATE 428D1、理由コード 37)。

(DLURLCOMPLETEWRITE または DLURLPATHWRITE スカラー関数を使用して) 書き込みアクセス用の DATALINK 値を検索しているときに、 DATALINK 列が WRITE PERMISSION ADMIN で定義される場合、ディレクトリー・アクセス特権がファイル・サーバー (データ・リンク・ファイル・マネージャー) でチェックされます。照会を発行するユーザーは、書き込みトークンが生成され、戻り DATALINK 値に組み込まれる前に、指定されたディレクトリーの下のファイルへの書き込み権限を持っていなければなりません。ユーザーが書き込み権限を持っていない場合、書き込みトークンは生成されず、SELECT ステートメントは失敗します (SQLSTATE 42511、理由コード 1)。

DATALINK 値の一部分を、スカラー関数 (DLLINKTYPE または DLURLPATH など) のアプリケーションの後のホスト変数に割り当てることができます。

通常、検索時にファイル・サーバーへのアクセスは試みられないことに注意してください。そのため、それ以降にファイル・システム・コマンドを使ってファイル・サーバーにアクセスを試みても、失敗する可能性があります。パスに関連した接頭部名を決定するのに、ファイル・サーバーにアクセスすることが必要になる場合があります。これはファイル・システムのマウント・ポイントを移動する場合に、ファイル・サーバーで変更することができます。サーバー上のファイルに始めてアクセスしたときに、必要な値がファイル・サーバーから検索されてデータベース・サーバーのキャッシュに置かれます。そして、それ以降の DATALINK 値の検索はそのファイル・サーバーで行われます。ファイル・サーバーにアクセスできない場合には、エラーが戻されます (SQLSTATE 57050)。

DATALINK 値を検索するためにスカラー関数 DLURLCOMPLETEWRITE または DLURLPATHWRITE を使用する場合、ユーザーのパス上にディレクトリー・アクセ

ス特権を決定するために、ファイル・サーバーにアクセスすることが必要になる場合があります。ファイル・サーバーにアクセスできない場合には、エラーが戻されます (SQLSTATE 57050)。

DATALINK の値を検索する場合、データベース・サーバーでのファイル・サーバーの登録がチェックされ、そのファイル・サーバーがまだそのデータベース・サーバーに登録されていることが確認されます (SQLSTATE 55022)。さらに、DATALINK 値を検索する場合に警告が戻される場合がありますが、それはその表が調整ペンディングまたは調整不能状態のためです (SQLSTATE 01627)。

ユーザー定義タイプの割り当て

ユーザー定義タイプをホスト変数へ割り当てる場合には、他の割り当てで使用される規則とは異なる規則が適用されます。

特殊タイプ: ホスト変数への割り当ては、特殊タイプのソース・タイプに基づいて行われます。つまり、以下の規則に従います。

- 割り当ての右辺に指定する特殊タイプの値は、その特殊タイプのソース・タイプが割り当ての左辺に指定するホスト変数に割り当て可能な場合にのみ、ホスト変数へ割り当てられます。

割り当ての宛先が、特殊タイプに基づく列である場合、ソース・データ型は、ターゲット・データ型へキャスト可能でなければなりません。

構造化タイプ: ホスト変数に対する割り当ては、ホスト変数の宣言済みタイプに基づきます。つまり、以下の規則に従います。

割り当ての右辺に指定する構造化タイプの値は、宣言済みタイプのホスト変数が構造化タイプ、または構造化タイプのスーパータイプである場合にのみ、左辺のホスト変数へ割り当てられます。

割り当ての宛先が、構造化タイプの列である場合、ソース・データ型は、ターゲット・データ型、またはターゲット・データ型のサブタイプでなければなりません。

参照タイプの割り当て

ターゲット・タイプ *T* を指定している参照タイプは、ターゲット・タイプ *S* を指定している参照タイプでもある参照タイプ列に割り当てることができます (*S* は *T* のスーパータイプ)。有効範囲が指定されている参照列または変数に割り当てが行われる場合、割り当てられている実際の値が、有効範囲で定義されているターゲット表またはターゲット・ビューに存在することを確認するためのチェックは行われません。

ホスト変数への割り当ては、参照タイプの表示タイプに基づいて行われます。つまり、以下の規則に従います。

- 割り当ての右側に指定する参照タイプの値は、ホスト参照変数の左側にも割り当て可能ですが、それはこの参照タイプの表示タイプがこのホスト変数に割り当て可能な場合だけです。

割り当てのターゲットが列で、その割り当ての右側にホスト変数が指定されている場合、そのホスト変数はそのターゲット列の参照タイプに明示的にキャストされなければなりません。

数値比較

数値は代数的に、つまり符号を考慮して比較されます。たとえば、-2 は +1 より小さい値として扱われます。

一方が整数で、もう一方が 10 進数の場合、 10 進数に変換された整数の一時コピーが比較に使用されます。

位取りの異なる 10 進数を比較する場合、比較は、一方の数値の小数部分が、他方の数値の小数部分と同じ桁数になるように、後続ゼロを使って拡張されたその数値の一時コピーを使用して行われます。

一方が浮動小数点数で、他方が整数か 10 進数の場合、この後者の数値を倍精度浮動小数点数に変換したものの一時コピーが比較に使用されます。

2 つの浮動小数点値が等しいのは、正規形のビット構成が同一の場合のみです。

ストリングの比較

文字ストリングは、データベースの作成時に指定された照合シーケンスに従って比較されます。ただし、FOR BIT DATA 属性の文字ストリングは例外で、そのような文字ストリングは常にビット値に従って比較されます。

長さの異なる文字ストリングを比較する場合に、 指定された照合シーケンスが UCA (Unicode Collation Algorithm) タイプのものでないと、短い方のストリングの右端に、長い方のストリングの長さになるまで 1 バイト・ブランクを埋め込んだものの論理コピーが比較に使用されます。この論理的な拡張は、FOR BIT DATA のタグの付いたものも含め、すべての文字ストリングに対して行われます。 照合シーケンスが UCA タイプであると、どんな文字を埋め込んでも比較結果に影響を与えるので、短いほうのストリングの右側で埋め込みは行われません。 UCA を使った比較は、コード・ポイントに対してではなく、ストリング内の各文字の 1 つ以上の重みに対して行われます。 UCA の詳細は、 Unicode Consortium Web サイト の Unicode Technical Standard #10 に記載されています。以下の解説は、UCA タイプではない照合シーケンスを対象としています。

文字ストリング (FOR BIT DATA のタグが付けられた文字ストリングを除く) は、データベースの作成時に指定された照合シーケンスに従って比較されます。たとえば、データベース・マネージャーによって指定されるデフォルトの照合シーケンスは、同じ文字の小文字と大文字に同じ重みを与えています。データベース・マネージャーは、完全に同一のストリングだけが相互に等しいものとして扱われるようにするために、 2 つの比較を実行します。まず、ストリングがデータベースの照合シーケンスに従って比較されます。ストリングの文字の重みが等しい場合、次の判断基準として、実際のコード・ポイント値に基づいてストリングを比較します。

2 つのストリングは、両方が空であるか、または対応するすべてのバイト数が等しい場合には、等しくなります。どちらかのオペランドが NULL 値の場合の結果は不明です。

基本比較演算子 (=、<>、<、>、<=、および >=) を使用する比較演算では、長ストリングおよび LOB ストリングはサポートされません。このようなストリングの比較は、 LIKE 述部と POSSTR 関数を使用した比較でサポートされています。

長ストリングおよび LOB ストリングのうち 4 000 バイト以下の部分は、 SUBSTR と VARCHAR のスカラー関数を使用して比較できます。たとえば、以下のような列を考えてみます。

```
MY_SHORT_CLOB   CLOB(300)
MY_LONG_VAR     LONG VARCHAR
```

この場合、以下の演算は有効です。

```
WHERE VARCHAR(MY_SHORT_CLOB) > VARCHAR(SUBSTR(MY_LONG_VAR,1,300))
```

例:

以下の例で、'A'、'Á'、'a'、および 'á' のコード・ポイント値はそれぞれ、X'41'、X'C1'、X'61'、および X'E1' です。

'A'、'Á'、'a'、'á' という文字の重みが 136、139、135、138 である照合シーケンスを考えてみます。このような場合は以下のようになります。

```
'a' < 'A' < 'á' < 'Á'
```

今度は D1、D2、D3、および D4 という 4 つの DBCS 文字を例にとって考えてみます。これらの文字はそれぞれ 0xC141、0xC161、0xE141、および 0xE161 というコード・ポイントを持っています。これらの DBCS 文字が CHAR 列に入っている場合、各文字のバイトが持っている照合重みに従った順序でソートされます。最初の 2 つのバイトの重みは 138 と 139 であるため、 D3 と D4 は D2 と D1 よりも前に来ます。続く 2 つのバイトの重みは 135 と 136 であるため、順序は以下のようになります。

```
D4 < D3 < D2 < D1
```

ただし、比較する値に FOR BIT DATA 属性がある場合や、これらの DBCS 文字が GRAPHIC 列に格納された場合は、照合重みは無視され、これらの文字が持っているコード・ポイントに従って文字が比較されます。以下のようになります。

```
'A' < 'a' < 'Á' < 'á'
```

DBCS 文字はコード・ポイントの順序でソートされます。以下のようになります。

```
D1 < D2 < D3 < D4
```

次に 'A'、'Á'、'a'、'á' という文字が、 74、75、74、および 75 の (ユニークでない) 重みを持つ照合シーケンスを考えてみます。照合重みだけに注目すると (第 1 のパス)、 'a' は 'A' に等しく、'á' は 'Á' に等しいですが、決着を付けるために文字のコード・ポイントを使用すると (第 2 のパス)、以下のようになります。

```
'A' < 'a' < 'Á' < 'á'
```

CHAR 列に入っている DBCS 文字は、最初は重みに従ったバイトの順序 (第 1 パス) でソートされます。それでも決着がつかない場合は、コード・ポイントに従ったバイトの順序 (第 2 パス) でソートされます。最初の 2 つのバイトは重みが同じであるため、コード・ポイント (0xC1 と 0xE1) で決着を付けることになります。結果として、文字 D1 と D2 は文字 D3 と D4 の前にソートされます。続く 2 つのバイトもこれと同じように比較されます。最終的な結果は以下のようになります。

```
D1 < D2 < D3 < D4
```

ここでも、比較する値に FOR BIT DATA 属性がある場合や、これらの DBCS 文字が GRAPHIC 列に格納された場合は、照合重みは無視され、これらの文字が持っているコード・ポイントに従って文字が比較されます。以下のようになります。

```
D1 < D2 < D3 < D4
```

この例では、照合重みが使用されたときと同じ結果が戻されていますが、実際の場面でいつもそのようになるとは限りません。

比較の際の変換規則: 2 つのストリングを比較する場合、必要なら、一方のストリングがまずもう一方のストリングのコード化スキームおよびコード・ページに変換されます。

結果の順序付け: 結果のソートが必要な場合、 120 ページの『ストリングの比較』で説明されているストリング比較規則に基づいて順序付けが行われます。比較はデータベース・サーバー側で実行されます。クライアント・アプリケーションに結果が戻される時点で、コード・ページ変換が実行されることがあります。後から行われるこのようなコード・ページ変換は、サーバーの決定した結果セットの順序には影響しません。

ストリング比較に関する MBCS の考慮事項: SBCS/MBCS 混合文字ストリングは、データベースの作成時に指定された照合シーケンスに従って比較されます。デフォルト (SYSTEM) 照合シーケンスで作成されたデータベースの場合、 1 バイトの ASCII 文字はすべて正しい順序で保管されますが、 2 バイト文字は必ずしもコード・ポイントの順序になっているとは限りません。 IDENTITY 順序で作成されたデータベースの場合、 2 バイト文字はすべてコード・ポイントの順序で保管され、1 バイトの ASCII 文字も同様にコード・ポイントの順序で保管されます。COMPATIBILITY 順序で作成されたデータベースの場合、ほとんどの 2 バイト文字について正しくソートを行い、 ASCII についてもほぼ正しい、中間的な順序が使用されます。これは、DB2 バージョン 2 ではデフォルトの照合表でした。

混合文字ストリングはバイトごとに比較されます。混合ストリング内に現われるマルチバイト文字では通常とは異なる結果になる場合がありますが、これは個々のバイトが別々に扱われるためです。

例:

この例で、'A'、'B'、'a'、および 'b' の 2 バイト文字のコード・ポイント値はそれぞれ、 X'8260'、X'8261'、X'8281'、および X'8282' です。

コード・ポイント X'8260'、X'8261'、X'8281'、および X'8282' の重みがそれぞれ 96、65、193、および 194 である照合シーケンスを考えてみます。 この場合は以下のようになります。

```
'B' < 'A' < 'a' < 'b'
```

および

```
'AB' < 'AA' < 'Aa' < 'Ab' < 'aB' < 'aA' < 'aa' < 'ab'
```

GRAPHIC ストリングの比較は、文字ストリングの場合と同じように処理されます。

GRAPHIC ストリングの比較は、 LONG VARGRAPHIC を除くすべての GRAPHIC ストリング・データ型の間で有効です。 LONG VARGRAPHIC および DBCLOB データ型は、比較演算では使用できません。

GRAPHIC ストリングに対しては、データベースの照合シーケンスは使用されません。その代わりに、GRAPHIC ストリングは、常に対応するバイトの数値 (バイナリー値) に基づいて比較されます。

前の例で、リテラルが GRAPHIC ストリングの場合、以下のような結果になります。

```
'A' < 'B' < 'a' < 'b'
```

および

```
'AA' < 'AB' < 'Aa' < 'Ab' < 'aA' < 'aB' < 'aa' < 'ab'
```

長さの異なる GRAPHIC ストリングを比較する場合、短い方のストリングの右端に長い方のストリングの長さになるまで、 2 バイト・ブランクを埋め込んだものの論理コピーが比較に使用されます。

2 つの GRAPHIC ストリングの値が等しくなるのは、両方が空であるか、または対応する GRAPHIC がすべて等しい場合です。どちらかのオペランドが NULL 値の場合の結果は不明です。 2 つの値が等しくない場合は、両者の関係は単純なバイナリー・ストリング比較によって決定されます。

この節で説明してきたとおり、バイトに基づくストリングの比較は誤った結果をもたらす場合があります。つまり、文字比較で得られる文字とは異なる結果が生じる場合があります。ここで示した一連の例は、同じ MBCS コード・ページであることを前提にしていますが、実際には、同じ言語を使用していても異なるマルチバイトのコード・ページを使用することがあるので、状況はもっと複雑であるといえます。たとえば、日本語 DBCS コード・ページと日本語 EUC コード・ページからのストリングを比較するというような場合が考えられます。

日付/時刻の比較

DATE、TIME、または TIMESTAMP 値は、同じデータ型の別の値か、そのデータ型のストリング表記と比較することができます。すべての比較は日時順に行われます。つまり、0001 年 1 月 1 日からの時間の経過の大きい方が値が大きいということです。

TIME 値と、時刻値のストリング表記とが関係する比較では、常に秒数が組み入れられます。ストリング表記で秒数を省略しているときは、暗黙のうちにゼロ秒が補われます。

TIMESTAMP 値に関する比較は、等しいと見なしてもよいような表示の考慮はしません。日時順に行われます。

例:

```
TIMESTAMP('1990-02-23-00.00.00') > '1990-02-22-24.00.00'
```

ユーザー定義タイプの比較

ユーザー定義特殊タイプの値は、完全に同じユーザー定義特殊タイプの値とのみ比較することができます。ユーザー定義特殊タイプは、WITH COMPARISONS 文節を使用して定義されていなければなりません。

例:

以下の YOUTH 特殊タイプおよび CAMP_DB2_ROSTER 表を想定します。

```
CREATE DISTINCT TYPE YOUTH AS INTEGER WITH COMPARISONS


CREATE TABLE CAMP_DB2_ROSTER
   ( NAME               VARCHAR(20),
     ATTENDEE_NUMBER    INTEGER NOT NULL,
     AGE                YOUTH,
     HIGH_SCHOOL_LEVEL  YOUTH)
```

以下の比較は有効です。

```
SELECT * FROM CAMP_DB2_ROSTER
   WHERE AGE > HIGH_SCHOOL_LEVEL
```

以下の比較は無効です。

```
SELECT * FROM CAMP_DB2_ROSTER
   WHERE AGE > ATTENDEE_NUMBER
```

ただし、特殊タイプとそのソース・タイプとの間では、キャストのための関数または CAST 指定を使用することによって、 AGE と ATTENDEE_NUMBER とを比較することができます。以下の比較はすべて有効です。

```
SELECT * FROM CAMP_DB2_ROSTER
   WHERE INTEGER(AGE) > ATTENDEE_NUMBER

 SELECT * FROM CAMP_DB2_ROSTER
   WHERE CAST( AGE AS INTEGER) > ATTENDEE_NUMBER

 SELECT * FROM CAMP_DB2_ROSTER
   WHERE AGE > YOUTH(ATTENDEE_NUMBER)

 SELECT * FROM CAMP_DB2_ROSTER
   WHERE AGE > CAST(ATTENDEE_NUMBER AS YOUTH)
```

ユーザー定義構造化タイプの値を、他の値と比較することはできません (NULL 述部および TYPE 述部が使えます)。

参照タイプの比較

参照タイプ値を比較できるのは、それらのターゲット・タイプが共通のスーパータイプを持っている場合だけです。その共通のスーパータイプのスキーマ名が関数パスに組み込まれている場合、適切な比較関数は検索されるだけです。比較は参照タイプの表示タイプを使用して行われます。参照の有効範囲は、比較では考慮されません。

関連資料:

- 64 ページの『ID』
- 234 ページの『LIKE 述部』
- 404 ページの『POSSTR』

- 96 ページの『日付/時刻の値』
- 107 ページの『データ型間のキャスト』
- 126 ページの『結果データ型の規則』
- 131 ページの『ストリング変換の規則』

結果データ型の規則

結果のデータ型は、演算のオペランドに適用される規則によって決定されます。ここでは、そのような規則について説明します。

これらの規則は以下に適用されます。

- 集合演算 (UNION、INTERSECT、および EXCEPT) の全選択における対応する列
- CASE 式の結果式
- スカラー関数 COALESCE (または VALUE) の引き数
- IN 述部のリストの式値
- 複数行の VALUES 文節の対応する式

これらの規則は、さまざまな演算での長ストリングに関するその他の制限にも従って、適用されます。

さまざまなデータ型に関係する規則を以下に示します。一部については、考えられる結果データ型を表に示します。

それらの表では、適用される長さまたは精度と位取りも含めて、結果データ型を示します。結果タイプは、オペランドを考慮して決定されます。オペランドの対が複数の場合は、まず最初の対から検討します。それによる結果タイプとその次のオペランドとの組み合わせが検討されて、次の結果タイプが決定される、というようになります。最後の中間結果タイプと最後のオペランドによって、その演算の最終的な結果タイプが決定されます。演算処理は左から右へ行われます。このため、演算が繰り返されるときは、中間結果タイプが重要になります。たとえば、以下のような演算を考えてみます。

```
CHAR(2) UNION CHAR(4) UNION VARCHAR(3)
```

最初の対の結果のタイプは CHAR(4) です。この結果の値は常に 4 文字になります。最終的な結果タイプは VARCHAR(4) です。最初の UNION 演算の結果の値は、常に長さが 4 になります。

文字ストリング

文字ストリングは他の文字ストリングと互換性があります。文字ストリングには、CHAR、VARCHAR、LONG VARCHAR、および CLOB データ型が組み込まれます。

一方のオペランド	他方のオペランド	結果のデータ型
CHAR(x)	CHAR(y)	CHAR(z)、ただし z = max(x,y)
CHAR(x)	VARCHAR(y)	VARCHAR(z)、ただし z = max(x,y)
VARCHAR(x)	CHAR(y) または VARCHAR(y)	VARCHAR(z)、ただし z = max(x,y)
LONG VARCHAR	CHAR(y)、VARCHAR(y)、または LONG VARCHAR	LONG VARCHAR
CLOB(x)	CHAR(y)、VARCHAR(y)、または CLOB(y)	CLOB(z)、ただし z = max(x,y)
CLOB(x)	LONG VARCHAR	CLOB(z)、ただし z = max(x,32700)

結果の文字ストリングのコード・ページは、ストリング変換の規則に基づいて導き出されます。

GRAPHIC ストリング

GRAPHIC ストリングは他の GRAPHIC ストリングと互換性があります。GRAPHIC ストリングには、GRAPHIC、VARGRAPHIC、LONG VARGRAPHIC、および DBCLOB データ型が入ります。

一方のオペランド	他方のオペランド	結果のデータ型
GRAPHIC(x)	GRAPHIC(y)	GRAPHIC(z)、ただし z = max(x,y)
VARGRAPHIC(x)	GRAPHIC(y) または VARGRAPHIC(y)	VARGRAPHIC(z)、ただし z = max(x,y)
LONG VARGRAPHIC	GRAPHIC(y)、VARGRAPHIC(y)、または LONG VARGRAPHIC	LONG VARGRAPHIC
DBCLOB(x)	GRAPHIC(y)、VARGRAPHIC(y)、または DBCLOB(y)	DBCLOB(z)、ただし z = max(x,y)
DBCLOB(x)	LONG VARGRAPHIC	DBCLOB(z)、ただし z = max(x,16350)

結果の GRAPHIC ストリングのコード・ページは、ストリング変換の規則に基づいて導き出されます。

Unicode データベース内の文字ストリングおよび GRAPHIC ストリング

Unicode データベースでは、文字ストリングと GRAPHIC ストリングは互換性があります。

一方のオペランド	他方のオペランド	結果のデータ型
GRAPHIC(x)	CHAR(y) または GRAPHIC(y)	GRAPHIC(z)、ただし z = max(x,y)
VARGRAPHIC(x)	CHAR(y) または VARCHAR(y)	VARGRAPHIC(z)、ただし z = max(x,y)
VARCHAR(x)	GRAPHIC(y) または VARGRAPHIC	VARGRAPHIC(z)、ただし z = max(x,y)
LONG VARGRAPHIC	CHAR(y)、VARCHAR(y)、または LONG VARCHAR	LONG VARGRAPHIC
LONG VARCHAR	GRAPHIC(y) または VARGRAPHIC(y)	LONG VARGRAPHIC
DBCLOB(x)	CHAR(y)、VARCHAR(y)、または CLOB(y)	DBCLOB(z)、ただし z = max(x,y)
DBCLOB(x)	LONG VARCHAR	DBCLOB(z)、ただし z = max(x,16350)

一方のオペランド	他方のオペランド	結果のデータ型
CLOB(x)	GRAPHIC(y) または VARGRAPHIC(y)	DBCLOB(z)、ただし z = max(x,y)
CLOB(x)	LONG VARGRAPHIC	DBCLOB(z)、ただし z = max(x,16350)

バイナリー・ラージ・オブジェクト (BLOB)

BLOB は別の BLOB とのみ互換であり、結果は BLOB になります。 BLOB タイプとして扱う必要がある場合、 BLOB スカラー関数を使用して他のタイプからキャストすることができます。結果の BLOB の長さは、すべてのデータ型の中で最大の長さです。

数値

数値タイプは他の数値タイプと互換性があります。数値タイプには、SMALLINT、INTEGER、BIGINT、DECIMAL、REAL および DOUBLE が入ります。

一方のオペランド	他方のオペランド	結果のデータ型
SMALLINT	SMALLINT	SMALLINT
INTEGER	INTEGER	INTEGER
INTEGER	SMALLINT	INTEGER
BIGINT	BIGINT	BIGINT
BIGINT	INTEGER	BIGINT
BIGINT	SMALLINT	BIGINT
DECIMAL(w,x)	SMALLINT	DECIMAL(p,x)、ただし p = x+max(w-x,5) [注 1]
DECIMAL(w,x)	INTEGER	DECIMAL(p,x)、ただし p = x+max(w-x,11) [注 1]
DECIMAL(w,x)	BIGINT	DECIMAL(p,x)、ただし p = x+max(w-x,19) [注 1]
DECIMAL(w,x)	DECIMAL(y,z)	DECIMAL(p,s)、ただし p = max(x,z)+max(w-x,y-z) [注 1]s = max(x,z)
REAL	REAL	REAL
REAL	DECIMAL、BIGINT、INTEGER、または SMALLINT	DOUBLE
DOUBLE	任意の数値	DOUBLE

[1] 精度は 31 以下でなければなりません。

DATE (日付)

日付は、別の日付、または日付の有効なストリング表記を値とする任意の CHAR または VARCHAR 式と互換性があります。結果のデータ型は DATE です。

TIME (時刻)

時刻は、別の時刻、または時刻の有効なストリング表記を値とする任意の CHAR または VARCHAR 式と互換性があります。結果のデータ型は TIME です。

TIMESTAMP (タイム・スタンプ)

タイム・スタンプは、別のタイム・スタンプ、またはタイム・スタンプの有効なストリング表記を値とする任意の CHAR または VARCHAR 式と互換性があります。結果のデータ型は TIMESTAMP です。

DATALINK (データ・リンク)

データ・リンクはほかのデータ・リンクと互換性があります。結果のデータ型は DATALINK です。結果の DATALINK の長さは、すべてのデータ型の中で最大の長さです。

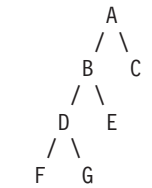
ユーザー定義タイプ

特殊タイプ: ユーザー定義特殊タイプは同じユーザー定義特殊タイプとしか互換性がありません。結果のデータ型はそのユーザー定義特殊タイプです。

参照タイプ: 参照タイプは、ほかの参照タイプと互換性がありますが、それは両方のターゲット・タイプが共通のスーパータイプを持っている場合に限ります。結果のデータ型は、共通のスーパータイプをターゲット・タイプとして持っている参照タイプです。すべてのオペランドに同じ有効範囲の表がある場合、結果は有効範囲の表になります。それ以外の場合、結果では効力範囲は指定されません。

構造化タイプ: 構造化タイプは、ほかの構造化タイプと互換性がありますが、それは両方が共通のスーパータイプを持っている場合に限ります。結果の構造化タイプ列の静的データ型は、いずれかの列の最小限の共通スーパータイプである構造化タイプです。

たとえば、以下の構造化タイプ階層について考えてみます。



静的タイプ E と F の構造化タイプは、結果の静的タイプ B と互換性があります。ただし、E および F の最小限の共通スーパータイプです。

結果の NULL 可能属性

INTERSECT と EXCEPT を除き、 2 つのオペランドの両方とも NULL 値が使用できないのでない限り、結果で NULL 値が可能です。

- INTERSECT で、どちらかのオペランドで NULL 値を使えない場合、結果での NULL 値の使用は認められません (論理積が NULL 値になることはありません)。
- EXCEPT では、最初のオペランドで NULL 値を使えない場合、結果での NULL 値の使用は認められません (結果は最初のオペランドの値しか取れないためです)。

関連資料:

- 303 ページの『BLOB』
- 131 ページの『ストリング変換の規則』

ストリング変換の規則

演算の実行に使用されるコード・ページは、その演算のオペランドに適用される規則によって決定されます。ここでは、そのような規則について説明します。

これらの規則は以下に適用されます。

- 集合演算 (UNION、INTERSECT、および EXCEPT) の全選択における対応するストリング列
- 連結のオペランド
- 述部のオペランド (LIKE を除く)
- CASE 式の結果式
- スカラー関数 COALESCE (および VALUE) の引き数
- IN 述部のリストの式値
- 複数行の VALUES 文節の対応する式

それぞれの場合で結果コード・ページはバインド時に決定されます。演算の実行時に、ストリングがそのコード・ページで識別されるコード・ページに変換されることがあります。有効な変換がなされていない文字は、置換文字にマップされ、文字セットと SQLWARN10 が SQLCA で 'W' に設定されます。

結果コード・ページは、オペランドのコード・ページによって決定されます。初めの 2 つのオペランドのコード・ページが中間結果コード・ページを決定し、 (該当する場合には) そのコード・ページと次のオペランドのコード・ページが新たな中間結果コード・ページを決定します。以下、同様になります。最後の中間結果コード・ページと最後のオペランドのコード・ページが、最終結果のストリングまたは列のコード・ページを決定します。コード・ページの各ペアについて、次の規則を順番に適用することで結果が決定されます。

- コード・ページが等しい場合、結果はそのコード・ページになります。
- コード・ページが BIT DATA (コード・ページ 0) 結果のコード・ページは BIT DATA になります。
- Unicode データベースでは、一方のコード・ページがもう一方のコード・ページとは異なるコード化スキーム内にデータを表示する場合、結果は、UTF-8 上の UCS-2 (つまり、文字データ型上の GRAPHIC データ型) になります。 (非 Unicode データベースでは、異なるコード化スキーム間での変換はサポートされません。)
- ホスト変数であるオペランド (コード・ページは BIT DATA ではない) の場合、結果コード・ページはデータベース・コード・ページになります。そのようなホスト変数からの入力データは、使用前に、アプリケーション・コード・ページからデータベース・コード・ページに変換されます。

以下については、必要なら結果のコード・ページへの変換が行われます。

- 連結演算子のオペランド
- スカラー関数 COALESCE (または VALUE) から選択された引き数
- CASE 式から選択された結果式
- IN 述部のリストの式
- 複数行の VALUES 文節の対応する式

- 集合演算に関係した対応する列

文字変換は、次の条件のすべてに該当する場合に必要になります。

- コード・ページが異なる
- いずれのストリングも BIT DATA ではない
- ストリングが NULL でも空でもない

例

例 1: コード・ページ 850 で作成されたデータベースで、以下の条件がある場合は、次のようになります。

式	タイプ	コード・ページ
COL_1	列	850
HV_2	ホスト変数	437

ここで、以下の述部を評価すると、

```
COL_1 CONCAT :HV_2
```

ホスト変数データは、使用前に、データベース・コード・ページに変換されるため、 2 つのオペランドの結果コード・ページは 850 になります。

例 2: 上記の例からの情報を使用して、述部を評価すると、

```
COALESCE(COL_1, :HV_2:NULLIND,)
```

結果コード・ページは 850 になります。したがって、 COALESCE スカラー関数の結果コード・ページはコード・ページ 850 になります。

パーティションの互換性データ型

パーティションの互換性 は、区分化キーの対応する列の基本データ型相互間で定義されます。パーティション互換データ型には、同じ値をもつ 2 つの変数 (タイプごとに 1 つずつ) が、同じパーティション関数によって同じ区分化マップ索引にマップされるという特性があります。

表 10 は、パーティションのデータ型の互換性を示しています。

パーティションの互換性には、次の特性があります。

- DATE、TIME、および TIMESTAMP には内部フォーマットが使用されます。内部フォーマットは相互に互換性がなく、CHAR との互換性がありません。
- パーティションの互換性は、NOT NULL または FOR BIT DATA 定義を伴う列の影響を受けません。
- 互換データ型の NULL 値は同じように取り扱われます。互換性のないデータ型の NULL の場合は異なる結果が生じることがあります。
- UDT の基本データ型は、パーティションの互換性を分析する場合に使用されます。
- 区分化キーの同一値の小数部は、位取りおよび精度が異なっている場合であっても、同一として取り扱われます。
- 文字ストリング (CHAR、VARCHAR、GRAPHIC または VARGRAPHIC) の末尾のブランクは、システムにより提供されるハッシュ関数によって無視されます。
- 長さが異なる CHAR または VARCHAR は、互換データ型です。
- 等しい REAL または DOUBLE の値は、精度が異なっていても同一として取り扱われます。

表 10. パーティションの互換性

オペランド	バイナリー整数	10 進数	浮動小数点数	文字ストリング	GRAPHIC ストリング	DATE	TIME	TIMESTAMP	特殊タイプ	構造化タイプ
バイナリー整数	Yes	No	No	No	No	No	No	No	[1]	No
10 進数	No	Yes	No	No	No	No	No	No	[1]	No
浮動小数点数	No	No	Yes	No	No	No	No	No	[1]	No
文字ストリング [3]	No	No	No	Yes[2]	No	No	No	No	[1]	No
GRAPHIC ストリング [3]	No	No	No	No	Yes	No	No	No	[1]	No
DATE	No	No	No	No	No	Yes	No	No	[1]	No
TIME	No	No	No	No	No	No	Yes	No	[1]	No
TIMESTAMP	No	No	No	No	No	No	No	Yes	[1]	No
特殊タイプ	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]	No
構造化タイプ [3]	No	No	No	No	No	No	No	No	No	No

表 10. パーティションの互換性 (続き)

オペランド	バイナリー整数	10 進数	浮動小数点数	文字ストリング	GRAPHIC ストリング	DATE	TIME	TIMESTAMP	特殊タイプ	構造化タイプ

注:

1. ユーザー定義特殊タイプ (UDT) の値は、UDT のソース・タイプ、もしくはパーティション互換ソース・タイプのその他の UDT との互換性があるパーティションです。
2. FOR BIT DATA 属性は、パーティションの互換性に影響を与えません。
3. ユーザー定義構造化タイプとデータ型 LONG VARCHAR、LONG VARGRAPHIC、CLOB、 DBCLOB、および BLOB は、区分化キーでサポートされていないので、パーティション互換性には適用されないことに注意してください。

定数

定数 (*リテラル* と呼ばれるときもあります) は、値を指定するものです。定数は、ストリング定数と数値定数に分けられる。 数値定数はさらに、整数、浮動小数点数、または 10 進数に分類されます。

定数は、すべて NOT NULL の属性を持ちます。

数値定数では、負のゼロ値 (-0) は符号のないゼロ (0) と同じ値です。

ユーザー定義タイプは強力なタイプ指定です。つまり、ユーザー定義タイプはそれ自体のタイプとしか互換性がありません。一方、定数には組み込みタイプがあります。このため、ユーザー定義タイプと定数が関与する演算が実行可能なのは、ユーザー定義タイプがその定数の組み込みタイプにキャストされている場合か、または定数がそのユーザー定義タイプにキャストされている場合のみです。たとえば、124 ページの『ユーザー定義タイプの比較』にある表と特殊タイプを使用する場合、定数 14 との以下の比較が有効です。

```
SELECT * FROM CAMP_DB2_ROSTER
  WHERE AGE > CAST(14 AS YOUTH)

SELECT * FROM CAMP_DB2_ROSTER
  WHERE CAST(AGE AS INTEGER) > 14
```

以下の比較は無効です。

```
SELECT * FROM CAMP_DB2_ROSTER
  WHERE AGE > 14
```

整数定数

整数定数 は、小数点を除き最大 19 桁の符号付きまたは符号なしの整数を指定します。整数定数の値が長精度整数の範囲内である場合、その整数定数のデータ型は長精度整数 (large integer) です。 整数定数の値が長精度整数の範囲外であるが、64 ビット整数の範囲内にある場合、その整数定数のデータ型は 64 ビット整数 (big integer) です。 64 ビット整数値の範囲外で定義された定数は、10 進定数と見なされます。

長精度整数定数の最小のリテラル表現は -2 147 483 647 であり、整数値の限界である -2 147 483 648 ではありません。同様に、64 ビット整数定数の最小のリテラル表現は、 -9 223 372 036 854 775 807 であり、 -9 223 372 036 854 775 808 (64 ビット整数値の限界) ではありません。

例:

```
64     -15     +100     32767     720176      12345678901
```

構文図で 'integer' (整数) という用語は、符号を使用してはならない長精度整数定数を指すために使用されます。

浮動小数点定数

浮動小数点定数 は、 E で区切られた 2 つの数値で浮動小数点数を指定します。最初の数値には符号と小数点を指定することができます。 2 番目の数値には符号を指定できますが、小数点を使用することはできません。浮動小数点定数のデータ型は

倍精度です。定数の値は、最初の数値と、2 番目の数値で指定される 10 の累乗との積であり、浮動小数点数の範囲内になければなりません。定数の文字数は 30 以下でなければなりません。

例:

```
15E1     2.E5     2.2E-1     +5.E+2
```

10 進定数

10 進定数 は、 31 桁以内の数字で構成される符号付きまたは符号なしの数値です。小数点を備えているか、または バイナリー整数の範囲に収まらないかのどちらかです。これは 10 進数の範囲内になければなりません。精度は桁数の合計数 (前後のゼロを含む)、位取りは小数点の右側の桁数 (後続ゼロを含む) です。

例:

```
25.5     1000.     -15.     +37589.3333333333
```

文字ストリング定数

文字ストリング定数 は、可変長の文字ストリングを指定するもので、アポストロフィ (') で始まりアポストロフィで終わる文字の並びで構成されます。 ストリング定数のこの形式では、ストリング区切り文字の間に文字ストリングを指定します。文字ストリングの長さは、32 672 バイト以下でなければなりません。文字ストリング内で 1 つのストリング区切り文字を表したいときは、ストリング区切り文字を 2 つ連続して使用します。

例:

```
'12/14/1985'
'32'
'DON''T CHANGE'
```

定数値は、データベースにバインドされるときに、必ずデータベース・コード・ページに変換されます。それは、データベース・コード・ページのものと見なされます。したがって、定数を FOR BIT DATA 列と結合してその結果が FOR BIT DATA となる式で使用される場合、定数値は使用時にそのデータベース・コード・ページ表記から変換されません。

16 進定数

16 進定数 は、セクション・コード・ページ内の可変長の文字ストリングを指定します。

16 進数定数のフォーマットは、X の後に、アポストロフィ (') で囲んだ一つながりの文字を続けたものです。アポストロフィの間にある文字は、偶数個の 16 進数字でなければなりません。 16 進数字の数は 16 336 以下でなければなりません。これを超えると、エラー (SQLSTATE -54002) になります。 1 桁の 16 進数字は 4 ビットを表します。これは、数字としてか、または A から F (大文字または小文字) の任意の文字として指定されます。ただし、 A はビット・パターン '1010' を、B は '1011' を表し、以後同様に続きます。 16 進定数のフォーマットに誤りがある (たとえば、無効な 16 進数字もしくは奇数の 16 進数字が入っている) 場合、エラーが生じます (SQLSTATE 42606)。

例:

```
X'FFFF'       ビット・パターン '1111111111111111' を表します。

X'4672616E6B' ASCII ストリングの VARCHAR パターン 'Frank' を表します。
```

GRAPHIC ストリング定数

*GRAPHIC ストリング定数*は可変長の GRAPHIC ストリングを指定しますが、このストリングを構成するのは、単一バイトのアポストロフィ (') で囲まれ、しかも先頭に単一バイトの G または N を付けられた一つながりの 2 バイト文字です。アポストロフィで囲まれた文字は偶数バイトでなければならず、 GRAPHIC ストリングの長さは 16 336 バイト以下でなければなりません。

例:

```
G'2 バイト文字ストリング'
N'2 バイト文字ストリング'
```

MBCS 文字の一部としては、単引用符 (') を使用しないでください。区切り文字と見なされてしまいます。

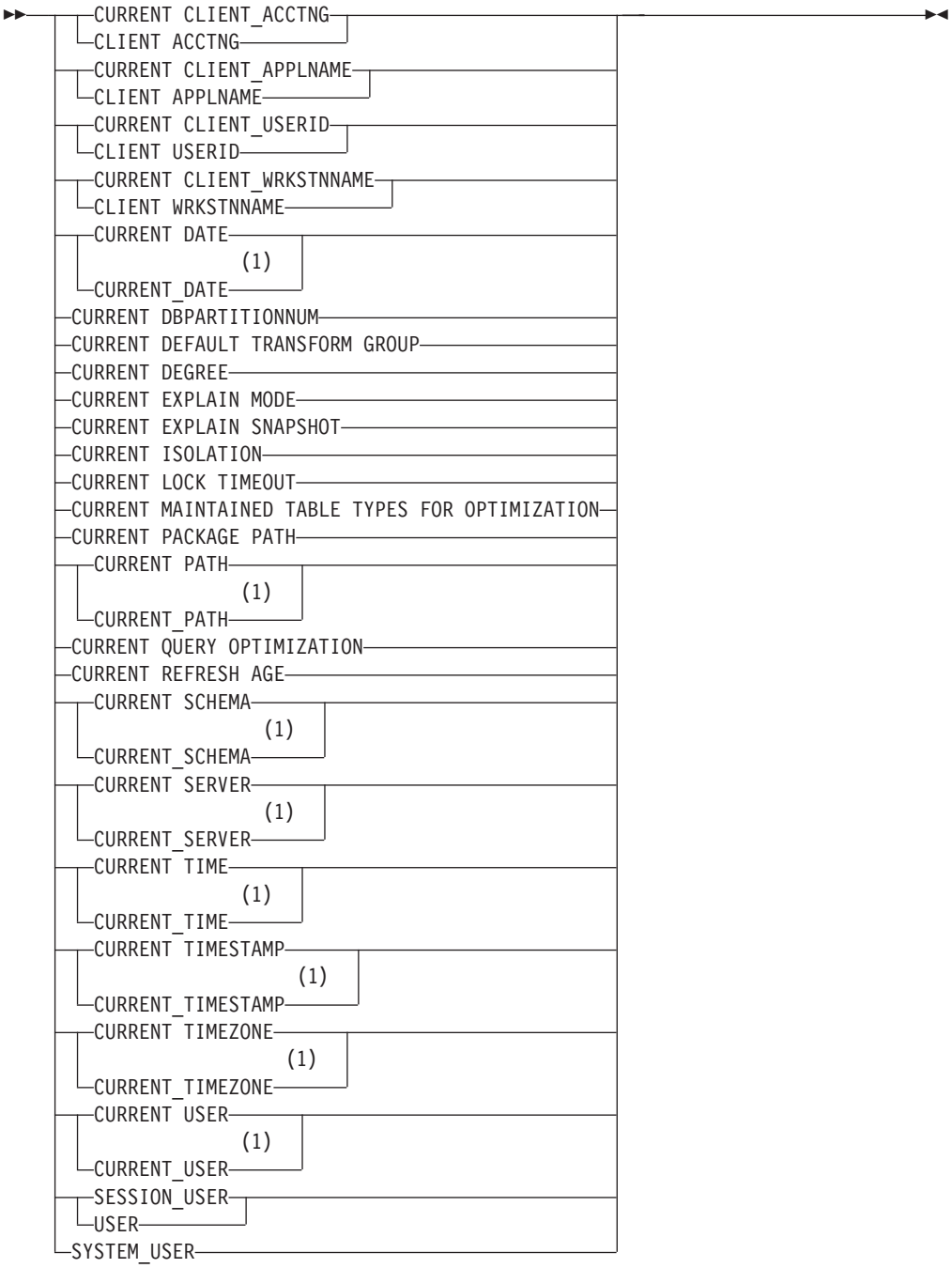
関連資料:

- 184 ページの『式』
- 110 ページの『割り当てと比較』

特殊レジスター

特殊レジスター

特殊レジスター は、データベース・マネージャーによってアプリケーション・プロセスに対して定義されるストレージです。それは、SQL ステートメントで参照可能な情報を保管するのに使用されます。特殊レジスターの参照は、現行サーバーから提供される値の参照になります。値がストリングの場合、その CCSID は、現行サーバーのデフォルト CCSID になります。特殊レジスターの参照は、次のようにして行うことができます。



注:

1 SQL 1999 Core 標準では、下線付きの書式が使用されます。

一部の特殊レジスターは、SET ステートメントを使用して更新できます。以下の表は、どの特殊レジスターを更新できるかを示しています。

表 11. 特殊レジスター

特殊レジスター	更新可能
CURRENT CLIENT_ACCTNG	No
CURRENT CLIENT_APPLNAME	No
CURRENT CLIENT_USERID	No
CURRENT CLIENT_WRKSTNNAME	No
CURRENT DATE	No
CURRENT DBPARTITIONNUM	No
CURRENT DEFAULT TRANSFORM GROUP	Yes
CURRENT DEGREE	Yes
CURRENT EXPLAIN MODE	Yes
CURRENT EXPLAIN SNAPSHOT	Yes
CURRENT ISOLATION	Yes
CURRENT LOCK TIMEOUT	Yes
CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION	Yes
CURRENT PACKAGE PATH	Yes
CURRENT PATH	Yes
CURRENT QUERY OPTIMIZATION	Yes
CURRENT REFRESH AGE	Yes
CURRENT SCHEMA	Yes
CURRENT SERVER	No
CURRENT TIME	No
CURRENT TIMESTAMP	No
CURRENT TIMEZONE	No
CURRENT USER	No
SESSION_USER	Yes
SYSTEM_USER	No
USER	Yes

特殊レジスターがルーチン内で参照されるとき、ルーチン内の特殊レジスターの値はその特殊レジスターが更新可能かどうかによって異なります。更新可能ではない特殊レジスターの場合、値はその特殊レジスターのデフォルト値に設定されます。更新可能な特殊レジスターの場合、初期値はルーチンの起動側から継承されて、ルーチン内の後続の SET ステートメントによって変更できます。

CURRENT CLIENT_ACCTNG

CURRENT CLIENT_ACCTNG (または CLIENT ACCTNG) 特殊レジスターには、この接続用に指定されたクライアント情報からのアカウンティング・ストリングの値が入ります。このレジスターのデータ型は VARCHAR(255) です。このレジスターのデフォルト値は空ストリングです。

Set Client Information (**sqleseti**) API を使用して、アカウンティング・ストリングの値を変更できます。

sqleseti API を使用して指定した値はアプリケーションのコード・ページに入れられ、特殊レジスターの値はデータベースのコード・ページで保管されることに注意してください。 クライアント情報の設定時に使用されるデータ値によっては、特殊レジスターに保管されているデータ値がコード・ページ変換の際に切り捨てられることがあります。

例: この接続のアカウンティング・ストリングの現行値を入手します。

```
 VALUES (CURRENT CLIENT_ACCTNG)
INTO :ACCT_STRING
```

CURRENT CLIENT_APPLNAME

CURRENT CLIENT_APPLNAME (または CLIENT APPLNAME) 特殊レジスターには、この接続用に指定されたクライアント情報からのアプリケーション名の値が入ります。このレジスターのデータ型は VARCHAR(255) です。このレジスターのデフォルト値は空ストリングです。

Set Client Information (**sqleseti**) API を使用して、アプリケーション名の値を変更できます。

sqleseti API を使用して指定した値はアプリケーションのコード・ページに入れられ、特殊レジスターの値はデータベースのコード・ページで保管されることに注意してください。 クライアント情報の設定時に使用されるデータ値によっては、特殊レジスターに保管されているデータ値がコード・ページ変換の際に切り捨てられることがあります。

例: この接続に使用されるアプリケーションを使用できる部門を選択します。

```
SELECT DEPT
  FROM DEPT_APPL_MAP
  WHERE APPL_NAME = CURRENT CLIENT_APPLNAME
```

CURRENT CLIENT_USERID

CURRENT CLIENT_USERID (または CLIENT USERID) 特殊レジスターには、この接続用に指定されたクライアント情報からのユーザー ID の値が入ります。このレジスターのデータ型は VARCHAR(255) です。このレジスターのデフォルト値は空ストリングです。

Set Client Information (**sqleseti**) API を使用して、クライアント・ユーザー ID の値を変更できます。

sqleseti API を使用して指定した値はアプリケーションのコード・ページに入れられ、特殊レジスターの値はデータベースのコード・ページで保管されることに注意してください。 クライアント情報の設定時に使用されるデータ値によっては、特殊レジスターに保管されているデータ値がコード・ページ変換の際に切り捨てられることがあります。

例: 現行のクライアント・ユーザー ID を使用している部門を検出します。

```
SELECT DEPT
  FROM DEPT_USERID_MAP
  WHERE USER_ID = CURRENT CLIENT_USERID
```

CURRENT CLIENT_WRKSTNNAME

CURRENT CLIENT_WRKSTNNAME (または CLIENT WRKSTNNAME) 特殊レジスターには、この接続用に指定されたクライアント情報からのワークステーション名の値が入ります。このレジスターのデータ型は VARCHAR(255) です。このレジスターのデフォルト値は空ストリングです。

Set Client Information (**sqleseti**) API を使用して、ワークステーション名の値を変更できます。

sqleseti API を使用して指定した値はアプリケーションのコード・ページに入れられ、特殊レジスターの値はデータベースのコード・ページで保管されることに注意してください。 クライアント情報の設定時に使用されるデータ値によっては、特殊レジスターに保管されているデータ値がコード・ページ変換の際に切り捨てられることがあります。

例: この接続で使用されているワークステーション名を入手します。

```
 VALUES (CURRENT CLIENT_WRKSTNNAME)
INTO :WS_NAME
```

CURRENT DATE

CURRENT DATE (または CURRENT_DATE) 特殊レジスターは、アプリケーション・サーバーで SQL ステートメントが実行される時点の、時刻機構の読み取り値にもとづく日付を指定します。この特殊レジスターが単一の SQL ステートメントで何度も使用される場合、または単一のステートメントで CURRENT TIME または CURRENT TIMESTAMP と共に使用される場合、その値はすべて時刻機構の 1 回の読み取りに基づきます。

ルーチン内部の SQL ステートメントで使用する場合、呼び出しステートメントからの CURRENT DATE の継承はありません。

フェデレーテッド・システムでは、データ・ソースでの使用を目的とした照会で CURRENT DATE を使用できます。この照会が処理されて戻される日付は、フェデレーテッド・サーバーの CURRENT DATE レジスターから取得されたもので、データ・ソースから取得されたものではありません。

例: 以下の例は、PROJECT 表を使用して、 MA2111 プロジェクト (PROJNO) のプロジェクト終了日付 (PRENDATE) に CURRENT DATE を設定しています。

```
UPDATE PROJECT
  SET PRENDATE = CURRENT DATE
  WHERE PROJNO = 'MA2111'
```

CURRENT DBPARTITIONNUM

CURRENT DBPARTITIONNUM 特殊レジスターは、ステートメントのコーディネーター・ノード番号を識別する INTEGER 値を指定します。アプリケーションから発行されるステートメントの場合は、アプリケーションの接続先のパーティションがコーディネーターになります。ルーチンから発行されるステートメントの場合は、ルーチンが呼び出されるパーティションがコーディネーターになります。

ルーチン内部の SQL ステートメントで使用する場合、呼び出しステートメントからの CURRENT DBPARTITIONNUM の継承はありません。

データベース・インスタンスがパーティション分割をサポートするように定義されていない場合、 CURRENT DBPARTITIONNUM は 0 を戻します。 (これはつまり、db2nodes.cfg ファイルが存在しない場合です。パーティション・データベースの場合は、db2nodes.cfg ファイルがあり、そこにパーティション、またはノードの定義が入っている場合です。)

CURRENT DBPARTITIONNUM は、ある一定の条件に該当する場合に限り、CONNECT ステートメントで変更できます。

バージョン 8 以前のバージョンと互換性を持たせるため、DBPARTITIONNUM の部分はキーワード NODE に置き換えられます。

例: 以下の例では、アプリケーションが接続しているパーティションの番号をホスト変数 APPL_NODE (整数) に設定しています。

```
VALUES CURRENT DBPARTITIONNUM
  INTO :APPL_NODE
```

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CONNECT (タイプ 1) ステートメント』

CURRENT DEFAULT TRANSFORM GROUP

CURRENT DEFAULT TRANSFORM GROUP 特殊レジスターは、VARCHAR (18) 値を指定します。ここには、ユーザー定義構造化タイプの値をホスト・プログラムと交換するときに、動的 SQL ステートメントで使用するトランスフォーム・グループの名前を指定します。この特殊レジスターでは、静的 SQL ステートメントで使用する、または外部関数やメソッドを使ったパラメーターと結果の交換で使用するトランスフォーム・グループを指定しません。

その値は SET CURRENT DEFAULT TRANSFORM GROUP ステートメントによって設定することができます。値を設定しない場合、特殊レジスターの初期値は、空ストリングになります (ゼロの長さの VARCHAR)。

動的 SQL ステートメント (つまり、ホスト変数と相互作用するもの) では、値を交換するときに使用するトランスフォーム・グループの名前は、このレジスターに空ストリングが入っていない限り、この特殊レジスターの値と同じになります。レジスターに空ストリングが入っている場合 (SET CURRENT DEFAULT TRANSFORM GROUP ステートメントを使用して、値が設定されていない場合)、トランスフォームのために、DB2_PROGRAM トランスフォーム・グループが使われます。 構造化タイプ・サブジェクト用に DB2_PROGRAM トランスフォーム・グループが定義されていない場合、ランタイムにエラーが生じます (SQLSTATE 42741)。

例:

デフォルトのトランスフォーム・グループを MYSTRUCT1 に設定します。MYSTRUCT1 トランスフォームで定義される TO SQL および FROM SQL 関数は、ユーザー定義構造化タイプ変数とホスト・プログラムを交換するときに使います。

```
SET CURRENT DEFAULT TRANSFORM GROUP = MYSTRUCT1
```

この特殊レジスターに割り当てられた、デフォルトのトランスフォーム・グループの名前を検索します。

```
VALUES (CURRENT DEFAULT TRANSFORM GROUP)
```

CURRENT DEGREE

CURRENT DEGREE 特殊レジスターは、動的 SQL ステートメントを実行するときの、パーティション内並列処理の度合いを指定します。 (静的 SQL の場合、DEGREE BIND オプションが同じ制御機能として働きます。) このレジスターのデータ型は CHAR(5) です。有効な値は、ANY、または 1 から 32 767 の範囲 (両端の値を含む) の整数のストリング表記です。

SQL ステートメントが動的に準備される時点で、整数として表現される CURRENT DEGREE の値が 1 である場合には、そのステートメントの実行にパーティション内並列処理は使用されません。

SQL ステートメントが動的に準備されるときに、整数として表される CURRENT DEGREE の値が 2 以上 32 767 以下である場合、そのステートメントの実行には、指定された度合いのパーティション内並列処理を伴う場合があります。

SQL ステートメントが動的に準備される時点で、 CURRENT DEGREE の値が ANY である場合、そのステートメントの実行には、データベース・マネージャーによって決定された度合いを用いたパーティション内並列処理を使用できます。

実際の実行時の並列処理の度合いは、以下の低い方になります。

- 最大照会度合 (*max_querydegree*) 構成パラメーターの値
- アプリケーション実行時の度合い
- SQL ステートメントのコンパイルの度合い

intra_parallel のデータベース・マネージャー構成パラメーターが NO に設定される場合、最適化のために CURRENT DEGREE 特殊レジスターの値は無視され、ステートメントはパーティション内並列処理を使用しません。

値は、SET CURRENT DEGREE ステートメントを呼び出すことによって変更できます。

CURRENT DEGREE の初期値は、*dft_degree* データベース構成パラメーターによって判別されます。

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『SET CURRENT DEGREE ステートメント』

CURRENT EXPLAIN MODE

CURRENT EXPLAIN MODE 特殊レジスターには、該当する動的 SQL ステートメントに関連のある Explain 機能の動作を制御するための VARCHAR(254) の値が入れられます。この機能は、Explain 情報を生成し、その情報を Explain 表に挿入します。この情報には、Explain スナップショットは入っていません。 使用できる値は、YES、EXPLAIN、NO、REOPT、RECOMMEND INDEXES、および EVALUATE INDEXES です。(静的 SQL の場合、EXPLAIN BIND オプションは同じ制御機能として働きます。 PREP および BIND コマンドの場合、EXPLAIN オプション値は YES、NO、および ALL です。)

YES Explain 機能を使用可能にし、動的 SQL ステートメントについての Explain 情報をそのステートメントのコンパイル時にキャプチャーします。

EXPLAIN
機能を使用可能にします。ただし、動的ステートメントは実行されません。

NO Explain 機能を使用不可にします。

REOPT
Explain 機能が使用可能になり、動的 (つまり増分バインド) SQL ステートメントに関する Explain 情報がそのステートメントのコンパイル時にキャプチャーされることになります。ただし、入力変数 (ホスト変数、特殊レジスター、またはパラメーター・マーカー) 用の実数を使ってこのステートメントが再最適化された場合に限ります。

RECOMMEND INDEXES
各動的照会に一連の索引を推奨します。 ADVISE_INDEX 表に一連の索引を移植します。

EVALUATE INDEXES
推奨されている索引が存在するかのように、動的照会を Explain します。使用される索引は ADVISE_INDEX 表から選出されます。

初期値は NO です。値は、SET CURRENT EXPLAIN MODE ステートメントを呼び出すことによって変更できます。

CURRENT EXPLAIN MODE と CURRENT EXPLAIN SNAPSHOT 特殊レジスター値は、 Explain 機能が呼び出されている場合に相互に作用します。 CURRENT EXPLAIN MODE 特殊レジスター値の方は、 EXPLAIN BIND オプションとも相互に作用します。 RECOMMEND INDEXES と EVALUATE INDEXES を設定できるのは、 CURRENT EXPLAIN MODE レジスターの場合だけです。これらを設定するには、 SET CURRENT EXPLAIN MODE ステートメントを使用します。

例: ホスト変数 EXPL_MODE (VARCHAR(254)) を CURRENT EXPLAIN MODE 特殊レジスターの現在の値に設定します。

```
VALUES CURRENT EXPLAIN MODE
  INTO :EXPL_MODE
```

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『SET CURRENT EXPLAIN MODE ステートメント』
- 779 ページの『付録 J. EXPLAIN レジスター値』

CURRENT EXPLAIN SNAPSHOT

CURRENT EXPLAIN SNAPSHOT 特殊レジスターには、 Explain スナップショット機能の動作を制御するための CHAR(8) の値が入れられます。この機能は、アクセス・プラン情報、演算子コスト、バインド実行時の統計などに関する情報を圧縮して生成するものです。

次に挙げるステートメントだけがこのレジスターの値として認められます。すなわち、DELETE、INSERT、SELECT、SELECT INTO、UPDATE、VALUES、および VALUES INTO です。使用できる値は、YES、EXPLAIN、NO、および REOPT です。 (静的 SQL の場合、EXPLSNAP BIND オプションが同じ制御機能として働きます。 PREP および BIND コマンドの場合、EXPLSNAP オプション値は YES、NO、および ALL です。)

YES Explain スナップショット機能を使用可能にし、動的 SQL ステートメントがコンパイルされるときの、そのステートメントの内部表記のスナップショットを取り出します。

EXPLAIN
Explain スナップショット機能を使用可能にします。ただし、動的ステートメントは実行されません。

NO Explain スナップショット機能を使用不可にします。

REOPT
Explain 機能が使用可能になり、動的 (つまり増分バインド) SQL ステートメントに関する Explain 情報がそのステートメントのコンパイル時にキャプチャーされることになります。ただし、入力変数 (ホスト変数、特殊レジスター、またはパラメーター・マーカー) 用の実数を使ってこのステートメントが再最適化された場合に限ります。

初期値は NO です。値は、SET CURRENT EXPLAIN SNAPSHOT ステートメントを呼び出すことによって変更できます。

CURRENT EXPLAIN SNAPSHOT と CURRENT EXPLAIN MODE 特殊レジスター値は、 Explain 機能が呼び出されている場合に相互に作用します。 CURRENT EXPLAIN SNAPSHOT 特殊レジスター値の方は、 EXPLSNAP BIND オプションとも相互に作用します。

例: 以下の例は、ホスト変数 EXPL_SNAP (char(8)) に、 CURRENT EXPLAIN SNAPSHOT 特殊レジスターの現在の値を設定するものです。

```
VALUES CURRENT EXPLAIN SNAPSHOT
  INTO :EXPL_SNAP
```

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『SET CURRENT EXPLAIN SNAPSHOT ステートメント』
- 779 ページの『付録 J. EXPLAIN レジスター値』

CURRENT ISOLATION

CURRENT ISOLATION 特殊レジスターは、現行のセッション内で発行された動的 SQL ステートメントの分離レベル (他の並行セッションに関する) を識別する CHAR(2) 値を収容します。

使用できる値は次のとおりです。

(ブランク)
未設定。パッケージの分離属性を使用します。

UR 非コミット読み取り

CS カーソル固定

RR 反復可能読み取り

RS 読み取り固定

CURRENT ISOLATION 特殊レジスターの値は、SET CURRENT ISOLATION ステートメントにより変更することができます。

SET CURRENT ISOLATION ステートメントがセッション内で発行されるまでか、または SET CURRENT ISOLATION に対して RESET が指定された後で、CURRENT ISOLATION 特殊レジスターはブランクに設定され、動的 SQL ステートメントには適用されません。使用される分離レベルは、動的 SQL ステートメントを発行したパッケージの分離属性から取られます。 SET CURRENT ISOLATION ステートメントが発行されると、CURRENT ISOLATION 特殊レジスターは、ステートメントを発行したパッケージの設定に関係なく、セッション内でコンパイルされた後続のすべての動的 SQL ステートメントのための分離レベルを提供します。これが有効であるのは、セッションが終了するまでか、または RESET オプションを使用して SET CURRENT ISOLATION ステートメントが発行されるまでです。

例: ホスト変数 ISOLATION_MODE (CHAR(2)) を CURRENT ISOLATION 特殊レジスターに保管されている現在の値に設定します。

```
VALUES CURRENT ISOLATION
  INTO :ISOLATION_MODE
```

関連概念:

- 14 ページの『分離レベル』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『SET CURRENT ISOLATION ステートメント』

CURRENT LOCK TIMEOUT

CURRENT LOCK TIMEOUT 特殊レジスターは、待機の開始から何秒経過したら、ロックをかけられなかったことを示すエラーが戻されるかを指定します。この特殊レジスターは、行、表、索引キー、および MDC ブロック・ロックに影響を与えます。このレジスターのデータ型は INTEGER です。

CURRENT LOCK TIMEOUT 特殊レジスターの有効値は、 -1 以上 32767 以下の整数です。この特殊レジスターは、NULL 値に設定することもできます。 -1 の値は、タイムアウトはとられないので、ロックの解放またはデッドロックの検出までアプリケーションは待機することを指定します。 0 の値は、アプリケーションがロックを待機しないことを指定します。ロックをかけられなかった場合、ただちにエラーが戻されます。

CURRENT LOCK TIMEOUT 特殊レジスターの値は、 SET CURRENT LOCK TIMEOUT ステートメントを起動して変更することができます。その初期値はヌルですが、その場合、 *locktimeout* データベース構成パラメーターの現行値がロックの待機時間に使われ、この値が、この特殊レジスターの値として戻されます。

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『SET CURRENT LOCK TIMEOUT ステートメント』

CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION

CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION 特殊レジスターは、 VARCHAR(254) の値を指定します。この値は、動的 SQL 照会の処理を最適化する際に考慮される表のタイプを識別するものです。マテリアライズ照会表は、静的組み込み SQL 照会で使用されることは決してありません。

CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION の初期値は SYSTEM です。その値は SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION ステートメントによって変更することができます。

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION ステートメント』
- 「*管理ガイド: パフォーマンス*」の『dft_mttb_types - 最適化用デフォルト保守表タイプ構成パラメーター』

CURRENT PACKAGE PATH

CURRENT PACKAGE PATH 特殊レジスターは、 SQL ステートメントの実行時に必要なパッケージの参照の解決に使用されるパスを特定する VARCHAR(4096) 値を指定します。

この値は、空またはブランクのストリングか、または二重引用符で区切られてさらにコンマで区切られた 1 つ以上のスキーマ名のリストのどちらでもかまいません。ストリング中で二重引用符を使用する場合はすべて、区切り ID の通常の使用法と同様に 2 つの二重引用符で表記する必要があります。区切り文字とコンマは、この特殊レジスターの長さの一部を成します。

この特殊レジスターは、静的と動的の両方のステートメントで使用できます。

ユーザー定義の関数、メソッド、またはプロシージャー内の CURRENT PACKAGE PATH の初期値は、呼び出し側アプリケーションから継承されます。すなわち、CURRENT PACKAGE PATH の初期値は空ストリングです。アプリケーションが SET CURRENT PACKAGE PATH ステートメントを使ってスキーマ・リストを明示的に指定していた場合のみ、その初期値はスキーマのリストになります。

例:

アプリケーションは、複数の SQLJ パッケージ (スキーマ SQLJ1 と SQLJ2 に入っています) と、 1 つの JDBC パッケージ (DB2JAVA に入っています) を使用することになっています。 CURRENT PACKAGE PATH 特殊レジスターを設定して SQLJ1、SQLJ2、および DB2JAVA をこの順序で調べます。

```
SET CURRENT PACKAGE PATH = "SQLJ1", "SQLJ2", "DB2JAVA"
```

ホスト変数 HVPKLIST を、CURRENT PACKAGE PATH 特殊レジスターに現在保管されている値に設定します。

```
VALUES CURRENT PACKAGE PATH INTO :HVPKLIST
```

関連資料:

- 154 ページの『CURRENT PATH』
- 「*SQL リファレンス 第 2 巻*」の『SET CURRENT PACKAGE PATH ステートメント』

CURRENT PATH

CURRENT PATH (または CURRENT_PATH) 特殊レジスターは、動的に作成された SQL ステートメントの関数参照やデータ型参照の解決に使用される SQL パスを識別する、 VARCHAR(254) 値を指定します。 CURRENT FUNCTION PATH は、CURRENT PATH の同義語です。 CURRENT PATH は、 CALL ステートメントのストアード・プロシージャー参照を解決するのにも使用されます。初期値は、後述するデフォルト値です。静的 SQL の場合は、FUNCPATH BIND オプションで関数およびデータ型の解決のための SQL パスを指定できます。

CURRENT PATH 特殊レジスターには、二重引用符で囲まれ、コンマで区切られた、1 つ以上のスキーマ名のリストが入っています。たとえば、データベース・マネージャーがまず FERMAT スキーマを参照し、次いで XGRAPHIC スキーマ、最後に SYSIBM スキーマを参照するように指定する SQL パスは、 CURRENT PATH 特殊レジスターでは以下のように戻されます。

```
"FERMAT","XGRAPHIC","SYSIBM"
```

デフォルト値は 『SYSIBM』、『SYSFUN』、『SYSPROC』、 *X* (*X* は USER 特殊レジスターの値) で、それぞれは二重引用符で区切られます。値は、SET CURRENT PATH ステートメントを呼び出すことによって変更できます。 SYSIBM スキーマを指定する必要はありません。 SQL パスにスキーマが指定されていなければ、暗黙的にそのスキーマが最初のスキーマとして見なされます。暗黙的に想定されている場合、SYSIBM は 254 文字を一切扱いません。

スキーマ名で修飾されないデータ型は、同じ非修飾名のデータ型が入っている SQL パスの最初のスキーマで暗黙的に修飾されます。この規則には、CREATE DISTINCT TYPE、CREATE FUNCTION、COMMENT、および DROP ステートメントの部分で概説されているように例外があります。

例: 以下のビューは、SYSCAT.VIEWS カタログ・ビューを使用して、 CURRENT PATH 特殊レジスターの現行値と同じ設定で作成されたすべてのプランを検索しています。

```
SELECT VIEWNAME, VIEWSCHEMA FROM SYSCAT.VIEWS
  WHERE FUNC_PATH = CURRENT PATH
```

関連資料:

- 166 ページの『関数』
- 「*SQL リファレンス 第 2 巻*」の『SET PATH ステートメント』

CURRENT QUERY OPTIMIZATION

CURRENT QUERY OPTIMIZATION 特殊レジスターには、動的 SQL ステートメントのバインド時に、データベース・マネージャーによって行われる照会最適化のクラスを制御する INTEGER 値が入れられます。 QUERYOPT BIND オプションは、静的 SQL ステートメントの照会クラスの最適化を制御します。使用できる値の範囲は 0 から 9 です。たとえば、照会最適化クラスが 0 (最小の最適化) に設定されている場合、この特殊レジスターの値は 0 です。デフォルト値は、*dft_queryopt* データベース構成パラメーターで決まります。値は、SET CURRENT QUERY OPTIMIZATION ステートメントを呼び出すことによって変更できます。

例: 以下の例は、SYSCAT.PACKAGES カタログ・ビューを使用して、 CURRENT QUERY OPTIMIZATION 特殊レジスターの現行値と同じ設定でバインドされたすべてのプランを検索しています。

```
SELECT PKGNAME, PKGSCHEMA FROM SYSCAT.PACKAGES
  WHERE QUERYOPT = CURRENT QUERY OPTIMIZATION
```

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『SET CURRENT QUERY OPTIMIZATION ステートメント』

CURRENT REFRESH AGE

CURRENT REFRESH AGE 特殊レジスターは、データ型が DECIMAL(20,6) のタイム・スタンプ期間値を指定します。これは、タイム・スタンプが作成された、キャッシュ・データ・オブジェクトでの特定のイベント (たとえば、システムが保守する REFRESH DEFERRED マテリアライズ照会表で REFRESH TABLE ステートメントを処理するなど) が起きた後、照会の処理の最適化にそのキャッシュ・データ・オブジェクトを使用できる最大期間です。 CURRENT REFRESH AGE の値が 99 999 999 999 999 (ANY) で、照会の最適化クラスが 5 以上の場合は、動的 SQL 照会の処理を最適化する際に、 CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION で指定されたタイプの表が考慮されます。

CURRENT REFRESH AGE の初期値はゼロです。値は、SET CURRENT REFRESH AGE ステートメントを呼び出すことによって変更できます。

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『SET CURRENT REFRESH AGE ステートメント』

CURRENT SCHEMA

CURRENT SCHEMA (または CURRENT_SCHEMA) 特殊レジスターは、動的に作成された SQL ステートメントで可能な場合に、データベース・オブジェクト参照を修飾するのに使用されるスキーマ名を識別する VARCHAR(128) 値を指定します。 DB2 for OS/390 との互換性を保つため、 CURRENT SQLID (または CURRENT_SQLID) は CURRENT SCHEMA の同義語になっています。

CURRENT SCHEMA の初期値は、現行セッション・ユーザーの許可 ID です。値は、SET SCHEMA ステートメントを呼び出すことによって変更できます。

QUALIFIER BIND オプションは、動的に作成された SQL ステートメントについて可能な場合に、データベース・オブジェクト参照を修飾するのに使用されるスキーマ名を制御します。

例: オブジェクト修飾のスキーマを 'D123' に設定します。

```
SET CURRENT SCHEMA =  'D123'
```

CURRENT SERVER

CURRENT SERVER (または CURRENT_SERVER) 特殊レジスターには、現在のアプリケーション・サーバーを識別する VARCHAR(18) の値が入れられます。このレジスターにはアプリケーション・サーバーの実際の名前 (別名ではない) が入れられます。

CURRENT SERVER は、ある一定の条件に該当する場合に限り、CONNECT ステートメントで変更できます。

ルーチン内部の SQL ステートメントで使用する場合、呼び出しステートメントからの CURRENT SERVER の継承はありません。

例: 以下の例は、アプリケーションが接続されているアプリケーション・サーバーの名前をホスト変数 APPL_SERVE (VARCHAR(18)) に設定しています。

```
VALUES CURRENT SERVER INTO :APPL_SERVE
```

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CONNECT (タイプ 1) ステートメント』

CURRENT TIME

CURRENT TIME (または CURRENT_TIME) 特殊レジスターは、アプリケーション・サーバーで SQL ステートメントが実行される時点の時刻機構の読み取り値に基づく時刻を指定します。この特殊レジスターが単一の SQL ステートメントで何度も使用される場合、または単一のステートメントで CURRENT DATE または CURRENT TIMESTAMP と共に使用される場合、その値はすべて時刻機構の 1 回の読み取りに基づく値です。

ルーチン内部の SQL ステートメントで使用する場合、呼び出しステートメントからの CURRENT TIME の継承はありません。

フェデレーテッド・システムでは、データ・ソースでの使用を目的とした照会で CURRENT TIME を使用できます。この照会が処理されて戻される時刻は、フェデレーテッド・サーバーの CURRENT TIME レジスターから取得されたもので、データ・ソースから取得されたものではありません。

例: 以下の例は、CL_SCHED 表を使って、その日のそれ以降に開始される (STARTING) すべてのクラス (CLASS_CODE) を選択するものです。現在のクラスの DAY 列の値は 3 です。

```
SELECT CLASS_CODE FROM CL_SCHED
  WHERE STARTING > CURRENT TIME AND DAY = 3
```

CURRENT TIMESTAMP

CURRENT TIMESTAMP (または CURRENT_TIMESTAMP) 特殊レジスターは、アプリケーション・サーバーで SQL ステートメントが実行される時点の、時刻機構の読み取り値にもとづくタイム・スタンプを指定します。この特殊レジスターが単一の SQL ステートメントで何度も使用される場合、または単一のステートメントで CURRENT DATE または CURRENT TIME と共に使用される場合、その値はすべて時刻機構の 1 回の読み取りに基づく値です。

ルーチン内部の SQL ステートメントで使用する場合、呼び出しステートメントからの CURRENT TIMESTAMP の継承はありません。

フェデレーテッド・システムでは、データ・ソースでの使用を目的とした照会で CURRENT TIMESTAMP を使用できます。この照会が処理されて戻されるタイム・スタンプは、フェデレーテッド・サーバーの CURRENT TIMESTAMP レジスターから取得されたもので、データ・ソースから取得されたものではありません。

例: 以下の例は、1 つの行を IN_TRAY 表に挿入するものです。RECEIVED 列の値は、その行の挿入時点を示すタイム・スタンプでなければなりません。他の 3 つの列の値は、ホスト変数 SRC (char(8))、SUB (char(64))、および TXT (VARCHAR(200)) から取られたものです。

```
INSERT INTO IN_TRAY
  VALUES (CURRENT TIMESTAMP, :SRC, :SUB, :TXT)
```

CURRENT TIMEZONE

CURRENT TIMEZONE (または CURRENT_TIMEZONE) 特殊レジスターには、UTC (協定世界時 (Coordinated Universal Time)。旧 GMT。) とアプリケーション・サーバーのローカル時との差が入れられます。この差は、時刻期間 (最初の 2 桁が時間数、次の 2 桁が分数、最後の 2 桁が秒数である 10 進数) によって表現されます。時間数の数値は -24 と 24 を除く -24 と 24 の間です。ローカル時刻から CURRENT TIMEZONE を減算すると、ローカル時刻が UTC に変換されます。時刻は、SQL ステートメントが実行されるときに、オペレーティング・システムの時刻から計算されます。 (CURRENT TIMEZONE の値は、C のランタイム関数によって決まります。)

CURRENT TIMEZONE 特殊レジスターは、時刻やタイム・スタンプの算術演算など、 DECIMAL(6,0) のデータ型の式が使用されるところならどこででも使用できます。

ルーチン内部の SQL ステートメントで使用する場合、呼び出しステートメントからの CURRENT TIMEZONE の継承はありません。

例: 以下の例は、RECEIVED 列の UTC タイム・スタンプを使って、 IN_TRAY 表にレコードを挿入します。

```
INSERT INTO IN_TRAY VALUES (
  CURRENT TIMESTAMP - CURRENT TIMEZONE,
  :source,
  :subject,
  :notetext )
```

CURRENT USER

CURRENT USER (または CURRENT_USER) 特殊レジスターは、ステートメントの許可に使用される許可 ID を指定します。静的 SQL ステートメントの場合、値は、パッケージをバインドする際に使用される許可 ID を表します。動的 SQL ステートメントの場合、値は、 DYNAMICRULES(RUN) BIND オプションでバインドされたパッケージの SESSION_USER 特殊レジスターの値と同じになります。このレジスターのデータ型は VARCHAR(128) です。

例: スキーマが CURRENT USER 特殊レジスターの値と一致する表名を選択します。

```
SELECT TABNAME FROM SYSCAT.TABLES
  WHERE TABSCHEMA = CURRENT USER AND TYPE = 'T'
```

このステートメントが静的 SQL ステートメントとして実行される場合は、このステートメントを収めたパッケージのバインド・プログラムと一致するスキーマ名の付いた表が戻されます。このステートメントが動的 SQL ステートメントとして実行される場合は、 SESSION_USER 特殊レジスターの現行の値と一致するスキーマ名の表が戻されます。

SESSION_USER

SESSION_USER 特殊レジスターは、現行セッションに使用される許可 ID を指定します。このレジスターの値は、DYNAMICRULES の実行動作がパッケージに対して有効な場合に、動的 SQL ステートメントの許可検査に使用されます。このレジスターのデータ型は VARCHAR(128) です。

新しい接続での SESSION_USER の初期値は、SYSTEM_USER 特殊レジスターの値と同じです。値は、SET SESSION AUTHORIZATION ステートメントを呼び出すことで変更できます。

SESSION_USER は、USER 特殊レジスターの同義語です。

例: 動的 SQL を使用してどのルーチンを実行できるかを判別します。DYNAMICRULES の実行振る舞いが、ルーチンを呼び出す動的 SQL の発行元パッケージに対して有効であることを想定します。

```
SELECT SCHEMA, SPECIFICNAME FROM SYSCAT.ROUTINEAUTH
  WHERE GRANTEE = SESSION_USER
  AND EXECUTEAUTH IN ('Y', 'G')
```

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『SET SESSION AUTHORIZATION ステートメント』

SYSTEM_USER

SYSTEM_USER 特殊レジスターは、データベースに接続するユーザーの許可 ID を指定します。このレジスターの値は、別の許可 ID を持つユーザーとして接続しない限り、変更できません。このレジスターのデータ型は VARCHAR(128) です。

SET SESSION AUTHORIZATION ステートメントの説明にある、『例』を参照してください。

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『SET SESSION AUTHORIZATION ステートメント』

USER

USER 特殊レジスターは、データベースでアプリケーションが始動するときにデータベース・マネージャーに渡されるランタイム許可 ID を指定します。このレジスターのデータ型は VARCHAR(128) です。

ルーチン内部の SQL ステートメントで使用する場合、呼び出しステートメントからの USER の継承はありません。

例: ユーザー自身が IN_TRAY 表に入れたすべてのメモを、表から選択します。

```
SELECT * FROM IN_TRAY
  WHERE SOURCE = USER
```

関数

データベース関数 とは、一連の入力データの値と一連の結果の値との間の関係です。たとえば TIMESTAMP 関数は、入力データ値として DATE および TIME データ型を受け取り、 TIMESTAMP という結果を生成します。 関数には組み込み関数とユーザー定義関数があります。

- データベース・マネージャーには*組み込み関数* が付属しています。これは、単一の結果値を返し、SYSIBM スキーマの一部と識別されます。そのような関数には、列関数 (たとえば AVG)、演算子関数 (たとえば +)、および cast 関数 (たとえば DECIMAL) などがあります。
- *ユーザー定義関数*は、 SYSCAT.ROUTINES のデータベースに登録されている (CREATE FUNCTION ステートメントを使って) 関数です。 ユーザー定義関数は SYSIBM スキーマの一部ではありません。このような関数の集合の 1 つに、データベース・マネージャーに用意されている SYSFUN という名前のスキーマがあります。

  ユーザー定義関数は、データベース・エンジンそのものにおいて適用できる関数定義 (ユーザーまたは第三者ベンダー作成の) を追加することで、データベース・システムの機能を拡張することができます。データベース関数が拡張されれば、アプリケーションが使用するのと同じ関数をデータベースがエンジン内で活用することができ、それによってアプリケーションとデータベースとの間の相互作用が強化されます。

外部、SQL、およびソース・ユーザー定義関数

ユーザー定義関数は、外部関数、SQL 関数、またはソース関数とすることができます。*外部関数* は、オブジェクト・コード・ライブラリーと、関数呼び出し時に実行されるそのライブラリー内の関数によってデータベースに対して定義されます。列関数を外部関数にすることはできません。 *SQL 関数* は、SQL RETURN ステートメントだけを使用して、データベースに対して定義されます。スカラー値、行、または表のいずれかを戻すことができます。 SQL 関数を、列関数とすることはできません。*ソース関数からの派生関数* は、データベースがすでに認識している別の組み込み関数またはユーザー定義関数への参照によって、データベースに対して定義されます。ソース関数からの派生関数はスカラー関数または列関数です。これらは、ユーザー定義タイプの既存の関数のサポートに有用です。

スカラー、列、行、および表のユーザー定義関数

各ユーザー定義関数は、スカラー関数、列関数、または表関数として類別することもできます。*スカラー関数* は、呼び出されるたびに単一値の応答を戻す関数です。たとえば、組み込み関数 SUBSTR() はスカラー関数です。スカラー UDF は、外部関数とソースからの派生関数のどちらであってもかまいません。

列関数 は、概念上、類似値の集合 (列) を渡され、単一値の応答を戻す関数です。DB2 では、場合によっては*総計関数* と呼ばれることもあります。列関数の例として、組み込み関数 AVG() があります。外部列 UDF を DB2 に対して定義することはできませんが、組み込み列関数のいずれかをソースとして派生する列 UDF を定義することができます。これは、特殊タイプに対して有用です。たとえば、特殊タ

イプ SHOESIZE が基本タイプ INTEGER を使用して定義されている場合、組み込み関数 AVG(INTEGER) をソースとする UDF AVG(SHOESIZE) を定義することができ、これは列関数になります。

行関数 とは、値を一行で戻す関数です。これは、構造化タイプの属性値を行の値に割り当てるトランスフォーム関数としてのみ、使うことができます。行関数は、SQL 関数と定義する必要があります。

表関数 は、その関数を参照する SQL ステートメントに表を戻す関数です。SELECT ステートメントの FROM 文節でのみ参照することができます。このような関数を使用して、DB2 データ以外のデータに SQL 言語処理能力を適用することや、このようなデータを DB2 表に変換することができます。 たとえば、ファイルを取り出してそれを表に変換することや、 WWW からデータをサンプリングしてそれを表にすること、あるいは Lotus Notes データベースにアクセスして、日付、送信元、メッセージのテキストなどのすべてのメール・メッセージに関する情報を戻すこと、などを行うことができます。このような情報は、データベースの他の表と結合することができます。表関数は、外部関数または SQL 関数と定義できます。 (表関数はソース関数であってはなりません。)

関数シグニチャー

関数は、そのスキーマ、関数名、パラメーター数、およびそのパラメーターのデータ型によって識別されます。これは*関数シグニチャー* と呼ばれ、データベース内でユニークである必要があります。 パラメーターの数やパラメーターのデータ型が違っていれば、 1 つのスキーマに同じ名前の関数が複数存在してもかまいません。複数の関数インスタンスのある関数名は、*多重定義* 関数と呼ばれます。ある関数名があるスキーマ内で多重定義される場合、そのスキーマにはその名前で 2 つ以上の関数があるということです。これらの関数は、それぞれ別々のパラメーター・タイプをもっていなければなりません。関数名は SQL パスにおいても多重定義可能です。その場合、そのパス内にその名前の付いた関数が 2 つ以上あることになりますが、必ずしもパラメーター・タイプがそれぞれ異なっている必要はありません。

関数を呼び出すには、後に括弧に入った引き数のリストの付いた修飾名 (スキーマ名および関数名) を参照します (使用可能なコンテキストで)。また、スキーマ名を指定せずに関数を呼び出すことも可能であり、その場合は、異なるスキーマの関数のうち、同じパラメーターまたは許容パラメーターをもつ可能な関数を選択できることになります。 このような場合、関数解決に役立つ *SQL パス* が使用されます。SQL パスとは、同じ名前、同じパラメーター数、および受け入れ可能なデータ型を持つ関数を識別するために探索されるスキーマのリストです。静的 SQL ステートメントに対する SQL パスは、 FUNCPATH BIND オプションを使って指定されます。動的 SQL ステートメントの場合、 SQL パスは CURRENT PATH 特殊レジスターの値です。

関数へのアクセスは、EXECUTE 特権を使って制御します。個々の関数または一連の関数を誰が実行できて誰ができないかを指定するには、 GRANT および REVOKE ステートメントを使用します。関数を呼び出すには、EXECUTE 特権 (つまり DBADM 権限) が必要です。関数の定義者には、自動的に EXECUTE 特権が与えられます。外部関数や、基礎となるすべてのオブジェクトでの WITH GRANT オプションをもつ SQL 関数の定義者には、関数に対する EXECUTE 特権をもった WITH GRANT オプションも与えられます。定義者 (または SYSADM または

DBADM) は次にそれを、任意の SQL ステートメントからの関数の呼び出し、任意の DDL ステートメント (たとえば CREATE VIEW、CREATE TRIGGER、または制約の定義時) の参照、またはこの関数をソースとして派生する別の関数の作成を行いたいと考えているユーザーに付与する必要があります。 EXECUTE 特権をユーザーに付与しないと、この関数は、一致性では優れていても関数解決アルゴリズムでの検討対象にはなりません。組み込み関数 (SYSIBM 関数) と SYSFUN 関数は、暗黙で PUBLIC に付与される EXECUTE 特権をもちます。

関数解決

関数呼び出しの後、データベース・マネージャーは、同じ名前をもつ呼び出し可能な関数の中でどれが『最適』かを判別する必要があります。これには、組み込み関数とユーザー定義関数の中から関数を解決する処理が関与します。

引き数 とは、呼び出し時に関数に渡される値です。SQL の中で呼び出されるとき、関数にはゼロ個以上の引き数のリストが渡されます。このような引き数は、引き数が引き数リストの位置によって決定されるというセマンティクスで定位置と言えます。パラメーター は、関数への入力の形式的な定義です。データベースに対して内部的に (組み込み関数) またはユーザーによって (ユーザー定義関数) 関数が定義されるときは、関数のパラメーターが (ゼロ個以上) 指定されます。また、パラメーターの定義の順序によって、パラメーターの位置とセマンティクスが定義されることになります。したがって、どのパラメーターも関数への特定の定位置入力になります。呼び出し時に、引き数リスト中のその位置によって、引き数は特定のパラメーターに対応します。

データベース・マネージャーは、呼び出しで指定される関数名、その関数に対する EXECUTE 特権、引き数の数とデータ型、SQL パスの中で同じ名前を持つすべての関数、対応するパラメーターのデータ型を使用して、ある関数を選択するかどうかの判断基準とします。メソッドを決定する過程で発生する可能性のある結果について以下に示します。

- 特定の関数が最適であると判断される場合。たとえば、名前が RISK、スキーマが TEST、シグニチャーが以下のように定義された関数を考えてみます。

  ```
  TEST.RISK(INTEGER)
  TEST.RISK(DOUBLE)
  ```

  SQL パスには TEST スキーマが入っており、以下のように関数が参照されたとします (DB は DOUBLE 列)。

  ```
  SELECT ... RISK(DB) ...
  ```

  この場合は、2 番目の RISK が選択されます。

  以下のように関数が参照される場合は (SI は SMALLINT 列)、

  ```
  SELECT ... RISK(SI) ...
  ```

  最初の RISK が選択されます。これは、SMALLINT は INTEGER にプロモート可能であり、優先順位リストでの順位がさらに下になる DOUBLE よりも一致性が高いためです。

構造化タイプである引き数について考慮する場合、優先順位リストには、静的タイプの引き数のスーパータイプが入っています。最適なのは、構造化タイプ階層の中で、静的タイプの関数引き数に最も近いスーパータイプ・パラメーターで定義する関数です。

- 許容できる適合性をもつ関数がないと判断される場合。たとえば、前出の例と同じ 2 つの関数が指定され、以下のように関数が参照されたとします (C は CHAR(5) 列)。

```
SELECT ... RISK(C) ...
```

この場合、引き数はどちらの RISK 関数のパラメーターとも整合性がありません。

- SQL パス、および呼び出し時に渡される引き数の数とデータ型に基づいて特定の関数が選択される場合。たとえば、名前が RANDOM で、シグニチャーが以下のように定義されている関数を考えてみます。

```
TEST.RANDOM(INTEGER)
PROD.RANDOM(INTEGER)
```

SQL パスは以下のとおりです。

```
"TEST","PROD"
```

この場合、次の関数参照では、

```
SELECT ... RANDOM(432) ...
```

TEST.RANDOM が選択されます。これは、どちらの RANDOM 関数も同程度に高い一致性を示し (この例の場合は完全一致)、どちらのスキーマも関数パスにありますが、 SQL パスにおいて TEST の方が PROD より先に指定されているためです。

最適の判別

引き数のデータ型と、対象とする関数のパラメーターに定義されているデータ型との比較は、似通った名前の関数の中でどれが『最適』かを決定する基準となります。関数の結果のデータ型または関数のタイプ (列、スカラー、または表) は、この決定には関係しないことに注意してください。

関数解決は、以下の手順で行われます。

1. まず、カタログ (SYSCAT.ROUTINES) と組み込み関数から、以下のすべての条件が真となるすべての関数を探します。
   - スキーマ名が指定された呼び出し (修飾子付き参照) の場合、スキーマ名と関数名が呼び出し名に一致する。
   - スキーマ名が指定されていない呼び出し (修飾子なし参照) の場合、関数名が呼び出し名に一致し、 SQL パス中のスキーマの 1 つに一致するスキーマ名がある。
   - 呼び出し側は、関数に対する EXECUTE 特権をもっている。
   - 定義済みパラメーターの数が呼び出しと一致している。
   - 呼び出しの各引き数のデータ型が、関数の対応する定義済みパラメーターのデータ型に一致するか、またはそのデータ型に『プロモート可能』である。

2. 次に、関数呼び出しの個々の引き数を左から右に検討していきます。引き数ごとに、その引き数に対して最適な一致ではない関数をすべて除去していきます。ある引き数の最適な一致とは、その引き数データ型に対応する優先順位リストの中で、そのデータ型のパラメーターをもつ関数が存在するデータ型のうち、最初に記述されているデータ型です。長さ、精度、位取り、および FOR BIT DATA 属性は、この比較では考慮されません。たとえば、DECIMAL(9,1) の引き数は DECIMAL(6,5) のパラメーターと完全に一致すると見なされ、また VARCHAR(19) の引き数は VARCHAR(6) のパラメーターと完全に一致すると見なされます。

   ユーザー定義構造化タイプ引き数に最適なのは、それ自身です。次に適しているのは、すぐ上のスーパータイプです。このことは、引き数のどのスーパータイプにも当てはまります。ここで考慮しているのは、静的タイプ (宣言済みタイプ) の構造化タイプ引き数であり、動的タイプ (最も特定的なタイプ) ではありません。
3. ステップ 2 の後でも 2 つ以上の関数が候補として残っているときは、残りの関数候補はすべて同じシグニチャーを持っていても、それぞれ異なるスキーマに存在しているはずです。スキーマがそのユーザーの SQL パスで最初に出現する関数が選択されます。
4. ステップ 2 の後で候補となる関数が残らなかった場合は、エラー (SQLSTATE 42884) になります。

組み込み関数の場合の関数パスに関する考慮事項

組み込み関数は SYSIBM という名前の特殊なスキーマ内に置かれています。SYSFUN および SYSPROC スキーマにはさらに別の使用可能な関数がありますが、それらは組み込み関数とは見なされません。それらはユーザー定義関数として開発されたものであり、処理上の特別な考慮事項がないためです。 SYSIBM 、SYSFUN、または SYSPROC スキーマに (あるいは SYS の文字で始まる名前の他のどのスキーマにも)、ユーザーがさらに関数を定義することはできません。

すでに説明したように、関数解決処理の中で、組み込み関数はユーザー定義関数とまったく同じように扱われます。関数解決の観点から見た組み込み関数とユーザー定義関数の違いの 1 つは、組み込み関数は関数解決で常に検討対象とする必要があるということです。したがって、パスから SYSIBM を省いても、 (関数およびデータ型の解決では) SYSIBM がパスの最初のスキーマであることが想定されることになります。

たとえば、ユーザーの SQL パスが以下のように定義されているとします。

```
"SHAREFUN","SYSIBM","SYSFUN"
```

また、引き数の数とタイプが SYSIBM.LENGTH と同じである LENGTH 関数が SHAREFUN スキーマに定義されているとします。この場合、このユーザーの SQL ステートメントで、修飾なしで LENGTH を参照すると、SHAREFUN.LENGTH が選択されます。一方、ユーザーの SQL パスが以下のように定義されている場合には、

```
"SHAREFUN","SYSFUN"
```

同じ SHAREFUN.LENGTH 関数が存在していたとしても、このユーザーの SQL ステートメントで修飾せずに LENGTH を参照すると、 SYSIBM.LENGTH が選択されることになります。これは、SYSIBM が暗黙のうちにパス中で最初に出現するためです。

この状況下では、次のようにすれば、問題を未然に最小化することができます。

- ユーザー定義関数には組み込み関数の名前を決して使用しないようにします。
- 何らかの理由で組み込み関数と同じ名前のユーザー定義関数を作成する必要のある場合は、そのような関数への参照には必ず修飾子を付けます。

関数解決の例

以下は、正常な関数解決の例を示しています。 (必要なキーワードがすべて示されているわけではないことに注意してください。)

3 つの異なるスキーマに 7 個の ACT 関数があり、以下のように登録されているとします。

```
  CREATE FUNCTION AUGUSTUS.ACT (CHAR(5), INT, DOUBLE)  SPECIFIC ACT_1 ...
CREATE FUNCTION AUGUSTUS.ACT (INT, INT, DOUBLE)      SPECIFIC ACT_2 ...
CREATE FUNCTION AUGUSTUS.ACT (INT, INT, DOUBLE, INT) SPECIFIC ACT_3 ...
CREATE FUNCTION JULIUS.ACT   (INT, DOUBLE, DOUBLE)   SPECIFIC ACT_4 ...
CREATE FUNCTION JULIUS.ACT   (INT, INT, DOUBLE)      SPECIFIC ACT_5 ...
CREATE FUNCTION JULIUS.ACT   (SMALLINT, INT, DOUBLE) SPECIFIC ACT_6 ...
CREATE FUNCTION NERO.ACT     (INT, INT, DEC(7,2))    SPECIFIC ACT_7 ...
```

以下のように関数が参照されるとします (I1 および I2 は INTEGER 列、 D は DECIMAL 列です)。

```
SELECT ... ACT(I1, I2, D) ...
```

この参照を行うアプリケーションの SQL パスが次のようになっているとします。

```
"JULIUS","AUGUSTUS","CAESAR"
```

アルゴリズムに従っていきます。

- スキーマ NERO が SQL パスに組み込まれていないため、特定の名前 ACT_7 の付いた関数は候補から除かれます。
- パラメーターの数が違っているため、ACT_3 は候補から除かれます。第 1 引き数が第 1 パラメーターのデータ型にプロモートできないため、 ACT_1 と ACT_6 はどちらも候補から除かれます。
- この時点で複数の候補が残っているため、次に引き数が順に検討されます。
- 最初の引き数については、残りのすべての関数 ACT_2、ACT_4、および ACT_5 がその引き数タイプと完全に一致します。この検討ではどの関数も検討の対象から除かれないため、次の引き数を検討する必要があります。
- 2 番目の引き数では、ACT_2 および ACT_5 が完全に一致していますが、ACT_4 は一致していないため、ACT_4 が検討の対象から除かれます。 ACT_2 と ACT_5 の間の何らかの差異を判別するために、さらに次の引き数が検討されます。
- 第 3 の最後の引き数では、 ACT_2 も ACT_5 も引き数のタイプと完全には一致していませんが、両方とも適合度は同程度です。

- この時点で、パラメーター・シグニチャーが同じである関数として ACT_2 と ACT_5 の 2 つが残っています。最終的な決定要因は、どちらの関数のスキーマが SQL パスで先に出現するかであり、この基準によって ACT_5 が最終的に選択されます。

関数の呼び出し

関数が選択された後も、いくつかの理由でその関数の使用が許可されない場合があります。 個々の関数は特定のデータ型の結果を返すように定義されています。この結果のデータ型が、関数が呼び出されるコンテキストと互換性がない場合、エラーが生じます。たとえば、今度は結果のデータ型が異なる STEP という名前の関数が次のように定義されているとします。

```
STEP(SMALLINT) returns CHAR(5)
STEP(DOUBLE)   returns INTEGER
```

以下のように関数が参照されると (S は SMALLINT 列)、

```
SELECT ... 3 + STEP(S) ...
```

引き数タイプが完全に一致しているため、最初の STEP が選択されます。しかし、結果のタイプが加法演算子の引き数として求められる数値タイプではなく CHAR(5) であるため、ステートメントではエラーになります。

この他にこの状態が起きる場合の例は次のとおりです。いずれの例もステートメントのエラーが生じます。

- 関数が FROM 文節で参照されたが、関数解決ステップで選択された関数がスカラー関数または列関数であった場合。
- その逆の場合。つまりコンテキストがスカラー関数または列関数を要求し、関数解決が表関数を選択する場合。

関数呼び出しの引き数が、選択された関数のパラメーターのデータ型と完全一致でない場合、列への割り当てと同じ規則を適用して、実行時に引き数がパラメーターのデータ型に変換されます。これには、引き数とパラメーターの間で精度、位取り、または長さが異なる場合も含まれます。

従来のバインディング・セマンティクス

ステートメントの処理時にルーチンとデータ型が解決されて、データベース・マネージャーはこの解決を繰り返せなければならない場合があります。これは、以下の場合に当てはまります。

- パッケージ内の静的 DML ステートメント
- ビュー
- トリガー
- チェック制約
- SQL ルーチン

パッケージ内の静的 DML ステートメントの場合、ルーチンおよびデータ型の参照はバインド操作時に解決されます。ビュー、トリガー、SQL ルーチン、およびチェック制約の中にあるルーチンとデータ型の参照は、データベース・オブジェクトの作成時に解決されます。

そのようなオブジェクト内のいずれかのルーチン参照に対してルーチン解決をもう一度実行すると、以下の場合はその動作が変わる可能性があります。

- 一致度のより高いシグニチャーをもつ新規のルーチンが追加されたけれども、実際の実行可能ファイルは別の操作で実行される場合。
- 一致度のより高いシグニチャーをもつルーチンに対する実行特権が定義者に付与されたけれども、実際の実行可能ファイルは別の操作で実行される場合。

同様に、これらのオブジェクト内のデータ型のいずれかで二度目の関数解決が実行される場合、他のスキーマにあるのと同じ名前 (SQL パスにも存在している) で新しいデータ型が追加されていると、動作が変わってしまうおそれがあります。これが起きないようにするためにデータベース・マネージャーは、必要であれば常に*従来のバインディング・セマンティクス* を適用します。それによって、ルーチンとデータ型の参照を解決するときは、必ず前のバインド時の解決先と同じ SQL パスと一連のルーチンが使用されることになります。解決時に検討対象となるルーチンとデータ型の作成タイム・スタンプは、ステートメントのバインド時の時刻よりも後にはなりません。 (バージョン 6.1 から追加された組み込み関数では、データベースの作成または移行の時刻に基づいた作成タイム・スタンプが押されます。) このようにして、ルーチン、およびデータ型のうち、ステートメントを最初に処理したときの解決時に考慮されたものだけが考慮されるようにします。したがって、従来のバインド・セマンティクスが適用される場合、後から作成されたルーチン、新たに許可されたルーチン、およびデータ型は考慮されません。

シグニチャー SCHEMA1.BAR(INTEGER) および SCHEMA2.BAR(DOUBLE) をもつ 2 つの関数を擁するデータベースについて考察してみます。 SQL パス内には、SCHEMA1 と SCHEMA2 の 2 つのスキーマ (SQL パス内でのその順序は重要ではありません) があると仮定します。 USER1 には、関数 SCHEMA2.BAR(DOUBLE) に対する EXECUTE 特権が付与されています。 USER1 は、BAR(INT_VAL) を呼び出すビューを作成すると仮定します。これは、関数 SCHEMA2.BAR(DOUBLE) に解決されます。そのビューの作成後に SCHEMA1.BAR(INTEGER) に対する EXECUTE 特権を誰かが USER1 に付与しても、そのビューは常に SCHEMA2.BAR(DOUBLE) を使用します。

パッケージ内の静的 DML の場合、パッケージの再バインドは暗黙的に行うことができますが、 REBIND コマンド (またはこれに対応する API) または BIND コマンド (またはこれに対応する API) を明示的に発行して行うこともできます。暗黙的な再バインドでは、常に保守的バインド・セマンティクスを適用して、ルーチン、およびデータ型の解決が実行されます。 REBIND コマンドには、従来のバインド・セマンティクスで解決するか (RESOLVE CONSERVATIVE オプション)、それとも新しいルーチンとデータ型を考慮して解決するか (デフォルト・オプションの RESOLVE ANY) を選択するオプションがあります。

パッケージを暗黙的に再バインドすると、常に同じルーチンが解決されます。一致性のより高いルーチンに対する EXECUTE 特権を付与されていても、そのルーチンは検討の対象にはなりません。パッケージを明示的に再バインドすると、異なるルーチンが選択されることになります。 (ただし、RESOLVE CONSERVATIVE を指定すると、従来のバインディング・セマンティクスに従ってルーチンは解決されます。)

ビュー、トリガー、制約、または SQL ルーチン本体の作成時にルーチンを指定すると、そのルーチンのどのインスタンスが使用されるかは、オブジェクトの作成時のルーチン解決で決まります。オブジェクトの作成が完了した後で EXECUTE 特権が付与されても、オブジェクトが使用する所定のルーチンは変わりません。

シグニチャー SCHEMA1.BAR(INTEGER) および SCHEMA2.BAR(DOUBLE) をもつ 2 つの関数を擁するデータベースについて考察してみます。 USER1 には、関数 SCHEMA2.BAR(DOUBLE) に対する EXECUTE 特権が付与されています。 USER1 は、BAR(INT_VAL) を呼び出すビューを作成すると仮定します。これは、関数 SCHEMA2.BAR(DOUBLE) に解決されます。そのビューの作成後に SCHEMA1.BAR(INTEGER) に対する EXECUTE 特権を誰かが USER1 に付与しても、そのビューは常に SCHEMA2.BAR(DOUBLE) を使用します。

その他のデータベース・オブジェクトでも、これと同じ動作が行われます。たとえば、パッケージを暗黙で再バインドした (おそらく索引のドロップ後に) 場合、その暗黙の再バインドの前でも後でもそのパッケージは同じ所定のルーチンを参照します。ただしパッケージを*明示的に* 再バインドすると、異なるルーチンが選択されることになります。

関連資料:

- 154 ページの『CURRENT PATH』
- 105 ページの『データ型のプロモーション』
- 110 ページの『割り当てと比較』

メソッド

構造化タイプのデータベース・メソッドは、一連の入力データ値と、一連の結果値との関連のことです。ここで、最初の入力値 (またはサブジェクト*引き数*) は、メソッドと同じ値になるか、サブジェクト・タイプ (サブジェクト・パラメーター ともいう) のサブタイプになります。たとえば、型が ADDRESS である CITY というメソッドは、型が VARCHAR の入力データ値に渡すことができます。結果は ADDRESS (または ADDRESS のサブタイプ) になります。

メソッドは、ユーザー定義構造化タイプの定義の一環として、暗黙的にあるいは明示的に定義されます。

暗黙的に定義されたメソッドは、構造化タイプごとに作成されます。また、構造化タイプの属性ごとに、*監視用メソッド* が定義されます。監視用メソッドを使うと、アプリケーション側は、該当タイプのインスタンスの属性値を知ることができます。*変更メソッド*も属性ごとに定義されます。これにより、アプリケーション側では、型インスタンスの属性の値を変更することによって型インスタンスを変更できます。上記の CITY メソッドは、型 ADDRESS の変更メソッドの一例です。

明示的に定義したメソッド、すなわち*ユーザー定義メソッド* は、 CREATE TYPE (または ALTER TYPE ADD METHOD) および CREATE METHOD ステートメントを組み合わせて使用して、 SYSCAT.ROUTINES のデータベースに登録されるメソッドです。 特定の構造化タイプ用に定義されたメソッドはすべて、その型と同じスキーマで定義されます。

構造化タイプ用のユーザー定義メソッドは、データベース・エンジンの構造化タイプインスタンスに適用できる (ユーザーもしくは第三者ベンダーによって提供された) メソッド定義を追加することによって、データベース・システムの機能を拡張します。データベース・メソッドを定義することにより、データベースはアプリケーションが使用するのと同じメソッドをエンジンで活用することができ、アプリケーションとデータベースとの間の相互作用が高まります。

外部および SQL ユーザー定義メソッド

ユーザー定義メソッドは、外部メソッドとするか、 SQL 式に基づくものとすることができます。外部メソッドは、オブジェクト・コード・ライブラリーと、メソッド呼び出し時に実行されるそのライブラリー内の関数によって、データベースに対して定義されます。 SQL 式に基づいたメソッドは、メソッドの呼び出し時に、その SQL 式の結果を戻します。そのようなメソッドでは、すべてが SQL で作成されているので、オブジェクト・コード・ライブラリーが必要ありません。

ユーザー定義メソッドでは、呼び出されるたびに単一値の応答を戻すことができます。この値は、構造化タイプとすることができます。また、メソッドを (SELF AS RESULT を使用して) *型保持* として定義し、メソッドの戻される型として、動的型のサブジェクト引き数を戻すようにすることが可能です。暗黙的に定義した変更メソッドは、型保持されます。

メソッド・シグニチャー

メソッドは、そのサブジェクト・タイプ、メソッド名、パラメーター数、およびそのパラメーターのデータ型によって識別されます。これはメソッド・シグニチャーと呼ばれ、データベース内でユニークである必要があります。

以下の場合に、同じ名前を付けられた構造化タイプのメソッドが、複数存在する可能性があります。

- パラメーターの数やパラメーターのデータ型が違う場合。または
- メソッドが同じメソッド階層の一部である場合 (つまり、メソッド同士がオーバーライド関係にあるか、同じ元のメソッドをオーバーライドしている場合)。または
- (最初のパラメーターとして、サブジェクト・タイプ、またはそのサブタイプかスーパータイプを使用した) 同じ関数シグニチャーが存在しない場合。

複数のメソッド・インスタンスを持つメソッド名のことを、*多重定義メソッド* といいます。あるメソッド名があるタイプ内で多重定義される場合、そのタイプには、その名前で 2 つ以上のメソッドがあるということです (それぞれのタイプにはすべて、異なるパラメーター・タイプがあります)。メソッド名はサブジェクト・タイプ階層においても多重定義可能です。その場合、そのタイプ階層にその名前のメソッドが 2 つ以上ありますが、それぞれのメソッドには異なるパラメーター・タイプがなければなりません。

メソッドを呼び出すときには (許可されているコンテキストで) 構造化タイプインスタンス (サブジェクト引き数) への参照と、二重ドット演算子の両方が先頭に記されているメソッド名を参照します。 その後に、括弧で囲まれた引き数のリストが続きます。実際に呼び出されるメソッドは、次の項で説明されているメソッド解決プロセスにより、静的タイプのサブジェクト・タイプに基づいて決定されます。関数呼び出しを使い、WITH FUNCTION ACCESS で定義されているメソッドを呼び出すこともできます。その場合、関数解決のための通常のルールが適用されます。

関数解決の結果が WITH FUNCTION ACCESS で定義されているメソッドであれば、メソッド呼び出しのその後のステップはすべて処理されます。

メソッドへのアクセスは EXECUTE 特権を通して制御されます。 GRANT および REVOKE ステートメントは、特定のメソッドまたはメソッドのセットを誰が実行できて誰ができないのかを指定するために使用します。メソッドを呼び出すには、EXECUTE 特権 (または DBADM 権限) が必要です。メソッドの定義者には、自動的に EXECUTE 特権が付与されます。すべての基礎オブジェクトに対する WITH GRANT オプションをもつ外部メソッドまたは SQL メソッドの定義者にはまた、メソッドに対する EXECUTE 特権付きの WITH GRANT オプションが GRANT されます。定義者 (または SYSADM または DBADM) は、これを、任意の SQL ステートメントからメソッドを呼び出したり、任意の DDL ステートメント (CREATE VIEW、CREATE TRIGGER、または制約を定義するときなど) でメソッドを参照したりするユーザーに付与します。ユーザーに EXECUTE 特権が付与されていない場合、そのメソッドは、たとえ一致の度合いが高くても、メソッド解決アルゴリズムによって考慮されません。

メソッド解決

メソッドの呼び出しの後、データベース・マネージャーは、同じ名前をもつ呼び出し可能なメソッドの中でどれが「最適」かを判別する必要があります。メソッド解決時には、関数 (組み込みまたはユーザー定義) は考慮されません。

*引き数*とは、呼び出し時にメソッドに渡される値です。 SQL の中で呼び出されるとき、メソッドには (特定構造化タイプの) サブジェクト引き数、およびゼロ個以上の引き数のリストが渡されます。このような引き数は、引き数が引き数リストの位置によって決定されるというセマンティクスで定位置と言えます。*パラメーター*は、メソッドへの入力の形式上の定義です。メソッドがデータベースに対して暗黙的に (特定タイプ用にシステム生成される)、またはユーザーによって (ユーザー定義メソッド) 定義されるとき、メソッドのパラメーターが (最初のパラメーターとしてのサブジェクト・パラメーター付きで) 指定されます。パラメーターの定義の順序がパラメーターの位置、およびその結果としてパラメーターのセマンティクスを定義することになります。したがって、どのパラメーターもメソッドの特定の定位置入力です。呼び出し時に、引き数リスト中のその位置によって、引き数は特定のパラメーターに対応します。

データベース・マネージャーは、呼び出しで指定されるメソッド名、メソッドに対する EXECUTE 特権、引き数の数とデータ型、サブジェクト引き数の静的タイプ (およびそのスーパータイプ) 用に同じ名前をもつすべてのメソッド、対応するパラメーターのデータ型を使用して、あるメソッドを選択するかどうかの判断基準とします。メソッドを決定する過程で発生する可能性のある結果について以下に示します。

- 特定のメソッドが最適であると判断される場合。たとえば、シグニチャーが以下のように定義されたタイプ SITE の RISK というメソッドを考えてみます。

  ```
  PROXIMITY(INTEGER) FOR SITE
  PROXIMITY(DOUBLE) FOR SITE
  ```

  続くメソッドの呼び出しは次のようになります (ここで、ST は SITE 列、DB は DOUBLE 列です)。

  ```
  SELECT ST..PROXIMITY(DB) ...
  ```

  この場合は、2 番目の PROXIMITY が選択されます。

  以下のようにメソッドが呼び出される場合は (SI は SMALLINT 列)、

  ```
  SELECT ST..PROXIMITY(SI) ...
  ```

  最初の PROXIMITY が選択されます。これは、SMALLINT は INTEGER にプロモート可能であり、優先順位リストでの順位がさらに下になる DOUBLE よりも一致性が高いためです。

  構造化タイプである引き数について考慮する場合、優先順位リストには、静的タイプの引き数のスーパータイプが入っています。最適なのは、構造化タイプ階層の中で、静的タイプの関数引き数に最も近いスーパータイプ・パラメーターで定義する関数です。
- 許容できる適合性をもつメソッドがないと判断される場合。たとえば、前出の例と同じ 2 つの関数が指定され、以下のように関数が参照されたとします (C は CHAR(5) 列)。

```
SELECT ST..PROXIMITY(C) ...
```

この場合、引き数はどちらの PROXIMITY 関数のパラメーターとも整合性がありません。

- タイプ階層のメソッド、および呼び出し時に渡される引き数の数とデータ型に基づいて特定のメソッドが選択される場合。たとえば、シグニチャーが以下のように定義された、タイプ SITE および DRILLSITE (SITE のサブタイプ) の RISK というメソッドを考えてみます。

```
RISK(INTEGER) FOR DRILLSITE
RISK(DOUBLE) FOR SITE
```

続くメソッドの呼び出しは次のようになります (ここで、DRST は DRILLSITE 列、DB は DOUBLE 列です)。

```
SELECT DRST..RISK(DB) ...
```

DRILLSITE は SITE へプロモートできるので、この場合は、2 番目の RISK が選択されます。

以下のようにメソッドが参照される場合は (SI は SMALLINT 列)、

```
SELECT DRST..RISK(SI) ...
```

最初の RISK が選択されます。これは、 SMALLINT は INTEGER にプロモート可能 (優先順位リストでの順位が DOUBLE よりも近い) であり、 DRILLSITE は、SITE よりも一致性が高いスーパータイプであるためです。

同じタイプ階層内のメソッドで同じシグニチャーを使い、サブジェクト・パラメーター以外のパラメーターで利用することはできません。

最適の判別

引き数のデータ型と、対象とするメソッドのパラメーターに定義されているデータ型との比較は、似通った名前のメソッドの中でどれが『最適』かを決定する基準となります。考慮しているメソッドの結果のデータ型は、この決定には関係しないことに注意してください。

メソッド解決は、以下の手順で実行されます。

1. まず、カタログ (SYSCAT.ROUTINES) から、以下のすべての条件が真となるすべてのメソッドを探します。
   - メソッド名が呼び出し名と同じで、サブジェクト・パラメーターが、サブジェクト引き数の静的タイプと同じであるか、そのスーパータイプである。
   - 呼び出し側がメソッドに対する EXECUTE 特権を持っている。
   - 定義済みパラメーターの数が呼び出しと一致している。
   - 呼び出しの各引き数のデータ型が、メソッドの対応する定義済みパラメーターのデータ型に一致するか、またはそのデータ型に『プロモート可能』である。
2. 次に、メソッド呼び出しの個々の引き数を左から右に検討していきます。左端の引き数 (すなわち最初の引き数) は、暗黙的な SELF パラメーターです。たとえば、タイプ ADDRESS_T に定義したメソッドには、暗黙的な最初のパラメーターとしてタイプ ADDRESS_T があります。引き数ごとに、その引き数に対して最適な一致ではない関数をすべて除去していきます。ある引き数の最適な一致と

は、その引き数データ型に対応する優先順位リストの中で、そのデータ型のパラメーターをもつ関数が存在するデータ型のうち、最初に記述されているデータ型です。長さ、精度、位取り、および FOR BIT DATA 属性は、この比較では考慮されません。たとえば、DECIMAL(9,1) の引き数は DECIMAL(6,5) のパラメーターと完全に一致すると見なされ、また VARCHAR(19) の引き数は VARCHAR(6) のパラメーターと完全に一致すると見なされます。

ユーザー定義構造化タイプ引き数に最適なのは、それ自身です。次に適しているのは、すぐ上のスーパータイプです。このことは、引き数の各スーパータイプに当てはまります。ここで考慮しているのは、静的タイプ (宣言済みタイプ) の構造化タイプ引き数であり、動的タイプ (最も特定的なタイプ) ではありません。

3. ほとんどの場合、ステップ 2 の実行後に候補メソッドが 1 つ残ります。このメソッドを選択します。
4. ステップ 2 の後で候補となるメソッドが残らなかった場合は、エラー (SQLSTATE 42884) になります。

メソッド解決の例

以下は、正常なメソッド解決の例を示しています。

GOVERNOR の階層内で定義された 3 つの構造化タイプには、 HEADOFSTATE のサブタイプとしての EMPEROR のサブタイプとして、 7 つの FOO メソッドがあります。それぞれ、以下のシグニチャーで登録されています。

```
CREATE METHOD FOO (CHAR(5), INT, DOUBLE)  FOR HEADOFSTATE SPECIFIC FOO_1 ...
CREATE METHOD FOO (INT, INT, DOUBLE)      FOR HEADOFSTATE SPECIFIC FOO_2 ...
CREATE METHOD FOO (INT, INT, DOUBLE, INT) FOR HEADOFSTATE SPECIFIC FOO_3 ...
CREATE METHOD FOO (INT, DOUBLE, DOUBLE)   FOR EMPEROR     SPECIFIC FOO_4 ...
CREATE METHOD FOO (INT, INT, DOUBLE)      FOR EMPEROR     SPECIFIC FOO_5 ...
CREATE METHOD FOO (SMALLINT, INT, DOUBLE) FOR EMPEROR     SPECIFIC FOO_6 ...
CREATE METHOD FOO (INT, INT, DEC(7,2))    FOR GOVERNOR    SPECIFIC FOO_7 ...
```

以下のようにメソッドが参照されるとします (I1 および I2 は INTEGER 列、 D は DECIMAL 列、そして E は EMPEROR 列です)。

```
SELECT E..FOO(I1, I2, D) ...
```

アルゴリズムに従っていきます。

- タイプ GOVERNOR は EMPEROR のサブタイプなので (スーパータイプではない)、 FOO_7 は候補から除かれます。
- パラメーターの数が違っているため、FOO_3 は候補から除かれます。
- 第 1 引き数 (サブジェクト引き数ではない) が第 1 パラメーターのデータ型にプロモートできないため、 FOO_1 と FOO_6 はどちらも候補から除かれます。この時点で複数の候補が残っているため、次に引き数が順に検討されます。
- サブジェクト引き数の場合、FOO_2 はスーパータイプですが、 FOO_4 と FOO_5 はサブジェクト引き数に一致しています。
- 最初の引き数については、残りのメソッド FOO_4 および FOO_5 がその引き数タイプと完全に一致します。この検討ではどのメソッドも検討の対象から除かれないため、次の引き数を検討する必要があります。
- 2 番目の引き数では、FOO_5 が完全に一致しているのに対し、 FOO_4 は一致していないため、FOO_4 が検討の対象から除かれます。これにより、メソッド FOO_5 が選ばれます。

メソッドの呼び出し

メソッドが選択された後も、いくつかの理由でそのメソッドの使用が許可されない場合があります。

個々のメソッドは特定のデータ型の結果を戻すように定義されています。この結果のデータ型が、メソッドが呼び出されるコンテキストと互換性がない場合、エラーが生じます。たとえば、STEP というメソッドが定義されていて、結果としてそれぞれが別々のデータ型を持っているとします。

```
STEP(SMALLINT) FOR TYPEA RETURNS CHAR(5)
STEP(DOUBLE) FOR TYPEA RETURNS INTEGER
```

以下のようにメソッドが参照されると (S は SMALLINT 列で TA は TYPEA の列)、

```
SELECT 3 + TA..STEP(S) ...
```

引き数タイプが完全に一致しているため、最初の STEP が選択されます。しかし、結果のタイプが加法演算子の引き数として求められる数値タイプではなく CHAR(5) であるため、ステートメントではエラーになります。

選択されたメソッドから、「メソッドの動的ディスパッチング」で記述されたアルゴリズムを使用して、コンパイル時にディスパッチ可能なメソッドのセットが構築されます。正確にどのメソッドが呼び出されるかは、「メソッドの動的ディスパッチング」で記述されます。

選択したメソッドがタイプ保持メソッドである場合、以下の点に注意してください。

- 関数解決に続く静的結果タイプは、メソッド呼び出しのサブジェクト引き数の静的タイプと同じです。
- メソッドを呼び出すときの動的結果タイプは、メソッド呼び出しのサブジェクト引き数の動的タイプと同じです。

これは、タイプ保持メソッド定義で指定した結果タイプのサブタイプになる可能性がありますが、逆に、メソッドの処理時に実際に戻される動的タイプのスーパータイプになる場合もあります。

メソッド呼び出しの引き数が、選択されたメソッドのパラメーターのデータ型と完全一致でない場合、列への割り当てと同じ規則を適用して、実行時に引き数がパラメーターのデータ型に変換されます。これには、引き数とパラメーターの間で精度、位取り、または長さが異なる場合も含まれます。ただし、引き数の動的タイプが、パラメーターの静的タイプのサブタイプである場合を除きます。

メソッドの動的ディスパッチング

メソッドは機能を提供し、あるタイプのデータをカプセル化します。メソッドはあるタイプに対して定義され、常にこのタイプと関連付けることができます。メソッドのパラメーターの 1 つは暗黙的な SELF パラメーターです。 SELF パラメーターは、メソッドが宣言されているタイプのパラメーターです。 DML ステートメントでメソッドが呼び出されるときに SELF 引き数として渡される引き数は、「サブジェクト」と呼ばれます。

メソッドがメソッド解決 (177 ページの『メソッド解決』を参照) によって選択されているか、あるいはメソッドが DDL ステートメントで指定されている場合、そのメソッドは「最も具体的な適用可能許可メソッド」として認識されます。サブジェクトが構造化タイプの場合、そのメソッドは 1 つ以上のオーバーライド・メソッドを持つ可能性があります。 DB2 は、ランタイムのサブジェクトの動的タイプ (最も具体的なタイプ) に基づいて、これらのどのメソッドを呼び出すかを決定しなければなりません。この決定は「最も具体的なディスパッチ可能メソッドの決定」と呼ばれます。このプロセスについて以下で説明します。

1. 最も具体的な適用可能許可メソッドを備えたメソッド階層で元のメソッドを見つけます。これは、*ルート・メソッド* と呼ばれます。
2. ディスパッチ可能メソッドのセットを作成します。これには以下が関与します。
   - 最も具体的な適用可能許可メソッド。
   - 最も具体的な適用可能許可メソッドをオーバーライドする任意のメソッド。これは、この呼び出しのサブジェクトのサブタイプであるタイプに対して定義されます。
3. 以下のように、最も具体的なディスパッチ可能メソッドを決定します。
   a. 一連のディスパッチ可能メソッドのエレメントで、サブジェクトの動的タイプか、そのいずれかのスーパータイプのメソッドである任意のメソッドを開始します。これは、初期の最も具体的なディスパッチ可能メソッドになります。
   b. 一連のディスパッチ可能メソッドのエレメントに関して上記を繰り返します。各メソッドについて、そのメソッドが、最も具体的なディスパッチ可能メソッドが定義されているタイプの適切なサブタイプのいずれかに対して定義されている場合、および、そのメソッドが、サブジェクトの最も具体的なタイプのスーパータイプのいずれかに対して定義されている場合は、そのメソッドを最も具体的なディスパッチ可能メソッドとして、ステップ 2 を繰り返します。
4. 最も具体的なディスパッチ可能メソッドを呼び出します。

例:

3 つのタイプ「Person」、「Employee」、および「Manager」があります。「Person」に対して定義された、人物の収入を計算する元のメソッド「income」があります。人物はデフォルトで無職 (子供や退職者など) です。したがって、タイプ「Person」に対する「income」は、常にゼロを戻します。タイプ「Employee」およびタイプ「Manager」の場合、収入を計算するには別のアルゴリズムを適用する必要があります。そのため、「Employee」および「Manager」では、「Person」に対するメソッド「income」がオーバーライドされます。

表を作成してデータを追加するには以下のようにします。

```
CREATE TABLE aTable (id integer, personColumn Person);
  INSERT INTO aTable VALUES (0, Person()), (1, Employee()), (2, Manager());
```

$40000 以上の収入を得ている人物をリストします。

```
SELECT id, person, name
  FROM aTable
  WHERE person..income() >= 40000;
```

タイプ「Person」に対するメソッド「income」は、メソッド解決によって、最も具体的な適用可能許可メソッドとして選択されています。

1. ルート・メソッドは「Person」に対する「income」自体です。
2. 上記のアルゴリズムの 2 番目のステップが実行され、一連のディスパッチ可能メソッドが構成されます。
   - タイプ「Person」に対するメソッド「income」は、最も具体的な適用可能許可メソッドなので、これは組み込まれます。
   - タイプ「Employee」に対するメソッド「income」とタイプ「Manager」に対するメソッド「income」の両メソッドは、ルート・メソッドをオーバーライドするもので、「Employee」と「Manager」は「Person」のサブタイプなので、これらのメソッドは組み込まれます。

   したがって、ディスパッチ可能メソッドのセットは、 {「Person」に対する「income」、「Employee」に対する「income」、「Manager」に対する「income」} です。
3. 最も具体的なディスパッチ可能メソッドを決定します。
   - 最も具体的なタイプが「Person」であるサブジェクトの場合:
     a. 初期の最も具体的なディスパッチ可能メソッドを、タイプ「Person」に対する「income」にします。
     b. ディスパッチ可能メソッドのセットには、「Person」の適切なサブタイプとサブジェクトの最も具体的なタイプのスーパータイプに対して定義された他のメソッドがないので、「Person」に対する「income」が最も具体的なディスパッチ可能メソッドになります。
   - 最も具体的なタイプが「Employee」であるサブジェクトの場合:
     a. 初期の最も具体的なディスパッチ可能メソッドを、タイプ「Person」に対する「income」にします。
     b. 一連のディスパッチ可能メソッドごとに繰り返します。タイプ「Employee」に対するメソッド「income」は、「Person」の適切なサブタイプとサブジェクト (注: タイプはこの独自のスーパータイプとサブタイプである) の最も具体的なタイプのスーパータイプに対して定義されているので、「Employee」に対するメソッド「income」が、最も具体的なディスパッチ可能メソッドにより合致するメソッドになります。タイプ「Employee」に対するメソッド「income」を最も適切なディスパッチ可能メソッドとしてこのステップを繰り返します。
     c. ディスパッチ可能メソッドのセットには、「Employee」の適切なサブタイプおよびサブジェクトの最も具体的なタイプのスーパータイプに対して定義された他のメソッドがないので、「Employee」に対するメソッド「income」が最も具体的なディスパッチ可能メソッドになります。
   - 最も具体的なタイプが「Manager」であるサブジェクトの場合:
     a. 初期の最も具体的なディスパッチ可能メソッドを、タイプ「Person」に対する「income」にします。
     b. 一連のディスパッチ可能メソッドごとに繰り返します。タイプ「Manager」に対するメソッド「income」は、「Person」の適切なサブタイプとサブジェクト (注: タイプはこの独自のスーパータイプとサブタイプである) の最も具体的なタイプのスーパータイプに対して定義されているので、

「Manager」に対するメソッド「income」が、最も具体的なディスパッチ可能メソッドにより合致するメソッドになります。タイプ「Manager」に対するメソッド「income」を最も適切なディスパッチ可能メソッドとしてこのステップを繰り返します。

c. ディスパッチ可能メソッドのセットには、「Manager」の適切なサブタイプおよびサブジェクトの最も具体的なタイプのスーパータイプに対して定義された他のメソッドがないので、「Manager」に対するメソッド「income」が最も具体的なディスパッチ可能メソッドになります。

4. 最も具体的なディスパッチ可能メソッドを呼び出します。

関連資料:

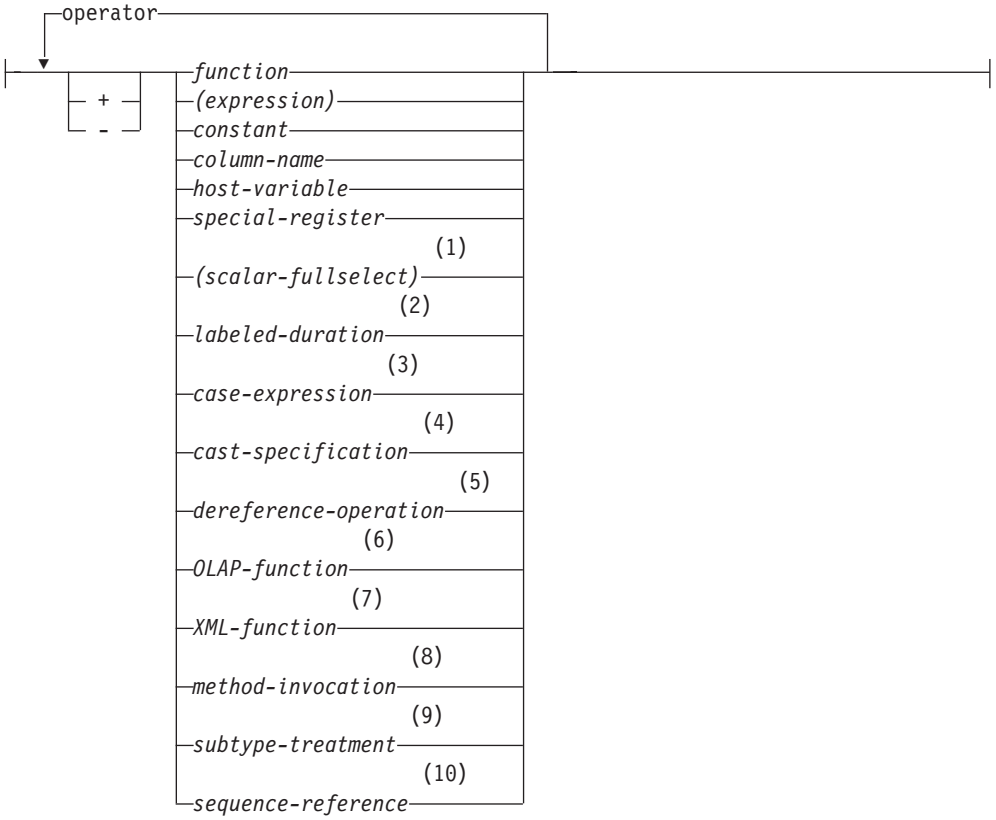
- 105 ページの『データ型のプロモーション』
- 110 ページの『割り当てと比較』

式

式は値を指定します。これは、定数や列名だけで構成される簡単な値にすることもでき、さらに複雑な値にすることも可能です。同じような複雑な式を繰り返し使用するのであれば、共通の式をカプセル化するための SQL 関数を検討対象とすることができます。

Unicode データベースでは、文字ストリングまたは GRAPHIC ストリングを受け入れる式は、変換をサポートされている任意のストリング・タイプを受け入れます。

expression:



operator:

注:

1 190 ページの『スカラー全選択』を参照。

2 191 ページの『ラベル付き期間』を参照。

3 196 ページの『CASE 式』を参照。

4 198 ページの『CAST 指定』を参照。

5 200 ページの『間接参照操作』を参照。

6 201 ページの『OLAP 関数』を参照。

7 207 ページの『XML 関数』を参照。

8 216 ページの『メソッドの呼び出し』を参照。

9 217 ページの『サブタイプの扱い』を参照。

10 218 ページの『シーケンス参照』を参照。

11 CONCAT の同義語として || を使用することができます。

演算子がない式

演算子を使用しない式では、指定した値が式の結果になります。

例:

```
SALARY:SALARY'SALARY'MAX(SALARY)
```

連結演算子がある式

連結演算子 (CONCAT) は、2 つのストリング・オペランドを連結して、 1 つのストリング式 にします。

連結するオペランドは、互換性のあるストリングでなければなりません。 FOR BIT DATA と定義されている文字ストリングも含め、バイナリー・ストリングを文字ストリングと連結することはできません (SQLSTATE 42884)。

Unicode データベースでは、文字ストリング・オペランドと GRAPHIC ストリング・オペランドの両方がかかわる連結の場合、まず文字オペランドが GRAPHIC オペランドに変換されます。非 Unicode データベースでは、文字と GRAPHIC の両方のオペランドが連結にかかわることはないことに注意してください。

いずれかのオペランドが NULL 値になる可能性がある場合は、結果も NULL 値になる可能性があり、いずれかが NULL 値なら結果は NULL 値になります。そうでない場合、結果は第 1 オペランド・ストリングの後に第 2 オペランド・ストリングが続いた形式となります。連結時に混合データが不正に形成されても、それに対する検査は行われません。

結果ストリングの長さは、オペランドの長さの合計になります。

結果のデータ型と長さ属性は、以下の表に示すように、オペランドのデータ型と長さ属性によって決まります。

表 12. 連結するオペランドのデータ型と長さ

オペランド	連結後の長さ属性	結果
CHAR(A) CHAR(B)	<255	CHAR(A+B)
CHAR(A) CHAR(B)	>254	VARCHAR(A+B)
CHAR(A) VARCHAR(B)	<4001	VARCHAR(A+B)
CHAR(A) VARCHAR(B)	>4000	LONG VARCHAR
CHAR(A) LONG VARCHAR	-	LONG VARCHAR

表 12. 連結するオペランドのデータ型と長さ (続き)

オペランド	連結後の長さ属性	結果
VARCHAR(A) VARCHAR(B)	<4001	VARCHAR(A+B)
VARCHAR(A) VARCHAR(B)	>4000	LONG VARCHAR
VARCHAR(A) LONG VARCHAR	-	LONG VARCHAR
LONG VARCHAR LONG VARCHAR	-	LONG VARCHAR
CLOB(A) CHAR(B)	-	CLOB(MIN(A+B, 2G))
CLOB(A) VARCHAR(B)	-	CLOB(MIN(A+B, 2G))
CLOB(A) LONG VARCHAR	-	CLOB(MIN(A+32K, 2G))
CLOB(A) CLOB(B)	-	CLOB(MIN(A+B, 2G))
GRAPHIC(A) GRAPHIC(B)	<128	GRAPHIC(A+B)
GRAPHIC(A) GRAPHIC(B)	>127	VARGRAPHIC(A+B)
GRAPHIC(A) VARGRAPHIC(B)	<2001	VARGRAPHIC(A+B)
GRAPHIC(A) VARGRAPHIC(B)	>2000	LONG VARGRAPHIC
GRAPHIC(A) LONG VARGRAPHIC	-	LONG VARGRAPHIC
VARGRAPHIC(A) VARGRAPHIC(B)	<2001	VARGRAPHIC(A+B)
VARGRAPHIC(A) VARGRAPHIC(B)	>2000	LONG VARGRAPHIC
VARGRAPHIC(A) LONG VARGRAPHIC	-	LONG VARGRAPHIC
LONG VARGRAPHIC LONG VARGRAPHIC	-	LONG VARGRAPHIC
DBCLOB(A) GRAPHIC(B)	-	DBCLOB(MIN(A+B, 1G))
DBCLOB(A) VARGRAPHIC(B)	-	DBCLOB(MIN(A+B, 1G))
DBCLOB(A) LONG VARGRAPHIC	-	DBCLOB(MIN(A+16K, 1G))
DBCLOB(A) DBCLOB(B)	-	DBCLOB(MIN(A+B, 1G))
BLOB(A) BLOB(B)	-	BLOB(MIN(A+B, 2G))

旧バージョンとの互換性を保つために、 LONG データ型に関連した結果は LOB データ型に自動的にエスカレーションされないことに注意してください。たとえば、CHAR(200) の値と、完全に文字の詰まった LONG VARCHAR の値とを連結した場合、 CLOB データ型へプロモートされるのではなくエラーになります。

結果のコード・ページは派生コード・ページと見なされ、そのオペランドのコード・ページによって決定されます。

一方のオペランドはパラメーター・マーカーにすることができます。パラメーター・マーカーが使用されている場合、そのオペランドのデータ型と長さ属性は、パラメーター・マーカーでないオペランドと同じであると見なされます。ネストした連結の場合、これらの属性を決定できるように演算の順序を考慮する必要があります。

例 1: FIRSTNME が Pierre で LASTNAME が Fermat である場合、以下のようになります。

```
FIRSTNME CONCAT ' ' CONCAT LASTNAME
```

Pierre Fermat の値が戻されます。

例 2: 以下を条件とします。

- COLA は、 'AA' の値を持つ VARCHAR(5) と定義されている。
- :host_var は、長さが 5 で値が 'BB ' である文字ホスト変数と定義されている。
- COLC は、値が 'CC' の CHAR(5) と定義されている。
- COLD は、値が 'DDDDD' の CHAR(5) と定義されている。

COLA CONCAT :host_var CONCAT COLC CONCAT COLD の値は、 'AABB CC DDDDD' です。

データ型が VARCHAR で、長さ属性は 17、結果コード・ページはデータベース・コード・ページとなります。

例 3: 以下を条件とします。

COLA は、CHAR(10) と定義する。

COLB は、VARCHAR(5) と定義する。

次の式の中のパラメーター・マーカーは、

```
COLA CONCAT COLB CONCAT ?
```

VARCHAR(15) と見なされます。これは、COLA CONCAT COLB が最初に評価され、その結果が 2 番目の CONCAT 演算の第 1 オペランドとなるためです。

ユーザー定義タイプ

ユーザー定義タイプは、ストリング・タイプのソース・データ型がある特殊タイプであっても、連結演算子は使用できません。 連結するためには、そのソースとしての CONCAT 演算子を使った関数を作成する必要があります。たとえば、TITLE と TITLE_DESCRIPTION という特殊タイプがあり、どちらも VARCHAR(25) データ型である場合は、以下に示すユーザー定義関数 ATTACH でそれらを連結することができます。

```
CREATE FUNCTION ATTACH (TITLE, TITLE_DESCRIPTION)
  RETURNS VARCHAR(50) SOURCE CONCAT (VARCHAR(), VARCHAR())
```

別の方法として、新規のデータ型を追加するユーザー定義関数を使用し、連結演算子を多重定義することもできます。

```
CREATE FUNCTION CONCAT (TITLE, TITLE_DESCRIPTION)
  RETURNS VARCHAR(50) SOURCE CONCAT (VARCHAR(), VARCHAR())
```

算術演算子がある式

算術演算子が使用されている場合、式の結果は、演算子をオペランドの値に適用して導かれた値となります。

いずれかのオペランドが NULL 値になる可能性がある場合、またはデータベースが DFT_SQLMATHWARN を yes に設定して構成されている場合、結果も NULL 値になる可能性があります。

どちらか一方のオペランドが NULL 値ならば、式の結果は NULL 値になります。

算術演算子は、符号付き数値タイプと日時タイプに適用できます (192 ページの『SQL における日付/時刻の算術演算』を参照)。たとえば、USER+2 は無効です。ソース関数については、符号付き数値タイプであるソース・タイプを持つ特殊タイプ上の算術演算子に定義できます。

接頭演算子、+ (単項加算) はそのオペランドを変更しません。接頭演算子、- (単項減算) は、ゼロ以外のオペランドの符号を逆にします。 A のデータ型が短整数である場合、 -A のデータ型は長精度整数になります。接頭演算子の後に続くトークンの先頭の文字は、正または負の符号であってはなりません。

挿入演算子 +、-、*、および / はそれぞれ、加算、減算、乗算、および除算を指定します。除算の第 2 オペランドの値はゼロにすることはできません。これらの演算子は関数としても扱われます。したがって、式 *"+"(a,b)* は、式 *a+b* の『演算子』の機能と同じ意味になります。

算術演算エラー

ゼロによる除算や数値のオーバーフローなどの算術演算エラーが、式の処理の過程で生じると、エラーが戻され、その式を処理する SQL ステートメントは失敗し、エラー (SQLSTATE 22003 または 22012) が出されます。

データベースは、算術演算エラーが生じた場合に式の値として NULL 値を戻すように構成することが可能で (DFT_SQLMATHWARN を yes に設定して)、警告 (SQLSTATE 01519 または 01564) を出して、その SQL ステートメントの処理を続けることができます。算術計算エラーが NULL 値として扱われる場合、SQL ステートメントの結果に影響があります。以下は、このような影響の例を示しています。

- 列関数の引き数の式で算術演算エラーが起きると、その列関数の結果を判別する際に行が無視されます。算術演算エラーがオーバーフローである場合、結果の値に大きな影響を与える場合があります。
- WHERE 文節の述部の式で算術演算エラーが起きると、結果に行が入っていない場合があります。
- チェック制約の述部の式で算術演算エラーが起きても、制約には誤りがないため更新または挿入は続行されます。

このようなタイプの影響が受け入れられない場合、算術演算エラーを処理するのに必要な他の処置を行って、受け入れ可能な結果を生成する必要があります。たとえば次のような処置です。

- ゼロによる除算の有無を検査するために CASE 式を追加して、このような状態に対応する必要な値を設定する。

- NULL 値を処理する述部を追加する (NULL 値が不能な列のチェック制約は次のようになります)。

```
check (c1*c2 is not null and c1*c2>5000)
```

(これにより、オーバーフローの制約に違反する場合があります。)

2 つの整数オペランド

算術演算子のオペランドが両方とも整数の場合、その演算はバイナリー数で実行され、いずれかの (または両方の) オペランドが 64 ビット整数 (big integer) でない限り、その結果は*長精度整数 (large integer)* になります。いずれかの (または両方の) オペランドが 64 ビット整数である場合は、結果は 64 ビット整数になります。除算の剰余は失われます。整数算術演算 (単項減算符号を含む) の結果は、結果タイプの範囲内でなければなりません。

整数と 10 進数オペランド

一方のオペランドが整数で、もう一方のオペランドが 10 進数の場合、その演算は、精度 *p* および位取り 0 の 10 進数に変換されたその整数の一時コピーを使用して、 10 進数で行われます。 *p* は、64 ビット整数 (big integer) の場合 19 であり、長精度整数 (large integer) の場合 11 であり、短整数 (small integer) の場合 5 です。

2 つの 10 進数オペランド

オペランドが両方とも 10 進数の場合、その演算は 10 進数で行われます。 10 進数の算術演算の結果は 10 進数であり、その結果の精度と位取りは、演算の種類およびオペランドの精度と位取りによって異なります。演算が加算または減算で、オペランドの位取りが同じでない場合は、オペランドの一方の一時コピーを使用して演算が行われます。短い方のオペランドの小数部分が、長い方のオペランドと同じ桁数になるように、短い方のオペランドのコピーに後続ゼロを加えて拡張されます。

10 進数演算の結果は、精度が 31 以下でなければなりません。 10 進数の加算、減算、および乗算の結果は、精度が 31 を超える一時結果から求められることがあります。一時結果の精度が 31 を超えない場合、最終結果は一時結果と同じです。

SQL での 10 進数演算

以下の公式により、SQL における 10 進数演算の結果の精度および位取りが決まります。記号 *p* と *s* は第 1 オペランドの精度と位取りを表し、記号 *p'* と *s'* は第 2 オペランドの精度と位取りを表します。

加算および減算

精度は min(31,max(*p-s,p'-s'*) +max(*s,s'*)+1) になります。加算および減算の結果の位取りは max(*s,s'*) です。

乗算

乗算結果の精度は (31,*p* +*p'*)、位取りは min(31,*s* +*s'*) です。

除算

除算結果の精度は 31 です。位取りは *31 -p +s -s '* です。位取りは負であってはなりません。

注: MIN_DEC_DIV_3 データベース構成パラメーターは、除法に関係する 10 進算術演算の位取りを変更します。パラメーター値を NO に設定した場合、位取りは *31 -p +s -s '* として計算されます。パラメーターを YES に設定した場合、位取りは MAX(3, *31 -p +s -s '*) として計算されます。これにより、10 進数の除算の結果は常に、少なくとも 3 の位取りを持つようになります (精度は常に 31 です)。

浮動小数点オペランド

算術演算子のいずれかのオペランドが浮動小数点の場合、演算は浮動小数点で行われ、必要に応じてオペランドが最初に倍精度の浮動小数点数に変換されます。したがって、式のエレメントのいずれかが浮動小数点数の場合、その式の結果は倍精度浮動小数点数になります。

浮動小数点数と整数に関与した演算は、倍精度浮動小数点に変換した整数の一時コピーを使って実行されます。浮動小数点数と 10 進数に関与した演算は、倍精度浮動小数点に変換した 10 進数の一時コピーを使って実行されます。浮動小数点数演算の結果は、浮動小数点数の範囲内でなければなりません。

オペランドとしてのユーザー定義タイプ

ユーザー定義タイプは、そのソース・データ型が数値であっても算術演算子には使用できません。 算術演算を実行するには、そのソースとしての算術演算子を使用する関数を作成する必要があります。たとえば、INCOME と EXPENSES という特殊タイプがあり、どちらも DECIMAL(8,2) データ型である場合は、以下に示すユーザー定義関数 REVENUE を使って一方からもう一方を減算することができます。

```
CREATE FUNCTION REVENUE (INCOME, EXPENSES)
  RETURNS DECIMAL(8,2) SOURCE "-" (DECIMAL, DECIMAL)
```

別の方法として、新規のデータ型を減算するユーザー定義関数を使って - (マイナス) 演算子を多重定義することも可能です。

```
CREATE FUNCTION "-" (INCOME, EXPENSES)
  RETURNS DECIMAL(8,2) SOURCE "-" (DECIMAL, DECIMAL)
```

スカラー全選択

式でサポートされるスカラー全*選択* は、括弧で囲まれる全選択であり、1 つの列値で構成される 1 つの行を戻します。全選択が行を戻さない場合、式の結果は NULL 値になります。選択リスト・エレメントが単なる列名か間接参照の式である場合、その列の名前に基づいて結果列の名前が付けられます。

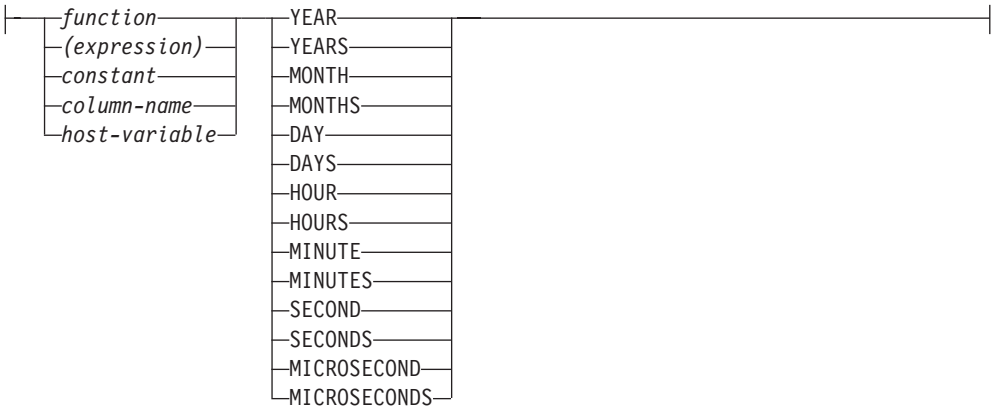
日付/時刻演算と期間

日付/時刻の値は、増分、減分、および減算を行うことができます。このような演算には、*期間* と呼ばれる 10 進数を伴う場合があります。期間の定義と、日付/時刻の算術演算に関する規則の仕様について以下に説明します。

期間とは、時間のインターバルを表す数値です。期間には以下の 4 つのタイプがあります。

ラベル付き期間

labeled-duration:



ラベル付き期間 (labeled-duration) は、特定の時間単位を表すもので、数値 (式の結果でも可) の後に 7 つの期間キーワード YEARS、 MONTHS、 DAYS、 HOURS、 MINUTES、 SECONDS、または MICROSECONDS のうちの 1 つを付けたものです。 (これらのキーワードの単数形 YEAR、MONTH、DAY、HOUR、MINUTE、SECOND、および MICROSECOND も可能です。) 指定した値は、DECIMAL(15,0) の数値へ割り当てられる場合と同様に変換されます。ラベル付き期間は、算術演算子の 1 つのオペランドとしてのみ使用でき、このときの他方のオペランドは DATE、TIME、または TIMESTAMP です。したがって、式 HIREDATE + 2 MONTHS + 14 DAYS は有効ですが、式 HIREDATE + (2 MONTHS + 14 DAYS) は有効ではありません。どちらの式でも 2 MONTHS と 14 DAYS がラベル付き期間です。

日付期間

日付期間 は、DECIMAL(8,0) の数値として表現される年数、月数、および日数を表します。正しく解釈されるには、この数値は *yyyymmdd* というフォーマットにする必要があります (*yyyy* は年数、 *mm* は月数、*dd* は日数を表します)。 (このフォーマットの期間は、DECIMAL データ型を示します。) 式 HIREDATE - BRTHDATE のように、ある日付値から別の日付値を減算した結果が日付期間です。

時刻期間

時刻期間 は、DECIMAL(6,0) の数値として表現される時間数、分数、および秒数を表します。正しく解釈されるには、この数値は *hhmmss.* というフォーマットにする必要があります (*hh* は時間数、 *mm* は分数、*ss* は秒数を表します)。 (このフォーマットの期間は、DECIMAL データ型を示します。) ある時刻値から別の時刻値を減算した結果が時刻期間です。

タイム・スタンプ期間

タイム・スタンプ期間 は、DECIMAL(20,6) の数値として表現され、年数、月数、日数、時間数、分数、秒数、およびマイクロ秒数を表します。正しく解釈されるには、この数値を *yyyymmddhhmmss.zzzzzz* というフォーマットにする必要があります

(*yyyy*、 *mm*、*dd*、 *hh*、*mm*、 *ss*、および *zzzzzz* はそれぞれ、年数、月数、日数、時間数、分数、秒数、およびマイクロ秒数を表します)。 あるタイム・スタンプ値から別のタイム・スタンプ値を減算した結果が、タイム・スタンプ期間です。

SQL における日付/時刻の算術演算

日付/時刻値に関して実行できる算術演算は加算と減算だけです。日付/時刻値が加算のオペランドである場合、他方のオペランドは期間でなければなりません。日付/時刻の値を使う加算演算子を使用するときには、次のような特有の規則があります。

- 一方のオペランドが日付である場合、もう一方のオペランドは日付期間、または YEARS、MONTHS、DAYS のラベル付き期間であることが必要です。
- 一方のオペランドが時刻である場合、もう一方のオペランドは時刻期間、または HOURS、MINUTES、SECONDS のラベル付き期間であることが必要です。
- 一方のオペランドがタイム・スタンプである場合、もう一方のオペランドは期間でなければなりません。この場合、期間のどのタイプでも有効です。
- 加算演算子のどちらのオペランドにも、パラメーター・マーカーは使用できません。

日付/時刻の値に減算演算子を使用する際の規則は、加算演算子の場合とは異なります。これは、日付/時刻の値を期間から引くことができないため、さらに 2 つの日付/時刻の値を差し引くことと期間を日付/時刻の値から差し引くこととは異なるためです。日付/時刻の値を使う減算演算子を使用するときには、次のような特有の規則があります。

- 第 1 オペランドが日付の場合、第 2 オペランドは日付、日付期間、日付のストリング表記、または YEARS、MONTHS、DAYS のラベル付き期間であることが必要です。
- 第 2 オペランドが日付の場合、第 1 オペランドは、日付または日付のストリング表記であることが必要です。
- 第 1 オペランドが時刻の場合、第 2 オペランドは、時刻、時刻期間、時刻のストリング表記、または HOURS、MINUTES、SECONDS のラベル付き期間であることが必要です。
- 第 2 オペランドが時刻の場合、第 1 オペランドは、時刻または時刻のストリング表記であることが必要です。
- 第 1 オペランドがタイム・スタンプの場合、第 2 オペランドは、タイム・スタンプまたはタイム・スタンプのストリング表記、または期間であることが必要です。
- 第 2 オペランドがタイム・スタンプの場合、第 1 オペランドは、タイム・スタンプまたはタイム・スタンプのストリング表記であることが必要です。
- 減算演算子のどちらのオペランドにも、パラメーター・マーカーは使用できません。

日付の算術演算

日付は、減算、増分、および減分を行うことができます。

日付の減算: ある日付 (DATE2) を別の日付 (DATE1) から減算した結果は、これら 2 つの日付の間の年数、月数、日数を示す日付期間です。結果のデータ型は DECIMAL(8,0) です。 DATE1 が DATE2 以上の場合、DATE1 から DATE2 が減

算されます。これに対し、DATE1 が DATE2 より小さい場合は、 DATE2 から DATE1 が減算され、結果の符号が負になります。演算 RESULT = DATE1 - DATE2 の実行ステップを、以下に順に示します。

```
If DAY(DATE2) <= DAY(DATE1)
then DAY(RESULT) = DAY(DATE1) - DAY(DATE2).

If DAY(DATE2) > DAY(DATE1)
then DAY(RESULT) = N + DAY(DATE1) - DAY(DATE2)
where N = the last day of MONTH(DATE2).
MONTH(DATE2) is then incremented by 1.

If MONTH(DATE2) <= MONTH(DATE1)
then MONTH(RESULT) = MONTH(DATE1) - MONTH(DATE2).

If MONTH(DATE2) > MONTH(DATE1)
then MONTH(RESULT) = 12 + MONTH(DATE1) - MONTH(DATE2).
YEAR(DATE2) is then incremented by 1.

YEAR(RESULT) = YEAR(DATE1) - YEAR(DATE2).
```

たとえば、DATE('3/15/2000') - '12/31/1999' の結果は 00000215 になります。 (すなわち、0 年 2 か月 15 日の期間です。)

日付の増分と減分: 日付に期間を加算したり、日付から期間を減算したりすると、結果自体は日付となります。 (この演算では、月はカレンダーのページに相当します。つまり、日付に月を加算することは、その日付のページから順にカレンダーをめくっていくようなものです。) 結果は、0001 年 1 月 1 日以後 9999 年 12 月 31 日以前の日付となる必要があります。

年の期間を加算または減算する場合、影響を受けるのは日付の年の部分だけです。月も日も変更されませんが、その結果がうるう年でない年の 2 月 29 日となった場合は別です。その場合は日が 28 に変更され、SQLCA の警告標識が日付調整の発生を示すように設定されます。

同様に、月の期間を加算または減算する場合、影響を受けるのは月の部分だけです。ただし、必要に応じて年の部分にも影響が及びます。日付の日の部分は変更されませんが、結果が無効な場合 (たとえば 9 月 31 日など) は別です。その場合は日がその月の最後の日に設定され、 SQLCA の警告標識が日付調整の発生を示すように設定されます。

日の期間を加算または減算すると、日付の中の日の部分は当然影響を受けますが、月および年も影響を受ける可能性があります。

日付期間も、正負にかかわらず、日付に対して加減算が行えます。ラベル付き期間の場合と同じように、結果は有効な日付となり、月末の調整が必要になれば SQLCA の警告標識が設定されます。

正の日付期間が日付に加算されるとき、または負の日付期間が日付から減算されるときは、日付は、指定した年数、月数、日数の順で増分されます。したがって、X が正の DECIMAL(8,0) の数値であるとき、 DATE1 + X は以下の式と同等です。

```
DATE1 + YEAR(X) YEARS + MONTH(X) MONTHS + DAY(X) DAYS.
```

正の日付期間を日付から減算するとき、または負の日付期間を日付に加算するとき、日付は、指定した日数、月数、年数の順で減分されます。したがって、X が正の DECIMAL(8,0) の数値であるとき、 DATE1 - X は以下の式と同等です。

```
DATE1 - DAY(X) DAYS - MONTH(X) MONTHS - YEAR(X) YEARS.
```

期間を日付に加算するとき、特定の日付に 1 か月を加算すると、1 か月後の同じ日付になります。ただし、1 か月後にその日付が存在しない場合は扱いが異なります。その場合、日付は 1 か月後の最後の日に設定されます。たとえば、1 月 28 日に 1 か月を加えると 2 月 28 日になります。 1 月 29、30、または 31 日に 1 か月を加えると通常の年では 2 月 28 日、うるう年では 2 月 29 日になります。

注: 特定の日付に 1 か月以上の月数を加算し、その結果から同じ月数を減算した場合、最終的な日付が元の日付と同じになるとは限りません。

時刻の算術演算

時刻は、減算、増分、または減分を行うことができます。

時刻値の減算: ある時刻 (TIME2) を別の時刻 (TIME1) から減算した結果は、それら 2 つの時刻の間の時間数、分数、秒数を示す時刻期間です。結果のデータ型は DECIMAL(6,0) です。

TIME1 が TIME2 と同じかまたはそれより大きい場合、 TIME1 から TIME2 が引かれます。

これに対し、TIME1 が TIME2 より小さい場合は、TIME2 から TIME1 が減算され、結果の符号が負になります。演算 RESULT = TIME1 - TIME2 の実行ステップを、以下に順に示します。

```
If SECOND(TIME2) <= SECOND(TIME1)
then SECOND(RESULT) = SECOND(TIME1) - SECOND(TIME2).

If SECOND(TIME2) > SECOND(TIME1)
then SECOND(RESULT) = 60 + SECOND(TIME1) - SECOND(TIME2).
MINUTE(TIME2) is then incremented by 1.

If MINUTE(TIME2) <= MINUTE(TIME1)
then MINUTE(RESULT) = MINUTE(TIME1) - MINUTE(TIME2).

If MINUTE(TIME1) > MINUTE(TIME1)
then MINUTE(RESULT) = 60 + MINUTE(TIME1) - MINUTE(TIME2).
HOUR(TIME2) is then incremented by 1.

HOUR(RESULT) = HOUR(TIME1) - HOUR(TIME2).
```

たとえば、TIME('11:02:26') - '00:32:56' の結果は 102930 になります。 (10 時間 29 分、30 秒の期間です。)

時刻値の増分と減分: 時刻に期間を加算したり、時刻から期間を減算したりすると、結果自体は時刻となります。時間数のオーバーフローやアンダーフローは捨てられ、これにより常に結果が時刻となります。時間数で指定する期間を加算または減算する場合、影響を受けるのは時間数の部分だけです。分数と秒数は変更されません。

同様に、分数で指定する期間を加算または減算する場合、影響を受けるのは分の部分だけです。ただし、必要に応じて時間数の部分にも影響が及びます。時刻の秒の部分は変更されません。

秒の期間を加算または減算すると、時刻の中の秒の部分は当然影響を受けますが、分および時も影響を受ける可能性があります。

時刻期間も、正負にかかわらず、時刻との加減算を行えます。結果は、指定した時間数、分数、秒数の順に増分または減分された時刻となります。 TIME1 + X (“X” は DECIMAL(6,0)) は次の式と同等です。

```
TIME1 + HOUR(X) HOURS + MINUTE(X) MINUTES + SECOND(X) SECONDS
```

注: 時刻 ’24:00:00’ は有効な値として受け付けられますが、時刻の加減算の結果として戻されることはありません。これは、期間オペランドがゼロであっても同じです (たとえば、時刻 (’24:00:00’)±0秒 =’00:00:00’ となります)。

タイム・スタンプの算術演算

タイム・スタンプは、減算、増分、または減分を行うことができます。

タイム・スタンプの減算: あるタイム・スタンプ (TS2) を別のタイム・スタンプ (TS1) から減算した結果は、それら 2 つのタイム・スタンプの間の年数、月数、日数、時間数、分数、秒数、およびマイクロ秒数を示すタイム・スタンプ期間です。結果のデータ型は DECIMAL(20,6) です。

TS1 が TS2 以上の場合、TS1 から TS2 が減算されます。これに対し、TS1 が TS2 より小さい場合は、TS2 から TS1 が減算され、結果の符号が負になります。演算 RESULT = TS1 - TS2 の実行ステップを、以下に順に示します。

```
If MICROSECOND(TS2) <= MICROSECOND(TS1)
then MICROSECOND(RESULT) = MICROSECOND(TS1) -
MICROSECOND(TS2).

If MICROSECOND(TS2) > MICROSECOND(TS1)
then MICROSECOND(RESULT) = 1000000 +
MICROSECOND(TS1) - MICROSECOND(TS2)
and SECOND(TS2) is incremented by 1.
```

タイム・スタンプの秒および分の部分は、時刻の減算規則で指定されたように減算されます。

```
If HOUR(TS2) <= HOUR(TS1)
then HOUR(RESULT) = HOUR(TS1) - HOUR(TS2).

If HOUR(TS2) > HOUR(TS1)
then HOUR(RESULT) = 24 + HOUR(TS1) - HOUR(TS2)
and DAY(TS2) is incremented by 1.
```

タイム・スタンプの日付の部分は、日付の減算規則での説明と同じようにして減算されます。

タイム・スタンプの増分と減分: タイム・スタンプに期間を加算したり、タイム・スタンプから期間を減算したりすると、結果自体はタイム・スタンプとなります。日付と時刻の算術演算はすでに説明したとおりに実行されますが、時間数のオーバーフローとアンダーフローは結果の日付の部分に繰り上げまたは繰り下げられ、有効な日付の範囲内に収められます。マイクロ秒のオーバーフローは秒に繰り上げられます。

演算の優先順位

括弧の中の式および間接参照操作が、最初に左から右へと評価されます。 (括弧は、subselect ステートメントや、検索条件、関数でも使用される点に注意してください。ただし、SQL ステートメント内でセクションを任意にグループ分けするのに使用することはできません。) 評価の順序が括弧で指定されていない場合は、まず

接頭演算子が乗算および除算に先立って行われ、次に乗算と除算が加算および減算に先立って行われます。同じ優先順位の演算子は左から右に行われます。

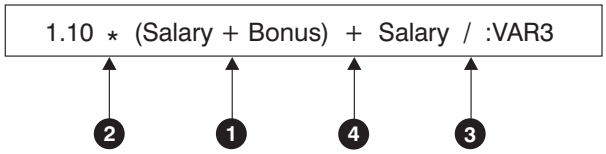
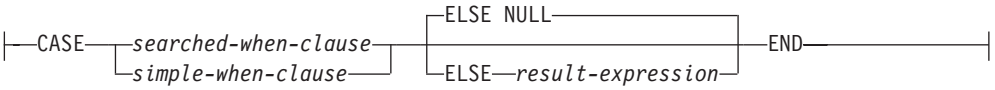


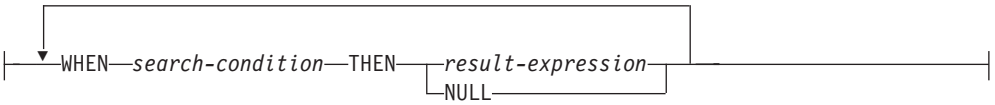
図 11. 演算の優先順位

CASE 式

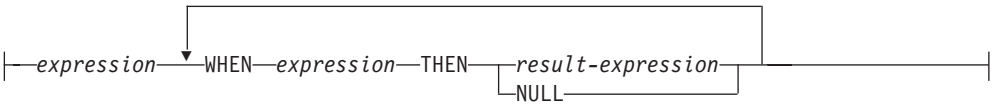
case-expression:



searched-when-clause:



simple-when-clause:



CASE 式は、1 つ以上の条件の評価に基づいて式を選択するためのものです。一般に、CASE 式の値は、評価が「真」である最初の (左端の) ケースの後に来る *result-expression* (結果式) の値になります。評価が「真」であるケースがなく、ELSE キーワードが指定されている場合、結果は ELSE の *result-expression* (結果式) または NULL になります。評価が「真」であるケースがなく、ELSE キーワードが指定されていない場合、結果は NULL になります。あるケースの評価が「不明」の場合 (NULL のため)、そのケースは「真」ではなく、したがって評価が「偽」であるケースと同じように扱われます。

CASE 式が VALUES 文節、IN 述部、GROUP BY 文節、または ORDER BY 文節中にある場合、 searched-when-clause の *search-condition* は、比較述部、全選択を使用する IN 述部、または EXISTS 述部 (SQLSTATE 42625) にすることはできません。

simple-when-clause (単純 WHEN 文節) を使用する場合は、最初の *WHEN* キーワードの前の *expression* (式) の値が、その WHEN キーワードの後にある *expression* の値と等しいかどうかが検査されます。このため、最初の *WHEN* キーワードの前の *expression* は、 WHEN キーワードの後に来るそれぞれの *expression* のデータ型と互換である必要があります。 *simple-when-clause* の中の最初の *WHEN* キーワードの前にある *expression* で、可変の関数または外部処理を伴う関数を使用することはできません (SQLSTATE 42845)。

result-expression (結果式) は、 THEN または ELSE キーワードの後に指定する式です。 CASE 式では、少なくとも 1 つの *result-expression* を指定する必要があります (すべてのケースに NULL を指定することはできません) (SQLSTATE 42625)。すべての結果式のデータ型は互換でなければなりません (SQLSTATE 42804)。

例:

- 以下の例では、部門番号の先頭文字が組織内の部を示すものとし、 CASE 式を使用して、各社員が属する部の正式名称を取り出します。

```
SELECT EMPNO, LASTNAME,
  CASE SUBSTR(WORKDEPT,1,1)
    WHEN 'A' THEN 'Administration'
    WHEN 'B' THEN 'Human Resources'
    WHEN 'C' THEN 'Accounting'
    WHEN 'D' THEN 'Design'
    WHEN 'E' THEN 'Operations'
  END
  FROM EMPLOYEE;
```

- 就学年数は、学歴レベルを示す目的で EMPLOYEE 表で使用されています。 CASE 式を使用して、これらを分類し、学歴レベルを示します。

```
SELECT EMPNO, FIRSTNME, MIDINIT, LASTNAME,
  CASE
    WHEN EDLEVEL < 15 THEN 'SECONDARY'
    WHEN EDLEVEL < 19 THEN 'COLLEGE'
    ELSE 'POST GRADUATE'
  END
  FROM EMPLOYEE
```

- CASE ステートメントの別の有効な使い方として、ゼロ除算によるエラーを防止することができます。たとえば以下のコードは、収入のすべてではないが 25% より多くを歩合で得ている社員を検索しています。

```
SELECT EMPNO, WORKDEPT, SALARY+COMM FROM EMPLOYEE
WHERE (CASE WHEN SALARY=0 THEN NULL
  ELSE COMM/SALARY
  END) > 0.25;
```

- 以下の 2 つの CASE 式は同じものです。

```
SELECT LASTNAME,
  CASE
  WHEN LASTNAME = 'Haas' THEN 'President'
  ...

SELECT LASTNAME,
  CASE LASTNAME
  WHEN 'Haas' THEN 'President'
  ...
```

CASE の機能の一部を処理する目的で、スカラー関数の NULLIF と COALESCE が特別に用意されています。 表 13 に、 CASE を使用した場合とそれらの関数を使用した場合とで同等の式を示します。

表 13. 同等の CASE 式

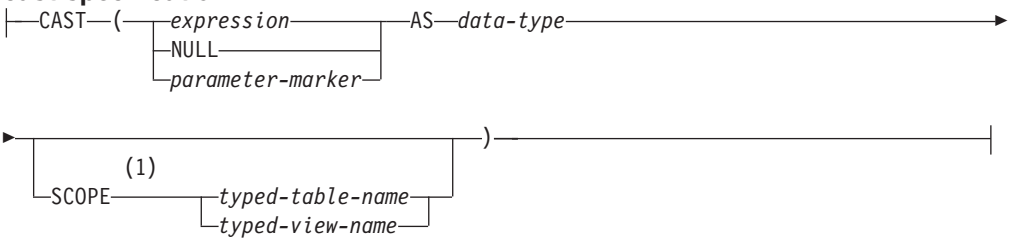
式	同等の式
CASE WHEN e1=e2 THEN NULL ELSE e1 END	NULLIF(e1,e2)
CASE WHEN e1 IS NOT NULL THEN e1 ELSE e2 END	COALESCE(e1,e2)

表 13. 同等の CASE 式 (続き)

式	同等の式
CASE WHEN e1 IS NOT NULL THEN e1 ELSE COALESCE(e2,...,eN) END	COALESCE(e1,e2,...,eN)

CAST 指定

cast-specification:



注:

1 SCOPE 文節が適用されるのは、REF データ型のみです。

CAST 指定は、*データ型* によって指定されたタイプにキャストされたキャスト・オペランド (第 1 オペランド) を戻します。キャストがサポートされていない場合、エラー (SQLSTATE 42846) が戻されます。

expression
キャスト・オペランドが式 (パラメーター・マーカーまたは NULL ではなく) である場合、結果は、指定された目的*データ型* に変換された引き数値です。

文字ストリング (CLOB 以外) を長さの異なる文字ストリングにキャストするとき、後続ブランク以外の文字が切り捨てられると、警告 (SQLSTATE 01004) が戻されます。 GRAPHIC ストリング (DBCLOB 以外) を長さの異なる GRAPHIC ストリングにキャストするとき、後続ブランク以外の文字が切り捨てられると、警告 (SQLSTATE 01004) が戻されます。キャスト・オペランドが BLOB、CLOB、および DBCLOB の場合、何らかの文字が切り捨てられると警告が発行されます。

NULL
キャスト・オペランドがキーワード NULL である場合、結果は、指定されたデータ*型* の NULL 値です。

parameter-marker
パラメーター・マーカー (疑問符で指定されるもの) は通常は式として見なされますが、ここでは特別な意味をもつため別個に説明します。キャスト・オペランドがパラメーター・マーカー である場合、指定されたデータ*型* は、指定されたデータ型に置き換えが割り当て可能である (ストリングの記憶割り当てを使用して) ことを示す合意であると見なされます。このようなパラメーター・マーカーは、*型付きパラメーター・マーカー* と見なされます。型付きパラメーター・マーカーは、関数解決、選択リストの DESCRIBE、または列割り当てを行う目的で、他の型付き値と同じように扱われます。

data type
既存のデータ型の名前。このタイプ名が修飾されていない場合は、 SQL パスを

使用してデータ型が解決されます。長さ、精度、および位取りなどの関連する属性を伴うデータ型には、データ型 の指定時にこのような属性を組み込む必要があります (指定されていない場合、 CHAR は長さ 1 にデフォルト解釈され、DECIMAL は精度 5 および位取り 0 にデフォルト解釈されます)。サポートされるデータ型に関する制限は、指定したキャスト・オペランドに基づいて適用されます。

- キャスト・オペランドが式 の場合にサポートされるターゲット・データ型は、キャスト・オペランドのデータ型 (ソース・データ型) によって異なります。
- キャスト・オペランドがキーワード NULL の場合、既存のどのデータ型でも指定できます。
- キャスト・オペランドがパラメーター・マーカーの場合、ターゲット・データ型は、既存の任意のデータ型とすることができます。データ型がユーザー定義特殊タイプの場合、パラメーター・マーカーを使用するアプリケーションは、そのユーザー定義特殊タイプのソース・データ型を使用します。データ型がユーザー定義構造化タイプの場合、パラメーター・マーカーを使用するアプリケーションは、そのユーザー定義構造化タイプの TO SQL トランスフォーム関数の入力パラメーター・タイプを使用します。

SCOPE

データ型が参照タイプの場合、有効範囲は参照のターゲット表またはターゲット・ビューを識別するように定義することができます。

typed-table-name

タイプ表の名前。表名はすでに指定されていなければなりません (SQLSTATE 42704)。キャストは *data-type* REF(*S*) にするものでなければなりません。ここでの *S* は *typed-table-name* (SQLSTATE 428DM) のタイプを表しています。

typed-view-name

タイプ・ビューの名前。そのビューは存在しているか、あるいはビュー定義の一部としてキャストを組み込むように作成されているビューと同じ名前でなければなりません (SQLSTATE 42704)。キャストは *data-type* REF(*S*) にするものでなければなりません。ここでの *S* は *typed-view-name* (SQLSTATE 428DM) のタイプを表しています。

数値データを文字データにキャストする場合、結果のデータ型は固定長文字ストリングです。文字データを数値データにキャストする場合、結果のデータ型は指定した数値のタイプによって異なります。たとえば整数へのキャストの場合、結果のデータ型は長精度整数になります。

例:

- アプリケーションが、 EMPLOYEE 表の SALARY (decimal(9,2) と定義) の整数部だけを使用するとします。社員番号や SALARY の整数値を備えた、以下のような照会が考えられます。

```
SELECT EMPNO, CAST(SALARY AS INTEGER) FROM EMPLOYEE
```

- SMALLINT に基づいて定義された T_AGE という名前の特殊タイプがあり、PERSONNEL 表に AGE 列を作成するために使用されるとします。さらに INTEGER に基づいて定義された R_YEAR という名前の特殊タイプがあり、

PERSONNEL 表に RETIRE_YEAR 列を作成するために使用されるとします。以下のような更新ステートメントが考えられます。

```
UPDATE PERSONNEL SET RETIRE_YEAR =?
  WHERE AGE = CAST( ? AS T_AGE)
```

第 1 パラメーターは、データ型 R_YEAR のタイプなしパラメーター・マーカーです。一方、アプリケーションはこのパラメーター・マーカーの整数部を使用します。この場合、これは割り当てなので、明示的な CAST 指定をする必要はありません。

2 番目のパラメーター・マーカーは、特殊タイプ T_AGE としてキャストされる型付きパラメーター・マーカーです。これにより、比較は互換データ型との間でなければならない、という要件が満たされます。アプリケーションは、ソース・データ型 (SMALLINT) を使用してこのパラメーター・マーカーを処理します。

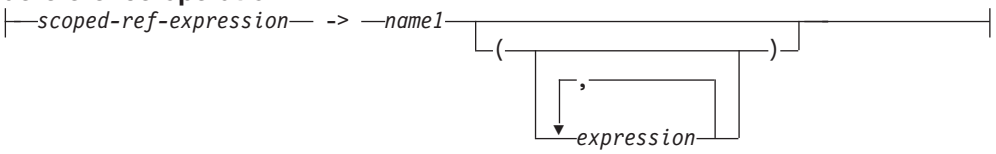
このステートメントの正常な処理では、関数パスには、 2 つの特殊タイプを定義した 1 つ以上のスキーマのスキーマ名が入っていることを前提としています。

- アプリケーションは、たとえばオーディオ・ストリームのような一つながりの値を提供しますが、その値は SQL ステートメントで使用される前にコード・ページの変換を経由してはなりません。アプリケーションは、次のような CAST を使用することができます。

```
CAST( ? AS VARCHAR(10000) FOR BIT DATA)
```

間接参照操作

dereference-operation:



有効範囲を指定した参照式の有効範囲は、ターゲット 表またはビューという表またはビューになります。効力範囲を指定した参照式は、*ターゲット行* を識別します。*ターゲット行* は、ターゲット表またはビューの (あるいは、副表かサブビューのいずれかの) 行です。この行のオブジェクト ID (OID) 列値は、参照式と一致しています。間接参照操作を使い、ターゲット行の列にアクセスしたり、そのターゲット行をメソッドのサブジェクトとして使用してメソッドを呼び出すことができます。間接参照操作の結果は、常に NULL になり得ます。 間接参照操作は、他のすべての操作よりも優先されます。

scoped-ref-expression
有効範囲を持っている参照タイプである式 (SQLSTATE 428DT)。式がホスト変数、パラメーター・マーカー、またはほかの有効範囲がない参照タイプ値の場合、 SCOPE 文節での CAST 指定で有効範囲の参照を指定する必要があります。

name1
修飾なしの ID を指定します。

name1 の後に括弧がなく、 *name1* がターゲット・タイプの属性名と一致している場合、間接参照操作の値は、ターゲット行の名前付き列の値になります。その場合、列のデータ型 (NULL 可能) の値により、間接参照操作の結果タイプが決まります。オブジェクト ID が参照式と一致するターゲット行がない場合、間接参照操作の結果は NULL になります。間接参照操作が選択リストで使用され、式の一部分としては組み込まれていない場合、 *name1* が結果の列名になります。

name1 の後に括弧があるか、*name1* がターゲット・タイプの属性名と一致しない場合、間接参照操作はメソッド呼び出しとして扱われます。呼び出されるメソッドの名前は *name1* です。メソッドのサブジェクトは、ターゲット行であり、その構造化タイプのインスタンスと見なされます。オブジェクト ID が参照式と一致するターゲット行がない場合、メソッドのサブジェクトは、ターゲット・タイプの NULL 値になります。括弧内に式があれば、それはメソッド呼び出しの残りのパラメーターを指定するものです。メソッド呼び出しの解決には、通常の処理が使われます。選択したメソッドの結果タイプ (NULL 可能) の値により、間接参照操作の結果タイプが決まります。

間接参照操作を使用するステートメントの許可 ID は、 *scoped-ref-expression* のターゲット表での SELECT 特権を持っていなければなりません (SQLSTATE 42501)。

間接参照操作では、データベース内の値を変更できません。間接参照操作を使用して変更メソッドを呼び出す場合、その変更メソッドはターゲット行のコピーを変更してコピーを戻しますが、データベースは未変更のままです。

例:

- DEPTREF という列がある EMPLOYEE 表 (属性 DEPTNAME を持っているタイプに基づくタイプ表を効力範囲とする参照タイプ) があるとします。表 EMPLOYEE の DEPTREF の値は、 DEPTREF 列のターゲット表にある OID 列値と対応していなければなりません。

```
SELECT EMPNO, DEPTREF->DEPTNAME
  FROM EMPLOYEE
```

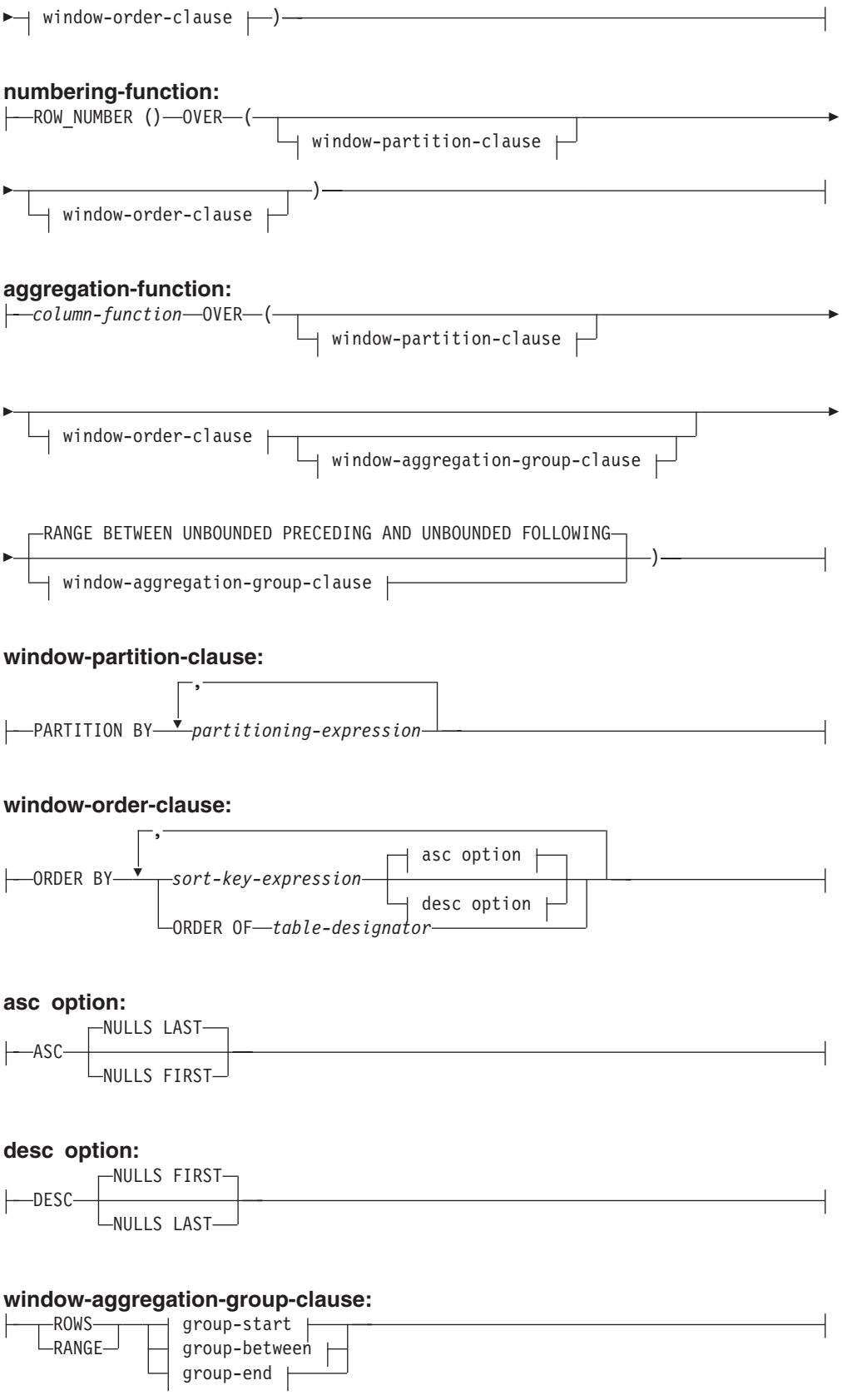
- 前の例と同じ表を使用し、間接参照操作を使って BUDGET というメソッドを呼び出します。その際に、ターゲット行をサブジェクト・パラメーターとして、そして '1997' を追加パラメーターとして指定します。

```
SELECT EMPNO, DEPTREF->BUDGET('1997') AS DEPTBUDGET97
  FROM EMPLOYEE
```

OLAP 関数

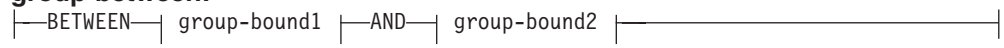
OLAP-function:

ranking-function:


window-order-clause
)
numbering-function:
ROW_NUMBER ()
OVER
(
window-partition-clause
window-order-clause
)
aggregation-function:
column-function
OVER
(
window-partition-clause
window-order-clause
window-aggregation-group-clause
RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING
window-aggregation-group-clause
)
window-partition-clause:
PARTITION BY
,
partitioning-expression
window-order-clause:
ORDER BY
,
sort-key-expression
asc option
desc option
ORDER OF
table-designator
asc option:
ASC
NULLS LAST
NULLS FIRST
desc option:
DESC
NULLS FIRST
NULLS LAST
window-aggregation-group-clause:
ROWS
RANGE
group-start
group-between
group-end

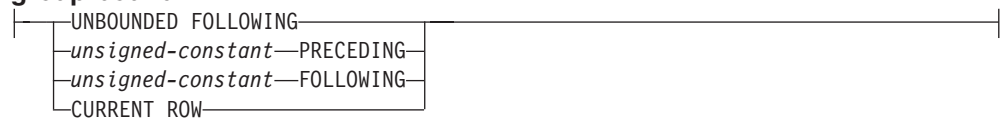
group-start:

group-between:

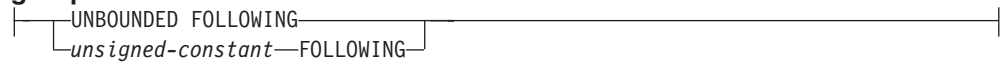


group-bound1:

group-bound2:



group-end:



OLAP (On-Line Analytical Processing) 関数には、照会の結果の中で、ランキング、行番号、および既存の列関数情報をスカラー値で戻す機能があります。 OLAP 関数は、select-list の式、または select-statement の ORDER BY 文節に組み込むことができます (SQLSTATE 42903)。 OLAP 関数を列関数の引き数として使うことはできません (SQLSTATE 42607)。 OLAP 関数を適用したときの照会の結果は、その OLAP 関数の入った最も内側の副選択の結果表です。

OLAP 関数を指定するときには、関数を適用する行を定義したり、その順序を定義する枠が指定されます。列関数とともに使用すると、該当する行をさらに詳細化して、現在行との相対関係で、その前後の行範囲または行数として扱うことができます。たとえば、月単位のパーティションでは、直前の四半期の平均を計算することができます。

ランキング関数は、枠内の行の序数ランクを計算します。それぞれの枠内での順序がはっきりしていない行は、同位に割り当てられます。ランキングの結果については、重複する値の結果の数値にギャップがあってもなくても定義できます。

RANK を指定すると、該当行に先行する行数に 1 を足した数で、行のランクが定義されます。したがって、順序がはっきりしていない行が 2 行以上あると、通しランク番号には、1 つ以上のギャップができます。

DENSE_RANK (または DENSERANK) を指定すると、順序の明確な先行行数に 1 を足して行のランクが定義されます。したがって、通しランク番号にはギャップはありません。

ROW_NUMBER (または ROWNUMBER) 関数は、最初の行を 1 行目とする順序付けで定義された枠内の行の通し番号を計算します。枠内で ORDER BY 文節を指定していない場合、 (SELECT ステートメントの ORDER BY 文節に基づくのではなく) 副選択で戻されたとおりに、任意の順番で行に行番号が割り当てられます。

RANK、DENSE_RANK、または ROW_NUMBER の結果のデータ型は BIGINT です。結果が NULL 値になることはありません。

PARTITION BY (*partitioning-expression***,...)**
関数を適用するときのパーティションを定義します。 *partitioning-expression* は、結果セットのパーティションを定義するときに使う式です。 partitioning-expression で参照されている各 *column-name* は、 OLAP 関数 subselect ステートメントの結果セット列をはっきり参照するものでなければなりません (SQLSTATE 42702 または 42703)。 partitioning-expression には、scalar-fullselect (SQLSTATE 42822) や、可変の関数または外部処理を伴う関数 (SQLSTATE 42845) を入れることはできません。

ORDER BY (*sort-key-expression***,...)**
OLAP 関数の値、または window-aggregation-group-clause の ROW 値の意味を決める、パーティション内の行の順序を定義します (照会結果セットの順序を定義するものではありません)。

sort-key-expression
枠のパーティション内の行の順序を定義するのに使う式。 sort-key-expression で参照されている各 column-name は、 OLAP 関数を含む副選択の結果セットの列をはっきり参照するものでなければなりません (SQLSTATE 42702 または 42703)。 sort-key-expression には、scalar-fullselect (SQLSTATE 42822) や、可変の関数または外部処理を伴う関数 (SQLSTATE 42845) を入れることはできません。 この文節は、RANK および DENSE_RANK 関数 (SQLSTATE 42601) で必要になります。

ASC
sort-key-expression の値を昇順に使用します。

DESC
sort-key-expression の値を降順に使用します。

NULLS FIRST
ウィンドウ配列において、ソート順序は、すべての非 NULL 値の*前に* NULL 値が置かれます。

NULLS LAST
ウィンドウ配列において、ソート順序は、すべての非 NULL 値の*後に* NULL 値が置かれます。

ORDER OF *table-designator*
表指定子 で使用されているのと同じ順序付けを、副選択の結果表にも適用する必要があることを指定します。この文節を指定する副選択の FROM 文節内には、*表指定子* に一致する表参照がなければなりません (SQLSTATE 42703)。指定の *表指定子* に対応する副選択 (または全選択) には、データに依存する ORDER BY 文節が入っていなければなりません (SQLSTATE 428FI)。ネストされた副選択 (または全選択) 内の ORDER BY 文節の列はさらに外側の副選択

(または全選択) 内に収容されていて、しかもそれらの列は、ORDER OF 文節の代わりに指定された列である場合と同じ順序付けが適用されます。

window-aggregation-group-clause
行 R の集約グループは、 (R のパーティションの行の順序付け内の) R に関連して定義されている行のセットです。その文節は集約グループを指定します。この文節が指定されていない場合、デフォルトは、累積集約結果を提供する RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW と同じになります。

ROWS
集約グループがカウント行によって定義されることを示します。

RANGE
集約グループがソート・キーからのオフセットによって定義されることを示します。

group-start
集約グループの開始点を指定します。集約グループの終了は current row です。 group-start 文節の仕様は、 ″BETWEEN group-start AND CURRENT ROW″ 形式の group-between 文節と同じです。

group-between
ROWS または RANGE に基づいて、集約グループの開始および終了を指定します。

group-end
集約グループの終了点を指定します。集約グループの開始は current row です。 group-end 文節の仕様は、 ″BETWEEN CURRENT ROW AND group-end″ 形式の group-between 文節と同じです。

UNBOUNDED PRECEDING
current row の前のパーティション全体を組み込みます。これは、ROWS または RANGE のいずれかと一緒に指定できます。 window-order-clause 内の複数の sort-key-expressions と一緒に指定することもできます。

UNBOUNDED FOLLOWING
current row に続くパーティション全体を組み込みます。これは、ROWS または RANGE のいずれかと一緒に指定できます。 window-order-clause 内の複数の sort-key-expressions と一緒に指定することもできます。

CURRENT ROW
current row に基づいて、集約グループの開始および終了を指定します。 ROWS が指定された場合、current row が集約グループ境界です。 RANGE が指定された場合、集約グループ境界には、 current row と同じ値を *sort-key-expressions* として持つ行のセットが組み込まれます。 *group-bound1* で *value* FOLLOWING が指定されている場合、この文節を *group-bound2* で指定することはできません。

value **PRECEDING**
current row の前の行の範囲または行数のいずれかを指定します。 ROWS が指定された場合、*value* は行数を示す正の整数です。 RANGE が指定された場合、*value* のデータ型は、 window-order-clause の sort-key-expression のタイプと互換性がなければなりません。 sort-key-expression は 1 つのみ

で、 sort-key-expression のデータ型は減算を許可しなければなりません。 *group-bound1* が CURRENT ROW または *value* FOLLOWING の場合、この文節を *group-bound2* で指定することはできません。

value **FOLLOWING**
current row の後の行の範囲または行数のいずれかを指定します。 ROWS が指定された場合、*value* は行数を示す正の整数です。 RANGE が指定された場合、*value* のデータ型は、 window-order-clause の sort-key-expression のタイプと互換性がなければなりません。 sort-key-expression は 1 つのみで、 sort-key-expression のデータ型は加算を許可しなければなりません。

例:

- 給与合計 (給与 + ボーナス) が $30,000 を超えている従業員のランキングを、それぞれの給与合計に基づいて、姓の順に表示します。

```
SELECT EMPNO, LASTNAME, FIRSTNME, SALARY+BONUS AS TOTAL_SALARY,
    RANK() OVER (ORDER BY SALARY+BONUS DESC) AS RANK_SALARY
  FROM EMPLOYEE WHERE SALARY+BONUS > 30000
  ORDER BY LASTNAME
```

  結果をランキング順に並べる場合、ORDER BY LASTNAME を以下のように置き換えます。

```
ORDER BY RANK_SALARY
```

  または

```
ORDER BY RANK() OVER (ORDER BY SALARY+BONUS DESC)
```

- それぞれの給与合計の平均に基づいて部門をランク付けします。

```
SELECT WORKDEPT, AVG(SALARY+BONUS) AS AVG_TOTAL_SALARY,
    RANK() OVER (ORDER BY AVG(SALARY+BONUS) DESC) AS RANK_AVG_SAL
  FROM EMPLOYEE
  GROUP BY WORKDEPT
  ORDER BY RANK_AVG_SAL
```

- それぞれの学歴に基づいて部門内で従業員をランク付けします。部門内で同じランクの従業員が複数いた場合は、次のランキング値を増やさないようにします。

```
SELECT WORKDEPT, EMPNO, LASTNAME, FIRSTNME, EDLEVEL,
    DENSE_RANK() OVER
      (PARTITION BY WORKDEPT ORDER BY EDLEVEL DESC) AS RANK_EDLEVEL
  FROM EMPLOYEE
  ORDER BY WORKDEPT, LASTNAME
```

- 照会の結果に行番号を示します。

```
SELECT ROW_NUMBER() OVER (ORDER BY WORKDEPT, LASTNAME) AS NUMBER,
    LASTNAME, SALARY
  FROM EMPLOYEE
  ORDER BY WORKDEPT, LASTNAME
```

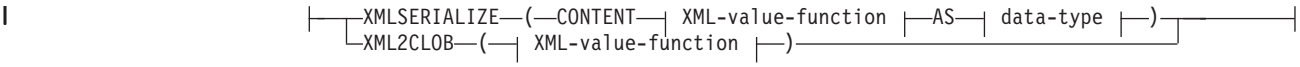
- 収入の多い上位 5 人をリストします。

```
SELECT EMPNO, LASTNAME, FIRSTNME, TOTAL_SALARY, RANK_SALARY
  FROM (SELECT EMPNO, LASTNAME, FIRSTNME, SALARY+BONUS AS TOTAL_SALARY,
    RANK() OVER (ORDER BY SALARY+BONUS DESC) AS RANK_SALARY
    FROM EMPLOYEE) AS RANKED_EMPLOYEE
  WHERE RANK_SALARY < 6
  ORDER BY RANK_SALARY
```

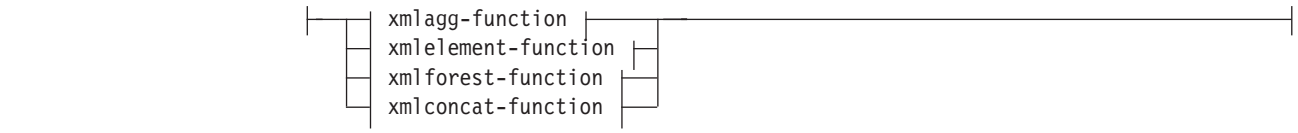
ランクを WHERE 文節で使うために、事前にそのランキングも含めた結果をまず計算するのに、ネストされた表の式が使われていることに注意してください。共通表式も使われています。

XML 関数

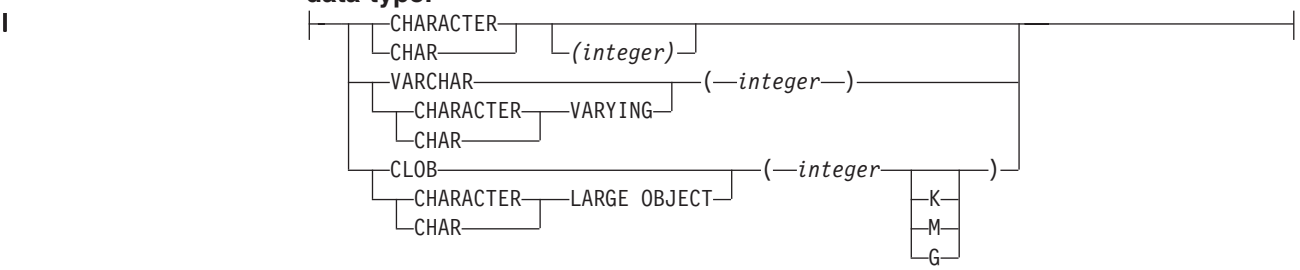
XML-function:



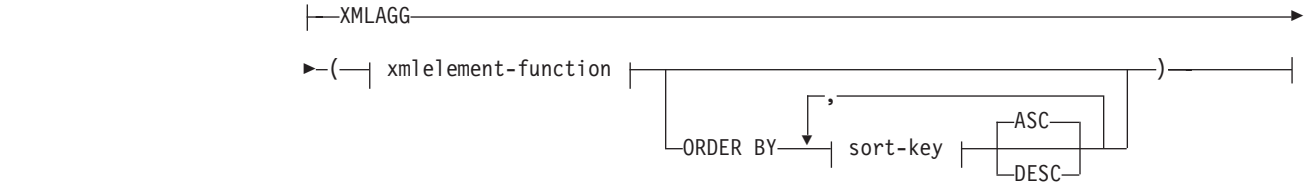
XML-value-function:



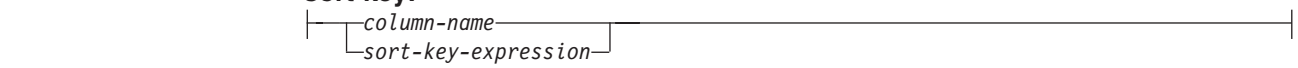
data-type:



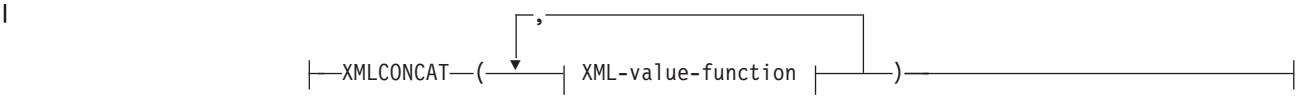
xmlagg-function:



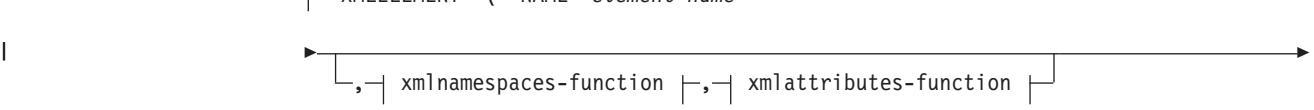
sort-key:



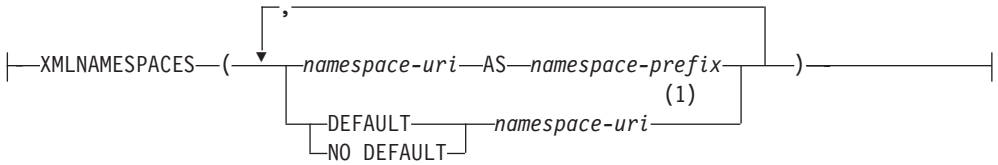
xmlconcat-function:



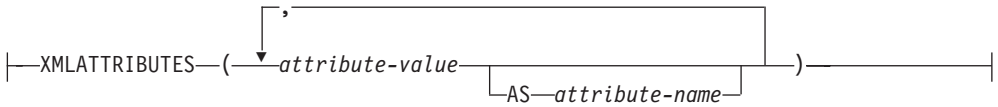
xmlelement-function:



xmlnamespaces-function:

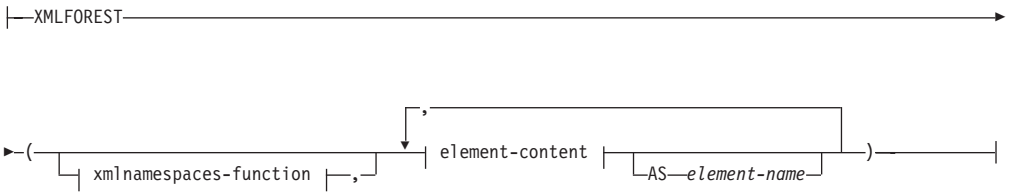


xmlattributes-function:



element-content:

xmlforest-function:



注:

1 DEFAULT または NO DEFAULT は、XMLNAMESPACES の引き数内で 1 度しか指定できません。

XMLSERIALIZE

関数の引き数の一部として指定されたデータ型の値として引き数を戻します。スキーマは SYSIBM です。引き数は、XML のデータ型の値を戻す式でなければなりません。結果は、ユーザーが指定したデータ型になります。
XML-value-function が NULL 値の場合、結果は NULL 値になります。

CONTENT

XML-value-function の値を、複数の最上位エレメントで構成できることを指定します。

AS *data-type*

サポートされているデータ型は CHAR、VARCHAR、および CLOB です。CHAR の長さを指定しなかった場合、長さのデフォルトは 1 になります。VARCHAR には長さの値を指定しなければなりません。 CLOB の長さを指定しなかった場合、長さのデフォルトは 1M になります。

XML2CLOB

CLOB 値で引き数を戻します。スキーマは SYSIBM です。引き数は、XML のデータ型の値を戻す式でなければなりません。結果は CLOB データ型になります。

XML2CLOB 関数は、旧リリースとの互換性を保つために備えられています。この代わりに、XMLSERIALIZE 関数を使用してください。

XMLAGG

一連の XML 値の連結を戻します。 スキーマは SYSIBM です。関数名を修飾名で指定することはできません。結果のデータ型は XML になり、その長さは 1 073 741 823 に設定されます。この XMLAGG 関数が空のセットに適用されると、結果は NULL 値になります。適用されない場合、結果はセット内の値の連結になります。

ORDER BY

集合内の処理対象の、同じグループ化集合に属する行の順序を指定します。ORDER BY 文節を省略した場合や、ORDER BY が列データの ORDER BY を特定できない場合、同一のグループ化集合内の行は任意の ORDER BY で並べられます。

sort-key

ソート・キーは、列名または sort-key-expression のどちらでもかまいません。ソート・キーが定数の場合、ソート・キーは出力列の位置を (通常の ORDER BY 文節におけるように) 参照しませんが、これは単なる定数でしかなく、ソート・キーではないことを意味することに注意してください。

XMLAGG 関数の使用に関する制約事項は次のとおりです。

- 列関数を直接入力に使用することはできません (SQLSTATE 42607)。
- XMLAGG を、OLAP 集約関数の列関数として使用することはできません (SQLSTATE 42601)。

XMLCONCAT

さまざまな数の XML 引き数の連結を戻します。スキーマは SYSIBM です。引き数は、データ型 XML の式でなければなりません。結果は、引き数と同じ内部 XML データ型をもちます。 XMLCONCAT 関数の入力引き数から戻される NULL 値は無視されます。 XMLCONCAT のすべての引き数が NULL であれば、結果は NULL 値になります。それ以外の場合、結果は XML 値の連結になります。

XMLELEMENT

引き数から XML エレメントを作成します。スキーマは SYSIBM です。関数名を修飾名で指定することはできません。この関数は、エレメント名、オプションのネーム・スペース宣言の集合、オプションの属性の集合、およびエレメントの内容を構成するゼロ個以上の引き数をとります。 エレメントの内容が NULL の場合、結果は空のエレメントになります。 結果のデータ型は XML になります。

NAME

このキーワードは、XML エレメントの名前の前に付きます。

element-name
XML エレメントの名前。 これは XML QName でなければなりません。有効名の詳細は、「W3C の XML ネーム・スペース仕様」を参照してください。

xmlnamespaces-function
XMLNAMESPACES 関数の結果である XML ネーム・スペース宣言。

xmlattributes-function
XMLATTRIBUTES 関数の結果である XML 属性。

element-content
生成されるエレメントの内容を、式によってか、または式リストによって指定します。式の結果のデータ型は、SMALLINT、INTEGER、BIGINT、DECIMAL、NUMERIC、REAL、DOUBLE、CHAR、VARCHAR、LONG VARCHAR、CLOB、GRAPHIC、VARGRAPHIC、LONG VARGRAPHIC、DBCLOB、DATE、TIME、TIMESTAMP、XML のうちのいずれかか、またはこれらのデータ型のうちのいずれかをソース・タイプとしてもつ特殊タイプでなければなりません。 FOR BIT DATA と定義されている文字ストリング・データは使用できません。式は任意の SQL 式でかまいませんが、スカラー全選択や副照会をその中に組み込むことはできません。

XMLATTRIBUTES
引き数から XML 属性を作成します。スキーマは SYSIBM です。関数名を修飾名で指定することはできません。結果は、引き数と同じ内部 XML データ型をもちます。

attribute-value
属性値は式です。式の結果のデータ型は、SMALLINT、INTEGER、BIGINT、DECIMAL、NUMERIC、REAL、DOUBLE、CHAR、VARCHAR、LONG VARCHAR、CLOB、GRAPHIC、VARGRAPHIC、LONG VARGRAPHIC、 DBCLOB、DATE、TIME、TIMESTAMP のうちのいずれかか、またはこれらのデータ型のうちのいずれかをソース・タイプとしてもつ特殊タイプでなければなりません。 FOR BIT DATA と定義されている文字ストリング・データは使用できません。式は任意の SQL 式でかまいませんが、スカラー全選択や副照会をその中に組み込むことはできません。式が単純な列参照でない場合、属性名を指定する必要があります。重複した属性名を使用することはできません (SQLSTATE 42713)。

attribute-name
属性名は SQL ID です。これは、 XML 修飾名の形式であるかまたは QName (SQLSTATE 42634) でなければなりません。有効名の詳細は、「W3C の XML ネーム・スペース仕様」を参照してください。 属性名を `xmlns` にしたり、その前に `xmlns:` を付けたりすることはできません。ネーム・スペースは、関数 XMLNAMESPACES を使って宣言します。

XMLFOREST
引き数から得られる XML エレメントのシーケンス (FOREST) を構成します。スキーマは SYSIBM です。結果は、引き数と同じ内部 XML データ型をもちます。 XMLFOREST 関数の入力引き数から戻される NULL 値は無視されます

(エレメントは生成されません)。 XMLFOREST のすべての引き数が NULL であれば、結果は NULL 値になります。それ以外の場合、結果は戻される XML エレメントのシーケンスになります。

xmlnamespaces-function
XMLNAMESPACES 関数の結果である XML ネーム・スペース宣言。

element-content
エレメント内容は式です。式の結果のデータ型は、 SMALLINT、 INTEGER、 BIGINT、 DECIMAL、 NUMERIC、 REAL、 DOUBLE、 CHAR、 VARCHAR、 LONG VARCHAR、 CLOB、 GRAPHIC、 VARGRAPHIC、 LONG VARGRAPHIC、 DBCLOB、 DATE、 TIME、 TIMESTAMP のうちのいずれかか、またはこれらのデータ型のうちのいずれかをソース・タイプとしてもつ特殊タイプでなければなりません。 FOR BIT DATA と定義されている文字ストリング・データは使用できません。式は任意の SQL 式でかまいませんが、スカラー全選択や副照会をその中に組み込むことはできません。式が単純な列参照でない場合、エレメント名を指定する必要があります。

element-name
エレメント名は SQL IDです。エレメント名は、 XML 修飾名の形式であるかまたは QName (SQLSTATE 42634) でなければなりません。有効名の詳細は、「W3C の XML ネーム・スペース仕様」を参照してください。

XMLNAMESPACES
引き数から XML ネーム・スペース宣言を作成します。スキーマは SYSIBM です。結果のデータ型は XML になります。ネーム・スペースの宣言は、XMLELEMENT 関数と XMLFOREST 関数で生成されたエレメントの有効範囲内にあります。事前定義のネーム・スペース接頭部である `xml` を、明示的に定義しなくても使用することができます。ただし `xml` 接頭部を再定義することはできません。

namespace-uri
ネーム・スペースの URI は、文字ストリング定数です。この文字ストリング定数は空であってはなりません。

namespace-prefix
ネーム・スペースの接頭部は SQL ID です。これは、XML NCName のフォームでなければなりません (SQLSTATE 42635)。有効名の詳細は、「W3C の XML ネーム・スペース仕様」を参照してください。接頭部を `xml` または `xmlns` にすることはできません。接頭部は、ネーム・スペース宣言リストの中でユニークでなければなりません。

DEFAULT *namespace-uri*
ネーム・スペースに NO DEFAULT を指定している *xmlnamespaces-functions* を除いて、この XML エレメントに対してと、この XML エレメントに入っているすべての XML エレメントに対して使用するデフォルトのネーム・スペースを指定します。

NO DEFAULT
ネーム・スペースに DEFAULT を指定している *xmlnamespaces-functions* を

除いて、この XML エレメントに対してと、この XML エレメントに入っているすべての XML エレメントに対して、デフォルトのネーム・スペースを使わないことを指定します。

例:

- XMLELEMENT 関数から戻された式から CLOB を作成します。以下に照会の例を示します。

```
SELECT e.empno, XMLSERIALIZE(CONTENT XMLELEMENT(NAME "Emp", e.firstnme || ' ' ||
    e.lastname) AS CLOB)
  AS "Result" FROM employee e
  WHERE e.edlevel = 12
```

上記は、次のような結果を生じます。

```
EMPNO      Result
000290     <Emp>JOHN PARKER</Emp>
000310     <Emp>MAUDE SETRIGHT</Emp>
```

- 社員の姓別にソートした社員リストを使って部門エレメント (部門別の) を作成します。

```
SELECT XMLSERIALIZE(CONTENT XMLELEMENT(NAME "Department",
                          XMLATTRIBUTES(e.workdept AS "name"),
                          XMLAGG(XMLELEMENT(NAME "emp", e.lastname)
                            ORDER BY e.lastname
                              )
                         ) AS CLOB ) AS "dept_list"
  FROM employee e
  WHERE e.workdept IN ('C01','E21')
  GROUP BY workdept
```

この照会は、次のような出力を生じます。結果内には、実際にはスペースや改行文字は生じないことに注意してください。以下の出力は、分かりやすいようにフォーマットされています。

```
dept_list
<Department name = "C01">
    <emp>KWAN</emp>
    <emp>NICHOLLS</emp>
    <emp>QUINTANA</emp>
</Department>
<Department name = "E21">
    <emp>GOUNOT</emp>
    <emp>LEE</emp>
    <emp>MEHTA</emp>
    <emp>SPENSER</emp>
</Department>
```

- 部門 A00 の配下にある各部門ごとに、MGRNO という ID 属性をもつ Mgr という名前の空の XML エレメントを作成します。以下に照会の例を示します。

```
SELECT d.deptno, XMLSERIALIZE(CONTENT XMLELEMENT(NAME "Mgr",
  XMLATTRIBUTES(d.mgrno AS "ID")) AS CLOB)
  AS "Result" FROM department d
  WHERE d.admrdept = 'A00'
```

上記は、次のような結果を生じます。

```
DEPTNO     Result
A00        <Mgr ID="000010"/>
B01        <Mgr ID="000020"/>
C01        <Mgr ID="000030"/>
D01        <Mgr/>
E01        <Mgr ID="000050">
```

- 各社員ごとに Emp という名前の XML エレメントを作成します。ただし、社員の氏名と社員の雇用日付のエレメントはネストされています。以下に照会の例を示します。

```
SELECT e.empno, XMLSERIALIZE
 (CONTENT XMLELEMENT(NAME "Emp",
  XMLELEMENT(NAME "name", e.firstnme || ' ' || e.lastname),
  XMLELEMENT(NAME "hiredate", e.hiredate)) AS CLOB)
  AS "Result" FROM employee e
  WHERE e.edlevel = 12
```

上記は、以下の結果を生じます (以下は分かりやすいようにフォーマットされています。出力の XML フラグメントには余計な空白文字はありません)。

```
EMPNO      Result
000290     <Emp>
           <name>JOHN PARKER</name>
           <hiredate>1980-05-30</hiredate>
           </Emp>
000310     <Emp>
           <name>MAUDE SETRIGHT</name>
           <hiredate>1964-09-12</hiredate>
           </Emp>
```

- XMLATTRIBUTES 関数を、XMLSERIALIZE 関数と XMLELEMENT 関数と一緒に使用して、XML 属性を作成します。以下に照会の例を示します。

```
SELECT XMLSERIALIZE(CONTENT XMLELEMENT(NAME "Emp_Exempt",
  XMLATTRIBUTES(e.firstnme, e.lastname AS "name:last", e.midinit)) AS CLOB)
    AS "Result"
  FROM employee e
  WHERE e.lastname='GEYER'
```

上記は、次のような結果を生じます。

```
<Emp_Exempt
   FIRSTNME="JOHN"
   name_last="GEYER"
   MIDINIT="B">
</Emp_Exempt>
```

- 有資格のそれぞれの社員ごとに Emp エレメントを生成します。それには、XMLFOREST 照会の引き数から生成される一連のサブエレメントが入ります。以下に照会の例を示します。

```
SELECT e.empno, XMLSERIALIZE(CONTENT XMLELEMENT(NAME "Emp",
    XMLFOREST(e.firstnme || ' ' || e.lastname AS "Name" , e.hiredate)) AS CLOB)
  AS "Result" FROM employee e
  WHERE e.edlevel = 12
```

上記は、以下の結果を生じます (以下は分かりやすいようにフォーマットされています。出力の XML には余計な空白文字はありません)。

```
EMPNO      Result
000290     <Emp>
             <Name>JOHN PARKER</Name>
             <HIDREDATE>1980-05-30</HIREDATE>
           </Emp>
000310     <Emp>
             <Name>MAUDE SETRIGHT</Name>
             <HIREDATE>1964-09-12</HIREDATE>
           </Emp>
```

- 資格のあるそれぞれの社員の姓名を生成します。以下に照会の例を示します。

```
SELECT e.empno, XMLSERIALIZE(CONTENT XMLCONCAT(XMLELEMENT(NAME "firstname",
      e.firstnme),
    XMLELEMENT(NAME "lastname", e.lastname)) AS CLOB)
  AS "Result" FROM employee e
  WHERE e.edlevel = 12
```

上記は、次のような出力を生じます。

```
EMPNO      Result
000290     <firstname>JOHN</firstname><lastname>PARKER</lastname>
000310     <firstname>MAUDE</firstname><lastname>SETRIGHT</lastname>
```

- adm:employee という名前の XML エレメントと、 XML 属性 adm:department を生成します。このどちらにも、 adm を接頭部としてもつネーム・スペースが関連付けられます。

```
SELECT empno, XMLSERIALIZE(CONTENT XMLELEMENT(NAME "adm:employee",
  XMLNAMESPACES('http://www.adm.com' AS "adm"),
  XMLATTRIBUTES(workdept AS "adm:department"), lastname) AS CLOB) AS "Result"
  FROM employee
  WHERE job = 'ANALYST'
```

上記の照会は、以下の出力 (ここでは分かりやすいようにフォーマットされています。出力の XML フラグメントには余計な空白文字はなく、通常は 1 行で示されます) におけるようなテキスト表現の XML 値を生じます。

```
EMPNO  Result
000130 <adm:employee xmlns:adm="http://www.adm.com"
                     adm:department="C01">QUINTANA
       </adm:employee>
000140 <adm:employee xmlns:adm="http://www.adm.com"
                     adm:department="C01">NICHOLLS
       </adm:employee>
```

- XMLFOREST の引き数から作成された一連のエレメントを生成します。また、最初のエレメントに関連付けられたデフォルトのネーム・スペースと、 2 番目のエレメントに関連付けられた d を接頭部としてもつネーム・スペースの宣言も行います。

```
SELECT empno, XMLSERIALIZE(CONTENT XMLFOREST
  (XMLNAMESPACES(DEFAULT 'http://hr.org',
    'http://fed.gov' AS "d"),
    lastname, job AS "d:job") AS CLOB) AS "Result"
  FROM employee
  WHERE edlevel = 12
```

上記の照会は、以下の出力 (ここでは分かりやすいようにフォーマットされています。出力の XML フラグメントには余計な空白文字はなく、通常は 1 行で示されます) におけるようなテキスト表現の XML 値を生成します。

```
EMPNO  Result
000290 <LASTNAME xmlns="http://hr.org" xmlns:d="http://fed.gov">PARKER
       </LASTNAME>
       <d:job xmlns="http://hr.org" xmlns:d="http://fed.gov">OPERATOR
       </d:job>
000310 <LASTNAME xmlns="http://hr.org" xmlns:d="http://fed.gov">SETRIGHT
       </LASTNAME>
       <d:job xmlns="http://hr.org" xmlns:d="http://fed.gov">OPERATOR
       </d:job>
```

- デフォルトのネーム・スペースに関連付けられた employee という名前の XML エレメントと、デフォルトのネーム・スペースを使用しない department という名前のサブエレメントを作成します。

```
SELECT emp.empno, XMLSERIALIZE(CONTENT XMLELEMENT(NAME "employee",
  XMLNAMESPACES(DEFAULT 'http://hr.org'), emp.lastname,
  XMLELEMENT(NAME "department",
    XMLNAMESPACES(NO DEFAULT),
      emp.workdept)) AS CLOB) AS "Result"
  FROM employee emp
  WHERE emp.edlevel = 12
```

上記の照会は、以下の出力 (ここでは分かりやすいようにフォーマットされています。出力の XML フラグメントには余計な空白文字はなく、通常は 1 行で示されます) におけるようにシリアライズ化された表現形式の XML 値を生成します。

```
EMPNO  Result
000290 <employee xmlns="http://hr.org">PARKER
              <department xmlns="">E11
              </department>
       </employee>
000310 <employee xmlns="http://hr.org">SETRIGHT
              <department xmlns="">E11
              </department>
       </employee>
```

- デフォルトのネーム・スペースに関連付けられた employee という名前の XML エレメントと、デフォルトのネーム・スペースを使用するサブエレメント department をもったデフォルトのネーム・スペースを使用しない job という名前のサブエレメントを作成します。

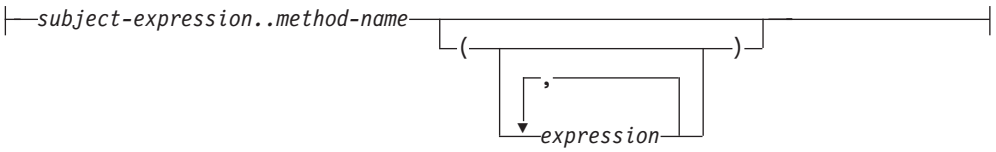
```
SELECT emp.empno, XMLSERIALIZE(CONTENT XMLELEMENT(NAME "employee",
  XMLNAMESPACES(DEFAULT 'http://hr.org'), emp.lastname,
  XMLELEMENT(NAME "job",
   XMLNAMESPACES(NO DEFAULT), emp.job,
   XMLELEMENT(NAME "department",
   XMLNAMESPACES(DEFAULT 'http://adm.org'), emp.workdept))) AS CLOB) AS "Result"
  FROM employee emp
  WHERE emp.edlevel = 12
```

上記の照会は、以下の出力 (ここでは分かりやすいようにフォーマットされています。出力の XML フラグメントには余計な空白文字はなく、通常は 1 行で示されます) におけるようにシリアライズ化された表現形式の XML 値を生成します。

```
EMPNO   Result
000290 <employee xmlns="http://hr.org">PARKER
           <job xmlns="">OPERATOR
              <department xmlns="http://adm.org">E11
              </department>
           </job>
       </employee>
000310 <employee xmlns="http://hr.org">SETRIGHT
           <job xmlns="">OPERATOR
              <department xmlns="http://adm.org">E11
              </department>
           </job>
       </employee>
```

メソッドの呼び出し

method-invocation:



システム生成による監視および変更メソッドの両方、さらにユーザー定義メソッドも、二重ドット演算子を使って呼び出されます。

subject-expression
ユーザー定義構造化タイプである静的結果タイプを持つ式。

method-name
修飾なしのメソッド名。 *subject-expression* の静的タイプまたはそのスーパータイプのいずれかに、指定した名前を持つメソッドが入っている必要があります。

(*expression***,...)**
括弧内に *method-name* の引き数を指定します。引き数がないことを示すときには、括弧内を空にしておくことができます。特定のメソッドを解決するときに、*subject-expression* の静的タイプに基づき、 *method-name* と、指定した引き数の式のデータ型を使用します。

メソッド呼び出しに使う二重ドット演算子は、優先順位が高い順に左から右へ列挙される挿入演算子です。たとえば、以下の 2 つの式は同じことを意味します。

```
a..b..c + x..y..z
```

および

```
((a..b)..c) + ((x..y)..z)
```

メソッドにサブジェクト以外のパラメーターがない場合、括弧があってもなくても呼び出すことができます。たとえば、以下の 2 つの式は同じことを意味します。

```
point1..x
point1..x()
```

メソッド呼び出しの NULL サブジェクトは、次のように扱われます。

- システム生成の変更メソッドが NULL サブジェクトで呼び出される場合、エラーになります (SQLSTATE 2202D)。
- システム生成の変更メソッド以外のメソッドが NULL サブジェクトで呼び出される場合、そのメソッドは実行されず、結果は NULL になります。この規則は、SELF AS RESULT を指定したユーザー定義メソッドにも当てはまります。

データベース・オブジェクト (パッケージ、ビュー、またはトリガーなど) を作成する場合、それぞれのメソッド呼び出しのための最適な方法を見つけられます。

注: 定義した WITH FUNCTION ACCESS タイプのメソッドは、通常の関数表記を使用して呼び出すこともできます。関数解決では、候補となる関数として、すべての関数だけでなく、関数アクセスのあるメソッドも考慮します。ただし、メソッド呼び出しを使用して関数を呼び出すことはできません。メソッド解決

では、候補となるメソッドとして、すべてのメソッドを考慮しますが、関数については考慮しません。適切な関数またはメソッドの解決に失敗すると、エラーになります (SQLSTATE 42884)。

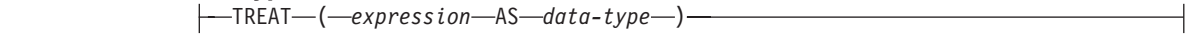
例:

- 二重ドット演算子を使用して、AREA というメソッドを呼び出します。構造化タイプ CIRCLE の列 CIRCLE_COL をもった RINGS という表が存在するとします。また、CIRCLE タイプのために、メソッド AREA が、 AREA() RETURNS DOUBLE としてあらかじめ定義されているとします。

```
SELECT CIRCLE_COL..AREA() FROM RINGS
```

サブタイプの扱い

subtype-treatment:



subtype-treatment は、構造化タイプの式を、そのサブタイプのいずれかへキャストするときに使用します。 *expression* の静的タイプは、ユーザー定義構造化タイプでなければなりません。このタイプは、*data-type* と同じタイプであるか、またはそのスーパータイプでなければなりません。 *data-type* のタイプ名が修飾されていない場合は、 SQL パスを使用してタイプ参照を解決します。 subtype-treatment の結果の静的タイプは *data-type* であり、 subtype-treatment の値は expression の値になります。ランタイムに、expression の動的タイプが *data-type* ではないか、 *data-type* のサブタイプでない場合、エラーが戻されます (SQLSTATE 0D000)。

例:

- 列 CIRCLE_COL のすべての列オブジェクト・インスタンスに、動的タイプ COLOREDCIRCLE があることを、アプリケーション側が認識している場合、次の照会を使って、そのようなオブジェクト上でメソッド RGB を呼び出します。構造化タイプ CIRCLE の列 CIRCLE_COL をもった RINGS という表が存在するとします。また、COLOREDCIRCLE は CIRCLE のサブタイプであり、COLOREDCIRCLE のために、メソッド RGB が RGB() RETURNS DOUBLE としてあらかじめ定義されているとします。

```
SELECT TREAT (CIRCLE_COL AS COLOREDCIRCLE)..RGB()
  FROM RINGS
```

ランタイムに、動的タイプ CIRCLE のインスタンスが存在する場合、エラーになります (SQLSTATE 0D000)。このエラーは、次に示すように、CASE 式の中で TYPE 述部を使うことで避けることができます。

```
SELECT (CASE
  WHEN CIRCLE_COL IS OF (COLOREDCIRCLE)
    THEN TREAT (CIRCLE_COL AS COLOREDCIRCLE)..RGB()
    ELSE NULL
  END)
  FROM RINGS
```

シーケンス参照

sequence-reference:

nextval-expression:

prevval-expression:

NEXT VALUE FOR *sequence-name*

NEXT VALUE 式は、 *sequence-name* で指定されたシーケンスの次の値を生成して返します。

PREVIOUS VALUE FOR *sequence-name*

PREVIOUS VALUE 式は、現行アプリケーション・プロセス内の直前のステートメントに指定されたシーケンスについて最後に生成された値を返します。この値は、シーケンスの名前が指定されている PREVIOUS VALUE 式を使用して、繰り返し参照することができます。単一ステートメント内に同じシーケンスを指定している PREVIOUS VALUE 式のインスタンスが複数存在する可能性があり、それらはすべて同じ値を返します。パーティション・データベース環境では、最も新しく生成された値が PREVIOUS VALUE 式によって返されない場合があります。

同じシーケンス名が指定されている NEXT VALUE 式がすでに現行アプリケーション・プロセスで、現在または前のトランザクションで参照されている場合のみ、 PREVIOUS VALUE 式を使用できます (SQLSTATE 51035)。

注意:

- ***互換性***
  - 以前のバージョンの DB2 との互換性:
    - NEXTVAL と PREVVAL は、NEXT VALUE と PREVIOUS VALUE の代わりに指定できます。
- NEXT VALUE 式がシーケンスの名前を指定していれば、そのシーケンスの新しい値が生成されます。ただし、照会の中に同じシーケンス名を指定している NEXT VALUE 式のインスタンスが複数ある場合、シーケンスのカウンターは結果の行ごとに 1 つずつ増えていき、 NEXT VALUE のすべてのインスタンスが結果の行に同じ値を戻します。
- 以下に示すように、同じシーケンス番号は、先頭の行の NEXT VALUE 式 (これはシーケンス値を生成します) およびその他の行の PREVIOUS VALUE 式 (PREVIOUS VALUE のインスタンスは現在のセッションで最後に生成されたシーケンス値を参照します) を使用してシーケンス番号を参照することによって、 2 つの異なる表内のユニーク・キー値として使用することができます。

```
INSERT INTO order(orderno, cutno)
  VALUES (NEXT VALUE FOR order_seq, 123456);

INSERT INTO line_item (orderno, partno, quantity)
  VALUES (PREVIOUS VALUE FOR order_seq, 987654, 1);
```

- NEXT VALUE 式と PREVIOUS VALUE 式は、以下の位置に指定できます。
  - select-statement または SELECT INTO ステートメント (ステートメントに DISTINCT キーワード、 GROUP BY 文節、ORDER BY 文節、UNION キーワード、INTERSECT キーワード、または EXCEPT キーワードが入っていなければ、select-clause 内)
  - INSERT ステートメント (VALUES 文節内)
  - INSERT ステートメント (全選択の select-clause 内)
  - UPDATE ステートメント (SET 文節内 (検索条件付き、または位置指定 UPDATE ステートメントのいずれか)、ただし NEXT VALUE は SET 文節にある式の全選択の select-clause 内に指定できない)
  - SET 変数ステートメント (式の全選択の select-clause 内を除きます。トリガー内に NEXT VALUE 式を指定することができますが、PREVIOUS VALUE 式は指定できません)。
  - VALUES INTO ステートメント (式の全選択の select-clause 内)
  - CREATE PROCEDURE ステートメント (SQL プロシージャーのルーチン本体内)
  - トリガー・アクション内の CREATE TRIGGER ステートメント (NEXT VALUE 式は指定できるが、 PREVIOUS VALUE 式は指定できない)
- NEXT VALUE 式と PREVIOUS VALUE 式は、以下の位置には指定できません。
  - 完全外部結合の結合条件
  - CREATE TABLE または ALTER TABLE ステートメント内の列の DEFAULT 値
  - CREATE TABLE または ALTER TABLE ステートメント内の生成された列定義
  - CREATE TABLE または ALTER TABLE ステートメント内のサマリー表定義
  - CHECK 制約の条件
  - CREATE TRIGGER ステートメント (NEXT VALUE 式は指定できるが、PREVIOUS VALUE 式は指定できない)
  - CREATE VIEW ステートメント
  - CREATE METHOD ステートメント
  - CREATE FUNCTION ステートメント
- また、以下の位置に NEXT VALUE 式を指定することはできません (SQLSTATE 428F9)。
  - CASE 式
  - 総計関数のパラメーター・リスト
  - それ以前に明示的に許可されていない場合、コンテキスト内の副照会
  - 外部 SELECT に DISTINCT 演算子を備えた SELECT ステートメント
  - 結合の結合条件

- 外部 SELECT に GROUP BY 文節を備えた SELECT ステートメント
- 外部 SELECT が UNION、INTERSECT、または EXCEPT セット演算子を使用して他の SELECT ステートメントと組み合わされている場合の SELECT ステートメント
- ネストされた表の式
- 表関数のパラメーター・リスト
- 最外部の SELECT ステートメントか、DELETE または UPDATE ステートメントの WHERE 文節
- 最外部の SELECT ステートメントの ORDER BY 文節
- UPDATE ステートメントの SET 文節にある式の全選択の select-clause
- SQL ルーチンにおける IF、 WHILE、 DO...UNTIL、または CASE ステートメント

- シーケンスについて値が生成されると、その値を再使用できなくなるため、次に値が要求されたときに新しい値が生成されます。 NEXT VALUE 式が組み込まれているステートメントが失敗した場合やロールバックされた場合でも、これが当てはまります。

  列の VALUES リストにある NEXT VALUE 式が INSERT ステートメントに組み込まれており、 INSERT 実行中のある時点でエラー (次のシーケンス値を生成しているときの問題、あるいは別の列の値に問題があると考えられる) が起こった場合、挿入は失敗し (SQLSTATE 23505)、シーケンスについて生成した値は再使用できないものと見なされます。場合によっては、同じ INSERT ステートメントを再発行することによって、正しく動作します。

  たとえば、NEXT VALUE が使用されていた列のユニーク索引が存在する結果としてエラーが起こり、すでに生成されているシーケンス値がその索引に存在するとします。シーケンスについて生成される次の値は、索引には存在しない値になることが考えられるため、後続の INSERT が正しく動作します。
- シーケンスの値の生成において、そのシーケンスが最大値 (または降順シーケンスの最小値) に達し、循環が許可されていない場合、エラーが起こります (SQLSTATE 23522)。この場合、ユーザーはシーケンスを ALTER して許容値の範囲を拡張、またはシーケンスの循環を可能にでき、あるいは値の範囲がより大きな、異なるデータ型を持つ新しいシーケンスを DROP および CREATE することができます。

  たとえば、シーケンスがデータ型 SMALLINT で定義されていて、その結果、そのシーケンスが割り当て可能な値を使い果たしてしまうことがあります。シーケンスを新しい定義で DROP および再作成して、そのシーケンスを INTEGER として再定義しなければならない場合があります。
- カーソルの SELECT ステートメント内の NEXT VALUE に対する参照は、結果表の行について生成される値を参照します。データベースから取り出される行ごとに NEXT VALUE 式のシーケンス値が生成されます。クライアントでブロッキングを行うと、サーバーで FETCH ステートメントの処理の前に値が生成されることがあります。この状況は、結果表の行がブロッキングされている場合に生じることがあります。クライアント・アプリケーションが、データベースでマテリアライズされている行をすべて明示的に FETCH しないと、 (マテリアライズされている行のうち戻されなかったものの) シーケンス値が生成されません。

- カーソルの SELECT ステートメント内の PREVIOUS VALUE に対する参照は、そのカーソルをオープンする前に、生成された指定シーケンスの値を参照します。しかしながら、カーソルをクローズすると、後続するステートメント内の、PREVIOUS VALUE によって戻される指定シーケンスの値に影響が生じることがあります。このことは、カーソルを再オープンした同じステートメントの場合でも生じることがあります。カーソルの SELECT ステートメントに入っている NEXT VALUE に対する参照中のシーケンス名が同じである場合はこのようになります。

例:

″order″ という表があり、 ″order_seq″ という以下のようなシーケンスが作成されると想定します。

```
CREATE SEQUENCE order_seq
  START WITH 1
  INCREMENT BY 1
  NO MAXVALUE
  NO CYCLE
  CACHE 24
```

以下は、NEXT VALUE 式で ″order_seq″ シーケンス番号を生成する方法例を示しています。

```
INSERT INTO order(orderno, custno)
  VALUES (NEXT VALUE FOR order_seq, 123456);
```

または

```
UPDATE order
  SET orderno = NEXT VALUE FOR order_seq
  WHERE custno = 123456;
```

または

```
VALUES NEXT VALUE FOR order_seq INTO :hv_seq;
```

関連資料:

- 64 ページの『ID』
- 240 ページの『TYPE 述部』
- 305 ページの『CHAR』
- 380 ページの『INTEGER』
- 511 ページの『全選択』
- 「*SQL リファレンス 第 2 巻*」の『CREATE TABLE ステートメント』
- 175 ページの『メソッド』
- 「*SQL リファレンス 第 2 巻*」の『CREATE FUNCTION (SQL スカラー、表、または行) ステートメント』
- 107 ページの『データ型間のキャスト』
- 110 ページの『割り当てと比較』
- 126 ページの『結果データ型の規則』
- 131 ページの『ストリング変換の規則』

述部

述部

述部 とは、特定の行またはグループに対して「真」、「偽」、または「不明」の条件を指定するものです。

以下の規則は、すべてのタイプの述部に適用されます。

- 述部の中で指定される値は、すべて互換でなければなりません。
- 基本、比較、IN、または BETWEEN 述部の式の結果が、長さ属性が 4 000 を超える文字ストリング、長さ属性が 2 000 を超える GRAPHIC ストリング、任意のサイズの LOB ストリングになってはなりません。
- ホスト変数の値は、NULL 値にすることができます (つまり、変数が負の標識変数を持つことがあります)。
- LIKE を除き、2 つ以上のオペランドを伴う述部のオペランドのコード・ページ変換は、ストリング変換の規則に従って行われます。
- DATALINK 値の使用は、NULL 述部に限定されています。
- 構造化タイプ値の使用は、NULL 述部と TYPE 述部に限定されています。
- Unicode データベースでは、文字または GRAPHIC ストリングを受け入れるすべての述部は、変換をサポートされている任意のストリング・タイプを受け入れます。

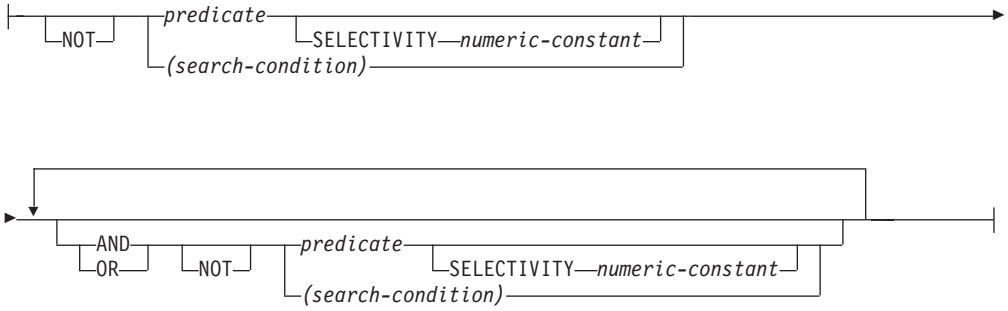
全選択は、SELECT ステートメントの 1 つの形式で、述部で使用されるとき、*副照会*とも呼ばれます。

関連資料:

- 511 ページの『全選択』
- 131 ページの『ストリング変換の規則』

検索条件

search-condition:



search-condition (検索条件) は、特定の行について「真」、「偽」、または「不明」となる条件を指定します。

検索条件の結果は、指定した各述部の結果に、指定した*論理演算子* (AND、OR、NOT) を適用することによって求められます。論理演算子の指定がない場合、検索条件の結果は指定された述部の結果になります。

AND と OR は、表 14 で定義されています。表中の P と Q は任意の述部です。

表 14. AND と OR の真理値表

P	Q	P AND Q	P OR Q
真	真	真	真
真	偽	偽	真
真	不明	不明	真
偽	真	偽	真
偽	偽	偽	偽
偽	不明	偽	不明
不明	真	不明	真
不明	偽	偽	不明
不明	不明	不明	不明

NOT(true) は偽、NOT(false) は真、NOT(unknown) は不明です。

括弧の中の検索条件が最初に評価されます。評価の順序を括弧によって指定していない場合、 NOT が AND の前に適用され、AND が OR の前に適用されます。同じ優先順位の演算子が評価される順序は、検索条件の最適化を図るために定義されていません。

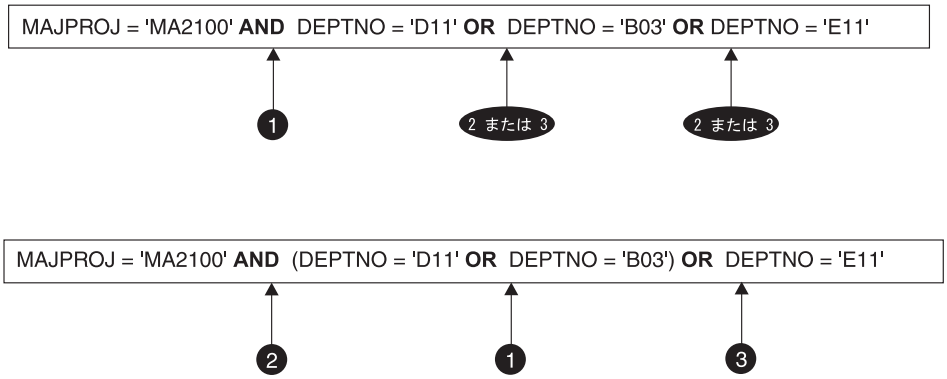


図 12. 検索条件の評価順序

SELECTIVITY *値*

SELECTIVITY 文節は、述部に指定する選択の予想パーセントを、 DB2 に知らせるときに使用します。 SELECTIVITY を指定できるのは、述部がユーザー定義述部である場合だけです。

ユーザー定義述部とは、述部が指定されているコンテキストの中で、ユーザー定義関数呼び出しで構成される述部のことです。これは、CREATE FUNCTION の PREDICATES 文節で指定した述部と一致します。たとえば、PREDICATES WHEN=1... で関数 foo が定義される場合、 SELECTIVITY を次のように使うことができます。

```
SELECT *
  FROM STORES
  WHERE foo(parm,parm) = 1 SELECTIVITY 0.004
```

この SELECTIVITY の値は、 0 から 1 の範囲の数値リテラル値でなければなりません (SQLSTATE 42615)。 SELECTIVITY を指定しない場合、デフォルトは 0.01 になります (つまり、ユーザー定義述部は、表内にあるすべての行の 1% を除いて、すべての行をフィルターで除外することになります)。 SYSSTAT.ROUTINES ビュー内の SELECTIVITY 列を更新すれば、どの関数の SELECTIVITY デフォルトでも変更することができます。ユーザー定義述部以外の述部に SELECTIVITY 文節を指定すると、エラーが戻されます (SQLSTATE 428E5)。

ユーザー定義関数 (UDF) はユーザー定義述部として使うことができるので、以下の場合、索引を利用するときにも使える可能性があります。

- CREATE FUNCTION ステートメントに述部の指定がある場合
- WHERE 文節で UDF が呼び出されていて、述部を指定したときの指定方法で (文法的に) 比較される場合
- 「否定」(NOT 演算子) がない場合

例:

次の照会では、WHERE 文節に within UDF が指定されていて、 3 つの条件がすべて満たされているので、ユーザー定義述部であると見なされます。

```
SELECT *
     FROM customers
     WHERE within(location, :sanJose) = 1 SELECTIVITY  0.2
```

ただし、次の照会に within を指定しても、「否定」が入っているため、索引を利用できません。これは、ユーザー定義述部とは見なされません。

```
SELECT *
     FROM customers
     WHERE NOT(within(location, :sanJose) = 1) SELECTIVITY  0.3
```

次の例では、相互が特定の距離内にいる顧客と店を識別します。特定の店から別の店の距離は、顧客が居住している都市の半径の範囲に基づいて計算されます。

```
SELECT *
     FROM customers, stores
     WHERE distance(customers.loc, stores.loc) <
        CityRadius(stores.loc) SELECTIVITY 0.02
```

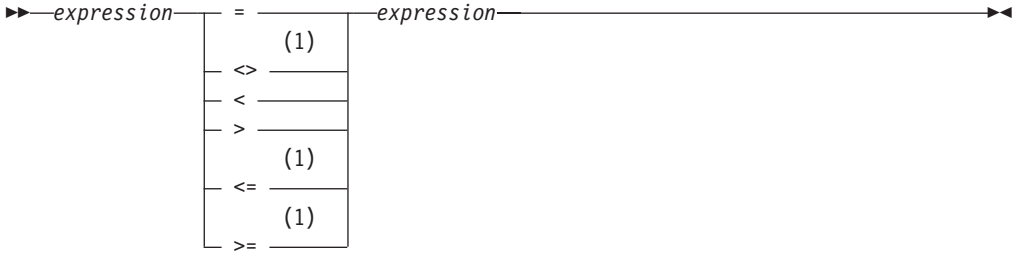
上記の照会では、WHERE 文節の述部は、ユーザー定義述部であると見なされます。 CityRadius による結果は、範囲を生成する関数に対する検索引き数として使われます。

しかし、CityRadius によって生成される結果は、範囲を生成する関数として使われるため、上記のユーザー定義述部では、 stores.loc 列で定義した索引の拡張機能を十分に利用することができません。したがって、UDF は customers.loc 列で定義した索引のみを利用します。

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE FUNCTION (外部スカラー) ステートメント』

基本述部



注:

1 基本述部および比較述部では、 ^=、^<、^>、!=、!<、および !> の形式の比較演算子もサポートされています。コード・ページ 437、819、および 850 では、 ¬=、¬<、および ¬> の形式もサポートされています。

このような製品固有の比較演算子の形式は、このような演算子を使用する既存の SQL をサポートすることのみを目的としており、新たに SQL ステートメントを書く場合には使用しないようお勧めします。

基本述部 は 2 つの値を比較します。

一方のオペランドの値が NULL 値の場合、述部の結果は不明です。それ以外の場合の結果は、真または偽のいずれかになります。

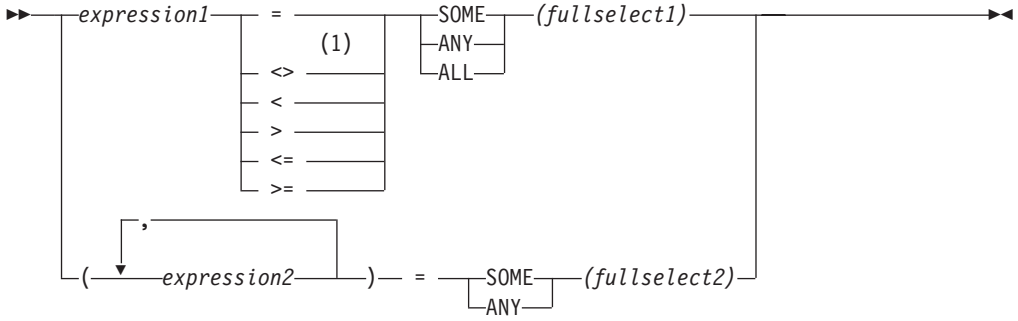
値 *x* および *y* について、次のような関係が成り立ちます。

述部	次の場合にのみ「真」になります。
x **=** *y*	*x* は *y* に等しい
x **<>** *y*	*x* は *y* に等しくない
x **<** *y*	*x* は *y* より小さい
x **>** *y*	*x* は *y* より大きい
x **>=** *y*	*x* は *y* より大きいか等しい
x **<=** *y*	*x* は *y* より小さいか等しい

例:

```
EMPNO='528671'
SALARY < 20000
PRSTAFF <> :VAR1
SALARY > (SELECT AVG(SALARY) FROM EMPLOYEE)
```

比較述部



注:

1 基本述部および比較述部では、 ^=、^<、^>、!=、!<、および !> の形式の比較演算子もサポートされています。コード・ページ 437、819、および 850 では、 ¬=、¬<、および ¬> の形式もサポートされています。

このような製品固有の比較演算子の形式は、このような演算子を使用する既存の SQL をサポートすることのみを目的としており、新たに SQL ステートメントを書く場合には使用しないようお勧めします。

比較述部 は、 1 つの値もしくは複数の値と、値の集合との間で比較を行います。

全選択は、述部演算子の左側に指定されている式の数と同じ数の列を識別しなければなりません (SQLSTATE 428C4)。全選択は、任意の行数を戻すことができます。

ALL を指定した場合、

- 全選択が値をまったく戻さない場合、または指定した関係が、全選択によって戻される値のすべてに対して「真」である場合、述部の結果は「真」となります。
- 指定した関係が、全選択によって戻される値の少なくとも 1 つに対して「偽」である場合、述部の結果は「偽」となります。
- 指定した関係が、全選択によって戻されるどの値に対しても「偽」でなく、少なくとも 1 つの比較が NULL 値のために「不明」である場合、述部の結果は「不明」となります。

SOME または ANY を指定した場合、

- 指定した関係が、全選択によって戻される少なくとも 1 つの行の各値に対して「真」である場合、述部の結果は「真」になります。
- 全選択が行を戻さない場合、または指定した関係が、全選択によって戻されるすべての行の少なくとも 1 つの値に対して「偽」である場合、述部の結果は「偽」になります。
- 指定した関係がどの行に対しても「真」でなく、少なくとも 1 つの比較が NULL 値のために「不明」である場合、述部の結果は「不明」になります。

例: 以降の例を参照する場合、次の表を使用してください。

TBL*AB*:

COL*A*	COL*B*
1	12
2	12
3	13
4	14
-	-

TBL*XY*:

COL*X*	COL*Y*
2	22
3	23

図 13.

例 1

```
SELECT COLA FROM TBLAB
   WHERE COLA = ANY(SELECT COLX FROM TBLXY)
```

結果は 2、3 です。副選択は (2,3) を戻します。行 2 と 3 の COLA は、それらの値の少なくとも 1 つに等しくなっています。

例 2

```
SELECT COLA FROM TBLAB
   WHERE COLA > ANY(SELECT COLX FROM TBLXY)
```

結果は 3、4 です。副選択は (2,3) を戻します。行 3 と 4 の COLA は、それらの値の少なくとも 1 つより大きくなっています。

例 3

```
SELECT COLA FROM TBLAB
   WHERE COLA > ALL(SELECT COLX FROM TBLXY)
```

結果は 4 です。副選択は (2,3) を戻します。それらの値の両方より大きいものは、行 4 の COLA しかありません。

例 4

```
SELECT COLA FROM TBLAB
   WHERE COLA > ALL(SELECT COLX FROM TBLXY
                    WHERE COLX<0)
```

結果は 1、2、3、4、NULL 値です。副選択は値を戻しません。したがって、述部は TBLAB のすべての行に対して真です。

例 5

```
SELECT * FROM TBLAB
   WHERE (COLA,COLB+10) = SOME (SELECT COLX, COLY FROM TBLXY)
```

副選択は TBLXY からすべての項目を戻します。述部は副選択に対して真であるため、結果は次のようになります。

```
COLA        COLB
----------- -----------
          2          12
          3          13
```

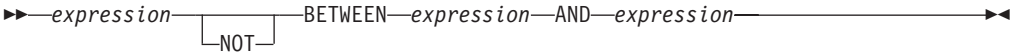
例 6

```
SELECT * FROM TBLAB
   WHERE (COLA,COLB) = ANY (SELECT COLX,COLY-10 FROM TBLXY)
```

副選択は TBLXY から COLX および COLY-10 を戻します。述部は副選択に対して真であるため、結果は次のようになります。

```
COLA        COLB
----------- -----------
          2          12
          3          13
```

BETWEEN 述部



BETWEEN 述部は、ある値を値の範囲と比較します。

次の BETWEEN 述部は、

```
value1 BETWEEN value2 AND value3
```

次の検索条件と同等です。

```
value1 >= value2 AND value1 <= value3
```

次の BETWEEN 述部は、

```
value1 NOT BETWEEN value2 AND value3
```

次の検索条件と同等です。

```
NOT(value1 BETWEEN value2 AND value3); that is,
value1 < value2 OR value1 > value3.
```

第 1 オペランド (expression) 内で、可変の関数または外部処理を伴う関数を使用することはできません (SQLSTATE 426804)。

日付/時刻値と日付/時刻値のストリング表記が混在している場合、すべての値は日付/時刻オペランドのデータ型に変換されます。

例:

例 1

```
EMPLOYEE.SALARY BETWEEN 20000 AND 40000
```

結果は $20,000.00 と $40,000.00 の間のすべての給与となります。

例 2

```
SALARY NOT BETWEEN 20000 + :HV1 AND 40000
```

:HV1 が 5000 であるとすると、結果は $25,000.00 より低いか $40,000.00 より高いすべての給与となります。

EXISTS 述部

EXISTS 述部は、特定の行の存在を調べるためのものです。

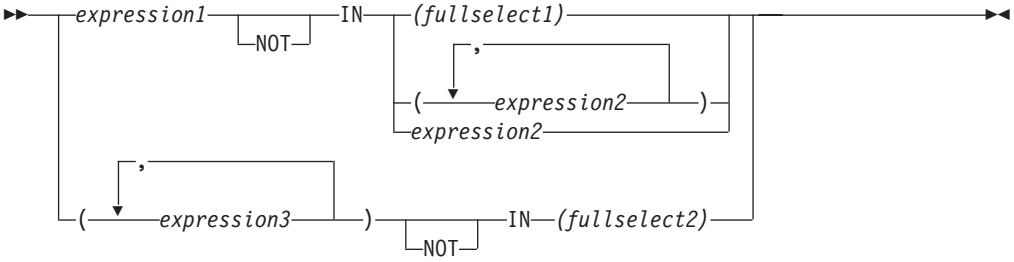
fullselect (全選択) には、必要な数の列を指定できます。

- fullselect に指定された行数がゼロでない場合にのみ、結果が「真」になります。
- 指定された行数がゼロの場合にのみ、結果が「偽」になります。
- 結果が「不明」になることはありません。

例:

```
EXISTS (SELECT * FROM TEMPL WHERE SALARY < 10000)
```

IN 述部



IN 述部は、1 つの値または複数の値を値の集合と比較します。

全選択は、 IN キーワードの左側に指定されている式の数と同じ数の列を識別しなければなりません (SQLSTATE 428C4)。全選択は、任意の行数を戻すことができます。

- 次の形式の IN 述部があるとします。

  ```
  expression IN expression
  ```

  上記は、以下の形式の基本述部と同等です。

  ```
  expression = expression
  ```

- 次の形式の IN 述部があるとします。

  ```
  expression IN (fullselect)
  ```

  上記は、以下の形式の比較述部と同等です。

  ```
  expression = ANY (fullselect)
  ```

- 次の形式の IN 述部があるとします。

  ```
  expression NOT IN (fullselect)
  ```

  上記は、以下の形式の比較述部と同等です。

  ```
  expression <> ALL (fullselect)
  ```

- 次の形式の IN 述部があるとします。

  ```
  expression IN (expressiona, expressionb, ..., expressionk)
  ```

  上記は、以下と同等です。

  ```
  expression = ANY (fullselect)
  ```

  この fullselect は、values 文節形式では次のようになります。

  ```
  VALUES (expressiona), (expressionb), ..., (expressionk)
  ```

- 次の形式の IN 述部があるとします。

  ```
  (expressiona, expressionb,..., expressionk) IN (fullselect)
  ```

  上記は、以下の形式の比較述部と同等です。

  ```
  (expressiona, expressionb,..., expressionk) = ANY (fullselect)
  ```

IN 述部の *expression1* および *expression2* の値、または *fullselect1* の列には、互換性が必要です。 IN 述部の *expression3* の各値、およびそれに対応する *fullselect2* の列にも互換性が必要です。結果データ型の規則を使って、比較で使用される結果の属性を判別することができます。

IN 述部の式の値 (全選択の対応する列を含めて) のコード・ページが異なっていても構いません。変換が必要な場合にコード・ページを判別するには、まず IN リストに対してストリング変換の規則を適用し、次に第 2 オペランドとして IN リストの派生コード・ページを使って同じ規則を述部に適用します。

例:

例 1: DEPTNO 列で評価の対象となる行の値に D01、B01、または C01 が入っている場合、以下は真であると評価されます。

```
DEPTNO IN ('D01', 'B01', 'C01')
```

例 2: 左側の EMPNO (従業員番号) が部門 E11 の従業員の EMPNO と一致する場合のみ、以下は真であると評価されます。

```
EMPNO IN (SELECT EMPNO FROM EMPLOYEE WHERE WORKDEPT = 'E11')
```

例 3: 以下の情報に基づき、COL_1 列の行の特定の値が、リスト内のいずれかの値と一致する場合、この例は真であると評価されます。

表 15. IN 述部の例

式	タイプ	コード・ページ
COL_1	列	850
HV_2	ホスト変数	437
HV_3	ホスト変数	437
CON_1	定数	850

ここで、次のような述部を評価します。

```
COL_1 IN (:HV_2, :HV_3, CON_4)
```

この場合、ストリング変換の規則に基づいて、2 個のホスト変数がコード・ページ 850 に変換されます。

例 4: EMENDATE に指定された年 (プロジェクトの従業員の活動が終了した日付) が、リストに指定された値のいずれか (現在の年または過去 2 年) と一致する場合、以下は真と評価されます。

```
YEAR(EMENDATE) IN (YEAR(CURRENT DATE),
                   YEAR(CURRENT DATE - 1 YEAR),
                   YEAR(CURRENT DATE - 2 YEARS))
```

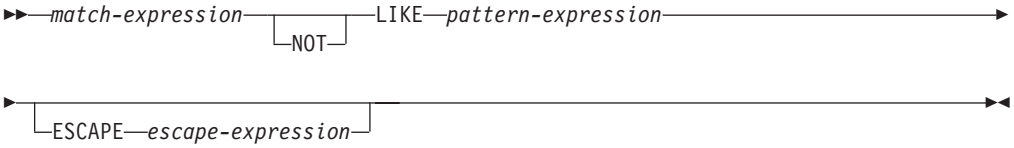
例 5: 左側の ID と DEPT の両方が、 ORG 表の任意の行の MANAGER と DEPTNUMB にそれぞれ一致する場合、以下は真と評価されます。

```
(ID, DEPT) IN (SELECT MANAGER, DEPTNUMB FROM ORG)
```

関連資料:

- 126 ページの『結果データ型の規則』
- 131 ページの『ストリング変換の規則』

LIKE 述部



LIKE 述部は、ある一定のパターンをもったストリングを探索するものです。パターンは、特殊な意味のある下線とパーセント記号を使ったストリングによって指定されます。パターンでは後続ブランクもパターンの一部です。

引き数うちのいずれかの値が NULL 値である場合、LIKE 述部の結果は不明になります。

match-expression (一致式)、 *pattern-expression* (パターン式)、および *escape-expression* (エスケープ式) の値は、互換性のある文字ストリング式です。サポートされる文字ストリング式のタイプは、各引き数ごとに少しずつ違います。式の有効なタイプは、各引き数の説明の中に示します。

どの式も特殊タイプを生成するものであってはなりません。ただし、特殊タイプをそのソース・タイプへキャストする関数は可能です。

match-expression
特定の文字パターンに適合するかどうか調べる対象のストリングを指定する式。

この式は、以下によって指定できます。

- 定数
- 特殊レジスター
- ホスト変数 (ロケーター変数またはファイル参照変数を含む)
- スカラー関数
- ラージ・オブジェクトのロケーター
- 列名
- 上記のいずれかを連結する式

pattern-expression
一致すべき基準となるストリングを指定する式。

この式は、以下によって指定できます。

- 定数
- 特殊レジスター
- ホスト変数
- 上記のいずれかをオペランドとするスカラー関数
- 上記のいずれかを連結する式

次のような制約があります。

- 式のエレメントに、LONG VARCHAR、CLOB、LONG VARGRAPHIC、または DBCLOB のタイプを使うことはできません。また、BLOB ファイル参照変数は使用できません。

- *pattern-expression* の実際の長さは、 32 672 バイト以下でなければなりません。

LIKE 述部について**簡単に説明**すると、これは *match-expression* の値のための適合基準を指定するために使用されるパターンです。これには以下の規則があります。

- 下線文字 (_) は、任意の 1 文字を表します。
- パーセント記号 (%) は、ゼロ個以上の文字から構成されるストリングを表します。
- その他の文字は、その文字自身を表します。

pattern-expression に下線またはパーセント文字を使用する必要がある場合は、*escape-expression* によって、パターンの中で下線またはパーセント文字に先行させる文字を指定します。

LIKE 述部について**厳密に説明**すると、以下のようになります。この説明では、*escape-expression* の使用は無視されています。それについては後で説明します。

- *m* が *match-expression* の値を、 *p* が *pattern-expression* の値を表すとします。ストリング *p* は、一連の最小数のサブストリング指定子と解釈されるので、 *p* の各文字はただ 1 つのサブストリング指定子の一部となります。サブストリング指定子とは、下線、パーセント記号、または下線およびパーセント記号以外の任意の空でない一連の文字です。

  m または *p* が NULL 値の場合は、述部の結果が不明になります。それ以外の場合の結果は、真か偽のどちらかになります。 *m* と *p* の両方が空ストリングの場合、または以下のようにして *m* をサブストリングにパーティション化したものが存在する場合、結果は真になります。

  – *m* のサブストリングがゼロ個以上の連続する文字の並びで、 *m* の各文字がちょうど 1 つのサブストリングの一部である。
  – *n* 番目のサブストリング指定子が下線の場合、 *m* の *n* 番目のサブストリング指定子は任意の 1 文字である。
  – *n* 番目のサブストリング指定子がパーセント記号の場合、 *m* の *n* 番目のサブストリング指定子は 0 個以上の文字の並びである。
  – *n* 番目のサブストリング指定子が下線でもパーセント記号でもない場合、*m* の *n* 番目のサブストリングは、対応するサブストリング指定子と等しく、同じ長さである。
  – *m* のサブストリングの数は、サブストリング指定子の数と同じである。

  したがって、*p* が空ストリングで、 *m* が空ストリングでない場合、結果は偽になります。同様に、*m* が空ストリングで、 *p* が空ストリングでない (% 記号だけから成るストリングを除く) 場合、結果は偽になります。

  述部 *m* NOT LIKE *p* は、検索条件 NOT (*m* LIKE *p*) と同等です。

escape-expression が指定されている場合、直後にエスケープ文字、下線文字、またはパーセント記号文字が続くのでない限り、 *pattern-expression* の中に、*escape-expression* で指定されるエスケープ文字が入っていてはなりません (SQLSTATE 22025)。

match-expression が MBCS データベースの文字ストリングの場合、それには混合データを収容することができます。この場合は、パターン内で SBCS 文字と非 SBCS 文字の両方を使用することができます。非 Unicode データベースの場合、パターンの中の特殊文字は、以下のようにして解釈されます。

- SBCS の半角下線文字は、1 つの SBCS 文字を表します。
- 非 SBCS の全角下線文字は、1 つの非 SBCS 文字を表します。
- SBCS の半角または非 SBCS の全角 % 記号文字は、0 以上の SBCS または非 SBCS 文字を表します。

Unicode データベースでは、「単一バイト」文字と「非単一バイト」文字の間に実質的な区別がありません。また UTF-8 フォーマットは Unicode 文字の「混合バイト」エンコード方式ですが、 UTF-8 では、SBCS 文字と非 SBCS 文字の間に実質的な区別がありません。 UTF-8 フォーマットでは、文字のバイト数に関係なく、すべての文字が Unicode 文字になります。

Unicode GRAPHIC 列では、半角下線文字 (U+005F) や半角 % 記号文字 (U+0025) を含め、補足文字以外のすべての文字が 2 バイト幅になります。Unicode データベースの場合、パターンの中の特殊文字は次のように解釈されます。

- 文字ストリングでは、半角下線文字 (X'5F') または全角下線文字 (X'EFBCBF') が 1 つの Unicode 文字を表し、半角 % 記号文字 (X'25') または全角 % 記号文字 (X'EFBC85') が 0 以上の Unicode 文字を表します。
- GRAPHIC ストリングでは、半角下線文字 (U+005F) または全角下線文字 (U+FF3F) が 1 つの Unicode 文字を表し、半角 % 記号文字 (U+0025) または全角 % 記号文字 (U+FF05) が 0 以上の Unicode 文字を表します。

GRAPHIC 列では、Unicode の補足 GRAPHIC 文字が UCS-2 文字 2 つ分で表されるため、この補足 GRAPHIC 文字と下線文字を対等にするには、下線文字が 2 つ必要です。しかし、文字の列では、下線文字 1 つが Unicode の補足文字と等しくなります。

escape-expression

これはオプションの引き数であり、 *pattern-expression* の中の下線 (_) 文字とパーセント (%) 文字の特別な意味を変更するための文字を指定する式です。これにより、実際にパーセントや下線文字の入った値との一致を調べるために LIKE 述部を使うことができます。

この式は、以下のいずれかによって指定できます。

- 定数
- 特殊レジスター
- ホスト変数
- 上記のいずれかをオペランドとするスカラー関数
- 上記のいずれかを連結する式

以下の制約があります。

- 式のエレメントに、LONG VARCHAR、CLOB、LONG VARGRAPHIC、または DBCLOB のタイプを使うことはできません。また、BLOB ファイル参照変数は使用できません。

- 文字の列の場合、式の結果は 1 文字、つまり 1 バイトだけの入ったバイナリー・ストリングになります (SQLSTATE 22019)。
- GRAPHIC 列の場合、式の結果は 1 つの文字になります (SQLSTATE 22019)。

パターン・ストリングにエスケープ文字が入っている場合、下線、パーセント記号、またはエスケープ文字は、それ自体のリテラル・オカレンスを表すことができます。これは、その文字が奇数個の連続したエスケープ文字に先行されている場合です。そうでない場合は当てはまりません。

パターンの中で、連続するエスケープ文字の並びは以下のように扱われます。

- S がそのような並びであり、さらに長く連続するエスケープ文字の並びの一部となってはいないものとします。また、S が合計 n 個の文字を収めているものとします。このとき、S に適用される規則は、n の値により以下のように異なります。
  - n が奇数の場合、S の後には下線またはパーセント記号がなければなりません (SQLSTATE 22025)。 S とその後に続く文字は、エスケープ文字の (n-1)/2 個のリテラル・オカレンスの後に下線記号またはパーセント記号のリテラル・オカレンスが続くことを表します。
  - n が偶数の場合、S はエスケープ文字の n/2 個のリテラル・オカレンスを表します。 n が奇数の場合とは異なり、S でパターンが終了する場合もあります。 S でパターンが終了しない場合、S の後にはどんな文字が続いても構いません (ただし、 S がさらに長く連続したエスケープ文字の並びの一部ではないという前提に違反するため、当然エスケープ文字は除外されます)。 S の後に下線記号またはパーセント記号が続く場合、その文字には特別な意味があります。

以下は、エスケープ文字 (この場合は、円記号 (¥)) の連続したオカレンスの影響を示しています。

パターン・ストリング	実際のパターン
\%	パーセント記号
¥¥%	1 つの円記号の後にゼロ個以上の任意の文字が続く
¥¥%	1 つの円記号の後に 1 つのパーセント記号が続く

比較で使用されるコード・ページは、 *match-expression* の値のコード・ページに基づいて決定されます。

- *match-expression* の値が変換されることはありません。
- *pattern-expression* のコード・ページが、 *match-expression* のコード・ページと異なる場合、 *pattern-expression* の値が *match-expression* のコード・ページに変換されます。ただし、どちらかのオペランドが FOR BIT DATA で定義されている場合は除きます (その場合は変換されません)。
- *escape-expression* のコード・ページが、 *match-expression* のコード・ページと異なる場合、 *escape-expression* の値が *match-expression* のコード・ページに変換されます。ただし、どちらかのオペランドが FOR BIT DATA で定義されている場合は除きます (その場合は変換されません)。

注:

- 末尾ブランクの数は、 *match-expression* と *pattern-expression* のどちらでも重要です。ストリング同士が同じ長さでない場合に、短いほうのストリングにブランクが埋め込まれることはありません。たとえば、'PADDED ' LIKE 'PADDED' という式は、一致していないことになります。
- LIKE 述部でパラメーター・マーカーをパターンとして指定した場合に、そのパラメーター・マーカーを固定長文字のホスト変数に置き換える場合は、正しい長さのホスト変数値を指定しなければなりません。正しい長さを指定しないと、所定の結果が SELECT から戻されません。

  たとえば、ホスト変数を CHAR(10) と定義した場合に、そのホスト変数に値 WYSE% を割り当てると、割り当ての際にそのホスト変数はブランクで埋め込まれます。以下のパターンが使用されます。

  ```
  'WYSE%     '
  ```

  データベース・マネージャーは、 WYSE で始まって 5 つのブランク・スペースで終わるすべての値を検索します。 WYSE で始まる値だけを検索するつもりの場合は、ホスト変数に値 WSYE%%%%%% を割り当てます。

例:

- PROJECT 表の PROJNAME 列で 'SYSTEMS' というストリングを探索します。

  ```
  SELECT PROJNAME FROM PROJECT
    WHERE PROJECT.PROJNAME LIKE '%SYSTEMS%'
  ```

- EMPLOYEE 表の FIRSTNME 列で、先頭の文字が 'J' で、長さがちょうど 2 文字のストリングを探索します。

  ```
  SELECT FIRSTNME FROM EMPLOYEE
    WHERE EMPLOYEE.FIRSTNME LIKE 'J_'
  ```

- EMPLOYEE 表の FIRSTNME 列で、先頭の文字が 'J' で、任意の長さのストリングを探索します。

  ```
  SELECT FIRSTNME FROM EMPLOYEE
    WHERE EMPLOYEE.FIRSTNME LIKE 'J%'
  ```

- CORP_SERVERS 表で、LA_SERVERS 列のストリングのうち、 CURRENT SERVER 特殊レジスターの値と一致するものを探索します。

  ```
  SELECT LA_SERVERS FROM CORP_SERVERS
    WHERE CORP_SERVERS.LA_SERVERS LIKE CURRENT SERVER
  ```

- 表 T の列 A 中の文字シーケンス %_¥ で始まるすべてのストリングを検索します。

  ```
  SELECT A FROM T
    WHERE T.A LIKE '¥;%¥_¥¥%' ESCAPE '¥'
  ```

- 一致ストリングとパターン・ストリングのデータ型 (どちらも BLOB) と互換である 1 バイトのエスケープ文字を獲得するには、次のように BLOB スカラー関数を使用します。

  ```
  SELECT COLBLOB FROM TABLET
    WHERE COLBLOB LIKE :pattern_var ESCAPE BLOB(X'0E')
  ```

NULL 述部

NULL 述部は、NULL 値かどうかを検査するものです。

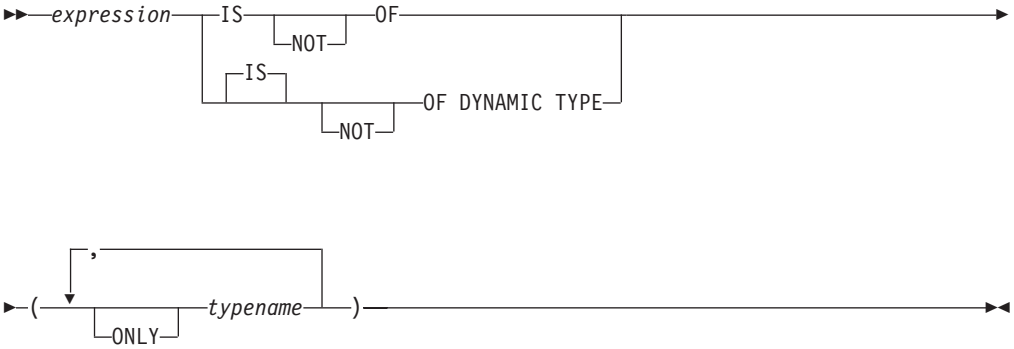
NULL 述部の結果が不明になることはありません。式の値が NULL 値の場合、結果は真になります。値が NULL 値でない場合、結果は偽になります。 NOT が指定されている場合、結果は逆になります。

例:

```
PHONENO IS NULL

SALARY IS NOT NULL
```

TYPE 述部



TYPE 述部 は、式のタイプと 1 つまたは複数のユーザー定義構造化タイプとを比較します。

参照タイプの間接参照に関与した式の動的タイプは、ターゲットのタイプ表またはビューにある参照される行の実際のタイプです。これは、その参照に関与した式のターゲット・タイプ (式の静的タイプと呼ばれる) とは異なる場合があります。

expression の値が NULL の場合、述部の結果は不明です。 *expression* の動的タイプが *typename* で指定された構造化タイプの 1 つのサブタイプの場合、述部の結果は「真」になり、そうでない場合は「偽」になります。 ONLY のあとに *typename* がある場合、その型の適切なサブタイプは考慮されません。

typename が修飾されていない場合、SQL パスを使用して解決されます。 *typename* は、 *expression* の静的タイプのタイプ階層にあるユーザー定義タイプを識別しなければなりません (SQLSTATE 428DU)。

DEREF 関数は、参照タイプの値に関与した式が TYPE 述部にある場合はいつでも、使用されなければなりません。 *expression* がこの形式の場合の静的タイプは、参照のターゲット・タイプです。

構文上の IS OF と OF DYNAMIC TYPE は、TYPE 述部では同じ働きをします。同様に、IS NOT OF と NOT OF DYNAMIC TYPE も TYPE 述部では同じ働きをします。

例:

ある表階層には、タイプ EMP のルート表 EMPLOYEE と、タイプ MGR の副表 MANAGER があります。別の表 ACTIVITIES は、 REF(EMP) SCOPE EMPLOYEE として定義されている WHO_RESPONSIBLE という列を備えています。WHO_RESPONSIBLE と対応する行が管理者の場合に、結果が「真」となるタイプ述部を以下に示します。

```
DEREF (WHO_RESPONSIBLE) IS OF (MGR)
```

表にタイプ EMP の列 EMPLOYEE が入っている場合、EMPLOYEE には、タイプ EMP の値だけではなく、MGR のようなサブタイプの値を使用することができます。次のような述部は、

```
EMPL IS OF (MGR)
```

EMPL が NULL ではなく、実際に管理職である場合に、「真」を戻します。

関連資料:

- 333 ページの『DEREF』

第 3 章 関数

関数の概要

関数 とは、関数名の後に 1 対の括弧で囲んだ引き数の指定を伴う演算です (引き数がない場合もあります)。

組み込み関数 は、データベース・マネージャーに提供されている関数です。この関数は、1 つの結果値を戻し、SYSIBM スキーマの一部として識別されます。組み込み関数には、列関数 (AVG など)、演算子関数 (『+』 など)、 cast 関数 (DECIMAL など)、およびその他の関数 (SUBSTR など) などがあります。

ユーザー定義関数 は、 (CREATE FUNCTION ステートメントを使用して) SYSCAT.ROUTINES のデータベースに登録されます。ユーザー定義関数は SYSIBM スキーマの一部ではありません。このような関数のセットは、1 つは SYSFUN という名前のスキーマで、もう 1 つは SYSPROC というスキーマで、データベース・マネージャーに付属しています。

関数は、集約 (列) 関数、スカラー関数、行関数、および表関数に分類されます。

- *列関数* の引き数は、互いに似通った値の集合です。列関数は、単一の値 (おそらくは NULL) を戻しますが、式を使用できる場所であればどこでも SQL ステートメント内に指定できます。
- *スカラー関数* の引き数は、それぞれ型や意味が異なる可能性のある個々のスカラー値です。スカラー関数は、単一の値 (おそらくは NULL) を戻しますが、式を使用できる場所であればどこでも SQL ステートメント内に指定できます。
- *行関数* の引き数は、構造化タイプです。行関数は、組み込みデータ型の行を戻しますが、構造化タイプの変換関数としてだけ指定できます。
- *表関数* の引き数は、それぞれ型や意味が異なる可能性のある個々のスカラー値です。表関数は、SQL ステートメントへ表を戻し、SELECT ステートメントの FROM 文節にのみ指定することができます。

関数の完全修飾名は、関数名とスキーマの組み合わせからなっています。スキーマ、関数名、および入力パラメーターの組み合わせが*関数シグニチャー* を構成します。

入力パラメーターのタイプは、特定の組み込みデータ型と指定される場合と、*any-numeric-type* (任意の数値タイプの意) のような汎用変数を使用して指定される場合があります。特定のデータ型が指定される場合は、指定されたデータ型に対してのみ正しい一致が成立します。一方、汎用変数が使用される場合は、その変数に関連付けられているどのデータ型に対しても正しい一致が成立します。

関数シグニチャーの 1 つをソースにして別のスキーマでユーザー定義関数を作成することにより、追加の関数も使用できます。アプリケーションで外部関数を作成することも可能です。

関連概念:

- 274 ページの『集約関数』

関連資料:

- 166 ページの『関数』
- 474 ページの『副選択』
- 「*SQL リファレンス 第 2 巻*」の『CREATE FUNCTION ステートメント』

サポートされている関数と SQL 管理ルーチン

表 16 は、サポートされている関数と SQL 管理ルーチンの詳細を要約しています。関数の完全修飾名は、関数名とスキーマの組み合わせからなっています。「入力パラメーター」列には、関数呼び出し時の各引き数の所定のデータ型を示します。関数の多くには、さまざまな入力パラメーターが備えられており、異なるデータ型または異なる数の引き数を使用することができます。スキーマ、関数名、および入力パラメーターの組み合わせが関数シグニチャーを構成します。「戻り値のタイプ」列には、関数から戻される値として使用可能なデータ型が示されます。

型別に分類されたサポート対象の組み込み関数のリストは、以下の表を参照してください。

- 集約関数 (267 ページの表 17)
- CAST スカラー関数 (268 ページの表 18)
- DATALINK スカラー関数 (268 ページの表 19)
- DATETIME スカラー関数 (269 ページの表 20)
- パーティション化スカラー関数 (270 ページの表 21)
- 数値スカラー関数 (271 ページの表 22)
- ストリング・スカラー関数 (272 ページの表 23)
- その他のスカラー関数 (273 ページの表 24)

表 16. サポートされている関数

関数名	スキーマ 入力パラメーター	説明 戻り値のタイプ
294 ページの『ABS または ABSVAL』	SYSIBM	このスカラー関数は、引き数の絶対値を戻します。
	組み込み数値データ型を返す式。	引き数と同じデータ型および長さ
294 ページの『ABS または ABSVAL』	SYSFUN	このスカラー関数は、引き数の絶対値を戻します。
	SMALLINT	SMALLINT
	INTEGER	INTEGER
	BIGINT	BIGINT
	DOUBLE	DOUBLE
295 ページの『ACOS』	SYSIBM	このスカラー関数は、引き数のアークコサイン (逆余弦) の角度を戻します (ラジアン単位)。
	DOUBLE	DOUBLE
ALTOBJ	SYSPROC	このプロシージャー (SQL 管理ルーチン) は、変更予定の既存表のターゲット・データ定義言語 (DDL) として働く入力 CREATE TABLE ステートメントの構文を解析します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
AM_BASE_RPT_RECOMS	SYSPROC	この表関数 (SQL 管理ルーチン) は、アクティビティー・モニターで使用されるアクティビティー報告書に関する勧告を戻します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
AM_BASE_RPTS	SYSPROC	この表関数 (SQL 管理ルーチン) は、アクティビティー・モニターで使用されるアクティビティー報告書を戻します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
AM_DROP_TASK	SYSPROC	このプロシージャー (SQL 管理ルーチン) は、モニター・タスクを削除します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	

表 16. サポートされている関数 (続き)

関数名	スキーマ	説明
	入力パラメーター	戻り値のタイプ
AM_GET_LOCK_CHN_TB	SYSPROC	このプロシージャー (SQL 管理ルーチン) は、表フォーマットのアプリケーション・ロック・チェーン・データを戻します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
AM_GET_LOCK_CHNS	SYSPROC	このプロシージャー (SQL 管理ルーチン) は、フォーマット設定ストリングを使って、指定のアプリケーション用のロック・チェーンを表示します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
AM_GET_LOCK_RPT	SYSPROC	このプロシージャー (SQL 管理ルーチン) は、アプリケーションのロックの詳細を表示します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
AM_GET_RPT	SYSPROC	このプロシージャー (SQL 管理ルーチン) は、レポートのアクティビティー・モニター・データを表示します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
AM_SAVE_TASK	SYSPROC	このプロシージャー (SQL 管理ルーチン) は、モニター・タスクを作成または修正します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
APP	DB2EAS	このプロシージャー (SQL 管理ルーチン) は、 DB2 用のアプリケーション・サーバー上で指定のアプリケーションを開始または停止します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
APPLICATION_ID	SYSFUN	このスカラー関数 (SQL 管理ルーチン) は、現行接続のアプリケーション ID を戻します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
296 ページの『ASCII』	SYSFUN	このスカラー関数は、引き数の左端の文字の ASCII コード値を整数として戻します。
	CHAR	INTEGER
	VARCHAR(4000)	INTEGER
	CLOB(1M)	INTEGER
297 ページの『ASIN』	SYSIBM	このスカラー関数は、引き数のアークサイン (逆正弦) の値をラジアン単位で戻します。
	DOUBLE	DOUBLE
298 ページの『ATAN』	SYSIBM	このスカラー関数は、引き数のアークタンジェント (逆正接) の角度を戻します (ラジアン単位)。
	DOUBLE	DOUBLE
300 ページの『ATANH』	SYSIBM	このスカラー関数は、引き数の双曲線アークタンジェント (逆正接) の値を戻します。引き数はラジアン単位の角度です。
	DOUBLE	DOUBLE
299 ページの『ATAN2』	SYSIBM	このスカラー関数は、最初と 2 番目の引き数によってそれぞれ指定された *x* 座標と *y* 座標のアークタンジェント (逆正接) の値をラジアン単位の角度として戻します。
	DOUBLE, DOUBLE	DOUBLE
275 ページの『AVG』	SYSIBM	この集約関数は、一連の数値の平均値を戻します。
	numeric-type[4]	*numeric-type*[1]
301 ページの『BIGINT』	SYSIBM	このスカラー関数は、数値または文字ストリングを 64 ビットで表した整数を、整数定数の形で戻します。
	numeric-type	BIGINT
	VARCHAR	BIGINT

表 16. サポートされている関数 (続き)

関数名	スキーマ / 入力パラメーター	説明 / 戻り値のタイプ
303 ページの『BLOB』	SYSIBM	このスカラー関数は、オプションの長さを指定してソース・タイプから BLOB にキャストします。
	string-type	BLOB
	string-type, INTEGER	BLOB
304 ページの『CEILING または CEIL』	SYSIBM	このスカラー関数は、引き数よりも大きいか等しい最小の整数を戻します。
	SMALLINT	SMALLINT
	INTEGER	INTEGER
	BIGINT	BIGINT
	DOUBLE	DOUBLE
305 ページの『CHAR』	SYSIBM	このスカラー関数は、ソース・タイプのストリング表記を戻します。
	character-type	CHAR
	character-type, INTEGER	CHAR(*integer*)
	datetime-type	CHAR
	datetime-type, *keyword*[2]	CHAR
	SMALLINT	CHAR(6)
	INTEGER	CHAR(11)
	BIGINT	CHAR(20)
	DECIMAL	CHAR(2+*precision*)
	DECIMAL、VARCHAR	CHAR(2+*precision*)
305 ページの『CHAR』	SYSFUN	このスカラー関数は、浮動小数点数の文字ストリング表記を戻します。
	DOUBLE	CHAR(24)
309 ページの『CHR』	SYSFUN	このスカラー関数は、引き数で指定された ASCII コード値をもつ文字を戻します。引き数の値は 0 から 255 の範囲でなければなりません。そうでないと、戻り値は NULL 値になります。
	INTEGER	CHAR(1)
310 ページの『CLOB』	SYSIBM	このスカラー関数は、オプションの長さを指定してソース・タイプから CLOB にキャストします。
	character-type	CLOB
	character-type, INTEGER	CLOB
311 ページの『COALESCE』[3]	SYSIBM	このスカラー関数は、一連の引き数のうち、NULL 以外の最初の引き数を戻します。
	any-type, *any-union-compatible-type*, ...	*any-type*
312 ページの『CONCAT』	SYSIBM	このスカラー関数は、2 つのストリング引き数を連結して戻します。
	string-type, *compatible-string-type*	*max-string-type*
277 ページの『CORRELATION』	SYSIBM	この集約関数は、一連の数値ペアの相関係数を戻します。
	numeric-type, *numeric-type*	DOUBLE
313 ページの『COS』	SYSIBM	このスカラー関数は、引き数のコサイン (余弦) を戻します。引き数はラジアン単位の角度です。
	DOUBLE	DOUBLE
314 ページの『COSH』	SYSIBM	このスカラー関数は、引き数の双曲線コサイン (余弦) を戻します。引き数はラジアン単位の角度です。
	DOUBLE	DOUBLE
315 ページの『COT』	SYSIBM	このスカラー関数は、引き数のコタンジェント (余接) を戻します。引き数はラジアン単位の角度です。
	DOUBLE	DOUBLE

表 16. サポートされている関数 (続き)

関数名	スキーマ	説明
	入力パラメーター	戻り値のタイプ
278 ページの『COUNT』	SYSIBM	この集約関数は、一連の行または値の中の行または値の数を戻します。
	any-builtin-type[4]	INTEGER
279 ページの『COUNT_BIG』	SYSIBM	この集約関数は、一連の行または値の中の行または値の数を戻します。その結果は整数の最大値より大きい場合があります。
	any-builtin-type[4]	DECIMAL(31,0)
281 ページの『COVARIANCE』	SYSIBM	この集約関数は、一連の数値ペアの共分散を戻します。
	numeric-type, *numeric-type*	DOUBLE
316 ページの『DATE』	SYSIBM	このスカラー関数は、単一の入力値から日付を戻します。
	DATE	DATE
	TIMESTAMP	DATE
	DOUBLE	DATE
	VARCHAR	DATE
317 ページの『DAY』	SYSIBM	このスカラー関数は、値の日の部分を戻します。
	VARCHAR	INTEGER
	DATE	INTEGER
	TIMESTAMP	INTEGER
	DECIMAL	INTEGER
318 ページの『DAYNAME』	SYSFUN	このスカラー関数は、db2start が発行された時点のロケールに基づいて、引き数の日の部分の曜日名から成る大文字小文字混合文字ストリング (たとえば、Friday) を戻します。
	VARCHAR(26)	VARCHAR(100)
	DATE	VARCHAR(100)
	TIMESTAMP	VARCHAR(100)
319 ページの『DAYOFWEEK』	SYSFUN	このスカラー関数は、引き数内の曜日を 1 から 7 の範囲の整数値で戻します。1 は日曜日を表します。
	VARCHAR(26)	INTEGER
	DATE	INTEGER
	TIMESTAMP	INTEGER
320 ページの『DAYOFWEEK_ISO』	SYSFUN	このスカラー関数は、引き数内の曜日を 1 から 7 の範囲の整数値で戻します。1 は月曜日を表します。
	VARCHAR(26)	INTEGER
	DATE	INTEGER
	TIMESTAMP	INTEGER
321 ページの『DAYOFYEAR』	SYSFUN	このスカラー関数は、引き数内の年間通算日を、1 から 366 の範囲の整数値で戻します。
	VARCHAR(26)	INTEGER
	DATE	INTEGER
	TIMESTAMP	INTEGER
322 ページの『DAYS』	SYSIBM	このスカラー関数は、日付の整数表記を戻します。
	VARCHAR	INTEGER
	TIMESTAMP	INTEGER
	DATE	INTEGER
DB_PARTITIONS	SYSPROC	この表関数 (SQL 管理ルーチン) は、表形式の db2nodes.cfg ファイルの内容を戻します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	

表 16. サポートされている関数 (続き)

関数名	スキーマ / 入力パラメーター	説明 / 戻り値のタイプ
323 ページの『DBCLOB』	SYSIBM	このスカラー関数は、ソース・タイプからオプションの長さを指定して DBCLOB にキャストします。
	graphic-type	DBCLOB
	graphic-type, INTEGER	DBCLOB
324 ページの『DBPARTITIONNUM』 [3]	SYSIBM	このスカラー関数は、行のデータベース・パーティション番号を戻します。引き数は表内の列名です。
	any-type	INTEGER
326 ページの『DECIMAL』	SYSIBM	このスカラー関数は、オプションの精度と位取りを指定した 10 進表記の数値を戻します。
	numeric-type	DECIMAL
	numeric-type, INTEGER	DECIMAL
	numeric-type INTEGER, INTEGER	DECIMAL
326 ページの『DECIMAL』	SYSIBM	このスカラー関数は、オプションの精度、位取り、および小数点文字を指定した 10 進数表記の文字ストリングを戻します。
	VARCHAR	DECIMAL
	VARCHAR, INTEGER	DECIMAL
	VARCHAR, INTEGER, INTEGER	DECIMAL
	VARCHAR, INTEGER, INTEGER, VARCHAR	DECIMAL
330 ページの『DECRYPT_BIN および DECRYPT_CHAR』	SYSIBM	このスカラー関数は、パスワード・ストリングを使用した暗号化データの暗号化解除の結果である値を返します。
	VARCHAR FOR BIT DATA	VARCHAR FOR BIT DATA
	VARCHAR FOR BIT DATA, VARCHAR	VARCHAR FOR BIT DATA
330 ページの『DECRYPT_BIN および DECRYPT_CHAR』	SYSIBM	このスカラー関数は、パスワード・ストリングを使用した暗号化データの暗号化解除の結果である値を返します。
	VARCHAR FOR BIT DATA	VARCHAR
	VARCHAR FOR BIT DATA, VARCHAR	VARCHAR
332 ページの『DEGREES』	SYSFUN	このスカラー関数は、ラジアン単位の引き数から変換した度数を戻します。
	DOUBLE	DOUBLE
333 ページの『DEREF』	SYSIBM	このスカラー関数は、参照タイプ引き数のターゲット・タイプのインスタンスを戻します。
	有効範囲を定義された REF(*any-structured-type*)	*any-structured-type (入力ターゲット・タイプと同じ)*
334 ページの『DIFFERENCE』	SYSFUN	このスカラー関数は、SOUNDEX 関数によって判別された 2 つの引き数ストリングの語の音の相違を戻します。 4 の値は、それらのストリングが同じ音であることを意味します。
	VARCHAR(4000), VARCHAR(4000)	INTEGER
335 ページの『DIGITS』	SYSIBM	このスカラー関数は、数値の文字ストリング表記を戻します。
	DECIMAL	CHAR
336 ページの『DLCOMMENT』	SYSIBM	このスカラー関数は、DATALINK 値のコメント属性を戻します。
	DATALINK	VARCHAR(254)
337 ページの『DLLINKTYPE』	SYSIBM	このスカラー関数は、DATALINK 値のリンク・タイプ属性を戻します。
	DATALINK	VARCHAR(4)
338 ページの『DLNEWCOPY』	SYSIBM	このスカラー関数は、参照されているファイルが変更されたことを示す属性をもった DATALINK 値を戻します。
	VARCHAR(254)	DATALINK

表 16. サポートされている関数 (続き)

関数名	スキーマ	説明
	入力パラメーター	戻り値のタイプ
340 ページの『DLPREVIOUSCOPY』	SYSIBM	このスカラー関数は、旧バージョンのファイルをリストアする必要があることを示す属性をもった DATALINK 値を戻します。
	VARCHAR(254)	DATALINK
342 ページの『DLREPLACECONTENT』	SYSIBM	このスカラー関数は DATALINK 値を戻します。この関数は、UPDATE ステートメント内の SET 文節の右辺にあるか、または INSERT ステートメント内の VALUES 文節にあるとき、戻された値を割り当てることによって、ファイルの内容を別のファイルで置き換え、その後そのファイルへのリンクを作成します。
	VARCHAR(200)	DATALINK
	VARCHAR(200), VARCHAR(200)	DATALINK
344 ページの『DLURLCOMPLETE』	SYSIBM	このスカラー関数は、DATALINK 値の完全 URL (アクセス・トークンを備えている) を戻します。
	DATALINK	VARCHAR
345 ページの『DLURLCOMPLETEONLY』	SYSIBM	このスカラー関数は、 URL のリンク・タイプを持つ DATALINK 値から、完全な URL (アクセス・トークンを備えていない) を戻します。
	DATALINK	VARCHAR(254)
346 ページの『DLURLCOMPLETEWRITE』	SYSIBM	このスカラー関数は、URL のリンク・タイプを持つ DATALINK 値から、指定されたファイルの変更に必要な完全な URL 値を戻します。
	DATALINK	VARCHAR(254)
347 ページの『DLURLPATH』	SYSIBM	このスカラー関数は、DATALINK 値のパスおよびファイル名 (アクセス・トークンを備えている) を戻します。
	DATALINK	VARCHAR
348 ページの『DLURLPATHONLY』	SYSIBM	このスカラー関数は、DATALINK 値のパスおよびファイル名 (アクセス・トークンを備えていない) を戻します。
	DATALINK	VARCHAR
349 ページの『DLURLPATHWRITE』	SYSIBM	このスカラー関数は、特定のサーバー内のファイルの修正に必要なパスおよびファイル名を、 URL のリンク・タイプの DATALINK 値から戻します。
	DATALINK	VARCHAR(254)
350 ページの『DLURLSCHEME』	SYSIBM	このスカラー関数は、DATALINK 値の URL 属性からスキームを戻します。
	DATALINK	VARCHAR
351 ページの『DLURLSERVER』	SYSIBM	このスカラー関数は、DATALINK 値の URL 属性からサーバーを戻します。
	DATALINK	VARCHAR
352 ページの『DLVALUE』	SYSIBM	このスカラー関数は、データ位置付け引き数、リンク・タイプ引き数、およびオプションのコメント・ストリング引き数から DATALINK 値を作成します。
	VARCHAR	DATALINK
	VARCHAR, VARCHAR	DATALINK
	VARCHAR, VARCHAR, VARCHAR	DATALINK
354 ページの『DOUBLE』	SYSIBM	このスカラー関数は、数値の浮動小数点表記を戻します。
	numeric-type	DOUBLE
354 ページの『DOUBLE』	SYSFUN	このスカラー関数は、数値の文字ストリング表記に対応する浮動小数点数を戻します。*引き数* に先行ブランクや後続ブランクがあっても、それは無視されます。
	VARCHAR	DOUBLE

表 16. サポートされている関数 (続き)

関数名	スキーマ	説明
	入力パラメーター	戻り値のタイプ
356 ページの『ENCRYPT』	SYSIBM	このスカラー関数は、データ・ストリング式の暗号化の結果である値を返します。
	VARCHAR	VARCHAR FOR BIT DATA
	VARCHAR, VARCHAR	VARCHAR FOR BIT DATA
	VARCHAR, VARCHAR, VARCHAR	VARCHAR FOR BIT DATA
359 ページの『EVENT_MON_STATE』	SYSIBM	このスカラー関数は、特定のイベント・モニターの作動状態を戻します。
	VARCHAR	INTEGER
EXEC_DB2_SCRIPT	SYSPROC	このプロシージャー (SQL 管理ルーチン) は、DB2 インスタンスで DB2 コマンドを実行します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
360 ページの『EXP』	SYSFUN	このスカラー関数は、引き数の指数関数を戻します。
	DOUBLE	DOUBLE
361 ページの『FLOAT』	SYSIBM	このスカラー関数は、DOUBLE と同じです。
362 ページの『FLOOR』	SYSIBM	このスカラー関数は、引き数よりも小さいか等しい最大の整数を戻します。
	SMALLINT	SMALLINT
	INTEGER	INTEGER
	BIGINT	BIGINT
	DOUBLE	DOUBLE
364 ページの『GENERATE_UNIQUE』	SYSIBM	このスカラー関数は、同じ関数の他の実行に照らし合わせてユニークなビット・データ文字ストリングを戻します。
	引き数なし	CHAR(13) FOR BIT DATA
GET_DB_CONFIG	SYSFUN	この表関数 (SQL 管理ルーチン) は、データベース構成情報を戻します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
GET_DBM_CONFIG	SYSFUN	この表関数 (SQL 管理ルーチン) は、データベース・マネージャー構成情報を戻します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
GET_DBSIZE_INFO	SYSTOOLS	このプロシージャー (SQL 管理ルーチン) は、データベース・サイズと最大容量を計算します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
GET_ROUTINE_OPTS	SYSPROC	このスカラー関数 (SQL 管理ルーチン) は、現行セッションでの SQL プロシージャーの作成に使われる予定のオプションの文字ストリング値を戻します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
GET_ROUTINE_SAR	SYSFUN	このプロシージャー (SQL 管理ルーチン) は、最低限同じレベルおよびオペレーティング・システムで実行している別のデータベース・サーバーに、同一のルーチンをインストールするのに必要な情報を戻します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
363 ページの『GETHINT』	SYSIBM	このスカラー関数は、パスワードのヒントが検出された場合にそれを戻します。
	VARCHAR または CLOB	VARCHAR
366 ページの『GRAPHIC』	SYSIBM	このスカラー関数は、オプションの長さを指定してソース・タイプから GRAPHIC にキャストします。
	graphic-type	GRAPHIC
	graphic-type, INTEGER	GRAPHIC

表 16. サポートされている関数 (続き)

関数名	スキーマ / 入力パラメーター	説明 / 戻り値のタイプ
282 ページの『GROUPING』	SYSIBM	この集約関数は、グループ化集約によって生成された小計行を示すために、グループ化集約およびスーパー・グループで使用されます。返される値は 0 または 1 です。 1 の値は、戻された行の引き数の値は NULL 値であり、行がグループ化集合用に生成されたことを意味します。生成されたこの行は、グループ化集合の小計を示します。
	any-type	SMALLINT
368 ページの『HASHEDVALUE』 [3]	SYSIBM	このスカラー関数は、行の区分化マップ索引 (0 から 4095) を戻します。引き数は表内の列名です。
	any-type	INTEGER
HEALTH_DB_HIC	SYSPROC	この表関数 (SQL 管理ルーチン) は、データベースのヘルス・スナップショットからヘルス・インディケーターを集めた情報を戻します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
HEALTH_DB_HIC_HIS	SYSPROC	この表関数 (SQL 管理ルーチン) は、データベースのヘルス・スナップショットからヘルス・インディケーター履歴を集めた情報を戻します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
HEALTH_CONT_HI	SYSPROC	この表関数 (SQL 管理ルーチン) は、データベースのヘルス・スナップショットからコンテナーに関するヘルス・インディケーター情報を載せた表を戻します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
HEALTH_CONT_HI_HIS	SYSPROC	この表関数 (SQL 管理ルーチン) は、データベースのヘルス・スナップショットからコンテナーに関するヘルス・インディケーター履歴情報を載せた表を戻します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
HEALTH_CONT_INFO	SYSPROC	この表関数 (SQL 管理ルーチン) は、データベースのヘルス・スナップショットからコンテナー情報を載せた表を戻します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
HEALTH_DB_HI	SYSPROC	この表関数 (SQL 管理ルーチン) は、データベースのヘルス・スナップショットからヘルス・インディケーター情報を載せた表を戻します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
HEALTH_DB_HI_HIS	SYSPROC	この表関数 (SQL 管理ルーチン) は、データベースのヘルス・スナップショットからヘルス・インディケーター履歴情報を載せた表を戻します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
HEALTH_DB_INFO	SYSPROC	この表関数 (SQL 管理ルーチン) は、データベースのヘルス・スナップショットからの情報を載せた表を戻します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
HEALTH_DBM_HI	SYSPROC	この表関数 (SQL 管理ルーチン) は、 DB2 データベース・マネージャーのヘルス・スナップショットからヘルス・インディケーター情報を載せた表を戻します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
HEALTH_DBM_HI_HIS	SYSPROC	この表関数 (SQL 管理ルーチン) は、 DB2 データベース・マネージャーのヘルス・スナップショットからヘルス・インディケーター履歴情報を載せた表を戻します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
HEALTH_DBM_INFO	SYSPROC	この表関数 (SQL 管理ルーチン) は、 DB2 データベース・マネージャーのヘルス・スナップショットから情報を示した表を戻します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	

表 16. サポートされている関数 (続き)

関数名	スキーマ 入力パラメーター	説明 戻り値のタイプ
HEALTH_HI_REC	SYSPROC	このプロシージャー (SQL 管理ルーチン) は、特定の DB2 オブジェクト上のアラート状態にあるヘルス・インディケーターに関連した一連の推奨事項を取り出します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
HEALTH_TBS_HI	SYSPROC	この表関数 (SQL 管理ルーチン) は、データベースのヘルス・スナップショットから表スペースに関するヘルス・インディケーター情報を載せた表を戻します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
HEALTH_TBS_HI_HIS	SYSPROC	この表関数 (SQL 管理ルーチン) は、データベースのヘルス・スナップショットから表スペースに関するヘルス・インディケーター履歴情報を載せた表を戻します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
HEALTH_TBS_INFO	SYSPROC	この表関数 (SQL 管理ルーチン) は、データベースのヘルス・スナップショットから表スペース情報を載せた表を戻します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
370 ページの『HEX』	SYSIBM	このスカラー関数は、値の 16 進表記を戻します。
	any-builtin-type	VARCHAR
372 ページの『HOUR』	SYSIBM	このスカラー関数は、値の時間の部分を戻します。
	VARCHAR	INTEGER
	TIME	INTEGER
	TIMESTAMP	INTEGER
	DECIMAL	INTEGER
373 ページの『IDENTITY_VAL_LOCAL』	SYSIBM	このスカラー関数は、ID 列に一番後で割り当てられた値を戻します。
		DECIMAL
378 ページの『INSERT』	SYSFUN	このスカラー関数は、*argument2* から始まる *argument3* バイトを *argument1* から削除し、 *argument2* から始まる *argument1* に *argument4* を挿入したストリングを戻します。
	VARCHAR(4000), INTEGER, INTEGER, VARCHAR(4000)	VARCHAR(4000)
	CLOB(1M), INTEGER, INTEGER, CLOB(1M)	CLOB(1M)
	BLOB(1M), INTEGER, INTEGER, BLOB(1M)	BLOB(1M)
INSTALLAPP	DB2EAS	このプロシージャー (SQL 管理ルーチン) は、 DB2 用のアプリケーション・サーバーに指定のアプリケーションをインストールします。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
380 ページの『INTEGER』	SYSIBM	このスカラー関数は、数値の整数表記を戻します。
	numeric-type	INTEGER
	VARCHAR	INTEGER
382 ページの『JULIAN_DAY』	SYSFUN	このスカラー関数は、紀元前 4712 年 1 月 1 日 (ユリウス暦の起点) からの経過日数を表す整数値を *argument* に指定された日付値に戻します。
	VARCHAR(26)	INTEGER
	DATE	INTEGER
	TIMESTAMP	INTEGER
384 ページの『LCASE (SYSFUN スキーマ)』	SYSFUN	このスカラー関数は、すべての文字が小文字に変換されたストリングを戻します。 LCASE は、不変セットの文字だけを処理します。したがって、LCASE(UCASE(string)) は LCASE(string) と同じ結果を戻すとはかぎりません。
	VARCHAR(4000)	VARCHAR(4000)
	CLOB(1M)	CLOB(1M)

表 16. サポートされている関数 (続き)

関数名	スキーマ	説明	
	入力パラメーター		戻り値のタイプ
383 ページの『LCASE または LOWER』	SYSIBM	このスカラー関数は、すべての文字が小文字に変換されたストリングを戻します。	
	CHAR		CHAR
	VARCHAR		VARCHAR
385 ページの『LEFT』	SYSFUN	このスカラー関数は、*argument1* の左端の *argument2* バイトからなるストリングを戻します。	
	VARCHAR(4000), INTEGER		VARCHAR(4000)
	CLOB(1M), INTEGER		CLOB(1M)
	BLOB(1M), INTEGER		BLOB(1M)
386 ページの『LENGTH』	SYSIBM	このスカラー関数は、オペランドの長さをバイト数で戻します (長さを文字数で戻す 2 バイト・ストリング型を除きます)。	
	any-builtin-type		INTEGER
387 ページの『LN』	SYSFUN	このスカラー関数は、引き数の自然対数値を戻します (LOG と同じ)。	
	DOUBLE		DOUBLE
388 ページの『LOCATE』	SYSFUN	このスカラー関数は、*argument2* の中の *argument1* の最初のオカレンスの開始位置を戻します。オプションの 3 番目の引き数を指定すると、その引き数は *argument2* の中で探索を開始する文字位置を示します。 *argument1* が *argument2* 内にない場合、値 0 が戻されます。	
	VARCHAR(4000), VARCHAR(4000)		INTEGER
	VARCHAR(4000), VARCHAR(4000), INTEGER		INTEGER
	CLOB(1M), CLOB(1M)		INTEGER
	CLOB(1M), CLOB(1M), INTEGER		INTEGER
	BLOB(1M), BLOB(1M)		INTEGER
	BLOB(1M), BLOB(1M), INTEGER		INTEGER
389 ページの『LOG』	SYSFUN	このスカラー関数は、引き数の自然対数値を戻します (LN と同じ)。	
	DOUBLE		DOUBLE
390 ページの『LOG10』	SYSFUN	このスカラー関数は、引き数の 10 を底とする対数を戻します。	
	DOUBLE		DOUBLE
391 ページの『LONG_VARCHAR』	SYSIBM	このスカラー関数は、LONG ストリングを戻します。	
	character-type		LONG VARCHAR
392 ページの『LONG_VARGRAPHIC』	SYSIBM	このスカラー関数は、ソース・タイプから LONG VARGRAPHIC にキャストします。	
	graphic-type		LONG VARGRAPHIC
394 ページの『LTRIM (SYSFUN スキーマ)』	SYSFUN	このスカラー関数は、先行ブランクを除去した引き数の文字を戻します。	
	VARCHAR(4000)		VARCHAR(4000)
	CLOB(1M)		CLOB(1M)
393 ページの『LTRIM』	SYSIBM	このスカラー関数は、先行ブランクを除去した引き数の文字を戻します。	
	CHAR		VARCHAR
	VARCHAR		VARCHAR
	GRAPHIC		VARGRAPHIC
	VARGRAPHIC		VARGRAPHIC
284 ページの『MAX』	SYSIBM	この集約関数は、一連の値の中の最大値を戻します。	
	any-builtin-type [5]		*入力タイプと同じ*

表 16. サポートされている関数 (続き)

関数名	スキーマ 入力パラメーター	説明 戻り値のタイプ
395 ページの『MICROSECOND』	SYSIBM	このスカラー関数は、値のマイクロ秒 (時間単位) の部分を戻します。
	VARCHAR	INTEGER
	TIMESTAMP	INTEGER
	DECIMAL	INTEGER
396 ページの『MIDNIGHT_SECONDS』	SYSFUN	このスカラー関数は、午前 0 時から *argument* に指定された時刻値までの秒数を表す 0 から 86 400 の範囲の整数値を戻します。
	VARCHAR(26)	INTEGER
	TIME	INTEGER
	TIMESTAMP	INTEGER
286 ページの『MIN』	SYSIBM	この集約関数は、一連の値の中の最小値を戻します。
	any-builtin-type [5]	*入力タイプと同じ*
397 ページの『MINUTE』	SYSIBM	このスカラー関数は、値の分の部分を戻します。
	VARCHAR	INTEGER
	TIME	INTEGER
	TIMESTAMP	INTEGER
	DECIMAL	INTEGER
398 ページの『MOD』	SYSFUN	このスカラー関数は、>*argument1* を *argument2* で除算した剰余 (モジュラス) を戻します。結果は、*argument1* が負の場合にのみ負になります。
	SMALLINT, SMALLINT	SMALLINT
	INTEGER, INTEGER	INTEGER
	BIGINT, BIGINT	BIGINT
399 ページの『MONTH』	SYSIBM	このスカラー関数は、値の月の部分を戻します。
	VARCHAR	INTEGER
	DATE	INTEGER
	TIMESTAMP	INTEGER
	DECIMAL	INTEGER
400 ページの『MONTHNAME』	SYSFUN	このスカラー関数は、データベース開始時のロケールに基づいて、日付またはタイム・スタンプである引き数の月の部分の月名 (January など) から成る大文字小文字混合文字ストリングを戻します。
	VARCHAR(26)	VARCHAR(100)
	DATE	VARCHAR(100)
	TIMESTAMP	VARCHAR(100)
MQPUBLISH	DB2MQ, DB2MQ1C	このスカラー関数 (SQL 管理ルーチン) は、MQSeries ロケーションに対してデータを公開します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
MQREAD	DB2MQ, DB2MQ1C	このスカラー関数 (SQL 管理ルーチン) は、MQSeries ロケーションからメッセージを戻します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
MQREADALL	DB2MQ, DB2MQ1C	この表関数 (SQL 管理ルーチン) は、 MQSeries ロケーションからメッセージとメッセージ・メタデータを示した表を戻します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
MQREADALLCLOB	DB2MQ	この表関数 (SQL 管理ルーチン) は、指定された MQSeries ロケーションからメッセージとメッセージ・メタデータを載せた表を戻します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	

表 16. サポートされている関数 (続き)

関数名	スキーマ	説明
	入力パラメーター	戻り値のタイプ
MQREADCLOB	DB2MQ	このスカラー関数 (SQL 管理ルーチン) は、指定された MQSeries ロケーションからメッセージを戻します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
MQRECEIVE	DB2MQ, DB2MQ1C	このスカラー関数 (SQL 管理ルーチン) は、MQSeries ロケーションからメッセージを戻し、それに関連したキューからメッセージを除去します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
MQRECEIVEALL	DB2MQ, DB2MQ1C	この表関数 (SQL 管理ルーチン) は、MQSeries ロケーションからメッセージとメッセージ・メタデータの入った表を戻し、それに関連したキューからメッセージを除去します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
MQRECEIVEALLCLOB	DB2MQ	この表関数 (SQL 管理ルーチン) は、指定された MQSeries ロケーションからメッセージとメッセージ・メタデータを載せた表を戻します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
MQRECEIVECLOB	DB2MQ	このスカラー関数 (SQL 管理ルーチン) は、指定された MQSeries ロケーションからメッセージを戻します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
MQSEND	DB2MQ, DB2MQ1C	このスカラー関数 (SQL 管理ルーチン) は、MQSeries ロケーションにデータを送信します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
MQSUBSCRIBE	DB2MQ, DB2MQ1C	このスカラー関数 (SQL 管理ルーチン) は、特定のトピックに関して公開された MQSeries メッセージにサブスクライブします。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
MQUNSUBSCRIBE	DB2MQ, DB2MQ1C	このスカラー関数 (SQL 管理ルーチン) は、特定のトピックに関して公開された MQSeries メッセージからアンサブスクライブします。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
401 ページの『MULTIPLY_ALT』	SYSIBM	このスカラー関数は、2 つの引き数の積を 10 進数で返します。この関数は、引き数の精度の合計が 31 より大きい場合に便利です。
	exact-numeric-type, *exact-numeric-type*	DECIMAL
403 ページの『NULLIF』 [3]	SYSIBM	このスカラー関数は、引き数が等しい場合は NULL 値、等しくない場合は最初の引き数を戻します。
	any-type [5], *any-comparable-type* [5]	*any-type*
404 ページの『POSSTR』	SYSIBM	このスカラー関数は、あるストリングが他のストリングに収容されている位置を戻します。
	string-type, *compatible-string-type*	INTEGER
406 ページの『POWER』	SYSFUN	このスカラー関数は、*argument1* の *argument2* 乗の値を戻します。
	INTEGER, INTEGER	INTEGER
	BIGINT, BIGINT	BIGINT
	DOUBLE, INTEGER	DOUBLE
	DOUBLE, DOUBLE	DOUBLE
PUT_ROUTINE_SAR	SYSFUN	このプロシージャー (SQL 管理ルーチン) は、データベース・サーバーで SQL ルーチンを作成したり定義したりするのに必要な情報を渡します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	

表 16. サポートされている関数 (続き)

関数名	スキーマ	説明	
	入力パラメーター		戻り値のタイプ
407 ページの『QUARTER』	SYSFUN	このスカラー関数は、引き数に指定され日付が属する 4 半期を示す 1 から 4 の範囲の整数値を戻します。	
	VARCHAR(26)		INTEGER
	DATE		INTEGER
	TIMESTAMP		INTEGER
408 ページの『RADIANS』	SYSFUN	このスカラー関数は、度単位の引き数をラジアン単位の角度に変換して戻します。	
	DOUBLE		DOUBLE
409 ページの『RAISE_ERROR』 [3]	SYSIBM	このスカラー関数は、SQLCA にエラーを発生させます。戻される sqlstate は *argument1* で指定します。 2 番目の引き数には、戻されるテキストが入れられます。	
	VARCHAR, VARCHAR		*any-type* [6]
411 ページの『RAND』	SYSFUN	このスカラー関数は、引き数をオプションのシード値として使用して、 0 から 1 のランダムな浮動小数点数値を戻します。	
	引き数を必要としない		DOUBLE
	INTEGER		DOUBLE
412 ページの『REAL』	SYSIBM	このスカラー関数は、数値の単精度浮動小数点表記を戻します。	
	numeric-type		REAL
REBIND_ROUTINE_PACKAGE	SYSPROC	このプロシージャー (SQL 管理ルーチン) は、SQL プロシージャーに関連したパッケージを再バインドします。	
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。		
413 ページの『REC2XML』	SYSIBM	このスカラー関数は、XML タグでフォーマット設定されて列名と列データを収めたストリングを戻します。	
	DECIMAL, VARCHAR, VARCHAR, *any-type*[7]		VARCHAR
287 ページの『回帰関数』	SYSIBM	REGR_AVGX 集約関数は、診断統計の計算に使用される数量を戻します。	
	numeric-type, *numeric-type*		DOUBLE
287 ページの『回帰関数』	SYSIBM	REGR_AVGY 集約関数は、診断統計の計算に使用される数量を戻します。	
	numeric-type, *numeric-type*		DOUBLE
287 ページの『回帰関数』	SYSIBM	REGR_COUNT 集約関数は、回帰直線をフィッティングするために使用される NULL ではない数字のペアの数を戻します。	
	numeric-type, *numeric-type*		INTEGER
287 ページの『回帰関数』	SYSIBM	REGR_INTERCEPT または REGR_ICPT 集約関数は、回帰直線の y 切片を戻します。	
	numeric-type, *numeric-type*		DOUBLE
287 ページの『回帰関数』	SYSIBM	REGR_R2 集約関数は、回帰を決定する係数を戻します。	
	numeric-type, *numeric-type*		DOUBLE
287 ページの『回帰関数』	SYSIBM	REGR_SLOPE 集約関数は、直線の傾きを戻します。	
	numeric-type, *numeric-type*		DOUBLE
287 ページの『回帰関数』	SYSIBM	REGR_SXX 集約関数は、診断統計の計算に使用される数量を戻します。	
	numeric-type, *numeric-type*		DOUBLE
287 ページの『回帰関数』	SYSIBM	REGR_SXY 集約関数は、診断統計の計算に使用される数量を戻します。	
	numeric-type, *numeric-type*		DOUBLE
287 ページの『回帰関数』	SYSIBM	REGR_SYY 集約関数は、診断統計の計算に使用される数量を戻します。	
	numeric-type, *numeric-type*		DOUBLE

表 16. サポートされている関数 (続き)

関数名	スキーマ / 入力パラメーター	説明 / 戻り値のタイプ
REORGCHK_IX_STATS	SYSPROC	このプロシージャー (SQL 管理ルーチン) は、索引統計を調べて、再編成の必要があるかどうかを判別します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
REORGCHK_TB_STATS	SYSPROC	このプロシージャー (SQL 管理ルーチン) は、表統計を調べて、再編成の必要があるかどうかを判別します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
418 ページの『REPEAT』	SYSFUN	このスカラー関数は、*argument2* 回繰り返された *argument1* から成る文字ストリングを戻します。
	VARCHAR(4000), INTEGER	VARCHAR(4000)
	CLOB(1M), INTEGER	CLOB(1M)
	BLOB(1M), INTEGER	BLOB(1M)
419 ページの『REPLACE』	SYSFUN	このスカラー関数は、 *argument1* 内に存在する *argument2* をすべて *argument3* に置き換えます。
	VARCHAR(4000), VARCHAR(4000), VARCHAR(4000)	VARCHAR(4000)
	CLOB(1M), CLOB(1M), CLOB(1M)	CLOB(1M)
	BLOB(1M), BLOB(1M), BLOB(1M)	BLOB(1M)
420 ページの『RIGHT』	SYSFUN	このスカラー関数は、*argument1* の右端の *argument2* バイトからなるストリングを戻します。
	VARCHAR(4000), INTEGER	VARCHAR(4000)
	CLOB(1M), INTEGER	CLOB(1M)
	BLOB(1M), INTEGER	BLOB(1M)
421 ページの『ROUND』	SYSIBM	このスカラー関数は、最初の引き数を小数点以下 *argument2* 桁目で丸めて戻します。 *argument2* が負の場合、 *argument1* は小数点の左側の *argument2* 桁数の絶対値に丸められます。
	INTEGER, INTEGER	INTEGER
	BIGINT, INTEGER	BIGINT
	DOUBLE, INTEGER	DOUBLE
424 ページの『RTRIM (SYSFUN スキーマ)』	SYSFUN	このスカラー関数は、末尾ブランクを除去した引き数の文字を戻します。
	VARCHAR(4000)	VARCHAR(4000)
	CLOB(1M)	CLOB(1M)
423 ページの『RTRIM』	SYSIBM	このスカラー関数は、末尾ブランクを除去した引き数の文字を戻します。
	CHAR	VARCHAR
	VARCHAR	VARCHAR
	GRAPHIC	VARGRAPHIC
	VARGRAPHIC	VARGRAPHIC
425 ページの『SECOND』	SYSIBM	このスカラー関数は、値の秒 (時間単位) の部分を戻します。
	VARCHAR	INTEGER
	TIME	INTEGER
	TIMESTAMP	INTEGER
	DECIMAL	INTEGER
SERVER	DB2EAS	このプロシージャー (SQL 管理ルーチン) は、DB2 用のアプリケーション・サーバーを開始または停止します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	

表 16. サポートされている関数 (続き)

関数名	スキーマ	説明	
	入力パラメーター		戻り値のタイプ
SET_ROUTINE_OPTS	SYSPROC	このプロシージャー (SQL 管理ルーチン) は、現行セッションでの SQL プロシージャーの作成に使われる予定のオプションを設定します。	
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。		
426 ページの『SIGN』	SYSFUN	このスカラー関数は、引き数の符号の標識を戻します。引き数が負の場合は、-1 が戻されます。引き数がゼロの場合は、0 が戻されます。引き数が正の場合には、1 が戻されます。	
	SMALLINT		SMALLINT
	INTEGER		INTEGER
	BIGINT		BIGINT
	DOUBLE		DOUBLE
427 ページの『SIN』	SYSIBM	このスカラー関数は、引き数のサイン (正弦) を戻します。引き数はラジアン単位の角度です。	
	DOUBLE		DOUBLE
428 ページの『SINH』	SYSIBM	このスカラー関数は、引き数の双曲線サイン (正弦) を戻します。引き数はラジアン単位の角度です。	
	DOUBLE		DOUBLE
429 ページの『SMALLINT』	SYSIBM	このスカラー関数は、数値の短整数表記を戻します。	
	numeric-type		SMALLINT
	VARCHAR		SMALLINT
SNAPSHOT_AGENT	SYSPROC	この表関数 (SQL 管理ルーチン) は、アプリケーション・スナップショットからエージェントに関する情報を載せた表を戻します。	
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。		
SNAPSHOT_APPL	SYSPROC	この表関数 (SQL 管理ルーチン) は、アプリケーション・スナップショットから一般情報の表を戻します。	
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。		
SNAPSHOT_APPL_INFO	SYSPROC	この表関数 (SQL 管理ルーチン) は、アプリケーション・スナップショットから一般情報の表を戻します。	
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。		
SNAPSHOT_BP	SYSPROC	この表関数 (SQL 管理ルーチン) は、バッファー・プール・スナップショットからの情報を載せた表を戻します。	
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。		
SNAPSHOT_CONTAINER	SYSPROC	この表関数 (SQL 管理ルーチン) は、表スペース・スナップショットからのコンテナー構成情報を載せた表を戻します。	
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。		
SNAPSHOT_DATABASE	SYSPROC	この表関数 (SQL 管理ルーチン) は、データベース・スナップショットからの情報を載せた表を戻します。	
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。		
SNAPSHOT_DBM	SYSPROC	この表関数 (SQL 管理ルーチン) は、 DB2 データベース・マネージャーのスナップショットから情報を示した表を戻します。	
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。		
SNAPSHOT_DYN_SQL	SYSPROC	この表関数 (SQL 管理ルーチン) は、動的 SQL スナップショットからの情報を載せた表を戻します。	
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。		
SNAPSHOT_FCM	SYSPROC	この表関数 (SQL 管理ルーチン) は、高速コミュニケーション・マネージャー (FCM) に関するデータベース・マネージャー・レベル情報を収めた表を戻します。	
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。		

表 16. サポートされている関数 (続き)

関数名	スキーマ	説明
	入力パラメーター	戻り値のタイプ
SNAPSHOT_FCMNODE	SYSPROC	この表関数 (SQL 管理ルーチン) は、データベース・マネージャー内の高速コミュニケーション・マネージャーのスナップショットからの情報を示した表を戻します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
SNAPSHOT_FILEW	SYSPROC	このプロシージャー (SQL 管理ルーチン) は、システム・スナップショット・データを、インスタンス・ディレクトリーの tmp サブディレクトリーにあるファイルに書き込みます。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
SNAPSHOT_LOCK	SYSPROC	この表関数 (SQL 管理ルーチン) は、ロック・スナップショットからの情報の入った表を戻します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
SNAPSHOT_LOCKWAIT	SYSPROC	この表関数 (SQL 管理ルーチン) は、アプリケーション・スナップショットからのロック待機情報を載せた表を戻します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
SNAPSHOT_QUIESCERS	SYSPROC	この表関数 (SQL 管理ルーチン) は、表スペース・スナップショットからの quiescer に関する情報を載せた表を戻します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
SNAPSHOT_RANGES	SYSPROC	この表関数 (SQL 管理ルーチン) は、範囲スナップショットからの情報を載せた表を戻します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
SNAPSHOT_STATEMENT	SYSPROC	この表関数 (SQL 管理ルーチン) は、アプリケーション・スナップショットからのステートメントに関する情報を載せた表を戻します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
SNAPSHOT_SUBSECT	SYSPROC	この表関数 (SQL 管理ルーチン) は、アプリケーション・スナップショットからのサブセクションに関する情報を載せた表を戻します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
SNAPSHOT_SWITCHES	SYSPROC	この表関数 (SQL 管理ルーチン) は、データベース・スナップショットの切り替え状態に関する情報を載せた表を戻します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
SNAPSHOT_TABLE	SYSPROC	この表関数 (SQL 管理ルーチン) は、表スナップショットからのアクティビティー情報の入った表を戻します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
SNAPSHOT_TBREORG	SYSPROC	この表関数 (SQL 管理ルーチン) は、表の再編成情報を載せた表を戻します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
SNAPSHOT_TBS	SYSPROC	この表関数 (SQL 管理ルーチン) は、表スペース・スナップショットからのアクティビティー情報の入った表を戻します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
SNAPSHOT_TBS_CFG	SYSPROC	この表関数 (SQL 管理ルーチン) は、表スペース・スナップショットからの構成情報を載せた表を戻します。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
430 ページの『SOUNDEX』	SYSFUN	このスカラー関数は、引き数内の語の音を表す 4 文字コードを戻します。この結果を、他のストリングの音と比較することができます。
	VARCHAR(4000)	CHAR(4)
431 ページの『SPACE』	SYSFUN	このスカラー関数は、*argument1* 個のブランクから成る文字ストリングを戻します。
	INTEGER	VARCHAR(4000)

表 16. サポートされている関数 (続き)

関数名	スキーマ	説明	
	入力パラメーター		戻り値のタイプ
SQLCACHE_SNAPSHOT	SYSFUN	この表関数 (SQL 管理ルーチン) は、 DB2 動的 SQL ステートメント・キャッシュのスナップショットを示した表を戻します。	
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。		
432 ページの『SQRT』	SYSFUN	このスカラー関数は、引き数の平方根を戻します。	
	DOUBLE		DOUBLE
290 ページの『STDDEV』	SYSIBM	この集約関数は、一連の数値の標準偏差を戻します。	
	DOUBLE		DOUBLE
433 ページの『SUBSTR』	SYSIBM	このスカラー関数は、*argument2* から始めてストリング *argument1* のサブストリングを戻します。このサブストリングは、*argument3* 字数の長さです。*argument3* を指定しないと、ストリングの残りが字数と想定されます。	
	string-type, INTEGER		*string-type*
	string-type, INTEGER, INTEGER		*string-type*
291 ページの『SUM』	SYSIBM	この集約関数は、一連の数値の和を戻します。	
	numeric-type[4]		*max-numeric-type*[1]
SYSINSTALLOBJECTS	SYSPROC	このプロシージャー (SQL 管理ルーチン) は、特定のツールで必要なデータベース・オブジェクトを作成またはドロップします。	
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。		
SYSINSTALLROUTINES	SYSPROC	このプロシージャー (SQL 管理ルーチン) は、データベースの移行後に **db2updv8** ユーティリティーが実行されなかった場合に、 SYSPROC スキーマ内に新規のプロシージャーと関数を作成します。	
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。		
436 ページの『TABLE_NAME』	SYSIBM	このスカラー関数は、*argument1* に指定したオブジェクト名と、 *argument2* に指定したオプションのスキーマ名に基づく表またはビューの非修飾名を戻します。戻された値は、別名の解決に使用されます。	
	VARCHAR		VARCHAR(128)
	VARCHAR, VARCHAR		VARCHAR(128)
437 ページの『TABLE_SCHEMA』	SYSIBM	このスカラー関数は、*argument1* 内のオブジェクト名と、 *argument2* 内のオプションのスキーマ名で指示された 2 つの部分からなる表名またはビュー名のスキーマ名部分を戻します。戻された値は、別名の解決に使用されます。	
	VARCHAR		VARCHAR(128)
	VARCHAR, VARCHAR		VARCHAR(128)
439 ページの『TAN』	SYSIBM	このスカラー関数は、引き数のタンジェント (正接) を戻します。引き数はラジアン単位の角度です。	
	DOUBLE		DOUBLE
440 ページの『TANH』	SYSIBM	このスカラー関数は、引き数の双曲線タンジェント (正接) を戻します。引き数はラジアン単位の角度です。	
	DOUBLE		DOUBLE
441 ページの『TIME』	SYSIBM	このスカラー関数は、値から時刻を戻します。	
	TIME		TIME
	TIMESTAMP		TIME
	VARCHAR		TIME

表 16. サポートされている関数 (続き)

関数名	スキーマ / 入力パラメーター	説明 / 戻り値のタイプ
442 ページの『TIMESTAMP』	SYSIBM	このスカラー関数は、1 つの値または値のペアからタイム・スタンプを戻します。
	TIMESTAMP	TIMESTAMP
	VARCHAR	TIMESTAMP
	VARCHAR, VARCHAR	TIMESTAMP
	VARCHAR, TIME	TIMESTAMP
	DATE, VARCHAR	TIMESTAMP
	DATE, TIME	TIMESTAMP
444 ページの『TIMESTAMP_FORMAT』	SYSIBM	このスカラー関数は、フォーマット・テンプレート (*argument2*) を使って解釈された文字ストリング (*argument1*) からタイム・スタンプを戻します。
	VARCHAR, VARCHAR	TIMESTAMP
446 ページの『TIMESTAMP_ISO』	SYSFUN	このスカラー関数は、日付、時刻、またはタイム・スタンプの引き数に基づいてタイム・スタンプ値を戻します。引き数が日付の場合は、時間エレメントのすべてにゼロが入れられます。引き数が時刻の場合、日付エレメントには CURRENT DATE の値、時刻の小数エレメントにはゼロが入れられます。
	DATE	TIMESTAMP
	TIME	TIMESTAMP
	TIMESTAMP	TIMESTAMP
	VARCHAR(26)	TIMESTAMP
447 ページの『TIMESTAMPDIFF』	SYSFUN	このスカラー関数は、2 つのタイム・スタンプの差に基づいて、タイプ *argument1* の推定インターバル数を戻します。 2 番目の引き数は、2 つのタイム・スタンプ・タイプの減算を行い、その結果を CHAR に変換した結果です。有効なインターバル・タイプは、以下のとおりです。 **1** 秒の小数部 **2** 秒 **4** 分 **8** 時間 **16** 日 **32** 週 **64** 月 **128** 四半期 **256** 年
	INTEGER, CHAR(22)	INTEGER
449 ページの『TO_CHAR』	SYSIBM	このスカラー関数は、タイム・スタンプの文字表記を戻します。
	VARCHAR_FORMAT と同じ。	VARCHAR_FORMAT と同じ。
450 ページの『TO_DATE』	SYSIBM	このスカラー関数は、文字ストリングからタイム・スタンプを戻します。
	TIMESTAMP_FORMAT と同じ。	TIMESTAMP_FORMAT と同じ。

表 16. サポートされている関数 (続き)

関数名	スキーマ / 入力パラメーター	説明 / 戻り値のタイプ
451 ページの『TRANSLATE』	SYSIBM	このスカラー関数は、1 つ以上の文字を他の文字に変換したストリングを戻します。
	CHAR	CHAR
	VARCHAR	VARCHAR
	CHAR, VARCHAR, VARCHAR	CHAR
	VARCHAR, VARCHAR, VARCHAR	VARCHAR
	CHAR, VARCHAR, VARCHAR, VARCHAR	CHAR
	VARCHAR, VARCHAR, VARCHAR, VARCHAR	VARCHAR
	GRAPHIC, VARGRAPHIC, VARGRAPHIC	GRAPHIC
	VARGRAPHIC, VARGRAPHIC, VARGRAPHIC	VARGRAPHIC
	GRAPHIC, VARGRAPHIC, VARGRAPHIC, VARGRAPHIC	GRAPHIC
	VARGRAPHIC, VARGRAPHIC, VARGRAPHIC, VARGRAPHIC	VARGRAPHIC
454 ページの『TRUNCATE または TRUNC』	SYSIBM	このスカラー関数は、*argument1* を小数点以下 *argument2* 桁目で丸めて戻します。 *argument2* が負の場合、 *argument1* は、小数点の左側の *argument2* 桁数の絶対値に切り捨てられます。
	INTEGER, INTEGER	INTEGER
	BIGINT, INTEGER	BIGINT
	DOUBLE, INTEGER	DOUBLE
455 ページの『TYPE_ID』 [3]	SYSIBM	このスカラー関数は、引き数の動的データ型の内部データ型 ID を戻します。この関数の結果はデータベース間で移動できないことに注意してください。
	any-structured-type	INTEGER
456 ページの『TYPE_NAME』 [3]	SYSIBM	このスカラー関数は、引き数の動的データ型の非修飾名を戻します。
	any-structured-type	VARCHAR(18)
457 ページの『TYPE_SCHEMA』 [3]	SYSIBM	このスカラー関数は、動的データ型の引き数のスキーマ名を戻します。
	any-structured-type	VARCHAR(128)
458 ページの『UCASE または UPPER』	SYSFUN	このスカラー関数は、すべての文字が大文字に変換されたストリングを戻します。
	VARCHAR	VARCHAR
458 ページの『UCASE または UPPER』	SYSIBM	このスカラー関数は、すべての文字が大文字に変換されたストリングを戻します。
	CHAR	CHAR
	VARCHAR	VARCHAR
UNINSTALLAPP	DB2EAS	このプロシージャー (SQL 管理ルーチン) は、 DB2 用のアプリケーション・サーバーで指定のアプリケーションをアンインストールします。
	この SQL 管理ルーチンの総合解説は、DB2 インフォメーション・センターに掲載されています。	
459 ページの『VALUE』 [3]	SYSIBM	このスカラー関数は、COALESCE と同じです。
460 ページの『VARCHAR』	SYSIBM	このスカラー関数は、最初の引き数の VARCHAR 表記を戻します。 2 番目の引き数を指定した場合、その値は結果の長さを指定します。
	character-type	VARCHAR
	character-type, INTEGER	VARCHAR
	datetime-type	VARCHAR
462 ページの『VARCHAR_FORMAT』	SYSIBM	このスカラー関数は、フォーマット・テンプレート (*argument2*) どおりにフォーマット設定されたタイム・スタンプ (*argument1*) の文字表現を戻します。
	TIMESTAMP, VARCHAR	VARCHAR
	VARCHAR, VARCHAR	VARCHAR

表 16. サポートされている関数 (続き)

関数名	スキーマ	説明
	入力パラメーター	**戻り値のタイプ**
464 ページの『VARGRAPHIC』	SYSIBM	このスカラー関数は、最初の引き数の VARGRAPHIC 表記を戻します。 2 番目の引き数を指定した場合、その値は結果の長さを指定します。
	graphic-type	VARGRAPHIC
	graphic-type, INTEGER	VARGRAPHIC
	VARCHAR	VARGRAPHIC
292 ページの『VARIANCE』	SYSIBM	この集約関数は、一連の数値の差異を戻します。
	DOUBLE	DOUBLE
466 ページの『WEEK』	SYSFUN	このスカラー関数は、引き数の年間通算週番号を、1 から 54 の範囲の整数値で戻します。
	VARCHAR(26)	INTEGER
	DATE	INTEGER
	TIMESTAMP	INTEGER
467 ページの『WEEK_ISO』	SYSFUN	このスカラー関数は、引き数の年間通算週番号を、1 から 53 の範囲の整数値で戻します。週の最初の日は月曜日です。第 1 週は、年の第 1 週目で木曜日が入っています。
	VARCHAR(26)	INTEGER
	DATE	INTEGER
	TIMESTAMP	INTEGER
468 ページの『YEAR』	SYSIBM	このスカラー関数は、値の年の部分を戻します。
	VARCHAR	INTEGER
	DATE	INTEGER
	TIMESTAMP	INTEGER
	DECIMAL	INTEGER
“+”	SYSIBM	2 つの数値オペランドを加算します。
	numeric-type, *numeric-type*	*max-numeric-type*
“+”	SYSIBM	単項加算演算子。
	numeric-type	*numeric-type*
“+”	SYSIBM	日付/時刻加算演算子
	DATE, DECIMAL(8,0)	DATE
	TIME, DECIMAL(6,0)	TIME
	TIMESTAMP, DECIMAL(20,6)	TIMESTAMP
	DECIMAL(8,0), DATE	DATE
	DECIMAL(6,0), TIME	TIME
	DECIMAL(20,6), TIMESTAMP	TIMESTAMP
	datetime-type, DOUBLE, *labeled-duration-code*	*datetime-type*
“-”	SYSIBM	2 つの数値オペランドを減算します。
	numeric-type, *numeric-type*	*max-numeric-type*
“-”	SYSIBM	単項減算演算子。
	numeric-type	*numeric-type* [1]

表 16. サポートされている関数 (続き)

関数名	スキーマ	説明
	入力パラメーター	**戻り値のタイプ**
“-”	SYSIBM	日付/時刻減算演算子。
	DATE, DATE	DECIMAL(8,0)
	TIME, TIME	DECIMAL(6,0)
	TIMESTAMP, TIMESTAMP	DECIMAL (20,6)
	DATE, VARCHAR	DECIMAL(8,0)
	TIME, VARCHAR	DECIMAL(6,0)
	TIMESTAMP, VARCHAR	DECIMAL (20,6)
	VARCHAR, DATE	DECIMAL(8,0)
	VARCHAR, TIME	DECIMAL(6,0)
	VARCHAR, TIMESTAMP	DECIMAL (20,6)
	DATE, DECIMAL(8,0)	DATE
	TIME, DECIMAL(6,0)	TIME
	TIMESTAMP, DECIMAL(20,6)	TIMESTAMP
	datetime-type, DOUBLE, *labeled-duration-code*	*datetime-type*
“*”	SYSIBM	2 つの数値オペランドを乗算します。
	numeric-type, *numeric-type*	*max-numeric-type*
“/”	SYSIBM	2 つの数値オペランドを除算します。
	numeric-type, *numeric-type*	*max-numeric-type*
“||”	SYSIBM	CONCAT と同じ。

注

- 長さで修飾されていないストリング・データ型への参照は、最大長のデータ型をサポートするものと見なされます。
- 精度と位取りを指定していない DECIMAL データ型への参照は、サポートされているすべての精度と位取りが可能であると見なされます。

表 16. サポートされている関数 (続き)

関数名	スキーマ / 入力パラメーター	説明 / 戻り値のタイプ

凡例

any-builtin-type 特殊タイプ以外の任意のデータ型。

any-type データベースに定義されている任意の型。

any-structured-type
データベースに定義されているユーザー定義の構造化タイプ。

any-comparable-type
110 ページの『割り当てと比較』 の定義どおりに、他の引き数タイプと比較可能なタイプ。

any-union-compatible-type
126 ページの『結果データ型の規則』 の定義どおりに、他の引き数タイプと互換性のあるタイプ。

character-type 文字ストリング・タイプ CHAR、VARCHAR、LONG VARCHAR、CLOB のいずれか。

compatible-string-type
他の引き数と同じグループのストリング・タイプ (たとえば、引き数の一方が *character-type* であれば、もう一方の引き数も *character-type* でなければなりません)。

datetime-type 日付/時刻タイプ DATE、TIME、TIMESTAMP のいずれか。

exact-numeric-type
厳密な数タイプ SMALLINT、INTEGER、BIGINT、DECIMAL のいずれか。

graphic-type 2 バイト文字ストリング・タイプ GRAPHIC、VARGRAPHIC、LONG VARGRAPHIC、 DBCLOB のいずれか。

labeled-duration-code
これは、タイプとしては SMALLINT です。関数が、プラスまたはマイナス演算子の挿入形式を使用して呼び出される場合は、『式』で定義されているラベル付き期間を使用することができます。プラスまたはマイナスの演算子文字を名前として使用しないソース関数の場合、関数を呼び出すときに labeled-duration-code 引き数に次の値を使用する必要があります。

1 YEAR または YEARS

2 MONTH または MONTHS

3 DAY または DAYS

4 HOUR または HOURS

5 MINUTE または MINUTES

6 SECOND または SECONDS

7 MICROSECOND または MICROSECONDS

LOB-type ラージ・オブジェクト・タイプ BLOB、CLOB、DBCLOB のいずれか。

max-numeric-type
引き数の最大数値タイプ。最大値は右端の *numeric-type* と定義されます。

max-string-type 引き数の最大ストリング・タイプ。最大値は右端の *character-type* または *graphic-type* と定義されます。引き数が BLOB の場合、*max-string-type* は BLOB です。

numeric-type 数値タイプ SMALLINT、INTEGER、BIGINT、DECIMAL、REAL、DOUBLE のいずれか。

string-type *character-type*、 *graphic-type*、または BLOB のいずれか。

表 16. サポートされている関数 (続き)

関数名	スキーマ / 入力パラメーター	説明 / 戻り値のタイプ

表の脚注

1 入力パラメーターが SMALLINT の場合、結果タイプは INTEGER です。入力パラメーターが REAL の場合、結果タイプは DOUBLE です。

2 使用できる keyword (キーワード) は、ISO、USA、EUR、JIS、および LOCAL です。この関数シグニチャーは、ソース関数からの派生関数としてはサポートされていません。

3 この関数は、ソース関数としては使用できません。

4 第 1 パラメーターの前にキーワード ALL または DISTINCT を使用できます。 DISTINCT を指定した場合、ユーザー定義構造化タイプ、 LONG ストリング・タイプまたは DATALINK タイプは使用できません。

5 ユーザー定義構造化タイプ、 LONG ストリング・タイプまたは DATALINK タイプは使用できません。

6 RAISE_ERROR によって戻されるタイプは、それを使うコンテキストによって異なります。特定のタイプにキャストしない場合、 RAISE_ERROR は、CASE 式内でのその呼び出しに適したタイプを戻します。

7 *graphic-type*、*LOB-type*、長ストリング・タイプ、および DATALINK タイプは使用できません。

表 17. 集約関数

関数	説明
275 ページの『AVG』	一連の数値の平均を戻します。
277 ページの『CORRELATION』	一連の数値の相関係数を戻します。
278 ページの『COUNT』	一連の行または値の中の行数または値の数を戻します。
279 ページの『COUNT_BIG』	一連の行または値の中の行数または値の数を戻します。その結果は整数の最大値より大きい場合があります。
281 ページの『COVARIANCE』	一連の数値ペアの共分散を戻します。
282 ページの『GROUPING』	グループ化集合によって生成された小計行を示すために、グループ化集合およびスーパー・グループで使用されます。返される値は 0 または 1 です。 1 の値は、戻された行の引き数の値は NULL 値であり、行がグループ化集合用に生成されたことを意味します。生成されたこの行は、グループ化集合の小計を示します。
284 ページの『MAX』	一連の値の最大値を戻します。
286 ページの『MIN』	一連の値の最小値を戻します。
287 ページの『回帰関数』	REGR_AVGX 集約関数は、診断統計の計算に使用される数量を戻します。
287 ページの『回帰関数』	REGR_AVGY 集約関数は、診断統計の計算に使用される数量を戻します。
287 ページの『回帰関数』	REGR_COUNT 集約関数は、回帰直線をフィッティングするために使用される NULL ではない数字のペアの数を戻します。
287 ページの『回帰関数』	REGR_INTERCEPT または REGR_ICPT 集約関数は、回帰直線の y 切片を戻します。
287 ページの『回帰関数』	REGR_R2 集約関数は、回帰を決定する係数を戻します。
287 ページの『回帰関数』	REGR_SLOPE 集約関数は、直線の傾きを戻します。

表 17. 集約関数 (続き)

関数	説明
287 ページの『回帰関数』	REGR_SXX 集約関数は、診断統計の計算に使用される数量を戻します。
287 ページの『回帰関数』	REGR_SXY 集約関数は、診断統計の計算に使用される数量を戻します。
287 ページの『回帰関数』	REGR_SYY 集約関数は、診断統計の計算に使用される数量を戻します。
290 ページの『STDDEV』	一連の数値の標準偏差を戻します。
291 ページの『SUM』	一連の数値の和を戻します。
292 ページの『VARIANCE』	一連の数値の差異を戻します。

表 18. CAST スカラー関数

関数	説明
301 ページの『BIGINT』	数値または文字ストリングを 64 ビットで表した整数を、整数定数の形で戻します。
303 ページの『BLOB』	任意の型のストリングの BLOB 表記を戻します。
305 ページの『CHAR』	値の文字表現を戻します。
310 ページの『CLOB』	値の CLOB 表現を戻します。
316 ページの『DATE』	値から日付を戻します。
323 ページの『DBCLOB』	ストリングの DBCLOB 表現を戻します。
326 ページの『DECIMAL』	数値の DECIMAL 表現を戻します。
354 ページの『DOUBLE』	数値の倍精度表現を戻します。
361 ページの『FLOAT』	数値の浮動小数点表現を戻します。
366 ページの『GRAPHIC』	ストリングの GRAPHIC 表現を戻します。
380 ページの『INTEGER』	数値の整数表記を戻します。
391 ページの『LONG_VARCHAR』	値の LONG VARCHAR 表現を戻します。
392 ページの『LONG_VARGRAPHIC』	値の LONG VARGRAPHIC 表現を戻します。
412 ページの『REAL』	数値の実数表現を戻します。
429 ページの『SMALLINT』	数値の SMALLINT 表現を戻します。
441 ページの『TIME』	値から時刻を戻します。
442 ページの『TIMESTAMP』	値または値のペアからタイム・スタンプを戻します。
460 ページの『VARCHAR』	値の VARCHAR 表現を戻します。
464 ページの『VARGRAPHIC』	値の VARGRAPHIC 表現を戻します。

表 19. DATALINK スカラー関数

関数	説明
336 ページの『DLCOMMENT』	データ・リンク値のコメント属性を戻します。

表 19. DATALINK スカラー関数 (続き)

関数	説明
337 ページの『DLLINKTYPE』	データ・リンク値のリンク・タイプ属性を戻します。
338 ページの『DLNEWCOPY』	参照されているファイルが変更されたことを示す属性をもった DATALINK 値を戻します。
340 ページの『DLPREVIOUSCOPY』	旧バージョンのファイルをリストアする必要があることを示す属性をもった DATALINK 値を戻します。
342 ページの『DLREPLACECONTENT』	DATALINK 値を戻します。この関数は、UPDATE ステートメント内の SET 文節の右辺にあるか、または INSERT ステートメント内の VALUES 文節にあるとき、戻された値を割り当てることによって、ファイルの内容を別のファイルで置き換え、その後そのファイルへのリンクを作成します。
344 ページの『DLURLCOMPLETE』	DATALINK 値の完全 URL (アクセス・トークンを含む) を戻します。
345 ページの『DLURLCOMPLETEONLY』	URL のリンク・タイプを持つ DATALINK 値から、完全な URL (アクセス・トークンを含まない) を戻します。
346 ページの『DLURLCOMPLETEWRITE』	URL のリンク・タイプを持つ DATALINK 値から、指定されたファイルの変更に必要な完全な URL 値を戻します。
347 ページの『DLURLPATH』	データ・リンク値のパスおよびファイル名 (アクセス・トークンを含む) を戻します。
348 ページの『DLURLPATHONLY』	データ・リンク値のパスおよびファイル名 (アクセス・トークンを含まない) を戻します。
349 ページの『DLURLPATHWRITE』	特定サーバー内のファイルを変更するために必要なパスおよびファイル名を、リンク・タイプが URL の DATALINK 値から戻します。
350 ページの『DLURLSCHEME』	データ・リンク値の URL 属性からスキームを戻します。
351 ページの『DLURLSERVER』	データ・リンク値の URL 属性からサーバーを戻します。
352 ページの『DLVALUE』	データ位置付け引き数、リンク・タイプ引き数、およびオプションのコメント・ストリング引き数からデータ・リンク値を作成します。

表 20. DATETIME スカラー関数

関数	説明
317 ページの『DAY』	値の日の部分を戻します。
318 ページの『DAYNAME』	db2start が 出された時点のロケールに基づいて、引き数の日の部分の曜日名から成る大文字小文字混合文字ストリング (たとえば、 Friday) を戻します。
319 ページの『DAYOFWEEK』	値から曜日を戻します。ただし 1 は Sunday、7 は Saturday です。
320 ページの『DAYOFWEEK_ISO』	値から曜日を戻します。ただし 1 は Monday、7 は Sunday です。
321 ページの『DAYOFYEAR』	値から、年頭からの通算日数を戻します。

表 20. DATETIME スカラー関数 (続き)

関数	説明
322 ページの『DAYS』	日付の整数表記を戻します。
372 ページの『HOUR』	値の時間の部分を戻します。
382 ページの『JULIAN_DAY』	紀元前 4712 年 1 月 1 日からの経過日数を表す整数値を、引き数に指定された日付値に戻します。
395 ページの『MICROSECOND』	値のマイクロ秒の部分を戻します。
396 ページの『MIDNIGHT_SECONDS』	午前 0 時から指定した時刻値までの秒数を表す整数値を戻します。
397 ページの『MINUTE』	値の分の部分を戻します。
399 ページの『MONTH』	値の月の部分を戻します。
400 ページの『MONTHNAME』	データベース開始時のロケールに基づいて、日付またはタイム・スタンプである引き数の月の部分の月名 (January など) などから成る大文字小文字混合文字ストリングを戻します。
407 ページの『QUARTER』	日付が属する四半期を表す整数を戻します。
425 ページの『SECOND』	値の秒の部分を戻します。
444 ページの『TIMESTAMP_FORMAT』	フォーマット・テンプレート (*argument2*) を使って解釈された文字ストリング (*argument1*) からタイム・スタンプを戻します。
446 ページの『TIMESTAMP_ISO』	日付、時刻、またはタイム・スタンプの引き数に基づいてタイム・スタンプ値を戻します。引き数が日付の場合は、時間エレメントのすべてにゼロが入れられます。引き数が時刻の場合、日付エレメントには CURRENT DATE の値、時刻の小数エレメントにはゼロが入れられます。
447 ページの『TIMESTAMPDIFF』	2 つのタイム・スタンプの差に基づいて、タイプ *argument1* の推定インターバル数を戻します。 2 番目の引き数は、2 つのタイム・スタンプ・タイプの減算を行い、その結果を CHAR に変換した結果です。
449 ページの『TO_CHAR』	タイム・スタンプの文字表現を戻します。
450 ページの『TO_DATE』	文字ストリングからタイム・スタンプを戻します。
462 ページの『VARCHAR_FORMAT』	フォーマット・テンプレート (*argument2*) どおりにフォーマット設定されたタイム・スタンプ (*argument1*) の文字表現を戻します。
466 ページの『WEEK』	値から、年頭からの通算週を戻します。ただしその週は、日曜日から始まります。
467 ページの『WEEK_ISO』	値から、年頭からの通算週を戻します。ただしその週は、月曜日から始まります。
468 ページの『YEAR』	値の年の部分を戻します。

表 21. パーティション化スカラー関数

関数	説明
324 ページの『DBPARTITIONNUM』	行のデータベース・パーティション番号を戻します。引き数は表内の列名です。
368 ページの『HASHEDVALUE』	行の区分化マップ索引 (0 から 4095) を戻します。引き数は表内の列名です。

表 22. 数値スカラー関数

関数	説明
294 ページの『ABS または ABSVAL』	数値の絶対値を戻します。
295 ページの『ACOS』	数値のラジアン単位のアークコサイン (逆余弦) を戻します。
297 ページの『ASIN』	数値のラジアン単位のアークサイン (逆正弦) を戻します。
298 ページの『ATAN』	数値のラジアン単位のアークタンジェント (逆正接) を戻します。
300 ページの『ATANH』	数値のラジアン単位の双曲線アークタンジェント (逆正接) を戻します。
299 ページの『ATAN2』	x 座標および y 座標のアークタンジェント (逆正接) の角度を戻します (ラジアン単位)。
304 ページの『CEILING または CEIL』	数値よりも大きいかまたは等しい最小整数値を戻します。
313 ページの『COS』	数値のコサイン (余弦) を戻します。
314 ページの『COSH』	数値の双曲線コサイン (余弦) を戻します。
315 ページの『COT』	引き数に対するコタンジェント (余接) の値を戻します。引き数はラジアン単位の角度です。
332 ページの『DEGREES』	角度の度数を戻します。
335 ページの『DIGITS』	数値の絶対値の文字ストリング表現を戻します。
360 ページの『EXP』	引き数で指定された累乗の自然対数 (e) の底である値を戻します。
362 ページの『FLOOR』	数値よりも大きいかまたは等しい最大整数値を戻します。
387 ページの『LN』	数値の自然対数値を戻します。
389 ページの『LOG』	数値の自然対数値を戻します (LN と同じ)。
390 ページの『LOG10』	数値の常用対数 (底 10) を戻します。
398 ページの『MOD』	最初の引き数を 2 番目の引き数で割った剰余を戻します。
401 ページの『MULTIPLY_ALT』	2 つの引き数の積を 10 進数として返します。この関数は、引き数の精度の合計が 31 より大きい場合に便利です。
406 ページの『POWER』	最初の引き数を 2 番目の引き数に累乗した結果を戻します。
408 ページの『RADIANS』	度単位で表された引き数のラジアン数を戻します。
411 ページの『RAND』	乱数を戻します。
421 ページの『ROUND』	指定された小数点以下の桁数に丸めた数値を戻します。
426 ページの『SIGN』	数値の符号を戻します。
427 ページの『SIN』	数値のサイン (正弦) を戻します。
428 ページの『SINH』	数値の双曲線サイン (正弦) を戻します。
432 ページの『SQRT』	数値の平方根を戻します。
439 ページの『TAN』	数値のタンジェント (正接) を戻します。
440 ページの『TANH』	数値の双曲線タンジェント (正接) を戻します。
454 ページの『TRUNCATE または TRUNC』	指定された小数点以下の桁数に切り捨てられた数値を戻します。

表 23. ストリング・スカラー関数

関数	説明
296 ページの『ASCII』	引き数の左端の文字の ASCII コード値を整数として戻します。
309 ページの『CHR』	引き数で指定される ASCII コード値の文字を戻します。
312 ページの『CONCAT』	2 つのストリングを連結したストリングを戻します。
330 ページの『DECRYPT_BIN および DECRYPT_CHAR』	パスワード・ストリングを使用した暗号化データの暗号化解除の結果である値を返します。
330 ページの『DECRYPT_BIN および DECRYPT_CHAR』	パスワード・ストリングを使用した暗号化データの暗号化解除の結果である値を返します。
334 ページの『DIFFERENCE』	SOUNDEX 関数で判別された 2 つの引き数ストリングの語の音の差を戻します。 4 の値は、それらのストリングが同じ音であることを意味します。
356 ページの『ENCRYPT』	データ・ストリング式の暗号化の結果である値を返します。
364 ページの『GENERATE_UNIQUE』	同じ関数の他の実行とは異なるユニークなビット・データ文字ストリングを戻します。
363 ページの『GETHINT』	パスワードのヒントが検出された場合にそれを戻します。
378 ページの『INSERT』	*argument2* から始まる *argument3* バイトを *argument1* から削除し、 *argument2* から始まる *argument1* に *argument4* を挿入したストリングを戻します。
383 ページの『LCASE または LOWER』	すべての文字を小文字に変換したストリングを戻します。
385 ページの『LEFT』	ストリングから左端の文字を戻します。
388 ページの『LOCATE』	別のストリング内にある 1 つのストリングの始動位置を戻します。
393 ページの『LTRIM』	ストリング式の先頭にあるブランクを除去します。
404 ページの『POSSTR』	別のストリング内にある 1 つのストリングの始動位置を戻します。
418 ページの『REPEAT』	*argument2* 回繰り返された *argument1* から成る文字ストリングを戻します。
419 ページの『REPLACE』	*argument1* 内に存在する *argument2* をすべて *argument3* に置き換えます。
420 ページの『RIGHT』	ストリングから右端の文字を戻します。
423 ページの『RTRIM』	ストリング式の末尾にあるブランクを除去します。
430 ページの『SOUNDEX』	引き数内の語の音を示す 4 文字コードを戻します。この結果を、他のストリングの音と比較することができます。
431 ページの『SPACE』	指定数のブランクから成る文字ストリングを戻します。
433 ページの『SUBSTR』	ストリングのサブストリングを戻します。
451 ページの『TRANSLATE』	1 つまたは複数の文字を他の文字に変換したストリングを戻します。
458 ページの『UCASE または UPPER』	すべての文字を大文字に変換したストリングを戻します。

表 24. その他のスカラー関数

関数	説明
311 ページの『COALESCE』	NULL 以外の最初の引き数を戻します。
333 ページの『DEREF』	参照タイプ引き数のターゲット・タイプのインスタンスを戻します。
359 ページの『EVENT_MON_STATE』	特定のイベント・モニターの作動状態を戻します。
370 ページの『HEX』	値の 16 進数表現を戻します。
373 ページの『IDENTITY_VAL_LOCAL』	ID 列に割り当てられた最新の値を戻します。
386 ページの『LENGTH』	値の長さを戻します。
403 ページの『NULLIF』	引き数が等しい場合は NULL 値を戻し、それ以外の場合には最初の引き数の値を戻します。
409 ページの『RAISE_ERROR』	SQLCA にエラーを発生させます。戻される sqlstate は *argument1* で指定します。 2 番目の引き数には、戻されるテキストが入れられます。
413 ページの『REC2XML』	XML タグでフォーマット設定されて列名と列データを収めたストリングを戻します。
436 ページの『TABLE_NAME』	*argument1* に指定したオブジェクト名と、 *argument2* に指定したオプションのスキーマ名に基づく表またはビューの非修飾名を戻します。戻された値は、別名の解決に使用されます。
437 ページの『TABLE_SCHEMA』	*argument1* 内のオブジェクト名と、 *argument2* 内のオプションのスキーマ名で指示された 2 つの部分からなる表名またはビュー名のスキーマ名部分を戻します。戻された値は、別名の解決に使用されます。
455 ページの『TYPE_ID』	引き数の動的データ型の内部データ型 ID を戻します。この関数の結果はデータベース間で移植できません。
456 ページの『TYPE_NAME』	引き数の動的データ型の非修飾名を戻します。
457 ページの『TYPE_SCHEMA』	動的データ型の引き数のスキーマ名を戻します。
459 ページの『VALUE』	NULL 値以外の最初の引き数を戻します。

集約関数

列関数の引き数は、1 つの式から派生する一連の値の集合です。式に列を組み込むことはできますが、*スカラー全選択*、または他の列関数を組み込むことはできません (SQLSTATE 42607)。その集合の有効範囲は、グループ、または中間結果表です。

GROUP BY 文節が照会内に指定されている場合に、 FROM、WHERE、GROUP BY および HAVING 文節からの中間結果が空のセットであると、列関数は適用されません。つまり、照会の結果は空のセットとなり、 SQLCODE は +100 に設定され、SQLSTATE は '02000' に設定されます。

GROUP BY 文節が照会の中に指定されておらず、FROM、WHERE、および HAVING の文節の中間結果が空のセットの場合、列関数はその空のセットに適用されます。

たとえば、次の SELECT ステートメントの結果は、部門 D01 の社員に対して重複しない JOBCODE 値の数となります。

```
SELECT COUNT(DISTINCT JOBCODE)
  FROM CORPDATA.EMPLOYEE
  WHERE WORKDEPT = 'D01'
```

キーワード DISTINCT は、関数の引き数ではなく、関数が適用される以前に実行される演算の指定と見なされます。 DISTINCT を指定すると、重複する値は除去されます。暗黙のうちにまたは明示的に ALL を指定すると、重複する値は削除されません。

列関数で、式を使用することができます。たとえば、次のような場合です。

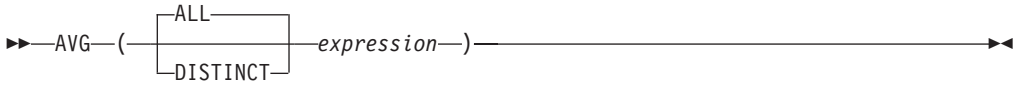
```
SELECT MAX(BONUS + 1000)
  INTO :TOP_SALESREP_BONUS
  FROM EMPLOYEE
  WHERE COMM > 5000
```

列関数は、スキーマ名 (たとえば SYSIBM.COUNT(*)) で修飾することができます。

関連資料:

• 473 ページの『SQL 照会』

AVG



スキーマは SYSIBM です。

AVG 関数は、一連の数値の平均値を戻します。

引き数の値は数値 (組み込みタイプのみ) でなければならず、その合計は、結果のデータ型の範囲内になければなりません。ただし、10 進数の結果データ型は例外です。 10 進数の結果の場合、合計は、精度 31 および引き数値と同一の位取りの 10 進数データ型でサポートされている範囲内になければなりません。結果は NULL 値の場合もあります。

結果のデータ型は、引き数値のデータ型と同じです。ただし、以下の場合を除きます。

- 引き数値が短整数 (small integer) の場合、結果は長精度整数 (large integer) になります。
- 引き数値が単精度浮動小数点の場合、結果は倍精度浮動小数点になります。

引き数値のデータ型が精度 *p* で位取りが *s* の 10 進数の場合、結果の精度は 31、位取りは 31-*p* +*s* となります。

この関数は、引き数の値から NULL 値を除いて求めた値の集合に対して適用されます。 DISTINCT を指定すると、重複する値は除去されます。

この関数が空のセットに適用されると、結果は NULL 値になります。それ以外の場合、結果はそのセットの平均値になります。

値が加算される順序は定義されていませんが、すべての中間結果は結果のデータ型の範囲内になければなりません。

結果のデータ型が整数であれば、平均値の小数部分は失われます。

例:

- PROJECT 表を使用して、部門 (DEPTNO) 'D11' におけるプロジェクトの平均スタッフ数 (PRSTAFF) を、ホスト変数 AVERAGE(decimal(5,2)) に設定します。

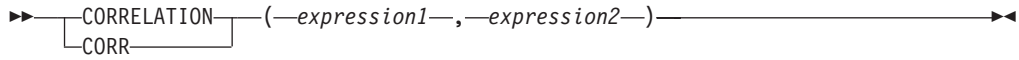
  ```
  SELECT AVG(PRSTAFF)
   INTO :AVERAGE
   FROM PROJECT
   WHERE DEPTNO = 'D11'
  ```

  サンプル表を使用してこの例を実行すると、結果として AVERAGE には、 4.25 (つまり、17/4) が設定されます。
- PROJECT 表を使用して、ホスト変数 ANY_CALC (decimal(5,2)) を、部門 (DEPTNO) 'D11' の中でのプロジェクトのスタッフ・レベル (PRSTAFF) のユニークな値についての平均値に設定します。

```
SELECT AVG(DISTINCT PRSTAFF)
 INTO :ANY_CALC
 FROM PROJECT
 WHERE DEPTNO = 'D11'
```

サンプル表を使用してこの例を実行すると、結果として ANY_CALC は 4.66 (つまり 14/3) に設定されます。

CORRELATION



スキーマは SYSIBM です。

CORRELATION 関数は、数値の組の集合に関する相関係数を戻します。

引き数には、数値を指定しなければなりません。

結果のデータ型は、倍精度の浮動小数点です。結果は NULL 値の場合もあります。値が NULL 値でない場合、結果は -1 から 1 になります。

この関数は、引き数の値から導出されたペアの集合 (*expression1*, *expression2*) から、 *expression1* または *expression2* のどちらかが NULL 値の対を除外したものに対して適用されます。

この関数が空のセットに適用された場合や、 STDDEV(*expression1*) または STDDEV(*expression2*) のどちらかがゼロに等しい場合、結果は NULL 値になります。それ以外の場合、結果はそのセット内にある値ペアの相関係数になります。結果は、以下の式と等しくなります。

```
COVARIANCE(expression1,expression2)/
(STDDEV(expression1)*
STDDEV(expression2))
```

値を集計する順序は定義されていませんが、すべての中間結果は結果のデータ型の範囲内になければなりません。

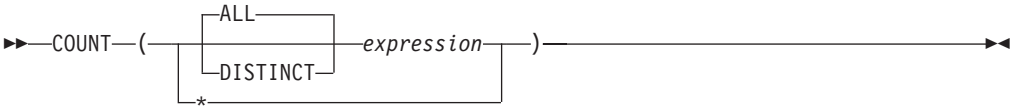
例:

- EMPLOYEE 表を使用して、ホスト変数 CORRLN (倍精度の浮動小数点数) を、部署 (WORKDEPT) 'A00' の従業員の給与と賞与の間に見られる相関に設定します。

  ```
  SELECT CORRELATION(SALARY, BONUS)
   INTO :CORRLN
    FROM EMPLOYEE
   WHERE WORKDEPT = 'A00'
  ```

  サンプル表を使用した場合、CORRLN はおよそ 9.99853953399538E-001 に設定されます。

COUNT



スキーマは SYSIBM です。

COUNT 関数は、行の集合または値の集合内にある行または値の数を戻します。

expression のデータ型は、 LONG VARCHAR、LONG VARGRAPHIC、BLOB、CLOB、DBCLOB、DATALINK、これらのタイプの特殊タイプ、または構造化タイプとすることはできません (SQLSTATE 42907)。

この関数の結果は長精度整数 (large integer) です。結果が NULL 値になることはありません。

COUNT(*) の引き数は行の集合になります。結果は、集合の行の数です。 NULL 値 (NULL 値) のみから成る行もカウントに組み入れられます。

COUNT(DISTINCT *expression*) の引き数は、値の集合です。この関数は、引き数の値から NULL 値と重複値を除いた値の集合に対して適用されます。結果は、その集合の中の異なる非 NULL 値の数です。

COUNT(*expression*) または COUNT(ALL *expression*) の引き数は、値の集合です。この関数は、引き数の値から NULL 値を除いて求めた値の集合に対して適用されます。結果は、その集合の中の NULL でない値の数です (重複値も含む)。

例:

- EMPLOYEE 表を使用して、 SEX 列の値が 'F' である行数をホスト変数 FEMALE(int) に設定します。

  ```
  SELECT COUNT(*)
   INTO :FEMALE
    FROM EMPLOYEE
   WHERE SEX = 'F'
  ```

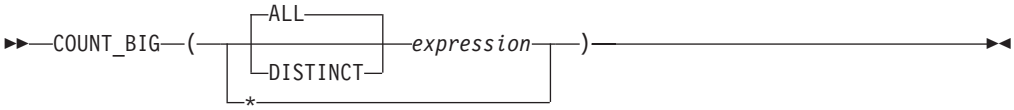
  サンプル表を使用してこの例を実行すると、結果として FEMALE に 13 が設定されます。

- EMPLOYEE 表を使用して、女性社員が少なくとも 1 人所属している部門 (WORKDEPT) の数を、ホスト変数 FEMALE_IN_DEPT (int) に設定します。

  ```
  SELECT COUNT(DISTINCT WORKDEPT)
   INTO :FEMALE_IN_DEPT
    FROM EMPLOYEE
   WHERE SEX = 'F'
  ```

  サンプル表を使用した場合、結果として FEMALE は 5 に設定されます。 (少なくとも 1 人の女性がいる部門は、A00、C01、D11、D21、および E11 です。)

COUNT_BIG



スキーマは SYSIBM です。

COUNT_BIG 関数は、一連の行または値の行の数または値の数を戻します。これは COUNT とほぼ同じですが、結果が整数の最大値より大きくなる場合がある点が異なります。

expression の結果データ型は、LONG VARCHAR、LONG VARGRAPHIC、BLOB、CLOB、DBCLOB、DATALINK、これらの型の特殊タイプ、または構造化タイプとすることはできません (SQLSTATE 42907)。

関数の結果は、精度 31 および位取り 0 の 10 進数です。結果が NULL 値になることはありません。

COUNT_BIG(*) の引き数は、行の集合です。結果は、集合の行の数です。 NULL 値 (NULL 値) のみから成る行もカウントに組み入れられます。

COUNT_BIG(DISTINCT *expression*) の引き数は、値の集合です。この関数は、引き数の値から NULL 値と重複値を除いた値の集合に対して適用されます。結果は、その集合の中の異なる非 NULL 値の数です。

COUNT_BIG(*expression*) または COUNT_BIG(ALL *expression*) の引き数は、値の集合です。この関数は、引き数の値から NULL 値を除いて求めた値の集合に対して適用されます。結果は、その集合の中の NULL でない値の数です (重複値も含む)。

例:

- COUNT の例を参照して、COUNT を COUNT_BIG に置き換えてください。結果のデータ型以外は、同じ結果になります。
- アプリケーションによっては、COUNT を使用する必要があるにもかかわらず、最大整数よりも大きい値をサポートする必要がある場合があります。これは、ソースから派生されたユーザー定義関数を使用し、 SQL パスを設定して行うことができます。以下の一連のステートメントは、 COUNT_BIG に基づいて COUNT(*) をサポートするソースからの派生関数を作成して、精度 15 の 10 進数を戻す方法を示しています。 COUNT_BIG に基づくソースからの派生関数が、ここに示されている照会などの後続のステートメントで使用されるように、 SQL パスが設定されます。

  ```
  CREATE FUNCTION RICK.COUNT() RETURNS DECIMAL(15,0)
       SOURCE SYSIBM.COUNT_BIG();
  SET CURRENT FUNCTION PATH RICK, SYSTEM PATH;
  SELECT COUNT(*) FROM EMPLOYEE;
  ```

  ソースからの派生関数は、 COUNT(*) をサポートするパラメーターを指定せずに定義されていることに注意してください。これが有効なのは、関数 COUNT を指定し、関数の使用時に関数をスキーマ名で修飾しない場合だけです。 COUNT 以外の名前を使用して COUNT(*) と同じ結果を得るためには、パラメーターを指定

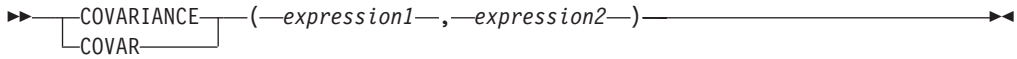
せずに関数を呼び出します。たがって、RICK.COUNT が RICK.MYCOUNT として定義されている場合、照会は以下のように書く必要があります。

```
SELECT MYCOUNT() FROM EMPLOYEE;
```

特定の列についてカウントを取る場合、ソースからの派生関数はその列のタイプを指定する必要があります。以下のステートメントによって作成されたソースからの派生関数は、 CHAR 列を引き数として取り、COUNT_BIG を使用してカウントを実行します。

```
CREATE FUNCTION RICK.COUNT(CHAR()) RETURNS DOUBLE
  SOURCE SYSIBM.COUNT_BIG(CHAR());
SELECT COUNT(DISTINCT WORKDEPT) FROM EMPLOYEE;
```

COVARIANCE



スキーマは SYSIBM です。

COVARIANCE 関数は、数値の組の集合に関する (集団) 共分散を戻します。

引き数には、数値を指定しなければなりません。

結果のデータ型は、倍精度の浮動小数点です。結果は NULL 値の場合もあります。

この関数は、引き数の値から導出された対の集合 (*expression1*, *expression2*) から、*expression1* または *expression2* のどちらかが NULL 値の対を除外したものに対して適用されます。

この関数が空のセットに適用されると、結果は NULL 値になります。それ以外の場合、結果はそのセット内の値ペアの共分散になります。結果は、次のようにして割り出されます。

1. avgexp1 を AVG(*expression1*) の結果に、 avgexp2 を AVG(*expression2*) の結果にします。
2. COVARIANCE(*expression1*, *expression2*) の結果は、 AVG(*expression1* - avgexp1) * (*expression2* - avgexp2) になります。

値を集計する順序は定義されていませんが、すべての中間結果は結果のデータ型の範囲内になければなりません。

例:

- EMPLOYEE 表を使用して、ホスト変数 COVARNCE (倍精度の浮動小数点) を、部署 (WORKDEPT) 'A00' の従業員の給与と賞与の間に見られる共分散に設定します。

```
SELECT COVARIANCE(SALARY, BONUS)
 INTO :COVARNCE
  FROM EMPLOYEE
 WHERE WORKDEPT = 'A00'
```

  サンプル表を使用した場合、COVARNCE はおよそ 1.68888888888889E+006 に設定されます。

GROUPING

スキーマは SYSIBM です。

GROUPING 関数は、グループ化集合およびスーパー・グループに関連して使用され、 GROUP BY 応答セットで戻された行が、 *expression* によって表される列を除外し、グループ化集合によって生成された行であるか否かを示す値を戻します。

引き数はどのようなタイプでも構いませんが、 GROUP BY 文節の項目でなければなりません。

この関数の結果は短整数 (small integer) です。結果は以下のいずれかの値に設定されます。

1 戻された行の *expression* の値は NULL 値であり、しかもその行はスーパー・グループによって生成されました。生成されたその行は、GROUP BY 式の小計の値を求めるのに使用することができます。

0 値は上記以外です。

例:

以下の照会を行うと、

```
SELECT SALES_DATE, SALES_PERSON,
    SUM(SALES) AS UNITS_SOLD,
    GROUPING(SALES_DATE) AS DATE_GROUP,
    GROUPING(SALES_PERSON) AS SALES_GROUP
 FROM SALES
 GROUP BY CUBE (SALES_DATE, SALES_PERSON)
 ORDER BY SALES_DATE, SALES_PERSON
```

結果は次のようになります。

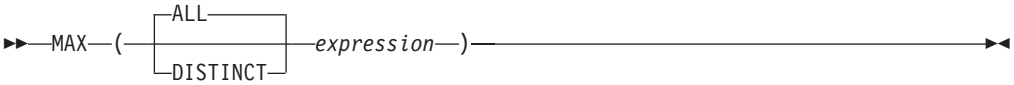
```
SALES_DATE SALES_PERSON    UNITS_SOLD  DATE_GROUP  SALES_GROUP
---------- --------------- ----------- ----------- -----------
12/31/1995 GOUNOT                    1           0           0
12/31/1995 LEE                       6           0           0
12/31/1995 LUCCHESSI                 1           0           0
12/31/1995 -                         8           0           1
03/29/1996 GOUNOT                   11           0           0
03/29/1996 LEE                      12           0           0
03/29/1996 LUCCHESSI                 4           0           0
03/29/1996 -                        27           0           1
03/30/1996 GOUNOT                   21           0           0
03/30/1996 LEE                      21           0           0
03/30/1996 LUCCHESSI                 4           0           0
03/30/1996 -                        46           0           1
03/31/1996 GOUNOT                    3           0           0
03/31/1996 LEE                      27           0           0
03/31/1996 LUCCHESSI                 1           0           0
03/31/1996 -                        31           0           1
04/01/1996 GOUNOT                   14           0           0
04/01/1996 LEE                      25           0           0
04/01/1996 LUCCHESSI                 4           0           0
04/01/1996 -                        43           0           1
-          GOUNOT                   50           1           0
-          LEE                      91           1           0
-          LUCCHESSI                14           1           0
-          -                       155           1           1
```

アプリケーションは、DATE_GROUP の値が 0 であり、 SALES_GROUP の値が 1 であることによって、 SALES_DATE の小計行を認識することができます。SALES_PERSON の小計行は、DATE_GROUP の値が 1 であり、 SALES_GROUP の値が 0 であることによって認識することができます。合計行は、DATE_GROUP と SALES_GROUP の両方の値が 1 であることによって認識することができます。

関連資料:

- 474 ページの『副選択』

MAX



スキーマは SYSIBM です。

MAX 関数は、値の集合の最大値を戻します。

引き数値は、LONG ストリングまたは DATALINK 以外の任意の組み込みタイプにすることができます。

expression の結果データ型は、LONG VARCHAR、LONG VARGRAPHIC、BLOB、CLOB、DBCLOB、DATALINK、これらのタイプの特殊タイプ、または構造化タイプとすることはできません (SQLSTATE 42907)。

結果のデータ型、長さ、およびコード・ページは、引き数値のデータ型、長さ、およびコード・ページと同じです。結果は、派生値と見なされ、NULL 値の場合もあります。

この関数は、引き数の値から NULL 値を除いて求めた値の集合に対して適用されます。

この関数が空のセットに適用されると、結果は NULL 値になります。それ以外の場合、結果はそのセットの中の最大値になります。

DISTINCT を指定しても結果に影響しないので、指定しないようにしてください。これは、他の関係システムとの互換性の目的で組み込まれています。

例:

- EMPLOYEE 表を使用して、月額の給与の最高額 (SALARY/12) の値をホスト変数 MAX_SALARY (decimal(7,2)) に設定します。

  ```
  SELECT MAX(SALARY) / 12
   INTO :MAX_SALARY
   FROM EMPLOYEE
  ```

  サンプル表を使用すると、結果として MAX_SALARY は 4395.83 に設定されます。

- PROJECT 表を使用して、照合シーケンスの最後にくるプロジェクト名 (PROJNAME) をホスト変数 LAST_PROJ (char(24)) に設定します。

  ```
  SELECT MAX(PROJNAME)
   INTO :LAST_PROJ
   FROM PROJECT
  ```

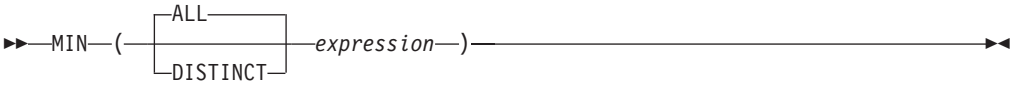
  サンプル表を使用すると、結果として LAST_PROJ は 'WELD LINE PLANNING' に設定されます。

- 前の例と同様に、プロジェクト名にホスト変数 PROJSUPP を連結した場合に照合シーケンスで最後になるプロジェクト名を、ホスト変数 LAST_PROJ (char(40)) に設定します。 PROJSUPP は '_Support' であり、データ型は char(8) です。

```
SELECT MAX(PROJNAME CONCAT PROJSUPP)
 INTO :LAST_PROJ
 FROM PROJECT
```

サンプル表を使用した場合、結果として LAST_PROJ は 'WELD LINE PLANNING_SUPPORT' に設定されます。

MIN



MIN 関数は、値の集合の最小値を戻します。

引き数値は、 LONG ストリングまたは DATALINK 以外の任意の組み込みタイプにすることができます。

expression の結果データ型は、LONG VARCHAR、LONG VARGRAPHIC、BLOB、CLOB、DBCLOB、DATALINK、これらのタイプの特殊タイプ、または構造化タイプとすることはできません (SQLSTATE 42907)。

結果のデータ型、長さ、およびコード・ページは、引き数値のデータ型、長さ、およびコード・ページと同じです。結果は、派生値と見なされ、NULL 値の場合もあります。

この関数は、引き数の値から NULL 値を除いて求めた値の集合に対して適用されます。

この関数が空のセットに適用されると、結果は NULL 値になります。それ以外の場合、結果はそのセット中の最小値になります。

DISTINCT を指定しても結果に影響しないので、指定しないようにしてください。これは、他の関係システムとの互換性の目的で組み込まれています。

例:

- EMPLOYEE 表を使用して、部門 (WORKDEPT) 'D11' の社員に対する最高と最低歩合 (COMM) の差をホスト変数 COMM_SPREAD (decimal(7,2)) に設定します。

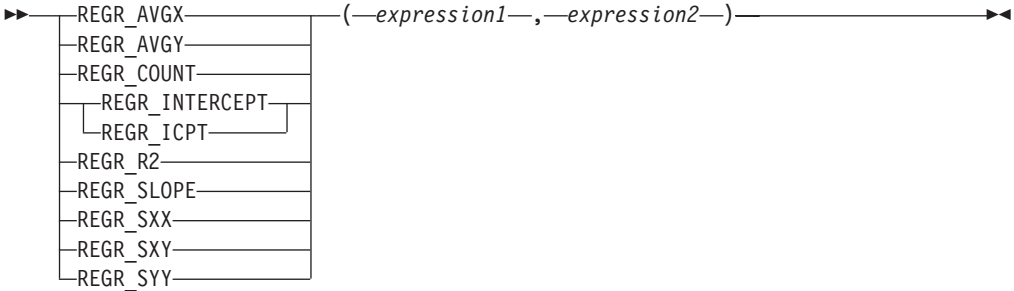
  ```
  SELECT MAX(COMM) - MIN(COMM)
   INTO :COMM_SPREAD
   FROM EMPLOYEE
   WHERE WORKDEPT = 'D11'
  ```

  サンプル表を使用すると、結果として COMM_SPREAD は 1118 (つまり 2580 - 1462) に設定されます。
- PROJECT 表を使用して、最初に完了するように予定されたプロジェクトの予定終了日 (PRENDATE) をホスト変数 (FIRST_FINISHED (char(10))) に設定します。

  ```
  SELECT MIN(PRENDATE)
   INTO :FIRST_FINISHED
   FROM PROJECT
  ```

  サンプル表を使用すると、結果として FIRST_FINISHED は '1982-09-15' に設定されます。

回帰関数



スキーマは SYSIBM です。

回帰関数は、通常の最小二乗法による回帰直線 (形式 y = a * x + b) を数値ペアの集合に当てはめることをサポートします。各ペアの最初のエレメント (*expression1*) は、従属変数の値 (つまり、″y 値″) と解釈されます。各ペアの 2 番目のエレメント (*expression2*) は、独立変数の値 (つまり、″x 値″) と解釈されます。

関数 REGR_COUNT は、回帰直線をフィッティングするために使用された NULL ではない数字のペアの数を戻します (下記参照)。

REGR_INTERCEPT (または REGR_ICPT) 関数は、回帰直線の y 切片 (前述の式では ″b″) を戻します。

REGR_R2 関数は、回帰に関する判別の係数 (″R 2 乗″ または ″適合度″ ともいう) を戻します。

REGR_SLOPE 関数は、直線の傾き (前述の式ではパラメーター ″a″) を戻します。

REGR_AVGX、REGR_AVGY、REGR_SXX、REGR_SXY、および REGR_SYY 関数は数量を戻します。そのデータを使用すれば、回帰モデルの質と統計としての有効性を評価するために必要な各種の診断統計を計算できます (下記参照)。

引き数には、数値を指定しなければなりません。

REGR_COUNT の結果のデータ型は整数です。その他の関数の場合、結果のデータ型は倍精度の浮動小数点です。結果は NULL 値の場合もあります。値が NULL 値でない場合、REGR_R2 の結果は 0 から 1 になります。 REGR_SXX と REGR_SYY の結果はどちらも正になります。

各関数は、 *expression1* または *expression2* のどちらかが NULL であるペアの除外によって、引き数の値から導出されたペアの集合 （*expression1*, *expression2*) へ適用されます。

集合が NULL ではなく、かつ VARIANCE(*expression2*) が正の場合、REGR_COUNT は NULL ではない数字のペアの数を集合に戻し、その他の関数は次のように定義された結果を戻します。

```
REGR_SLOPE(expression1,expression2) =
COVARIANCE(expression1,expression2)/VARIANCE(expression2)

REGR_INTERCEPT(expression1, expression2) =
AVG(expression1) - REGR_SLOPE(expression1, expression2) * AVG(expression2)
```

```
REGR_R2(expression1, expression2) =
POWER(CORRELATION(expression1, expression2), 2) if VARIANCE(expression1)>0
REGR_R2(expression1, expression2) = 1 if VARIANCE(expression1)=0

REGR_AVGX(expression1, expression2) = AVG(expression2)

REGR_AVGY(expression1, expression2) = AVG(expression1)

REGR_SXX(expression1, expression2) =
REGR_COUNT(expression1, expression2) * VARIANCE(expression2)

REGR_SYY(expression1, expression2) =
REGR_COUNT(expression1, expression2) * VARIANCE(expression1)

REGR_SXY(expression1, expression2) =
REGR_COUNT(expression1, expression2) * COVARIANCE(expression1, expression2)
```

集合が空ではなく、かつ VARIANCE(*expression2*) がゼロに等しい場合、回帰直線は無限の傾きになるか、未定義の状態になります。その場合、関数 REGR_SLOPE、REGR_INTERCEPT、および REGR_R2 はそれぞれ NULL 値を戻し、その他の関数は上記のように定義された戻り値を戻します。集合が空の場合は、REGR_COUNT はゼロを戻し、その他の関数は NULL 値を戻します。

値を集計する順序は定義されていませんが、すべての中間結果は結果のデータ型の範囲内になければなりません。

回帰関数はすべて、1 回のデータ・パススルーでまとめて計算されます。一般に、回帰関数を使用して回帰分析に必要な統計を計算した方が、 AVERAGE、VARIANCE、 COVARIANCE などの通常の列関数を使用して同じ計算を実行するよりも効率的です。

線形回帰分析に関係する一般的な診断統計についても、上記の関数を使用して計算できます。たとえば、次のようにします。

調整を加えた R2
: 1 - ((1 - REGR_R2) * ((REGR_COUNT - 1) / (REGR_COUNT - 2)))

標準誤差
: SQRT((REGR_SYY-
(POWER(REGR_SXY,2)/REGR_SXX))/(REGR_COUNT-2))

2 乗和の合計
: REGR_SYY

2 乗和の回帰
: POWER(REGR_SXY,2) / REGR_SXX

2 乗和の残差
: (2 乗和の合計) - (回帰 2 乗和)

傾きについての t 統計
: REGR_SLOPE * SQRT(REGR_SXX) / (標準誤差)

y 切片についての t 統計
: REGR_INTERCEPT/(標準誤差)*
SQRT((1/REGR_COUNT)+(POWER(REGR_AVGX,2)/REGR_SXX))

例:

- EMPLOYEE 表を使用して、部門 (WORKDEPT) 'A00' の従業員の賞与を表す回帰直線 (通常の最小二乗法による) を、従業員の給与の線形関数として計算しま

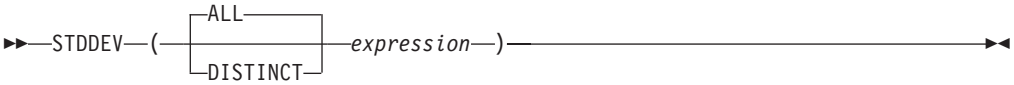
す。ホスト変数 SLOPE、ICPT、RSQR (いずれも倍精度浮動小数点数) をそれぞれ、回帰直線の判別の傾き、切片、係数に設定します。また、ホスト変数 AVGSAL を部門 'A00' の従業員の平均給与、ホスト変数 AVGBONUS を部門 'A00' の従業員の平均賞与に設定します。さらに、ホスト変数 CNT (整数) を、部門 'A00' の従業員のうち、給与データと賞与データが両方とも存在している従業員の数に設定します。その他の回帰統計はホスト変数 SXX、SYY、および SXY に格納します。

```
SELECT REGR_SLOPE(BONUS,SALARY), REGR_INTERCEPT(BONUS,SALARY),
REGR_R2(BONUS,SALARY), REGR_COUNT(BONUS,SALARY),
REGR_AVGX(BONUS,SALARY), REGR_AVGY(BONUS,SALARY),
REGR_SXX(BONUS,SALARY), REGR_SYY(BONUS,SALARY),
REGR_SXY(BONUS,SALARY)
INTO :SLOPE, :ICPT,
:RSQR, :CNT,
:AVGSAL, :AVGBONUS,
:SXX, :SYY,
:SXY
FROM EMPLOYEE
WHERE WORKDEPT = 'A00'
```

サンプル表を使用する場合、ホスト変数は以下の概数値に設定されます。

```
SLOPE:  +1.71002671916749E-002
ICPT:  +1.00871888623260E+002
RSQR:  +9.99707928128685E-001
CNT:  3
AVGSAL:  +4.28333333333333E+004
AVGBONUS:  +8.33333333333333E+002
SXX:  +2.96291666666667E+008
SYY:  +8.66666666666667E+004
SXY:  +5.06666666666667E+006
```

STDDEV



スキーマは SYSIBM です。

STDDEV 関数は、一連の数値の標準偏差を戻します。

引き数には、数値を指定しなければなりません。

結果のデータ型は、倍精度の浮動小数点です。結果は NULL 値の場合もあります。

この関数は、引き数の値から NULL 値を除いて求めた値の集合に対して適用されます。 DISTINCT を指定すると、重複する値は除去されます。

この関数が空のセットに適用されると、結果は NULL 値になります。それ以外の場合、結果はそのセットの値の標準偏差になります。

値を集計する順序は定義されていませんが、すべての中間結果は結果のデータ型の範囲内になければなりません。

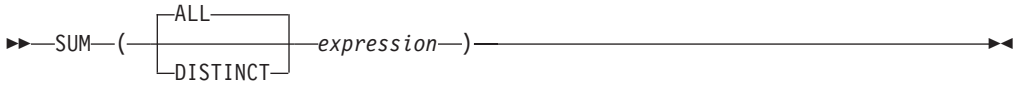
例:

- EMPLOYEE 表を使用して、ホスト変数 DEV (倍精度の浮動小数点) を部署 (WORKDEPT) 'A00' の従業員の給与の標準偏差に設定します。

```
SELECT STDDEV(SALARY)
 INTO :DEV
 FROM EMPLOYEE
 WHERE WORKDEPT = 'A00'
```

  サンプル表を使用した場合、結果として DEV はおよそ 9938.00 に設定されます。

SUM



スキーマは SYSIBM です。

SUM 関数は、一連の数値の合計値を戻します。

引き数の値は数値 (組み込みタイプのみ) でなければならず、その合計は、結果のデータ型の範囲内になければなりません。

結果のデータ型は、引き数値のデータ型と同じです。ただし、以下の点が異なります。

- 引き数値が短整数 (small integer) の場合、結果は長精度整数 (large integer) になります。
- 引き数値が単精度浮動小数点の場合、結果は倍精度浮動小数点になります。

引き数値のデータ型が 10 進数の場合、結果の精度は 31、位取りは引き数値の位取りと同じになります。結果は NULL 値の場合もあります。

この関数は、引き数の値から NULL 値を除いて求めた値の集合に対して適用されます。 DISTINCT を指定すると、重複する値も除去されます。

この関数が空のセットに適用されると、結果は NULL 値になります。それ以外の場合、結果はそのセットの値の合計になります。

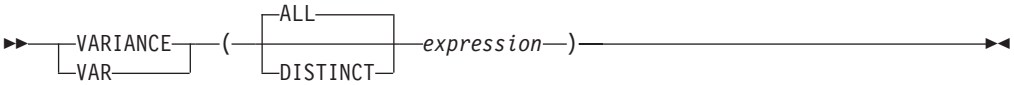
例:

- EMPLOYEE 表を使用して、事務職員 (JOB='CLERK') に支払われるボーナス (BONUS) の総額をホスト変数 JOB_BONUS (decimal(9,2)) に設定します。

```
SELECT SUM(BONUS)
 INTO :JOB_BONUS
 FROM EMPLOYEE
 WHERE JOB = 'CLERK'
```

  サンプル表を使用すると、結果として JOB_BONUS は 2800 に設定されます。

VARIANCE



スキーマは SYSIBM です。

VARIANCE 関数は、一連の数値の差異を戻します。

引き数には、数値を指定しなければなりません。

結果のデータ型は、倍精度の浮動小数点です。結果は NULL 値の場合もあります。

この関数は、引き数の値から NULL 値を除いて求めた値の集合に対して適用されます。 DISTINCT を指定すると、重複する値は除去されます。

この関数が空のセットに適用されると、結果は NULL 値になります。それ以外の場合、結果はそのセットの値の差異になります。

値が加算される順序は定義されていませんが、すべての中間結果は結果のデータ型の範囲内になければなりません。

例:

- EMPLOYEE 表を使用して、ホスト変数 VARNCE (倍精度の浮動小数点) を部署 (WORKDEPT) 'A00' の従業員の給与の差異に設定します。

```
SELECT VARIANCE(SALARY)
 INTO :VARNCE
 FROM EMPLOYEE
 WHERE WORKDEPT = 'A00'
```

  サンプル表を使用した場合、結果として VARNCE はおよそ 98763888.88 に設定されます。

スカラー関数

スカラー関数は、式を使用できる個所であればどこにでも使用することができます。ただし、式と列関数の使用に適用される制約事項は、スカラー関数の中で式または列関数を使用する場合にも適用されます。たとえば、スカラー関数の引き数を列関数にすることができるのは、スカラー関数が使用されるコンテキストで列関数の使用が許されている場合だけです。

列関数の使用法に関する制約事項がスカラー関数に適用されないのは、スカラー関数が、値の集合ではなく、単一の値を対象にするからです。

次の SELECT ステートメントの結果には、部門 D01 の従業員の数と同じ数の行が入っています。

```
SELECT EMPNO, LASTNAME, YEAR(CURRENT DATE - BRTHDATE)
  FROM EMPLOYEE
  WHERE WORKDEPT = 'D01'
```

スカラー関数は、スキーマ名 (たとえば SYSIBM.CHAR(123)) で修飾することができます。

Unicode データベースでは、文字ストリングまたは GRAPHIC ストリングを受け入れるスカラー関数はすべて、変換をサポートされている任意のストリング・タイプを受け入れます。

ABS または ABSVAL

スキーマは SYSIBM です。

この関数は、バージョン 7.1 のフィックスパック 2 から有効になりました。 ABS 関数の SYSFUN バージョン (または ABSVAL) 関数は引き続き使用可能です。

引き数の絶対値を返します。引き数は、任意の組み込み数値データ型にすることができます。

結果のデータ型と長さ属性は、引き数と同じになります。結果は NULL 値になることがあります。引き数が NULL 値である場合、その結果は NULL 値になります。引き数が SMALLINT、INTEGER、または BIGINT の最大負数値であると、その結果はオーバーフロー・エラーになります。

例:

```
ABS(-51234)
```

は値 51234 の INTEGER を返します。

ACOS

スキーマは SYSIBM です。 (ACOS 関数の SYSFUN バージョンは引き続き使用可能です。)

引き数のアークコサイン (逆余弦) の角度を戻します (ラジアン単位)。

引き数は、任意の組み込み数値データ型にすることができます。引き数は、関数での処理に必要な倍精度浮動小数点数に変換されます。

関数の結果は倍精度浮動小数点数になります。引き数が NULL になる可能性があるか、またはデータベース構成パラメーターで DFT_SQLMATHWARN が YES に設定されている場合には、結果は NULL になる可能性があります。引き数が NULL の場合、結果は NULL 値になります。

例:

ホスト変数 ACOSINE は、0.070737202 の値をもつ DECIMAL(10,9) ホスト変数であると仮定します。

```
SELECT ACOS(:ACOSINE)
  FROM SYSIBM.SYSDUMMY1
```

このステートメントは、近似値 1.49 を戻します。

ASCII

スキーマは SYSFUN です。

引き数の左端の文字の ASCII コード値を整数として戻します。

引き数は、任意の組み込み文字ストリング型にすることができます。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。 VARCHAR の場合、最大長は 4 000 バイトです。 CLOB の場合、最大長は 1 048 576 バイトです。 LONG VARCHAR は、関数による処理に必要な CLOB に変換されます。

関数の結果は常に INTEGER になります。

結果は NULL 値になることがあります。引き数が NULL 値である場合、その結果は NULL 値になります。

ASIN

スキーマは SYSIBM です。 (ASIN 関数の SYSFUN バージョンは引き続き使用可能です。)

引き数のアークサイン (逆正弦) の角度を戻します (ラジアン単位)。

引き数は、任意の組み込み数値タイプにすることができます。引き数は、関数での処理に必要な倍精度浮動小数点数に変換されます。

関数の結果は倍精度浮動小数点数になります。引き数が NULL になる可能性があるか、またはデータベース構成パラメーターで DFT_SQLMATHWARN が YES に設定されている場合には、結果は NULL になる可能性があります。引き数が NULL の場合、結果は NULL 値になります。

ATAN

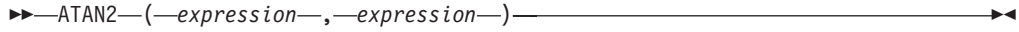
スキーマは SYSIBM です。 (ATAN 関数の SYSFUN バージョンは引き続き使用可能です。)

引き数のアークタンジェント (逆正接) の角度を戻します (ラジアン単位)。

引き数は、任意の組み込み数値データ型にすることができます。引き数は、関数での処理に必要な倍精度浮動小数点数に変換されます。

関数の結果は倍精度浮動小数点数になります。引き数が NULL になる可能性があるか、またはデータベース構成パラメーターで DFT_SQLMATHWARN が YES に設定されている場合には、結果は NULL になる可能性があります。引き数が NULL の場合、結果は NULL 値になります。

ATAN2



スキーマは SYSIBM です。 (ATAN2 関数の SYSFUN バージョンは引き続き使用可能です。)

x 座標および y 座標のアークタンジェント (逆正接) の角度を戻します (ラジアン単位)。 x 座標および y 座標はそれぞれ、最初と 2 番目の引き数によって指定されます。

最初と 2 番目の引き数は、任意の組み込み数値データ型にすることができます。いずれの引き数も、関数での処理に必要な倍精度浮動小数点数に変換されます。

関数の結果は倍精度浮動小数点数になります。引き数が NULL になる可能性があるか、またはデータベース構成パラメーターで DFT_SQLMATHWARN が YES に設定されている場合には、結果は NULL になる可能性があります。引き数が NULL の場合、結果は NULL 値になります。

ATANH

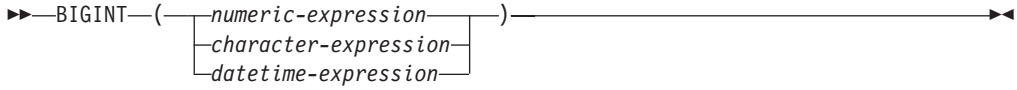
スキーマは SYSIBM です。

引き数に対する双曲線アークタンジェント (逆正接) の値を戻します。引き数はラジアン単位の角度です。

引き数は、任意の組み込み数値データ型にすることができます。引き数は、関数による処理に必要な倍精度浮動小数点数に変換されます。

関数の結果は倍精度浮動小数点数になります。引き数が NULL になる可能性があるか、またはデータベース構成パラメーターで DFT_SQLMATHWARN が YES に設定されている場合には、結果は NULL になる可能性があります。引き数が NULL の場合、結果は NULL 値になります。

BIGINT



スキーマは SYSIBM です。

BIGINT 関数は、数値、文字ストリング、日付、時刻、またはタイム・スタンプを 64 ビットで表した整数を、整数定数の形で戻します。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

numeric-expression
組み込み数値データ型の値を返す式。

引き数が*数値式* の場合、結果は、引き数を 64 ビット整数 (big integer) の列または変数に割り当てた場合と同じ数値になります。引き数の整数部分が整数の範囲内にない場合、エラーになります。引き数に小数部分がある場合は、切り捨てられます。

character-expression
文字定数の最大長以下の長さの文字ストリング値を戻す式。先行ブランクと末尾ブランクは削除されます。その結果のストリングは、 SQL 整数定数を形成するための規則に従うものでなければなりません (SQLSTATE 22018)。文字ストリングとして、LONG ストリングを使うことはできません。

引き数が文字式 の場合、結果は、対応する整数定数を 64 ビット整数 (big integer) の列または変数に割り当てた場合の数値と同じになります。

datetime-expression
次のデータ型のいずれかの式。

- DATE。結果は、日付を *yyyymmdd* で表した BIGINT 値になります。
- TIME。結果は、時間を *hhmmss* で表した BIGINT 値になります。
- TIMESTAMP。結果は、タイム・スタンプを *yyyymmddhhmmss* で表した BIGINT 値になります。タイム・スタンプのマイクロ秒の部分は、結果には入っていません。

関数の結果は 64 ビット整数です。引き数が NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引き数が NULL 値であれば、結果は NULL 値です。

例:

- ORDERS_HISTORY 表から、注文の数を数えて結果を 64 ビット整数値として戻します。

```
SELECT BIGINT (COUNT_BIG(*) )
 FROM ORDERS_HISTORY
```

- EMPLOYEE 表を使用して、アプリケーションでさらに処理を行うために、EMPNO を 64 ビット整数形式として選択します。

```
SELECT BIGINT (EMPNO) FROM EMPLOYEE
```

- RECEIVED (timestamp) 列に、 '1988-12-22-14.07.21.136421' に相当する内部値が入っていると想定します。

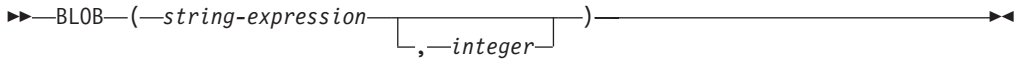
  ```
  BIGINT(RECEIVED)
  ```

  結果は、19 881 222 140 721 の値になります。
- STARTTIME (time) 列に、 '12:03:04' に相当する内部値が入っていると想定します。

  ```
  BIGINT(STARTTIME)
  ```

  結果は 120 304 の値になります。

BLOB



スキーマは SYSIBM です。

BLOB 関数は、任意のタイプのストリングの BLOB 表記を戻します。

string-expression
文字ストリング、GRAPHIC ストリング、またはバイナリー・ストリングの値をもつストリング式。

integer
結果の BLOB データ型の長さ属性を指定する整数値。 *integer* を指定しない場合、結果の長さ属性は入力の長さと同じになります。ただし、入力が GRAPHIC ストリングの場合は除きます。その場合、結果の長さ属性は入力の長さの 2 倍になります。

関数の結果は BLOB です。引き数が NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引き数が NULL 値であれば、結果は NULL 値です。

例

- TOPOGRAPHIC_MAP という名前の BLOB 列と、 MAP_NAME という名前の VARCHAR 列をもつ表を使用して、 'Pellow Island' というストリングの入ったマップ (MAP) を探し出し、実際のマップの先頭にマップ名を連結した単一のバイナリー・ストリングを戻します。

```
SELECT BLOB(MAP_NAME  || ':  ') || TOPOGRAPHIC_MAP
  FROM ONTARIO_SERIES_4
  WHERE TOPOGRAPHIC_MAP LIKE BLOB('%Pellow Island%')
```

CEILING または CEIL

スキーマは SYSIBM です。 (CEILING または CEIL 関数の SYSFUN バージョンは引き続き使用可能です。)

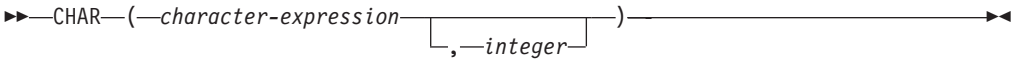
引き数よりも大きいか、または等しい整数で、最小の値を戻します。

引き数は、任意の組み込み数値タイプにすることができます。引き数が DECIMAL の場合は位取りは 0 になる以外は、関数の結果のデータ型と長さ属性は、引き数と同じになります。たとえば、データ型が DECIMAL(5,5) の引き数は DECIMAL(5,0) を戻します。

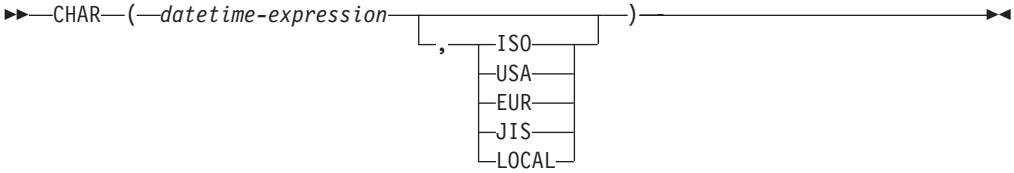
引き数が NULL になる可能性があるか、またはデータベース構成パラメーター DFT_SQLMATHWARN が YES に設定されている場合には、結果は NULL になる可能性があります。引き数が NULL の場合、結果は NULL 値になります。

CHAR

文字→文字:



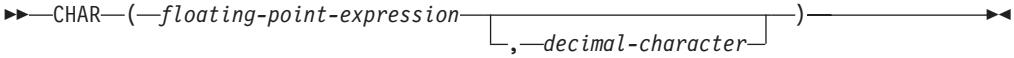
日付/時刻→文字:



整数→文字:

10 進数→文字:

浮動小数点数→文字:



スキーマは SYSIBM です。 SYSFUN.CHAR(*floating-point-expression*) シグニチャーは、引き続き使用可能です。その場合、小数点文字はロケール追従であるため、データベース・サーバーのロケールに応じてピリオドまたはコンマが戻されます。

CHAR 関数は、以下の固定長文字ストリング表記を戻します。

- 文字ストリング (最初の引き数がいずれかのタイプの文字ストリングの場合)
- 日付/時刻 (最初の引き数が日付、時刻、またはタイム・スタンプの場合)
- 整数 (最初の引き数が SMALLINT、INTEGER、または BIGINT の場合)
- 10 進数 (最初の引き数が 10 進数の場合)
- 倍精度浮動小数点 (最初の引き数が DOUBLE または REAL の場合)

最初の引き数は組み込みデータ型でなければなりません。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

注: CAST 式を使用してストリング式を戻すこともできます。

関数の結果は、固定長文字ストリングです。最初の引き数が NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。最初の引き数が NULL 値の場合には、結果も NULL 値です。

文字→文字

character-expression
CHAR、VARCHAR、LONG VARCHAR、または CLOB のいずれかのデータ型の値を戻す式。

integer
結果の固定長文字ストリングの長さ属性。値は 0 から 254 の範囲でなければなりません。

文字式の長さが結果の長さ属性より短い場合、結果の長さになるまでブランクが結果に埋められます。文字式の長さが結果の長さ属性より長い場合、切り捨てが行われます。その場合、切り捨てられた文字がすべてブランクで、文字式が LONG ストリング (LONG VARCHAR または CLOB) でない限り、警告 (SQLSTATE 01004) が戻されます。

日付/時刻→文字

datetime-expression
次の 3 つのデータ型のいずれかの式。

日付 (date)
結果は、2 番目の引き数によって指定された形式の日付の文字ストリング表記になります。結果の長さは 10 文字です。 2 番目の引き数が指定され、その値が有効な値でない場合には、エラーが戻されます (SQLSTATE 42703)。

時刻 (time)
結果は、2 番目の引き数によって指定された形式の時刻の文字ストリング表記になります。結果の長さは 8 文字です。 2 番目の引き数が指定され、その値が有効な値でない場合には、エラーが戻されます (SQLSTATE 42703)。

タイム・スタンプ (timestamp)
結果は、タイム・スタンプの文字ストリング表記になります。結果の長さは 26 文字です。2 番目の引き数は適用されないので、指定してはなりません (SQLSTATE 42815)。

ストリングのコード・ページは、アプリケーション・サーバーのデータベースのコード・ページになります。

整数→文字

integer-expression
整数データ型の値 (SMALLINT、INTEGER または BIGINT のいずれか) を戻す式。

結果は、SQL 整数定数の形式による引き数の文字ストリング表記になります。結果は、引き数内の有効桁数を表す *n* 個の文字で構成されます。引き数が負の場合は、負符号 (-) が前に付けられます。結果は、左そろえになります。

- 最初の引き数が短整数 (small integer) の場合、 その結果の長さは 6 になります。
- 最初の引き数が長精度整数 (large integer) の場合、 その結果の長さは 11 になります。

- 最初の引き数が 64 ビット整数 (big integer) の場合、その結果の長さは 20 になります。

結果内の文字数が、結果に定義されていた長さ未満の場合、結果の右側にブランクが埋められます。

ストリングのコード・ページは、アプリケーション・サーバーのデータベースのコード・ページになります。

10 進数→文字

decimal-expression
10 進数データ型の値を戻す式。別の精度と位取りが必要であれば、まず DECIMAL スカラー関数を使用して変更を行うことができます。

decimal-character
結果文字ストリングの中で 10 進数を区切るために使用する 1 バイト文字定数を指定します。文字定数を数字、正記号 (+)、負記号 (-)、またはブランク文字にすることはできません (SQLSTATE 42815)。デフォルトはピリオド (.) 文字です。

結果は、引き数の固定長文字ストリング表記になります。結果は、小数点文字と *p* 桁の数字からなります (*p* は *decimal-expression* の精度)。引き数が負の場合は、前に負符号が付けられます。結果の長さは 2+*p* です (*p* は *decimal-expression* の精度)。つまり、正の値には常に 1 個の末尾のブランクが付けられます。

ストリングのコード・ページは、アプリケーション・サーバーのデータベースのコード・ページになります。

浮動小数点数→文字

floating-point-expression
浮動小数点データ型 (DOUBLE または REAL) である値を戻す式。

decimal-character
結果文字ストリングの中で 10 進数を区切るために使用する 1 バイト文字定数を指定します。文字定数を数字、正記号 (+)、負記号 (-)、またはブランク文字にすることはできません (SQLSTATE 42815)。デフォルトはピリオド (.) 文字です。

結果は、浮動小数点定数形式の引き数の固定長文字ストリング表記になります。結果の長さは 24 文字です。引き数が負である場合、結果の先頭の文字は負符号 (-) になります。それ以外の場合、最初の文字は数字になります。引き数値がゼロの場合、結果は 0E0 になります。それ以外の場合の結果は、 *decimal-character* と一連の数字が後に続くゼロ以外の 1 桁の数字で小数部が構成されることで引き数の値を表すことのできる最小の文字数になります。結果の文字数が 24 未満の場合、結果の右側にブランクが埋められます。

ストリングのコード・ページは、アプリケーション・サーバーのデータベースのコード・ページになります。

例:

- PRSTDATE 列には、1988-12-25 に相当する内部値が入っているとします。以下の関数は、値 '12/25/1988' を戻します。

```
CHAR(PRSTDATE, USA)
```

- STARTING 列には 17:12:30 に相当する内部値が入っており、ホスト変数 HOUR_DUR (decimal(6,0)) は、050000 (すなわち 5 時間) の値をもった時刻期間であると仮定します。以下の関数は、値 '5:12 PM' を戻します。

```
CHAR(STARTING, USA)
```

  以下の関数は、値 '10:12 PM' を戻します。

```
CHAR(STARTING + :HOUR_DUR, USA)
```

- RECEIVED 列 (TIMESTAMP) には、列 PRSTDATE と列 STARTING を組み合わせたものに相当する内部値が入っているとします。以下の関数は、値 '1988-12-25-17.12.30.000000' を戻します。

```
CHAR(RECEIVED)
```

- LASTNAME 列は VARCHAR(15) と定義されています。以下の関数は、 10 文字の長さの固定長文字ストリングでこの列内に値を戻します。 10 文字を超える長さ (末尾ブランクを除く) の LASTNAME 値は切り捨てられて、警告が戻されます。

```
SELECT CHAR(LASTNAME,10) FROM EMPLOYEE
```

- EDLEVEL 列は SMALLINT と定義されています。以下の関数は、固定長文字ストリングでこの列内に値を戻します。 EDLEVEL 値が 18 であれば、CHAR(6) 値 '18 ' ('18' の後に 4 つのブランクが続く) で戻されます。

```
SELECT CHAR(EDLEVEL) FROM EMPLOYEE
```

- SALARY 列は、9 の精度と 2 の位取りをもった DECIMAL と定義されています。現行値 (18357.50) は、小数点文字としてコンマを使って表示されることになります (18357,50)。以下の関数は、値 '00018357,50' を戻します。

```
CHAR(SALARY, ',')
```

- SALARY 列内の値が 20000.25 から減算されて、デフォルトの小数点文字付きで表示されることになります。以下の関数は、値 '-0001642.75' を戻します。

```
CHAR(20000.25 - SALARY)
```

- ホスト変数 SEASONS_TICKETS は INTEGER と定義されていて、 10000 の値をもっていると想定します。以下の関数は、値 '10000.00 ' を戻します。

```
CHAR(DECIMAL(:SEASONS_TICKETS,7,2))
```

- ホスト変数 DOUBLE_NUM は DOUBLE と定義されていて、 -987.654321E-35 の値をもっていると想定します。以下の関数は、値 '-9.87654321E-33' を戻します。結果のデータ型は CHAR(24) であるので、結果には 9 つの末尾ブランクがあります。

```
CHAR(:DOUBLE_NUM)
```

関連資料:

- 184 ページの『式』
- 460 ページの『VARCHAR』

CHR

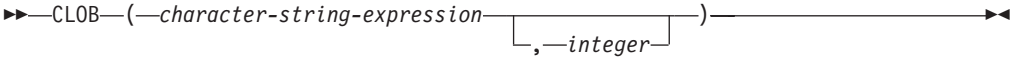
スキーマは SYSFUN です。

引き数で指定される ASCII コード値の文字を戻します。

引き数は INTEGER または SMALLINT のいずれかです。引き数の値は 0 から 255 の範囲でなければなりません。そうでない場合、戻り値は NULL 値になります。

関数の結果は CHAR(1) です。結果は NULL 値になることがあります。引き数が NULL 値である場合、その結果は NULL 値になります。

CLOB



スキーマは SYSIBM です。

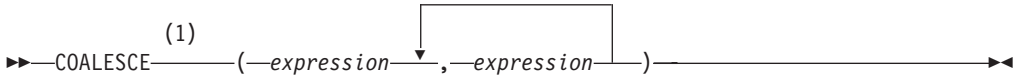
CLOB 関数は、文字ストリング・タイプの CLOB 表記を戻します。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

character-string-expression
文字ストリングである値を戻す式。

integer
結果の CLOB データ型の長さ属性を指定する整数値。値は 0 から 2 147 483 647 の範囲でなければなりません。 *integer* を指定しない場合、結果の長さは、最初の引き数の長さと同じになります。

関数の結果は CLOB です。引き数が NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引き数が NULL であれば、結果は NULL 値です。

COALESCE



注:

1 VALUE は COALESCE の同義語です。

スキーマは SYSIBM です。

COALESCE は、その値が NULL 値以外の最初の引き数を戻します。

引き数は指定された順序で評価され、関数の結果は NULL 値以外の最初の引き数になります。結果は、すべての引き数が NULL 値の場合にのみ NULL 値になります。選択された引き数は、必要に応じて結果の属性に変換されます。

引き数は互いに互換性がなければなりません。また、組み込みまたはユーザー定義のどちらのデータ型でもかまいません。 (この関数は、ユーザー定義関数の作成時にソース関数として使用することはできません。この関数は、すべての互換データ型を引き数として受け入れるので、ユーザー定義特殊型をサポートするための追加のシグニチャーを作成する必要はありません。)

例:

- DEPARTMENT 表のすべての行のすべての値を選択する場合に、部門の管理者 (MGRNO) が欠落しているときには (つまり NULL 値なら)、 'ABSENT' という値を戻すようにします。

```
SELECT DEPTNO, DEPTNAME, COALESCE(MGRNO, 'ABSENT'), ADMRDEPT
  FROM DEPARTMENT
```

- EMPLOYEE 表のすべての行から従業員番号 (EMPNO) と給与 (SALARY) を選択する場合に、給与が欠落していれば (つまり NULL 値なら)、値としてゼロを戻すようにします。

```
SELECT EMPNO, COALESCE(SALARY, 0)
  FROM EMPLOYEE
```

関連資料:

- 126 ページの『結果データ型の規則』

CONCAT

注:

1 CONCAT の同義語として || を使用することができます。

スキーマは SYSIBM です。

2 つのストリング引き数を連結した値を戻します。この 2 つの引き数のタイプには互換性がなければなりません。

関数の結果はストリングです。結果の長さは、2 つの引き数の長さの合計です。いずれかの引き数が NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引き数が NULL 値であれば、結果も NULL 値です。

関連資料:

• 184 ページの『式』

COS

スキーマは SYSIBM です。 (COS 関数の SYSFUN バージョンは引き続き使用可能です。)

引き数に対するコサイン (余弦) の値を戻します。引き数はラジアン単位の角度です。

引き数は、任意の組み込み数値タイプにすることができます。引き数は、関数での処理に必要な倍精度浮動小数点数に変換されます。

関数の結果は倍精度浮動小数点数になります。引き数が NULL になる可能性があるか、またはデータベース構成パラメーターで DFT_SQLMATHWARN が YES に設定されている場合には、結果は NULL になる可能性があります。引き数が NULL の場合、結果は NULL 値になります。

COSH

スキーマは SYSIBM です。

引き数に対する双曲線コサイン (余弦) の値を戻します。引き数はラジアン単位の角度です。

引き数は、任意の組み込み数値データ型にすることができます。引き数は、関数による処理に必要な倍精度浮動小数点数に変換されます。

関数の結果は倍精度浮動小数点数になります。引き数が NULL になる可能性があるか、またはデータベース構成パラメーターで DFT_SQLMATHWARN が YES に設定されている場合には、結果は NULL になる可能性があります。引き数が NULL の場合、結果は NULL 値になります。

COT

スキーマは SYSIBM です。 (COT 関数の SYSFUN バージョンは引き続き使用可能です。)

引き数に対するコタンジェント (余接) の値を戻します。引き数はラジアン単位の角度です。

引き数は、任意の組み込み数値タイプにすることができます。引き数は、関数での処理に必要な倍精度浮動小数点数に変換されます。

関数の結果は倍精度浮動小数点数になります。引き数が NULL になる可能性があるか、またはデータベース構成パラメーターで DFT_SQLMATHWARN が YES に設定されている場合には、結果は NULL になる可能性があります。引き数が NULL の場合、結果は NULL 値になります。

DATE

スキーマは SYSIBM です。

DATE 関数は、値から日付を戻します。

引き数は、日付、タイム・スタンプ、3 652 059 以下の正の整数、日付またはタイム・スタンプの有効なストリング表記、または CLOB、LONG VARCHAR、DBCLOB、または LONG VARGRAPHIC ではない長さ 7 文字のストリングのいずれかでなければなりません。

Unicode データベースだけが、日付またはタイム・スタンプの GRAPHIC ストリング表現である引き数をサポートします。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

引き数が長さ 7 のストリングの場合、 *yyyynnn* という形式の有効な日付を表していなければなりません。ここで、*yyyy* は年を示す数字、 *nnn* は年間通算日を示す 001 から 366 までの数字です。

関数の結果は日付です。引き数が NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引き数が NULL 値であれば、結果は NULL 値です。

その他の規則は、引き数のデータ型に応じて以下のように異なります。

- 引き数が日付、タイム・スタンプ、または日付やタイム・スタンプの有効なストリング表記の場合
  - 結果はその値の日付部分です。
- 引き数が数値の場合
  - 結果は、0001 (1 月 1 日) から数えて *n* -1 日後の日付です (*n* は数字の整数部分)。
- 引き数が長さ 7 のストリングの場合
  - 結果は、そのストリングで表された日付になります。

例:

列 RECEIVED (タイム・スタンプ) には、 '1988-12-25-17.12.30.000000' に相当する内部値が入っているものとします。

- 以下の例の結果は、'1988-12-25' の内部表記になります。

```
DATE(RECEIVED)
```

- 以下の例の結果は、'1988-12-25' の内部表記になります。

```
DATE('1988-12-25')
```

- 以下の例の結果は、'1988-12-25' の内部表記になります。

```
DATE('25.12.1988')
```

- 以下の例の結果は、'0001-02-04' の内部表記になります。

```
DATE(35)
```

DAY

スキーマは SYSIBM です。

DAY 関数は、値の日の部分を戻します。

引き数は、日付、タイム・スタンプ、日付期間、タイム・スタンプ期間であるか、または CLOB でも LONG VARCHAR でもない日付あるいはタイム・スタンプの有効な文字ストリング表記でなければなりません。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

この関数の結果は長精度整数 (large integer) です。引き数が NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引き数が NULL 値であれば、結果は NULL 値です。

その他の規則は、引き数のデータ型に応じて以下のように異なります。

- 引き数が日付、タイム・スタンプ、または日付やタイム・スタンプの有効なストリング表記の場合
  - 結果は、値の日の部分 (1 から 31 の整数) になります。
- 引き数が日付期間またはタイム・スタンプ期間の場合
  - 結果は、値の日の部分 (-99 から 99 の整数) になります。ゼロ以外の結果の符号は、引き数と同じになります。

例:

- PROJECT 表を使用して、 WELD LINE PLANNING プロジェクト (PROJNAME) の終了予定日 (PRENDATE) をホスト変数 END_DAY (短精度整数) に設定します。

```
SELECT DAY(PRENDATE)
 INTO :END_DAY
 FROM PROJECT
 WHERE PROJNAME = 'WELD LINE PLANNING'
```

  サンプル表を使用した場合、結果として END_DAY は 15 に設定されます。

- 列 DATE1 (日付) には、2000-03-15 に相当する内部値が入っていて、列 DATE2 (日付) には、1999-12-31 に相当する内部値が入っているとします。

```
DAY(DATE1 - DATE2)
```

  結果は、15 の値になります。

DAYNAME

スキーマは SYSFUN です。

データベースの開始時点のロケールに基づいて、引き数の日の部分の曜日名から成る大文字小文字混合文字ストリング (たとえば、 Friday) を戻します。

引き数は、日付またはタイム・スタンプであるか、または CLOB でも LONG VARCHAR でもない日付あるいはタイム・スタンプの有効な文字ストリング表記でなければなりません。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

関数の結果は VARCHAR(100) です。結果は NULL 値になることがあります。引き数が NULL 値である場合、その結果は NULL 値になります。

DAYOFWEEK

引き数の曜日を 1 から 7 の範囲の整数値として戻します。1 は日曜日を表します。

引き数は、日付またはタイム・スタンプであるか、または CLOB でも LONG VARCHAR でもない日付あるいはタイム・スタンプの有効な文字ストリング表記でなければなりません。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

関数の結果は INTEGER になります。結果は NULL 値になることがあります。引き数が NULL 値である場合、その結果は NULL 値になります。

DAYOFWEEK_ISO

スキーマは SYSFUN です。

引き数の曜日を 1 から 7 の範囲の整数値として戻します。1 は月曜日を表します。

引き数は、日付またはタイム・スタンプであるか、または CLOB でも LONG VARCHAR でもない日付あるいはタイム・スタンプの有効な文字ストリング表記でなければなりません。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

関数の結果は INTEGER になります。結果は NULL 値になることがあります。引き数が NULL 値である場合、その結果は NULL 値になります。

DAYOFYEAR

スキーマは SYSFUN です。

引き数の年間通算日を、1 から 366 の範囲の整数値として戻します。

引き数は、日付またはタイム・スタンプであるか、または CLOB でも LONG VARCHAR でもない日付あるいはタイム・スタンプの有効な文字ストリング表記でなければなりません。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

関数の結果は INTEGER になります。結果は NULL 値になることがあります。引き数が NULL 値である場合、その結果は NULL 値になります。

DAYS

スキーマは SYSIBM です。

DAYS 関数は、日付の整数表記を戻します。

引き数は、日付またはタイム・スタンプであるか、または CLOB でも LONG VARCHAR でもない日付あるいはタイム・スタンプの有効な文字ストリング表記でなければなりません。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

この関数の結果は長精度整数 (large integer) です。引き数が NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引き数が NULL 値であれば、結果は NULL 値です。

結果は、1 月 1 日 (0001) から *D* までの日数に、 1 を加えた数になります (*D* は、引き数に DATE 関数を適用した場合の結果となる日付)。

例:

- PROJECT 表を使用して、プロジェクト (PROJNO) 'IF2000' に要する見積日数 (PRENDATE - PRSTDATE) をホスト変数 EDUCATION_DAYS (整数) に設定します。

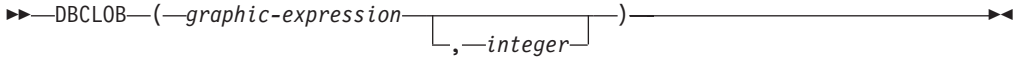
  ```
  SELECT DAYS(PRENDATE) - DAYS(PRSTDATE)
   INTO :EDUCATION_DAYS
   FROM PROJECT
   WHERE PROJNO = 'IF2000'
  ```

  結果として EDUCATION_DAYS は 396 に設定されます。
- PROJECT 表を使用して、ホスト変数 TOTAL_DAYS (int) に、部署 (DEPTNO) 'E21' のすべてのプロジェクトについての経過日数見積もり (PRENDATE - PRSTDATE) の合計を設定します。

  ```
  SELECT SUM(DAYS(PRENDATE) - DAYS(PRSTDATE))
   INTO :TOTAL_DAYS
   FROM PROJECT
   WHERE DEPTNO = 'E21'
  ```

  サンプル表を使用した場合、結果として TOTAL_DAYS は 1584 に設定されます。

DBCLOB



スキーマは SYSIBM です。

DBCLOB 関数は、GRAPHIC ストリング・タイプの DBCLOB 表記を戻します。

Unicode データベースでは、指定した引き数が文字ストリングであると、まず GRAPHIC ストリングに変換されてから、関数が実行されます。最後の文字が高サロゲートになるように出力ストリングが切り捨てられた場合、そのサロゲートは次のいずれかになります。

- 指定した引き数が文字ストリングの場合は現状のままになる。
- 指定した引き数が GRAPHIC ストリングの場合はブランク文字 (X'0020') に変換される。

今後のリリースでこの動作は変更される可能性があります。

関数の結果は DBCLOB です。引き数が NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引き数が NULL 値であれば、結果は NULL 値です。

graphic-expression
GRAPHIC ストリング値を戻す式。

integer
結果の DBCLOB データ型の長さ属性を指定する整数値。値は 0 から 1 073 741 823 の範囲でなければなりません。 *integer* を指定しない場合、結果の長さは、最初の引き数の長さと同じになります。

DBPARTITIONNUM

スキーマは SYSIBM です。

DBPARTITIONNUM 関数は、行のパーティション番号を戻します。 たとえば、SELECT 文節で使用すると、 SELECT ステートメントの結果の生成に使用された表の各行のパーティション番号を戻します。

遷移変数および表に戻されるパーティション番号は、区分化キー列の現行遷移値から得られます。たとえば、挿入前トリガーにおいて、新しい遷移変数の現行値があれば、関数は予定パーティション番号を戻します。ただし、区分化キー列の値はそれ以後の挿入前トリガーによって変更される場合があります。したがって、データベースに挿入される時点での行の最終パーティション番号は、予定値とは異なるかもしれません。

引き数は、表内の列の修飾された名前または無修飾の名前でなければなりません。該当の列は、どのようなデータ型であっても構いません。 (この関数は、ユーザー定義関数の作成時にソース関数として使用することはできません。この関数は、すべてのデータ型を引き数として受け入れるので、ユーザー定義特殊タイプをサポートするための追加のシグニチャーを作成する必要はありません。) *column-name* がビューの列を参照する場合、その列のビューの式は基本を成す基本表の列を参照する必要があり、そのビューは削除可能でなければなりません。ネストされているか、または共通の表式は、ビューと同じ規則に従います。

DBPARTITIONNUM 関数によってパーティション番号が戻される特定の行 (または表) は、この関数を使用する SQL ステートメントのコンテキストから判別されます。

結果のデータ型は INTEGER であり、NULL 値にはなりません。行レベルの情報が戻されるので、どの列が表に指定されるかに関係なく、結果は同じです。
db2nodes.cfg ファイルがない場合、結果は 0 になります。

DBPARTITIONNUM 関数は、複製された表、チェック制約内、または生成された列の定義で使用することはできません (SQLSTATE 42881)。

バージョン 8 より前の旧バージョンとの互換性を保つために、
DBPARTITIONNUM をキーワード NODENUMBER に置き換えてもかまいません。

例:

- EMPLOYEE の行が DEPARTMENT の従業員の部門記述とは異なるパーティション上にある行の数を数えます。

```
SELECT COUNT(*) FROM DEPARTMENT D, EMPLOYEE E
  WHERE D.DEPTNO=E.WORKDEPT
  AND DBPARTITIONNUM(E.LASTNAME) <> DBPARTITIONNUM(D.DEPTNO)
```

- 2 つの表の行が同じパーティションにある EMPLOYEE および DEPARTMENT の表を結合します。

```
SELECT * FROM DEPARTMENT D, EMPLOYEE E
  WHERE DBPARTITIONNUM(E.LASTNAME) = DBPARTITIONNUM(D.DEPTNO)
```

- 表 EMPLOYEE で前トリガーを作成して、従業員の挿入の際には必ず、EMPINSERTLOG1 という表に従業員番号と新しい行の予定パーティション番号を記録します。

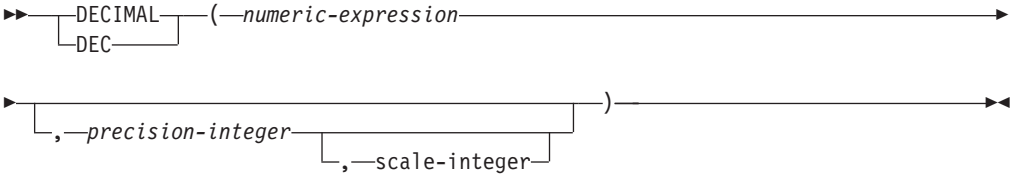
```
CREATE TRIGGER EMPINSLOGTRIG1
BEFORE INSERT ON EMPLOYEE
REFERENCING NEW AW NEWTABLE
  FOR EACH ROW
INSERT INTO EMPINSERTLOG1
VALUES(NEWTABLE.EMPNO, DBPARTITIONNUM
(NEWTABLE.EMPNO))
```

関連資料:

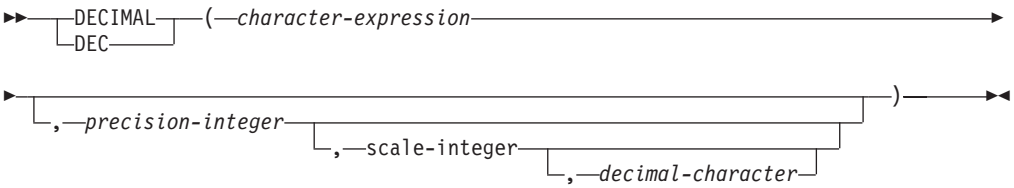
- 「*SQL リファレンス 第 2 巻*」の『CREATE VIEW ステートメント』

DECIMAL

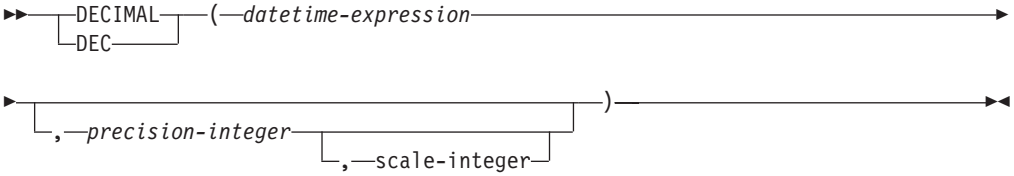
数値→ 10 進数:



文字→ 10 進数:



日付/時刻→ 10 進数:



スキーマは SYSIBM です。

DECIMAL 関数は、以下の 10 進表記を戻します。

- 数値
- 10 進数の文字ストリング表記
- 整数の文字ストリング表記
- 浮動小数点数の文字ストリング表記
- 日付/時刻 (引き数が日付、時刻、またはタイム・スタンプの場合)

Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

関数の結果は、精度 *p*、位取り *s* の 10 進数になります (*p* と *s* はそれぞれ 2 番目と 3 番目の引き数)。最初の引き数を NULL にすることができる場合、結果も NULL にすることができます。最初の引き数が NULL であれば、結果は NULL 値になります。

数値→ 10 進数

numeric-expression
数値データ型の値を戻す式。

precision-integer
1 から 31 の範囲の値の整数定数。

precision-integer のデフォルト値は、 *numeric-expression* のデータ型によって異なります。

- 浮動小数点および 10 進数の場合は 15
- 64 ビット整数の場合は 19
- 長精度整数の場合は 11
- 短整数の場合は 5

scale-integer
0 から *precision-integer* の値までの範囲の整数定数。デフォルト値はゼロです。

結果は、最初の引き数が精度 *p*、位取り *s* の 10 進数の列または変数に割り当てられた場合と同じ数になります (*p* と *s* は、それぞれ 2 番目と 3 番目の引き数)。数値の整数部分を表すために必要な有効 10 進桁数が、 *p* - *s* より大きい場合はエラーになります。

文字→ 10 進数

character-expression
長さが文字定数の最大長 (4 000 バイト) 以下の文字ストリングである値を戻す式。 CLOB または LONG VARCHAR データ型にすることはできません。先行ブランクと末尾のブランクは、ストリングから除去されます。この結果のサブストリングは、 SQL 整数または 10 進定数を形成するための規則に準拠していなければなりません (SQLSTATE 22018)。

定数 *decimal-character* のコード・ページに一致させるために必要であれば、 *character-expression* はデータベース・コード・ページに変換されます。

precision-integer
結果の精度を指定する整数定数 (値の範囲は 1 から 31)。この指定がない場合のデフォルト値は 15 です。

scale-integer
結果の位取りを指定する整数定数 (値の範囲は 0 から *precision-integer*)。 この指定がない場合のデフォルト値は 0 です。

decimal-character
character-expression の小数部分と整数部分とを区切るために使用する 1 バイト文字定数を指定します。この文字には、数字、プラス (+)、マイナス (-)、またはブランクを使用できず、 *character-expression* の中に最高で 1 回しか使用することができません (SQLSTATE 42815)。

結果は、精度 *p*、位取り *s* の 10 進数になります (*p* と *s* は、それぞれ 2 番目と 3 番目の引き数)。小数点より右側の数字の桁数が、位取りより多い場合、 10 進数の終わりから数字が切り捨てられます。 *character-expression* の小数点文字の左側にある有効数字 (数値の整数部分) の桁数が *p* - *s* よりも多い場合は、エラーになります (SQLSTATE 22003)。*decimal-character* 引き数に別の値が指定されている場合、サブストリングに使われているデフォルトの小数点文字は無効になります (SQLSTATE 22018)。

日付/時刻→ 10 進数

datetime-expression
次のデータ型のいずれかの式。

- DATE。結果は、日付を *yyyymmdd* で表した DECIMAL(8,0) 値になります。
- TIME。結果は、時間を *hhmmss* で表した DECIMAL(6,0) 値になります。
- TIMESTAMP。結果は、タイム・スタンプを *yyyymmddhhmmss.nnnnnn* で表した DECIMAL(20,6) 値になります。

この関数を使ってユーザーは、精度を指定したり、精度と位取りを指定したりすることができます。ただし、精度を指定しないと、位取りを指定することはできません。 (precision,scale) のデフォルト値は、DATE の場合は (8,0)、 TIME の場合は (6,0)、TIMESTAMP の場合は (20,6) です。

結果は、精度 *p*、位取り *s* の 10 進数になります (*p* と *s* は、それぞれ 2 番目と 3 番目の引き数)。小数点より右側の数字の桁数が、位取りより多い場合、終わりから数字が切り捨てられます。 *datetime-expression* の小数点文字の左側にある有効数字 (数値の整数部分) の桁数が *p - s* よりも多い場合は、エラーになります (SQLSTATE 22003)。

例:

- EMPLOYEE 表の EDLEVEL 列 (データ型 = SMALLINT) の選択リストに (精度が 5 で、位取りが 2 の) DECIMAL データ型が必ず戻されるように DECIMAL 関数を使用します。選択リストには、EMPNO 列も入れておいてください。

```
SELECT EMPNO, DECIMAL(EDLEVEL,5,2)
  FROM EMPLOYEE
```

- ホスト変数 PERIOD のタイプが INTEGER であるとします。その値を日付期間として使用するためには、それを decimal(8,0) として「キャスト」する必要があります。

```
SELECT PRSTDATE + DECIMAL(:PERIOD,8)
  FROM PROJECT
```

- SALARY 列の更新内容が、小数点文字にコンマを使用した文字ストリングとして、ウィンドウから入力されるものとします (ユーザーは、たとえば 21400,50 と入力します)。アプリケーションが妥当性検査を行った後、これを、CHAR(10) として定義されたホスト変数 newsalary に設定します。

```
UPDATE STAFF
  SET SALARY = DECIMAL(:newsalary, 9, 2, ',')
  WHERE ID = :empid;
```

  newsalary の値は 21400.50 になります。

- 値にデフォルト値の小数点文字 (.) を追加します。

```
DECIMAL('21400,50', 9, 2, '.')
```

  この例では、小数点文字としてピリオド (.) を指定しているのに、第 1 引き数の中で区切り文字としてコンマ (,) が使われているため、エラーになります。

- STARTING (time) 列に、 '12:10:00' に相当する内部値が入っていると想定します。

```
DECIMAL(STARTING)
```

  結果は、121 000 の値になります。

- RECEIVED (timestamp) 列に、 '1988-12-22-14.07.21.136421' に相当する内部値が入っていると想定します。

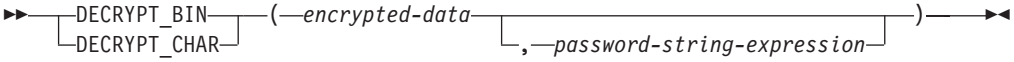
```
DECIMAL(RECEIVED)
```

  結果は、19 881 222 140 721.136421 の値になります。

- 以下の表は、各種の日時入力値の 10 進数結果とその結果の精度と位取りを示しています。

DECIMAL(引き数)	精度と位取り	結果
DECIMAL(2000-03-21)	(8,0)	20000321
DECIMAL(2000-03-21, 10)	(10,0)	20000321
DECIMAL(2000-03-21, 12, 2)	(12,2)	20000321.00
DECIMAL(12:02:21)	(6,0)	120221
DECIMAL(12:02:21, 10)	(10,0)	120221
DECIMAL(12:02:21, 10, 2)	(10,2)	120221.00
DECIMAL(2000-03-21-12.02.21.123456)	(20, 6)	20000321120221.123456
DECIMAL(2000-03-21-12.02.21.123456, 23)	(23, 6)	20000321120221.123456
DECIMAL(2000-03-21-12.02.21.123456, 23, 4)	(23, 4)	20000321120221.1234

DECRYPT_BIN および DECRYPT_CHAR



スキーマは SYSIBM です。

DECRYPT_BIN 関数と DECRYPT_CHAR 関数はどちらも、 *encrypted-data* 暗号化解除の結果である値を返します。暗号化解除に使用されるパスワードは、*password-string-expression*、または SET ENCRYPTION PASSWORD ステートメントで割り当てられた ENCRYPTION PASSWORD 値のいずれかです。 DECRYPT_BIN および DECRYPT_CHAR 関数は、 ENCRYPT 関数を使って暗号化された値のみを暗号化解除できます (SQLSTATE 428FE)。

encrypted-data
CHAR FOR BIT DATA 値または VARCHAR FOR BIT DATA 値を、暗号化された完全なデータ・ストリングとして戻す式です。データ・ストリングは、ENCRYPT 関数を使って暗号化されたものでなければなりません。

password-string-expression
少なくとも 6 バイトで 127 バイトを超えない CHAR または VARCHAR 値を返す式です (SQLSTATE 428FC)。この式は、データの暗号化に使用されたパスワードと同じでなければなりません。それ以外の場合、暗号化解除はエラーになります (SQLSTATE 428FD)。パスワード引き数が NULL、または与えられていない場合、セッションに設定されていなければならない ENCRYPTION PASSWORD 値を使用してデータは暗号化されます (SQLSTATE 51039)。

DECRYPT_BIN 関数の結果は VARCHAR FOR BIT DATA です。DECRYPT_CHAR 関数の結果は VARCHAR です。 *encrypted-data* にヒントが組み込まれている場合、そのヒントは関数によって返されません。結果の長さ属性は、*encrypted-data* のデータ型の長さマイナス 8 バイトになります。関数によって実際に返される長さの値は、暗号化されたオリジナル・ストリングの長さに一致します。暗号化ストリングを超えるバイトが *encrypted-data* に入っている場合、それらのバイトは関数によって返されません。

最初の引き数が NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。最初の引き数が NULL 値の場合には、結果も NULL 値です。

データが暗号化されたときのコード・ページとは異なるコード・ページを使用する、別のシステムでデータが暗号化解除された場合、暗号化解除された値をデータベース・コード・ページに変換するとき、長さが超過してしまう可能性があります。この場合、*encrypted-data* 値をより大きなバイト数の VARCHAR ストリングに cast する必要があります。

例:

例 1: この例では、暗号化パスワードを保持するために ENCRYPTION PASSWORD 値が使用されています。

```
SET ENCRYPTION PASSWORD = 'Ben123';
INSERT INTO EMP(SSN) VALUES ENCRYPT('289-46-8832');
SELECT DECRYPT_CHAR(SSN)
   FROM EMP;
```

返される値は '289-46-8832' です。

例 2: この例では、暗号化パスワードが明示的に渡されています。

```
INSERT INTO EMP (SSN) VALUES ENCRYPT('289-46-8832','Ben123','');
SELECT DECRYPT(SSN,'Ben123')
   FROM EMP;
```

この例は、値 '289-46-8832' を返します。

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『SET ENCRYPTION PASSWORD ステートメント』
- 356 ページの『ENCRYPT』
- 363 ページの『GETHINT』

DEGREES

スキーマは SYSFUN です。

ラジアン単位の引き数を度単位の角度に変換して戻します。

引き数は、任意の組み込み数値タイプにすることができます。引き数は、関数での処理に必要な倍精度浮動小数点数に変換されます。

関数の結果は倍精度浮動小数点数になります。結果は NULL 値になることがあります。引き数が NULL 値である場合、その結果は NULL 値になります。

DEREF

DEREF 関数は引き数のターゲット・タイプのインスタンスを戻します。

引き数は、有効範囲が定義された参照データ型が指定されている任意の値にすることができます (SQLSTATE 428DT)。

結果の静的データ型は、引き数のターゲット・タイプです。結果の動的データ型は、引き数のターゲット・タイプのサブタイプです。結果は NULL 値の場合もあります。 *expression* が NULL 値か、または *expression* が突き合わせする OID がターゲット表にない参照の場合、結果は NULL 値になります。

結果は参照のターゲット・タイプのサブタイプのインスタンスです。結果は参照値と突き合わせするオブジェクト ID がある参照の、ターゲット表またはターゲット・ビューの行を検出することにより決定されます。この行のタイプによって結果の動的タイプが決定されます。結果のタイプがターゲット表の副表の行またはターゲット・ビューのサブビューの行に基づく場合があるため、ステートメントの許可 ID にはターゲット表とその副表のすべて、またはターゲット・ビューとそのサブビューのすべての SELECT 特権が必要です (SQLSTATE 42501)。

例:

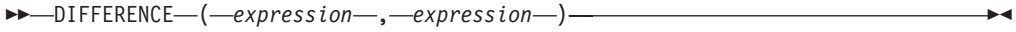
EMPLOYEE はタイプ EMP の表であり、そのオブジェクト ID 列は EMPID であるとします。次の照会は、EMPLOYEE 表 (およびその副表) の行ごとに、タイプ EMP (またはそのサブタイプのいずれか) のオブジェクトを戻します。この照会では、EMPLOYEE およびその副表すべてに対する SELECT 権限が必要です。

```
SELECT DEREF(EMPID) FROM EMPLOYEE
```

関連資料:

- 456 ページの『TYPE_NAME』

DIFFERENCE



スキーマは SYSFUN です。

ストリングへの SOUNDEX 関数の適用に基づいて、 2 つのストリングの音の差を示す 0 から 4 の値を戻します。 4 の値は可能な限り最良の音の一致を示します。

引き数は、CHAR または VARCHAR いずれかの文字ストリング (最大長 4 000 バイト) にすることができます。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

関数の結果は INTEGER になります。結果は NULL 値になることがあります。引き数が NULL 値である場合、その結果は NULL 値になります。

例:

```
VALUES (DIFFERENCE('CONSTRAINT','CONSTANT'),SOUNDEX('CONSTRAINT'),
SOUNDEX('CONSTANT')),
(DIFFERENCE('CONSTRAINT','CONTRITE'),SOUNDEX('CONSTRAINT'),
SOUNDEX('CONTRITE'))
```

この例では、以下を戻します。

```
 1           2    3
----------- ---- ----
4 C523 C523
2 C523 C536
```

最初の行で、SOUNDEX の語は同じ結果になりますが、2 行目の語は類似性があるにすぎません。

関連資料:

• 430 ページの『SOUNDEX』

DIGITS

スキーマは SYSIBM です。

DIGITS 関数は、数値の文字ストリング表記を戻します。

引き数は、SMALLINT、INTEGER、BIGINT または DECIMAL の型の値を戻す式でなければなりません。

引き数が NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引き数が NULL 値であれば、結果は NULL 値です。

この関数の結果は、引き数の位取りに関係なく、引き数の絶対値を表す固定長文字ストリングになります。結果には、符号も小数点文字も示されません。結果は、必要に応じてストリングを埋めるための先行ゼロの付いた数字だけで構成されます。ストリングの長さは次のとおりです。

- 引き数が短整数 (small integer) の場合は 5
- 引き数が長精度整数 (large integer) の場合は 10
- 引き数が 64 ビット整数 (big integer) の場合は 19
- 引き数が精度 *p* の 10 進数の場合は *p*

例:

- 表 TABLEX に、INTCOL という INTEGER 列があり、その値が 10 桁の数値であるとします。列 INTCOL に入っている最初の 4 桁の数字からなる、異なる 4 文字の組み合わせすべてのリストを作成します。

  ```
  SELECT DISTINCT SUBSTR(DIGITS(INTCOL),1,4)
   FROM TABLEX
  ```

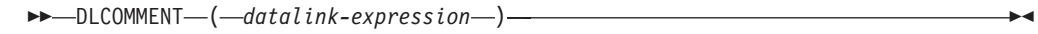
- COLUMNX のデータ型が DECIMAL(6,2) であり、その値の 1 つが -6.28 であると想定します。この値に対して次の関数を実行すると、

  ```
  DIGITS(COLUMNX)
  ```

  値 '000628' が戻されます。

  この結果は、ストリングをこの長さまで埋めるための先行ゼロの付いた、長さ 6 (列の精度) のストリングになります。符号も小数点文字も結果には示されません。

DLCOMMENT



スキーマは SYSIBM です。

DLCOMMENT 関数は、DATALINK 値からコメント値を戻します (コメント値がある場合)。

引き数は DATALINK のデータ型の値を戻す式でなければなりません。

関数の結果は VARCHAR(254) です。引き数が NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引き数が NULL 値であれば、結果は NULL 値です。

例:

- HOCKEY_GOALS 表から日付、記述およびコメントを選択する (ARTICLES 列内のリンクから) ステートメントを準備します。選択する行は、Richard 兄弟 (Maurice または Henri) のどちらかが得点したゴールの行です。

```
stmtvar = "SELECT DATE_OF_GOAL, DESCRIPTION, DLCOMMENT(ARTICLES)
             FROM HOCKEY_GOALS
             WHERE BY_PLAYER = 'Maurice Richard'
             OR BY_PLAYER = 'Henri Richard' ";
EXEC SQL PREPARE HOCKEY_STMT FROM :stmtvar;
```

- スカラー関数を使用して、表 TBLA にある行の列 COLA に DATALINK 値を挿入したとします。

```
DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','A comment')
```

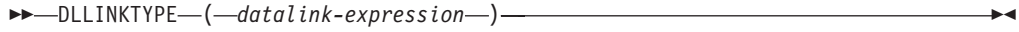
そして、その値を次の関数で処理します。

```
DLCOMMENT(COLA)
```

次の値が戻されます。

```
A comment
```

DLLINKTYPE



スキーマは SYSIBM です。

DLLINKTYPE 関数は、DATALINK 値からリンク・タイプ値を戻します。

引き数は、 DATALINK のデータ型の値を戻す式でなければなりません。

関数の結果は VARCHAR(4) になります。引き数が NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引き数が NULL 値であれば、結果は NULL 値です。

例:

- スカラー関数を使用して、表 TBLA にある行の列 COLA に DATALINK 値を挿入したとします。

  ```
  DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','a comment')
  ```

  そして、その値を次の関数で処理します。

  ```
  DLLINKTYPE(COLA)
  ```

  次の値が戻されます。

  ```
  URL
  ```

DLNEWCOPY

スキーマは SYSIBM です。

DLNEWCOPY 関数は、参照先ファイルが変更されたことを示す属性を持つ DATALINK 値を戻します。 SQL UPDATE ステートメントの結果そのような値が DATALINK 列に割り当てられた場合、リンク先のファイルの更新が完了したことが DB2 に通知されます。 DATALINK 列が RECOVERY YES を指定されて定義された場合、リンク先ファイルの新規バージョンは非同期でアーカイブされます。 SQL INSERT ステートメントの結果そのような値が DATALINK 列に割り当てられた場合、エラー (SQLSTATE 428D1) が戻されます。

data-location
完全な URL 値の入った可変長文字ストリングを指定する VARCHAR(200) 式。値は事前に SELECT ステートメントによって DLURLCOMPLETEWRITE 関数を介して取得されている場合があります。

has-token
データ・ロケーションに書き込みトークンが入っているかどうかを示す INTEGER 値。

0 データ・ロケーションに書き込みトークンは入っていません。

1 データ・ロケーションには書き込みトークンが入ります。

値が 0 または 1 ではない場合 (SQLSTATE 42815)、またはデータ・ロケーションに組み込まれたトークンが無効の場合 (SQLSTATE 428D1)、エラーが生じます。

この関数の結果は、書き込みトークンを持たない DATALINK 値となります。 *data-location* と *has-token* のどちらも、NULL になれません。

WRITE PERMISSION ADMIN REQUIRING TOKEN FOR UPDATE を使用して定義した DATALINK 列の場合、 SQL UPDATE ステートメント (SQLSTATE 428D1) を完了するには、書き込みトークンはデータ・ロケーションの中になければなりません。一方 WRITE PERMISSION ADMIN NOT REQUIRING TOKEN FOR UPDATE の場合は、書き込みトークンは必要ありませんがデータ・ロケーション内に置くことはできます。

WRITE PERMISSION ADMIN REQUIRING TOKEN FOR UPDATE を指定して定義された DATALINK 列の場合、指定のファイルが開かれているのであれば、書き込みトークンはそのファイルを開くために使用したものと同じでなければなりません (SQLSTATE 428D1)。

任意の WRITE PERMISSION ADMIN 列で、書き込みトークンが有効期限切れであっても、そのトークンが指定のファイルを開くために使用したものと同じであれば、それは書き込みアクセスのためにまだ有効と見なされます。

ファイル更新が行われなかった場合、または DATALINK ファイルが WRITE PERMISSION BLOCKED/FS や NO LINK CONTROL などの他のオプションとリンクしている場合、この関数は DLVALUE と同様に動作します。

例:

- スカラー関数を使用して、表 TBLA にある列 COLA (WRITE PERMISSION ADMIN REQUIRING TOKEN FOR UPDATE を指定して定義された) に DATALINK 値を挿入したとします。

```
DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','a comment')
```

  スカラー関数 DLURLCOMPLETEWRITE を使用して値をフェッチします。

```
SELECT DLURLCOMPLETEWRITE(COLA)
  FROM TBLA
  WHERE ...
```

  次の値が返されます。

```
HTTP://DLFS.ALMADEN.IBM.COM/x/y/****************;a.b
```

  ここで ********** は書き込みトークンを表します。

  上記の値を使用してファイルを見付け、その内容を更新します。次の SQL UPDATE ステートメントを出して、ファイルが正常に変更されたことを示します。

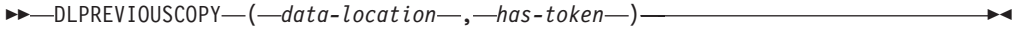
```
UPDATE TBLA
  SET COLA = DLNEWCOPY('http://dlfs.almaden.ibm.com/x/y/********
    ********;a.b', 1)
  WHERE ...
```

  ここで ********** は、URL 値の参照先ファイルを変更するために使用したものと同じトークンを表します。 COLA が WRITE PERMISSION ADMIN NOT REQUIRING TOKEN FOR UPDATE を指定して定義された場合、上記の例では書き込みトークンが必要ないことに注意してください。

- 2 番目の引き数 (*has-token*) の値は、以下の CASE ステートメントで代用することができます。 URL 値が *url_file* という変数に入っていると想定します。次の SQL UPDATE ステートメントを出して、ファイルが正常に変更されたことを示します。

```
EXEC SQL UPDATE TBLA
  SET COLA = DLNEWCOPY(:url_file,
    (CASE
      WHEN LENGTH(:url_file) = LENGTH(DLURLCOMPLETEONLY(COLA))
      THEN 0
      ELSE 1
     END))
  WHERE ...
```

DLPREVIOUSCOPY



スキーマは SYSIBM です。

DLPREVIOUSCOPY 関数は、DATALINK 値を戻します。この値には、以前のバージョンのファイルをリストアするように指示する属性があります。その値が SQL UPDATE の結果として DATALINK 列に割り当てられる場合、 DB2 が起動され、リンクされたファイルを以前にコミットされたバージョンからリストアします。SQL INSERT ステートメントの結果そのような値が DATALINK 列に割り当てられた場合、エラー (SQLSTATE 428D1) が戻されます。

data-location
完全な URL 値の入った可変長文字ストリングを指定する VARCHAR(200) 式。この値は、DLURLCOMPLETEWRITE 関数の SELECT ステートメントによって、以前取得している場合があります。

has-token
データ・ロケーションに書き込みトークンが入っているかどうかを示す整数値。

0 データ・ロケーションに書き込みトークンは入っていません。

1 データ・ロケーションに書き込みトークンは入っています。

値が 0 でも 1 でもない場合 (SQLSTATE 42815)、またはデータ・ロケーションに組み込まれたトークンが無効の場合 (SQLSTATE 428D1)、エラーが発生します。

この関数の結果は、書き込みトークンなしの DATALINK 値となります。*data-location* と *has-token* のどちらも、NULL になれません。

WRITE PERMISSION ADMIN REQUIRING TOKEN FOR UPDATE を使用して定義した DATALINK 列の場合、 SQL UPDATE ステートメント (SQLSTATE 428D1) を完了するには、書き込みトークンはデータ・ロケーションの中になければなりません。他方、WRITE PERMISSION ADMIN NOT REQUIRING TOKEN FOR UPDATE の場合、書き込みトークンは必要ありませんが、データ・ロケーション内に置くことはできます。

WRITE PERMISSION ADMIN REQUIRING TOKEN FOR UPDATE を使用して定義した DATALINK 列の場合、書き込みトークンは指定したファイルをオープンするために使用したトークンと同じでなければなりません (ファイルをオープンした場合) (SQLSTATE 428D1)。

すべての WRITE PERMISSION ADMIN 列に関して、書き込みトークンの有効期限が切れた場合でも、書き込みアクセス用に指定されたファイルを同じトークンを使用してオープンする限り、そのトークンは依然有効と見なされます。

例:

- スカラー関数を使用して、 (WRITE PERMISSION ADMIN REQUIRING TOKEN FOR UPDATE および RECOVERY YES を使用して定義した) 列 COLA に DATALINK 値を挿入したとします。

```
DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','a comment')
```

スカラー関数 DLURLCOMPLETEWRITE を使用して、値を取り出します。

```
SELECT DLURLCOMPLETEWRITE(COLA)
  FROM TBLA
  WHERE ...
```

これは、以下の値を戻します。

```
HTTP://DLFS.ALMADEN.IBM.COM/x/y/****************;a.b
```

********** は、書き込みトークンを表します。

上記の値を使用して、ファイルの内容を見つけ、更新します。以下の SQL UPDATE ステートメントを発行し、ファイル変更をバックアウトして、以前にコミットされたバージョンにリストアします。

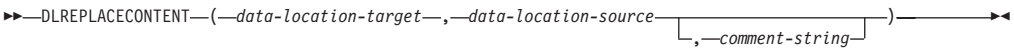
```
UPDATE TBLA
  SET COLA = DLPREVIOUSCOPY('http://dlfs.almaden.ibm.com/x/y/********
    ********;a.b', 1)
  WHERE ...
```

********** は、 URL 値によって参照されたファイルを変更する際に使用したのと同じ書き込みトークンです。 WRITE PERMISSION ADMIN NOT REQUIRING TOKEN FOR UPDATE を使用して定義した COLA の場合、上記の例の書き込みトークンは必要ありません。

- 2 番目の引き数の値 (*has-token*) は、以下の CASE ステートメントによって置換されます。 URL 値が *url_file* という変数に入っていると想定します。以下の SQL UPDATE ステートメントを発行し、ファイル変更をバックアウトして、以前にコミットされたバージョンにリストアします。

```
EXEC SQL UPDATE TBLA
  SET COLA = DLPREVIOUSCOPY(:url_file,
    (CASE
      WHEN LENGTH(:url_file) = LENGTH(DLURLCOMPLETEONLY(COLA))
      THEN 0
      ELSE 1
     END))
  WHERE ...
```

DLREPLACECONTENT



スキーマは SYSIBM です。

DLREPLACECONTENT 関数は DATALINK 値を戻します。この関数は、UPDATE ステートメント内の SET 文節の右辺にあるか、または INSERT ステートメント内の VALUES 文節にあるとき、戻された値を割り当てることによって、ファイルの内容を別のファイルで置き換え、その後そのファイルへのリンクを作成します。実際のファイル置換プロセスは、現行トランザクションのコミット処理中に行われます。

data-location-target
完全な URL 値の入った可変長文字ストリングを指定する VARCHAR(200) 式。

data-location-source
URL フォーマットのファイルのデータ・ロケーションを指定する VARCHAR 式。 UPDATE または INSERT ステートメントでの割り当ての結果として、このファイルは、*data-location-target* の指すファイル名に名前変更されます。ターゲット・ファイルの所有権および許可属性は保存されます。

data-location-source は以下のいずれかにしかならない、という制限があります。

- ゼロ長ストリング
- NULL 値
- *data-location-target* の値に接尾部のストリングを加えたもの。接尾部のストリングは 20 文字までです。接尾部ストリングの文字は URL 文字セットに属していなければなりません。さらに、UNC スキームではストリングに 『¥』 を入れることはできず、その他の有効スキームではストリングに 『/』 を入れることはできません (SQLSTATE 428D1)。

comment-string
コメントまたは追加のロケーション情報の入ったオプションの VARCHAR 値。

この関数の結果は DATALINK 値となります。いずれかの引き数が NULL になる可能性がある場合、結果も NULL になる可能性があります。 *data-location-target* が NULL であれば、結果は NULL 値です。

data-location-source が NULL、ゼロ長ストリング、または *data-location-target* と全く同じの場合、 DLREPLACECONTENT は DLVALUE と同じ効果を示します。

例:

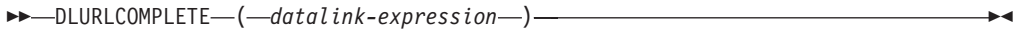
- リンクされたファイルの内容を別のファイルと置き換えます。以下の INSERT ステートメントを使用して、表 TBLA にある列 PICT_FILE に DATALINK 値を挿入したとします。

```
EXEC SQL INSERT INTO TBLA (PICT_ID, PICT_FILE)
  VALUES(1000, DLVALUE('HTTP://HOSTA.COM/dlfs/image-data/pict1.gif'));
```

  以下の SQL UPDATE ステートメントを発行して、このファイルの内容を別のファイルに置き換えます。

```
EXEC SQL UPDATE TBLA
  SET PICT_FILE =
    DLREPLACECONTENT('HTTP://HOSTA.COM/dlfs/image-data/pict1.gif',
                     'HTTP://HOSTA.COM/dlfs/image-data/pict1.gif.new')
  WHERE PICT_ID = 1000;
```

DLURLCOMPLETE



DLURLCOMPLETE 関数は、リンク・タイプの URL を持つ DATALINK 値から、データ位置属性を戻します。 *datalink-expression* が、属性 READ PERMISSION DB で定義された DATALINK 列であると、この値にはファイル・アクセス・トークンが入っています。

引き数は、 DATALINK のデータ型の値を戻す式でなければなりません。

関数の結果は VARCHAR(254) です。引き数が NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引き数が NULL 値であれば、結果は NULL 値です。

DATALINK 値にコメントしか組み込まれていない場合、長さがゼロのストリングが戻されます。

例:

- スカラー関数を使用して、表 TBLA にある行の列 COLA に DATALINK 値を挿入したとします。

  ```
  DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','a comment')
  ```

  その値を次の関数で処理します。

  ```
  DLURLCOMPLETE(COLA)
  ```

  上記は以下を戻します。

  ```
  HTTP://DLFS.ALMADEN.IBM.COM/x/y/****************;a.b
  ```

  ここで ********** はアクセス・トークンを表します。

DLURLCOMPLETEONLY

スキーマは SYSIBM です。

DLURLCOMPLETEONLY 関数は、リンク・タイプの URL を持つ DATALINK 値から、データ位置属性を戻します。戻される値にはファイル・アクセス・トークンは決して入っていません。

引き数は、 DATALINK のデータ型の値を戻す式でなければなりません。

関数の結果は VARCHAR(254) です。引き数が NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引き数が NULL 値であれば、結果は NULL 値です。

DATALINK 値にコメントしか組み込まれていない場合、長さがゼロのストリングです。

例:

- スカラー関数を使用して、表 TBLA にある DATALINK 列 COLA (READ PERMISSION DB を指定して定義された) に DATALINK 値を挿入したとします。

  ```
  DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','a comment')
  ```

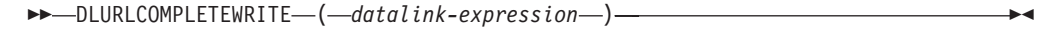
  その値を次の関数で処理します。

  ```
  DLURLCOMPLETEONLY(COLA)
  ```

  次の値を戻します。

  ```
  HTTP://DLFS.ALMADEN.IBM.COM/x/y/a.b
  ```

DLURLCOMPLETEWRITE



スキーマは SYSIBM です。

DLURLCOMPLETEWRITE 関数は、リンク・タイプの URL を持つ DATALINK 値から、完全な URL 値を戻します。 *datalink-expression* から生成された DATALINK 値が WRITE PERMISSION ADMIN を指定して定義された DATALINK 列から取得された場合、書き込みトークンが戻り値に組み入れられます。戻り値を使用して、リンク先ファイルを見付け、それを更新することができます。

DATALINK 列が他の WRITE PERMISSION オプション (ADMIN ではない) または NO LINK CONTROL を指定して定義された場合、DLURLCOMPLETEWRITE は書き込みトークンのない URL 値だけを戻します。ファイル参照が WRITE PERMISSION FS を指定して定義された DATALINK 列から派生した場合、ファイル許可はファイル・システムによって制御されるので、ファイルに書き込むためにトークンは必要ありません。ファイル参照が WRITE PERMISSION BLOCKED を指定して定義された DATALINK 列から派生した場合、ファイルにはまったく書き込むことができません。

引き数は、 DATALINK のデータ型の値を戻す式でなければなりません。

関数の結果は VARCHAR(254) です。引き数が NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引き数が NULL 値であれば、結果は NULL 値です。

DATALINK 値にコメントしか組み込まれていない場合、長さがゼロのストリングです。

例:

- スカラー関数を使用して、表 TBLA にある DATALINK 列 COLA (WRITE PERMISSION ADMIN を指定して定義された) に DATALINK 値を挿入したとします。

  ```
  DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','a comment')
  ```

  その値を次の関数で処理します。

  ```
  DLURLCOMPLETEWRITE(COLA)
  ```

  次の値を戻します。

  ```
  HTTP://DLFS.ALMADEN.IBM.COM/x/y/****************;a.b
  ```

  ここで ********** は書き込みトークンを表します。 COLA が WRITE PERMISSION ADMIN を指定して定義された場合、書き込みトークンは存在しません。

DLURLPATH

スキーマは SYSIBM です。

DLURLPATH 関数は、特定サーバー内のファイルにアクセスするために必要なパスおよびファイル名を、リンク・タイプが URL の DATALINK 値から戻します。*datalink-expression* が、属性 READ PERMISSION DB で定義された DATALINK 列であると、この値にはファイル・アクセス・トークンが入っています。

引き数は、 DATALINK のデータ型の値を戻す式でなければなりません。

関数の結果は VARCHAR(254) です。引き数が NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引き数が NULL 値であれば、結果は NULL 値です。

DATALINK 値にコメントしか組み込まれていない場合、長さがゼロのストリングが戻されます。

例:

- スカラー関数を使用して、表 TBLA にある行の列 COLA に DATALINK 値を挿入したとします。

  ```
  DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','a comment')
  ```

  そして、その値を次の関数で処理します。

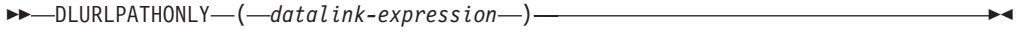
  ```
  DLURLPATH(COLA)
  ```

  次の値が戻されます。

  ```
  /x/y/****************;a.b
  ```

  (ここで ********** はアクセス・トークンを表します)

DLURLPATHONLY



スキーマは SYSIBM です。

DLURLPATHONLY 関数は、特定サーバー内のファイルにアクセスするために必要なパスおよびファイル名を、リンク・タイプが URL の DATALINK 値から戻します。戻される値にはファイル・アクセス・トークンは入っていません。

引き数は、 DATALINK のデータ型の値を戻す式でなければなりません。

関数の結果は VARCHAR(254) です。引き数が NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引き数が NULL 値であれば、結果は NULL 値です。

DATALINK 値にコメントしか組み込まれていない場合、長さがゼロのストリングが戻されます。

例:

- スカラー関数を使用して、表 TBLA にある行の列 COLA に DATALINK 値を挿入したとします。

  ```
  DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','a comment')
  ```

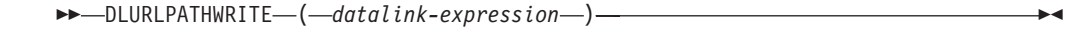
  そして、その値を次の関数で処理します。

  ```
  DLURLPATHONLY(COLA)
  ```

  次の値が戻されます。

  ```
  /x/y/a.b
  ```

DLURLPATHWRITE



スキーマは SYSIBM です。

DLURLPATHWRITE 関数は、特定サーバー内のファイルにアクセスするために必要なパスおよびファイル名を、リンク・タイプが URL の DATALINK 値から戻します。 *datalink_expression* から生成される DATALINK 値が WRITE PERMISSION ADMIN を使用して定義した DATALINK 列から、戻される値の中に書き込みトークンが入っています。

DATALINK 列が他の WRITE PERMISSION オプション (ADMIN ではない)、または NO LINK CONTROL を使用して定義した場合、 DLURLPATHWRITE は書き込みトークンを添付しないでパスとファイル名を戻します。 WRITE PERMISSION FS を使用して定義した DATALINK 列からファイルが参照される場合、ファイルに書き込むためのトークンは必要ありません。なぜなら、書き込み許可は、ファイル・システムによって制御されるからです。 WRITE PERMISSION BLOCKED を使用して定義した DATALINK 列からファイルが参照される場合、ファイルの書き込みは全く行えません。

引き数は、 DATALINK のデータ型の値を戻す式でなければなりません。

関数の結果は VARCHAR(254) です。引き数が NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引き数が NULL 値であれば、結果は NULL 値です。

DATALINK 値にコメントしか組み込まれていない場合、長さがゼロのストリングです。

例:

- スカラー関数を使用して、 (WRITE PERMISSION ADMIN を使用して定義した) DATALINK 列 COLA に DATALINK 値を挿入したとします。

  ```
  DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','a comment')
  ```

  その値を次の関数で処理します。

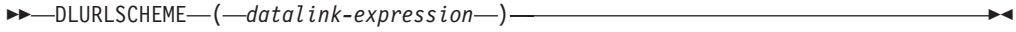
  ```
  DLURLPATHWRITE(COLA)
  ```

  これは、以下のものを戻します。

  ```
  /x/y/****************;a.b
  ```

  ********** は、書き込みトークンを表します。 COLA の定義で WRITE PERMISSION ADMIN を使用していない場合、書き込むトークンは存在しません。

DLURLSCHEME



スキーマは SYSIBM です。

DLURLSCHEME 関数は、リンク・タイプが URL の DATALINK 値からスキームを戻します。この値は必ず大文字になります。

引き数は、 DATALINK のデータ型の値を戻す式でなければなりません。

関数の結果は VARCHAR(20) になります。引き数が NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引き数が NULL 値であれば、結果は NULL 値です。

DATALINK 値にコメントしか組み込まれていない場合、長さがゼロのストリングが戻されます。

例:

- スカラー関数を使用して、表 TBLA にある行の列 COLA に DATALINK 値を挿入したとします。

  ```
  DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','a comment')
  ```

  そして、その値を次の関数で処理します。

  ```
  DLURLSCHEME(COLA)
  ```

  次の値が戻されます。

  ```
  HTTP
  ```

DLURLSERVER

スキーマは SYSIBM です。

DLURLSERVER 関数は、リンク・タイプが URL の DATALINK 値からファイル・サーバーを戻します。この値は必ず大文字になります。

引き数は、 DATALINK のデータ型の値を戻す式でなければなりません。

関数の結果は VARCHAR(254) です。引き数が NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引き数が NULL 値であれば、結果は NULL 値です。

DATALINK 値にコメントしか組み込まれていない場合、長さがゼロのストリングが戻されます。

例:

- スカラー関数を使用して、表 TBLA にある行の列 COLA に DATALINK 値を挿入したとします。

  ```
  DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','a comment')
  ```

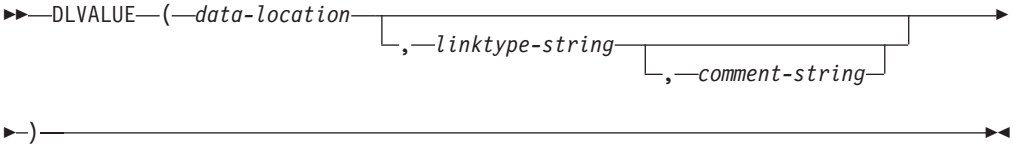
  そして、その値を次の関数で処理します。

  ```
  DLURLSERVER(COLA)
  ```

  次の値が戻されます。

  ```
  DLFS.ALMADEN.IBM.COM
  ```

DLVALUE



スキーマは SYSIBM です。

DLVALUE 関数は DATALINK 値を戻します。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。この関数は、UPDATE ステートメント内の SET 文節の右辺にあるか、または INSERT ステートメント内の VALUES 文節にあるとき、通常ファイルへのリンクも作成します。ただし、コメントだけが指定されている (data-location がゼロ長ストリングである) 場合、 DATALINK 値はリンケージ属性が空のまま作成されるため、ファイル・リンクはありません。

data-location
リンク・タイプが URL である場合、これは完全 URL 値の入った可変長文字ストリングになる式です。

linktype-string
DATALINK 値のリンク・タイプを指定する任意指定の VARCHAR 式。 'URL' (SQLSTATE 428D1) だけが有効な値です。

comment-string
コメントまたは追加の位置情報を提供する任意指定の VARCHAR(254) 値。 *data-location* の長さに *comment-string* を加えたものは 200 バイトを超えてはなりません。

この関数の結果は DATALINK 値となります。 DLVALUE 関数のいずれかの引き数が NULL 値の可能性がある場合、結果も NULL 値になる可能性があります。 *data-location* が NULL 値の場合、その結果は NULL 値になります。

この関数を使用して DATALINK 値を定義する際には、ターゲット値の最大長を考慮してください。たとえば、列が DATALINK(200) で定義されている場合、 *data-location* に *comment* を加えた最大長は 200 バイトとなります。

例:

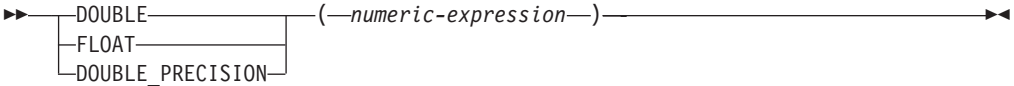
- 1 つの行を表に挿入します。最初の 2 つのリンクの URL 値は、 url_article および url_snapshot という名前の変数内に入っています。 url_snapshot_comment という名前の変数には、スナップショット・リンクに付随するコメントが入っています。現時点では、ムービー用のリンクはなく、 url_movie_comment という名前の変数にコメントだけを組み入れることができます。

```
EXEC SQL
  INSERT INTO HOCKEY_GOALS
    VALUES('Maurice Richard',
           'Montreal Canadien',
           '?',
           'Boston Bruins,
           '1952-04-24',
           'Winning goal in game 7 of Stanley Cup final',
```

```
DLVALUE(:url_article),
DLVALUE(:url_snapshot, 'URL', :url_snapshot_comment),
DLVALUE('', 'URL', :url_movie_comment) );
```

DOUBLE

数値 → 倍精度:



文字ストリング → 倍精度:

スキーマは SYSIBM です。ただし、DOUBLE(*string-expression*) のスキーマは SYSFUN です。

DOUBLE 関数は、以下に対応する浮動小数点数を戻します。

- 引き数が数式の場合は、数値。
- 引き数が文字ストリング式の場合は、数値の文字ストリング表記。

Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

数値 → 倍精度

numeric-expression
引き数は、組み込み数値データ型の値を返す式です。

関数の結果は倍精度浮動小数点数になります。引き数が NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引き数が NULL 値であれば、結果は NULL 値です。

結果は、引き数が倍精度浮動小数点の列、または変数に割り当てられた場合の結果と同じ数値になります。

文字ストリング → 倍精度

string-expression
引き数のタイプは、数値定数形式の CHAR または VARCHAR にすることができます。引き数に先行ブランクや後続ブランクがあっても、それは無視されます。

関数の結果は倍精度浮動小数点数になります。結果は NULL 値になることがあります。引き数が NULL 値である場合、その結果は NULL 値になります。

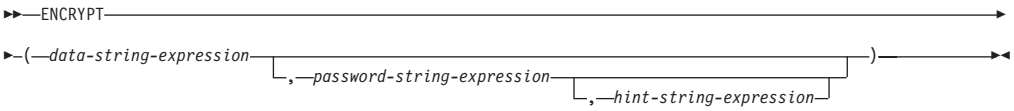
結果は、ストリングが定数と見なされ、倍精度浮動小数点の列または変数に割り当てられる場合の結果と同じ数値になります。

例:

EMPLOYEE 表を使用して、歩合がゼロではない従業員の給与と歩合の比率を計算します。関係する列 (SALARY と COMM) のデータ型は DECIMAL です。結果が範囲外にならないようにするため、DOUBLE を SALARY に適用して、除算が浮動小数点数で実行されるようにします。

```
SELECT EMPNO, DOUBLE(SALARY)/COMM
  FROM EMPLOYEE
  WHERE COMM > 0
```

ENCRYPT



スキーマは SYSIBM です。

ENCRYPT 関数は、*data-string-expression* 暗号化の結果である値を返します。暗号化解除に使用されるパスワードは、 *password-string-expression* か (SET ENCRYPTION PASSWORD ステートメントで割り当てられた) ENCRYPTION PASSWORD 値のいずれかです。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

data-string-expression
暗号化する CHAR または VARCHAR 値を返す式です。 *data-string-expression* のデータ型の長さ属性は、 *hint-string-expression* 引き数がなければ 32663 に、*hint-string-expression* 引き数が指定されていれば 32631 に制限されています (SQLSTATE 42815)。

password-string-expression
少なくとも 6 バイトで 127 バイトを超えない CHAR または VARCHAR 値を返す式です (SQLSTATE 428FC)。この値は、*data-string-expression* を暗号化するために使用されるパスワードを表します。パスワード引き数が NULL、または与えられていない場合、セッションに設定されていなければならない ENCRYPTION PASSWORD 値を使用してデータは暗号化されます (SQLSTATE 51039)。

hint-string-expression
データ所有者がパスワードを思い出す (たとえば、'Ocean' が 'Pacific' を思い出すヒントになります) ために役立つ、32 バイトまでの CHAR または VARCHAR 値を返す式です。ヒントの値が与えられると、そのヒントは結果に組み込まれ、GETHINT 関数を使用して取り出すことができます。この引き数が NULL、または与えられていない場合、ヒントは結果に組み込まれません。

関数の結果データ型は VARCHAR FOR BIT DATA です。

結果の長さ属性:

- オプションのヒント・パラメーターが指定されている場合、非暗号化データの長さ属性 + 8 バイト + 次の 8 バイト境界までのバイト数 + ヒントの長さ 32 バイト
- ヒント・パラメーターがない場合、非暗号化データの長さ属性 + 8 バイト + 次の 8 バイト境界までのバイト数

最初の引き数を NULL にすることができる場合、結果も NULL にすることができます。最初の引き数が NULL であれば、結果は NULL 値になります。

暗号化された結果が *data-string-expression* 値よりも長くなることに注意してください。そのため、暗号化された値を割り当てる際、その暗号化された値全体が入るだけの十分なサイズでターゲットを宣言してください。

注:

- ***暗号化アルゴリズム:*** 使用されている内部暗号化アルゴリズムは埋め込み付きの RC2 ブロック暗号で、128 ビットの秘密鍵は MD2 メッセージ・ダイジェストによってパスワードから派生します。
- ***暗号化パスワードおよびデータ:*** パスワードの管理はユーザーの責任です。データが暗号化されると、そのデータを暗号化解除するために使用できるのは、暗号化に使用したパスワードだけです (SQLSTATE 428FD)。ブランクで埋め込むことが可能なため、CHAR 変数を使用してパスワードを設定する際には注意してください。暗号化された結果には、NULL 終止符およびその他の表示不可文字が組み込まれていることがあります。
- ***表列定義:*** 暗号化データを組み込む列およびタイプを定義するとき、以下のようにして常に長さ属性を計算してください。

  ヒントのない暗号化データの場合:

  非暗号化データの最大長 + 8 バイト + 次の 8 バイト境界までのバイト数 = 暗号化データの列の長さ

  ヒントが組み込まれた暗号化データの場合:

  非暗号化データの最大長 + 8 バイト + 次の 8 バイト境界までのバイト数 + ヒントの長さ 32 バイト = 暗号化データの列の長さ

  推奨されているデータ長よりも短い長さへの割り当てまたは cast は、暗号化解除の失敗やデータ脱落の原因となります。ブランクは、短すぎる列に保管するときに切り捨てられる有効な暗号化データ値です。

  以下に、列の長さの計算例をいくつか示します。

  ```
  非暗号データの最大長                    6 バイト
  8 バイト                                8 バイト
  次の 8 バイト境界までのバイト数         2 バイト
                                        ---------
  暗号化データの列の長さ                 16 バイト

  非暗号データの最大長                   32 バイト
  8 バイト                                8 バイト
  次の 8 バイト境界までのバイト数         8 バイト
                                        ---------
  暗号化データの列の長さ                 48 バイト
  ```

- ***暗号化データの管理:*** 暗号化データは、 ENCRYPT 関数に対応する暗号化解除関数をサポートしているサーバーでのみ暗号化解除できます。そのため、暗号化データの入った列の複製は、 DECRYPT_BIN または DECRYPT_CHAR 関数をサポートしているサーバーで行わなければなりません。

例:

例 1 : この例では、暗号化パスワードを保持するために ENCRYPTION PASSWORD 値が使用されています。

```
SET ENCRYPTION PASSWORD = 'Ben123';
INSERT INTO EMP(SSN) VALUES ENCRYPT('289-46-8832');
```

例 2 : この例では、暗号化パスワードが明示的に渡されています。

```
INSERT INTO EMP(SSN) VALUES ENCRYPT('289-46-8832','Ben123');
```

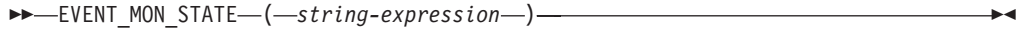
例 3 : 暗号化パスワード 'Pacific' を思い浮かべることができるよう、ヒント 'Ocean' が保管されます。

```
INSERT INTO EMP(SSN) VALUES ENCRYPT('289-46-8832','Pacific','Ocean');
```

関連資料:

- 330 ページの『DECRYPT_BIN および DECRYPT_CHAR』
- 363 ページの『GETHINT』

EVENT_MON_STATE



スキーマは SYSIBM です。

EVENT_MON_STATE 関数は、イベント・モニターの現在の状態を戻します。

引き数は、結果のタイプが CHAR または VARCHAR で、値がイベント・モニターの名前であるストリング式です。指定したイベント・モニターが SYSCAT.EVENTMONITORS カタログ表にない場合は、 SQLSTATE 42704 が戻されます。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

結果は、次のいずれかの値の整数になります。

0 イベント・モニターは非アクティブ状態です。

1 イベント・モニターはアクティブです。

引き数が NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引き数が NULL 値であれば、結果は NULL 値です。

例:

次の例は、定義済みのイベント・モニターすべてを選択して、各モニターがアクティブか非アクティブかを示すものです。

```
SELECT EVMONNAME,
  CASE
    WHEN EVENT_MON_STATE(EVMONNAME) = 0 THEN 'Inactive'
    WHEN EVENT_MON_STATE(EVMONNAME) = 1 THEN 'Active'
  END
  FROM SYSCAT.EVENTMONITORS
```

EXP

スキーマは SYSFUN です。

引き数に対する指数関数値を戻します。

引き数は、任意の組み込み数値データ型にすることができます。引き数は、関数での処理に必要な倍精度浮動小数点数に変換されます。

関数の結果は倍精度浮動小数点数になります。結果は NULL 値になることがあります。引き数が NULL 値である場合、その結果は NULL 値になります。

FLOAT

スキーマは SYSIBM です。

FLOAT 関数は、数値の浮動小数点数表記を戻します。 FLOAT は DOUBLE の同義語です。

関連資料:

• 354 ページの『DOUBLE』

FLOOR

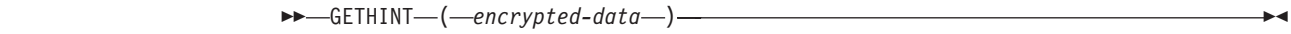
スキーマは SYSIBM です。 (FLOOR 関数の SYSFUN バージョンは引き続き使用可能です。)

引き数よりも小さいか等しい整数で、最大の整数値を戻します。

引き数が DECIMAL の場合は位取りは 0 になる以外は、関数の結果のデータ型と長さ属性は、引き数と同じになります。たとえば、データ型が DECIMAL(5,5) の引き数は DECIMAL(5,0) を戻します。

引き数が NULL になる可能性があるか、またはデータベース構成パラメーターで DFT_SQLMATHWARN が YES に設定されている場合には、結果は NULL になる可能性があります。引き数が NULL の場合、結果は NULL 値になります。

GETHINT



スキーマは SYSIBM です。

パスワード・ヒントが *encrypted-data* 内で見つかった場合、 GETHINT 関数をそのヒントを返します。パスワード・ヒントとは、データ所有者がパスワードを思い出すのに役立つ語句です。たとえば 'Ocean' は、'Pacific' を思い出すヒントになります。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

encrypted-data
完全な、暗号化データ・ストリングである CHAR FOR BIT DATA または VARCHAR FOR BIT DATA 値を返す式です。データ・ストリングは、ENCRYPT 関数を使って暗号化されたものでなければなりません (SQLSTATE 428FE)。

関数の結果は VARCHAR(32) です。結果は NULL になることがあります。ヒント・パラメーターが ENCRYPT 関数によって *encrypted-data* に追加されなかった場合、または最初の引き数が NULL の場合には、結果が NULL 値になります。

例:

この例では、暗号化パスワード 'Pacific' を思い出すことができるよう、ヒント 'Ocean' が保管されます。

```
INSERT INTO EMP (SSN) VALUES ENCRYPT('289-46-8832', 'Pacific','Ocean');
SELECT GETHINT(SSN)
   FROM EMP;
```

返される値は 'Ocean' です。

関連資料:

- 330 ページの『DECRYPT_BIN および DECRYPT_CHAR』
- 356 ページの『ENCRYPT』

GENERATE_UNIQUE

スキーマは SYSIBM です。

GENERATE_UNIQUE 関数は、同一関数の他の実行と比較してユニークである、 13 バイト長のビット・データ文字ストリング (CHAR(13) FOR BIT DATA) を戻します。(システムの刻時機構は、関数が実行されるパーティション番号とともに、内部の世界標準時 (UTC) タイム・スタンプを生成するのに使用されます。実際のシステム刻時機構をリバースへ動かす調整を行うと、値が重複する場合があります。) この関数は、deterministic 関数ではないものと定義されます。

この関数には引き数がありません (空の括弧を指定する必要があります)。

関数の結果は、内部形式の世界標準時 (UTC)、および関数が処理されたパーティション番号から成るユニーク値になります。結果が NULL 値になることはありません。

この関数の結果を使用して、表のユニーク値を用意することができます。後続の各値は直前の値より大きくなり、表で使用できる順序列を提供します。値には、関数が実行されたパーティションの番号が組み込まれ、それにより、複数のパーティションにまたがってパーティション化された表もある順序列のユニーク値を持つことになります。この順序列は、関数が実行された時刻に基づいています。

この関数は、特殊レジスター CURRENT TIMESTAMP を使用する場合とは異なります。この特殊レジスターの場合、複数行の挿入ステートメントまたは全選択を伴う挿入ステートメントの各行についてユニーク値が生成されます。

この関数の結果の一部であるタイム・スタンプ値は、 GENERATE_UNIQUE の結果を引き数にする TIMESTAMP スカラー関数を使用して決定することができます。

例:

- 行ごとにユニークな列から成る表を作成します。 GENERATE_UNIQUE 関数を使用してこの列を移植します。 UNIQUE_ID 列には、列をビット・データ文字ストリングとして識別するために ″FOR BIT DATA″ が指定されていることに注意してください。

  ```
  CREATE TABLE EMP_UPDATE
    (UNIQUE_ID CHAR(13) FOR BIT DATA,
    EMPNO CHAR(6),
    TEXT VARCHAR(1000))
    INSERT INTO EMP_UPDATE
      VALUES (GENERATE_UNIQUE(), '000020', 'Update entry...'),
      (GENERATE_UNIQUE(), '000050', 'Update entry...')
  ```

  この表には、行ごとにユニークな ID があります。ただし、UNIQUE_ID 列が、常に GENERATE_UNIQUE を使用して設定されている場合です。これは、表にトリガーを導入することによって行うことができます。

  ```
  CREATE TRIGGER EMP_UPDATE_UNIQUE
    NO CASCADE BEFORE INSERT ON EMP_UPDATE
    REFERENCING NEW AS NEW_UPD
    FOR EACH ROW
    SNEW_UPD.UNIQUE_ID = GENERATE_UNIQUE()
  ```

このトリガーを定義すると、以下のように最初の列を指定せずに上記の INSERT ステートメントを出すことができます。

```
INSERT INTO EMP_UPDATE (EMPNO, TEXT)
  VALUES ('000020', 'Update entry 1...'),
  ('000050', 'Update entry 2...')
```

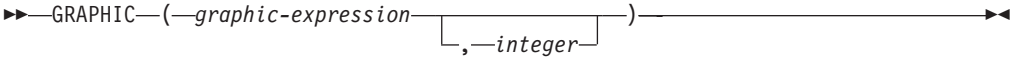
行が EMP_UPDATE に追加された時点のタイム・スタンプ (UTC における) は、以下を使用して戻すことができます。

```
SELECT TIMESTAMP (UNIQUE_ID), EMPNO, TEXT
  FROM EMP_UPDATE
```

したがって、表内のタイム・スタンプ列を行の挿入時に記録する必要はありません。

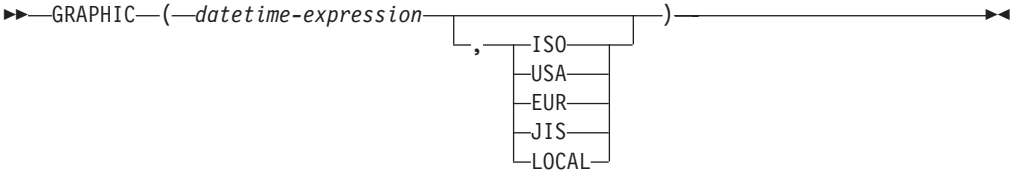
GRAPHIC

GRAPHIC → GRAPHIC:



文字 → GRAPHIC:

日付/時刻→GRAPHIC:



スキーマは SYSIBM です。

GRAPHIC 関数は、以下を表す固定長の GRAPHIC ストリングを戻します。

- GRAPHIC ストリング (最初の引き数がいずれかのタイプの GRAPHIC ストリングの場合)
- 日付/時刻 (Unicode データベースのみ) (最初の引き数が日付、時刻、またはタイム・スタンプの場合)

Unicode データベースでは、指定した引き数が文字ストリングであると、まず GRAPHIC ストリングに変換されてから、関数が実行されます。最後の文字が高サロゲートになるように出力ストリングが切り捨てられた場合、そのサロゲートはブランク文字 (X'0020') に変換されます。この動作を過信しないでください。今後のリリースで変更される可能性があるからです。

この関数の結果は、固定長 GRAPHIC ストリング (GRAPHIC データ型) になります。最初の引き数を NULL にすることができる場合、結果も NULL にすることができます。最初の引き数が NULL であれば、結果は NULL 値になります。

GRAPHIC → GRAPHIC:

graphic-expression
GRAPHIC ストリング値を戻す式。

integer
結果の GRAPHIC データ型の長さ属性を指定する整数値。値は 1 から 127 の範囲でなければなりません。値を指定しない場合の結果の長さ属性は、最初の引き数の長さ属性と同じになります。

文字→ GRAPHIC:

character-expression
LONG VARCHAR または CLOB 以外の文字ストリングのデータ型である必要のある値をもち、しかも 16 336 バイトを最大長としてもつ式。

結果の長さ属性は、引き数の長さ属性と同じになります。

日付/時刻 → GRAPHIC

datetime-expression
次の 3 つのデータ型のいずれかの式。

日付 (date)
結果は、2 番目の引き数によって指定された形式の日付の GRAPHIC ストリング表記になります。結果の長さは 10 文字です。 2 番目の引き数が指定され、その値が有効な値でない場合には、エラーが戻されます (SQLSTATE 42703)。

時刻 (time)
結果は、2 番目の引き数によって指定された形式の時刻の GRAPHIC ストリング表記になります。結果の長さは 8 文字です。 2 番目の引き数が指定され、その値が有効な値でない場合には、エラーが戻されます (SQLSTATE 42703)。

タイム・スタンプ (timestamp)
結果は、タイム・スタンプの GRAPHIC ストリング表記になります。結果の長さは 26 文字です。2 番目の引き数は適用されないので、指定してはなりません (SQLSTATE 42815)。

ストリングのコード・ページは、アプリケーション・サーバーのデータベースのコード・ページになります。

関連資料:

• 464 ページの『VARGRAPHIC』

HASHEDVALUE

スキーマは SYSIBM です。

HASHEDVALUE 関数は、パーティション関数を行の区分化キー値に適用することによって入手された行の区分化マップ索引を戻します。 たとえば、SELECT 文節で使用すると、その SELECT ステートメントの結果の生成に使用された表の各行の区分化マップ索引を戻します。

遷移変数および表に戻される区分化マップ索引は、区分化キー列の現行遷移値から派生します。たとえば、挿入前トリガーにおいて、新しい遷移変数の現行値があれば、関数は予定区分化マップ索引を戻します。ただし、区分化キー列の値はそれ以後の挿入前トリガーによって変更される場合があります。したがって、データベースに挿入される時点で、行の最終区分化マップ索引は予定値と異なるかもしれません。

引き数は、表内の列の修飾された名前または無修飾の名前でなければなりません。該当の列は、どのようなデータ型であっても構いません。 (この関数は、ユーザー定義関数の作成時にソース関数として使用することはできません。この関数は、すべてのデータ型を引き数として受け入れるので、ユーザー定義特殊タイプをサポートするための追加のシグニチャーを作成する必要はありません。 *column-name* がビューの列を参照する場合、その列のビューの式は基本表の列を参照する必要があり、そのビューは削除可能でなければなりません。ネストされているか、または共通の表式は、ビューと同じ規則に従います。

HASHEDVALUE 関数によって区分化マップ索引が戻される特定の行 (または表) は、この関数を使用する SQL ステートメントのコンテキストから判別されます。

結果のデータ型は、0 から 4095 の範囲の INTEGER です。区分化キーのない表の場合、結果は常に 0 になります。 NULL 値が戻されることはありません。行レベルの情報が戻されるので、どの列が表に指定されるかに関係なく、結果は同じです。

HASHEDVALUE 関数は、複製された表、チェック制約内、または生成された列の定義で使用することはできません (SQLSTATE 42881)。

バージョン 8 より前の旧バージョンとの互換性を保つために、 HASHEDVALUE を関数名 PARTITION に置き換えてもかまいません。

例:

- 区分化マップ索引が 100 であるすべての行について、 EMPLOYEE 表から従業員番号 (EMPNO) をリストします。

```
SELECT EMPNO FROM EMPLOYEE
  WHERE HASHEDVALUE(PHONENO) = 100
```

- 表 EMPLOYEE で前トリガーを作成して、従業員の挿入の際には必ず、EMPINSERTLOG2 という表に従業員番号と新しい行の予定区分化マップ索引を記録します。

```
CREATE TRIGGER EMPINSLOGTRIG2
  BEFORE INSERT ON EMPLOYEE
  REFERENCING NEW AW NEWTABLE
  FOR EACH ROW
  INSERT INTO EMPINSERTLOG2
    VALUES(NEWTABLE.EMPNO, HASHEDVALUE(NEWTABLE.EMPNO))
```

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE VIEW ステートメント』

HEX

スキーマは SYSIBM です。

HEX 関数は、値の 16 進表記を文字ストリングとして戻します。

引き数には、最大長 16 336 バイトの任意の組み込みデータ型の値である式を使うことかできます。

この関数の結果は文字ストリングです。引き数が NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引き数が NULL 値である場合、結果は NULL 値です。

コード・ページはデータベース・コード・ページになります。

結果は、16 進数字のストリングです。最初の 2 つは引き数の最初のバイト、次の 2 つは引き数の 2 番目のバイトを表します。以下同様です。引き数が日付/時刻値または数値である場合、結果は引き数の内部形式の 16 進表記になります。戻される 16 進表記は、関数が実行されるアプリケーション・サーバーによって異なる場合があります。違いが生じる場合としては、次のような場合があります。

- EBCDIC サーバーに対する ASCII クライアントまたは ASCII サーバーに対する EBCDIC クライアントで、文字ストリング引き数を指定して HEX 関数を実行したとき。
- クライアント・システムとサーバー・システムとで数値のバイト・オーダーが異なる場合に、 HEX 関数に数値引き数を指定したとき (場合によります)。

結果のタイプと長さは、文字ストリング引き数のタイプと長さによって異なります。

- 文字ストリング
  - 127 以下の固定長
    - 結果は、引き数について定義されている長さの 2 倍の長さの固定長文字ストリングになります。
  - 127 を超える固定長
    - 結果は、引き数について定義されている長さの 2 倍の長さの可変長文字ストリングになります。
  - 可変長
    - 結果は、引き数について定義されている最大長さの 2 倍を最大長とする可変長文字ストリングになります。
- GRAPHIC ストリング
  - 63 以下の固定長
    - 結果は、引き数について定義されている長さの 4 倍の長さの固定長文字ストリングになります。
- 63 を超える固定長
  - 結果は、引き数について定義されている長さの 4 倍の長さの可変長文字ストリングになります。

- 可変長
  - 結果は、引き数について定義されている最大の長さの 4 倍を最大長とする可変長文字ストリングになります。

例:

以下の例では、DB2 for AIX アプリケーション・サーバーを使用することを前提にしています。

- DEPARTMENT 表を使用して、ホスト変数 HEX_MGRNO (char(12)) に、'PLANNING' 部門 (DEPTNAME) の管理者番号 (MGRNO) の 16 進表記を設定します。

```
SELECT HEX(MGRNO)
 INTO :HEX_MGRNO
 FROM DEPARTMENT
 WHERE DEPTNAME = 'PLANNING'
```

  サンプル表を使用した場合、HEX_MGRNO は '303030303230' に設定されます (文字値は '000020')。
- COL_1 が、データ型 char(1)、値 'B' の列であるとします。英字 'B' の 16 進数表記は X'42' です。 HEX(COL_1) は 2 文字のストリング '42' を戻します。
- COL_3 が、データ型 decimal(6,2)、値 40.1 の列であるとします。 10 進数値 40.1 の内部表記に HEX 関数を適用した結果は、 8 文字のストリング '0004010C' になります。

HOUR

スキーマは SYSIBM です。

HOUR 関数は、値の時の部分を戻します。

引き数は、時刻、タイム・スタンプ、時刻期間、タイム・スタンプ期間であるか、または CLOB でも LONG VARCHAR でもない時刻またはタイム・スタンプの有効な文字ストリング表記でなければなりません。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

この関数の結果は長精度整数 (large integer) です。引き数が NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引き数が NULL 値であれば、結果は NULL 値です。

その他の規則は、引き数のデータ型に応じて以下のように異なります。

- 引き数が、時刻、タイム・スタンプ、または時刻やタイム・スタンプの有効なストリング表記の場合
  - 結果は、値の時間の部分 (0 から 24 の整数) になります。
- 引き数が時刻期間またはタイム・スタンプ期間の場合
  - 結果は、値の時の部分 (-99 から 99 の整数) になります。ゼロ以外の結果の符号は、引き数と同じになります。

例:

CL_SCHED サンプル表を使用して、午後に始まるすべてのクラスを選択します。

```
SELECT * FROM CL_SCHED
 WHERE HOUR(STARTING) BETWEEN 12 AND 17
```

IDENTITY_VAL_LOCAL

スキーマは SYSIBM です。

IDENTITY_VAL_LOCAL 関数は、非 deterministic 関数であり、ID 列に割り当てられた最新の値を戻します。この割り当ては、VALUES 文節を使用した単一行 INSERT ステートメントの結果として発生したものです。関数には入力パラメーターはありません。

対応する ID 列の実際のデータ型に関係なく、結果は DECIMAL(31,0) です。

関数によって返される値は、最新の単一行 INSERT ステートメントで識別されている表の ID 列に割り当てられた値です。 INSERT ステートメントには、ID 列の入った表の VALUES 文節が入っていなければなりません。また、INSERT ステートメントは、同じレベルで発行される必要もあります。つまり、値は、次に割り当てられる値に置き換えられるまでは、割り当てられたレベルでローカルに使用できなければなりません。 (レベル、トリガーやルーチンが呼び出されるたびに新しく開始されます。)

割り当てられる値は (ID 列が GENERATED BY DEFAULT で定義される場合は) ユーザーによって提供されるか、あるいは DB2 で生成された識別値が提供されます。

VALUES 文節をともなう単一行 INSERT ステートメントが、現行処理レベルで ID 列の入った表に対して発行されていない場合は、関数は NULL 値を戻します。

関数の結果が以下のステートメントによって影響を受けることはありません。

- ID 列のない表の、VALUES 文節をともなう単一行 INSERT ステートメント
- VALUES 文節をともなう複数行 INSERT ステートメント
- fullselect を持つ INSERT ステートメント
- ROLLBACK TO SAVEPOINT ステートメント

注:

- INSERT ステートメントの VALUES 文節内の式は、INSERT ステートメントのターゲット列の割り当ての前に評価されます。そのため、INSERT ステートメントの VALUES 文節にある IDENTITY_VAL_LOCAL 関数の呼び出しでは、前の INSERT ステートメントからの ID 列として、最新の割り当て値が使用されます。 ID 列の入った表の VALUES 文節をともなう単一行 INSERT ステートメントが、 IDENTITY_VAL_LOCAL 関数と同じレベル内で実行されていない場合、関数は NULL 値を戻します。
- トリガーが定義されている表の ID 列の値は、ID 列のトリガー遷移変数を参照することにより、トリガー内で判別できます。
- 挿入トリガーのトリガー条件から IDENTITY_VAL_LOCAL 関数を呼び出した結果は、 NULL 値になります。
- 複数の BEFORE または AFTER 挿入トリガーが 1 つの表について存在することが可能です。この場合、各トリガーは別々に処理され、 IDENTITY_VAL_LOCAL

関数を使用して、あるトリガー処置によって割り当てられている値を別のトリガー処置に使用することはできません。概念上、複数のトリガー処置が同じレベルで定義されている場合でも、これは当てはまります。

- 一般的に、BEFORE 挿入トリガーの本体に IDENTITY_VAL_LOCAL 関数を使用することはお勧めしません。 BEFORE 挿入トリガーのトリガー処置から IDENTITY_VAL_LOCAL 関数を呼び出した結果は、NULL 値になります。トリガーが定義されている表の ID 列の値は、BEFORE 挿入トリガーのトリガー処置から IDENTITY_VAL_LOCAL 関数を呼び出すことによって得ることはできません。ただし、ID 列のトリガー遷移変数を参照することにより、その ID 列の値をトリガー処置で得ることができます。
- AFTER 挿入トリガーのトリガー処置から IDENTITY_VAL_LOCAL 関数を呼び出した結果は、ID 列の入った表の VALUES 文節をともなう、同じトリガー処置で呼び出された最新の単一行 INSERT ステートメントで識別されている表の ID 列に割り当てられた値になります。 (これは、トリガーの挿入後、FOR EACH ROW と FOR EACH STATEMENT の両方に適用されます。) ID 列の入った表の VALUES 文節をともなう単一行 INSERT ステートメントが、IDENTITY_VAL_LOCAL 関数呼び出しの前に、同じトリガー処置の中で実行されなかった場合、関数は NULL 値を返します。
- IDENTITY_VAL_LOCAL 関数は deterministic 関数ではないため、カーソルの SELECT ステートメント内の IDENTITY_VAL_LOCAL 関数の呼び出しの結果は、各 FETCH ステートメントによって異なります。
- 割り当て値は、ID 列に実際に割り当てられる値 (つまり、その後に続く SELECT ステートメントで戻される値です)。この値は必ずしも、INSERT ステートメントの VALUES 文節で提供される値、あるいは DB2 によって生成される値とは限りません。割り当て値は、 ID 列に関連するトリガー遷移変数の、トリガー挿入前の本体内の SET 遷移変数ステートメントに指定されている値にすることができます。
- VALUES 文節をともなう単一行 INSERT を ID 列と一緒に表に入れるのに失敗した後の関数によって戻される値は予測不能です。値は、失敗した INSERT の前に呼び出された関数から戻された値である場合もあれば、あるいは、続く INSERT に割り当てられる値の場合もあります。戻される実際の値は、失敗のロケーションにより異なるので、予測不能です。

例:

例 1: 変数 IVAR を、EMPLOYEE 表内の ID 列に割り当てられた値に設定します。これが EMPLOYEE 表への最初の挿入である場合は、IVAR の値は 1 です。

```
CREATE TABLE EMPLOYEE
   (EMPNO   INTEGER GENERATED ALWAYS AS IDENTITY,
    NAME CHAR(30),
    SALARY DECIMAL(5,2),
    DEPTNO  SMALLINT)
```

例 2: INSERT ステートメントで呼び出された IDENTITY_VAL_LOCAL 関数は、ID 列を持つ表の VALUES 文節をともなう直前の単一行 INSERT ステートメントと関連する値です。 T1 と T2 の 2 つの表があると想定してみてください。T1 と T2 の両方に、C1 という名前の ID 列があります。 DB2 は、表 T1 の C1 列に 1 で開始する値を順に生成し、表 T2 の C1 列に 10 で開始する値を順に生成します。

```
CREATE TABLE T1
  (C1 INTEGER GENERATED ALWAYS AS IDENTITY,
   C2 INTEGER)

CREATE TABLE T2
  (C1 DECIMAL(15,0) GENERATED BY DEFAULT AS IDENTITY
    (START WITH 10),
   C2 INTEGER)

INSERT INTO T1 (C2) VALUES (5)

INSERT INTO T1 (C2) VALUES (6)

SELECT * FROM T1
```

結果は次のとおりです。

```
C1           C2
-----------  ----------
          1           5
          2           6
```

ここで、変数 IVAR の関数を宣言します。

```
VALUES IDENTITY_VAL_LOCAL() INTO :IVAR
```

この時点で、IDENTITY_VAL_LOCAL 関数は IVAR に 値 2 を戻します。 DB2 によって割り当てられた最新の値であるためです。以下の INSERT ステートメントでは、単一行を T2 に挿入します。ここで、列 C2 は IDENTITY_VAL_LOCAL 関数から値 2 を得ます。

```
INSERT INTO T2 (C2) VALUES (IDENTITY_VAL_LOCAL())

SELECT * FROM T2
  WHERE C1 = DECIMAL(IDENTITY_VAL_LOCAL(),15,0)
```

戻される結果は次のとおりです。

```
C1                  C2
------------------  ----------
              10.           2
```

この挿入後の IDENTITY_VAL_LOCAL 関数の呼び出しにより、結果は、DB2 が T2 の列 C1 用に生成した値 10 となります。

トリガーに関係するネストされた環境では、ID 列がより低いレベルで割り当てられていても、ある特定のレベルで割り当てられている識別値を検索するには IDENTITY_VAL_LOCAL 関数を使用してください。 3 つの表 EMPLOYEE、EMP_ACT、および ACCT_LOG があると仮定します。 EMP_ACT および ACCT_LOG 表に挿入をさらに行う AFTER 挿入トリガーが EMPLOYEE に定義されています。

```
CREATE TABLE EMPLOYEE
  (EMPNO SMALLINT GENERATED ALWAYS AS IDENTITY (START WITH 1000),
   NAME CHAR(30),
   SALARY DECIMAL(5,2),
   DEPTNO SMALLINT);

CREATE TABLE EMP_ACT
  (ACNT_NUM SMALLINT GENERATED ALWAYS AS IDENTITY (START WITH 1),
   EMPNO SMALLINT);

CREATE TABLE ACCT_LOG
```

```
    (ID SMALLINT GENERATED ALWAYS AS IDENTITY (START WITH 100),
     ACNT_NUM SMALLINT,
     EMPNO SMALLINT);

  CREATE TRIGGER NEW_HIRE
    AFTER INSERT ON EMPLOYEE
    REFERENCING NEW AS NEW_EMP
    FOR EACH ROW
    BEGIN ATOMIC
      INSERT INTO EMP_ACT (EMPNO)
      VALUES (NEW_EMP.EMPNO);
      INSERT INTO ACCT_LOG (ACNT_NUM EMPNO)
      VALUES (IDENTITY_VAL_LOCAL(), NEW_EMP.EMPNO);
    END
```

最初に起動される INSERT ステートメントは、行を EMP_ACT 表に挿入します。この INSERT ステートメントは、EMPLOYEE 表の EMPNO 列のトリガー遷移変数を使用して、 EMPLOYEE 表の EMPNO 列の識別値を EMP_ACT 表の EMPNO 列にコピーするよう指示します。 EMPLOYEE 表の EMPNO 列に割り当てられている値を得るために、 IDENTITY_VAL_LOCAL 関数を使用することはできません。これは、このネストのレベルで INSERT ステートメントが出されていないことが原因で、 IDENTITY_VAL_LOCAL 関数が EMP_ACT の INSERT の VALUES 文節で呼び出された場合、NULL 値が返されます。この EMP_ACT 表の INSERT ステートメントの結果、ACNT_NUM 列の新しい ID 列値も生成されます。

2 番目に起動される INSERT ステートメントは、行を ACCT_LOG 表に挿入します。このステートメントは IDENTITY_VAL_LOCAL 関数を呼び出し、トリガー処置の前の INSERT ステートメントで EMP_ACT 表の ACNT_NUM 列に割り当てられた値を、 ACCT_LOG 表の ACNT_NUM 列にコピーするよう指示します。 EMPNO 列は、EMPLOYEE 表の EMPNO 列と同じ値が割り当てられています。

呼び出しアプリケーション (つまり、EMPLOYEE への INSERT が出されるレベル) から、オリジナル INSERT ステートメントによって EMPLOYEE 表の EMPNO 列に割り当てられた値に IVAR 変数を設定します。

```
  INSERT INTO EMPLOYEE (NAME, SALARY, DEPTNO)
    VALUES ('Rupert', 989.99, 50);
```

オリジナル INSERT ステートメント、およびトリガー処置すべてが処理された後の 3 つの表の内容:

```
  SELECT EMPNO, SUBSTR(NAME,10) AS NAME, SALARY, DEPTNO
     FROM EMPLOYEE;

EMPNO       NAME        SALARY                            DEPTNO
----------- ----------- --------------------------------- -----------
       1000 Rupert                                 989.99          50


  SELECT ACNT_NUM, EMPNO
     FROM EMP_ACT;

ACNT_NUM    EMPNO
----------- -----------
          1        1000


  SELECT * FROM ACCT_LOG;
```

```
ID          ACNT_NUM    EMPNO
----------- ----------- -----------
        100           1        1000
```

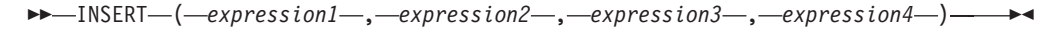
IDENTITY_VAL_LOCAL 関数の結果は、同じネスト・レベルで ID 列に割り当てられた最新の値になります。オリジナル INSERT ステートメント、およびトリガー処置すべてが処理された後、 EMPLOYEE 表の EMPNO 列に割り当てられている値が 1000 であるため、 IDENTITY_VAL_LOCAL 関数は値 1000 を返します。下の VALUES ステートメントによって、IVAR が 1000 に設定されます。 EMP_ACT 表への挿入 (EMPLOYEE 表への挿入の後、より低いネスト・レベルで行われる) は、この IDENTITY_VAL_LOCAL 関数の呼び出しによって何が返されるかには影響しません。

```
VALUES IDENTITY_VAL_LOCAL() INTO :IVAR;
```

関連サンプル:

- 『fnuse.out -- HOW TO USE BUILT-IN SQL FUNCTIONS (C)』
- 『fnuse.sqc -- How to use built-in SQL functions (C)』
- 『fnuse.out -- HOW TO USE FUNCTIONS (C++)』
- 『fnuse.sqC -- How to use built-in SQL functions (C++)』

INSERT



スキーマは SYSFUN です。

expression2 から始まる *expression1* から *expression3* に等しいバイト数が削除され、 *expression2* から始まる *expression1* に *expression4* が挿入されたストリングが戻されます。結果のストリングの長さが戻りタイプの最大値を超える場合、エラーが戻されます (SQLSTATE 38552)。

最初の引き数は、文字ストリングまたはバイナリー・ストリング型です。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。 2 番目および 3 番目の引き数は、データ型が SMALLINT または INTEGER の数値でなければなりません。最初の引き数は、文字ストリングです。そして、4 番目の引き数も、文字ストリングでなければなりません。最初の引き数がバイナリー・ストリングである場合、 4 番目の引き数はバイナリー・ストリングでなければなりません。 VARCHAR の場合、最大長は 4 000 バイトです。 CLOB またはバイナリー・ストリングの場合、最大長は 1 048 576 バイトです。最初の引き数および 4 番目の引き数については、CHAR が VARCHAR に変換され、 LONG VARCHAR が CLOB(1M) に変換されます。 2 番目および 3 番目の引き数の場合、この関数による処理に必要なので、SMALLINT が INTEGER に変換されます。

結果は、以下のように引き数のタイプに基づきます。

- 最初および 4 番目の引き数がいずれも (4 000 バイトを超えない) VARCHAR または CHAR である場合、 VARCHAR(4000) になります。
- 最初または 4 番目の引き数のいずれかが CLOB または LONG VARCHAR である場合、 CLOB(1M) になります。
- 最初および 4 番目の引き数がいずれも BLOB である場合、BLOB(1M) になります。

結果は NULL 値になることがあります。いずれかの引き数が NULL 値である場合、結果は NULL 値になります。

例:

- 語 'DINING' から 1 文字を削除して、'VID' を挿入します。いずれも 3 番目の文字から始まります。

  ```
  VALUES CHAR(INSERT('DINING', 3, 1, 'VID'), 10)
  ```

  この例では、以下を戻します。

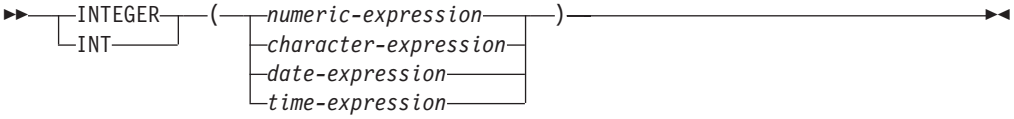
  ```
  1
  ----------
  DIVIDING
  ```

  前述のように、INSERT 関数の出力は VARCHAR(4000) になります。この例の場合、関数 CHAR が、INSERT の出力を 10 バイトに制限するために使用されています。特定のストリングの開始位置は、LOCATE 関数を使用して見つけることができます。

関連資料:

- 388 ページの『LOCATE』

INTEGER



スキーマは SYSIBM です。

INTEGER 関数は、数値、文字ストリング、日付、または時刻を表す整数を、整数定数の形で戻します。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

numeric-expression
組み込み数値データ型の値を返す式。

引き数が*数値式* の場合、結果は、引き数を長精度整数 (large integer) の列または変数に割り当てた場合と同じ数値になります。引き数の整数部分が整数の範囲内にない場合、エラーになります。引き数に小数部分がある場合は、切り捨てられます。

character-expression
文字定数の最大長以下の長さの文字ストリング値を戻す式。先行ブランクと末尾のブランクは削除されます。その結果のストリングは、 SQL 整数定数を形成するための規則に従うものでなければなりません (SQLSTATE 22018)。文字ストリングとして、LONG ストリングを使うことはできません。

引き数が*文字式* の場合、結果は、対応する整数定数を長精度整数の列または変数に割り当てた場合の数値と同じになります。

date-expression
DATE データ型の値を戻す式。結果は、日付を *yyyymmdd* で表した INTEGER 値になります。

time-expression
TIME データ型の値を戻す式。結果は、時間を *hhmmss* で表した INTEGER 値になります。

この関数の結果は長精度整数 (large integer) です。引き数が NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引き数が NULL 値であれば、結果は NULL 値です。

例:

- EMPLOYEE 表を使用して、給与 (SALARY) を学歴 (EDLEVEL) 値で除算した値を示したリストを選択します。計算では小数部をすべて切り捨てます。リストには、計算に使用される値と従業員番号 (EMPNO) も入っていなければなりません。リストは、計算値の降順に配列する必要があります。

```
SELECT INTEGER (SALARY / EDLEVEL), SALARY, EDLEVEL, EMPNO
  FROM EMPLOYEE
  ORDER BY 1 DESC
```

- EMPLOYEE 表を使用して、アプリケーションでさらに処理を行うために、EMPNO 列を整数形式として選択します。

```
SELECT INTEGER(EMPNO) FROM EMPLOYEE
```

- BIRTHDATE (date) 列に、 '1964-07-20' に相当する内部値が入っていると想定します。

  ```
  INTEGER(BIRTHDATE)
  ```

  結果は `19 640 720` の値になります。
- STARTTIME (time) 列に、 '12:03:04' に相当する内部値が入っていると想定します。

  ```
  INTEGER(STARTTIME)
  ```

  結果は `120 304` の値になります。

JULIAN_DAY

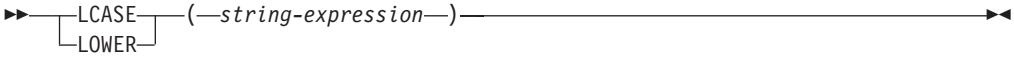
スキーマは SYSFUN です。

紀元前 4713 年 1 月 1 日 (ユリウス暦の起点) からの経過日数を表す整数値を引き数で指定された日付値に戻します。

引き数は、日付またはタイム・スタンプであるか、または CLOB でも LONG VARCHAR でもない日付あるいはタイム・スタンプの有効な文字ストリング表記でなければなりません。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

関数の結果は INTEGER になります。結果は NULL 値になることがあります。引き数が NULL 値である場合、その結果は NULL 値になります。

LCASE または LOWER



スキーマは SYSIBM です。 (この関数の SYSFUN バージョンでは、 LONG VARCHAR 引き数と CLOB 引き数のサポートが引き続き有効です。)

LCASE または LOWER 関数は、すべての SBCS 文字が小文字に変換されたストリングを戻します。つまり、文字 A から Z は文字 a から z に変換され、発音符付きの文字は小文字に相当するものがあればその文字に変換されます。たとえば、コード・ページ 850 では、É は é に変換されます。必ずすべての文字が変換されるわけではないので、 LCASE(UCASE(*string-expression*)) の結果が LCASE(*string-expression*) と同じになるとは限りません。

引き数は、値が CHAR または VARCHAR データ型の式でなければなりません。

関数の結果のデータ型と長さ属性は、引き数と同じになります。引き数が NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引き数が NULL であれば、結果は NULL 値です。

Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

例:

EMPLOYEE 表の列 JOB の値で使用されている文字を小文字に変換します。

```
SELECT LCASE(JOB)
       FROM EMPLOYEE
  WHERE EMPNO = '000020';
```

結果は、値 'manager' になります。

関連資料:

- 384 ページの『LCASE (SYSFUN スキーマ)』

LCASE (SYSFUN スキーマ)

スキーマは SYSFUN です。

文字 A から Z のすべてが文字 a から z に変換されたストリングを戻します (分音符号のついた文字は変換されません)。したがって、LCASE(UCASE(string)) の結果が LCASE(string) と同じになるとは限らない点に注意してください。

引き数は、任意の組み込み文字ストリング型にすることができます。 VARCHAR の場合、最大長は 4 000 バイトです。 CLOB の場合、最大長は 1 048 576 バイトです。

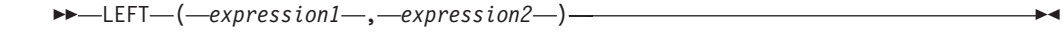
関数の結果は次のとおりです。

- 引き数が VARCHAR (4 000 バイトを超えない) または CHAR である場合、VARCHAR(4000) になります。
- 引き数が CLOB または LONG VARCHAR の場合は CLOB(1M) になります。

結果は NULL 値になることがあります。引き数が NULL 値である場合、その結果は NULL 値になります。

Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

LEFT



スキーマは SYSFUN です。

expression1 の左端 *expression2* バイトからなるストリングを戻します。 *expression1* 値の右側には必要な数のブランク文字が効率的に付加されて、 *expression1* のうちの指定したサブストリングが常に存在するようにされます。

最初の引き数は、文字ストリングまたはバイナリー・ストリング型です。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。 VARCHAR の場合、最大長は 4 000 バイトです。 CLOB またはバイナリー・ストリングの場合、最大長は 1 048 576 バイトです。 2 番目の引き数のデータ型は INTEGER または SMALLINT でなければなりません。

関数の結果は次のとおりです。

- 引き数が VARCHAR (4 000 バイトを超えない) または CHAR である場合、VARCHAR(4000) になります。
- 引き数が CLOB または LONG VARCHAR の場合は CLOB(1M) になります。
- 引き数が BLOB の場合は BLOB(1M) になります。

結果は NULL 値になることがあります。いずれかの引き数が NULL 値である場合、結果は NULL 値になります。

LENGTH

スキーマは SYSIBM です。

LENGTH 関数は、値の長さを戻します。

引き数としては、任意の組み込みデータ型の値を戻す式を使えます。

この関数の結果は長精度整数 (large integer) です。引き数が NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引き数が NULL 値であれば、結果は NULL 値です。

結果は引き数の長さです。その長さには、NULL 値が可能な列引き数の NULL 値標識バイトは入っていません。ストリングの長さにブランクは入りますが、可変長ストリングの長さ制御フィールドは入りません。可変長ストリングの長さは、最大長ではなく実際の長さです。

GRAPHIC ストリングの長さは、DBCS 文字の数になります。それ以外のすべての値の場合の長さは、その値を表すために使用されるバイトの数になります。

- 短整数の場合は 2
- 長精度整数の場合は 4
- 精度 *p* の 10 進数の場合は (*p*/2)+1
- バイナリー・ストリングの場合は、そのストリングの長さ
- 文字ストリングの場合は、そのストリングの長さ
- 単精度浮動小数点の場合は 4
- 倍精度浮動小数点の場合は 8
- 日付の場合は 4
- 時刻の場合は 3
- タイム・スタンプの場合は 10

例:

- ホスト変数 ADDRESS が、 '895 Don Mills Road' という値を持つ可変長文字ストリングであると想定します。

  ```
  LENGTH(:ADDRESS)
  ```

  戻り値は 18 です。
- START_DATE が DATE タイプの列であると想定します。

  ```
  LENGTH(START_DATE)
  ```

  この例では、4 の値を戻します。
- 次の例は、10 の値を戻します。

  ```
  LENGTH(CHAR(START_DATE, EUR))
  ```

LN

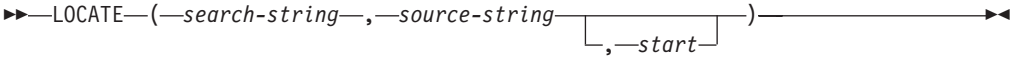
スキーマは SYSFUN です。

引き数の自然対数値を戻します (LOG と同じ)。

引き数は、任意の組み込み数値データ型にすることができます。引き数は、関数での処理に必要な倍精度浮動小数点数に変換されます。

関数の結果は倍精度浮動小数点数になります。結果は NULL 値になることがあります。引き数が NULL 値である場合、その結果は NULL 値になります。

LOCATE



スキーマは SYSFUN です。

source-string 内の *search-string* の最初のオカレンスの開始位置を戻します。オプションの *start* が指定されている場合、その値は、探索を開始する *source-string* の文字位置を示します。これで、LOCATE 関数は以下と同等になります。

```
POSSTR(SUBSTR(source-string, start), search-string) + start - 1
```

search-string が *source-string* 内にない場合、値 0 が戻されます。

最初の引き数が文字ストリングである場合、 2 番目の引き数も文字ストリングでなければなりません。 VARCHAR の場合、最大長は 4 000 バイトです。 CLOB の場合、最大長は 1 048 576 バイトです。最初の引き数がバイナリー・ストリングである場合、2 番目の引き数は、 1 048 576 バイトの最大長をもつバイナリー・ストリングでなければなりません。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。 3 番目の引き数は INTEGER または SMALLINT でなければなりません。

関数の結果は INTEGER になります。結果は NULL 値になることがあります。いずれかの引き数が NULL 値である場合、結果は NULL 値になります。

例:

- 語 'DINING' 内の英字 'N' (最初の出現) の位置を検出します。

  ```
  VALUES LOCATE ('N', 'DINING')
  ```

  この例では、以下を戻します。

  ```
  1
  -----------
            3
  ```

LOG

スキーマは SYSFUN です。

引き数の自然対数値を戻します (LN と同じ)。

引き数は、任意の組み込み数値データ型にすることができます。引き数は、関数での処理に必要な倍精度浮動小数点数に変換されます。

関数の結果は倍精度浮動小数点数になります。結果は NULL 値になることがあります。引き数が NULL 値である場合、その結果は NULL 値になります。

LOG10

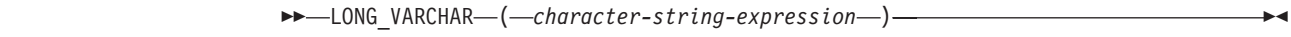
スキーマは SYSFUN です。

引き数に対する、10 を底とする対数 (常用対数) を戻します。

引き数は、任意の組み込み数値タイプにすることができます。引き数は、関数での処理に必要な倍精度浮動小数点数に変換されます。

関数の結果は倍精度浮動小数点数になります。結果は NULL 値になることがあります。引き数が NULL 値である場合、その結果は NULL 値になります。

LONG_VARCHAR



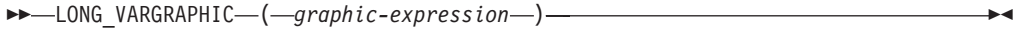
スキーマは SYSIBM です。

LONG_VARCHAR 関数は、文字ストリング・データ型の LONG VARCHAR 表記を戻します。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

character-string-expression
32 700 バイトを最大長とする文字ストリング値を戻す式。

関数の結果は LONG VARCHAR になります。引き数が NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引き数が NULL 値であれば、結果は NULL 値です。

LONG_VARGRAPHIC



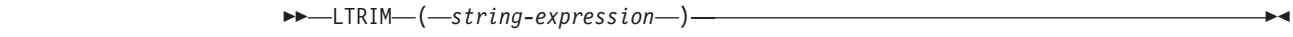
スキーマは SYSIBM です。

LONG_VARGRAPHIC 関数は、2 バイト文字ストリングの LONG VARGRAPHIC 表記を戻します。

graphic-expression
16 350 個の 2 バイト文字を最大長とする GRAPHIC ストリング値を戻す式。

関数の結果は LONG VARGRAPHIC になります。引き数が NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引き数が NULL 値であれば、結果は NULL 値です。

LTRIM



スキーマは SYSIBM です。 (この関数の SYSFUN バージョンでは、 LONG VARCHAR 引き数と CLOB 引き数のサポートが引き続き有効です。)

LTRIM 関数は、*string-expression* の先頭からブランクを除去します。

引き数には CHAR、VARCHAR、GRAPHIC、または VARGRAPHIC データ型を使用できます。

- 引き数が DBCS または EUC データベースの GRAPHIC ストリングである場合は、先行 2 バイト・ブランクが除去されます。
- 引き数が Unicode データベースの GRAPHIC ストリングである場合は、先行 UCS-2 ブランクが除去されます。
- それ以外の場合は、先行 1 バイト・ブランクが除去されます。

この関数の結果のデータ型は次のとおりです。

- *string-expression* のデータ型が VARCHAR または CHAR の場合は VARCHAR になります。
- *string-expression* のデータ型が VARGRAPHIC または GRAPHIC の場合は VARGRAPHIC になります。

戻される型の長さパラメーターは、引き数のデータ型の長さパラメーターと同じになります。

結果が文字ストリングである場合の実際の長さは、除去されるブランク文字のバイト数を *string-expression* から引いた値になります。結果が GRAPHIC ストリングである場合の実際の長さは、除去される 2 バイト・ブランク文字の数を *string-expression* から引いた値 (2 バイト文字単位) になります。すべての文字が除去された場合、結果は空になり、可変長ストリング (長さゼロ) が戻されます。

引き数が NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引き数が NULL 値であれば、結果は NULL 値です。

例:

ホスト変数 HELLO が CHAR(9) と定義されており、値は 'Hello' であるものとします。

```
VALUES LTRIM(:HELLO)
```

結果は 'Hello' になります。

関連資料:

- 394 ページの『LTRIM (SYSFUN スキーマ)』

LTRIM (SYSFUN スキーマ)

スキーマは SYSFUN です。

引き数の文字から先行ブランクを除去して戻します。

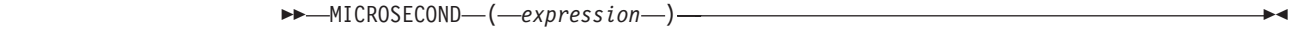
引き数は、任意の組み込み文字ストリング・タイプにすることができます。 VARCHAR の場合、最大長は 4 000 バイトです。 CLOB の場合、最大長は 1 048 576 バイトです。

関数の結果は次のとおりです。

- 引き数が VARCHAR (4 000 バイトを超えない) または CHAR である場合、VARCHAR(4000) になります。
- 引き数が CLOB または LONG VARCHAR の場合は CLOB(1M) になります。

結果は NULL 値になることがあります。引き数が NULL 値である場合、その結果は NULL 値になります。

MICROSECOND



スキーマは SYSIBM です。

MICROSECOND 関数は、値のマイクロ秒の部分を戻します。

引き数は、タイム・スタンプまたはタイム・スタンプ期間であるか、または CLOB でも LONG VARCHAR でもないタイム・スタンプの有効な文字ストリング表記でなければなりません。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

この関数の結果は長精度整数 (large integer) です。引き数が NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引き数が NULL 値であれば、結果は NULL 値です。

その他の規則は、引き数のデータ型に応じて以下のように異なります。

- 引き数がタイム・スタンプまたはタイム・スタンプの有効なストリング表記の場合
  - 整数の範囲は 0 から 999 999 となります。
- 引き数が期間の場合
  - 結果には、 -999 999 から 999 999 の間の整数値としてのマイクロ秒部分が反映されます。ゼロ以外の結果の符号は、引き数と同じになります。

例:

- 表 TABLEA に、タイプが TIMESTAMP の TS1 および TS2 という 2 つの列が入っているものとします。 TS1 のマイクロ秒部分がゼロではなく、 TS1 と TS2 の秒部分が同じである行すべてを選択します。

```
SELECT * FROM TABLEA
  WHERE MICROSECOND(TS1) <> 0
    AND
  SECOND(TS1) = SECOND(TS2)
```

MIDNIGHT_SECONDS

スキーマは SYSFUN です。

午前 0 時から引き数で指定した時刻値までの秒数を表す 0 から 86 400 の範囲の整数値を戻します。

引き数は、時刻またはタイム・スタンプであるか、または CLOB でも LONG VARCHAR でもない時刻またはタイム・スタンプの有効な文字ストリング表記でなければなりません。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

関数の結果は INTEGER になります。結果は NULL 値になることがあります。引き数が NULL 値である場合、その結果は NULL 値になります。

例:

- 午前 0 時から 00:10:10 までの秒数、および午前 0 時から 13:10:10 までの秒数を求めます。

  ```
  VALUES (MIDNIGHT_SECONDS('00:10:10'), MIDNIGHT_SECONDS('13:10:10'))
  ```

  この例では、以下を戻します。

  ```
  1           2
  ----------- -----------
          610       47410
  ```

  1 分は 60 秒なので、午前 0 時から指定された時刻までは 610 秒です。 2 番目の例でも同じです。 1 時間は 3600 秒であり、1 分は 60 秒なので、指定された時刻から午前 0 時までは 47410 秒です。

- 午前 0 時から 24:00:00 までの秒数、および午前 0 時から 00:00:00 までの秒数を求めます。

  ```
  VALUES (MIDNIGHT_SECONDS('24:00:00'), MIDNIGHT_SECONDS('00:00:00'))
  ```

  この例では、以下を戻します。

  ```
  1           2
  ----------- -----------
        86400           0
  ```

  これらの 2 つの値は、同じポイント・イン・タイムを表しているにもかかわらず、 MIDNIGHT_SECONDS 値が異なっていることに注意してください。

MINUTE

スキーマは SYSIBM です。

MINUTE 関数は、値の分の部分を戻します。

引き数は、時刻、タイム・スタンプ、時刻期間、タイム・スタンプ期間であるか、または CLOB でも LONG VARCHAR でもない時刻またはタイム・スタンプの有効な文字ストリング表記でなければなりません。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

この関数の結果は長精度整数 (large integer) です。引き数が NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引き数が NULL 値であれば、結果は NULL 値です。

その他の規則は、引き数のデータ型に応じて以下のように異なります。

- 引き数が、時刻、タイム・スタンプ、または時刻やタイム・スタンプの有効なストリング表記の場合
  - 結果は、値の分の部分 (0 から 59 の整数) になります。
- 引き数が時刻期間またはタイム・スタンプ期間の場合
  - 結果は、値の分の部分 (-99 から 99 の整数) になります。ゼロ以外の結果の符号は、引き数と同じになります。

例:

- CL_SCHED サンプル表を使用して、授業時間が 50 分未満のクラスを全選択します。

```
SELECT * FROM CL_SCHED
  WHERE HOUR(ENDING - STARTING) = 0
    AND
  MINUTE(ENDING - STARTING) < 50
```

MOD

スキーマは SYSFUN です。

最初の引き数を 2 番目の引き数で割った剰余を戻します。結果は、最初の引き数が負である場合にのみ負になります。

関数の結果は次のとおりです。

- 両方の引き数が SMALLINT の場合は SMALLINT になります
- 一方の引き数が INTEGER で、他方が INTEGER または SMALLINT の場合は INTEGER になります
- 一方の引き数が BIGINT で、他方が BIGINT、INTEGER または SMALLINT の場合は BIGINT になります。

結果は NULL 値になることがあります。いずれかの引き数が NULL 値である場合、結果は NULL 値になります。

MONTH

スキーマは SYSIBM です。

MONTH 関数は、値の月の部分を戻します。

引き数は、日付、タイム・スタンプ、日付期間、タイム・スタンプ期間であるか、または CLOB でも LONG VARCHAR でもない日付あるいはタイム・スタンプの有効な文字ストリング表記でなければなりません。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

この関数の結果は長精度整数 (large integer) です。引き数が NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引き数が NULL 値であれば、結果は NULL 値です。

その他の規則は、引き数のデータ型に応じて以下のように異なります。

- 引き数が日付、タイム・スタンプ、または日付やタイム・スタンプの有効なストリング表記の場合
  - 結果は、値の月の部分 (1 から 12 の整数) になります。
- 引き数が日付期間またはタイム・スタンプ期間の場合
  - 結果は、値の月の部分 (-99 から 99 の整数) になります。ゼロ以外の結果の符号は、引き数と同じになります。

例:

- EMPLOYEE 表から、12 月に生まれた (BIRTHDATE) 社員の行を全選択します。

```
SELECT * FROM EMPLOYEE
  WHERE MONTH(BIRTHDATE) = 12
```

MONTHNAME

スキーマは SYSFUN です。

データベースの開始時点のロケールに基づいて、引き数の月の部分の月名から成る大文字小文字混合文字ストリング (たとえば、 January) を戻します。

引き数は、日付またはタイム・スタンプであるか、または CLOB でも LONG VARCHAR でもない日付あるいはタイム・スタンプの有効な文字ストリング表記でなければなりません。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

関数の結果は VARCHAR(100) です。結果は NULL 値になることがあります。引き数が NULL 値である場合、その結果は NULL 値になります。

MULTIPLY_ALT

スキーマは SYSIBM です。

MULTIPLY_ALT スカラー関数は、2 つの引き数の積を 10 進数として返します。これは、引き数の精度の合計が 31 を超えるときに特に合わせて、乗算の演算子の代わりとして用意されています。

引き数には組み込み数値データ型 (DECIMAL、BIGINT、INTEGER、または SMALLINT) を指定できます。

関数の結果は DECIMAL です。結果の精度と位取りは、以下のように決定されます。記号 *p* および *s* を使用して最初の引き数の精度と位取りを、記号 *p* ' および *s* ' を使用して 2 番目の引き数の精度と位取りを指定します。

精度は MIN(31, *p* + *p* ')

位取りは:

- 両方の引き数が 0 の場合は 0
- *p* + *p* ' が 31 以下であれば MIN(31, *s* + *s* ')
- *p* + *p* ' が 31 より大きい場合は MAX(MIN(3, *s* + *s* '), 31 - (*p* - *s* + *p* ' - *s* '))

少なくとも 1 つの引き数が NULL になる可能性があるか、またはデータベース構成パラメーターで DFT_SQLMATHWARN が YES に設定されている場合には、結果は NULL になる可能性があります。引き数の 1 つが NULL の場合、結果は NULL 値になります。

少なくとも 3 の位取りが必要で、精度の合計が 31 を超えるような 10 進数の計算を実行するときには、乗算演算子ではなく MULTIPLY_ALT 関数の使用が推奨されます。このような場合、内部計算が実行されるため、オーバーフローが回避されます。最終結果は、位取りを合わせるために必要に応じて切り捨てを使用して、結果データ型に割り当てられます。最終結果のオーバーフローは、位取りが 3 のときはまだ起こり得ることに注意してください。

以下は、MULTIPLY_ALT と乗算演算子を使用した結果タイプの比較の例です。

引き数タイプ 1	引き数タイプ 2	MULTIPLY_ALT を使用した結果	乗算演算子を使用した結果
DECIMAL(31,3)	DECIMAL(15,8)	DECIMAL(31,3)	DECIMAL(31,11)
DECIMAL(26,23)	DECIMAL(10,1)	DECIMAL(31,19)	DECIMAL(31,24)
DECIMAL(18,17)	DECIMAL(20,19)	DECIMAL(31,29)	DECIMAL(31,31)
DECIMAL(16,3)	DECIMAL(17,8)	DECIMAL(31,9)	DECIMAL(31,11)
DECIMAL(26,5)	DECIMAL(11,0)	DECIMAL(31,3)	DECIMAL(31,5)
DECIMAL(21,1)	DECIMAL(15,1)	DECIMAL(31,2)	DECIMAL(31,2)

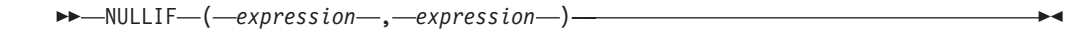
例:

最初の引き数のデータ型が DECIMAL(26,3)、 2 番目の引き数のデータ型が DECIMAL(9,8) の 2 つの値を乗算します。結果のデータ型は DECIMAL(31,7) です。

```
values multiply_alt(98765432109876543210987.654,5.43210987)
1
--------------------------------
536504678578875294857887.5277415
```

これら 2 つの数値の積の全体は 536504678578875294857887.52774154498 ですが、最後の 4 桁は、結果のデータ型の位取りに一致させるために切り捨てられます。同じ値を使って乗算演算子を使用すると、算術オーバーフローが発生します。これは、結果のデータ型が DECIMAL(31,11) で、結果の値の小数点の左側が 24 桁になるものの、結果のデータ型が 20 桁しかサポートしないためです。

NULLIF



スキーマは SYSIBM です。

NULLIF 関数は、引き数が等しい場合は NULL 値を戻し、それ以外の場合には最初の引き数の値を戻します。

2 つの引き数は比較可能でなければなりません。それらは、組み込みデータ型 (LONG ストリングまたは DATALINK 以外) か、まったく異なるデータ型 (LONG ストリングまたは DATALINK に基づくもの以外) のいずれかにすることができます。 (この関数は、ユーザー定義関数の作成時にソース関数として使用することはできません。この関数は、すべての互換データ型を引き数として受け入れるので、ユーザー定義特殊型をサポートするための追加のシグニチャーを作成する必要はありません。) 結果の属性は、最初の引き数の属性になります。

NULLIF(e1,e2) を使用した結果は、次の式を使用した結果と同じになります。

```
CASE WHEN e1=e2 THEN NULL ELSE e1 END
```

一方または両方の引き数が NULL で、e1=e2 が不明と評価されると、 CASE 式はこれを真ではないと見なします。したがって、この場合、NULLIF は最初の引き数の値を戻します。

例:

- ホスト変数 PROFIT、CASH、および LOSSES のデータ型が DECIMAL で、値がそれぞれ 4500.00、500.00、および 5000.00 であるとします。

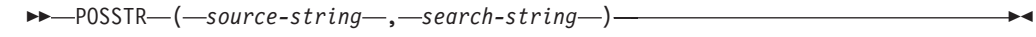
  ```
  NULLIF (:PROFIT + :CASH , :LOSSES )
  ```

  NULL 値が戻されます。

関連資料:

- 110 ページの『割り当てと比較』

POSSTR



スキーマは SYSIBM です。

POSSTR 関数は、あるストリング (*source-string*、ソース・ストリングと呼ばれる) の中で、別のストリング (*search-string*、探索ストリングと呼ばれる) の最初の出現箇所の開始位置を戻します。 *search-string* の位置を示す数値は、1 から始まります (0 ではない)。

この関数の結果は長精度整数 (large integer) です。引き数のいずれかが NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引き数のいずれかが NULL 値の場合、その結果は NULL 値です。

source-string
探索が行われるロケーションとしてのソース・ストリングを指定します。

この式は、以下のいずれかによって指定できます。

- 定数
- 特殊レジスター
- ホスト変数 (ロケーター変数またはファイル参照変数を含む)
- スカラー関数
- ラージ・オブジェクトのロケーター
- 列名
- 上記のいずれかを連結する式

search-string
検索対象のストリングを指定する式。

この式は、以下のいずれかによって指定できます。

- 定数
- 特殊レジスター
- ホスト変数
- 上記のいずれかをオペランドとするスカラー関数
- 上記のいずれかを連結する式

以下の制約があります。

- 式のエレメントに、LONG VARCHAR、CLOB、LONG VARGRAPHIC、または DBCLOB のタイプを使うことはできません。また、BLOB ファイル参照変数は使用できません。
- *search-string* の実際の長さは 4 000 バイト以下でなければなりません。

search-string と *source-string* には、いずれも、ゼロ個以上の連続した位置があります。ストリングが文字ストリングまたはバイナリー・ストリングの場合、 1 つの位置は 1 バイトを表します。ストリングが GRAPHIC ストリングの場合、位置は GRAPHIC (DBCS) 文字になります。

POSSTR 関数は混合データ・ストリングを受け入れます。ただし、 POSSTR は、厳密にバイト・カウント単位で計算し、 1 バイト文字とマルチバイト文字の間の変更は感知しません。

以下の規則が適用されます。

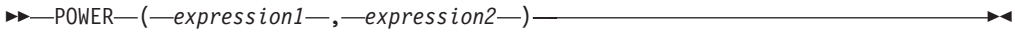
- *source-string* と *search-string* のデータ型には、互換性がある必要があります。そうでない場合、エラーになります (SQLSTATE 42884)。
  - *source-string* が文字ストリングの場合、 *search-string* は CLOB または LONG VARCHAR 以外の文字ストリングでなければならず、実際の長さが 32 672 バイト以下でなければなりません。
  - *source-string* が GRAPHIC ストリングの場合、 *search-string* は DBCLOB または LONG VARGRAPHIC 以外の GRAPHIC ストリングでなければならず、実際の長さが 16 336 以下でなければなりません。
  - *source-string* がバイナリー・ストリングである場合、*search-string* は、実際の長さが 32 672 バイト以下のバイナリー・ストリングでなければなりません。
- *search-string* の長さがゼロの場合、この関数によって戻される結果は 1 です。
- それ以外の場合は、次のとおりです。
  - *source-string* の長さがゼロの場合、関数によって戻される結果はゼロです。
  - それ以外の場合は、次のとおりです。
    - *search-string* が *source-string* の値のうち、連続する複数の位置の同じ長さのサブストリングに等しい場合、この関数によって戻される結果は、*source-string* 値の中でそのようなサブストリングのうち最初の開始位置になります。
    - それ以外の場合、この関数によって戻される結果は 0 です。

例:

- IN_TRAY 表の項目のうち、 NOTE_TEXT 列に 'GOOD BEER' という語句が入っている項目について、 RECEIVED 列と SUBJECT 列、およびその語句の開始位置を選択します。

```
SELECT RECEIVED, SUBJECT, POSSTR(NOTE_TEXT, 'GOOD BEER')
 FROM IN_TRAY
 WHERE POSSTR(NOTE_TEXT, 'GOOD BEER') <> 0
```

POWER



スキーマは SYSFUN です。

expression1 の *expression2* 乗の値を戻します。

引き数は、任意の組み込み数値データ型にすることができます。 DECIMAL および REAL の引き数は倍精度浮動小数点数に変換されます。

関数の結果は次のとおりです。

- 両方の引き数が INTEGER または SMALLINT の場合は INTEGER になります。
- 一方の引き数が BIGINT で、他方が BIGINT、INTEGER または SMALLINT の場合は BIGINT になります。
- それ以外の場合は DOUBLE になります。

結果は NULL 値になることがあります。いずれかの引き数が NULL 値である場合、結果は NULL 値になります。

QUARTER

スキーマは SYSFUN です。

引き数に指定した日付が属する四半期を示す 1 から 4 の範囲の整数値を戻します。

引き数は、日付またはタイム・スタンプであるか、または CLOB でも LONG VARCHAR でもない日付あるいはタイム・スタンプの有効な文字ストリング表記でなければなりません。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

関数の結果は INTEGER になります。結果は NULL 値になることがあります。引き数が NULL 値である場合、その結果は NULL 値になります。

RADIANS

スキーマは SYSFUN です。

度単位の引き数をラジアン単位の角度に変換して戻します。

引き数は、任意の組み込み数値データ型にすることができます。引き数は、関数での処理に必要な倍精度浮動小数点数に変換されます。

関数の結果は倍精度浮動小数点数になります。結果は NULL 値になることがあります。引き数が NULL 値である場合、その結果は NULL 値になります。

RAISE_ERROR

スキーマは SYSIBM です。

RAISE_ERROR 関数は、指定された SQLSTATE、SQLCODE -438、および *diagnostic-string* のエラーが、この関数を備えたステートメントから戻されるようにします。 RAISE_ERROR 関数は、未定義データ型では常に NULL を戻します。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

sqlstate
厳密に 5 文字の文字ストリング。これは、長さ 5 と定義された CHAR 型、または長さ 5 以上と定義された VARCHAR 型でなければなりません。 *sqlstate* の値は、次のように、アプリケーション定義の SQLSTATE の規則に従っていなければなりません。

- 各文字は、数字 ('0' から '9')、またはアクセントのない大文字の英字 ('A' から 'Z') でなければなりません。
- SQLSTATE クラス (最初の 2 文字) は、 '00'、'01'、または '02' であってはなりません (これらの値はエラー・クラスではないので)。
- SQLSTATE クラス (最初の 2 文字) が文字 '0' から '6' または 'A' から 'H' で始まっている場合、サブクラス (最後の 3 文字) は 'I' から 'Z' の範囲の文字で始まっていなければなりません。
- SQLSTATE クラス (最初の 2 文字) が文字 '7'、'8'、'9'、または 'I' から 'Z' で始まっている場合、サブクラス (最後の 3 文字) として '0' から '9' または 'A' から 'Z' のいずれでも使用することができます。

SQLSTATE がこれらの規則に従っていない場合は、エラーになります (SQLSTATE 428B3)。

diagnostic-string
エラー条件を記述する最高 70 バイトの文字ストリングを戻すタイプ CHAR または VARCHAR の式。ストリングが 70 バイトを超える場合には切り捨てられます。

この関数を、結果データ型の規則が適用されないコンテキスト (選択リストで単独の場合など) で使用するには、 Cast 指定を使用して、NULL 値の戻される値にデータ型に割り当てる必要があります。 CASE 式は、RAISE_ERROR 関数を使う最適のロケーションといえます。

例:

従業員番号と学歴のリストを、学歴を Post Graduate、Graduate、および Diploma として示します。学歴が 20 を超える場合は、エラーになります。

```
SELECT EMPNO,
  CASE WHEN EDUCLVL < 16 THEN 'Diploma'
       WHEN EDUCLVL < 18 THEN 'Graduate'
       WHEN EDUCLVL < 21 THEN 'Post Graduate'
```

```
      ELSE RAISE_ERROR('70001',
        'EDUCLVL has a value greater than 20')
  END
FROM EMPLOYEE
```

RAND

スキーマは SYSFUN です。

RAND 関数は、0 から 1 の浮動小数点値を戻します。

式を指定すると、シード値として使用されます。式は、 0 から 2 147 483 647 までの値をもった組み込み SMALLINT または INTEGER データ型でなければなりません。

結果のデータ型は、倍精度の浮動小数点です。引き数が NULL の場合、結果は NULL 値になります。

特定のシード値の場合、照会の実行のたびにその照会中の RAND 関数の特定のインスタンスに対して同じ一連の乱数が生成されます。シード値を指定しない場合は、同一セッション内での照会の実行のたびに別の一連の乱数が生成されます。セッションごとに異なる一連の乱数を生成するには、たとえば現在時刻に基づいてランダム・シードを指定します。

RAND は一律の結果を生じない関数です。

REAL

スキーマは SYSIBM です。

REAL 関数は、数値の単精度浮動小数点表記を戻します。

引き数は、組み込み数値データ型の値を返す式です。

関数の結果は単精度浮動小数点数になります。引き数が NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引き数が NULL 値であれば、結果は NULL 値です。

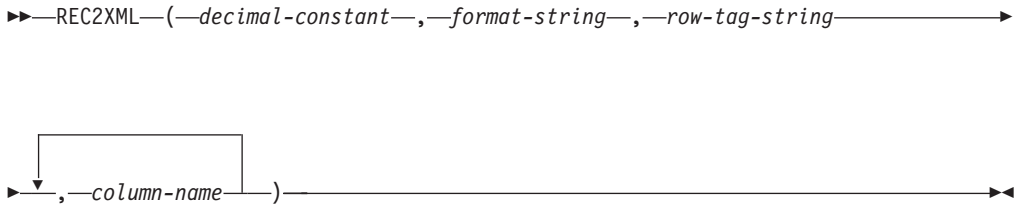
結果は、引き数が単精度浮動小数点の列または変数に割り当てられた場合の結果と同じ数値になります。

例:

EMPLOYEE 表を使用して、歩合がゼロではない従業員の給与と歩合の比率を計算します。関係する列 (SALARY と COMM) のデータ型は DECIMAL です。単精度浮動小数点の結果が必要です。したがって、除算が浮動小数点 (実際には倍精度) で実行されるように REAL が SALARY に適用され、次に単精度の浮動小数点で結果を戻すために REAL が式全体に適用されます。

```
SELECT EMPNO, REAL(REAL(SALARY)/COMM)
  FROM EMPLOYEE
  WHERE COMM > 0
```

REC2XML



スキーマは SYSIBM です。

REC2XML 関数は、XML タグで形式設定されて列名と列データを収めたストリングを戻します。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

decimal-constant
列データ文字の置換用の拡張係数。この 10 進値は 0.0 より大きく、6.0 以下でなければなりません (SQLSTATE 42820)。

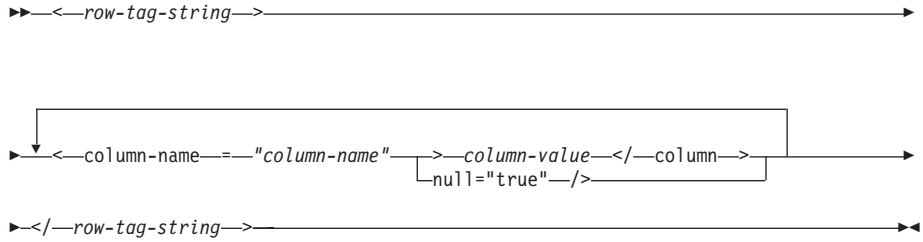
decimal-constant は、関数の結果の長さを計算するために使われます。文字データ型のそれぞれの列ごとに、列の長さ属性が結果の長さに挿入される前に、長さ属性にこの拡張係数を掛けます。

拡張しないことを指定するには、値 1.0 を使用します。 1.0 より小さい値を指定すると、結果の長さが短く計算されます。結果ストリングの実際の長さが、関数の計算された結果の長さよりも長い場合には、エラーが発生します (SQLSTATE 22001)。

format-string
関数を実行する際、どのフォーマットを使用するかを指定するストリング定数。

format-string は大文字小文字を区別するため、以下に示す値が認識されるようにするには、大文字で指定する必要があります。

COLATTVAL または **COLATTVAL_XML**
これらのフォーマットは、列を属性値とするストリングを戻します。



列名は、有効な XML 属性値である場合と、そうでない場合があります。列名が有効な XML 属性値ではない場合、列名が結果ストリングに挿入される前に、列名の文字置換が行われます。

列の値は、有効な XML エレメント名である場合と、そうでない場合があります。 *format-string* に COLATTVAL を指定すると、列名が有効な XML エレメント値ではない場合、列値が結果ストリングに挿入される前に、列値の文字置換

が行われます。 *format-string* に COLATTVAL_XML を指定すると、列値の文字置換は行われません (ただし、列名の文字置換は行われます)。

row-tag-string
各行に使用するタグを指定するストリング定数。空ストリングを指定すると、値 'row' と想定されます。

1 つまたは複数のブランク文字の入ったストリングを指定すると、結果ストリングには、最初の *row-tag-string* も最後の *row-tag-string* も表示されません (不等号括弧の区切り文字を含む)。

column-name
表列の名前 (修飾子付きまたは修飾子なし)。列のデータ型は、以下のいずれかでなければなりません (SQLSTATE 42815)。

- 数値 (SMALLINT、INTEGER、BIGINT、DECIMAL、REAL、DOUBLE)
- 文字ストリング (CHAR、VARCHAR サブタイプ BIT DATA の文字ストリングは使用できません。)
- 日時 (DATE、TIME、TIMESTAMP)
- 上記のいずれかの型に基づくユーザー定義タイプ

同じ列名を 2 度以上指定することはできません (SQLSTATE 42734)。

関数の結果は VARCHAR です。最大長は 32 672 バイトです (SQLSTATE 54006)。

以下のような呼び出しの場合、

REC2XML (dc, fs, rt, c_1, c_2, ..., c_n)

″fs″ の値を ″COLATTVAL″ または ″COLATTVAL_XML″ のいずれかにすると、結果は次の式と同じになります。

```
'<' CONCAT rt CONCAT '>' CONCAT y1 CONCAT y2
CONCAT ... CONCAT yn CONCAT '</' CONCAT rt CONCAT '>'
```

ここで y_n は以下と同等です。

```
'<column name="' CONCAT xvcn CONCAT vn
```

さらに vn は以下と同等です。

```
'">' CONCAT rn CONCAT '</column>'
```

(列が非 NULL の場合)

```
'" null="true"/>'
```

(列値が NULL の場合)

xvc_n は、 c_n の列名のストリング表記と同等です。 415 ページの表 26 に示されたすべての文字は、対応する表記に置換されます。これによって、結果ストリングは必ず有効な XML 属性またはエレメント値のトークンになります。

r_n は、 表 25 に示されているストリング表記と同等です。

表 25. 列値のストリング結果

c_n のデータ型	**r_n**
CHAR、VARCHAR	値はストリングです。 *format-string* が文字 "_XML" で終わらない場合、 c_n 内の各文字は、 表 26 に示される置換表記によって置換されます。長さ属性は、[dc] に [c_n の長さ属性] を乗算したものになります。
SMALLINT、 INTEGER、 BIGINT、 DECIMAL、 NUMERIC、 REAL、 DOUBLE	値は LTRIM(RTRIM(CHAR(c_n))) です。長さ属性は、CHAR(c_n) の結果の長さです。小数点文字は必ずピリオド ('.') 文字です。
DATE	値は CHAR(c_n,ISO) です。長さ属性は、CHAR(c_n,ISO) の結果の長さです。
TIME	値は CHAR(c_n,JIS) です。長さ属性は、CHAR(c_n,JIS) の結果の長さです。
TIMESTAMP	値は CHAR(c_n) です。長さ属性は、CHAR(c_n) の結果の長さです。

文字の置換:

format-string に指定される値によっては、列名を有効な XML 属性値にして、列値を有効な XML エレメント値にするために、列名と列値の一部の文字が置換されます。

表 26. XML 属性値およびエレメント値の文字置換

文字	**置換**
<	<
>	>
"	"
&	&
'	'

例:

注: REC2XML は、出力の中に改行文字を挿入しません。例の出力はすべて、見やすくするために書式を整えています。

- サンプル・データベースの DEPARTMENT 表を使用して、部門 'D01' の部門表の行 (DEPTNAME 列と LOCATION 列を除く) を、 XML ストリングにフォーマット設定します。データの中には置換の必要な文字が入っていないため、拡張係数は 1.0 (拡張なし) です。さらに、この行の MGRNO 値が NULL であることに注意してください。

```
SELECT REC2XML (1.0, 'COLATTVAL', '', DEPTNO, MGRNO, ADMRDEPT)
   FROM DEPARTMENT
   WHERE DEPTNO = 'D01'
```

  この例は、以下の VARCHAR(117) ストリングを戻します。

```
<row>
<column name="DEPTNO">D01</column>
<column name="MGRNO" null="true"/>
<column name="ADMRDEPT">A00</column>
</row>
```

- 5 日制の大学のスケジュールで、'&43<FIE' という名前の授業を、CLASS_CODE 列用の新しいフォーマットを使用して表 CL_SCHED に追加します。この例では REC2XML 関数を使用して、この新しい授業のデータの入った XML ストリングを形式設定します (授業の終了時刻を除く)。

  REC2XML 呼び出しの長さ属性 (下記を参照) に拡張係数 1.0 を掛けて、 128 とします ('<row>' と '</row>' のオーバーヘッドに 11、列名に 21、 '<column name='、'>'、'</column>'、二重引用符に合わせて 75、 CLASS_CODE データに 7、DAY データに 6、STARTING データに 8)。文字 '&' および '<' は置換されるので、拡張係数 1.0 では不十分でしょう。関数の長さ属性は、新しいフォーマットの CLASS_CODE データ用に 7 文字から 14 文字への増加をサポートする必要があるでしょう。

  しかし、DAY 値が決して 1 桁より多くならないことがわかっているので、使用されない余分な 5 単位の長さが合計に加えられます。したがって、2 の増加のみを拡張で扱えばいいことになります。引き数リストの中では CLASS_CODE が唯一の文字ストリング列ですから、拡張係数が適用される列データはこれだけです。長さを 2 だけ増加させるには、拡張係数 9/7 (約 1.2857) が必要でしょう。そこで、拡張係数 1.3 を使用します。

  ```
  SELECT REC2XML (1.3, 'COLATTVAL', 'record', CLASS_CODE, DAY, STARTING)
     FROM CL_SCHED
     WHERE CLASS_CODE = '&43<FIE'
  ```

  この例は、以下の VARCHAR(167) ストリングを戻します。

  ```
  <record>
  <column name="CLASS_CODE">&amp;43&lt;FIE</column>
  <column name="DAY">5</column>
  <column name="STARTING">06:45:00</column>
  </record>
  ```

- サンプル・データベースの EMP_RESUME 表に、新しい行がいくつか追加されたとします。新しい行は、履歴書を有効な (妥当な) XML ストリングとして保管します。文字置換が実行されないように、*format-string* には COLATTVAL_XML を使用します。履歴書の長さは、3500 文字を超えることはありません。以下の照会を使用して、EMP_RESUME 表から履歴書の XML バージョンを選択し、それを XML 文書の一部としてフォーマット設定します。

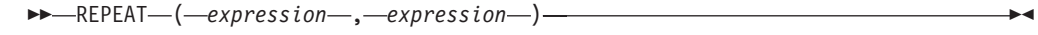
  ```
  SELECT REC2XML (1.0, 'COLATTVAL_XML', 'row', EMPNO, RESUME_XML)
     FROM (SELECT EMPNO, CAST(RESUME AS VARCHAR(3500)) AS RESUME_XML
      FROM EMP_RESUME
      WHERE RESUME_FORMAT = 'XML')
     AS EMP_RESUME_XML
  ```

  この例は、XML フォーマットの履歴書がある各従業員ごとに、行を戻します。戻される各行は、次のフォーマットのストリングになります。

  ```
  <row>
  <column name="EMPNO">{employee number}</column>
  <column name="RESUME_XML">{resume in XML}</column>
  </row>
  ```

ここで ″{employee number}″ は列の実際の EMPNO 値、 ″{resume in XML}″ は履歴書である実際の XML フラグメントのストリング値です。

REPEAT



スキーマは SYSFUN です。

2 番目の引き数によって指定された回数だけ繰り返した最初の引き数で構成される文字ストリングを戻します。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

最初の引き数は、文字ストリングまたはバイナリー・ストリング型です。 VARCHAR の場合、最大長は 4 000 バイトです。 CLOB またはバイナリー・ストリングの場合、最大長は 1 048 576 バイトです。 2 番目の引き数は SMALLINT または INTEGER にすることができます。

関数の結果は次のとおりです。

- 最初の引き数が VARCHAR (4 000 バイトを超えない) または CHAR である場合、 VARCHAR(4000) になります。
- 最初の引き数が CLOB または LONG VARCHAR の場合は CLOB(1M) になります。
- 最初の引き数が BLOB の場合は BLOB(1M) になります。

結果は NULL 値になることがあります。いずれかの引き数が NULL 値である場合、結果は NULL 値になります。

例:

- 句 'REPEAT THIS' を 5 回リストします。

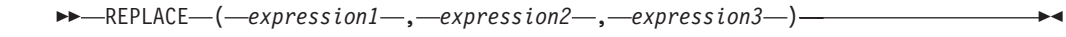
```
VALUES CHAR(REPEAT('REPEAT THIS', 5), 60)
```

この例では以下が戻されます。

```
1
------------------------------------------------------------
REPEAT THISREPEAT THISREPEAT THISREPEAT THISREPEAT THIS
```

前述のように、REPEAT 関数の出力は VARCHAR(4000) になります。上記の例の場合、 REPEAT の出力を 60 バイトまでに制限するために CHAR 関数が使用されています。

REPLACE



スキーマは SYSFUN です。

expression1 における *expression2* のすべての出現箇所を *expression3* に置き換えます。

最初の引き数は、任意の組み込み文字ストリングまたはバイナリー・ストリング・タイプにすることができます。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。 VARCHAR の場合、最大長は 4 000 バイトです。 CLOB またはバイナリー・ストリングの場合、最大長は 1 048 576 バイトです。 CHAR は VARCHAR に変換され、LONG VARCHAR は CLOB(1M) に変換されます。 2 番目と 3 番目の引き数の型は、最初の引き数のものと同じです。

関数の結果は次のとおりです。

- 最初、2 番目、および 3 番目の引き数が VARCHAR または CHAR の場合は VARCHAR(4000) になります。
- 最初、2 番目、および 3 番目の引き数が CLOB または LONG VARCHAR の場合は CLOB(1M) になります。
- 最初、2 番目、および 3 番目の引き数が BLOB の場合は BLOB(1M) になります。

結果は NULL 値になることがあります。いずれかの引き数が NULL 値である場合、結果は NULL 値になります。

例:

語 'DINING' 内の英字 'N' のすべての出現箇所を 'VID' に置き換えます。

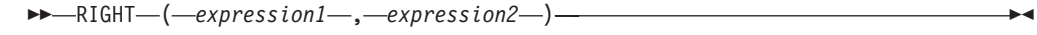
```
VALUES CHAR (REPLACE ('DINING', 'N', 'VID'), 10)
```

この例では、以下を戻します。

```
1
----------
DIVIDIVIDG
```

前述のように、REPLACE 関数の出力は VARCHAR(4000) になります。上記の例の場合、 REPLACE の出力を 10 バイトまでに制限するために CHAR 関数が使用されています。

RIGHT



expression1 の右端の *expression2* バイトからなるストリングを戻します。 *expression1* 値の右側には必要な数のブランク文字が効率的に付加されて、 *expression1* のうちの指定したサブストリングが常に存在するようにされます。

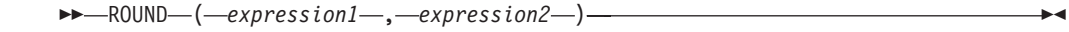
最初の引き数は、文字ストリングまたはバイナリー・ストリング型です。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。 VARCHAR の場合、最大長は 4 000 バイトです。 CLOB またはバイナリー・ストリングの場合、最大長は 1 048 576 バイトです。 2 番目の引き数は INTEGER または SMALLINT にすることができます。

関数の結果は次のとおりです。

- 最初の引き数が VARCHAR (4 000 バイトを超えない) または CHAR である場合、 VARCHAR(4000) になります。
- 最初の引き数が CLOB または LONG VARCHAR の場合は CLOB(1M) になります。
- 最初の引き数が BLOB の場合は BLOB(1M) になります。

結果は NULL 値になることがあります。いずれかの引き数が NULL 値である場合、結果は NULL 値になります。

ROUND



スキーマは SYSIBM です。 (ROUND 関数の SYSFUN バージョンは引き続き使用可能です。)

ROUND 関数は、*expression2* が正の場合は小数点の右側、 *expression2* がゼロまたは負の場合は小数点の左側の、 *expression2* の桁数に丸められた *expression1* を返します。

expression1 が正の場合、5 以上の端数は、次に大きい正の数に丸められます。たとえば ROUND(3.5,0) = 4 です。 *expression1* が負の場合、5 以上の端数は、次に小さい負の数に丸められます。たとえば、ROUND(-3.5,0) = -4 のようになります。

expression1
組み込み数値データ型の値を返す式。

expression2
短整数または長精度整数を返す式。*expression2* の値が負ではないときは、小数点の右側のその桁数に丸めることを指定します。 *expression2* の値が負のときは、小数点の左側の、 *expression2* の位置の絶対値に丸めることを指定します。

expression2 が負ではない場合、 expression1 は小数点の右側の、*expression2* の桁数の絶対値に丸められます。 *expression2* の値が *expression1* の位取りより大きい場合、 1 大きい精度を持つ結果値を除いて、値は変更されません。たとえば、 ROUND(748.58,5) = 748.58 のようになります。ここでの精度は 6 で、位取りは 2 のままです。

expression2 が負の場合、*expression1* は小数点の左側の、 *expression2* +1 の桁数の絶対値に丸められます。

負の *expression2* の絶対値が小数点の左側の桁数より大きい場合、結果は 0 になります。たとえば、ROUND(748.58,-4) = 0 のようになります。

結果のデータ型および長さ属性は、最初の引き数のデータ型および長さ属性と同じになります。ただし、*expression1* が DECIMAL であり、精度が 31 より小さいときに精度が 1 増加する場合を除きます。

たとえば、データ型が DECIMAL(5,2) の引き数の結果は DECIMAL(6,2) になります。データ型 DECIMAL(31,2) の引き数は DECIMAL(31,2) になります。位取りは最初の引き数の位取りと同じです。

引き数が NULL か、またはデータベースの構成で DFT_SQLMATHWARN が YES に設定されている場合、結果は NULL になります。引き数が NULL の場合、結果は NULL 値になります。

例:

値 873.726 を小数点以下 2, 1, 0, -1, -2, -3, および -4 桁に、それぞれ丸めます。

```
VALUES (
  ROUND(873.726, 2),
  ROUND(873.726, 1),
  ROUND(873.726, 0),
```

```
     ROUND(873.726,-1),
     ROUND(873.726,-2),
     ROUND(873.726,-3),
     ROUND(873.726,-4) )
```

この例は次の値を返します。

```
1         2         3         4         5         6         7
--------- --------- --------- --------- --------- --------- ---------
  873.730   873.700   874.000   870.000   900.000  1000.000     0.000
```

正と負の両方の数を使って計算します。

```
   VALUES (
     ROUND(3.5, 0),
     ROUND(3.1, 0),
     ROUNDROUND(-3.1, 0),
     ROUND(-3.5,0) )
```

この例は次の値を返します。

```
1    2    3    4
---- ---- ---- ----
 4.0  3.0 -3.0 -4.0
```

RTRIM

スキーマは SYSIBM です。 (この関数の SYSFUN バージョンでは、 LONG VARCHAR 引き数と CLOB 引き数のサポートが引き続き有効です。)

RTRIM 関数は、*string-expression* の末尾からブランクを除去します。

引き数には CHAR、VARCHAR、GRAPHIC、または VARGRAPHIC データ型を使用できます。

- 引き数が DBCS または EUC データベースの GRAPHIC ストリングである場合は、後続 2 バイト・ブランクが除去されます。
- 引き数が Unicode データベースの GRAPHIC ストリングである場合は、後続 UCS-2 ブランクが除去されます。
- それ以外の場合は、後続 1 バイト・ブランクが除去されます。

この関数の結果のデータ型は次のとおりです。

- *string-expression* のデータ型が VARCHAR または CHAR の場合は VARCHAR になります。
- *string-expression* のデータ型が VARGRAPHIC または GRAPHIC の場合は VARGRAPHIC になります。

戻される型の長さパラメーターは、引き数のデータ型の長さパラメーターと同じになります。

結果が文字ストリングである場合の実際の長さは、除去されるブランク文字のバイト数を *string-expression* から引いた値になります。結果が GRAPHIC ストリングである場合の実際の長さは、除去される 2 バイト・ブランク文字の数を *string-expression* から引いた値 (2 バイト文字単位) になります。すべての文字が除去された場合、結果は空になり、可変長ストリング (長さゼロ) が戻されます。

引き数が NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引き数が NULL 値であれば、結果は NULL 値です。

例: ホスト変数 HELLO が CHAR(9) と定義されており、値は 'Hello ' であるものとします。

```
VALUES RTRIM(:HELLO)
```

結果は 'Hello' になります。

関連資料:

- 424 ページの『RTRIM (SYSFUN スキーマ)』

RTRIM (SYSFUN スキーマ)

スキーマは SYSFUN です。

引き数の文字から後続ブランクを除去した文字を戻します。

引き数は、任意の組み込み文字ストリング・データ型にすることができます。 VARCHAR の場合、最大長は 4 000 バイトです。 CLOB の場合、最大長は 1 048 576 バイトです。

関数の結果は次のとおりです。

- 引き数が VARCHAR (4 000 バイトを超えない) または CHAR である場合、VARCHAR(4000) になります。
- 引き数が CLOB または LONG VARCHAR の場合は CLOB(1M) になります。

結果は NULL 値になることがあります。引き数が NULL 値である場合、その結果は NULL になります。

SECOND

スキーマは SYSIBM です。

SECOND 関数は、値の秒の部分を戻します。

引き数は、時刻、タイム・スタンプ、時刻期間、タイム・スタンプ期間であるか、または CLOB でも LONG VARCHAR でもない時刻またはタイム・スタンプの有効な文字ストリング表記でなければなりません。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

この関数の結果は長精度整数 (large integer) です。引き数が NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引き数が NULL 値であれば、結果は NULL 値です。

その他の規則は、引き数のデータ型に応じて以下のように異なります。

- 引き数が、時刻、タイム・スタンプ、または時刻やタイム・スタンプの有効なストリング表記の場合
  - 結果は、値の秒の部分 (0 から 59 の整数) になります。
- 引き数が時刻期間またはタイム・スタンプ期間の場合
  - 結果は、値の秒の部分 (-99 から 99 の整数) になります。ゼロ以外の結果の符号は、引き数と同じになります。

例:

- ホスト変数 TIME_DUR (decimal(6,0)) の値が 153045 と想定します。

  ```
  SECOND(:TIME_DUR)
  ```

  戻り値は 45 です。
- RECEIVED (timestamp) 列に、 1988-12-25-17.12.30.000000 に相当する内部値が入っていると想定します。

  ```
  SECOND(RECEIVED)
  ```

  この例では 30 の値を戻します。

SIGN

引き数の符号の標識を戻します。引き数が負の場合は、-1 が戻されます。引き数がゼロの場合は、0 が戻されます。引き数が正の場合には、1 が戻されます。

引き数は、任意の組み込み数値データ型にすることができます。 DECIMAL および REAL 値は、関数での処理用に倍精度浮動小数点数に変換されます。

関数の結果は次のとおりです。

- 引き数が SMALLINT の場合は SMALLINT になります。
- 引き数が INTEGER の場合は INTEGER になります。
- 引き数が BIGINT の場合は BIGINT になります。
- それ以外の場合は DOUBLE になります。

結果は NULL 値になることがあります。引き数が NULL 値である場合、その結果は NULL 値になります。

SIN

スキーマは SYSIBM です。 (SIN 関数の SYSFUN バージョンは引き続き使用可能です。)

引き数のサイン (正弦) の値を戻します。引き数は、ラジアン単位の角度です。

引き数は、任意の組み込み数値データ型にすることができます。引き数は、関数での処理に必要な倍精度浮動小数点数に変換されます。

関数の結果は倍精度浮動小数点数になります。引き数が NULL になる可能性があるか、またはデータベース構成パラメーターで DFT_SQLMATHWARN が YES に設定されている場合には、結果は NULL になる可能性があります。引き数が NULL の場合、結果は NULL 値になります。

SINH

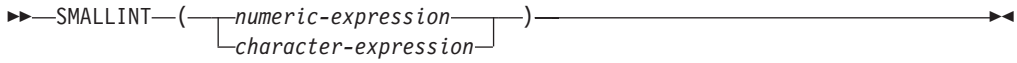
スキーマは SYSIBM です。

引き数に対する双曲線サイン (正弦) の値を戻します。引き数はラジアン単位の角度です。

引き数は、任意の組み込み数値データ型にすることができます。引き数は、関数による処理に必要な倍精度浮動小数点数に変換されます。

関数の結果は倍精度浮動小数点数になります。引き数が NULL になる可能性があるか、またはデータベース構成パラメーターで DFT_SQLMATHWARN が YES に設定されている場合には、結果は NULL になる可能性があります。引き数が NULL の場合、結果は NULL 値になります。

SMALLINT



スキーマは SYSIBM です。

SMALLINT 関数は、短整数 (small integer) 定数の形式の数値または文字ストリングの短整数 (small integer) 表記を戻します。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

numeric-expression
組み込み数値データ型の値を返す式。

引き数が*数値式* の場合、結果は、引き数を短整数 (small integer) の列または変数に割り当てた場合と同じ数値になります。引き数の整数部分が短整数 (small integer) の範囲内にない場合、エラーになります。引き数に小数部分がある場合は、切り捨てられます。

character-expression
文字定数の最大長以下の長さの文字ストリング値を戻す式。先行ブランクと末尾のブランクは削除されます。その結果のストリングは、 SQL 整数定数を形成するための規則に従うものでなければなりません (SQLSTATE 22018)。ただし、定数の値は短整数 (small integer) の範囲内になければなりません (SQLSTATE 22003)。文字ストリングとして、LONG ストリングを使うことはできません。

引き数が文字式 の場合、結果は、対応する整数定数を短整数 (small integer) の列または変数に割り当てた場合の数値と同じになります。

この関数の結果は短整数 (small integer) です。引き数が NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引き数が NULL 値であれば、結果は NULL 値です。

SOUNDEX

スキーマは SYSFUN です。

引き数内の語の音を示す 4 文字コードを戻します。この結果は、他のストリングの音との比較に使用することができます。

引き数は、 CHAR または VARCHAR (4 000 バイトを超えない) のいずれかの文字ストリングです。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

関数の結果は CHAR(4) です。結果は NULL 値になることがあります。引き数が NULL 値である場合、その結果は NULL 値になります。

SOUNDEX 関数は、音が分かっているが、正確なつづりが不明なストリングを検出する場合に有用です。文字および文字の組合わせがどのように聞こえるかを前提とするものであり、類似する音の語の探索に役立ちます。比較は、直接行うことができますが、ストリングを引き数として DIFFERENCE 関数へ渡すことによって行うこともできます。

例:

EMPLOYEE 表を使って、 'Loucesy' に似通った音の姓をもつ従業員の EMPNO および LASTNAME を見つけます。

```
SELECT EMPNO, LASTNAME FROM EMPLOYEE
  WHERE SOUNDEX(LASTNAME) = SOUNDEX('Loucesy')
```

この例では、以下を戻します。

```
EMPNO  LASTNAME
------ ---------------
000110 LUCCHESSI
```

関連資料:

- 334 ページの『DIFFERENCE』

SPACE

スキーマは SYSFUN です。

2 番目の引き数によって指定された長さのブランクで構成される文字ストリングを戻します。

引き数は SMALLINT または INTEGER にすることができます。

関数の結果は VARCHAR(4000) になります。結果は NULL 値になることがあります。引き数が NULL 値である場合、その結果は NULL 値になります。

SQRT

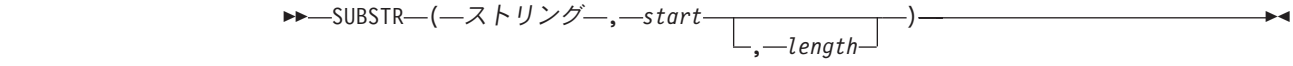
スキーマは SYSFUN です。

引き数の平方根を戻します。

引き数は、任意の組み込み数値データ型にすることができます。引き数は、関数での処理に必要な倍精度浮動小数点数に変換されます。

関数の結果は倍精度浮動小数点数になります。結果は NULL 値になることがあります。引き数が NULL 値である場合、その結果は NULL 値になります。

SUBSTR



SUBSTR 関数は、ストリングのサブストリングを戻します。

string が文字ストリングの場合、関数の結果は、その最初の引き数のコード・ページで示された文字ストリングになります。バイナリー・ストリングの場合には、関数の結果もバイナリー・ストリングになります。 GRAPHIC ストリングの場合は、関数の結果も最初の引き数のコード・ページで示された GRAPHIC ストリングになります。最初の引き数がホスト変数であると、結果のコード・ページは、データベースのコード・ページになります。 SUBSTR 関数のいずれかの引き数が NULL 値の可能性がある場合、結果も NULL 値になる可能性があります。いずれかの引き数が NULL 値の場合、結果は NULL 値になります。

string
結果を取り出すストリングを指定する式。

string が文字ストリングまたはバイナリー・ストリングの場合、 *string* のサブストリングは、ゼロ個以上の連続したバイトからなるストリング になります。 *string* が GRAPHIC ストリングの場合、*string* のサブストリングは、ゼロ個以上の連続する 2 バイト文字からなるストリング になります。

start
結果の最初のバイト位置 (文字ストリングまたはバイナリー・ストリングの場合)、あるいは結果の最初の文字の位置 (GRAPHIC ストリングの場合) を指定する式。 *start* は、*string* が固定長か可変長かに応じて、 1 から *string* の最大長までの整数でなければなりません (この範囲外の場合は SQLSTATE 22011)。データベース・コード・ページのコンテキスト内にあるバイト数として指定しなければなりません。アプリケーション・コード・ページのコンテキストで指定してはなりません。

length
結果の長さを指定する式。この式を指定する場合、 *length* は、0 から *n* の範囲の バイナリー整数でなければなりません。ただし、*n* は、 (*string* の長さ属性) - *start* + 1 です (その範囲外の場合は SQLSTATE 22011)。

length を明示的に指定した場合、 *string* の右側に必要な数のブランク文字 (文字ストリングの場合は 1 バイト、 GRAPHIC ストリングの場合は 2 バイト) が、また BLOB ストリングの場合には必要な数の 16 進ゼロ文字がそれぞれ効率的に付加されて、 *string* のうちの指定したサブストリングが常に存在するようにされます。 *length* のデフォルト値は、文字ストリングまたはバイナリー・ストリングの場合は、 *start* で指定されたバイト位置から *string* の最後のバイト位置までのバイト数、 GRAPHIC ストリングの場合には、 *start* で指定された文字位置から *string* の最後の文字位置までの 2 バイト文字の数です。ただし、*string* が可変長ストリングで、その長さが *start* 未満の場合、デフォルト値はゼロになり、結果は空ストリングになります。データベース・コード・ページのコンテキスト内にあるバイト数として指定しなければなりません。アプリケーション・コード・ページのコンテキストで指定してはなりません。 (たとえば、データ型 VARCHAR(18)、値 'MCKNIGHT' の列 NAME の場合、`SUBSTR(NAME,10)` では空ストリングが戻されます。)

表 27 に、入力のタイプと属性ごとに、 SUBSTR 関数の結果タイプと長さがどうなるかを示しています。

表 27. SUBSTR の結果のデータ型と長さ

ストリング引き数のデータ型	長さ引き数	結果のデータ型
CHAR(A)	定数 (*l*<255)	CHAR(*l*)
CHAR(A)	指定しない。*start* 引き数は定数	CHAR(A-*start*+1)
CHAR(A)	定数以外	VARCHAR(A)
VARCHAR(A)	定数 (*l*<255)	CHAR(*l*)
VARCHAR(A)	定数 (254<*l*<32673)	VARCHAR(*l*)
VARCHAR(A)	定数以外、または指定しない。	VARCHAR(A)
LONG VARCHAR	定数 (*l*<255)	CHAR(*l*)
LONG VARCHAR	定数 (254<*l*<4001)	VARCHAR(*l*)
LONG VARCHAR	定数 (*l*>4000)	LONG VARCHAR
LONG VARCHAR	定数以外、または指定しない。	LONG VARCHAR
CLOB(A)	定数 (*l*)	CLOB(*l*)
CLOB(A)	定数以外、または指定しない。	CLOB(A)
GRAPHIC(A)	定数 (*l*<128)	GRAPHIC(*l*)
GRAPHIC(A)	指定しない。*start* 引き数は定数	GRAPHIC(A-*start*+1)
GRAPHIC(A)	定数以外	VARGRAPHIC(A)
VARGRAPHIC(A)	定数 (*l*<128)	GRAPHIC(*l*)
VARGRAPHIC(A)	定数 (127<*l*<16337)	VARGRAPHIC(*l*)
VARGRAPHIC(A)	定数以外	VARGRAPHIC(A)
LONG VARGRAPHIC	定数 (*l*<128)	GRAPHIC(*l*)
LONG VARGRAPHIC	定数 (127<*l*<2001)	VARGRAPHIC(*l*)
LONG VARGRAPHIC	定数 (*l*>2000)	LONG VARGRAPHIC
LONG VARGRAPHIC	定数以外、または指定しない。	LONG VARGRAPHIC
DBCLOB(A)	定数 (*l*)	DBCLOB(*l*)
DBCLOB(A)	定数以外、または指定しない。	DBCLOB(A)
BLOB(A)	定数 (*l*)	BLOB(*l*)
BLOB(A)	定数以外、または指定しない。	BLOB(A)

string が固定長ストリングの場合に *length* を省略すると、暗黙に LENGTH(*string*) - *start* + 1 が指定されます。 *string* が可変長ストリングの場合に *length* を省略すると、暗黙にゼロまたは LENGTH(*string*) - *start* + 1 のいずれか大きい方が指定されます。

例:

- ホスト変数 NAME (VARCHAR(50)) の値が 'BLUE JAY' で、ホスト変数 SURNAME_POS (int) の値が 6 と想定します。

  ```
  SUBSTR(:NAME, :SURNAME_POS)
  ```

  値 'JAY' が戻されます。

  ```
  SUBSTR(:NAME, :SURNAME_POS,1)
  ```

  この例では 'J' の値を戻します。

- PROJECT 表から、語 'OPERATION' で始まるプロジェクト名 (PROJNAME) の行を全選択します。

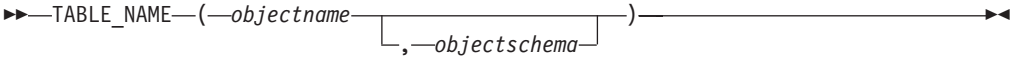
  ```
  SELECT * FROM PROJECT
   WHERE SUBSTR(PROJNAME,1,10) = 'OPERATION '
  ```

  定数の最後にあるスペースは、 'OPERATIONS' などの語で始まるものを除外するために必要です。

注:

1. 動的 SQL では、*string*、*start*、および *length* が、パラメーター・マーカー (?) によって表される場合があります。 *string* にパラメーター・マーカーが使用されると、オペランドのデータ型は VARCHAR になり、オペランドは NULL 可能になります。
2. 上記の結果定義には明確には述べられていませんが、 *string* が 1 バイト文字/マルチバイト文字混合ストリングの場合、 *start* と *length* の値によっては、結果にマルチバイト文字のフラグメントが入ることになる場合があります。つまり、結果が 2 バイト文字の 2 番目のバイトから始まったり、 2 バイト文字の最初のバイトで終わったりする可能性があるということです。 SUBSTR 関数は、このようなフラグメント化の検出を行わず、またこのようなフラグメント化があっても特別な処理は何も行われません。

TABLE_NAME



スキーマは SYSIBM です。

TABLE_NAME 関数は、別名チェーンが解決された後で検出されたオブジェクトの非修飾名を戻します。指定された *objectname* (および *objectschema*) が、解決の開始点として使用されます。開始点が別名を参照していない場合は、開始点の非修飾名が戻されます。結果の名前は、表、ビュー、または未定義オブジェクトのいずれかになります。Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

objectname
解決しようとする非修飾名 (通常は既存の別名) を表す文字式。 *objectname* は、CHAR または VARCHAR のデータ型、 1 文字以上 129 文字未満の長さでなければなりません。

objectschema
指定された *objectname* の解決前の値を修飾するのに使うスキーマを表す文字式。 *objectschema* は、CHAR または VARCHAR のデータ型、 1 文字以上 129 文字未満の長さでなければなりません。

objectschema を指定しない場合は、修飾子にデフォルトのスキーマが使用されます。

この関数の結果のデータ型は VARCHAR(128) です。 *objectname* が NULL 値の可能性がある場合は、結果も NULL 値になる可能性があります。*objectname* が NULL 値であれば、結果も NULL 値になります。 *objectschema* が NULL 値の場合は、デフォルトのスキーマ名が使用されます。結果は、非修飾名を表す文字ストリングになります。結果の名前は、次のいずれかを表す可能性があります。

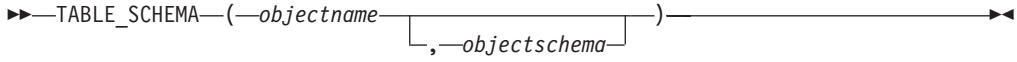
表 *objectname* の値が、表名 (入力値が戻される) であったか、あるいは解決結果が表となり、その表名が戻されることになる別名であった。

ビュー *objectname* の値が、ビュー名 (入力値が戻される) であったか、あるいは解決結果がビューとなり、そのビュー名が戻されることになる別名であった。

未定義オブジェクト
objectname の値が、未定義オブジェクト (入力値が戻される) であったか、あるいは解決結果が未定義オブジェクトとなり、その名前が戻されることになる別名であった。

したがって、NULL 以外の値がこの関数に指定された場合、結果名のオブジェクトが存在していなくても、常にその値が戻されます。

TABLE_SCHEMA



スキーマは SYSIBM です。

TABLE_SCHEMA 関数は、別名チェーンが解決された後で検出されるオブジェクトのスキーマ名を戻します。指定された *objectname* (および *objectschema*) が、解決の開始点として使用されます。開始点が別名を参照していない場合は、開始点のスキーマ名が戻されます。結果のスキーマ名は、表、ビュー、または未定義オブジェクトのいずれかになります。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

objectname
解決しようとする非修飾名 (通常は既存の別名) を表す文字式。 *objectname* は、CHAR または VARCHAR のデータ型、 1 文字以上 129 文字未満の長さでなければなりません。

objectschema
指定された *objectname* の解決前の値を修飾するのに使うスキーマを表す文字式。 *objectschema* は、CHAR または VARCHAR のデータ型、 1 文字以上 129 文字未満の長さでなければなりません。

objectschema を指定しない場合は、修飾子にデフォルトのスキーマが使用されます。

この関数の結果のデータ型は VARCHAR(128) です。 *objectname* が NULL 値の可能性がある場合は、結果も NULL 値になる可能性があります。*objectname* が NULL 値であれば、結果も NULL 値になります。 *objectschema* が NULL 値の場合は、デフォルトのスキーマ名が使用されます。結果は、スキーマ名を表す文字ストリングになります。結果のスキーマは、次のいずれかのスキーマ名を表します。

表 *objectname* の値が、表名 (*objectschema* の入力値またはデフォルト値が戻される) であったか、あるいは解決結果が表となり、そのスキーマ名が戻されることになる別名であった。

ビュー *objectname* の値が、ビュー名 (*objectschema* の入力値またはデフォルト値が戻される) であったか、あるいは解決結果がビューとなり、そのスキーマ名が戻されることになる別名であった。

未定義オブジェクト
objectname の値が、未定義オブジェクト名 (*objectschema* の入力値またはデフォルト値が戻される) であったか、あるいは解決結果が未定義オブジェクトとなり、そのスキーマ名が戻されることになる別名であった。

したがって、NULL 以外の *objectname* 値がこの関数に指定された場合、結果名のスキーマ名でのオブジェクトが存在していなくても、常に値が戻されます。たとえば、`TABLE_SCHEMA('DEPT', 'PEOPLE')` は、カタログ項目が見つからない場合には、'PEOPLE ' を戻します。

例:

- PBIRD は、表 HEDGES.T1 に定義されている別名 PBIRD.A1 を使用して、SYSCAT.TABLES から指定した表の統計値を選択しようとしています。

```
SELECT NPAGES, CARD FROM SYSCAT.TABLES
 WHERE TABNAME = TABLE_NAME ('A1')
 AND TABSCHEMA = TABLE_SCHEMA ('A1')
```

  HEDGES.T1 について要求された統計値が、カタログから取り出されます。

- HEDGES.X1 というオブジェクトの統計値を、 HEDGES.X1 を使用して SYSCAT.TABLES から選択します。 HEDGES.X1 が別名か表かが分からないため、 TABLE_NAME と TABLE_SCHEMA を使用します。

```
SELECT NPAGES, CARD FROM SYSCAT.TABLES
 WHERE TABNAME = TABLE_NAME ('X1','HEDGES')
 AND TABSCHEMA = TABLE_SCHEMA ('X1','HEDGES')
```

  HEDGES.X1 が表であるとすると、 HEDGES.X1 について要求された統計がカタログから取り出されます。

- HEDGES.T2 に対して定義された別名 PBIRD.A2 を使用して、 SYSCAT.TABLES から指定した表の統計を選択しますが、HEDGES.T2 は存在していません。

```
SELECT NPAGES, CARD FROM SYSCAT.TABLES
 WHERE TABNAME = TABLE_NAME ('A2','PBIRD')
 AND TABSCHEMA = TABLE_SCHEMA ('A2',PBIRD')
```

  TABNAME = 'T2' および TABSCHEMA = 'HEDGES' である項目が SYSCAT.TABLES の中に見つからないため、このステートメントからは 0 個のレコードが戻されます。

- SYSCAT.TABLES 内の各項目の修飾名、および別名項目については最終参照名を選択します。

```
SELECT TABSCHEMA AS SCHEMA,  TABNAME AS NAME,
  TABLE_SCHEMA (BASE_TABNAME, BASE_TABSCHEMA) AS REAL_SCHEMA,
  TABLE_NAME (BASE_TABNAME, BASE_TABSCHEMA) AS REAL_NAME
  FROM SYSCAT.TABLES
```

  このステートメントは、カタログ内の各オブジェクトの修飾名と、別名項目については最終参照名 (別名が解決された後の名前) を戻します。別名でないすべての項目については、 BASE_TABNAME および BASE_TABSCHEMA が NULL 値であるため、 REAL_SCHEMA 列と REAL_NAME 列は NULL 値になります。

TAN

スキーマは SYSIBM です。 (TAN 関数の SYSFUN バージョンは引き続き使用可能です。)

引き数のタンジェント (正接) の値を戻します。引き数は、ラジアン単位の角度です。

引き数は、任意の組み込み数値データ型にすることができます。引き数は、関数での処理に必要な倍精度浮動小数点数に変換されます。

関数の結果は倍精度浮動小数点数になります。引き数が NULL になる可能性があるか、またはデータベース構成パラメーターで DFT_SQLMATHWARN が YES に設定されている場合には、結果は NULL になる可能性があります。引き数が NULL の場合、結果は NULL 値になります。

TANH

スキーマは SYSIBM です。

引き数に対する双曲線タンジェント (正接) の値を戻します。引き数はラジアン単位の角度です。

引き数は、任意の組み込み数値データ型にすることができます。引き数は、関数による処理に必要な倍精度浮動小数点数に変換されます。

関数の結果は倍精度浮動小数点数になります。引き数が NULL になる可能性があるか、またはデータベース構成パラメーターで DFT_SQLMATHWARN が YES に設定されている場合には、結果は NULL になる可能性があります。引き数が NULL の場合、結果は NULL 値になります。

TIME

スキーマは SYSIBM です。

TIME 関数は、値から時刻を戻します。

引き数は、時刻、タイム・スタンプであるか、または CLOB、LONG VARCHAR、DBCLOB、または LONG VARGRAPHIC ではない時刻またはタイム・スタンプの有効なストリング表記でなければなりません。

Unicode データベースだけが、時刻またはタイム・スタンプの GRAPHIC ストリング表現である引き数をサポートします。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

関数の結果は時刻です。引き数が NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引き数が NULL 値であれば、結果は NULL 値です。

その他の規則は、引き数のデータ型に応じて以下のように異なります。

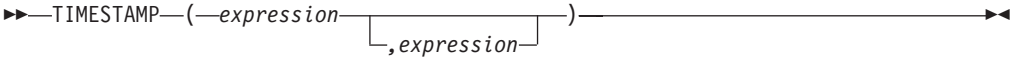
- 引き数が時刻の場合
  - 結果は、指定した時刻になります。
- 引き数がタイム・スタンプの場合
  - 結果はタイム・スタンプの時刻の部分になります。
- 引き数がストリングの場合
  - 結果は、そのストリングによって表される時刻になります。

例:

- IN_TRAY サンプル表から、 (任意の日の) 現在の時刻よりも 1 時間後以降に受け取ったすべての注を選択します。

```
SELECT * FROM IN_TRAY
  WHERE TIME(RECEIVED) >= CURRENT TIME + 1 HOUR
```

TIMESTAMP



スキーマは SYSIBM です。

TIMESTAMP 関数は、1 つの値または 2 つの値からタイム・スタンプを戻します。

Unicode データベースだけが、日付、時刻、またはタイム・スタンプの GRAPHIC ストリング表現である引き数をサポートします。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

引き数に関する規則は、2 番目の引き数を指定するか否かによって異なります。

- 引き数を 1 つだけ指定した場合
  - タイム・スタンプ、タイム・スタンプの有効なストリング表記、あるいは CLOB、LONG VARCHAR、DBCLOB、または LONG VARGRAPHIC ではない長さ 14 のストリングのいずれかでなければなりません。

    長さ 14 のストリングは、有効な日付と時刻を *yyyyxxddhhmmss* という形式で表した数字のストリングであることが必要です (ここで、 *yyyy* は年、 *xx* は月、 *dd* は日、 *hh* は時、 *mm* は分、そして *ss* は秒を表します)
- 引き数を 2 つとも指定する場合
  - 最初の引き数は日付または日付の有効なストリング表記でなければならず、 2 番目の引き数は時刻または時刻の有効なストリング表記でなければなりません。

関数の結果はタイム・スタンプです。引き数のいずれかが NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引き数のいずれかが NULL 値の場合、その結果は NULL 値です。

その他の規則は、2 番目の引き数を指定するか否かによって以下のように異なります。

- 引き数を 2 つとも指定する場合
  - 結果は、最初の引き数によって日付が指定され、 2 番目の引き数によって時刻が指定されたタイム・スタンプです。タイム・スタンプのマイクロ秒部分はゼロです。
- 引き数が 1 つだけ指定され、それがタイム・スタンプの場合
  - 結果は、指定したタイム・スタンプになります。
- 引き数が 1 つだけ指定され、それがストリングの場合
  - 結果は、ストリングによって表されるタイム・スタンプになります。引き数が長さ 14 のストリングの場合、タイム・スタンプのマイクロ秒部分はゼロになります。

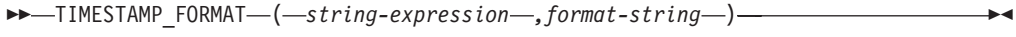
例:

- START_DATE (日付) 列が 1988-12-25 に等しい値、 START_TIME (時刻) 列が 17.12.30 に等しい値であると想定します。

```
TIMESTAMP(START_DATE, START_TIME)
```

この例は、値 '1988-12-25-17.12.30.000000' を戻します。

TIMESTAMP_FORMAT



スキーマは SYSIBM です。

TIMESTAMP_FORMAT 関数は、文字テンプレートを使って解釈された文字ストリングからタイム・スタンプを戻します。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

string-expression

format-string によって指定されたフォーマットでタイム・スタンプ値を表す文字式。 (*string-expression* がタイプ無しパラメーター・マーカーの場合、タイプは最大長 254 を持つ VARCHAR であると想定されます。) ストリング式は、254 未満の最大長をもつ CHAR または VARCHAR の値を戻します (SQLSTATE 42815)。先頭および末尾のブランクは *string-expression* から除かれ、ブランクが除かれたそのサブストリングは、 *format-string* によって指定されたフォーマットを使うタイム・スタンプと解釈されます。先頭のゼロは、年を除くすべてのタイム・スタンプ・コンポーネントから省略することができます。それらのコンポーネントでは、先頭のゼロの代わりにブランクを使用できます。たとえば、以下のストリングはいずれも、'YYYY-MM-DD HH24:MI:SS' というフォーマットで 9 a.m. on January 1, 2000 (2000 年 1 月 1 日午前 9 時) を表したものですが、どれも受け入れられます。

```
'2000-1-01 09:00:00'    (月が 1 桁)
'2000- 1-01 09:00:00'   (月が 1 桁で、前にブランクがある)
'2000-1-1 09:00:00'     (月および日が 1 桁)
'2000-01-01 9:00:00'    (時間が 1 桁)
'2000-01-01 09:0:0'     (分および秒が 1 桁)
'2000- 1- 1 09: 0: 0'   (月、日、分および秒が 1 桁で、
                         前にブランクがある)
'2000-01-01 09:00:00'   (各エレメントが最大桁数)
```

format-string

タイム・スタンプ値としてストリング式をどのように解釈するかに関するテンプレートを収めた文字定数。書式制御ストリングの長さは、254 (SQLSTATE 42815) 以下でなければなりません。先頭および末尾のブランクは *format-string* から除かれ、ブランクが除かれたそのサブストリングは、タイム・スタンプ値の有効なテンプレートになるはずです (SQLSTATE 42815)。 *format-string* の内容は英大文字小文字混合で指定できます。

有効な書式制御ストリングは次のとおりです。

```
'YYYY-MM-DD HH24:MI:SS'
```

YYYY は 4 桁の年の値を表し、 *MM* は 2 桁の月の値を表します (01 から 12。
January (1 月) = 01)。 *DD* は 2 桁の日の値を表し (01 から 31)、 *HH24* は 2 桁の時間の値を表します (00 から 24。 24 時間制の場合、分および秒の値はゼロになります)。 *MI* は 2 桁の分の値を表し (00 から 59)、 *SS* は 2 桁の秒の値を表します (00 から 59)。

関数の結果はタイム・スタンプです。最初の引き数を NULL にすることができる場合、結果も NULL にすることができます。最初の引き数が NULL であれば、結果は NULL 値になります。

例:

- 2000 年が始まる前に (December 31, 1999 at 23:59:59 (1999 年 12 月 31 日 23:59:59))、 1 秒と同等の受信タイム・スタンプを使って、in_tray 表に行を挿入します。

```
INSERT INTO in_tray (received)
  VALUES (TIMESTAMP_FORMAT('1999-12-31 23:59:59',
    'YYYY-MM-DD HH24:MI:SS'))
```

TIMESTAMP_ISO

スキーマは SYSFUN です。

日付、時刻、またはタイム・スタンプの引き数に基づいてタイム・スタンプ値を戻します。引き数が日付の場合は、時間エレメントのすべてにゼロが入れられます。引き数が時刻の場合、日付エレメントには CURRENT DATE 特殊レジスターの値が入れられ、時刻の小数エレメントにはゼロが入れられます。

引き数は、日付、時刻、またはタイム・スタンプであるか、または CLOB でも LONG VARCHAR でもない日付、時刻、またはタイム・スタンプの有効な文字ストリング表現でなければなりません。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

関数の結果は TIMESTAMP になります。結果は NULL 値になることがあります。引き数が NULL 値である場合、その結果は NULL 値になります。

TIMESTAMPDIFF

スキーマは SYSFUN です。

2 つのタイム・スタンプ間の差に基づいて、最初の引き数によって定義されたタイプのインターバル数の見積もりが戻されます。

最初の引き数は INTEGER または SMALLINT のいずれかです。インターバル (最初の引き数) の有効な値は次のとおりです。

1 秒の小数部

2 秒

4 分

8 時間

16 日

32 週

64 月

128 四半期

256 年

2 番目の引き数は、2 つのタイム・スタンプの減算を行い、その結果を CHAR(22) に変換した結果です。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

関数の結果は INTEGER になります。結果は NULL 値になることがあります。引き数が NULL 値である場合、その結果は NULL 値になります。

以下の前提事項が、差の見積もりに使用されます。

- 1 年は 365 日である。
- 1 か月は 30 日である。
- 1 日は 24 時間である。
- 1 時間は 60 分である。
- 1 分は 60 秒である。

上記の前提は、タイム・スタンプ期間である 2 番目の引き数の情報を最初の引き数で指定されたインターバル・タイプに変換する際に使用されます。戻される見積もりが、日数によって異なる場合があります。たとえば、 '1997-03-01-00.00.00' と '1997-02-01-00.00.00' の差の日数 (インターバル 16) が要求された場合、結果は 30 になります。これは、タイム・スタンプ相互間の差は 1 か月であり、 1 か月は 30 日であるという前提が適用されるからです。

例 :

以下の例は、2 つのタイム・スタンプにはさまれた分数である 4277 を戻します。

```
TIMESTAMPDIFF(4,CHAR(TIMESTAMP('2001-09-29-11.25.42.483219') -
  TIMESTAMP('2001-09-26-12.07.58.065497')))
```

TO_CHAR

スキーマは SYSIBM です。

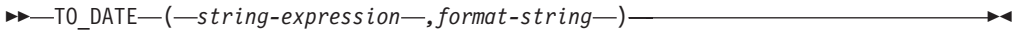
TO_CHAR 関数は、文字テンプレートを使ってフォーマットされたタイム・スタンプの文字表現を戻します。

TO_CHAR は VARCHAR_FORMAT の同義語です。

関連資料:

- 462 ページの『VARCHAR_FORMAT』

TO_DATE



スキーマは SYSIBM です。

TO_DATE 関数は、文字テンプレートを使って解釈された文字ストリングからタイム・スタンプを戻します。

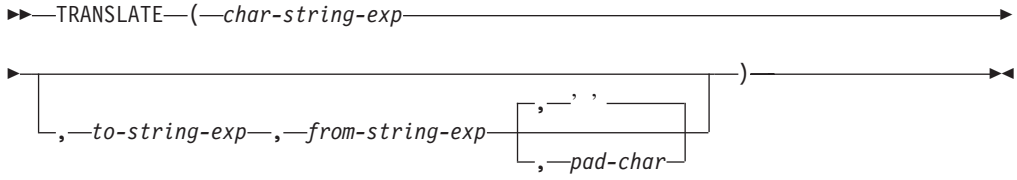
TO_DATE は TIMESTAMP_FORMAT の同義語です。

関連資料:

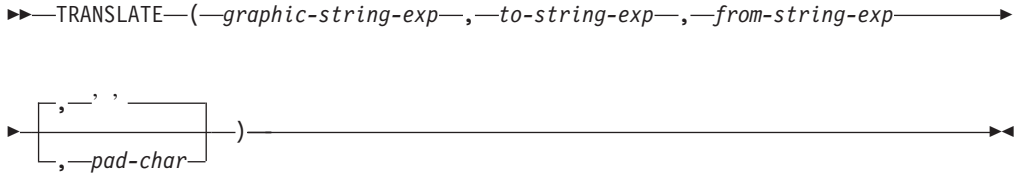
• 444 ページの『TIMESTAMP_FORMAT』

TRANSLATE

文字ストリング式:



GRAPHIC ストリング式:



スキーマは SYSIBM です。

TRANSLATE 関数は、ストリング式内の 1 つまたは複数の文字が他の文字に変換された値を戻します。

関数の結果のデータ型とコード・ページは、最初の引き数と同じです。最初の引き数がホスト変数であると、結果のコード・ページは、データベースのコード・ページになります。結果の長さ属性は、最初の引き数と同じになります。指定した式のいずれかが NULL 値の可能性がある場合、結果も NULL の可能性があります。指定した式のいずれかが NULL 値の場合、結果も NULL になります。

char-string-exp または *graphic-string-exp*
変換されるストリング。

to-string-exp
char-string-exp の特定文字の変換後の文字ストリング。

to-string-exp を指定せず、データ型が GRAPHIC でない場合、 *char-string-exp* 内の文字すべてが大文字変換されます。つまり、文字 a から z は文字 A から Z に変換され、発音符付きの文字は大文字に相当するものがあればその文字に変換されます。たとえば、コード・ページ 850 では、é は É に変換されますが、 ÿ は変換されません。これは、コード・ページ 850 に Ÿ が組み込まれていないためです。

from-string-exp
このストリングの文字が *char-string-exp* の中にある場合、それは *to-string-exp* の中の対応する文字に変換されることになります。 *from-string-exp* に重複する文字が入っている場合は、最初に見つかった文字が使用され、重複は無視されます。 *to-string-exp* が *from-string-exp* より長い場合、余分な文字は無視されます。 *to-string-exp* を指定する場合は、 *from-string-exp* も指定する必要があります。

pad-char-exp
これは、*to-string-exp* が *from-string-exp* より短い場合に、 *to-string-exp* への埋

め込みに使用する単一の文字です。 *pad-char-exp* は長さ属性が 1 でなければなりません。そうでない場合には、エラーになります。これを指定しない場合、1 バイトのブランクと見なされます。

引き数には、データ型 CHAR または VARCHAR のいずれかのストリングか、あるいはデータ型 GRAPHIC または VARGRAPHIC の GRAPHIC ストリングを指定できます。データ型 LONG VARCHAR、LONG VARGRAPHIC、BLOB、 CLOB、または DBCLOB は使用できません。

graphic-string-exp を指定する場合、 *pad-char-exp* だけがオプションです (これを指定しない場合、 2 バイトのブランクが使用されます)。埋め込み文字を含め、各引き数は GRAPHIC データ型でなければなりません。

結果は、*from-string-exp* の中で出現する *char-string-exp* または *graphic-string-exp* 内の文字すべてを、 *to-string-exp* 内の対応する文字に変換した結果のストリング、あるいは対応する文字がない場合は *pad-char-exp* で指定される埋め込み文字に変換した結果のストリングになります。

TRANSLATE の結果のコード・ページは、最初のオペランドのコード・ページと同じになります。バージョン 8 の時点では、最初のオペランドがホスト変数であると、結果のコード・ページは、データベースのコード・ページになります。それ以外のどのオペランドの場合も、それ自身または最初のオペランドが FOR BIT DATA と定義されて (この場合は変換は行われません) いない限り、結果のコード・ページに変換されます。

引き数のCHAR または VARCHAR のデータ型 の場合、 *to-string-exp* と *from-string-exp* に対応する文字は、同じバイト数でなければなりません。たとえば、1 バイト文字をマルチバイト文字に変換することや、マルチバイト文字を 1 バイト文字に変換することは無効です。そのような変換を行うと、エラーになります。 *pad-char-exp* として有効なマルチバイト文字の第 1 バイトを指定することはできません。そのように指定すると、SQLSTATE 42815 が戻されます。 *pad-char-exp* を指定しない場合は、 1 バイトのブランクが使用されます。

char-string-exp のみを指定した場合、 1 バイト文字は大文字変換され、マルチバイト文字はそのままになります。

例:

- ホスト変数 SITE (VARCHAR(30)) の値が 'Hanauma Bay' であると想定します。

  ```
  TRANSLATE(:SITE)
  ```

  この例では、値 'HANAUMA BAY' を戻します。

  ```
  TRANSLATE(:SITE 'j','B')
  ```

  この例では、'Hanauma jay' が戻されます。

  ```
  TRANSLATE(:SITE,'ei','aa')
  ```

  この例では、値 'Heneume Bey' が戻されます。

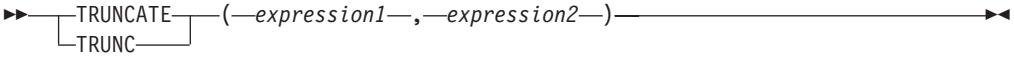
  ```
  TRANSLATE(:SITE,'bA','Bay','%')
  ```

  この例では、値 'HAnAumA bA%' が戻されます。

```
TRANSLATE(:SITE,'r','Bu')
```

この例では、値 'Hana ma ray' が戻されます。

TRUNCATE または TRUNC



スキーマは SYSIBM です。 (TRUNCATE または TRUNC 関数の SYSFUN バージョンは引き続き使用可能です。)

expression2 が正の場合は小数点の右側、 *expression2* がゼロまたは負の場合は小数点の左側の、 *expression2* の桁数に丸められた *expression1* を返します。

expression1
組み込み数値データ型の値を返す式。

expression2
短整数または長精度整数を返す式。整数の絶対値は、*expression2* が正の場合は小数点の右側、 *expression2* が負の場合は小数点の左側の結果桁数を指定します。

expression2 の絶対値が小数点の左側の桁数より大きい場合、結果は 0 になります。たとえば:

```
TRUNCATE(748.58,-4) = 0
```

結果のデータ型および長さ属性は、最初の引き数のデータ型および長さ属性と同じになります。

引き数が NULL になる可能性があるか、またはデータベース構成パラメーター DFT_SQLMATHWARN が YES に設定されている場合には、結果は NULL になる可能性があります。引き数が NULL の場合、結果は NULL 値になります。

例:

- EMPLOYEE 表を使って、最高給与額の社員の平均月給を計算します。結果の小数点より右側の 2 桁を切り捨てます。

  ```
  SELECT TRUNCATE(MAX(SALARY)/12,2)
    FROM EMPLOYEE;
  ```

  最高給与額の社員は年収が $52750.00 であるので、この例では 4395.83 が戻されます。

- 873.726 をそれぞれ小数点以下 2、1、0、-1、および -2 桁に切り捨てた数字が表示されます。

  ```
  VALUES (
    TRUNC(873.726,2),
    TRUNC(873.726,1),
    TRUNC(873.726,0),
    TRUNC(873.726,-1),
    TRUNC(873.726,-2),
    TRUNC(873.726,-3) );
  ```

  この例では、873.720、873.700、873.000、870.000、800.000、および 0.000 が戻されます。

TYPE_ID

スキーマは SYSIBM です。

TYPE_ID 関数は、*expression* の動的データ型の内部タイプ ID を戻します。

引き数はユーザー定義の構造化タイプでなければなりません。 (この関数は、ユーザー定義関数の作成時にソース関数として使用することはできません。この関数は、すべての構造化データ型を引き数として受け入れるので、別のユーザー定義タイプをサポートするための追加のシグニチャーを作成する必要はありません。)

この関数の結果のデータ型は INTEGER です。 *expression* が NULL 値の可能性がある場合は、結果も NULL 値になる可能性があります。 *expression* が NULL であれば、結果も NULL 値になります。

TYPE_ID 関数が戻した値は、データベース間で移動することはできません。動的データ型のタイプ・スキーマおよびタイプ名が同じであっても、値が異なる可能性があります。移植性を考慮してコーディングを行う場合、タイプ・スキーマおよびタイプ名の判別には TYPE_SCHEMA および TYPE_NAME 関数を使用してください。

例:

- ある表階層には、タイプ EMP のルート表 EMPLOYEE と、タイプ MGR の副表 MANAGER があります。別の表 ACTIVITIES は、 REF(EMP) SCOPE EMPLOYEE と定義されている WHO_RESPONSIBLE という列を収めています。ACTIVITIES を参照するたびに、参照に対応する行の内部タイプ ID を表示します。

  ```
  SELECT TASK, WHO_RESPONSIBLE->NAME,
      TYPE_ID(DEREF(WHO_RESPONSIBLE))
   FROM ACTIVITIES
  ```

  DEREF 関数は、行に対応するオブジェクトを戻すときに使用します。

TYPE_NAME

スキーマは SYSIBM です。

TYPE_ID 関数は、*expression* の動的データ型の非修飾名を戻します。

引き数はユーザー定義の構造化タイプでなければなりません。 (この関数は、ユーザー定義関数の作成時にソース関数として使用することはできません。この関数は、すべての構造化データ型を引き数として受け入れるので、別のユーザー定義タイプをサポートするための追加のシグニチャーを作成する必要はありません。)

この関数の結果のデータ型は VARCHAR(18) です。 *expression* が NULL 値の可能性がある場合は、結果も NULL 値になる可能性があります。 *expression* が NULL であれば、結果も NULL 値になります。 TYPE_SCHEMA 関数を使用して、TYPE_NAME が戻したタイプ名のスキーマ名を判別してください。

例:

- ある表階層には、タイプ EMP のルート表 EMPLOYEE と、タイプ MGR の副表 MANAGER があります。別の表 ACTIVITIES は、 REF(EMP) SCOPE EMPLOYEE と定義されている WHO_RESPONSIBLE という列を収めています。 ACTIVITIES を参照するたびに、参照に対応する行のタイプを表示します。

  ```
  SELECT TASK, WHO_RESPONSIBLE->NAME,
      TYPE_NAME(DEREF(WHO_RESPONSIBLE)),
      TYPE_SCHEMA(DEREF(WHO_RESPONSIBLE))
   FROM ACTIVITIES
  ```

  DEREF 関数は、行に対応するオブジェクトを戻すときに使用します。

TYPE_SCHEMA

スキーマは SYSIBM です。

TYPE_SCHEMA 関数は、*expression* の動的データ型のスキーマ名を戻します。

引き数はユーザー定義の構造化タイプでなければなりません。この関数は、ユーザー定義関数の作成時にソース関数として使用することはできません。この関数は、すべての構造化データ型を引き数として受け入れるので、別のユーザー定義タイプをサポートするための追加のシグニチャーを作成する必要はありません。

この関数の結果のデータ型は VARCHAR(128) です。 *expression* が NULL 値の可能性がある場合は、結果も NULL 値になる可能性があります。 *expression* が NULL であれば、結果も NULL 値になります。 TYPE_NAME 関数を使用して、TYPE_SCHEMA が戻したスキーマ名に関連するタイプ名を判別してください。

関連資料:

• 456 ページの『TYPE_NAME』

UCASE または UPPER

スキーマは SYSIBM です。 (この関数の SYSFUN バージョンでは、上向き互換性が引き続き有効です。この点の説明については、バージョン 5 の資料を参照してください。)

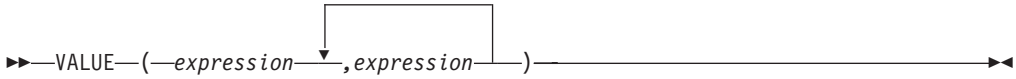
UCASE または UPPER 関数は、最初の引き数 (*char-string-exp*) だけが指定されるという点を除けば、 TRANSLATE 関数と同じです。

Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

関連資料:

- 451 ページの『TRANSLATE』

VALUE



スキーマは SYSIBM です。

VALUE 関数は、NULL 値以外の最初の引き数を戻します。

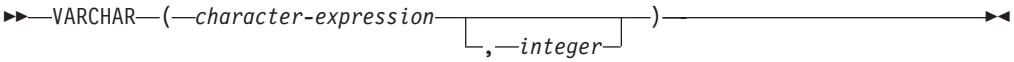
VALUE は COALESCE の同義語です。

関連資料:

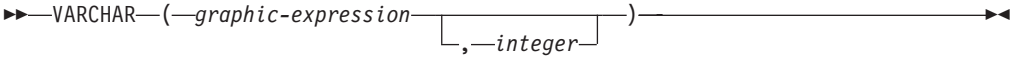
• 311 ページの『COALESCE』

VARCHAR

文字 → VARCHAR:



GRAPHIC → VARCHAR:



日付/時刻 → VARCHAR:

スキーマは SYSIBM です。

VARCHAR 関数は、以下の可変長文字ストリング表記を戻します。

- 文字ストリング。これは、最初の引き数がいずれかのタイプの文字ストリングの場合です。
- GRAPHIC ストリング (Unicode データベースのみ)。これは、最初の引き数がいずれかのタイプの GRAPHIC ストリングの場合です。
- 日付/時刻。これは、引き数が日付、時刻、またはタイム・スタンプの場合です。

Unicode データベースでは、複数バイト文字を介して出力ストリングが途中で切り捨てられると、次のようになります。

- 入力が文字ストリングであった場合、部分的な文字は 1 つ以上のブランクに置き換えられます。
- 入力が GRAPHIC ストリングであった場合、部分的な文字は空ストリングに置き換えられます。

このどちらの動作も過信しないでください。今後のリリースで変更される可能性があるからです。

関数の結果は、可変長文字ストリングです。最初の引き数が NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。最初の引き数が NULL 値の場合には、結果も NULL 値です。

文字→ VARCHAR

character-expression
32 672 バイトの最大長の文字ストリング・データ型でなければならない値をもつ式。

integer
結果の可変長文字ストリングの長さ属性。値は 0 から 32 672 の範囲でなければなりません。この引き数を指定しない場合、結果の長さ属性は引き数の長さ属性と同じになります。

GRAPHIC → VARCHAR

graphic-expression
LONG VARGRAPHIC または DBCLOB 以外の文字ストリングのデータ型である必要のある値をもち、しかも 16 336 個の 2 バイト文字を最大長としてもつ式。

integer
結果の可変長文字ストリングの長さ属性。値は 0 から 32 672 の範囲でなければなりません。この引き数を指定しない場合、結果の長さ属性は引き数の長さ属性と同じになります。

日付/時刻 → VARCHAR

datetime-expression
日付、時刻、またはタイム・スタンプのいずれかのデータ型でなければならない値をもつ式。

例:

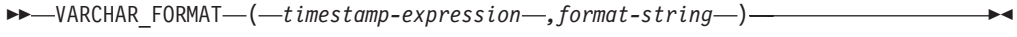
- VARCHAR(8) と定義されたホスト変数 JOB_DESC (VARCHAR(8)) を、社員 Dolores Quintana に関するジョブ記述と同等の VARCHAR (JOB 列の値) に設定します。

```
SELECT VARCHAR(JOB)
  INTO :JOB_DESC
  FROM EMPLOYEE
  WHERE LASTNAME = 'QUINTANA'
```

関連資料:

- 305 ページの『CHAR』

VARCHAR_FORMAT



スキーマは SYSIBM です。

VARCHAR_FORMAT 関数は、文字テンプレートを使ってフォーマットされたタイム・スタンプの文字表現を戻します。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

timestamp-expression
タイム・スタンプを戻す式。引き数は、タイム・スタンプであるか、または CLOB でも LONG VARCHAR でもないタイム・スタンプのストリング表記でなければなりません。 (*string-expression* がタイプ無しパラメーター・マーカーの場合、タイプは TIMESTAMP であると想定されます。) ストリング式は、254 未満の最大長をもつ CHAR または VARCHAR の値を戻します (SQLSTATE 42815)。先頭および末尾のブランクは *string-expression* から除かれ、ブランクが除かれたそのサブストリングは、 *format-string* によって指定されたフォーマットを使うタイム・スタンプと解釈されます。先頭のゼロは、年を除くすべてのタイム・スタンプ・コンポーネントから省略することができます。それらのコンポーネントでは、先頭のゼロの代わりにブランクを使用できます。たとえば、以下のストリングはいずれも、'YYYY-MM-DD HH24:MI:SS' というフォーマットで 9 a.m. on January 1, 2000 (2000 年 1 月 1 日午前 9 時) を表したものですが、すべて受け入れられます。

```
'2000-1-01 09:00:00'    (月が 1 桁)
'2000- 1-01 09:00:00'   (月が 1 桁で、前にブランクがある)
'2000-1-1 09:00:00'     (月および日が 1 桁)
'2000-01-01 9:00:00'    (時間が 1 桁)
'2000-01-01 09:0:0'     (分および秒が 1 桁)
'2000- 1- 1 09: 0: 0'   (月、日、分および秒が 1 桁で、
                         前にブランクがある)
'2000-01-01 09:00:00'   (各エレメントが最大桁数)
```

format-string
結果のフォーマット方法に関するテンプレートの入った文字定数。書式制御ストリングの長さは、254 (SQLSTATE 42815) 以下でなければなりません。先頭および末尾のブランクは *format-string* から除かれ、ブランクが除かれたそのサブストリングは、タイム・スタンプ値の有効なテンプレートになるはずです (SQLSTATE 42815)。 *format-string* の内容は英大文字小文字混合で指定できます。

有効な書式制御ストリングは次のとおりです。

```
'YYYY-MM-DD HH24:MI:SS'
```

YYYY は 4 桁の年の値を表し、 *MM* は 2 桁の月の値を表します (01 から 12。 January (1 月) = 01)。 *DD* は 2 桁の日の値を表し (01 から 31)、 *HH24* は 2 桁の時間の値を 表します (00 から 24。 24 時間制の場合、分および秒の値はゼロになります)。 *MI* は 2 桁の分の値を表し (00 から 59)、 *SS* は 2 桁の秒の値を表します (00 から 59)。

この関数の結果は、形式設定されたタイム・スタンプ式からなる可変長文字ストリングです。また、書式制御ストリングは、長さ属性や実際の結果の長さも決定しま

す。 format-string が 'YYYY-MM-DD HH24:MI:SS' の場合、長さ属性は 19 です。結果は、19 文字で次のような形式になります。

```
YYYY-MM-DD HH:MI:SS
```

たとえば、10 a.m. on January 1, 2000 (2000 年 1 月 1 日午前 10 時) という日時の場合、 'YYYY-MM-DD HH24:MI:SS' というフォーマットを使うと以下が戻されます。

```
'2000-01-01 10:00:00'
```

月および日の値が 1 桁のみを必要とする場合でも、この例では、それぞれの有効数字の前にゼロが付いています。また、分と秒の値のどちらもゼロの場合でも、それぞれに最大桁数が使用され、結果のこれらの部分にはそれぞれ '00' が戻されます。

最初の引き数を NULL にすることができる場合、結果も NULL にすることができます。最初の引き数が NULL であれば、結果は NULL 値になります。結果の CCSID は、システムの SBCS CCSID です。

例:

- 名前が 'SYSU' で始まるすべてのシステム表の名前と作成タイム・スタンプを表示します。

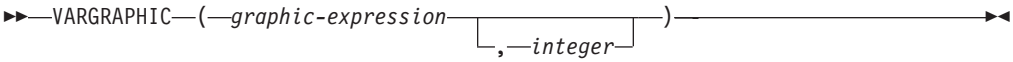
```
SELECT VARCHAR(name, 20) AS TABLE_NAME,
  VARCHAR_FORMAT(ctime, 'YYYY-MM-DD HH24:MI:SS') AS CREATION_TIME
  FROM SYSCAT.TABLES
  WHERE name LIKE 'SYSU%'
```

この例では、以下を戻します。

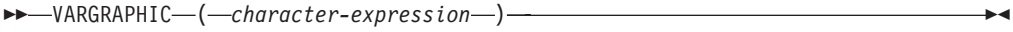
```
TABLE_NAME           CREATION_TIME
-------------------- -------------------
SYSUSERAUTH          2000-05-19 08:18:56
SYSUSEROPTIONS       2000-05-19 08:18:56
```

VARGRAPHIC

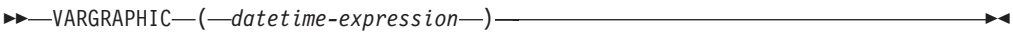
GRAPHIC → VARGRAPHIC:



文字→ VARGRAPHIC:



日付/時刻 → VARGRAPHIC:



スキーマは SYSIBM です。

VARGRAPHIC 関数は、以下の可変長 GRAPHIC ストリング表記を戻します。

- GRAPHIC ストリング。これは、最初の引き数がいずれかの型の GRAPHIC ストリングの場合です。
- 文字ストリング。最初の引き数がいずれかの型の文字ストリングの場合には、 1 バイト文字は 2 バイト文字に変換されます。
- 日付/時刻 (Unicode データベースのみ)。これは、引き数が日付、時刻、またはタイム・スタンプの場合です。

Unicode データベースでは、指定した引き数が文字ストリングであると、まず GRAPHIC ストリングに変換されてから、関数が実行されます。最後の文字が高サロゲートになるように出力ストリングが切り捨てられた場合、そのサロゲートはブランク文字 (X'0020') に変換されます。この動作を過信しないでください。今後のリリースで変更される可能性があるからです。

この関数の結果は、可変長 GRAPHIC ストリング (VARGRAPHIC データ型) です。最初の引き数を NULL にすることができる場合、結果も NULL にすることができます。最初の引き数が NULL であれば、結果は NULL 値になります。

GRAPHIC → VARGRAPHIC

graphic-expression
GRAPHIC ストリング値を戻す式。

integer
結果の VARGRAPHIC データ型の長さ属性を指定する整数値。値は 0 から 16 336 の範囲でなければなりません。値を指定しない場合の結果の長さ属性は、最初の引き数の長さ属性と同じになります。

GRAPHIC 式の長さが結果の長さ属性より長い場合、結果は切り捨てられます。その場合、切り捨てられた文字がすべてブランクで、 GRAPHIC 式が LONG ストリング (LONG VARGRAPHIC または DBCLOB) でない限り、警告 (SQLSTATE 01004) が戻されます。

文字→ VARGRAPHIC

character-expression
LONG VARCHAR または CLOB 以外の文字ストリングのデータ型である必要のある値をもち、しかも 16 336 バイトを最大長としてもつ式。

結果の長さ属性は、引き数の長さ属性と同じになります。

character-expression 内の各 1 バイト文字は、結果においては、それに相当する 2 バイト表記または 2 バイト置換文字に変換されます。 *character-expression* 内の各 2 バイト文字は、それ以外の変換なしでマップされます。 2 バイト文字の最初の 1 バイトが *character-expression* の最後のバイトとして示される場合、それは 2 バイトの置換文字に変換されます。 *character-expression* 内の文字順序はそのまま維持されます。

Unicode データベースの場合、この関数は文字ストリングをオペランドのコード・ページから UCS-2 へと変換します。 2 バイト文字をはじめとして、オペランドのすべての文字が変換されます。 2 番目の引き数の値を指定すると、結果のストリングに必要な長さが指定されます (UCS-2 文字)。

VARGRAPHIC 関数による 2 バイト・コード・ポイントへの変換は、オペランドのコード・ページに基づいています。

オペランドの 2 バイト文字は変換されません。その他の文字はすべて、それに対応する同等の 2 バイト文字に変換されます。対応する 2 バイトで相当するものが存在しない場合には、コード・ページ用の 2 バイト置換文字が使用されます。

1 つ以上の 2 バイト置換文字が結果内に戻されても警告やエラー・コードは生成されません。

日付/時刻 → VARGRAPHIC

datetime-expression
日付、時刻、またはタイム・スタンプのいずれかのデータ型でなければならない値をもつ式。

関連資料:

- 366 ページの『GRAPHIC』
- 801 ページの『付録 N. 日本語および中国語 (繁体字) の拡張 UNIX コード (EUC) の考慮事項』

WEEK

引き数の年間通算週番号を 1 から 54 の範囲の整数値として戻します。週は日曜日から始まります。

引き数は、日付またはタイム・スタンプであるか、または CLOB でも LONG VARCHAR でもない日付あるいはタイム・スタンプの有効な文字ストリング表記でなければなりません。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

関数の結果は INTEGER になります。結果は NULL 値になることがあります。引き数が NULL 値である場合、その結果は NULL 値になります。

WEEK_ISO

スキーマは SYSFUN です。

引き数の年間通算週番号を、1 から 53 の範囲の整数値で戻します。週は月曜日から始まり、常に 7 日から成ります。第 1 週は、木曜日の入った年の第 1 週目であり、それは 1 月 4 日の入った第 1 週と同等です。よって、年の初めの 3 日間までを、前の年の最終週と表すことができます。逆に、年の最後の 3 日間までを、翌年の最初の週と表すこともできます。

引き数は、日付またはタイム・スタンプであるか、または CLOB でも LONG VARCHAR でもない日付あるいはタイム・スタンプの有効な文字ストリング表記でなければなりません。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

関数の結果は INTEGER になります。結果は NULL 値になることがあります。引き数が NULL 値である場合、その結果は NULL 値になります。

例:

以下のリストは、WEEK_ISO および DAYOFWEEK_ISO の結果例です。

```
DATE       WEEK_ISO    DAYOFWEEK_ISO
---------- ----------- -------------
1997-12-28          52             7
1997-12-31           1             3
1998-01-01           1             4
1999-01-01          53             5
1999-01-04           1             1
1999-12-31          52             5
2000-01-01          52             6
2000-01-03           1             1
```

YEAR

スキーマは SYSIBM です。

YEAR 関数は、指定された値の年の部分を戻します。

引き数は、日付、タイム・スタンプ、日付期間、タイム・スタンプ期間であるか、または CLOB でも LONG VARCHAR でもない日付あるいはタイム・スタンプの有効な文字ストリング表記でなければなりません。 Unicode データベースでは、指定した引き数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

この関数の結果は長精度整数 (large integer) です。引き数が NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引き数が NULL であれば、結果は NULL 値です。

その他の規則は、指定した引き数のデータ型に応じて以下のように異なります。

- 引き数が日付、タイム・スタンプ、または日付やタイム・スタンプの有効なストリング表記の場合
  - 結果は、指定した値の年の部分 (1 から 9 999 の整数) になります。
- 引き数が日付期間またはタイム・スタンプ期間の場合
  - 結果は、指定した値の年の部分 (-9 999 から 9 999 の整数) になります。ゼロ以外の結果の符号は、引き数と同じになります。

例:

- PROJECT 表から、同一暦年内に開始 (PRSTDATE) および終了 (PRENDATE) が予定されているプロジェクトを全選択します。

```
SELECT * FROM PROJECT
 WHERE YEAR(PRSTDATE) = YEAR(PRENDATE)
```

- PROJECT 表から、1 年未満での完了が予定されているプロジェクトを全選択します。

```
SELECT * FROM PROJECT
 WHERE YEAR(PRENDATE - PRSTDATE) < 1
```

表関数

表関数は、ステートメントの FROM 文節でしか使用できません。表関数は、表の列を戻します。これは、単純な CREATE TABLE ステートメントが作成する表に似ています。表関数はスキーマ名で修飾することができます。

モニター表関数はすべて、現行セッションで使用しているものとは異なる別個のインスタンス接続を使用します。したがって、デフォルトのデータベース・マネージャー・モニター・スイッチのみが有効になります。現行セッションまたはアプリケーションから動的にオンまたはオフにできるどのモニター・スイッチも無効になります。

プロシージャー

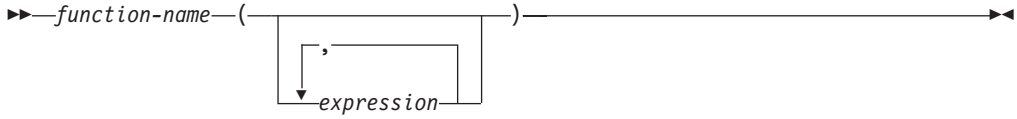
プロシージャーとは、 SQL CALL ステートメントを使って呼び出すことのできるアプリケーション・プログラムです。プロシージャーはプロシージャー名で指定され、プロシージャー名の後には括弧で囲まれた引き数が付くこともあります。

プロシージャーの引き数 (複数の場合あり) は個別のスカラー値で、タイプや意味が異なっていることもあります。引き数を使って、プロシージャーに値を渡したり、プロシージャーから戻り値を受け取ったりします (またはその両方を行います)。

ユーザー定義プロシージャーとは、 CREATE PROCEDURE ステートメントを使って SYSCAT.ROUTINES のデータベースに登録されるプロシージャーです。このような関数のセットは、1 つは SYSFUN という名前のスキーマで、もう 1 つは SYSPROC というスキーマで、データベース・マネージャーに付属しています。

プロシージャーは、スキーマ名で修飾することができます。

ユーザー定義関数



ユーザー定義関数 (UDF) は、SQL 言語の既存の組み込み関数に対する拡張または追加です。 ユーザー定義関数は、呼び出されるたびに単一値を戻すスカラー関数、互いに似通った一連の値を渡されてその中から単一値を戻す列関数、1 行を戻す行関数、または表を戻す表関数のいずれでかです。

SYSFUN および SYSPROC スキーマでは、多数のユーザー定義関数が提供されています。

UDF が既存の列関数をソースとして派生される場合にのみ、 UDF は列関数になります。 UDF は、修飾または無修飾の関数名およびその後に括弧で囲んだその関数の引き数を指定することによって参照します。データベースに登録されたユーザー定義の列関数またはスカラー関数は、組み込み関数を使用できるのと同じコンテキストで参照することができます。ユーザー定義の行関数は、ユーザー定義タイプのトランスフォーム関数として登録しておく場合に限り、暗黙的に参照できます。データベースに登録されたユーザー定義の表関数は、 SELECT ステートメントの FROM 文節でのみ参照することができます。

関数の引き数の数および位置は、データベースに登録された時点のユーザー定義関数に指定されたパラメーターと対応していなければなりません。さらに、引き数は、対応する定義済みパラメーターのデータ型にプロモート可能なデータ型でなければなりません

関数の結果は、RETURNS 文節に指定されます。 RETURNS 文節 (UDF の登録時に定義される) は、関数が表関数かどうかを決定します。関数登録時に RETURNS NULL ON NULL INPUT 文節が指定されていると (あるいはデフォルトでそうなっていると)、引き数のいずれかが NULL 値の場合には、結果は NULL 値になります。表関数の場合、これは、戻される表は行を備えていない (つまり、空の表) という意味に解釈されます。

以下に、ユーザー定義関数例をいくつか示します。

- ADDRESS というスカラー UDF は、スクリプト・フォーマットで保管されているレジュームからホーム・アドレスを抽出します。この ADDRESS 関数には、CLOB 引き数を指定し、VARCHAR(4000) の値が戻されます。

  ```
  SELECT EMPNO, ADDRESS(RESUME) FROM EMP_RESUME
    WHERE RESUME_FORMAT = 'SCRIPT'
  ```

- 表 T2 には、数値列 A があります。前の例の ADDRESS というスカラー UDF を以下のように呼び出すと、

  ```
  SELECT ADDRESS(A) FROM T2
  ```

  名前が一致して引き数からプロモート可能なパラメーターを持つ関数がないので、エラー (SQLSTATE 42884) が生じます。

- WHO という表 UDF は、ステートメントの実行時にアクティブであった、サーバー・マシン上のセッションに関する情報を戻します。 WHO 関数は、キーワー

ド TABLE および必須相関変数からなる FROM 文節内で呼び出されます。WHO() 表の列名は、CREATE FUNCTION ステートメントで定義されます。

```
SELECT ID, START_DATE, ORIG_MACHINE
FROM TABLE( WHO() ) AS QQ
WHERE START_DATE LIKE 'MAY%'
```

関連資料:

- 474 ページの『副選択』
- 「*SQL リファレンス 第 2 巻*」の『CREATE FUNCTION ステートメント』

第 4 章 照会

SQL 照会

照会 は結果表を指定します。照会は、いくつかの特定の SQL ステートメントのコンポーネントです。照会には、次の 3 つの形式があります。

- 副選択
- 全選択
- select ステートメント

許可

照会で参照される表、ビュー、またはニックネームのそれぞれに対して、ステートメントの許可 ID は少なくとも以下の 1 つを持っている必要があります。

- SYSADM または DBADM 権限
- CONTROL 特権
- SELECT 特権

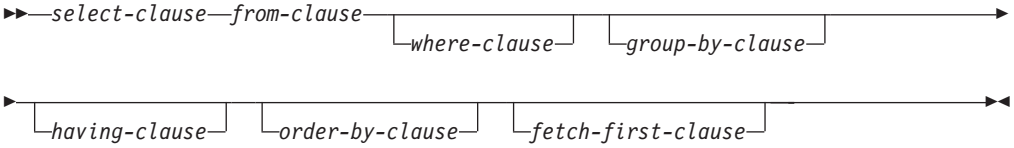
PUBLIC を除き、静的 SQL ステートメントに入っている照会のグループ特権はチェックされません。

ニックネームの場合、ニックネームが参照するオブジェクトでのデータ・ソースの許可要件は、照会が処理される時に適用されます。ステートメントの許可 ID は、データ・ソースに存在する別の許可 ID にマップすることができます。

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『SELECT INTO ステートメント』

副選択



副選択 は、全選択のコンポーネントの 1 つです。

副選択は、FROM 文節で指定される表、ビュー、またはニックネームから派生する結果表を指定します。この派生の方法は、各操作の結果が次の演算の入力になるような、一連の操作として記述することができます。 (これは、副選択を記述する 1 つの方法にすぎません。派生操作を実行するために使用されるメソッドは、この記述とはまったく異なる場合があります。)

副選択の文節は以下の順序で処理されます。

1. FROM 文節
2. WHERE 文節
3. GROUP BY 文節
4. HAVING 文節
5. SELECT 文節
6. ORDER BY 文節
7. FETCH FIRST 文節

ORDER BY または FETCH FIRST 文節を備えた副選択は、以下では指定できません。

- ビューの最外部の全選択。
- マテリアライズ照会表。
- 副選択が括弧で囲まれていない場合。

たとえば、以下は無効です (SQLSTATE 428FJ)。

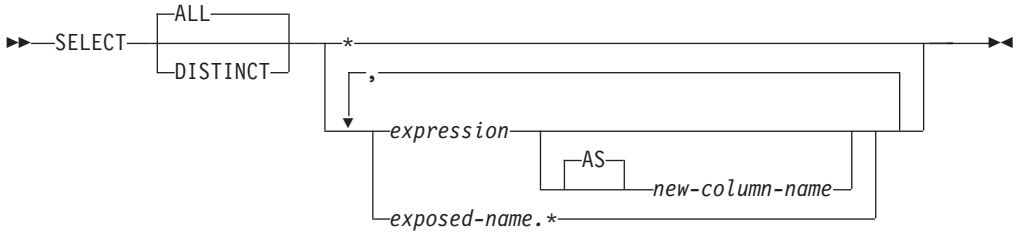
```
SELECT * FROM T1
   ORDER BY C1
UNION
SELECT * FROM T2
   ORDER BY C1
```

以下の例は有効です。

```
(SELECT * FROM T1
   ORDER BY C1)
UNION
(SELECT * FROM T2
   ORDER BY C1)
```

注: 副選択内の ORDER BY 文節は、照会によって戻される行の順序には影響を与えません。 ORDER BY 文節が影響を与えるのは、最外部の全選択で指定された場合に戻される行の ORDER BY だけです。

SELECT 文節



SELECT 文節は、最終結果表の列を指定します。列値は、*選択リスト* を R に適用することによって作成されます。選択リストとは、SELECT 文節に指定された名前または式であり、 R は副選択のうち直前の操作の結果です。たとえば、指定されている文節が SELECT、FROM、および WHERE だけの場合、 R は WHERE 文節の結果になります。

ALL
最終結果表の行すべてをそのまま保持し、冗長な重複を削除しません。これはデフォルトです。

DISTINCT
最終結果表の中に重複行があれば、その中の 1 つを除き、それ以外のすべてを削除します。 DISTINCT を使用した場合、結果表のストリング列を LONG VARCHAR、LONG VARGRAPHIC、DATALINK、LOB といったタイプにすることはできず、それらのタイプのいずれかを基にした特殊タイプ、または構造化タイプにすることもできません。 DISTINCT は、1 つの副選択で複数回使用することができます。これには、SELECT DISTINCT、選択リストまたは HAVING 文節の列関数での DISTINCT の使用、および副選択の副照会などがあります。

2 つの行が互いに重複していると言えるのは、最初の行の各値が 2 番目の行の対応する値に等しい場合だけです。重複を判別する場合、2 つの NULL 値は等しいものと見なされます。

選択リストの表記法

* 表 R の列を識別する名前のリストを表します。リスト内の最初の名前は R の最初の列、2 番目の名前は R の 2 番目の列、というようになります。

名前のリストは、その SELECT 文節の入ったプログラムのバインド時に確立されます。したがって、* (アスタリスク) では表参照の入ったステートメントのバインド後に表に追加された列は識別されません。

expression
結果列の値を指定します。有効な SQL 言語エレメントである任意の式にできますが、普通は列名を入れます。選択リストで使用される各列名は、 R の列をあいまいなところなく識別するものでなければなりません。

new-column-name または **AS** *new-column-name*
列名の名前を指定または変更します。この名前は修飾してはならず、ユニークである必要もありません。列名の後の使用方法は、次のように限定されています。

- AS 文節に指定された新しい列名 (new-column-name) は、その名前がユニークであれば、ORDER BY 文節で使用することができます。
- 選択リストの AS 文節に指定された新しい列名を、副選択の他の文節 (WHERE 文節、GROUP BY 文節、または HAVING 文節) で使用することはできません。
- AS 文節に指定された新しい列名を、UPDATE 文節で使用することはできません。
- AS 文節に指定された新しい列名は、ネストした表式、共通表式、および CREATE VIEW の全選択の外部で認識されます。

*name.**

結果表の列を指定する名前のリストを表します。この名前は *exposed-name* によって示されます。 *exposed-name* は、表名、ビュー名、ニックネーム、または相関名のいずれかにすることができ、FROM 文節で指定された表、ビュー、またはニックネームを指定するものでなければなりません。リスト内の最初の名前は表、ビュー、あるいはニックネームの最初の列、 2 番目の名前は表、ビュー、またはニックネームの 2 番目の列を識別する、というようになります。

名前のリストは、その SELECT 文節の入ったステートメントのバインド時に確立されます。したがって、ステートメントのバインド後に表に追加された列は、* によって識別されません。

SELECT の結果の列の数は、演算形式の選択リスト (つまり、ステートメントの準備時に設定されたリスト) の式の数と同じであり、 500 (4K ページ・サイズの場合) または 1012 (8K、16K、 32K ページ・サイズの場合) を超えることはできません。

ストリング列に関する制限

選択リストの制限については、『Restrictions Using Varying-Length Character Strings』 を参照してください。

選択リストの適用

選択リストを R に適用した結果は、 GROUP BY または HAVING が使用されているかどうかによって異なる場合があります。その結果について、次の 2 つのリストで説明します。

GROUP BY または HAVING が使用されている場合:

- 選択リストで使用される式 *X* (列関数ではない) には、以下を指定した GROUP BY 文節が必要です。
  - 各列名が列 R を明確に識別する*グループ化式* (488 ページの『GROUP BY 文節』を参照) または
  - 個別の*グループ化式* として *X* で参照される R の各列
- 選択リストは R のそれぞれのグループに対して適用され、その結果には、R にあるグループと同数の行が入ります。選択リストが R の 1 つのグループに適用されるとき、そのグループは、選択リストの列関数の引き数のソースになります。

GROUP BY または HAVING のどちらも使用されていない場合:

- 選択リストに列関数が入っていないか、または選択リスト内の各 *column-name* が列関数の中に指定されているか、あるいは相関列参照であるかのいずれかでなければなりません。
- 選択リストが列関数を備えていない場合、選択リストは R のそれぞれの行に対して適用され、その結果には R にある行と同数の行が示されます。
- 選択リストが列関数のリストである場合、関数の引き数は R から与えられ、選択リストを適用した結果は 1 行となります。

どちらの場合も、結果の *n* 番目の列には、命令形式の選択リストにある *n* 番目の式を適用することによって指定された値が入ります。

結果列の NULL 属性: 結果列は、以下から得られた場合には、NULL 値を使用できません。

- NULL 値が許されない列
- 定数
- COUNT または COUNT_BIG 関数
- 標識変数を持たないホスト変数
- NULL を使用できるオペランドを組み込まれていないスカラー関数または式

結果列が以下から得られた場合は、NULL 値を使用できます。

- COUNT または COUNT_BIG 以外の列関数
- NULL 値が可能な列
- NULL が可能なオペランドが組み込まれたスカラー関数または式
- 等しい値を引き数とする NULLIF 関数
- 標識変数を持つホスト変数
- 選択リスト内の対応項目の少なくとも 1 つが NULL 可能な場合のセット演算の結果
- 演算式から得られた演算式またはビューの列で、そのデータベースが DFT_SQLMATHWARN を yes に 設定して構成されている。
- 間接参照操作

結果列の名前:

- AS 文節が指定されている場合、結果列の名前は、AS 文節に指定された名前になります。
- AS 文節が指定されておらず、結果列が列から派生している場合、結果列の列名は、その列の非修飾名になります。
- AS 文節が指定されておらず、結果列が間接参照操作を使用して派生している場合、結果列の列名がその間接参照操作のターゲット列の非修飾名になります。
- それ以外の結果列には、名前が付けられません。システムは、これらの列に対して一時的な数字を (文字ストリングとして) 割り当てます。

結果列のデータ型: SELECT の結果の各列は、その列の派生元の式のデータ型となります。

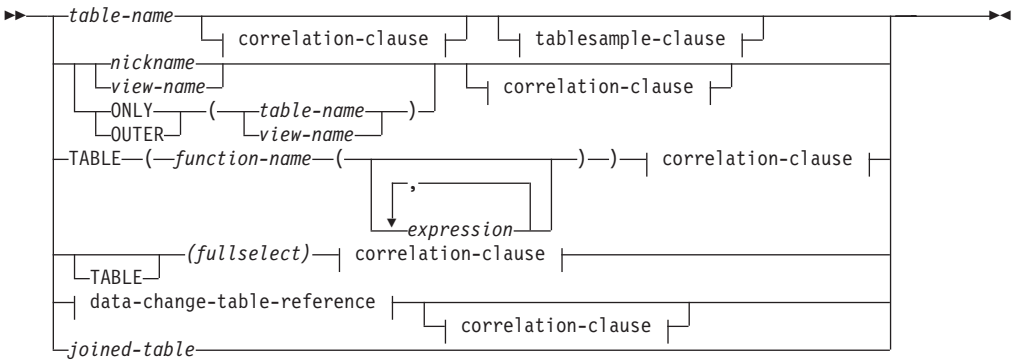
式	結果列のデータ型
数値列の名前	列のデータ型と同じ。DECIMAL 列の場合は精度と位取りも同じ。
整数定数	INTEGER。
10 進定数	定数の精度および位取りを持つ DECIMAL。
浮動小数点定数	DOUBLE。
任意の数値変数の名前	変数のデータ型と同じで、10 進変数については精度と位取りも同じ。
n バイトを表す 16 進定数	VARCHAR(n); コード・ページはデータベース・コード・ページになります。
ストリング列の名前	列のデータ型と同じで、長さ属性も同じ。
ストリング変数の名前	変数のデータ型と同じ (同じ長さ属性); 変数のデータ型が SQL データ型と同じではない場合 (たとえば、 C において NUL 文字で終了するストリング)、結果列は可変長ストリングになります。
長さ *n* の文字ストリング定数	VARCHAR(*n*)。
長さ *n* の GRAPHIC ストリング定数	VARGRAPHIC(*n*)。
日付/時刻の列の名前	列のデータ型と同じ。
ユーザー定義タイプ列の名前	列のデータ型と同じ。
参照タイプ列の名前	列のデータ型と同じ。

FROM 文節

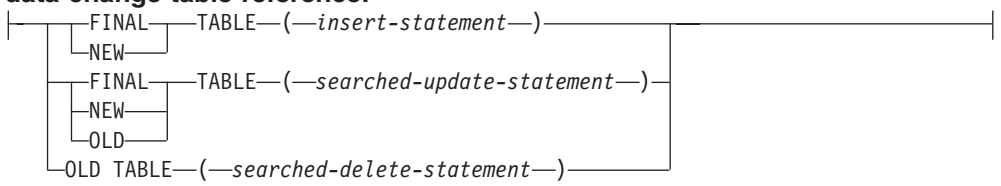
FROM 文節は、中間結果表を指定します。

表参照 (table-reference) が 1 つ指定されている場合、中間結果表は単に、その表参照の結果です。複数の表参照が指定されている場合、中間結果表は、指定された表参照の行の可能なすべての組み合わせ (カルテシアン積) からなります。結果の各行は、最初の表参照の行を 2 番目の表参照の行に連結し、それを 3 番目の表参照の行に連結し、以下同様の手順で連結した行です。結果の行数は、すべての表参照の行数の積です。*表参照* については、『表参照』 を参照してください。

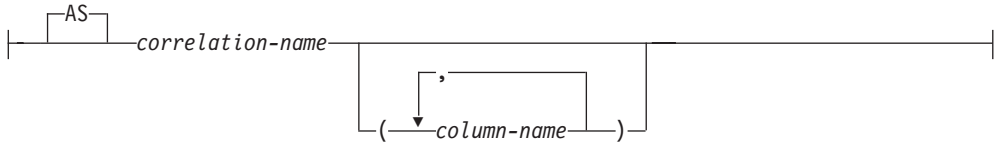
表参照



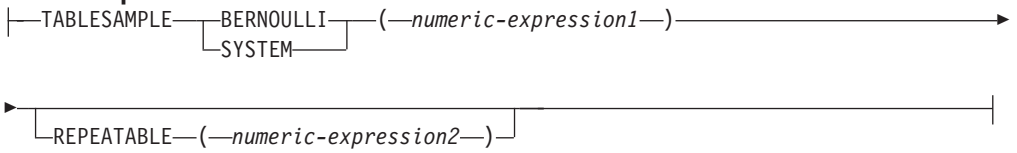
data-change-table-reference:



correlation-clause:



tablesample-clause:



表参照として指定する各 *table-name* (表名)、*view-name* (ビュー名)、または *nickname* (ニックネーム) は、アプリケーション・サーバーの既存の表、ビュー、またはニックネーム、あるいは表参照の入った全選択の前に定義された共通表式の *table-name* を指定するものでなければなりません。 *table-name* (表名) がタイプ表を参照する場合、その名前はその表およびその表の副表 (*table-name* の列のみ) の UNION ALL (全合併) を表します。同様に、*view-name* (ビュー名) がタイプ・ビューを参照する場合、その名前はそのビューおよびそのビューのサブビュー (*view-name* の列のみ) の UNION ALL を表します。

ONLY(*table-name*) または ONLY(*view-name*) を使用した場合は、適正な副表またはサブビューの行は組み込まれません。 ONLY に指定した *table-name* に副表がない場合、 ONLY(*table-name*) は *table-name* を指定することと同じになります。 ONLY に指定した *view-name* にサブビューがない場合、 ONLY(*view-name*) は *view-name* を指定することと同じになります。

OUTER(*table-name*) または OUTER(*view-name*) を指定した場合、それは仮想表を表します。 OUTER に指定した *table-name* または *view-name* に副表またはサブビュ

ーがない場合は、 OUTER を指定してもしなくても同じです。
OUTER(*table-name*) は、次のように *table-name* から派生します。

- 列には、*table-name* の列に続いて、副表によって導入された追加列 (副表がある場合) が組み込まれます。副表階層を階層深度の大きい順にスキャンし、追加列は右側に追加されます。共通の親を持つ副表の場合は、タイプの作成順にスキャンします。
- 行には、*table-name* のすべての行、およびその表の副表のすべての行が組み込まれます。その行の副表にない列には、NULL が戻されます。

上記の点は OUTER(*view-name*) にも当てはまります。その場合、*table-name* を *view-name* に、副表をサブビューに読み替えてください。

ONLY または OUTER を使用するときには、 *table-name* の副表または *view-name* のサブビューごとに、 SELECT 権限が必要です。

表参照として指定された各 *function-name* (関数名) およびその引き数のタイプは、アプリケーション・サーバーの既存の表関数に解決されなければなりません。

括弧内の全選択 (fullselect) とその後に続く相関名 (correlation name) は、*ネストされた表式* と呼ばれます。

joined-table (結合表) は、 1 つまたは複数の結合演算の結果である中間結果セットを指定します。詳細については、 486 ページの『結合表』を参照してください。

すべての表参照の直接的な名前はユニークでなければなりません。直接的な名前とは、以下の名前です。

- *相関名*
- 後に*相関名* の付いていない*表名*
- 後に*相関名* の付いていない*ビュー名*
- 後に*相関名* の付いていない*ニックネーム*
- 後に*相関名* の付いていない*別名*

各 *correlation-name* (相関名) は、直前の*表名*、*ビュー名*、*ニックネーム*、*関数名* の参照またはネストした表式の指定子と定義されます。表、ビュー、表関数、またはネストした表式の列に対する修飾参照では、必ず直接的な名前を使用しなければなりません。同じ表名、ビュー名、またはニックネームを 2 回指定する場合は、その少なくとも 1 回の指定の後には*相関名* を付ける必要があります。*相関名* は、表、ビュー、またはニックネームの*列*に対する参照を修飾するのに使用されます。*相関名* が指定されている場合、*表名*、*ビュー名*、*ニックネーム*、*関数名* の参照、あるいはネストした表式の列に名前を指定するために、*列名* を指定することもできます。

通常、表関数、およびネストされた表の式は、FROM 文節にのみ指定することができます。表関数およびネストされた表の式からの列は、選択リストの中および残りの副選択で、指定する必要のある相関名を使用して参照することができます。この相関名の有効範囲は、FROM 文節の他の表名、ビュー名、またはニックネームの相関名と同じです。ネストされた表式は、次の場合に使用することができます。

- ビューの代わりに使用して、ビューが作成されないようにする場合 (ビューを一般的に使用する必要がない場合)

- 必要な結果表がホスト変数に基づいたものである場合

全選択内にあるデータ変更ステートメントで参照されているか、またはターゲットとされているネストされた表の式の選択リストにある式は、以下が入っていない場合に限り、有効です。

- SQL データの読み取りまたは変更を行う関数
- 非決定性の関数
- 外部アクションを指定する関数
- OLAP 関数

ビューが、FROM 文節のデータ変更ステートメントで直接参照されている場合や、このステートメント内のネストされた表の式のターゲットとして参照されている場合、ビューは、シンメトリック (WITH CHECK OPTION が指定されている) であるか、 WITH CHECK OPTION ビューの制限を満たしている必要があります。

FROM 文節のデータ変更ステートメントのターゲットがネストされた表の式である場合、変更された行は再修飾されず、WHERE 文節の述部は再評価されません。また、ORDER BY 操作や FETCH FIRST 操作は再実行されません。

オプションの tablesample-clause を使用すると、指定された *table-name* のすべての内容ではなく、その *table-name* の行のランダム・サブセット (サンプル) をこの照会のために取得できます。このサンプリングは、 *where-clause* で指定されたすべての述部に加えて行われます。オプションの REPEATABLE 文節が指定されていない限り、照会を実行するたびに、通常、異なるサンプルが生成されます。ただし変性の場合、サンプル・サイズに比べて表が小さすぎるので、すべてのサンプルは同じ行を戻します。サンプル・サイズは括弧内の *numeric-expression1* で制御されます。これは、表の中で戻すべきおおよそのパーセント (P) を示します。サンプルを取得する方法は TABLESAMPLE キーワードの後に指定され、 BERNOULLI (ベルヌーイ) または SYSTEM のいずれかが可能です。どちらの方法を使用しても、実際のサンプルの正確な行数は照会を実行するたびに異なることがあります。しかし平均では、述部によって行数がさらに削減される前の段階で、表のおよそ P パーセントになります。

table-name はすでに保管されている表でなければなりません。マテリアライズ照会表 (MQT) の名前も指定できますが、 MQT の定義に関連している副選択または表式は指定できません。これは、データベース・マネージャーがその副選択に対応する MQT にアクセス・パスを指定するとは限らないためです。

セマンティクス的には、表のサンプリングは、述部の適用または結合の実行などの、他のすべての照会処理よりも前に実行されます。 1 つの照会の実行中にサンプリングされた同じ表に繰り返しアクセスする場合 (たとえば、ネストされたループ結合や相関副照会の場合)、常に同じサンプルが戻されます。 1 つの照会で複数の表をサンプリングすることができます。

BERNOULLI (ベルヌーイ) サンプリングではそれぞれの行が個別に考慮されます。サンプル内の各行は、他の行とは関係なく、P/100 の確率 (P は *numeric-expression1* の値) で組み込まれて、1 - P/100 の確率で除外されます。したがって、*numeric-expression1* の値が 10 (つまり 10 % のサンプル) と評価された場合、各行は 0.1 の確率で組み込まれて、0.9 の確率で除外されます。

SYSTEM サンプリングの場合、サンプリングを実行する最も効率的な方法をデータベース・マネージャーに判断させます。ほとんどの場合、*table-name* に SYSTEM サンプリングを適用すると、 *table-name* の各ページがサンプルに組み込まれる確率は P/100、除外される確率は 1 - P/100 です。組み込まれた各ページ内のすべての行がサンプルの対象となります。 *table-name* に SYSTEM サンプリングを適用する方が、検索対象のデータ・ページがより少なくなるため、ほとんどの場合 BERNOULLI サンプリングよりもかなり速く実行されます。ただし、集約関数 (たとえば SUM(SALES)) の見積精度が低くなる可能性が高くなります (特に、*table-name* の行が、その照会で参照されるいずれかの列にクラスター化されている場合)。場合によっては、 SYSTEM サンプリングを BERNOULLI サンプリングであるかのように実行した方がより効率的だとオプティマイザーが判断するかもしれません。たとえば、*table-name* の述部が索引によって適用され、サンプリング率 P よりもかなり限定的になる場合です。

numeric-expression1 は、 *table-name* から得られるサンプルのサイズを指定します (パーセントとして表されます)。これは定数式でなければならず、列、パラメーター・マーカー、またはホスト変数を中で使用することはできません。式では 100 以下の正数を評価する必要があります (1 と 0 の間の値でも差し支えありません)。たとえば、値 0.01 は 1 % 100 分の 1 を表し、平均して 10 000 行につき 1 行がサンプルに組み入れられることを意味します。 *numeric-expression1* が値 100 を評価する場合、 tablesample-clause が指定されていないかのように扱われます。*numeric-expression1* が NULL 値を評価したり、または 100 を超えたり 0 を下回ったりする値を評価する場合には、エラーが戻されます (SQLSTATE 2202H)。

照会の実行ごとにサンプリングを繰り返すのが望ましい場合があります。たとえば、レグレッション・テスト中や照会の「デバッグ」中などが該当します。これを行うには、REPEATABLE 文節を指定します。 REPEATABLE 文節では、括弧の中に *numeric-expression2* を指定する必要があります。この数式は、乱数発生ルーチンのシードと同じ役割を果たします。 *table-name* の tablesample-clause に REPEATABLE 文節を追加すれば、 (*numeric-expression2* に同じ値を使用して) その照会を繰り返し実行した場合、同じサンプルが戻されるようになります (もちろん、データそのものが更新、再編成、または再パーティションされないことが前提です)。複数の照会の間で *table-name* の同じサンプルが使われるようにするには、グローバル一時表の使用をお勧めします。他の方法として、複数の照会を 1 つの照会に結合することもできます。その場合、WITH 文節を使って定義された 1 つのサンプルへの複数の参照があります。

以下は、いくつかの例です。

例 1: 監査のために、 Sales 表から 10 % のベルヌーイ・サンプルを抽出します。

```
SELECT * FROM Sales
  TABLESAMPLE BERNOULLI(10)
```

例 2: 北東部 (Northeast) 地域での、それぞれの商品カテゴリーごとの売上総額 (Sales.Revenue) を計算します。その際、Sales 表のランダム 1 % SYSTEM サンプリングを使用します。セマンティクス的に、SUM はサンプルそのものに対する処理です。したがって、Sales 表全体の売上を推定するには、照会においてその SUM をサンプリング率 (0.01) で除算する必要があります。

```
SELECT SUM(Sales.Revenue) / (0.01)
  FROM Sales TABLESAMPLE SYSTEM(1)
  WHERE Sales.RegionName = 'Northeast'
  GROUP BY Sales.ProductCategory
```

例 3: REPEATABLE 文節を使って、上記の照会を変更し、照会が実行されるたびに同じ (ランダムな) 結果が得られるようにします。 (括弧で囲まれた定数は任意の値です。)

```
SELECT SUM(Sales.Revenue) / (0.01)
  FROM Sales TABLESAMPLE SYSTEM(1) REPEATABLE(3578231)
  WHERE Sales.RegionName = 'Northeast'
  GROUP BY Sales.ProductCategory
```

表関数参照

一般的に、表関数とその引き数値は、表やビューとまったく同じ方法で SELECT の FROM 文節で参照することができます。ただし、特殊な考慮事項が適用されます。

- 表関数の列名

  相関名 の後に代替列名を指定する場合を除いて、表関数の列名は、 CREATE FUNCTION ステートメントの RETURNS 文節に指定された列名になります。これは、 CREATE TABLE ステートメントに定義されている表の列名に類似したものです。

- 表関数解決

  表関数参照に指定された引き数は関数名と共に、*関数解決* と呼ばれるアルゴリズムによって、使用する正確な関数を判別するのに用いられます。これは、ステートメントで使用される他の関数 (たとえば、スカラー関数) の場合に行われるのと同じです。

- 表関数の引き数

  スカラー関数の引き数の場合と同じように、通常、表関数の引き数は有効な SQL 式です。次の例は正しい構文です。

  例 1:
  ```
  SELECT c1
  FROM TABLE( tf1('Zachary') ) AS z
  WHERE c2 = 'FLORIDA';
  ```

  例 2:
  ```
  SELECT c1
  FROM TABLE( tf2 (:hostvar1, CURRENT DATE) ) AS z;
  ```

  例 3:
  ```
  SELECT c1
  FROM t
  WHERE c2 IN
           (SELECT c3 FROM
            TABLE( tf5(t.c4) ) AS z  -- 直前の FROM 文節への
           )                         -- 相関参照
  ```

- SQL データを変更する表関数

  MODIFIES SQL DATA で指定された表関数は、SET ステートメントの副選択、SELECT INTO、または *row-fullselect* である *select-statement*、*common-table-expression*、または RETURN ステートメントで、最後の表参照としてのみ使用できます。 1 つの FROM 文節中で使用できる表関数は 1 つだけであり、表関数の引き数は、副選択内の他のすべての表参照と相関していなければなりません (SQLSTATE 429BL)。以下の例は、MODIFIES SQL DATA プロパティーをもった表関数の有効な構文です。

```
例 1:       SELECT c1
            FROM TABLE( tfmod('Jones') ) AS z

例 2:       SELECT c1
            FROM t1, t2, TABLE( tfmod(t1.c1, t2.c1) ) AS z

例 3:       SET var =
            (SELECT c1
            FROM TABLE( tfmod('Jones') ) AS z

例 4:       RETURN SELECT c1
            FROM TABLE( tfmod('Jones') ) AS z

例 5:       WITH v1(c1) AS
            (SELECT c1
            FROM TABLE( tfmod(:hostvar1) ) AS z)
            SELECT c1
            FROM v1, t1 WHERE v1.c1 = t1.c1
```

表参照における相関参照

相関参照は、ネストされた表の式や、表関数の引き数として使用することができます。両方の場合に適用される基本的な規則は、相関参照は、副照会の階層のより高いレベルにある*表参照* から行う必要があるということです。この階層には、FROM 文節の左から右への処理ですでに解決されている表参照が入っています。ネストされた表の式の場合、TABLE キーワードが全選択の前に指定されなければなりません。したがって、次の例は正しい構文です。

```
例 1:       SELECT t.c1, z.c5
            FROM t, TABLE( tf3(t.c2) ) AS z      -- FROM において
            WHERE t.c3 = z.c4;                   -- t が tf3 に先行するので、
                                                 -- t.c2 は既知

例 2:       SELECT t.c1, z.c5
            FROM t, TABLE( tf4(2 * t.c2) ) AS z  -- FROM において
            WHERE t.c3 = z.c4;                   -- t が tf3 に先行するので、
                                                 -- t.c2 は既知

例 3:       SELECT d.deptno, d.deptname,
                   empinfo.avgsal, empinfo.empcount
            FROM department d,
                 TABLE (SELECT AVG(e.salary) AS avgsal,
                               COUNT(*) AS empcount
                        FROM employee e           -- department が先行し、
                        WHERE e.workdept=d.deptno -- TABLE が指定されて
                       ) AS empinfo;              -- いないので、
                                                  -- d.deptno は既知
```

しかし、以下は正しくない例です。

```
例 4:       SELECT t.c1, z.c5
            FROM TABLE( tf6(t.c2) ) AS z, t     -- t.c2 の t を解決できない!
            WHERE t.c3 = z.c4;                  -- 上記例 1 と比較

例 5:       SELECT a.c1, b.c5
            FROM TABLE( tf7a(b.c2) ) AS a, TABLE( tf7b(a.c6) ) AS b
            WHERE a.c3 = b.c4;                   -- b.c2 の b を解決できない!

例 6:       SELECT d.deptno, d.deptname,
                   empinfo.avgsal, empinfo.empcount
            FROM department d,
                 (SELECT AVG(e.salary) AS avgsal,
                               COUNT(*) AS empcount
                        FROM employee e           -- department が先行し、
```

```
WHERE e.workdept=d.deptno -- TABLE が指定されて
) AS empinfo;             -- いないので、
                          -- d.deptno は不明
```

データ変更表参照

data-change-table-reference 文節は、中間結果表を指定します。この表は、文節に入っている検索済み UPDATE、検索済み DELETE、または INSERT ステートメントで直接変更された行に基づいています。 *data-change-table-reference* は、*select-statement*、 SELECT INTO ステートメント、または共通表式で使用される外部全選択の FROM 文節で、唯一の *table-reference* として指定できます。また、割り当てステートメントの唯一の全選択として *data-change-table-reference* を指定することもできます (SQLSTATE 428FL)。データ変更ステートメントのターゲットになる表やビューは、照会で参照される表やビューとして見なされるため、照会の許可 ID には、ターゲットとなる表やビューでの SELECT 特権が必要になります。

共通表式で定義される一時ビューは、 UPDATE、DELETE、または INSERT ステートメントのターゲットにできません (SQLSTATE 42807)。

FINAL TABLE
中間結果表の行がデータ変更ステートメントの完了時に表示されるとき、これらの行が、SQL データ変更ステートメントによって変更された一連の行を表すことを示します。 AFTER トリガーや参照制約があり、その結果として SQL データ変更ステートメントのターゲットになっている表に対して追加の操作が発生する場合は、エラーが戻されます (SQLSTATE 57058、SQLSTATE 560C6)。SQL データ変更ステートメントのターゲットがデータ変更タイプの INSTEAD OF トリガーで定義されたビューであった場合は、エラーが戻されます (SQLSTATE 428G3)。

NEW TABLE
中間結果表の行が、参照制約や AFTER トリガーのアプリケーションより前に SQL データ変更ステートメントによって変更された一連の行を表していることを示します。参照制約や AFTER トリガーに対する追加の処理のため、ステートメントの完了時にターゲット表にあるデータは、中間結果表のデータと一致しないことがあります。

OLD TABLE
中間結果表の行が、データ変更ステートメントのアプリケーションよりも前に存在していたために SQL データ変更ステートメントによって変更された一連の行を表すことを示します。

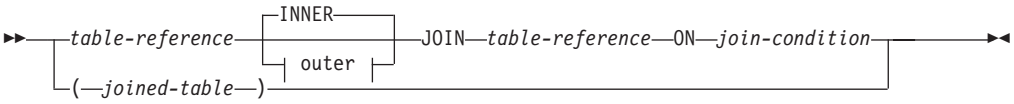
(searched-update-statement)
検索済み UPDATE ステートメントを指定します。 UPDATE ステートメントの WHERE 文節や SET 文節に、 UPDATE ステートメント外の列への相関参照を入れることはできません。

(searched-delete-statement)
検索済み DELETE ステートメントを指定します。 DELETE ステートメントの WHERE 文節に、 DELETE ステートメント外の列への相関参照を入れることはできません。

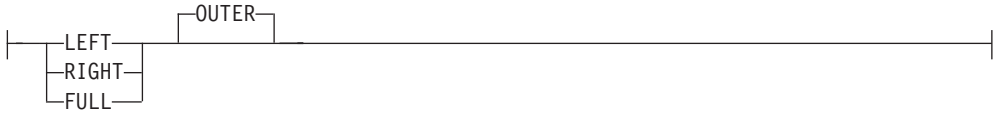
(insert-statement)
INSERT ステートメントを指定します。 INSERT ステートメントの全選択内で、 INSERT ステートメントの全選択に入っていない列への相関参照を使用することはできません。

data-change-table-reference の中間結果表の内容は、カーソルが開かれたときに決定されます。中間結果表には、指定されたターゲット表またはビューのすべての列を含め、処理されたすべての行が示されます。 SQL データ変更ステートメントのターゲット表またはビューのすべての列は、ターゲット表またはビューから列名を使用してアクセスできます。データ変更ステートメントで INCLUDE 文節が指定されている場合、中間結果表には、これらの追加の列が示されます。

結合表



outer:



結合表 (joined table) は、内部結合または外部結合のいずれかの結果である中間結果の表を指定します。この表は、結合演算子 INNER、LEFT OUTER、RIGHT OUTER、または FULL OUTER のいずれかをそのオペランドに適用して得ることができます。

内部結合は、表のクロス製品 (左側の表の各行を右側の表の各行に結合する) と見なすことができ、結合条件が真である行のみを保持します。結果表では、結合された表の一方または両方からの行が欠落している場合があります。外部結合には、内部結合が組み込まれて、このような欠落行を保持します。外部結合には、次の 3 種類のものがあります。

- ***左外部結合***。内部結合から欠落している左側の表の行が入っています。
- ***右外部結合***。内部結合から欠落している右側の表の行が入っています。
- ***全外部結合***。内部結合から欠落している左側および右側の表の行が入っています。

結合演算子の指定がない場合、INNER が暗黙の指定になります。複数の結合が行われる順序は、結果に影響を与えます。結合は、他の結合内にネストすることができます。結合の処理順序は、通常、左から右方向ですが、必要な結合条件の位置に基づきます。ネストされた結合の順序を読みやすくするために、括弧の使用をお勧めします。たとえば、次のようにします。

```
TB1 LEFT JOIN TB2 ON TB1.C1=TB2.C1
  RIGHT JOIN TB3 LEFT JOIN TB4 ON TB3.C1=TB4.C1
    ON TB1.C1=TB3.C1
```

これは、以下と同じです。

```
(TB1 LEFT JOIN TB2 ON TB1.C1=TB2.C1)
  RIGHT JOIN (TB3 LEFT JOIN TB4 ON TB3.C1=TB4.C1)
    ON TB1.C1=TB3.C1
```

結合表は、SELECT ステートメントの形式のいずれかが使用されるコンテキストであれば、どのようなコンテキストでも使用することができます。その SELECT ステートメントに結合表が入っている場合には、ビューまたはカーソルは読み取り専用です。

join-condition (結合条件) は *search-condition* (検索条件) です。ただし、以下の点で異なっています。

- 結合条件には、副照会 (スカラーなど) を入れることができません。
- 参照値がオブジェクト ID 列以外の場合、間接参照操作または DEREF 関数を組み込むことはできません。
- SQL 関数を組み入れることができません。
- *結合条件* の式で参照される列は、関連する結合のオペランド表のいずれかの列でなければなりません (同じ結合表の文節の範囲内)。
- 全外部結合の*結合条件* の式で参照される関数は、決定的なものでなければならず、外部アクションは持ちません。

結合条件が上記の規則にしたがっていない場合、エラーが生じます (SQLSTATE 42972)。

列参照は、列名の修飾子を解決するための規則を使用して解決されます。述部に適用されるのと同じ規則が、結合条件にも適用されます。

結合演算

結合条件 は、T1 と T2 のペアを指定します。ここで、T1 および T2 は、*結合条件* の JOIN 演算子の左と右のオペランド表です。 T1 および T2 の行のすべての組み合わせについて、*結合条件* が真であれば、T1 の行は T2 の行とペアになります。 T1 の行が T2 の行に結合する場合、結果の行は、T1 の行の値が T2 の行の値と連結された値で構成されます。この実行には、NULL 行の生成が関与する場合があります。表の NULL 行は、列が NULL 値を許すか否かに関係なく、表の各列の NULL 値で構成されます。

以下に、結合演算の結果を要約します。

- T1 INNER JOIN T2 の結果は、結合条件が真であるペアの行で構成されます。
- T1 LEFT OUTER JOIN T2 の結果は、結合条件が真であるペアの行、およびペアになっていない T1 の行ごとに、その行を T2 の NULL 行に連結したもので構成されます。 T2 から得られるすべての列には NULL 値を使用することができます。
- T1 RIGHT OUTER JOIN T2 の結果は、結合条件が真であるペアの行、およびペアになっていない T2 の行ごとに、その行を T1 の NULL 行に連結したもので構成されます。 T1 から得られるすべての列には NULL 値を使用することができます。
- T1 FULL OUTER JOIN T2 の結果は、ペアの行、およびペアになっていない T2 の行ごとにその行を T1 の NULL 行に連結したもの、およびペアになっていな

い T1 の行ごとにその行を T2 の NULL 行に連結したもので構成されます。 T1 および T2 から得られるすべての列には NULL 値を使用することができます。

WHERE 文節

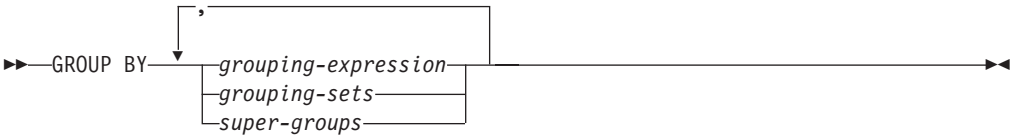
WHERE 文節は、 *search-condition (検索条件)* が真である R の行で構成される中間結果表を指定します。 R は、その副選択の FROM 文節の結果です。

search-condition は、以下の規則に適合していなければなりません。

- 各 *列名* は、R の列をあいまいなところなく指定するか、あるいは相関参照でなければなりません。副選択外の*表参照* の列を識別している*列名* は、相関参照となります。
- WHERE 文節が HAVING 文節の副照会に指定されていて、関数の引き数がグループに対する相関参照であるのでない限り、列関数を指定することはできません。

search-condition 内の副照会は、R の各行に対して実際に実行され、*search-condition* を R のその行に適用するときにその結果が使用されます。副照会が実際に R の各行に対して実行されるのは、その中に相関参照が組み込まれている場合だけです。実際、相関参照のない副照会は 1 回しか実行されませんが、相関参照のある副照会は、各行ごとに 1 回ずつ実行されます。

GROUP BY 文節



GROUP BY 文節は、R の行のグループ化で構成される中間結果表を指定します。R は、副選択のそれ以前の文節の結果です。

最も単純な形式では、 GROUP BY 文節に *grouping expression* (グループ化式) が入っています。グループ化式とは、R のグループ化の定義で使用される式 のことです。グループ化式に入っている各*列名* は、 R の列を明確に識別ししていなければなりません (SQLSTATE 42702 または 42703)。グループ化式には、スカラー全選択 (SQLSTATE 42822)、または可変の関数または外部処理を伴う関数 (SQLSTATE 42845) を組み入れることはできません。

複雑な形式の GROUP BY 文節には、 *grouping-sets* (グループ化集合) および *super-groups* (スーパー・グループ) が組み入れられます。これらの形式の説明については、それぞれ 489 ページの『グループ化集合』 と 490 ページの『スーパー・グループ』 を参照してください。

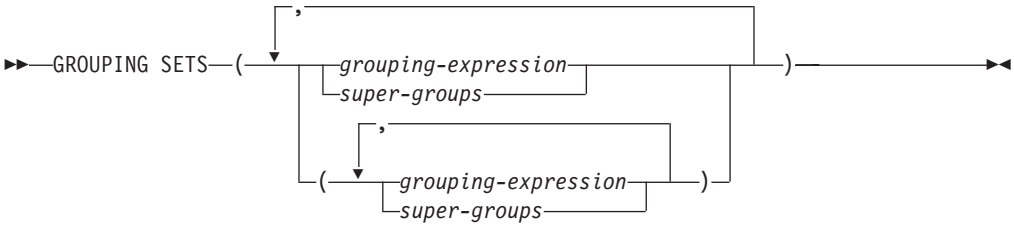
GROUP BY の結果は、いくつかの行グループの集まりです。この結果の各行は、*グループ化式* が等しい行の集合を表します。グループ化では、*グループ化式* の NULL 値はすべて等しいものと見なされます。

グループ化式 は、HAVING 文節の検索条件、SELECT 文節の式、または ORDER BY 文節の *sort-key-expression* (ソート・キー式) (詳細については、 495 ページの『ORDER BY 文節』 を参照) で使用することができます。いずれの場合も、その参照は各グループの 1 つの値だけを指定します。たとえば、*グループ化式* が *col1+col2* である場合、選択リストで使用できる式は、*col1+col2+3* になります。式の関連性規則では、類似した式 *3+col1+col2* を使用することを許可しません。ただし、対応する式を同じ順序で評価するために括弧を使用する場合を除きます。したがって、選択リストでは *3+(col1+col2)* も使用することができます。連結演算子を使用する場合、*グループ化式* は、選択リストで指定された式とまったく同じように使用しなければなりません。

グループ化式 に末尾のブランクの付いた可変長ストリングが入っている場合、そのグループの値は末尾のブランクの数が異なる場合があり、必ずしも同じ長さになるとはかぎりません。そのような場合でも、*グループ化式* への参照は各グループに 1 つの値だけを指定しますが、グループの値は使用可能な値の集合の中から任意に選択されます。したがって、結果値の実際の長さは予測できません。

前述のように、GROUP BY 文節が、 SELECT 文節で指定された列を式 (スカラー全選択、可変または外部処理関数) として直接参照することができない場合があります。そのような式を使用してグループ化を行うには、ネストした表式または共通表式を使用して、まず結果の列が式となる結果表を指定します。 ネストした表式の使用例については、 500 ページの例 A9 を参照してください。

グループ化集合



grouping-sets (グループ化集合) の指定により、複数のグループ化文節を単一ステートメントで指定することができます。これは、行の複数のグループを単一の結果セットにまとめたものと見なすことができます。これは、1 つのグループ化集合に対応する各副選択ごとに GROUP BY 文節を持つ複数の副選択を合併したものと論理的に等しくなります。グループ化集合は、単一のエレメント、もしくは括弧によって区切られる複数のエレメントのリストになります。この場合、エレメントはグループ化式、またはスーパー・グループのいずれかです。*グループ化集合* を使用して、基本表の単一パスを使用してグループを計算することができます。

グループ化集合 の指定により、単純な*グループ化式* を使用するか、またはより複雑な形式の*スーパー・グループ* を使用することができます。*スーパー・グループ* については、 490 ページの『スーパー・グループ』 を参照してください。

グループ化集合は、GROUP BY 演算の基礎的な構築ブロックであることに注意してください。単純な列を使用する単純な GROUP BY は、 1 つのエレメントを持つグループ化集合と見なすことができます。たとえば、次のようにします。

```
GROUP BY a
```

これは、以下と同じです。

```
GROUP BY GROUPING SETS((a))
```

および

```
GROUP BY a,b,c
```

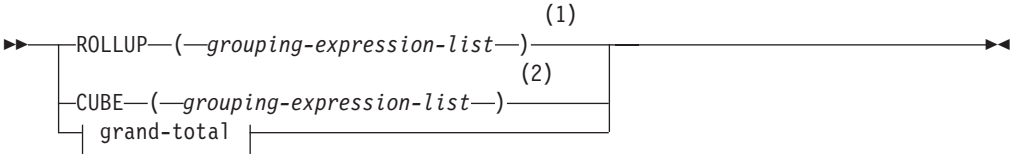
これは、以下と同じです。

```
GROUP BY GROUPING SETS((a,b,c))
```

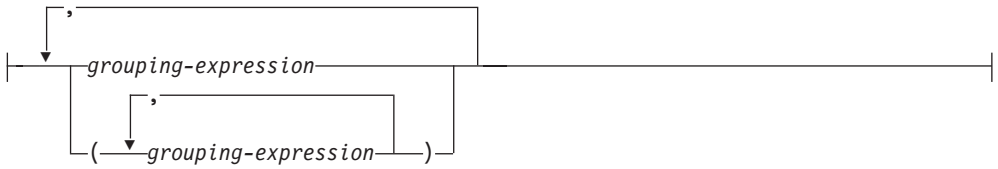
グループ化集合から除外される副選択の選択リストの非集約の列は、そのグループ化集合用に生成される各行の列に NULL 値を戻します。これは、集約が列の値を考慮せずに行われたことを表しています。

503 ページの例 C2 から 507 ページの例 C7 は、グループ化集合の使用法を示しています。

スーパー・グループ



grouping-expression-list:



grand-total:

注:

1 GROUP BY 文節で単独で使用される代替仕様は、 grouping-expression-list WITH ROLLUP です。

2 GROUP BY 文節で単独で使用される代替仕様は、 grouping-expression-list WITH CUBE です。

ROLLUP (*grouping-expression-list*)

ROLLUP grouping (ROLLUP グループ化) は GROUP BY 文節の拡張であり、「通常の」グループ化された行に加えて*小計* 行の入った結果セットを生成します。*小計* 行は、グループ化行を入手するのに使用されたのと同じ列関数を適用して値が得られる集合体が入っている「スーパー集約」行です。これらの行は小計が最も一般的な用途なので小計行と呼ばれますが、集約には任意の列関数を使用することができます。たとえば、 509 ページの例 C8では MAX および AVG が使用されます。

ROLLUP グループ化は、一連の*グループ化集合* です。 *n* 個のエレメントを持つ ROLLUP の一般的な仕様は、次のとおりです。

```
GROUP BY ROLLUP(C1,C2,...,Cn-1,Cn)
```

これは、以下と同じ意味になります。

```
GROUP BY GROUPING SETS((C1,C2,...,Cn-1,Cn)
                       (C1,C2,...,Cn-1)
                       ...
                       (C1,C2)
                       (C1)
                       () )
```

ROLLUP の *n* のエレメントは、 *n* +1 のグループ化集合に変換される点に注意してください。*グループ化式* が指定されている順序が、 ROLLUP にとって重要である点も注意してください。たとえば、次のようにします。

```
GROUP BY ROLLUP(a,b)
```

これは、以下と同じ意味になります。

```
GROUP BY GROUPING SETS((a,b)
                       (a)
                       () )
```

一方、

```
GROUP BY ROLLUP(b,a)
```

これは、以下と同じです。

```
GROUP BY GROUPING SETS((b,a)
                       (b)
                       () )
```

ORDER BY 文節は、結果セットの行の ORDER BY を確保する唯一の方法です。

504 ページの例 C3 は ROLLUP の使用法を示しています。

CUBE (*grouping-expression-list*)

CUBE grouping (CUBE グループ化) は GROUP BY 文節の拡張であり、すべての ROLLUP 集約行に加えて、「クロス集計」行の入った結果セットを生成します。*クロス集計* 行は、小計による集約の一部ではない「スーパー集約行」です。

ROLLUP と同じように、CUBE グループ化も、一連の*グループ化集合* と見なすことができます。 CUBE の場合、*グループ化式リスト* のすべての順列が総計とともに計算されます。したがって、CUBE の *n* 個のエレメントは、 2***n* (2 の *n* 乗) の*グループ化集合* に変換されます。たとえば、次のように指定するとします。

```
GROUP BY CUBE(a,b,c)
```

これは、以下と同じ意味になります。

```
GROUP BY GROUPING SETS((a,b,c)
                       (a,b)
                       (a,c)
                       (b,c)
                       (a)
                       (b)
                       (c)
                       () )
```

CUBE の 3 個のエレメントは、8 個のグループ化集合に変換されることに注意してください。

エレメントの指定の順序は、CUBE の場合、重要ではありません。 CUBE (DayOfYear, Sales_Person)' と 'CUBE (Sales_Person, DayOfYear)' は同じ結果セットを生成します。「同じ」とは、結果セットの順序ではなく、結果セットの内容を指します。 ORDER BY 文節は、結果セットの行の ORDER BY を確保する唯一の方法です。
504 ページの例 C4 に、CUBE の使用例が示されています。

grouping-expression-list

grouping-expression-list (グループ化式リスト) は、 CUBE または ROLLUP 演算のエレメント数を定義するために、 CUBE または ROLLUP 文節で使用されます。これは、複数の*グループ化式* を持つエレメントを区切るために括弧を使用して制御されます。

グループ化式 の規則については、 488 ページの『GROUP BY 文節』 で説明しています。たとえば、照会が、County ではなく Province 内の City の ROLLUP について合計費用を戻すものと想定します。しかし、以下の文節の場合、

```
GROUP BY ROLLUP(Province, County, City)
```

County についての不要な小計行が生じます。 以下の文節では、

```
GROUP BY ROLLUP(Province, (County, City))
```

複合 (County, City) が ROLLUP の 1 つのエレメントを形成するので、この文節を使用する照会は、必要な結果を生じます。つまり、次のように 2 つのエレメントからなる ROLLUP の場合、

```
GROUP BY ROLLUP(Province, (County, City))
```

以下を生成します。

```
GROUP BY GROUPING SETS((Province, County, City)
                       (Province)
                       () )
```

一方、3 つのエレメントからなる ROLLUP の場合は、次のようになります。

```
GROUP BY GROUPING SETS((Province, County, City)
                       (Province, County)
                       (Province)
                       () )
```

503 ページの例 C2 でも、複合列値が使用されています。

grand-total

CUBE および ROLLUP は、全体の集約 (総計) である行を戻します。これは、GROUPING SET 文節に空の括弧を使用して別々に指定することができます。また、GROUP BY 文節に直接指定することもできます。これは照会の結果には影響しません。 504 ページの例 C4 では、総計構文を使用しています。

グループ化集合の結合

これは、任意のタイプの GROUP BY 文節を組み合わせるのに使用することができます。単純な*グループ化式* のフィールドを他のグループと組み合わせる場合、結果

のグループ化集合 の始めに「追加」されます。 ROLLUP 式または CUBE 式を組み合わせる場合、それらの式は、残りの式では「乗数」のように動作し、 ROLLUP または CUBE の定義に応じて追加のグループ化集合項目を生成します。

たとえば、グループ化式 のエレメントを組み合わせると、次のようになります。

```
GROUP BY a, ROLLUP(b,c)
```

これは、以下と同じ意味になります。

```
GROUP BY GROUPING SETS((a,b,c)
                       (a,b)
                       (a) )
```

もしくは

```
GROUP BY a, b, ROLLUP(c,d)
```

これは、以下と同じ意味になります。

```
GROUP BY GROUPING SETS((a,b,c,d)
                       (a,b,c)
                       (a,b) )
```

ROLLUP エレメントを組み合わせると、次のようになります。

```
GROUP BY ROLLUP(a), ROLLUP(b,c)
```

これは、以下と同じ意味になります。

```
GROUP BY GROUPING SETS((a,b,c)
                       (a,b)
                       (a)
                       (b,c)
                       (b)
                       () )
```

同様に、

```
GROUP BY ROLLUP(a), CUBE(b,c)
```

これは、以下と同じ意味になります。

```
GROUP BY GROUPING SETS((a,b,c)
                       (a,b)
                       (a,c)
                       (a)
                       (b,c)
                       (b)
                       (c)
                       () )
```

CUBE と *ROLLUP* のエレメントの組み合わせは次のようになります。

```
GROUP BY CUBE(a,b), ROLLUP(c,d)
```

これは、以下と同じ意味になります。

```
GROUP BY GROUPING SETS((a,b,c,d)
                       (a,b,c)
                       (a,b)
                       (a,c,d)
                       (a,c)
                       (a)
                       (b,c,d)
                       (b,c)
```

```
(b)
(c,d)
(c)
() )
```

単純な*グループ化式* の場合のように、グループ化集合を組み合わせると、各グループ化集合内で重複したものが除去されます。たとえば、次のようになります。

```
GROUP BY a, ROLLUP(a,b)
```

これは、以下と同じ意味になります。

```
GROUP BY GROUPING SETS((a,b)
                       (a) )
```

グループ化集合を組み合わせる場合の詳しい例は、完全な CUBE 集約について戻される特定行を除去する結果セットを構成する場合です。

たとえば、以下の GROUP BY 文節について考えてみます。

```
GROUP BY Region,
         ROLLUP(Sales_Person, WEEK(Sales_Date)),
         CUBE(YEAR(Sales_Date), MONTH (Sales_Date))
```

GROUP BY のすぐ右側にリストされている列は単純にグループ化され、 ROLLUP の後の括弧内の列がロールアップされ、 CUBE の後の括弧内の列は 3 乗されます。したがって、上記の文節の結果は、YEAR 内の MONTH のキューブが生成されてから、 REGION 内の Sales_Person 内の WEEK の集約内でロールアップが行われます。この結果は、Region、 Sales_Person または WEEK(Sales_Date) の総計行にもクロス集計行にもならないため、生成される行は、以下の文節より少なくなります。

```
GROUP BY ROLLUP (Region, Sales_Person, WEEK(Sales_Date),
                 YEAR(Sales_Date), MONTH(Sales_Date) )
```

HAVING 文節

HAVING 文節は、 *search-condition (検索条件)* が真である R のグループで構成される中間結果表を指定します。 R は、副選択のそれ以前の文節の結果です。その文節が GROUP BY ではない場合、 R はグループ化列のない単一のグループと見なされます。

検索条件内の各*列名* は、以下のいずれかを満たすものであることが必要です。

- R のグループ化列を明確に識別すること。
- 列関数内で指定されていること。
- 相関参照であること。副選択外の*表参照* の列を識別している*列名* は、相関参照となります。

検索条件が適用される R のグループは、検索条件内の各列関数 (引き数が相関参照である関数を除く) の引き数を提供するものとなります。

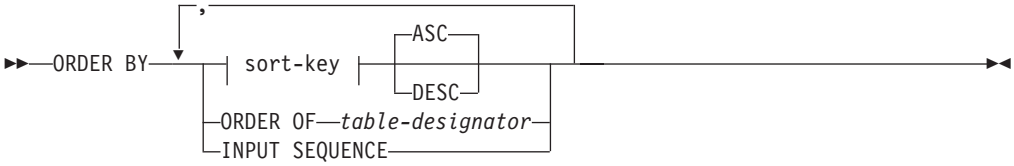
検索条件に副照会が入っている場合、その副照会は、検索条件が R のグループに適用されるたびに実行され、その結果は検索条件の適用において使用されるものと見なすことができます。実際には、副照会が各グループごとに実行されるのは、その

中に相関参照が入っている場合だけです。この違いについては、 499 ページの例 A6 および 499 ページの例 A7 を参照してください。

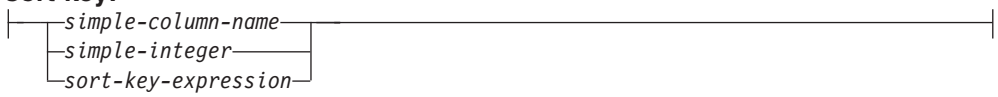
R のグループに対する相関参照は、グループ化列を指定するものであるか、あるいは列関数に入っているものでなければなりません。

HAVING が GROUP BY なしで使用される場合、選択リストには、列関数内の列名、相関列参照、リテラル、または特殊レジスターしか使用できません。

ORDER BY 文節



sort-key:



ORDER BY 文節は、結果表の行の ORDER BY を指定します。単一のソート指定 (方向が関連した 1 つの *sort-key* (ソート・キー)) が指定された場合、行は、そのソート指定の値によって順序付けられます。複数のソート指定を指定すると、行の順序は、最初に指定されたソート指定の値によってソートされ、次に 2 番目に指定されたソート指定の値によってソートされます。各 *sort-key* のデータ型は、LONG VARCHAR、CLOB、 LONG VARGRAPHIC、DBCLOB、BLOB、DATALINK、これらのタイプの特殊タイプ、または構造化タイプとすることはできません (SQLSTATE 42907)。

選択リストに指定された列は、 *simple-integer* または *simple-column-name* であるソート・キー によって識別することができます。選択リストに指定されていない列は、*simple-integer*、もしくは場合によっては、選択リストの式と一致する *sort-key-expression* (*sort-key-expression* の詳細を参照) によって識別されなければなりません。列は、AS 文節の指定がなく、しかもその列が定数、演算子の入った式、または関数から派生した列の場合には無名です。

順序付けは、比較規則に従って行われます。 NULL 値は、他のどのような値よりも高位として扱われます。 ORDER BY 文節で行が完全に順序付けされない場合、指定されたすべての列の値が重複する複数の行は、任意の ORDER BY で表示されます。

simple-column-name

通常、結果表の列を識別します。この場合、*simple-column-name (単純列名)* は、選択リストに指定された列の列名でなければなりません。

また、照会が副選択である場合、*simple-column-name* として、 FROM 文節で識別される表、ビュー、またはネストされた表の列名も指定することができます。副選択が以下の場合は、エラーが生じます。

- SELECT 文節に DISTINCT を指定する場合 (SQLSTATE 42822)

- グループ化された結果を生成する場合に、*単純列名* が*グループ化式* ではない場合 (SQLSTATE 42803)

結果の順序付けにどの列を使用するかの決定については、以下の『ソート・キーの列名』で説明されています。

simple-integer
0 より大きく、結果表の列の数以下でなければなりません (SQLSTATE 42805)。整数 *n* は、結果表の *n* 番目の列を指定します。

sort-key-expression
単なる列名または符号なし整数定数ではない式。順序付けが適用される照会は、この形式のソート・キーを使用するためには*副選択* でなければなりません。*sort-key-expression (*ソート・キー式*)* には、相関スカラー全選択 (SQLSTATE 42703)、または外部処理を伴う関数 (SQLSTATE 42845) を組み入れることはできません。

ソート・キー式 内の列名は、以下の『ソート・キーの列名』で説明されている規則に従っていなければなりません。

指定可能な式にさらに制約が加わる特殊な場合があります。

- DISTINCT が、副選択の SELECT 文節に指定されている (SQLSTATE 42822)。

  ソート・キー式は、副選択の選択リスト内の式と完全に一致しなければなりません (スカラー全選択は一致しません)。
- 副選択がグループ化されている (SQLSTATE 42803)。

  ソート・キー式は以下が可能です。
  - 副選択の選択リスト内の式である。
  - 副選択の GROUP BY 文節の*グループ化式* が組み込まれる。
  - 列関数、定数、またはホスト変数が組み込まれる。

ASC
列の値を昇順に使用します。これはデフォルトです。

DESC
列の値を降順に使用します。

ORDER OF *table-designator*
表指定子 で使用される同じ順序を、副選択の結果表に適用するように指定します。この文節を指定している副選択の FROM 文節の中に、*表指定子* に一致している表参照がなければなりません (SQLSTATE 42703)。指定された *表指定子* に対応する副選択 (または全選択) には、データに依存する ORDER BY 文節が入っていなければなりません (SQLSTATE 428FI)。適用される ORDER BY は、ネストされた副選択 (または全選択) 内の ORDER BY 文節の列が外部副選択 (または全選択) に入っていた場合、およびそれらの列が ORDER OF 文節の代わりに指定された場合と同じです。

この形式は、全選択 (全選択の変性形式を除く) では許可されていないので注意してください。たとえば、以下は無効です。

```
(SELECT C1 FROM T1
ORDER BY C1)
   UNION
 SELECT C1 FROM T2
ORDER BY ORDER OF T1
```

以下の例は有効です。

```
SELECT C1 FROM
   (SELECT C1 FROM T1
      UNION
    SELECT C1 FROM T2
    ORDER BY C1 ) AS UTABLE
ORDER BY ORDER OF UTABLE
```

INPUT SEQUENCE

INSERT ステートメントの場合に、結果行が配列済みデータ行の入力配列を反映していることを示します。 INPUT SEQUENCE 配列は、 FROM 文節で INSERT ステートメントが指定されている場合にのみ使用できます (SQLSTATE 428G4)。 478 ページの『表参照』を参照してください。 INPUT SEQUENCE が指定されていても、入力データが配列されていない場合は、 INPUT SEQUENCE 文節は無視されます。

注:

- **ソート・キーの列名**
  - 列名が修飾されている場合

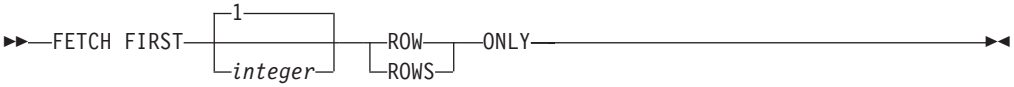
    照会は*副選択* でなければなりません (SQLSTATE 42877)。列名は、副選択の FROM 文節の表、ビュー、またはネストされた表の列を明確に識別する必要があります (SQLSTATE 42702)。列の値は、ソート指定の値を計算するのに使用されます。
  - 列名が無修飾の場合
    - 照会は副選択です。

      列名が結果表の複数の列の名前と同一である場合、この列名は、順序付け副選択の FROM 文節内の表、ビュー、またはネストされた表の列を明確に識別する必要があります (SQLSTATE 42702)。列名が 1 つの列と同一である場合、その列は、ソート指定の値を計算するのに使用されます。列名が結果表の列と同一でない場合、この列名は、選択ステートメントの全選択の FROM 文節の表、ビュー、またはネストされた表の列を明確に識別する必要があります (SQLSTATE 42702)。
    - 照会は、副選択ではありません (UNION、EXCEPT、または INTERSECT などの SET 演算が組み入れられます)。

      列名は、結果表の複数の列の名前と同一にすることはできません (SQLSTATE 42702)。列名は、結果表のちょうど 1 つの列と同一でなければなりません (SQLSTATE 42707)。この列は、ソート指定の値を計算するのに使用されます。
- **制限**: *ソート・キー式* またはその列が選択リストにない*単純列名* を使用すると、ソートに使用される一時表にその列または式が追加される場合があります。

これにより、表の列の数の制限、または表の行のサイズの制限に到達する場合があります。ソート操作を行うのに一時表が必要になる場合、このような制限を超えると、エラーが生じます。

FETCH FIRST 文節



FETCH FIRST 文節 は、検索できる最大行数を設定します。この文節は、アプリケーションが最大 *integer* 行数までしか検索しないことを、データベース・マネージャーに認識させます。これは、この文節が指定されていない場合に、結果表にある行数に影響されません。 *integer* 行より多くの行を取り出そうとすると、通常のデータの終わりと同じように処理されます (SQLSTATE 02000)。 *integer* の値は、正の整数 (ゼロを除く) でなければなりません。

結果表を最初の *integer* 行に限定することにより、パフォーマンスが向上します。データベース・マネージャーは、一度最初の *integer* 行を判別すると、照会処理を停止します。 *fetch-first-clause* と *optimize-for-clause* の両方が指定されている場合には、これらの文節の *integer* 値のうちの小さい方を使用して通信バッファー・サイズが決定されます。これらの値は最適化処理専用です。

全選択に FROM 文節の SQL データ変更ステートメントが入っている場合は、フェッチされる行の数の限度に関係なく、すべての行が変更されます。

副選択の例

例 A1: EMPLOYEE 表の列および行を全選択します。

```
SELECT * FROM EMPLOYEE
```

例 A2: EMP_ACT 表および EMPLOYEE 表を結合し、 EMP_ACT 表からすべての列を選択して、 EMPLOYEE 表の従業員の姓 (LASTNAME) を結果の各行に追加します。

```
SELECT EMP_ACT.*, LASTNAME
  FROM EMP_ACT, EMPLOYEE
  WHERE EMP_ACT.EMPNO = EMPLOYEE.EMPNO
```

例 A3: EMPLOYEE 表と DEPARTMENT 表を結合し、 1930 年よりも前に生まれた (BIRTHDATE) 従業員すべての従業員番号 (EMPNO)、従業員の姓 (LASTNAME)、部門番号 (EMPLOYEE 表の WORKDEPT と DEPARTMENT 表の DEPTNO)、および部門名 (DEPTNAME) を選択します。

```
SELECT EMPNO, LASTNAME, WORKDEPT, DEPTNAME
   FROM EMPLOYEE, DEPARTMENT
   WHERE WORKDEPT = DEPTNO
   AND YEAR(BIRTHDATE) < 1930
```

例 A4: EMPLOYEE 表の中で同じジョブ・コードをもつ行のグループごとに、ジョブ (JOB) と給与 (SALARY) の最低額と最高額を選択します。ただし、グループの中でも、複数の行を備えていて、しかも給与の最高額が 27000 以上のグループについてのみ選択を行います。

```
SELECT JOB, MIN(SALARY), MAX(SALARY)
                FROM EMPLOYEE
  GROUP BY JOB
  HAVING COUNT(*) > 1
  AND MAX(SALARY) >= 27000
```

例 A5: EMP_ACT 表の中から、部門 (WORKDEPT) 'E11' の従業員 (EMPNO) についてのすべての行を選択します。 (従業員部門番号は、EMPLOYEE 表に示されています。)

```
SELECT *
   FROM EMP_ACT
  WHERE EMPNO IN
          (SELECT EMPNO
                FROM EMPLOYEE
               WHERE WORKDEPT = 'E11')
```

例 A6: EMPLOYEE 表から、給与の最高額が従業員全体の平均給与に満たないすべての部門について、部門番号 (WORKDEPT) と部門別給与 (SALARY) の最高額を選択します。

```
SELECT WORKDEPT, MAX(SALARY)
                FROM EMPLOYEE
  GROUP BY WORKDEPT
  HAVING MAX(SALARY) < (SELECT AVG(SALARY)
                        FROM EMPLOYEE)
```

この例では、HAVING 文節の副照会は一度だけ実行されることになります。

例 A7: EMPLOYEE 表を使用して、部門別給与の最高額が他のすべての部門の平均給与より少ない部門の部門番号 (WORKDEPT) とその部門別給与 (SALARY) の最高額を選択します。

```
SELECT WORKDEPT, MAX(SALARY)
   FROM EMPLOYEE EMP_COR
    GROUP BY WORKDEPT
    HAVING MAX(SALARY) < (SELECT AVG(SALARY)
                FROM EMPLOYEE
                          WHERE NOT WORKDEPT = EMP_COR.WORKDEPT)
```

例 A6 とは反対に、HAVING 文節の副照会は、各グループごとに実行する必要があります。

例 A8: セールス担当員の従業員番号と給与、およびその部門の給与平均額と人数とを調べます。

この照会では、まずネストされた表式 (DINFO) を作成して、AVGSALARY 列と EMPCOUNT 列、また WHERE 文節で使用される DEPTNO 列を入手する必要があります。

```
SELECT THIS_EMP.EMPNO, THIS_EMP.SALARY, DINFO.AVGSALARY, DINFO.EMPCOUNT
 FROM EMPLOYEE THIS_EMP,
      (SELECT OTHERS.WORKDEPT AS DEPTNO,
                        AVG(OTHERS.SALARY) AS AVGSALARY,
                        COUNT(*) AS EMPCOUNT
        FROM EMPLOYEE OTHERS
        GROUP BY OTHERS.WORKDEPT
       ) AS DINFO
 WHERE THIS_EMP.JOB = 'SALESREP'
  AND THIS_EMP.WORKDEPT = DINFO.DEPTNO
```

このような場合には、ネストした表式を使用することによって、 DINFO ビューを正規のビューとして作成することによりオーバーヘッドを軽減することができます。ステートメントの準備中に、ビューのカタログにはアクセスされません。これは、照会の残りの部分のコンテキストにより、ビューによって考慮する必要があるのはセールス担当の部門の行だけだからです。

例 A9: 5 つの従業員のランダム・グループについて平均的な教育レベルと給与を表示します。

この照会では、各従業員のランダム値を GROUP BY 文節で使用できるようにするために、ネストした表式を使用してこのランダム値を設定する必要があります。

```
SELECT RANDID , AVG(EDLEVEL), AVG(SALARY)
  FROM ( SELECT EDLEVEL, SALARY, INTEGER(RAND()*5) AS RANDID
                FROM EMPLOYEE
       ) AS EMPRAND
  GROUP BY RANDID
```

例 A10: EMP_ACT 表を照会して、その中の従業員の給与が全従業員の中で上位 10 人に入るプロジェクト番号を戻します。

```
SELECT EMP_ACT.EMPNO,PROJNO
   FROM EMP_ACT
   WHERE EMP_ACT.EMPNO IN
       (SELECT EMPLOYEE.EMPNO
                FROM EMPLOYEE
        ORDER BY SALARY DESC
        FETCH FIRST 10 ROWS ONLY)
```

結合の例

例 B1: この例では、表 J1 および J2 を使用した種々の結合の結果を示しています。これらの表には、以下の行が入っています。

```
SELECT * FROM J1

W   X
--- ------
A       11
B       12
C       13

SELECT * FROM J2

Y   Z
--- ------
A       21
C       22
D       23
```

以下の照会では、両方の表の最初の列が一致する J1 および J2 の内部結合を行っています。

```
SELECT * FROM J1 INNER JOIN J2 ON W=Y

W   X      Y   Z
--- ------ --- ------
A       11 A       21
C       13 C       22
```

この内部結合の例では、J1 の列 W='C' の行および J2 の列 Y='D' の行が、結果に入っていません。これは、これらの行がもう一方の表に一致するものがないからです。次のような代替形式の内部結合照会で同じ結果が生成されることに注意してください。

```
SELECT * FROM J1, J2 WHERE W=Y
```

以下の左外部結合では、J2 の列が NULL 値である J1 の欠落行を戻します。 J1 のすべての行が組み入れられます。

```
SELECT * FROM J1 LEFT OUTER JOIN J2 ON W=Y

W   X      Y   Z
--- ------ --- ------
A       11 A       21
B       12 -        -
C       13 C       22
```

以下の右外部結合では、J1 の列が NULL 値である J2 の欠落行を戻します。 J2 のすべての行が組み入れられます。

```
SELECT * FROM J1 RIGHT OUTER JOIN J2 ON W=Y

W   X      Y   Z
--- ------ --- ------
A       11 A       21
C       13 C       22
-        - D       23
```

以下の全外部結合では、NULL 値である J1 と J2 の両方の欠落行を戻します。 J1 と J2 の両方のすべての行が組み入れられます。

```
SELECT * FROM J1 FULL OUTER JOIN J2 ON W=Y

W   X      Y   Z
--- ------ --- ------
A       11 A       21
C       13 C       22
-        - D       23
B       12 -        -
```

例 B2: 上記の例の表 J1 および J2 を使用して、述部が検索条件に追加されるとどうなるかを調べます。

```
SELECT * FROM J1 INNER JOIN J2 ON W=Y AND X=13

W   X      Y   Z
--- ------ --- ------
C       13 C       22
```

条件を追加すると、内部結合は、 500 ページの例 B1 の内部結合と比較して 1 行のみを選択します。

全外部結合に対するこの影響に注意してください。

```
SELECT * FROM J1 FULL OUTER JOIN J2 ON W=Y AND X=13

W   X      Y   Z
--- ------ --- ------
-        - A       21
C       13 C       22
-        - D       23
A       11 -        -
B       12 -        -
```

内部結合には 1 行のみがあり、両方の表のすべての行を戻す必要があるため、結果は 5 行になります (追加の述部がない場合の 4 行と比較して)。

以下の照会では、同じ述部を WHERE 文節に追加することにより、まったく異なる結果を生成される場合を示しています。

```
SELECT * FROM J1 FULL OUTER JOIN J2 ON W=Y
   WHERE X=13

W   X      Y   Z
--- ------ --- ------
C       13 C       22
```

WHERE 文節は、全外部結合の中間結果の後に適用されます。この中間結果は、500 ページの例 B1 の全外部結合照会の結果と同じになります。 WHERE 文節は、この中間結果に適用され、X=13 の行を除くすべての行を除去します。外部結合を行う場合に、述部の位置の選択によって、結果に大きな影響を与える可能性があります。述部が X=13 ではなく X=12 であるとどうなるかを考えてみます。以下の内部結合は行を戻しません。

```
SELECT * FROM J1 INNER JOIN J2 ON W=Y AND X=12
```

したがって、全外部結合は 6 行を返します。つまり、J2 の列が NULL 値である J1 の 3 行と、J1 の列が NULL 値である J2 の 3 行です。

```
SELECT * FROM J1 FULL OUTER JOIN J2 ON W=Y AND X=12

W   X      Y   Z
--- ------ --- ------
-        - A       21
-        - C       22
-        - D       23
A       11 -        -
B       12 -        -
C       13 -        -
```

追加の述部が WHERE 文節にある場合には、1 行が戻されます。

```
SELECT * FROM J1 FULL OUTER JOIN J2 ON W=Y
   WHERE X=12

W   X      Y   Z
--- ------ --- ------
B       12 -        -
```

例 B3: 管理者のいない部門も含めて、すべての部門を従業員番号と管理者の姓と共にリストします。

```
SELECT DEPTNO, DEPTNAME, EMPNO, LASTNAME
  FROM DEPARTMENT LEFT OUTER JOIN EMPLOYEE
       ON MGRNO = EMPNO
```

例 B4: 管理者のいない従業員も含めて、すべての従業員の番号と姓を管理者の従業員番号と姓と共にリストします。

```
SELECT E.EMPNO, E.LASTNAME, M.EMPNO, M.LASTNAME
  FROM EMPLOYEE E LEFT OUTER JOIN
                                      DEPARTMENT INNER JOIN EMPLOYEE M
       ON MGRNO = M.EMPNO
       ON E.WORKDEPT = DEPTNO
```

内部結合は、DEPARTMENT 表で識別されるすべての管理者の姓を判別し、左外部結合によって、対応する部門が DEPARTMENT にない場合であっても各従業員を必ずリストすることができます。

グループ化集合、CUBE、および ROLLUP の例

例 C1 から 504 ページの例 C4 の照会では、述部 'WEEK(SALES_DATE) = 13' に基づいて SALES 表の行のサブセットを使用しています。

```
SELECT WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       SALES_PERSON, SALES AS UNITS_SOLD
FROM SALES
WHERE WEEK(SALES_DATE) = 13
```

これは次の結果になります。

```
WEEK        DAY_WEEK    SALES_PERSON    UNITS_SOLD
----------- ----------- --------------- -----------
         13           6 LUCCHESSI                 3
         13           6 LUCCHESSI                 1
         13           6 LEE                       2
         13           6 LEE                       2
         13           6 LEE                       3
         13           6 LEE                       5
         13           6 GOUNOT                    3
         13           6 GOUNOT                    1
         13           6 GOUNOT                    7
         13           7 LUCCHESSI                 1
         13           7 LUCCHESSI                 2
         13           7 LUCCHESSI                 1
         13           7 LEE                       7
         13           7 LEE                       3
         13           7 LEE                       7
         13           7 LEE                       4
         13           7 GOUNOT                    2
         13           7 GOUNOT                   18
         13           7 GOUNOT                    1
```

例 C1: これは、3 つの列に対して基本の GROUP BY 文節を使用している照会です。

```
SELECT WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
      SALES_PERSON, SUM(SALES) AS UNITS_SOLD
FROM SALES
WHERE WEEK(SALES_DATE) = 13
GROUP BY WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE), SALES_PERSON
ORDER BY WEEK, DAY_WEEK, SALES_PERSON
```

これは次のような結果になります。

```
WEEK        DAY_WEEK    SALES_PERSON    UNITS_SOLD
----------- ----------- --------------- -----------
         13           6 GOUNOT                   11
         13           6 LEE                      12
         13           6 LUCCHESSI                 4
         13           7 GOUNOT                   21
         13           7 LEE                      21
         13           7 LUCCHESSI                 4
```

例 C2: SALES 表の行の 2 つのグループ化集合に基づいて結果を生成します。

```
SELECT WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
      SALES_PERSON, SUM(SALES) AS UNITS_SOLD
FROM SALES
WHERE WEEK(SALES_DATE) = 13
GROUP BY GROUPING SETS ( (WEEK(SALES_DATE), SALES_PERSON),
                         (DAYOFWEEK(SALES_DATE), SALES_PERSON))
ORDER BY WEEK, DAY_WEEK, SALES_PERSON
```

これは次のような結果になります。

```
WEEK        DAY_WEEK    SALES_PERSON    UNITS_SOLD
----------- ----------- --------------- -----------
         13           - GOUNOT                   32
         13           - LEE                      33
         13           - LUCCHESSI                 8
          -           6 GOUNOT                   11
          -           6 LEE                      12
          -           6 LUCCHESSI                 4
          -           7 GOUNOT                   21
          -           7 LEE                      21
          -           7 LUCCHESSI                 4
```

WEEK 13 の行は、最初のグループ化集合のものであり、それ以外の行は 2 番目のグループ化集合のものです。

例 C3: 503 ページの例 C2 のグループ化集合に関与した 3 つの特殊な列を使用して、 ROLLUP を実行する場合、(WEEK、DAY_WEEK、SALES_PERSON)、(WEEK、DAY_WEEK)、 (WEEK) および総計のグループ化集合を見ることができます。

```
SELECT WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
      SALES_PERSON, SUM(SALES) AS UNITS_SOLD
FROM SALES
WHERE WEEK(SALES_DATE) = 13
GROUP BY ROLLUP ( WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE), SALES_PERSON )
ORDER BY WEEK, DAY_WEEK, SALES_PERSON
```

これは次のような結果になります。

```
WEEK        DAY_WEEK    SALES_PERSON    UNITS_SOLD
----------- ----------- --------------- -----------
         13           6 GOUNOT                   11
         13           6 LEE                      12
         13           6 LUCCHESSI                 4
         13           6 -                        27
         13           7 GOUNOT                   21
         13           7 LEE                      21
         13           7 LUCCHESSI                 4
         13           7 -                        46
         13           - -                        73
          -           - -                        73
```

例 C4: 例 C3 と同じ照会を実行して ROLLUP を CUBE に置き換える場合、結果に (WEEK、SALES_PERSON)、 (DAY_WEEK、SALES_PERSON)、(DAY_WEEK)、(SALES_PERSON) の追加のグループ化集合を見ることができます。

```
SELECT WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
      SALES_PERSON, SUM(SALES) AS UNITS_SOLD
```

```
FROM SALES
WHERE WEEK(SALES_DATE) = 13
GROUP BY CUBE ( WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE), SALES_PERSON )
ORDER BY WEEK, DAY_WEEK, SALES_PERSON
```

これは次のような結果になります。

```
WEEK        DAY_WEEK    SALES_PERSON    UNITS_SOLD
----------- ----------- --------------- -----------
         13           6 GOUNOT                   11
         13           6 LEE                      12
         13           6 LUCCHESSI                 4
         13           6 -                        27
         13           7 GOUNOT                   21
         13           7 LEE                      21
         13           7 LUCCHESSI                 4
         13           7 -                        46
         13           - GOUNOT                   32
         13           - LEE                      33
         13           - LUCCHESSI                 8
         13           - -                        73
          -           6 GOUNOT                   11
          -           6 LEE                      12
          -           6 LUCCHESSI                 4
          -           6 -                        27
          -           7 GOUNOT                   21
          -           7 LEE                      21
          -           7 LUCCHESSI                 4
          -           7 -                        46
          -           - GOUNOT                   32
          -           - LEE                      33
          -           - LUCCHESSI                 8
          -           - -                        73
```

例 C5: SALES 表から選択された行の総計と、 SALES_PERSON および MONTH によって集計された行のグループの入った結果セットを入手します。

```
SELECT SALES_PERSON,
       MONTH(SALES_DATE) AS MONTH,
       SUM(SALES) AS UNITS_SOLD
FROM SALES
GROUP BY GROUPING SETS ( (SALES_PERSON, MONTH(SALES_DATE)),
                         ()
                       )
ORDER BY SALES_PERSON, MONTH
```

これは次のような結果になります。

```
SALES_PERSON    MONTH       UNITS_SOLD
--------------- ----------- -----------
GOUNOT                    3          35
GOUNOT                    4          14
GOUNOT                   12           1
LEE                       3          60
LEE                       4          25
LEE                      12           6
LUCCHESSI                 3           9
LUCCHESSI                 4           4
LUCCHESSI                12           1
          - -                       155
```

例 C6: この例では、2 つの単純な ROLLUP 照会を示し、その後に、2 つの ROLLUP を単一結果セットのグループ化集合として扱い、グループ化集合に入っている列ごとに行の順序を指定する照会を示しています。

例 C6-1:

```
SELECT WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       SUM(SALES) AS UNITS_SOLD
FROM SALES
GROUP BY ROLLUP ( WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE) )
ORDER BY WEEK, DAY_WEEK
```

結果は次のようになります。

```
WEEK        DAY_WEEK    UNITS_SOLD
----------- ----------- -----------
         13           6          27
         13           7          46
         13           -          73
         14           1          31
         14           2          43
         14           -          74
         53           1           8
         53           -           8
          - -                     155
```

例 C6-2:

```
SELECT MONTH(SALES_DATE) AS MONTH,
       REGION,
       SUM(SALES) AS UNITS_SOLD
FROM SALES
GROUP BY ROLLUP ( MONTH(SALES_DATE), REGION );
ORDER BY MONTH, REGION
```

結果は次のようになります。

```
MONTH       REGION          UNITS_SOLD
----------- --------------- -----------
          3 Manitoba                 22
          3 Ontario-North             8
          3 Ontario-South            34
          3 Quebec                   40
          3 -                       104
          4 Manitoba                 17
          4 Ontario-North             1
          4 Ontario-South            14
          4 Quebec                   11
          4 -                        43
         12 Manitoba                  2
         12 Ontario-South             4
         12 Quebec                    2
         12 -                         8
          - -                       155
```

例 C6-3:

```
SELECT WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       MONTH(SALES_DATE) AS MONTH,
       REGION,
       SUM(SALES) AS UNITS_SOLD
FROM SALES
GROUP BY GROUPING SETS ( ROLLUP( WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE) ),
                         ROLLUP( MONTH(SALES_DATE), REGION )  )
ORDER BY WEEK, DAY_WEEK, MONTH, REGION
```

結果は次のようになります。

```
   WEEK        DAY_WEEK    MONTH       REGION          UNITS_SOLD
----------- ----------- ----------- --------------- -----------
         13           6           - -                          27
         13           7           - -                          46
         13           -           - -                          73
         14           1           - -                          31
         14           2           - -                          43
         14           -           - -                          74
         53           1           - -                           8
         53           -           - -                           8
          -           -           3 Manitoba                   22
          -           -           3 Ontario-North               8
          -           -           3 Ontario-South              34
          -           -           3 Quebec                     40
          -           -           3 -                         104
          -           -           4 Manitoba                   17
          -           -           4 Ontario-North               1
          -           -           4 Ontario-South              14
          -           -           4 Quebec                     11
          -           -           4 -                          43
          -           -          12 Manitoba                    2
          -           -          12 Ontario-South               4
          -           -          12 Quebec                      2
          -           -          12 -                           8
          -           -           - -                         155
          -           -           - -                         155
```

2 つの ROLLUP をグループ化集合として使用すると、結果に重複した行が組み入れられます。総計行も 2 つになります。

ORDER BY を使用すると結果にどのような影響があるかを調べてみます。

- 最初のグループ化集合では、week 53 が最後に位置変更されています。
- 2 番目のグループ化集合では、month 12 が最後に位置付けられ、地域がアルファベット順になっています。
- NULL 値は上位にソートされます。

例 C7: 単一のパスで複数の ROLLUP を実行する照会 (たとえば 506 ページの例 C6-3) では、各行を生成したのがどのグループ化集合であるかを示すことができます。以下のステップは、結果セット内の各行の発生点を示す列 (GROUP と呼ばれます) を提供する方法を示しています。結果セットの行を生成したのが、2 つのグループ化集合のいずれであるかということです。

ステップ 1: VALUES 文節から選択する照会 (代替形式の全選択) を使用して、新しいデータ値を「生成する」方法を導入します。この照会は、2 つの列 ″R1″ と ″R2″、および 1 行のデータがある、 ″X″ と呼ばれる表を得る方法を示しています。

```
SELECT R1,R2
FROM (VALUES('GROUP 1','GROUP 2')) AS X(R1,R2);
```

結果は次のようになります。

```
  R1      R2
------- -------
GROUP 1 GROUP 2
```

ステップ 2: SALES 表を使ってこの表 ″X″ のクロス製品を生成します。これにより、各行に列 ″R1″ および ″R2″ が追加されます。

```
SELECT R1, R2, WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       MONTH(SALES_DATE) AS MONTH,
       REGION,
       SALES AS UNITS_SOLD
FROM SALES,(VALUES('GROUP 1','GROUP 2')) AS X(R1,R2)
```

これにより、各行に列 ″R1″ および ″R2″ が追加されます。

ステップ 3: これで、これらの列をグループ化集合と組み合わせて、 ROLLUP 分析にこれらの列を組み入れることができるようになりました。

```
SELECT R1, R2,
       WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       MONTH(SALES_DATE) AS MONTH,
       REGION, SUM(SALES) AS UNITS_SOLD
FROM SALES,(VALUES('GROUP 1','GROUP 2')) AS X(R1,R2)
GROUP BY GROUPING SETS ((R1, ROLLUP(WEEK(SALES_DATE),
                         DAYOFWEEK(SALES_DATE))),
                       (R2,ROLLUP( MONTH(SALES_DATE), REGION ) )  )
ORDER BY WEEK, DAY_WEEK, MONTH, REGION
```

結果は次のようになります。

```
R1      R2      WEEK     DAY_WEEK  MONTH     REGION          UNITS_SOLD
------- ------- -------- --------- --------- --------------- -----------
GROUP 1 -             13         6         - -                          27
GROUP 1 -             13         7         - -                          46
GROUP 1 -             13         -         - -                          73
GROUP 1 -             14         1         - -                          31
GROUP 1 -             14         2         - -                          43
GROUP 1 -             14         -         - -                          74
GROUP 1 -             53         1         - -                           8
GROUP 1 -             53         -         - -                           8
-       GROUP 2        -         -         3 Manitoba                   22
-       GROUP 2        -         -         3 Ontario-North               8
-       GROUP 2        -         -         3 Ontario-South              34
-       GROUP 2        -         -         3 Quebec                     40
-       GROUP 2        -         -         3 -                         104
-       GROUP 2        -         -         4 Manitoba                   17
-       GROUP 2        -         -         4 Ontario-North               1
-       GROUP 2        -         -         4 Ontario-South              14
-       GROUP 2        -         -         4 Quebec                     11
-       GROUP 2        -         -         4 -                          43
-       GROUP 2        -         -        12 Manitoba                    2
-       GROUP 2        -         -        12 Ontario-South               4
-       GROUP 2        -         -        12 Quebec                      2
-       GROUP 2        -         -        12 -                           8
-       GROUP 2        -         -         - -                         155
GROUP 1 -              -         -         - -                         155
```

ステップ 4: R1 および R2 が異なるグループ化集合で使用されるため、 R1 の結果が非 NULL である場合は常に R2 は NULL であり、 R2 の結果が非 NULL である場合は常に R1 は NULL になることに注意してください。つまり、 COALESCE 関数を使用すれば、これらの列を単一列に統合できるということです。また、ORDER BY 文節でこの列を使用すれば、 2 つのグループ化集合の結果をまとめることもできます。

```
SELECT COALESCE(R1,R2) AS GROUP,
       WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       MONTH(SALES_DATE) AS MONTH,
       REGION, SUM(SALES) AS UNITS_SOLD
```

```
FROM SALES,(VALUES('GROUP 1','GROUP 2')) AS X(R1,R2)
GROUP BY GROUPING SETS ((R1, ROLLUP(WEEK(SALES_DATE),
                        DAYOFWEEK(SALES_DATE))),
                       (R2,ROLLUP( MONTH(SALES_DATE), REGION ) )  )
ORDER BY GROUP, WEEK, DAY_WEEK, MONTH, REGION;
```

結果は次のようになります。

```
GROUP   WEEK        DAY_WEEK    MONTH       REGION          UNITS_SOLD
------- ----------- ----------- ----------- --------------- ----------
GROUP 1          13           6           - -                         27
GROUP 1          13           7           - -                         46
GROUP 1          13           -           - -                         73
GROUP 1          14           1           - -                         31
GROUP 1          14           2           - -                         43
GROUP 1          14           -           - -                         74
GROUP 1          53           1           - -                          8
GROUP 1          53           -           - -                          8
GROUP 1           -           -           - -                        155
GROUP 2           -           -           3 Manitoba                  22
GROUP 2           -           -           3 Ontario-North              8
GROUP 2           -           -           3 Ontario-South             34
GROUP 2           -           -           3 Quebec                    40
GROUP 2           -           -           3 -                        104
GROUP 2           -           -           4 Manitoba                  17
GROUP 2           -           -           4 Ontario-North              1
GROUP 2           -           -           4 Ontario-South             14
GROUP 2           -           -           4 Quebec                    11
GROUP 2           -           -           4 -                         43
GROUP 2           -           -          12 Manitoba                   2
GROUP 2           -           -          12 Ontario-South              4
GROUP 2           -           -          12 Quebec                     2
GROUP 2           -           -          12 -                          8
GROUP 2           -           -           - -                        155
```

例 C8: 以下の例は、CUBE を実行する場合の種々の列関数の使用例を示しています。また、この例は、cast 関数および round 関数を利用して、妥当な精度と位取りで 10 進数の結果を生成します。

```
SELECT MONTH(SALES_DATE) AS MONTH,
       REGION,
       SUM(SALES) AS UNITS_SOLD,
       MAX(SALES) AS BEST_SALE,
       CAST(ROUND(AVG(DECIMAL(SALES)),2) AS DECIMAL(5,2)) AS AVG_UNITS_SOLD
FROM SALES
GROUP BY CUBE(MONTH(SALES_DATE),REGION)
ORDER BY MONTH, REGION
```

これは次のような結果になります。

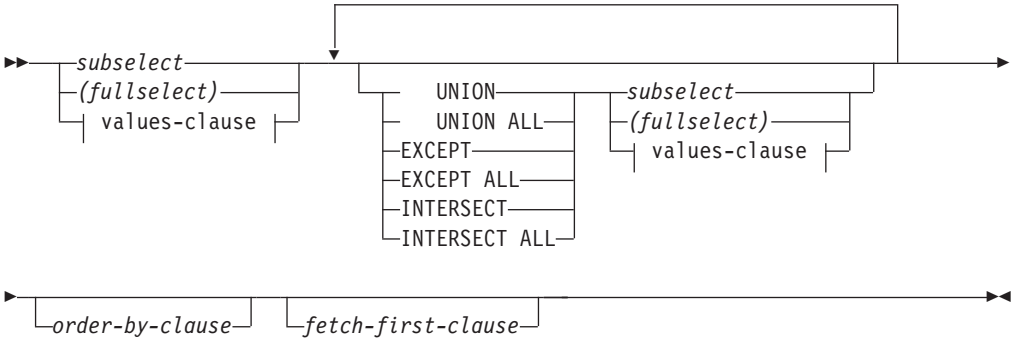
```
MONTH       REGION          UNITS_SOLD  BEST_SALE   AVG_UNITS_SOLD
----------- --------------- ----------- ----------- --------------
          3 Manitoba                 22           7           3.14
          3 Ontario-North             8           3           2.67
          3 Ontario-South            34          14           4.25
          3 Quebec                   40          18           5.00
          3 -                       104          18           4.00
          4 Manitoba                 17           9           5.67
          4 Ontario-North             1           1           1.00
          4 Ontario-South            14           8           4.67
          4 Quebec                   11           8           5.50
          4 -                        43           9           4.78
         12 Manitoba                  2           2           2.00
         12 Ontario-South             4           3           2.00
         12 Quebec                    2           1           1.00
         12 -                         8           3           1.60
```

```
- Manitoba                41          9          3.73
- Ontario-North            9          3          2.25
- Ontario-South           52         14          4.00
- Quebec                  53         18          4.42
- -                      155         18          3.87
```

関連資料:

- 64 ページの『ID』
- 166 ページの『関数』
- 282 ページの『GROUPING』
- 511 ページの『全選択』
- 516 ページの『Select-statement』
- 「*SQL リファレンス 第 2 巻*」の『DELETE ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『INSERT ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『UPDATE ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『CREATE FUNCTION (SQL スカラー、表、または行) ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『CREATE FUNCTION (外部表) ステートメント』
- 90 ページの『文字ストリング』
- 110 ページの『割り当てと比較』
- 222 ページの『述部』

全選択



values-clause:

Values-row:

全選択 (fullselect) は、select ステートメント、 INSERT ステートメント、および CREATE VIEW ステートメントのコンポーネントの 1 つです。また、これはステートメントのコンポーネントである、特定の述部のコンポーネントともなります。述部のコンポーネントである全選択は、*副照会 (subquery)* と呼ばれ、括弧で囲んだ全選択 (fullselect) も、副照会と呼ばれることがあります。

セット演算子である UNION、EXCEPT、および INTERSECT は、関係演算子の合併、差、積に対応しています。

全選択は結果表を指定します。セット演算子を使用しない全選択の結果は、指定した副選択または VALUES 文節の結果になります。

Values-clause

結果表の行の各列ごとに式を使用して実際の値を指定することによって、結果表を派生させます。複数の行を指定することができます。

NULL は、複数の *Values-row* でのみ使用することができ、同一列の少なくとも 1 行は NULL 以外でなければなりません (SQLSTATE 42826)。

Values-row は以下によって指定されます。

- 結果表の単一の列についての単一の式
- コンマで区切った *n* 個の式 (または NULL) を括弧で囲んだもの (*n* は結果表の列の数)

複数行からなる Values-clause には、各 *Values-row* に同数の式が必要です (SQLSTATE 42826)。

以下に、Values-clause の例とその意味を示します。

```
VALUES (1),(2),(3)          - 3 rows of 1 column
VALUES 1, 2, 3              - 3 rows of 1 column
VALUES (1, 2, 3)            - 1 row of 3 columns
VALUES (1,21),(2,22),(3,23) - 3 rows of 2 columns
```

Values-clause は、 *n* 個の指定の *Values-row* RE_1 から RE_n (*n* は 2 以上) で構成され、以下と同等です。

RE_1 UNION ALL RE_2 ... UNION ALL RE_n

これは、各 *Values-row* に対応する式は比較可能でなければなりません (SQLSTATE 42825)。

UNION または **UNION ALL**
2 つの結果表 (R1 と R2) を組み合わせて、新たな結果表を導きます。 UNION ALL を指定すると、結果は R1 と R2 のすべての行から構成されるものになります。 ALL オプションなしで UNION を指定すると、結果は R1 または R2 のいずれかの行すべての集合から、重複行を除去したものになります。しかしいずれにしても、UNION 表の各行は R1 か R2 のどちらかから取られた行です。

EXCEPT または **EXCEPT ALL**
2 つの結果表 (R1 と R2) を組み合わせて、新たな結果表を導きます。 EXCEPT ALL を指定すると、結果は、重複行の数を勘定に入れつつ、 R2 の中に対応する行のないすべての行で構成されるものになります。 ALL オプションなしで EXCEPT を指定すると、結果は、それぞれの重複行を除去してから R1 にのみ存在する行を取り出したもので構成されます。

INTERSECT または **INTERSECT ALL**
2 つの結果表 (R1 と R2) を組み合わせて、新たな結果表を導きます。 INTERSECT ALL を指定すると、結果は R1 と R2 の両方に入っている行すべてで構成されるものになります。 ALL オプションなしで INTERSECT を指定すると、結果は、R1 と R2 の両方にある行すべての集合から重複行を除去したものになります。

ORDER BY 文節
ORDER BY または FETCH FIRST 文節の入った全選択は、以下では指定できません。

- マテリアライズ照会表
- ビューの最外部の全選択 (SQLSTATE 428FJ)。

注: 全選択内の ORDER BY 文節は、照会によって戻される行の順序には影響を与えません。 ORDER BY 文節が影響を与えるのは、最外部の全選択で指定された場合に戻される行の ORDER BY だけです。

結果表 R1 の中の列の数と R2 の中の列の数は、同じでなければなりません (SQLSTATE 42826)。 ALL キーワードを指定していない場合、 LONG VARCHAR、CLOB、LONG VARGRAPHIC、DBCLOB、 BLOB、DATALINK の各

データ型、これらの型の特殊型、または構造化タイプをもつ列を R1 および R2 に組み入れることはできません (SQLSTATE 42907)。

結果の列の名前は、次のようになります。

- R1 の *n* 番目の列と R2 の *n* 番目の列の結果列の名前が同じ場合、 R の *n* 番目の列が結果列の名前になります。
- R1 の *n* 番目の列と R2 の *n* 番目の列の結果列の名前が異なる場合は、名前が生成されます。この名前を、ORDER BY 文節または UPDATE 文節の列名として使用することはできません。

生成された名前を調べるには、SQL ステートメントの DESCRIBE を実行して、SQLNAME フィールドを参照します。

2 つの行が互いに重複していると言えるのは、最初の行の各値が 2 番目の行の対応する値に等しい場合です。 (重複を判別する場合、2 つの NULL 値は等しいものと見なされます。)

複数の演算を 1 つの式の中に結合した場合は、括弧内の演算が先に実行されます。括弧がない場合、演算は左から右に実行されますが、例外として、すべての INTERSECT 演算は UNION または EXCEPT の演算の前に実行されます。

次の例では、表 R1 と R2 の値を左端に示しています。他にリストされている見出しは、R1 と R2 の種々のセット演算の結果の値を示しています。

R1	R2	UNION ALL	UNION	EXCEPT ALL	EXCEPT	INTER-SECT ALL	INTER-SECT
1	1	1	1	1	2	1	1
1	1	1	2	2	5	1	3
1	3	1	3	2		3	4
2	3	1	4	2		4	
2	3	1	5	4			
2	3	2		5			
3	4	2					
4		2					
4		3					
5		3					
		3					
		3					
		3					
		4					
		4					
		4					
		5					

全選択の例

例 1: EMPLOYEE 表からすべての列と行を選択します。

```
SELECT * FROM EMPLOYEE
```

例 2: EMPLOYEE 表の従業員で、その部門番号 (WORKDEPT) が 'E' で始まる部門に属しているか、**または**プロジェクト番号 (PROJNO) が 'MA2100'、 'MA2110'、または 'MA2112' である EMP_ACT 表のプロジェクトに割り当てられている従業員すべての従業員番号 (EMPNO) をリストします。

```
   SELECT EMPNO
     FROM EMPLOYEE
     WHERE WORKDEPT LIKE 'E%'
UNION
SELECT EMPNO
   FROM EMP_ACT
   WHERE PROJNO IN('MA2100', 'MA2110', 'MA2112')
```

例 3: 例 2 と同じ照会を行い、さらに EMPLOYEE 表の行には 'emp'、EMP_ACT 表の行には 'emp_act' という “タグ” を付けます。例 2 の結果とは異なり、この照会では、同じ EMPNO が複数回戻され、付加される “タグ” によりどの表からとられたかが示されます。

```
SELECT EMPNO, 'emp'
    FROM EMPLOYEE
   WHERE WORKDEPT LIKE 'E%'
UNION
SELECT EMPNO, 'emp_act' FROM EMP_ACT
   WHERE PROJNO IN('MA2100', 'MA2110', 'MA2112')
```

例 4: 例 2 と同じの照会を行いますが、重複行が除去されないように UNION ALL を使用します。

```
   SELECT EMPNO
     FROM EMPLOYEE
     WHERE WORKDEPT LIKE 'E%'
UNION ALL
SELECT EMPNO
   FROM EMP_ACT
   WHERE PROJNO IN('MA2100', 'MA2110', 'MA2112')
```

例 5: 例 3 と同じ照会を行いますが、現在どの表にもない 2 人の従業員を追加して、それらの行に ″new″ というタグを付けます。

```
SELECT EMPNO, 'emp'
    FROM EMPLOYEE
   WHEREWORKDEPTLIKE 'E%'
UNION
SELECT EMPNO, 'emp_act'
   FROM EMP_ACT
   WHERE PROJNO IN('MA2100', 'MA2110', 'MA2112')
UNION
   VALUES ('NEWAAA', 'new'), ('NEWBBB', 'new')
```

例 6: この EXCEPT の例は、T1 に存在し、T2 に存在しない行をすべて生成します。

```
  (SELECT * FROM T1)
  EXCEPT ALL
  (SELECT * FROM T2)
```

NULL 値が関与していない場合、この例は次の例と同じ結果を戻します。

```
SELECT ALL *
   FROM T1
   WHERE NOT EXISTS (SELECT * FROM T2
                WHERE T1.C1 = T2.C1 AND T1.C2 = T2.C2 AND...)
```

例 7: この INTERSECT の例は、表 T1 と T2 の両方にあるすべての行を生成し、重複した行を除去します。

```
(SELECT * FROM T1)
INTERSECT
(SELECT * FROM T2)
```

NULL 値が関与していない場合、この例は次の例と同じ結果を戻します。

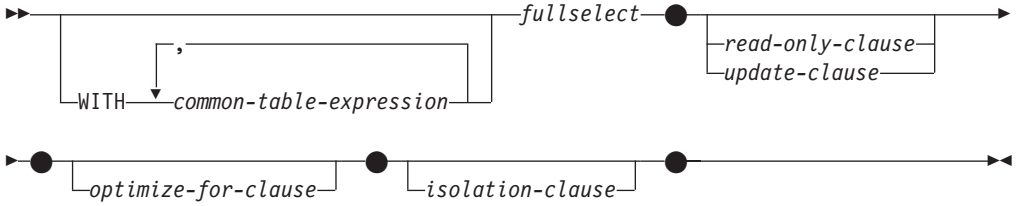
```
SELECT DISTINCT * FROM T1
   WHERE EXISTS (SELECT * FROM T2
                WHERE T1.C1 = T2.C1 AND T1.C2 = T2.C2 AND...)
```

ここで、C1、C2、などは T1 と T2 の列を表します。

関連資料:

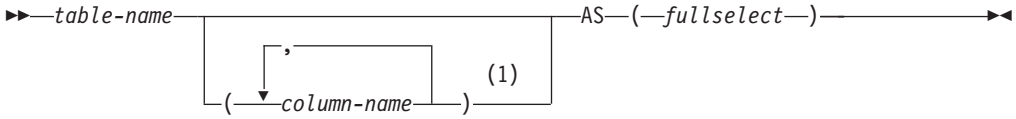
- 126 ページの『結果データ型の規則』
- 131 ページの『ストリング変換の規則』

Select-statement



select-statement は、DECLARE CURSOR ステートメントに直接指定することや、準備した後で DECLARE CURSOR ステートメントで参照することができる形式の照会です。また、選択ステートメントはコマンド行プロセッサー (または同種のツール) を使用する 動的 SQL ステートメントを使って発行することができ、それにより、結果表を画面に表示することもできます。いずれの場合も、*select-statement* によって指定される表は、全選択 (fullselect) の結果です。

common-table-expression



注:

1 共通表式 (common table expression) が再帰的である場合、あるいは全選択の結果として列名が重複する場合は、列名を指定する必要があります。

*共通表式*を使用すると、結果表を *table-name (表名)* によって定義して、それをその後に続く全選択の任意の FROM 文節に指定できるようにすることができます。単一の WITH キーワードの後に、複数の共通表式を指定することができます。指定する各共通表式は、それ以降の共通表式の FROM 文節の中でも名前によって参照することができます。

列のリストを指定する場合、その中の列の名前の数は、全選択の結果表内の列数と同じ数でなければなりません。各 *column-name* (列名) は、ユニークで、しかも非修飾でなければなりません。これらの列名を指定しない場合、共通表式の定義に使用された全選択の選択リストから名前が得られます。

共通表式の *table-name* は、同じステートメントの他の共通表式の *table-name* すべてと異なるものでなければなりません (SQLSTATE 42726)。共通表式が INSERT ステートメントに指定されている場合、*table-name* を、その挿入の対象である表またはビューの名前にすることはできません (SQLSTATE 42726)。共通表式の *table-name* は、その全選択を通じて、どの FROM 文節の中でも表名として指定することができます。共通表式の *table-name* は、(カタログの中で) 同じ修飾名の既存の表、ビュー、または別名をオーバーライドするものとなります。

同じステートメントの中に複数の共通表式が定義されている場合、共通表式相互間の循環参照があってはなりません (SQLSTATE 42835)。*循環参照* が生じるのは、 2 つの共通表式 *dt1* と *dt2* が作成された場合に、 *dt1* が *dt2* を参照し、 *dt2* が *dt1* を参照するようになる場合です。

| 共通表式の全選択の FROM 文節に *data-change-table-reference* が入っている場合
| は、その共通表式がデータを変更するように指示を受けます。データを変更する共
| 通表式は、その共通表式がステートメントの別の箇所で使用されているかどうかに
| 関係なく、ステートメントが処理されるときに常に評価されます。データの読み取
| りまたは変更を行う共通表式が 1 つでもある場合は、すべての共通表式が発生した
| 順に処理されます。そして、データの読み取りまたは変更を行う各共通表式は、制
| 約やトリガーもすべて含めて完全に実行されます。 1 つの共通表式が完全に実行さ
| れるまで、次の共通表式は実行されません。

共通表式は、 CREATE VIEW および INSERT の全選択 (fullselect) の前でもオプションとして使用できます。

共通表式は、以下のような場合に使用することができます。

- ビューの代わりに使用して、ビューが作成されないようにするため (ビューを一般的に使用する必要がなく、定位置の更新や削除を使わない場合)
- スカラー副選択や可変の関数または外部処理を伴う関数から得られる列によりグループ化できるようにする場合
- 必要な結果表がホスト変数に基づいたものである場合
- 同じ結果表を全選択で共用する必要がある場合
- 結果表を再帰的に派生させる必要がある場合
- 照会の中で複数の SQL データ変更ステートメントを処理する必要がある場合

共通表式の全選択の FROM 文節の中にそれ自体への参照が入っている場合、その共通表式は、*再帰的共通表式* です。再帰処理を使用した照会は、部品表 (BOM)、予約システム、およびネットワーク・プランなどのアプリケーションをサポートする上で役立ちます。

再帰的共通表式では、以下のことが成り立っていなければなりません。

- 再帰サイクルの一部をなす各全選択は、 SELECT または SELECT ALL で始まっていなければなりません。 SELECT DISTINCT は使用できません (SQLSTATE 42925)。また、集合の和を求める場合には UNION ALL を使用する必要があります (SQLSTATE 42925)。
- 共通表式の *table-name (表名)* の後には、必ず列名を指定する必要があります (SQLSTATE 42908)。
- 最初の UNION の最初の全選択 (初期化全選択) には、どの FROM 文節の共通表式のどの列に対する参照も入っていてはなりません (SQLSTATE 42836)。
- 共通表式の列名が反復全選択において参照される場合、その列のデータ型、長さ、およびコード・ページは、初期化全選択に基づいて決められます。反復全選択の中の対応する列のデータ型と長さは、初期化全選択に基づいて決められたデータ型と長さと同じでなければならず、コード・ページは一致していなければなりません (SQLSTATE 42825)。ただし、文字ストリング・タイプの場合は、 2 つのデータ型の長さが違っても構いません。この場合、反復全選択の列の長さは、初期化全選択から決められた長さに常に割り当て可能な長さでなければなりません。
- 再帰サイクルの一部をなす各全選択には、列関数、GROUP BY 文節、または HAVING 文節が入っていてはなりません (SQLSTATE 42836)。

これらの全選択の FROM 文節には、再帰サイクルの一部である共通表式に対する参照を多くても 1 つまで組み入れることができます (SQLSTATE 42836)。

- 反復全選択および全体再帰的全選択には、 ORDER BY 文節を組み入れることはできません (SQLSTATE 42836)。
- 副照会 (スカラーまたは定量化されたもの) が再帰サイクルの一部であってはなりません (SQLSTATE 42836)。

再帰的共通表式を開発するときには、無限再帰サイクル (ループ) が作成される恐れについて常に注意してください。再帰サイクルは、必ず終了するようにしてください。これは、関係しているデータが循環している場合に特に重要です。再帰的共通表式には、無限ループを防止する述部を組み入れるようにしてください。再帰的共通表式には、以下のものを組み入れるようにしてください。

- 反復全選択の中に、定数ずつ増分される整数列。
- ″counter_col < constant″ または ″counter _col < :hostvar″ の形式の反復全選択の WHERE 文節の述部。

この構文が再帰的共通表式に見つからないなら、警告が出されます (SQLSTATE 01605)。

再帰の例: 部品表:

部品表 (BOM) のアプリケーションは、多くの業務環境において一般的に必要になります。 BOM アプリケーションの再帰的共通表式の機能を示すため、部品とそれに関連する副部品、そして各部品に必要な副部品の数量を示す表について考えてみます。この例では、以下のようにして表を作成します。

```
CREATE TABLE PARTLIST
               (PART VARCHAR(8),
                SUBPART VARCHAR(8),
                QUANTITY INTEGER);
```

この例の照会結果を示すために、PARTLIST 表には次のようなデータが入っているものとします。

```
PART     SUBPART  QUANTITY
-------- -------- -----------
00       01                5
00       05                3
01       02                2
01       03                3
01       04                4
01       06                3
02       05                7
02       06                6
03       07                6
04       08               10
04       09               11
05       10               10
05       11               10
06       12               10
06       13               10
07       14                8
07       12                8
```

例 1: 単一レベルの展開

最初の例は、単一レベルの展開と呼ばれるものです。これは「'01' で示されている部品を作成するにはどの部品が必要か」という質問に答えるものです。このリストには、直接の副部品、副部品の副部品などが入ります。しかし、ある部品が何回も使用される場合でも、その副部品は 1 回しかリストに示されません。

```
WITH RPL (PART, SUBPART, QUANTITY) AS
     (  SELECT ROOT.PART, ROOT.SUBPART, ROOT.QUANTITY
       FROM PARTLIST ROOT
       WHERE ROOT.PART = '01'
       UNION ALL
        SELECT CHILD.PART, CHILD.SUBPART, CHILD.QUANTITY
          FROM RPL PARENT, PARTLIST CHILD
          WHERE PARENT.SUBPART = CHILD.PART
     )
SELECT DISTINCT PART, SUBPART, QUANTITY
 FROM RPL
  ORDER BY PART, SUBPART, QUANTITY;
```

上記の照会では、*RPL* という名前で指定されている共通表式が組み込まれており、それによってこの照会の再帰的な部分が表されています。この照会には、再帰的共通表式の基本的なエレメントが示されています。

UNION の第 1 オペランド (全選択) は*初期化全選択* と呼ばれるもので、それによって部品 '01' の直接の子が求まります。この全選択の FROM 文節ではソース表が参照されていますが、それ自身 (この場合は *RPL*) を参照することはありません。この最初の全選択の結果が、共通表式 *RPL* (再帰的 PARTLIST) の中に入れられることになります。この例の場合、UNION は常に UNION ALL でなければなりません。

UNION の第 2 オペランド (全選択) は、 *RPL* を使って、副部品の副部品を計算しています。これは、FROM 文節で共通表式 *RPL* とソース表 (CHILD: 子) の部品を、 *RPL* (PARENT: 親) に入っている現行の結果の副部品に結び付けることによります。この結果は、再度 *RPL* に入れられます。このようにして、UNION の第 2 オペランドは、子が存在しなくなるまで繰り返し使用されます。

この照会の主要な全選択の SELECT DISTINCT では、同じ部品/副部品が 2 回以上リストに現れることがないようにしています。

照会結果は、次のようになります。

```
PART     SUBPART  QUANTITY
-------- -------- -----------
01       02                 2
01       03                 3
01       04                 4
01       06                 3
02       05                 7
02       06                 6
03       07                 6
04       08                10
04       09                11
05       10                10
05       11                10
06       12                10
06       13                10
07       12                 8
07       14                 8
```

この結果では、部品 '01' から '02' へ、そこからさらに '06' へと進むようになっています。さらに、部品 '06' へは、'01' から直接に 1 回、 '02' から 1 回の計 2 回達することに注意してください。しかしこの出力では、そのサブコンポーネントが 1 回しかリストに現れないようになっています (これは SELECT DISTINCT を使用した結果です)。

再帰的共通表式では、*無限ループ* になる可能性を必ず考慮に入れてください。この例で、親表と子表を結合する第 2 オペランドの検索条件を以下のようにコーディングしたとすると、無限ループが作成されることになります。

```
PARENT.SUBPART = CHILD.SUBPART
```

無限ループが発生するこの例は、意図したとおりにコーディングされていない場合であることは明らかです。しかし、再帰サイクルが必ず終了するようにコーディングすることにも注意を払ってください。

この例の照会によって得られる結果は、再帰的共通表式を使用しなくても、アプリケーション・プログラム内で作成することができます。しかし、そのような方法では、すべての再帰レベルごとに新しい照会を開始する必要があります。さらに、すべての結果をデータベースに入れ、その結果を並べ替えることを、アプリケーションで行う必要があります。そのような方法では、アプリケーションのロジックが複雑になり、パフォーマンスはよくありません。要約正展開やインデント正展開の照会など、その他の部品表の照会では、アプリケーションのロジックがさらに複雑で効率の悪いものとなってしまいます。

例 2: 要約正展開

2 番目の例は、要約正展開の例です。ここでの質問は、「部品 '01' の作成には各部品が合計どれくらい必要か」というものです。単一レベル正展開と異なる主な点は、数量を集計する必要があるということです。例 1 は、部品が必要になったときにそれに必要な副部品の数量を示すものです。部品 '01' を作成するのに、副部品が結局どれだけ必要かは示されていません。

```
WITH RPL (PART, SUBPART, QUANTITY) AS
   (
      SELECT ROOT.PART, ROOT.SUBPART, ROOT.QUANTITY
       FROM PARTLIST ROOT
       WHERE ROOT.PART = '01'
       UNION ALL
      SELECT PARENT.PART, CHILD.SUBPART, PARENT.QUANTITY*CHILD.QUANTITY
          FROM RPL PARENT, PARTLIST CHILD
          WHERE PARENT.SUBPART = CHILD.PART
   )
SELECT PART, SUBPART, SUM(QUANTITY) AS "Total QTY Used"
 FROM RPL
  GROUP BY PART, SUBPART
  ORDER BY PART, SUBPART;
```

上記の照会では、 *RPL* という名前で指定されている再帰的共通表式の中の UNION の第 2 オペランドの選択リストによって、数量の合計が示されています。副部品の使用量を求めるには、親の数量に、親 1 個当たりの子の数量を乗算します。 1 つの部品が異なる複数のロケーションで何回も使用される場合は、もう 1 つ最終的な集計が必要になります。これは、共通表式 *RPL* をグループ化し、主要全選択の選択リストの中で SUM 列関数を使用することによって行います。

照会結果は、次のようになります。

```
PART     SUBPART  Total Qty Used
-------- -------- --------------
01       02                    2
01       03                    3
01       04                    4
01       05                   14
01       06                   15
01       07                   18
01       08                   40
01       09                   44
01       10                  140
01       11                  140
01       12                  294
01       13                  150
01       14                  144
```

この出力のうち、副部品が '06' の行に注目してください。合計使用量の値 15 は、部品 '01' のための直接の数 3 と、部品 '02' のための数 (6) に部品 '01' の数 (2) を掛けたものとを加えた数です。

例 3: 深さの制御

この表の中に存在する部品のレベルが、とりあえず照会で必要なレベルより深い場合はどうなるのでしょうか。つまり、「'01' で指定される部品を作成するために必要な部品の最初の 2 つのレベルはどんなものか」という質問に答えるためには、どんな照会を作成したらよいでしょうか。例をわかりやすくするため、レベル番号を結果に組み入れることにします。

```
WITH RPL (LEVEL, PART, SUBPART, QUANTITY) AS
      (
         SELECT 1,               ROOT.PART, ROOT.SUBPART, ROOT.QUANTITY
       FROM PARTLIST ROOT
       WHERE ROOT.PART = '01'
       UNION ALL
         SELECT PARENT.LEVEL+1, CHILD.PART, CHILD.SUBPART, CHILD.QUANTITY
          FROM RPL PARENT, PARTLIST CHILD
          WHERE PARENT.SUBPART = CHILD.PART
            AND PARENT.LEVEL < 2
      )
 SELECT PART, LEVEL, SUBPART, QUANTITY
   FROM RPL;
```

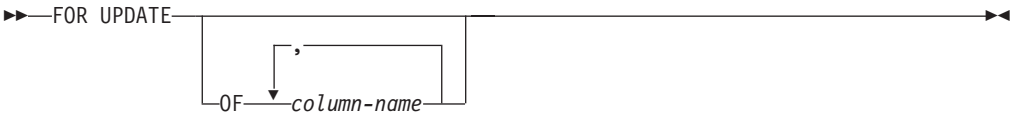
この照会は例 1 に似ています。元の部品からのレベルを示すために、列 *LEVEL* を使っています。初期化全選択では、*LEVEL* 列の値を 1 に初期化しています。それ以降の全選択では、親のレベルに 1 ずつ加算します。次に、結果のレベル数を制御するため、2 番目の全選択に、親のレベルが 2 未満でなければならないという条件が組み入れられています。これによって、2 番目の全選択では、子の処理が 2 次レベルまでしか行われないことになります。

照会結果は、次のようになります。

```
PART     LEVEL       SUBPART  QUANTITY
-------- ----------- -------- -----------
01                 1 02                 2
01                 1 03                 3
01                 1 04                 4
01                 1 06                 3
02                 2 05                 7
02                 2 06                 6
03                 2 07                 6
```

```
04              2 08            10
04              2 09            11
06              2 12            10
06              2 13            10
```

update-clause



FOR UPDATE 文節は、その後実行される UPDATE ステートメントで更新可能となる列を指定します。各 *column-name (列名)* は非修飾でなければならず、全選択の最初の FROM 文節で指定された表またはビューの列を指定するものでなければなりません。列名のない FOR UPDATE 文節を指定すると、全選択の最初の FROM 文節に指定された表またはビューの列のうち更新可能な列すべてが組み込まれます。

以下のいずれかに該当する場合、FOR UPDATE 文節は使用できません。

- 選択ステートメントに関連付けられているカーソルが削除不能である。
- 選択された列のいずれかがカタログ表の更新不能な列であり、 FOR UPDATE 文節を使ってその列が除外されていない。

read-only-clause

FOR READ ONLY 文節は、結果表が読み取り専用であり、カーソルは UPDATE ステートメントおよび DELETE ステートメントで参照されません。 FOR FETCH ONLY も同じ意味です。

結果表の中には、最初から読み取り専用のものがあります。 (読み取り専用ビューに基づく表など。) FOR READ ONLY は、このような表にも指定できますが、指定しても効果はありません。

更新と削除ができない結果表の場合、FOR READ ONLY (または FOR FETCH ONLY) を指定すると、データベース・マネージャーが、ブロッキングを行うことができるため、 FETCH 操作のパフォーマンスが向上する可能性があります。たとえば、FOR READ ONLY 文節または ORDER BY 文節のない動的 SQL ステートメントの入ったプログラムでは、 FOR UPDATE 文節が指定されたかのようにして、データベース・マネージャーがカーソルをオープンする場合があります。したがって、 UPDATE または DELETE ステートメントで照会を使用する場合以外は、パフォーマンスを向上させるために FOR READ ONLY 文節を使用するようにしてください。

読み取り専用結果表は、それが最初から読み取り専用であるか、それとも FOR READ ONLY (FOR FETCH ONLY) として指定されたのかには関係なく、定位置 UPDATE または DELETE ステートメントで参照することはできません。

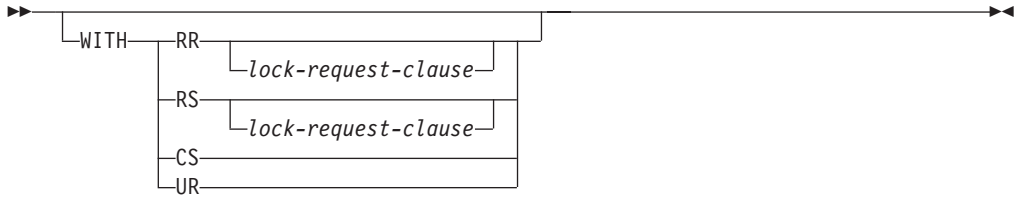
optimize-for-clause

OPTIMIZE FOR 文節は、*選択ステートメント* の特殊な処理を要求します。この文節が省略されると、結果表のすべての行が検索されることが想定されます。指定されている場合には、検索される行数はおそらく *n* を超えないことを前提としています。ここで、*n* は *integer* の値です。 *n* の値は、正の整数でなければなりません。OPTIMIZE FOR 文節を使用すると、 *n* 個の行が検索されることを前提とする照会の最適化に影響を与えます。さらに、ブロックされているカーソルの場合、この文節は、各ブロックで戻される行の数に影響を与えます (すなわち、各ブロックで戻される行の数は *n* 行以下になります)。 *fetch-first-clause* と *optimize-for-clause* の両方が指定されている場合には、これらの文節の integer 値のうちの小さい方を使用して通信バッファー・サイズが決定されます。これらの値は最適化処理専用です。

この文節を指定しても、取り出される行の数が制限されることはなく、パフォーマンス以外ではどんな点でも結果に影響を与えることはありません。 OPTIMIZE FOR *n* ROWS を使用した場合、 *n* 個以下の行を取り出す場合にはパフォーマンスが向上することがありますが、 *n* 個を超える行を取り出す場合にはパフォーマンスが低下する可能性があります。

n の値に行のサイズを乗算した値が、通信バッファーのサイズを超える場合、OPTIMIZE FOR 文節はデータ・バッファーに影響を与えません。通信バッファーのサイズは、RQRIOBLK または ASLHEAPSZ 構成パラメーターによって定義されます。

isolation-clause

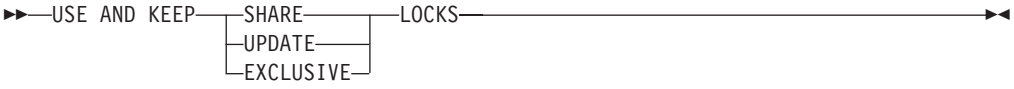


オプションの *isolation-clause* は、ステートメントが実行される分離レベルを指定し、さらに特定のタイプのロックをかけるかどうかを指定します。

- RR - 反復可能読み取り
- RS - 読み取り固定
- CS - カーソル固定
- UR - 非コミット読み取り

ステートメントのデフォルト分離レベルは、ステートメントがバインドされているパッケージの分離レベルです。

lock-request-clause



オプションの *lock-request-clause* は、データベース・マネージャーがかけたままにするロックのタイプを指定します。

SHARE 並行処理プロセスが、データに対する SHARE または UPDATE ロックをかけることができます。

UPDATE 並行処理プロセスが、データに SHARE ロックをかけることができますが、並行処理中のいずれのプロセスも UPDATE または EXCLUSIVE ロックをかけることはできません。

EXCLUSIVE 並行処理プロセスは、データにロックをかけることはできません。

lock-request-clause を適用される対象は、副照会、SQL 関数、および SQL メソッド内のものも含め、照会に必要なすべての基本表と索引スキャンです。これは、プロシージャー、外部関数、または外部メソッドでかけられたロックに対しては効力をもちません。このステートメントで呼び出される (直接または間接に) SQL 関数または SQL メソッドはすべて、 INHERIT ISOLATION LEVEL WITH LOCK REQUEST で作成されたものでなければなりません(SQLSTATE 42601)。トリガーを呼び出す可能性があるか、または参照保全検査を必要とする修正照会と一緒に *lock-request-clause* を使用することはできません (SQLSTATE 42601)。

SELECT ステートメントの例

例 1: EMPLOYEE 表からすべての列と行を選択します。

```
SELECT * FROM EMPLOYEE
```

例 2: PROJECT 表からプロジェクト名 (PROJNAME)、開始日 (PRSTDATE)、および終了日 (PRENDATE) を選択します。その日付が最新の終了日から順に結果表を配列します。

```
SELECT PROJNAME, PRSTDATE, PRENDATE
 FROM PROJECT
 ORDER BY PRENDATE DESC
```

例 3: EMPLOYEE 表のすべての部門の部門番号 (WORKDEPT) と部門別給与 (SALARY) の平均額を選択します。結果表は、部門別給与の平均額の昇順に配列します。

```
SELECT WORKDEPT, AVG(SALARY)
          FROM EMPLOYEE
   GROUP BY WORKDEPT
   ORDER BY 2
```

例 4: C プログラムで使用する UP_CUR という名前のカーソルを宣言して、PROJECT 表の開始日 (PRSTDATE) と終了日 (PRENDATE) の列を更新します。プログラムは、各行のこれらの 2 つの値と、プロジェクト番号 (PROJNO) とを受け取る必要があります。

```
EXEC SQL  DECLARE UP_CUR CURSOR FOR
            SELECT PROJNO, PRSTDATE, PRENDATE
    FROM PROJECT
              FOR UPDATE OF PRSTDATE, PRENDATE;
```

例 5: この例では、SAL+BONUS+COMM に TOTAL_PAY という名前を付けます。

```
SELECT SALARY+BONUS+COMM AS TOTAL_PAY
          FROM EMPLOYEE
  ORDER BY TOTAL_PAY
```

例 6: セールス担当員の従業員番号と給与、およびその部門の給与平均額と人数とを調べます。また、部門別給与平均額と、平均額の最高値も調べます。

ここでは、共通表式を使用することによって、 DINFO ビューを正規のビューとして作成したときのオーバーヘッドを軽減します。ステートメントの作成中に、ビューのカタログにはアクセスされません。これは、全選択の残りの部分のコンテキストにより、ビューによって考慮する必要があるのはセールス担当の部門の行だけだからです。

```
WITH
   DINFO (DEPTNO, AVGSALARY, EMPCOUNT) AS
       (SELECT OTHERS.WORKDEPT, AVG(OTHERS.SALARY), COUNT(*)
         FROM EMPLOYEE OTHERS
         GROUP BY OTHERS.WORKDEPT
       ),
   DINFOMAX AS
       (SELECT MAX(AVGSALARY) AS AVGMAX FROM DINFO)
SELECT THIS_EMP.EMPNO, THIS_EMP.SALARY,
       DINFO.AVGSALARY, DINFO.EMPCOUNT, DINFOMAX.AVGMAX
 FROM EMPLOYEE THIS_EMP, DINFO, DINFOMAX
 WHERE THIS_EMP.JOB = 'SALESREP'
 AND THIS_EMP.WORKDEPT = DINFO.DEPTNO
```

例 7: EMPLOYEE と PROJECT という 2 つの表があるとして、従業員 SALLY を新しい従業員 GEORGE に置き換え、 SALLY が指揮していたプロジェクトをすべて GEORGE に割り当て、更新されたプロジェクトの名前を戻します。

```
WITH
  NEWEMP AS (SELECT EMPNO FROM NEW TABLE
              (INSERT INTO EMPLOYEE(EMPNO, FIRSTNME)
                  VALUES(NEXT VALUE FOR EMPNO_SEQ, 'GEORGE'))),
  OLDEMP AS (SELECT EMPNO FROM EMPLOYEE WHERE FIRSTNME = 'SALLY'),
  UPPROJ AS (SELECT PROJNAME FROM NEW TABLE
              (UPDATE PROJECT
                  SET RESPEMP = (SELECT EMPNO FROM NEWEMP)
                  WHERE RESPEMP = (SELECT EMPNO FROM OLDEMP))),
  DELEMP AS (SELECT EMPNO FROM OLD TABLE
              (DELETE FROM EMPLOYEE
                  WHERE EMPNO = (SELECT EMPNO FROM OLDEMP)))
SELECT PROJNAME FROM UPPROJ;
```

例 8: DEPT 表からデータを取り出します。次にそのデータは、取り出された更新内容に合わせて更新されますが、照会の実行時にはロックされていなければなりません。

```
SELECT DEPTNO, DEPTNAME, MGRNO
FROM DEPT
WHERE ADMRDEPT ='A00'
FOR READ ONLY WITH RS USE AND KEEP EXCLUSIVE LOCKS
```

関連資料:

- 474 ページの『副選択』
- 「*SQL リファレンス 第 2 巻*」の『DECLARE CURSOR ステートメント』

付録 A. SQL の制限値

以下の表は、特定の SQL 制限値を示しています。最も制限が厳しい場合に準拠することによって、プログラマーは容易に移植できるアプリケーション・プログラムを設計することができます。

表 28. ID の長さの制限値

説明	制限値 (バイト単位)
許可名の最大長 (1 バイト文字のみ可)	30
制約名の最大長	18
相関名の最大長	128
条件名の最大長	64
カーソル名の最大長	18
非修飾の列名 (ニックネーム) の最大長	128
データ・ソースの索引名の最大長	128
データ・ソース名の最大長	128
データ・ソースの表名 (*リモート表名*) の最大長	128
外部プログラム名の最大長	8
ホスト識別名の最大長 注 1	255
データ・ソースのユーザー (*リモート許可名*) の ID の最大長	30
ラベル名の最大長	64
メソッド名の最大長	18
パラメーター名の最大長 注 2	128
データ・ソースにアクセスするパスワードの最大長	32
セーブポイント名の最大長	128
スキーマ名の最大長 注 3	30
サーバー名 (データベース別名) の最大長	8
SQL 変数名の最大長	64
ステートメント名の最大長	18
トランスフォーム・グループ名の最大長	18
非修飾の列名の最大長	30
非修飾のパッケージ名の最大長	8
非修飾のユーザー定義タイプ、ユーザー定義関数、ユーザー定義のメソッド、バッファー・プール、表スペース、データベース・パーティション・グループ、トリガー、索引、または索引指定の名前の最大長	18
非修飾表名、ビュー名、ストアード・プロシージャー名、ニックネーム、または別名の最大長	128
ラッパー名の最大長	128

表 28. ID の長さの制限値 (続き)

説明	制限値 (バイト単位)

注:

1. 注 1 ホスト言語コンパイラーによっては、変数名に関してより厳しい制限がある場合があります。
2. 注 2 SQL プロシージャー内のパラメーター名の場合、最大長は 64 バイトです。
3. 注 3 ユーザー定義タイプのスキーマ名の場合、最大長は 8 バイトです。

表 29. 数値の制限値

説明	制限値
INTEGER (整数) の最小値	-2 147 483 648
INTEGER (整数) の最大値	+2 147 483 647
BIGINT (64 ビット整数) の最小値	-9 223 372 036 854 775 808
BIGINT (64 ビット整数) の最大値	+9 223 372 036 854 775 807
SMALLINT (短整数) の最小値	-32 768
SMALLINT (短整数) の最大値	+32 767
10 進数の精度の最大値	31
DOUBLE (倍精度) の最小値	-1.79769E+308
DOUBLE (倍精度) の最大値	+1.79769E+308
DOUBLE (倍精度) の正の最小値	+2.225E-307
DOUBLE (倍精度) の負の最大値	-2.225E-307
REAL (実数) の最小値	-3.402E+38
REAL (実数) の最大値	+3.402E+38
REAL (実数) の正の最大値	+1.175E-37
REAL (実数) の負の最大値	-1.175E-37

表 30. ストリングの制限値

説明	制限値
CHAR の最大長 (バイト単位)	254
VARCHAR の最大長 (バイト単位)	32 672
LONG VARCHAR の最大長 (バイト単位)	32 700
CLOB の最大長 (バイト単位)	2 147 483 647
GRAPHIC の最大長 (文字単位)	127
VARGRAPHIC の最大長 (文字単位)	16 336
LONG VARGRAPHIC の最大長 (文字単位)	16 350
DBCLOB の最大長 (文字単位)	1 073 741 823
BLOB の最大長 (バイト単位)	2 147 483 647
文字定数の最大長	32 672
GRAPHIC 定数の最大長	16 336
連結後の文字ストリングの最大長	2 147 483 647
連結後の GRAPHIC ストリングの最大長	1 073 741 823
連結後のバイナリー・ストリングの最大長	2 147 483 647

表 30. ストリングの制限値 (続き)

説明	制限値
16 進定数の最大桁数	16 336
カタログ・コメントの最大サイズ (バイト単位)	254
ランタイムの構造化タイプ列オブジェクトの最大インスタンス (ギガバイト単位)	1

表 31. 日付/時刻の制限値

説明	制限値
DATE (日付) の最小値	0001-01-01
DATE (日付) の最大値	9999-12-31
TIME (時刻) の最小値	00:00:00
TIME (時刻) の最大値	24:00:00
TIMESTAMP (タイム・スタンプ) の最小値	0001-01-01-00.00.00.000000
TIMESTAMP (タイム・スタンプ) の最大値	9999-12-31-24.00.00.000000

表 32. データベース・マネージャーの制限値

説明	制限値
表の列の最大数 注 7	1 012
ビューの列の最大数 注 1	5 000
すべてのオーバーヘッドを含む行の最大長 注 2 注 7	32 677
パーティション当たりの表の最大サイズ (ギガバイト単位) 注 3 注 7	512
パーティション当たりの索引の最大サイズ (ギガバイト単位)	512
パーティション当たりの表の行の最大数	4×10^9
索引キーの最大長 (すべてのオーバーヘッドを含む) (バイト単位)	1 024
索引キーの列の最大数	16
表の索引の最大数	32 767 またはストレージのサイズによる
SQL ステートメントまたはビューで参照される表の最大数	ストレージ
プリコンパイル済みプログラム中のホスト変数宣言の最大数 注 3	ストレージ
SQL ステートメント中のホスト変数参照の最大数	32 767
挿入または更新に使用するホスト変数の最大長 (バイト単位)	2 147 483 647
SQL ステートメントの最大長 (バイト単位)	2 097 152
選択リストでのエレメントの最大数 注 7	1 012
WHERE または HAVING 文節の述部の最大数	ストレージ
GROUP BY 文節中の列の最大数 注 7	1 012
GROUP BY 文節中の列の合計長の最大値 (バイト単位) 注 7	32 677
ORDER BY 文節中の列の最大数 注 7	1 012
ORDER BY 文節中の列の合計長の最大値 (バイト単位) 注 7	32 677
SQLDA の最大サイズ (バイト単位)	ストレージ

表 32. データベース・マネージャーの制限値 (続き)

説明	制限値
準備されるステートメントの最大数	ストレージ
プログラムで宣言できるカーソルの最大数	ストレージ
1 度にオープンできるカーソルの最大数	ストレージ
SMS 表スペース中の表の最大数	65 534
表に対する制約の最大数	ストレージ
副照会ネストの最大レベル	ストレージ
単一のステートメント中の副照会の最大数	ストレージ
INSERT ステートメント中の値の最大数 注 7	1 012
単一の UPDATE ステートメント中の SET 文節の最大数 注 7	1 012
ユニーク (UNIQUE) 制約の列の最大数 (ユニーク索引によってサポートされる)	16
ユニーク (UNIQUE) 制約の列の結合後の最大長 (ユニーク索引によってサポートされる) (バイト単位)	1 024
外部キーで参照される列の最大数	16
外部キーで参照される列の結合後の最大長 (バイト単位)	1 024
チェック制約の指定の最大長 (バイト単位)	65 535
区分化キーの列の最大数 注 5	500
作業単位で変更される行の最大数	ストレージ
パッケージの最大数	ストレージ
ステートメント中の定数の最大数	ストレージ
サーバーの並行ユーザーの最大数 注 4	64 000
ストアード・プロシージャー中のパラメーターの最大数	32 767
ユーザー定義関数のパラメーターの最大数	90
トリガーのカスケード・ランタイムの最大の深さ	16
同時に起動できるイベント・モニターの最大数	32
REGULAR DMS 表スペース中の最大サイズ (ギガバイト単位) 注 3 注 7	512
長形式 DMS 表スペースの最大サイズ (テラバイト単位) 注 3	2
一時 DMS 表スペースの最大サイズ (テラバイト単位) 注 3	2
インスタンス当たりの並行使用可能なデータベースの最大数	256
インスタンス当たりの並行ユーザーの最大数	64 000
データベース当たりの並行アプリケーションの最大数	60 000
DB2 クライアント内のプロセス当たりの接続の最大数	512
トリガーのカスケードの最大の深さ	16
最大パーティション番号	999
DMS 表スペース中の表オブジェクトの最大数 注 6	51 000
変数の索引キー部分の最大長 (バイト単位) 注 8	1022 またはストレージのサイズによる

表 32. データベース・マネージャーの制限値 (続き)

説明	制限値
ニックネームによって参照されるデータ・ソース表またはビューにある列の最大数	5 000
32 ビット・リリースでのバッファー・プールの最大 NPAGES	1 048 576
64 ビット・リリースでのバッファー・プールの最大 NPAGES	2 147 483 647
すべてのバッファー・プール・スロットの最大合計サイズ (4K)	2 147 483 646
ストアード・プロシージャーの最大ネスト・レベル	16
データベース内の表スペースの最大数	32 768
構造化タイプ内の属性の最大数	4082
トランザクション内の同時にオープンできる LOB ロケーターの最大数	32 100

注:

1. 注 1 この最大値は、 CREATE VIEW ステートメントで結合を使うことによって達成できます。 そのようなビューからの選択は、選択リスト内のエレメントの最大数の制限値によって制限されます。
2. 注 2 BLOB、CLOB、LONG VARCHAR、DBCLOB、および LONG VARGRAPHIC の列の実際のデータは、このカウントには含まれません。 しかし、そのデータの格納場所についての情報のために、行内に一定のスペースが確保されます。
3. 注 3 示されている数値は、アーキテクチャー上の制限値であり、近似値です。 実際の制限値はもっと小さい場合があります。
4. 注 4 実際の値は、MAXAGENTS 構成パラメーターの値になります。
5. 注 5 これは、アーキテクチャー上の制限値です。 実際上の制限値としては、索引キーの列の最大数に関する制限値を使用する必要があります。
6. 注 6 表オブジェクトには、データ、索引、LONG VARCHAR または VARGRAPHIC 列、および LOB 列が組み入れられます。 表データと同じ表スペース中にある表オブジェクトは、制限値に対して余分のオブジェクトとしてカウントされません。しかし、表データとは異なる表スペースにある各表オブジェクトは、表オブジェクトがある表スペース中の表当たりの表オブジェクト・タイプの制限値に対して、実際に 1 個のオブジェクトとしてカウントされます。
7. 注 7 ページ・サイズのユニークな値については、 532 ページの表 33 を参照してください。
8. 注 8 これには、オーバーヘッドがすべて組み入れられており、最も長い索引キーによってのみ制限されます (バイト単位)。索引キー部分の数が増えるにつれて、各キー部分の最大長も増加します。

表 33. データベース・マネージャーのページ・サイズのユニークな制限値

説明	4K ページ・サイズの制限値	8K ページ・サイズの制限値	16K ページ・サイズの制限値	32K ページ・サイズの制限値
表の列の最大数	500	1 012	1 012	1 012
すべてのオーバーヘッドを含む行の最大長	4 005	8 101	16 293	32 677
パーティション当たりの表の最大サイズ (ギガバイト単位)	64	128	256	512
パーティション当たりの索引の最大サイズ (ギガバイト単位)	64	128	256	512
選択リストのエレメントの最大数	500	1 012	1 012	1 012
GROUP BY 文節中の列の最大数	500	1 012	1 012	1 012
GROUP BY 文節中の列の合計長の最大値 (バイト単位)	4 005	8 101	16 293	32 677
ORDER BY 文節中の列の最大数	500	1 012	1 012	1 012
ORDER BY 文節中の列の合計長の最大値 (バイト単位)	4 005	8 101	16 293	32 677
INSERT ステートメント中の値の最大数	500	1 012	1 012	1 012
単一の UPDATE ステートメント中の SET 文節の最大数	500	1 012	1 012	1 012
通常 DMS 表スペース中の最大サイズ (ギガバイト単位)	64	128	256	512

関連資料:

- 「*管理ガイド: パフォーマンス*」の『maxagents - 「エージェントの最大数」構成パラメーター』

付録 B. SQLCA (SQL 連絡域)

SQLCA は、すべての SQL ステートメントのそれぞれ実行の終了時に更新される変数の集まりです。実行可能な SQL ステートメントを収めていて、オプション LANGLEVEL SAA1 (デフォルト) または MIA を指定してプリコンパイルされたプログラムは、 1 つだけの SQLCA を用意する必要があります。ただし、マルチスレッドのアプリケーションでは、スレッドごとに 1 つの SQLCA を用意し、その結果 SQLCA が複数になることがあります。

オプション LANGLEVEL SQL92E を指定してプログラムをプリコンパイルした場合、 SQLCODE 変数または SQLSTATE 変数を SQL 宣言セクションで宣言することができ、また SQLCODE 変数をプログラムで宣言することができます。

LANGLEVEL SQL92E を使用すると、SQLCA は用意されません。 REXX 以外のすべての言語では、 SQL INCLUDE ステートメントを使用して SQLCA を宣言することができます。 REXX では、自動的に SQLCA が用意されます。

コマンド行プロセッサーでそれぞれのコマンドを実行した後に SQLCA を表示するには、コマンド **db2 -a** を実行します。これにより、SQLCA は後続のコマンドの出力の一部として表示されます。同時に SQLCA は db2diag.log ファイルにダンプされます。

SQLCA フィールド記述

表 34. SQLCA のフィールド： この表によって示されているフィールド名は、 INCLUDE ステートメントによって得られる SQLCA のものです。

名前	データ・タイプ	フィールドの値
sqlcaid	CHAR(8)	'SQLCA' の入ったストレージ・ダンプの「目印」。 SQL プロシージャー本体の解析から行番号情報が戻される場合、6 番目のバイトは 'L' になります。
sqlcabc	INTEGER	SQLCA の長さ (136)。
sqlcode	INTEGER	SQL 戻りコード。 **コード 意味** **0** 正常に実行された (1 つ以上の SQLWARN 標識が設定される場合があります)。 **正の値** 正常に実行されたが、警告状態である。 **負の値** エラー状態。
sqlerrml	SMALLINT	*sqlerrmc* の長さ標識 (0 から 70 の範囲)。 0 は、*sqlerrmc* の値が関係ないことを示します。

表 34. SQLCA のフィールド (続き): この表によって示されているフィールド名は、INCLUDE ステートメントによって得られる SQLCA のものです。

名前	データ・タイプ	フィールドの値
sqlerrmc	VARCHAR (70)	X'FF' で区切られた 1 つまたは複数のトークンが入り、エラー条件の説明の中の変数を置き換えます。 このフィールドは、正常に接続が完了した場合にも使用されます。 NOT ATOMIC コンパウンド SQL ステートメントが発行されると、最大 7 つのエラーに関する情報を入れることができます。 最後のトークンの後に X'FF' が続くことがあります。*sqlerrml* 値には、後続の X'FF' が付きます。
sqlerrp	CHAR(8)	製品を示す 3 文字の ID の後に、製品のバージョン、リリース、および修正レベルを示す 5 桁の数字が続きます。たとえば SQL08010 は、DB2 Universal Database バージョン 8 リリース 1 修正レベル 0 を指します。 SQLCODE がエラー条件を示している場合、そのエラーを戻したモジュールがこのフィールドに示されます。 このフィールドは、正常に接続が完了した場合にも使用されます。
sqlerrd	ARRAY	診断情報の提供に使用される 6 個の INTEGER 変数。これらの値は、エラーがない場合は通常は空ですが、パーティション・データベースの sqlerrd(6) は例外です。
sqlerrd(1)	INTEGER	接続が正常に起動された場合、アプリケーションのコード・ページからデータベースのコード・ページに変換する際に予想される混合文字データ (CHAR データ型) の長さの差の最大値が入ります。値 0 または 1 は、拡張がないことを示します。 1 よりも大きい値は、その長さの分の拡張が見込まれることを示します。負の値は、切り捨てが見込まれることを示します。 SQL プロシージャーから正常に戻された場合、その SQL プロシージャーからの戻り状況値が入ります。
sqlerrd(2)	INTEGER	接続が正常に起動された場合、データベースのコード・ページからアプリケーションのコード・ページに変換する際に予想される混合文字データ (CHAR データ型) の長さの差の最大値が入ります。値 0 または 1 は、拡張がないことを示します。 1 よりも大きい値は、その長さの分の拡張が見込まれることを示します。負の値は、切り捨てが見込まれることを示します。SQLCA がエラーの発生した NOT ATOMIC コンパウンド SQL ステートメントの結果である場合、この値はエラーの発生したステートメントの番号に設定されます。

表 34. SQLCA のフィールド (続き): この表によって示されているフィールド名は、INCLUDE ステートメントによって得られる SQLCA のものです。

名前	データ・タイプ	フィールドの値
sqlerrd(3)	INTEGER	PREPARE を呼び出して正常に実行した場合は、戻される行の数の見積もりが入ります。 INSERT、UPDATE、DELETE、または MERGE を実行した後は、実際にその操作のために修飾された行の数になります。コンパウンド SQL を呼び出した場合は、すべてのサブステートメント行の累計が入ります。 CONNECT が呼び出され、データベースが更新可能な場合は 1、データベースが読み取り専用の場合は 2 が入ります。 OPEN ステートメントが呼び出され、カーソルに SQL データ変更ステートメントが入っている場合、このフィールドには、組み込まれた挿入、更新、削除、またはマージ操作のために修飾された行の合計が入ります。 SQL プロシージャーで CREATE PROCEDURE を呼び出したものの、 SQL プロシージャー本体の解析でエラーが発生した場合、エラーが発生した行番号が入ります。この場合、有効な行番号を表すために、sqlcaid の 6 バイト目は必ず 'L' になります。
sqlerrd(4)	INTEGER	PREPARE が正常に呼び出された場合、ステートメントを処理するのに必要なリソースの相対コスト見積もりが入ります。コンパウンド SQL を呼び出した場合は、正常に実行されたサブステートメントの数のカウントが入ります。 CONNECT を呼び出した場合は、下位レベルのクライアントからの 1 フェーズ・コミットなら 0、 1 フェーズ・コミットなら 1、1 フェーズの読み取り専用コミットなら 2、 2 フェーズ・コミットなら 3 が入ります。
sqlerrd(5)	INTEGER	以下の両方の結果として削除、挿入、または更新された行の合計数が入ります。 • 削除操作が正常に実行された後の制約の実施 • 起動したトリガーから引き起こされた SQL ステートメントの処理 コンパウンド SQL を呼び出した場合は、すべてのサブステートメントの上記のような行の累計が入ります。場合によっては、エラーが発生した場合に、内部エラー・ポインターである負の値が入ります。 CONNECT が呼び出された場合に入る値は、サーバー認証の場合は 0、クライアント認証の場合は 1、 DB2 Connect を使用した認証の場合は 2、SERVER_ENCRYPT 認証の場合は 4、暗号化を指定された DB2 Connect を使った認証では 5、 KERBEROS 認証の場合は 7、KRB_SERVER_ENCRYPT 認証の場合は 8、GSSPLUGIN 認証の場合は 9、GSS_SERVER_ENCRYPT 認証の場合は 10、 認証が指定されなければ 255 です。

表 34. SQLCA のフィールド (続き): この表によって示されているフィールド名は、INCLUDE ステートメントによって得られる SQLCA のものです。

名前	データ・タイプ	フィールドの値
sqlerrd(6)	INTEGER	パーティション・データベースの場合、エラーまたは警告が出されたパーティションのパーティション番号が入ります。エラーまたは警告が出されなかった場合は、このフィールドにはコーディネーター・ノードのパーティション番号が入ります。 このフィールドに入る番号は、 db2nodes.cfg ファイルでパーティションに対して指定されたものと同じです。
sqlwarn	配列	一連の警告標識で、それぞれの標識はブランクか W です。コンパウンド SQL を呼び出した場合は、すべてのサブステートメントについての警告標識の累積が入ります。
sqlwarn0	CHAR(1)	他の標識がすべてのブランクの場合はブランク、 1 つでもブランクでない標識があれば W。
sqlwarn1	CHAR(1)	ホスト変数への割り当て時にストリング列の値が切り捨てられた場合は W になります。 NULL 終止符が切り捨てられた場合は N になります。 CONNECT または ATTACH が正常に実行され、その接続の許可名が 8 バイトを超える場合は A になります。 sqlerrd(4) 内に保管されている PREPARE ステートメントの相対コスト見積もりが、INTEGER 内に保管できたであろう値を超える場合や、1 より小さかった場合に、 CURRENT EXPLAIN MODE または CURRENT EXPLAIN SNAPSHOT 特殊レジスターが NO 以外の値に設定されていると、 P になります。
sqlwarn2	CHAR(1)	関数の引き数から NULL 値が削除された場合、W が入ります。[a]
sqlwarn3	CHAR(1)	列の数とホスト変数の数が等しくない場合、W が入ります。 ASSOCIATE LOCATORS ステートメントに指定されている結果セット・ロケーターが、プロシージャーから戻された結果セットの数より少ない場合、Z が入ります。
sqlwarn4	CHAR(1)	準備された UPDATE または DELETE ステートメントに WHERE 文節が指定されていない場合に、 W が入ります。
sqlwarn5	CHAR(1)	将来の使用のために予約済み。
sqlwarn6	CHAR(1)	存在しない日付を避けるために日付演算の結果が調整された場合に W が入ります。
sqlwarn7	CHAR(1)	将来の使用のために予約済み。 CONNECT が呼び出されて正常に実行された場合に、DYN_QUERY_MGMT データベース構成パラメーターが使用可能であれば、'E' が入ります。
sqlwarn8	CHAR(1)	変換できなかった文字が置換文字に置き換えられた場合に W になります。
sqlwarn9	CHAR(1)	エラーのある算術式が列関数の処理時に無視された場合に W になります。
sqlwarn10	CHAR(1)	SQLCA 内のフィールドのいずれかで、文字データ値の変換時に変換エラーが起きた場合、W が入ります。

表 34. SQLCA のフィールド (続き): この表によって示されているフィールド名は、INCLUDE ステートメントによって得られる SQLCA のものです。

名前	データ・タイプ	フィールドの値
sqlstate	CHAR(5)	直前に実行された SQL ステートメントの結果を示す戻りコード。

[a] 一部の関数では、NULL 値が除去されても SQLWARN2 が W にならないことがあります。これは、結果が NULL 値の除去によるものでない場合に生じます。

エラー報告

エラー報告の順序は、以下のとおりです。

1. 重大エラー条件は必ず報告されます。重大エラーが報告される場合、SQLCA に追加されるものはありません。
2. 重大エラーが発生しなかった場合、デッドロック・エラーはその他のエラーよりも優先します。
3. その他のエラーの場合、最初の負の SQL コードの SQLCA が戻されます。
4. 負の SQL コードが検出されない場合、最初の警告の SQLCA (つまり正の SQL コード) が戻されます。

   パーティション・データベース・システムで、あるパーティションでは空で、他のパーティションにはデータがある表に対してデータ操作コマンドが呼び出される場合は、この規則の例外です。すべてのパーティションで表が空であるため、または UPDATE ステートメントの WHERE 文節の条件を満たす行がもうないために、すべてのパーティションのエージェントが SQL0100W を返した場合のみ、アプリケーションに SQLCODE +100 が返されます。

パーティション・データベース・システムでの SQLCA の使用法

パーティション・データベース・システムでは、異なるパーティションにある多くのエージェントが 1 つの SQL ステートメントを実行する場合があり、それぞれのエージェントが異なるエラーまたは警告についての異なる SQLCA を戻す場合があります。また、コーディネーターのエージェントには独自の SQLCA があります。

アプリケーションについての表示に一貫性を持たせるために、 SQLCA の値はすべて 1 つの構造の中に組み合わされ、 SQLCA のフィールドは以下のようにグローバルなカウントを表示します。

- エラーと警告のすべてについて、*sqlwarn* フィールドにはすべてのエージェントから受け取った警告標識が入ります。
- 行カウントを表示する *sqlerrd* フィールドの値は、すべてのエージェントからの累計です。

トリガーにより実行された SQL ステートメントの処理中に発生したエラーのケースによっては、 SQLSTATE 09000 が戻されないことがあるので注意してください。

付録 C. SQLDA (SQL 記述子域)

SQLDA は、SQL DESCRIBE ステートメントの実行に必要な変数の集まりです。 SQLDA 変数は、PREPARE、OPEN、FETCH、および EXECUTE ステートメントでは、オプションとして使用できます。 SQLDA は、動的 SQL との間で情報を伝えます。これは、DESCRIBE ステートメントで使用され、ホスト変数のアドレスで変更して、 FETCH または EXECUTE ステートメントで再び使用することができます。

SQLDA は、すべての言語でサポートされていますが、事前定義の宣言を使用できるのは、 C、REXX、FORTRAN、および COBOL に対してだけです。

SQLDA の情報の意味は、その使用法によって異なります。 PREPARE と DESCRIBE の場合、 SQLDA は準備されるステートメントに関する情報をアプリケーション・プログラムに提供します。 OPEN、EXECUTE、および FETCH の場合、SQLDA はホスト変数を記述します。

DESCRIBE および PREPARE において、記述する列のいずれかが LOB タイプ (LOB ロケーターおよびファイル参照変数では、 2 倍の SQLDA は必要ありません)、参照タイプ、またはユーザー定義タイプの場合、 SQLDA 全体の SQLVAR 項目の数を 2 倍にする必要があります。たとえば:

- 3 つの VARCHAR の列と 1 つの INTEGER の列の入っている表を記述する場合、 SQLVAR 項目は 4 つになります。
- 2 つの VARCHAR の列と 1 つの CLOB の列、および 1 つの INTEGER の列の入っている表を記述する場合には、 SQLVAR 項目は 8 つになります。

EXECUTE、FETCH、および OPEN において、記述する変数のいずれかが LOB タイプ (LOB ロケーターおよびファイル参照変数では、 2 倍の SQLDA は必要ありません) または構造化タイプの場合、 SQLDA 全体の SQLVAR 項目の数を 2 倍にする必要があります。特殊タイプと参照タイプは、これらの場合には関係ありません。これは、それらのタイプの場合、データベースが 2 倍の数の項目の追加情報を必要とはしないためです。)

SQLDA フィールド記述

SQLDA は、4 つの変数と、その後に SQLVAR と総称して呼ばれる変数の任意の数のオカレンスによって構成されています。 OPEN、FETCH、および EXECUTE では、 SQLVAR の各オカレンスによってホスト変数が記述されます。 DESCRIBE と PREPARE では、 SQLVAR の各オカレンスによって結果表またはパラメーター・マーカーの列が記述されます。 SQLVAR の項目には、以下の 2 つのタイプがあります。

- **基本 SQLVAR:** これらの項目は常に存在します。これらの項目には、データ型のコード、長さ属性、列名、ホスト変数のアドレス、および標識変数アドレスなどの列、パラメーター・マーカー、またはホスト変数に関する基本的な情報が入れられます。

- **副次 SQLVAR:** これらの項目は、上記で概説した規則に従って SQLVAR 項目の数が 2 倍になった場合にのみ存在します。ユーザー定義タイプ (特殊または構造) の場合は、ユーザー定義タイプ名が入ります。参照タイプの場合は、参照のターゲット・タイプが入れられます。 LOB の場合は、ホスト変数の長さ属性と、実際の長さの入っているバッファーを指すポインターが入れられます。 (特殊タイプと LOB の情報が重なり合うことはないので、 DESCRIBE においては、SQLVAR 項目を 3 倍にしなくても、LOB に基づく特殊タイプを使用できます。) ロケーターまたはファイル参照変数を使用して LOB を示す場合、これらの項目は必要ありません。

SQLDA に上記 2 つのタイプの項目が両方とも入っている場合、基本 SQLVAR は副次 SQLVAR のブロックの前のブロックに入れられます。それぞれの場合において、項目の数は SQLD の値で示されます (副次 SQLVAR 項目の多くは使用されない場合があります)。

DESCRIBE によって SQLVAR 項目が設定される環境については、 544 ページの『SQLDA に対する DESCRIBE の効果』に示されています。

SQLDA ヘッダーのフィールド

表 35. SQLDA ヘッダーのフィールド

C での名前	SQL データ型	DESCRIBE および PREPARE で使用する場合 (SQLN を除き、データベース・マネージャーで設定)	FETCH、OPEN、および EXECUTE で使用する場合 (ステートメントの実行前にアプリケーションで設定)
sqldaid	CHAR(8)	このフィールドの 7 番目のバイトは、SQLDOUBLED という名前のフラグ・バイトです。データベース・マネージャーは、それぞれの列に対して 2 つの SQLVAR 項目が作成された場合は SQLDOUBLED を文字 '2' に設定し、その他の場合はブランク (ASCII では X'20'、 EBCDIC では X'40' に設定します)。 SQLDOUBLED がいつ設定されるかについては、 544 ページの『SQLDA に対する DESCRIBE の効果』を参照してください。	このフィールドの 7 番目のバイトは、SQLVAR の数が 2 倍になった場合に使用されます。これは SQLDOUBLED という名前のフィールドです。記述されるホスト変数が構造化タイプ、BLOB、CLOB、または DBCLOB の場合、この 7 番目のバイトは文字 '2' に設定され、それ以外の場合は任意の文字に設定できます (ブランクの使用をお勧めします)。
sqldabc	INTEGER	32 ビットの場合、SQLDA の長さ = SQLN*44+16。 16。 64 ビットの場合、SQLDA の長さ = SQLN*56+16。	32 ビットの場合、SQLDA の長さ >= SQLN*44+16。 64 ビットの場合、SQLDA の長さ >= SQLN*56+16。
sqln	SMALLINT	データベース・マネージャーはこれを変更しません。 DESCRIBE ステートメントを実行する前に、ゼロまたはゼロより大きい値を設定する必要があります。 これは、SQLVAR のオカレンスの合計数を示します。	SQLDA の SQLVAR のオカレンスの合計。 SQLN には、ゼロまたはゼロより大きい値を設定する必要があります。
sqld	SMALLINT	データベース・マネージャーによって、結果表の列の数またはパラメーター・マーカーの数に設定されます。	SQLVAR のオカレンスにより記述されるホスト変数の数

基本 SQLVAR のオカレンスのフィールド

表 36. 基本 SQLVAR のフィールド

資料名	データ型	DESCRIBE および PREPARE で使用する場合	FETCH、OPEN、および EXECUTE で使用する場合
sqltype	SMALLINT	列のデータ型と、その列またはパラメーター・マーカーが NULL 可能能かどうかを示します。(パラメーター・マーカーは常に NULL 可能と見なされます。) 545 ページの表 38 は、許される値とその意味をリストしています。 特殊タイプまたは参照タイプの場合は、その基本タイプのデータ型がこのフィールドに入れられます。構造化タイプの場合は、そのタイプの変換グループ (CURRENT DEFAULT TRANSFORM GROUP 特殊レジスターに基づく) の FROM SQL 変換関数が入れられます。基本 SQLVAR には、それがユーザー定義タイプまたは参照タイプの記述の一部であるかどうかを示す標識はありません。	ホスト変数の場合と同じ。日付/時刻の値のホスト変数は、文字ストリング変数でなければなりません。 FETCH の場合、日付/時刻のタイプ・コードは、固定長文字ストリングを意味します。 sqltype が偶数値の場合、sqlind フィールドは無視されます。
sqllen	SMALLINT	列またはパラメーター・マーカーの長さ属性。日付/時刻の列またはパラメーター・マーカーの場合は、値のストリング表記の長さ。 545 ページの表 38 を参照してください。 ラージ・オブジェクト・ストリングの場合、この値は 0 に設定されます。その長さ属性が 2 バイト整数に入る小さいものであっても、設定値はやはり 0 になります。	ホスト変数の長さ属性。 545 ページの表 38 を参照してください。 CLOB、DBCLOB、および BLOB の列の場合、データベース・マネージャーはこの値を無視します。代わりに、副次 SQLVAR の len.sqllonglen フィールドが使用されます。
sqldata	ポインター	ストリング SQLVARS の場合、 sqldata にはコード・ページが組み入れられます。文字ストリング SQLVAR の場合、 FOR BIT DATA 属性で列を定義すると、 sqldata は 0 になります。ほかの文字ストリング SQLVARS の場合、 sqldata には、 SBCS データの SBCS コード・ページか、 MBCS データの複合 MBCS コード・ページに関連づけられた SBCS コード・ページのいずれかが入っています。日本語の EUC、中国語 (繁体字) の EUC、および Unicode の UTF-8 文字ストリング SQLVARS では、 sqldata にそれぞれ 954、964、および 1208 が組み入れられます。 他のすべてのタイプの列の場合、sqldata は未定義です。	ホスト変数のアドレスが入っています (取り出したデータを保管するロケーション)。

基本 SQLVAR のオカレンスのフィールド

表 36. 基本 SQLVAR のフィールド (続き)

資料名	データ型	DESCRIBE および PREPARE で使用する場合	FETCH、OPEN、および EXECUTE で使用する場合
sqlind	ポインター	文字ストリング SQLVARS の場合、sqlind は 0 になります。ただし、sqlind が複合 DBCS コード・ページに関連付けられた DBCS コード・ページの場合の、 MBCS データは例外です。 他のすべてのタイプの場合、sqlind は未定義です。	関連する標識変数があれば、そのアドレスが入ります。それ以外の場合は、使用されません。 sqltype が偶数値の場合、sqlind フィールドは無視されます。
sqlname	VARCHAR (30)	非修飾の列名またはパラメーター・マーカー名。 システム生成の名前を持つ列およびパラメーター・マーカーの場合、 30 番目のバイトが X'FF' に設定されます。 AS 文節によって列名が指定された場合は、このバイトは X'00' になります。	DB2 Connect を使用してサーバーにアクセスする場合、 FOR BIT DATA ストリングを指定するには、sqlname を以下のように設定します。 • sqlname の長さは 8 • sqlname の最初の 4 バイトは X'00000000' • sqlname の残りのバイトは予約済み (現在は無視される)

副次 SQLVAR のオカレンスのフィールド

表 37. 副次 SQLVAR のフィールド

資料名	データ型	DESCRIBE および PREPARE で使用する場合	FETCH、OPEN、および EXECUTE で使用する場合
len.sqllonglen	INTEGER	BLOB、CLOB、または DBCLOB の列またはパラメーター・マーカーの長さ属性。	BLOB、CLOB、または DBCLOB ホスト変数の長さ属性。データベース・マネージャーは、それらのデータ型に対しては基本 SQLVAR の SQLLEN フィールドを無視します。 長さ属性は、BLOB または CLOB ではバイト数、DBCLOB では文字数になります。
reserve2	32 ビットの場合は CHAR(3)、 64 ビットの場合は CHAR(11)。	使用されません。	使用されません。
sqlflag4	CHAR(1)	SQLVAR の表している参照タイプが sqldatatype_name に指定されたターゲット・タイプに関連付けられたものである場合、この値は X'01' になります。 SQLVAR の表している構造化タイプで、 sqldatatype_name にユーザー定義タイプ名が指定されている場合、値は X'12' になります。それ以外の場合は、値は X'00' です。	SQLVAR の表している参照タイプが sqldatatype_name に指定されたターゲット・タイプに関連付けられたものである場合、 X'01' に設定されます。 SQLVAR の表している構造化タイプで、 sqldatatype_name にユーザー定義タイプ名が指定されている場合、 X'12' に設定されます。それ以外の場合は、値は X'00' です。

表 37. 副次 SQLVAR のフィールド (続き)

資料名	データ型	DESCRIBE および PREPARE で使用する場合	FETCH、OPEN、および EXECUTE で使用する場合
sqldatalen	ポインター	使用されません。	BLOB、CLOB、および DBCLOB ホスト変数でのみ使用されます。 このフィールドが NULL (NULL 値) の場合は、データの直前に実際の長さ (文字単位) を 4 バイトで保管し、SQLDATA はフィールド長の最初のバイトを指すようにする必要があります。 このフィールドが NULL (NULL 値) でない場合は、対応する基本 SQLVAR 内の SQLDATA フィールドの指すバッファー内のデータの実際の長さ (バイト単位、 DBCLOB の場合も含む) の入っている 4 バイト長のバッファーを指すポインターが入れられます。 このフィールドを使用するか否かに関係なく、 len.sqllonglen フィールドは設定する必要があります。
sqldatatype_name	VARCHAR(27)	ユーザー定義タイプの場合、データベース・マネージャーはこれを完全修飾ユーザー定義タイプ名に設定します。注 1 参照タイプの場合は、データベース・マネージャーはこれを参照のターゲット・タイプの完全修飾タイプ名に設定します。	構造化タイプの場合、表の注[1] で示されているフォーマットの完全修飾ユーザー定義タイプ名に設定されます。
予約済み	CHAR(3)	使用されません。	使用されません。

[1] 最初の 8 バイトには、タイプのスキーマ名が入れられます (必要に応じて右側にスペースが入れられます)。 バイト 9 はドット文字 (.) です。 10 から 27 バイトには、タイプ名のうちの下位部分が入れられます。それは、右側にスペースを入れて拡張することはできません。

このフィールドの主な目的は、タイプの名前を入れることですが、 IBM の定義済みデータ型用に設定することもできます。この場合、スキーマ名は SYSIBM、名前の下位部分は DATATYPES カタログ・ビューの TYPENAME 列に保管されている名前になります。たとえば:

```
type name        length   sqldatatype_name
---------        ------   ----------------
A.B              10       A       .B
INTEGER          16       SYSIBM  .INTEGER
"Frank's".SMINT  13       Frank's .SMINT
MY."type  "      15       MY      .type
```

SQLDA に対する DESCRIBE の効果

DESCRIBE OUTPUT または PREPARE OUTPUT INTO ステートメントの場合、データベース・マネージャーは、常に SQLD を結果セットの列の数、または出力パラメーター・マーカーの数に設定します。 DESCRIBE INPUT または PREPARE INPUT INTO ステートメントの場合、データベース・マネージャーは、常に SQLD をステートメント内の入力パラメーター・マーカーの数に設定します。 CALL ステートメント内の INOUT パラメーターに対応するパラメーター・マーカーは、入力記述子と出力記述子の両方で記述されるので注意してください。

SQLDA の SQLVAR は、以下の場合に設定されます。

- SQLN >= SQLD で、しかも LOB、ユーザー定義タイプ、または参照タイプの項目がない

  最初の SQLD SQLVAR 項目が設定され、SQLDOUBLED はブランクに設定されます。
- SQLN >= 2*SQLD で、しかも少なくとも 1 つの項目が LOB、ユーザー定義タイプ、または参照タイプである

  2 倍の数の SQLD SQLVAR 項目が設定され、SQLDOUBLED は '2' に設定されます。
- SQLD <= SQLN < 2*SQLD で、しかも少なくとも 1 つの項目が特殊タイプまたは参照タイプで、 LOB の項目または構造化タイプの項目はない

  最初の SQLD SQLVAR 項目が設定され、SQLDOUBLED はブランクに設定されます。 SQLWARN BIND オプションが YES の場合は、警告 SQLCODE +237 (SQLSTATE 01594) が出されます。

SQLDA の SQLVAR は、以下の場合には設定されません (さらに多くのスペースの割り振りと別の DESCRIBE が必要)。

- SQLN < SQLD で、しかも LOB、ユーザー定義タイプ、または参照タイプの項目がない

  SQLVAR 項目は設定されず、SQLDOUBLED はブランクに設定されます。 SQLWARN BIND オプションが YES の場合は、警告 SQLCODE +236 (SQLSTATE 01005) が出されます。

  DESCRIBE が正常に実行される場合には、SQLD 個の SQLVAR が割り振られます。
- SQLN < SQLD で、しかも少なくとも 1 つの項目が特殊タイプまたは参照タイプで、 LOB の項目または構造化タイプの項目はない

  SQLVAR 項目は設定されず、SQLDOUBLED はブランクに設定されます。 SQLWARN BIND オプションが YES の場合は、警告 SQLCODE +239 (SQLSTATE 01005) が出されます。

  特殊タイプ名および参照タイプのターゲット・タイプを組み込まれた DESCRIBE が正常に実行されると、 2*SQLD 個の SQLVAR が割り振られます。
- SQLN < 2*SQLD で、しかも少なくとも 1 つの項目が LOB または構造化タイプである

SQLVAR 項目は設定されず、SQLDOUBLED はブランクに設定されます。(SQLWARN BIND オプションの設定に関係なく) 警告 SQLCODE +238 (SQLSTATE 01005) が出されます。

DESCRIBE が正常に実行される場合には、2*SQLD 個の SQLVAR が割り振られます。

上記リストでの「LOB 項目」には、ソース・タイプが LOB タイプである特殊タイプの項目も入ります。

DESCRIBE (または PREPARE INTO) から警告 SQLCODE +236、+237、+239 を戻すかどうかを制御するには、 BIND または PREP コマンドの SQLWARN オプションを使用します。 使用するアプリケーション・コードでは、これらの SQLCODE がいつ戻されてもよいようにしておいてください。選択リストに LOB または構造化タイプの項目が入っていて、 SQLDA の中の SQLVAR が不足している場合には、常に警告 SQLCODE +238 が戻されます。これは、結果セット内に LOB または構造化タイプの項目があるために SQLVAR 数を 2 倍にする必要があることをアプリケーションに認識させる唯一の方法です。

構造化タイプの項目を記述しようとしているものの、 FROM SQL トランスフォームが定義されていない場合 (CURRENT DEFAULT TRANSFORM GROUP 特殊レジスターを使用した TRANSFORM GROUP の指定が行われていない (SQLSTATE 42741) か、またはその名前グループが FROM SQL トランスフォーム関数を定義していない (SQLSTATE 42744) ため)、 DESCRIBE はエラーを戻します。このエラーは、構造化タイプの項目がある表の DESCRIBE で戻されるエラーと同じです。

SQLTYPE と SQLLEN

表 38 に、 SQLDA の SQLTYPE フィールドと SQLLEN フィールドに現れる値を示します。 DESCRIBE と PREPARE INTO においては、 SQLTYPE の値が偶数ならその列では NULL 値が使えないこと、また奇数ならその列で NULL 値が可能であることを意味しています。 FETCH、OPEN、および EXECUTE において、SQLTYPE の値が偶数の場合には標識変数がないこと、奇数の場合には SQLIND に標識変数のアドレスが入れられていることを意味しています。

表 38. SQLTYPE と SQLLEN の値 (DESCRIBE、FETCH、OPEN、および EXECUTE の場合)

	DESCRIBE および PREPARE INTO の場合		**FETCH、OPEN、および EXECUTE の場合**	
SQLTYPE	**列のデータ・タイプ**	**SQLLEN**	**ホスト変数のデータ型**	**SQLLEN**
384/385	日付	10	日付の固定長文字ストリング表示	ホスト変数の長さ属性
388/389	time	8	時刻の固定長文字ストリング表示	ホスト変数の長さ属性
392/393	timestamp	26	タイム・スタンプの固定長文字ストリング表示	ホスト変数の長さ属性
396/397	DATALINK	列の長さ属性	DATALINK	ホスト変数の長さ属性

表 38. SQLTYPE と SQLLEN の値 (DESCRIBE、FETCH、OPEN、および EXECUTE の場合) (続き)

	DESCRIBE および PREPARE INTO の場合		FETCH、OPEN、および EXECUTE の場合	
SQLTYPE	列のデータ・タイプ	SQLLEN	ホスト変数のデータ型	SQLLEN
400/401	N/A	N/A	NUL 文字で終了する GRAPHIC ストリング	ホスト変数の長さ属性
404/405	BLOB	0 *	BLOB	使用されません。*
408/409	CLOB	0 *	CLOB	使用されません。*
412/413	DBCLOB	0 *	DBCLOB	使用されません。*
448/449	可変長文字ストリング	列の長さ属性	可変長文字ストリング	ホスト変数の長さ属性
452/453	固定長文字ストリング	列の長さ属性	固定長文字ストリング	ホスト変数の長さ属性
456/457	長形式可変長文字ストリング	列の長さ属性	長形式可変長文字ストリング	ホスト変数の長さ属性
460/461	N/A	N/A	NUL 文字で終了する文字ストリング	ホスト変数の長さ属性
464/465	可変長 GRAPHIC ストリング	列の長さ属性	可変長 GRAPHIC ストリング	ホスト変数の長さ属性
468/469	固定長 GRAPHIC ストリング	列の長さ属性	固定長 GRAPHIC ストリング	ホスト変数の長さ属性
472/473	長形式可変長 GRAPHIC ストリング	列の長さ属性	長形式 GRAPHIC ストリング	ホスト変数の長さ属性
480/481	浮動小数点数	倍精度の場合は 8、単精度の場合は 4	浮動小数点数	倍精度の場合は 8、単精度の場合は 4
484/485	パック 10 進数	バイト 1 は精度、バイト 2 は位取り	パック 10 進数	バイト 1 は精度、バイト 2 は位取り
492/493	64 ビット整数	8	64 ビット整数	8
496/497	64 ビット整数	4	64 ビット整数	4
500/501	短整数	2	短整数	2
916/917	該当なし	該当なし	BLOB ファイル参照変数	267
920/921	該当なし	該当なし	CLOB ファイル参照変数	267
924/925	該当なし	該当なし	DBCLOB ファイル参照変数	267
960/961	該当なし	該当なし	BLOB ロケーター	4
964/965	該当なし	該当なし	CLOB ロケーター	4
968/969	該当なし	該当なし	DBCLOB ロケーター	4

注:

- 副次 SQLVAR の len.sqllonglen フィールドに、列の長さ属性が入れられます。
- SQLTYPE は、DB2 での移植性のために旧バージョンから変更されました。旧バージョンの値 (旧バージョンの SQL リファレンスを参照) は、引き続きサポートされています。

認識されない非サポート SQLTYPE

SQLDA の SQLTYPE フィールドに表示される値は、データの送信側および受信側で使用可能なデータ型サポートのレベルによって異なります。 これは、新しいデータ型が製品に追加される場合に特に重要です。

新しいデータ型は、データの送信側または受信側にサポートされることもあれば、サポートされないこともあり、データの送信側や受信側に認識されないことさえあります。 状況に応じて、新しいデータ型が戻されたり、送信側と受信側の両方が認めた互換データ型が戻されたり、あるいは結果としてエラーが発生したりします。

送信側と受信局が互換データ型の使用に同意する場合、以下に示すマッピングが実行されます。このマッピングは、送信側または受信側の少なくともどちらかが指定データ型をサポートしない場合に実行されます。 非サポート・データ型は、アプリケーションまたはデータベース・マネージャーのどちらかによって提供されます。

データ型	互換データ型
BIGINT	DECIMAL(19, 0)
ROWID[1]	VARCHAR(40) FOR BIT DATA

[1] ROWID は、 DB2 Universal Database (z/OS および OS/390 版) バージョン 6 によってサポートされています。

SQLDA では、データ型が置換されたことは示されないので注意してください。

パック 10 進数

パック 10 進数は、一種のバイナリー・コードによる 10 進数 (BCD) 表記で保管されます。 BCD においては、1 ニブル (4 ビット) で 10 進数の 1 桁が表されます。たとえば、0001 0111 1001 は 179 を表します。したがって、パック 10 進数の値はニブルごとに読む必要があります。 値の保管はバイト単位で行い、16 進数表記としてそれらのバイトを読み、それを 10 進数に戻します。たとえば、0001 0111 1001 は、バイナリー表記では 00000001 01111001 となります。 この数値を 16 進数として読むと、0179 になります。

小数点は、位取りによって決まります。 たとえば、DEC(12,5) の列の場合、小数点より右側に 5 桁あることになります。

符号は、桁数を表すニブルの右側のニブルで示します。 正記号または負記号は、以下のように示します。

表 39. パック 10 進数の符号標識の値

	表記		
符号	バイナリー	10 進	16 進
正符号 (+)	1100	12	C
負符号 (-)	1101	13	D

まとめ:

- 値を保管するためには、*p*/2+1 バイトを割り振ります。*p* は精度です。

- 値を表すために、ニブルを左から右へ割り当てます。 数値の精度が偶数の場合は、最初にニブルを追加します。この割り当てには、先行 (無効な) ゼロと後続 (有効な) ゼロの桁が入ります。
- 符号ニブルは、最後のバイトの第 2 ニブルになります。

たとえば、以下のようになります。

列	値	バイトごとにグループにした 16 進のニブル
DEC(8,3)	6574.23	00 65 74 23 0C
DEC(6,2)	-334.02	00 33 40 2D
DEC(7,5)	5.2323	05 23 23 0C
DEC(5,2)	-23.5	02 35 0D

10 進数の SQLLEN フィールド

SQLLEN フィールドには、 10 進数の列の精度 (第 1 バイト) と位取り (第 2 バイト) が入れられます。 アプリケーションを移植可能にするには、精度のバイトと位取りのバイトを短整数として一度に設定するのではなく、個々に設定するようにしてください。これによって、整数のバイト反転の問題が回避されます。

たとえば、C の場合には以下のようにします。

```
((char *)&(sqlda->sqlvar[i].sqllen))[0] = precision;
((char *)&(sqlda->sqlvar[i].sqllen))[1] = scale;
```

関連資料:

- 305 ページの『CHAR』

付録 D. カタログ・ビュー

この付録は、列名とデータ型を含めて、個々のシステム・カタログ・ビューについて説明しています。

システム・カタログ・ビュー

データベース・マネージャーは、基本システム・カタログ表の上に定義される 2 組のシステム・カタログのビューを作成し、保守しています。

- SYSCAT ビューは、SYSCAT スキーマに存在する読み取り専用のカタログ・ビューです。これらのビューに対する SELECT 特権は、デフォルトで PUBLIC に付与されています。
- SYSSTAT ビューは、SYSSTAT スキーマに存在する更新可能なカタログ・ビューです。更新可能なビューには、オプティマイザーで使用される統計情報が入れられます。これらのビューのいくつかの列内の値を変更して、パフォーマンスをテストすることができます。 (統計を変更する前に、RUNSTATS コマンドを呼び出してすべての統計が現行状態を反映するようにすることをお勧めします。) アプリケーションは、基本のカタログ表に対してではなく、 SYSSTAT ビューを対象に作成する必要があります。

すべてのシステム・カタログ・ビューは、データベースの作成時に作成されます。カタログ・ビューは、明示的に作成またはドロップすることはできません。ビューは、SQL データ定義ステートメント、環境ルーチン、および特定のユーティリティーに対応する通常の操作の過程で更新されます。システム・カタログ・ビューのデータは、通常の SQL 照会機能によって使用可能です。システム・カタログ・ビューは (一部の更新可能なカタログ・ビューを除いて)、通常の SQL データ操作コマンドを使用して変更することはできません。

ユーザーの更新可能カタログ・ビューにオブジェクト (表、列、関数、または索引) が現れるのは、そのユーザーがそれを作成した場合か、そのユーザーにそのオブジェクトに対する CONTROL 特権、または明示的な DBADM 権限が与えられている場合だけです。

ビュー内での列の順序はリリースによって変更されることがあります。これによりプログラミング・ロジックが影響を受けないようにするためには、選択リスト内で明示的に列を指定して、SELECT * の使用を避けます。列は、記述するオブジェクトのタイプに基づいて、一貫性のある名前を持ちます。

記述されるオブジェクト	列名
表	TABSCHEMA, TABNAME
索引	INDSCHEMA, INDNAME
ビュー	VIEWSCHEMA, VIEWNAME
制約	CONSTSCHEMA, CONSTNAME
トリガー	TRIGSCHEMA, TRIGNAME

パッケージ	PKGSCHEMA, PKGNAME
タイプ	TYPESCHEMA, TYPENAME, TYPEID
関数	ROUTINESCHEMA, ROUTINENAME, ROUTINEID
メソッド	ROUTINESCHEMA, ROUTINENAME, ROUTINEID
手順	ROUTINESCHEMA, ROUTINENAME, ROUTINEID
列	COLNAME
スキーマ	SCHEMANAME
表スペース	TBSPACE
データベース・パーティション・グループ	DPGNAME
バッファー・プール	BPNAME
イベント・モニター	EVMONNAME
タイム・スタンプの作成	CREATE_TIME

カタログ・ビューのロードマップ

表 40. 読み取り専用のカタログ・ビューのロードマップ

説明	カタログ・ビュー
構造化データ型の属性	556 ページの『SYSCAT.ATTRIBUTES』
データベースに対する権限	577 ページの『SYSCAT.DBAUTH』
データベース・パーティション・グループのバッファー・プールの構成	558 ページの『SYSCAT.BUFFERPOOLS』
データベース・パーティションのバッファー・プールのサイズ	557 ページの『SYSCAT.BUFFERPOOLDBPARTITIONS』
cast 関数	559 ページの『SYSCAT.CASTFUNCTIONS』
チェック制約	560 ページの『SYSCAT.CHECKS』
列の特権	561 ページの『SYSCAT.COLAUTH』
列	569 ページの『SYSCAT.COLUMNS』
チェック制約によって参照される列	562 ページの『SYSCAT.COLCHECKS』
ディメンションで使用される列	573 ページの『SYSCAT.COLUSE』
キーで使用される列	603 ページの『SYSCAT.KEYCOLUSE』
制約の従属関係	574 ページの『SYSCAT.CONSTDEP』
データベース・パーティション・グループのデータベース・パーティション	579 ページの『SYSCAT.DBPARTITIONGROUPDEF』
データベース・パーティション・グループの定義	580 ページの『SYSCAT.DBPARTITIONGROUPS』
データ型	575 ページの『SYSCAT.DATATYPES』
列グループ統計値の詳細	564 ページの『SYSCAT.COLGROUPDIST』
	565 ページの『SYSCAT.COLGROUPDISTCOUNTS』
	566 ページの『SYSCAT.COLGROUPS』
列オプションの詳細	568 ページの『SYSCAT.COLOPTIONS』
列統計値の詳細	563 ページの『SYSCAT.COLDIST』
イベント・モニターの定義	581 ページの『SYSCAT.EVENTMONITORS』
現在モニター中のイベント	583 ページの『SYSCAT.EVENTS』
	584 ページの『SYSCAT.EVENTTABLES』
関数の従属関係 注 1	621 ページの『SYSCAT.ROUTINEDEP』
関数マッピング	588 ページの『SYSCAT.FUNCMAPPINGS』
関数マッピングのオプション	586 ページの『SYSCAT.FUNCMAPOPTIONS』
関数パラメーター・マッピングのオプション	587 ページの『SYSCAT.FUNCMAPPARMOPTIONS』
関数パラメーター 注 1	622 ページの『SYSCAT.ROUTINEPARMS』
関数 注 1	624 ページの『SYSCAT.ROUTINES』
階層 (タイプ、表、ビュー)	589 ページの『SYSCAT.HIERARCHIES』
	585 ページの『SYSCAT.FULLHIERARCHIES』

表 40. 読み取り専用のカタログ・ビューのロードマップ (続き)

説明	カタログ・ビュー
ID 列	567 ページの『SYSCAT.COLIDENTATTRIBUTES』
索引列	591 ページの『SYSCAT.INDEXCOLUSE』
索引の従属関係	592 ページの『SYSCAT.INDEXDEP』
索引活用	597 ページの『SYSCAT.INDEXEXPLOITRULES』
索引拡張従属関係	598 ページの『SYSCAT.INDEXEXTENSIONDEP』
索引拡張パラメーター	600 ページの『SYSCAT.INDEXEXTENSIONPARMS』
索引拡張検索メソッド	599 ページの『SYSCAT.INDEXEXTENSIONMETHODS』
索引拡張	601 ページの『SYSCAT.INDEXEXTENSIONS』
索引オプション	602 ページの『SYSCAT.INDEXOPTIONS』
索引特権	590 ページの『SYSCAT.INDEXAUTH』
索引	593 ページの『SYSCAT.INDEXES』
メソッドの従属関係 注 1	621 ページの『SYSCAT.ROUTINEDEP』
メソッド・パラメーター 注 1	624 ページの『SYSCAT.ROUTINES』
メソッド 注 1	624 ページの『SYSCAT.ROUTINES』
オブジェクト・マッピング	604 ページの『SYSCAT.NAMEMAPPINGS』
パッケージの従属関係	606 ページの『SYSCAT.PACKAGEDEP』
パッケージ特権	605 ページの『SYSCAT.PACKAGEAUTH』
パッケージ	607 ページの『SYSCAT.PACKAGES』
区分化マップ	612 ページの『SYSCAT.PARTITIONMAPS』
パススルー特権	613 ページの『SYSCAT.PASSTHRUAUTH』
述部指定	614 ページの『SYSCAT.PREDICATESPECS』
プロシージャーのオプション	615 ページの『SYSCAT.PROCOPTIONS』
プロシージャーのパラメーター・オプション	616 ページの『SYSCAT.PROCPARMOPTIONS』
プロシージャー・パラメーター 注 1	622 ページの『SYSCAT.ROUTINEPARMS』
プロシージャー 注 1	624 ページの『SYSCAT.ROUTINES』
DB2 Universal Database for z/OS and OS/390 の互換性を提供する	555 ページの『SYSIBM.SYSDUMMY1』
参照制約	617 ページの『SYSCAT.REFERENCES』
リモート表オプション	646 ページの『SYSCAT.TABOPTIONS』
リバース・データ型マッピング	618 ページの『SYSCAT.REVTYPEMAPPINGS』
ルーチンの従属関係	621 ページの『SYSCAT.ROUTINEDEP』
ルーチン・パラメーター 注 1	622 ページの『SYSCAT.ROUTINEPARMS』
ルーチン特権	620 ページの『SYSCAT.ROUTINEAUTH』

表 40. 読み取り専用のカタログ・ビューのロードマップ (続き)

説明	カタログ・ビュー
ルーチン 注 1	624 ページの『SYSCAT.ROUTINES』
スキーマ特権	630 ページの『SYSCAT.SCHEMAAUTH』
スキーマ	631 ページの『SYSCAT.SCHEMATA』
シーケンス特権	632 ページの『SYSCAT.SEQUENCEAUTH』
シーケンス	633 ページの『SYSCAT.SEQUENCES』
サーバー・オプション	634 ページの『SYSCAT.SERVEROPTIONS』
サーバー・オプションの値	653 ページの『SYSCAT.USEROPTIONS』
パッケージ中のステートメント	636 ページの『SYSCAT.STATEMENTS』
ストアード・プロシージャー	624 ページの『SYSCAT.ROUTINES』
システム・サーバー	635 ページの『SYSCAT.SERVERS』
表の制約	639 ページの『SYSCAT.TABCONST』
表の従属関係	640 ページの『SYSCAT.TABDEP』
表の特権	637 ページの『SYSCAT.TABAUTH』
表スペースの使用特権	647 ページの『SYSCAT.TBSPACEAUTH』
表スペース	645 ページの『SYSCAT.TABLESPACES』
表	641 ページの『SYSCAT.TABLES』
トランスフォーム	648 ページの『SYSCAT.TRANSFORMS』
トリガーの従属関係	649 ページの『SYSCAT.TRIGDEP』
トリガー	650 ページの『SYSCAT.TRIGGERS』
タイプ・マッピング	651 ページの『SYSCAT.TYPEMAPPINGS』
ユーザー定義関数	624 ページの『SYSCAT.ROUTINES』
ビュー	641 ページの『SYSCAT.TABLES』
	654 ページの『SYSCAT.VIEWS』
ラッパー・オプション	655 ページの『SYSCAT.WRAPOPTIONS』
ラッパー	656 ページの『SYSCAT.WRAPPERS』

注 1 DB2 バージョン 7.1 以前の表、メソッド、およびプロシージャーのカタログ・ビューがまだ存在します。しかし、これらのビューは DB2 バージョン 7.1 以降の変更を反映していません。そのようなビューを以下に示します。

```
Functions:  SYSCAT.FUNCTIONS, SYSCAT.FUNCDEP, SYSCAT.FUNCPARMS
Methods:    SYSCAT.FUNCTIONS, SYSCAT.FUNCDEP, SYSCAT.FUNCPARMS
Procedures: SYSCAT.PROCEDURES, SYSCAT.PROCPARMS
```

表 41. 更新可能なカタログ・ビューのロードマップ

Description	カタログ・ビュー
列	659 ページの『SYSSTAT.COLUMNS』
列統計値の詳細	657 ページの『SYSSTAT.COLDIST』
索引	662 ページの『SYSSTAT.INDEXES』
ルーチン 注 1	666 ページの『SYSSTAT.ROUTINES』
表	668 ページの『SYSSTAT.TABLES』

表 41. 更新可能なカタログ・ビューのロードマップ (続き)

Description	カタログ・ビュー

注 1 関数およびメソッドの統計を更新するための、 SYSSTAT.FUNCTIONS カタログ・ビューがまだ存在します。しかし、このビューは DB2 バージョン 7.1 以降の変更を反映していません。

SYSIBM.SYSDUMMY1

1 つの行が入っています。このビューは、DB2 Universal Database for z/OS and OS/390 との互換性を必要とするアプリケーションで使用できます。

表 42. SYSCAT.DUMMY1 カタログ・ビュー

列名	データ型	NULL 可能	説明
IBMREQD	CHAR(1)		Y

SYSCAT.ATTRIBUTES

ユーザー定義の構造化データ型に定義されている各属性 (継承された属性の中で適用できるものを含む) ごとに 1 つの行が入ります。

表 43. SYSCAT.ATTRIBUTES カタログ・ビュー

列名	データ型	NULL 可能	説明
TYPESCHEMA	VARCHAR(128)		属性の入った構造化データ型の修飾名。
TYPENAME	VARCHAR(128)		
ATTR_NAME	VARCHAR(128)		属性名。
ATTR_TYPESCHEMA	VARCHAR(128)		属性のタイプの修飾名。
ATTR_TYPENAME	VARCHAR(128)		
TARGET_TYPESCHEMA	VARCHAR(128)	Yes	属性のタイプが REFERENCE の場合、ターゲット・タイプの修飾名。属性のタイプが REFERENCE でない場合は、NULL 値。
TARGET_TYPENAME	VARCHAR(128)	Yes	
SOURCE_TYPESCHEMA	VARCHAR(128)		属性が使われたデータ型階層における、データ型の修飾名。非継承属性の場合、これらの列は TYPESCHEMA および TYPENAME と同じです。
SOURCE_TYPENAME	VARCHAR(128)		
ORDINAL	SMALLINT		構造化データ型の定義における属性の位置 (ゼロから開始する)。
LENGTH	INTEGER		データの最大長。特殊タイプの場合は 0。LENGTH 列は、DECIMAL フィールドに対する精度を示します。
SCALE	SMALLINT		DECIMAL フィールドの位取り。 DECIMAL 以外の場合は 0 になります。
CODEPAGE	SMALLINT		属性のコード・ページ。 FOR BIT DATA 属性を指定して定義されていない文字ストリング属性の場合、値はデータベース・コード・ページです。GRAPHIC ストリング属性の場合、値は (複合) データベース・コード・ページによる暗黙の DBCS コード・ページです。それ以外の場合の値は 0 です。
LOGGED	CHAR(1)		LOB タイプまたは LOB に基づく特殊タイプの属性だけに適用される。それ以外はブランクです。 Y = 属性のログ記録を取る。 N = 属性のログ記録を取らない。
COMPACT	CHAR(1)		LOB タイプまたは LOB に基づく特殊タイプの属性だけに適用される。それ以外はブランクです。 Y = ストレージ内で属性を圧縮する。 N = 属性を圧縮しない。
DL_FEATURES	CHAR(10)		DATALINK タイプ属性だけに適用されます。REFERENCE タイプ属性の場合はブランクで、それ以外は NULL です。リンク・タイプ、制御モード、リカバリー、およびリンク解除特性など、さまざまな DATALINK 機能をエンコードします。

SYSCAT.BUFFERPOOLDBPARTITIONS

データベース・パーティションに対するバッファー・プールのサイズが、SYSCAT.BUFFERPOOLS の列 NPAGES のデフォルトのサイズと異なるバッファー・プール内のデータベース・パーティションごとに、 1 つの行が入っています。

表 44. SYSCAT.BUFFERPOOLDBPARTITIONS カタログ・ビュー

列名	データ型	NULL 可能	説明
BUFFERPOOLID	INTEGER		内部バッファー・プール ID。
DBPARTITIONNUM	SMALLINT		データベース・パーティション番号。
NPAGES	INTEGER		このデータベース・パーティションに対するバッファー・プールのページ数。

SYSCAT.BUFFERPOOLS

データベース・パーティション・グループ内のバッファー・プールごとに 1 つの行が入っています。

表 45. SYSCAT.BUFFERPOOLS カタログ・ビュー

列名	データ型	NULL 可能	説明
BPNAME	VARCHAR(128)		バッファー・プールの名前。
BUFFERPOOLID	INTEGER		内部バッファー・プール ID。
DPGNAME	VARCHAR(128)	Yes	データベース・パーティション・グループ名 (そのバッファー・プールが、データベース内のすべてのデータベース・パーティションに対して存在する場合は NULL)。
NPAGES	INTEGER		バッファー・プールのページ数。
PAGESIZE	INTEGER		このバッファー・プールのページ・サイズ。
ESTORE	CHAR(1)		N = このバッファー・プールは拡張ストレージを使用しない。 Y = このバッファー・プールは拡張ストレージを使用する。

SYSCAT.CASTFUNCTIONS

cast 関数ごとに 1 つの行が入っています。組み込み cast 関数は組み入れられません。

表 46. SYSCAT.CASTFUNCTIONS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
FROM_TYPESCHEMA	VARCHAR(128)		パラメーターのデータ型の修飾名。
FROM_TYPENAME	VARCHAR(128)		
TO_TYPESCHEMA	VARCHAR(128)		キャスト後の結果のデータ型を示す修飾名。
TO_TYPENAME	VARCHAR(128)		
FUNCSCHEMA	VARCHAR(128)		関数の修飾名。
FUNCNAME	VARCHAR(128)		
SPECIFICNAME	VARCHAR(128)		関数インスタンスの名前。
ASSIGN_FUNCTION	CHAR(1)		Y = 暗黙的な割り当て関数 N = 割り当て関数ではない

SYSCAT.CHECKS

チェック制約ごとに 1 つの行が入っています。

表 47. SYSCAT.CHECKS カタログ・ビュー

列名	データ型	NULL 可能	説明
CONSTNAME	VARCHAR(18)		チェック制約の名前 (表内でユニーク)。
DEFINER	VARCHAR(128)		そのチェック制約を定義した許可 ID。
TABSCHEMA	VARCHAR(128)		この制約が適用される表の修飾名。
TABNAME	VARCHAR(128)		
CREATE_TIME	TIMESTAMP		制約が定義された時刻。この制約で使用される関数の解決に使用されます。制約の定義後に作成された関数は選択されません。
QUALIFIER	VARCHAR(128)		オブジェクト定義時のデフォルト・スキーマの値。非修飾参照を完了するために使用します。
TYPE	CHAR(1)		チェック制約のタイプ。 A = GENERATED ALWAYS 列のシステム生成チェック制約 C = チェック制約 F = 関数の従属関係 O = 制約はオブジェクト・プロパティー
FUNC_PATH	VARCHAR(254)		制約作成時に使用された現行 SQL パス。
TEXT	CLOB(64K)		CHECK 文節のテキスト。注 1 を参照。

注:

1. カタログ・ビューでは、CHECK 文節のテキストは常にデータベース・コード・ページ内で示されますが、その中で置換文字を使用することができます。チェック制約は常にターゲット表のコード・ページ内で適用されますが、適用時には置換文字を収容しません。 (チェック制約は、置換文字の入っていない可能性のあるターゲット表のコード・ページ内のオリジナル・テキストに基づいて適用されます。)

SYSCAT.COLAUTH

列レベルの特権が与えられているユーザーまたはグループごとに 1 つまたは複数の行が入っており、特権のタイプとその特権が付与可能か否かを示します。

表 48. SYSCAT.COLAUTH カタログ・ビュー

列名	データ型	NULL 可能	説明
GRANTOR	VARCHAR(128)		特権を付与したユーザーの許可 ID、または SYSIBM。
GRANTEE	VARCHAR(128)		特権を付与されたユーザーまたはグループの許可 ID。
GRANTEETYPE	CHAR(1)		U = GRANTEE は個々のユーザー。 G = GRANTEE はグループ。
TABSCHEMA	VARCHAR(128)		表またはビューの修飾名。
TABNAME	VARCHAR(128)		
COLNAME	VARCHAR(128)		この特権が適用される列の名前。
COLNO	SMALLINT		表またはビューの中のこの列の番号。
PRIVTYPE	CHAR(1)		表またはビューに対する特権のタイプを次のように示します。 U = 更新特権 R = 参照特権
GRANTABLE	CHAR(1)		特権が付与可能か否かを示します。 G = 付与可能 N = 付与不能

SYSCAT.COLCHECKS

各行は、チェック制約によって参照される列を表します。

表 49. SYSCAT.COLCHECKS カタログ・ビュー

列名	データ型	NULL 可能	説明
CONSTNAME	VARCHAR(18)		チェック制約の名前。 (表内でユニーク。システム生成の名前も含む。)
TABSCHEMA	VARCHAR(128)		参照される列の入った表の修飾名。
TABNAME	VARCHAR(128)		
COLNAME	VARCHAR(128)		列の名前。
USAGE	CHAR(1)		D = 関数の従属関係の子列 P = 関数の従属関係の親列 R = 列はチェック制約内で参照されます。 S = 列は生成される列をサポートするシステム生成チェック制約のソース列です。 T = 列は生成される列をサポートするシステム生成チェック制約のターゲット列です。

SYSCAT.COLDIST

オプティマイザーで使用される列の詳細な統計値が入れられます。各行は、列の N 番目の最大頻度を記述しています。

表 50. SYSCAT.COLDIST カタログ・ビュー

列名	データ型	NULL 可能	説明
TABSCHEMA	VARCHAR(128)		この項目が適用される表の修飾名。
TABNAME	VARCHAR(128)		
COLNAME	VARCHAR(128)		この項目が適用される列の名前。
TYPE	CHAR(1)		F = 頻度 (最大頻度) Q = 変位値
SEQNO	SMALLINT		• TYPE=F の場合、この列の N は N 番目の最大頻度。 • TYPE=Q の場合、この列の N は N 番目の変位値。
COLVALUE	VARCHAR(254)	Yes	データ値 (文字リテラルまたは NULL 値)。注 1 を参照。
VALCOUNT	BIGINT		• TYPE=F の場合、VALCOUNT は、その列の中の COLVALUE の出現回数。 • TYPE=Q の場合、VALCOUNT は、値が COLVALUE 以下の行の数。
DISTCOUNT	BIGINT	Yes	TYPE=Q の場合、この列は COLVALUE 以下の値の種類数 (入手不能の場合は NULL) を記録します。

注:

1. カタログ・ビュー内の COLVALUE の値は常にデータベース・コード・ページ中に示されますが、これには置換文字を収めることができます。ただし、列の表のコード・ページ内で内部的に統計が収集されるので、照会の最適化時に適用するときは実際の列値が使用されます。

SYSCAT.COLGROUPDIST

列グループ中の *n* 番目に高い頻度の値または列グループ中の *n* 番目の変位を構成する列グループ内の各値ごと 1 つずつ行を収容します。

表 51. SYSCAT.COLGROUPDIST カタログ・ビュー

列名	データ型	NULL 可能	説明
COLGROUPID	INTEGER		列グループの内部 ID。
TYPE	CHAR(1)		F = 頻度 Q = 変位値
ORDINAL	SMALLINT		グループ内の列の順序数。
SEQNO	SMALLINT		*n* 番目の TYPE 値を表すシーケンス番号 *n*。
COLVALUE	VARCHAR(254)	Yes	データ値 (文字リテラルまたは NULL 値)。

SYSCAT.COLGROUPDISTCOUNTS

列グループ中の *n* 番目に高い頻度の値または列グループ中の *n* 番目の変位に適用される分散統計の行を収容します。

表 52. SYSCAT.COLGROUPDISTCOUNTS カタログ・ビュー

列名	データ型	NULL 可能	説明
COLGROUPID	INTEGER		列グループの内部 ID。
TYPE	CHAR(1)		F = 頻度 Q = 変位値
SEQNO	SMALLINT		*n* 番目の TYPE 値を表すシーケンス番号 *n*。
VALCOUNT	BIGINT		TYPE=F の場合、VALCOUNT は、この SEQNO を持つ列グループの中の COLVALUE の出現回数です。 TYPE=Q の場合、VALCOUNT は、値が この SEQNO を持つ列グループの COLVALUE 以下の行の数です。
DISTCOUNT	BIGINT	Yes	TYPE=Q の場合、この列はこの SEQNO を持つ列グループの COLVALUE 以下の値の種類数 (入手不能の場合は NULL) を記録します。

SYSCAT.COLGROUPS

すべての列グループに対する行、および列グループ全体に適用される統計が入っています。

表 53. SYSCAT.COLGROUPS カタログ・ビュー

列名	データ型	NULL 可能	説明
COLGROUPSCHEMA	VARCHAR(128)		列グループの修飾名。
COLGROUPNAME	VARCHAR(128)		
COLGROUPID	INTEGER		列グループの内部 ID。
COLGROUPCARD	BIGINT		列グループのカーディナリティー。
NUMFREQ_VALUES	SMALLINT		列グループに関して収集された頻度の数。
NUMQUANTILES	SMALLINT		列グループに関して収集された変位の数。

SYSCAT.COLIDENTATTRIBUTES

表に定義されている各 ID 列ごとに 1 つの行が入ります。

表 54. SYSCAT.COLIDENTATTRIBUTES カタログ・ビューの列

列名	データ型	NULL 可能	説明
TABSCHEMA	VARCHAR(128)		この列を収めた表またはビューの修飾名。
TABNAME	VARCHAR(128)		
COLNAME	VARCHAR(128)		列名。
START	DECIMAL(31,0)		開始値
INCREMENT	DECIMAL(31,0)		増分値
MINVALUE	DECIMAL(31,0)		最小値
MAXVALUE	DECIMAL(31,0)		最大値。
CYCLE	CHAR(1)		境界に達したときにサイクリングを行うかどうか。 Y - サイクリングを行う N - サイクリングを行わない
CACHE	INTEGER		アクセスを高速化するために、メモリーに事前割り当てするシーケンス値の数。 0 は値が事前割り当てされないことを示します。
SEQID	INTEGER		シーケンスの内部 ID。

SYSCAT.COLOPTIONS

各行には、列固有のオプション値が入ります。

表 55. SYSCAT.COLOPTIONS カタログ・ビュー

列名	データ型	NULL 可能	説明
TABSCHEMA	VARCHAR(128)		列の修飾されたニックネーム。
TABNAME	VARCHAR(128)		
COLNAME	VARCHAR(128)		ローカル列名。
OPTION	VARCHAR(128)		列オプションの名前。
SETTING	VARCHAR(255)		列オプションの値。

SYSCAT.COLUMNS

表またはビューで定義されている列 (該当する場合は継承された列も含む) ごとに 1 行が入っています。どのカタログ・ビューにも、SYSCAT.COLUMNS 表内に項目があります。

表 56. SYSCAT.COLUMNS カタログ・ビュー

列名	データ型	NULL 可能	説明
TABSCHEMA	VARCHAR(128)		この列の入った表またはビューの修飾名。
TABNAME	VARCHAR(128)		
COLNAME	VARCHAR(128)		列名。
COLNO	SMALLINT		表またはビューにおけるその列の位置を示す番号。番号は、0 から始まります。
TYPESCHEMA	VARCHAR(128)		列のデータ型が特殊型の場合、その修飾名が入っています。それ以外の場合、TYPESCHEMA には値 SYSIBM が入り、 TYPENAME にはその列のデータ型 (CHARACTER などの長い形式で) が入ります。 FLOAT または *n* が 24 より大きい FLOAT(*n*) が指定された場合、 TYPENAME は DOUBLE に名前変更されます。 *n* が 25 より小さい FLOAT(*n*) が指定された場合は、 TYPENAME は REAL に名前変更されます。また、NUMERIC は DECIMAL に名前変更されます。
TYPENAME	VARCHAR(18)		
LENGTH	INTEGER		データの最大長。特殊タイプの場合は 0。LENGTH 列は、DECIMAL フィールドに対する精度を示します。
SCALE	SMALLINT		DECIMAL フィールドの位取り。 DECIMAL 以外の場合は 0 になります。
DEFAULT	VARCHAR(254)	Yes	列のデータ型に適した定数、特殊レジスター、または cast 関数で表された表の列のデフォルト値。キーワード NULL の場合もあります。 値は、デフォルト値として指定された値から変換されることがあります。たとえば、日時の定数は ISO フォーマットで表示され、cast 関数名はスキーマ名で修飾され、ID は区切られています (注 3 を参照)。 DEFAULT 文節が指定されていないか、または列がビューの列の場合は、NULL 値になります。

表 56. SYSCAT.COLUMNS カタログ・ビュー (続き)

列名	データ型	NULL 可能	説明
NULLS	CHAR(1)		Y = 列は NULL 可能。 N = 列は NULL 不可。 式または関数から派生したビューの列の場合、値は N である可能性があります。それでも、このビューを使用するステートメントが処理され、算術計算エラーの警告を出された場合は、この列に NULL 値が入ります。 注 1 を参照。
CODEPAGE	SMALLINT		列のコード・ページ。 FOR BIT DATA 属性を指定して定義されていない文字ストリング列の場合、値はデータベース・コード・ページです。 GRAPHIC ストリング列の場合、値は (複合) データベース・コード・ページによる暗黙の DBCS コード・ページです。それ以外の場合の値は 0 です。
LOGGED	CHAR(1)		LOB タイプまたは LOB に基づく特殊タイプの列だけに適用されます (それ以外はブランク)。 Y = 列のログ記録を取る。 N = 列のログ記録を取らない。
COMPACT	CHAR(1)		LOB タイプまたは LOB に基づく特殊タイプの列だけに適用されます (それ以外はブランク)。 Y = ストレージ内で列を圧縮する。 N = 列を圧縮しない。
COLCARD	BIGINT		列の特殊値の数。統計が収集されていない場合は -1。継承列および階層表の列の場合は -2。
HIGH2KEY	VARCHAR(254)	Yes	列の 2 番目の最高値。統計が収集されていない場合や、継承列および階層表の列の場合、このフィールドは空です。注 2 および 4 を参照。
LOW2KEY	VARCHAR(254)	Yes	列の 2 番目の最低値。統計が収集されていない場合や、継承列および階層表の列の場合、このフィールドは空です。注 2 および 4 を参照。
AVGCOLLEN	INTEGER		列の長さに必要な平均のスペース。長形式フィールドまたは LOB の場合、または統計が収集されていない場合は -1。継承列および階層表の列の場合は -2。
KEYSEQ	SMALLINT	Yes	表の主キー内の列の位置番号。副表および階層表の場合、このフィールドは NULL 値です。
PARTKEYSEQ	SMALLINT	Yes	表の区分化キー内の列の位置番号。列が区分化キーの一部でない場合、このフィールドは NULL 値または 0。副表および階層表の場合も、このフィールドは NULL 値です。
NQUANTILES	SMALLINT		この列の SYSCAT.COLDIST に記録された変位値の数。統計がない場合は -1。継承列および階層表の列の場合は -2。

表 56. SYSCAT.COLUMNS カタログ・ビュー (続き)

列名	データ型	NULL 可能	説明
NMOSTFREQ	SMALLINT		この列の SYSCAT.COLDIST に記録された最大頻度の数。統計がない場合は -1。継承列および階層表の列の場合は -2。
NUMNULLS	BIGINT		列の NULL 値の数が入っています。統計が収集されていない場合は -1。
TARGET_TYPESCHEMA	VARCHAR(128)	Yes	列の型が REFERENCE の場合、ターゲット・タイプの修飾名。列の型が REFERENCE でない場合は、NULL 値。
TARGET_TYPENAME	VARCHAR(18)	Yes	
SCOPE_TABSCHEMA	VARCHAR(128)	Yes	列の型が REFERENCE の場合、有効範囲 (ターゲット表) の修飾名。列の型が REFERENCE でない場合や、有効範囲が定義されていない場合は、NULL 値。
SCOPE_TABNAME	VARCHAR(128)	Yes	
SOURCE_TABSCHEMA	VARCHAR(128)		列が使われたそれぞれの表階層における、表またはビューの修飾名。非継承列の場合、値は TBCREATOR および TBNAME と同じです。非タイプ表およびビューの列の場合は NULL 値。
SOURCE_TABNAME	VARCHAR(128)		
DL_FEATURES	CHAR(10)	Yes	DATALINK 型の列だけに適用されます。それ以外の場合、NULL 値です。各文字の位置は、次のようにして定義されます。 1. リンク・タイプ (URL の場合は U) 2. リンク制御 (ファイルの場合は F、ファイルでない場合は N) 3. 保全性 (すべての場合は A、なしの場合は N) 4. 読み取り許可 (ファイル・システムの場合は F、データベースの場合は D) 5. 書き込み許可 (ファイル・システムの場合は F、ブロック済みの場合は B、更新のトークンが必要な admin の場合は A、更新のトークンが必要ない admin の場合は A) 6. リカバリー (する場合は Y、しない場合は N) 7. リンク解除時の処理 (リストアの場合は R、削除の場合は D、適用しない場合は N) 文字 8 から 10 は、将来の使用のために予約されています。
SPECIAL_PROPS	CHAR(8)	Yes	REFERENCE 型の列のみに適用されます。それ以外の場合、NULL 値です。各文字の位置は、次のようにして定義されます。 オブジェクト ID (OID) 列 (yes の場合は Y、no の場合は N) ユーザー生成またはシステム生成 (ユーザーの場合は U、システムの場合は S)

表 56. SYSCAT.COLUMNS カタログ・ビュー (続き)

列名	データ型	NULL 可能	説明
HIDDEN	CHAR(1)		隠し列のタイプ。 S = システムに管理される隠し列。 列が隠されていない場合はブランク。
INLINE_LENGTH	INTEGER		基本表の行と一緒に保持できる構造化タイプの列の長さ。 ALTER/CREATE TABLE ステートメントで明示的に設定された値がない場合は 0 です。
IDENTITY	CHAR(1)		'Y' はその列が ID 列であることを示し、'N' はその列が ID 列ではないことを示しています。
GENERATED	CHAR(1)		生成される列の型。 A = 列の値が常に生成される。 D = 列の値がデフォルトで生成される。 列が生成されない場合はブランク。
COMPRESS	CHAR(1)		S = システム・デフォルト値を圧縮する。 O = 圧縮をオフにする。
TEXT	CLOB(64K)		キーワード AS で始まる、生成される列のテキストが入ります。
REMARKS	VARCHAR(254)	Yes	ユーザー提供のコメント。
AVGDISTINCTPERPAGE	DOUBLE	Yes	将来の利用。
PAGEVARIANCERATIO	DOUBLE	Yes	将来の利用。
SUB_COUNT	SMALLINT		サブエレメントの平均数。文字カラムの場合のみ適用可能。たとえば、 'database simulation analytical business intelligence' というストリングについて考えてみます。この例では、ストリング内に 5 個のサブエレメントがあるため、 SUB_COUNT = 5 です。
SUB_DELIM_LENGTH	SMALLINT		各サブエレメントを区切っている各区切り文字の平均の長さ。文字カラムの場合のみ適用可能。たとえば、 'database simulation analytical business intelligence' というストリングについて考えてみます。この例では、各区切り文字は単一ブランクなので、 SUB_DELIM_LENGTH = 1 です。

注:

1. バージョン 2 以降では、値 D (デフォルト値が NULL でないことを示す) は使用されなくなりました。代わりに、WITH DEFAULT の使用は、 DEFAULT 欄の値が NULL 以外であることによって示されます。
2. バージョン 2 以降では、数値データの表現が文字リテラルに変更されました。サイズが 16 バイトから 33 バイトに拡大されました。
3. バージョン 2.1.0 では cast-function 名は区切られていなかったため、 DEFAULT 列にこのように表示される場合があります。また、ビュー列にはデフォルト値が入っていたため、これも DEFAULT 列に表示されます。
4. カタログ・ビューでは、HIGH2KEY と LOW2KEY の値は常にデータベース・コード・ページ中に示され、これには置換文字を収めることができます。ただし、列の表のコード・ページ内で内部的に統計が収集されるので、照会の最適化時に適用するときは実際の列値が使用されます。

SYSCAT.COLUSE

CREATE TABLE ステートメントの DIMENSIONS 文節に関与する各列ごとに 1 行ずつが入っています。

表 57. SYSCAT.COLUSE カタログ・ビュー

列名	データ型	NULL 可能	説明
TABSCHEMA	VARCHAR(128)		列の入った表の修飾名
TABNAME	VARCHAR(128)		
COLNAME	VARCHAR(128)		列の名前
DIMENSION	SMALLINT		DIMENSIONS 節に指定されたディメンション順序に基づくディメンション番号 (最初の位置 = 0)。複合ディメンションでは、この値はディメンションのどのコンポーネントでも同じです。
COLSEQ	SMALLINT		列が属するディメンション内での列の位置番号 (最初の位置 = 0)。複合ではないディメンションの単一列では、この値は 0 です。
TYPE	CHAR(1)		ディメンションのタイプ。 C = クラスタリング/マルチディメンション・クラスタリング (MDC)

SYSCAT.CONSTDEP

他の特定のオブジェクトへの制約の従属関係ごとに行が 1 つずつ入っています。

表 58. SYSCAT.CONSTDEP カタログ・ビュー

列名	データ型	NULL 可能	説明
CONSTNAME	VARCHAR(18)		制約の名前。
TABSCHEMA	VARCHAR(128)		制約が適用される表の修飾名。
TABNAME	VARCHAR(128)		
BTYPE	CHAR(1)		この制約が従属しているオブジェクトのタイプ。可能な値: F = 関数インスタンス I = 索引インスタンス R = 構造化タイプ
BSCHEMA	VARCHAR(128)		この制約が依存するオブジェクトの修飾名。
BNAME	VARCHAR(18)		

SYSCAT.DATATYPES

組み込み型およびユーザー定義型をはじめとするすべてのデータ型ごとに行が 1 つずつ入っています。

表 59. SYSCAT.DATATYPES カタログ・ビュー

列名	データ型	NULL 可能	説明
TYPESCHEMA	VARCHAR(128)		データ型の修飾名 (組み込み型の場合、TYPESCHEMA は SYSIBM です)。
TYPENAME	VARCHAR(18)		
DEFINER	VARCHAR(128)		タイプ作成時の許可 ID。
SOURCESCHEMA	VARCHAR(128)	Yes	特殊タイプのソース・タイプの修飾名。構造化タイプの参照表現として使用される参照型として使用される組み込み型の修飾名。その他の場合は NULL 値。
SOURCENAME	VARCHAR(18)	Yes	
METATYPE	CHAR(1)		S = システムで定義済みのタイプ T = 特殊タイプ R = 構造化タイプ
TYPEID	SMALLINT		システムで生成されたデータ型の内部 ID。
SOURCETYPEID	SMALLINT	Yes	ソース・タイプの内部タイプ ID (組み込みタイプの場合は NULL 値)。ユーザー定義構造化タイプの場合、これは参照表示タイプの内部タイプ ID になります。
LENGTH	INTEGER		タイプの最大長。システムで定義済みのパラメーター化タイプ (DECIMAL や VARCHAR など) の場合は 0。ユーザー定義構造化タイプの場合、これは参照表示タイプの長さを示します。
SCALE	SMALLINT		システムで定義済みの DECIMAL タイプに基づく特殊タイプまたは参照表示タイプの位取り。他のすべてのタイプの場合は 0 (DECIMAL 自体を含む)。ユーザー定義構造化タイプの場合、これは参照表示タイプの長さを示します。
CODEPAGE	SMALLINT		文字特殊タイプおよび GRAPHIC 特殊タイプ、または参照表示タイプの場合はコード・ページ。それ以外の場合は 0。
CREATE_TIME	TIMESTAMP		データ型の作成時刻。
ATTRCOUNT	SMALLINT		データ型内の属性の数。
INSTANTIABLE	CHAR(1)		Y = タイプをインスタンス化できる。 N = タイプをインスタンス化できない。
WITH_FUNC_ACCESS	CHAR(1)		Y = 関数表記を使ってこのタイプ用のメソッドすべてを呼び出せる。 N = 関数表記を使ってこのタイプ用のメソッドすべてを呼び出せない。

表 59. SYSCAT.DATATYPES カタログ・ビュー (続き)

列名	データ型	NULL 可能	説明
FINAL	CHAR(1)		Y = ユーザー定義タイプにサブタイプを指定できない。 N = ユーザー定義タイプにサブタイプを指定できる。
INLINE_LENGTH	INTEGER		基本表の行で保持できる構造化タイプの長さ。CREATE TYPE ステートメントで明示的に設定された値がない場合は 0 です。
NATURAL_INLINE_LENGTH	INTEGER		システムによって計算された、構造化タイプのインラインの長さ。
REMARKS	VARCHAR(254)	Yes	ユーザー提供のコメントまたは NULL 値。

SYSCAT.DBAUTH

ユーザーが持つデータベース権限を記録します。

表 60. SYSCAT.DBAUTH カタログ・ビュー

列名	データ型	NULL 可能	説明
GRANTOR	VARCHAR(128)		SYSIBM または特権を付与したユーザーの許可 ID。
GRANTEE	VARCHAR(128)		特権を付与されたユーザーまたはグループの許可 ID。
GRANTEETYPE	CHAR(1)		U = GRANTEE は個々のユーザー。 G = GRANTEE はグループ。
DBADMAUTH	CHAR(1)		GRANTEE がデータベースに対する DBADM 権限を持っているか否か。 Y = 権限を持っている。 N = 権限を持っていない。
CREATETABAUTH	CHAR(1)		GRANTEE がデータベースに表を作成できるか否か (CREATETAB)。 Y = 特権がある。 N = 特権がない。
BINDADDAUTH	CHAR(1)		GRANTEE がデータベース中に新しいパッケージを作成できるか否か (BINDADD)。 Y = 特権がある。 N = 特権がない。
CONNECTAUTH	CHAR(1)		GRANTEE がデータベースに接続できるか否か (CONNECT)。 Y = 特権がある。 N = 特権がない。
NOFENCEAUTH	CHAR(1)		GRANTEE に fenced でない関数を作成する特権があるか否か。 Y = 特権がある。 N = 特権がない。
IMPLSCHEMAAUTH	CHAR(1)		GRANTEE がデータベース内に暗黙的にスキーマを作成できるか否か (IMPLICIT_SCHEMA)。 Y = 特権がある。 N = 特権がない。
LOADAUTH	CHAR(1)		GRANTEE がデータベースに対する LOAD 権限を持っているか否か。 Y = 権限を持っている。 N = 権限を持っていない。

表 60. SYSCAT.DBAUTH カタログ・ビュー (続き)

列名	データ型	NULL 可能	説明
EXTERNALROUTINEAUTH	CHAR(1)		GRANTEE が外部ルーチンを作成できるか否か (CREATE_EXTERNAL_ROUTINE)。 Y = 権限を持っている。 N = 権限を持っていない。
QUIESCECONNECTAUTH	CHAR(1)		GRANTEE がデータベースに接続できるか否か (QUIESCE_CONNECT)。 Y = 権限を持っている。 N = 権限を持っていない。

SYSCAT.DBPARTITIONGROUPDEF

データベース・パーティション・グループに入っているパーティションごとに 1 つの行が入っています。

表 61. SYSCAT.DBPARTITIONGROUPDEF カタログ・ビュー

列名	データ型	NULL 可能	説明
DBPGNAME	VARCHAR(18)		データベース・パーティションの入ったデータベース・パーティション・グループの名前。
DBPARTITIONNUM	SMALLINT		データベース・パーティション・グループに属するパーティションのパーティション番号。有効なパーティション番号は、0 以上 999 以下です。
IN_USE	CHAR(1)		データベース・パーティションの状況。 A = 新しく追加されたパーティションは区分化マップに入っていないが、データベース・パーティション・グループ内の表スペースにコンテナーが作成された。パーティションは、データベース・パーティション・グループ再分散操作が正常に完了した場合に、区分化マップに追加されます。 D = パーティションは、データベース・パーティション・グループ再分散操作の完了時にドロップされる。 T = 新しく追加されたパーティションは、区分化マップに入っておらず、WITHOUT TABLESPACES 文節を使用して追加された。このデータベース・パーティション・グループの表スペースに、コンテナーを明示的に追加する必要があります。 Y = パーティションは区分化マップに入っている。

SYSCAT.DBPARTITIONGROUPS

データベース・パーティション・グループごとに 1 つの行が入ります。

表 62. SYSCAT.DBPARTITIONGROUPS カタログ・ビュー

列名	データ型	NULL 可能	説明
DBPGNAME	VARCHAR(18)		データベース・パーティション・グループの名前。
DEFINER	VARCHAR(128)		データベース・パーティション・グループの定義者の許可 ID。
PMAP_ID	SMALLINT		SYSCAT.PARTITIONMAPS 内の区分化マップの ID。
REDISTRIBUTE_PMAP_ID	SMALLINT		再分散用に現在使用されている区分化マップの ID。再分散が現在進行中でない場合は、値は -1。
CREATE_TIME	TIMESTAMP		データベース・パーティション・グループの作成時刻。
REMARKS	VARCHAR(254)	Yes	ユーザーが入力したコメント。

SYSCAT.EVENTMONITORS

定義されているイベント・モニターごとに 1 つの行が入ります。

表 63. SYSCAT.EVENTMONITORS カタログ・ビュー

列名	データ型	NULL 可能	説明
EVMONNAME	VARCHAR(18)		イベント・モニターの名前。
DEFINER	VARCHAR(128)		イベント・モニターの定義者の許可 ID。
TARGET_TYPE	CHAR(1)		イベント・データの書き込み先 (ターゲット) のタイプ。値: F = ファイル P = パイプ T = 表
TARGET	VARCHAR(246)		イベント・データの書き込み先 (ターゲット) の名前。ファイルの絶対パス名、またはパイプの絶対名。
MAXFILES	INTEGER	Yes	このイベント・モニターの 1 つのイベント・パスで許されるイベント・ファイルの最大数。最大値がない場合、または書き込み先が FILE でない場合は NULL 値。
MAXFILESIZE	INTEGER	Yes	各イベント・ファイルの最大サイズ (4K ページ単位)。このサイズに達すると、イベント・モニターは新しいファイルを作成します。最大値がない場合、または書き込み先が FILE でない場合は NULL 値。
BUFFERSIZE	INTEGER	Yes	ファイルに出力する場合、イベント・モニターの使用するバッファー・サイズ (4K ページ単位)。それ以外の場合は NULL 値。
IO_MODE	CHAR(1)	Yes	ファイル入出力のモード。 B = ブロック化 N = 非ブロック化 宛先タイプが FILE でない場合は NULL 値。
WRITE_MODE	CHAR(1)	Yes	このイベント・モニターの起動時に、モニターが既存のイベント・データを処理する方法。値: A = 追加 R = 置換 宛先タイプが FILE でない場合は NULL 値。
AUTOSTART	CHAR(1)		データベース開始時にイベント・モニターが自動的に活動化されるか否か。 Y = Yes N = No
DBPARTITIONNUM	SMALLINT		イベント・モニターが稼働してイベントをログ記録するデータベース・パーティションの番号。

表 63. SYSCAT.EVENTMONITORS カタログ・ビュー (続き)

列名	データ型	NULL 可能	説明
MONSCOPE	CHAR(1)		モニターの有効範囲: L = ローカル G = グローバル T = 表スペースが存在するデータベース・パーティションごと ブランク文字 (WRITE TO TABLE イベント・モニターの場合にのみ有効)
EVMON_ACTIVATES	INTEGER		このイベント・モニターが活動化された回数
REMARKS	VARCHAR(254)	Yes	将来の使用のために予約済み。

SYSCAT.EVENTS

モニターするイベントごとに 1 つの行が入ります。一般に、1 つのイベント・モニターは複数のイベントをモニターします。

表 64. SYSCAT.EVENTS カタログ・ビュー

列名	データ型	NULL 可能	説明
EVMONNAME	VARCHAR(18)		このイベントをモニターするイベント・モニターの名前。
TYPE	VARCHAR(18)		モニターされるイベントのタイプ。可能な値: DATABASE CONNECTIONS TABLES STATEMENTS TRANSACTIONS DEADLOCKS DETAILDEADLOCKS TABLESPACES
FILTER	CLOB(32K)	Yes	このイベントに適用される WHERE 文節のテキスト全体。

SYSCAT.EVENTTABLES

SQL 表に書き込むイベント・モニターのターゲット表ごとに 1 つの行が入っています。

表 65. SYSCAT.EVENTTABLES カタログ・ビュー

列名	データ型	NULL 可能	説明
EVMONNAME	VARCHAR(128)		イベント・モニターの名前。
LOGICAL_GROUP	VARCHAR(18)		論理データ・グループの名前。これは、以下のいずれかになります。 BUFFERPOOL CONN CONNHEADER CONTROL DB DEADLOCK DLCONN DLLOCK STMT SUBSECTION TABLE TABLESPACE XACT
TABSCHEMA	VARCHAR(128)		ターゲット表の修飾名。
TABNAME	VARCHAR(128)		
PCTDEACTIVATE	SMALLINT		イベント・モニターは、DMS 表スペースがこの比率の値を超えると、自動的に非活動化されます。 SMS 表スペースには、100 に設定します。

SYSCAT.FULLHIERARCHIES

各行は、副表とスーパー表、サブタイプとスーパータイプ、またはサブビューとスーパービューの関係を表しています。このビューには、直接の関係をはじめとする、すべての階層関係が入っています。

表 66. SYSCAT.FULLHIERARCHIES カタログ・ビュー

列名	データ型	NULL 可能	説明
METATYPE	CHAR(1)		次のように関係のタイプをエンコードします。 R = 構造化タイプの相互関係 U = タイプ表の相互関係 W = タイプ・ビューの相互関係
SUB_SCHEMA	VARCHAR(128)		サブタイプ、副表、またはサブビューの修飾名。
SUB_NAME	VARCHAR(128)		
SUPER_SCHEMA	VARCHAR(128)		スーパータイプ、スーパー表、またはスーパービューの修飾名。
SUPER_NAME	VARCHAR(128)		
ROOT_SCHEMA	VARCHAR(128)		階層のルートにある表、ビュー、またはタイプの修飾名。
ROOT_NAME	VARCHAR(128)		

SYSCAT.FUNCMAPOPTIONS

各行には、関数マッピングのオプション値が入っています。

表 67. SYSCAT.FUNCMAPOPTIONS カタログ・ビュー

列名	データ型	NULL 可能	説明
FUNCTION_MAPPING	VARCHAR(18)		関数マッピングの名前。
OPTION	VARCHAR(128)		関数マッピングのオプション名。
SETTING	VARCHAR(255)		関数マッピングのオプションの値。

SYSCAT.FUNCMAPPARMOPTIONS

各行には、関数マッピングのパラメーター・オプションの値が入っています。

表 68. SYSCAT.FUNCMAPPARMOPTIONS カタログ・ビュー

列名	データ型	NULL 可能	説明
FUNCTION_MAPPING	VARCHAR(18)		関数マッピングの名前。
ORDINAL	SMALLINT		パラメーターの位置。
LOCATION	CHAR(1)		L = ローカル R = リモート
OPTION	VARCHAR(128)		関数マッピングのパラメーター・オプションの名前。
SETTING	VARCHAR(255)		関数マッピングのパラメーター・オプションの値。

SYSCAT.FUNCMAPPINGS

各行には関数マッピングが入っています。

表 69. SYSCAT.FUNCMAPPINGS カタログ・ビュー

列名	データ型	NULL 可能	説明
FUNCTION_ MAPPING	VARCHAR(128)		関数マッピングの名前 (システム生成の場合もある)。
FUNCSCHEMA	VARCHAR(128)	Yes	関数のスキーマ。システム組み込み関数の場合は NULL 値。
FUNCNAME	VARCHAR(1024)	Yes	ローカル関数 (組み込みまたはユーザー定義) の名前。
FUNCID	INTEGER	Yes	内部的に割り当てられた ID。
SPECIFICNAME	VARCHAR(128)	Yes	ローカル関数インスタンスの名前。
DEFINER	VARCHAR(128)		このマッピングの作成に使用された許可 ID。
WRAPNAME	VARCHAR(128)	Yes	マッピングが適用されるラッパー名。
SERVERNAME	VARCHAR(128)	Yes	データ・ソースの名前。
SERVERTYPE	VARCHAR(30)	Yes	マッピングが適用されるデータ・ソースのタイプ。
SERVERVERSION	VARCHAR(18)	Yes	マッピングが適用されるサーバー・タイプのバージョン。
CREATE_TIME	TIMESTAMP	Yes	マッピングが作成された時刻。
REMARKS	VARCHAR(254)	Yes	ユーザー提供のコメントまたは NULL 値。

SYSCAT.HIERARCHIES

各行は、副表とすぐ上のスーパー表、サブタイプとすぐ上のスーパータイプ、またはサブビューとすぐ上のスーパービューの関係を表しています。このビューには、直接の階層関係しか入っていません。

表 70. SYSCAT.HIERARCHIES カタログ・ビュー

列名	データ型	NULL 可能	説明
METATYPE	CHAR(1)		次のように関係のタイプをエンコードします。 R = 構造化タイプの相互関係 U = タイプ表の相互関係 W = タイプ・ビューの相互関係
SUB_SCHEMA	VARCHAR(128)		サブタイプ、副表、またはサブビューの修飾名。
SUB_NAME	VARCHAR(128)		
SUPER_SCHEMA	VARCHAR(128)		スーパータイプ、スーパー表、またはスーパービューの修飾名。
SUPER_NAME	VARCHAR(128)		
ROOT_SCHEMA	VARCHAR(128)		階層のルートにある表、ビュー、またはタイプの修飾名。
ROOT_NAME	VARCHAR(128)		

SYSCAT.INDEXAUTH

索引に関する特権ごとに 1 つの行が入ります。

表 71. SYSCAT.INDEXAUTH カタログ・ビュー

列名	データ型	NULL 可能	説明
GRANTOR	VARCHAR(128)		特権を与えたユーザーの許可 ID。
GRANTEE	VARCHAR(128)		特権を付与されたユーザーまたはグループの許可 ID。
GRANTEETYPE	CHAR(1)		U = GRANTEE は個々のユーザー。 G = GRANTEE はグループ。
INDSCHEMA	VARCHAR(128)		索引の修飾名。
INDNAME	VARCHAR(18)		
CONTROLAUTH	CHAR(1)		GRANTEE に索引に対する CONTROL 特権があるか否か。 Y = 特権がある。 N = 特権がない。

SYSCAT.INDEXCOLUSE

索引に関与するすべての列をリストします。

表 72. SYSCAT.INDEXCOLUSE カタログ・ビュー

列名	データ型	NULL 可能	説明
INDSCHEMA	VARCHAR(128)		索引の修飾名。
INDNAME	VARCHAR(18)		
COLNAME	VARCHAR(128)		列の名前。
COLSEQ	SMALLINT		索引内の列の位置番号 (最初の位置 = 1)。
COLORDER	CHAR(1)		索引内のこの列にある値の順序。値: A = 昇順 D = 降順 I = INCLUDE 列 (順序は無視される)

SYSCAT.INDEXDEP

各行は、何らかの他のオブジェクトに対する索引の従属関係を表します。

表 73. SYSCAT.INDEXDEP カタログ・ビュー

列名	データ型	NULL 可能	記述
INDSCHEMA	VARCHAR(128)		別のオブジェクトに従属する索引の修飾名。
INDNAME	VARCHAR(18)		
BTYPE	CHAR(1)		索引が従属するオブジェクトのタイプ。 A = 別名 F = 関数インスタンス O = 表またはビュー階層内のすべての副表またはサブビューに対する特権の従属関係 R = 構造化タイプ S = マテリアライズ照会表 T = 表 U = タイプ表 V = ビュー W = タイプ・ビュー X = 拡張索引
BSCHEMA	VARCHAR(128)		索引が従属しているオブジェクトの修飾名。
BNAME	VARCHAR(128)		
TABAUTH	SMALLINT	Yes	BTYPE = O、S、T、U、V、または W の場合、従属索引が必要とする表またはビューに対する特権をエンコードする。それ以外の場合は NULL。

SYSCAT.INDEXES

表に定義されている索引 (継承された列の中で適用できるものを含む) ごとに 1 行が入ります。

表 74. SYSCAT.INDEXES カタログ・ビュー

列名	データ型	NULL 可能	説明
INDSCHEMA	VARCHAR(128)		索引の名前。
INDNAME	VARCHAR(18)		
DEFINER	VARCHAR(128)		索引を作成したユーザー。
TABSCHEMA	VARCHAR(128)		索引が定義されている表またはニックネームの修飾名。
TABNAME	VARCHAR(128)		
COLNAMES	VARCHAR(640)		列名の先頭に、昇順か降順かを示す + または - を付けたもののリスト。警告: この列は将来除去されます。詳しくは、SYSCAT.INDEXCOLUSE を参照してください。
UNIQUERULE	CHAR(1)		ユニーク値に関する規則: D = 重複可 P = 1 次索引 U = ユニーク項目のみ可
MADE_UNIQUE	CHAR(1)		Y = 索引は元は非ユニークだったが、ユニーク・キー制約または主キー制約をサポートするために、ユニーク索引に変換された。制約がドロップされると、この索引は非ユニークに戻る。 N = 索引は作成時のまま。
COLCOUNT	SMALLINT		キー内の列数と組み込み列 (存在する場合) の数の合計。
UNIQUE_COLCOUNT	SMALLINT		ユニーク・キーに必要な列の数。常に <=COLCOUNT。組み込み列がある場合にのみ < COLCOUNT。索引にユニーク・キーがない場合は -1 (重複可能)。
INDEXTYPE	CHAR(4)		索引のタイプ。 CLUS = クラスタリング REG = 通常 DIM = ディメンション・ブロック索引 BLOK = ブロック索引
ENTRYTYPE	CHAR(1)		H = 階層表の索引 L = タイプ表の論理索引 非タイプ表の索引の場合は、ブランク
PCTFREE	SMALLINT		索引を最初に作成する際に予約する索引リーフ・ページのパーセント。このスペースは、索引の作成後に行う挿入用に使用可能です。
IID	SMALLINT		索引の内部 ID。

表 74. SYSCAT.INDEXES カタログ・ビュー (続き)

列名	データ型	NULL 可能	説明
NLEAF	INTEGER		リーフ・ページの数。統計が収集されていない場合は -1。
NLEVELS	SMALLINT		索引レベルの数。統計が収集されていない場合は -1。
FIRSTKEYCARD	BIGINT		最初のキーの値の種類数。統計が収集されていない場合は -1。
FIRST2KEYCARD	BIGINT		索引の最初の 2 つの列を使用するキーの種類数。統計がない場合、または適用されない場合は -1。
FIRST3KEYCARD	BIGINT		索引の最初の 3 つの列を使用するキーの種類数。統計がない場合、または適用されない場合は -1。
FIRST4KEYCARD	BIGINT		索引の最初の 4 つの列を使用するキーの種類数。統計がない場合、または適用されない場合は -1。
FULLKEYCARD	BIGINT		個別の全キー値の数。統計が収集されていない場合は -1。
CLUSTERRATIO	SMALLINT		索引によるデータ・クラスタリングの程度。統計が収集されていない場合、または詳細な索引統計が収集されている場合は -1 (それらの場合は CLUSTERFACTOR のほうが使用されます)。
CLUSTERFACTOR	DOUBLE		クラスタリングの程度の詳細測定値。詳細索引統計が収集されていない場合、またはニックネームの索引が定義されていない場合は -1。
SEQUENTIAL_PAGES	INTEGER		索引キーの順序でディスクに存在し、それらの間に大きなギャップがなく、わずかなギャップしかないリーフ・ページの数。統計が使用できない場合は -1。
DENSITY	INTEGER		索引によって占有されているページの範囲内の、ページ数に対する SEQUENTIAL_PAGES の比率。パーセントで表現される (0 から 100 の整数。統計が入手できない場合は -1。)
USER_DEFINED	SMALLINT		この索引がユーザー定義であってドロップされていない場合は 1、それ以外の場合は 0。

表 74. SYSCAT.INDEXES カタログ・ビュー (続き)

列名	データ型	NULL 可能	説明
SYSTEM_REQUIRED	SMALLINT		有効な値は以下のとおりです。 次の条件のいずれかを満たす場合は 1。 – この索引が主キー制約またはユニーク・キー制約で必要である、またはこの索引がマルチディメンション・クラスタリング (MDC) 表のディメンション・ブロック索引または複合ブロック索引である。 – これがタイプ表の (OID) 列上の索引である。 次の条件の両方を満たす場合は 2。 – この索引が主キー制約またはユニーク・キー制約で必要である、またはこの索引が MDC 表のディメンション・ブロック索引または複合ブロック索引である。 – これがタイプ表の (OID) 列上の索引である。 それ以外の場合は 0。
CREATE_TIME	TIMESTAMP		索引の作成された時刻。
STATS_TIME	TIMESTAMP	Yes	この索引について記録されている統計値が最後に変更された時刻。統計が使用可能でない場合は、NULL 値。
PAGE_FETCH_PAIRS	VARCHAR(254)		文字形式で表された整数ペアのリスト。それぞれのペアは、仮のバッファー内のページ数と、その仮のバッファーを使用した表のスキャンに必要なページ取り出しの回数を表しています。 (データが入手できない場合は、長さ 0 のストリング。)
MINPCTUSED	SMALLINT		ゼロでない場合は、オンライン索引デフラグが使用可能になり、その値は、ページをマージをする前に使用される最小スペースのしきい値です。
REVERSE_SCANS	CHAR(1)		Y = 索引は逆スキャンをサポートする N = 索引は逆スキャンをサポートしない
INTERNAL_FORMAT	SMALLINT		有効な値は以下のとおりです。 索引にリバース・ポインターがない場合は 1。 索引にリバース・ポインターがある場合は >= 2。 索引が複合ブロック索引の場合は 6。
REMARKS	VARCHAR(254)	Yes	ユーザー提供のコメントまたは NULL 値。
IESCHEMA	VARCHAR(128)	Yes	索引拡張の修飾名。通常の索引の場合は NULL。
IENAME	VARCHAR(18)	Yes	
IEARGUMENTS	CLOB(32K)	Yes	索引の作成時に指定されるパラメーターの外部情報。通常の索引の場合は NULL。
INDEX_OBJECTID	INTEGER		表の索引オブジェクト ID
NUMRIDS	BIGINT		索引内の行 ID (RID) の合計数。

表 74. SYSCAT.INDEXES カタログ・ビュー (続き)

列名	データ型	NULL 可能	説明
NUMRIDS_DELETED	BIGINT		削除対象としてマークされている、索引内の行 ID の合計数 (すべての行 ID が削除対象としてマークされている、リーフ・ページ上の行 ID は除く)。
NUM_EMPTY_LEAFS	BIGINT		すべての行 ID が削除対象としてマークされている、索引リーフ・ページの合計数。
AVERAGE_RANDOM_ FETCH_PAGES	DOUBLE		索引を使用してフェッチする際の、順次ページ・アクセス間のランダム表ページの平均数。不明の場合は -1。注 1 および 2 を参照。
AVERAGE_RANDOM_ PAGES	DOUBLE		順次索引ページ・アクセス間のランダム索引ページの平均数。不明の場合は -1。注 2 を参照。
AVERAGE_SEQUENCE_ GAP	DOUBLE		索引ページ・シーケンス間のギャップ。各ギャップは索引リーフ・ページのスキャンにより検出され、索引ページ・シーケンスの間でランダムにフェッチしなければならない索引ページの平均数を表します。不明の場合は -1。注 2 を参照。
AVERAGE_SEQUENCE_ FETCH_GAP	DOUBLE		索引を使用してフェッチする際の、表ページ・シーケンス間のギャップ。各ギャップは索引リーフ・ページのスキャンにより検出され、表ページ・シーケンスの間でランダムにフェッチしなければならない表ページの平均数を表します。不明の場合は -1。注 1 および 2 を参照。
AVERAGE_SEQUENCE_ PAGES	DOUBLE		順次にアクセス可能な索引ページの平均数 (つまり、プリフェッチャーが順次として検出する索引ページの数)。不明の場合は -1。注 2 を参照。
AVERAGE_SEQUENCE_ FETCH_PAGES	DOUBLE		索引を使用してフェッチする際の、順次にアクセス可能な表ページの平均数 (つまり、プリフェッチャーが順次として検出する表ページの数)。不明の場合は -1。注 1 および 2 を参照。
TBSPACEID	INTEGER		索引表スペースの内部 ID。

注:

1. DMS 表スペースの使用時には、この統計は計算されません。
2. LOAD...STATISTICS YES または CREATE INDEX...COLLECT STATISTICS の操作時や、データベース構成パラメーター *seqdetect* がオフになっているときは、プリフェッチ統計は集められません。

関連資料:

• 591 ページの『SYSCAT.INDEXCOLUSE』

SYSCAT.INDEXEXPLOITRULES

各行は、索引活用を表します。

表 75. SYSCAT.INDEXEXPLOITRULES カタログ・ビュー

列名	データ型	NULL 可能	説明
FUNCID	INTEGER		関数 ID。
SPECID	SMALLINT		CREATE FUNCTION ステートメント内の述部指定の数。
IESCHEMA	VARCHAR(128)		索引拡張の修飾名。
IENAME	VARCHAR(18)		
RULEID	SMALLINT		ユニークな活用規則 ID。
SEARCHMETHODID	SMALLINT		特定の索引拡張での検索メソッド ID。
SEARCHKEY	VARCHAR(320)		索引を活用するのに使用するキー。
SEARCHARGUMENT	VARCHAR(1800)		索引活用で使用する検索引き数。
EXACT	CHAR(1)		述部の評価から判断して索引参照数が正確かどうかを示します。使用できる値は以下のとおりです。 Y = 索引参照数は正確である。 N = 索引参照数は正確でない。

SYSCAT.INDEXEXTENSIONDEP

さまざまなデータベース・オブジェクトに対する索引拡張の従属関係ごとに、1 つの行が入ります。

表 76. SYSCAT.INDEXEXTENSIONDEP カタログ・ビュー

列名	データ型	NULL 可能	説明
IESCHEMA	VARCHAR(128)		別のオブジェクトに従属する索引拡張の修飾名。
IENAME	VARCHAR(18)		
BTYPE	CHAR(1)		索引拡張が従属しているオブジェクトのタイプ。 A = 別名 F = 関数インスタンスまたはメソッド・インスタンス J = サーバー定義 O = 階層 SELECT 特権への "外部" 従属関係 R = 構造化タイプ S = マテリアライズ照会表 T = 表 (型付きではない) U = タイプ表 V = ビュー (型付きではない) W = タイプ・ビュー X = 拡張索引
BSCHEMA	VARCHAR(128)		索引拡張が従属しているオブジェクトの修飾名。(BTYPE='F' の場合、これは関数の固有名になります。)
BNAME	VARCHAR(128)		
TABAUTH	SMALLINT	Yes	BTYPE='O'、'T'、'U'、'V'、または 'W' の場合、従属トリガーで必要な表 (またはビュー) の特権をエンコードします。それ以外の場合は NULL。

SYSCAT.INDEXEXTENSIONMETHODS

各行は、検索メソッドを表します。 1 つの索引拡張に、複数の検索メソッドを組み入れることができます。

表 77. SYSCAT.INDEXEXTENSIONMETHODS カタログ・ビュー

列名	データ型	NULL 可能	説明
METHODNAME	VARCHAR(18)		検索メソッドの名前。
METHODID	SMALLINT		索引拡張内のメソッドの数。
IESCHEMA	VARCHAR(128)		索引拡張の修飾名。
IENAME	VARCHAR(18)		
RANGEFUNCSCHEMA	VARCHAR(128)		範囲指定関数の修飾名。
RANGEFUNCNAME	VARCHAR(18)		
RANGESPECIFICNAME	VARCHAR(18)		範囲指定関数の固有名。
FILTERFUNCSCHEMA	VARCHAR(128)		フィルター関数の修飾名。
FILTERFUNCNAME	VARCHAR(18)		
FILTERSPECIFICNAME	VARCHAR(18)		フィルター関数の関数固有名。
REMARKS	VARCHAR(254)	Yes	ユーザー提供または NULL。

SYSCAT.INDEXEXTENSIONPARMS

各行は、索引拡張のインスタンス・パラメーターまたはソース・キー定義を表します。

表 78. SYSCAT.INDEXEXTENSIONPARMS カタログ・ビュー

列名	データ型	NULL 可能	説明
IESCHEMA	VARCHAR(128)		索引拡張の修飾名。
IENAME	VARCHAR(18)		
ORDINAL	SMALLINT		パラメーターまたはソース・キーのシーケンス番号。
PARMNAME	VARCHAR(18)		パラメーターまたはソース・キーの名前。
TYPESCHEMA	VARCHAR(128)		インスタンス・パラメーターまたはソース・キーのデータ型の修飾名。
TYPENAME	VARCHAR(18)		
LENGTH	INTEGER		インスタンス・パラメーターまたはソース・キーのデータ型の長さ。
SCALE	SMALLINT		インスタンス・パラメーターまたはソース・キーのデータ型の位取り。適用できない場合はゼロ (0)。
PARMTYPE	CHAR(1)		以下のように、行によって表されるタイプ。 P = 索引拡張パラメーター K = キー列
CODEPAGE	SMALLINT		索引拡張パラメーターのコード・ページ。ストリング・タイプでない場合はゼロ。

SYSCAT.INDEXEXTENSIONS

索引拡張ごとに 1 つの行が入ります。

表 79. SYSCAT.INDEXEXTENSIONS カタログ・ビュー

列名	データ型	NULL 可能	説明
IESCHEMA	VARCHAR(128)		索引拡張の修飾名。
IENAME	VARCHAR(18)		
DEFINER	VARCHAR(128)		索引拡張の定義時の許可 ID。
CREATE_TIME	TIMESTAMP		索引拡張が定義された時刻。
KEYGENFUNCSCHEMA	VARCHAR(128)		キー生成関数の修飾名。
KEYGENFUNCNAME	VARCHAR(18)		
KEYGENSPECIFICNAME	VARCHAR(18)		キー生成関数の固有名。
TEXT	CLOB(64K)		CREATE INDEX EXTENSION ステートメントのテキスト全体。
REMARKS	VARCHAR(254)		ユーザー提供のコメントまたは NULL 値。

SYSCAT.INDEXOPTIONS

各行には、索引固有のオプション値が入ります。

表 80. SYSCAT.INDEXOPTIONS カタログ・ビュー

列名	データ型	NULL 可能	説明
INDSCHEMA	VARCHAR(128)		索引のスキーマ名。
INDNAME	VARCHAR(18)		索引のローカル名。
OPTION	VARCHAR(128)		索引オプションの名前。
SETTING	VARCHAR(255)		値。

SYSCAT.KEYCOLUSE

ユニーク・キー、主キー、または外部キーの制約によって定義されるキー (継承された主キーまたはユニーク・キーの中で適用できるものを含む) に関与する列をすべてリストします。

表 81. SYSCAT.KEYCOLUSE カタログ・ビュー

列名	データ型	NULL 可能	説明
CONSTNAME	VARCHAR(18)		制約の名前 (表内でユニーク)。
TABSCHEMA	VARCHAR(128)		列の入った表の修飾名。
TABNAME	VARCHAR(128)		
COLNAME	VARCHAR(128)		列の名前。
COLSEQ	SMALLINT		キー内の列の位置番号 (最初の位置 = 1)。

SYSCAT.NAMEMAPPINGS

各行は、論理オブジェクトと、論理オブジェクトを実装するそれぞれの実装オブジェクトのマッピングを表します。

表 82. SYSCAT.NAMEMAPPINGS カタログ・ビュー

列名	データ型	NULL 可能	記述
TYPE	CHAR(1)		C = 列 I = 索引 U = タイプ表
LOGICAL_SCHEMA	VARCHAR(128)		論理オブジェクトの修飾名。
LOGICAL_NAME	VARCHAR(128)		
LOGICAL_COLNAME	VARCHAR(128)	Yes	TYPE = C の場合は、論理列の名前です。それ以外の場合は NULL 値。
IMPL_SCHEMA	VARCHAR(128)		論理オブジェクトを実装する実装オブジェクトの修飾名。
IMPL_NAME	VARCHAR(128)		
IMPL_COLNAME	VARCHAR(128)	Yes	TYPE = C の場合は、実装列の名前です。それ以外の場合は NULL 値。

SYSCAT.PACKAGEAUTH

パッケージに関する特権ごとに 1 つの行が入っています。

表 83. SYSCAT.PACKAGEAUTH カタログ・ビュー

列名	データ型	NULL 可能	説明
GRANTOR	VARCHAR(128)		特権を与えたユーザーの許可 ID。
GRANTEE	VARCHAR(128)		特権を付与されたユーザーまたはグループの許可 ID。
GRANTEETYPE	CHAR(1)		U = GRANTEE は個々のユーザー。 G = GRANTEE はグループ。
PKGSCHEMA	VARCHAR(128)		特権の対象となるパッケージの名前。
PKGNAME	CHAR(8)		
CONTROLAUTH	CHAR(1)		GRANTEE に、パッケージに対する CONTROL 特権があるか否か。 Y = 特権がある。 N = 特権がない。
BINDAUTH	CHAR(1)		GRANTEE に、パッケージに対する BIND 特権があるか否か。 Y = 特権がある。 N = 特権がない。 G = 特権があり、付与可能。
EXECUTEAUTH	CHAR(1)		GRANTEE に、パッケージに対する EXECUTE 特権があるか否か。 Y = 特権がある。 N = 特権がない。 G = 特権があり、付与可能。

SYSCAT.PACKAGEDEP

索引、表、ビュー、トリガー、関数、別名、タイプ、および階層に対するパッケージの従属関係ごとに 1 つの行が入っています。

表 84. SYSCAT.PACKAGEDEP カタログ・ビュー

列名	データ型	NULL 可能	説明
PKGSCHEMA	VARCHAR(128)		パッケージの名前。
PKGNAME	CHAR(8)		
UNIQUEID	CHAR(8)		パッケージを最初に作成された時点を示す内部日付/時刻情報。複数のパッケージで同じ名前が存在する場合に、特定のパッケージを識別するのに便利です。
PKGVERSION	VARCHAR(64)		パッケージのバージョン ID。
BINDER	VARCHAR(128)	Yes	パッケージをバインドしたユーザー。
BTYPE	CHAR(1)		オブジェクト BNAME のタイプ。 A = 別名 B = トリガー D = サーバー定義 F = 関数インスタンス I = 索引 M = 関数マッピング N = ニックネーム O = 表またはビュー階層内のすべての副表またはサブビューに対する特権の従属関係 P = ページ・サイズ R = 構造化タイプ S = マテリアライズ照会表 T = 表 U = タイプ表 V = ビュー W = タイプ・ビュー
BSCHEMA	VARCHAR(128)		パッケージが従属しているオブジェクトの修飾名。
BNAME	VARCHAR(128)		
TABAUTH	SMALLINT	Yes	BTYPE が O、S、T、U、V または W の場合、このパッケージで必要な特権 (SELECT、INSERT、DELETE、UPDATE) をエンコードします。

注: 従属関係のある関数インスタンスがドロップされると、パッケージは「作動不能」状態になり、明示的に再バインドする必要があります。従属関係のあるほかのオブジェクトがドロップされると、パッケージは「無効」状態になり、最初の参照時に、システムによってパッケージの再バインドが自動的に試みられます。

SYSCAT.PACKAGES

アプリケーション・プログラムをバインドして作成されたパッケージごとに 1 つの行が入っています。

表 85. SYSCAT.PACKAGES カタログ・ビュー

列名	データ型	NULL 可能	説明
PKGSCHEMA	VARCHAR(128)		パッケージ名のスキーマ修飾子。
PKGNAME	CHAR(8)		パッケージ名の非修飾 ID。
PKGVERSION	VARCHAR(64)		パッケージ名のバージョン ID。
BOUNDBY	VARCHAR(128)		パッケージをバインドしたユーザーの許可 ID (OWNER)。
DEFINER	VARCHAR(128)		パッケージをバインドしたユーザーのユーザー ID。
DEFAULT_SCHEMA	VARCHAR(128)		静的 SQL ステートメントの非修飾名に使用されるデフォルト・スキーマ (QUALIFIER) の名前。
VALID	CHAR(1)		Y = 有効 N = 無効 X = パッケージが従属している関数インスタンスがドロップされたので、パッケージは作動不能。明示的な再バインドが必要です。 (従属関係のある関数インスタンスがドロップされた場合、パッケージは「作動不能」状態になり、明示的に再バインドする必要があります。従属関係のある他のオブジェクトがドロップされた場合、パッケージは「作動不能」状態になり、システムはこのパッケージが最初に参照されるときに自動的にこのパッケージの再バインドを試みます。
UNIQUE_ID	CHAR(8)		パッケージを最初に作成された時点を示す内部日付/時刻情報。同じ名前を持つ複数のパッケージが存在している場合に、特定のパッケージを識別するのに役立つ。
TOTAL_SECT	SMALLINT		パッケージのセクションの合計数。
FORMAT	CHAR(1)		パッケージに関連した日付と時刻のフォーマット。 0 = クライアントのテリトリー・コードに関連したフォーマット 1 = USA 形式の日付、時刻 2 = EUR 形式の日付、時刻 3 = ISO 形式の日付、時刻 4 = JIS 形式の日付、時刻 5 = LOCAL 形式の日付、時刻

表 85. SYSCAT.PACKAGES カタログ・ビュー (続き)

列名	データ型	NULL 可能	説明
ISOLATION	CHAR(2)	Yes	分離レベル。 RR = 反復可能読み取り RS = 読み取り固定 CS = カーソル固定 UR = 非コミット読み取り
BLOCKING	CHAR(1)	Yes	カーソル・ブロッキング・オプション。 N = ブロッキングなし U = 確定カーソルをブロック化 B = すべてのカーソルをブロック化
INSERT_BUF	CHAR(1)		バインド時に使用された挿入オプション。 Y = 挿入内容はバッファーに入れられる N = 挿入内容はバッファーに入れられない
REOPTVAR	CHAR(1)		以下の可変値を使って実行時にアクセス・パスが再最適化されるかどうかを示します。 A = OPEN または EXECUTE 要求ごとにアクセス・パスは再最適化されます。 N = アクセス・パスは実行時には再最適化されません。 O = 最初の OPEN または EXECUTE 要求でのみアクセス・パスは再最適化されます。これはその後キャッシュに入れられます。
OS_PTR_SIZE	INTEGER		次のような、パッケージの作成場所であるプラットフォームのワード・サイズを示します。 32 = パッケージは 32 ビット・パッケージです。 64 = パッケージは 64 ビット・パッケージです。
LANG_LEVEL	CHAR(1)	Yes	BIND 時に使用された LANGLEVEL 値。 0 = SAA1 1 = MIA 2 = SQL92E
FUNC_PATH	VARCHAR(254)		このパッケージの最後の BIND コマンドで使用された SQL パス。これは REBIND のデフォルトのパスとして使用されます。バージョン 2 より前のパッケージでは SYSIBM。
QUERYOPT	INTEGER		このパッケージをバインドした最適化クラス。再バインドに使用されます。 0、1、3、5、および 9 のクラスがあります。
EXPLAIN_LEVEL	CHAR(1)		EXPLAIN または EXPLSNAP BIND オプションを使用して、 Explain が要求されたか否か。 P = プラン選択レベル Explain が要求されていない場合は、ブランク

表 85. SYSCAT.PACKAGES カタログ・ビュー (続き)

列名	データ型	NULL 可能	説明
EXPLAIN_MODE	CHAR(1)		EXPLAIN BIND オプションの値。 Y = Yes (静的) N = No A = All (静的と動的)
EXPLAIN_SNAPSHOT	CHAR(1)		EXPLSNAP BIND オプションの値。 Y = Yes (静的) N = No A = All (静的と動的)
SQLWARN	CHAR(1)		動的 SQL ステートメントからアプリケーションに戻される結果の正の SQLCODE。 Y = Yes N = No (抑制される)
SQLMATHWARN	CHAR(1)		バインド時のデータベース構成パラメーター DFT_SQLMATHWARN の値。静的 SQL ステートメントの算術計算エラーと検索変換エラーを、警告を伴う NULL 値として扱うか否か。 Y = Yes N = No (抑制される)
EXPLICIT_BIND_TIME	TIMESTAMP		このパッケージを最後に明示的にバインドまたは再バインドした時刻。パッケージを暗黙に再バインドすると、この時点以降に作成された関数インスタンスは選択されません。
LAST_BIND_TIME	TIMESTAMP		このパッケージが最後に明示的にバインドまたは再バインドされた時刻。
CODEPAGE	SMALLINT		バインド時のアプリケーションのコード・ページ (不明の場合は -1)。
DEGREE	CHAR(5)		パッケージのバインド時のパーティション内並列処理に関する限界の指定 (BIND オプションとしての)。 1 = パーティション内並列処理はない。 2 から 32 767 = パーティション内並列処理の度合い。 ANY = 度合いはデータベース・マネージャーによって決定される。
MULTINODE_PLANS	CHAR(1)		Y = パッケージは複数パーティションの環境でバインドされた。 N = パッケージは単一パーティションの環境でバインドされた。

表 85. SYSCAT.PACKAGES カタログ・ビュー (続き)

列名	データ型	NULL 可能	説明
INTRA_PARALLEL	CHAR(1)		パッケージ内の静的 SQL ステートメントによるパーティション内並列処理の使用を示します。 Y = パッケージ内の 1 つまたは複数の静的 SQL ステートメントがパーティション内並列処理を使用する。 N = パッケージ内の静的 SQL ステートメントはパーティション内並列処理を使用しない。 F = パッケージ内の 1 つまたは複数の静的 SQL ステートメントはパーティション内並列処理を使用可能。この並列処理は、パーティション内並列処理を使用するように構成されていないシステムでは使用不可になっています。
VALIDATE	CHAR(1)		B = すべての検査は BIND 実行時に行う必要がある。 R = 予約済み。

表 85. SYSCAT.PACKAGES カタログ・ビュー (続き)

列名	データ型	NULL 可能	説明
DYNAMICRULES	CHAR(1)		B = BIND。動的 SQL ステートメントはバインド動作で実行されます。 D = DEFINEBIND。ルーチン・コンテキスト内でパッケージが実行される場合、パッケージ内の動的 SQL ステートメントは定義動作で実行されます。パッケージがルーチン・コンテキスト内で実行されない場合は、パッケージ内の動的 SQL ステートメントはバインド動作で実行されます。 E = DEFINERUN。ルーチン・コンテキスト内でパッケージが実行される場合、パッケージ内の動的 SQL ステートメントは定義動作で実行されます。パッケージがルーチン・コンテキスト内で実行されない場合は、パッケージ内の動的 SQL ステートメントは実行動作で実行されます。 H = INVOKEBIND。ルーチン・コンテキスト内でパッケージが実行される場合、パッケージ内の動的 SQL ステートメントは呼び出し動作で実行されます。パッケージがルーチン・コンテキスト内で実行されない場合は、パッケージ内の動的 SQL ステートメントはバインド動作で実行されます。 I = INVOKERUN。ルーチン・コンテキスト内でパッケージが実行される場合、パッケージ内の動的 SQL ステートメントは呼び出し動作で実行されます。パッケージがルーチン・コンテキスト内で実行されない場合は、パッケージ内の動的 SQL ステートメントは実行動作で実行されます。 R = RUN。動的 SQL ステートメントは実行動作で実行されます。これがデフォルトです。
SQLERROR	CHAR(1)		パッケージをバインドまたは再バインドした最新のサブコマンドの SQLERROR オプション。 C = 予約済み N = パッケージなし
REFRESHAGE	DECIMAL (20,6)		タイム・スタンプ期間。REFRESH TABLE ステートメントがマテリアライズ照会表に実行されてから、マテリアライズ照会表が基本表の代わりに使用されるまでの最大時間を示す。
TRANSFORMGROUP	CHAR(1024)	Yes	トランスフォーム・グループ BIND オプションの入ったストリング。
REMARKS	VARCHAR(254)	Yes	ユーザー提供のコメントまたは NULL 値。

SYSCAT.PARTITIONMAPS

表の区分化キーのハッシュに基づいて、データベース・パーティション・グループ内のパーティションの間で表の行を再分散するために使用される区分化マップごとに、 1 つの行が入ります。

表 86. SYSCAT.PARTITIONMAPS カタログ・ビュー

列名	データ型	NULL 可能	説明
PMAP_ID	SMALLINT		区分化マップの ID。
PARTITIONMAP	LONG VARCHAR FOR BIT DATA		実際の区分化マップ。複数のパーティション・データベースのデータベース・パーティション・グループの場合は、4 096 個の 2 バイト整数から成るベクトル。単一のパーティション・データベースのデータベース・パーティション・グループの場合は、単一パーティションのパーティション番号を示す項目が 1 つあります。

SYSCAT.PASSTHRUAUTH

このカタログ・ビューには、パススルー・セッション内のデータ・ソースを照会できる権限に関する情報が示されます。基本表の制約により、 SERVER の値は、SYSCAT.SERVERS の SERVER 列の値に対応している必要があります。SYSCAT.PASSTHRUAUTH には NULL 可能なフィールドはありません。

表 87. SYSCAT.PASSTHRUAUTH カタログ・ビューの列

列名	データ型	NULL 可能	記述
GRANTOR	VARCHAR(128)		特権を付与したユーザーの許可 ID。
GRANTEE	VARCHAR(128)		特権を保持するユーザーまたはグループの許可 ID。
GRANTEETYPE	CHAR(1)		GRANTEE のタイプを指定する文字: U = GRANTEE は個々のユーザー。 G = GRANTEE はグループ。
SERVERNAME	VARCHAR(128)		ユーザーまたはグループが許可を付与されるデータ・ソースの名前。

SYSCAT.PREDICATESPECS

各行は述部指定を表します。

表 88. SYSCAT.PREDICATESPECS カタログ・ビュー

列名	データ型	NULL 可能	説明
FUNCSCHEMA	VARCHAR(128)		関数の修飾名。
FUNCNAME	VARCHAR(128)		
SPECIFICNAME	VARCHAR(128)		関数インスタンスの名前。
FUNCID	INTEGER		関数 ID。
SPECID	SMALLINT		この述部指定の ID。
CONTEXTOP	CHAR(8)		比較演算子は組み込み関係演算子 (=、<、>= など) のいずれか。
CONTEXTEXP	CLOB(32K)		定数、または SQL 式。
FILTERTEXT	CLOB(32K)	Yes	データ・フィルター式のテキスト。

SYSCAT.PROCOPTIONS

各行には、プロシージャー固有のオプション値が入ります。

表 89. SYSCAT.PROCOPTIONS カタログ・ビュー

列名	データ型	NULL 可能	記述
PROCSCHEMA	VARCHAR(128)		ストアード・プロシージャーの名前またはニックネームの修飾子。
PROCNAME	VARCHAR(128)		ストアード・プロシージャーの名前またはニックネーム。
OPTION	VARCHAR(128)		ストアード・プロシージャー・オプションの名前。
SETTING	VARCHAR(255)		ストアード・プロシージャー・オプションの値。

SYSCAT.PROCPARMOPTIONS

各行には、プロシージャー・パラメーター固有のオプション値が入ります。

表 90. SYSCAT.PROCPARMOPTIONS カタログ・ビュー

列名	データ型	NULL 可能	記述
PROCSCHEMA	VARCHAR(128)		プロシージャーの修飾名またはニックネーム。
PROCNAME	VARCHAR(128)		
ORDINAL	SMALLINT		プロシージャーのシグニチャー内でのパラメーターの位置番号。
OPTION	VARCHAR(128)		ストアード・プロシージャーのパラメーター・オプションの名前。
SETTING	VARCHAR(255)		ストアード・プロシージャーのパラメーター・オプションの値。

SYSCAT.REFERENCES

定義された参照制約ごとに 1 つの行が入っています。

表 91. SYSCAT.REFERENCES カタログ・ビュー

列名	データ型	NULL 可能	説明
CONSTNAME	VARCHAR(18)		制約の名前。
TABSCHEMA	VARCHAR(128)		表の修飾名。
TABNAME	VARCHAR(128)		
DEFINER	VARCHAR(128)		制約を作成したユーザー。
REFKEYNAME	VARCHAR(18)		親キーの名前。
REFTABSCHEMA	VARCHAR(128)		親表の修飾名。
REFTABNAME	VARCHAR(128)		
COLCOUNT	SMALLINT		外部キーを構成する列の数。
DELETERULE	CHAR(1)		削除規則。 A = NO ACTION (アクションなし) C = CASCADE (削除ルール : カスケード) N = SET NULL (NULL 設定) R = RESTRICT (削除ルール : 削除禁止)
UPDATERULE	CHAR(1)		更新規則。 A = NO ACTION (アクションなし) R = RESTRICT (削除ルール : 削除禁止)
CREATE_TIME	TIMESTAMP		参照制限が定義されたタイム・スタンプ。
FK_COLNAMES	VARCHAR (640)		外部キーの列名のリスト。警告: この列は将来除去されます。詳しくは、SYSCAT.KEYCOLUSE を参照してください。
PK_COLNAMES	VARCHAR (640)		親キーの列名のリスト。警告: この列は将来除去されます。詳しくは、SYSCAT.KEYCOLUSE を参照してください。

注: SYSCAT.REFERENCES ビューは、バージョン 1 の SYSIBM.SYSRELS 表に基づいています。

関連資料:

- 603 ページの『SYSCAT.KEYCOLUSE』

SYSCAT.REVTYPEMAPPINGS

各行には、リバース・データ型マッピング (ローカルに定義されたデータ型から、データ・ソースのデータ型にマップすること) が入ります。このバージョンにデータはありません。将来の利用のためにデータ型マッピングを使って定義されます。

表 92. SYSCAT.REVTYPEMAPPINGS カタログ・ビュー

列名	データ型	NULL 可能	記述
TYPE_MAPPING	VARCHAR(18)		リバース・タイプ・マッピングの名前 (システム生成の名前の場合もある)。
TYPESCHEMA	VARCHAR(128)	Yes	タイプのスキーマ名。システム組み込みタイプの場合は NULL 値。
TYPENAME	VARCHAR(18)		リバース・タイプ・マッピングでのローカル・タイプの名前。
TYPEID	SMALLINT		タイプ ID。
SOURCETYPEID	SMALLINT		ソース・タイプ ID。
DEFINER	VARCHAR(128)		このタイプ・マッピングの作成に使用された許可 ID。
LOWER_LEN	INTEGER	Yes	ローカル・タイプの長さ/精度の下限。
UPPER_LEN	INTEGER	Yes	ローカル・タイプの長さ/精度の上限。 NULL 値であれば、システムは最善の長さ/精度属性を判別します。
LOWER_SCALE	SMALLINT	Yes	ローカルの 10 進数データ型の位取りの下限。
UPPER_SCALE	SMALLINT	Yes	ローカルの 10 進数データ型の位取りの上限。 NULL 値であれば、システムは最善の位取りの属性を判別します。
S_OPR_P	CHAR(2)	Yes	ローカルの位取りとローカルの精度の関係。基本比較演算子を使用できます。 NULL 値であれば、特定の関係は不要です。
BIT_DATA	CHAR(1)	Yes	Y = 型はビット・データ用。 N = 型はビット・データ用ではない。 NULL = 文字データ型ではなく、システムはビット・データ属性を判別しない。
WRAPNAME	VARCHAR(128)	Yes	このデータ・アクセス・プロトコルにマッピングが適用されます。
SERVERNAME	VARCHAR(128)	Yes	データ・ソースの名前。
SERVERTYPE	VARCHAR(30)	Yes	このタイプのデータ・ソースにマッピングが適用されます。
SERVERVERSION	VARCHAR(18)	Yes	このバージョンの SERVERTYPE にマッピングが適用されます。
REMOTE_TYPESCHEMA	VARCHAR(128)	Yes	リモート・タイプのスキーマ名。
REMOTE_TYPENAME	VARCHAR(128)		データ・ソースに対して定義されたデータ型の名前。

表 92. SYSCAT.REVTYPEMAPPINGS カタログ・ビュー (続き)

列名	データ型	NULL 可能	記述
REMOTE_META_TYPE	CHAR(1)	Yes	S = リモート・タイプはシステム組み込みタイプ。 T = リモート・タイプは特殊タイプ。
REMOTE_LENGTH	INTEGER	Yes	リモート 10 進タイプの最大桁数とリモート・文字タイプの最大文字数。それ以外の場合は NULL 値。
REMOTE_SCALE	SMALLINT	Yes	小数点以下に使用できる最大桁数 (リモート 10 進タイプの場合)。それ以外の場合は NULL 値。
REMOTE_BIT_DATA	CHAR(1)	Yes	Y = 型はビット・データ用。 N = 型はビット・データ用ではない。 NULL = 文字データ型ではなく、システムはビット・データ属性を判別しない。
USER_DEFINED	CHAR(1)		ユーザーによって定義される。
CREATE_TIME	TIMESTAMP		このマッピングが作成された時刻。
REMARKS	VARCHAR(254)	Yes	ユーザー提供のコメントまたは NULL 値。

SYSCAT.ROUTINEAUTH

データベースの特定のルーチンに関する特権が付与されているユーザーまたはグループごとに 1 つまたは複数の行が入っています。

表 93. SYSCAT.ROUTINEAUTH カタログ・ビュー

列名	データ型	NULL 可能	説明
GRANTOR	VARCHAR(128)		特権を付与したユーザーの許可 ID、または SYSIBM。
GRANTEE	VARCHAR(128)		特権を保持するユーザーまたはグループの許可 ID。
GRANTEETYPE	CHAR(1)		U = GRANTEE は個々のユーザー。 G = GRANTEE はグループ。
SCHEMA	VARCHAR(128)		ルーチンの修飾子。
SPECIFICNAME	VARCHAR(128)	Yes	ルーチンの名前を指定します。 SPECIFICNAME が NULL で ROUTINETYPE が M ではない場合、特権は ROUTINETYPE で指定されたタイプのスキーマ内のすべてのルーチンに適用されます。 SPECIFICNAME が NULL で ROUTINETYPE が M である場合、特権は サブジェクト・タイプ TYPENAME のスキーマ内のすべてのメソッドに適用されます。
TYPESCHEMA	VARCHAR(128)	Yes	メソッドのタイプ名の修飾子。 ROUTINETYPE が M ではない場合、TYPESCHEMA は NULL です。
TYPENAME	VARCHAR(18)	Yes	メソッドのタイプ名。 ROUTINETYPE が M ではない場合、TYPENAME は NULL です。 TYPENAME が NULL で ROUTINETYPE が M の場合、特権はスキーマ TYPESCHEMA 内のサブジェクト・タイプに適用されます。
ROUTINETYPE	CHAR(1)		ルーチンのタイプ F = 関数 M = メソッド P = プロシージャー
EXECUTEAUTH	CHAR(1)		GRANTEE に、関数またはメソッドに対する EXECUTE 特権があるか否か。 Y = 特権がある。 G = 特権があり、付与可能。 N = 特権がない。
GRANT_TIME	TIMESTAMP		EXECUTE 特権が付与された時刻。

SYSCAT.ROUTINEDEP

各行は、何らかの他のオブジェクトに対するルーチンの従属関係を表します。 (このカタログ・ビューは SYSCAT.FUNCDEP に取って代わります。他のビューは存在しますが、DB2 バージョン 7.1 のときのままになります。)

表 94. SYSCAT.FUNCDEP カタログ・ビュー

列名	データ型	NULL 可能	記述
ROUTINESCHEMA	VARCHAR(128)		別のオブジェクトに従属するルーチンの修飾名。
ROUTINENAME	VARCHAR(128)		
BTYPE	CHAR(1)		ルーチンが依存するオブジェクトのタイプ。 A = 別名 F = ルーチン・インスタンス O = 表またはビュー階層内のすべての副表またはサブビューに対する特権の従属関係 R = 構造化タイプ S = マテリアライズ照会表 T = 表 U = タイプ表 V = ビュー W = タイプ・ビュー X = 拡張索引
BSCHEMA	VARCHAR(128)		関数またはメソッドが従属しているオブジェクトの修飾名 (BTYPE = F の場合、これはルーチンの固有名です)。
BNAME	VARCHAR(128)		
TABAUTH	SMALLINT	Yes	BTYPE = O、S、T、U、V、または W の場合、従属ルーチンで必要な表またはビューの特権をエンコードします。それ以外の場合は NULL 値。

SYSCAT.ROUTINEPARMS

SYSCAT.ROUTINES に定義されているルーチンの、パラメーターまたは結果ごとに、1 つの行が入れられます。 (このカタログ・ビューは、 SYSCAT.FUNCPARMS および SYSCAT.PROCPARMS に代わるものです。他のビューは存在しますが、DB2 バージョン 7.1 では旧来のまま残されます。

表 95. SYSCAT.ROUTINEPARMS カタログ・ビュー

列名	データ型	NULL 可能	説明
ROUTINESCHEMA	VARCHAR(128)		ルーチンの修飾名。
ROUTINENAME	VARCHAR(128)		
SPECIFICNAME	VARCHAR(128)		ルーチン・インスタンスの名前 (システム生成の場合もある)。
PARMNAME	VARCHAR(128)	Yes	パラメーターまたは結果の列の名前、あるいは名前が存在しない場合は NULL 値。
ROWTYPE	CHAR(1)		B = 入力と出力の両方のパラメーター C = キャスト後の結果 O = 出力パラメーター P = 入力パラメーター R = キャスト前の結果
ORDINAL	SMALLINT		ROWTYPE = B、O、または P の場合、ルーチン・シグニチャー内のパラメーターの位置番号。ROWTYPE= R であり、なおかつルーチンが表関数である場合、その結果表内の列の位置番号。それ以外の場合は 0。
TYPESCHEMA	VARCHAR(128)		パラメーターまたは結果のデータ型の修飾名。
TYPENAME	VARCHAR(128)		
LOCATOR	CHAR(1)		Y = パラメーターまたは結果は、ロケーターの形式で渡される N = パラメーターまたは結果は、ロケーターの形式で渡されない
LENGTH	INTEGER		パラメーターまたは結果の長さ。パラメーターまたは結果が特殊タイプの場合は 0。注 1 を参照。
SCALE	SMALLINT		パラメーターまたは結果の位取り。パラメーターまたは結果が特殊タイプの場合は 0。注 1 を参照。
CODEPAGE	SMALLINT		パラメーターまたは結果のコード・ページ。該当しない場合、または FOR BIT DATA 属性を指定して宣言された文字データのパラメーターか結果の場合は 0。
CAST_FUNCSCHEMA	VARCHAR(128)	Yes	引き数また結果のキャストに使用される関数の修飾名。ソース関数および外部関数に適用され、それ以外の場合は NULL。
CAST_FUNCSPECIFIC	VARCHAR(128)	Yes	

表 95. SYSCAT.ROUTINEPARMS カタログ・ビュー (続き)

列名	データ型	NULL 可能	説明
TARGET_TYPESCHEMA	VARCHAR(128)	Yes	パラメーターまたは結果のタイプが REFERENCE の場合、ターゲット・タイプの修飾名。パラメーターまたは結果のタイプが REFERENCE でない場合は、NULL 値。
TARGET_TYPENAME	VARCHAR(128)	Yes	
SCOPE_TABSCHEMA	VARCHAR(128)	Yes	パラメーターまたは結果のタイプが REFERENCE の場合、有効範囲 (ターゲット表) の修飾名。パラメーターまたは結果のタイプが REFERENCE でない場合や、有効範囲が定義されていない場合は、NULL 値。
SCOPE_TABNAME	VARCHAR(128)	Yes	
TRANSFORMGRPNAME	VARCHAR(128)	Yes	構造化タイプのパラメーターまたは結果の場合は、トランスフォーム・グループの名前。
REMARKS	VARCHAR(254)	Yes	パラメーターの注釈。

注:

1. ソース関数からの派生関数 (他の関数を参照して定義された関数) はソースのパラメーターの長さと位取りを継承するので、そのような関数の LENGTH と SCALE は 0 に設定されます。

SYSCAT.ROUTINES

ユーザー定義関数 (スカラー、表、またはソース)、システム生成メソッド、ユーザー定義メソッド、またはプロシージャーごとに、1 つの行が入ります。組み込み関数は組み入れられません。 (このカタログ・ビューは、 SYSCAT.FUNCTIONS および SYSCAT.PROCEDURES に代わるものです。ほかのビューが存在するものの、DB2 バージョン 7.1 のものとして残っています。)

表 96. SYSCAT.ROUTINES カタログ・ビュー

列名	データ型	NULL 可能	説明
ROUTINESCHEMA	VARCHAR(128)		ルーチンの修飾名。
ROUTINENAME	VARCHAR(128)		
ROUTINETYPE	CHAR(1)		F = 関数 M = メソッド P = プロシージャー
DEFINER	VARCHAR(128)		ルーチンの定義者の許可 ID。
SPECIFICNAME	VARCHAR(128)		ルーチン・インスタンスの名前 (システム生成の場合もある)。
ROUTINEID	INTEGER		内部割り当てによるルーチン ID。
RETURN_TYPESCHEMA	VARCHAR(128)	Yes	スカラー関数またはメソッドの場合、戻りタイプの修飾名。
RETURN_TYPENAME	VARCHAR(128)	Yes	
ORIGIN	CHAR(1)		B = 組み込み E = ユーザー定義、外部 M = テンプレート Q = SQL 本体 U = ユーザー定義、ソース関数に基づく S = システム生成 T = システム生成トランスフォーム
FUNCTIONTYPE	CHAR(1)		C = 列関数 R = 行関数 S = スカラー関数またはメソッド T = 表関数 ブランク = プロシージャー
PARM_COUNT	SMALLINT		パラメーター数。
LANGUAGE	CHAR(8)		ルーチン本体のインプリメンテーション言語。使用できる値は C、COBOL、JAVA、 OLE、OLEDB、または SQL。 ORIGIN が E または Q でない場合はブランク。

表 96. SYSCAT.ROUTINES カタログ・ビュー (続き)

列名	データ型	NULL 可能	説明
SOURCESCHEMA	VARCHAR(128)	Yes	ORIGIN=U かつルーチンがユーザー定義関数の場合、ソース関数の修飾名。 ORIGIN=U で、しかもソース関数が組み込み関数の場合、SOURCESCHEMA は 'SYSIBM'、SOURCESPECIFIC は '組み込み関数の場合は N/A'。 ORIGIN が U でない場合は、NULL 値。
SOURCESPECIFIC	VARCHAR(128)	Yes	
DETERMINISTIC	CHAR(1)		Y = 決定論 (結果は一貫している) N = 非決定論 (結果は毎回同じとは限らない) ORIGIN が E または Q でない場合はブランク。
EXTERNAL_ACTION	CHAR(1)		E = 関数に外部副次作用がある (呼び出しの数が重要) N = 副次作用がない。 ORIGIN が E または Q でない場合はブランク。
NULLCALL	CHAR(1)		Y = CALLED ON NULL INPUT N = RETURNS NULL ON NULL INPUT (オペランドに NULL 値が入っている場合、暗黙のうちに結果は NULL 値) ORIGIN が E または Q でない場合はブランク。
CAST_FUNCTION	CHAR(1)		Y = cast 関数 N = cast 関数ではない
ASSIGN_FUNCTION	CHAR(1)		Y = 暗黙的な割り当て関数 N = 割り当て関数ではない
SCRATCHPAD	CHAR(1)		Y = このルーチンはスクラッチ・パッドあり N = このルーチンはスクラッチ・パッドなし ORIGIN が E または ROUTINETYPE が P でない場合はブランク。
SCRATCHPAD_LENGTH	SMALLINT		*n* = スクラッチパッドの長さ (バイト単位) 0 = SCRATCHPAD が N -1 = LANGUAGE が OLEDB
FINALCALL	CHAR(1)		Y = ステートメント終了の実行時に、この関数に対して最終呼び出しが行われる N = 最終呼び出しを行わない ORIGIN が E でない場合はブランク
PARALLEL	CHAR(1)		Y = 関数を並列して実行できる N = 関数を並列して実行できない ORIGIN が E でない場合はブランク

表 96. SYSCAT.ROUTINES カタログ・ビュー (続き)

列名	データ型	NULL 可能	説明
PARAMETER_STYLE	CHAR(8)		ルーチンの作成時に宣言されたパラメーターのスタイル。値: DB2SQL SQL DB2GENRL GENERAL JAVA DB2DARI GNRLNULL ORIGIN が E でない場合はブランク
FENCED	CHAR(1)		Y = fenced N = fenced でない ORIGIN が E でない場合はブランク
SQL_DATA_ACCESS	CHAR(1)		C = CONTAINS SQL: SQL データの読み取りまたは変更を行わない SQL に限り許可されている。 M = MODIFIES SQL DATA: ルーチンで許可されているすべての SQL が許可されている。 N = NO SQL: SQL は許可されていない。 R = READS SQL DATA: SQL データを読み取る SQL に限り許可されている。
DBINFO	CHAR(1)		Y = DBINFO は渡される N = DBINFO は渡されない
PROGRAMTYPE	CHAR(1)		M = メイン S = サブルーチン
COMMIT_ON_RETURN	CHAR(1)		N = プロシージャーの完了後に変更がコミットされない ROUTINETYPE が P でない場合はブランク
RESULT_SETS	SMALLINT		戻される結果セットの上限の見積もり。
SPEC_REG	CHAR(1)		I = INHERIT SPECIAL REGISTERS: ステートメントの呼び出しから、特殊レジスターをそれらの値を使用して開始する。 ORIGIN が E または Q でない場合はブランク。
FEDERATED	CHAR(1)		使用されません。
THREADSAFE	CHAR(1)		Y = ルーチンをほかルーチンと同じプロセスで実行できる。 N = ルーチンをほかのルーチンとは別のプロセスで実行しなければならない。 ORIGIN が E でない場合はブランク

表 96. SYSCAT.ROUTINES カタログ・ビュー (続き)

列名	データ型	NULL 可能	説明
VALID	CHAR(1)		Y = SQL プロシージャーが有効。 N = SQL プロシージャーが無効。 X = SQL プロシージャーが必要とする何らかのオブジェクトがドロップされたので、その SQL プロシージャーは作動不能。その SQL プロシージャーを明示的にドロップし、再作成しなければならない。 ORIGIN が Q でない場合はブランク。
METHODIMPLEMENTED	CHAR(1)		Y = メソッドがインプリメントされている。 N = インプリメントは行われずにメソッドが指定されている。 ROUTINETYPE が M でない場合はブランク。
METHODEFFECT	CHAR(2)		MU = Mutator メソッド OB = オブザーバー・メソッド CN = コンストラクター・メソッド FUNCTIONTYPE が T ではない場合、ブランク。
TYPE_PRESERVING	CHAR(1)		Y = 戻りタイプは ″type-preserving″ パラメーターで管理される。システム生成の mutator メソッドはすべてタイプ保持です。 N = 戻りタイプはメソッドの宣言された戻りタイプ。 ROUTINETYPE が M でない場合はブランク。
WITH_FUNC_ACCESS	CHAR(1)		Y = このメソッドは関数表記を使用して呼び出すことができます。 N = このメソッドは関数表記を使用して呼び出すことはできません。 ROUTINETYPE が M でない場合はブランク。
OVERRIDEN_METHODID	INTEGER	Yes	将来の使用のために予約済み。
SUBJECT_TYPESCHEMA	VARCHAR(128)	Yes	メソッドの対象となるタイプ。
SUBJECT_TYPENAME	VARCHAR(128)	Yes	
CLASS	VARCHAR(128)	Yes	LANGUAGE=JAVA の場合、このルーチンをインプリメントするクラス。それ以外の場合、NULL 値です。
JAR_ID	VARCHAR(128)	Yes	LANGUAGE=JAVA の場合、このルーチンをインプリメントする jar ファイル。それ以外の場合、NULL 値です。
JARSCHEMA	VARCHAR(128)	Yes	LANGUAGE=JAVA の場合、このルーチンをインプリメントする jar ファイルのスキーマ。それ以外の場合、NULL 値です。

表 96. SYSCAT.ROUTINES カタログ・ビュー (続き)

列名	データ型	NULL 可能	説明
JAR_SIGNATURE	VARCHAR(1024)	Yes	LANGUAGE=JAVA の場合、このルーチンをインプリメントする Java メソッドのシグニチャー。それ以外の場合、NULL 値です。
CREATE_TIME	TIMESTAMP		ルーチン作成時のタイム・スタンプ。バージョン 1 の関数の場合は 0 に設定されます。
ALTER_TIME	TIMESTAMP		最新のルーチン変更のタイム・スタンプ。ルーチンが変更されていない場合、 CREATE_TIME に設定します。
FUNC_PATH	VARCHAR(254)	Yes	ルーチンの定義された時点の SQL パス。
QUALIFIER	VARCHAR(128)		オブジェクト定義時のデフォルト・スキーマの値。
IOS_PER_INVOC	DOUBLE		呼び出しごとの入出力回数の見積もり。不明の場合は -1 (デフォルト値は 0)。
INSTS_PER_INVOC	DOUBLE		呼び出しごとの命令の数の見積もり。不明の場合は -1 (デフォルト値は 450)。
IOS_PER_ARGBYTE	DOUBLE		入力引き数 1 バイトごとの入出力回数の見積もり。不明の場合は -1 (デフォルト値は 0)。
INSTS_PER_ARGBYTE	DOUBLE		入力引き数 1 バイトごとの命令数の見積もり。不明の場合は -1 (デフォルト値は 0)。
PERCENT_ARGBYTES	SMALLINT		ルーチンが実際に読み取る入力引き数バイトの平均パーセント値の見積もり。不明の場合は -1 (デフォルト値は 100)。
INITIAL_IOS	DOUBLE		最初/最後のルーチン呼び出し時に実行される入出力回数の見積もり。不明の場合は -1 (デフォルト値は 0)。
INITIAL_INSTS	DOUBLE		最初/最後のルーチン呼び出し時に実行される命令の数の見積もり。不明の場合は -1 (デフォルト値は 0)。
CARDINALITY	BIGINT		表関数の予測されるカーディナリティー。不明の場合、またはルーチンが表関数でない場合は -1。
SELECTIVITY	DOUBLE		ユーザー定義述部での使用。ユーザー定義述部がない場合は -1。注 1 を参照。
RESULT_COLS	SMALLINT		表関数 (ROUTINETYPE = F および TYPE = T) の場合は結果表の列数。ほかの関数またはメソッド (ROUTINETYPE = F または M) の場合は 1。プロシージャー (ROUTINETYPE = P) の場合は 0。
IMPLEMENTATION	VARCHAR(254)	Yes	ORIGIN=E の場合、この関数を実現するためのパス/モジュール/関数。 ORIGIN=U、かつソース関数が組み込み関数の場合、この列に含まれるソース関数の名前とシグニチャー。それ以外の場合、NULL です。
LIB_ID	INTEGER	Yes	将来の使用のために予約されています。

表 96. SYSCAT.ROUTINES カタログ・ビュー (続き)

列名	データ型	NULL 可能	説明
TEXT_BODY_OFFSET	INTEGER		LANGUAGE = SQL の場合、 CREATE ステートメントのテキスト全体の中の、 SQL プロシージャー本体の開始位置に対するオフセット。 LANGUAGE が SQL でない場合は -1。
TEXT	CLOB(2M)	Yes	LANGUAGE = SQL の場合、 CREATE FUNCTION、CREATE METHOD、または CREATE PROCEDURE ステートメントのテキスト。
NEWSAVEPOINTLEVEL	CHAR(1)		呼び出し時にルーチンが新しいセーブポイント・レベルを開始するかどうかを指示します。 Y = ルーチンの呼び出し時に新しいセーブポイント・レベルが開始される。 N = ルーチンの呼び出し時に新しいセーブポイント・レベルが開始されない。ルーチンは既存のセーブポイント・レベルを使用します。 ORIGIN が E または Q でない場合はブランク。
DEBUG_MODE	CHAR(3)		OFF = このルーチンでデバッグをオフにする。 ON = このルーチンでデバッグをオンにする。
TRACE_LEVEL	CHAR(1)		将来の使用のために予約されています。
DIAGNOSTIC_LEVEL	CHAR(1)		将来の使用のために予約されています。
CHECKOUT_USERID	VARCHAR(128)	Yes	オブジェクトのチェックアウトを実行したユーザーのユーザー ID。チェックアウトが行われていない場合は NULL。
PRECOMPILE_OPTIONS	VARCHAR(1024)	Yes	ルーチンに指定されたプリコンパイル・オプション。
COMPILE_OPTIONS	VARCHAR(1024)	Yes	ルーチンに指定されたコンパイル・オプション。
REMARKS	VARCHAR(254)	Yes	ユーザー提供のコメントまたは NULL 値。

注:

1. どのようなユーザー定義ルーチンでも、パック記述子およびシステム・カタログでの移行中は、この列は -1 に設定されます。ユーザー定義述部の場合、システム・カタログでの選択性は -1 になります。この場合、オプティマイザーが使用する選択性の値は 0.01 です。

SYSCAT.SCHEMAAUTH

データベースの特定のスキーマに関する特権が付与されているユーザーまたはグループごとに 1 つまたは複数の行が入っています。特定の認可者から、特定の被認可者に与えられた 1 つのスキーマに関するすべてのスキーマ特権が 1 つの行に示されます。

表 97. SYSCAT.SCHEMAAUTH カタログ・ビュー

列名	データ型	NULL 可能	説明
GRANTOR	VARCHAR(128)		特権を付与したユーザーの許可 ID、または SYSIBM。
GRANTEE	VARCHAR(128)		特権を付与されたユーザーまたはグループの許可 ID。
GRANTEETYPE	CHAR(1)		U = GRANTEE は個々のユーザー。 G = GRANTEE はグループ。
SCHEMANAME	VARCHAR(128)		スキーマの名前。
ALTERINAUTH	CHAR(1)		GRANTEE に、スキーマに対する ALTERIN 特権があるか否か。 Y = 特権がある。 G = 特権があり、付与可能。 N = 特権がない。
CREATEINAUTH	CHAR(1)		GRANTEE に、スキーマに対する CREATEIN 特権があるか否か。 Y = 特権がある。 G = 特権があり、付与可能。 N = 特権がない。
DROPINAUTH	CHAR(1)		GRANTEE に、スキーマに対する DROPIN 特権があるか否か。 Y = 特権がある。 G = 特権があり、付与可能。 N = 特権がない。

SYSCAT.SCHEMATA

スキーマごとに 1 つの行が入っています。

表 98. SYSCAT.SCHEMATA カタログ・ビュー

列名	データ型	NULL 可能	説明
SCHEMANAME	VARCHAR(128)		スキーマの名前。
OWNER	VARCHAR(128)		スキーマの許可 ID。暗黙的に作成されたスキーマの値は SYSIBM。
DEFINER	VARCHAR(128)		スキーマを作成したユーザー。
CREATE_TIME	TIMESTAMP		オブジェクトが作成された時点を示すタイム・スタンプ。
REMARKS	VARCHAR(254)	Yes	ユーザーが入力したコメント。

SYSCAT.SEQUENCEAUTH

シーケンスを使用または変更するために使用できる許可 ID ごとの行が入っています。

表 99. SYSCAT.SEQUENCEAUTH カタログ・ビュー

列名	データ型	NULL 可能	説明
GRANTOR	VARCHAR(128)		特権を付与した SYSIBM または許可 ID。
GRANTEE	VARCHAR(128)		特権を保持する許可 ID。
GRANTEETYPE	CHAR(1)		特権を保持する許可 ID のタイプ。 U = GRANTEE は個々のユーザー
SEQSCHEMA	VARCHAR(128)		シーケンスの修飾名。
SEQNAME	VARCHAR(128)		
USAGEAUTH	CHAR(1)		Y = 特権がある N = 特権がない G = 特権があり、付与可能
ALTERAUTH	CHAR(1)		Y = 特権がある N = 特権がない G = 特権があり、付与可能

SYSCAT.SEQUENCES

データベースで定義された各シーケンスまたは ID 列の行が入っています。

表 100. SYSCAT.SEQUENCES カタログ・ビューの列

列名	データ型	NULL 可能	記述
SEQSCHEMA	VARCHAR(128)		シーケンスの修飾名 (ID 列に DB2 が生成した名前)。
SEQNAME	VARCHAR(128)		
DEFINER	VARCHAR(128)		シーケンスの定義者。
OWNER	VARCHAR(128)		シーケンスの所有者。
SEQID	INTEGER		シーケンスの内部 ID。
SEQTYPE	CHAR(1)		シーケンス・タイプ。 S = 通常のシーケンス I = ID シーケンス
INCREMENT	DECIMAL(31,0)		増分値。
START	DECIMAL(31,0)		開始値。
MAXVALUE	DECIMAL(31,0)		最大値。
MINVALUE	DECIMAL(31,0)		最小値。
CYCLE	CHAR(1)		境界に達したときにサイクリングを行うかどうか。 Y - サイクリングを行う N - サイクリングを行わない
CACHE	INTEGER		アクセスを高速化するために、メモリーに事前割り当てするシーケンス値の数。 0 は値が事前割り当てされないことを示します。
ORDER	CHAR(1)		シーケンス番号を要求の順序で生成するかどうか。 Y - 要求の順序でシーケンス番号を生成します。 N - 要求の順序でシーケンス番号を生成しません。
DATATYPEID	INTEGER		組み込みタイプの場合は、組み込みタイプの内部 ID。特殊タイプの場合は、特殊タイプの内部 ID。
SOURCETYPEID	INTEGER		組み込みタイプの場合、この値は 0 になります。特殊タイプの場合は、特殊タイプのソース・タイプである組み込みタイプの内部 ID です。
CREATE_TIME	TIMESTAMP		シーケンスが作成された時刻。
ALTER_TIME	TIMESTAMP		最後の ALTER SEQUENCE ステートメントがこのシーケンスについて実行された時刻。
PRECISION	SMALLINT		シーケンスのデータ型の精度。 SMALLINT では 5、INTEGER では 10、 BIGINT では 19 になります。 DECIMAL では、指定した DECIMAL データ型の精度です。
ORIGIN	CHAR(1)		シーケンス原点。 U - ユーザー生成シーケンス S - システム生成シーケンス
REMARKS	VARCHAR(254)	Yes	ユーザー提供のコメントまたは NULL 値。

SYSCAT.SERVEROPTIONS

各行には、サーバー・レベルの構成オプションが入ります。

表 101. SYSCAT.SERVEROPTIONS カタログ・ビューの列

列名	データ型	NULL 可能	記述
WRAPNAME	VARCHAR(128)	Yes	ラッパー名。
SERVERNAME	VARCHAR(128)	Yes	サーバーの名前。
SERVERTYPE	VARCHAR(30)	Yes	サーバー・タイプ。
SERVERVERSION	VARCHAR(18)	Yes	サーバーのバージョン。
CREATE_TIME	TIMESTAMP		項目が作成された時刻。
OPTION	VARCHAR(128)		サーバー・オプションの名前。
SETTING	VARCHAR(2048)		サーバー・オプションの値。
SERVEROPTIONKEY	VARCHAR(18)		行を固有識別する。
REMARKS	VARCHAR(254)	Yes	ユーザー提供のコメントまたは NULL 値。

SYSCAT.SERVERS

各行はデータ・ソースを表します。このカタログ表の入った同じインスタンスに保管されている表の場合、カタログ項目は不要です。

表 102. SYSCAT.SERVERS カタログ・ビューの列

資料名	データ型	NULL 可能	記述
WRAPNAME	VARCHAR(128)		ラッパー名。
SERVERNAME	VARCHAR(128)		システムに認識されているデータ・ソースの名前。
SERVERTYPE	VARCHAR(30)	Yes	データ・ソースのタイプ (常に大文字)。
SERVERVERSION	VARCHAR(18)	Yes	データ・ソースのバージョン。
REMARKS	VARCHAR(254)	Yes	ユーザー提供のコメントまたは NULL 値。

SYSCAT.STATEMENTS

データベースの各パッケージの中の各 SQL ステートメントごとに、 1 つまたは複数の行が入っています。

表 103. SYSCAT.STATEMENTS カタログ・ビュー

列名	データ型	NULL 可能	説明
PKGSCHEMA	VARCHAR(128)		パッケージの名前。
PKGNAME	CHAR(8)		
UNIQUEID	CHAR(8)		パッケージを最初に作成された時点を示す内部日付/時刻情報。同じ名前を持つ複数のパッケージが存在している場合に、特定のパッケージを識別するのに役立つ。
PKGVERSION	VARCHAR(64)		パッケージのバージョン ID。
STMTNO	INTEGER		アプリケーション・プログラムのソース・モジュールにおける SQL ステートメントの行番号。
SECTNO	SMALLINT		この SQL ステートメントを収めたパッケージ・セクションの番号。
SEQNO	SMALLINT		常に 1。
TEXT	CLOB(64K)		SQL ステートメントのテキスト。

SYSCAT.TABAUTH

データベース内の特定の表またはビューに関する特権を付与されているユーザーまたはグループごとに 1 つまたは複数の行が入っています。特定の認可者から特定の被認可者に付与された 1 つの表またはビューに関するすべての表特権が 1 つの行に示されます。

表 104. SYSCAT.TABAUTH カタログ・ビュー

列名	データ型	NULL 可能	説明
GRANTOR	VARCHAR(128)		特権を付与したユーザーの許可 ID、または SYSIBM。
GRANTEE	VARCHAR(128)		特権を付与されたユーザーまたはグループの許可 ID。
GRANTEETYPE	CHAR(1)		U = GRANTEE は個々のユーザー。 G = GRANTEE はグループ。
TABSCHEMA	VARCHAR(128)		表またはビューの修飾名。
TABNAME	VARCHAR(128)		
CONTROLAUTH	CHAR(1)		GRANTEE に、表またはビューに対する CONTROL 特権があるか否か。 Y = 特権がある。 N = 特権がない。
ALTERAUTH	CHAR(1)		GRANTEE に、表に対する ALTER 特権があるか否か。 Y = 特権がある。 N = 特権がない。 G = 特権があり、付与可能。
DELETEAUTH	CHAR(1)		GRANTEE に、表またはビューに対する DELETE 特権があるか否か。 Y = 特権がある。 N = 特権がない。 G = 特権があり、付与可能。
INDEXAUTH	CHAR(1)		GRANTEE に、表に対する INDEX 特権があるか否か。 Y = 特権がある。 N = 特権がない。 G = 特権があり、付与可能。
INSERTAUTH	CHAR(1)		GRANTEE に、表またはビューに対する INSERT 特権があるか否か。 Y = 特権がある。 N = 特権がない。 G = 特権があり、付与可能。

表 104. SYSCAT.TABAUTH カタログ・ビュー (続き)

列名	データ型	NULL 可能	説明
SELECTAUTH	CHAR(1)		GRANTEE に、表またはビューに対する SELECT 特権があるか否か。 Y = 特権がある。 N = 特権がない。 G = 特権があり、付与可能。
REFAUTH	CHAR(1)		GRANTEE に、表またはビューに対する REFERENCE 特権があるか否か。 Y = 特権がある。 N = 特権がない。 G = 特権があり、付与可能。
UPDATEAUTH	CHAR(1)		GRANTEE に、表またはビューに対する UPDATE 特権があるか否か。 Y = 特権がある。 N = 特権がない。 G = 特権があり、付与可能。

SYSCAT.TABCONST

それぞれの行は、タイプ CHECK、 UNIQUE、PRIMARY KEY、または FOREIGN KEY の表制約を示しています。

表 105. SYSCAT.TABCONST カタログ・ビュー

列名	データ型	NULL 可能	説明
CONSTNAME	VARCHAR(18)		制約の名前 (表内でユニーク)。
TABSCHEMA	VARCHAR(128)		この制約が適用される表の修飾名。
TABNAME	VARCHAR(128)		
DEFINER	VARCHAR(128)		制約の定義時の許可 ID。
TYPE	CHAR(1)		制約の種類。 F = 外部キー I = 関数の従属関係 K = 検査 P = 主キー U = ユニーク
REMARKS	VARCHAR(254)	Yes	ユーザー提供のコメントまたは NULL 値。
ENFORCED	CHAR(1)		Y = 制約を実施 N = 制約を実施しない
CHECKEXISTINGDATA	CHAR(1)		D = 既存のデータの検査を据え置く I = 即時に既存のデータを検査する N = 既存のデータを検査しない
ENABLEQUERYOPT	CHAR(1)		Y = 照会最適化は使用可能 N = 照会最適化は使用不能

SYSCAT.TABDEP

他の特定のオブジェクトへのビューまたはマテリアライズ照会表の従属関係ごとに 1 つの行が入っています。また、このビューに対する特権が、基礎表またはビューに対する特権にどのように従属しているかもエンコードします。 (VIEWDEP カタログ・ビューもまだ (ただし、バージョン 7.1 レベルでのみ) 使用可能です。)

表 106. SYSCAT.TABDEP カタログ・ビュー

列名	データ型	NULL 可能	説明
TABSCHEMA	VARCHAR(128)		基本表への従属関係を持つビューまたはマテリアライズ照会表の名前。
TABNAME	VARCHAR(128)		
DTYPE	CHAR(1)		S = マテリアライズ照会表 V = ビュー (非タイプ付き) W = タイプ・ビュー
DEFINER	VARCHAR(128)	Yes	ビューの作成者の許可 ID。
BTYPE	CHAR(1)		オブジェクト BNAME のタイプ。 A = 別名 F = 関数インスタンス N = ニックネーム O = 表またはビュー階層内のすべての副表またはサブビューに対する特権の従属関係 I = 索引 (基本表への従属関係を記録する場合) R = 構造化タイプ S = マテリアライズ照会表 T = 表 U = タイプ表 V = ビュー W = タイプ・ビュー
BSCHEMA	VARCHAR(128)		ビューが従属しているオブジェクトの修飾名。
BNAME	VARCHAR(128)		
TABAUTH	SMALLINT	Yes	BTYPE= N、O、S、T、U、V、W の場合、この表が依存する基礎表または基礎ビューに対する特権をエンコードします。それ以外の場合は NULL 値。

SYSCAT.TABLES

作成されている表、ビュー、ニックネーム、または別名ごとに 1 つの行が入ります。カタログ表とカタログ・ビューはすべて、SYSCAT.TABLES カタログ・ビューに項目を持っています。

表 107. SYSCAT.TABLES カタログ・ビュー

列名	データ型	NULL 可能	説明
TABSCHEMA	VARCHAR(128)		表、ビュー、ニックネーム、または別名の修飾名。
TABNAME	VARCHAR(128)		
DEFINER	VARCHAR(128)		表、ビュー、ニックネーム、または別名を作成したユーザー。
TYPE	CHAR(1)		オブジェクトの種類。 A = 別名 H = 階層表 N = ニックネーム S = マテリアライズ照会表 T = 表 U = タイプ表 V = ビュー W = タイプ・ビュー
STATUS	CHAR(1)		オブジェクトのチェック・ペンディング状況。 N = 通常の表、ビュー、別名、またはニックネーム C = 表またはニックネームについてのチェック・ペンディング X = 作動不能のビューまたはニックネーム
DROPRULE	CHAR(1)		N = 規則はなし R = ドロップに制約規則が適用される
BASE_TABSCHEMA	VARCHAR(128)	Yes	TYPE=A の場合、これらの列は該当の別名によって参照される表、ビュー、別名、またはニックネームを示し、その他の場合は NULL 値です。
BASE_TABNAME	VARCHAR(128)	Yes	
ROWTYPESCHEMA	VARCHAR(128)	Yes	該当する場合、この表の行タイプの修飾名が入ります。それ以外の場合、NULL 値です。
ROWTYPENAME	VARCHAR(18)		
CREATE_TIME	TIMESTAMP		オブジェクトが作成された時点を示すタイム・スタンプ。
STATS_TIME	TIMESTAMP	Yes	この表に対する何らかの変更が最後に統計に記録された時刻。統計が利用できない場合は、NULL。
COLCOUNT	SMALLINT		表の列の数。
TABLEID	SMALLINT		表の内部 ID。
TBSPACEID	SMALLINT		この表の PRIMARY 表スペースの内部 ID。

表 107. SYSCAT.TABLES カタログ・ビュー (続き)

列名	データ型	NULL 可能	説明
CARD	BIGINT		表の中の行の総数。表階層内の表の場合は、指定されたレベルの階層にある行の数。統計が収集されていないか、あるいは行がビューまたは別名を記述している場合は -1。階層表 (H 表) の場合は -2。
NPAGES	INTEGER		表の行が存在しているページの総数。統計が収集されていないか、この行がビューまたは別名を記述している場合は -1。副表または表階層の場合は -2。
FPAGES	INTEGER		ページの総数。統計が収集されていないか、この行がビューまたは別名を記述している場合は -1。副表または表階層の場合は -2。
OVERFLOW	INTEGER		表のオーバーフロー・レコードの総数。統計が収集されていないか、この行がビューまたは別名を記述している場合は -1。副表または表階層の場合は -2。
TBSPACE	VARCHAR(18)	Yes	表の PRIMARY 表スペースの名前。他の表スペースが指定されていない場合、表のすべての部分がこの表スペースに保管されます。別名およびビューの場合は、NULL 値です。
INDEX_TBSPACE	VARCHAR(18)	Yes	この表に作成されたすべての索引が保管されている表スペースの名前。別名およびビューの場合、または INDEX IN 文節の指定がないか、または INDEX IN 文節に CREATE TABLE ステートメントの IN 文節と同じ値が指定されている場合は、NULL 値。
LONG_TBSPACE	VARCHAR(18)	Yes	この表のすべての長形式データ (LONG または LOB の列タイプ) が入っている表スペースの名前。別名およびビューの場合、または LONG IN 文節が指定されていないか、または LONG IN 文節に CREATE TABLE ステートメントの IN 文節と同じ値が指定されている場合は、NULL 値。
PARENTS	SMALLINT	Yes	この表の親表の数 (この表が従属表になっている参照制約の数)。
CHILDREN	SMALLINT	Yes	この表に従属している表の数 (この表が親表になっている参照制約の数)。
SELFREFS	SMALLINT	Yes	この表に対する自己参照になっている参照制約の数 (この表が親表であり、また従属表でもある参照制約の数)。
KEYCOLUMNS	SMALLINT	Yes	表の主キーを構成する列の数。
KEYINDEXID	SMALLINT	Yes	1 次索引の索引 ID。主キーがない場合は NULL 値または 0。
KEYUNIQUE	SMALLINT		この表に定義されたユニーク制約 (主キー以外) の数。
CHECKCOUNT	SMALLINT		この表に定義されたチェック制約の数。

表 107. SYSCAT.TABLES カタログ・ビュー (続き)

列名	データ型	NULL 可能	説明
DATACAPTURE	CHAR(1)		Y = 表はデータ複製に関与している N = 関与していない L = 表はデータ複製 (LONG VARCHAR および LONG VARGRAPHIC 列の複製を含む) に関与している
CONST_CHECKED	CHAR(32)		バイト 1 は、外部キー制約を表します。バイト 2 は、チェック制約を表します。バイト 5 は、マテリアライズ照会表を表します。バイト 6 は生成される列を表します。バイト 7 は、ステージング表を表します。他のバイトは予約済み。チェックに関する制約情報をエンコードします。値: Y = システムによるチェック U = ユーザーによるチェック N = チェックなし (ペンディング) W = 表がチェック・ペンディング (ペンディング) になったときに 'U' の状態だった F = バイト 5 では、マテリアライズ照会表をインクリメンタル更新できない。バイト 7 では、ステージング表の内容は不完全で、関連したマテリアライズ照会表のインクリメンタル更新に使用することができません。
PMAP_ID	SMALLINT	Yes	この表が使用する区分化マップの ID。別名およびビューの場合は、NULL 値です。
PARTITION_MODE	CHAR(1)		パーティション・データベースの表に使用されるモード。 H = 区分化キーのハッシュ R = データベース・パーティション間で複製された表 別名、ビュー、および区分化キーが定義されていない単一のパーティション・データベースのデータベース・パーティション・グループの表の場合は、ブランク。ニックネームの場合もブランク。
LOG_ATTRIBUTE	CHAR(1)		0 = デフォルトのロギング 1 = 初期にログ記録を取らずに作成された表
PCTFREE	SMALLINT		将来の挿入用に予約されたページのパーセント。ALTER TABLE によって変更可能。
APPEND_MODE	CHAR(1)		ページ上での行の挿入方法。 N = 新規行は使用可能な既存スペースに挿入される Y = 新規行はデータの終わりに追加される 初期値は N。

表 107. SYSCAT.TABLES カタログ・ビュー (続き)

列名	データ型	NULL 可能	説明
REFRESH	CHAR(1)		リフレッシュ・モード。 D = 据え置き I = 即時 O = 1 回 マテリアライズ照会表でなければブランク。
REFRESH_TIME	TIMESTAMP	Yes	REFRESH=D または O の場合、データを最後に最新表示した REFRESH TABLE ステートメントのタイム・スタンプ。それ以外の場合は NULL 値。
LOCKSIZE	CHAR(1)		DML ステートメントがアクセスしたときに優先する表ロックの細分性を示します。表にのみ適用されます。使用できる値は次のとおりです。 R = 行 T = 表 該当しない場合はブランク。 初期値は R。
VOLATILE	CHAR(1)		C = 表のカーディナリティーは揮発性 該当しない場合はブランク。
ROW_FORMAT	CHAR(1)		使用されません。
PROPERTY	VARCHAR(32)		表のプロパティー。単一ブランクは、表にプロパティーがないことを示します。
STATISTICS_PROFILE	CLOB(32K)	Yes	表の統計プロファイルの登録に使用された RUNSTATS コマンド。
COMPRESSION	CHAR(1)		V = 値の圧縮がアクティブで、圧縮をサポートする行フォーマットが使用されている N = 圧縮はなし。圧縮をサポートしない行フォーマットが使用されています。
ACCESS_MODE	CHAR(1)		オブジェクトのアクセス・モード。このアクセス・モードは STATUS フィールドと連結して用いられて、以下の 4 ついずれかの状態を表します。使用できる値は次のとおりです。 N = アクセスはなし (状況値 C に対応) R = 読み取り専用 (状況値 C に対応) D = データ移動はなし (状況値 N に対応) F = 全アクセス権限 (状況値 N に対応)
CLUSTERED	CHAR(1)	Yes	Y = マルチディメンション・クラスタリング (MDC) 表 非 MDC 表の場合は NULL。
ACTIVE_BLOCKS	INTEGER	Yes	MDC 表の中の使用中のブロックの総数。統計が収集されていない場合は -1。
MAXFREESPACESEARCH	SMALLINT		将来の使用のために予約されています。
REMARKS	VARCHAR(254)	Yes	ユーザーが入力したコメント。

SYSCAT.TABLESPACES

表スペースごとに 1 つの行が入っています。

表 108. SYSCAT.TABLESPACES カタログ・ビュー

列名	データ型	NULL 可能	説明
TBSPACE	VARCHAR(18)		表スペースの名前
DEFINER	VARCHAR(128)		表スペースの定義者の許可 ID。
CREATE_TIME	TIMESTAMP		表スペースの作成時刻。
TBSPACEID	INTEGER		表スペースの内部 ID。
TBSPACETYPE	CHAR(1)		表スペースのタイプ。 S = システム管理スペース D = データベース管理スペース
DATATYPE	CHAR(1)		保管できるデータのタイプ。 A = すべてのデータ型の永続データ L = ラージ・データ - 長形式データまたは索引データ T = システム一時表のみ U = 宣言された一時表のみ
EXTENTSIZE	INTEGER		サイズ PAGESIZE をページ単位とするエクステント・サイズ。表スペースの中のコンテナーにこの数のページが書き込まれたら、次のコンテナーに切り替わるようになります。
PREFETCHSIZE	INTEGER		プリフェッチの実行時に読み取るサイズ PAGESIZE のページ数。プリフェッチ・サイズが AUTOMATIC の場合は -1。
OVERHEAD	DOUBLE		コントローラー・オーバーヘッドおよびディスク・シークおよび待ち時間 (ミリ秒単位)。
TRANSFERRATE	DOUBLE		サイズ PAGESIZE から成る 1 ページをバッファーに読み込む時間。
PAGESIZE	INTEGER		表スペースのページのサイズ (バイト単位)。
DBPGNAME	VARCHAR(18)		表スペースのデータベース・パーティション・グループの名前。
BUFFERPOOLID	INTEGER		この表スペースで使用されるバッファー・プールの ID。1 はデフォルトのバッファー・プールを示します。
DROP_RECOVERY	CHAR(1)		N = 表は DROP TABLE ステートメントの後にリカバリー不能 Y = 表は DROP TABLE ステートメントの後にリカバリー可能
REMARKS	VARCHAR(254)	Yes	ユーザーが入力したコメント。

SYSCAT.TABOPTIONS

各行には、リモート表に関連するオプションが入ります。

表 109. SYSCAT.TABOPTIONS カタログ・ビュー

列名	データ型	NULL 可能	記述
TABSCHEMA	VARCHAR(128)		表、ビュー、別名、またはニックネームの修飾名。
TABNAME	VARCHAR(128)		
OPTION	VARCHAR(128)		表、ビュー、別名、またはニックネームのオプション名。
SETTING	VARCHAR(255)		値。

SYSCAT.TBSPACEAUTH

データベース内の特定の表スペースに対する USE 特権を付与されているユーザーまたはグループごとに 1 つの行が入ります。

表 110. SYSCAT.TBSPACEAUTH カタログ・ビュー

列名	データ型	NULL 可能	記述
GRANTOR	CHAR(128)		特権を付与したユーザーの許可 ID、または SYSIBM。
GRANTEE	CHAR(128)		特権を付与されたユーザーまたはグループの許可 ID。
GRANTEETYPE	CHAR(1)		U = GRANTEE は個々のユーザー。 G = GRANTEE はグループ。
TBSPACE	VARCHAR(18)		表スペースの名前
USEAUTH	CHAR(1)		GRANTEE に、表スペースに対する USE 特権があるか否か。 G = 特権があり、付与可能。 N = 特権がない。 Y = 特権がある。

SYSCAT.TRANSFORMS

指名されたトランスフォーム・グループに入っているユーザー定義タイプ内のトランスフォーム関数タイプごとに、 1 つの行が入ります。

表 111. SYSCAT.TRANSFORMS カタログ・ビュー

列名	データ型	NULL 可能	説明
TYPEID	SMALLINT		SYSCAT.DATATYPES で定義されている内部タイプ ID。
TYPESCHEMA	VARCHAR(128)		特定のユーザー定義構造化タイプの修飾名。
TYPENAME	VARCHAR(128)		
GROUPNAME	VARCHAR(128)		トランスフォーム・グループ名。
FUNCID	INTEGER	Yes	SYSCAT.ROUTINES で定義されている、関連するトランスフォーム関数の内部ルーチン ID。 NULL になるのは、内部システム関数の場合だけです。
FUNCSCHEMA	VARCHAR(128)		関連するトランスフォーム関数の修飾名。
FUNCNAME	VARCHAR(128)		
SPECIFICNAME	VARCHAR(128)		関数の特定名 (インスタンス名)。
TRANSFORMTYPE	VARCHAR(8)		'FROM SQL' = トランスフォーム関数は SQL から構造化タイプをトランスフォームする 'TO SQL' = トランスフォーム関数は SQL に構造化タイプをトランスフォームする
FORMAT	CHAR(1)		'U' = ユーザー定義
MAXLENGTH	INTEGER	Yes	FROM SQL トランスフォームからの出力の最大長 (バイト単位)。 TO SQL トランスフォームの場合は NULL。
ORIGIN	CHAR(1)		'O' = オリジナル・トランスフォーム・グループ (ユーザー定義またはシステム定義) 'R' = 再定義
REMARKS	VARCHAR(254)	Yes	ユーザー提供コメントまたは NULL。

SYSCAT.TRIGDEP

他の特定のオブジェクトへのトリガーの従属関係ごとに 1 つの行が入っています。

表 112. SYSCAT.TRIGDEP カタログ・ビュー

列名	データ型	NULL 可能	説明
TRIGSCHEMA	VARCHAR(128)		トリガーの修飾名。
TRIGNAME	VARCHAR(18)		
BTYPE	CHAR(1)		オブジェクト BNAME のタイプ。 A = 別名 B = トリガー F = 関数インスタンス N = ニックネーム O = 表またはビュー階層内のすべての副表またはサブビューに対する特権の従属関係 R = 構造化タイプ S = マテリアライズ照会表 T = 表 U = タイプ表 V = ビュー W = タイプ・ビュー X = 拡張索引
BSCHEMA	VARCHAR(128)		トリガーが従属しているオブジェクトの修飾名。
BNAME	VARCHAR(128)		
TABAUTH	SMALLINT	Yes	BTYPE= O、S、T、U、V、または W の場合、従属索引で必要な表またはビューの特権をエンコードします。それ以外の場合は NULL。

SYSCAT.TRIGGERS

トリガーごとに 1 つの行が入っています。表階層の場合、トリガーはそれぞれ作成された階層レベルでのみ記録されます。

表 113. SYSCAT.TRIGGERS カタログ・ビュー

列名	データ型	NULL 可能	説明
TRIGSCHEMA	VARCHAR(128)		トリガーの修飾名。
TRIGNAME	VARCHAR(18)		
DEFINER	VARCHAR(128)		トリガーの定義時の許可 ID。
TABSCHEMA	VARCHAR(128)		このトリガーを適用される表またはビューの修飾名。
TABNAME	VARCHAR(128)		
TRIGTIME	CHAR(1)		トリガーを起動したイベントに関連して、いつトリガー・アクションが該当の基本表に適用するか。 A = トリガーはイベントの後に適用される B = トリガーはイベントの前に適用される I = トリガーはイベントの代わりに適用される
TRIGEVENT	CHAR(1)		トリガーを起動するイベント。 I = 挿入 D = 削除 U = 更新
GRANULARITY	CHAR(1)		トリガーが実行される単位。 S = ステートメント R = 行
VALID	CHAR(1)		Y = トリガーは有効 X = トリガーは作動不能で、再作成が必要
CREATE_TIME	TIMESTAMP		トリガーが定義された時刻。関数とタイプの解決に使用されます。
QUALIFIER	VARCHAR(128)		オブジェクト定義時のデフォルト・スキーマの値が入ります。
FUNC_PATH	VARCHAR(254)		トリガーの定義された時点の関数パス。関数とタイプの解決に使用されます。
TEXT	CLOB(64K)		入力されたとおりの CREATE TRIGGER ステートメントのテキスト全体。
REMARKS	VARCHAR(254)	Yes	ユーザー提供のコメントまたは NULL 値。

SYSCAT.TYPEMAPPINGS

各行には、リモート組み込みデータ型からローカル組み込みデータ型へのユーザー定義マッピングが入ります。

表 114. SYSCAT.TYPEMAPPINGS カタログ・ビュー

列名	データ型	NULL 可能	記述
TYPE_MAPPING	VARCHAR(18)		タイプ・マッピングの名前 (システム生成の場合もある)。
TYPESCHEMA	VARCHAR(128)	Yes	タイプのスキーマ名。システム組み込みタイプの場合は NULL 値。
TYPENAME	VARCHAR(18)		データ型マッピングでのローカル・タイプの名前。
TYPEID	SMALLINT		タイプ ID。
SOURCETYPEID	SMALLINT		ソース・タイプ ID。
DEFINER	VARCHAR(128)		このタイプ・マッピングの作成に使用された許可 ID。
LENGTH	INTEGER	Yes	データ型の最大長または精度。 NULL 値の場合、システムは最善の長さ/精度を判別します。
SCALE	SMALLINT	Yes	DECIMAL フィールドの位取り。 NULL 値の場合、システムは最善の位取りの属性を判別します。
BIT_DATA	CHAR(1)	Yes	Y = 型はビット・データ用。 N = 型はビット・データ用ではない。 NULL = 文字データ型ではなく、システムはビット・データ属性を判別しない。
WRAPNAME	VARCHAR(128)	Yes	このラッパーにマッピングが適用されます。
SERVERNAME	VARCHAR(128)	Yes	データ・ソースの名前。
SERVERTYPE	VARCHAR(30)	Yes	このタイプのデータ・ソースにマッピングが適用されます。
SERVERVERSION	VARCHAR(18)	Yes	SERVERTYPE に指定されたタイプをもつこのバージョンのデータ・ソースにマッピングが適用されます。
REMOTE_TYPESCHEMA	VARCHAR(128)	Yes	リモート・タイプのスキーマ名。
REMOTE_TYPENAME	VARCHAR(128)		データ・ソースに対して定義されたデータ型の名前。
REMOTE_META_TYPE	CHAR(1)	Yes	S = リモート・タイプはシステム組み込みタイプ。 T = リモート・タイプは特殊タイプ。
REMOTE_LOWER_LEN	INTEGER	Yes	リモート 10 進数タイプの長さまたは精度の下限。文字データ型の場合、このフィールドは文字数を示します。 -1 は、デフォルトの長さまたは精度が使われることを示すか、またはリモート型は長さも精度ももっていないことを示します。

表 114. SYSCAT.TYPEMAPPINGS カタログ・ビュー (続き)

列名	データ型	NULL 可能	記述
REMOTE_UPPER_LEN	INTEGER	Yes	リモート 10 進数タイプの長さまたは精度の上限。文字データ型の場合、このフィールドは文字数を示します。 -1 は、デフォルトの長さまたは精度が使われることを示すか、またはリモート型は長さも精度ももっていないことを示します。
REMOTE_LOWER_SCALE	SMALLINT	Yes	リモート・タイプの位取りの下限。
REMOTE_UPPER_SCALE	SMALLINT	Yes	リモート・タイプの位取りの上限。
REMOTE_S_OPR_P	CHAR(2)	Yes	リモートの位取りとリモートの精度の関係。基本比較演算子を使用できます。 NULL であれば、特定の関係は不要です。
REMOTE_BIT_DATA	CHAR(1)	Yes	Y = 型はビット・データ用。 N = 型はビット・データ用ではない。 NULL = これは文字データ型ではなく、システムはビット・データ属性を判別しない。
USER_DEFINED	CHAR(1)		ユーザーが提供する定義。
CREATE_TIME	TIMESTAMP		マッピングが作成された時刻。
REMARKS	VARCHAR(254)	Yes	ユーザー提供のコメントまたは NULL 値。

SYSCAT.USEROPTIONS

各行には、サーバー固有のオプション値が入ります。

表 115. SYSCAT.USEROPTIONS カタログ・ビュー

列名	データ型	NULL 可能	記述
AUTHID	VARCHAR(128)		ローカル許可 ID (常に大文字)。
SERVERNAME	VARCHAR(128)		ユーザーが定義されたサーバーの名前。
OPTION	VARCHAR(128)		ユーザー・オプションの名前。
SETTING	VARCHAR(255)		値。

SYSCAT.VIEWS

作成されたビューごとに 1 つまたは複数の行が入っています。

表 116. SYSCAT.VIEWS カタログ・ビュー

列名	データ型	NULL 可能	説明
VIEWSCHEMA	VARCHAR(128)		マテリアライズ照会表またはステージング表の定義に使用されるビューまたは表の修飾名。
VIEWNAME	VARCHAR(128)		
DEFINER	VARCHAR(128)		ビューの作成者の許可 ID。
SEQNO	SMALLINT		常に 1。
VIEWCHECK	CHAR(1)		ビューの検査のタイプ。 N = 検査オプションはない L = ローカル検査オプション C = カスケード検査オプション
READONLY	CHAR(1)		Y = ビューはその定義により読み取り専用 N = ビューは読み取り専用ではない
VALID	CHAR(1)		Y = ビューまたはマテリアライズ照会表の定義が有効である。 X = ビューまたはマテリアライズ照会表の定義が作動不能である。再作成が必要。
QUALIFIER	VARCHAR(128)		オブジェクト定義時のデフォルト・スキーマの値が入ります。
FUNC_PATH	VARCHAR(254)		ビュー作成時のビュー作成者の SQL パス。ビューをデータ操作ステートメントで使用する場合、ビューにおける関数呼び出しを解決するのにこのパスを使用する必要があります。バージョン 2 より前に作成されたビューの場合は SYSIBM です。
TEXT	CLOB(64k)		CREATE VIEW ステートメントのテキスト。

SYSCAT.WRAPOPTIONS

各行には、ラッパー固有のオプションが入ります。

表 117. SYSCAT.WRAPOPTIONS カタログ・ビュー

列名	データ型	NULL 可能	記述
WRAPNAME	VARCHAR(128)		ラッパー名。
OPTION	VARCHAR(128)		ラッパー・オプションの名前。
SETTING	VARCHAR(255)		値。

SYSCAT.WRAPPERS

各行には、登録済みラッパーに関する情報が入ります。

表 118. SYSCAT.WRAPPERS カタログ・ビュー

列名	データ型	NULL 可能	記述
WRAPNAME	VARCHAR(128)		ラッパー名。
WRAPTYPE	CHAR(1)		N = 非リレーショナル R = リレーショナル
WRAPVERSION	INTEGER		ラッパーのバージョン。
LIBRARY	VARCHAR(255)		このラッパーに関連したデータ・ソースとの通信に使用するコードが入っているファイルの名前。
REMARKS	VARCHAR(254)	Yes	ユーザー提供のコメントまたは NULL 値。

SYSSTAT.COLDIST

各行は、列の N 番目の最大頻度または N 番目の変位値を記述しています。タイプ表の継承列の場合、統計は記録されません。

表 119. SYSSTAT.COLDIST カタログ・ビュー

列名	データ・タイプ	NULL 可能	記述	更新可能
TABSCHEMA	VARCHAR(128)		この項目が適用される表の修飾名。	
TABNAME	VARCHAR(128)			
COLNAME	VARCHAR(128)		この項目が適用される列の名前。	
TYPE	CHAR(1)		収集する統計のタイプ。 F = 頻度 (最大頻度) Q = 変位値	
SEQNO	SMALLINT		TYPE=F の場合、この列の N は N 番目の最大頻度。 TYPE=Q の場合、この列の N は N 番目の変位値。	
COLVALUE	VARCHAR(254)	Yes	データ値 (文字リテラルまたは NULL 値)。注 1 を参照。 この列は、統計に関連している列に対応する値の有効な表記を用いて更新可能です。必要な頻出度が NULL 値の場合は、この列を NULL に設定する必要があります。	Yes
VALCOUNT	BIGINT		TYPE=F の場合、VALCOUNT は、その列の中の COLVALUE の出現回数。 TYPE=Q の場合、VALCOUNT は、値が COLVALUE 以下の行の数。 この列は、次の値でのみ更新可能です。 • >= 0 (ゼロ)	Yes
DISTCOUNT	BIGINT		TYPE=q の場合、この列は COLVALUE に等しいかより小さい特殊値の数 (入手不能の場合は NULL) を記録します。値が COLVALUE より小さいか等しい行の数。	Yes

表 119. SYSSTAT.COLDIST カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	記述	更新可能

注:

1. カタログ・ビュー内の COLVALUE の値は常にデータベース・コード・ページ中に示されますが、これには置換文字を収めることができます。ただし、列の表のコード・ページ内で内部的に統計が収集されるので、照会の最適化時に適用するときは実際の列値が使用されます。

   UPDATE ステートメントを使って値を変更する場合、データベース・コード・ページ内の文字のみを使用しなければなりません。そうしないと、置換文字が発生し、これが照会の最適化中の統計の適用において使用されます。

SYSSTAT.COLUMNS

統計を更新できる列ごとに 1 つの行が入っています。タイプ表の継承列の場合、統計は記録されません。

表 120. SYSSTAT.COLUMNS カタログ・ビュー

列名	データ・タイプ	NULL 可能	記述	更新可能
TABSCHEMA	VARCHAR(128)		列の入った表の修飾名。	
TABNAME	VARCHAR(128)			
COLNAME	VARCHAR(128)		列名。	
COLCARD	BIGINT		列の値の種類数。統計が収集されていない場合は -1。継承列および階層表の列の場合は -2。 どのような列でも、COLCARD は、その列の入った表のカーディナリティーを超える値になることはありません。 この列は、次の値でのみ更新可能です。 • -1 または >= 0 (ゼロ)	Yes
HIGH2KEY	VARCHAR(33)	Yes	列の 2 番目の最高値。継承列および階層表の列の統計が収集されていない場合、このフィールドは空です。 この列は、統計に関連している列に適した値の有効な表記を用いて更新することができます。 注 1 および 2 を参照。	Yes
LOW2KEY	VARCHAR(33)	Yes	列の 2 番目の最低値。継承列および階層表の列の統計が収集されていない場合、このフィールドは空です。 この列は、統計に関連している列に適した値の有効な表記を用いて更新することができます。 注 1 および 2 を参照。	Yes

表 120. SYSSTAT.COLUMNS カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	記述	更新可能
AVGCOLLEN	INTEGER		列の平均の長さ。長形式フィールドまたは LOB の場合、または統計が収集されていない場合は -1。継承列および階層表の列の場合は -2。 この列は、次の値でのみ更新可能です。 • -1 または >= 0 (ゼロ)	Yes
NUMNULLS	BIGINT		列の NULL 値の数が入っています。統計が収集されていない場合は -1。 この列は、次の値でのみ更新可能です。 • -1 または >= 0 (ゼロ)	Yes
SUB_COUNT	SMALLINT		サブエレメントの平均数。文字カラムにのみ適用可能。たとえば、 'database simulation analytical business intelligence' というストリングについて考えてみます。この例では、ストリング内に 5 個のサブエレメントがあるため、SUB_COUNT = 5 です。	Yes
SUB_DELIM_ LENGTH	SMALLINT		各サブエレメントを分ける各区切り文字の平均長。文字列にのみ該当します。たとえば、 'database simulation analytical business intelligence' というストリングについて考えてみます。この例では、各区切り文字は単一のブランクなので、SUB_DELIM_LENGTH = 1 です。	Yes

表 120. SYSSTAT.COLUMNS カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	記述	更新可能

注:

1. HIGH2KEY と LOW2KEY はそれぞれ、列内の 2 番目に大きい値と 2 番目に小さい値の ASCII バージョンを表します。整数値の ASCII 表現は、符号も入れて 1 から 11 文字の長さになります。 HIGH2KEY 値と LOW2KEY 値は、UPDATE ステートメントの SET 文節で使用できるように保管されます。文字ストリングの場合、これはすなわち、ストリングの前後に単一引用符が付けられて、ストリング内にすでにある各単一引用符の前にさらに単一引用符が挿入されるということです。列の値の最初の 33 文字だけが、検討対象になります。

   HIGH2KEY 値と LOW2KEY 値に手動でデータを追加する予定の場合、以下を確認してください。

   - HIGH2KEY と LOW2KEY は、それに対応するユーザー列と同じデータ型の有効値である。
   - HIGH2KEY 値と LOW2KEY 値の長さは、ターゲット列のデータ型の最大長か、またはストリングの長さを最大 68 に増加するための、追加の単一引用符を除く 33 のどちらか小さいほうでなければならない。
   - 対応する列に 3 つ以上の値が存在する場合は、常に HIGH2KEY のほうが LOW2KEY より大きくなければならない。列内に存在する値が 3 つ未満の場合は、HIGH2KEY と LOW2KEY が等しくてもかまいません。

2. カタログ・ビューでは、HIGH2KEY と LOW2KEY の値は常にデータベース・コード・ページ中に示されますが、これには置換文字を収めることができます。ただし、列の表のコード・ページ内で内部的に統計が収集されるので、照会の最適化時に適用するときは実際の列値が使用されます。

   UPDATE ステートメントを使って値を変更する場合、データベース・コード・ページ内の文字のみを使用しなければなりません。そうしないと、置換文字が発生し、これが照会の最適化中の統計の適用において使用されます。

SYSSTAT.INDEXES

表に定義されている各索引ごとに 1 つの行が入ります。

表 121. SYSSTAT.INDEXES カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明	更新可能
INDSCHEMA	VARCHAR(128)		索引の修飾名。	
INDNAME	VARCHAR(18)			
TABSCHEMA	VARCHAR(128)		表名の修飾子。	
TABNAME	VARCHAR(128)		索引が定義されている表またはニックネームの名前。	
COLNAMES	CLOB(1M)		+ または - の接頭部が付いた列名のリスト。	
NLEAF	INTEGER		リーフ・ページの数。統計が収集されていない場合は -1。 この列は、以下の値でのみ更新可能です。 • -1 または > 0 (ゼロ)	Yes
NLEVELS	SMALLINT		索引レベルの数。統計が収集されていない場合は -1。 この列は、以下の値でのみ更新可能です。 • -1 または > 0 (ゼロ)	Yes
FIRSTKEYCARD	BIGINT		最初のキーの値の種類数。統計が収集されていない場合は -1。 この列は、以下の値でのみ更新可能です。 • -1 または >= 0 (ゼロ)	Yes
FIRST2KEYCARD	BIGINT		索引の最初の 2 つの列を使用するキーの種類数 (統計がない場合、または適用されない場合は -1)。 この列は、以下の値でのみ更新可能です。 • -1 または >= 0 (ゼロ)	Yes
FIRST3KEYCARD	BIGINT		索引の最初の 3 つの列を使用するキーの種類数 (統計がない場合、または適用されない場合は -1)。 この列は、以下の値でのみ更新可能です。 • -1 または >= 0 (ゼロ)	Yes
FIRST4KEYCARD	BIGINT		索引の最初の 4 つの列を使用するキーの種類数 (統計がない場合、または適用されない場合は -1)。 この列は、以下の値でのみ更新可能です。 • -1 または >= 0 (ゼロ)	Yes

表 121. SYSSTAT.INDEXES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明	更新可能
FULLKEYCARD	BIGINT		個別の全キー値の数。統計が収集されていない場合は -1。 この列は、以下の値でのみ更新可能です。 • -1 または >= 0 (ゼロ)	Yes
CLUSTERRATIO	SMALLINT		この列は、オプティマイザーにより使用されます。索引のデータ・クラスタリングの程度を示し、統計が収集されていない場合、または詳細な索引統計が収集されている場合は -1。 この列は、以下の値でのみ更新可能です。 • -1 または 0 から 100	Yes
CLUSTERFACTOR	DOUBLE		この列は、オプティマイザーにより使用されます。クラスタリングの程度の詳細測定値。詳細索引統計が収集されていない場合は -1。 この列は、以下の値でのみ更新可能です。 • -1 または 0 から 1	Yes
SEQUENTIAL_PAGES	INTEGER		索引キーの順序でディスクに存在し、それらの間に大きなギャップがなく、わずかなギャップしかないリーフ・ページの数。統計が使用できない場合は -1。 この列は、以下の値でのみ更新可能です。 • -1 または >= 0 (ゼロ)	Yes
DENSITY	INTEGER		索引によって占有されているページの範囲内の、ページ数に対する SEQUENTIAL_PAGES の比率。パーセントで表現される (0 から 100 の整数。統計が入手できない場合は -1。) この列は、以下の値でのみ更新可能です。 • -1 または 0 から 100	Yes

表 121. SYSSTAT.INDEXES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明	更新可能
PAGE_FETCH_PAIRS	VARCHAR(254)		文字形式で表された整数ペアのリスト。それぞれのペアは、仮のバッファーのページ数と、その仮のバッファーを使用した索引のスキャンに必要なページ取り出しの回数を表しています。 (データが入手できない場合は、長さ 0 のストリング。) この列は、以下の入力値でのみ更新できます。 • 対の区切り文字および対の区切り記号は、数字以外の文字だけが受け入れられます。 • ブランクは、対の区切り文字および対の区切り記号として認識される唯一の文字です。 • 各数値項目には、対応するパートナーの数値項目がなければならず、それら 2 つは対区切り記号で区切る必要があります。 • 対と対の間は、対区切り文字で区切る必要があります。 • 予期される数値項目は、0 から 9 の正の数でなければなりません。	Yes
NUMRIDS	BIGINT		索引の中の RID の数。統計が収集されていない場合は -1。	Yes
NUMRIDS_DELETED	BIGINT		削除済みとしてマークされた索引の中の RID の数 (ただし、すべての RID が削除済みとしてマークされたページの RID は除く)。統計が収集されていない場合は -1。	Yes
NUM_EMPTY_LEAFS	BIGINT		すべての RID が削除済みとしてマークされた索引内のリーフ・ページの数。統計が収集されていない場合は -1。	Yes
AVERAGE_RANDOM_FETCH_PAGES	DOUBLE		索引を使用してフェッチする際の、順次ページ・アクセス間のランダム表ページの平均数。不明の場合は -1。注 1 および 2 を参照。	
AVERAGE_RANDOM_PAGES	DOUBLE		順次索引ページ・アクセス間のランダム索引ページの平均数。不明の場合は -1。注 2 を参照。	
AVERAGE_SEQUENCE_GAP	DOUBLE		索引ページ・シーケンス間のギャップ。各ギャップは索引リーフ・ページのスキャンにより検出され、一連の索引ページの間でランダムにフェッチしなければならない索引ページの平均数を表します。不明の場合は -1。注 2 を参照。	

表 121. SYSSTAT.INDEXES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明	更新可能
AVERAGE_SEQUENCE_FETCH_GAP	DOUBLE		索引を使用してフェッチする際の、表ページ・シーケンス間のギャップ。各ギャップは索引リーフ・ページのスキャンにより検出され、一連の表ページの間でランダムにフェッチしなければならない表ページの平均数を表します。不明の場合は -1。注 1 および 2 を参照。	
AVERAGE_SEQUENCE_PAGES	DOUBLE		順次にアクセス可能な索引ページの平均数 (つまり、プリフェッチャーが順次として検出する索引ページの数)。不明の場合は -1。注 2 を参照。	
AVERAGE_SEQUENCE_FETCH_PAGES	DOUBLE		索引を使用してフェッチする際の、順次にアクセス可能な表ページの平均数 (つまり、プリフェッチャーが順次として検出する表ページの数)。不明の場合は -1。注 1 および 2 を参照。	

注:

1. DMS 表スペースの使用時には、この統計は計算されません。
2. LOAD...STATISTICS YES または CREATE INDEX...COLLECT STATISTICS の操作時や、データベース構成パラメーター *seqdetect* がオフになっているときは、プリフェッチ統計は集められません。

SYSSTAT.ROUTINES

ユーザー定義関数 (スカラー、表、またはソース)、システム生成メソッド、ユーザー定義メソッド、またはプロシージャーごとに、1 つの行が入ります。組み込み関数は組み入れられません。 (SYSSTAT.FUNCTIONS はこのカタログ・ビューによって置き換えられています。他のビューも存在しますが、DB2 バージョン 7.1 におけるのと同じ状態のままです。)

表 122. SYSSTAT.ROUTINES カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明	更新可能
ROUTINESCHEMA	VARCHAR(128)		ルーチンの修飾名。	
ROUTINENAME	VARCHAR(128)			
ROUTINETYPE	CHAR(1)		F = 関数 M = メソッド P = プロシージャー	
SPECIFICNAME	VARCHAR(128)		ルーチン・インスタンスの名前 (システム生成の場合もある)。	
IOS_PER_INVOC	DOUBLE		呼び出しごとの入出力回数の見積もり。不明の場合は -1 (デフォルト値は 0)。この列は、-1 または >= 0 (ゼロ) でのみ更新できます。	Yes
INSTS_PER_INVOC	DOUBLE		呼び出しごとの命令の数の見積もり。不明の場合は -1 (デフォルト値は 450)。この列は、-1 または >= 0 (ゼロ) でのみ更新できます。	Yes
IOS_PER_ARGBYTE	DOUBLE		入力引き数 1 バイトごとの入出力回数の見積もり。不明の場合は -1 (デフォルト値は 0)。この列は、-1 または >= 0 (ゼロ) でのみ更新できます。	Yes
INSTS_PER_ARGBYTE	DOUBLE		入力引き数 1 バイトごとの命令数の見積もり。不明の場合は -1 (デフォルト値は 0)。この列は、-1 または >= 0 (ゼロ) でのみ更新できます。	Yes
PERCENT_ARGBYTES	SMALLINT		ルーチンが実際に読み取る入力引き数バイトの平均パーセント値の見積もり。不明の場合は -1 (デフォルト値は 100)。この列は、-1 か、あるいは 0 (ゼロ) と 100 の間の数でのみ更新できます。	Yes
INITIAL_IOS	DOUBLE		ルーチンが最初/最後に呼び出されたときに実行される入出力回数の見積もり。不明の場合は -1 (デフォルト値は 0)。この列は、-1 または >= 0 (ゼロ) でのみ更新できます。	Yes
INITIAL_INSTS	DOUBLE		最初/最後のルーチン呼び出し時に実行される命令の数の見積もり。不明の場合は -1 (デフォルト値は 0)。この列は、-1 または >= 0 (ゼロ) でのみ更新できます。	Yes

表 122. SYSSTAT.ROUTINES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明	更新可能
CARDINALITY	BIGINT		表関数の予測されるカーディナリティー。不明の場合、またはルーチンが表関数でない場合は -1。この列は、-1 または >= 0 (ゼロ) でのみ更新できます。	Yes
SELECTIVITY	DOUBLE		ユーザー定義述部に使用される。ユーザー定義述部がない場合は -1 です。注 1 を参照。	

注:

1. どのようなユーザー定義関数でも、システム・カタログでの DB2 バージョン 5.2 から 8.1 への移行中は、この列は -1 に設定されます。ユーザー定義述部の場合、システム・カタログでの選択性は -1 になります。この場合、オプティマイザーが使用する選択性の値は 0.01 です。

SYSSTAT.TABLES

基本 表ごとに 1 つの行が入っています。ビューまたは別名は組み入れられません。タイプ表の場合、このビューに示されるのは表階層のルート表だけです。タイプ表の継承列の場合、統計は記録されません。 CARD 値は、他の統計が表階層全体に適用される場合にのみ、ルート表に適用されます。

表 123. SYSSTAT.TABLES カタログ・ビュー

列名	データ・タイプ	NULL 可能	記述	更新可能
TABSCHEMA	VARCHAR(128)		表の修飾名。	
TABNAME	VARCHAR(128)			
CARD	BIGINT		表の中の行の総数。統計が収集されていない場合は -1。表の CARD を更新する場合、その表の列のいずれかの COLCARD の値よりも小さい値を割り当ててはなりません。値 -2 は変更できません。また、列の値は直接 -2 に設定できません。この列は、次の値でのみ更新可能です。 • -1 または >= 0 (ゼロ)	Yes
NPAGES	INTEGER		表の行が存在しているページの総数。統計が収集されていない場合は -1。副表および階層表の場合は -2。値 -2 は変更できません。また、列の値は直接 -2 に設定できません。この列は、次の値でのみ更新可能です。 • -1 または >= 0 (ゼロ)	Yes
FPAGES	INTEGER		ファイルのページの総数。統計が収集されていない場合は -1。副表および階層表の場合は -2。値 -2 は変更できません。また、列の値は直接 -2 に設定できません。この列は、次の値でのみ更新可能です。 • -1 または > 0 (ゼロ)	Yes
OVERFLOW	INTEGER		表のオーバーフロー・レコードの総数。統計が収集されていない場合は -1。副表および階層表の場合は -2。値 -2 は変更できません。また、列の値は直接 -2 に設定できません。この列は、次の値でのみ更新可能です。 • -1 または >= 0 (ゼロ)	Yes
CLUSTERED	CHAR(1)	Yes	Y = テーブルはマルチディメンションに分けてクラスター化される (1 つしかディメンションがない場合でも)。 NULL = テーブルはクラスター化されない。	

表 123. SYSSTAT.TABLES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	記述	更新可能
ACTIVE_ BLOCKS	INTEGER		マルチディメンション・クラスタリング (MDC) 表の中の使用中のブロックの総数。統計が収集されていない場合は -1。	Yes

付録 E. フェデレーテッド・システム

SQL ステートメントで有効なサーバーのタイプ

サーバー定義がどのような種類のデータ・ソースを表すかは、サーバー・タイプによって示されます。サーバー・タイプは、ベンダー別、目的別、またオペレーティング・システム別に異なります。サポートされる値は使用するラッパーで異なります。

大半のデータ・ソースの場合、有効なサーバー・タイプを CREATE SERVER ステートメントに指定する必要があります。

BioRS ラッパー

BioRS データ・ソース

サーバー・タイプ	データ・ソース
CREATE SERVER ステートメントでは不要	BioRS

BLAST ラッパー

BLAST デーモンでサポートされている BLAST データ・ソース。

サーバー・タイプ	データ・ソース
BLASTN	塩基配列データベースの内容に対して塩基配列を比較して、元の配列領域に一致する領域をもった配列を見つけるための BLAST 検索。
BLASTP	アミノ酸配列データベースの内容に対してアミノ酸配列を比較して、元の配列領域に一致する領域をもった配列を見つけるための BLAST 検索。
BLASTX	アミノ酸配列データベースの内容に対して塩基配列を比較して、元の配列領域に一致する領域をもった配列を見つけるための BLAST 検索。
TBLASTN	塩基配列データベースの内容に対してアミノ酸配列を比較して、元の配列領域に一致する領域をもった配列を見つけるための BLAST 検索。
TBLASTX	塩基配列データベースの内容に対して塩基配列を比較して、元の配列領域に一致する領域をもった配列を見つけるための BLAST 検索。

CTLIB ラッパー

CTLIB Client Software でサポートされる Sybase データ・ソース

サーバー・タイプ	データ・ソース
SYBASE	Sybase

Documentum ラッパー

Documentum Client API/Library でサポートされる Documentum データ・ソース。

サーバー・タイプ	データ・ソース
DCTM	Documentum

DRDA ラッパー

DB2 ファミリー・データ・ソース

表 124. DB2 for Linux、DB2 for UNIX、および DB2 for Windows

サーバー・タイプ	データ・ソース
DB2/UDB	IBM DB2 Universal Database
DB2/6000	IBM DB2 for AIX
DB2/AIX	IBM DB2 for AIX
DB2/HPUX	IBM DB2 for HP-UX
DB2/HP	IBM DB2 for HP-UX
DB2/NT	IBM DB2 for Windows NT
DB2/EEE	IBM DB2 エンタープライズ - 拡張エディション
DB2/SUN	IBM DB2 for Solaris
DB2/PE	IBM DB2 for Personal Edition
DB2/2	IBM DB2 for OS/2
DB2/LINUX	IBM DB2 for Linux
DB2/PTX	IBM DB2 for NUMA-Q
DB2/SCO	IBM DB2 for SCO Unixware

表 125. DB2 for iSeries および IBM DB2 for AS/400

サーバー・タイプ	データ・ソース
DB2/400	IBM DB2 for iSeries および IBM DB2 for AS/400

表 126. DB2 for z/OS and OS/390

サーバー・タイプ	データ・ソース
DB2/ZOS	IBM DB2 for z/OS
DB2/390	IBM DB2 for OS/390

表 126. DB2 for z/OS and OS/390 (続き)

サーバー・タイプ	データ・ソース
DB2/MVS	IBM DB2 for MVS

表 127. DB2 Server for VM および DB2 Server for VSE

サーバー・タイプ	データ・ソース
DB2/VM	IBM DB2 for VM
DB2/VSE	IBM DB2 for VSE
SQL/DS	IBM SQL/DS

Entrez ラッパー

Entrez データ・ソース。

サーバー・タイプ	データ・ソース
NUCLEOTIDE	Entrez
PUBMED	Entrez

Excel ラッパー

Microsoft Excel 97、2000、および 2002 でサポートされる Excel データ・ソース。

サーバー・タイプ	データ・ソース
CREATE SERVER ステートメントでは不要	Microsoft Excel

Extended Search ラッパー

Extended Search Client Library でサポートされる Extended Search データ・ソース。

サーバー・タイプ	データ・ソース
CREATE SERVER ステートメントでは不要	IBM Lotus Extended Search

HMMER ラッパー

HMMER デーモンでサポートされている HMMER データ・ソース。

サーバー・タイプ	データ・ソース
PFAM	HMMER
SEARCH	HMMER

Informix ラッパー

Informix Client SDK ソフトウェアでサポートされる Informix データ・ソース

サーバー・タイプ	データ・ソース
INFORMIX	Informix

MSSQLODBC3 ラッパー

DataDirect Connect ODBC 3.6 ドライバーまたは ODBC 3.0 以上のドライバーでサポートされる Microsoft SQL Server データ・ソース

サーバー・タイプ	データ・ソース
MSSQLSERVER	Microsoft SQL Server

NET8 ラッパー

Oracle NET8 Client Software でサポートされる Oracle データ・ソース

サーバー・タイプ	データ・ソース
ORACLE	Oracle バージョン 8.0. またはこれ以上

ODBC ラッパー

ODBC 3.x ドライバーでサポートされる ODBC データ・ソース。

サーバー・タイプ	データ・ソース
ODBC	ODBC

OLE DB ラッパー

Microsoft OLE DB 2.0 以上に準拠する OLE DB Provider。

サーバー・タイプ	データ・ソース
CREATE SERVER ステートメントでは不要	任意の OLE DB provider

表構造ファイル・ラッパー

表構造ファイル・データ・ソース。

サーバー・タイプ	データ・ソース
CREATE SERVER ステートメントでは不要	表構造ファイル

Teradata ラッパー

Teradata V2R3、V2R4、および V2R5 クライアント・ソフトウェアでサポートされる Teradata データ・ソース

サーバー・タイプ	データ・ソース
TERADATA	Teradata

Web サービス・ラッパー

Web サービス・データ・ソース。

サーバー・タイプ	データ・ソース
CREATE SERVER ステートメントでは不要	任意の Web サービス・データ・ソース

WebSphere Business Integration ラッパー

WeSphere Business Integration ラッパーでサポートされるビジネス・アプリケーションのデータ・ソース。

サーバー・タイプ	データ・ソース
WBI	WebSphere Business Integration 2.2 または 2.3

XML ラッパー

XML データ・ソース

サーバー・タイプ	データ・ソース
CREATE SERVER ステートメントでは不要	XML

フェデレーテッド・システムのニックネーム列

CREATE NICKNAME または ALTER NICKNAME ステートメントで列情報を指定するには、ニックネーム列オプションというパラメーターを使用します。

以下の表は、各データ・ソースごとのニックネーム列オプションを一覧で示しています。

表 128. 選択可能なニックネーム列オプション

データ・ソース	ALL_VALUES	DEFAULT	DELIMITER	DOCUMENT	ESCAPE_INPUT	FOREIGN_KEY	INDEX	IS_REPEATING	NUMERIC_STRING	PRIMARY_KEY	REMOTE_NAME	SOAPACTIONCOLUMN	TEMPLATE	URLCOLUMN	VARCHAR_NO_TRAILING_ BLANKS	XPATH
BLAST		X	X				X									
DB2 Universal Database for iSeries									X							

表 128. 選択可能なニックネーム列オプション (続き)

データ・ソース	ALL_VALUES	DEFAULT	DELIMITER	DOCUMENT	ESCAPE_INPUT	FOREIGN_KEY	INDEX	IS_REPEATING	NUMERIC_STRING	PRIMARY_KEY	REMOTE_NAME	SOAPACTIONCOLUMN	TEMPLATE	URLCOLUMN	VARCHAR_NO_TRAILING_ BLANKS	XPATH
DB2 Universal Database for z/OS and OS/390									X							
DB2 Universal Database for VM and VSE									X							
DB2 Universal Database for Linux、UNIX、Windows									X							
Documentum	X		X					X			X					
Informix									X							
Microsoft SQL Server									X							
ODBC									X							
OLE DB																
Oracle									X						X	
Sybase									X							
表構造ファイル				X												
Teradata									X							
WebSphere Business Integration					X	X				X			X			X
Web サービス					X	X				X		X	X	X		X
XML				X		X				X						X

表 129. ニックネーム列オプションおよびその設定値

オプション	説明および有効設定値	デフォルトの設定値
ALL_VALUES	反復属性の値はすべて、指定の区切り文字で区切られて戻されることを指定します。このオプションが欠落しているか、または N であると、反復属性の最後の値だけが戻されます。 ALL_VALUES を指定できるのは、IS_REPEATING オプションが 'Y' である VARCHAR 列に対してのみです (しかも IS_REG_TABLE = 'Y' の場合に有効)。	

表 129. ニックネーム列オプションおよびその設定値 (続き)

オプション	説明および有効設定値	デフォルトの設定値
DEFAULT	以下の入力固定列の新しいデフォルト値を指定します。 • E_value • QueryStrands • GapAlign • NMisMatchPenalty • NMatchReward • Matrix • FilterSequence • NumberOfAlignments • GapCost • ExtendedGapCost • WordSize • ThresholdEx この新しい値によって、事前設定のデフォルト値はオーバーライドされます。新しいデフォルト値は、該当する列で指示された値と同じ型でなければなりません。	
DELIMITER	Documentum の場合: 反復属性の複数の値を連結するのに使う区切り文字ストリングを指定します。区切り文字には 1 つ以上の文字を使用することができます。このオプションが有効なのは、IS_REPEATING オプションを Y に設定されて VARCHAR データ型をもつオブジェクトの属性の場合のみです。 BLAST の場合: このオプションが置かれている列の定義行情報のエンドポイントの指定に使う区切り文字。このオプション値内で複数の文字が使われている場合、その中のいずれか 1 つの文字の最初の出現箇所がこのフィールドの情報の末尾のしるしになります。デフォルトは行の終わりです。指定した最後の列に定義行の残りを入れたいのでないかぎり、このオプションは必須です。	Documentum の場合: デフォルトの区切り文字はコンマです。 BLAST の場合: デフォルトの区切り文字は行の終わりです。

表 129. ニックネーム列オプションおよびその設定値 (続き)

オプション	説明および有効設定値	デフォルトの設定値
DOCUMENT	表構造ファイルの場合: 表構造ファイルの種類を指定します。このオプションの場合、値 FILE だけがこのラッパーでサポートされています。ニックネームごとに DOCUMENT オプションで指定できる列は 1 つだけです。DOCUMENT オプションに関連付ける列は、VARCHAR または CHAR のデータ型でなければなりません。 FILE_PATH ニックネーム・オプションではなく DOCUMENT ニックネーム列オプションを使用すると、そのニックネームに対応するファイルを照会の実行中に提供することを暗黙指定します。 DOCUMENT オプションに FILE 値を指定した場合、照会の実行時に提供されるのは、そのニックネームのニックネーム定義に一致するスキーマをもったファイルの絶対パスになります。 XML の場合: この列は DOCUMENT 列であることを指定します。DOCUMENT 列の値は、照会の実行時にニックネームに与えられる XML ソース・データの型を指示します。このオプションが受け入れられるのは、ルート・ニックネーム (XML 文書の最上位のエレメントを特定するニックネーム) の列の場合だけです。ニックネームごとに DOCUMENT オプションで指定できる列は 1 つだけです。 DOCUMENT オプションと関連付けられる列は、VARCHAR データ型でなければなりません。 FILE_PATH または DIRECTORY_PATH ニックネーム・オプションではなく DOCUMENT 列オプションを使用すると、照会の実行時にはそのニックネームに対応する文書が提供されます。 DOCUMENT オプションの有効値は次のとおりです。 **FILE** ニックネーム列の値を、ファイルのパス名にバインドすることを指定します。照会の実行時にはそのファイル内のデータが提供されます。 **DIRECTORY** ニックネーム列の値を、複数の XML データ・ファイルの入ったディレクトリーのパス名にバインドすることを指定します。照会の実行時にはその複数のファイル内の XML データが提供されます。そのデータは、指定されたディレクトリー・パスの XML ファイルに入っています。 XML ラッパーは、ユーザーによって指定されたディレクトリーに置かれた .xml 拡張子の付いたファイルだけを使用します。 XML ラッパーは、そのディレクトリー内の他のファイルはすべて無視します。 **URI** ニックネーム列の値を、 URI によって参照されているリモート XML ファイルのパス名にバインドすることを指定します。 URI アドレスによって Web 上のその XML ファイルのリモート・ロケーションが示されます。 **COLUMN** リレーショナル列内に XML 文書を保管することを指定します。	
ELEMENT_NAME	BioRS エレメント名を指定します。その名前で大文字小文字を区別するかどうかは、BioRS サーバーでの大文字小文字の区別によってと、CASE_SENSITIVE サーバー・オプションの値によって決まります。 BioRS エレメント名を指定する必要があるのは、それが列名と異なる場合だけです。	
ESCAPE_INPUT	XML 入力値内の XML 特殊文字を置き換えるかどうかを指定します。繰り返しエレメントをもった XML フラグメントなどの XML フラグメントを入力として組み込みたい場合に、このオプションを使用します。ESCAPE_INPUT 列オプションを使用する列上に TEMPLATE 列オプションを指定する必要があります。列データ型は VARCHAR または CHAR でなければなりません。 有効な値は次のとおりです。 Y XML 入力内に特殊文字が入っている場合、その入力文字を表すのに XML で使われる対応文字に置き換えられます。 N 入力文字は、表示されたとおりに保存されます。	Y

表 129. ニックネーム列オプションおよびその設定値 (続き)

オプション	説明および有効設定値	デフォルトの設定値
FOREIGN_KEY	このニックネームは子ニックネームであることを示し、それに対応する親ニックネームを指定します。ニックネームには、最大 1 つの FOREIGN_KEY 列オプションを付けることができます。このオプションの値では、大文字小文字が区別されます。このオプションで指定された表には、ラッパーで生成されたキーが収容されます。この列には XPATH オプションを指定してはなりません。親ニックネームと子ニックネームを結合するためにだけこの列を使用することができます。 親ニックネームに別のスキーマ名が付いていると、 FOREIGN_KEY オプションを指定した CREATE NICKNAME ステートメントは失敗します。 FOREIGN_KEY 文節内で参照されているニックネームが、対応する CREATE NICKNAME ステートメント内で引用符で囲まれて小文字または大文字小文字混合と明示的に定義されていないかぎり、そのニックネームを FOREIGN_KEY 文節内で参照するときは、大文字でニックネームを指定する必要があります。 列上にこのオプションを設定した場合、列には他のオプションは設定できません。	
INDEX	定義行列グループ内でこのオプションを置く列の序数。このオプションは必須です。	
IS_INDEXED	対応する列に索引を付けるかどうか (述部内でその列を参照できるかどうか) を指示します。有効値は Y と N です。値 Y を指定できるのは、BioRS サーバーによって索引を付けられた対応エレメントをもつ列の場合のみです。	ニックネームの作成時には、BioRS で索引を付けられたエレメントに対応するすべての列に、値を Y に設定されたこのオプションが自動的に付け加えられます。
IS_REPEATING	列が多値かどうかを指示します。有効値は、Y と N です。 以下の場合は最後の値だけが戻されます。 • VARCHAR 以外の繰り返し属性 • ALL_VALUES 'N' の指定時の VARCHAR 列 この制限事項に対処するために、繰り返し属性列用の二重定義を作成することができます。	N
NUMERIC_STRING	数字のストリングを列に入れるかどうかを指定します。 Y この列には数字のストリング '0'、 '1'、 '2'、 .... '9' が入る。ブランクは入らない。この列に、後に末尾ブランクが付いた数値ストリングのみが入る場合、Y を指定しないでください。 列に対して NUMERIC_STRING を Y に設定すると、列データをソートする場合に支障となるブランクがこの列には入らないことを、オプティマイザーに知らせることになります。データ・ソースの照合シーケンスがフェデレーテッド・サーバーで使用される照合シーケンスとは異なる場合に、このオプションを使用してください。照合シーケンスが異なるため、このオプションを使用する列は、リモート評価から除外されません。 N この列は数値ストリング列ではないか、またはブランクの入った数値ストリング列である。	N
PRIMARY_KEY	このニックネームは親ニックネームであることを示します。列データ型は VARCHAR(16) でなければなりません。ニックネームには、最大 1 つの PRIMARY_KEY 列オプションを付けることができます。有効値は YES のみです。このオプションで指定された列には、ラッパーで生成されたキーが保存されます。この列には XPATH オプションを指定してはなりません。親ニックネームと子ニックネームを結合するためにだけこの列を使用することができます。 列上にこのオプションを設定した場合、列には他のオプションは設定できません。	

表 129. ニックネーム列オプションおよびその設定値 (続き)

オプション	説明および有効設定値	デフォルトの設定値
REFERENCED_OBJECT	このオプションが有効なのは、BioRS データ型が参照の列の場合のみです。このオプションは、現在の列によって参照される BioRS データ・バンクの名前を指定します。その名前で大文字小文字を区別するかどうかは、BioRS サーバーでの大文字小文字の区別によってと、 CASE_SENSITIVE サーバー・オプションの値によって決まります。	
REMOTE_NAME	対応する Documentum 属性または列の名前を指定します。このオプションは、リモートの属性名または列名をローカル DB2 UDB 列名にマップします。	DB2 UDB 列名。
SOAPACTIONCOLUMN	Web サービス記述言語 (WSDL) フォーマットから URI SOAPACTION 属性を動的に指定する列。このオプションは、ルート・ニックネームでのみ指定します。 列上にこのオプションを設定した場合、列には他のオプションは設定できません。	
TEMPLATE	XML 入力文書の作成に使用する列テンプレート・フラグメント。このフラグメントは、指定されたテンプレート構文に適合していなければなりません。	
URLCOLUMN	照会の実行時に Web サービス・エンドポイントの URL を動的に指定するための列。このオプションは、ルート・ニックネームでのみ指定します。 列上にこのオプションを設定した場合、列には他のオプションは設定できません。	
VARCHAR_NO_ TRAILING_BLANKS	このオプションを用いるのは、比較時に長さを埋めるための末尾ブランクが使用されない可変長文字データ型のデータ・ソースに対してです。 Oracle などの一部のデータ・ソースは、DB2 UDB for Linux、UNIX、Windows の比較セマンティクスと同じ結果を戻すブランク埋め込み文字比較のセマンティクスを備えていません。データ・ソース・オブジェクト内の特定の VARCHAR または VARCHAR2 列だけに適用したい場合にこのオプションを設定してください。 Y これらの VARCHAR 列には末尾ブランクが欠如しているか、あるいは、フェデレーテッド・サーバーのセマンティクスに似たブランク埋め込み文字比較セマンティクスがデータ・ソースに備わっています。 フェデレーテッド・サーバーは、処理対象の文字比較操作をデータ・ソースに送信します。 N これらの VARCHAR 列には末尾ブランクがあり、しかも、フェデレーテッド・サーバーのものとは異なるブランク埋め込み文字比較セマンティクスがデータ・ソースに備わっています。 同等のセマンティクスに合うように補正できなければ、フェデレーテッド・サーバーが文字比較操作を処理します。たとえば、述部を書き直します。	影響を受けるデータ・ソースの場合は N。
XPATH	この列に対応するデータの入った XML 文書内の XPath 式を指定します。CREATE NICKNAME ステートメントがこの XPATH ニックネーム・オプションから XPath 式を適用した後で、ラッパーは XPath 式を評価します。	

関連概念:

- 「フェデレーテッド・システム・ガイド」の『プッシュダウン分析』

関連タスク:

- 「フェデレーテッド・システム・ガイド」の『グローバルな最適化』

フェデレーテッド・システムの関数マッピング・オプション

DB2 Information Integrator は、既存の組み込みデータ・ソース関数と組み込み DB2 関数間のデフォルトのマッピングを備えています。ほとんどのデータ・ソースの場合、ラッパー内にデフォルトの関数マッピングがあります。フェデレーテッド・サーバーが認識しないデータ・ソース関数を使用するには、データ・ソース関数と、フェデレーテッド・データベースでの対応する関数との間に、関数マッピングを作成する必要があります。

関数マッピング・オプションの主目的は、データ・ソースでデータ・ソース関数を実行した場合の潜在的な費用についての情報を提供することです。プッシュダウン分析により、データ・ソースの関数が照会内にある関数を実行できるかどうかを判別します。照会オプティマイザーは、関数処理をデータ・ソースにプッシュダウンした場合、最小の費用で済むかどうかを判断します。

関数マッピング定義で提供される統計情報は、照会オプティマイザーが、データ・ソース関数を実行する費用見積もりを、 DB2 関数を実行する費用見積もりと比較するのに役立ちます。

表 130. 関数マッピング・オプションおよびその設定値

オプション	有効な設定値	デフォルトの設定値
DISABLE	デフォルト関数マッピングを使用不可にする。有効な値は 'Y' および 'N'。	'N'
INITIAL_INSTS	データ・ソース関数が呼び出された最初の時および最後の時に、処理される命令数の見積もり。	'0'
INITIAL_IOS	データ・ソース関数が呼び出された最初の時および最後の時に、実行される入出力数の見積もり。	'0'
IOS_PER_ARGBYTE	データ・ソース関数に渡された引き数セットの、それぞれのバイトごとに費やされる入出力数の見積もり。	'0'
IOS_PER_INVOC	データ・ソース関数の呼び出しごとの入出力数の見積もり。	'0'
INSTS_PER_ARGBYTE	データ・ソース関数に渡された引き数セットの、それぞれのバイトごとに処理される命令数の見積もり。	'0'
INSTS_PER_INVOC	データ・ソース関数の呼び出しごとに処理される命令数の見積もり。	'450'
PERCENT_ARGBYTES	データ・ソース関数が実際に読み取る入力引き数バイトの平均パーセントの見積もり。	'100'
REMOTE_NAME	データ・ソース関数の名前。	ローカル名

フェデレーテッド・システムのサーバー・オプション

サーバー・オプションは、データ・ソース・サーバーを記述するために使用されます。サーバー・オプションは、データ保全性、ロケーション、セキュリティー、およびパフォーマンス情報を指定します。すべてのデータ・ソースに利用できるサーバー・オプションがある一方で、データ・ソース別のサーバー・オプションもあります。

リレーショナル・データ・ソースに共通のフェデレーテッド・サーバー・オプションは以下のとおりです。

- 互換性オプション。COLLATING_SEQUENCE、IGNORE_UDT
- データ保全性オプション。IUD_APP_SVPT_ENFORCE
- データと時刻のオプション。
  DATEFORMAT、TIMEFORMAT、TIMESTAMPFORMAT
- ロケーション・オプション。CONNECTSTRING、DBNAME、IFILE
- セキュリティー・オプション。FOLD_ID、FOLD_PW、INFORMIX_LOCK_MODE
- パフォーマンス・オプション。COMM_RATE、CPU_RATIO、
  DB2_MAXIMAL_PUSHDOWN、IO_RATIO、LOGIN_TIMEOUT、PACKET_SIZE、
  PLAN_HINTS、PUSHDOWN、TIMEOUT、VARCHAR_NO_TRAILING_BLANKS

以下の表は、各リレーショナル・データ・ソースごとに適したサーバー定義サーバー・オプションを一覧で示しています。

表 131. リレーショナル・データ・ソース用のサーバー・オプション

データ・ソース	CODEPAGE	COLLATING_SEQUENCE	COMM_RATE	CONNECTSTRING	CPU_RATIO	DATEFORMAT	DB2_MAXIMAL_PUSHDOWN	DBNAME	FOLD_ID	FOLD_PW	IFILE	INFORMIX_LOCK_MODE	IO_RATIO	IUD_APP_SVPT_ENFORCE	LOGIN_TIMEOUT	NODE	PACKET_SIZE	PASSWORD	PLAN_HINTS	PUSHDOWN	TIMEOUT	TIMEFORMAT	TIMESTAMPFORMAT	VARCHAR_NO_TRAILING_BLANKS
DB2 UDB for iSeries		X	X		X		X	X	X	X			X	X				X		X				X
DB2 UDB for z/OS and OS/390		X	X		X		X	X	X	X			X	X				X		X				X
DB2 for VM and VSE		X	X		X		X	X	X	X			X	X				X		X				X
DB2 UDB for Linux、UNIX、Windows		X	X		X		X	X	X	X			X	X				X		X				X

表 131. リレーショナル・データ・ソース用のサーバー・オプション (続き)

データ・ソース	CODEPAGE	COLLATING_SEQUENCE	COMM_RATE	CONNECTSTRING	CPU_RATIO	DATEFORMAT	DB2_MAXIMAL_PUSHDOWN	DBNAME	FOLD_ID	FOLD_PW	IFILE	INFORMIX_LOCK_MODE	IO_RATIO	IUD_APP_SVPT_ENFORCE	LOGIN_TIMEOUT	NODE	PACKET_SIZE	PASSWORD	PLAN_HINTS	PUSHDOWN	TIMEOUT	TIMEFORMAT	TIMESTAMPFORMAT	VARCHAR_NO_TRAILING_BLANKS
Informix		X	X		X		X	X	X	X		X	X	X		X		X		X				
Microsoft SQL Server	X	X	X		X		X	X	X	X			X	X		X		X		X				
ODBC	X	X	X		X	X	X	X	X	X			X	X		X		X		X		X	X	X
OLE DB		X		X																				
Oracle		X	X		X		X		X	X			X			X		X	X	X				X
Sybase		X	X		X		X	X	X	X	X		X		X	X	X	X	X	X				
Teradata		X	X		X		X						X	X		X				X				

以下の表は、 WebSphere Business Integration を除く各非リレーショナル・データ・ソースごとに適したサーバー定義サーバー・オプションを一覧で示しています。 WebSphere Business Integration 用のサーバー定義サーバー・オプションは、 684 ページの表 133 に一覧で示されています。

表 132. 非リレーショナル・データ・ソース用のサーバー・オプション

データ・ソース	CASE_SENSITIVE	CONTENT_DIR	DAEMON_PORT	ES_HOST	ES_PORT	ES_TRACING	ES_TRACELEVEL	ES_TRACEFILENAME	HMMPFAM_OPTIONS	HMMSEARCH_OPTIONS	MAX_ROWS	NODE	OS_TYPE	PORT	PROCESSORS	PROXU_AUTHID	PROXY_PASSWORD	PROXY_SERVER_NAME	PROXY_SERVER_PORT	PROXY_TYPE	RDBMS_TYPE	SOCKET_TIMEOUT	TIMEOUT	TRANSACTIONS	USE_CLOB_SEQUENCE
BioRS	X											X		X									X		
BLAST			X									X													X
Documentum		X										X	X								X			X	
Entrez											X					X	X	X	X	X		X			
Excel																									
Extended Search				X	X	X	X	X																	
HMMER			X						X	X		X			X										X
表構造ファイル																									

表 132. 非リレーショナル・データ・ソース用のサーバー・オプション (続き)

データ・ソース	CASE_SENSITIVE	CONTENT_DIR	DAEMON_PORT	ES_HOST	ES_PORT	ES_TRACING	ES_TRACELEVEL	ES_TRACEFILENAME	HMMPFAM_OPTIONS	HMMSEARCH_OPTIONS	MAX_ROWS	NODE	OS_TYPE	PORT	PROCESSORS	PROXU_AUTHID	PROXY_PASSWORD	PROXY_SERVER_NAME	PROXY_SERVER_PORT	PROXY_TYPE	RDBMS_TYPE	SOCKET_TIMEOUT	TIMEOUT	TRANSACTIONS	USE_CLOB_SEQUENCE
Web サービス																									
XML																X	X	X	X	X		X			

以下の表は、 WebSphere Business Integration データ・ソースに適したサーバー定義サーバー・オプションを一覧で示しています。

表 133. WebSphere Business Integration データ・ソース用のサーバー・オプション

データ・ソース	APP_TYPE	FAULT_QUEUE	MQ_CONN_NAME	MQ_MANAGER	MQ_RESPONSE_TIMEOUT	MQ_SVRCONN_CHANNELNAME	REQUEST_QUEUE	RESPONSE_QUEUE
WebSphere Business Integration	X	X	X	X	X	X	X	X

以下の表は、各サーバー・オプションを説明し、有効設定値とデフォルト設定値を一覧で示しています。

表 134. サーバー・オプションとその設定値

オプション	説明および有効設定値	デフォルトの設定値
APP_TYPE	リモート・アプリケーションのタイプ。有効値は 'PSOFT'、'SAP'、および 'SIEBEL' です。このオプションは必須です。	なし

表 134. サーバー・オプションとその設定値 (続き)

オプション	説明および有効設定値	デフォルトの設定値
CASE_SENSITIVE	BioRS サーバーが、大文字小文字を区別して名前を扱うかどうかを指定します。有効値は、Y または N です。 'Y' BioRS サーバーは、大文字小文字を区別する方式を使って名前を扱います。 'N' BioRS サーバーは、大文字小文字を区別する方式で名前を扱いません。 BioRS 製品では、 BioRS サーバーに保管されるデータでの大文字小文字の区別は、構成パラメーターで制御されます。 CASE_SENSITIVE オプションは、 BioRS システムの構成パラメーターに相当する DB2 Information Integrator のオプションです。 BioRS サーバーでの大文字小文字の区別の構成設定は、 BioRS システムにおいてと DB2 Information Integrator において同期する必要があります。 BioRS と DB2 Information Integrator の間で、大文字小文字の区別の構成設定の同期を保っておかないと、 DB2 Information Integrator を通して BioRS データにアクセスしようとしたときにエラーが生じます。 DB2 Information Integrator において新規の BioRS サーバーを作成した後は、 CASE_SENSITIVE オプションの変更も削除もできません。 CASE_SENSITIVE オプションを変更する必要が生じた場合、サーバー全体をいったんドロップしてから作成しなおす必要があります。 BioRS サーバーをドロップした場合、対応する BioRS ニックネームもすべてもう一度作成する必要があります。 DB2 Information Integrator は、ドロップされたサーバーに対応するすべてのニックネームを自動的にドロップします。	Y
CODEPAGE	データ・ソースのクライアント構成のコード化文字セットに対応する DB2 コード・ページ ID を指定します。クライアントのコード・ページとフェデレーテッド・データベースのコード・ページが一致しない場合、クライアントのコード・ページを指定する必要があります。 Unicode をサポートするデータ・ソースの場合、 CODEPAGE オプションを、データ・ソース・クライアントのサポートされている Unicode エンコード方式に対応する DB2 コード・ページ ID に設定することができます。	Unicode 以外のフェデレーテッド・データベースの UNIX または Windows システムの場合: フェデレーテッド・データベースのコード・ページ。 Unicode フェデレーテッド・データベースの UNIX システムの場合: 1208 Unicode フェデレーテッド・データベースの Windows システムの場合: 1202

表 134. サーバー・オプションとその設定値 (続き)

オプション	説明および有効設定値	デフォルトの設定値
COLLATING_ SEQUENCE	データ・ソースがフェデレーテッド・データベースと同じデフォルト照合シーケンスを使用するかどうかを、 NLS コード・セットと国別/地域別情報に基づいて指定します。 'Y' データ・ソースは DB2 フェデレーテッド・データベースと同じ照合シーケンスをもちます。 'N' データ・ソースは DB2 フェデレーテッド・データベースの照合シーケンスとは別の照合シーケンスをもちます。 'I' データ・ソースは DB2 フェデレーテッド・データベースの照合シーケンスとは別の照合シーケンスを持ち、データ・ソースの照合シーケンスは大文字小文字の区別をしません (たとえば、 'STEWART' と 'StewART' は等しいと見なされます)。	'N'
COMM_RATE	フェデレーテッド・サーバーとデータ・ソース・サーバー間の通信レートを指定します。秒当たりの MB 単位で表されます。 有効な値は 0 より大きく、$1x10^{23}$ より小さい値です。任意の有効な REAL 表記で値を表すことができます。	'2'
CONTENT_DIR	GET_FILE、GET_FILE_DEL、GET_RENDITION、および GET_RENDITION_DEL 疑似列で取り出されたコンテンツ・ファイルの保管用のローカル側からアクセス可能なルート・ディレクトリーの名前を指定します。これは、その疑似列を使用できるすべてのユーザーから書き込み可能でなければなりません。	UNIX の場合: '/tmp' Windows 2000 の場合: 'C:¥temp'
CONNECTSTRING	OLE DB Provider への接続に必要な初期化プロパティーを指定します。	なし

表 134. サーバー・オプションとその設定値 (続き)

オプション	説明および有効設定値	デフォルトの設定値
CPU_RATIO	データ・ソースの CPU がフェデレーテッド・サーバーの CPU よりどれほど速いまたは遅いかを示します。 有効な値は 0 より大きく、$1x10^{23}$ より小さい値です。任意の有効な REAL 表記で値を表すことができます。 1 を設定した場合、DB2 フェデレーテッド CPU の速度とデータ・ソースの CPU の速度は、 1:1 の比率で同じ CPU 速度をもつことを示します。 .5 を設定した場合、DB2 フェデレーテッド CPU の速度のほうが、データ・ソースの CPU 速度よりも 50% 遅いことを示します。 2 を設定した場合、DB2 フェデレーテッド CPU の速度のほうが、データ・ソースの CPU 速度よりも 2 倍早いことを示します。	'1.0'
DATEFORMAT	データ・ソースで使用される日付フォーマット。日付の数値形式を表す 'DD'、'MM'、および 'YY' または 'YYYY' を使って、フォーマットを入力します。また、スペースやコンマなどの区切り文字を指定する必要もあります。たとえば、'2003-01-01' の日付フォーマットを表すには、'YYYY-MM-DD' を使用します。このフィールドは NULL 可能です。	なし
DAEMON_PORT	デーモンが BLAST または HMMER のジョブ要求を listen するポートの番号を指定します。このポート番号は、デーモンの構成ファイルの DAEMON_PORT オプションに指定されたものと同じ番号でなければなりません。	BLAST: '4007'、HMMER: '4098'

表 134. サーバー・オプションとその設定値 (続き)

オプション	説明および有効設定値	デフォルトの設定値
DB2_MAXIMAL_ PUSHDOWN	アクセス・プランの選択時に照会オプティマイザーが使用する 1 次基準を指定します。照会オプティマイザーは、コストに基づいてか、またはリモート・データ・ソースで可能な限り多くの照会を実行するためのユーザー要求に基づいて、アクセス・プランを選択することができます。 'Y' 照会オプティマイザーは、他のアクセス・プランより多くの照会操作をデータ・ソースにプッシュダウンするアクセス・プランを選びます。同数のプッシュダウンを行うアクセス・プランが複数ある場合、照会オプティマイザーは最もコストの低いプランを選びます。 フェデレーテッド・サーバー上のマテリアライズ照会表 (MQT) が一部または全部の照会を処理できる場合、そのようなマテリアライズ照会表を備えたアクセス・プランが使用される可能性があります。フェデレーテッド・データベースは、カルテシアン積に帰結した照会をプッシュダウンしません。 'N' 照会オプティマイザーは、コストに基づいてアクセス・プランを選びます。	'N'
DBNAME	フェデレーテッド・サーバーにアクセスさせるデータ・ソース・データベースの名前。 DB2 データベースの場合、この値は初期のリモート DB2 データベース接続での特定のデータベースに対応します。その特定のデータベースは、 CATALOG DATABASE コマンドを使うかまたは DB2 構成アシスタントを使ってフェデレーテッド・サーバーでカタログされたリモート DB2 データベースのデータベース別名です。 Oracle インスタンスに入っているのは 1 つのデータベースだけなので、Oracle データ・ソースには該当しません。	なし
ES_HOST	検索しようとしている Extended Search サーバーの完全修飾ホスト名または IP アドレスを指定します。このオプションは必須です。	なし
ES_PORT	Extended Search サーバーが要求を listen するポートの番号を指定します。このオプションは任意のオプションです。	'6001'

表 134. サーバー・オプションとその設定値 (続き)

オプション	説明および有効設定値	デフォルトの設定値
ES_TRACING	リモート Extended Search サーバーで生成されるエラー・メッセージ、警告メッセージ、および通知メッセージのトレースを使用可能にするかどうかを指定します。有効な値は次のとおりです。 'OFF' トレース・メッセージをログ記録しません。 'ON' トレース・メッセージをログ記録します。 このオプションは任意のオプションです。	'OFF'

ES_TRACELEVEL	トレースを使用可能にした場合のこのオプションは、ログ・ファイルに書き込まれるメッセージのタイプを指定します。以下のトレース・レベルをそれぞれ単独で使用可能または使用禁止にすることができます。
	'C' クリティカル・エラー・メッセージ
	'N' クリティカル以外のエラー・メッセージ
	'W' 警告メッセージ
	'I' 通知メッセージ
	たとえば、以下のようにします。
	`ES_TRACELEVEL 'W'`
	`ES_TRACELEVEL 'CN'`
	このオプションは任意のオプションです。
		'C'

ES_TRACEFILENAME	トレースを使用可能にした場合のこのオプションは、メッセージの書き込み先のディレクトリーとファイルの名前を指定します。このオプションは任意のオプションです。	UNIX オペレーティング・システムの場合: $INSTHOME/sqllib/log/ESWrapper.log Windows オペレーティング・システムの場合: %DB2TEMPDIR%\ESWrapper.log
FAULT_QUEUE	アダプターからラッパーにエラー・メッセージを送付する障害キューの名前。この名前は、WebSphere MQ のキュー名の仕様に準じていなければなりません。これは必須指定のオプションです。	なし

表 134. サーバー・オプションとその設定値 (続き)

オプション	説明および有効設定値	デフォルトの設定値
FOLD_ID (この表の最後にある注 1 および 4 を参照。)	フェデレーテッド・サーバーが認証のためにデータ・ソース・サーバーに送信するユーザー ID に適用される。 有効な値は次のとおりです。 'U' フェデレーテッド・サーバーは、ユーザー ID をデータ・ソースに送信する前に、大文字に変換します。これは、DB2 ファミリーおよび Oracle データ・ソースについては当然の選択です (この表の最後にある注 2 を参照。) 'N' フェデレーテッド・サーバーは、ユーザー ID をデータ・ソースに送信する前に、ユーザー ID に対して何の処理も行いません。 (この表の最後の注 2 を参照。) 'L' フェデレーテッド・サーバーは、ユーザー ID をデータ・ソースに送信する前に、小文字に変換します。 これらの設定値のいずれも使用しない場合は、フェデレーテッド・サーバーはユーザー ID を大文字にしてデータ・ソースに送信しようとします。そのユーザー ID を正常に送信できない場合は、サーバーはユーザー ID を小文字で送信しようとします。	なし
FOLD_PW (この表の最後にある注 1、3 および 4 を参照。)	フェデレーテッド・サーバーが認証のためにデータ・ソースに送信するパスワードに適用される。有効な値は次のとおりです。 'U' フェデレーテッド・サーバーは、パスワードをデータ・ソースに送信する前に、大文字に変換します。これは、DB2 ファミリーおよび Oracle データ・ソースについては当然の選択です。 'N' フェデレーテッド・サーバーは、パスワードをデータ・ソースに送信する前に、パスワードに対して何の処理も行いません。 'L' フェデレーテッド・サーバーは、パスワードをデータ・ソースに送信する前に、小文字に変換します。 これらの設定値のいずれも使用しない場合は、フェデレーテッド・サーバーはパスワードを大文字にしてデータ・ソースに送信しようとします。そのパスワードを正常に送信できない場合は、サーバーはパスワードを小文字で送信しようとします。	なし

表 134. サーバー・オプションとその設定値 (続き)

オプション	説明および有効設定値	デフォルトの設定値
HMMPFAM_OPTIONS	オプションを列名にマップする参照表内に対応する列名をもたない --null2、--pvm、および --xnu などの hmmpfam オプションを指定します。 たとえば、以下のようにします。 `HMMPFAM_OPTIONS '--xnu --pvm'` この例ではデーモンは、照会の WHERE 文節と、さらに別のオプション --xnu --pvm を指定した HMMPFAM プログラムを実行します。	
HMMSEARCH_ OPTIONS	ユーザーは、追加のコマンド行オプションを hmmsearch コマンドに指定することができます。SEARCH タイプのもののみが有効です。詳細は、「HMMER User's Guide」を参照してください。	なし。
IFILE	Sybase Open Client インターフェース・ファイルのパスと名前を指定します。 Windows NT フェデレーテッド・サーバーでは、デフォルトは %DB2PATH%¥interfaces です。 UNIX フェデレーテッド・サーバーでは、デフォルトのパスと名前の値は、$DB2INSTANCE/sqllib/interfaces です。	なし。
INFORMIX_LOCK_ MODE	Informix データ・ソース用に設定するロック・モードを指定します。 Informix ラッパーは、 Informix データ・ソースへの接続を確立した後ただちに 'SET LOCK MODE' コマンドを発行します。有効な値は次のとおりです。 **'W'** Informix ロック・モードを WAIT に設定します。ロックされた表または行へのアクセスをラッパーが試みた場合、 Informix はそのロックが解除されるまで待機します。 **'N'** Informix ロック・モードを NOWAIT に設定します。ロックされた表または行へのアクセスをラッパーが試みた場合、 Informix はエラーを戻します。 **'n'** Informix ロック・モードを WAIT *n* 秒に設定します。ロックされた表または行へのアクセスをラッパーが試みた場合に指定の秒数以内にロックが解除されないと、 Informix はエラーを戻します。	'W'

表 134. サーバー・オプションとその設定値 (続き)

オプション	説明および有効設定値	デフォルトの設定値
IO_RATIO	データ・ソースの入出力システムがフェデレーテッド・サーバーの入出力システムより、どれほど実行が速いまたは遅いかを表します。 有効な値は 0 より大きく、$1x10^{23}$ より小さい値です。任意の有効な REAL 表記で値を表すことができます。 1 を設定した場合、DB2 フェデレーテッド入出力の速度とデータ・ソースの入出力の速度は、 1:1 の比率で同じ入出力速度をもつことを示します。 .5 を設定した場合、DB2 フェデレーテッド入出力の速度のほうが、データ・ソースの入出力速度よりも 50% 遅いことを示します。 2 を設定した場合、DB2 フェデレーテッド入出力の速度のほうが、データ・ソースの入出力速度よりも 2 倍早いことを示します。	'1.0'
IUD_APP_SVPT_ ENFORCE	DB2 フェデレーテッド・システムが、アプリケーションのセーブポイント・ステートメントの検出または作成を実施すべきかどうかを指定します。 SET SERVER OPTION ステートメントを使って設定した場合のこのサーバー・オプションは、静的 SQL ステートメントに対しては効力を持ちません。 'Y' 挿入、更新、または削除操作においてエラーが生じたときに、データ・ソースがアプリケーション・セーブポイント・ステートメントを実施していなかった場合は、フェデレーテッド・サーバーは挿入、更新、または削除トランザクションをロールバックします。 SQL エラー・コード SQL1476N が戻されます。 'N' エラーが起きても、フェデレーテッド・サーバーはトランザクションをロールバックしません。ユーザーのアプリケーションでエラー・リカバリーを処理する必要があります。	'Y'
LOGIN_TIMEOUT	ログイン要求に対して、Sybase Open Client からの応答を DB2 フェデレーテッド・サーバーが待つ秒数を指定します。デフォルト値は TIMEOUT と同じです。	'0'

表 134. サーバー・オプションとその設定値 (続き)

オプション	説明および有効設定値	デフォルトの設定値
MAX_ROWS	Entrez ラッパーを使用する照会においてフェデレーテッド・サーバーが戻す行数を指定します。 正数とゼロのみを指定することができます。このオプションをゼロに設定すると、 NCBI Web サイトから制限のない数の行を照会で取り出すことができます。ただし MAX_ROWS サーバー・オプションをゼロまたは非常に大きい数値に設定した場合、照会のパフォーマンスが影響を受けることがあります。 MAX_ROWS サーバー・オプションは必須ではありません。	Microsoft Windows オペレーティング・システム: 2000 行。 UNIX ベース・オペレーティング・システム: 5000 行。
MQ_CONN_NAME	Websphere MQ サーバーが稼働するコンピューターのホスト名またはネットワーク・アドレス。接続名の例には 9.30.76.151(1420) があります。ただし 1420 はポート番号です。ポート番号を除外した場合、1414 のデフォルト値が使われます。このオプションは任意のオプションです。このオプションを省略した場合、MQSERVER 環境変数 (db2dj.ini ファイル内に指定されている場合) が使われて、チャネル定義が選択されます。 MQSERVER が設定されていないと、クライアントのチャネル表が使われます。	ラッパーは MQSERVER 環境変数 (db2dj.ini ファイルに指定されている場合) を使って、チャネル定義を選択します。MQSERVER 環境変数が設定されていないと、ラッパーはクライアントのチャネル表を使います。
MQ_MANAGER	WebSphere MQ マネージャーの名前。任意の有効な WebSphere MQ マネージャーの名前。このオプションは必須です。	なし。
MQ_RESPONSE_ TIMEOUT	ラッパーが応答キューからの応答メッセージを待機する時間。この値はミリ秒単位です。タイムアウト期間はないことを指示するのに特殊値 -1 を指定することができます。このオプションは任意のオプションです。	10000
MQ_SVRCONN_ CHANNELNAME	ラッパーが接続を試みる必要のある Webspehere MQ Manager 上のサーバー接続チャネルの名前。このパラメーターを指定できるのは、MQ_CONN_NAME サーバー・オプションを指定した場合だけです。このオプションを省略した場合、デフォルトのサーバー接続チャネル SYSTEM.DEF.SVRCONN が使われます。	SYSTEM.DEF.SVRCONN

表 134. サーバー・オプションとその設定値 (続き)

オプション	説明および有効設定値	デフォルトの設定値
NODE	リレーショナル・データ・ソース: RDBMS に対してデータ・ソースをインスタンスとして定義するための名前。 Documentum: Documentum Docbase の実際の名前を指定します。このオプションは必須です。 BLAST: BLAST デーモン・プロセスが稼働するシステムのホスト名を指定します。このオプションは必須です。 HMMER: HMMER デーモン・プロセスが稼働するサーバーのホスト名を指定します。このオプションは必須です。 BioRS: BioRS 照会ツールを利用できるシステムのホスト名を指定します。このオプションは任意のオプションです。	BioRS: *localhost*
OS_TYPE	Docbase サーバーのオペレーティング・システムを指定します。有効値は AIX、SOLARIS、および WINDOWS です。このオプションは必須です。	なし
PACKET_SIZE	Sybase インターフェース・ファイルのパケット・サイズをバイト単位で指定します。指定したパケット・サイズをデータ・ソースがサポートしない場合、接続は失敗します。各レコードが非常に大きい場合 (たとえば、大きな表に行を挿入する場合など) にパケット・サイズを増やすと、パフォーマンスが非常によくなります。バイト・サイズは数値です。	
PASSWORD	パスワードがデータ・ソースに送信されるかどうかを指定します。 'Y' パスワードは、データ・ソースに送られて妥当性検査されます。 'N' パスワードは、データ・ソースに送信されることも妥当性検査されることもありません。	'Y'

表 134. サーバー・オプションとその設定値 (続き)

オプション	説明および有効設定値	デフォルトの設定値
PLAN_HINTS	プラン・ヒント を使用可能にするかどうかを指定します。プラン・ヒントはステートメントの一部分であり、データ・ソース・オプティマイザーに対しての追加情報を提供します。特定の照会タイプについてこの情報を利用すれば、照会パフォーマンスを改善することができます。プラン・ヒントは、データ・ソース・オプティマイザーが索引を使用するかどうか、どの索引を使用するか、またはどの表結合シーケンスを使うかを判別するのに役立ちます。 'Y' データ・ソースがプラン・ヒントをサポートしている場合は、プラン・ヒントを使用可能にします。 'N' プラン・ヒントはデータ・ソースで使用可能になりません。 このオプションを使用できるのは、Oracle および Sybase データ・ソースの場合のみです。	'N'
PORT	ラッパーが BioRS サーバーへの接続に使用するポートの番号を指定します。このオプションは任意のオプションです。	'5014'
PROCESSORS	HMMER プログラムが使用するプロセッサーの数を指定します。このオプションは、hmmpfam コマンドの --cpu オプションと同等です。	なし
PROXY_AUTHID	PROXY_TYPE の値が 'SOCKS5' の場合に使用するユーザー名を指定します。このフィールドは、PROXY_TYPE の値が 'SOCKS5' の場合はオプションです。使用するユーザー名に関しては、ネットワーク管理者に問い合わせてください。 PROXY_TYPE が 'SOCKS5' でなければ、このオプションは無効です。	なし
PROXY_PASSWORD	PROXY_TYPE の値が 'SOCKS5' の場合に使用するパスワードを指定します。このフィールドは、PROXY_TYPE の値が 'SOCKS5' の場合はオプションです。使用するパスワードに関しては、ネットワーク管理者に問い合わせてください。 PROXY_TYPE が 'SOCKS5' でなければ、このオプションは無効です。	なし
PROXY_SERVER_ NAME	プロキシー・サーバー名または IP アドレスを指定します。このフィールドは、PROXY_TYPE の値が 'HTTP'、'SOCKS4'、または 'SOCKS5' の場合は必須です。プロキシー・サーバー名または IP アドレスに関しては、ネットワーク管理者に問い合わせてください。	なし

表 134. サーバー・オプションとその設定値 (続き)

オプション	説明および有効設定値	デフォルトの設定値
PROXY_SERVER_ PORT	プロキシー・サーバーのポート番号を指定します。このフィールドは、PROXY_TYPE の値が 'HTTP'、'SOCKS4'、または 'SOCKS5' の場合は必須です。使用する必要のあるプロキシー・サーバーのポート番号に関しては、ネットワーク管理者に問い合わせてください。	なし
PROXY_TYPE	ファイアウォールの背後にいる場合にインターネットへのアクセスに使用するプロキシー・タイプを指定します。有効な値は 'NONE'、'HTTP'、'SOCKS4'、または 'SOCKS5' です。デフォルト値は 'NONE' です。使用するプロキシーのタイプに関しては、ネットワーク管理者に問い合わせてください。	'NONE'
PUSHDOWN	'Y' DB2 UDB は、データ・ソースによる操作の評価を検討します。 'N' DB2 UDB は、列名を指定した SELECT のみの入った データ・ソース SQL ステートメントを送信します。述部 (WHERE= など) 列およびスカラー関数 (MAX や MIN など)、ソート (ORDER BY や GROUP BY など)、および結合は、データ・ソースに送られるどの SQL にも入れられません。	'Y'
RDBMS_TYPE	Docbase で使用される RDBMS を指定します。有効値は DB2、INFORMIX、ORACLE、SQLSERVER、または SYBASE です。このオプションは必須です。	なし
RESPONSE_QUEUE	アダプターからラッパーに照会結果を送付する応答キューの名前。この名前は、WebSphere MQ のキュー名の仕様に準じていなければなりません。このオプションは必須です。	なし
REQUEST_QUEUE	ラッパーからアダプターに照会要求を送付する要求キューの名前。この名前は、WebSphere MQ のキュー名の仕様に準じていなければなりません。このオプションは必須です。	なし
SOCKET_TIMEOUT	DB2 フェデレーテッド・サーバーがプロキシー・サーバーからの結果を待機する分単位の最長時間を指定します。有効値は、ゼロ以上の任意の数値です。デフォルトはゼロ '0' です。ゼロの値は、無制限の期間待機することを表します。	0

表 134. サーバー・オプションとその設定値 (続き)

オプション	説明および有効設定値	デフォルトの設定値
TIMEFORMAT	データ・ソースで使用される時刻フォーマット。 'hh12'、'hh24'、'mm'、'ss'、'AM'、または 'A.M' を使ってフォーマットを入力します。たとえば、'16:00:00' の時刻フォーマットを表すには、'hh24:mm:ss' を使います。 '8:00:00 AM' の時刻フォーマット を表すには、'hh12:mm:ss AM' を使います。このフィールドは NULL 可能です。	なし
TIMESTAMPFORMAT	データ・ソースで使用されるタイム・スタンプ・フォーマット。このフォーマットは日付および時刻のフォーマットに準じますが、 1/10 秒の 'n'、1/100 秒の 'nn'、ミリ秒の 'nnn' などをはじめとして、マイクロ秒の 'nnnnnn' までが付け加えられます。たとえば、'2003-01-01-24:00:00.000000' のタイム・スタンプ・フォーマットを表すには、'YYYY-MM-DD-hh24:mm:ss.nnnnnn' を使います。このフィールドは NULL 可能です。	なし
TIMEOUT	Sybase: 何らかの SQL ステートメントに対する Sybase Open Client からの応答を、 DB2 フェデレーテッド・サーバーが待つ秒数を指定します。 *seconds* の値は、DB2 Universal Database の整数範囲内にある、正の整数です。指定するタイムアウト値は、使用するラッパーにより異なります。 Sybase ラッパーの TIMEOUT オプションのデフォルト動作は 0 ですが、その場合、DB2 UDB は無期限に応答を待つことになります。 BioRS: BioRS ラッパーが BioRS サーバーからの応答を待機する時間を分単位で指定します。デフォルト値は 10 です。 このオプションは任意のオプションです。	'0'; BioRS: '10'
TRANSACTIONS	サーバー・トランザクション・モードを指定します。有効値は次のとおりです。 'NONE' トランザクションは使用可能になっていません。 'QUERY' Dctm_Query メソッドのトランザクションのみが使用可能になります。 **'ALL'** Dctm_Query メソッドのトランザクションが使用可能になります。現リリースでは、ALL は QUERY と同じ機能をもちます。	'QUERY'

表 134. サーバー・オプションとその設定値 (続き)

オプション	説明および有効設定値	デフォルトの設定値
USE_CLOB_ SEQUENCE	このオプションは、BlastSeq または HmmQSeq 列に対してフェデレーテッド・サーバーが使用するデータ型を指定します。その値は 'Y' か 'N' のどちらかにすることができます。 CREATE NICKNAME または ALTER NICKNAME ステートメントを使用して、BlastSeq または HmmQSeq 列のデフォルトのデータ型をオーバーライドすることができます。	'Y'
VARCHAR_NO_ TRAILING_BLANKS	このオプションを用いるのは、比較時に長さを埋めるための末尾ブランクが使用されない可変長文字データ型のデータ・ソースに対してです。 Oracle などの一部のデータ・ソースは、DB2 for Linux、DB2 for UNIX、および DB2 for Windows の比較セマンティクスと同じ結果を戻すブランク埋め込み文字比較のセマンティクスを備えていません。このオプションは、指定されたサーバーからアクセスされるデータ・ソース・オブジェクト内の、すべての VARCHAR および VARCHAR2 列にこれを適用したい場合に設定してください。これにはビューも入ります。 Y これらの VARCHAR 列には末尾ブランクが欠如しているか、あるいは、フェデレーテッド・サーバーのセマンティクスに似たブランク埋め込み文字比較セマンティクスがデータ・ソースに備わっています。 フェデレーテッド・サーバーは、処理のために文字比較操作をデータ・ソースにプッシュダウンします。 N これらの VARCHAR 列には末尾ブランクがあり、しかも、フェデレーテッド・サーバーのものとは異なるブランク埋め込み文字比較セマンティクスがデータ・ソースに備わっています。 同等のセマンティクスに合うように補正できなければ、フェデレーテッド・サーバーが文字比較操作を処理します。たとえば、述部を書き直します。	影響を受けるデータ・ソースの場合は N。

この表に関する注

1. このフィールドは、認証に指定される値に関係なく適用されます。
2. DB2 UDB はユーザー ID を大文字で保管するので、値 'N' と 'U' は論理的に互いに同等です。
3. パスワードの設定が 'N' の場合は、 FOLD_PW を設定しても効果はありません。パスワードが送信されないので、大文字小文字の区別は意味をなしません。

4. これらのいずれのオプションでも、NULL 値を設定することは避けてください。NULL 値を設定すると、 DB2 UDB はユーザー ID とパスワードの解決を複数回試みることになるので、望ましい設定のように思えますが、パフォーマンスが低下する可能性があります (DB2 UDB が ユーザー ID とパスワードを 4 回送信した後で、ようやくデータ・ソース認証に成功するということもあり得ます)。

関連概念:

- 「フェデレーテッド・システム・ガイド」の『プッシュダウンの可能性に影響を与えるサーバー特性』
- 「フェデレーテッド・システム・ガイド」の『グローバルな最適化に影響を与えるサーバー特性』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『DROP ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『ALTER SERVER ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『CREATE SERVER ステートメント』

フェデレーテッド・システムのユーザー・マッピング・オプション

これらのオプションは、すべてのリレーショナル・データ・ソースで有効です。非リレーショナル・データ・ソースの場合、REMOTE_AUTHID オプションと REMOTE_PASSWORD オプションは、 BioRS、Documentum、Extended Search、および Web サービスの各データ・ソースに対して有効です。 GUEST オプションは BioRS データ・ソースに対して有効です。

これらのオプションは、CREATE USER MAPPING ステートメントと ALTER USER MAPPING ステートメントで使用します。

表 135. ユーザー・マッピング・オプションとその設定値

オプション	有効な設定値	デフォルトの設定値
ACCOUNTING	DRDA: DRDA アカウント情報ストリングの指定に使用します。有効な設定値は、長さが 255 以下の任意のストリングです。このオプションは、アカウント情報を渡す必要がある場合のみ必要です。詳細は、「DB2 Connect ユーザーズ・ガイド」を参照してください。	なし
GUEST	ラッパーが BioRS サーバーへのゲスト・アクセス・モードを使用するかどうかを指定します。 **Y** ラッパーは BioRS サーバーへのゲスト・アクセス・モードを使用します。 **N** ラッパーは BioRS サーバーへのゲスト・アクセス・モードを使用しません。 このオプションは、Y の値に設定されると、REMOTE_AUTHID オプションと REMOTE_PASSWORD オプションに対して相互に排他的になります。	N
REMOTE_AUTHID	データ・ソースで使用される許可 ID を表します。有効な設定値は、長さが 255 以下の任意のストリングです。	DB2 Universal Database への接続に使用する許可 ID。

表 135. ユーザー・マッピング・オプションとその設定値 (続き)

オプション	有効な設定値	デフォルトの設定値
REMOTE_DOMAIN	Documentum: Documentum データ・ソースに接続するユーザーの認証に使用する Windows NT ドメインを指示します。有効な設定値は、任意の有効な Windows NT ドメイン・ネームです。	Documentum データベースのデフォルト認証ドメイン。
REMOTE_PASSWORD	データ・ソースで使用される許可パスワードを表します。有効な設定値は、長さが 32 以下の任意のストリングです。 以下の条件を満たせば、このオプションを設定する必要はありません。 • データベース・マネージャー構成パラメーター AUTHENTICATON が SERVER に設定されている。 • DB2 データベースへの接続時に許可 ID とパスワードを指定した。 パスワードを必要とするサーバーの場合にこのオプションを設定しないときは、上記の条件を両方とも満たしていることを確認してください。満たしていないと、接続は失敗します。	有効設定列に一覧で示した条件が両方とも満たされている場合に DB2 Universal Database への接続に使用するパスワード。

関連概念:

- 「*DB2 Connect ユーザーズ・ガイド*」の『DB2 Connect と DRDA』
- 「*DB2 Connect ユーザーズ・ガイド*」の『DRDA とデータ・アクセス』

フェデレーテッド・システムのラッパー・オプション

ラッパー・オプションは、ラッパーの構成や、フェデレーテッド・サーバーでのラッパーの使用法を定義するために使用されます。ラッパーを作成または変更するときに、ラッパー・オプションを設定することができます。

リレーショナルおよび非リレーショナルのすべてのデータ・ソースで、DB2_FENCED ラッパー・オプションが使われます。 ODBC データ・ソースでは、MODULE ラッパー・オプションが使われます。 Entrez データ・ソースでは、EMAIL ラッパー・オプションが使われます。

表 136. ラッパー・オプションとその設定値

オプション	有効な設定値		デフォルトの設定値
DB2_FENCED	fenced またはトラステッドのどちらのモードでラッパーを実行するかを指定します。		リレーショナル・ラッパー: N。
	Y	ラッパーを fenced モードで実行します。	IBM 製の非リレーショナル・ラッパー: N。
	N	ラッパーをトラステッド・モードで実行します。	サード・パーティー製の非リレーショナル・ラッパー: Y。

表 136. ラッパー・オプションとその設定値 (続き)

オプション	有効な設定値	デフォルトの設定値
EMAIL	Entrez ラッパーの登録時に E メール・アドレスを指定します。この E メール・アドレスは、すべての照会に組み込まれるので、照会数が多すぎて NCBI サーバーが過負荷になっているなどの問題がある場合に NCBI からユーザーに通知することができます。このオプションは必須です。	
MODULE	ODBC Driver Manager インプリメンテーションまたは SQL/CLI インプリメンテーションを収めたライブラリーの絶対パスを指定します。UNIX フェデレーテッド・サーバー上の ODBC ラッパーで必要。	Windows でのデフォルト値は odbc32.dll です。

関連タスク:

- 「フェデレーテッド・システム・ガイド」の『ラッパーの変更』

デフォルトのフォワード・データ型マッピング

データ・ソースのデータ型とフェデレーテッド・データベースのデータ型の間のマッピングには、フォワード・タイプのマッピングとリバース・タイプのマッピングの 2 種類があります。フォワード・タイプ・マッピング では、マッピングはリモートのタイプから対応するローカル・タイプへのマッピングです。

デフォルトのタイプ・マッピングをオーバーライドすることも、 CREATE TYPE MAPPING ステートメントを使用して新しいタイプ・マッピングを作成することもできます。

これらのマッピングは、特に注記のないかぎり、サポート対象のすべてのバージョンで有効です。

データ・ソースから DB2 for Linux、DB2 for UNIX、および DB2 for Windows へのデフォルトのどのフォワード・データ型マッピングでも、DB2 フェデレーテッド・スキーマは SYSIBM です。

下表は、DB2 for Linux、DB2 for UNIX、および DB2 for Windows のデータ型とデータ・ソースのデータ型間のデフォルトのフォワード・マッピングを示しています。

DB2 for z/OS and OS/390 データ・ソース

表 137. DB2 for z/OS and OS/390 のデフォルトのフォワード・データ型マッピング (表示されていない列があります)

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHAR	1	254	-	-	-	-	CHAR	-	0	N
CHAR	255	32672	-	-	-	-	VARCHAR	-	0	N
CHAR	1	254	-	-	Y	-	CHAR	-	0	Y
CHAR	255	32672	-	-	Y	-	VARCHAR	-	0	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	-	-	-	-	-	DATE	-	0	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
FLOAT	4	-	-	-	-	-	REAL	-	-	-
FLOAT	8	-	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	1	127	-	-	-	-	GRAPHIC	-	0	N
INTEGER	-	-	-	-	-	-	INTEGER	-	0	-
ROWID	-	-	-	-	Y	-	VARCHAR	40	-	Y
SMALLINT	-	-	-	-	-	-	SMALLINT	-	0	-
TIME	-	-	-	-	-	-	TIME	-	0	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
TIMESTMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	0	N
VARCHAR	1	32672	-	-	Y	-	VARCHAR	-	0	Y
VARG	1	16336	-	-	-	-	VARGRAPHIC	-	0	N
VARGRAPHIC	1	16336	-	-	-	-	VARGRAPHIC	-	0	N

DB2 for iSeries データ・ソース

表 138. DB2 for iSeries のデフォルトのフォワード・データ型マッピング (表示されていない列があります)

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHAR	1	254	-	-	-	-	CHAR	-	0	N
CHAR	255	32672	-	-	-	-	VARCHAR	-	0	N
CHAR	1	254	-	-	Y	-	CHAR	-	0	Y
CHAR	255	32672	-	-	Y	-	VARCHAR	-	0	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	-	-	-	-	-	DATE	-	0	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
FLOAT	4	-	-	-	-	-	REAL	-	-	-
FLOAT	8	-	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	1	127	-	-	-	-	GRAPHIC	-	0	N
GRAPHIC	128	16336	-	-	-	-	VARGRAPHIC	-	0	N
INTEGER	-	-	-	-	-	-	INTEGER	-	0	-
NUMERIC	-	-	-	-	-	-	DECIMAL	-	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	-	0	-
TIME	-	-	-	-	-	-	TIME	-	0	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
TIMESTMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	0	N
VARCHAR	1	32672	-	-	Y	-	VARCHAR	-	0	Y
VARG	1	16336	-	-	-	-	VARGRAPHIC	-	0	N
VARGRAPHIC	1	16336	-	-	-	-	VARGRAPHIC	-	0	N

DB2 Server for VM and VSE データ・ソース

表 139. DB2 Server for VM and VSE のデフォルトのフォワード・データ型マッピング (表示されていない列があります)

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHAR	1	254	-	-	-	-	CHAR	-	0	N
CHAR	1	254	-	-	Y	-	CHAR	-	0	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	-	-	-	-	-	DATE	-	0	-
DBAHW	-	-	-	-	-	-	SMALLINT	-	0	-
DBAINT	-	-	-	-	-	-	INTEGER	-	0	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
FLOAT	4	-	-	-	-	-	REAL	-	-	-
FLOAT	8	-	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	1	127	-	-	-	-	GRAPHIC	-	0	N
INTEGER	-	-	-	-	-	-	INTEGER	-	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	-	-	-
TIME	-	-	-	-	-	-	TIME	-	0	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
TIMESTMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	0	N
VARCHAR	1	32672	-	-	Y	-	VARCHAR	-	0	Y
VARGRAPHIC	1	16336	-	-	-	-	VARGRAPHIC	-	0	N
VARGRAPH	1	16336	-	-	-	-	VARGRAPHIC	-	0	N

DB2 for Linux、DB2 for UNIX、および DB2 for Windows データ・ソース

表 140. DB2 for Linux、DB2 for UNIX、および DB2 for Windows のデフォルトのフォワード・データ型マッピング (表示されていない列があります)

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
BIGINT	-	-	-	-	-	-	BIGINT	-	0	-
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHAR	-	-	-	-	-	-	CHAR	-	0	N
CHAR	-	-	-	-	Y	-	CHAR	-	0	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	-	-	-	-	-	DATE	-	0	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	-	-	-	-	-	DOUBLE	-	-	-
FLOAT	-	-	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	0	N
INTEGER	-	-	-	-	-	-	INTEGER	-	0	-
LONGVAR	-	-	-	-	N	-	CLOB	-	-	-
LONGVAR	-	-	-	-	Y	-	BLOB	-	-	-
LONGVARG	-	-	-	-	-	-	DBCLOB	-	-	-
REAL	-	-	-	-	-	-	REAL	-	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	-	0	-
TIME	-	-	-	-	-	-	TIME	-	0	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
TIMESTMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	0	N
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	0	Y
VARGRAPH	-	-	-	-	-	-	VARGRAPHIC	-	0	N
VARGRAPHIC	-	-	-	-	-	-	VARGRAPHIC	-	0	N

Informix データ・ソース

表 141. Informix のデフォルトのフォワード・データ型マッピング (表示されていない列があります)

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
BLOB	-	-	-	-	-	-	BLOB	2147483647	-	-
BOOLEAN	-	-	-	-	-	-	CHARACTER	1	-	-
BYTE	-	-	-	-	-	-	BLOB	2147483647	-	-
CHAR	1	254	-	-	-	-	CHARACTER	-	-	-
CHAR	255	32672	-	-	-	-	VARCHAR	-	-	-
CLOB	-	-	-	-	-	-	CLOB	2147483647	-	-
DATE	-	-	-	-	-	-	DATE	4	-	-
DATETIME	0	4	0	4	-	-	DATE	4	-	-
DATETIME	6	10	6	10	-	-	TIME	3	-	-
DATETIME	0	4	6	15	-	-	TIMESTAMP	10	-	-
DATETIME	6	10	11	15	-	-	TIMESTAMP	10	-	-
DECIMAL	1	31	0	31	-	-	DECIMAL	-	-	-
DECIMAL	32	130	-	-	-	-	DOUBLE	8	-	-
FLOAT	-	-	-	-	-	-	DOUBLE	8	-	-
INTEGER	-	-	-	-	-	-	INTEGER	4	-	-
INTERVAL	-	-	-	-	-	-	VARCHAR	25	-	-
INT8	-	-	-	-	-	-	BIGINT	19	0	-
LVARCHAR	1	32672	-	-	-	-	VARCHAR	-	-	-
MONEY	1	31	0	31	-	-	DECIMAL	-	-	-
MONEY	32	32	-	-	-	-	DOUBLE	8	-	-
NCHAR	1	254	-	-	-	-	CHARACTER	-	-	-
NCHAR	255	32672	-	-	-	-	VARCHAR	-	-	-
NVARCHAR	1	32672	-	-	-	-	VARCHAR	-	-	-
REAL	-	-	-	-	-	-	REAL	4	-	-
SERIAL	-	-	-	-	-	-	INTEGER	4	-	-
SERIAL8	-	-	-	-	-	-	BIGINT	-	-	-
SMALLFLOAT	-	-	-	-	-	-	REAL	4	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	2	-	-
TEXT	-	-	-	-	-	-	CLOB	2147483647	-	-
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	-	-

表 141. *Informix* のデフォルトのフォワード・データ型マッピング (表示されていない列があります) (続き)

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA

注:

| • Informix DATETIME データ型の場合、DB2 UNIX および Windowsフェデレーテッド・サーバーは、 Informix の高
| 水準修飾子を REMOTE_LENGTH として使用し、 Informix の低水準修飾子を REMOTE_SCALE として使用しま
| す。

| Informix 修飾子は、Informix Client SDK `datatime.h` ファイルに定義されている "TU_" 定数です。その定数は次の
| とおりです。

0 = YEAR	8 = MINUTE	13 = FRACTION(3)
2 = MONTH	10 = SECOND	14 = FRACTION(4)
4 = DAY	11 = FRACTION(1)	15 = FRACTION(5)
6 = HOUR	12 = FRACTION(2)	

Microsoft SQL Server データ・ソース

表 142. *Microsoft SQL Server* のデフォルトのフォワード・データ型マッピング

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
bigint [4]	-	-	-	-	-	-	BIGINT	-	-	-
binary	1	254	-	-	-	-	CHARACTER	-	-	Y
binary	255	8000	-	-	-	-	VARCHAR	-	-	Y
bit	-	-	-	-	-	-	SMALLINT	2	-	-
char	1	254	-	-	-	-	CHAR	-	-	N
char	255	8000	-	-	-	-	VARCHAR	-	-	N
datetime	-	-	-	-	-	-	TIMESTAMP	10	-	-
datetimen	-	-	-	-	-	-	TIMESTAMP	10	-	-
decimal	1	31	0	31	-	-	DECIMAL	-	-	-

表 142. *Microsoft SQL Server* のデフォルトのフォワード・データ型マッピング *(続き)*

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
decimal	32	38	0	38	-	-	DOUBLE	-	-	-
decimaln	1	31	0	31	-	-	DECIMAL	-	-	-
decimaln	32	38	0	38	-	-	DOUBLE	-	-	-
DUMMY65 [1]	1	38	-84	127	-	-	DOUBLE	-	-	-
DUMMY2000 [3]	1	38	-84	127	-	-	DOUBLE	-	-	-
float	-	8	-	-	-	-	DOUBLE	8	-	-
floatn	-	8	-	-	-	-	DOUBLE	8	-	-
float	-	4	-	-	-	-	REAL	4	-	-
floatn	-	4	-	-	-	-	REAL	4	-	-
image	-	-	-	-	-	-	BLOB	2147483647	-	Y
int	-	-	-	-	-	-	INTEGER	4	-	-
intn	-	-	-	-	-	-	INTEGER	4	-	-
money	-	-	-	-	-	-	DECIMAL	19	4	-
moneyn	-	-	-	-	-	-	DECIMAL	19	4	-
nchar	1	127	-	-	-	-	CHAR	-	-	N
nchar	128	4000	-	-	-	-	VARCHAR	-	-	N
numeric	1	31	0	31	-	-	DECIMAL	-	-	-
numeric	32	38	0	38	-	-	DOUBLE	8	-	-
numericn	32	38	0	38	-	-	DOUBLE	-	-	-
numericn	1	31	0	31	-	-	DECIMAL	-	-	-
ntext [2]	-	-	-	-	-	-	CLOB	2147483647	-	Y
nvarchar	1	4000	-	-	-	-	VARCHAR	-	-	N
real	-	-	-	-	-	-	REAL	4	-	-
smallint	-	-	-	-	-	-	SMALLINT	2	-	-
smalldatetime	-	-	-	-	-	-	TIMESTAMP	10	-	-
smallmoney	-	-	-	-	-	-	DECIMAL	10	4	-
smallmoneyn	-	-	-	-	-	-	DECIMAL	10	4	-
SQL_BIGINT	-	-	-	-	-	-	DECIMAL	-	-	-
SQL_BIGINT [4]	-	-	-	-	-	-	BIGINT	-	-	-
SQL_BINARY	1	254	-	-	-	-	CHARACTER	-	-	Y
SQL_BINARY	255	8000	-	-	-	-	VARCHAR	-	-	Y
SQL_BIT	-	-	-	-	-	-	SMALLINT	2	-	-

表 142. *Microsoft SQL Server* のデフォルトのフォワード・データ型マッピング (続き)

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
SQL_CHAR	1	254	-	-	-	-	CHAR	-	-	N
SQL_CHAR	255	8000	-	-	-	-	VARCHAR	-	-	N
SQL_DATE	-	-	-	-	-	-	DATE	4	-	-
SQL_DECIMAL	1	31	0	31	-	-	DECIMAL	-	-	-
SQL_DECIMAL	32	38	0	38	-	-	DOUBLE	8	-	-
SQL_DECIMAL	32	32	0	31	-	-	DOUBLE	8	-	-
SQL_DOUBLE	-	-	-	-	-	-	DOUBLE	8	-	-
SQL_FLOAT	-	-	-	-	-	-	DOUBLE	8	-	-
SQL_GUID [2]	1	4000	-	-	Y	-	VARCHAR	16	-	Y
SQL_INTEGER	-	-	-	-	-	-	INTEGER	4	-	-
SQL_LONGVARCHAR	-	-	-	-	-	-	CLOB	2147483647	-	N
SQL_LONGVARBINARY	-	-	-	-	-	-	BLOB	-	-	Y
SQL_NUMERIC	1	31	0	31	-	-	DECIMAL	-	-	-
SQL_REAL	-	-	-	-	-	-	DOUBLE	8	-	-
SQL_SMALLINT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_TIME	-	-	-	-	-	-	TIME	3	-	-
SQL_TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	10	-	-
SQL_TINYINT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_VARBINARY	1	8000	-	-	-	-	VARCHAR	-	-	Y
SQL_VARCHAR	1	8000	-	-	-	-	VARCHAR	-	-	N
text	-	-	-	-	-	-	CLOB	-	-	N
timestamp	-	-	-	-	-	-	VARCHAR	8		Y
tinyint	-	-	-	-	-	-	SMALLINT	2	-	-
固有 ID [2]	1	4000	-	-	Y	-	VARCHAR	16	-	Y
varbinary	1	8000	-	-	-	-	VARCHAR	-	-	Y
varchar	1	8000	-	-	-	-	VARCHAR	-	-	N

注:

1. このタイプ・マッピングは Microsoft SQL Server バージョン 6.5 でのみ有効。
2. このタイプ・マッピングは Microsoft SQL Server バージョン 7 およびバージョン 2000 でのみ有効。
3. このタイプ・マッピングは Windows 2000 オペレーティング・システムでのみ有効。
4. このタイプ・マッピングは Microsoft SQL Server バージョン 2000 でのみ有効。

ODBC データ・ソース

表 143. ODBC のデフォルトのフォワード・データ型マッピング (表示されていない列があります)

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
SQL_BIGINT	-	-	-	-	-	-	BIGINT	8	-	-
SQL_BINARY	1	254	-	-	-	-	CHARACTER	-	-	Y
SQL_BINARY	255	32672	-	-	-	-	VARCHAR	-	-	Y
SQL_BIT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_CHAR	1	254	-	-	-	-	CHAR	-	-	N
SQL_CHAR	255	32672	-	-	-	-	VARCHAR	-	-	N
SQL_DECIMAL	1	31	0	31	-	-	DECIMAL	-	-	-
SQL_DECIMAL	32	38	0	38	-	-	DOUBLE	8	-	-
SQL_DOUBLE	-	-	-	-	-	-	DOUBLE	8	-	-
SQL_FLOAT	-	-	-	-	-	-	DOUBLE	8	-	-
SQL_INTEGER	-	-	-	-	-	-	INTEGER	4	-	-
SQL_LONGVARCHAR	-	-	-	-	-	-	CLOB	2147483647	-	N
SQL_LONGVARBINARY	-	-	-	-	-	-	BLOB	-	-	Y
SQL_NUMERIC	1	31	0	31	-	-	DECIMAL	-	-	-
SQL_NUMERIC	32	32	0	31	-	-	DOUBLE	8	-	-
SQL_REAL	-	-	-	-	-	-	REAL	4	-	-
SQL_SMALLINT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_TYPE_DATE	-	-	-	-	-	-	DATE	4	-	-
SQL_TYPE_TIME	-	-	-	-	-	-	TIME	3	-	-
SQL_TYPE_TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	10	-	-
SQL_TINYINT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_VARBINARY	1	32672	-	-	-	-	VARCHAR	-	-	Y
SQL_VARCHAR	1	32672	-	-	-	-	VARCHAR	-	-	N
SQL_WCHAR	1	127	-	-	-	-	CHAR	-	-	N
SQL_WCHAR	128	16336	-	-	-	-	VARCHAR	-	-	N
SQL_WVARCHAR	1	16336	-	-	-	-	VARCHAR	-	-	N
SQL_WLONGVARCHAR	-	1073741823	-	-	-	-	CLOB	2147483647	-	N

Oracle NET8 データ・ソース

表 144. Oracle NET8 のデフォルトのフォワード・データ型マッピング

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
BLOB	0	0	0	0	-	¥0	BLOB	2147483647	0	Y
CHAR	1	254	0	0	-	¥0	CHAR	0	0	N
CHAR	255	2000	0	0	-	¥0	VARCHAR	0	0	N
CLOB	0	0	0	0	-	¥0	CLOB	2147483647	0	N
DATE	0	0	0	0	-	¥0	TIMESTAMP	0	0	N
FLOAT	1	126	0	0	-	¥0	DOUBLE	0	0	N
LONG	0	0	0	0	-	¥0	CLOB	2147483647	0	N
LONG RAW	0	0	0	0	-	¥0	BLOB	2147483647	0	Y
MLSLABEL	0	0	0	0	-	¥0	VARCHAR	255	0	N
NUMBER	1	38	-84	127	-	¥0	DOUBLE	0	0	N
NUMBER	1	31	0	31	-	>=	DECIMAL	0	0	N
NUMBER	1	4	0	0	-	¥0	SMALLINT	0	0	N
NUMBER	5	9	0	0	-	¥0	INTEGER	0	0	N
NUMBER	-	10	0	0	-	¥0	DECIMAL	0	0	N
RAW	1	2000	0	0	-	¥0	VARCHAR	0	0	Y
ROWID	0	0	0	NULL	-	¥0	CHAR	18	0	N
TIMESTAMP [1]	-	-	-	-	-	-	TIMESTAMP	10	-	-
VARCHAR2	1	4000	0	0	-	¥0	VARCHAR	0	0	N

注:

1. このタイプ・マッピングは、Oracle 9i (またはそれ以上) のクライアントとサーバー構成でのみ有効です。

Sybase データ・ソース

表 145. Sybase CTLIB のデフォルトのフォワード・データ型マッピング

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
binary	1	254	-	-	-	-	CHAR	-	-	Y
binary	255	16384	-	-	-	-	VARCHAR	-	-	Y
bit	-	-	-	-	-	-	SMALLINT	-	-	-
char	1	254	-	-	-	-	CHAR	-	-	N
char	255	16384	-	-	-	-	VARCHAR	-	-	N
char null (varchar を参照)										
datetime	-	-	-	-	-	-	TIMESTAMP	-	-	-
datetimn	-	-	-	-	-	-	TIMESTAMP	-	-	-
decimal	1	31	0	31	-	-	DECIMAL	-	-	-
decimal	32	38	0	38	-	-	DOUBLE	-	-	-
decimaln	1	31	0	31	-	-	DECIMAL	-	-	-
decimaln	32	38	0	38	-	-	DOUBLE	-	-	-
float	-	4	-	-	-	-	REAL	-	-	-
float	-	8	-	-	-	-	DOUBLE	-	-	-
floatn	-	4	-	-	-	-	REAL	-	-	-
floatn	-	8	-	-	-	-	DOUBLE	-	-	-
image	-	-	-	-	-	-	BLOB	-	-	-
int	-	-	-	-	-	-	INTEGER	-	-	-
intn	-	-	-	-	-	-	INTEGER	-	-	-
money	-	-	-	-	-	-	DECIMAL	19	4	-
moneyn	-	-	-	-	-	-	DECIMAL	19	4	-
nchar	1	254	-	-	-	-	CHAR	-	-	N
nchar	255	16384	-	-	-	-	VARCHAR	-	-	N
nchar null (nvarchar を参照)										
numeric	1	31	0	31	-	-	DECIMAL	-	-	-
numeric	32	38	0	38	-	-	DOUBLE	-	-	-
numericn	1	31	0	31	-	-	DECIMAL	-	-	-
numericn	32	38	0	38	-	-	DOUBLE	-	-	-
nvarchar	1	16384	-	-	-	-	VARCHAR	-	-	N

表 145. *Sybase CTLIB* のデフォルトのフォワード・データ型マッピング *(続き)*

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
real	-	-	-	-	-	-	REAL	-	-	-
smalldatetime	-	-	-	-	-	-	TIMESTAMP	-	-	-
smallint	-	-	-	-	-	-	SMALLINT	-	-	-
smallmoney	-	-	-	-	-	-	DECIMAL	10	4	-
sysname	1	254	-	-	-	-	CHAR	-	-	N
text	-	-	-	-	-	-	CLOB	-	-	-
timestamp	-	-	-	-	-	-	VARCHAR	8	-	Y
tinyint	-	-	-	-	-	-	SMALLINT	-	-	-
unichar[1]	1	254	-	-	-	-	CHAR	-	-	N
unichar[1]	255	16384	-	-	-	-	VARCHAR	-	-	N
unichar null (univarchar を参照)										
univarchar[1]	1	16384	-	-	-	-	VARCHAR	-	-	N
varbinary	1	16384	-	-	-	-	VARCHAR	-	-	Y
varchar	1	16384	-	-	-	-	VARCHAR	-	-	N

注:

1. Unicode 以外のフェデレーテッド・データベースで有効。

Teradata データ・ソース

表 146. *Teradata* のデフォルトのフォワード・データ型マッピング *(表示されていない列があります)*

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
BYTE	1	254	-	-	-	-	CHAR	-	-	Y
BYTE	255	32672	-	-	-	-	VARCHAR	-	-	Y

表 146. Teradata のデフォルトのフォワード・データ型マッピング (表示されていない列があります) (続き)

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
BYTE	32673	64000	-	-	-	-	BLOB	-	-	-
BYTEINT	-	-	-	-	-	-	SMALLINT	-	-	-
CHAR	1	254	-	-	-	-	CHARACTER	-	-	-
CHAR	255	32672	-	-	-	-	VARCHAR	-	-	-
CHAR	32673	64000	-	-	-	-	CLOB	-	-	-
DATE	-	-	-	-	-	-	DATE	-	-	-
DECIMAL	1	18	0	18	-	-	DECIMAL	-	-	-
DOUBLE PRECISION	-	-	-	-	-	-	DOUBLE	-	-	-
FLOAT	-	-	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	1	127	-	-	-	-	GRAPHIC	-	-	-
GRAPHIC	128	16336	-	-	-	-	VARGRAPHIC	-	-	-
GRAPHIC	16337	32000	-	-	-	-	DBCLOB	-	-	-
INTEGER	-	-	-	-	-	-	INTEGER	-	-	-
INTERVAL	-	-	-	-	-	-	CHAR	-	-	-
NUMERIC	1	18	0	18	-	-	DECIMAL	-	-	-
REAL	-	-	-	-	-	-	DOUBLE	-	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	-	-	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	-	-	-
VARBYTE	1	32762	-	-	-	-	VARCHAR	-	-	Y
VARBYTE	32763	64000	-	-	-	-	BLOB	-	-	-
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	-	-
VARCHAR	32673	64000	-	-	-	-	CLOB	-	-	-
VARGRAPHIC	1	16336	-	-	-	-	VARGRAPHIC	-	-	-
VARGRAPHIC	16337	32000	-	-	-	-	DBCLOB	-	-	-

関連概念:

- 「フェデレーテッド・システム・ガイド」の『順方向データ・タイプ・マッピングと逆方向データ・タイプ・マッピング』

関連資料:

- 「フェデレーテッド・システム・ガイド」の『長形式データ・タイプから varchar データ・タイプへの変更』

- 「フェデレーテッド・システム・ガイド」の『Unicode のデフォルトの順方向データ・タイプ・マッピング - NET8 ラッパー』
- 「フェデレーテッド・システム・ガイド」の『Unicode のデフォルトの順方向データ・タイプ・マッピング - Sybase ラッパー』
- 「フェデレーテッド・システム・ガイド」の『Unicode のデフォルトの順方向データ・タイプ・マッピング - ODBC ラッパー』
- 「フェデレーテッド・システム・ガイド」の『Unicode のデフォルトの順方向データ・タイプ・マッピング - Microsoft SQL Server ラッパー』

デフォルトのリバース・データ型マッピング

データ・ソースのデータ型とフェデレーテッド・データベースのデータ型の間のマッピングには、フォワード・タイプのマッピングとリバース・タイプのマッピングの 2 種類があります。*フォワード・タイプ・マッピング* では、マッピングはリモートのタイプから対応するローカル・タイプへのマッピングです。 マッピングのもう一方のタイプは*リバース・タイプ・マッピング* であり、これはリモートの表を作成または変更するために、トランスペアレント DDL で使用されます。

ほとんどのデータ・ソースの場合、ラッパー内にデフォルトのタイプ・マッピングがあります。 DB2 ファミリーのデータ・ソース用のデフォルトのタイプ・マッピングは、 DRDA ラッパーにあります。Informix 用のデフォルトのタイプ・マッピングは INFORMIX ラッパーなどにあります。

DB2 フェデレーテッド・データベースにリモートの表またはビューを定義する際、その定義には、リバース・タイプ・マッピングが入ります。マッピングは、それぞれの列の*ローカル* の DB2 for Linux、DB2 for UNIX、および DB2 for Windows のデータ型、およびそれに対応するリモートのデータ型から行われます。たとえば、ローカル型 REAL が Informix 型 SMALLFLOAT を指す、デフォルトのリバース・タイプ・マッピングがあります。

DB2 for Linux、DB2 for UNIX、および DB2 for Windows フェデレーテッド・サーバーは、データ型 LONG VARCHAR、LONG VARGRAPHIC、DATALINK、およびユーザー定義タイプのマッピングをサポートしません。

CREATE TABLE ステートメントを使用してリモート表を作成するときに、リモート表に入れたいローカル・データ型を指定します。これらのデフォルトのリバース・タイプ・マッピングは、これらの列に対応するリモート・タイプを割り当てます。たとえば、CREATE TABLE ステートメントを使用して、列 C2 を持つ Informix の表を定義するとします。ステートメント内で C2 のデータ型として BIGINT を指定します。 BIGINT のデフォルトのリバース・タイプ・マッピングは、どのバージョンの Informix で表を作成しているかにより異なります。 Informix バージョン 8 では Informix 表の C2 のマッピングは DECIMAL となり、Informix バージョン 9 では INT8 になります。

デフォルトのリバース・タイプ・マッピングをオーバーライドすることも、CREATE TYPE MAPPING ステートメントを使用して新しいリバース・タイプ・マッピングを作成することもできます。

下表は、DB2 for Linux、DB2 for UNIX、および DB2 for Windows のローカル・データ型とデータ・ソースのリモート・データ型の間のデフォルトのリバース・マッピングを示しています。

これらのマッピングは、特に注記のないかぎり、サポート対象のすべてのバージョンで有効です。

DB2 for z/OS and OS/390 データ・ソース

表 147. DB2 for z/OS and OS/390 のデフォルトのリバース・データ型マッピング (表示されていない列があります)

FEDERATED_TYPENAME	FEDERATED_LOWER_LEN	FEDERATED_UPPER_LEN	FEDERATED_LOWER_SCALE	FEDERATED_UPPER_SCALE	FEDERATED_BIT_DATA	FEDERATED_DATA_OPERATORS	REMOTE_TYPENAME	REMOTE_LENGTH	REMOTE_SCALE	REMOTE_BIT_DATA
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHARACTER	-	-	-	-	-	-	CHAR	-	-	N
CHARACTER	-	-	-	-	Y	-	CHAR	-	-	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	4	-	-	-	-	DATE	-	-	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	8	-	-	-	-	DOUBLE	-	-	-
FLOAT	-	8	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	N
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
REAL	-	4	-	-	-	-	REAL	-	-	-
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
TIME	-	3	-	-	-	-	TIME	-	-	-
TIMESTAMP	-	10	-	-	-	-	TIMESTAMP	-	-	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	N
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	-	Y
VARGRAPHIC	-	-	-	-	-	-	VARGRAPHIC	-	-	N

DB2 for iSeries データ・ソース

表 148. DB2 for iSeries のデフォルトのリバース・データ型マッピング (表示されていない列があります)

FEDERATED_TYPENAME	FEDERATED_LOWER_LEN	FEDERATED_UPPER_LEN	FEDERATED_LOWER_SCALE	FEDERATED_UPPER_SCALE	FEDERATED_BIT_DATA	FEDERATED_DATA_OPERATORS	REMOTE_TYPENAME	REMOTE_LENGTH	REMOTE_SCALE	REMOTE_BIT_DATA
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHARACTER	-	-	-	-	-	-	CHARACTER	-	-	N
CHARACTER	-	-	-	-	Y	-	CHARACTER	-	-	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	4	-	-	-	-	DATE	-	-	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	NUMERIC	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	8	-	-	-	-	FLOAT	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	N
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
REAL	-	4	-	-	-	-	FLOAT	-	-	-
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
TIME	-	3	-	-	-	-	TIME	-	-	-
TIMESTAMP	-	10	-	-	-	-	TIMESTAMP	-	-	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	N
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	-	Y
VARGRAPHIC	-	-	-	-	-	-	VARG	-	-	N

DB2 for VM and VSE データ・ソース

表 149. DB2 for VM and VSE のデフォルトのリバース・データ型マッピング (表示されていない列があります)

FEDERATED_TYPENAME	FEDERATED_LOWER_LEN	FEDERATED_UPPER_LEN	FEDERATED_LOWER_SCALE	FEDERATED_UPPER_SCALE	FEDERATED_BIT_DATA	FEDERATED_DATA_OPERATORS	REMOTE_TYPENAME	REMOTE_LENGTH	REMOTE_SCALE	REMOTE_BIT_DATA
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHARACTER	-	-	-	-	-	-	CHAR	-	-	-
CHARACTER	-	-	-	-	Y	-	CHAR	-	-	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	4	-	-	-	-	DATE	-	-	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	8	-	-	-	-	FLOAT	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	N
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
REAL	-	4	-	-	-	-	REAL	-	-	-
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
TIME	-	3	-	-	-	-	TIME	-	-	-
TIMESTAMP	-	10	-	-	-	-	TIMESTAMP	-	-	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	-
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	-	Y
VARGRAPH	-	-	-	-	-	-	VARGRAPH	-	-	N

DB2 for Linux、DB2 for UNIX、および DB2 for Windows データ・ソース

表 150. DB2 for Linux、DB2 for UNIX、および DB2 for Windows のデフォルトのリバース・データ型マッピング (表示されていない列があります)

FEDERATED_TYPENAME	FEDERATED_LOWER_LEN	FEDERATED_UPPER_LEN	FEDERATED_LOWER_SCALE	FEDERATED_UPPER_SCALE	FEDERATED_BIT_DATA	FEDERATED_DATA_OPERATORS	REMOTE_TYPENAME	REMOTE_LENGTH	REMOTE_SCALE	FEDERATED_BIT_DATA
BIGINT	-	8	-	-	-	-	BIGINT	-	-	-
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHARACTER	-	-	-	-	-	-	CHAR	-	-	N
CHARACTER	-	-	-	-	Y	-	CHAR	-	-	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	4	-	-	-	-	DATE	-	-	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	8	-	-	-	-	DOUBLE	-	-	-
FLOAT	-	8	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	N
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
REAL	-	-	-	-	-	-	REAL	-	-	-
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
TIME	-	3	-	-	-	-	TIME	-	-	-
TIMESTAMP	-	10	-	-	-	-	TIMESTAMP	-	-	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	N
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	-	Y
VARGRAPH	-	-	-	-	-	-	VARGRAPHIC	-	-	N
VARGRAPHIC	-	-	-	-	-	-	VARGRAPHIC	-	-	-

Informix データ・ソース

表 151. Informix のデフォルトのリバース・データ型マッピング

FEDERATED_TYPENAME	FEDERATED_LOWER_LEN	FEDERATED_UPPER_LEN	FEDERATED_LOWER_SCALE	FEDERATED_UPPER_SCALE	FEDERATED_BIT_DATA	FEDERATED_DATA_OPERATORS	REMOTE_TYPENAME	REMOTE_LENGTH	REMOTE_SCALE	REMOTE_BIT_DATA
BIGINT[1]	-	-	-	-	-	-	DECIMAL	19	-	-
BIGINT [2]	-	-	-	-	-	-	INT8	-	-	-
BLOB	1	2147483647	-	-	-	-	BYTE	-	-	-
CHARACTER	-	-	-	-	N	-	CHAR	-	-	-
CHARACTER	-	-	-	-	Y	-	BYTE	-	-	-
CLOB	1	2147483647	-	-	-	-	TEXT	-	-	-
DATE	-	4	-	-	-	-	DATE	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	8	-	-	-	-	FLOAT	-	-	-
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
REAL	-	4	-	-	-	-	SMALLFLOAT	-	-	-
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
TIME	-	3	-	-	-	-	DATETIME	6	10	-
TIMESTAMP	-	10	-	-	-	-	DATETIME	0	15	-
VARCHAR	1	254	-	-	N	-	VARCHAR	-	-	-
VARCHAR	255	32672	-	-	N	-	TEXT	-	-	-
VARCHAR	-	-	-	-	Y	-	BYTE	-	-	-
VARCHAR [2]	255	2048	-	-	N	-	LVARCHAR	-	-	-
VARCHAR [2]	2049	32672	-	-	N	-	TEXT	-	-	-

注:

1. このタイプ・マッピングは Informix サーバーのバージョン 8 (以下) でのみ有効。
2. このタイプ・マッピングは Informix サーバーのバージョン 9 でのみ有効。

Informix DATETIME データ型の場合、DB2 UNIX および Windowsフェデレーテッド・サーバーは、 Informix の高水準修飾子を REMOTE_LENGTH として使用し、 Informix の低水準修飾子を REMOTE_SCALE として使用します。

Informix 修飾子は、Informix Client SDK `datatime.h` ファイルに定義されている "TU_" 定数です。その定数は次のとおりです。

0 = YEAR	8 = MINUTE	13 = FRACTION(3)
2 = MONTH	10 = SECOND	14 = FRACTION(4)
4 = DAY	11 = FRACTION(1)	15 = FRACTION(5)

表 151. *Informix* のデフォルトのリバース・データ型マッピング (続き)

FEDERATED_TYPENAME	FEDERATED_LOWER_LEN	FEDERATED_UPPER_LEN	FEDERATED_LOWER_SCALE	FEDERATED_UPPER_SCALE	FEDERATED_BIT_DATA	FEDERATED_DATA_OPERATORS	REMOTE_TYPENAME	REMOTE_LENGTH	REMOTE_SCALE	REMOTE_BIT_DATA
6 = HOUR		12 = FRACTION(2)								

Microsoft SQL Server データ・ソース

表 152. *Microsoft SQL Server* のデフォルトのリバース・データ型マッピング (表示されていない列があります)

FEDERATED_TYPENAME	FEDERATED_LOWER_LEN	FEDERATED_UPPER_LEN	FEDERATED_LOWER_SCALE	FEDERATED_UPPER_SCALE	FEDERATED_BIT_DATA	FEDERATED_DATA_OPERATORS	REMOTE_TYPENAME	REMOTE_LENGTH	REMOTE_SCALE	REMOTE_BIT_DATA
BIGINT[1]	-	-	-	-	-	-	bigint	-	-	-
BLOB	-	-	-	-	-	-	image	-	-	-
CHARACTER	-	-	-	-	Y	-	binary	-	-	-
CHARACTER	-	-	-	-	N	-	char	-	-	-
CLOB	-	-	-	-	-	-	text	-	-	-
DATE	-	4	-	-	-	-	datetime	-	-	-
DECIMAL	-	-	-	-	-	-	decimal	-	-	-
DOUBLE	-	8	-	-	-	-	float	-	-	-
INTEGER	-	-	-	-	-	-	int	-	-	-
SMALLINT	-	-	-	-	-	-	smallint	-	-	-
REAL	-	4	-	-	-	-	real	-	-	-
TIME	-	3	-	-	-	-	datetime	-	-	-
TIMESTAMP	-	10	-	-	-	-	datetime	-	-	-
VARCHAR	1	8000	-	-	N	-	varchar	-	-	-
VARCHAR	8001	32672	-	-	N	-	text	-	-	-
VARCHAR	1	8000	-	-	Y	-	varbinary	-	-	-

表 152. Microsoft SQL Server のデフォルトのリバース・データ型マッピング (表示されていない列があります) (続き)

FEDERATED_TYPENAME	FEDERATED_LOWER_LEN	FEDERATED_UPPER_LEN	FEDERATED_LOWER_SCALE	FEDERATED_UPPER_SCALE	FEDERATED_BIT_DATA	FEDERATED_DATA_OPERATORS	REMOTE_TYPENAME	REMOTE_LENGTH	REMOTE_SCALE	REMOTE_BIT_DATA
VARCHAR	8001	32672	-	-	Y	-	image	-	-	-

注:

1. このタイプ・マッピングは Microsoft SQL Server バージョン 2000 でのみ有効。

Oracle NET8 データ・ソース

表 153. Oracle NET8 のデフォルトのリバース・データ型マッピング

FEDERATED_TYPENAME	FEDERATED_LOWER_LEN	FEDERATED_UPPER_LEN	FEDERATED_LOWER_SCALE	FEDERATED_UPPER_SCALE	FEDERATED_BIT_DATA	FEDERATED_DATA_OPERATORS	REMOTE_TYPENAME	REMOTE_LENGTH	REMOTE_SCALE	REMOTE_BIT_DATA
BLOB	0	2147483647	0	0	Y	¥0	BLOB	0	0	Y
CHARACTER	1	254	0	0	N	¥0	CHAR	0	0	N
CHARACTER	1	254	0	0	Y	¥0	RAW	0	0	Y
CLOB	0	2147483647	0	0	N	¥0	CLOB	0	0	N
DATE	0	4	0	0	N	¥0	DATE	0	0	N
DECIMAL	0	0	0	0	N	¥0	NUMBER	0	0	N
DOUBLE	0	8	0	0	N	¥0	FLOAT	126	0	N
FLOAT	0	8	0	0	N	¥0	FLOAT	126	0	N
INTEGER	0	4	0	0	N	¥0	NUMBER	9	0	N
REAL	0	4	0	0	N	¥0	FLOAT	63	0	N
SMALLINT	0	2	0	0	N	¥0	NUMBER	4	0	N
TIME	0	3	0	0	N	¥0	DATE	0	0	N
TIMESTAMP	0	10	0	0	N	¥0	DATE	0	0	N
VARCHAR	1	4000	0	0	N	¥0	VARCHAR2	0	0	N

表 153. Oracle NET8 のデフォルトのリバース・データ型マッピング (続き)

FEDERATED_TYPENAME	FEDERATED_LOWER_LEN	FEDERATED_UPPER_LEN	FEDERATED_LOWER_SCALE	FEDERATED_UPPER_SCALE	FEDERATED_BIT_DATA	FEDERATED_DATA_OPERATORS	REMOTE_TYPENAME	REMOTE_LENGTH	REMOTE_SCALE	REMOTE_BIT_DATA
VARCHAR	1	2000	0	0	Y	¥0	RAW	0	0	Y

注: DB2 Universal Database for Linux, UNIX, and Windows の BIGINT データ型はトランスペアレント (ユーザーが意識しない) DDL では使用できません。リモート Oracle 表を作成する時に CREATE TABLE ステートメントで BIGINT データ型を指定することはできません。

Sybase データ・ソース

表 154. Sybase CTLIB のデフォルトのリバース・データ型マッピング

FEDERATED_TYPENAME	FEDERATED_LOWER_LEN	FEDERATED_UPPER_LEN	FEDERATED_LOWER_SCALE	FEDERATED_UPPER_SCALE	FEDERATED_BIT_DATA	FEDERATED_DATA_OPERATORS	REMOTE_TYPENAME	REMOTE_LENGTH	REMOTE_SCALE	REMOTE_BIT_DATA
BIGINT	-	-	-	-	-	-	decimal	19	0	-
BLOB	-	-	-	-	-	-	image	-	-	-
CHARACTER	-	-	-	-	N	-	char	-	-	-
CHARACTER	-	-	-	-	Y	-	binary	-	-	-
CLOB	-	-	-	-	-	-	text	-	-	-
DATE	-	-	-	-	-	-	datetime	-	-	-
DECIMAL	-	-	-	-	-	-	decimal	-	-	-
DOUBLE	-	-	-	-	-	-	float	-	-	-
GRAPHIC	-	-	-	-	-	-	unichar	-	-	-
VARGRAPHIC	-	-	-	-	-	-	univarchar	-	-	-
INTEGER	-	-	-	-	-	-	integer	-	-	-
REAL	-	-	-	-	-	-	real	-	-	-
SMALLINT	-	-	-	-	-	-	smallint	-	-	-
TIME	-	-	-	-	-	-	datetime	-	-	-

表 154. Sybase CTLIB のデフォルトのリバース・データ型マッピング (続き)

FEDERATED_TYPENAME	FEDERATED_LOWER_LEN	FEDERATED_UPPER_LEN	FEDERATED_LOWER_SCALE	FEDERATED_UPPER_SCALE	FEDERATED_BIT_DATA	FEDERATED_DATA_OPERATORS	REMOTE_TYPENAME	REMOTE_LENGTH	REMOTE_SCALE	REMOTE_BIT_DATA
TIMESTAMP	-	-	-	-	-	-	datetime	-	-	-
VARCHAR[1]	1	255	-	-	N	-	varchar	-	-	-
VARCHAR[1]	256	32672	-	-	N	-	text	-	-	-
VARCHAR [2]	1	16384	-	-	N	-	varchar	-	-	-
VARCHAR [2]	16385	32672	-	-	N	-	text	-	-	-
VARCHAR[1]	1	255	-	-	Y	-	varbinary	-	-	-
VARCHAR[1]	256	32672	-	-	Y	-	image	-	-	-
VARCHAR [2]	1	16384	-	-	Y	-	varbinary	-	-	-
VARCHAR [2]	16385	32672	-	-	Y	-	image	-	-	-

注:

1. このタイプ・マッピングが有効なのは、Sybase サーバーのバージョン 12.0 以前での CTLIB の場合のみです。
2. このタイプ・マッピングが有効なのは、Sybase サーバーのバージョン 12.5 以降での CTLIB の場合のみです。

Teradata データ・ソース

表 155. Teradata のデフォルトのリバース・データ型マッピング (表示されていない列があります)

FEDERATED_TYPENAME	FEDERATED_LOWER_LEN	FEDERATED_UPPER_LEN	FEDERATED_LOWER_SCALE	FEDERATED_UPPER_SCALE	FEDERATED_BIT_DATA	FEDERATED_DATA_OPERATORS	REMOTE_TYPENAME	REMOTE_LENGTH	REMOTE_SCALE	FEDERATED_BIT_DATA
BLOB [1]	1	64000	-	-	-	-	VARBYTE	-	-	-
CHARACTER	-	-	-	-	-	-	CHARACTER	-	-	-
CHARACTER	-	-	-	-	Y	-	BYTE	-	-	-
CLOB [2]	1	64000	-	-		-	VARCHAR	-	-	-
DATE	-	-	-	-	-	-	DATE	-	-	-

表 155. *Teradata* のデフォルトのリバース・データ型マッピング *(*表示されていない列があります*) (*続き*)*

FEDERATED_TYPENAME	FEDERATED_LOWER_LEN	FEDERATED_UPPER_LEN	FEDERATED_LOWER_SCALE	FEDERATED_UPPER_SCALE	FEDERATED_BIT_DATA	FEDERATED_DATA_OPERATORS	REMOTE_TYPENAME	REMOTE_LENGTH	REMOTE_SCALE	FEDERATED_BIT_DATA
DBCLOB [3]	1	32000	-	-	-	-	VARGRAPHIC	-	-	-
DECIMAL	1	18	0	18	-	-	DECIMAL	-	-	-
DECIMAL	19	31	0	31	-	-	FLOAT	-	-	-
DOUBLE	-	-	-	-	-	-	FLOAT	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	-
INTEGER	-	-	-	-	-	-	INTEGER	-	-	-
REAL	-	-	-	-	-	-	FLOAT	-	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	-	-	-
TIME	-	-	-	-	-	-	TIME	-	-	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	-	-	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	-
VARCHAR	-	-	-	-	Y	-	VARBYTE	-	-	-
VARGRAPHIC	-	-	-	-	-	-	VARGRAPHIC	-	-	-

注:

1. Teradata VARBYTE データ型には、指定長 (1 から 64000 まで) の DB2 BLOB データ型のみを入れることができます。
2. Teradata VARCHAR データ型には、指定長 (1 から 64000 まで) の DB2 CLOB データ型のみを入れることができます。
3. Teradata VARGRAPHIC データ型には、指定長 (1 から 32000 まで) の DB2 DBCLOB データ型のみを入れることができます。

関連概念:

- 「フェデレーテッド・システム・ガイド」の『順方向データ・タイプ・マッピングと逆方向データ・タイプ・マッピング』

付録 F. SAMPLE データベース

DB2 資料の中のコード例の多くは、 SAMPLE データベースを使用します。以下は、SAMPLE データベース内の各表についての説明です。データベースを作成およびドロップする方法についての説明もあります。各表の初期データ値も示されています。ハイフン (-) は NULL 値を表しています。

SAMPLE データベースの作成

SAMPLE データベースを作成するには、DB2SAMPL コマンドを使用します。データベースを作成するためには、SYSADM 権限が必要です。

- **UNIX 系のプラットフォームを使う場合**

  オペレーティング・システムのコマンド・プロンプトを使っている場合は、データベース・マネージャーのインスタンス所有所のホーム・ディレクトリーから以下を発行します。

  ```
  sqllib/bin/db2sampl <path>
  ```

  path は、 SAMPLE データベースの作成先のパスを指定するオプション・パラメーターです。 path パラメーターを指定しないなら、 SAMPLE データベースはデータベース・マネージャー構成ファイルの中の DFTDBPATH パラメーターによって指定されるデフォルト・パスに作成されます。 DB2SAMPL のスキーマは、CURRENT SCHEMA 特殊レジスターの値です。

- **Windows プラットフォームを使う場合**

  オペレーティング・システムのコマンド・プロンプトを使っている場合は、以下を発行します。

  ```
  db2sampl e
  ```

  ここで、*e* はデータベースを作成するドライブを指定するオプション・パラメーターです。ドライブ・パラメーターの指定がない場合は、 SAMPLE データベースは、DB2 と同じドライブに作成されます。

SAMPLE データベースの消去

SAMPLE データベースにアクセスする必要がなくなった場合には、 DROP DATABASE コマンドを使用して、SAMPLE データベースを削除することができます。

```
db2 drop database sample
```

CL_SCHED 表

名前:	CLASS_CODE	DAY	STARTING	ENDING
タイプ:	char(7)	smallint	time	time

CL_SCHED 表

名前:	CLASS_CODE	DAY	STARTING	ENDING
説明:	クラス・コード (教室: 教師)	4 日単位の予定の日番号	クラス開始時刻	クラス終了時刻

DEPARTMENT 表

名前:	DEPTNO	DEPTNAME	MGRNO	ADMRDEPT	LOCATION
タイプ:	char(3) 非 NULL 値	varchar(29) 非 NULL 値	char(6)	char(3) 非 NULL 値	char(16)
説明:	部門番号	部門の一般的な活動を記述する名前	部門管理者の従業員番号 (EMPNO)	この部門の報告提出先の部門 (DEPTNO)	遠隔地の名前
値:	A00	SPIFFY COMPUTER SERVICE DIV.	000010	A00	-
	B01	PLANNING	000020	A00	-
	C01	INFORMATION CENTER	000030	A00	-
	D01	DEVELOPMENT CENTER	-	A00	-
	D11	MANUFACTURING SYSTEMS	000060	D01	-
	D21	ADMINISTRATION SYSTEMS	000070	D01	-
	E01	SUPPORT SERVICES	000050	A00	-
	E11	OPERATIONS	000090	E01	-
	E21	SOFTWARE SUPPORT	000100	E01	-

EMPLOYEE 表

名前:	EMPNO	FIRSTNME	MIDINIT	LASTNAME	WORKDEPT	PHONENO	HIREDATE
タイプ:	char(6) 非 NULL 値	varchar(12) 非 NULL 値	char(1) 非 NULL 値	varchar(15) 非 NULL 値	char(3)	char(4)	date
説明:	従業員番号	ファーストネーム (名)	ミドルネームのイニシャル	ラストネーム (姓)	従業員の所属部門 (DEPTNO)	電話番号	入社日

JOB	EDLEVEL	SEX	BIRTHDATE	SALARY	BONUS	COMM
char(8)	smallint 非 NULL 値	char(1)	date	dec(9,2)	dec(9,2)	dec(9,2)
担当業務	学校教育の年数	性別 (男性は M、女性は F)	誕生日	年収	年間ボーナス	年間歩合給

以下の表には、EMPLOYEE 表の値が入っています。

EMPLOYEE 表

EMPNO	FIRSTNME	MID INIT	LASTNAME	WORK DEPT	PHONE NO	HIREDATE	JOB	ED LEVEL	SEX	BIRTHDATE	SALARY	BONUS	COMM
char(6) 非 NULL 値	varchar(12) 非 NULL 値	char(1) 非 NULL 値	varchar(15) 非 NULL 値	char(3)	char(4)	date	char(8)	smallint 非 NULL 値	char(1)	date	dec(9,2)	dec(9,2)	dec(9,2)
000010	CHRISTINE	I	HAAS	A00	3978	1965-01-01	PRES	18	F	1933-08-24	52750	1000	4220
000020	MICHAEL	L	THOMPSON	B01	3476	1973-10-10	MANAGER	18	M	1948-02-02	41250	800	3300
000030	SALLY	A	KWAN	C01	4738	1975-04-05	MANAGER	20	F	1941-05-11	38250	800	3060
000050	JOHN	B	GEYER	E01	6789	1949-08-17	MANAGER	16	M	1925-09-15	40175	800	3214
000060	IRVING	F	STERN	D11	6423	1973-09-14	MANAGER	16	M	1945-07-07	32250	500	2580
000070	EVA	D	PULASKI	D21	7831	1980-09-30	MANAGER	16	F	1953-05-26	36170	700	2893
000090	EILEEN	W	HENDERSON	E11	5498	1970-08-15	MANAGER	16	F	1941-05-15	29750	600	2380
000100	THEODORE	Q	SPENSER	E21	0972	1980-06-19	MANAGER	14	M	1956-12-18	26150	500	2092
000110	VINCENZO	G	LUCCHESSI	A00	3490	1958-05-16	SALESREP	19	M	1929-11-05	46500	900	3720
000120	SEAN		O'CONNELL	A00	2167	1963-12-05	CLERK	14	M	1942-10-18	29250	600	2340
000130	DOLORES	M	QUINTANA	C01	4578	1971-07-28	ANALYST	16	F	1925-09-15	23800	500	1904
000140	HEATHER	A	NICHOLLS	C01	1793	1976-12-15	ANALYST	18	F	1946-01-19	28420	600	2274
000150	BRUCE		ADAMSON	D11	4510	1972-02-12	DESIGNER	16	M	1947-05-17	25280	500	2022
000160	ELIZABETH	R	PIANKA	D11	3782	1977-10-11	DESIGNER	17	F	1955-04-12	22250	400	1780
000170	MASATOSHI	J	YOSHIMURA	D11	2890	1978-09-15	DESIGNER	16	M	1951-01-05	24680	500	1974
000180	MARILYN	S	SCOUTTEN	D11	1682	1973-07-07	DESIGNER	17	F	1949-02-21	21340	500	1707
000190	JAMES	H	WALKER	D11	2986	1974-07-26	DESIGNER	16	M	1952-06-25	20450	400	1636
000200	DAVID		BROWN	D11	4501	1966-03-03	DESIGNER	16	M	1941-05-29	27740	600	2217
000210	WILLIAM		JONES	D11	0942	1979-04-11	DESIGNER	17	M	1953-02-23	18270	400	1462
000220	JENNIFER	K	LUTZ	D11	0672	1968-08-29	DESIGNER	18	F	1948-03-19	29840	600	2387
000230	JAMES	J	JEFFERSON	D21	2094	1966-11-21	CLERK	14	M	1935-05-30	22180	400	1774
000240	SALVATORE	M	MARINO	D21	3780	1979-12-05	CLERK	17	M	1954-03-31	28760	600	2301
000250	DANIEL	S	SMITH	D21	0961	1969-10-30	CLERK	15	M	1939-11-12	19180	400	1534
000260	SYBIL	P	JOHNSON	D21	8953	1975-09-11	CLERK	16	F	1936-10-05	17250	300	1380
000270	MARIA	L	PEREZ	D21	9001	1980-09-30	CLERK	15	F	1953-05-26	27380	500	2190
000280	ETHEL	R	SCHNEIDER	E11	8997	1967-03-24	OPERATOR	17	F	1936-03-28	26250	500	2100
000290	JOHN	R	PARKER	E11	4502	1980-05-30	OPERATOR	12	M	1946-07-09	15340	300	1227
000300	PHILIP	X	SMITH	E11	2095	1972-06-19	OPERATOR	14	M	1936-10-27	17750	400	1420
000310	MAUDE	F	SETRIGHT	E11	3332	1964-09-12	OPERATOR	12	F	1931-04-21	15900	300	1272
000320	RAMLAL	I	MEHTA	E21	9990	1965-07-07	FIELDREP	16	M	1932-08-11	19950	400	1596
000330	WING		LEE	E21	2103	1976-02-23	FIELDREP	14	M	1941-07-18	25370	500	2030
000340	JASON	R	GOUNOT	E21	5698	1947-05-05	FIELDREP	16	M	1926-05-17	23840	500	1907

EMP_ACT 表

名前:	EMPNO	PROJNO	ACTNO	EMPTIME	EMSTDATE	EMENDATE
タイプ:	char(6) 非 NULL 値	char(6) 非 NULL 値	smallint 非 NULL 値	dec(5,2)	date	date
説明:	従業員番号	プロジェクト番号	活動番号	従業員がプロジェクトに費やす時間の割合	活動開始日	活動終了日
値:	000010	AD3100	10	.50	1982-01-01	1982-07-01
	000070	AD3110	10	1.00	1982-01-01	1983-02-01
	000230	AD3111	60	1.00	1982-01-01	1982-03-15
	000230	AD3111	60	.50	1982-03-15	1982-04-15
	000230	AD3111	70	.50	1982-03-15	1982-10-15
	000230	AD3111	80	.50	1982-04-15	1982-10-15
	000230	AD3111	180	1.00	1982-10-15	1983-01-01
	000240	AD3111	70	1.00	1982-02-15	1982-09-15
	000240	AD3111	80	1.00	1982-09-15	1983-01-01
	000250	AD3112	60	1.00	1982-01-01	1982-02-01
	000250	AD3112	60	.50	1982-02-01	1982-03-15
	000250	AD3112	60	.50	1982-12-01	1983-01-01
	000250	AD3112	60	1.00	1983-01-01	1983-02-01
	000250	AD3112	70	.50	1982-02-01	1982-03-15
	000250	AD3112	70	1.00	1982-03-15	1982-08-15
	000250	AD3112	70	.25	1982-08-15	1982-10-15
	000250	AD3112	80	.25	1982-08-15	1982-10-15
	000250	AD3112	80	.50	1982-10-15	1982-12-01
	000250	AD3112	180	.50	1982-08-15	1983-01-01
	000260	AD3113	70	.50	1982-06-15	1982-07-01
	000260	AD3113	70	1.00	1982-07-01	1983-02-01
	000260	AD3113	80	1.00	1982-01-01	1982-03-01
	000260	AD3113	80	.50	1982-03-01	1982-04-15
	000260	AD3113	180	.50	1982-03-01	1982-04-15
	000260	AD3113	180	1.00	1982-04-15	1982-06-01
	000260	AD3113	180	.50	1982-06-01	1982-07-01
	000270	AD3113	60	.50	1982-03-01	1982-04-01
	000270	AD3113	60	1.00	1982-04-01	1982-09-01
	000270	AD3113	60	.25	1982-09-01	1982-10-15
	000270	AD3113	70	.75	1982-09-01	1982-10-15
	000270	AD3113	70	1.00	1982-10-15	1983-02-01
	000270	AD3113	80	1.00	1982-01-01	1982-03-01
	000270	AD3113	80	.50	1982-03-01	1982-04-01
	000030	IF1000	10	.50	1982-06-01	1983-01-01
	000130	IF1000	90	1.00	1982-01-01	1982-10-01
	000130	IF1000	100	.50	1982-10-01	1983-01-01
	000140	IF1000	90	.50	1982-10-01	1983-01-01
	000030	IF2000	10	.50	1982-01-01	1983-01-01

名前:	EMPNO	PROJNO	ACTNO	EMPTIME	EMSTDATE	EMENDATE
	000140	IF2000	100	1.00	1982-01-01	1982-03-01
	000140	IF2000	100	.50	1982-03-01	1982-07-01
	000140	IF2000	110	.50	1982-03-01	1982-07-01
	000140	IF2000	110	.50	1982-10-01	1983-01-01
	000010	MA2100	10	.50	1982-01-01	1982-11-01
	000110	MA2100	20	1.00	1982-01-01	1982-03-01
	000010	MA2110	10	1.00	1982-01-01	1983-02-01
	000200	MA2111	50	1.00	1982-01-01	1982-06-15
	000200	MA2111	60	1.00	1982-06-15	1983-02-01
	000220	MA2111	40	1.00	1982-01-01	1983-02-01
	000150	MA2112	60	1.00	1982-01-01	1982-07-15
	000150	MA2112	180	1.00	1982-07-15	1983-02-01
	000170	MA2112	60	1.00	1982-01-01	1983-06-01
	000170	MA2112	70	1.00	1982-06-01	1983-02-01
	000190	MA2112	70	1.00	1982-02-01	1982-10-01
	000190	MA2112	80	1.00	1982-10-01	1983-10-01
	000160	MA2113	60	1.00	1982-07-15	1983-02-01
	000170	MA2113	80	1.00	1982-01-01	1983-02-01
	000180	MA2113	70	1.00	1982-04-01	1982-06-15
	000210	MA2113	80	.50	1982-10-01	1983-02-01
	000210	MA2113	180	.50	1982-10-01	1983-02-01
	000050	OP1000	10	.25	1982-01-01	1983-02-01
	000090	OP1010	10	1.00	1982-01-01	1983-02-01
	000280	OP1010	130	1.00	1982-01-01	1983-02-01
	000290	OP1010	130	1.00	1982-01-01	1983-02-01
	000300	OP1010	130	1.00	1982-01-01	1983-02-01
	000310	OP1010	130	1.00	1982-01-01	1983-02-01
	000050	OP2010	10	.75	1982-01-01	1983-02-01
	000100	OP2010	10	1.00	1982-01-01	1983-02-01
	000320	OP2011	140	.75	1982-01-01	1983-02-01
	000320	OP2011	150	.25	1982-01-01	1983-02-01
	000330	OP2012	140	.25	1982-01-01	1983-02-01
	000330	OP2012	160	.75	1982-01-01	1983-02-01
	000340	OP2013	140	.50	1982-01-01	1983-02-01
	000340	OP2013	170	.50	1982-01-01	1983-02-01
	000020	PL2100	30	1.00	1982-01-01	1982-09-15

EMP_PHOTO 表

名前:	EMPNO	PHOTO_FORMAT	PICTURE
タイプ:	char(6) 非 NULL 値	varchar(10) 非 NULL 値	blob(100k)
説明:	従業員番号	写真のフォーマット	従業員の写真
値:	000130	bitmap	db200130.bmp
	000130	gif	db200130.gif
	000130	xwd	db200130.xwd

EMP_PHOTO 表

名前:	EMPNO	PHOTO_FORMAT	PICTURE
	000140	bitmap	db200140.bmp
	000140	gif	db200140.gif
	000140	xwd	db200140.xwd
	000150	bitmap	db200150.bmp
	000150	gif	db200150.gif
	000150	xwd	db200150.xwd
	000190	bitmap	db200190.bmp
	000190	gif	db200190.gif
	000190	xwd	db200190.xwd

EMP_RESUME 表

名前:	EMPNO	RESUME_FORMAT	RESUME
タイプ:	char(6) 非 NULL 値	varchar(10) 非 NULL 値	clob(5k)
説明:	従業員番号	履歴書のフォーマット	従業員の履歴書
値:	000130	ascii	db200130.asc
	000130	script	db200130.scr
	000140	ascii	db200140.asc
	000140	script	db200140.scr
	000150	ascii	db200150.asc
	000150	script	db200150.scr
	000190	ascii	db200190.asc
	000190	script	db200190.scr

IN_TRAY 表

名前:	RECEIVED	SOURCE	SUBJECT	NOTE_TEXT
タイプ:	timestamp	char(8)	char(64)	varchar(3000)
説明:	受取日時	メモの送信先のユーザー ID	要旨	メモ

ORG 表

名前:	DEPTNUMB	DEPTNAME	MANAGER	DIVISION	LOCATION
タイプ:	smallint 非 NULL 値	varchar(14)	smallint	varchar(10)	varchar(13)
説明:	部門番号	部門名	管理者番号	企業の部門	市
値:	10	Head Office	160	Corporate	New York
	15	New England	50	Eastern	Boston
	20	Mid Atlantic	10	Eastern	Washington
	38	South Atlantic	30	Eastern	Atlanta
	42	Great Lakes	100	Midwest	Chicago
	51	Plains	140	Midwest	Dallas

名前:	DEPTNUMB	DEPTNAME	MANAGER	DIVISION	LOCATION
	66	Pacific	270	Western	San Francisco
	84	Mountain	290	Western	Denver

PROJECT 表

名前:	PROJNO	PROJNAME	DEPTNO	RESPEMP	PRSTAFF	PRSTDATE	PRENDATE	MAJPROJ
タイプ:	char(6) 非 NULL 値	varchar(24) 非 NULL 値	char(3) 非 NULL 値	char(6) 非 NULL 値	dec(5,2)	date	date	char(6)
説明:	プロジェクト番号	プロジェクト名	担当部門	責任者	スタッフ平均人数の見積もり	開始日付の見積もり	終了日付の見積もり	主プロジェクト (副プロジェクトの場合)
値:	AD3100	ADMIN SERVICES	D01	000010	6.5	1982-01-01	1983-02-01	-
	AD3110	GENERAL ADMIN SYSTEMS	D21	000070	6	1982-01-01	1983-02-01	AD3100
	AD3111	PAYROLL PROGRAMMING	D21	000230	2	1982-01-01	1983-02-01	AD3110
	AD3112	PERSONNEL PROGRAMMING	D21	000250	1	1982-01-01	1983-02-01	AD3110
	AD3113	ACCOUNT PROGRAMMING	D21	000270	2	1982-01-01	1983-02-01	AD3110
	IF1000	QUERY SERVICES	C01	000030	2	1982-01-01	1983-02-01	-
	IF2000	USER EDUCATION	C01	000030	1	1982-01-01	1983-02-01	-
	MA2100	WELD LINE AUTOMATION	D01	000010	12	1982-01-01	1983-02-01	-
	MA2110	W L PROGRAMMING	D11	000060	9	1982-01-01	1983-02-01	MA2100
	MA2111	W L PROGRAM DESIGN	D11	000220	2	1982-01-01	1982-12-01	MA2110
	MA2112	W L ROBOT DESIGN	D11	000150	3	1982-01-01	1982-12-01	MA2110
	MA2113	W L PROD CONT PROGS	D11	000160	3	1982-02-15	1982-12-01	MA2110
	OP1000	OPERATION SUPPORT	E01	000050	6	1982-01-01	1983-02-01	-
	OP1010	OPERATION	E11	000090	5	1982-01-01	1983-02-01	OP1000
	OP2000	GEN SYSTEMS SERVICES	E01	000050	5	1982-01-01	1983-02-01	-
	OP2010	SYSTEMS SUPPORT	E21	000100	4	1982-01-01	1983-02-01	OP2000
	OP2011	SCP SYSTEMS SUPPORT	E21	000320	1	1982-01-01	1983-02-01	OP2010
	OP2012	APPLICATIONS SUPPORT	E21	000330	1	1982-01-01	1983-02-01	OP2010
	OP2013	DB/DC SUPPORT	E21	000340	1	1982-01-01	1983-02-01	OP2010
	PL2100	WELD LINE PLANNING	B01	000020	1	1982-01-01	1982-09-15	MA2100

SALES 表

名前:	SALES_DATE	SALES_PERSON	REGION	SALES
タイプ:	date	varchar(15)	varchar(15)	int
説明:	売上日	従業員の姓	売上地域	売上数量
値:	12/31/1995	LUCCHESSI	Ontario-South	1
	12/31/1995	LEE	Ontario-South	3
	12/31/1995	LEE	Quebec	1
	12/31/1995	LEE	Manitoba	2
	12/31/1995	GOUNOT	Quebec	1
	03/29/1996	LUCCHESSI	Ontario-South	3
	03/29/1996	LUCCHESSI	Quebec	1
	03/29/1996	LEE	Ontario-South	2
	03/29/1996	LEE	Ontario-North	2
	03/29/1996	LEE	Quebec	3
	03/29/1996	LEE	Manitoba	5
	03/29/1996	GOUNOT	Ontario-South	3
	03/29/1996	GOUNOT	Quebec	1
	03/29/1996	GOUNOT	Manitoba	7
	03/30/1996	LUCCHESSI	Ontario-South	1
	03/30/1996	LUCCHESSI	Quebec	2
	03/30/1996	LUCCHESSI	Manitoba	1
	03/30/1996	LEE	Ontario-South	7
	03/30/1996	LEE	Ontario-North	3
	03/30/1996	LEE	Quebec	7
	03/30/1996	LEE	Manitoba	4
	03/30/1996	GOUNOT	Ontario-South	2
	03/30/1996	GOUNOT	Quebec	18
	03/30/1996	GOUNOT	Manitoba	1
	03/31/1996	LUCCHESSI	Manitoba	1
	03/31/1996	LEE	Ontario-South	14
	03/31/1996	LEE	Ontario-North	3
	03/31/1996	LEE	Quebec	7
	03/31/1996	LEE	Manitoba	3
	03/31/1996	GOUNOT	Ontario-South	2
	03/31/1996	GOUNOT	Quebec	1
	04/01/1996	LUCCHESSI	Ontario-South	3
	04/01/1996	LUCCHESSI	Manitoba	1
	04/01/1996	LEE	Ontario-South	8
	04/01/1996	LEE	Ontario-North	-
	04/01/1996	LEE	Quebec	8
	04/01/1996	LEE	Manitoba	9
	04/01/1996	GOUNOT	Ontario-South	3
	04/01/1996	GOUNOT	Ontario-North	1
	04/01/1996	GOUNOT	Quebec	3
	04/01/1996	GOUNOT	Manitoba	7

STAFF 表

名前:	ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
タイプ:	smallint 非 NULL 値	varchar(9)	smallint	char(5)	smallint	dec(7,2)	dec(7,2)
説明:	従業員番号	従業員名	部門番号	職種	勤続年数	現在の給与	歩合給
値:	10	Sanders	20	Mgr	7	18357.50	-
	20	Pernal	20	Sales	8	18171.25	612.45
	30	Marenghi	38	Mgr	5	17506.75	-
	40	O'Brien	38	Sales	6	18006.00	846.55
	50	Hanes	15	Mgr	10	20659.80	-
	60	Quigley	38	Sales	-	16808.30	650.25
	70	Rothman	15	Sales	7	16502.83	1152.00
	80	James	20	Clerk	-	13504.60	128.20
	90	Koonitz	42	Sales	6	18001.75	1386.70
	100	Plotz	42	Mgr	7	18352.80	-
	110	Ngan	15	Clerk	5	12508.20	206.60
	120	Naughton	38	Clerk	-	12954.75	180.00
	130	Yamaguchi	42	Clerk	6	10505.90	75.60
	140	Fraye	51	Mgr	6	21150.00	-
	150	Williams	51	Sales	6	19456.50	637.65
	160	Molinare	10	Mgr	7	22959.20	-
	170	Kermisch	15	Clerk	4	12258.50	110.10
	180	Abrahams	38	Clerk	3	12009.75	236.50
	190	Sneider	20	Clerk	8	14252.75	126.50
	200	Scoutten	42	Clerk	-	11508.60	84.20
	210	Lu	10	Mgr	10	20010.00	-
	220	Smith	51	Sales	7	17654.50	992.80
	230	Lundquist	51	Clerk	3	13369.80	189.65
	240	Daniels	10	Mgr	5	19260.25	-
	250	Wheeler	51	Clerk	6	14460.00	513.30
	260	Jones	10	Mgr	12	21234.00	-
	270	Lea	66	Mgr	9	18555.50	-
	280	Wilson	66	Sales	9	18674.50	811.50
	290	Quill	84	Mgr	10	19818.00	-
	300	Davis	84	Sales	5	15454.50	806.10
	310	Graham	66	Sales	13	21000.00	200.30
	320	Gonzales	66	Sales	4	16858.20	844.00
	330	Burke	66	Clerk	1	10988.00	55.50
	340	Edwards	84	Sales	7	17844.00	1285.00
	350	Gafney	84	Clerk	5	13030.50	188.00

STAFFG 表 (2 バイト・コード・ページのみ)

名前:	ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
タイプ:	smallint 非 NULL 値	vargraphic(9)	smallint	graphic(5)	smallint	dec(9,0)	dec(9,0)
説明:	従業員番号	従業員名	部門番号	職種	勤続年数	現在の給与	歩合給
値:	10	Sanders	20	Mgr	7	18357.50	-
	20	Pernal	20	Sales	8	18171.25	612.45
	30	Marenghi	38	Mgr	5	17506.75	-
	40	O'Brien	38	Sales	6	18006.00	846.55
	50	Hanes	15	Mgr	10	20659.80	-
	60	Quigley	38	Sales	-	16808.30	650.25
	70	Rothman	15	Sales	7	16502.83	1152.00
	80	James	20	Clerk	-	13504.60	128.20
	90	Koonitz	42	Sales	6	18001.75	1386.70
	100	Plotz	42	Mgr	7	18352.80	-
	110	Ngan	15	Clerk	5	12508.20	206.60
	120	Naughton	38	Clerk	-	12954.75	180.00
	130	Yamaguchi	42	Clerk	6	10505.90	75.60
	140	Fraye	51	Mgr	6	21150.00	-
	150	Williams	51	Sales	6	19456.50	637.65
	160	Molinare	10	Mgr	7	22959.20	-
	170	Kermisch	15	Clerk	4	12258.50	110.10
	180	Abrahams	38	Clerk	3	12009.75	236.50
	190	Sneider	20	Clerk	8	14252.75	126.50
	200	Scoutten	42	Clerk	-	11508.60	84.20
	210	Lu	10	Mgr	10	20010.00	-
	220	Smith	51	Sales	7	17654.50	992.80
	230	Lundquist	51	Clerk	3	13369.80	189.65
	240	Daniels	10	Mgr	5	19260.25	-
	250	Wheeler	51	Clerk	6	14460.00	513.30
	260	Jones	10	Mgr	12	21234.00	-
	270	Lea	66	Mgr	9	18555.50	-
	280	Wilson	66	Sales	9	18674.50	811.50
	290	Quill	84	Mgr	10	19818.00	-
	300	Davis	84	Sales	5	15454.50	806.10
	310	Graham	66	Sales	13	21000.00	200.30
	320	Gonzales	66	Sales	4	16858.20	844.00
	330	Burke	66	Clerk	1	10988.00	55.50
	340	Edwards	84	Sales	7	17844.00	1285.00
	350	Gafney	84	Clerk	5	13030.50	188.00

BLOB および CLOB データ型の入ったサンプル・ファイル

ここに示すのは、 EMP_PHOTO ファイル (従業員の写真) と EMP_RESUME ファイル (従業員の履歴書) に入っているデータです。

Quintana の写真

図 14. Dolores M. Quintana

Quintana の履歴書

以下のテキストは、 db200130.asc ファイルと db200130.scr ファイルに入っています。

Resume: Dolores M. Quintana

Personal Information

Address:	1150 Eglinton Ave Mellonville, Idaho 83725
Phone:	(208) 555-9933
Birthdate:	September 15, 1925
Sex:	Female
Marital Status:	Married
Height:	5’2″
Weight:	120 lbs.

Department Information

Employee Number:	000130
Dept Number:	C01
Manager:	Sally Kwan
Position:	Analyst
Phone:	(208) 555-4578
Hire Date:	1971-07-28

Education

1965	Math and English, B.A. Adelphi University
1960	Dental Technician Florida Institute of Technology

Work History

10/91 - present	Advisory Systems Analyst Producing documentation tools for engineering department.

12/85 - 9/91	Technical Writer, Writer, text programmer, and planner.
1/79 - 11/85	COBOL Payroll Programmer Writing payroll programs for a diesel fuel company.

Interests

- Cooking
- Reading
- Sewing
- Remodeling

Nicholls の写真

図 15. Heather A. Nicholls

Nicholls の履歴書

以下のテキストは、 db200140.asc ファイルと db200140.scr ファイルに入っています。

Resume: Heather A. Nicholls

Personal Information

Address:	844 Don Mills Ave Mellonville, Idaho 83734
Phone:	(208) 555-2310
Birthdate:	January 19, 1946
Sex:	Female
Marital Status:	Single
Height:	5’8″
Weight:	130 lbs.

Department Information

Employee Number:	000140
Dept Number:	C01
Manager:	Sally Kwan

Position:	Analyst
Phone:	(208) 555-1793
Hire Date:	1976-12-15

Education

1972	Computer Engineering, Ph.D. University of Washington
1969	Music and Physics, M.A. Vassar College

Work History

2/83 - present	Architect, OCR Development Designing the architecture of OCR products.
12/76 - 1/83	Text Programmer Optical character recognition (OCR) programming in PL/I.
9/72 - 11/76	Punch Card Quality Analyst Checking punch cards met quality specifications.

Interests

- Model railroading
- Interior decorating
- Embroidery
- Knitting

Adamson の写真

図 16. Bruce Adamson

Adamson の履歴書

以下のテキストは、 db200150.asc ファイルと db200150.scr ファイルに入っています。

Resume: Bruce Adamson

Personal Information

Address:	3600 Steeles Ave Mellonville, Idaho 83757

Adamson の履歴書

Phone:	(208) 555-4489
Birthdate:	May 17, 1947
Sex:	Male
Marital Status:	Married
Height:	6’0″
Weight:	175 lbs.
Department Information	
Employee Number:	000150
Dept Number:	D11
Manager:	Irving Stern
Position:	Designer
Phone:	(208) 555-4510
Hire Date:	1972-02-12
Education	
1971	Environmental Engineering, M.Sc. Johns Hopkins University
1968	American History, B.A. Northwestern University
Work History	
8/79 - present	Neural Network Design Developing neural networks for machine intelligence products.
2/72 - 7/79	Robot Vision Development Developing rule-based systems to emulate sight.
9/71 - 1/72	Numerical Integration Specialist Helping bank systems communicate with each other.

Interests

- Racing motorcycles
- Building loudspeakers
- Assembling personal computers
- Sketching

Walker の写真

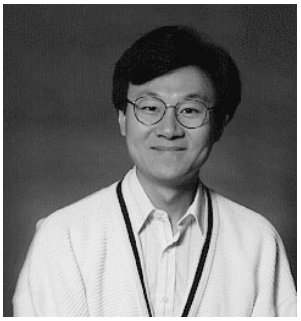


図 17. James H. Walker

Walker の履歴書

以下のテキストは、 db200190.asc ファイルと db200190.scr ファイルに入っています。

Resume: James H. Walker

Personal Information

Address:	3500 Steeles Ave Mellonville, Idaho 83757
Phone:	(208) 555-7325
Birthdate:	June 25, 1952
Sex:	Male
Marital Status:	Single
Height:	5'11″
Weight:	166 lbs.

Department Information

Employee Number:	000190
Dept Number:	D11
Manager:	Irving Stern
Position:	Designer
Phone:	(208) 555-2986
Hire Date:	1974-07-26

Education

1974	Computer Studies, B.Sc. University of Massachusetts
1972	Linguistic Anthropology, B.A. University of Toronto

Work History

6/87 - present	Microcode Design Optimizing algorithms for mathematical functions.

4/77 - 5/87 Printer Technical Support Installing and supporting laser printers.

9/74 - 3/77 Maintenance Programming Patching assembly language compiler for mainframes.

Interests

- Wine tasting
- Skiing
- Swimming
- Dancing

付録 G. 予約済みスキーマ名と予約語

データベース・マネージャーに必要な特定の名前の使用には制限があります。名前によっては、予約済みで、アプリケーション・プログラムで使用できない名前があります。また、データベース・マネージャーによって、その使用は禁止されてはいませんが、アプリケーション・プログラムによる使用をお勧めできない名前もあります。

予約済みのスキーマ名は以下のとおりです。

- SYSCAT
- SYSFUN
- SYSIBM
- SYSSTAT
- SYSPROC

'SYS' は規則によりシステムで予約されている領域を示すのに使用されるので、'SYS' の接頭部で始まるスキーマ名は使用しないようにしてください。ユーザー定義関数、ユーザー定義タイプ、トリガー、または別名を、 'SYS' で始まる名前のスキーマに入れることはできません (SQLSTATE 42939)。

DB2QP スキーマおよび SYSTOOLS スキーマは、 DB2 ツールが使用するために確保されています。データベース・マネージャーによってこれらのスキーマの使用は禁止されてはいませんが、ユーザーがそれらを明示的に定義することはお勧めできません。

さらに、SESSION はスキーマ名としては使用しないようお勧めします。宣言済み一時表は SESSION で修飾しなければならないため、アプリケーションが持続表の名前と同じ名前を指定して一時表を宣言してしまい、アプリケーションのロジックが複雑になってしまう場合があります。このような事態を避けるため、宣言済み一時表を扱う場合を除いて、スキーマ SESSION を使用することは避けてください。

DB2 バージョン 8 には明確に予約された語はありません。キーワードも、それが SQL キーワードとして解釈されることになるコンテキスト以外であれば、通常 ID として使用することができます。キーワードとして解釈されるコンテキストの場合は、その語を区切り ID として指定する必要があります。たとえば、SELECT ステートメントに、区切り ID でない COUNT を列名として使用することはできません。

ISO/ANSI SQL99 および他の DB2 Universal Database 製品には、 DB2 Universal Database が予約していない予約語が装備されていますが、移植性が低下するので、通常 ID として使用しないようお勧めします。

DB2 Universal Database 製品間での移植性に関しては、以下を予約語と見なしてください。

```
ADD             DETERMINISTIC  LEAVE         RESTART
AFTER           DISALLOW       LEFT          RESTRICT
ALIAS           DISCONNECT     LIKE          RESULT
```

ALL	DISTINCT	LINKTYPE	RESULT_SET_LOCATOR
ALLOCATE	DO	LOCAL	RETURN
ALLOW	DOUBLE	LOCALE	RETURNS
ALTER	DROP	LOCATOR	REVOKE
AND	DSNHATTR	LOCATORS	RIGHT
ANY	DSSIZE	LOCK	ROLLBACK
APPLICATION	DYNAMIC	LOCKMAX	ROUTINE
AS	EACH	LOCKSIZE	ROW
ASSOCIATE	EDITPROC	LONG	ROWS
ASUTIME	ELSE	LOOP	RRN
AUDIT	ELSEIF	MAXVALUE	RUN
AUTHORIZATION	ENCODING	MICROSECOND	SAVEPOINT
AUX	END	MICROSECONDS	SCHEMA
AUXILIARY	END-EXEC	MINUTE	SCRATCHPAD
BEFORE	END-EXEC1	MINUTES	SECOND
BEGIN	ERASE	MINVALUE	SECONDS
BETWEEN	ESCAPE	MODE	SECQTY
BINARY	EXCEPT	MODIFIES	SECURITY
BUFFERPOOL	EXCEPTION	MONTH	SELECT
BY	EXCLUDING	MONTHS	SENSITIVE
CACHE	EXECUTE	NEW	SET
CALL	EXISTS	NEW_TABLE	SIGNAL
CALLED	EXIT	NO	SIMPLE
CAPTURE	EXTERNAL	NOCACHE	SOME
CARDINALITY	FENCED	NOCYCLE	SOURCE
CASCADED	FETCH	NODENAME	SPECIFIC
CASE	FIELDPROC	NODENUMBER	SQL
CAST	FILE	NOMAXVALUE	SQLID
CCSID	FINAL	NOMINVALUE	STANDARD
CHAR	FOR	NOORDER	START
CHARACTER	FOREIGN	NOT	STATIC
CHECK	FREE	NULL	STAY
CLOSE	FROM	NULLS	STOGROUP
CLUSTER	FULL	NUMPARTS	STORES
COLLECTION	FUNCTION	OBID	STYLE
COLLID	GENERAL	OF	SUBPAGES
COLUMN	GENERATED	OLD	SUBSTRING
COMMENT	GET	OLD_TABLE	SYNONYM
COMMIT	GLOBAL	ON	SYSFUN
CONCAT	GO	OPEN	SYSIBM
CONDITION	GOTO	OPTIMIZATION	SYSPROC
CONNECT	GRANT	OPTIMIZE	SYSTEM
CONNECTION	GRAPHIC	OPTION	TABLE
CONSTRAINT	GROUP	OR	TABLESPACE
CONTAINS	HANDLER	ORDER	THEN
CONTINUE	HAVING	OUT	TO
COUNT	HOLD	OUTER	TRANSACTION
COUNT_BIG	HOUR	OVERRIDING	TRIGGER
CREATE	HOURS	PACKAGE	TRIM
CROSS	IDENTITY	PARAMETER	TYPE
CURRENT	IF	PART	UNDO
CURRENT_DATE	IMMEDIATE	PARTITION	UNION
CURRENT_LC_CTYPE	IN	PATH	UNIQUE
CURRENT_PATH	INCLUDING	PIECESIZE	UNTIL
CURRENT_SERVER	INCREMENT	PLAN	UPDATE
CURRENT_TIME	INDEX	POSITION	USAGE
CURRENT_TIMESTAMP	INDICATOR	PRECISION	USER
CURRENT_TIMEZONE	INHERIT	PREPARE	USING
CURRENT_USER	INNER	PRIMARY	VALIDPROC
CURSOR	INOUT	PRIQTY	VALUES
CYCLE	INSENSITIVE	PRIVILEGES	VARIABLE
DATA	INSERT	PROCEDURE	VARIANT
DATABASE	INTEGRITY	PROGRAM	VCAT
DAY	INTO	PSID	VIEW
DAYS	IS	QUERYNO	VOLUMES
DB2GENERAL	ISOBID	READ	WHEN
DB2GENRL	ISOLATION	READS	WHERE

DB2SQL	ITERATE	RECOVERY	WHILE
DBINFO	JAR	REFERENCES	WITH
DECLARE	JAVA	REFERENCING	WLM
DEFAULT	JOIN	RELEASE	WRITE
DEFAULTS	KEY	RENAME	YEAR
DEFINITION	LABEL	REPEAT	YEARS
DELETE	LANGUAGE	RESET	
DESCRIPTOR	LC_CTYPE	RESIGNAL	

| 上のリストに載っていない、ISO/ANSI SQL99 予約語のリストです。

ABSOLUTE	DESCRIBE	MODULE	SESSION
ACTION	DESTROY	NAMES	SESSION_USER
ADMIN	DESTRUCTOR	NATIONAL	SETS
AGGREGATE	DIAGNOSTICS	NATURAL	SIZE
ARE	DICTIONARY	NCHAR	SMALLINT
ARRAY	DOMAIN	NCLOB	SPACE
ASC	EQUALS	NEXT	SPECIFICTYPE
ASSERTION	EVERY	NONE	SQLEXCEPTION
AT	EXEC	NUMERIC	SQLSTATE
BIT	FALSE	OBJECT	SQLWARNING
BLOB	FIRST	OFF	STATE
BOOLEAN	FLOAT	ONLY	STATEMENT
BOTH	FOUND	OPERATION	STRUCTURE
BREADTH	GROUPING	ORDINALITY	SYSTEM_USER
CASCADE	HOST	OUTPUT	TEMPORARY
CATALOG	IGNORE	PAD	TERMINATE
CLASS	INITIALIZE	PARAMETERS	THAN
CLOB	INITIALLY	PARTIAL	TIME
COLLATE	INPUT	POSTFIX	TIMESTAMP
COLLATION	INT	PREFIX	TIMEZONE_HOUR
COMPLETION	INTEGER	PREORDER	TIMEZONE_MINUTE
CONSTRAINTS	INTERSECT	PRESERVE	TRAILING
CONSTRUCTOR	INTERVAL	PRIOR	TRANSLATION
CORRESPONDING	LARGE	PUBLIC	TREAT
CUBE	LAST	REAL	TRUE
CURRENT_ROLE	LATERAL	RECURSIVE	UNDER
DATE	LEADING	REF	UNKNOWN
DEALLOCATE	LESS	RELATIVE	UNNEST
DEC	LEVEL	ROLE	VALUE
DECIMAL	LIMIT	ROLLUP	VARCHAR
DEFERRABLE	LOCALTIME	SCOPE	VARYING
DEFERRED	LOCALTIMESTAMP	SCROLL	WHENEVER
DEPTH	MAP	SEARCH	WITHOUT
DEREF	MATCH	SECTION	WORK
DESC	MODIFY	SEQUENCE	ZONE

付録 H. トリガーと制約の相互作用

この付録では、更新操作の結果として生じる可能性のある参照制約およびチェック制約とトリガーとの相互作用について説明します。 図 18 とその後の説明は、データベースのデータを更新する SQL ステートメントに対して行われる典型的な処理を示しています。

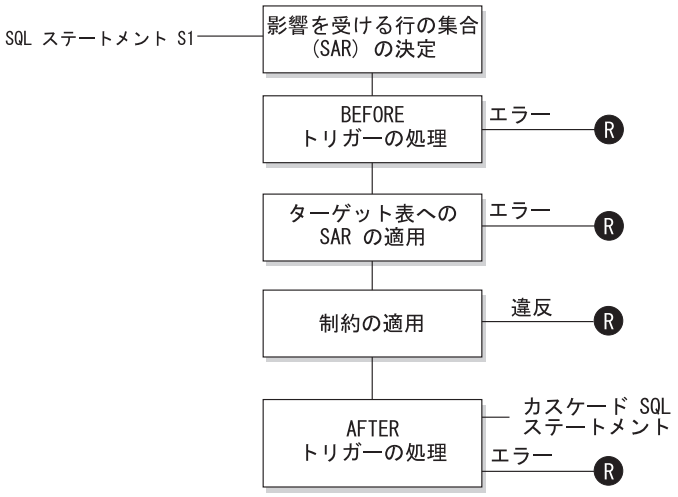


図 18. 関連するトリガーと制約を伴う SQL ステートメントの処理

図 18 は、表を更新する SQL ステートメントの一般的な処理の順序を示しています。ここでは、表に、「BEFORE」のトリガー、参照制約、チェック制約、およびカスケードする「AFTER」トリガーとが組み込まれることを想定しています。 図 18 に示されているボックスやその他の項目について、以下に説明します。

- SQL ステートメント S_1

  これは、その処理を開始する DELETE、INSERT、または UPDATE ステートメントです。 SQL ステートメント S_1 は、この説明においてターゲット表 と呼ばれる表 (または表に対する更新可能なビュー) を指定しています。

- 影響を受ける行の集合 (*SAR*) の決定

  このステップは、CASCADE および SET NULL の参照制約の削除規則と、AFTER トリガーからカスケードした SQL ステートメントについて繰り返される処理の開始点です。

  このステップの目的は、その SQL ステートメントで*影響を受ける行の集合* を決定することです。 *SAR* に組み込まれる一連の行は、以下の基準をベースにします。

  – DELETE の場合、ステートメントの検索条件を満たしているすべての行 (位置指定 DELETE の場合は現在行)

  – INSERT の場合、VALUES 文節または全選択によって指定される行

– UPDATE の場合、検索条件を満たしているすべての行 (位置指定 UPDATE の場合は現在行)

SAR が空の場合、 BEFORE トリガー、ターゲット表に適用される変更、または SQL ステートメントの処理に対する制約はありません。

- BEFORE トリガーの処理

  BEFORE トリガーの処理はすべて作成の昇順で行われます。各 BEFORE トリガーは、 *SAR* 内の各行ごとに 1 回ずつトリガー・アクションを処理します。

  トリガー・アクションの処理の過程でエラーが生じることがあり、そのような場合には元の SQL ステートメント S_1 の結果としての変更内容 (これまでの) がすべてロールバックされます。

  BEFORE トリガーがない場合、または *SAR* が空の場合、このステップはスキップされます。

- ターゲット表への *SAR* の適用

  実際の削除、挿入、または更新は、 *SAR* を使ってデータベース内のターゲット表に適用されます。

  SAR の適用時にエラーが生じることがあり (ユニーク索引のあるロケーションに重複するキーを持つ行を挿入しようとした場合など)、そのような場合は元の SQL ステートメント S_1 の結果としての変更内容 (これまでの) がすべてロールバックされます。

- 制約の適用

  SAR が空でない場合には、ターゲット表に関連した制約が適用されます。この制約には、ユニーク制約、ユニーク索引、参照制約、チェック制約、ビューに対する WITH CHECK OPTION に関連した検査などがあります。カスケードまたは NULL 設定の削除規則の参照制約があると、さらに別のトリガーが起動される場合があります。

  索引または WITH CHECK OPTION に違反があると、エラーが発生し、 S_1 の結果としての変更内容 (これまでの) がすべてロールバックされます。

- AFTER トリガーの処理

  S_1 によって活動化された AFTER トリガーは、作成の昇順に処理されます。

  FOR EACH STATEMENT トリガーでは、*SAR* が空の場合にも、 1 回だけトリガー・アクションが処理されます。 FOR EACH ROW トリガーでは、 *SAR* 内の各行ごとに 1 回ずつトリガー・アクションが処理されます。

  トリガー・アクションの処理の過程でエラーが生じることがあり、そのような場合は元の S_1 の結果としての変更内容 (これまでの) がすべてロールバックされます。

  トリガーのトリガー・アクションには、トリガーによって実行される DELETE、INSERT、または UPDATE などの SQL ステートメントが入っている場合があります。この説明では、そのようなステートメントは、*カスケードした SQL ステートメント* と見なされます。

カスケードした SQL ステートメントとは、 AFTER トリガーのトリガー・アクションの一部として処理される DELETE、 INSERT、または UPDATE ステートメントのことです。そのステートメントによって、カスケード・レベルのトリガー処理が開始されます。これは、新しい S_1 としてトリガー SQL ステートメントを割り当てて、ここで説明した手順をすべて再帰的に実行することと見なすことができます。

各 S_1 ごとに起動されるすべての AFTER トリガーによって実行されるすべての SQL ステートメントの処理が完了すると、元の S_1 の処理が完了します。

- **R** = 変更を S_1 の前までロールバックする操作

制約違反も含めて、処理中にエラーが発生すると、元の SQL ステートメント S_1 の結果として直接または間接になされたすべての変更がロールバックされます。その場合、データベースは、元の SQL ステートメント S_1 の実行直前と同じ状態に戻ります。

付録 I. Explain 表

Explain 表

Explain 表は、Explain 機能が活動化された時点のアクセス・プランをキャプチャーします。 Explain 表は、Explain 機能を呼び出す前に作成しておく必要があります。文書化された表定義を使用してこの表を作成することができますが、 sqllib ディレクトリーの misc サブディレクトリーの中の EXPLAIN.DDL ファイルに用意されているコマンド行プロセッサー (CLP) 入力スクリプトの例を呼び出して作成することもできます。スクリプトを呼び出すには、 Explain 表が要求されているデータベースに接続してから、次のコマンドを出します。

```
db2 -tf EXPLAIN.DDL
```

Explain 機能によって Explain 表にデータを追加しても、トリガーが起動したり、参照制約またはチェック制約が起動したりすることはありません。たとえば、EXPLAIN_INSTANCE 表に対する挿入トリガーを定義して、該当するステートメントが Explain されても、そのトリガーは起動しません。

パーティション・データベース・システムにおいて Explain 機能のパフォーマンスを改善するには、単一のデータベース・パーティション・グループ内に Explain 表を、できれば照会のコンパイル時に接続する予定のものと同じデータベース・パーティション上で作成することをお勧めします。

関連資料:

- 752 ページの『EXPLAIN_ARGUMENT 表』
- 758 ページの『EXPLAIN_OBJECT 表』
- 761 ページの『EXPLAIN_OPERATOR 表』
- 763 ページの『EXPLAIN_PREDICATE 表』
- 768 ページの『EXPLAIN_STREAM 表』
- 770 ページの『ADVISE_INDEX 表』
- 778 ページの『ADVISE_WORKLOAD 表』
- 756 ページの『EXPLAIN_INSTANCE 表』
- 765 ページの『EXPLAIN_STATEMENT 表』
- 773 ページの『ADVISE_INSTANCE 表』
- 774 ページの『ADVISE_MQT 表』
- 776 ページの『ADVISE_PARTITION 表』
- 777 ページの『ADVISE_TABLE 表』

EXPLAIN_ARGUMENT 表

EXPLAIN_ARGUMENT 表は、個々の演算子にユニークな特性がある場合、それを示します。

表 156. EXPLAIN_ARGUMENT 表： PK は、列が主キーの一部であることを示します。FK は、列が外部キーの一部であることを示します。

列名	データ型	NULL 可能?	キー?	説明
EXPLAIN_REQUESTER	VARCHAR(128)	No	FK	この Explain 要求を開始した許可 ID。
EXPLAIN_TIME	TIMESTAMP	No	FK	Explain 要求の開始時刻。
SOURCE_NAME	VARCHAR(128)	No	FK	動的ステートメントに Explain 要求を出したときに実行していたパッケージの名前、または静的 SQL に Explain 要求を出したときのソース・ファイルの名前。
SOURCE_SCHEMA	VARCHAR(128)	No	FK	Explain 要求のソースのスキーマ、または修飾子。
SOURCE_VERSION	VARCHAR(64)	No	FK	Explain 要求のソースのバージョン。
EXPLAIN_LEVEL	CHAR(1)	No	FK	この行に関連する Explain 情報のレベル。
STMTNO	INTEGER	No	FK	この Explain 情報に関連するパッケージ内のステートメントの番号。
SECTNO	INTEGER	No	FK	この Explain 情報に関連したパッケージ内のセクションの番号。
OPERATOR_ID	INTEGER	No	No	この照会内の演算子のユニークな ID。
ARGUMENT_TYPE	CHAR(8)	No	No	この演算子の引き数のタイプ。
ARGUMENT_VALUE	VARCHAR(1024)	Yes	No	この演算子の引き数の値。値が LONG_ARGUMENT_VALUE にある場合は NULL。
LONG_ARGUMENT_VALUE	CLOB(2M)	Yes	No	この演算子の引き数の値。(テキストが ARGUMENT_VALUE に収まらない場合。) 値が ARGUMENT_VALUE にある場合は NULL。

表 157. ARGUMENT_TYPE および ARGUMENT_VALUE 列の値

ARGUMENT_TYPE の値	可能な ARGUMENT_VALUE 値	説明
AGGMODE	COMPLETE PARTIAL INTERMEDIATE FINAL	部分集約標識。
BITFLTR	INTEGERFALSE	ハッシュ結合で使われるビット・フィルターのサイズ。
BLD_LEVEL	DB2 ビルド ID	ソース・コード・バージョンの内部識別ストリング。
BLKLOCK	EXCLUSIVE INTENT EXCLUSIVE INTENT SHARE NONE SHARE UPDATE	ブロック・レベル・ロック意図。
CSERQY	TRUE FALSE	リモート照会が共通副次式。
CSETEMP	TRUE FALSE	共通副次式フラグに対する一時表。
DIRECT	TRUE	取り出し標識を指示する。
DSTSEVER	サーバー名	宛先 (配送元) サーバー。

表 157. ARGUMENT_TYPE および ARGUMENT_VALUE 列の値 (続き)

ARGUMENT_TYPE の値	可能な ARGUMENT_VALUE 値	説明
DUPLWARN	TRUE FALSE	警告標識を複写する。
EARLYOUT	LEFT RIGHT NONE	EARLYOUT 標識。
ENVVAR	このタイプの各行には以下のものが入ります。 • 環境変数名 • 環境変数値	オプティマイザーに影響する環境変数
FETCHMAX	IGNORE INTEGER	FETCH 演算子の MAXPAGES 引き数の値をオーバーライドする。
GREEDY	TRUE	オプティマイザーが貪欲型アルゴリズムを使用してアクセスをプランしたことを示す。
GROUPBYC	TRUE FALSE	Group By 列が与えられているかどうか。
GROUPBYN	Integer	比較列の数。
GROUPBYR	このタイプの各行には以下のものが入ります。 • group by 文節内の列の順序値 (後に、コロンとスペースが続く) • 列の名前。	Group By 要件。
HASHCODE	24 32	ハッシュ結合に使われるハッシュ・コードのサイズ (ビット単位)。
INNERCOL	このタイプの各行には以下のものが入ります。 • 配列内の列の順序値 (後に、コロンとスペースが続く) • 列の名前。 • 昇順逆順指定値 **(A)** 昇順 **(D)** 降順	内部配列の列。
ISCANMAX	IGNORE INTEGER	ISCAN 演算子の MAXPAGES 引き数の値をオーバーライドする。
JN_INPUT	INNER OUTER	演算子が内部または外部のどちらの結合を送る演算子であるかを示す。
LISTENER	TRUE FALSE	Listener 表キューの標識。
MAXPAGES	ALL NONE INTEGER	プリフェッチのための最大ページ数。
MAXRIDS	NONE INTEGER	個々のリスト・プリフェッチ要求に組み込まれる最大行 ID。
NUMROWS	INTEGER	ソートされるべき行の数。
ONEFETCH	TRUE FALSE	1 つの取り出し標識。

表 157. ARGUMENT_TYPE および ARGUMENT_VALUE 列の値 (続き)

ARGUMENT_TYPE の値	可能な ARGUMENT_VALUE 値	説明
OUTERCOL	このタイプの各行には以下のものが入ります。 • 配列内の列の順序値 (後に、コロンとスペースが続く) • 列の名前。 • 昇順逆順指定値 **(A)** 昇順 **(D)** 降順	外部配列の列。
OUTERJN	LEFT RIGHT FULLLEFT (ANTI) RIGHT (ANTI)	外部結合の標識。
PARTCOLS	列の名前	演算子のパーティション列。
PREFETCH	LIST NONE SEQUENTIAL	プリフェッチ有資格属性のタイプ。
REOPT	ALWAYSONCE	パラメーター・マーカー、ホスト変数、および特殊レジスターに bind-in 値を使ってステートメントを最適化します。
RMTQTEXT	照会テキスト	リモート照会テキスト。
RNG_PROD	関数名	拡張索引アクセスのための範囲生成関数。
ROWLOCK	EXCLUSIVE NONE REUSE SHARE SHORT (INSTANT) SHARE UPDATE	行ロック意図。
ROWWIDTH	INTEGER	ソートされる行の幅。
RSUFFIX	照会テキスト	リモート SQL の接尾部。
SCANDIR	FORWARD REVERSE	スキャンの方向。
SCANGRAN	INTEGER	パーティション内並列処理、パーティション内並列処理のスキャンの細分性。 SCANUNIT の単位で表される。
SCANTYPE	LOCAL PARALLEL	パーティション内並列処理、索引または表のスキャン。
SCANUNIT	ROW PAGE	パーティション内並列処理、スキャンの細分性の単位。
SHARED	TRUE	パーティション内並列処理、共用 TEMP 標識。
SLOWMAT	TRUE FALSE	低速マテリアライズ・フラグ。
SNGLPROD	TRUE FALSE	パーティション内並列処理、単一エージェントによるソートまたは一時作成。

表 157. ARGUMENT_TYPE および ARGUMENT_VALUE 列の値 (続き)

ARGUMENT_TYPE の値	可能な ARGUMENT_VALUE 値	説明
SORTKEY	このタイプの各行には以下のものが入ります。 • キー内の列の順序値 (後に、コロンとスペースが続く) • 列の名前。 • 昇順逆順指定値 **(A)** 昇順 **(D)** 降順	ソート・キーの列。
SORTTYPE	PARTITIONED SHARED ROUND ROBIN REPLICATED	パーティション内並列処理、ソート・タイプ。
SRCSEVER	サーバー名	ソース (配送先) サーバー。
SPILLED	INTEGER	SORT スピル内の推定ページ数。
SQLCA	警告情報	Explain 操作中に発行された警告と理由コード。
STMTHEAP	INTEGER	ステートメントのコンパイルの開始時のステートメント・ヒープのサイズ。
STREAM	TRUE FALSE	リモート・ソースがストリーミング。
TABLOCK	EXCLUSIVE INTENT EXCLUSIVE INTENT NONE INTENT SHARE REUSE SHARE SHARE INTENT EXCLUSIVE SUPER EXCLUSIVE UPDATE	表ロック意図。
TEMPSIZE	INTEGER	一時表のページ・サイズ。
TQDEGREE	INTEGER	パーティション内並列処理、表キューにアクセスするサブエージェントの数。
TQMERGE	TRUE FALSE	マージする (ソート済み) 表キューの標識。
TQREAD	READ AHEAD STEPPING SUBQUERY STEPPING	表キューの読み取り特性。
TQSEND	BROADCAST DIRECTED SCATTER SUBQUERY DIRECTED	表キューの送信特性。
TQTYPE	LOCAL	パーティション内並列処理、表キュー。
TRUNCSRT	TRUE	切り捨てソート (作成される行の数を制限する)。
UNIQUE	TRUE FALSE	固有性の標識。
UNIQKEY	このタイプの各行には以下のものが入ります。 • キー内の列の順序値 (後に、コロンとスペースが続く) • 列の名前	ユニーク・キーの列。
VOLATILE	TRUE	揮発性表。

EXPLAIN_INSTANCE 表

EXPLAIN_INSTANCE 表は、すべての Explain 情報用の主コントロール表です。Explain 表中のデータの各行は、この表内のあるユニークな 1 行に明示的にリンクされます。 EXPLAIN_INSTANCE 表は、Explain 対象の SQL ステートメントのソースに関する基本情報、および Explain 機能の環境に関する情報を提供します。

表 158. EXPLAIN_INSTANCE 表： PK は、その列が主キーの一部であることを示します。 FK は、列が外部キーの一部であることを示します。

列名	データ型	NULL 可能?	キー?	説明
EXPLAIN_REQUESTER	VARCHAR(128)	No	PK	この Explain 要求を開始した許可 ID。
EXPLAIN_TIME	TIMESTAMP	No	PK	Explain 要求の開始時刻。
SOURCE_NAME	VARCHAR(128)	No	PK	動的ステートメントに Explain 要求を出したときに実行していたパッケージの名前、または静的 SQL に Explain 要求を出したときのソース・ファイルの名前。
SOURCE_SCHEMA	VARCHAR(128)	No	PK	Explain 要求のソースのスキーマ、または修飾子。
SOURCE_VERSION	VARCHAR(64)	No	PK	Explain 要求のソースのバージョン。
EXPLAIN_OPTION	CHAR(1)	No	No	この要求に関して要求された Explain 情報の内容を示します。 使用できる値は以下のとおりです。 **P** PLAN SELECTION
SNAPSHOT_TAKEN	CHAR(1)	No	No	この要求に関して Explain スナップショットが取られたかどうかを示します。 使用できる値は以下のとおりです。 **Y** Yes。1 つまたは複数の Explain スナップショットが取られ、 EXPLAIN_STATEMENT 表に保管されました。正規の Explain 情報もキャプチャーされました。 **N** Explain スナップショットは取られませんでした。正規の Explain 情報がキャプチャーされました。 **O** Explain スナップショットだけが取られました。正規の Explain 情報はキャプチャーされませんでした。
DB2_VERSION	CHAR(7)	No	No	この Explain 要求を処理した DB2 Universal Database の製品リリース番号。フォーマットは vv.rr.m です。ただし、 **vv** バージョン番号 **rr** リリース番号 **m** 保守リリース番号
SQL_TYPE	CHAR(1)	No	No	Explain インスタンスが静的と動的のどちらの SQL に関するものであったかどうかを示します。 使用できる値は以下のとおりです。 **S** 静的 SQL **D** 動的 SQL
QUERYOPT	INTEGER	No	No	Explain 呼び出しの時点で SQL コンパイラーが使用する照会最適化クラスを示します。値は、Explain 中の SQL ステートメントについて SQL コンパイラーが実行したのが、どのレベルの照会最適化であるかを示します。

表 158. EXPLAIN_INSTANCE 表 (続き): PK は、その列が主キーの一部であることを示します。 FK は、列が外部キーの一部であることを示します。

列名	データ型	NULL 可能?	キー?	説明
BLOCK	CHAR(1)	No	No	SQL ステートメントのコンパイル時に使用されたカーソルのブロッキング・タイプを示します。詳細については、SYSCAT.PACKAGES の BLOCK 列を参照してください。 使用できる値は以下のとおりです。 **N** ブロッキングなし **U** 確定カーソルのブロック **B** すべてのカーソルのブロック
ISOLATION	CHAR(2)	No	No	SQL ステートメントのコンパイル時に使用された分離レベルの種類を示します。詳細については、SYSCAT.PACKAGES の ISOLATION 列を参照してください。 使用できる値は以下のとおりです。 **RR** 反復可能読み取り **RS** 読み取り固定 **CS** カーソル固定 **UR** 非コミット読み取り
BUFFPAGE	INTEGER	No	No	Explain の呼び出しの時点で設定された BUFFPAGE データベース構成の値が入っています。
AVG_APPLS	INTEGER	No	No	Explain 呼び出し時に、 AVG_APPLS データベース構成パラメーターの値が入れられます。
SORTHEAP	INTEGER	No	No	Explain の呼び出しの時点で設定された SORTHEAP データベース構成の値が入っています。
LOCKLIST	INTEGER	No	No	Explain の呼び出しの時点で設定された LOCKLIST データベース構成の値が入っています。
MAXLOCKS	SMALLINT	No	No	Explain の呼び出しの時点で設定された MAXLOCKS データベース構成の値が入っています。
LOCKS_AVAIL	INTEGER	No	No	オプティマイザーによって各ユーザーごとに使用可能と見なされるロックの数が入っています。 (LOCKLIST および MAXLOCKS から派生したもの。)
CPU_SPEED	DOUBLE	No	No	Explain の呼び出しの時点で設定された CPUSPEED データベース・マネージャー構成設定の値が入っています。
REMARKS	VARCHAR(254)	Yes	No	ユーザーが入力したコメント。
DBHEAP	INTEGER	No	No	Explain 呼び出し時に DBHEAP データベース構成設定値が入れられます。
COMM_SPEED	DOUBLE	No	No	Explain 呼び出し時に COMM_BANDWIDTH データベース構成設定値が入れられます。
PARALLELISM	CHAR(2)	No	No	使用できる値は以下のとおりです。 • N = 並列処理なし • P = パーティション内並列処理 • IP = パーティション間並列処理 • BP = パーティション内並列処理とパーティション間並列処理
DATAJOINER	CHAR(1)	No	No	使用できる値は以下のとおりです。 • N = 非フェデレーテッド・システム・プラン • Y = フェデレーテッド・システム・プラン

EXPLAIN_OBJECT 表

EXPLAIN_OBJECT 表は、 SQL ステートメントを満たすために生成されるアクセス・プランが必要とするデータ・オブジェクトを指定します。

表 159. EXPLAIN_OBJECT 表： PK は、列が主キーの一部であることを示します。FK は、列が外部キーの一部であることを示します。

列名	データ・タイプ	NULL 可能?	キー?	説明
EXPLAIN_REQUESTER	VARCHAR(128)	No	FK	この Explain 要求を開始した許可 ID。
EXPLAIN_TIME	TIMESTAMP	No	FK	Explain 要求の開始時刻。
SOURCE_NAME	VARCHAR(128)	No	FK	動的ステートメントに Explain 要求を出したときに実行していたパッケージの名前、または静的 SQL に Explain 要求を出したときのソース・ファイルの名前。
SOURCE_SCHEMA	VARCHAR(128)	No	FK	Explain 要求のソースのスキーマ、または修飾子。
SOURCE_VERSION	VARCHAR(64)	No	FK	Explain 要求のソースのバージョン。
EXPLAIN_LEVEL	CHAR(1)	No	FK	この行に関連する Explain 情報のレベル。
STMTNO	INTEGER	No	FK	パッケージ内のステートメントのうち、この Explain 情報に関連するもののステートメント番号。
SECTNO	INTEGER	No	FK	この Explain 情報に関連したパッケージのセクション番号。
OBJECT_SCHEMA	VARCHAR(128)	No	No	このオブジェクトが属しているスキーマ。
OBJECT_NAME	VARCHAR(128)	No	No	オブジェクトの名前。
OBJECT_TYPE	CHAR(2)	No	No	オブジェクトのタイプの記述ラベル。
CREATE_TIME	TIMESTAMP	Yes	No	オブジェクトの作成時刻。表関数の場合は NULL。
STATISTICS_TIME	TIMESTAMP	Yes	No	このオブジェクトを最後に更新した時刻。このオブジェクトに統計がない場合は NULL 値。
COLUMN_COUNT	SMALLINT	No	No	このオブジェクトの列数。
ROW_COUNT	INTEGER	No	No	このオブジェクトの行数の見積もり。
WIDTH	INTEGER	No	No	オブジェクトの平均幅 (バイト数)。索引の場合は -1 に設定します。
PAGES	INTEGER	No	No	オブジェクトがバッファー・プールで占有するページ数の見積もり。表関数には、-1 に設定します。
DISTINCT	CHAR(1)	No	No	オブジェクト内の行が固有かどうか (つまり、重複しているかどうか) を示します。 使用できる値は以下のとおりです。 **Y** Yes **N** No
TABLESPACE_NAME	VARCHAR(128)	Yes	No	このオブジェクトが保管されている表スペースの名前。表スペースが関係していない場合は、NULL 値に設定する。
OVERHEAD	DOUBLE	No	No	指定された表スペースにランダム入出力を 1 回行うためのオーバーヘッドの見積合計 (ミリ秒)。コントローラー・オーバーヘッド、ディスク・シーク、および待ち時間が組み入れられます。表スペースが関係していない時は -1 に設定します。
TRANSFER_RATE	DOUBLE	No	No	指定の表スペースからデータ・ページを読み取るための時間の見積もり (ミリ秒単位)。表スペースが関係していない時は -1 に設定します。

表 159. EXPLAIN_OBJECT 表 (続き): PK は、列が主キーの一部であることを示します。FK は、列が外部キーの一部であることを示します。

列名	データ・タイプ	NULL 可能?	キー?	説明
PREFETCHSIZE	INTEGER	No	No	プリフェッチの実行時に読み取るデータ・ページの数。表関数には、-1 に設定します。
EXTENTSIZE	INTEGER	No	No	データ・ページを単位とするエクステント・サイズ。表スペースの中のコンテナーにこの数のページが書き込まれたら、次のコンテナーに切り替わります。表関数には、-1 に設定します。
CLUSTER	DOUBLE	No	No	索引とのデータ・クラスタリングの程度。 >= 1 の場合は CLUSTERRATIO。 0 以上かつ 1 より小さい場合、これは CLUSTERFACTOR。表、表関数について、またはこの統計が使用不可の場合は、-1 に設定します。
NLEAF	INTEGER	No	No	この索引オブジェクトの値が占めるリーフ・ページの数。表、表関数について、またはこの統計が使用不可の場合は、-1 に設定します。
NLEVELS	INTEGER	No	No	この索引オブジェクトのツリー内の索引レベルの数。表、表関数について、またはこの統計が使用不可の場合は、-1 に設定します。
FULLKEYCARD	BIGINT	No	No	この索引オブジェクトに組み込まれる個別フル・キー値の数。表、表関数について、またはこの統計が使用不可の場合は、-1 に設定します。
OVERFLOW	INTEGER	No	No	表内のオーバーフロー・レコードの合計数。索引、表関数について、またはこの統計が使用不可である場合は、-1 に設定します。
FIRSTKEYCARD	BIGINT	No	No	最初のキーの値の種類数。表、表関数について、またはこの統計が使用不可の場合は、-1 に設定します。
FIRST2KEYCARD	BIGINT	No	No	索引の最初の {2,3,4} 列を使用する最初のキー値の種類数。表、表関数について、またはこの統計が使用不可の場合は、-1 に設定します。
FIRST3KEYCARD	BIGINT	No	No	
FIRST4KEYCARD	BIGINT	No	No	
SEQUENTIAL_PAGES	INTEGER	No	No	索引キーの順序でディスクに存在し、それらの間に大きなギャップがないか、わずかなギャップしかないリーフ・ページの数。表、表関数について、またはこの統計が使用不可の場合は、-1 に設定します。
DENSITY	INTEGER	No	No	索引によって占有されているページの範囲内の、ページ数に対する SEQUENTIAL_PAGES の比率。パーセントで表現される (0 から 100 の整数)。表、表関数について、またはこの統計が使用不可の場合は、-1 に設定します。
STATS_SRC	CHAR(1)	No	No	統計のソースを示します。単一のノードからの場合は 1 に設定します。
AVERAGE_SEQUENCE_GAP	DOUBLE	No	No	シーケンス間のギャップ。
AVERAGE_SEQUENCE_FETCH_GAP	DOUBLE	No	No	索引を使用してフェッチするときの、シーケンス間のギャップ。
AVERAGE_SEQUENCE_PAGES	DOUBLE	No	No	順次にアクセスできる索引ページの平均数。
AVERAGE_SEQUENCE_FETCH_PAGES	DOUBLE	No	No	索引を使用してフェッチする際の、順次にアクセスできる表ページの平均数。
AVERAGE_RANDOM_PAGES	DOUBLE	No	No	順次ページ・アクセスの間のランダム索引ページの平均数。
AVERAGE_RANDOM_FETCH_PAGES	DOUBLE	No	No	索引を使用してフェッチする際の、順次ページ・アクセスの間のランダム表ページの平均数。

表 159. EXPLAIN_OBJECT 表 (続き): PK は、列が主キーの一部であることを示します。FK は、列が外部キーの一部であることを示します。

列名	データ・タイプ	NULL 可能?	キー?	説明
NUMRIDS	BIGINT	No	No	索引内の行 ID の合計数。
NUMRIDS_DELETED	BIGINT	No	No	索引内の疑似削除された行 ID の合計数。
NUM_EMPTY_LEAFS	BIGINT	No	No	索引内の空白リーフ・ページの合計数。
ACTIVE_BLOCKS	BIGINT	No	No	表内のアクティブなマルチディメンション・クラスタリング (MDC) ブロックの合計数。

表 160. 可能な OBJECT_TYPE 値

値	説明
IX	索引
NK	ニックネーム
RX	RCT 索引
TA	表
TF	表関数
+A	コンパイラー参照の別名
+C	コンパイラー参照の制約
+F	コンパイラー参照の関数
+G	コンパイラー参照のトリガー
+N	コンパイラー参照のニックネーム
+T	コンパイラー参照の表
+V	コンパイラー参照のビュー

EXPLAIN_OPERATOR 表

EXPLAIN_OPERATOR 表は、 SQL コンパイラーが SQL ステートメントを満たすために必要とするすべての演算子を格納します。

表 161. EXPLAIN_OPERATOR 表： PK は、列が主キーの一部であることを示します。FK は、列が外部キーの一部であることを示します。

列名	データ・タイプ	NULL 可能?	キー?	説明
EXPLAIN_REQUESTER	VARCHAR(128)	No	FK	この Explain 要求を開始した許可 ID。
EXPLAIN_TIME	TIMESTAMP	No	FK	Explain 要求の開始時刻。
SOURCE_NAME	VARCHAR(128)	No	FK	動的ステートメントに Explain 要求を出したときに実行していたパッケージの名前、または静的 SQL に Explain 要求を出したときのソース・ファイルの名前。
SOURCE_SCHEMA	VARCHAR(128)	No	FK	Explain 要求のソースのスキーマ、または修飾子。
SOURCE_VERSION	VARCHAR(64)	No	FK	Explain 要求のソースのバージョン。
EXPLAIN_LEVEL	CHAR(1)	No	FK	この行に関連する Explain 情報のレベル。
STMTNO	INTEGER	No	FK	パッケージ内のステートメントのうち、この Explain 情報に関連するもののステートメント番号。
SECTNO	INTEGER	No	FK	この Explain 情報に関連したパッケージのセクション番号。
OPERATOR_ID	INTEGER	No	No	この照会内の演算子のユニークな ID。
OPERATOR_TYPE	CHAR(6)	No	No	演算子のタイプの記述ラベル。
TOTAL_COST	DOUBLE	No	No	この演算子に至るまで (この演算子を含む) の、選択したアクセス・プランの実行にかかる合計コスト (timeron 単位) の累積の見積もり。
IO_COST	DOUBLE	No	No	この演算子に至るまで (この演算子を含む) の、選択したアクセス・プランの実行にかかる入出力コスト (データ・ページの入出力単位) の累積の見積もり。
CPU_COST	DOUBLE	No	No	選択したアクセス・プランをこの演算子まで実行した場合にかかる CPU コスト (命令数) の累積の見積もり。
FIRST_ROW_COST	DOUBLE	No	No	この演算子までのアクセス・プランで 1 行目を取り出した場合にかかる累積合計（timeron 数）の見積もり。この値には、必要な初期オーバーヘッドが入ります。
RE_TOTAL_COST	DOUBLE	No	No	この演算子に至るまで (この演算子を含む) の、選択したアクセス・プランの次の行の取り出しにかかるコスト (timeron 単位) の累積の見積もり。
RE_IO_COST	DOUBLE	No	No	この演算子に至るまで (この演算子を含む) の、選択したアクセス・プランの次の行の取り出しにかかる入出力コスト (データ・ページの入出力単位) の累積の見積もり。
RE_CPU_COST	DOUBLE	No	No	この演算子に至るまで (この演算子を含む) の、選択したアクセス・プランの次の行のフェッチにかかる CPU コスト (命令数) の累積の見積もり。
COMM_COST	DOUBLE	No	No	この演算子に至るまで (この演算子を含む) の、選択したアクセス・プランの実行にかかる通信コスト (TCP/IP フレーム単位) の累積の見積もり。
FIRST_COMM_COST	DOUBLE	No	No	この演算子に至るまで (この演算子を含む) の、選択したアクセス・プランの最初の行の取り出しにかかる通信コスト (TCP/IP フレーム単位) の累積の見積もり。この値には、必要な初期オーバーヘッドが入ります。
BUFFERS	DOUBLE	No	No	この演算子とその入力に必要なバッファーの見積もり。

表 161. EXPLAIN_OPERATOR 表 (続き): PK は、列が主キーの一部であることを示します。FK は、列が外部キーの一部であることを示します。

列名	データ・タイプ	NULL 可能?	キー?	説明
REMOTE_TOTAL_COST	DOUBLE	No	No	リモート・データベース操作の実行にかかる合計コスト (timeron 単位) の累積の見積もり。
REMOTE_COMM_COST	DOUBLE	No	No	この演算子に至るまで (この演算子を含む) の、選択したリモート・アクセス・プランの実行にかかる通信コストの累積の見積もり。

表 162. OPERATOR_TYPE の値

値	説明
DELETE	削除
EISCAN	拡張索引スキャン
FETCH	取り出し
FILTER	行フィルター
GENROW	生成行
GRPBY	Group By
HSJOIN	ハッシュ結合
INSERT	挿入
IXAND	動的ビットマップ索引 ANDing
IXSCAN	索引スキャン
MSJOIN	マージ結合
NLJOIN	ネスト・ループの結合
RETURN	結果
RIDSCN	行 ID (RID) スキャン
SHIP	リモート・システムへの照会の配送
SORT	ソート
TBSCAN	表スキャン
TEMP	一時表構造
TQ	テーブル・キュー
UNION	UNION
UNIQUE	複写除去
UPDATE	更新

EXPLAIN_PREDICATE 表

EXPLAIN_PREDICATE は、特定の演算子によって適用される述部を指定します。

表 163. EXPLAIN_PREDICATE 表： PK は、列が主キーの一部であることを示します。FK は、列が外部キーの一部であることを示します。

列名	データ・タイプ	NULL 可能?	キー?	説明
EXPLAIN_REQUESTER	VARCHAR(128)	No	FK	この Explain 要求を開始した許可 ID。
EXPLAIN_TIME	TIMESTAMP	No	FK	Explain 要求の開始時刻。
SOURCE_NAME	VARCHAR(128)	No	FK	動的ステートメントに Explain 要求を出したときに実行していたパッケージの名前、または静的 SQL に Explain 要求を出したときのソース・ファイルの名前。
SOURCE_SCHEMA	VARCHAR(128)	No	FK	Explain 要求のソースのスキーマ、または修飾子。
SOURCE_VERSION	VARCHAR(64)	No	FK	Explain 要求のソースのバージョン。
EXPLAIN_LEVEL	CHAR(1)	No	FK	この行に関連する Explain 情報のレベル。
STMTNO	INTEGER	No	FK	パッケージ内のステートメントのうち、この Explain 情報に関連するもののステートメント番号。
SECTNO	INTEGER	No	FK	この Explain 情報に関連したパッケージのセクション番号。
OPERATOR_ID	INTEGER	No	No	この照会内の演算子のユニークな ID。
PREDICATE_ID	INTEGER	No	No	特定の演算子のための述部のユニークな ID。
HOW_APPLIED	CHAR(5)	No	No	特定の演算子によって述部がどのように使用されているか。
WHEN_EVALUATED	CHAR(3)	No	No	この述部で使用される副照会をいつ評価するかの指示。 使用できる値は次のとおりです。 **ブランク** この述部には副照会が入っていません。 **EAA** この述部で使用される副照会は、適用時に評価されます (EAA)。つまり、指定の演算子によって行が処理されるたびに、述部が適用されるので副照会が再評価されます。 **EAO** この述部で使用される副照会は、オープン時に評価されます (EAO)。つまり、指定の演算子に対して副照会は 1 回だけ再評価され、結果はそれぞれの行へ述部を適用する際に再利用されます。 **MUL** この述部の副照会のタイプは複数あります。
RELOP_TYPE	CHAR(2)	No	No	この述部で使用される関係演算子のタイプ。
SUBQUERY	CHAR(1)	No	No	この述部に対して、副照会からのデータ・ストリームが要求されているか。複数の副照会ストリームが要求されることがあります。 使用できる値は次のとおりです。 **N** 副照会ストリームは要求されていない **Y** 1 つ以上の副照会ストリームが要求されている
FILTER_FACTOR	DOUBLE	No	No	この述部が修飾する小数部の行数の見積もり。

表 163. EXPLAIN_PREDICATE 表 (続き): PK は、列が主キーの一部であることを示します。FK は、列が外部キーの一部であることを示します。

列名	データ・タイプ	NULL 可能?	キー?	説明
PREDICATE_TEXT	CLOB(2M)	Yes	No	SQL ステートメントの内部表示から再作成された述部のテキスト。 ステートメントのコンパイル時にホスト変数、特殊レジスター、またはパラメーター・マーカーの値が使われると、大括弧に囲まれた述部テキストの末尾にこの値が示されます。 値が EXPLAIN_PREDICATE 表に保管されるのは、DBADM 権限をもったユーザーがステートメントを実行した場合か、または DB2 レジストリー変数 DB2_VIEW_REOPT_VALUES が YES に設定されている場合のみです。それ以外の場合、述部テキストの末尾に空の大括弧が表示されます。 使用不可の場合は NULL 値。

表 164. 可能な HOW_APPLIED 値

値	記述
BSARG	各ブロックにつき 1 つの検索引き数述部と評価された。
JOIN	以前は表を結合した。
RESID	残余述部と評価された。
SARG	索引またはデータ・ページの検索引き数述部と評価された。
START	開始条件として使用された。
STOP	停止条件として使用された。

表 165. 可能な RELOP_TYPE の値

値	記述
ブランク	適用されない
EQ	等しい
GE	より大きいかまたは等しい
GT	より大きい
IN	リストされている
LE	より小さいかまたは等しい
LK	類似
LT	より小さい
NE	等しくない
NL	NULL 値
NN	NULL 値以外

EXPLAIN_STATEMENT 表

EXPLAIN_STATEMENT 表には、さまざまなレベルの Explain 情報に関する SQL ステートメントのテキストが入ります。この表には、ユーザーが入力した元の SQL ステートメントと、その SQL ステートメントを満たすアクセス・プランを選択するのに (オプティマイザーで) 使用されるバージョンとが保管されます。後のバージョンは、書き直されているか、 SQL コンパイラーで判別された追加の述部によって拡張されているか、またはその両方であるので、元のバージョンとはあまり類似していないことがあります。

表 166. EXPLAIN_STATEMENT 表： PK は、その列が主キーの一部であることを示します。 FK は、列が外部キーの一部であることを示します。

列名	データ型	NULL 可能?	キー?	説明
EXPLAIN_REQUESTER	VARCHAR(128)	No	PK、FK	この Explain 要求を開始した許可 ID。
EXPLAIN_TIME	TIMESTAMP	No	PK、FK	Explain 要求の開始時刻。
SOURCE_NAME	VARCHAR(128)	No	PK、FK	動的ステートメントに Explain 要求を出したときに実行していたパッケージの名前、または静的 SQL に Explain 要求を出したときのソース・ファイルの名前。
SOURCE_SCHEMA	VARCHAR(128)	No	PK、FK	Explain 要求のソースのスキーマ、または修飾子。
SOURCE_VERSION	VARCHAR(64)	No	FK	Explain 要求のソースのバージョン。
EXPLAIN_LEVEL	CHAR(1)	No	PK	この行に関連する Explain 情報のレベル。 有効な値は次のとおりです。 **O** 元のテキスト (ユーザーが入力したもの) **P** PLAN SELECTION
STMTNO	INTEGER	No	PK	この Explain 情報に関連したパッケージ内のステートメントの番号。動的 Explain SQL ステートメントの場合は 1。静的 SQL ステートメントの場合、この値は SYSCAT.STATEMENTS カタログ・ビューで使用されているものと同じです。
SECTNO	INTEGER	No	PK	パッケージ内のセクションのうち、この SQL ステートメントを収めたもののセクション番号です。動的 Explain SQL ステートメントの場合、これは、実行時にこのステートメントのセクションを保持するのに使用されるセクション番号です。静的 SQL ステートメントの場合、この値は SYSCAT.STATEMENTS カタログ・ビューで使用されているものと同じです。
QUERYNO	INTEGER	No	No	Explain 対象の SQL ステートメントの数値 ID。 CLP または CLI を介して発行された動的 SQL ステートメントの場合 (EXPLAIN SQL ステートメントを除く)、デフォルト値は 1 ずつ順番に大きくなる値です。そうでない場合、デフォルト値は静的 SQL ステートメントでは STMTNO の値で、動的 SQL ステートメントでは 1 です。

表 166. EXPLAIN_STATEMENT 表 (続き): PK は、その列が主キーの一部であることを示します。 FK は、列が外部キーの一部であることを示します。

列名	データ型	NULL 可能?	キー?	説明
QUERYTAG	CHAR(20)	No	No	Explain 対象の各 SQL ステートメントの ID タグ。 CLP を介して発行された動的 SQL ステートメントの場合 (EXPLAIN SQL ステートメントは除く)、デフォルト値は 'CLP' です。 CLI を介して発行された動的 SQL ステートメントの場合 (EXPLAIN SQL ステートメントは除く)、デフォルト値は「CLI」です。それ以外の場合、使用されるデフォルト値はブランクです。
STATEMENT_TYPE	CHAR(2)	No	No	Explain 対象の照会のタイプの記述ラベル。 使用できる値は以下のとおりです。 **S** 選択 **D** 削除 **DC** カーソルの現在位置の削除 **I** 挿入 **U** 更新 **UC** カーソルの現在位置の更新
UPDATABLE	CHAR(1)	No	No	このステートメントが更新可能であると見なされるかどうかを示します。これは特に、潜在的に更新可能であると見なされる可能性のある SELECT ステートメントに関係しています。 使用できる値は以下のとおりです。 ' ' 該当しない (ブランク) **N** No **Y** Yes
DELETABLE	CHAR(1)	No	No	このステートメントが削除可能であると見なされるかどうかを示します。これは特に、潜在的に削除可能であると見なされる可能性のある SELECT ステートメントに関係しています。 使用できる値は以下のとおりです。 ' ' 該当しない (ブランク) **N** No **Y** Yes
TOTAL_COST	DOUBLE	No	No	このステートメントについて選択されたアクセス・プランの実行のための合計コストの見積もり (timeron 単位)。 EXPLAIN_LEVEL が *O* (オリジナル・テキスト) の場合は、この時点で選択されているアクセス・プランがないため、ゼロに設定されます。
STATEMENT_TEXT	CLOB(2M)	No	No	Explain 対象の SQL ステートメントのテキスト、またはその一部。 Explain 機能のプラン選択レベルで表示されるテキストは、内部表記から再構成されたものであり、本質的に SQL ステートメントに類似したものです。再構成されたステートメントは、正しい SQL 構文に必ず準拠しているとは限りません。

表 166. EXPLAIN_STATEMENT 表 (続き)： PK は、その列が主キーの一部であることを示します。 FK は、列が外部キーの一部であることを示します。

列名	データ型	NULL 可能?	キー?	説明
SNAPSHOT	BLOB(10M)	Yes	No	示されている Explain_Level での、この SQL ステートメントの内部表記のスナップショット。 この列は、DB2 Visual Explain で使用することを意図したものです。 EXPLAIN_LEVEL が *0* (元のステートメント) の場合は、ステートメントのこの特定バージョンがキャプチャーされた時点でアクセス・プランが選択されていないため、この列は NULL 値に設定されます。
QUERY_DEGREE	INTEGER	No	No	Explain の呼び出し時のパーティション内並列処理の度合い。元のステートメントの場合、ここには指定された度合いのパーティション内並列処理が入ります。 PLAN SELECTION の場合、ここには使用のプランに応じて生成されたパーティション内並列処理の度合いが入ります。

EXPLAIN_STREAM 表

EXPLAIN_STREAM 表は、個々の演算子とデータ・オブジェクトの間の、入出力データ・ストリームを表します。データ・オブジェクト自体は、EXPLAIN_OBJECT 表に示されています。データ・ストリームに関連する演算子は、EXPLAIN_OPERATOR 表にあります。

表 167. EXPLAIN_STREAM 表： PK は、列が主キーの一部であることを示します。FK は、列が外部キーの一部であることを示します。

列名	データ型	NULL 可能?	キー?	説明
EXPLAIN_REQUESTER	VARCHAR(128)	No	FK	この Explain 要求を開始した許可 ID。
EXPLAIN_TIME	TIMESTAMP	No	FK	Explain 要求の開始時刻。
SOURCE_NAME	VARCHAR(128)	No	FK	動的ステートメントに Explain 要求を出したときに実行していたパッケージの名前、または静的 SQL に Explain 要求を出したときのソース・ファイルの名前。
SOURCE_SCHEMA	VARCHAR(128)	No	FK	Explain 要求のソースのスキーマ、または修飾子。
SOURCE_VERSION	VARCHAR(64)	No	FK	Explain 要求のソースのバージョン。
EXPLAIN_LEVEL	CHAR(1)	No	FK	この行に関連する Explain 情報のレベル。
STMTNO	INTEGER	No	FK	この Explain 情報に関連したパッケージ内のステートメントの番号。
SECTNO	INTEGER	No	FK	この Explain 情報に関連したパッケージのセクション番号。
STREAM_ID	INTEGER	No	No	このデータ・ストリームに対する特定の演算子のユニーク ID。
SOURCE_TYPE	CHAR(1)	No	No	このデータ・ストリームのソースを示します。 **O** 演算子 **D** データ・オブジェクト
SOURCE_ID	SMALLINT	No	No	このデータ・ストリームのソースである照会内の、演算子に対するユニーク ID。 SOURCE_TYPE が「D」の場合は -1 に設定されます。
TARGET_TYPE	CHAR(1)	No	No	このデータ・ストリームのターゲットを示します。 **O** 演算子 **D** データ・オブジェクト
TARGET_ID	SMALLINT	No	No	このデータ・ストリームのターゲットである照会内の、演算子に対するユニーク ID。 TARGET_TYPE が「D」の場合は -1 に設定されます。
OBJECT_SCHEMA	VARCHAR(128)	Yes	No	影響を受けるデータ・オブジェクトが属するスキーマ。 SOURCE_TYPE および TARGET_TYPE が共に「O」である場合、NULL に設定します。
OBJECT_NAME	VARCHAR(128)	Yes	No	データ・ストリームのサブジェクトであるオブジェクトの名前。 SOURCE_TYPE および TARGET_TYPE が共に「O」である場合、NULL に設定します。
STREAM_COUNT	DOUBLE	No	No	データ・ストリームのカーディナリティー推定値。
COLUMN_COUNT	SMALLINT	No	No	データ・ストリーム内の列数。
PREDICATE_ID	INTEGER	No	No	ストリームが述部に対する副照会の一部である場合、述部 ID が反映される。それ以外は列は -1 に設定される。

表 167. EXPLAIN_STREAM 表 (続き): PK は、列が主キーの一部であることを示します。FK は、列が外部キーの一部であることを示します。

列名	データ型	NULL 可能?	キー?	説明
COLUMN_NAMES	CLOB(2M)	Yes	No	この列には、このストリームに関連した列の名前や配列情報が入っています。 名前は以下のフォーマットに従います。 `NAME1(A)+NAME2(D)+NAME3+NAME4` ここで、*(A)* は昇順の列、 *(D)* は降順の列を示し、配列情報がないものは、列が配列されていないか、配列が関係ないかのいずれかを示します。
PMID	SMALLINT	No	No	区分化マップの ID。
SINGLE_NODE	CHAR(5)	Yes	No	このデータ・ストリームが単一または複数のパーティションにあるかどうかを示します。 **MULT** 複数のパーティションにある **COOR** コーディネーター・パーティションにある **HASH** ハッシュを使用して指定される **RID** 行 ID を使用して指定される **FUNC** 関数を使用して指定される (HASHEDVALUE() または DBPARTITIONNUM()) **CORR** 相関値を使用して指定される **Numberic** 事前に決められた単一データベース・パーティションに指定される
PARTITION_COLUMNS	CLOB(2M)	Yes	No	このデータ・ストリームがパーティション分割される列のリスト。

ADVISE_INDEX 表

ADVISE_INDEX 表は、推奨索引を示しています。

表 168. ADVISE_INDEX 表： PK は、列が主キーの一部であることを示します。FK は、列が外部キーの一部であることを示します。

列名	データ・タイプ	NULL 可能?	キー?	説明
EXPLAIN_REQUESTER	VARCHAR(128)	No	No	この Explain 要求を開始した許可 ID。
EXPLAIN_TIME	TIMESTAMP	No	No	Explain 要求の開始時刻。
SOURCE_NAME	VARCHAR(128)	No	No	動的ステートメントに Explain 要求を出したときに実行していたパッケージの名前、または静的 SQL に Explain 要求を出したときのソース・ファイルの名前。
SOURCE_SCHEMA	VARCHAR(128)	No	No	Explain 要求のソースのスキーマ、または修飾子。
SOURCE_VERSION	VARCHAR(64)	No	No	Explain 要求のソースのバージョン。
EXPLAIN_LEVEL	CHAR(1)	No	No	この行に関連する Explain 情報のレベル。
STMTNO	INTEGER	No	No	パッケージ内のステートメントのうち、この Explain 情報に関連するもののステートメント番号。
SECTNO	INTEGER	No	No	この Explain 情報に関連したパッケージのセクション番号。
QUERYNO	INTEGER	No	No	Explain 対象の SQL ステートメントの数値 ID。 CLP または CLI を介して発行された動的 SQL ステートメントの場合 (EXPLAIN SQL ステートメントを除く)、デフォルト値は 1 ずつ順番に大きくなる値です。そうでない場合、デフォルト値は静的 SQL ステートメントでは STMTNO の値で、動的 SQL ステートメントでは 1 です。
QUERYTAG	CHAR(20)	No	No	Explain 対象の個々の SQL ステートメントの ID タグ。CLP を介して発行された動的 SQL ステートメントの場合 (EXPLAIN SQL ステートメントは除く)、デフォルト値は 'CLP' です。 CLI を介して発行された動的 SQL ステートメントの場合 (EXPLAIN SQL ステートメントは除く)、デフォルト値は「CLI」です。それ以外の場合、使用されるデフォルト値はブランクです。
NAME	VARCHAR(128)	No	No	索引の名前。
CREATOR	VARCHAR(128)	No	No	索引名の修飾子。
TBNAME	VARCHAR(128)	No	No	索引が定義されている表またはニックネームの名前。
TBCREATOR	VARCHAR(128)	No	No	表名の修飾子。
COLNAMES	CLOB(2M)	No	No	列名のリスト。
UNIQUERULE	CHAR(1)	No	No	ユニーク値に関する規則。 D = 重複可 P = 1 次索引 U = ユニークな項目のみ可
COLCOUNT	SMALLINT	No	No	キー内の列数と組み込み列 (もしあれば) の数の合計。
IID	SMALLINT	No	No	索引の内部 ID。
NLEAF	INTEGER	No	No	リーフ・ページの数。統計が収集されていない場合は -1。
NLEVELS	SMALLINT	No	No	索引レベルの数。統計が収集されていない場合は -1。
FIRSTKEYCARD	BIGINT	No	No	第 1 キーの値の数。統計が収集されていない場合は -1。
FULLKEYCARD	BIGINT	No	No	個別の全キー値の数。統計が収集されていない場合は -1。

表 168. ADVISE_INDEX 表 (続き): PK は、列が主キーの一部であることを示します。FK は、列が外部キーの一部であることを示します。

列名	データ・タイプ	NULL 可能?	キー?	説明
CLUSTERRATIO	SMALLINT	No	No	索引によるデータ・クラスタリングの程度。統計が収集されていない場合、または詳細な索引統計が収集されている場合は -1 (それらの場合は CLUSTERFACTOR の方が使用されます)。
CLUSTERFACTOR	DOUBLE	No	No	より高い計算精度のクラスタリング。詳細索引統計を収集していない場合、あるいはニックネームに索引が定義されていない場合は -1。
USERDEFINED	SMALLINT	No	No	ユーザーによる定義。
SYSTEM_REQUIRED	SMALLINT	No	No	次の条件のいずれかを満たす場合は 1。 – この索引が主キー制約またはユニーク・キー制約で必要です。またはこの索引がマルチディメンション・クラスタリング (MDC) 表のディメンション・ブロック索引または複合ブロック索引です。 – これはタイプ表のオブジェクト ID (OID) 列に対する索引です。 次の条件の両方を満たす場合は 2。 – この索引は主キー制約またはユニーク・キー制約に必要です。または、この索引は MDC 表に対するディメンション・ブロック索引またはコンポジット・ブロック索引です。 – これはタイプ表のオブジェクト ID (OID) 列に対する索引です。 それ以外の場合は 0。
CREATE_TIME	TIMESTAMP	No	No	索引の作成された時刻。
STATS_TIME	TIMESTAMP	Yes	No	この索引について記録されている統計値が最後に変更された時刻。統計が利用できない場合は、NULL。
PAGE_FETCH_PAIRS	VARCHAR(254)	No	No	文字形式で表された整数ペアのリスト。それぞれのペアは、仮のバッファー内のページ数と、その仮のバッファーを使用した表のスキャンに必要なページ取り出しの回数を表しています。 (データが利用できない場合は、長さ 0 のストリング。)
REMARKS	VARCHAR(254)	Yes	No	ユーザー提供のコメントまたは NULL 値。
DEFINER	VARCHAR(128)	No	No	索引を作成したユーザー。
CONVERTED	CHAR(1)	No	No	将来の使用のために予約済み。
SEQUENTIAL_PAGES	INTEGER	No	No	索引キーの順序でディスクに存在し、それらの間に大きなギャップがないか、わずかなギャップしかないリーフ・ページの数。 (統計が入手できない場合は -1。)
DENSITY	INTEGER	No	No	索引によって占有されているページの範囲内の、ページ数に対する SEQUENTIAL_PAGES の比率。パーセントで表されます (0 から 100 の整数。統計が入手できない場合は -1)。
FIRST2KEYCARD	BIGINT	No	No	索引の最初の 2 つの列を使用するキーの種類数 (統計がない場合、または適用されない場合は -1)。
FIRST3KEYCARD	BIGINT	No	No	索引の最初の 3 つの列を使用するキーの種類数 (統計がない場合、または適用されない場合は -1)。
FIRST4KEYCARD	BIGINT	No	No	索引の最初の 4 つの列を使用するキーの種類数 (統計がない場合、または適用されない場合は -1)。

表 168. ADVISE_INDEX 表 (続き): PK は、列が主キーの一部であることを示します。FK は、列が外部キーの一部であることを示します。

列名	データ・タイプ	NULL 可能?	キー?	説明
PCTFREE	SMALLINT	No	No	索引を最初に作成する際に予約する索引リーフ・ページのパーセント。このスペースは、索引の作成後に行う挿入用に使用可能です。
UNIQUE_COLCOUNT	SMALLINT	No	No	ユニーク・キーに必要な列の数。常に <=COLCOUNT。組み込み列がある場合にのみ < COLCOUNT。索引にユニーク・キーがない場合は -1 (重複可能)。
MINPCTUSED	SMALLINT	No	No	ゼロでない場合は、オンライン索引デフラグが使用可能になり、その値は、ページをマージをする前に使用される最小スペースのしきい値です。
REVERSE_SCANS	CHAR(1)	No	No	Y = 索引は逆スキャンをサポートする N = 索引は逆スキャンをサポートしない
USE_INDEX	CHAR(1)	Yes	No	Y = 推奨または評価された索引 N = 推奨されない索引 R = 既存のクラスタリング RID 索引の非クラスター化が推奨された (設計アドバイザーによって)。これは、表に対して新規のクラスタリング RID 索引が推奨されたケースです。
CREATION_TEXT	CLOB(2M)	No	No	索引の作成に使用された SQL ステートメント。
PACKED_DESC	BLOB(1M)	Yes	No	表の内部記述。
RUN_ID	TIMESTAMP	Yes	FK	ADVISE_INSTANCE 表内の行の START_TIME に対応し、同じ設計アドバイザーの実行にリンクする値。
INDEXTYPE	VARCHAR(4)	No	No	索引のタイプ。 CLUS = クラスタリング REG = 通常 DIM = ディメンション・ブロック索引 BLOK = ブロック索引
EXISTS	CHAR(1)	No	No	データベース・カタログ内に索引が存在する場合は、'Y' に設定します。
RIDTOBLOCK	CHAR(1)	No	No	設計アドバイザー内にブロック索引を作成するのに RID 索引が使われていた場合は、'Y' に設定します。

ADVISE_INSTANCE 表

ADVISE_INSTANCE 表には、開始時刻などの **db2advis** の実行に関する情報が入っています。 **db2advis** の実行ごとに行が 1 つずつ入ります。他の ADVISE 表には、同じ設計アドバイザーの実行中に作成された行ごとに、 ADVISE_INSTANCE 表の START_TIME 列にリンクする外部キー (RUN_ID) が添付されます。

表 169. ADVISE_INSTANCE 表： PK は、その列が主キーの一部であることを意味します。 FK は、その列が外部キーの一部であることを意味します。

列名	データ型	NULL 可能 ?	キー ?	記述
START_TIME	TIMESTAMP	No	PK	**db2advis** の実行が開始された時刻。
END_TIME	TIMESTAMP	No	No	**db2advis** の実行が終了した時刻。
MODE	VARCHAR(4)	No	No	設計アドバイザー上で -m オプションを使って指定された値。たとえば、'MC' は MQT と MDC を指定します。
WKLD_COMPRESSION	CHAR(4)	No	No	設計アドバイザーの実行で使われたワークロード圧縮。
STATUS	CHAR(9)	No	No	設計アドバイザーの実行の状況。状況は'STARTED' または 'COMPLETED' (正常完了時) になりますが、内部エラーの場合は 'EI' を、外部エラーの場合は 'EX' を前に付けられたエラー番号になります。後者の場合のエラー番号は SQLCODE を表します。

ADVISE_MQT 表

ADVISE_MQT 表には、設計アドバイザーから推奨されたマテリアライズ照会表 (MQT) に関する情報が入っています。

表 170. ADVISE_MQT 表： PK は、その列が主キーの一部であることを意味します。 FK は、その列が外部キーの一部であることを意味します。

列名	データ型	NULL 可能 ?	キー ?	記述
EXPLAIN_REQUESTER	VARCHAR(128)	No	No	この Explain 要求を開始した許可 ID。
EXPLAIN_TIME	TIMESTAMP	No	No	Explain 要求の開始時刻。
SOURCE_NAME	VARCHAR(128)	No	No	動的ステートメントに Explain 要求を出したときに実行していたパッケージの名前、または静的 SQL に Explain 要求を出したときのソース・ファイルの名前。
SOURCE_SCHEMA	VARCHAR(128)	No	No	Explain 要求のソースのスキーマ、または修飾子。
SOURCE_VERSION	VARCHAR(64)	No	No	Explain 要求のソースのバージョン。
EXPLAIN_LEVEL	CHAR(1)	No	No	この行に関連する Explain 情報のレベル。
STMTNO	INTEGER	No	No	この Explain 情報に関連するパッケージ内のステートメントの番号。
SECTNO	INTEGER	No	No	この Explain 情報に関連するパッケージ内のステートメントの番号。
NAME	VARCHAR(128)	No	No	MQT 名。
CREATOR	VARCHAR(128)	No	No	MQT 作成者名
IID	SMALLINT	No	No	内部 ID
CREATE_TIME	TIMESTAMP	No	No	MQT が作成された時刻。
STATS_TIME	TIMESTAMP	Yes	No	統計がとられた時刻。
NUMROWS	DOUBLE	No	No	MQT 内の推定行数。
NUMCOLS	SMALLINT	No	No	MQT に定義されている列の数。
ROWSIZE	DOUBLE	No	No	MQT 内の行の平均の長さ (バイト単位)。
BENEFIT	FLOAT	No	No	将来の使用のために予約されています。
USE_MQT	CHAR(1)	Yes	No	MQT が推奨された場合は 'Y' に設定します。
MQT_SOURCE	CHAR(1)	Yes	No	MQT 候補が生成された場所を示します。 MQT 候補が即時リフレッシュ MQT の場合は 'I' に、リフレッシュの据え置き MQT としてのみ作成できる場合は 'D' に設定します。
QUERY_TEXT	CLOB(2M)	No	No	MQT を定義する照会が入っています。
CREATION_TEXT	CLOB(2M)	No	No	MQT 用の CREATE TABLE DDL が入っています。
SAMPLE_TEXT	CLOB(2M)	No	No	MQT に関する詳細な統計を得るのに使われるサンプリング照会が入っています。詳細な統計が設計アドバイザーで必要な場合のみ使用します。その結果のサンプル化統計がこの表に示されます。これが NULL の場合、この MQT のサンプリング照会は作成されませんでした。
COLSTATS	CLOB(2M)	No	No	MQT の列統計が入っています (NULL でない場合)。その統計は XML フォーマットになっていて、中には列名、列カーディナリティー、および選択しだいで HIGH2KEY 値と LOW2KEY 値が組み入れられます。
EXTRA_INFO	BLOB(2M)	No	No	その他の出力用に予約されています。
TBSPACE	VARCHAR(128)	No	No	MQT に対して推奨される表スペース。
RUN_ID	TIMESTAMP	Yes	FK	ADVISE_INSTANCE 表内の行の START_TIME に対応し、同じ設計アドバイザーの実行にリンクする値。

表 170. ADVISE_MQT 表 (続き)： PK は、その列が主キーの一部であることを意味します。 FK は、その列が外部キーの一部であることを意味します。

列名	データ型	NULL 可能 ?	キー ?	記述
REFRESH_TYPE	CHAR(1)	No	No	即時の場合は 'I' に、据え置きの場合は 'D' に設定します。
EXISTS	CHAR(1)	No	No	データベース・カタログ内に MQT が存在する場合は、'Y' に設定します。

ADVISE_PARTITION 表

ADVISE_PARTITION 表には、設計アドバイザーから推奨されたデータベース・パーティションに関する情報が入りますが、この表にデータを追加できるのはパーティション・データベース環境においてのみです。

表 171. ADVISE_PARTITION 表： PK は、その列が主キーの一部であることを意味します。 FK は、その列が外部キーの一部であることを意味します。

列名	データ型	NULL 可能 ?	キー ?	記述
EXPLAIN_REQUESTER	VARCHAR(128)	No	No	この Explain 要求を開始した許可 ID。
EXPLAIN_TIME	TIMESTAMP	No	No	Explain 要求の開始時刻。
SOURCE_NAME	VARCHAR(128)	No	No	動的ステートメントに Explain 要求を出したときに実行していたパッケージの名前、または静的 SQL に Explain 要求を出したときのソース・ファイルの名前。
SOURCE_SCHEMA	VARCHAR(128)	No	No	Explain 要求のソースのスキーマ、または修飾子。
SOURCE_VERSION	VARCHAR(64)	No	No	Explain 要求のソースのバージョン。
EXPLAIN_LEVEL	CHAR(1)	No	No	この行に関連する Explain 情報のレベル。
STMTNO	INTEGER	No	No	この Explain 情報に関連するパッケージ内のステートメントの番号。
SECTNO	INTEGER	No	No	この Explain 情報に関連するパッケージ内のステートメントの番号。
QUERYNO	INTEGER	No	No	Explain 対象の SQL ステートメントの数値 ID。 CLP または CLI を介して発行された動的 SQL ステートメントの場合 (EXPLAIN SQL ステートメントを除く)、デフォルト値は 1 ずつ順番に大きくなる値です。そうでない場合、デフォルト値は静的 SQL ステートメントに対する STMTNO の値、また動的 SQL ステートメントに対しては 1 となります。
QUERYTAG	CHAR(20)	No	No	Explain 対象の各 SQL ステートメントの ID タグ。 CLP を介して発行された動的 SQL ステートメントの場合 (EXPLAIN SQL ステートメントは除く)、デフォルト値は 'CLP' です。 CLI を介して発行された動的 SQL (Explain SQL ステートメント以外) に対するデフォルト値は 'CLI' です。それ以外の場合、使用されるデフォルト値はブランクです。
TBNAME	VARCHAR(128)	Yes	No	表名を指定します。
TBCREATOR	VARCHAR(128)	Yes	No	表作成者名を指定します。
PMID	SMALLINT	Yes	No	区分化マップ ID を指定します。
TBSPACE	VARCHAR(128)	Yes	No	表を置く表スペースのページ・サイズを指定します。
COLNAMES	CLOB(2M)	Yes	No	パーティション列名をコンマで区切って指定します。
COLCOUNT	SMALLINT	Yes	No	パーティション列の数を指定します。
REPLICATE	CHAR(1)	Yes	No	パーティションを複製するかどうかを指定します。
COST	DOUBLE	Yes	No	パーティションの使用コストを指定します。
USEIT	CHAR(1)	Yes	No	EVALUATE PARTITION モードでパーティションを使用するかどうかを指定します。 USEIT を 'Y' または 'y' に設定した場合は、パーティションを使用します。
RUN_ID	TIMESTAMP	Yes	FK	ADVISE_INSTANCE 表内の行の START_TIME に対応し、同じ設計アドバイザーの実行にリンクする値。

ADVISE_TABLE 表

ADVISE_TABLE 表は、マテリアライズ照会表 (MQT) に関する設計アドバイザーの最終勧告、マルチディメンション・クラスター表 (MDC)、およびパーティション化を使用して表を作成するためのデータ定義言語 (DDL) を保管します。

表 172. ADVISE_TABLE 表： PK は、その列が主キーの一部であることを意味します。 FK は、その列が外部キーの一部であることを意味します。

列名	データ型	NULL 可能 ?	キー ?	記述
RUN_ID	TIMESTAMP	Yes	FK	ADVISE_INSTANCE 表内の行の START_TIME に対応し、同じ設計アドバイザーの実行にリンクする値。
TABLE_NAME	VARCHAR(128)	No	No	表の名前。
TABLE_SCHEMA	VARCHAR(128)	No	No	表作成者の名前。
TABLESPACE	VARCHAR(128)	No	No	表の作成場所である表スペース。
SELECTION_FLAG	VARCHAR(4)	No	No	勧告タイプを示します。有効値は、MQT の場合は 'M'、パーティションの場合は 'P'、MDC の場合は 'C' です。このフィールドには、これらの値の任意のサブセットを入れることができます。たとえば、'MC' は、表を MQT 表と MDC 表とすることが推奨されたことを示します。
TABLE_EXISTS	CHAR(1)	No	No	データベース・カタログ内に表が存在する場合は、'Y' に設定します。
USE_TABLE	CHAR(1)	No	No	表に関する設計アドバイザーからの勧告がある場合は、'Y' に設定します。
GEN_COLUMNS	CLOB(2M)	No	No	表 DDL の作成内に生成された列を必要とする MDC 勧告がこの行に入っている場合、生成された列ストリングが入ります。
ORGANIZE_BY	CLOB(2M)	No	No	MDC 勧告の場合は、表 DDL の作成の ORGANIZE BY 文節が入ります。
CREATION_TEXT	CLOB(2M)	No	No	表 DDL の作成が入ります。
ALTER_COMMAND	CLOB(2M)	No	No	表の ALTER TABLE ステートメントが入ります。

ADVISE_WORKLOAD 表

ADVISE_WORKLOAD 表は、ワークロードを構成するステートメントを示しています。

表 173. ADVISE_WORKLOAD 表： PK は、列が主キーの一部であることを示します。FK は、列が外部キーの一部であることを示します。

列名	データ型	NULL 可能?	キー?	説明
WORKLOAD_NAME	CHAR(128)	No	No	このステートメントが属している SQL ステートメント (ワークロード) の集合の名前。
STATEMENT_NO	INTEGER	No	No	ワークロード内のステートメントのうち、この Explain 情報に関連するもののステートメント番号。
STATEMENT_TEXT	CLOB(1M)	No	No	SQL ステートメントの内容。
STATEMENT_TAG	VARCHAR(256)	No	No	Explain 対象の個々の SQL ステートメントの ID タグ。
FREQUENCY	INTEGER	No	No	ワークロード内でこのステートメントが使用される回数。
IMPORTANCE	DOUBLE	No	No	ステートメントの重要性。
WEIGHT	DOUBLE	No	No	ステートメントの優先度。
COST_BEFORE	DOUBLE	Yes	No	推奨された索引が作成されない場合の照会にかかるコスト (timerons 単位)。
COST_AFTER	DOUBLE	Yes	No	推奨された索引が作成される場合の照会にかかるコスト (timerons 単位)。
COMPILABLE	CHAR(17)	Yes	No	ステートメントを準備しようとしている間に生じた照会コンパイル・エラーを示します。この列が NULL であるか、または SQLCA で始まっていない場合、 SQL 照会は db2advis でコンパイルできます。コンパイル・エラーが db2advis または設計アドバイザーによって検出される場合、 COMPILABLE 列値は、8 文字の SQLCA.sqlcaid フィールド、コロン (:)、および 8 文字の SQLCA.sqlstate フィールドで構成されますが、この値は SQL ステートメントの戻りコードです。

付録 J. EXPLAIN レジスター値

以下は CURRENT EXPLAIN MODE および CURRENT EXPLAIN SNAPSHOT 特殊レジスターの値の相互作用、また PREP および BIND コマンドとの相互作用の説明です。

動的 SQL で CURRENT EXPLAIN MODE および CURRENT EXPLAIN SNAPSHOT 特殊レジスターの値には以下の相互作用があります。

表 174. Explain 特殊レジスターの値の相互作用 (動的 SQL)

EXPLAIN SNAPSHOT の値	EXPLAIN MODE の値					
	NO	YES	EXPLAIN	REOPT	RECOMMEND INDEXES	EVALUATE INDEXES
NO	• 戻された照会の結果	• データを追加された Explain 表 • 戻された照会の結果	• データを追加された Explain 表 • 戻されなかった照会 (実行されなかった動的ステートメント) の結果	• 実行時にステートメントが再最適化の対象となったときにデータを追加される Explain 表。 • 戻された照会の結果	• データを追加された Explain 表 • 戻されなかった照会 (実行されなかった動的ステートメント) の結果 • 推奨された索引	• データを追加された Explain 表 • 戻されなかった照会 (実行されなかった動的ステートメント) の結果 • 評価された索引
YES	• とられた Explain スナップショット • 戻された照会の結果	• データを追加された Explain 表 • とられた Explain スナップショット • 戻された照会の結果	• データを追加された Explain 表 • とられた Explain スナップショット • 戻されなかった照会 (実行されなかった動的ステートメント) の結果	• 実行時にステートメントが再最適化の対象となったときにデータを追加される Explain 表。 • とられた Explain スナップショット • 戻された照会の結果	• データを追加された Explain 表 • とられた Explain スナップショット • 戻されなかった照会 (実行されなかった動的ステートメント) の結果 • 推奨された索引	• データを追加された Explain 表 • とられた Explain スナップショット • 戻されなかった照会 (実行されなかった動的ステートメント) の結果 • 評価された索引

表 174. Explain 特殊レジスターの値の相互作用 (動的 SQL) (続き)

EXPLAIN SNAPSHOT の値	EXPLAIN MODE の値					
	NO	YES	EXPLAIN	REOPT	RECOMMEND INDEXES	EVALUATE INDEXES
EXPLAIN	• とられた Explain スナップショット • 戻されなかった照会 (実行されなかった動的ステートメント) の結果	• データを追加された Explain 表 • とられた Explain スナップショット • 戻されなかった照会 (実行されなかった動的ステートメント) の結果	• データを追加された Explain 表 • とられた Explain スナップショット • 戻されなかった照会 (実行されなかった動的ステートメント) の結果	• 実行時にステートメントが再最適化の対象となったときにデータを追加される Explain 表。 • 実行時にステートメントが再最適化の対象となったときにとられる Explain スナップショット。 • 戻されなかった照会 (実行されなかった動的または追加バインド・ステートメント) の結果	• データを追加された Explain 表 • とられた Explain スナップショット • 戻されなかった照会 (実行されなかった動的ステートメント) の結果 • 推奨された索引	• データを追加された Explain 表 • とられた Explain スナップショット • 戻されなかった照会 (実行されなかった動的ステートメント) の結果 • 評価された索引

表 174. Explain 特殊レジスターの値の相互作用 (動的 SQL) (続き)

EXPLAIN SNAPSHOT の値	EXPLAIN MODE の値					
	NO	YES	EXPLAIN	REOPT	RECOMMEND INDEXES	EVALUATE INDEXES
REOPT	• 実行時にステートメントが再最適化の対象となったときにとられる Explain スナップショット。 • 戻された照会の結果	• データを追加された Explain 表 • 実行時にステートメントが再最適化の対象となったときにとられる Explain スナップショット。 • 戻された照会の結果	• データを追加された Explain 表 • 実行時にステートメントが再最適化の対象となったときにとられる Explain スナップショット。 • 戻されなかった照会 (実行されなかった動的または追加バインド・ステートメント) の結果	• 実行時にステートメントが再最適化の対象となったときにデータを追加される Explain 表。 • 実行時にステートメントが再最適化の対象となったときにとられる Explain スナップショット。 • 戻された照会の結果	• データを追加された Explain 表 • 実行時にステートメントが再最適化の対象となったときにとられる Explain スナップショット。 • 戻されなかった照会 (実行されなかった動的または追加バインド・ステートメント) の結果 • 推奨された索引	• データを追加された Explain 表 • 実行時にステートメントが再最適化の対象となったときにとられる Explain スナップショット。 • 戻されなかった照会 (実行されなかった動的または追加バインド・ステートメント) の結果 • 評価された索引

CURRENT EXPLAIN MODE 特殊レジスターは、動的 SQL に対して以下のような方法で EXPLAIN BIND オプションと相互作用します。

表 175. EXPLAIN BIND オプションと CURRENT EXPLAIN MODE の相互作用

EXPLAIN MODE の値	EXPLAIN BIND オプションの値			
	NO	YES	REOPT	ALL
NO	• 戻された照会の結果	• 静的 SQL のデータを追加された Explain 表 • 戻された照会の結果	• 実行時にステートメントが再最適化の対象となったときに静的 SQL のデータを追加される Explain 表。 • 実行時にステートメントが再最適化の対象となったときに動的 SQL のデータを追加される Explain 表。 • 戻された照会の結果	• 静的 SQL のデータを追加された Explain 表 • 動的 SQL のデータを追加された Explain 表 • 戻された照会の結果

表 175. EXPLAIN BIND オプションと CURRENT EXPLAIN MODE の相互作用 (続き)

EXPLAIN MODE の値	EXPLAIN BIND オプションの値			
	NO	**YES**	**REOPT**	**ALL**
YES	• 動的 SQL のデータを追加された Explain 表 • 戻された照会の結果	• 静的 SQL のデータを追加された Explain 表 • 動的 SQL のデータを追加された Explain 表 • 戻された照会の結果	• 実行時にステートメントが再最適化の対象となったときに静的 SQL のデータを追加される Explain 表。 • 実行時にステートメントが再最適化の対象となったときに動的 SQL のデータを追加される Explain 表。 • 戻された照会の結果	• 静的 SQL のデータを追加された Explain 表 • 動的 SQL のデータを追加された Explain 表 • 戻された照会の結果
EXPLAIN	• 動的 SQL のデータを追加された Explain 表 • 戻されなかった照会 (実行されなかった動的ステートメント) の結果	• 静的 SQL のデータを追加された Explain 表 • 動的 SQL のデータを追加された Explain 表 • 戻されなかった照会 (実行されなかった動的ステートメント) の結果	• 実行時にステートメントが再最適化の対象となったときに静的 SQL のデータを追加される Explain 表。 • 実行時にステートメントが再最適化の対象となったときに動的 SQL のデータを追加される Explain 表。 • 戻されなかった照会 (実行されなかった動的ステートメント) の結果	• 静的 SQL のデータを追加された Explain 表 • 動的 SQL のデータを追加された Explain 表 • 戻されなかった照会 (実行されなかった動的ステートメント) の結果
REOPT	• 実行時にステートメントが再最適化の対象となったときに動的 SQL のデータを追加される Explain 表。 • 戻された照会の結果	• 実行時にステートメントが再最適化の対象となったときに静的 SQL のデータを追加される Explain 表。 • 実行時にステートメントが再最適化の対象となったときに動的 SQL のデータを追加される Explain 表。 • 戻された照会の結果	• 実行時にステートメントが再最適化の対象となったときに静的 SQL のデータを追加される Explain 表。 • 実行時にステートメントが再最適化の対象となったときに動的 SQL のデータを追加される Explain 表。 • 戻された照会の結果	• 実行時にステートメントが再最適化の対象となったときに静的 SQL のデータを追加される Explain 表。 • 実行時にステートメントが再最適化の対象となったときに動的 SQL のデータを追加される Explain 表。 • 戻された照会の結果

表 175. EXPLAIN BIND オプションと CURRENT EXPLAIN MODE の相互作用 (続き)

EXPLAIN MODE の値	EXPLAIN BIND オプションの値			
	NO	YES	REOPT	ALL
RECOMMEND INDEXES	• 動的 SQL のデータを追加された Explain 表 • 戻されなかった照会 (実行されなかった動的ステートメント) の結果 • 推奨索引	• 静的 SQL のデータを追加された Explain 表 • 動的 SQL のデータを追加された Explain 表 • 戻されなかった照会 (実行されなかった動的ステートメント) の結果 • 推奨索引	• 実行時にステートメントが再最適化の対象となったときに静的 SQL のデータを追加される Explain 表。 • 実行時にステートメントが再最適化の対象となったときに動的 SQL のデータを追加される Explain 表。 • 戻されなかった照会 (実行されなかった動的ステートメント) の結果 • 推奨索引	• 静的 SQL のデータを追加された Explain 表 • 動的 SQL のデータを追加された Explain 表 • 戻されなかった照会 (実行されなかった動的ステートメント) の結果 • 推奨索引
EVALUATE INDEXES	• 動的 SQL のデータを追加された Explain 表 • 戻されなかった照会 (実行されなかった動的ステートメント) の結果 • 索引の評価	• 静的 SQL のデータを追加された Explain 表 • 動的 SQL のデータを追加された Explain 表 • 戻されなかった照会 (実行されなかった動的ステートメント) の結果 • 索引の評価	• 実行時にステートメントが再最適化の対象となったときに静的 SQL のデータを追加される Explain 表。 • 実行時にステートメントが再最適化の対象となったときに動的 SQL のデータを追加される Explain 表。 • 戻されなかった照会 (実行されなかった動的ステートメント) の結果 • 索引の評価	• 静的 SQL のデータを追加された Explain 表 • 動的 SQL のデータを追加された Explain 表 • 戻されなかった照会 (実行されなかった動的ステートメント) の結果 • 索引の評価

CURRENT EXPLAIN SNAPSHOT 特殊レジスターは、動的 SQL に関して次ページのような方法で EXPLSNAP BIND オプションと相互作用します。

表 176. EXPLSNAP BIND オプションと CURRENT EXPLAIN SNAPSHOT の相互作用

EXPLAIN SNAPSHOT の値	EXPLSNAP BIND オプションの値			
	NO	YES	REOPT	ALL
NO	• 戻された照会の結果	• とられた静的 SQL の Explain スナップショット • 戻された照会の結果	• 実行時にステートメントが再最適化の対象となったときにとられた静的 SQL の Explain スナップショット。 • 実行時にステートメントが再最適化の対象となったときにとられた動的 SQL の Explain スナップショット。 • 戻された照会の結果	• とられた静的 SQL の Explain スナップショット • とられた動的 SQL の Explain スナップショット • 戻された照会の結果
YES	• とられた動的 SQL の Explain スナップショット • 戻された照会の結果	• とられた静的 SQL の Explain スナップショット • とられた動的 SQL の Explain スナップショット • 戻された照会の結果	• 実行時にステートメントが再最適化の対象となったときにとられた静的 SQL の Explain スナップショット。 • 実行時にステートメントが再最適化の対象となったときにとられた動的 SQL の Explain スナップショット。 • 戻された照会の結果	• とられた静的 SQL の Explain スナップショット • とられた動的 SQL の Explain スナップショット • 戻された照会の結果
EXPLAIN	• とられた動的 SQL の Explain スナップショット • 戻されなかった照会 (実行されなかった動的ステートメント) の結果	• とられた静的 SQL の Explain スナップショット • とられた動的 SQL の Explain スナップショット • 戻されなかった照会 (実行されなかった動的ステートメント) の結果	• 実行時にステートメントが再最適化の対象となったときにとられた静的 SQL の Explain スナップショット。 • 実行時にステートメントが再最適化の対象となったときにとられた動的 SQL の Explain スナップショット。 • 戻されなかった照会 (実行されなかった動的ステートメント) の結果	• とられた静的 SQL の Explain スナップショット • とられた動的 SQL の Explain スナップショット • 戻されなかった照会 (実行されなかった動的ステートメント) の結果

表 176. EXPLSNAP BIND オプションと CURRENT EXPLAIN SNAPSHOT の相互作用 (続き)

EXPLAIN SNAPSHOT の値	EXPLSNAP BIND オプションの値			
	NO	YES	REOPT	ALL
REOPT	• 実行時にステートメントが再最適化の対象となったときにとられた動的 SQL の Explain スナップショット。	• 実行時にステートメントが再最適化の対象となったときにとられた静的 SQL の Explain スナップショット。	• 実行時にステートメントが再最適化の対象となったときにとられた静的 SQL の Explain スナップショット。	• 実行時にステートメントが再最適化の対象となったときにとられた静的 SQL の Explain スナップショット。
	• 戻された照会の結果	• 実行時にステートメントが再最適化の対象となったときにとられた動的 SQL の Explain スナップショット。	• 実行時にステートメントが再最適化の対象となったときにとられた動的 SQL の Explain スナップショット。	• 実行時にステートメントが再最適化の対象となったときにとられた動的 SQL の Explain スナップショット。
		• 戻された照会の結果	• 戻された照会の結果	• 戻された照会の結果

付録 K. 例外表

例外表は、 IMMEDIATE CHECKED オプションを指定した SET INTEGRITY を使用して検査する対象として指定された表の定義を模倣して作成されたユーザー作成の表です。例外表は、検査対象の表の行のうち制約に違反しているもののコピーを保管するのに使用されます。

LOAD で使用する例外表は、ここで使用するものと同じです。したがってそれらは、SET INTEGRITY ステートメントでの検査中に再使用できます。

例外表の作成規則

例外表を作成する際の規則は、次のとおりです。

- 例外表の最初の『n』列は、検査対象の表の列と同じです。名前、タイプ、および長さなど、すべての列属性が同じでなければなりません。
- 例外表のすべての列は、制約やトリガーに束縛されないようにする必要があります。制約には、参照保全、チェック制約、さらには挿入時にエラーの原因となるユニーク索引の制約が付帯します。
- 例外表の第『(n+1)』列は、オプションの TIMESTAMP 列です。これは、データを検査するための SET INTEGRITY を発行する前に例外表内の行が削除されなかった場合に、同じ表の SET INTEGRITY ステートメントによる検査の連続呼び出しを検出するのに使用します。
- 第『(n+2)』列は、CLOB(32K) タイプまたはそれより大きいタイプでなければなりません。この列は、行内のデータが違反している制約の名前を示すために使用されるもので、オプションではありますが、なるべく使用するようにしてください。この列を用意しなかった場合 (たとえば、元の表の列数がすでに可能な最大値になっていた場合にはそれが可能)、制約違反が検出された行だけがコピーされます。
- 『(n+1)』列と『(n+2)』列の両方を備えた例外表を作成する必要があります。
- 上記の追加列の特定の名前には制約がありません。しかし、タイプの指定は、正確でなければなりません。
- それ以外の列は使用できません。
- 元の表に DATALINK 列がある場合は、例外表の対応する列に NO LINK CONTROL を指定する必要があります。これを指定しておけば、行 (DATALINK 列を持つ) が挿入されてもファイルにリンクされることはなく、例外表から行が選択されてもアクセス・トークンは生成されません。
- 生成される列 (IDENTITY プロパティーも含む) が元の表にある場合は、例外表の対応する列に、生成されるプロパティーを指定しないでください。
- また、SET INTEGRITY を呼び出してデータを検査するユーザーには、例外表に対して INSERT 特権が必要です。

「メッセージ」列の情報は、次ページの構造に従うものになります。

例外表の作成規則

表 177. 例外表のメッセージ列の構造

フィールド番号	内容	サイズ	コメント
1	制約違反の数	5 バイト	先頭に '0' を付加して右そろえ
2	最初の制約違反の種類	1 文字	'K' - チェック制約違反 'F' - 外部キー違反 'G' - 生成列違反 'I' - ユニーク索引違反 [a] 'L' - DATALINK ロード違反 'D' - 削除ルール : カスケード違反
3	制約/列 [b]/ 索引 ID[c] /DLVDESC[d] の長さ	5 バイト	先頭に '0' を付加して右そろえ
4	制約名/列名 [b]/ 索引 ID[c] /DLVDESC[d]	直前のフィールドで指定される長さ	
5	区切り記号	3 文字	<スペース><コロン><スペース>
6	次の制約違反の種類	1 文字	'K' - チェック制約違反 'F' - 外部キー違反 'G' - 生成列違反 'I' - ユニーク索引違反 'L' - DATALINK ロード違反 'D' - 削除ルール : カスケード違反
7	制約/列/索引 ID/ DLVDESC の長さ	5 バイト	先頭に '0' を付加して右そろえ
8	制約名/列名/制約 ID/ DLVDESC	直前のフィールドで指定される長さ	
.....	.....	.....	違反ごとにフィールド 5 から 8 を繰り返す。

- [a] SET INTEGRITY を使用した検査ではユニーク索引違反は起こりません。しかし、FOR EXCEPTION オプションを選択した場合に LOAD を実行すると、それが報告されます。一方、LOAD はチェック制約、生成列、および外部キーに関する違反を例外表に報告しません。
- [b] 生成列の式をカタログ・ビューから取り出すには、 select ステートメントを使用します。たとえば、フィールド 4 が MYSCHEMA.MYTABLE.GEN_1 の場合、 SELECT SUBSTR(TEXT, 1, 50) FROM SYSCAT.COLUMNS WHERE TABSCHEMA='MYSCHEMA' AND TABNAME='MYNAME' AND COLNAME='GEN_1'; は、式の最初の 50 文字を ″AS (<expression>)″ の形式で戻します。
- [c] カタログ・ビューから索引 ID を取り出すには、 select ステートメントを使用します。たとえば、フィールド 4 が 1234 であれば、 SELECT INDSCHEMA, INDNAME FROM SYSCAT.INDEXES WHERE IID=1234 となります。
- [d] DLVDESC は、次の表で説明する DATALINK ロード違反記述子 (DATALINK Load Violation DESCriptor) です。

表 178. DATALINK ロード違反記述子 (DLVDESC)

フィールド番号	内容	サイズ	コメント
1	違反している DATALINK 列の数	4 文字	先頭に '0' を付加して右そろえ
2	最初に違反している列の DATALINK 列番号	4 文字	先頭に '0' を付加して右そろえ

表 178. DATALINK ロード違反記述子 (DLVDESC) (続き)

フィールド番号	内容	サイズ	コメント
2	2 番目に違反している列の DATALINK 列番号	4 文字	先頭に '0' を付加して右そろえ
.....	.....	.....	違反している列番号ごとに繰り返し

注:

- DATALINK 列番号は、該当する表の SYSCAT.COLUMNS に示される COLNO です。

例外表での行の処理

例外表の情報は、必要な任意の方法で処理できます。その行を使用することにより、データを修正したり、元の表にその行を再び挿入したりすることができます。

元の表に INSERT トリガーがないなら、例外表に対する副照会の入った INSERT ステートメントを発行することによって、修正した行を転送します。

INSERT トリガーがあり、トリガーを起動することなく例外表からの修正済みの行によるロードを完了したい場合は、次のような方法があります。

- 目的に合わせて明示的に定義された列において、値に応じて起動されるように INSERT トリガーを設計します。
- 例外表からのデータをアンロードして、LOAD を使用してそれらを付加します。このケースでデータを再検査する場合、 DB2 バージョン 8 では制約違反検査は付加された行のみに限定されません。
- 関連するカタログ表のトリガー・テキストを保存します。次に、INSERT トリガーをドロップし、INSERT を使用して修正された行を例外表から転送します。最後に、保存した情報を使ってもう一度トリガーを作成します。

DB2 バージョン 8 では、例外表から行を挿入する際にトリガーが起動しないように防止する明示的な機能はありません。

ユニーク索引の違反に対しては、行ごとに 1 つの違反しか報告されません。

ロングタイプ・ストリングまたは LOB データ型の値が表の中に入っている場合、ユニーク索引違反があっても、その値は例外表に挿入されません。

例外表の照会

例外表のメッセージ列の構造は、前述の制約の名前、長さ、および区切り文字を連結したリストです。この情報についての照会を作成したい場合があるかもしれません。

たとえば、すべての違反のリストを作成し、各行ごとにその制約名だけを示すための照会を作成してみましょう。元の表 T1 に 2 つの列 C1 および C2 があるとします。また、対応する例外表 E1 には、T1 内のそれらの列に対応する列 C1 およ

び C2 と、メッセージ列としての MSGCOL があるとします。以下の照会では再帰を使っており、行ごとに 1 つの制約名を示します (複数の違反については行を反復します)。

```
WITH IV  (C1, C2, MSGCOL, CONSTNAME, I, J) AS
  (SELECT C1, C2, MSGCOL,
       CHAR(SUBSTR(MSGCOL, 12,
                   INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,7,5)),5,0)))),
       1,
       15+INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,7,5)),5,0))
     FROM E1
   UNION ALL
   SELECT C1, C2, MSGCOL,
       CHAR(SUBSTR(MSGCOL, J+6,
                 INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,J+1,5)),5,0)))),
       I+1,
       J+9+INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,J+1,5)),5,0))
     FROM IV
     WHERE I < INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,1,5)),5,0))
 ) SELECT C1, C2, CONSTNAME FROM IV;
```

特定の制約に違反したすべての行のリストを作成したい場合は、次のようにしてこの照会を拡張します。

```
WITH IV  (C1, C2, MSGCOL, CONSTNAME, I, J) AS
  (SELECT C1, C2, MSGCOL,
       CHAR(SUBSTR(MSGCOL, 12,
                   INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,7,5)),5,0)))),
       1,
       15+INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,7,5)),5,0))
     FROM E1
   UNION ALL
   SELECT C1, C2, MSGCOL,
       CHAR(SUBSTR(MSGCOL, J+6,
                 INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,J+1,5)),5,0)))),
       I+1,
       J+9+INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,J+1,5)),5,0))
     FROM IV
     WHERE I < INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,1,5)),5,0))
 ) SELECT C1, C2, CONSTNAME FROM IV WHERE CONSTNAME = 'constraintname';
```

チェック制約のすべての違反のリストを作成するには、以下のものを実行します。

```
WITH IV  (C1, C2, MSGCOL, CONSTNAME, CONSTTYPE, I, J) AS
  (SELECT C1, C2, MSGCOL,
       CHAR(SUBSTR(MSGCOL, 12,
                   INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,7,5)),5,0)))),
       CHAR(SUBSTR(MSGCOL, 6, 1)),
       1,
       15+INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,7,5)),5,0))
     FROM E1
   UNION ALL
   SELECT C1, C2, MSGCOL,
       CHAR(SUBSTR(MSGCOL, J+6,
                 INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,J+1,5)),5,0)))),
       CHAR(SUBSTR(MSGCOL, J, 1)),
       I+1,
       J+9+INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,J+1,5)),5,0))
     FROM IV
     WHERE I < INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,1,5)),5,0))
 ) SELECT C1, C2, CONSTNAME FROM IV WHERE CONSTTYPE = 'K';
```

付録 L. ルーチンで使用可能な SQL ステートメント

下記の表は、SQL ステートメント (第 1 列で指定されている) を、指定された SQL データ・アクセス指示を使ってルーチンで実行できるかどうかを示します。NO SQL と定義されたルーチン内に実行可能な SQL ステートメントが出現すると、 SQLSTATE 38001 が戻されます。その他の実行コンテキストの場合、どのコンテキストでもサポートされていない SQL ステートメントは SQLSTATE 38003 を戻します。 CONTAINS SQL コンテキスト内で使用できないその他の SQL ステートメントの場合は SQLSTATE 38004 が戻されます。 READS SQL DATA コンテキストの場合は SQLSTATE 38002 が戻されます。 SQL ルーチンの作成中、SQL データ・アクセス指示と一致しないステートメントによって、 SQLSTATE 42985 が戻されます。

ステートメントがルーチンを呼び出す際、ステートメントの有効 SQL データ・アクセス指示は、以下の SQL データ・アクセス指示よりも優先されます。

- 下記の表のステートメントの SQL データ・アクセス指示。
- ルーチンの作成時に指定されたルーチンの SQL データ・アクセス指示。

たとえば、CALL ステートメントには CONTAINS SQL の SQL データ・アクセス指示があります。ただし、READS SQL DATA と定義されたストアード・プロシージャーが呼び出されると、 CALL ステートメントの有効 SQL データ・アクセス指示は READS SQL DATA になります。

ルーチンが SQL ステートメントを呼び出す際、ステートメントの有効 SQL データ・アクセス指示は、ルーチンに対して宣言された SQL データ・アクセス指示を超えてはなりません。たとえば、READS SQL DATA と定義された関数は、MODIFIES SQL DATA と定義されたストアード・プロシージャーを呼び出せません。

表 179. SQL ステートメントと SQL データ・アクセス指示

SQL ステートメント	NO SQL	CONTAINS SQL	READS SQL DATA	MODIFIES SQL DATA
ALTER...	N	N	N	Y
BEGIN DECLARE SECTION	Y(1)	Y	Y	Y
CALL	N	Y	Y	Y
CLOSE	N	N	Y	Y
COMMENT ON	N	N	N	Y
COMMIT	N	N(4)	N(4)	N(4)
COMPOUND SQL	N	Y	Y	Y
CONNECT(2)	N	N	N	N
CREATE	N	N	N	Y
DECLARE CURSOR	Y(1)	Y	Y	Y
DECLARE GLOBAL TEMPORARY TABLE	N	N	N	Y
DELETE	N	N	N	Y

表 179. SQL ステートメントと SQL データ・アクセス指示 (続き)

SQL ステートメント	NO SQL	CONTAINS SQL	READS SQL DATA	MODIFIES SQL DATA
DESCRIBE	N	Y	Y	Y
DISCONNECT(2)	N	N	N	N
DROP ...	N	N	N	Y
END DECLARE SECTION	Y(1)	Y	Y	Y
EXECUTE	N	Y(3)	Y(3)	Y
EXECUTE IMMEDIATE	N	Y(3)	Y(3)	Y
EXPLAIN	N	N	N	Y
FETCH	N	N	Y	Y
FREE LOCATOR	N	Y	Y	Y
FLUSH EVENT MONITOR	N	N	N	Y
GRANT ...	N	N	N	Y
INCLUDE	Y(1)	Y	Y	Y
INSERT	N	N	N	Y
LOCK TABLE	N	Y	Y	Y
OPEN	N	N	Y (5)	Y
PREPARE	N	Y	Y	Y
REFRESH TABLE	N	N	N	Y
RELEASE CONNECTION(2)	N	N	N	N
RELEASE SAVEPOINT	N	N	N	Y
RENAME TABLE	N	N	N	Y
REVOKE ...	N	N	N	Y
ROLLBACK	N	N(4)	N(4)	N(4)
ROLLBACK TO SAVEPOINT	N	N	N	Y
SAVEPOINT	N	N	N	Y
SELECT INTO	N	N	Y (5)	Y
SET CONNECTION(2)	N	N	N	N
SET INTEGRITY	N	N	N	Y
SET 特殊レジスター	N	Y	Y	Y
UPDATE	N	N	N	Y
VALUES INTO	N	N	Y	Y
WHENEVER	Y(1)	Y	Y	Y

注:

1. NO SQL オプションは SQL ステートメントを指定できないことを暗黙指定しますが、実行不能ステートメントに対する制限はありません。
2. どのルーチン実行コンテキストでも、接続管理ステートメントは使用できません。
3. これは、実行しようとするステートメントによって異なります。 EXECUTE ステートメントで指定するステートメントは、有効な個々の SQL アクセス・レベ

ルのコンテキストで使用できるものでなければなりません。たとえば、SQL アクセス・レベル READS SQL DATA が有効な場合は、ステートメントを INSERT、UPDATE、または DELETE にできません。

4. TO SAVEPOINT 文節以外の COMMIT ステートメントおよび ROLLBACK ステートメントは、ストアード・プロシージャーで使用できます。ただし、ストアード・プロシージャーがアプリケーションから直接、またはアプリケーションからネスト・ストアード・プロシージャー呼び出しを介して間接的に呼び出される場合に限ります。 (トリガー、関数、メソッド、またはアトミック・コンパウンド・ステートメントのいずれかが、ストアード・プロシージャーに対する呼び出しチェーンにある場合、作業単位の COMMIT または ROLLBACK は使用できません。)
5. SQL アクセス・レベル READS SQL DATA が有効な場合は、 SELECT INTO ステートメントや、OPEN ステートメントで参照されるカーソルに、一切 SQL データ変更ステートメントを組み込むことができません。

付録 M. コンパイル済みステートメントから呼び出される CALL

データベースに保管されたプロシージャーを呼び出します。たとえば、ストアード・プロシージャーは、データベース側で実行されて、クライアント・アプリケーションにデータを戻します。

SQL CALL ステートメントを使用するプログラムは、クライアントとサーバーの 2 つの部分で実行されるように設計されます。データベースのサーバー・プロシージャーは、クライアント・アプリケーションと同じトランザクション内で実行されます。クライアント・アプリケーションとストアード・プロシージャーが同じパーティションにある場合、ストアード・プロシージャーはローカルに実行されます。

注: この形式の CALL ステートメントは推奨されていません。この形式は、DB2 の以前のバージョンと互換性を保つことのみを目的として提供されています。

呼び出し:

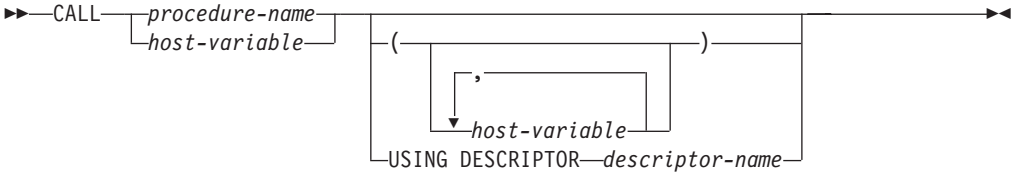
この形式の CALL ステートメントは、CALL_RESOLUTION DEFERRED を指定してプリコンパイルされたアプリケーション・プログラムに組み込む方法のみ可能です。これをトリガー、SQL プロシージャー、または他の非アプリケーション・コンテキストで使用することはできません。これは、動的に作成できない実行可能ステートメントです。ただし、プロシージャー名はホスト変数によって指定することができます。これは、USING DESCRIPTOR 文節の使用とを組み合わせることによって、プロシージャー名とパラメーター・リストの両方をランタイムに用意することができます。これで、動的準備が可能なステートメントに似た効果が得られます。

許可:

CALL ステートメントの許可 ID の特権には、ランタイムに以下の特権のうち少なくとも 1 つが入っていなければなりません。

- ストアード・プロシージャーに関連したパッケージの EXECUTE 特権 (ストアード・プロシージャー上の EXECUTE 特権はチェックされません。)
- ストアード・プロシージャーに関連したパッケージの CONTROL 特権
- SYSADM または DBADM 権限

構文:



説明:

procedure-name **または** *host-variable*
呼び出すプロシージャーを指定します。プロシージャー名は、直接指定するかまたはホスト変数の中に指定できます。指定するプロシージャーは、現行サーバーに存在していなければなりません (SQLSTATE 42724)。

procedure-name を指定する場合、それは 254 バイト以下の通常 ID でなければなりません。通常 ID だけを指定できるので、ブランクや特殊文字を使用することができません。値は大文字に変換されます。小文字名、ブランク、または特殊文字を使用する必要がある場合は、名前を *host-variable* (ホスト変数) によって指定する必要があります。

host-variable が指定されている場合、それは、長さ属性が 254 バイト以下の文字ストリング変数でなければならず、標識変数を組み入れることはできません。値は大文字に変換されません。文字ストリングは、左そろえでなければなりません。

プロシージャー名は、いくつかの形式のいずれかを使用して指定できます。

procedure-name
実行するプロシージャーの名前 (拡張子なし)。呼び出されるプロシージャーは、以下のように決定されます。

1. *procedure-name* を使用して、一致するプロシージャーがあるかどうか定義済みプロシージャー (SYSCAT.ROUTINES の) が探索されます。一致するプロシージャーは、以下の手順で決定されます。
   a. カタログ (SYSCAT.ROUTINES) から、 ROUTINENAME が指定の *procedure-name* と一致し、 ROUTINESCHEMA が SQL パス (CURRENT PATH 特殊レジスター) 内のスキーマ名であるプロシージャー (ROUTINETYPE は 'P') を見つけます。スキーマ名が明示的に指定されている場合、SQL パスは無視され、指定されたスキーマ名のプロシージャーのみが考慮されます。
   b. 次に、CALL ステートメントで指定された引き数の数と同数のパラメーターを持たないプロシージャーをすべて除去します。
   c. 残りのプロシージャーから、SQL パスの最初のプロシージャーを選択します。

   プロシージャーが選択されると、 DB2 は外部名によって定義されたそのプロシージャーを呼び出します。
2. マッチングするプロシージャーが見つからない場合、 *procedure-name* は、ストアード・プロシージャー・ライブラリーの名前と、そのライブラリー中の関数名の両方として使用されます。たとえば、*procedure-name* が `proclib` の場合、 DB2 サーバーは、 `proclib` という名前のストアード・プロシージャー・ライブラリーをロードし、そのライブラリー中の関数ルーチン `proclib()` を実行します。

   UNIX ベースのシステムでは、ストアード・プロシージャー・ライブラリーのデフォルト・ディレクトリーは `sqllib/function` です。囲いのないストアード・プロシージャーのデフォルト・ディレクトリーは、`sqllib/function/unfenced` です。

   Windows ベースのシステムでは、ストアード・プロシージャー・ライブラリーのデフォルト・ディレクトリーは `sqllib¥function` です。囲いのないストアード・プロシージャーのデフォルト・ディレクトリーは、`sqllib¥function¥unfenced` です。

ライブラリーまたは関数が見つからない場合、エラー (SQLSTATE 42884) になります。

procedure-library!function-name
感嘆符 (!) は、ストアード・プロシージャーのライブラリー名と関数名との間の区切り文字です。たとえば、`proclib!func` と指定した場合は、`proclib` がメモリーにロードされ、そのライブラリー中の関数 `func` が実行されます。これによって、1 つのストアード・プロシージャー・ライブラリーの中に複数の関数を入れることができるようになります。

procedure-name の部分で説明したように、ストアード・プロシージャー・ライブラリーは、ディレクトリーに入れるか、あるいは LIBPATH 変数で指定されます。

absolute-path!function-name
absolute-path (絶対パス) には、ストアード・プロシージャー・ライブラリーまでの絶対パス名を指定します。

たとえば、UNIX 系システムの場合に、`/u/terry/proclib!func` が指定されると、ストアード・プロシージャー・ライブラリーの `proclib` がディレクトリー `/u/terry` から取り出され、そのライブラリー内の関数 `func` が実行されます。

いずれの場合も、暗黙の絶対パスまたは明示指定の絶対パスの入ったプロシージャー名の全体の長さは、 254 バイトを超えてはなりません。

(host-variable,...)
それぞれの *host-variable* (ホスト変数) の指定は、CALL ステートメントのパラメーターです。 CALL の *n* 番目のパラメーターは、サーバーのストアード・プロシージャーの *n* 番目のパラメーターに対応します。

各 *host-variable* は、クライアントとサーバー間の双方向のデータ交換に使用されるものと見なされます。クライアントとサーバー間での不要なデータ送信を回避するため、クライアント・アプリケーションでは、各パラメーターに標識変数を指定して、そのパラメーターがストアード・プロシージャーへのデータの送信に使用されない場合にその標識を -1 に設定する必要があります。ストアード・プロシージャーは、クライアント・アプリケーションにデータを返すために使用されないパラメーターすべてについて、標識変数を -128 に設定する必要があります。

サーバーが DB2 Universal Database の場合、パラメーターのデータ型は、クライアントとサーバーの両方のプログラムで一致している必要があります。

USING DESCRIPTOR *descriptor-name*
ホスト変数の有効な記述の入った SQLDA を指定します。 *n* 番目の SQLVAR エレメントは、サーバーのストアード・プロシージャーの *n* 番目のパラメーターに対応します。

CALL ステートメントが処理される前に、アプリケーションでは、 SQLDA 中の以下のフィールドを設定する必要があります。

- SQLDA に用意する SQLVAR のエレメント数を示す SQLN
- SQLDA に割り振るストレージのバイト数を示す SQLDABC
- ステートメントの処理時にその SQLDA の使用される変数の数を示す SQLD

- 変数の属性を示す SQLVAR オカレンス。渡される各基本 SQLVAR エレメントの次のフィールドは、初期化しておく必要があります。
  - SQLTYPE
  - SQLLEN
  - SQLDATA
  - SQLIND

  渡される各 2 次 SQLVAR エレメントの次のフィールドは、初期化しておく必要があります。
  - LEN.SQLLONGLEN
  - SQLDATALEN
  - SQLDATATYPE_NAME

SQLDA は、クライアントとサーバーの間の双方向のデータ交換に使用されるものと見なされます。クライアントとサーバー間での不要なデータの送信を避けるため、クライアント・アプリケーションでは、ストアード・プロシージャーへのデータの送信にパラメーターが使用されない場合に、 SQLIND フィールドを -1 に設定する必要があります。ストアード・プロシージャーは、クライアント・アプリケーションにデータを返すために使用されないパラメーターすべてについて、 SQLIND フィールドを -128 に設定する必要があります。

注:

- ***ラージ・オブジェクト (LOB) データ型の使用***

  クライアントとサーバー・アプリケーションで、 SQLDA から LOB データを指定して、SQLVAR 項目数の 2 倍を割り振る必要があります。

  LOB データ型は、DB2 バージョン 2 以降ストアード・プロシージャーによってサポートされています。 LOB データ型は、それより下位レベルのクライアントまたはサーバーでは、サポートされていません。

- ***SQL プロシージャーからの RETURN_STATUS の検索***

  SQL プロシージャーが RETURN ステートメントを状況値とともに正常に発行すると、この値が SQLCA の最初の SQLERRD フィールドに戻されます。 SQL プロシージャーで CALL ステートメントが発行される場合、 GET DIAGNOSTICS ステートメントを使用して RETURN_STATUS 値を検索します。SQLSTATE がエラーを示す場合は、値は -1 になります。

- ***ストアード・プロシージャーから戻される結果セット***

  クライアント・アプリケーション・プログラムが CLI を使用して作成されている場合、結果セットをクライアント・アプリケーションに直接戻すことができます。ストアード・プロシージャーは、結果セットにカーソルを宣言して、その結果セットでカーソルをオープンし、プロシージャー終了時にカーソルをオープンしたままにすることによって、結果セットを戻すよう指定します。

  プロシージャーの終了時に
  - オープンされたままのカーソルのすべてについて、結果セットがアプリケーションに戻されます。

- 複数のカーソルがオープンされたままの場合、結果セットは、それらのカーソルがオープンされた順序で戻されます。
- 未読の行だけが戻されます。たとえば、カーソルの結果セットに 500 行が入っていて、そのうち 150 行がストアード・プロシージャーの終了時にストアード・プロシージャーによって読み取られた場合、第 151 行から第 500 行までがストアード・プロシージャーに戻されます。

- ***特殊レジスターの取り扱い***

  呼び出し側の特殊レジスターの設定値は、起動時にストアード・プロシージャーに継承され、呼び出し側に戻されるとただちにリストアされます。ストアード・プロシージャー内で特殊レジスターを変更してもかまいませんが、その変更で呼び出し側に影響を与えることはありません。ただし、既存のストアード・プロシージャー (パラメーター・スタイル DB2DARI で定義されているか、またはデフォルト・ライブラリーにあるもの) の場合はそうではなく、プロシージャー内で特殊レジスターに対して加えた変更は、呼び出し側の設定値になります。

- ***互換性***

  アプリケーションに組み込むこと (アプリケーションを CALL_RESOLUTION IMMEDIATE オプションを指定してプリコンパイルすることより)、または動的に準備することが可能な、新しく、推奨される形式の CALL ステートメントがあります。

例:

例 1:

C において、TEAMWINS というプロシージャーを ACHIEVE ライブラリーから呼び出し、ホスト変数 HV_ARGUMENT に保管されているパラメーターをそれに渡します。

```
strcpy(HV_PROCNAME, "ACHIEVE!TEAMWINS");
CALL :HV_PROCNAME (:HV_ARGUMENT);
```

例 2:

C において、:SALARY_PROC というプロシージャーを、 INOUT_SQLDA という名前の SQLDA を使用して呼び出します。

```
struct sqlda *INOUT_SQLDA;
/* Setup code for SQLDA variables goes here */
CALL :SALARY_PROC
USING DESCRIPTOR :*INOUT_SQLDA;
```

例 3:

Java ストアード・プロシージャーが、以下のステートメントを使用してデータベースに定義されています。

```
CREATE PROCEDURE PARTS_ON_HAND (IN PARTNUM INTEGER,
                                OUT COST DECIMAL(7,2),
                                OUT QUANTITY INTEGER)
   EXTERNAL NAME 'parts!onhand'
   LANGUAGE JAVA
   PARAMETER STYLE DB2GENERAL;
```

Java アプリケーションは、以下のコードを使用してこのストアード・プロシージャーを呼び出します。

```
...
CallableStatement stpCall;

String sql = "CALL PARTS_ON_HAND (?,?,?)";

stpCall = con.prepareCall( sql ) ; /* con is the connection */

stpCall.setInt( 1, variable1 ) ;
stpCall.setBigDecimal( 2, variable2 ) ;
stpCall.setInt( 3, variable3 ) ;

stpCall.registerOutParameter( 2, Types.DECIMAL, 2 ) ;
stpCall.registerOutParameter( 3, Types.INTEGER ) ;

stpCall.execute() ;

variable2 = stpCall.getBigDecimal(2) ;
variable3 = stpCall.getInt(3) ;
...
```

このアプリケーションのコード部分は、クラス *parts* の Java メソッド *onhand* を呼び出します。これは、CALL ステートメントで指定されたプロシージャー名がデータベースで検出され、外部名 'parts!onhand' を持っているためです。

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CALL ステートメント』

付録 N. 日本語および中国語 (繁体字) の拡張 UNIX コード (EUC) の考慮事項

日本語および中国語 (繁体字) の拡張 UNIX コード (EUC) では、 1 から 4 文字セットをサポートできるエンコード規則の集合を定義しています。日本語 EUC (eucJP) や中国語 (繁体字) EUC (eucTW) など、場合によっては 2 バイトより多いバイト数を使用して文字がエンコードされている場合もあります。このようなエンコード・スキームを使用するのは、データベース・サーバーまたはデータベース・クライアントのコード・ページとして使用する場合です。主な考慮事項には、以下のものがあります。

- EUC コード・ページと 2 バイトのコード・ページとの間で変換されるときに拡張または短縮されるストリング
- eucJP (日本語) または eucTW (中国語 (繁体字)) コード・ページで定義されたデータベース・サーバー中に格納された図形データのコード・ページとして使用される汎用文字セット 2 (UCS-2)

これらの考慮事項を除き、EUC を使用しても、 2 バイト文字セット (DBCS) サポートと矛盾しません。本書 (および他の資料) を通じて、 *2 バイト文字* に言及している箇所は、*マルチバイト文字* に変更されています。このため、2 バイトより多いバイト数を必要とする文字表示を可能にするエンコード規則のサポートが可能になります。日本語および繁体字中国語 EUC のサポートについての詳細な考慮事項は、この付録に載せられてます。この情報は、EUC データベース・サーバーまたは EUC データベース・クライアントで SQL を使用する場合に考慮する必要があり、アプリケーション開発情報とともに使用します。

言語エレメント

文字

各マルチバイト文字は*文字* と見なされますが、例外として 2 バイトのブランク文字は*特殊文字* と見なされます。

トークン

マルチバイトの英小文字は大文字に変換されません。ただし、通常大文字に変換されるトークン中の 1 バイトの英小文字は例外です。

ID

SQL ID

2 バイトのコード・ページと EUC コード・ページとの間の変換により、 2 バイト文字を 2 バイトより大きいバイト数でエンコードされるマルチバイト文字に変換できます。その結果、2 バイトのコード・ページの最大長に合う ID が、 EUC コード・ページの長さを超えることがあります。このタイプの環境の ID は慎重に選択し、ID の最大長を超えないようにする必要があります。

データ型

文字ストリング

MBCS データベースでは、文字ストリングには 1 バイト文字セット (SBCS) とマルチバイト文字セット (MBCS) の文字が混合して入る場合があります。そのようなストリングを使用する場合、それらが文字ベース (データを文字単位で処理) であるか、またはバイト・ベース (データをバイト単位で処理) であるかによって同じ操作でも結果が異なることがあります。関数または操作の説明を調べて、混合ストリングを処理する方法を決めてください。

GRAPHIC ストリング

GRAPHIC ストリングは、一連の 2 バイト文字データと定義されます。日本語または中国語 (繁体字) の EUC データを GRAPHIC 列に記憶できるようにするために、 EUC は UCS-2 でエンコードされています。サポートされているすべてのエンコード・スキーム (たとえば、 PC または EBCDIC DBCS) の中で 2 バイト文字ではない文字は、GRAPHIC 列では使用できません。 2 バイト文字以外の文字を使用すると、その結果変換中に代用文字によって置換されることがあります。そのようなデータを検索しても、入力された値と同じ値は戻されません。

割り当てと比較

ストリング割り当て: ストリングの変換は、割り当ての前に行われます。eucJP/eucTW コード・ページおよび DBCS コード・ページが入っている場合、文字ストリングは (DBCS から eucJP/eucTW へ) 長くなるか、または (eucJP/eucTW から DBCS へ) 短くなります。これにより、ストレージの割り当て時にエラーが生じ、検索割り当て時に切り捨てが生じることがあります。変換時の拡張のためにストレージ割り当てのエラーが生じる場合、 SQLSTATE 22001 の代わりに SQLSTATE 22524 が返されます。

同様に、GRAPHIC ストリングを割り当てると、その結果 UCS-2 エンコード 2 バイト文字が、対応する 2 バイト文字を持たない文字用に、 PC または EBCDIC DBCS コード・ページ中の代用文字に変換されることがあります。 SQLCA の SQLWARN10 フィールドを 'W' に設定すると、文字を代用文字に置換する割り当てはこのようになります。

マルチバイト文字ストリングが関与した検索割り当ての過程における切り捨ての場合、切り捨てのポイントはマルチバイト文字の一部になることがあります。この場合、文字の一部である各バイトは、1 バイトのブランクに置換されます。このため、複数の 1 バイトのブランクが、切り捨てられた文字ストリングの終わりに現れることがあります。

ストリングの比較: ストリングの比較は、バイト・ベースで行われます。文字ストリングは、データベース用に定義された照合シーケンスも使用します。 GRAPHIC ストリングは照合シーケンスを使用せず、eucJP または eucTW データベースでは、UCS-2 を使用してエンコードされます。このように、たとえ同じ文字が入っていても、2 つの混合文字ストリングの比較と、 2 つの GRAPHIC ストリングの比較とでは異なる結果になることがあります。同様に、その結果生じる混合文字列および GRAPHIC 列のソート順序は異なることがあります。

結果データ型の規則

文字ストリングの結果データ型は、ストリングが拡張しても影響を受けません。たとえば、2 つの CHAR オペランドを結合しても CHAR のままです。しかし、文字ストリングのオペランドの 1 つを変換して最大拡張により長さ属性が 2 つのオペランドのうち最大になるようにする場合、結果文字ストリングの長さ属性は影響を受けます。たとえば、VARCHAR(100) と VARCHAR(120) のデータ型の CASE 式の結果式を考えます。 VARCHAR(100) 式は (変換する必要があるかもしれない) 混合ストリングのホスト変数であり、 VARCHAR(120) 式は eucJP データベース中の列であると仮定します。可能な変換に備えて VARCHAR(100) が 2 バイトになっているため、結果データ型は VARCHAR(200) です。同じシナリオで eucJP データベースまたは eucTW データベースを使用しない場合、結果タイプは VARCHAR(120) になります。

ホスト変数長が 2 倍になっていることは、データベース・サーバーが日本語 EUC または繁体字中国語 EUC であるということに基づいていることに注意してください。クライアントが eucJP または eucTW であったとしても、ホスト変数長は 2 倍にされます。これにより、同じアプリケーション・パッケージを 2 バイトまたはマルチバイトのクライアントが使用できます。

ストリング変換の規則

SQL リファレンスの該当するセクションにリストされているタイプの操作は、オペランドをアプリケーションまたはデータベースのコード・ページに変換することがあります。

日本語または繁体字中国語 EUC から成る混合コード・ページ環境でそのような操作を行うと、混合文字ストリング・オペランドが長くなったり短くなったりすることがあります。そのため、結果のデータ型は可能なら、最大の拡張を収容する長さ属性を持ちます。データ型の長さ属性に制約がある場合、そのデータ型に許された最大長が使用されます。たとえば、最大拡張が 2 倍の環境では、VARCHAR(200) ホスト変数は VARCHAR(400) として、 CHAR(200) ホスト変数は CHAR(254) として処理されます。変換されたストリングがデータ型の最大長を超える場合、変換のランタイムにランタイム・エラーが生じることがあります。たとえば、CHAR(200) と CHAR(10) を結合すると、結果データ型は CHAR(254) になります。 254 文字より多い文字が必要な場合、結合した左側の値が変換されると、エラーが生じます。

場合によっては、変換のための最大増加分の余裕が取られるために、長さ属性が限界を超えることがあります。たとえば、結合に許可される列は 254 バイトまでです。したがって、可変長文字ストリング 128 バイト長と定義された DBCS 混合文字ストリングであるホスト変数が列リスト (:hv1 と呼ぶ) に入っている結合を使用して照会を行うと、アプリケーション内での照会が 254 バイトより大きい列を持っていないように思えたとしても、データ型が VARCHAR(256) に設定され、照会の準備をしているときにエラーが発生してしまいます。実際のストリングが 254 バイトを超えて拡張する可能性が低い場合は、ステートメントの作成に以下のものを使用できます。

```
SELECT CAST(:hv1 CONCAT ' AS VARCHAR(254)), C2 FROM T1
UNION
SELECT C1, C2 FROM T2
```

NULL 値ストリングをホスト変数に連結すると、キャストが実行される前に変換が行われます。この照会は、ランタイムに切り捨てエラーが生じることがありますが、 DBCS で eucJP/eucTW 環境に対して作成できます。

この技法 (NULL 値ストリングとキャストとの連結) を使用して、 SELECT DISTINCT の同じ 254 バイト限界を処理するか、または ORDER BY または GROUP BY 文節の列を使用することができます。

定数

GRAPHIC ストリング定数

日本語または中国語 (繁体字) EUC クライアントの場合、 1 バイト文字またはマルチバイト文字を入れることができます (混合文字ストリングと同様)。ストリングに 2 000 文字より多くの文字を使用することはできません。 GRAPHIC 定数では、すべての関連 PC および EBCDIC 2 バイト・コード・ページの 2 バイト文字に変換される文字だけを使用することをお勧めします。 SQL ステートメント中の GRAPHIC ストリング定数は、クライアント・コード・ページからデータベース・サーバーの 2 バイト・エンコードに変換されます。日本語または繁体字中国語 EUC サーバーでは、定数は UCS-2、すなわち GRAPHIC ストリングに使用される 2 バイト・エンコードに変換されます。 2 バイト・サーバーの場合、定数はクライアントのコード・ページからサーバーの DBCS コード・ページに変換されます。

関数

ユーザー定義関数の設計においては、パラメーターのデータ型に日本語 EUC または中国語 (繁体字) EUC をサポートすることの影響を考慮する必要があります。関数解決の一部では、関数呼び出しに対する引き数のデータ型を考慮します。日本語または繁体字中国語 EUC クライアントから成る混合文字ストリング引き数では、引き数を指定するために追加バイトが必要になることがあります。これには、長さを大きくするためにデータ型の変更が必要になることがあります。たとえば、サーバーの VARCHAR(4000) ストリングに合うアプリケーション (LONG VARCHAR) の文字ストリングを表示するのに、 4001 バイトが必要になることがあります。引き数が LONG VARCHAR であることを示す関数シグニチャーが入っていない場合、関数解決は関数を見つけることができません。

種々の理由で長ストリングが認められていない関数があります。そのような関数に LONG VARCHAR または CLOB 引き数を使用しても成功しません。たとえば、組み込み POSSTR 関数の 2 番目の引き数として LONG VARCHAR を使用すると、関数解決が失敗します (SQLSTATE 42884)。

式

連結演算子

連結オペランドのいずれかが拡張すると、日本語または繁体字中国語 EUC データベース・サーバーを備えた環境では、連結オペランドのデータ型と長さが変更されます。たとえば、ホスト変数の値の長さを 2 倍にする EUC サーバーを使用する場合、以下の例を考慮してください。

```
CHAR200 CONCAT :char50
```

列 *CHAR200* は CHAR(200) タイプです。ホスト変数 *char50* は CHAR(50) と定義されています。この連結演算の結果タイプは通常 CHAR(250) になります。しかし、eucJP または eucTW データベース・サーバーでは、ストリングが拡張して長さが 2 倍になると仮定しています。このため、*char50* は CHAR(100) として処理され、結果データ型は VARCHAR(300) です。結果データ型が VARCHAR であっても、それは末尾ブランクを入れて 300 バイトのデータを常にもっていることに注意してください。余分の後続ブランクが必要ない場合、ホスト変数を CHAR(50) ではなく VARCHAR(50) と定義してください。

述部

LIKE 述部

EUC データベースに混合文字ストリングが組み入れられた LIKE 述部の場合、

- SBCS の半角下線文字は、1 つの SBCS 文字を表します。
- 非 SBCS の全角下線文字は、1 つの非 SBCS 文字を表します。
- SBCS の半角または非 SBCS の全角 % 記号文字は、0 以上の SBCS または非 SBCS 文字を表します。

エスケープ文字は、1 つの SBCS または非 SBCS 文字でなければなりません。文字の列の場合、エスケープ文字は、正確に 1 バイトの入ったバイナリー・ストリングとすることもできます。

下線文字を使用すると、 LIKE 操作のコード・ページによって異なる結果が生じる場合があることに注意してください。たとえば、カタカナ文字は、日本語 EUC ではマルチバイト文字 (CS2) ですが、日本語 DBCS コード・ページでは 1 バイト文字です。 *pattern-expression* の 1 バイトの下線で照会すると、カタカナ文字のオカレンスが日本語 DBCS サーバーから下線の位置に戻されます。しかし、日本語 EUC サーバーの同じ表の同じ行は戻されません。カタカナ文字は 2 バイトの下線にしか一致しないからです。

EUC データベースに GRAPHIC ストリングを組み入れられた LIKE 述部の場合、

- 全角下線文字 (U+FF3F) は、1 つの Unicode 文字を表します。
- 全角の % 記号文字 (U+FF05) は、0 以上の Unicode 文字を表します。

関数

LENGTH

この関数の処理は、EUC 環境の混合文字ストリングでは変わりません。戻される値は、引き数のコード・ページでのストリングの長さです。バージョン 8 では、引き数がホスト変数である場合、戻される値はデータベース・コード・ページでのストリングの長さです。この関数を使用して値の長さを決める場合、その長さを使用する方法を十分考慮する必要があります。これは混合ストリング定数を使用する場合は特に当てはまります。長さが文字単位ではなくてバイト単位で与えられているからです。たとえば、LENGTH 関数によって戻された DBCS データベースの混合ストリング列の長さは、 eucJP または eucTW クライアントのその列の検索値の長さよりも短いことがあります。それは、いくつかの DBCS 文字がマルチバイト eucJP または eucTW 文字に変換されるためです。

SUBSTR

SUBSTR 関数は、混合文字ストリング上でバイト単位で演算を行います。そのため、結果ストリングの最初または終わりにマルチバイト文字のフラグメントが付いていることがあります。文字のフラグメントの検出または処理は行われません。

TRANSLATE

TRANSLATE 関数は、マルチバイト文字を含む混合文字ストリングをサポートします。 *to-string-exp* と *from-string-exp* の対応する文字は、同じ数のバイトを持っていなければならず、マルチバイト文字の一部で終了することはできません。

char-string-exp が文字ストリングである場合、 *pad-char-exp* の結果は 1 バイト文字でなければなりません。 TRANSLATE は *char-string-exp* のコード・ページで実行されるため、 *pad-char-exp* はマルチバイト文字から 1 バイト文字に変換されることがあります。

マルチバイト文字の一部で終了する *char-string-exp* は、変換されたバイトを持ちません。

VARGRAPHIC

日本語 EUC または中国語 (繁体字) EUC のコード・ページの文字ストリング・オペランドに対する VARGRAPHIC 関数は、 UCS-2 コード・ページの GRAPHIC ストリングを戻します。

- 1 バイト文字は、まず最初に、帰属する (eucJP または eucTW) コード・セットの該当する 2 バイト文字に変換されます。その後、該当する UCS-2 表示に変換されます。 2 バイト表示がない場合、文字は、UCS-2 表示に変換される前に、そのコード・セットに定義された 2 バイトの代用文字に変換されます。
- カタカナ (eucJP CS2) である eucJP の文字は実際には、あるエンコード・スキームでは 1 バイト文字です。このように、それらの文字は UCS-2 に変換される前に、 eucJP の該当する 2 バイト文字または 2 バイトの代用文字に変換されます。
- マルチバイト文字は、UCS-2 表示に変換されます。

ステートメント

CONNECT

クライアントまたはサーバーに日本語または繁体字中国語 EUC コード・ページが組み込まれている環境でアプリケーションがデータを処理する可能性がある場合に、 CONNECT ステートメントを正常に処理すると、重要な SQLCA の情報が戻されます。 *SQLERRD(1)* フィールドは、アプリケーションのコード・ページからデータベース・コード・ページに変換するときに混合文字ストリングを最大拡張します。 *SQLERRD(2)* フィールドは、データベース・コード・ページからアプリケーション・コード・ページに変換するときに混合文字ストリングを最大拡張します。拡張が生じる場合の値は正で、短縮が生じる場合の値は負になります。値が負の場合、短縮できず、変換後にストリングの全長が必要になる最悪の状況に備え、値は

常に -1 にしてあります。正の値は 2 と同じ大きさになることがあります。これは、変換後に文字ストリングのストリング長を 2 倍にする必要が生じるという最悪の状況に備えたものです。

アプリケーション・サーバーおよびアプリケーション・クライアントのコード・ページは、 SQLCA の SQLERRMC フィールドでも使用できます。

PREPARE

タイプなしパラメーター・マーカー用に決められたデータ型は、日本語または繁体字中国語 EUC を使用する環境では変更されません。その結果、ある場合、タイプ付きパラメーター・マーカーを使用して、 eucJP または eucTW の混合文字ストリングの十分な長さを提供することが必要になることがあります。たとえば、CHAR(10) 列への挿入を考慮してください。以下のステートメントを準備すると、

```
INSERT INTO T1 (CH10) VALUES (?)
```

その結果パラメーター・マーカーのデータ型は CHAR(10) になります。クライアントが eucJP または eucTW であった場合、挿入するストリングを表示するのに 10 バイトより大きいバイト数が必要になることがありますが、データベースの DBCS コード・ページでは 10 バイトを超えることはありません。この場合、準備するステートメントには、 10 より大きい長さのタイプ付きパラメーター・マーカーが入っている必要があります。このため、以下のステートメントを準備すると、

```
INSERT INTO T1 (CH10) VALUES (CAST(? AS VARCHAR(20))
```

パラメーター・マーカーのデータ型は VARCHAR(20) になります。

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『PREPARE ステートメント』

付録 O. DATALINK のバッカス正規形式 (BNF) 仕様

DATALINK 値はカプセル化された値であり、データベース以外のロケーションに保管されているファイルへのデータベースからの論理参照を収めています。

このカプセル化された値のデータ位置属性は、URL の形式でのファイルへの論理参照です。この属性の値は、以下の BNF によって指定される、URL の構文に準拠しています。これは、RFC 1738 : Uniform Resource Locators (URL), T. Berners-Lee, L. Masinter, M. McCahill, December 1994 に基づいています。 (BNF は ″バッカス正規形式″ (特定の言語の構文を記述するための正式な表記) の頭字語です。)

BNF 指定では、以下の規則が使用されます。

- 代替を指定するのに "|" を使用する
- オプションまたは繰り返されるエレメントの周りに大括弧 [ ] を使用する
- リテラルは "" で囲む
- エレメントの前に [*n*] * を付ければ、その直後のエレメントを *n* 回以上反復させることができる。 *n* が指定されていない場合、デフォルトは 0 です。

DATALINK での BNF 指定を以下に示します。

URL

```
url          =    httpurl | fileurl | uncurl | emptyurl
```

HTTP

```
httpurl      =    "http://" hostport [ "/" hpath ]
hpath        =    hsegment *[ "/" hsegment ]
hsegment     =    *[ uchar | ";" | ":" | "@" | "&" | "=" ]
```

RFC1738 の元の BNF での検索エレメントは除去されている点に注意してください。これは、そのエレメントがファイル参照の本質的な部分ではなく、DATALINK のコンテキストでは意味をなさないためです。

FILE

```
fileurl      =    "file://" host "/" fpath
fpath        =    fsegment *[ "/" fsegment ]
fsegment     =    *[ uchar | "?" | ":" | "@" | "&" | "=" ]
```

RFC1738 とは異なり、host がオプションではなく、 ″localhost″ ストリングが特別な意味を持たない点に注意してください。これにより、″localhost″ の解釈で、クライアント/サーバー構成とパーティション・データベース構成で混乱が生じるのを避けることができます。

UNC

```
uncurl       =    "unc:¥¥" hostname "¥" sharename "¥" uncpath
sharename    =    *uchar
uncpath      =    fsegment *[ "¥" fsegment ]
```

Windows では、一般的に使用されている UNC 命名規則をサポートしています。これは RFC1738 での標準方式ではありません。

EMPTYURL

```
emptyurl     =    ""
hostport     =    host [ ":" port ]
host         =    hostname | hostnumber
hostname     =    *[ domainlabel "." ] toplabel
domainlabel  =    alphadigit | alphadigit *[ alphadigit | "-" ] alphadigit
toplabel     =    alpha | alpha *[ alphadigit | "-" ] alphadigit
alphadigit   =    alpha | digit
hostnumber   =    digits "." digits "." digits "." digits
port         =    digits
```

DATALINK 値では、空の (長さゼロ) URL もサポートされています。これらは、調整例外が報告されて NULL 不可の DATALINK 列が呼び出される場合に、 DATALINK 列を更新するのに便利です。長さゼロの URL を使用すれば、その列が更新され、ファイルへのリンクが解除されます。

各種の定義

```
lowalpha      =   "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" |
                  "i" | "j" | "k" | "l" | "m" | "n" | "o" | "p" |
                  "q" | "r" | "s" | "t" | "u" | "v" | "w" | "x" |
                  "y" | "z"
hialpha       =   "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" |
                  "I" | "J" | "K" | "L" | "M" | "N" | "O" | "P" |
                  "Q" | "R" | "S" | "T" | "U" | "V" | "W" | "X" |
                  "Y" | "Z"
alpha         =   lowalpha | hialpha
digit         =   "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |
                  "8" | "9"
safe          =   "$" | "-" | "_" | "." | "+"
extra         =   "!" | "*" | "'" | "(" | ")" | ","
hex           =   digit | "A" | "B" | "C" | "D" | "E" | "F" |
                  "a" | "b" | "c" | "d" | "e" | "f"
escape        =   "%" hex hex
unreserved    =   alpha | digit | safe | extra
uchar         =   unreserved | escape
digits        =   1*digit
```

先行および末尾ブランク文字は、構文解析中に DB2 によって切り詰められます。また、スキーム名 ('HTTP'、'FILE'、'UNC') および host では大文字小文字が区別されず、常に大文字でデータベースに保管されます。

付録 P. DB2 Universal Database 技術情報

DB2 資料とヘルプ

DB2® 技術情報は、以下のツールと方法を介して利用できます。

- DB2 インフォメーション・センター
  - トピック
  - DB2 ツールのヘルプ
  - サンプル・プログラム
  - チュートリアル
- ダウンロード可能な PDF ファイル、CD 上の PDF ファイル、および印刷された資料
  - ガイド
  - リファレンス・マニュアル
- コマンド行ヘルプ
  - コマンド・ヘルプ
  - メッセージ・ヘルプ
  - SQL 状態ヘルプ
- インストール済みソース・コード
  - サンプル・プログラム

ibm.com® にある技術資料、白書、 Redbooks™ その他の DB2 Universal Database™ 技術情報にオンラインでアクセスできます。 DB2 Information Management ソフトウェア・ライブラリー・サイト (www.ibm.com/software/data/pubs/) にアクセスしてください。

DB2 資料の更新

IBM® は、 DB2 インフォメーション・センターの資料のフィックスパックやその他の資料更新を定期的に発行しています。 DB2 インフォメーション・センター (http://publib.boulder.ibm.com/infocenter/db2help/) にアクセスすれば、常に最新の情報が掲載されます。 DB2 インフォメーション・センターをローカル・インストールしている場合、更新記事を表示するには、まず手動で更新をインストールしてください。新しい情報が発表されたときに資料を更新することにより、 *DB2 インフォメーション・センター CD* からインストールした情報を更新することができます。

インフォメーション・センターの方が、PDF 資料やハードコピー資料よりも頻繁に更新されます。 DB2 の最新の技術情報を入手するには、資料更新が発行されたときにそれをインストールするか、または www.ibm.com サイトの DB2 インフォメーション・センターにアクセスしてください。

関連概念:

- 「*コール・レベル・インターフェース ガイドおよびリファレンス 第 1 巻*」の『CLI サンプル・プログラム』

- 「*アプリケーション開発ガイド アプリケーションの構築および実行*」の『Java サンプル・プログラム』
- 812 ページの『DB2 インフォメーション・センター』

関連タスク:

- 833 ページの『DB2 ツールからコンテキスト・ヘルプを呼び出す』
- 823 ページの『コンピューターまたはイントラネット・サーバーへの DB2 インフォメーション・センターの更新インストール』
- 834 ページの『コマンド行プロセッサーからメッセージ・ヘルプを呼び出す』
- 834 ページの『コマンド行プロセッサーからコマンド・ヘルプを呼び出す』
- 835 ページの『コマンド行プロセッサーから SQL 状態ヘルプを呼び出す』

関連資料:

- 825 ページの『DB2 PDF 資料および印刷された資料』

DB2 インフォメーション・センター

DB2® インフォメーション・センターを使用すると、 DB2 Universal Database™、 DB2 Connect™、 DB2 Information Integrator および DB2 Query Patroller™ などの DB2 ファミリー製品を最大限に活用するのに必要なすべての情報にアクセスできます。また、DB2 インフォメーション・センターは、 DB2 の主な機能とコンポーネントに関する情報を提供します (レプリケーション、データウェアハウジング、および DB2 の種々の Extender など)。

Mozilla 1.0 以上または Microsoft® Internet Explorer 5.5 以上で表示する場合、 DB2 インフォメーション・センターには以下の機能があります。以下のいくつかの機能では、 JavaScript™ のサポートを使用可能にする必要があります:

柔軟なインストール・オプション
以下の中から、 ご使用の環境に最も適したオプションを使って DB2 資料を表示できます。

- 最新の資料を常に自動的に利用できるようにするには、 IBM® の Web サイト (http://publib.boulder.ibm.com/infocenter/db2help/) にある DB2 インフォメーション・センターからすべての資料に直接アクセスします。
- 更新処理を最小化し、イントラネット内のネットワーク・トラフィックだけに制限するには、イントラネット上の 1 つのサーバーに DB2 資料をインストールします。
- 柔軟性を改善し、ネットワーク接続への依存を軽減するには、個々のコンピューターに DB2 資料をインストールします。

検索 「**検索**」テキスト・フィールドに検索語を入力することにより、 DB2 インフォメーション・センターのすべてのトピックを検索できます。複数の語句を引用符で囲めば、完全一致を検索できます。また、ワイルドカード演算子 (*、?) とブール演算子 (AND、 NOT、OR) を使用して検索を絞り込むことができます。

タスク指向の目次
単一の目次の中から、DB2 資料のトピックを見付けることができます。目

次は、主に実行するタスクの種類に従って編成されていますが、そのほかに製品概要、特定のゴール (目的) の情報、参照情報、索引、および用語集も含まれます。

- 製品概要では、DB2 ファミリーで使用可能な製品間の関係、そうした各製品で提供される機能、および各製品の最新リリース情報について説明されています。
- インストール、管理および開発などのゴール・カテゴリーには、タスクを迅速に完了し、そのための背景情報をよく理解できるようにするトピックが含まれています。
- 「参照」トピックでは、その対象に関する詳細な情報 (ステートメントとコマンドの構文、メッセージ・ヘルプ、構成パラメーターなど) が説明されています。

現在のトピックを目次に表示する

現在のトピックが目次のどの部分に該当するかを表示するには、目次フレーム内の**「リフレッシュ/現在のトピックの表示 (Refresh/Show Current Topic)」**ボタンをクリックするか、コンテンツ・フレーム内の**「目次に表示 (Show in Table of Contents)」**ボタンをクリックします。幾つかのファイルで関連トピックへの複数のリンクをたどった場合、または検索結果からトピックにアクセスした場合には、この機能が役立ちます。

索引 索引から、すべての資料にアクセスすることができます。索引では、用語が 50 音順に編成されています。

用語集 用語集を見れば、DB2 資料で使われているさまざまな用語の定義を調べることができます。用語集では、用語が 50 音順に編成されています。

組み込まれているローカライズ情報

DB2 インフォメーション・センターは、ブラウザーで設定された言語でトピックを表示します。設定された言語のトピックが利用できない場合、DB2 インフォメーション・センターにはそのトピックの英語版が表示されます。

iSeries™ 技術情報については、 IBM eServer™ iSeries Information Center (www.ibm.com/eserver/iseries/infocenter/) を参照してください。

関連概念:

- 814 ページの『DB2 インフォメーション・センターのインストール・シナリオ』

関連タスク:

- 823 ページの『コンピューターまたはイントラネット・サーバーへの DB2 インフォメーション・センターの更新インストール』
- 824 ページの『DB2 インフォメーション・センターにおける特定の言語でのトピックの表示』
- 822 ページの『DB2 インフォメーション・センターの呼び出し』
- 816 ページの『DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (UNIX)』
- 819 ページの『DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (Windows)』

DB2 インフォメーション・センターのインストール・シナリオ

さまざまに異なる業務環境のもとでは、 DB2® 情報にどのようにアクセスするかの要件もそれぞれ異なります。 DB2 インフォメーション・センターにアクセスするには、 IBM® の Web サイト、サーバーまたは組織のネットワーク、あるいはコンピューターへのインストールという 3 つの方法が可能です。この 3 つのケースのいずれも、資料は DB2 インフォメーション・センター内に置かれます。インフォメーション・センターは、ブラウザーを使って表示できるように設計されたトピック・ベースの情報の Web サイトです。デフォルトでは、DB2 製品から、 IBM Web サイト上の DB2 インフォメーション・センターにアクセスします。これに対して、 イントラネット・サーバーまたはご自分のコンピューターから DB2 インフォメーション・センターにアクセスしたい場合、製品メディア・パック内にある DB2 インフォメーション・センター CD から DB2 インフォメーション・センターをインストールする必要があります。以下では、DB2 資料へのアクセス・オプションの要約、および 3 つのインストール・シナリオを示します。これを参考にして、お客様の業務環境で DB2 インフォメーション・センターにアクセスするにはどの方法が最適か、どのようなインストール上の問題に配慮する必要があるかを判別してください。

DB2 資料にアクセスするオプションの要約:

以下の表は、お客様の実際の業務環境で、 DB2 インフォメーション・センターの DB2 製品情報にアクセスする方法としてどんなオプションが推奨されるかを示します。

インターネット・アクセス	イントラネット・アクセス	推奨されるアクション
はい	はい	IBM Web サイト上の DB2 インフォメーション・センターへのアクセス、またはイントラネット・サーバーにインストール済みの DB2 インフォメーション・センターへのアクセス
はい	いいえ	IBM Web サイト上の DB2 インフォメーション・センターへのアクセス
いいえ	はい	イントラネット・サーバーにインストール済みの DB2 インフォメーション・センターへのアクセス
いいえ	いいえ	ローカル・コンピューター上の DB2 インフォメーション・センターへのアクセス

シナリオ: コンピューター上の DB2 インフォメーション・センターへのアクセス:

Tsu-Chen 氏は小さな町で工場を経営していますが、その町には、インターネット・アクセスを提供する地元のインターネット・サービス・プロバイダーがありません。彼は、在庫、製品オーダー、銀行口座情報、および営業経費を管理するために DB2 Universal Database™ を購入しました。 Tsu-Chen 氏は以前に DB2 製品を利用したことがないので、 DB2 の使用方法を習得するために、DB2 製品資料を参照する必要があります。

Tsu-Chen 氏は 標準インストール・オプションを使って DB2 Universal Database を自分のコンピューターにインストールした後、 DB2 資料にアクセスしようとします。しかし、開こうとしているページが見つからないというエラー・メッセージがブラウザーから通知されました。 Tsu-Chen 氏は DB2 製品のインストール・マニュアルを調べた結果、 DB2 資料を自分のコンピューター上で利用するには、 DB2 インフォメーション・センターをインストールしなければならないことに気付きます。そしてメディア・パックの中にあった *DB2 インフォメーション・センター CD* を見つけ出して、インストールしました。

これで、Tsu-Chen 氏はオペレーティング・システムのアプリケーション・ランチャーから DB2 インフォメーション・センターにアクセスできるようになり、より良い業務成果をあげるために DB2 製品を利用する方法を習得できます。

シナリオ: IBM Web サイト上の DB2 インフォメーション・センターへのアクセス:

Colin は、あるセミナー企業に所属する情報技術コンサルタントです。彼の専門はデータベース・テクノロジーおよび SQL で、 DB2 Universal Database を使って北米一帯の企業を対象にこれらの科目のセミナーを開催しています。 Colin のセミナーでは、教材として DB2 資料も使用されます。たとえば、SQL の講習コースでは、データベース照会の基本構文と拡張構文を教えるために SQL に関する DB2 資料が使用されます。

Colin が教えている企業の大半はインターネット・アクセスを配備しています。このような状況から判断して、Colin は、最新バージョンの DB2 Universal Database を自分のモバイル・コンピューターにインストールしたとき、 IBM Web サイト上の DB2 インフォメーション・センターにアクセスするよう構成しました。この構成によって、Colin はセミナーで教えるときに最新の DB2 資料にオンライン・アクセスすることができます。

しかし、時折、Colin は移動中にインターネット・アクセスを利用できないことがあります。これは問題となります。担任するセミナーの準備のために DB2 資料にアクセスする必要のある場合には、とくにそうです。このような事態が起きないようにするために、 Colin は自分のモバイル・コンピューターに DB2 インフォメーション・センターのコピーをインストールしました。

こうして、Colin は常に DB2 資料のコピーを自在に活用できるようになりました。**db2set** コマンドを使って自分のモバイル・コンピューターのレジストリー変数を簡単に構成し、どこにいるかに応じて、IBM Web サイトまたは自分のモバイル・コンピューターから DB2 インフォメーション・センターにアクセスできます。

シナリオ: イントラネット・サーバー上の DB2 インフォメーション・センターへのアクセス:

Eva は、生命保険会社のデータベース上級管理者です。彼女は管理業務の一環として、会社の UNIX® データベース・サーバーに最新バージョンの DB2 Universal Database をインストールおよび構成します。彼女の会社は最近、セキュリティー上の理由から、インターネット・アクセスをもはや業務で利用できないようにすると社員に通知しました。同社はネットワーク環境を装備しているため、 Eva は DB2 インフォメーション・センターのコピーをイントラネット・サーバー上にインスト

ールして、社内のデータウェアハウスを定期的に利用するすべての社員 (営業担当者、営業部長、および業務分析担当者) から DB2 資料へのアクセスを可能にすることにしました。

Eva は、応答ファイルを使って全社員のコンピューター上に最新バージョンの DB2 Universal Database をインストールするようデータベース・チームに指示します。その際、イントラネット・サーバーのホスト名とポート番号を使って DB2 インフォメーション・センターにアクセスできるよう、確実に各コンピューターを構成します。

しかし、Eva のチームの下級データベース管理者である Migual の誤解によって、数人の社員のコンピューター上で、イントラネット・サーバーの DB2 インフォメーション・センターにアクセスするよう DB2 Universal Database を構成する代わりに、 DB2 インフォメーション・センターのコピーをそれらのコンピューターにインストールしてしまいました。これを訂正するために、Eva は、 **db2set** コマンドを使ってこれらのコンピューター上の DB2 インフォメーション・センターのレジストリー変数 (ホスト名は DB2_DOCHOST、ポート番号は DB2_DOCPORT) を変更するよう Migual に指示しました。これで、ネットワーク上の適切なすべてのコンピューターが DB2 インフォメーション・センターにアクセスできるようになり、社員は DB2 に関する質問の答えを DB2 資料から見つけることができます。

関連概念:

- 812 ページの『DB2 インフォメーション・センター』

関連タスク:

- 823 ページの『コンピューターまたはイントラネット・サーバーへの DB2 インフォメーション・センターの更新インストール』
- 816 ページの『DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (UNIX)』
- 819 ページの『DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (Windows)』
- 『DB2 インフォメーション・センターへのアクセスのロケーションの設定: Common GUI help』

関連資料:

- 「コマンド・リファレンス」の『db2set - DB2 プロファイル・レジストリー・コマンド』

DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (UNIX)

DB2 製品資料にアクセスする方法として、 IBM Web サイト、イントラネット・サーバー、またはコンピューターにインストールしたバージョンの 3 つがあります。デフォルトでは、DB2 製品は IBM Web サイト上の DB2 資料にアクセスします。イントラネット・サーバーまたはコンピューター上の DB2 資料にアクセスしたい場合には、 *DB2 インフォメーション・センター CD* から資料をインストールする必要があります。 DB2 セットアップ・ウィザードを使用すれば、インストール設

定を定義し、 UNIX オペレーティング・システムを使用するコンピューターに DB2 インフォメーション・センターをインストールできます。

前提条件:

このセクションでは、UNIX コンピューターに DB2 インフォメーション・センターをインストールするためのハードウェア、オペレーティング・システム、ソフトウェア、および通信の諸要件を一覧で示します。

- **ハードウェア要件**

  以下のいずれかのプロセッサーが必要です。

  – PowerPC (AIX)
  – HP 9000 (HP-UX)
  – Intel 32 ビット (Linux)
  – Solaris UltraSPARC コンピューター (Solaris オペレーティング環境)

- **オペレーティング・システム要件**

  以下のいずれかのオペレーティング・システムが必要です。

  – IBM AIX 5.1 (PowerPC 上)
  – HP-UX 11i (HP 9000 上)
  – Red Hat Linux 8.0 (Intel 32 ビット上)
  – SuSE Linux 8.1 (Intel 32 ビット上)
  – Sun Solaris バージョン 8 (Solaris オペレーティング環境の UltraSPARC コンピューター上)

  注: DB2 インフォメーション・センターは、 DB2 クライアントをサポートする UNIX オペレーティング・システム上で稼動します。このため、IBM Web サイトから DB2 インフォメーション・センターにアクセスするか、イントラネット・サーバーに DB2 インフォメーション・センターをインストールしてそれにアクセスすることをお勧めします。

- **ソフトウェア要件**

  – 以下のブラウザーがサポートされています。

    - Mozilla バージョン 1.0 以上

- DB2 セットアップ・ウィザードは、グラフィック・インストーラーです。ご使用のマシンで DB2 セットアップ・ウィザードのグラフィカル・ユーザー・インターフェースを表示可能にする X Window システム・ソフトウェアをインプリメントする必要があります。 DB2 セットアップ・ウィザードを実行する前に、ディスプレイを正しくエクスポートしたことを確認してください。たとえば、コマンド・プロンプトで

  ```
  export DISPLAY=9.26.163.144:0.
  ```

  というコマンドを入力します。

- **通信要件**

  – TCP/IP

手順:

DB2 セットアップ・ウィザードを使用して DB2 インフォメーション・センターをインストールするには、以下のようにします。

1. システムにログオンします。
2. DB2 インフォメーション・センター製品 CD を挿入してシステムにマウントします。
3. 次のコマンドを入力して、CD がマウントされているディレクトリーに移動します。

   ```
   cd /cd
   ```

   /cd は、CD のマウント・ポイントを表します。
4. **./db2setup** コマンドを入力して、 DB2 セットアップ・ウィザードを開始します。
5. IBM DB2 セットアップ・ランチパッドが開きます。 DB2 インフォメーション・センターのインストールに直接進むには、**「製品のインストール」**をクリックします。残りのステップについて説明しているオンライン・ヘルプを利用できます。オンライン・ヘルプを呼び出すには、**「ヘルプ」**をクリックします。**「キャンセル」**をクリックすれば、いつでもインストールを終了できます。
6. **「インストールしたい製品を選択します」**ページでは、**「次へ」**をクリックします。
7. **「DB2 セットアップ・ウィザードにようこそ (Welcome to the DB2 Setup wizard)」**ページで、**「次へ」**をクリックします。 DB2 セットアップ・ウィザードは、プログラムのセットアップ操作を案内します。
8. インストールを続行するには、使用許諾条件に同意する必要があります。**「ご使用条件」**ページで、**「ご使用条件に同意します (I accept the terms in the license agreement)」** をクリックして、**「次へ」**をクリックします。
9. **「インストール・アクションの選択」**で、**「このコンピューターに DB2 インフォメーション・センターをインストールする (Install DB2 Information Center on this computer)」**を選択します。応答ファイルを使用して、このコンピューターまたは他のコンピューターに DB2 インフォメーション・センターをあとでインストールしたい場合には、**「設定を応答ファイルに保管する」**を選択します。**「次へ」**をクリックします。
10. **「インストールする言語の選択」**ページでは、 DB2 インフォメーション・センターをインストールする言語を選択します。**「次へ」**をクリックします。
11. **「DB2 インフォメーション・センター・ポートの指定」**ページでは、 DB2 インフォメーション・センターへの着信通信を構成します。**「次へ」**をクリックしてインストールを続けます。
12. **「ファイルのコピーの開始」**ページでは、インストールの選択項目を確認します。設定を変更するには、**「戻る」**をクリックします。**「インストール」**をクリックすると、 DB2 インフォメーション・センターのファイルがコンピューターにコピーされます。

このほか、応答ファイルを使って DB2 インフォメーション・センターをインストールすることもできます。

インストール・ログ db2setup.his、 db2setup.log、および db2setup.err は、デフォルトでは /tmp ディレクトリーに置かれます。

db2setup.log ファイルは、エラーも含めた DB2 製品のインストール情報をすべてキャプチャーします。 db2setup.his ファイルは、コンピューター上の DB2 製品インストール内容をすべて記録します。 DB2 は、db2setup.log ファイルを db2setup.his に付加します。 db2setup.err ファイルは、 Java から戻されるすべてのエラー出力 (例外やトラップの情報など) をキャプチャーします。

インストールが完了したら、ご使用の UNIX オペレーティング・システムに応じて、 DB2 は以下のいずれかのディレクトリーにインストールされます。

- AIX: /usr/opt/db2_08_01
- HP-UX: /opt/IBM/db2/V8.1
- Linux: /opt/IBM/db2/V8.1
- Solaris オペレーティング環境: /opt/IBM/db2/V8.1

関連概念:

- 812 ページの『DB2 インフォメーション・センター』
- 814 ページの『DB2 インフォメーション・センターのインストール・シナリオ』

関連タスク:

- 「*インストールおよび構成 補足*」の『応答ファイルによる DB2 のインストール (UNIX)』
- 823 ページの『コンピューターまたはイントラネット・サーバーへの DB2 インフォメーション・センターの更新インストール』
- 824 ページの『DB2 インフォメーション・センターにおける特定の言語でのトピックの表示』
- 822 ページの『DB2 インフォメーション・センターの呼び出し』
- 819 ページの『DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (Windows)』

DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (Windows)

DB2 製品資料にアクセスする方法として、 IBM Web サイト、イントラネット・サーバー、またはコンピューターにインストールしたバージョンの 3 つがあります。デフォルトでは、DB2 製品は IBM Web サイト上の DB2 資料にアクセスします。イントラネット・サーバーまたはコンピューター上の DB2 資料にアクセスしたい場合には、 *DB2 インフォメーション・センター CD* から DB2 資料をインストールする必要があります。 DB2 セットアップ・ウィザードを使用すれば、インストール設定を定義し、 Windows オペレーティング・システムを使用するコンピューターに DB2 インフォメーション・センターをインストールできます。

前提条件:

このセクションでは、Windows に DB2 インフォメーション・センターをインストールするためのハードウェア、オペレーティング・システム、ソフトウェア、および通信の諸要件を一覧で示します。

- **ハードウェア要件**

  以下のいずれかのプロセッサーが必要です。

  – 32 ビット・コンピューター: Pentium または Pentium 互換の CPU

- **オペレーティング・システム要件**

  以下のいずれかのオペレーティング・システムが必要です。

  – Windows 2000

  – Windows XP

  注: DB2 インフォメーション・センターは、 DB2 クライアントをサポートする Windows オペレーティング・システム上で稼動します。このため、IBM Web サイトの DB2 インフォメーション・センターにアクセスするか、イントラネット・サーバーに DB2 インフォメーション・センターをインストールしてそれにアクセスすることをお勧めします。

- **ソフトウェア要件**

  – 以下のブラウザーがサポートされています。

    - Mozilla 1.0 以上

    - Internet Explorer バージョン 5.5 または 6.0 (Windows XP の場合はバージョン 6.0)

- **通信要件**

  – TCP/IP

制約事項:

- DB2 インフォメーション・センターをインストールするには、管理権限をもつアカウントが必要です。

手順:

DB2 セットアップ・ウィザードを使用して DB2 インフォメーション・センターをインストールするには、以下のようにします。

1. DB2 インフォメーション・センターのインストールで定義したアカウントで、システムにログオンします。
2. CD をドライブに挿入します。自動実行機能が使用可能になっていれば、IBM DB2 セットアップ・ランチパッドが起動します。
3. DB2 セットアップ・ウィザードは、システム言語を判別して、その言語用のセットアップ・プログラムを立ち上げます。英語以外の言語でセットアップ・プログラムを実行したい場合、またはセットアップ・プログラムの自動始動が失敗した場合には、 DB2 セットアップ・ウィザードを手動で開始できます。

   次のようにして、DB2 セットアップ・ウィザードを手動で開始します。

   a. **「スタート」**をクリックし、**「ファイル名を指定して実行」**を選択します。

   b. **「開く」**フィールドで、以下のコマンドを入力します。

      ```
      x:¥setup.exe /i 2-letter language identifier
      ```

ここで、*x:* は CD ドライブ、 *2-letter language identifier* (2 文字の言語識別子) はセットアップ・プログラムを実行する言語を表します。

c. **「OK」**をクリックします。

4. IBM DB2 セットアップ・ランチパッドが開きます。 DB2 インフォメーション・センターのインストールに直接進むには、**「製品のインストール」**をクリックします。残りのステップについて説明しているオンライン・ヘルプを利用できます。オンライン・ヘルプを呼び出すには、**「ヘルプ」**をクリックします。**「キャンセル」**をクリックすれば、いつでもインストールを終了できます。
5. **「インストールしたい製品を選択します」**ページでは、**「次へ」**をクリックします。
6. **「DB2 セットアップ・ウィザードにようこそ (Welcome to the DB2 Setup wizard)」**ページで、**「次へ」**をクリックします。 DB2 セットアップ・ウィザードは、プログラムのセットアップ操作を案内します。
7. インストールを続行するには、使用許諾条件に同意する必要があります。**「ご使用条件」**ページで、**「ご使用条件に同意します (I accept the terms in the license agreement)」** をクリックして、**「次へ」**をクリックします。
8. **「インストール・アクションの選択」**で、**「このコンピューターに DB2 インフォメーション・センターをインストールする (Install DB2 Information Center on this computer)」**を選択します。応答ファイルを使用して、このコンピューターまたは他のコンピューターに DB2 インフォメーション・センターをあとでインストールしたい場合には、**「設定を応答ファイルに保管する」**を選択します。**「次へ」**をクリックします。
9. **「インストールする言語の選択」**ページでは、 DB2 インフォメーション・センターをインストールする言語を選択します。**「次へ」**をクリックします。
10. **「DB2 インフォメーション・センター・ポートの指定」**ページでは、 DB2 インフォメーション・センターへの着信通信を構成します。**「次へ」**をクリックしてインストールを続けます。
11. **「ファイルのコピーの開始」**ページでは、インストールの選択項目を確認します。設定を変更するには、**「戻る」**をクリックします。**「インストール」**をクリックすると、 DB2 インフォメーション・センターのファイルがコンピューターにコピーされます。

応答ファイルを使って DB2 インフォメーション・センターをインストールすることができます。また、**db2rspgn** コマンドを使って、既存のインストール内容に基づく応答ファイルを生成することもできます。

インストール時に検出されるエラーの詳細については、「マイ ドキュメント」¥DB2LOG¥ ディレクトリー内の db2.log ファイルと db2wi.log ファイルを参照してください。「マイ ドキュメント」 ディレクトリーの場所は、ご使用のコンピューターの設定によって異なります。

db2wi.log ファイルは、 DB2 の最新のインストール情報をキャプチャーします。db2.log は、DB2 製品のインストールの履歴をキャプチャーします。

関連概念:

• 812 ページの『DB2 インフォメーション・センター』

- 814 ページの『DB2 インフォメーション・センターのインストール・シナリオ』

関連タスク:

- 「*インストールおよび構成 補足*」の『応答ファイルによる DB2 製品のインストール (Windows)』
- 823 ページの『コンピューターまたはイントラネット・サーバーへの DB2 インフォメーション・センターの更新インストール』
- 824 ページの『DB2 インフォメーション・センターにおける特定の言語でのトピックの表示』
- 822 ページの『DB2 インフォメーション・センターの呼び出し』
- 816 ページの『DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (UNIX)』

関連資料:

- 「コマンド・リファレンス」の『db2rspgn - 応答ファイル生成プログラム・コマンド』

DB2 インフォメーション・センターの呼び出し

DB2 インフォメーション・センターは、Linux、UNIX、および Windows オペレーティング・システム用の DB2 製品 (DB2 Universal Database、 DB2 Connect、DB2 Information Integrator、 DB2 Query Patroller など) を使用するために必要なすべての情報を提供します。

DB2 インフォメーション・センターは、以下の場所から呼び出すことができます。

- DB2 UDB クライアントまたはサーバーがインストールされているコンピューター
- DB2 インフォメーション・センターがインストールされているイントラネット・サーバーまたはローカル・コンピューター
- IBM の Web サイト

前提条件:

DB2 インフォメーション・センターを呼び出すための要件は、以下のとおりです。

- *オプショナル*: 希望する言語でトピックを表示するようブラウザーを構成する
- *オプショナル*: コンピューターまたはイントラネット・サーバーにインストール済みの DB2 インフォメーション・センターを使用するよう DB2 クライアントを構成する

手順:

DB2 UDB クライアントまたはサーバーがインストールされているコンピューターから DB2 インフォメーション・センターを呼び出すには、以下のようにします。

- (Windows オペレーティング・システムの)「スタート」メニューから: **「スタート」→「プログラム」→「IBM DB2」→「情報」→「インフォメーション・センター」**をクリックします。
- コマンド行プロンプトから:
  - Linux および UNIX オペレーティング・システムの場合、 **db2icdocs** コマンドを発行します。

– Windows オペレーティング・システムの場合、 **db2icdocs.exe** コマンドを発行します。

イントラネット・サーバーまたはローカル・コンピューターにインストール済みの DB2 インフォメーション・センターを Web ブラウザーで開くには、以下のようにします。

- Web ページ http://<host-name>:<port-number>/ を開きます (<host-name> はホスト名、 <port-number> は DB2 インフォメーション・センターを利用可能なポート番号)。

IBM Web サイトにある DB2 インフォメーション・センターを Web ブラウザーで開くには、以下のようにします。

- Web ページ publib.boulder.ibm.com/infocenter/db2help/ を開きます。

関連概念:

- 812 ページの『DB2 インフォメーション・センター』
- 814 ページの『DB2 インフォメーション・センターのインストール・シナリオ』

関連タスク:

- 824 ページの『DB2 インフォメーション・センターにおける特定の言語でのトピックの表示』
- 833 ページの『DB2 ツールからコンテキスト・ヘルプを呼び出す』
- 823 ページの『コンピューターまたはイントラネット・サーバーへの DB2 インフォメーション・センターの更新インストール』
- 834 ページの『コマンド行プロセッサーからコマンド・ヘルプを呼び出す』
- 『DB2 インフォメーション・センターへのアクセスのロケーションの設定: Common GUI help』

関連資料:

- 「*コマンド・リファレンス*」の『HELP コマンド』

コンピューターまたはイントラネット・サーバーへの DB2 インフォメーション・センターの更新インストール

http://publib.boulder.ibm.com/infocenter/db2help/ から利用できる DB2 インフォメーション・センターは、資料の新規追加または変更によって定期的に更新されます。さらに、更新された DB2 インフォメーション・センターをコンピューターまたはイントラネット・サーバーにダウンロードしてインストールできる場合もあります。DB2 インフォメーション・センターを更新しても、 DB2 クライアント製品またはサーバー製品は更新されません。

前提条件:

インターネットに接続されたコンピューターへのアクセスが必要です。

手順:

DB2 インフォメーション・センターの更新をコンピューターまたはイントラネット・サーバーにインストールするには、以下のようにします。

1. IBM の Web サイト (http://publib.boulder.ibm.com/infocenter/db2help/) にある DB2 インフォメーション・センターを開きます。
2. 「DB2 インフォメーション・センターにようこそ」ページの見出し「サービスおよびサポート」の「ダウンロード」セクションで、**「DB2 資料」**リンクをクリックします。
3. 最新のドキュメンテーション・イメージのレベルと、インストール済みのドキュメンテーション・レベルを比較して、 DB2 インフォメーション・センターを更新する必要があるかどうかを確認します。「DB2 インフォメーション・センターにようこそ」ページに、インストール済みのドキュメンテーションのレベルがリストされます。
4. より新しいバージョンの DB2 インフォメーション・センターが存在する場合、ご使用のオペレーティング・システムに対応する最新の *DB2 インフォメーション・センター・イメージをダウンロードします。
5. 最新の *DB2 インフォメーション・センター・イメージをインストールするには、 Web ページの指示に従ってください。

関連概念:

- 814 ページの『DB2 インフォメーション・センターのインストール・シナリオ』

関連タスク:

- 822 ページの『DB2 インフォメーション・センターの呼び出し』
- 816 ページの『DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (UNIX)』
- 819 ページの『DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (Windows)』

DB2 インフォメーション・センターにおける特定の言語でのトピックの表示

DB2 インフォメーション・センターでは、ブラウザーの設定で指定した言語でのトピックの表示が試みられます。トピックがその指定言語に翻訳されていない場合は、 DB2 インフォメーション・センターでは英語でトピックが表示されます。

手順:

Internet Explorer Web ブラウザーで、指定どおりの言語でトピックを表示するには、以下のようにします。

1. Internet Explorer の**「ツール」**—> **「インターネット オプション」**—> **「言語...」**ボタンをクリックします。「言語の優先順位」ウィンドウがオープンします。
2. 該当する言語が、言語リストの先頭の項目に指定されていることを確認します。
   - リストに新しい言語を追加するには、**「追加...」**ボタンをクリックします。

     注: 言語を追加しても、特定の言語でトピックを表示するのに必要なフォントがコンピューターに備えられているとはかぎりません。
   - リストの先頭に新しい言語を移動するには、その言語を選択してから、その言語が言語リストに先頭に行くまで**「上へ」**ボタンをクリックします。

3. 使いたい言語で DB2 インフォメーション・センターを表示するには、ページをリフレッシュします。

Mozilla Web ブラウザーの場合に、使いたい言語でトピックを表示するには、以下のようにします。

1. Mozilla の**「編集」**—> **「設定」**—> **「言語」**ボタンをクリックします。「設定」ウィンドウに「言語」パネルが表示されます。
2. 該当する言語が、言語リストの先頭の項目に指定されていることを確認します。
   - リストに新しい言語を追加するには、**「追加...」**ボタンをクリックしてから、「言語を追加」ウィンドウで言語を選択します。
   - リストの先頭に新しい言語を移動するには、その言語を選択してから、その言語が言語リストに先頭に行くまで**「上に移動」**ボタンをクリックします。
3. 使いたい言語で DB2 インフォメーション・センターを表示するには、ページをリフレッシュします。

関連概念:

- 812 ページの『DB2 インフォメーション・センター』

DB2 PDF 資料および印刷された資料

以下の表は、正式な資料名、資料番号、および PDF ファイル名を示しています。ハードコピー版の資料を注文するには、正式な資料名を知っておく必要があります。 PDF ファイルを印刷するには、PDF ファイル名を知っておく必要があります。

DB2 資料は、以下のカテゴリーに分類されています。

- DB2 中核情報
- 管理情報
- アプリケーション開発情報
- ビジネス・インテリジェンス情報
- DB2 Connect 情報
- 入門情報
- チュートリアル情報
- オプショナル・コンポーネント情報
- リリース・ノート

以下の表は、DB2 ライブラリー内の各資料について、その資料のハードコピー版を注文したり、PDF 版を印刷または表示したりするのに必要な情報を示しています。DB2 ライブラリー内の各資料に関する詳細な説明については、www.ibm.com/shop/publications/order にある IBM Publications Center にアクセスしてください。

DB2 の基本情報

こうした資料の情報は、すべての DB2 ユーザーに基本的なもので、プログラマーおよびデータベース管理者にとって役立つ情報であるとともに、DB2 Connect、DB2 Warehouse Manager、または他の DB2 製品を使用するユーザーにとっても役

立つ内容です。

表 180. DB2 の基本情報

資料名	資料番号	PDF ファイル名
「*IBM DB2 Universal Database コマンド・リファレンス*」	SC88-9140	db2n0j81
「*IBM DB2 Universal Database 用語集*」	資料番号なし	db2t0j81
「*IBM DB2 Universal Database メッセージ・リファレンス 第 1 巻*」	GC88-9152 (ハードコピーなし)	db2m1j81
「*IBM DB2 Universal Database メッセージ・リファレンス 第 2 巻*」	GC88-9153 (ハードコピーなし)	db2m2j81
「*IBM DB2 Universal Database 新機能*」	SC88-9158	db2q0j81

管理情報

これらの資料の情報は、DB2 データベース、データウェアハウス、およびフェデレーテッド・システムを効果的に設計し、インプリメントし、保守するために必要なトピックを扱っています。

表 181. 管理情報

資料名	資料番号	PDF ファイル名
「*IBM DB2 Universal Database 管理ガイド: プランニング*」	SC88-9135	db2d1j81
「*IBM DB2 Universal Database 管理ガイド: インプリメンテーション*」	SC88-9133	db2d2j81
「*IBM DB2 Universal Database 管理ガイド: パフォーマンス*」	SC88-9134	db2d3j81
「*IBM DB2 Universal Database 管理 API リファレンス*」	SC88-9136	db2b0j81
「*IBM DB2 Universal Database データ移動ユーティリティー ガイドおよびリファレンス*」	SC88-9142	db2dmj81
「*IBM DB2 Universal Database データ・リカバリーと高可用性 ガイドおよびリファレンス*」	SC88-9143	db2haj81
「*IBM DB2 Universal Database データウェアハウス・センター 管理ガイド*」	SC88-9165	db2ddj81
「*IBM DB2 Universal Database SQL リファレンス 第 1 巻*」	SC88-9155	db2s1j81
「*IBM DB2 Universal Database SQL リファレンス 第 2 巻*」	SC88-9156	db2s2j81

表 181. 管理情報 (続き)

資料名	資料番号	PDF ファイル名
「*IBM DB2 Universal Database システム・モニター ガイドおよびリファレンス*」	SC88-9157	db2f0j81

アプリケーション開発情報

これらの資料の情報は、 DB2 Universal Database (DB2 UDB) のアプリケーション開発者またはプログラマーが特に関心を持つ内容です。サポートされるさまざまなプログラミング・インターフェース (組み込み SQL、ODBC、JDBC、SQLJ、CLI など) を使用して DB2 UDB にアクセスするのに必要な資料とともに、サポートされる言語およびコンパイラーについても紹介されています。また、DB2 インフォメーション・センターをご使用の場合には、サンプル・プログラムのソース・コードの HTML バージョンにアクセスすることもできます。

表 182. アプリケーション開発情報

資料名	資料番号	PDF ファイル名
「*IBM DB2 Universal Database アプリケーション開発ガイド アプリケーションの構築および実行*」	SC88-9137	db2axj81
「*IBM DB2 Universal Database アプリケーション開発ガイド クライアント・アプリケーションのプログラミング*」	SC88-9138	db2a1j81
「*IBM DB2 Universal Database アプリケーション開発ガイド サーバー・アプリケーションのプログラミング*」	SC88-9139	db2a2j81
「*IBM DB2 Universal Database コール・レベル・インターフェース ガイドおよびリファレンス 第 1 巻*」	SC88-9159	db2l1j81
「*IBM DB2 Universal Database コール・レベル・インターフェース ガイドおよびリファレンス 第 2 巻*」	SC88-9160	db2l2j81
「*IBM DB2 Universal Database データウェアハウス・センター アプリケーション統合ガイド*」	SC88-9166	db2adj81
「*IBM DB2 Universal Database XML Extender 管理およびプログラミングのガイド*」	SC88-9172	db2sxj81

ビジネス・インテリジェンス情報

これらの資料の情報は、さまざまなコンポーネントを使用して、 DB2 Universal Database のデータウェアハウジング機能および分析機能を拡張する方法を説明しています。

表 183. ビジネス・インテリジェンス情報

資料名	資料番号	PDF ファイル名
「*IBM DB2 Warehouse Manager Standard Edition インフォメーション・カタログ・センター 管理ガイド*」	SC88-9167	db2dij81
「*IBM DB2 Warehouse Manager Standard Edition インストール・ガイド*」	GC88-9164	db2idj81
「*IBM DB2 Warehouse Manager Standard Edition DB2 Warehouse Manager を使用時の ETI ソリューション・コンバージョン・プログラムの管理*」	SC88-9894	iwhe1mstx80

DB2 Connect 情報

このカテゴリーの情報は、 DB2 Connect Enterprise Edition または DB2 Connect Personal Edition を使用して、メインフレーム・サーバーおよびミッドレンジ・サーバー上のデータにアクセスする方法を説明しています。

表 184. DB2 Connect 情報

資料名	資料番号	PDF ファイル名
「*IBM コネクティビティー 補足*」	資料番号なし	db2h1j81
「*IBM DB2 Connect Enterprise Edition 概説およびインストール*」	GC88-9145	db2c6j81
「*IBM DB2 Connect Personal Edition 概説およびインストール*」	GC88-9146	db2c1j81
「*IBM DB2 Connect ユーザーズ・ガイド*」	SC88-9147	db2c0j81

入門情報

このカテゴリーの情報は、サーバー、クライアント、および他の DB2 製品をインストールして構成する場合に役立ちます。

表 185. 入門情報

資料名	資料番号	PDF ファイル名
「*IBM DB2 Universal Database DB2 クライアント機能 概説およびインストール*」	GC88-9144 (ハードコピーなし)	db2itj81
「*IBM DB2 Universal Database DB2 サーバー機能 概説およびインストール*」	GC88-9148	db2isj81
「*IBM DB2 Universal Database DB2 Personal Edition 概説およびインストール*」	GC88-9150	db2i1j81
「*IBM DB2 Universal Database インストールおよび構成 補足*」	GC88-9149 (ハードコピーなし)	db2iyj81
「*IBM DB2 Universal Database DB2 Data Links Manager 概説およびインストール*」	GC88-9141	db2z6j81

チュートリアル情報

チュートリアル情報は、DB2 機能を紹介し、さまざまなタスクを実行する方法を示します。

表 186. チュートリアル情報

資料名	資料番号	PDF ファイル名
「*ビジネス・インテリジェンス・チュートリアル: データウェアハウス・センターの紹介*」	資料番号なし	db2tuj81
「*ビジネス・インテリジェンス・チュートリアル: データウェアハウジングの上級者向けガイド*」	資料番号なし	db2taj81
「*インフォメーション・カタログ・センター チュートリアル*」	資料番号なし	db2aij81
「*Video Central for e-business チュートリアル*」	資料番号なし	db2twj81
「*Visual Explain チュートリアル*」	資料番号なし	db2tvj81

オプショナル・コンポーネント情報

このカテゴリーの情報は、 DB2 のオプショナル・コンポーネントを使用する方法について説明しています。

表 187. オプショナル・コンポーネント情報

資料名	資料番号	PDF ファイル名
「*IBM DB2 Cube Views Guide and Reference*」	SC18–7298	db2aax81
「*IBM DB2 Query Patroller インストール、管理、使用法のガイド*」	GC88–9154	db2dwj81
「*IBM DB2 Spatial Extender and Geodetic Extender ユーザーズ・ガイドおよびリファレンス*」	SC88-9171	db2sbj81
「*IBM DB2 Universal Database Data Links Manager 管理ガイドおよびリファレンス*」	SC88-9169	db2z0x82
「*DB2 Net Search Extender 管理およびユーザーズ・ガイド*」 **注:** この資料の HTML 版は、HTML ドキュメンテーション CD からインストールされません。	SH88-8546	N/A

リリース・ノート

リリース・ノートは、ご使用の製品のリリースおよびフィックスパック・レベルに特有の追加情報を紹介します。また、リリース・ノートには、各リリース、アップデート、およびフィックスパックで組み込まれた資料上の更新の要約も含まれています。

表 188. リリース・ノート

資料名	資料番号	PDF ファイル名
「*DB2 リリース・ノート*」	「注」を参照。	「注」を参照。
「*DB2 インストール情報*」	製品 CD-ROM でのみ参照可能。	使用できません。

注: リリース・ノートは以下の形式で入手できます。

- XHTML およびテキスト形式 (製品 CD 内)
- PDF 形式 (PDF ドキュメンテーション CD 内)

さらに、リリース・ノートの中で、『*既知の問題と予備手段*』および『*リリース間の非互換性*』に関する部分は DB2 インフォメーション・センターにも表示されます。

UNIX ベースのプラットフォームでテキスト形式でリリース・ノートを確認するには、 Release.Notes ファイルを参照してください。このファイルは、DB2DIR/Readme/*%L* ディレクトリーに収録されています。 *%L* はロケール名を表しています。 DB2DIR は以下になります。

- AIX オペレーティング・システムの場合: /usr/opt/db2_08_01
- その他のすべての UNIX ベースのオペレーティング・システムの場合: /opt/IBM/db2/V8.1

関連概念:

- 811 ページの『DB2 資料とヘルプ』

関連タスク:

- 831 ページの『PDF ファイルからの DB2 資料の印刷方法』
- 832 ページの『DB2 の印刷資料の注文方法』
- 833 ページの『DB2 ツールからコンテキスト・ヘルプを呼び出す』

PDF ファイルからの DB2 資料の印刷方法

DB2 PDF ドキュメンテーション CD に収録されている DB2 資料を印刷することができます。 Adobe Acrobat Reader を使用すれば、資料全体または特定のページを印刷できます。

前提条件:

Adobe Acrobat Reader がインストールされていることを確認してください。 Adobe Acrobat Reader をインストールする必要がある場合、 Adobe Web サイト (www.adobe.com) から入手できます。

手順:

PDF ファイルから DB2 資料を印刷するには以下のようにします。

1. *DB2 PDF* ドキュメンテーション CD をドライブに挿入します。 UNIX オペレーティング・システムの場合、 DB2 PDF ドキュメンテーション CD をマウントします。 UNIX オペレーティング・システムで CD をマウントする方法については、「*概説およびインストール*」を参照してください。
2. index.htm を開きます。ブラウザー・ウィンドウにファイルが開きます。
3. 参照したい PDF のタイトルをクリックします。 Acrobat Reader で PDF が開きます。
4. **「ファイル」 → 「印刷」**を選択して、所要の資料の任意の部分を印刷します。

関連概念:

- 812 ページの『DB2 インフォメーション・センター』

関連タスク:

- 「*DB2 Universal Database サーバー機能 概説およびインストール*」の『CD-ROM のマウント (AIX)』
- 「*DB2 Universal Database サーバー機能 概説およびインストール*」の『HP-UX 上での CD-ROM のマウント』
- 「*DB2 Universal Database サーバー機能 概説およびインストール*」の『CD-ROM のマウント (Linux)』
- 832 ページの『DB2 の印刷資料の注文方法』

- 「*DB2 Universal Database サーバー機能 概説およびインストール*」の『CD-ROM のマウント (Solaris)』

関連資料:

- 825 ページの『DB2 PDF 資料および印刷された資料』

DB2 の印刷資料の注文方法

ハードコピー版の資料を望む場合には、以下のいずれかの方法で注文できます。

印刷資料の注文方法:

一部の国または地域では、印刷された資料を注文することもできます。お客様がお住まいの国または地域でこのサービスが利用可能かどうかを確認するには、お住まいの国または地域の IBM Publications Web サイトをご覧ください。資料のご注文が可能な場合、以下のようにすることができます。

- 正規の IBM 製品販売業者または営業担当員に連絡してください。お客様がお住まいの地域の IBM 担当員の情報については、お手数ですが IBM の Web サイト (www.ibm.com/planetwide) の IBM Worldwide Directory of Contacts で確認してください。
- IBM Publications Center (http://www.ibm.com/shop/publications/order) にアクセスしてください。なお、IBM Publications Center から資料を注文できない国もあります。

DB2 製品がご利用可能になった時点で、印刷された資料は *DB2 PDF* ドキュメンテーション *CD* にある PDF 形式の資料と同じものです。さらに、 *DB2* インフォメーション・センター *CD* に収録されている印刷された資料の内容もまた、これらと同じです。ただし、DB2 インフォメーション・センター CD には、 PDF 資料にない追加情報も含まれます (たとえば、SQL 管理作業や HTML サンプル)。 DB2 PDF ドキュメンテーション CD に収録されている資料の中には、ハードコピーとしてご注文できない資料もあります。

注: DB2 インフォメーション・センターは、 PDF またはハードコピー の資料よりも頻繁に更新されます。ドキュメンテーションの更新が入手可能になった時点でインストールするか、 DB2 インフォメーション・センター (http://publib.boulder.ibm.com/infocenter/db2help/) を参照して最新の情報を入手してください。

関連タスク:

- 831 ページの『PDF ファイルからの DB2 資料の印刷方法』

関連資料:

- 825 ページの『DB2 PDF 資料および印刷された資料』

DB2 ツールからコンテキスト・ヘルプを呼び出す

コンテキスト・ヘルプは、特定のウィンドウ、ノートブック、ウィザード、またはアドバイザーに関連したタスクまたはコントロールの情報を提供します。コンテキスト・ヘルプは、グラフィカル・ユーザー・インターフェースのある DB2 管理ツールおよび開発ツールから利用できます。コンテキスト・ヘルプには、以下の 2 種類があります。

- それぞれのウィンドウまたはノートブックにある**「ヘルプ」**ボタンからアクセス可能なヘルプ
- infopop (ポップアップ情報ウィンドウ)。これは、マウス・カーソルを特定のフィールドまたはコントロール上に置いたとき、またはウィンドウ、ノートブック、ウィザード、アドバイザー内でフィールドまたはコントロールを選択して F1 を押すと表示されます。

「ヘルプ」ボタンを押すと、概説、前提条件、およびタスク情報が表示されます。infopop は、それぞれのフィールドおよびコントロールについて説明します。

手順:

コンテキスト・ヘルプを呼び出すには、以下のようにします。

- ウィンドウおよびノートブックのヘルプを表示するには、いずれかの DB2 ツールを開始して、任意のウィンドウまたはノートブックを開きます。ウィンドウまたはノートブックの右下隅にある**「ヘルプ」**ボタンをクリックして、コンテキスト・ヘルプを呼び出します。

  また、それぞれの DB2 ツール・センターの上部にある**「ヘルプ」**メニュー項目からコンテキスト・ヘルプにアクセスすることもできます。

  ウィザードおよびアドバイザーでは、最初のページの「タスクの概要」リンクをクリックすると、コンテキスト・ヘルプを表示できます。
- ウィンドウまたはノートブック上の各コントロールの infopop ヘルプを表示するには、コントロールをクリックしてから、**F1** を押します。コントロールの詳細情報を示すポップアップ情報が、黄色いウィンドウに表示されます。

  注: フィールドまたはコントロールにマウス・カーソルを置いておくだけで infopops が表示されるようにするには、「ツール設定」ノートブックの**「文書 (Documentation)」**ページの**「infopops の自動表示」**チェック・ボックスを選択します。

  infopop に似た別のコンテキスト・ヘルプに、診断ポップアップ情報があります。これにはデータ入力規則が示されます。診断ポップアップ情報は、無効または不十分なデータが入力されたとき、紫色のウィンドウに表示されます。診断ポップアップ情報は、以下に関して表示されます。
  - 必須フィールド。
  - 日付フィールドのように、正確なフォーマットを必要とするデータのフィールド。

関連タスク:

- 822 ページの『DB2 インフォメーション・センターの呼び出し』
- 834 ページの『コマンド行プロセッサーからメッセージ・ヘルプを呼び出す』

- 834 ページの『コマンド行プロセッサーからコマンド・ヘルプを呼び出す』
- 835 ページの『コマンド行プロセッサーから SQL 状態ヘルプを呼び出す』
- 『DB2 インフォメーション・センターへのアクセス: Concepts help』
- 『DB2 UDB ヘルプの使用法: Common GUI help』
- 『DB2 インフォメーション・センターへのアクセスのロケーションの設定: Common GUI help』
- 『DB2 コンテキスト・ヘルプと資料へのアクセスを設定する: Common GUI help』

コマンド行プロセッサーからメッセージ・ヘルプを呼び出す

メッセージ・ヘルプは、メッセージが出された原因と、エラーへの応答として実行すべきアクションを説明します。

手順:

メッセージ・ヘルプを呼び出すには、コマンド行プロセッサーを開いて以下のように入力します。

```
? XXXnnnnn
```

ここで、*XXXnnnnn* は有効なメッセージ ID を表します。

たとえば、`? SQL30081` と入力すると、メッセージ SQL30081 に関するヘルプを表示します。

関連概念:

- 「*メッセージ・リファレンス 第 1 巻*」の『メッセージの概要』

関連資料:

- 「コマンド・リファレンス」の『db2 - コマンド行プロセッサーの呼び出しコマンド』

コマンド行プロセッサーからコマンド・ヘルプを呼び出す

コマンド・ヘルプは、コマンド行プロセッサーでのコマンドの構文を説明します。

手順:

コマンド・ヘルプを呼び出すには、コマンド行プロセッサーを開いて以下のように入力します。

```
? command
```

ここで *command* はキーワードまたはコマンド全体を表します。

たとえば、`? catalog` と入力すると、すべての CATALOG コマンドに関するヘルプが表示され、 `? catalog database` と入力すると、 CATALOG DATABASE コマンドのヘルプだけが表示されます。

関連タスク:

- 833 ページの『DB2 ツールからコンテキスト・ヘルプを呼び出す』
- 822 ページの『DB2 インフォメーション・センターの呼び出し』
- 834 ページの『コマンド行プロセッサーからメッセージ・ヘルプを呼び出す』
- 835 ページの『コマンド行プロセッサーから SQL 状態ヘルプを呼び出す』

関連資料:

- 「コマンド・リファレンス」の『db2 - コマンド行プロセッサーの呼び出しコマンド』

コマンド行プロセッサーから SQL 状態ヘルプを呼び出す

DB2 Univerrsal Database は、 SQL ステートメントの結果の原因となったと考えられる条件の SQLSTATE 値を戻します。 SQLSTATE ヘルプは、SQL 状態および SQL 状態クラス・コードの意味を説明します。

手順:

SQL 状態ヘルプを呼び出すには、コマンド行プロセッサーを開いて以下のように入力します。

```
? sqlstate または ? class code
```

ここで、*sqlstate* は有効な 5 桁の SQL 状態を、*class code* は SQL 状態の最初の 2 桁を表します。

たとえば、`? 08003` を指定すると SQL 状態 08003 のヘルプが表示され、 `? 08` を指定するとクラス・コード 08 のヘルプが表示されます。

関連タスク:

- 822 ページの『DB2 インフォメーション・センターの呼び出し』
- 834 ページの『コマンド行プロセッサーからメッセージ・ヘルプを呼び出す』
- 834 ページの『コマンド行プロセッサーからコマンド・ヘルプを呼び出す』

DB2 チュートリアル

DB2® チュートリアルは、 DB2 Universal Database のさまざまな機能について学習するのを支援します。このチュートリアルでは、アプリケーションの開発、SQL 照会のパフォーマンス調整、データウェアハウスの処理、メタデータの管理、および DB2 を使用した Web サービスの開発の各分野で、段階的なレッスンが用意されています。

はじめに:

インフォメーション・センター (http://publib.boulder.ibm.com/infocenter/db2help/) から、このチュートリアルの XHTML 版を表示できます。

チュートリアルの中で、サンプル・データまたはサンプル・コードを使用する場合があります。個々のタスクの前提条件については、それぞれのチュートリアルを参照してください。

DB2 Universal Database チュートリアル:

以下に示すチュートリアルのタイトルをクリックすると、そのチュートリアルを表示できます。

ビジネス・インテリジェンス・チュートリアル: データウェアハウス・センターの紹介
データウェアハウス・センターを使用して簡単なデータウェアハウジング・タスクを実行します。

ビジネス・インテリジェンス・チュートリアル: データウェアハウジングの上級者向けガイド
データウェアハウス・センターを使用して高度なデータウェアハウジング・タスクを実行します。

インフォメーション・カタログ・センター・チュートリアル
インフォメーション・カタログを作成および管理して、インフォメーション・カタログ・センターを使用してメタデータを配置し使用します。

Visual Explain チュートリアル
Visual Explain を使用して、パフォーマンスを向上させるために SQL ステートメントを分析し、最適化し、調整します。

DB2 トラブルシューティング情報

DB2® 製品を使用する際に役立つ、トラブルシューティングおよび問題判別に関する広範囲な情報を利用できます。

DB2 ドキュメンテーション
トラブルシューティング情報は、DB2 インフォメーション・センター、および DB2 ライブラリーに含まれる PDF 資料の中でご利用いただけます。DB2 インフォメーション・センターで、 (ブラウザー・ウィンドウの左側の) ナビゲーション・ツリーの「サポートおよびトラブルシューティング (Support and troubleshooting)」ブランチを参照すると、 DB2 トラブルシューティング・ドキュメンテーションの詳細なリストが見つかります。

DB2 Technical Support の Web サイト
現在問題が発生していて、考えられる原因とソリューションを検索したい場合は、 DB2 Technical Support の Web サイトを参照してください。Technical Support サイトには、最新の DB2 出版物、TechNotes、プログラム診断依頼書 (APAR)、フィックスパック、DB2 内部エラー・コードの最新リスト、その他のリソースが用意されています。この知識ベースを活用して、問題に対する有効なソリューションを探し出すことができます。

DB2 Technical Support の Web サイト (http://www.ibm.com/software/data/db2/udb/winos2unix/support) にアクセスしてください。

DB2 Problem Determination Tutorial Series
DB2 製品で作業中に直面するかもしれない問題を素早く識別し、解決する方法に関する情報を見つけるには、 DB2 Problem Determination Tutorial Series の Web サイトを参照してください。あるチュートリアルでは、使用可能な DB2 問題判別機能およびツールを紹介し、それらをいつ使用すべきかを判断する助けを与えます。別のチュートリアルは、『データベース・エ

ンジン問題判別 (Database Engine Problem Determination)』、『パフォーマンス問題判別 (Performance Problem Determination)』、『アプリケーション問題判別 (Application Problem Determination)』などの関連トピックを扱っています。

DB2 Technical Support (http://www.ibm.com/software/data/support/pdm/db2tutorials.html) には、 DB2 問題判別チュートリアルがすべて揃っています。

関連概念:

- 812 ページの『DB2 インフォメーション・センター』
- 「*問題判別の手引き*」の中の『Introduction to Problem Determination - DB2 テクニカル・サポートのチュートリアル』

アクセス支援

アクセス支援機能は、身体に障害のある (身体動作が制限されている、視力が弱いなど) ユーザーがソフトウェア製品を十分活用できるように支援します。 DB2® バージョン 8 製品に備わっている主なアクセス支援機能は、以下のとおりです。

- すべての DB2 機能は、マウスの代わりにキーボードを使ってナビゲーションできます。詳細については、『キーボードによる入力およびナビゲーション』を参照してください。
- DB2 ユーザー・インターフェースのフォント・サイズおよび色をカスタマイズすることができます。詳細については、 838 ページの『アクセスしやすい表示』を参照してください。
- DB2 製品は、Java™ Accessibility API を使用するアクセス支援アプリケーションをサポートします。詳細については、 838 ページの『支援テクノロジーとの互換性』を参照してください。
- DB2 資料は、アクセスしやすい形式で提供されています。詳細については、 838 ページの『アクセスしやすい資料』を参照してください。

キーボードによる入力およびナビゲーション

キーボード入力

キーボードだけを使用して DB2 ツールを操作できます。マウスを使って実行できる操作は、キーまたはキーの組み合わせによっても実行できます。標準のオペレーティング・システム・キー・ストロークを使用して、標準のオペレーティング・システム操作を実行できます。

キーまたはキーの組み合わせによって操作を実行する方法について、詳しくは キーボード・ショートカットおよびアクセラレーター: Common GUI help を参照してください。

キーボード・ナビゲーション

キーまたはキーの組み合わせを使用して、 DB2 ツールのユーザー・インターフェースをナビゲートできます。

キーまたはキーの組み合わせによって DB2 ツールをナビゲートする方法の詳細については、キーボード・ショートカットおよびアクセラレーター: Common GUI help を参照してください。

キーボード・フォーカス

UNIX® オペレーティング・システムでは、アクティブ・ウィンドウの中で、キー・ストロークによって操作できる領域が強調表示されます。

アクセスしやすい表示

DB2 ツールには、視力の弱いユーザー、その他の視力障害をもつユーザーのためにアクセシビリティーを向上させる機能が備わっています。これらのアクセシビリティー拡張機能には、フォント・プロパティーのカスタマイズを可能にする機能も含まれています。

フォントの設定

「ツール設定」ノートブックを使用して、メニューおよびダイアログ・ウィンドウに使用されるテキストの色、サイズ、およびフォントを選択できます。

フォント設定に関する詳細情報は、メニューおよびテキストのフォントを変更する: Common GUI help を参照してください。

色に依存しない

本製品のすべての機能を使用するために、ユーザーは必ずしも色を識別する必要はありません。

支援テクノロジーとの互換性

DB2 ツールのインターフェースは、Java Accessibility API をサポートします。これによって、スクリーン・リーダーその他の支援テクノロジーを DB2 製品で利用できるようになります。

アクセスしやすい資料

DB2 形式は、ほとんどの Web ブラウザーで表示可能な XHTML 1.0 形式で提供されています。 XHTML により、ご使用のブラウザーに設定されている表示設定に従って資料を表示できます。さらに、スクリーン・リーダーや他の支援テクノロジーを使用することもできます。

シンタックス・ダイアグラムはドット 10 進形式で提供されます。この形式は、スクリーン・リーダーを使用してオンライン・ドキュメンテーションにアクセスする場合にのみ使用できます。

関連概念:

- 839 ページの『ドット 10 進シンタックス・ダイアグラム』

関連タスク:

- 『キーボード・ショートカットおよびアクセラレーター: Common GUI help』
- 『メニューおよびテキストのフォントを変更する: Common GUI help』

ドット 10 進シンタックス・ダイアグラム

スクリーン・リーダーを使用してインフォメーション・センターを利用するユーザーのために、シンタックス・ダイアグラムがドット 10 進形式で提供されます。

ドット 10 進形式では、各シンタックス・エレメントは別々の行に書き込まれます。複数のシンタックス・エレメントが常に同時に存在する (または常に同時に不在の) 場合、単一のコンパウンド・シンタックス・エレメントとみなせるので同一行に表示できます。

各行は、ドット 10 進数で開始します。たとえば、3 または 3.1 ないしは 3.1.1 です。こうした数を適切に聞き取るには、スクリーン・リーダーが句読点を読み取るように設定されていることを確認してください。同じドット 10 進数を持つすべてのシンタックス・エレメント (たとえば、 3.1 という数値を持つすべてのシンタックス・エレメント) は、相互に排他的な代替エレメントです。 3.1 USERID および 3.1 SYSTEMID という行を聞き取る場合、シンタックスには両方ではなく USERID または SYSTEMID のどちらかが含まれることが分かります。

ドット 10 進レベルは、ネストのレベルを表示します。たとえば、ドット 10 進数 3 のシンタックス・エレメントの後に、一連のドット 10 進数 3.1 のシンタックス・エレメントが続きます。 3.1 の番号が付されたシンタックス・エレメントすべては、番号 3 の付されたシンタックス・エレメントに従属します。

シンタックス・エレメントに関する情報を追加するため、ドット 10 進数の次に特定のワードおよびシンボルが使用されます。 時折、こうしたワードおよびシンボルはエレメントの最初に表示される場合もあります。 簡単に識別するため、ワードやシンボルがシンタックス・エレメントの一部である場合には、円記号 (¥) 文字が先頭に付きます。 * シンボルはドット 10 進数の次に使用でき、シンタックス・エレメントが反復することを示します。 たとえば、ドット 10 進数 3 のシンタックス・エレメント *FILE は、3 ¥* FILE という形式になります。 3* FILE という形式は、シンタックス・エレメント FILE が反復されることを示します。 3* ¥* FILE という形式は、シンタックス・エレメント * FILE が反復されることを示します。

シンタックス・エレメントのストリングを分離するのに使用されるコンマなどの文字は、シンタックス内の分離する項目の直前に表示されます。 こうした文字は、それぞれの項目と同一行に表示するか、同じドット 10 進数を持つ関連する項目のある別の行に表示できます。 またその行には、シンタックス・エレメントに関する情報を提供する別のシンボルを表示することも可能です。 たとえば、複数の LASTRUN および DELETE シンタックス・エレメントを使用している場合には、5.1*、5.1 LASTRUN、および 5.1 DELETE という行は、エレメントをコンマで区切る必要があります。区切り文字が指定されないと、各シンタックス・エレメントを区切るのにブランクが使用されると想定されます。

シンタックス・エレメントの前に % シンボルが付く場合、他の箇所で定義されている参照であることを示します。 % シンボルの後のストリングは、リテラルではなくシンタックス・フラグメントの名前です。 たとえば、2.1 %OP1 という行は別のシンタックス・フラグメント OP1 を参照すべきことを意味します。

以下のワードおよびシンボルが、ドット 10 進数の次に使用されます。

- ? は、オプションのシンタックス・エレメントであることを表します。? シンボルが後に続くドット 10 進数は、対応するドット 10 進数のシンタックス・エレメント、および任意の従属のシンタックス・エレメントがオプションであることを示します。ドット 10 進数の付いたシンタックス・エレメントが 1 つしかない場合、 ? シンボルはそのシンタックス・エレメントと同じ行に表示されます (たとえば、5? NOTIFY)。ドット 10 進数の付いたシンタックス・エレメントが複数ある場合、 ? シンボルだけで行に表示され、その後にオプションのシンタックス・エレメントが続きます。たとえば、「5 ?、5 NOTIFY、および 5 UPDATE」という行を聞き取る場合、シンタックス・エレメント NOTIFY および UPDATE がオプションである、つまりそのいずれかを選択でき、どちらも選択しないこともできることが分かります。 ? シンボルは、線路型ダイアグラムのバイパス線に相当します。
- ! は、デフォルトのシンタックス・エレメントであることを表します。! シンボルおよびシンタックス・エレメントが後に続くドット 10 進数は、そのシンタックス・エレメントが、同じドット 10 進数を共用するシンタックス・エレメントすべてのデフォルト・オプションであることを示します。同じドット 10 進数を共用するシンタックス・エレメントのうち 1 つだけに、 ! シンボルを指定できます。たとえば、「2? FILE、2.1! (KEEP)、および 2.1 (DELETE)」という行を聞き取る場合、 FILE キーワードのデフォルト・オプションは (KEEP) になります。この例では、FILE キーワードを含めてもオプションを指定しない場合には、デフォルト・オプション KEEP が適用されます。デフォルト・オプションは、次に高位のドット 10 進数にも適用されます。 この例の場合、FILE キーワードが省略されると、デフォルトの FILE(KEEP) が使用されます。しかし、「2? FILE、2.1、2.1.1! (KEEP)、および 2.1.1 (DELETE)」という行を聞き取る場合、デフォルト・オプション KEEP は次に高位のドット 10 進数 2.1 (関連キーワードを持っていない) にのみ適用され、 2? FILE には適用されません。キーワード FILE が省略されると、どれも使用されません。
- * は、0 回以上反復できるシンタックス・エレメントを示します。 * シンボルが後に続くドット 10 進数は、このシンタックス・エレメントが 0 回以上使用できること、つまりオプションであり、なおかつ反復できることを表します。たとえば、5.1* データ域という行を聞き取る場合、 1 つまたは複数のデータ域を含めるか、またはデータ域を全く含めないことが可能です。「3*、3 HOST、および 3 STATE」という行を聞き取る場合、 HOST、STATE をどちらか一方または両方同時に含めるか、どちらも含めないことができます。

  注:

  1. ドット 10 進数の後にアスタリスク (*) が付き、ドット 10 進数の付いた項目が 1 つしかない場合には、同じ項目を複数回反復できます。
  2. ドット 10 進数の後にアスタリスクが付き、ドット 10 進数の付いた項目が複数ある場合、リストから複数の項目を使用できますが、各項目を複数回使用することはできません。 前述の例では、HOST STATE と書くことはできますが、HOST HOST とは書けません。
  3. * シンボルは、線路型シンタックス・ダイアグラムのループバック線に相当します。
- + は、1 回以上含める必要のあるシンタックス・エレメントであることを示します。 + シンボルが後に続くドット 10 進数は、このシンタックス・エレメントを 1 回以上含める必要があること、つまり少なくとも 1 回は含める必要があり、反

復できることを表します。たとえば、「6.1+ データ域」という行を聞き取る場合、データ域を少なくとも 1 回は含めなければなりません。「2+、2 HOST、および 2 STATE」という行を聞き取る場合には、 HOST、STATE、またはその両方を含める必要があります。 * シンボルと同様に、+ シンボルは、ドット 10 進数の付いた項目が 1 つしかない場合に限り、その特定の項目のみを反復できます。 * シンボルと同様、 + シンボルは線路型シンタックス・ダイアグラムのループバック線に相当します。

関連概念:

- 837 ページの『アクセス支援』

関連タスク:

- 『キーボード・ショートカットおよびアクセラレーター: Common GUI help』

関連資料:

- x ページの『構文図の見方』

DB2 Universal Database 製品の共通基準認証

DB2 Universal Database は、 Common Criteria の評価検定レベル 4 (EAL4) で認証の評価を受けています。 Common Criteria の詳細については、以下の Common Criteria の Web サイトを参照してください。 http://niap.nist.gov/cc-scheme/

付録 Q. 特記事項

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒106-0032
東京都港区六本木 3-2-31
IBM World Trade Asia Corporation
Licensing

以下の保証は、国または地域の法律に沿わない場合は、適用されません。 IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム(本プログラムを含む)との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Canada Limited
Office of the Lab Director
8200 Warden Avenue
Markham, Ontario
L6G 1C7
CANADA

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、 IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。 IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生した創作物には、次のように、著作権表示を入れていただく必要があります。

商標

以下は、IBM Corporation の商標です。

ACF/VTAM
AISPO
AIX
AIXwindows
AnyNet
APPN
AS/400
BookManager
C Set++
C/370
CICS
Database 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extenders
DB2 OLAP Server
DB2 Information Integrator
DB2 Query Patroller
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eServer
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
iSeries
LAN Distance
MVS
MVS/ESA
MVS/XA
Net.Data
NetView
OS/390
OS/400
PowerPC
pSeries
QBIC
QMF
RACF
RISC System/6000
RS/6000
S/370
SP
SQL/400
SQL/DS
System/370
System/390
SystemView
Tivoli
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WebSphere
WIN-OS/2
z/OS
zSeries

以下は、それぞれ各社の商標または登録商標です。

Microsoft、Windows、Windows NT および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

Pentium は、Intel Corporation の米国およびその他の国における商標です。

Java およびすべての Java 関連の商標およびロゴは、Sun Microsystems, Inc. の米国およびその他の国における商標または登録商標です。

UNIX は、The Open Group の米国およびその他の国における登録商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

[カ行]

[サ行]

[タ行]

[ナ行]

[ハ行]

[マ行]

[ヤ行]

[ラ行]

[ワ行]

[数字]

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

[特殊文字]

IBM と連絡をとる

技術上の問題がある場合は、お客様サポートにご連絡ください。

製品情報

DB2 Universal Database 製品に関する情報は、
http://www.ibm.com/software/data/db2/udb から入手できます。

このサイトには、技術ライブラリー、資料の注文方法、製品のダウンロード、ニュースグループ、フィックスパック、ニュース、および Web リソースへのリンクに関する最新情報が掲載されています。

米国以外の国で IBM に連絡する方法については、 IBM Worldwide ページ (www.ibm.com/planetwide) にアクセスしてください。

Printed in Japan

SC88-9155-01

日本アイ・ビー・エム株式会社
〒106-8711　東京都港区六本木3-2-12