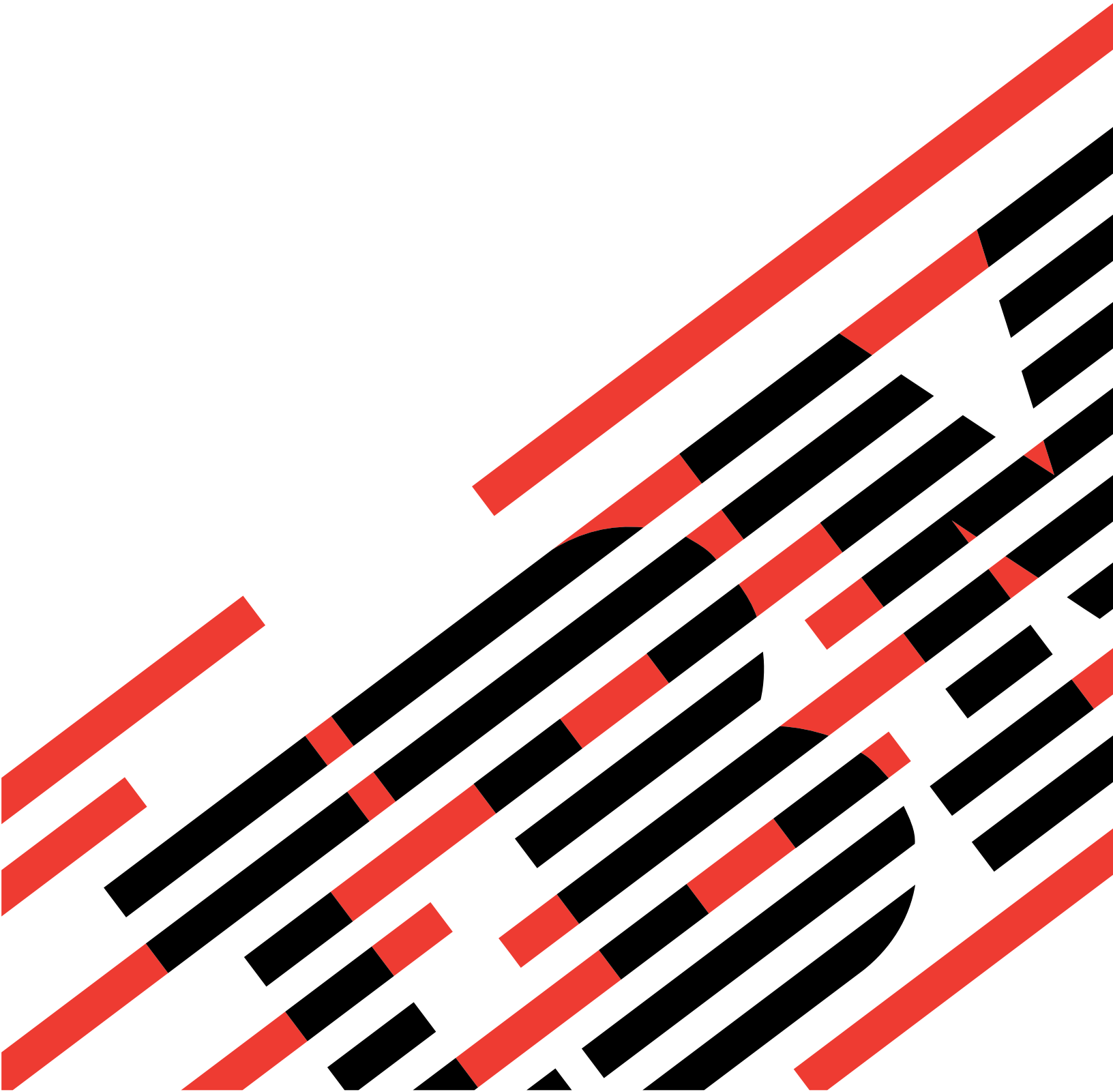
IBM

eserver

iSeries

DB2 Universal Database for iSeries SQL 解説書

バージョン 5



IBM

@server

iSeries

DB2 Universal Database for iSeries SQL 解説書

バージョン 5

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)

原　典： RBAF–Z000–03
iSeries
DB2 Universal Database for iSeries SQL Reference
Version 5

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 2002.8

目次

本書 (DB2 UDB for iSeries SQL 解説書) について

本書は、DB2 Query Manager and SQL Development Kit によってサポートされている構造化照会言語 (SQL) について説明しています。本書には、システムの管理、データベースの管理、アプリケーション・プログラミング、および操作のタスクに関する参照情報が記載されています。また、システムで使用する SQL ステートメントそれぞれについて、その構文、使用上の注意、キーワード、および例を示しています。

詳細については、以下のセクションを参照してください。

- 『標準への準拠』
- 『本書の対象読者』
- xiv ページの『SQL ステートメントの例に関する前提事項』
- xv ページの『構文図の見方』
- xvii ページの『混合データの値の記述規則』
- xvii ページの『SQL アクセシビリティー』
- xviii ページの『本書での変更箇所』

標準への準拠

DB2 UDB for iSeries のバージョン 5 リリース 2 は、以下の IBM およびその他の SQL 標準に準拠しています。

- ISO (国際標準化機構) 9075: 1992、データベース言語 SQL - 項目レベル
- ISO (国際標準化機構) 9075-4: 1996、データベース言語 SQL - 第 4 部: 永続的保管モジュール (SQL/PSM)
- ISO (国際標準化機構) 9075: 1999、データベース言語 SQL - コア
- ANSI (米国規格協会) X3.135-1992、データベース言語 SQL - 項目レベル
- ANSI (米国規格協会) X3.135-4: 1996、データベース言語 SQL - 第 4 部: 永続的保管モジュール (SQL/PSM)
- ANSI (米国規格協会) X3.135-1999、データベース言語 SQL - コア
- *IBM SQL Reference Version 2*, SC26-8416.

標準を厳守するために、標準オプションを使用するようにしてください。詳しくは、770 ページの『SET OPTION』の SQLCURRULE、および SQL プリコンパイラー・コマンドの解説を参照してください。

本書の対象読者

本書は、SQL を使用して iSeries のデータベースにアクセスするアプリケーションを作成する必要があるプログラマーの方々を対象としています。

本書では、SQL プログラミング 概念で説明されているシステム管理、データベース管理、または iSeries のアプリケーション・プログラミングに関する知識を持っているとともに、以下の事項についてもある程度の知識を持っていることを前提としています。

- COBOL for iSeries
- ILE C コンパイラー
- ILE C++ コンパイラー
- ILE COBOL コンパイラー
- Toolbox for Java または Developer Kit for Java
- ILE RPG コンパイラー
- iSeries PL/I
- REXX
- RPG III (RPG for iSeries の一部)
- 構造化照会言語 (SQL)

本書において、RPG および COBOL という用語は、一般の RPG または COBOL 言語を指しています。 COBOL for iSeries、ILE COBOL for iSeries、RPG for iSeries、または RPG III (RPG for iSeries の一部) は、特定の要素が互いに異なるプロダクトの場合を指しています。

本書は解説用というよりも、むしろ参照用の資料です。したがって、読者がすでに SQL プログラミングをある程度理解しているものとして説明を進めています。また、本書では、iSeries 用に限定したアプリケーションの作成を想定しています。

SQL ステートメント、ステートメントの構文、およびパラメーターの使用法の詳細を知りたい場合は、SQL プログラミング 概念を参照してください。

他の IBM 環境へ移植可能なアプリケーションを計画している場合には、本書のほかに、該当の環境に関する資料 (たとえば、*IBM SQL Reference Version 2* (SC26-8416) など) を参照することが必要になります。

詳しくは、以下のセクションを参照してください。

- 『SQL ステートメントの例に関する前提事項』
- xv ページの『構文図の見方』

SQL ステートメントの例に関する前提事項

本書で示している SQL ステートメントの例は、SQL プログラミング 概念の付録 A に示されているサンプル表に基づいており、以下を想定しています。

- 例は対話式 SQL 環境で示すか、または COBOL で書かれています。 EXEC SQL および EXEC SQL および END-EXEC は、COBOL プログラムにおける SQL ステートメントの範囲を定めるために使用されます。 COBOL プログラムにおける SQL ステートメントの使用法については、DB2 UDB サーバー (AS/400 版) ホスト言語での SQL プログラミングで説明しています。
- 各 SQL ステートメントの例は、そのステートメントの各文節ごとに行を変えて、数行にまたがって示されています。
- SQL のキーワードは、太字で示されています。

- 例で使用されている表名は、SQL プログラミング概念の付録 A に示されているサンプル表で、スキーマ CORPDATA で使用されます。この付録に示されていない表名は、ユーザーが作成するスキーマを作成する必要があります。 ユーザー独自のスキーマで次の SQL ステートメントを発行することにより、1 組のサンプル表を作成できます。

  ```
  CALL QSYS.CREATE_SQL_SAMPLE ('ユーザー作成のスキーマ名')
  ```

- 計算対象の列は、括弧、() で囲んでいます。
- SQL の命名規則を使用しています。
- (COBOL では、デフォルトではありませんが)、プリコンパイラー・オプションの APOST と APOSTSQL を前提としています。 SQL およびホスト言語のステートメント内の文字ストリング定数は、アポストロフィ (') で区切られています。
- *HEX のソート順序を使用します。

これらの前提事項と異なる例では、必ずその旨を明記しています。

『コードに関する特記事項』も参照してください。

コードに関する特記事項

本書には、プログラミングの例が含まれています。

IBM は、お客様に、すべてのプログラム・コードのサンプルを使用することができる非独占的な使用権を許諾します。お客様は、このサンプル・コードから、お客様独自の特別のニーズに合わせた類似のプログラムを作成することができます。

すべてのサンプル・コードは、例として示す目的でのみ、IBM により提供されます。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。

ここに含まれるすべてのプログラムは、現存するままの状態で提供され、いかなる保証条件も適用されません。 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の保証の適用も一切ありません。

構文図の見方

本書全体を通じて、構文を記述するときには、以下のように定義されている構文図を使用します。

- 構文図は、直線で示される経路にしたがって、左から右、上から下の方向に読んでください。

  ▸▸── の記号は、ステートメントの始まりを示します。

  ──▸ の記号は、ステートメントの構文が次の行に継続することを示します。

  ▸── の記号は、ステートメントが前の行から継続していることを示します。

  ──▸◂ の記号は、ステートメントの終わりを示します。

  部分構文を示す図は、▸── の記号で始まり、──▸ の記号で終わります。
- 必須項目は、次のように水平方向の線 (メインパス) 上に示します。

  ▸▸─*必須項目*──▸◂

- 任意指定項目は、次のようにメインパスの下に示します。

メインパスより上に示される任意指定項目は、単に読みやすさのために使用される項目で、ステートメントの実行には影響を与えません。

- 複数の項目からユーザーが選択できる場合は、それらの項目を縦方向に並べて示します。

  項目の中から必ずどれか 1 つを*選択しなければならない* 場合は、縦方向に並んでいる選択項目のうちの 1 つをメインパス上に示します。

  項目を選択しても選択しなくてもよい場合は、縦方向に並んでいる選択項目をすべてメインパスの下に示します。

  項目の中にデフォルトの選択項目がある場合は、その選択項目をメインパスの上に示し、残りの選択項目をメインパスの下に示します。

- メインパスの上を通って左に戻る矢印は、反復可能な項目を示します。

  反復可能を示す矢印にコンマが入っている場合は、反復可能な項目を指定するときに、項目相互間をコンマで区切る必要があります。

  縦方向に並べられた項目群の上に反復可能を示す矢印がある場合は、その項目群にある項目を反復して指定できることを示します。

- キーワードは大文字で示されます (例: FROM)。キーワードは、構文図に示されているとおりのつづりで正確に入力する必要があります。変数は小文字で現れます (たとえば、*列名*)。これらはユーザーが指定する名前または値を表しています。
- 構文図に句読記号、括弧、算術演算子、または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しなければなりません。
- 構文図には、優先キーワードまたは標準キーワードのみが含まれています。標準キーワードに加えて、標準外の同義語もサポートされている場合、そのような標準外同義語の説明は、構文図の中でなく、**注**のセクションで扱っています。最大の移植性を確保するためには、優先キーワードまたは標準キーワードのみを使用してください。

混合データの値の記述規則

混合データの値を例の中で示す場合は、以下のような規則が適用されています。

規則	意味
S_O	EBCDIC シフトアウト制御文字 (X'0E') を表す
S_I	EBCDIC シフトイン制御文字 (X'0F') を表す
SBCS ストリング	ゼロまたは複数の 1 バイト文字のストリングを表す
DBCS ストリング	ゼロまたは複数の 2 バイト文字のストリングを表す
'	DBCS のアポストロフィ (EBCDIC X'427D') を表す
G	DBCS の G (EBCDIC X'42C7') を表す

SQL アクセシビリティー

IBM では、身体に障害のある方々にとってアクセスしやすいインターフェースおよびドキュメンテーションを提供することをコミットしています。 IBM のアクセシビリティー・サポートに関する一般情報については、http://www.ibm.com/able  の Accessibility Center を参照してください。

SQL アクセシビリティー・サポートは、次の 2 つの主要なカテゴリーに分かれています。

- iSeries ナビゲーターは、iSeries および DB2 UDB へのグラフィカル・ユーザー・インターフェース (GUI) です。 Windows のグラフィカル・ユーザー・インターフェースでサポートされるアクセシビリティー機能については、Windows のヘルプの索引から「アクセシビリティー」を参照してください。
- オンライン・ドキュメンテーション、オンライン・ヘルプ、およびプロンプト形式の SQL インターフェースには、Windows リーダー・プログラム (IBM ホームページ・リーダーなど) を使用してアクセスすることができます。 IBM ホームページ・リーダーおよびその他のツールの詳細については、

  Accessibility Center  を参照してください。

IBM ホームページ・リーダーを使用すると、本書に収めてあるすべての本文、SQL Information Center に収めたすべての記事、およびすべての SQL メッセージにアクセスすることができます。ただし、SQL の構文図は、性質が複雑なため、リーダーではスキップされます。使い勝手に応じて選択できる方法が、このほかにも 2 つあります。

- 対話式 SQL および Query Manager

  対話式 SQL および Query Manager は、SQL ステートメントに関するプロンプトを提供する従来型のファイル・インターフェースです。これらの機能は、DB2

UDB Query Manager および SQL 開発キットに組み込まれているものです。対話式 SQL および Query Manager についての詳細は、「SQL プログラミング 概念」および 「Query Manager ご使用の手引き」 を参照してください。

- SQL 支援

  SQL 支援は、SQL ステートメントへの指示インターフェースを提供するグラフィカル・ユーザー・インターフェースです。これは iSeries ナビゲーターの中の一機能です。詳細については、iSeries ナビゲーターのオンライン・ヘルプおよび Information Center を参照してください。

本書での変更箇所

本書で説明している主要な新機能には、以下のものが含まれます。

- ROWID データ・タイプおよび ROWID スカラー関数
- IDENTITY 列属性
- CREATE TABLE AS (副選択)
- DECLARE GLOBAL TEMPORARY 表
- ユーザー定義の表関数
- COMMIT ON RETURN プロシージャー
- ビューの UNION
- スカラー副選択の機能強化
- SET TRANSACTION の READ ONLY および READ WRITE
- SQL プロシージャー、SQL 関数、および SQL トリガーの中の ITERATE ステートメントとネストされた複合ステートメント
- 派生表および共通表式における全選択
- ラベル付き期間内のパラメーター・マーカー
- 保管ポイント
- SET SCHEMA および SET SQLID
- HOLD LOCATOR
- 選択リストに必須ではない ORDER BY 式
- 派生表および共通表式における ORDER BY および FETCH FIRST n ROWS ONLY
- SQL ステートメントの長さが 64K に拡大
- 区切り列名 ID の長さの拡大
- SUBSTRING の機能強化
- VARCHAR 連結機能の強化
- SQL プロシージャー、SQL 関数、および SQL トリガーのオリジナル・ソース・ステートメントのデバッグ
- iSeries の複数のリレーショナル・データベース
- 標準、ODBC、および JDBC のカタログ・ビュー
- C の派生変数

第 1 章 概念

本書では、以下の概念について説明します。

リレーショナル・データベース

リレーショナル・データベースは、一組の表と見なすことができ、データの関係モデルにしたがって扱うことができるデータベースです。 リレーショナル・データベースには、データの保管、アクセス、および管理に使用される一組のオブジェクトが含まれます。 このようなオブジェクトには、表、ビュー、索引、別名、特殊タイプ、関数、プロシージャー、およびパッケージが含まれます。

ユーザーが iSeries システムからアクセスできるリレーショナル・データベースには、以下の 3 つのタイプがあります。

システム・リレーショナル・データベース

どのような iSeries システムにも、デフォルトのリレーショナル・データベースが 1 つあります。システム・リレーショナル・データベースは、常に

iSeries システムにとってローカルのデータベースです。このデータベースは、iSeries に接続しているディスクに存在しているデータベース・オブジェクトのうち、独立補助記憶域プールに保管されているものを除くすべてのオブジェクトから成っています。独立補助記憶域プールについての詳細は、iSeries Information Center のシステム管理カテゴリーを参照してください。

デフォルトでは、システム・リレーショナル・データベースの名前は iSeries システム名と同じです。ただし、ADDRDBDIRE (RDB ディレクトリー項目追加) コマンドまたは iSeries ナビゲーターを使用して、別の名前を割り当てることもできます。

ユーザー・リレーショナル・データベース

ユーザーは、システム上に独立補助記憶域プールを構成することにより、iSeries システム上に追加のリレーショナル・データベースを作成することができます。個々の 1 次独立補助記憶域プールが、それぞれ 1 つのリレーショナル・データベースとなります。このデータベースには、独立補助記憶域プール・ディスク上にあるすべてのデータベース・オブジェクトが含まれます。さらに、独立補助記憶域プールが接続している iSeries システムのシステム・リレーショナル・データベース内のすべてのデータベース・オブジェクトも、論理的にユーザー・リレーショナル・データベースに含まれます。したがって、1 つのユーザー・リレーショナル・データベース内に作成するスキーマの名前は、そのユーザー・リレーショナル・データベースまたはそれに関連したシステム・リレーショナル・データベースの中にすでに存在する名前であってはなりません。

システム・リレーショナル・データベース内のオブジェクトは、論理的にはユーザー・リレーショナル・データベースに含まれていますが、以下に示すように、システム・リレーショナル・データベースとユーザー・リレーショナル・データベースの間で、オブジェクト相互間に依存関係を持たせることができない場合が幾つかあります。

- ビューを作成するときは、そのビューが参照する表、ビュー、または関数と同じリレーショナル・データベース内にあるスキーマの中に作成する必要があります。
- 索引を作成するときは、その索引が参照する表と同じリレーショナル・データベース内にあるスキーマの中に作成する必要があります。
- トリガーまたは制約を作成するときは、その基礎表と同じリレーショナル・データベース内にあるスキーマの中に作成する必要があります。
- 1 つの参照制約の中の親表および従属表は、両方が同じリレーショナル・データベースの中にあることが必要です。
- 表を作成するときは、その表が参照する特殊タイプと同じリレーショナル・データベース内にあるスキーマの中に作成する必要があります。

システム・リレーショナル・データベースとユーザー・リレーショナル・データベースの間の、その他のオブジェクト相互依存関係は使用することができます。例えば、ユーザー・リレーショナル・データベース内のスキーマの中にあるプロシージャーが、システム・リレーショナル・データベース内のオブジェクトを参照することはできます。しかし、相手方のリレーショナル・データベースが使用可能になっていないと、このようなオブジェクトに

対する操作は失敗することがあります。例えば、ユーザー・リレーショナル・データベースをオフに変更し、さらに別のシステムに対してオンに変更した場合などが、これに相当します。

ユーザー・リレーショナル・データベースは、独立補助記憶域プールがオンにされている間は、iSeries システムにとってローカルの関係にあります。独立補助記憶域プールは、1 つの iSeries システムではオフに変更し、別の iSeries システムではオンに変更することができます。したがって、同じユーザー・リレーショナル・データベースが、特定の iSeries システムにとってある時点ではローカルになり、別の時点ではリモートになることがあります。独立補助記憶域プールについての詳細は、iSeries Information Center の システム管理 カテゴリーを参照してください。

デフォルトでは、ユーザー・リレーショナル・データベースの名前は独立補助記憶域プール名と同じです。ただし、ADDRDBDIRE (RDB ディレクトリー項目追加) コマンドまたは iSeries ナビゲーターを使用して、別の名前を割り当てることもできます。

リモート・リレーショナル・データベース

他の iSeries システムおよび iSeries 以外のシステム上にあるリレーショナル・データベースには、リモート・アクセスすることができます。この種のリレーショナル・データベースは、ADDRDBDIRE (RDB ディレクトリー項目追加) コマンドまたは iSeries ナビゲーターを使用して登録する必要があります。

データベース・マネージャーとは、リレーショナル・データベースを管理する iSeries ライセンス内部コードおよび DB2 UDB for iSeries のコード部分を指すのに使用する総称的な名前です。

構造化照会言語 (SQL)

構造化照会言語 (SQL) は、リレーショナル・データベース内のデータを定義および操作するための標準化言語です。データの関係モデルの概念にしたがうと、データベースは表の集合であると考えられます。また、表内の値によって関連が表現されるとともに、1 つまたは複数の基礎表から得られる結果表を指定することによってデータが検索されます。

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・マネージャーには、結果表の指定を、データ検索を最適化する一連の内部命令に変換する機能があります。SQL ステートメントを*準備* するときに、この変換が行われます。この変換を行うことを、*バインドする* とも言います。

SQL ステートメントを実行するには、あらかじめ実行可能 SQL ステートメントがすべて準備されている必要があります。準備の結果は、ステートメントの実行可能形式、つまり実行形式です。SQL は、SQL ステートメントを準備する方式とステートメントの実行形式の持続期間によって、静的 SQL と動的 SQL に区別されます。

静的 SQL

静的 SQL ステートメントのソース (原始) 形式は、ホスト言語 (COBOL など) で書かれたアプリケーション・プログラム内部に組み込まれます。このステートメン

トは、アプリケーション・プログラムの実行前に準備されます。また、このステートメントの実行形式は、アプリケーション・プログラムが実行された後も残ります。

静的 SQL ステートメントが入っているソース (原始) プログラムは、コンパイルする前に、SQL プリコンパイラーで処理する必要があります。プリコンパイラーは、SQL ステートメントの構文をチェックしてホスト言語のコメントに変換し、さらにデータベース・マネージャーを呼び出すホスト言語のステートメントを生成します。

SQL アプリケーション・プログラムの準備には、プリコンパイル、その静的 SQL ステートメントの準備、および変更されたソース・プログラムのコンパイルが含まれます。

動的 SQL

動的 SQL ステートメントは、SQL アプリケーションの実行時に準備されます。このステートメントの実行形式は、最後の SQL プログラムが呼び出しスタックから出るまで存続します。動的 SQL ステートメントのソース形式は、静的 SQL ステートメントの PREPARE または EXECUTE IMMEDIATE を使用して、プログラムからデータベース・マネージャーに文字ストリングとして渡されます。

REXX アプリケーションに組み込まれた SQL ステートメントは動的 SQL です。対話式 SQL 機能にサブミットされる SQL ステートメントも、動的 SQL ステートメントであるといえます。

拡張動的 SQL

拡張動的 SQL ステートメントは、完全に静的でもなく、完全に動的でもありません。QSQPRCED API は、拡張動的 SQL 機能を提供します。動的 SQL 機能と同様に、この API を使用して、ステートメントを準備し、記述し、実行することができます。動的 SQL とは異なり、この API によってパッケージ中に準備された SQL ステートメントは、そのパッケージまたはステートメントが明示的に削除されるまで、存続します。詳細については、iSeries Information Center の**プログラミング**・カテゴリーの OS/400 API 情報を参照してください。

対話式 SQL

対話式 SQL 機能は、すべてのデータベース・マネージャーに関連付けられています。あらゆる対話式 SQL 機能は、本質的には、端末からステートメントを読み取り、ステートメントを動的に準備して実行し、その結果をユーザーに表示する SQL アプリケーション・プログラムです。このような SQL ステートメントは、*対話式* に 出されるステートメントと呼ばれます。DB2 UDB for iSeries の対話式機能は、STRSQL コマンド、STRQM コマンド、または iSeries ナビゲーターの SQL スクリプト・サポートによって呼び出されます。 SQL の対話機能についての詳細は、

「SQL プログラミング 概念」および「Query Manager ご使用の手引き」  を参照してください。

SQL 呼び出しレベル・インターフェース (CLI) と Open Database Connectivity (ODBC)

DB2 呼び出しレベル・インターフェースは、動的 SQL ステートメントを処理するためにアプリケーション・プログラムに関数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。DB2 CLI により、ユーザーはどの ILE 言語を使用している場合でも、DB2 UDB for iSeries が提供するサービス・プログラムへのプロシージャー呼び出しを使用して、SQL 機能に直接アクセスできます。CLI プログラムは、Microsoft や他のベンダーから入手できる、ODBC データ・ソースへのアクセスを可能にする Open Database Connectivity (ODBC) ソフトウェア開発キットを使用してコンパイルすることもできます。組み込み SQL とは異なり、プリコンパイルの必要はありません。このインターフェースを使用して開発されたアプリケーションは、個々のデータベースごとにコンパイルせずに、さまざまなデータベースに対して実行できます。インターフェースを通して、アプリケーションは実行時にプロシージャーを呼び出し、データベースに接続し、SQL ステートメントを実行し、戻されたデータや状況に関する情報を入手します。

DB2 CLI インターフェースは、組み込み SQL では利用不能な多くの機能を備えています。これには例えば以下のようなものがあります。

- CLI が提供する関数呼び出しは、DB2 データベース管理システム・ファミリーに共通の、一貫した方法でのデータベース・システム・カタログ情報の照会、取り出しをサポートしています。これにより、データベース・サーバー特定のカタログ照会を作成する必要性が減ります。
- 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たストアード・プロシージャーは、これらのプログラムに対して結果セットを戻すことができます。

使用できる機能とその構文の詳細については、「DB2 UDB for iSeries SQL 呼び出しレベル・インターフェース」を参照してください。

Java Database Connectivity (JDBC) および Embedded SQL for Java (SQLJ) プログラム

DB2 UDB for iSeries は、標準に準拠した 2 つの Java プログラミング API、つまり、Java Database Connectivity (JDBC) と embedded SQL for Java (SQLJ) をインプリメントしています。どちらも、DB2 にアクセスする Java アプリケーションやアプレットを作成できます。

JDBC 呼び出しは、Java 固有の方式によって、DB2 CLI への呼び出しに変換されます。iSeries データベースには、2 つの JDBC ドライバーを介してアクセスできます。IBM Developer Kit for Java ドライバーまたは IBM Toolbox for Java JDBC ドライバーです。IBM Toolbox for Java JDBC ドライバーについての詳しい情報は、「IBM Toolbox for Java」を参照してください。

JDBC では、静的 SQL は使用できません。SQLJ アプリケーションは、データベースへの接続や SQL エラーの処理などの作業の基礎として JDBC を使用しますが、SQLJ ソース・ファイルに 静的 SQL ステートメントを含めることもできます。SQLJ ソース・ファイルは、まず SQLJ 変換プログラムを使って変換しないと、得られた Java ソース・コードをコンパイルできません。

JDBC および SQLJ アプリケーションの詳細については、「 Developer Kit for Java」を参照してください。

スキーマ

スキーマとは、名前付きオブジェクトの集合をいいます。スキーマは、リレーショナル・データベース内のオブジェクトを論理的にソートするものです。スキーマに含まれるオブジェクトとしては、表、ビュー、別名、関数、プロシージャー、型、パッケージなどが挙げられます。スキーマは、コレクションまたはライブラリーとも呼ばれます。

スキーマは、リレーショナル・データベース内のオブジェクトでもあります。CREATE SCHEMA ステートメントを使用して、明示的に作成します。[1]

スキーマ名 は、2 つの部分からなるオブジェクト名の高位部分として使用されます。スキーマに含まれるオブジェクトは、オブジェクトが作成されるときに、スキーマに割り当てられます。割り当てられるスキーマは、オブジェクトの名前 (スキーマ名で特別に修飾されている場合) またはデフォルトのスキーマ名 (修飾されていない場合) によって決まります。

例えば、ユーザーが C という名前のスキーマを作成するとします。

```
CREATE SCHEMA C
```

その後、次のステートメントを実行すると、スキーマ C の中に X と呼ばれる表を作成できます。

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

表

表は、ユーザー・データを保管するオブジェクトです。表は、データベース・マネージャーによって保守される論理構造です。表は、列と行から形成されます。表の各行には、固有の順序はありません。列と行が交わったところには、必ず *値* と呼ばれる特定のデータ項目があります。*列* とは、同一のタイプに属する値の集まりを指します。*行* とは、先頭から *n* 番目の値が、表の *n* 番目の列の値になるように並んでいる一連の値を指します。

基礎表 とは、CREATE TABLE ステートメントによって作成される表であり、持続的なユーザー・データを保持するのに使用されます。*結果表* とは、データベース・マネージャーが 1 つまたは複数の基礎表から選択または生成を行った列の集まりです。

基礎表は、名前を持ち、異なるシステム名を持つ場合があります。システム名は、OS/400 によって使用される名前です。SQL ステートメントで*表名* を指定する場所には、どちらの名前でも使用できます。詳細については、542 ページの『CREATE TABLE』を参照してください。

1. スキーマは CRTLIB CL コマンドを使って作成することもできますが、CREATE SCHEMA ステートメントで作成されるカタログ・ビューやジャーナルは、CRTLIB では作成されません。

基礎表の列は、名前を持ち、異なるシステム列名を持つ場合があります。システム列名は、OS/400 によって使用される名前です。SQL ステートメントで*列名* を指定する場所には、どちらの名前でも使用できます。詳細については、542 ページの『CREATE TABLE』を参照してください。

分散表 とは、そのデータがノード・グループに分配されている表を指します。*ノード・グループ* とは、複数のシステムが集まった論理的なグループを提供するオブジェクトです。*区分化*キー とは、分散表内の 1 つまたは複数の列の集まりであり、このキーを使用して行がどのシステムに属するべきかを判別します。 分散表の詳細については、「DB2 UDB for iSeries マルチ・システム」を参照してください。

宣言済み一時表 は DECLARE GLOBAL TEMPORARY TABLE ステートメントにより作成されるもので、単一のアプリケーションに代わって一時データを保留するために使用されます。この表は、そのアプリケーションがデータベースから切断されたときに、暗黙的に除去されます。

キー

キー とは、索引、固有制約、または参照制約の記述で識別されている 1 つまたは複数の列を指します。同一の列が複数のキーに含まれることもあります。複数の列から構成されるキーを複合キーと呼びます。

複合キー とは、同一の表にある列の集まりを順序付けたものです。列の順序は、表内での順序に制約されません。*値* という用語は、複合キーに関して使用した場合には、複合値のことを指します。したがって、『外部キーの値は、基本キーの値に等しくなければならない』 などの規則は、外部キーの値の各コンポーネントが、基本キーの値の対応するコンポーネントに等しくなければならないことを意味しています。

基本キーと固有キー

固有制約 は、キーの値が固有な場合にのみ有効であることを示す規則です。固有の値を持つように制約されているキーは、*固有*キー と呼ばれ、CREATE UNIQUE INDEX ステートメントを使用して定義することができます。結果の固有索引は、INSERT や UPDATE ステートメントの実行の過程でデータベース・マネージャーによって使用され、キーの固有性が確保されます。固有キーは、次のいずれかによって定義することができます。

- CREATE TABLE または ALTER TABLE ステートメントを使用して、基本キーとして定義する。1 つの表で複数の基本キーを持つことはできません。基本キーを構成する桁には NULL 値が許されないという規則を強制する、検査制約が暗黙的に追加されます。基本キーに基づく固有索引は、基本索引と呼ばれます。
- CREATE TABLE または ALTER TABLE ステートメントの UNIQUE 文節を使用して定義する。表は任意の数の UNIQUE キーを持つことができます。

参照制約の外部キーによって参照される固有キーは、親キーと呼ばれます。親キーは、基本キー、または UNIQUE キーのいずれかです。表が、参照制約において親として定義される場合、その基本キーが、デフォルトの親キーになります。

参照保全

参照保全 とは、すべての外部キーのすべての値が有効な場合のデータベースの状態を指しています。*外部キー* は、参照制約の定義の一部であるキーです。*参照制約* は、外部キーの値が以下の場合にのみ有効であることを示す規則です。

- それらが、親キーの値として現れている。または、
- 外部キーのコンポーネントにヌルのコンポーネントがある。

親キーを含む表は、参照制約の *親表* と呼ばれ、その外部キーを含む表は、その表に *従属* していると呼ばれます。

参照制約は、任意指定であり、CREATE TABLE ステートメントや ALTER TABLE ステートメントで定義することができます。参照制約は、INSERT、UPDATE、および DELETE ステートメントの実行の過程で、データベース・マネージャーによって課せられます。参照制約は、行が処理されるときに課せられる RESTRICT の削除や更新の規則の場合を除き、ステートメントの完了時に効果的に課せられます。

RESTRICT の削除や更新の規則を伴う参照制約は、常に、他の参照制約よりも前に課せられます。他の参照制約は、順序に関係のない形で課せられます。すなわち、その順序が操作の結果に影響することはありません。1 つの SQL ステートメント内で、

- 行に、CASCADE の削除規則を伴う任意の数の参照制約によって削除のマークを付けることができます。
- 行は、SET NULL または SET DEFAULT の削除規則を伴う 1 つの参照制約によってのみ更新することができます。
- ある参照制約によって更新された行には、CASCADE の削除規則を伴う他の参照制約によって削除のマークを付けることはできません。

参照保全の規則には、以下の概念や用語が関連します。

親キー 参照制約の基本キーまたは固有キー。

親行 少なくとも 1 つの従属行を持つ行。

親表 少なくとも 1 つの参照制約における親である表。表は、任意の数の参照制約で親として定義することができます。

従属表 少なくとも 1 つの参照制約において従属である表。表は、任意の数の参照制約で従属として定義することができます。従属表は、親表になることもできます。

下層表 表が、表 T の従属であるか、または表 T の従属の下層である場合、その表は、表 T の下層です。

従属行 少なくとも 1 つの親行をもつ行。

下層行 行が、行 p の従属であるか、または行 p の従属の下層である場合、その行は、行 p の下層です。

参照サイクル その集合の各表がそれ自身の下層である参照制約の集合。

自己参照行 それ自身の親である行。

自己参照表 同一の参照制約で親であると同時に従属である表。このような制約は、*自己参照制約* と呼ばれます。

参照制約の挿入規則では、外部キーの非ヌルの挿入値は、親表の親キーのいずれかの値に一致しなければなりません。複合外部キーのコンポーネントのいずれかの値がヌルである場合、その複合外部キーの値はヌルになります。

参照制約の更新規則は、その参照制約を定義する時点で指定します。選択できる項目は、NO ACTION と RESTRICT です。更新規則は、親または従属表が更新される時点で適用されます。更新規則では、外部キーの非ヌルの更新値は、親表の親キーのいずれかの値に一致しなければなりません。複合外部キーのコンポーネントのいずれかの値がヌルである場合、その複合外部キーの値はヌルになります。

参照制約の削除規則は、その参照制約を定義する時点で指定します。選択できる項目は、NO ACTION、RESTRICT、CASCADE、SET NULL または SET DEFAULT です。SET NULL は、外部キーの列にヌル値が許される列がある場合にのみ指定することができます。

参照制約の削除規則は、親表の行が削除される時点で適用されます。厳密には、この規則は、親表の行が、削除または波及削除操作の対象で (以下で説明)、しかもその行が参照制約の従属表に従属している場合に適用されます。P は親表を、D は従属表を、また p は削除あるいは波及削除操作の対象である親行を表すものとします。削除規則が、

- RESTRICT または NO ACTION の場合、エラーが生じ、行の削除は行われません。
- CASCADE の場合、削除操作は、D の p の従属行に波及します。
- SET NULL の場合、D の p の各従属行の外部キーのヌル可能な各列はヌルに設定されます。
- SET DEFAULT の場合、D の p の各従属行の外部キーの各列はそのデフォルト値に設定されます。

表が親である各参照制約は、それ自体の削除規則を持ち、適用可能なすべての削除規則が、削除操作の結果の判別に使用されます。したがって、行が RESTRICT または NO ACTION の削除規則を伴う参照制約に従属する場合、または削除が RESTRICT または NO ACTION の削除規則を伴う参照制約に従属する下層のいずれかにカスケードする場合は、その行は削除できません。

親表 P からの行の削除は、他の表を巻き込み、それらの表の行に影響を与えることがあります。

- 表 D が P に従属し、削除規則が RESTRICT または NO ACTION の場合、D はその操作に関与しますが、その操作による影響を受けません。
- D が P に従属し、削除規則が SET NULL の場合、D はその操作に関与し、D の行はその操作の過程で更新されることがあります。
- D が P に従属し、削除規則が SET DEFAULT の場合、D はその操作に関与し、D の行はその操作の過程で更新されることがあります。
- D が P に従属し、削除規則が CASCADE の場合、D はその操作に関与し、D の行はその操作の過程で削除されることがあります。

D の行が削除される場合は、P に対する削除操作が D に波及すると言われます。D が親表でもある場合は、このリストに記述したアクションは D の従属にも適用されることになります。

表が P に対する削除操作に関与することがある場合は、その表は P に対して*削除関係にある* と言います。したがって、ある表が表 P に従属しているか、または P からの削除操作のカスケード先である表に従属している場合は、その表は表 P に対して削除関係にあることになります。

検査制約

検査制約 は、表のすべての行の 1 つまたは複数の列で許される値を指定する規則です。検査制約は、任意指定であり、CREATE TABLE および ALTER TABLE の SQL ステートメントで定義することができます。検査制約の定義は、検索条件の制限付き形式です。制限の 1 つとして、表 T の検査制約の列名は T の列を識別する必要があります。

表は任意の数の検査制約を持つことができます。データベース・マネージャーは、次の場合に検査制約を強制します。

- 表に行が挿入される場合
- 表の行が更新される場合

挿入または更新されるそれぞれの行に対して、検査制約の検索条件を適用することによって、検査制約が強制されます。いずれかの行に対する検索条件の結果が FALSE であれば、エラーが生じます。

トリガー

トリガー は、指定の表に対する削除、挿入、または更新操作が行われるたびに自動的に実行される一組のアクションを定義します。そうした SQL 操作の実行時に、トリガーが起動されるといいます。[2]

この一組のアクションには、システム上で可能なほとんどすべての操作を含めることができます。許されない操作は、以下のような数少ない操作です。

- コミットまたはロールバック (同一のコミットメント定義がトリガーのアクション、およびトリガー対象のイベントで使用されている場合)
- CONNECT、SET CONNECTION、DISCONNECT、および RELEASE ステートメント

制約の詳細なリストについては、DB2 UDB for iSeries データベース・プログラミング を参照してください。

トリガーは、データ保全性の規則を適用するために、参照制約や検査制約と合わせて使用できます。トリガーを使用すると、他の表を更新する、挿入または更新される行の値を自動的に生成または変換する、あるいは DB2 の内部と外部の両方の操作を実行する関数を呼び出す、といったこともできるので、制約より強力です。例

2. ADDPFTRG CL コマンドも、読み取り操作時に起動されるトリガーを定義します。

えば、新規の値が所定の数量を超えている場合、トリガーでは、列の更新を防ぐ代わりに、有効な値で置き換え、管理者に無効な更新について通知することができます。

トリガーは、異なる状態のデータが含まれる過渡的ビジネス規則 (例えば、給与は 10 % までしか増やせない) を定義し、適用するのに便利なメカニズムです。このような制限は、増加前と増加後の給与の値を比較することが必要です。1 つの状態のデータしか含まれていない規則の場合は、参照制約や検査制約の使用を考えてください。

トリガーを使用すると、ビジネス規則を適用するのに必要なアプリケーション論理をデータベース内に移動することができるので、アプリケーションの開発が迅速になり、保守も容易になります。データベース内の論理 (例えば、前述のような、表の給与列の増加に関する制限) を使用して、DB2 はアプリケーションが給与列に加える変更の妥当性を検査します。また、論理が変更されても、アプリケーション・プログラムを変更する必要がありません。

トリガーはオプションであり、CREATE TRIGGER ステートメントまたは ADDPFTRG (物理ファイル・トリガーの追加) の CL コマンドを使用して定義します。トリガーは、DROP TRIGGER ステートメントまたは RMVPFTRG (物理ファイル・トリガーの除去) の CL コマンドを使用して除去できます。トリガーの作成について詳しくは、CREATE TRIGGER ステートメントの項を参照してください。トリガー全般についての詳しい情報は、575 ページの『CREATE TRIGGER』 ステートメントの項の説明、または「SQL プログラミング 概念」および「DB2 UDB for iSeries データベース・プログラミング」を参照してください。

トリガーの作成時に定義される、トリガーを起動する時期を決めるのに使われる基準が、いくつかあります。

- *対象表* は、トリガーが定義される表を定義します。
- *トリガー・イベント* は、対象表を変更する特定の SQL 操作を定義します。この操作としては、削除、挿入、更新が可能です。
- *トリガー起動時* は、トリガーを起動するのは、対象表に対するトリガー・イベントの実行前であるか、実行後であるかを定義します。

トリガーを起動するステートメントには、*影響を受ける行の集合* が含まれています。これらは、対象表の中の削除、挿入、または更新される行を表します。*トリガー細分性* は、トリガー・アクションを実行するのは、そのステートメントに対して一度であるのか、影響を受ける行集合の各行ごとに一度であるのかを定義します。

トリガー・アクション は、オプションの検索条件と、トリガーが起動されるたびに実行される 1 組みの SQL ステートメントから構成されます。SQL ステートメントは、検索条件が真と評価されたときにだけ実行されます。

トリガー・アクションは、影響を受ける行集合の値を参照することもできます。これは、*遷移変数* の使用を通してサポートされます。遷移変数は、対象表の中の列名を使用し、その参照が古い値 (更新前) に対するものか、新しい値 (更新後) に対するものかを識別する指定名によって修飾します。新しい値は、更新または挿入トリガーの前に、SET 遷移変数ステートメントを使用して変更することもできます。影響を受ける行集合の値を参照するもう 1 つの方法として、*遷移表* の使用がありま

す。遷移表も対象表の列名を使用しますが、影響を受ける行集合全体を 1 つの表として扱うことができる指定名を持っています。遷移表はトリガーの後でしか使用できません。古い値と新しい値に対して別々の遷移表を定義することも可能です。

表、イベント、起動時の 1 つの組み合わせに対して、複数のトリガーを指定できます。トリガーが起動される順序は、トリガーが作成された順序と同じです。つまり、最新に作成されたトリガーが、最後に起動されるトリガーになります。

トリガーの起動によって、*トリガー・カスケード* が生じることがあります。これは、あるトリガーの起動によって、SQL ステートメントが実行され、その実行によって別のトリガーが起動されたり、同じトリガーが再度起動されたりする結果起きるものです。トリガー・アクションでは、最初の変更の結果として更新が行われ、その結果としてさらにトリガーが起動されるといったことも起こります。トリガー・カスケードを使用すると、有効なトリガー・チェーンを起動することが可能で、単一の削除、挿入、または更新ステートメントによって、データベースに対する多数の変更を行うことができます。

トリガーとして実行されるアクションは、トリガーの実行を引き起こす操作の一環であると見なされます。したがって、分離レベルが NC (コミット不可) 以外で、トリガーのアクションがトリガー・イベントと同一のコミットメントを使用して行われる場合には、

- データベース・マネージャーは、操作とその操作の結果として実行されるトリガーがいずれも、すべて完了であるか、またはすべてバックアウトであることを確認します。トリガーの実行を引き起こす操作に先立って行われた操作は、影響を受けません。
- データベース・マネージャーは、該当の操作および関連するトリガーが実行された後で、すべての制約 (RESTRICT 削除規則を伴う制約を除く) を効果的にチェックします。

トリガーの実行を引き起こす SQL ステートメントにすでに挿入あるいは更新された行について、削除または更新を許可するかどうかを指定する属性がトリガーにはあります。

- トリガーを定義した際に ALWREPCHG(*YES) が指定されている場合、1 つの SQL ステートメント内で、
  - 同じ SQL ステートメントで挿入または更新した行がある場合、トリガーはその行を更新あるいは削除することができます。これには、トリガーが挿入または更新した行や同じ SQL ステートメントで生じた参照制約も含まれます。
- トリガーを定義した際に ALWREPCHG(*NO) が指定されている場合、1 つの SQL ステートメントで、
  - 行は、その行が、同一のその SQL ステートメントによって挿入、または更新されていない場合にのみ、トリガーによって削除することができます。分離レベルが NC (コミット不可) 以外で、トリガーのアクションがトリガー・イベントと同一のコミットメント定義を使用して行われる場合には、トリガー、または同一の SQL ステートメントにより生じた参照制約による挿入や更新が含まれます。
  - 行は、その行が、同一の SQL ステートメントによって挿入、または更新されていない場合にのみ、トリガーによって更新することができます。分離レベルが NC (コミット不可) 以外で、トリガーのアクションがトリガー・イベント

と同一のコミットメント定義を使用して行われる場合には、トリガー、または同一の SQL ステートメントにより生じた参照制約による挿入や更新が含まれます。

CREATE TRIGGER ステートメントを使用して作成されたトリガーは、すべて暗黙的に ALWREPCHG(*YES) 属性をもっています。

索引

索引 とは、基礎表の各行に対するポインターの集まりです。それぞれの索引は、1 つまたは複数の表の列内のデータ値をもとにしています。*索引* は、表内のデータとは独立のオブジェクトです。ユーザーが索引を要求すると、データベース・マネージャーは、索引の構造を構築し、それを自動的に維持管理します。

索引は、名前を持ち、さらに別のシステム名を持つことができます。システム名は、OS/400 によって使用される名前です。SQL ステートメントで索引の名前を指定する場合には、どちらの名前を使用しても構いません。詳細については、508 ページの『CREATE INDEX』を参照してください。

データベース・マネージャーは、次の 2 つのタイプの索引を使用します。

- 2 進基数ツリー索引

  2 進基数ツリー索引は、表の行に対して特定の順序を与えます。データベース・マネージャーは、これを使用して以下のことを行います。

  – パフォーマンスを向上させる。ほとんどの場合、索引を使用した方がデータへのアクセスは高速になります。
  – 固有性を確実にする。固有索引がある表には、同一のキーを持つ行を入れることはできません。
- コード化ベクトル索引

  コード化ベクトル索引は、表の行に対して特定の順序を与えることはありません。データベース・マネージャーは、パフォーマンス向上のためにのみ、この索引を使用します。

コード化ベクトルのアクセス・パスはコード化ベクトル索引の助けにより機能し、コードを異なるキー値に割り当ててからこれらの値を配列で表すことによって、データベース・ファイルへのアクセスを提供します。配列のエレメントは、表すべき異なる値の数によって、長さは 1、2、または 4 バイトになります。このサイズが小さく、比較的単純であるために、コード化ベクトルのアクセス・パスによってスキャンが高速化され、並列処理をより容易に行うことができるようになります。

SQL CREATE INDEX ステートメントを使用して、コード化ベクトルのアクセス・パスを作成します。 コード化ベクトル索引を用いた照会の高速化  の詳細については、DB2 UDB for iSeries の Web ページを参照してください。

ビュー

ビュー は、1 つまたは複数の表のデータを見るための代替方法を提供します。

ビューは、結果表の名前の付いた指定です。その指定は、ビューが SQL ステートメントで参照される時点で実際に実行される SELECT ステートメントです。したがって、ビューは、基礎表と同様に、列や行を持つものとして考えることができます。検索の場合、すべてのビューを基礎表と同様に使用することができます。挿入、更新、または削除の操作で、ビューを使用できるか否かは、CREATE VIEW の説明で述べるようにその定義に依存します。詳細については、 589 ページの『CREATE VIEW』を参照してください。

ビューに対して索引を作成することはできません。ただし、ビューの基礎となる表に対して作成された索引は、そのビューの操作のパフォーマンスを向上させることがあります。

ビューの列が、基礎表の列から直接派生する場合、その列は、基礎表の列に適用される制約をいずれも継承します。例えば、ビューがその基礎表の外部キーを含む場合、そのビューを使用する INSERT および UPDATE 操作は、基礎表と同じ参照制約が課せられます。同様に、ビューの基礎表が親表である場合、そのビューを使用する DELETE 操作は、基礎表に関する DELETE 操作と同一の規則に従います。また、ビューは、その基礎表に適用されるトリガーをいずれも継承します。例えば、ビューの基礎表が更新トリガーを持つ場合、そのトリガーは、そのビューに更新が行われる時点で実行されます。

ビューは名前を持ち、また異なるシステム名を持つこともできます。システム名は、OS/400 によって使用される名前です。 SQL ステートメントで、ビュー名を指定する個所には、どちらの名前でも使用することができます。詳細については、589 ページの『CREATE VIEW』を参照してください。

ビューの列は名前を持ち、また異なるシステム列名を持つことができます。システム列名は、OS/400 によって使用される名前です。SQL ステートメントで、列名を指定する個所には、どちらの名前でも使用することができます。詳細は、 589 ページの『CREATE VIEW』を参照してください。

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可能な場合には、別名を使用して表またはビューを参照することができます。[3] 表およびビューと同様に、別名は作成し、除去し、それに関係した注記あるいはラベルを付けることができます。別名を使用する際に、権限は不要です。ただし、別名が参照している表およびビューをアクセスするには、現行のステートメントに関する適切な権限がやはり必要になります。

別名は、名前を持ち、さらに別のシステム名を持つことができます。システム名は、OS/400 によって使用される名前です。SQL ステートメントで別名を指定する場合には、どちらの名前を使用しても構いません。詳細については、 436 ページの『CREATE ALIAS』を参照してください。

3. 別名はすべての文脈で使用できるわけではありません。例えば、データベース・ファイルの個々のメンバーを参照している別名は、データ定義言語 (DDL) ステートメントでは使用できません。

パッケージとアクセス・プラン

分散 SQL プログラムの場合、パッケージ は、SQL ステートメントを実行するために使用する制御構造を含むオブジェクトです。パッケージは、プログラムを準備するときに作成されます。パッケージ内の制御構造は、SQL ステートメントがバインドされた形式 (つまり SQL ステートメントの実行形式) であると考えることができます。パッケージ内のすべての制御構造は、単一のソース・プログラム内に組み込まれている SQL ステートメントをもとにして作成されます。

パッケージは、QSQPRCED API によって作成することもできます。QSQPRCED API によって作成されたパッケージの使用は、QSQPRCED API によって使用する場合に限定されます。このようなパッケージは、DRDA プロトコルを用いてサーバーで使用することはできません。詳細については、iSeries Information Center の**プログラミング**・カテゴリーの OS/400 API 情報を参照してください。

QSQPRCED API は iSeries Access for Windowsで使用され、SQL ODBC、JDBC および SQLJ インターフェースを介して実行される SQL ステートメントをキャッシュに入れるためのパッケージを作成します。

分散 SQL 以外の SQL プログラムの場合は、SQL ステートメントの実行に使用される制御構造は、その非分散 SQL プログラムの関連スペースに保管されます。

アクセス・プラン という用語は一般に、SQL プログラム、または SQL ステートメントの実行に使用する SQL パッケージに関連するスペースの制御構造を指しています。

プロシージャー

プロシージャー (ストアード・プロシージャーとも呼ばれます) は、一組の命令を行うために呼び出すことができるプログラミング構造です。これらの命令には、ホスト言語ステートメントおよび SQL ステートメントを含めることができます。

プロシージャーは、一般的に SQL プロシージャーと外部プロシージャーに類別されます。SQL プロシージャーには、SQL ステートメントだけが含まれます。外部プロシージャーは、SQL ステートメントを含む場合も、含まない場合もあるホスト言語プログラム (REXX の場合には、ソース・ファイル・メンバー) を参照します。外部プロシージャーおよび SQL プロシージャーの両方とも、DB2 UDB for iSeries でサポートされています。

SQL のプロシージャーは、ホスト言語のプロシージャーと同様の利点をもたらします。すなわち、共通のコード部分を一度だけ作成し、メインテナンスすることによって、複数のプログラムから呼び出すことができます。ホスト言語、および SQL の両者は、そのローカル・システムに存在するプロシージャーを呼び出すことができます。ただし、SQL は、リモート・システムに存在するプロシージャーも呼び出すことができます。事実、SQL のプロシージャーの主要な利点は、プロシージャーを分散アプリケーションのパフォーマンス特性の向上に使用することができる点にあります。

リモート・システムで、複数の SQL ステートメントの実行が必要であると想定します。最初の SQL ステートメントが実行されると、アプリケーション・リクエス

ター (要求元) は、サーバーにその命令の実行要求を送ります。アプリケーション・リクエスターは、該当のステートメントが正しく実行されたか否かを示す応答を待ち、必要に応じて結果を戻します。2 番目およびそれ以降の SQL ステートメントが実行される時点で、アプリケーション・リクエスターは、別の要求を送り、その応答を待ちます。同一の SQL ステートメントがサーバー側のプロシージャーに保管されている場合は、そのリモート・プロシージャーを参照する CALL ステートメントを実行することができます。その CALL ステートメントが実行されると、アプリケーション・リクエスターは、そのプロシージャーを呼び出す単一の要求を現行サーバーに送ります。その上で、アプリケーション・リクエスターは、そのプロシージャーが正常に実行されたか否かを示す単一の応答を待ち、必要に応じて結果を戻します。

次の 2 つの図は、分散アプリケーションでストアード・プロシージャーを使用することによって、リモート要求の数をどのように減らすことができるかを示しています。

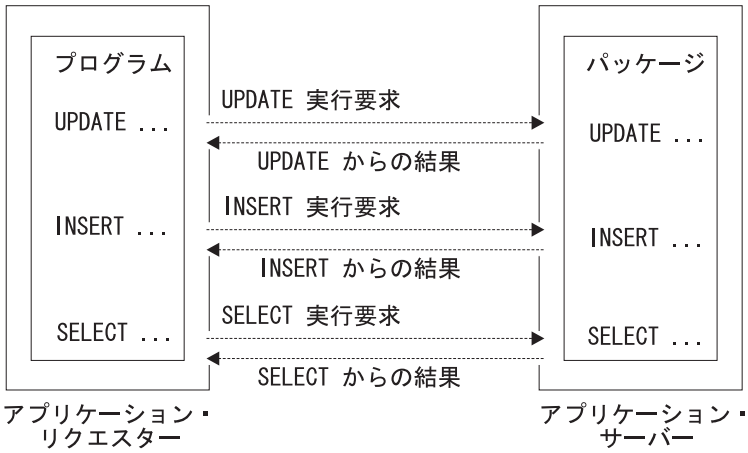


図 1. リモート・プロシージャーを持たないアプリケーション

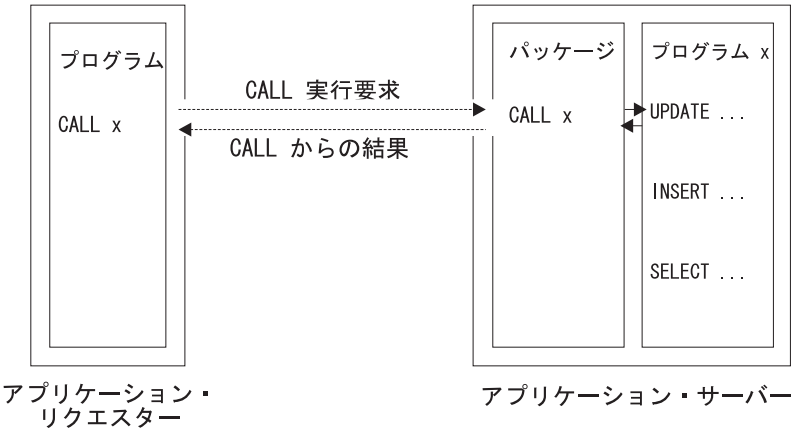


図 2. リモート・プロシージャーを持つアプリケーション

カタログ

データベース・マネージャーは、データベース中のデータに関する情報が入っている一組の表を維持管理しています。これらの表をまとめて*カタログ* と呼びます。*カタログ表* には、システムに存在する表、パラメーター、プロシージャー、パッケージ、ビュー、索引および制約についての情報が入っています。

データベース・マネージャーは、カタログ表に関連するビューを用意しています。これらのビューは、他の IBM SQL プロダクトのカタログ・ビューや、ANSI および ISO 標準のカタログ・ビュー (標準では情報スキーマと呼ばれている) との高い整合性を備えています。 QSYS2 のカタログ・ビューには、システムに存在するすべての表、パッケージ、ビュー、索引、および制約についての情報が入っています。さらに、あるスキーマの表、パッケージ、ビュー、索引、および制約についての情報のみを含む一組のビューが入っている SQL スキーマがあります。

カタログの表およびビューは、他のデータベース表およびデータベース・ビューと類似しています。権限を持っているユーザーであれば、システムの他の表からデータを検索するときと同じように、SQL ステートメントを使用してカタログ・ビューのデータを見ることができます。 データベース・マネージャーの働きによって、カタログには、データベース内の各オブジェクトに関する正確な記述が常に入るようになっています。

カタログの表およびビューの詳細については、929 ページの 『付録 G. DB2 UDB for iSeries のカタログ・ビュー』を参照してください。

アプリケーション・プロセス、並行性、および回復

SQL プログラムはすべてがアプリケーション・プロセスの一環として実行されます。OS/400 では、アプリケーション・プロセスは、ジョブと呼ばれています。ODBC、JDBC、および DRDA の場合は、使用しているジョブが終了していなくて再使用可能であっても、接続が終了した時点でアプリケーション・プロセスは終了します。アプリケーション・プロセスは、1 つまたは複数の活動化グループから成り立っています。活動化グループには、それぞれに 1 つまたは複数のプログラムの実行が含まれます。プログラムの実行は、非デフォルトの活動化グループ、またはデフォルトの活動化グループのもとで行われます。ILE コンパイラーにより作成されたプログラムを除き、すべてのプログラムはデフォルトの活動化グループのもとで実行されます。

活動化グループの詳細については、ILE 概念 を参照してください。

コミットメント制御を使用するアプリケーション・プロセスは、その実行に 1 つまたは複数のコミットメント定義を使用することができます。コミットメント定義を使用すると、コミットメント制御を活動化グループ・レベルまたはジョブ・レベルの範囲で行うことができます。コミットメント制御を使用する活動化グループは、一時点で、ただ 1 つのコミットメント定義に関連付けられます。

コミットメント定義を明示的に開始するには、コミットメント制御開始 (STRCMTCTL) コマンドを使用します。まだ開始されていないコミットメント定義の場合は、COMMIT(*NONE) 以外の分離レベルのもとで最初に SQL ステートメン

トが実行される時点で、暗黙に開始されます。1 つのジョブ・コミットメント定義を複数の活動化グループで共用することができます。

図 3 は、アプリケーション・プロセス、そのアプリケーション・プロセス内の活動化グループ、およびコミットメント定義の関係を示しています。活動化グループの A と B は、その活動化グループを有効範囲とするコミットメント制御を伴って実行されます。これらの活動化グループは、それぞれ独自のコミットメント定義をもっています。活動化グループ C の実行はどのようなコミットメント制御も伴いません。この活動化グループには、コミットメント定義がありません。

ジョブ・レベル・コミットメント定義のないアプリケーション・プロセス

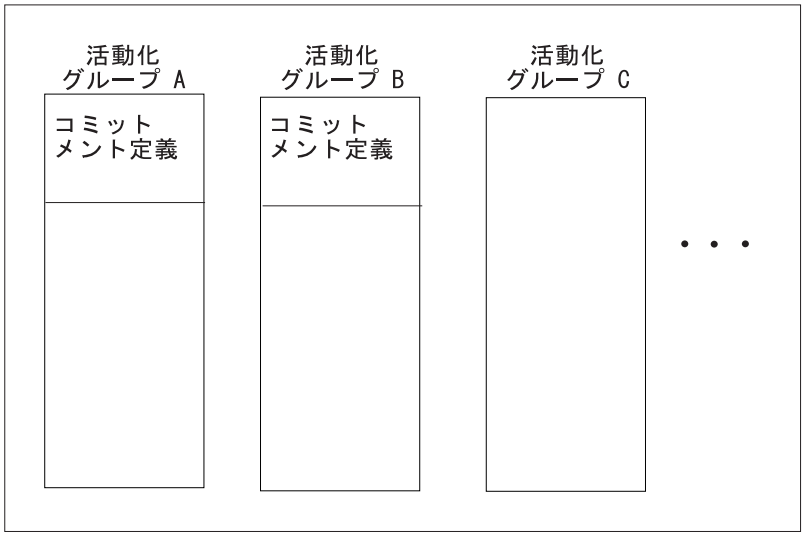


図 3. ジョブのコミットメント定義のない活動化グループ

19 ページの図 4 は、アプリケーション・プロセス、そのアプリケーション・プロセス内の活動化グループ、およびコミットメント定義を示しています。活動化グループの中には、ジョブのコミットメント定義によって実行されているものがあります。活動化グループの A と B は、ジョブのコミットメント定義のもとで実行されています。コミットメント制御は同じコミットメント定義によって行われるので、活動化グループの A または B におけるコミットまたはロールバック操作は、両方の活動化グループが影響を与えます。この例の活動化グループ C は、別のコミットメント定義をもっています。この活動化グループで行われるコミットおよびロールバック操作は、C における操作にのみ影響します。

ジョブ・レベル・コミットメント制御のないアプリケーション・プロセス

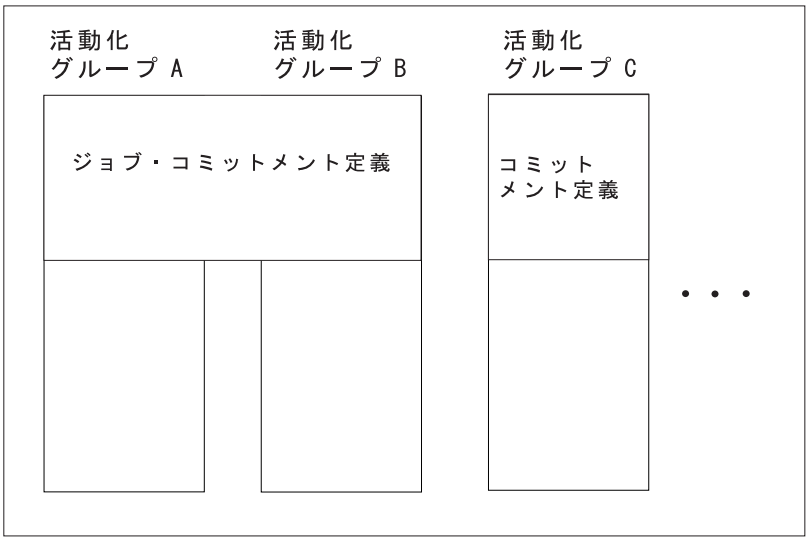


図 4. ジョブのコミットメント定義のある活動化グループ

コミットメント定義についての詳細は、コミットメント制御のトピックを参照してください。

ロック、コミット、およびロールバック

別のコミットメント定義を使用するアプリケーション・プロセスおよび活動化グループは、同時に同じデータに対するアクセスを要求することができます。このような状況でデータの保全性を維持するために、ロックが使用されます。ロックによって、2 つのアプリケーション・プロセスが同じデータの行を同時に更新するような事態を防止できます。

データベース・マネージャーは、異なるコミットメント定義を使用する活動化グループからは検出されないある活動化グループによるコミットされていない変更を保持するために、ロックを獲得します。オブジェクトのロックおよびその他のリソースは、活動化グループに割り振られます。行のロックは、コミットメント定義に割り振られます。

デフォルトの活動化グループ以外の活動化グループが*正常に* 終了すると、データベース・マネージャーは、その活動化グループによるロックをすべて解除します。ユーザーは、ロックをより迅速に解除することを明示的に要求することもできます。この操作をコミットと呼びます。コミット後もオープンのままのカーソルと関連するオブジェクトのロックは、解除されません。

データベース・マネージャーの回復機能には、コミットメント定義で行われた変更がまだコミットされていない場合に、その変更を取り消す方法が用意されています。データベース・マネージャーは以下のような場合に、コミットされていない変更を暗黙にバックアウトすることがあります。

- アプリケーション・プロセスが終了すると、デフォルトの活動化グループに関連するコミットメント定義のもとで行われたすべての変更はバックアウトされま

す。デフォルトの活動化グループ以外の活動化グループが、*異常* 終了すると、その活動化グループに関連するコミットメント定義のもとで行われた変更はすべてバックアウトされます。

- 分散作業単位を使用し、リモート・システムで変更をコミットしようとした時点で障害が起こると、リモート接続に関連するコミットメント定義のもとで行われた変更はすべてバックアウトされます。
- 分散作業単位を使用し、リモート・システムでの障害によりリモート・システムからバックアウトの要求を受け取った場合には、リモート接続に関連するコミットメント定義のもとで行われた変更はすべてバックアウトされます。

ユーザーは、データベースの変更のバックアウトを明示的に要求することができます。この操作を*ロールバック*と呼びます。

活動化グループに代わってデータベース・マネージャーが獲得したロックは、その作業単位が終了するまで保持されます。LOCK TABLE ステートメントによって明示的に獲得されたロックは、COMMIT HOLD または ROLLBACK HOLD を使用して作業単位を終了させると、作業単位の終了後も保持することができます。

カーソルによって、カーソルの置かれている行が暗黙のうちにロックされる場合があります。このロックによって、次のような事態が防止されます。

- 別のコミットメント定義に関連した、他のカーソルによる同一行のロック。
- 別のコミットメント定義に関連した、DELETE または UPDATE ステートメントによる同一行のロック。

作業単位

作業単位 (*論理作業単位* または*回復単位* とも呼ばれます) は、回復可能な一連を操作を指します。それぞれのコミットメント定義には、1 つまたは複数の作業単位の実行が含まれます。どのような時点をとっても、1 つのコミットメント定義には 1 つの作業単位があります。

作業単位は、コミットメント定義が開始された時点、または前の作業単位がコミットやロールバック操作によって終了した時点で開始されます。作業単位は、コミット操作、ロールバック操作、または活動化グループ終了のいずれか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、そのコミットまたはロールバックによって終了する作業単位内で行われたデータベースの変更にのみ影響します。変更のコミットが済まない間は、分離レベル COMMIT(*CS)、COMMIT(*RS)、および COMMIT(*RR) のもとで実行されている異なるコミットメント定義を使用する他の活動化グループは、変更を認知することができません。コミットが行われるまでは、変更を取り消すことができます。変更のコミットが済むと、異なるコミットメント定義で実行されている活動化グループからその変更にアクセスできるようになり、取り消しは不能になります。

1 つの作業単位の開始と終了によって、活動化グループ内の整合点が定義されます。例えば、銀行業務のトランザクションで、ある口座から別の口座に送金を行う場合があります。このようなトランザクションでは、最初の口座から差し引いた金額を、2 番目の口座に加算する必要があります。最初の口座から金額を差し引いたステップの後では、データに整合性はありません。2 番目の口座に金額を加算して初めて、再度整合性が確立されます。この両方のステップが完了した時点で、コミ

ット操作を使用して作業単位を終了させることができます。コミット操作が終わると、異なるコミットメント定義を使用する活動化グループが変更を使用できるようになります。

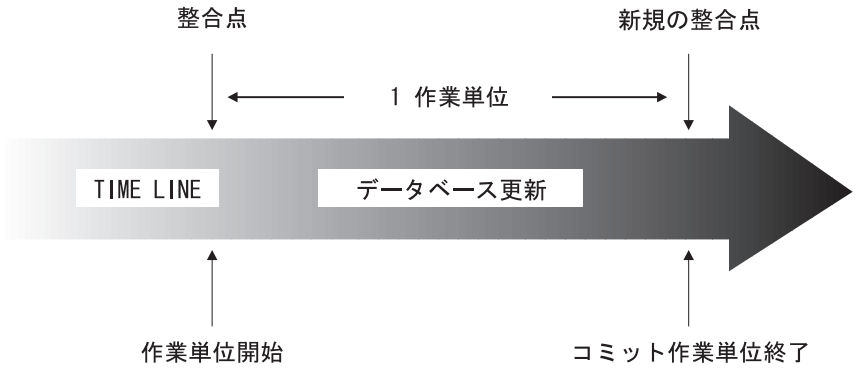


図 5. COMMIT (コミット) ステートメントと作業単位

作業のロールバック

データベース・マネージャーは、1 つの作業単位内で行われたすべての変更、または選択した一部の変更のみをバックアウトすることができます。ただし、整合点が達成されるのはすべての変更をバックアウトした場合だけです。

すべての変更のロールバック

TO SAVEPOINT 文節を伴わない SQL ROLLBACK ステートメントを使用すると、フル・ロールバック操作が行われます。このようなロールバック操作が正常に実行されると、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をバックアウトして、作業単位の開始時点で存在していたものと見なされるデータ一貫性を復元します。つまり、データベース・マネージャー は、以下の図のように作業を取り消します。

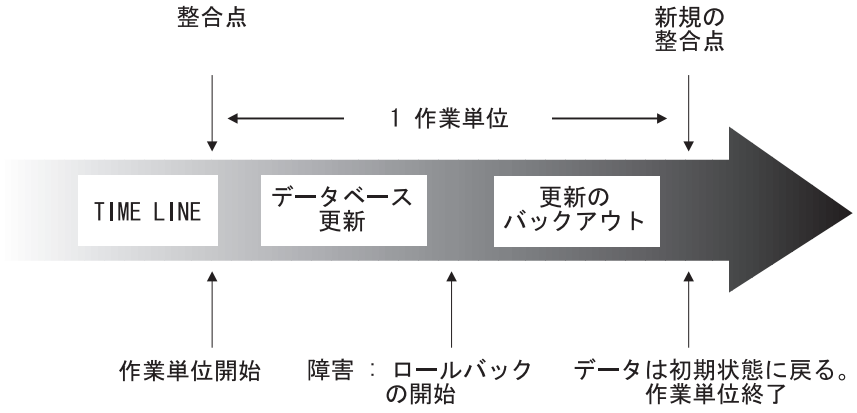


図 6. ROLLBACK (ロールバック) ステートメントと作業単位

保管ポイントを使用して選択した変更のロールバック

保管ポイント (savepoint) は、1 つの作業単位の中の特定時点におけるデータの状態を表します。アプリケーション・プロセスは、作業単位内に保管ポイントを設定しておき、ロジックの指示に従って、特定の保管ポイントの設定後に行われた変更のみをロールバックすることができます。例えば、旅行予約トランザクションには、航空券予約とホテルの予約が含まれることがあります。ここで、航空券は予約できたが、ホテルが予約できなかったという場合、アプリケーション・プロセスで航空券予約だけを取り消して、航空券予約より前にトランザクション内で行われたデータベース変更は取り消さないようにしたい場合があります。このような場合に、SQL プログラムは、SQL SAVEPOINT ステートメントを使用して保管ポイントを設定し、TO SAVEPOINT 文節を伴う SQL ROLLBACK ステートメントを使用して特定の保管ポイントまたは最後に設定された保管ポイントまで変更を取り消し、そして、RELEASE SAVEPOINT ステートメントを使用して保管ポイントを削除することができます。

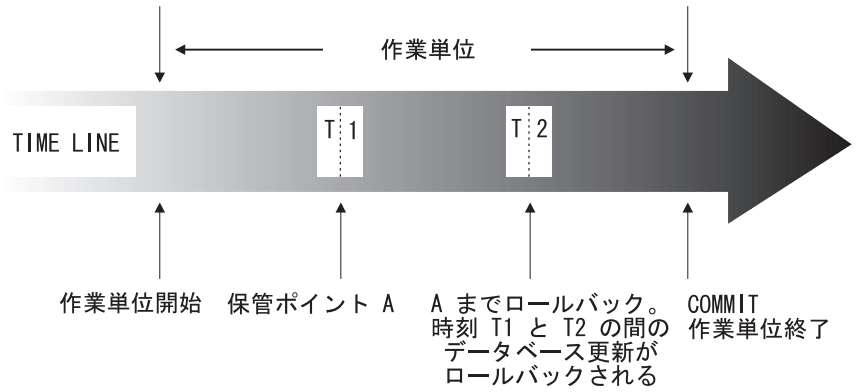


図 7. ROLLBACK (ロールバック) ステートメントおよび SAVEPOINT ステートメントと作業単位

スレッド

OS/400 では、アプリケーション・プロセスも 1 つまたは複数のスレッドから構成することができます。デフォルトでは、スレッドは同じコミットメント定義を共用し、そのジョブにおける他のスレッドのようにロックします。このため、1 つのスレッドがコミットあるいはロールバックする場合、そのスレッドはすべてのスレッドが行ったすべての変更をコミットあるいはロールバックすることができるように、それぞれのスレッドは同じ作業単位で機能することができます。このタイプの処理は、複数のスレッドで協調して単一のタスクを並列に実行する場合に便利です。

その他のケースとしては、あるスレッドがジョブ内の他のスレッドとは独立して、変更を行うような場合に便利です。この場合、そのスレッドではコミットメント定義を共用したり、他のスレッドとともにロックする必要はありません。さらに、複数データベースの接続とトランザクション情報について、よりすぐれた、きめ細かい制御を行うために、ジョブは SQL サーバー・モードを使用することができま

す。典型的なマルチスレッドのジョブでは、このような制御が必要になる場合があります。このタイプの処理方法は、いくつかあります。

- スレッドで実行するプログラムは、必ず、別の活動化グループを使用するようにします (ACTGRP(*NEW) を使用しないように注意します)。
- 最初の SQL ステートメントを出す前から、ジョブは、必ず、SQL サーバー・モードで実行しているようにします。SQL サーバー・モードは、アプリケーションでデータ・アクセスが生じる前に以下のいずれかのメカニズムを使用することによって、ジョブに対して活動化することができます。
  - データ・アクセスが行われる前に、ODBC API、SQLSetEnvAttr() を使用して、SQL_ATTR_SERVER_MODE 属性を SQL_TRUE に設定する。
  - データ・アクセスが行われる前に、Change Job API、QWTCHGJB() を使用して、'Server mode for Structured Query Language (SQL のサーバー・モード)' キーを設定する。
  - JAVA を使用し、JDBC を介してデータベースをアクセスする。JDBC は、自動的にサーバー・モードを使用し、必要な JDBC の意味体系を維持する。

SQL サーバー・モードが確立されると、すべての SQL ステートメントは、要求を取り扱う独立したサーバー・ジョブに渡されます。SQL 動作に関するサーバー・モード動作には、以下のものが含まれます。

- 組み込み SQL の場合、ジョブの各スレッドはデータベースに対する唯一の接続を (したがって、それ自体のコミット可能なトランザクションも) 暗黙的に取得します。
- ODBC/CLI および JDBC の場合、それぞれの接続はデータベースに対する独立型の接続を表しており、別々のエンティティーとしてコミット可能であり、使用することができます。

詳細については、DB2 UDB for iSeries SQL 呼び出しレベル・インターフェースを参照してください。

以下の SQL サポートは、スレッド・セーフではありません。

- DRDA 経由のリモート・アクセス
- ALTER TABLE
- COMMENT
- CREATE ALIAS
- CREATE DISTINCT TYPE
- CREATE FUNCTION
- CREATE INDEX
- CREATE PROCEDURE
- CREATE SCHEMA
- CREATE TABLE
- CREATE TRIGGER
- CREATE VIEW
- DECLARE GLOBAL TEMPORARY TABLE
- DROP
- GRANT

- LABEL
- RENAME
- REVOKE

詳細については、iSeries Information Center のプログラミング・トピックのマルチスレッド・アプリケーションを参照してください。

分離レベル

SQL ステートメントの実行中に使用する*分離レベル* によって、活動化グループが並行して実行される他の活動化グループから分離される度合いが決定されます。したがって、活動化グループ P が SQL ステートメントを実行すると、分離レベルによって次のことが決定されます。

- P によって検索される行、および P によって行われるデータベースの変更が、並行して実行される他の活動化グループで使用できる度合い。
- 並行して実行される活動化グループによって行われるデータベースの変更が、P に影響を及ぼす度合い。

分離レベルは、SQL プログラムまたは SQL パッケージの属性として指定され、その SQL パッケージまたは SQL プログラムを使用する活動化グループに適用されます。DB2 UDB for iSeries では、分離レベルを指定するために、いくつかの方法を提供しています。

- デフォルトの分離レベルを指定する場合は、CRTSQLxxx、STRSQL、および RUNSQLSTM コマンドで COMMIT パラメーターを使用します。
- 組み込み SQL を持つソース・モジュールまたはソース・プログラム内でデフォルトの分離レベルを指定する場合は、SET OPTION ステートメントを使用します。
- 作業単位内で分離レベルを指定変更する場合は、SET TRANSACTION ステートメントを使用します。その作業単位が終了すると、分離レベルはその作業単位の開始時点での値に戻ります。
- 特定のステートメントまたはカーソルについてのデフォルトの分離レベルを指定変更する場合は、SELECT、SELECT INTO、INSERT、UPDATE、DELETE、および DECLARE CURSOR ステートメントで分離文節を使用します。分離レベルは、分離文節を持つステートメントを実行する場合にのみ有効であり、現行の作業単位における保留中の変更に対しては無効です。

これらの分離レベルは、該当するデータを自動的にロックすることによってサポートされます。ロックのタイプに応じて、異なるコミットメント定義を使用して、並行して実行される活動化グループによるデータのアクセスが制約、または禁止されます。それぞれのデータベース・マネージャーでは、少なくとも次に示す 2 つのロックのタイプをサポートしています。

共用 異なるコミットメント定義を使用する、並行して実行される活動化グループを、データに対する読み取り専用操作に限定します。

排他 異なるコミットメント定義を使用する、並行して実行される活動化グループによるデータの更新および削除を防止します。並行して実行される活動化グループが、COMMIT(*RS)、COMMIT(*CS)、または COMMIT(*RR) を実行している異なるコミットメント定義を使用する場合は、それによるデータの

読み取りを防止します。並行して実行される活動化グループが、COMMIT(*UR) または COMMIT(*NC) を実行している異なるコミットメント定義を使用する場合は、それによるデータの読み取りを許します。

分離レベルに関する以下の説明は、行単位で行われるデータのロックについて述べています。個々のインプリメンテーションでは、基礎表の行よりも大きな物理単位でデータをロックすることができる場合があります。ただし、論理的には、ロックはすべての製品において基礎表の行レベルで行われます。同様に、データベース・マネージャーでは、ロックをより上位のレベルにまで拡大することができます。活動化グループには、少なくとも要求される最低限のロック・レベルが保証されます。

DB2 UDB for iSeries では、5 つの分離レベルをサポートします。コミット不可以外のすべての分離レベルで、データベース・マネージャーは、挿入、更新、または削除されるすべての行に排他ロックします。これにより、ある作業単位の過程で変更された行は、その作業単位が完了するまで、異なるコミットメント定義を使用する他の活動化グループにより変更されることはありません。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)

  分離レベル RR では、以下のことが確認されます。

  – ある作業単位の過程で読み取られた行は、その作業単位が完了するまで、別のコミットメント定義を使用する他の活動化グループにより変更されることはない。[4]
  – 別のコミットメント定義を使用する他の活動化グループによって変更された行 (あるいは現在 UPDATE のための行ロックでロックされている行) は、その行がコミットされるまで読み取ることはできない。

  分離レベル RR で実行されている活動化グループは、任意の排他ロックに加えて、少なくともその活動化グループが読み取ったすべての行に共用ロックします。さらに、その活動化グループが、異なるコミットメント定義を使用する、並行して実行される活動化グループの影響から完全に分離されるようにロックされます。

  DB2 UDB for iSeries では、COMMIT(*RR) によって反復可能読み取りをサポートします。分離レベル「反復可能読み取り」は、読み取りまたは更新の対象となる行が含まれている表にロックすることによってサポートされます。 ANS および ISO 標準では、反復可能読み取りは逐次化可能と呼ばれています。

- 読み取り固定 (RS)

  分離レベル RR と同様に、分離レベル RS では以下のことが確認されます。

  – ある作業単位の過程で読み取られた行は、その作業単位が完了するまで、別のコミットメント定義を使用する他の活動化グループにより変更されることはない。[4]

4. **WITH HOLD** カーソルの場合、この規則は実際に読み取られたときに適用されます。読み取り専用 **WITH HOLD** カーソルの場合は、事前の作業単位で行がすでに実際に読み取られている場合があります。

– 別のコミットメント定義を使用する他の活動化グループによって変更された行 (あるいは現在 UPDATE のための行ロックでロックされている行) は、その行がコミットされるまで読み取ることはできない。

RR とは異なり、分離レベル RS では、同時に実行されている別のコミットメント定義を使用する活動化グループの影響から、完全には分離されません。分離レベル RS では、活動化グループから同じ照会を複数回出すと追加の行を見ることがあります。この追加の行は、*単独読み取り行* と呼ばれます。

例えば、単独読み取り行が発生するのは次のような場合です。

1. 活動化グループ P1 で、何らかの検索条件を満たす行 *n* の集合を読み取る。
2. 次に、活動化グループ P2 で上記の検索条件を満たす 1 つまたは複数の行を挿入し、その挿入をコミットする。
3. ここで、P1 から前回と同じ検索条件で行の集合を読み取ると、当初の行と P2 によって挿入された行の両方が入手される。

分離レベル RS で実行されている活動化グループは、それ自体が読み取るすべての行に、たとえ排他ロックされている場合でも、それに加えて少なくとも共用ロックします。

DB2 UDB for iSeries では、COMMIT(*ALL) または COMMIT(*RS) によって読み取り固定をサポートします。 ANS および ISO 標準では、読み取り固定は反復可能読み取りと呼ばれています。

- カーソル固定 (CS)

  分離レベル RR および RS と同様に、分離レベル CS では、別のコミットメント定義を使用して他の活動化グループが変更した行 (あるいは現在 UPDATE 行ロックでロックされている行) は、コミットされるまで読み取ることはできません。ただし、RR および RS の場合とは異なり、分離レベル CS で保証されるのは、すべての更新可能なカーソルの現在行が、異なるコミットメント定義を使用する他の活動化グループによって変更されることはないということだけです。したがって、ある作業単位の過程で読み取られた行を、別のコミットメント定義を使用する別の活動化グループにより変更することができます。分離レベル CS で実行されている活動化グループには、任意の排他ロックに加えて、すべてのカーソルの現行行に対する共用ロックを獲得することもできます。

  DB2 UDB for iSeries では、COMMIT(*CS) によってカーソル固定をサポートします。 ANS および ISO 標準では、カーソル固定はコミット読み取りと呼ばれています。

- 非コミット読み取り (UR)

  SELECT INTO、読み取り専用カーソル付きの FETCH、副照会、または INSERT ステートメントで使用される副選択の場合は、分離レベル UR で以下のことが可能になります。

  – ある作業単位の過程で読み取られた行は、別のコミットメント定義の下で実行されている他の活動化グループにより変更できる。
  – 別のコミットメント定義の下で実行されている他の活動化グループによって変更された行 (あるいは現在 UPDATE 行ロックでロックされている行) は、いずれも、変更のコミットメントが行われていない場合でも読み取ることができる。

それ以外の操作の場合は、分離レベル CS の規則が適用されます。

DB2 UDB for iSeries では、COMMIT(*CHG) または COMMIT(*UR) によって非コミット読み取りをサポートします。ANS および ISO 標準では、非コミット読み取りは、読み取り非コミットと呼ばれています。

- コミット不可 (NC)

  すべての操作に関して、以下を除いて、分離レベル UR の規則が適用されます。

  – SQL ステートメントで、コミットおよびロールバックの操作は無効です。カーソルはクローズされず、また LOCK TABLE のロックは解除されません。ただし、解除保留状態の接続は終了します。
  – 変更はいずれも、正常に行われた各変更操作の終了時に効果的にコミットされ、異なるコミットメント定義を使用する他のアプリケーション・グループによるアクセスまたは変更をただちに行うことができます。

  DB2 UDB for iSeries では、COMMIT(*NONE) または COMMIT(*NC) によってコミット不可をサポートします。

  レコードのロック持続期間についての詳細は、資料「SQL プログラミング 概念」のコミットメント制御のトピックの説明および表を参照してください。

注: *(分散アプリケーションに関する注意)* 要求した分離レベルがサーバーによってサポートされていない場合は、分離レベルは、その次にサポートされている最上位の分離レベルまで拡大されます。例えば、サーバーで分離レベル RS がサポートされていない場合は、分離レベル RR が使用されます。

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、一組の表とその他のオブジェクト (別個ではあるが相互に接続されているコンピューター・システムにまたがって存在している) で構成されます。各コンピューター・システムには、それぞれその環境で表を管理するリレーショナル・データベース・マネージャーがあります。これらのデータベース・マネージャーは、相互に情報を交換し連携することによって、それぞれのデータベース・マネージャーが別のコンピューター・システムに対して SQL ステートメントを実行することができる仕組みになっています。

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 (要求元) とサーバーのプロトコルおよび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* (要求元) は、接続のアプリケーション側をサポートします。アプリケーション・リクエスターは、アプリケーションのデータベース要求を、分散データベース・ネットワークによる使用に適した通信プロトコルに変換します。これらの要求は、接続のもう一方の側のサーバー によって受信され処理されます。[5] アプリケーション・リクエスターとサーバーは、協力して通信やロケーションの問題を処理し、それによって、アプリケーションは、それらの問題から解放され、ローカルのデータベースをアクセスするかのように操作することが可能になります。 28 ページの図 8は、単純な分散リレーショナル・データベース環境を示しています。

5. これは、*アプリケーション・サーバー* とも呼ばれます。

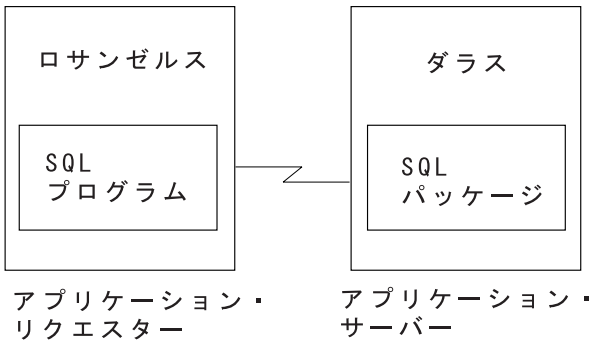


図 8. 分散リレーショナル・データベース環境

分散リレーショナル・データベース・アーキテクチャー (DRDA) 通信プロトコルについての詳細は、「 Distributed Relational Database Architecture Reference」を参照してください。

データベース・サーバー

表やビューを参照する SQL ステートメントの実行に先立って、活動化グループをデータベース・マネージャーのサーバーに接続しなければなりません。

接続 とは、活動化グループと、ローカルまたはリモートのサーバーとの間の結び付きを言います。接続は、アプリケーションにより管理されます。CONNECT ステートメントを使用して、サーバーとの接続を確立し、そのサーバーを活動化グループの現行サーバーとすることができます。

サーバーは、活動化グループが開始される環境に対して、ローカルでもリモートでも構いません (サーバーは、分散リレーショナル・データベースを使用しない場合でも存在しています)。該当の環境には、CONNECT ステートメントで識別できるサーバーを記述するローカル・ディレクトリーがあります。ディレクトリーの詳細については、以下の iSeries Information Center トピックのディレクトリー・コマンド (ADDRDBDIRE、CHGRDBDIRE、DSPRDBDIRE、RMVRDBDIRE、および WRKRDBDIRE) を参照してください。

- SQL プログラミング 概念
- 分散データベース・プログラミング
- CL 解説書

表またはビューを参照する静的 SQL ステートメントを実行する場合、サーバーは、そのステートメントのバインド済みの形式を使用します。このバインド済みのステートメントは、前もってデータベース・マネージャーがバインド操作によって作成したパッケージから得られます。以下の組み合わせによって適切なパッケージが決定されます。

- CRTSQLxxx コマンドの SQLPKG パラメーターによって指定されたパッケージの名前。CRTSQLxxx コマンドの説明に関しては、DB2 UDB for iSeries ホスト言語での SQL プログラミングを参照してください。
- 内部の整合性トークン (パッケージおよびプログラムが、同時に同じソースから作成されたことを確認する)。

IBM のリレーショナル・データベースの製品はすべて、IBM SQL の拡張機能をサポートしています。このような拡張機能には、製品に特有の機能や複数の製品に共通する機能があります。

多くの場合、アプリケーションは、そのアプリケーションが現在接続されているサーバーのデータベース・マネージャーによってサポートされているステートメントや文節を使用することができます。(アプリケーションが、それらのステートメントや文節のいくつかをサポートしないデータベース・マネージャーの適用アプリケーション・リクエスターを介して実行されている場合でも)。制限については、913 ページの 『付録 F. SQL ステートメントの特性』に示されています。

CONNECT (タイプ 1) および CONNECT (タイプ 2)

構文は同じで、意味が異なる 2 つのタイプの CONNECT ステートメントがあります。

- CONNECT (タイプ 1) は、リモート作業単位に対して使用されます。 425 ページの『CONNECT (タイプ 1)』を参照してください。
- CONNECT (タイプ 2) は、分散作業単位に対して使用されます。 431 ページの『CONNECT (タイプ 2)』を参照してください。

これらの相違点についての要約は、 926 ページの『CONNECT (タイプ 1) と CONNECT (タイプ 2) の相違点』を参照してください。

リモート作業単位

リモート作業単位 機能は、リモートでの SQL ステートメントの準備と実行を可能にします。コンピューター・システム A の活動化グループは、コンピューター・システム B のサーバーに接続することができます。そうすると、1 つまたは複数の作業単位内で、その活動化グループは、B にある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。B の作業単位が終了した後、活動化グループはコンピューター・システム C のサーバーに接続できるというようになります。

ほとんどの SQL ステートメントは、以下の制約を伴うものの、リモートで準備し実行することができます。

- 1 つの SQL ステートメントで参照されるオブジェクトはすべて、同じサーバーによって管理される必要があります。
- 1 つの作業単位のすべての SQL ステートメントは、同じサーバーによって実行される必要があります。

リモート作業単位の接続管理

活動化グループは、必ず以下の 3 つの状態のいずれかになります。

接続可能 / 接続状態
接続不能 / 接続状態
接続可能 / 未接続状態

次の図は、状態の推移を示しています。

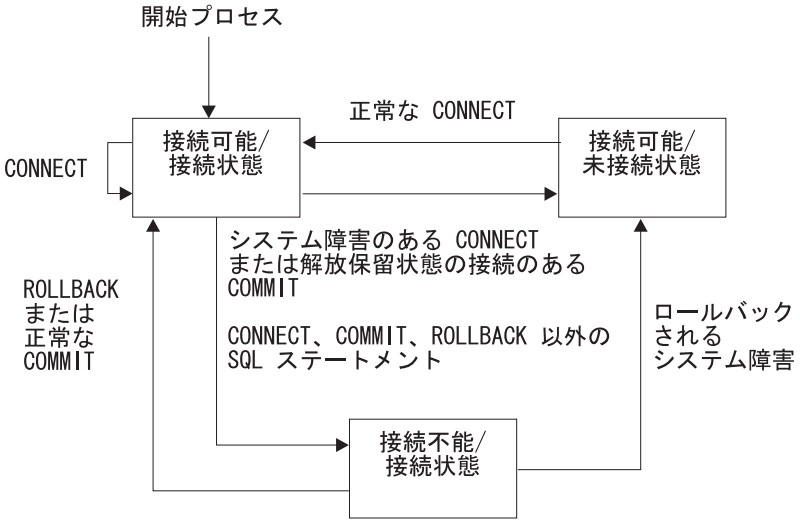


図 9. リモート作業単位の活動化グループの接続状態の推移

活動化グループの初期状態は、*接続可能/接続状態* です。活動化グループの接続先のサーバー は、CRTSQLxxx および STRSQL コマンドの RDB パラメーターによって決定され、また暗黙の CONNECT 操作が関与する場合があります。暗黙の CONNECT 操作は、すでに暗黙または明示の CONNECT 操作が行われ、成功または不成功になっている場合には、行われません。したがって、ある活動化グループが複数回にわたって 1 つのサーバーに暗黙接続されることはあり得ません。

接続可能 / 接続状態: 活動化グループがサーバーに接続され、CONNECT ステートメントが実行できる状態です。活動化グループがこの状態に入るのは、活動化グループが接続不能 / 接続状態からロールバックまたは正常なコミットを完了したとき、または CONNECT ステートメントが接続可能 / 未接続状態から正常に実行されたときです。

接続不能 / 接続状態: 活動化グループはサーバーに接続されているが、CONNECT ステートメントが正常に実行できないので、サーバーを変更できない状態です。活動化グループは、CONNECT、COMMIT、または ROLLBACK 以外の SQL ステートメントを実行すると、接続可能 / 接続状態からこの状態に入ります。

接続可能 / 未接続状態: 活動化グループはサーバーに接続されていません。この状態で実行できる SQL ステートメントは、CONNECT だけです。

活動化グループは、以下の場合にこの状態に入ります。

- 前もって接続が解除されており、正常な COMMIT が実行される。
- SQL DISCONNECT ステートメントを使用して、接続が切り離される。
- 接続が接続可能状態であったが CONNECT ステートメントの実行が失敗した。

CONNECT ステートメントは、連続して使用しても正常に実行されます。これは、CONNECT ステートメントは接続可能状態からその活動化グループを除去しないからです。活動化グループが現在接続されているサーバーに対する CONNECT は、他の CONNECT ステートメントと同様に実行されます。 CONNECT、 COMMIT、 DISCONNECT、 SET CONNECTION、 RELEASE、または ROLLBACK (COMMIT(*NC) を指定して実行する場合を除く) 以外の SQL ステートメントが前

に実行されていると、CONNECT は正常に実行できません。エラーを避けるために、CONNECT ステートメントを実行する前に、コミットまたはロールバック操作を実行してください。

アプリケーション指向の分散作業単位

アプリケーション指向の分散作業単位機能 は、リモート作業単位の場合と同じように、リモートでの SQL ステートメントの準備と実行を可能にします。リモート作業単位と同様に、コンピューター・システム A の活動化グループは、コンピューター・システム B のサーバーに接続でき、作業単位の終了に先立って、B にある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。1 つの SQL ステートメントで参照されるオブジェクトはすべて、同じサーバー によって管理される必要があります。ただし、リモート作業単位と異なり、同じ作業単位にいくつかのサーバーが関与することができます。コミット、またはロールバックの操作により、作業単位は終了します。

分散作業単位は APPC および TCP/IP 接続用に完全サポートされています。

アプリケーション指向の分散作業単位の接続の管理

どのような場合でも、

- 活動化グループは、必ず*接続状態* または*未接続状態* にあり、ゼロまたはそれ以上の接続からなる一組の接続をもっています。活動化グループの各接続は、接続先のサーバーの名前によって、個々に識別されます。
- SQL 接続は、常に以下の状態のいずれかです。
  - 現行 / 保留
  - 現行 / 解除保留
  - 休止 / 保留
  - 休止 / 解除保留

活動化グループの初期状態: 活動化グループは最初は接続状態にあり、接続は 1 つだけです。接続の初期状態は、*現行 / 保留* です。

次の図は、状態の推移を示しています。

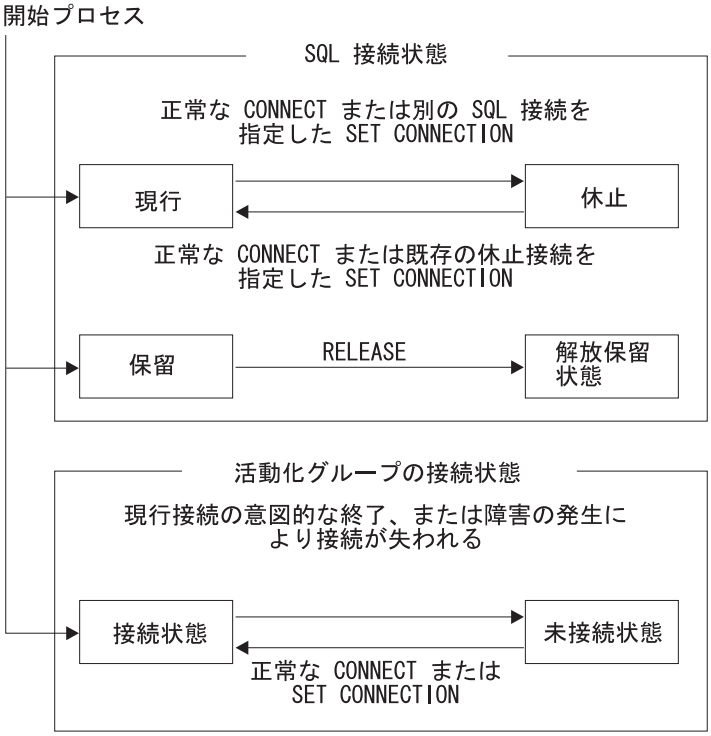


図 10. アプリケーション指向の分散作業単位の接続および活動化グループの接続状態の推移

接続状態

アプリケーション・プロセスが CONNECT ステートメントを正常に実行した場合 :

- 現行接続が休止 / 保留状態になる。
- そのサーバー名が接続の組に追加され、その新たな接続が現行 / 保留状態になる。

該当のサーバー名が、活動化グループの既存の接続のセットにすでに存在する場合には、エラーが起こります。

休止状態の接続は、SET CONNECTION ステートメントの使用により現行状態になります。ある接続が現行状態になると、それ以前の現行接続 (存在する場合) は休止状態になります。どのような時点でも、活動化グループの既存の接続のセットの複数の接続が、現行状態になることはありません。接続の状態を現行から休止へ、または休止から現行へ変更しても、その保留状態や解除保留状態には影響しません。

接続は、RELEASE ステートメントによって解除保留状態になります。活動化グループがコミット操作を実行すると、その活動化グループの解除保留状態の接続はすべて終了します。接続の状態を保留から解除保留へ変更しても、その現行状態や休止状態には影響しません。したがって、解除保留状態の接続は、次のコミット操作まで引き続き使用できます。接続の状態を解除保留から保留へ変更する手段はありません。

活動化グループの接続状態

CONNECT ステートメントの暗黙、または明示的な実行によって、異なるサーバーに接続が可能です。以下の規則が適用されます。

- 活動化グループは、同時に同じサーバーに対し、複数の接続を行うことはできません。
- 活動化グループが SET CONNECTION ステートメントを実行する場合、指定するロケーション名は、その活動化グループの既存の接続のセットに存在する既存の接続でなければなりません。
- 活動化グループが CONNECT ステートメントを実行する場合、指定するサーバー名は、その活動化グループの既存の接続のセットに存在する既存の接続であってはなりません。

活動化グループが現行接続を持つ場合、その活動化グループは*接続* 状態です。特殊レジスター CURRENT SERVER には、その現行接続のサーバーの名前が入っています。その活動化グループは、その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することができます。

未接続状態の活動化グループが CONNECT または SET CONNECTION ステートメントを実行し、成功すると、接続状態になります。

活動化グループが現行接続を持たない場合、その活動化グループは*未接続* 状態です。特殊レジスター CURRENT SERVER の内容は、ブランクに等しくなります。実行できる SQL ステートメントは、CONNECT、DISCONNECT、SET CONNECTION、RELEASE、COMMIT、および ROLLBACK のみです。

接続状態の活動化グループが未接続状態に入るのは、その現行接続を意図的に終了させた場合、または現行サーバーのロールバック操作または接続の消失の原因となる障害のために、SQL ステートメントの実行が正常になされなかった場合です。接続を意図的に終了させるのは、活動化グループがコミット操作を正常に実行し、しかもその接続が解除保留状態にある場合、またはアプリケーション・プロセスが DISCONNECT ステートメントを正常に実行した場合です。

接続が終了する場合

接続が終了すると、その接続を介して活動化グループが獲得していたすべてのリソース、およびその接続の確立や維持に使用されていたすべてのリソースが割り振り解除されます。例えば、アプリケーション・プロセス P がサーバー X への接続を解放保留状態にした場合は、その接続が次のコミット操作中に終了すると、X にある P のカーソルはすべてクローズし、割り振りは解除されます。

また、接続は通信障害の結果として終了することもあり、その場合、該当の活動化グループは、未接続状態になります。活動化グループが終了すると、その活動化グループの接続はすべて終了します。

データ表現に関する考慮事項

システムが異なれば、データの表現方法も異なります。あるシステムから別のシステムにデータを移す場合に、データ変換が必要になることがあります。DRDA をサポートするプロダクトでは、データ変換が必要な場合、その変換は受信側システムで自動的に行われます。

数字データの変換を行うのに必要な情報は、データ・タイプと送信側システムの環境タイプです。例えば、DB2 UDB for iSeriesのアプリケーション・リクエスターか

らの浮動小数点変数を OS/390 サーバーの表の列に割り当てると、その数値は IEEE 形式から System/370* (システム/370) 形式に変換されます。

文字データや図形データの場合には、データ・タイプと送信側システムの環境タイプだけでは十分ではありません。文字や図形のストリングを変換するには、さらに情報が必要になります。ストリングの変換は、データのコード化文字セットおよびそのデータに対して行われる操作の両方に基づいて行われます。ストリングの変換は、IBM 文字データ表現体系 (CDRA) に従って行われます。文字変換に関する詳細は、*Character Data Representation Architecture Level 1 Reference,* SC09-1390 を参照してください。

文字変換

ストリング とは、文字を表す一連のバイトを指します。 1 つのストリングの中では、すべての文字が共通のコード表示で表されます。これらの文字を別のコード表示に変換しなければならない場合があります。変換の処理を文字変換 と呼びます。[6]

SQL ステートメントがリモートで実行される場合には、文字変換が行われる可能性があります。例えば、次の 2 つの場合を考えてみます。

- ホスト変数の値が、アプリケーション・リクエスターから現行サーバーに送信される。
- 結果の列の値が、現行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る。

上記のどちらの場合も、送信側と受信側のシステムでストリングの表現が異なる可能性があります。同一のシステムにおけるストリング操作でも、変換が行われる場合があります。

以下のリストは、文字変換の説明で使用される用語のいくつかを定義しています。

文字セット

定義された文字の集合。例えば、次のような文字セットを持つコード・ページがあります。

- A から Z までのアクセントなしの文字 (26 文字)
- a から z までのアクセントなしの文字 (26 文字)
- 0 から 9 までの数字
- . , : ; ? ( ) ’ ″ / - _ (下線) & + % * = < >

コード・ページ

コード・ポイントに対して文字を割り当てた集合。例えば、EBCDIC では、"A" がコード・ポイント X'C1' に割り当てられ、"B" がコード・ポイント X'C2' に割り当てられています。 1 つのコード・ページ内では、それぞれのコード・ポイントが特定の意味を 1 つだけ持ちます。

6. 文字変換は、必要に応じて自動的に行われ、変換が正常に行われる場合は、アプリケーションに影響を与えることはありません。したがって、ステートメントの実行に関連するすべてのストリングが同一の方法で表現されている場合には、変換の知識は必要ありません。したがって、多くの読者の場合、文字変換の知識は必要ないはずです。

コード・ポイント
文字を表す固有のビット・パターン。

コード化文字セット
文字セットを確立するとともに、セット内の文字とそのコード表示との間に 1 対 1 の関係を確立する明確な規則の集合。

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合。これには例えば以下のようなものがあります。

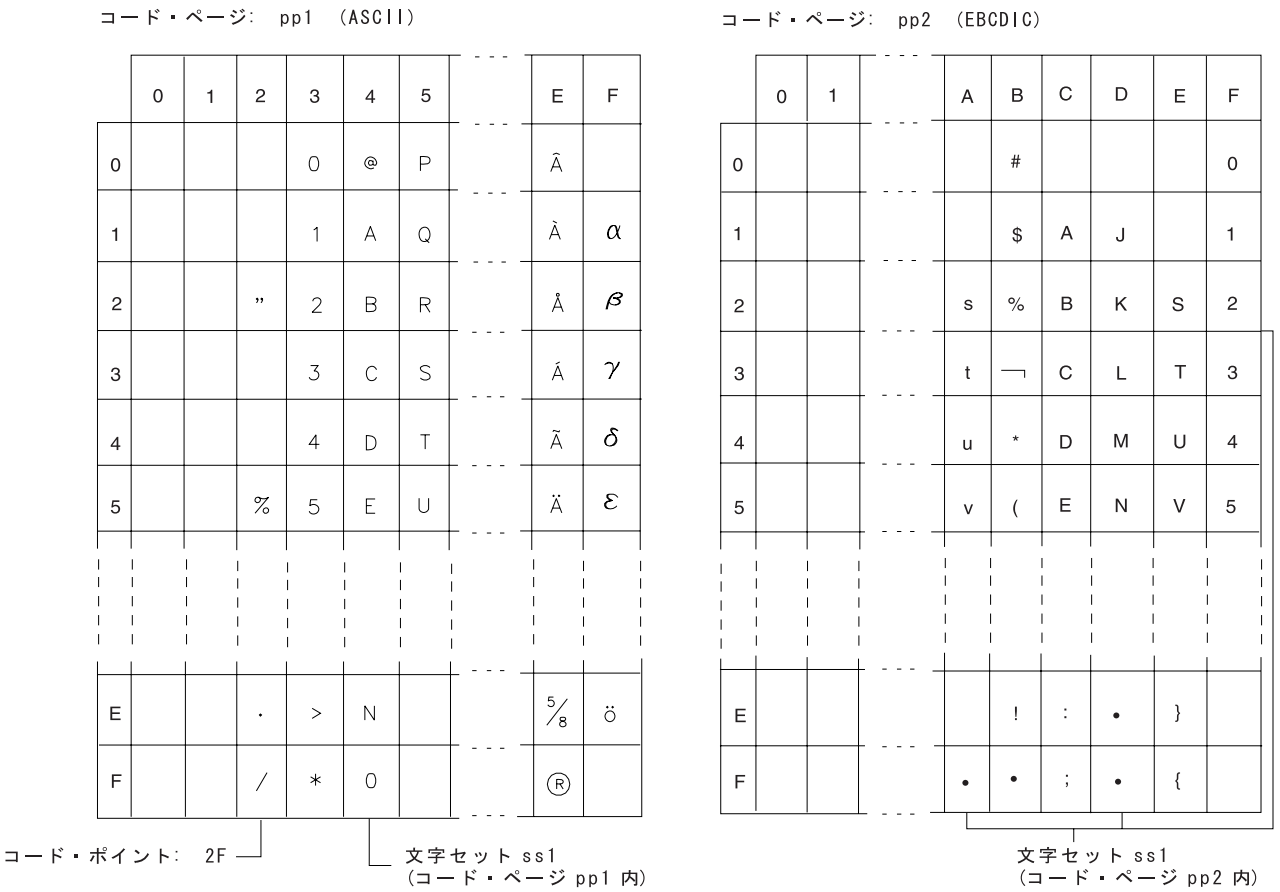
- 1 バイト EBCDIC
- 1 バイト ASCII[7]
- 1 バイト EBCDIC および 2 バイト EBCDIC 混合
- 1 バイト ASCII および 2 バイト ASCII 混合
- UCS-2 (汎用コード化文字セット)

置換文字
文字変換で、ソースのコード表示の文字に対応する文字が、ターゲットのコード表示に存在しない場合に、その文字に置き換わる固有の文字。

文字セットとコード・ページ

次の例は、典型的な文字セットが、2 つの異なるコード・ページの種々のコード・ポイントにどのようにマップされるかを示しています。

7. ASCII という用語は、本書全体を通して、IBM-PC データまたは ISO 8 データを指すのに使用されています。



同一のエンコード・スキームを使用している場合でも、異なるコード化文字セットが数多くあります。また、同一のコード・ポイントが別のコード化文字セットでは異なる文字を表すことがあります。さらに、文字ストリング中の 1 バイトが、必ずしも 1 バイト文字セット (SBCS) にある 1 文字を表すとは限りません。さらに、文字ストリングは、混合データ (1 バイト文字と 2 バイト文字の混合) や、どのような文字セットにも関連しないデータ (ビット・データと呼ばれる) にも使用されます。これはグラフィック・ストリングの場合には該当しません。その理由は、データベース・マネージャーでは、グラフィック・ストリングの場合には常に、そのバイトの対のすべてが、2 バイト文字セット (DBCS) または汎用コード化文字セット (UCS-2) の文字を表しているものと想定しているためです。

固有エンコード・スキームの CCSID は、データをそのサイトで保管できるコード化文字セットの 1 つです。外部エンコード・スキームの CCSID は、データをそのサイトで保管できないコード化文字セットの 1 つです。例えば、DB2 UDB for iSeries は、データを EBCDIC エンコード・スキームをもつ CCSID には保管できますが、ASCII エンコード・スキームには保管できません。

外部エンコード・スキームのデータを含むホスト変数は、関数または選択リストの中で使用される場合は、常に固有エンコード・スキームの CCSID に変換されます。また、外部エンコード・スキームのデータを含むホスト変数は、比較またはストリングを組み合わせる操作の中で使用される場合も、固有エンコード・スキームの CCSID に効果的に変換されます。データが固有エンコード・スキームのどの CCSID に変換されるかは、外部 CCSID およびデフォルト CCSID によって決まります。

コード化文字セットと CCSID

IBM の文字データ表現体系 (CDRA) では、ストリング表現およびコード化の差異に対処しています。この体系のキー・エレメントには、コード化文字セットID (CCSID) があります。CCSID は、2 バイト (無符号) の 2 進数で、文字セットとコード・ページの 1 つまたは複数の対およびエンコード・スキームを固有のものとして識別します。

長さがストリングの属性であるのとまったく同じように、CCSID はストリングの属性です。1 つのストリング列にあるすべての値は、同一の CCSID を持ちます。

それぞれのデータベース・マネージャーでは、文字変換のために *CCSID 変換選択表* を使用します。変換選択表には、ソースとターゲットの有効な組み合わせのリストが入っています。変換選択表には、CCSID の対ごとに、あるコード化文字セットを他のコード化文字セットに変換するのに使用する情報が入っています。この情報には、変換が必要かどうかを示す標識も含まれています。(対象となるストリングがそれぞれ異なる CCSID を持っていても、変換が不要な場合もあります。)

デフォルト CCSID

すべてのサーバーおよびアプリケーション・リクエスターには、デフォルト CCSID が 1 つあります (DBCS データをサポートするシステムには、複数のデフォルト CCSID があります)。以下のタイプのストリングの CCSID は、現行サーバーで決まります。

- ソースの CCSID が外部エンコード・スキームである場合のストリング定数 (日付/時刻の値を表すストリング定数を含む)
- ストリングの値をもつ特殊レジスター (USER や CURRENT SERVER など)
- CAST、CHAR、DIGITS、および HEX スカラー関数の結果[8]
- CCSID が引き数として定義されていない場合の VARCHAR、GRAPHIC、および VARGRAPHIC スカラー関数の結果
- CCSID が引き数として定義されていない場合の CLOB および DBCLOB スカラー関数の結果
- CREATE TABLE または ALTER TABLE ステートメントで定義されているストリング列で、列について CCSID が明示指定されていない場合[9]

分散 SQL プログラムでは、アプリケーション要求側によってホスト変数のデフォルト CCSID が決まります。非分散 SQL プログラムでは、ホスト変数のデフォルト CCSID はサーバーによって決定されます。OS/400 では、デフォルト CCSID は CCSID ジョブ属性によって決定されます。 CCSID の詳細については、iSeries Information Center のグローバリゼーション・セクションの中の CCSID の処理トピックを参照してください。

---

8. デフォルト CCSID が 65535 であり、関数が CLOB または DBCLOB への CAST である場合、CCSID として、DFTCCSID ジョブ属性の値が使用されます。

9. デフォルト CCSID が 65535 の場合、文字ストリング列は 65535 を使用しません。代わりに、CCSID として DFTCCSID ジョブ属性の値が使用されます。

ソート順序

ソート順序は、文字セット中の文字の比較や順序付けを行う場合に、文字が互いにどのような関係になっているかを定義するものです。ある特定の言語にしたがってデータの順序付けを行う場合は、別のソート順序を使用するのが便利です。例えば、リストをある特定の言語で通常見られる順序でリストすることができます。ソート順序を使用して、ある文字 (例えば、**a** と **A**) を同等として扱うこともできます。ソート順序は、以下のものを含むすべての比較で機能します。

- SBCS 文字データ (ビット・データを含む)
- 混合データの SBCS 部分
- UCS-2 グラフィック・データ

SBCS ソート順序は、256 バイトの表を使用してサポートされています。この表では、それぞれのバイトが 1 つのコード・ポイントまたは SBCS コード・ページ内の文字に対応しています。ソート順序は文字データに適用されるので、表には CCSID を関連付けておかなければなりません。ソート順序表中のバイトは、各コード・ポイントとそのコード・ページ内の他のコード・ポイントとの対比に基づいて設定されています。例えば、文字 **a** と **A** を比較の際に同等として扱いたい場合には、ソート順序表のこれらのコード・ポイントに対応するバイトには同じ値、すなわち、同じ重みが入ります。

UCS-2 ソート順序は、マルチバイトの表を使用してサポートされています。表内の一対のバイトが、UCS-2 コード・ページの 1 文字に対応します。 UCS-2 の数千ある文字の 1 つのサブセットだけが、代表して表に表されます。比較して異なる文字だけが (おそらく、同じ区分内の他の文字も)、表に表されます。ソート順序表中のバイトは、それぞれの文字と UCS-2 内の他の文字との対比に基づいて設定されます。

ソート順序表の複数のバイト (あるいは、UCS-2 の場合は一対のバイト) に同じ値が入っている場合、そのソート順序は共用重みソート順序です。ソート順序表中のすべてのバイト (あるいは、UCS-2 の場合は一対のバイト) が固有の値を持つソート順序は、固有重みソート順序です。システムでは、多くの言語に対応する固有重みおよび共用重みのソート順序が、オペレーティング・システムの一部として出荷されています。他の言語や要件に対応するソート順序が必要な場合には、表作成 (CRTTBL) コマンドを用いてそのソート順序を定義してください。

ソート順序によってデータ自体が変わるわけではないことを覚えておいてください。データの重み付け表現は、比較に使用されます。SQL では、ソート順序は CRTSQLxxx、STRSQL、および RUNSQLSTM コマンドで指定します。 SET OPTION ステートメントを使用して、組み込み SQL を含むプログラムのソース内にソート順序を指定することができます。指定されたソート順序は、SQL ステートメントで実行されるすべての文字比較に適用されます。システムのデフォルトのソート順序は、文字の 16 進表示を使用する際に生じる内部順序です。これは、SRTSEQ(*HEX) を指定した場合の順序です。バージョン 2 リリース 3 より前のプロダクトのリリースによってプリコンパイルされたプログラムの場合、ソート順序は *HEX になります。

ソート順序は、FOR BIT DATA 列や BLOB 列には適用されません。

CCSID の詳細については、iSeries Information Center のグローバリゼーション・セクションの中の CCSID の処理トピックを参照してください。 ソート順序、およびシステムと共に出荷されるソート順序については、iSeries Information Center の ソート順序表のトピックを参照してください。

権限と特権

指定した機能を行う権限がある場合にのみ、SQL ステートメントを正しく実行することができます。表を作成するには、表を作成する権限が必要であり、表を除去するには、表を除去する権限が必要であるという次第です。

管理権限を持つ担当者は、データベース・マネージャーを制御する作業を担当し、データの保全性や整合性について責任を持ちます。管理権限を持つ担当者は、データベース・マネージャーにアクセスできるユーザーとそのアクセスの範囲の両方を管理します。管理権限を持つ担当者は、特定の特権が与えられているか否かに関係なく、すべてのオブジェクトに対してすべての操作を行うことができる権限を持っています。セキュリティー担当者、および *ALLOBJ 権限を持つすべてのユーザーは、管理権限を持っています。

特権 とは、管理権限によってユーザーに許されている活動を指します。権限を持つユーザーは、任意のオブジェクトを作成し、ユーザー自身が所有するオブジェクトにアクセスし、また GRANT ステートメントを使用して、そのユーザー自身が所有するオブジェクトに関する特権を他のユーザーに与えることができます。REVOKE ステートメントを使用して、以前に与えた特権を取り消すことができます。

オブジェクトを作成すると、1 つの権限 ID にそのオブジェクトの*所有権* が割り当てられます。所有権を持つユーザーは、オブジェクトを完全に管理することができます (オブジェクトを除去する特権も持ちます)。所有者は、自分自身から所有するオブジェクトに対する特権を取り消すことができます。この場合、その所有者は、その特権を必要とする操作を一時的に行うことができなくなります。ただし、所有者なので、常に、その特権を自分自身に復権することができます。

SQL オブジェクトの *PUBLIC に認可される権限は、オブジェクトを作成した時に使用した命名規則に依存します。 *SYS 命名規則を使用している場合には、*PUBLIC はそのオブジェクトが作成されたライブラリーの作成権限 ((CRTAUT)) を獲得します。 *SQL 命名規則を使用した場合は、*PUBLIC は *EXCLUDE 権限を獲得します。

本書の権限の項では、オブジェクトの所有者が、その作成以降にオブジェクトからどのような特権も取り消していないことを前提にしています。オブジェクトがビューの場合には、そのビューが、直接、または間接に従属する表やビューからシステム権限の *READ を取り消していないことを前提にしています。所有者は、そのビューの定義で参照されている表やビューのすべてに対してシステム権限の *READ を持ち、またあるビューが参照されている場合には、そのビューの定義で参照されている表やビューのすべてに対してシステム権限の *READ を持ちます。以下同様です。権限と特権についての詳細は、「iSeries 機密保護解説書」  を参照してください。

記憶構造

iSeries システムは、オブジェクト・ベースのシステムです。DB2 UDB for iSeries のすべてのデータベース・オブジェクト (例えば、表および索引) は、OS/400 のオブジェクトです。単一レベルの記憶管理機能がデータベースのすべての記憶を管理しているため、データベース特有の記憶構造 (例えば、表スペース) は不要です。

区分表または分散表により、異なるデータベース区画にまたがって、データを置くことができます。含まれる区画は、表の作成または変更時に指定されるノード・グループによって決まります。ノード・グループとは、1 つまたは複数の iSeries システムのことです。区分化されたマップは、それぞれのノード・グループに関連しています。区分化されたマップは、データベース・マネージャーが使用し、ノード・グループのどのシステムが所定のデータ行を格納するかを決めます。ノード・グループおよびデータ区分化についての詳細は、DB2 UDB for iSeries マルチ・システムを参照してください。

また、表には、外部ファイルに保管されたデータへのリンクを登録する列も含めることができます。これについてのメカニズムは、DataLink データ・タイプです。通常の表に記録された DataLink 値が、外部ファイル・サーバーに保管されたファイルを指しています。

ファイル・サーバー上の DB2 UDB ファイル・マネージャーが、DB2 UDB と共同で以下の任意選択の機能を提供します。

- 現在、DB2 UDB にリンクしているファイルが削除または名前変更されないようにするための参照保全
- DataLink 列で適切な SQL 特権を持ったものだけが、その列にリンクしたファイルを読み取ることができるようにするためのセキュリティー

DataLinker は、次の 2 つの機能から成っています。

DataLink ファイル・マネージャー
DB2 UDB にリンクした特定のファイル・サーバーのすべてのファイルを登録する。

DataLink フィルター
ファイル・システム・コマンドをフィルターに掛け、登録されたファイルが削除または名前変更されないように確認する。任意選択として、コマンドをフィルターに掛け、適切なアクセス権限のあることを確認する。

第 2 章 言語エレメント

この章では、SQL の基本構文および多くの SQL ステートメントに共通する言語エレメントを定義しています。

詳細については、以下のセクションを参照してください。

- 『文字』
- 43 ページの『トークン』
- 45 ページの『ID』
- 47 ページの『命名規則』
- 57 ページの『スキーマと SQL パス』
- 58 ページの『別名』
- 60 ページの『権限 ID と権限名』
- 62 ページの『データ・タイプ』
- 80 ページの『データ・タイプのプロモーション』
- 82 ページの『データ・タイプ間のキャスト』
- 85 ページの『割り当ておよび比較』
- 99 ページの『結果のデータ・タイプに関する規則』
- 103 ページの『ストリングを結合する演算に適用される変換規則』
- 105 ページの『定数』
- 111 ページの『特殊レジスター』
- 121 ページの『変数に対する参照』
- 127 ページの『C、C++、COBOL、PL/I、および RPG におけるホスト構造』
- 129 ページの『C、C++、COBOL、PL/I、および RPG におけるホスト構造配列』
- 130 ページの『関数』
- 136 ページの『式』
- 153 ページの『述部』

文字

SQL 言語のキーワードや演算子の基本的な記号は、IBM のリレーショナル・データベース製品によってサポートされるすべての文字セットの一環である 1 バイト文字[10]を使用しています。言語で使用される文字は、文字、数字、あるいは特殊文字に類別されます。

10. SQL ステートメントが UCS-2 データとしてコード化されている場合は、ストリング定数を除くステートメントのすべての文字が処理の前に単一バイト文字に変換されることに注意してください。ストリング定数を表すトークンは UCS-2 グラフィック・ストリングとして処理され、単一バイトには変換されません。

文字

文字 とは、英語のアルファベットの 26 の大文字 (A 〜 Z) と 26 の小文字 (a 〜 z) の任意の文字を指します。[11]

数字 は、0 から 9 までの任意の文字です。

特殊文字 は、次に示す文字のいずれかです。[12]

	スペース	-	負符号
″	引用符または二重引用符	.	ピリオド
%	パーセント	/	斜線 (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	<	より小さい
)	右括弧	=	等号
*	アスタリスク	>	より大きい
+	正符号	?	疑問符
,	コンマ	_	下線
| [13]	縦線		

11. 文字には、各国言語用にアルファベットの拡張として 3 つのコード・ポイントが含まれています (米国の場合、#、@、および $)。これらの 3 つのコード・ポイントは、CCSID によって異なる文字を表すので、使用を避けてください。

12. 否定の記号 (¬) および感嘆符 (!) も、DB2 UDB for iSeries によって使用される特殊文字です。これらの記号は可変文字なので、使用を避けてください。

13. 縦線 (|) 文字の使用は、IBM リレーショナル・データベース製品間のコードの移植性を阻害することがあります。連結記号 (||) の代わりに、CONCAT 演算子を使用することをお勧めします。縦線は可変文字であるため、使用しないようにしてください。

トークン

言語の基本的な構文単位のことを、*トークン* と呼びます。 1 つのトークンは、1 つまたは複数の文字から構成されます。ただし、この文字にはブランクや制御文字は入りません。また、ストリング定数または区切り文字付き ID 内の文字も除きます。(これらの用語については後述します。)

トークンは、*通常トークン* と*区切りトークン* 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 、またキーワードを指します。
- *区切りトークン* とは、ストリング定数、区切り文字付きID、演算記号、または構文図に示される任意の特殊文字を指します。疑問符 (?) も、 725 ページの『PREPARE』で説明しているようなパラメーター・マーカーとして使用される場合は、区切りトークンとなります。

スペース: *スペース* とは、1 つまたは複数のブランク文字を並べたものです。

制御文字: *制御文字* は、ストリングの位置合わせに使用される特殊文字です。次の表は、データベース・マネージャーが取り扱う制御文字を示しています。

表 1. 制御文字

制御文字	EBCDIC 16 進値	UCS-2 16 進値
タブ	05	0009
用紙送り	0C	000C
復帰	0D	000D
改行	15	0085
行送り (改行)	25	000A

ストリング定数および特定の区切り文字付き ID 以外のトークンには、制御文字またはスペースを含めてはなりません。制御文字またはスペースは、トークンの後ろに続けることができます。区切りトークン、制御文字、またはスペースが、すべての通常トークンの後に続かなければなりません。構文上、通常トークンの後に区切りトークンを続けることが許されない場合には、その通常トークンの後に制御文字またはスペースを続ける必要があります。ここで述べた規則について、以下に例を示します。

次に示すのは、通常トークンの例です。

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

上記の通常トークンをいくつか組み合わせると、トークンが事実上変化してしまいます。その例を次に示します。

```
1.1       .1+2       SELECTE       .1E       E3       SELECT1
```

このため、通常トークンの後には必ず区切りトークンまたはスペースを置かなければなりません。

次に示すのは、区切りトークンの例です。

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

上記の通常トークンと区切りトークンを組み合わせると、トークンが事実上変化してしまうことがあります。この例を次に示します。

```
1.        .3
```

名前の修飾でピリオド (.) を 区切り記号として使用すると、そのピリオドは区切りトークンになります。上記の例では、ピリオドを通常トークンの数値定数と組み合わせて使用しています。このため、通常トークンの後に区切りトークンを置くことは構文上許されません。このような場合は、通常トークンの後に、区切りトークンではなくスペースを置かなければなりません。

108 ページの『小数点』で説明するように、小数点がコンマとして定義されている場合は、コンマが小数点として解釈されます。この場合の数値定数の例を、次に示します。

```
1,2       ,1       1,       1,e1
```

'1,2' および '1,e1' がそれぞれ 2 つの項目を表す場合には、コンマが小数点として解釈されるのを防ぐために、通常トークン (1) の後と区切りトークン (,) の後の両方にスペースを置かなければなりません。コンマは、通常は区切りトークンですが、小数点として解釈される場合には数値の一部となります。したがって、通常トークン (1) の後に区切りトークン (,) を続けることは構文上許されません。このような場合は、通常トークンの後に区切りトークンではなくスペースを置く必要があります。

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 のコメントを含めることができます。動的 SQL ステートメントの中に SQL コメントを組み込むことができます。どちらのタイプのコメントも、スペースを指定できる場所であればどこでも指定できますが、区切りトークンの中またはキーワードの EXEC と SQL の間は例外です。SQL コメントには、次の 2 つのタイプがあります。

単純コメント

単純コメントは、2 つの連続するハイフン (--) で始まります。単純コメントは、その行の終わりを超えて続けることはできません。詳細については、375 ページの『SQL のコメント』を参照してください。

括弧付きのコメント

括弧付きのコメントは、/* で始まり、*/ で終わります。括弧付きのコメントは、その行の終わりを超えて続けることができます。詳細については、375 ページの『SQL のコメント』を参照してください。

大文字と小文字: C ホスト変数以外の通常トークンで使用される小文字は、大文字に変換されます。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の場合、

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

変換後は、以下のステートメントと同等になります。

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

ID

ID とは、名前を形成するのに使用するトークンを指します。SQL ステートメントの ID は、次のタイプの 1 です。

- 『SQL ID』
- 『システム ID』
- 46 ページの『ホスト ID』

注: $、@、#、および他のすべての可変文字は、それらの文字を表すのに使用するコード・ポイントがそれらの文字を含むストリングの CCSID によって変わるため、IDで使用してはなりません。これらの文字を使用すると、予期しない結果が起こる可能性があります。可変文字の詳細については、iSeries Information Center の可変文字トピックを参照してください。

SQL ID

SQL ID には、*通常 ID* と*区切り文字付き ID* の 2 つのタイプがあります。

- *通常 ID* の場合、1 つの大文字の後に、大文字、数字、または下線文字がゼロ個または複数続きます。通常 ID は、大文字に変換されるので注意してください。通常 ID は、予約語であってはなりません。予約語のリストについては、 893 ページの『付録 D. 予約語』を参照してください。予約語を SQL における ID として使用する場合は、大文字で指定し、SQL エスケープ文字で囲む必要があります。
- *区切り文字付き ID* は、1 つまたは複数の文字の並びを SQL エスケープ文字で囲んだものです。このシーケンスは、1 つまたは複数の文字で構成されていなければなりません。シーケンスの先行ブランクは、意味を持ちます。シーケンスの末尾のブランクは、意味を持ちません。 2 つの SQL エスケープ文字は、区切り文字付き ID の長さには含まれません。区切り文字付き ID は、大文字に変換されないので注意してください。エスケープ文字には引用符 (") を使用します。ただし、次のような場合は例外で、アポストロフィ (') をエスケープ文字として使用します。
  - 対話式 SQL で、SQL ストリング区切り文字が、COBOL 構文検査ステートメント・モードで引用符に設定されている場合。
  - COBOL プログラムにおける動的 SQL で、CRTSQLCBL または CRTSQLCBLI のパラメーター OPTION(*QUOTESQL) が、ストリング区切り文字を引用符 (") として指定している場合。
  - COBOL のアプリケーション・プログラムで、CRTSQLCBL または CRTSQLCBLI のパラメーター OPTION(*QUOTESQL) が、ストリング区切り文字を引用符 (") として指定している場合。

  区切り文字付き ID の中では、以下の文字は使用することはできません。
  - X'00' から X'3F' まで、および X'FF'

システム ID

システム ID は、OS/400 のシステム・オブジェクトの名前を形成するのに使用されます。2 つのタイプのシステム ID があります。すなわち、通常 ID と区切り文字付き ID です。

- 通常システム ID の形成に関する規則は、SQL 通常 ID の場合の規則と同じです。
- 区切り文字付きシステム ID の形成に関する規則は、以下の点を除き、SQL 区切り文字付き ID の場合の規則と同じです。
  - 次の特殊文字は、区切り文字付きシステム ID には使用できません。
    - ブランク (X'40')
    - アスタリスク (X'5C')
    - アポストロフィ (X'7D')
    - 疑問符 (X'6F')
    - 引用符 (X'7F')
  - エスケープ文字用に必要なバイト数は、その区切り文字内の文字が通常 ID を形成する場合を除き、その ID の長さに含まれます。

    例えば、“PRIVILEGES” は大文字であり、区切り文字で囲まれている文字が通常 ID を形成するので、長さが 10 バイトで、列の有効なシステム名になります。これに対して、“privileges” は小文字であり、区切り文字に必要なバイト数を ID の長さに含めなければならないため、長さが 12 バイトになり、列の有効なシステム名にはなりません。

例

```
WKLYSAL     WKLY_SAL     "WKLY_SAL"     "UNION"     "wkly_sal"
```

ホスト ID

ホスト ID とは、ホスト・プログラムで宣言されている名前を指します。 ホスト ID を形成する際の規則は、ホスト言語の規則に従います。ただし、ホスト ID には、DBCS 文字は使用できません。例えば、COBOL プログラムでホスト ID を形成する際の規則は、COBOL プログラムでユーザー定義語を形成する際の規則と同じです。プリコンパイラーでは、'SQ'[14]、'SQL'、'sql'、'RDI'、または 'DSN' という文字で始まるホスト変数を生成するため、これらの文字で始まる名前を使用してはなりません。

14. 'SQ' は、C、COBOL、および PL/I では使用できますが、RPG では使用できません。

命名規則

名前を形成する規則は、その名前によって指定されるオブジェクトのタイプと命名オプション (*SQL または *SYS) に依存します。命名オプションは、CRTSQLxxx、RUNSQLSTM、および STRSQL コマンドに指定します。SET OPTION ステートメントを使用して、組み込み SQL を含むプログラムのソース内に命名オプションを指定することができます。構文図では、名前のタイプに応じてさまざまな用語が使用されています。以下にそれらの用語の定義を示します。

別名

別名を示す修飾名または非修飾名です。別名の修飾形式は、命名オプションに依存します。SQL 命名規則の場合、修飾形式は、スキーマ名の後にピリオド (.) と SQL ID が続きます。システム命名規則の場合、修飾形式は、スキーマ名の後にスラッシュ (/) と SQL ID が続きます。

非修飾形式は、SQL ID です。非修飾形式は、54 ページの『非修飾オブジェクト名の修飾』で指定された規則に基づいて暗黙的に修飾されます。

別名では、別名の名前、または別名のシステム・オブジェクトの名前のいずれかを指定することができます。

権限名

ユーザー、またはユーザーのグループを指定するシステム ID。権限名は、サーバー上のユーザー・プロファイル名です。この名前は、小文字または特殊文字を含む区切り文字付き ID であってはなりません。権限名と権限 ID の相違については、60 ページの 『権限 ID と権限名』を参照してください。

列名

表またはビューの列を示す修飾名または非修飾名です。非修飾形式の列名は、SQL ID です。修飾形式の列名は、修飾名の後にピリオドと SQL ID を付けたものになります。この場合の修飾名は、表名、ビュー名、または相関名です。

COMMENT および LABEL ステートメントにおける場合を除き、*スキーマ名/表名.列名* という形式を使って列名をシステム名によって修飾することはできません。ステートメントで相関名を使用できる場合、列名を修飾する必要があれば、相関名を使用して列を修飾しなければなりません。

列名は、表、またはビューの列名、あるいはシステム列名を指定します。列名が区切られている場合、名前の長さを判別するときに、区切り文字は名前の一部と見なされます。

制約名

表についての制約を指定する修飾または非修飾の名前です。制約名の修飾形式は、命名オプションに依存します。 SQL 命名規則の場合、修飾形式は、スキーマ名の後にピリオド (.) と システム ID を付

けたものです。 システム命名規則の場合、修飾形式は、スキーマ名の後にスラッシュ (/) と SQL ID が続きます。

非修飾形式は、SQL ID です。非修飾形式は、 54 ページの『非修飾オブジェクト名の修飾』で指定された規則に基づいて暗黙的に修飾されます。

相関名

表、ビュー、または表やビューの個々の行を示す SQL ID です。

カーソル名

SQL カーソルを示す SQL ID です。

記述子名

コロンとその後に SQL 記述子域 (SQLDA) を指定するホスト ID です。ホスト ID の説明については、 121 ページの『ホスト変数に対する参照』を参照してください。SQL 記述子域を示すホスト変数には、標識変数を指定してはなりません。*:ホスト変数 :標識変数* という形式は使用できません。

特殊タイプ名

特殊タイプ名を示す修飾名または非修飾名です。特殊タイプ名の修飾形式は、命名オプションに依存します。 SQL 命名規則の場合、修飾形式は、スキーマ名の後にピリオド (.) と SQL ID が続きます。システム命名規則の場合、修飾形式は、スキーマ名の後にスラッシュ (/) と SQL ID が続きます。

非修飾形式は、SQL ID です。非修飾形式は、 54 ページの『非修飾オブジェクト名の修飾』で指定された規則に基づいて暗黙的に修飾されます。

システム命名規則では、特殊タイプ名は、SQL ルーチンのパラメーター・データ・タイプ内で使用されている場合、あるいは SQL 関数、SQL プロシージャー、またはトリガーの SQL 変数宣言内で使用されている場合は、修飾することはできません。

外部プログラム名

これは、外部プログラムを指す修飾名、非修飾名、または文字ストリングです。外部プログラム名の修飾形式は、命名オプションに依存します。 SQL 命名規則の場合、修飾形式は、スキーマ名の後にピリオド (.) と システム ID を付けたものです。 システム命名規則の場合、修飾形式は、スキーマ名の後にスラッシュ (/) とシステム ID が続きます。

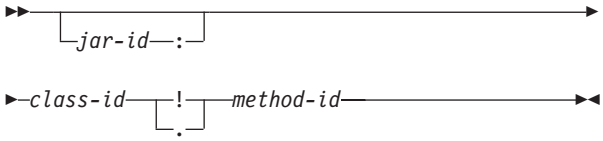
非修飾形式は、システム ID です。非修飾形式は、54 ページの『非修飾オブジェクト名の修飾』で指定された規則に基づいて暗黙的に修飾されます。

文字ストリングの形式は、次のいずれかです。

- OS/400 の修飾プログラム名 ('ライブラリー名/プログラム名')。
- 後に括弧で囲まれた OS/400 メンバー名を伴う OS/400 修飾ソース・ファイル名 ('ライブラリー

名/ソース・ファイル名 (メンバー名)')。この形式は、REXX プロシージャーを呼び出す場合にのみ有効です。

- 後に括弧で囲まれた OS/400 エントリー・ポイント名を伴う OS/400 修飾サービス・プログラム名 ('ライブラリー名/サービス・プログラム名 (エントリー・ポイント名)')。この形式は、関数の場合にのみ有効です。
- 任意選択の jar-id の後に、クラス ID、その後に感嘆符またはピリオド、その後にメソッド ID ('class-id!method-id' または 'class-id.method-id') を付けたものです。



jar-id は、jar スキーマを識別します (データベースにインストールされた場合)。これは、単純な ID またはスキーマ修飾 ID のどちらも可能です。例えば、'myJar' や 'myCollection.myJar' のようになります。
クラス ID は、Java オブジェクトのクラス ID を識別します。クラスがパッケージの一部である場合は、クラス ID には完全なパッケージ・プレフィックスが含まれていなければなりません。例えば、クラス ID が 'myPackage.StoredProcs' である場合、Java 仮想計算機は、以下のディレクトリー内で StoredProcs クラスを検索します。

```
'/QIBM/UserData/OS400/SQLLib/
  Function/myPackage/StoredProcs/'
```

メソッド ID は、呼び出される Java オブジェクトのメソッド名を識別します。

この形式は、Java プロシージャーおよび Java 関数の場合にのみ有効です。

関数名

ユーザー定義の関数、特殊タイプが作成されたときに生成されたキャスト関数、または組み込み関数を指す修飾名または非修飾名です。関数名の修飾形式は、命名オプションに依存します。SQL 命名規則の場合、修飾形式は、スキーマ名の後にピリオド (.) と SQL ID が続きます。システム命名規則の場合、修飾形式は、スキーマ名の後にスラッシュ (/) と SQL ID が続きます。

非修飾形式は、SQL ID です。非修飾形式は、54 ページの『非修飾オブジェクト名の修飾』で指定された規則に基づいて暗黙的に修飾されます。

システム命名規則の場合、
CREATE、COMMENT、DROP、GRANT、または
REVOKE ステートメントで名前を使用する場合
に、関数名をスキーマ名/関数名の形式でのみ、修飾
することができます。

ホスト・ラベル
ホスト・プログラムのラベルを示すトークンです。

ホスト変数
ホスト変数を示す一連のトークンです。 121 ページの『ホスト変数に対する参照』で説明されているように、1 つのホスト変数は少なくとも 1 つのホスト ID を持ちます。

索引名
索引を示す修飾名または非修飾名。索引名の修飾形式は、命名オプションに依存します。 SQL 命名規則の場合、修飾形式は、スキーマ名の後にピリオド (.) と SQL ID が続きます。システム命名規則の場合、修飾形式は、スキーマ名の後にスラッシュ (/) と SQL ID が続きます。

非修飾形式は、SQL ID です。非修飾形式は、 54 ページの『非修飾オブジェクト名の修飾』で指定された規則に基づいて暗黙的に修飾されます。

ノード・グループ名
ノード・グループを示す修飾名または非修飾名です。ノード・グループは、表が配布される iSeries サーバーすべてに及ぶグループを指します。分散表およびノード・グループに関する詳細については、「DB2 UDB for iSeries マルチ・システム」を参照してください。

ノード・グループ名の修飾形式は、命名オプションに依存します。SQL 命名規則の場合、修飾形式は、スキーマ名の後にピリオド (.) と システム ID を付けたものです。 システム命名規則の場合、修飾形式は、スキーマ名の後にスラッシュ (/) とシステム ID が続きます。

非修飾形式は、システム ID です。非修飾形式は、54 ページの『非修飾オブジェクト名の修飾』で指定された規則に基づいて暗黙的に修飾されます。

パッケージ名
パッケージを示す修飾名または非修飾名です。パッケージ名の修飾形式は、命名オプションに依存します。 SQL 命名規則の場合、修飾形式は、スキーマ名の後にピリオド (.) と システム ID を付けたものです。システム命名規則の場合、修飾形式は、スキーマ名の後にスラッシュ (/) とシステム ID が続きます。

非修飾形式は、システム ID です。非修飾形式は、54 ページの『非修飾オブジェクト名の修飾』で指定された規則に基づいて暗黙的に修飾されます。

パラメーター名

関数またはプロシージャーのパラメーターを指す通常 ID です。プロシージャー用のパラメーターの場合、IDはコロンの後に続きます。

プロシージャー名

プロシージャーを指す修飾名または非修飾名です。プロシージャー名の修飾形式は、命名オプションに依存します。SQL 命名規則の場合、修飾形式は、スキーマ名の後にピリオド (.) と SQL ID が続きます。システム命名規則の場合、修飾形式は、スキーマ名の後にスラッシュ (/) と SQL ID が続きます。

非修飾形式は、SQL ID です。非修飾形式は、 54 ページの『非修飾オブジェクト名の修飾』で指定された規則に基づいて暗黙的に修飾されます。

保管ポイント名

保管ポイントを指す非修飾 ID です。

スキーマ名

SQL オブジェクトを論理的にグループ化するための修飾名または非修飾名です。スキーマは、表、ビュー、索引、プロシージャー、関数、トリガー、制約、別名、タイプ、またはパッケージの修飾子として使用されます。スキーマ名の非修飾形式は、システム ID です。スキーマ名の修飾形式は、命名オプションに依存します。

SQL 名を使用している場合、SQL ステートメント中の非修飾のスキーマ名は、サーバー名によって暗黙のうちに修飾されます。修飾形式は、サーバー名の後に、(.) およびシステム ID を付けたものです。 このサーバー名は、現行サーバーを識別するものでなければなりません。

システム名を使用している場合、SQL ステートメント中の非修飾のスキーマ名は、サーバー名によって暗黙のうちに修飾されます。修飾形式は、サーバー名の後に、スラッシュ (/) およびシステム ID が続きます。このサーバー名は、現行サーバーを識別するものでなければなりません。

注: スキーマ名は、CREATE SCHEMA ステートメントによって作成されたスキーマ、あるいは OS/400 ライブラリーのいずれかを指します。

サーバー名

サーバーを示す SQL ID です。この ID には、小文字や特殊文字を使用してはなりません。

特定名

プロシージャーまたは関数を一意的に識別する修飾名または非修飾名です。特定名の修飾形式は、命名オプションに依存します。 SQL 命名規則の場合、修飾形式は、スキーマ名の後にピリオド (.) と SQL ID が続きます。システム命名規則の場合、修飾形式は、スキーマ名の後にスラッシュ (/) と SQL ID が続きます。

非修飾形式は、SQL ID です。非修飾形式は、 54 ページの『非修飾オブジェクト名の修飾』で指定された規則に基づいて暗黙的に修飾されます。

SQL ラベル

SQL プロシージャー、SQL 関数、またはトリガー本体のラベルを示す非修飾名です。SQL ラベル名は、SQL ID です。

SQL パラメーター名

SQL ルーチン本体のパラメーターを示す修飾名または非修飾名です。非修飾形式の SQL パラメーター名は、SQL ID です。修飾形式は、プロシージャー名の後にピリオド (.) と SQL ID が続きます。

SQL 変数名

SQL ルーチン本体の変数を示す修飾名または非修飾名です。非修飾形式の SQL 変数名は、SQL ID です。修飾形式は、SQL ラベルの後にピリオド (.) と SQL ID が続きます。

ステートメント名

準備済み SQL ステートメントを示す SQL ID です。

システム列名

表またはビューの OS/400 列名を示す非修飾名です。システム列名はシステム ID です。システム列名は、区切り文字付き ID でも構いませんが、区切り文字で囲まれた内部に小文字や特殊文字が入っていてはなりません。

システム・オブジェクト名

表、ビュー、索引、または別名の OS/400 名を示す非修飾名です。システム・オブジェクト名はシステム ID です。

表、ビュー、索引、またはビューの非修飾名が有効なシステム ID である場合、システム・オブジェクト名は、表、ビュー、索引、またはビューの非修飾名になります。

表名

表を示す修飾名または非修飾名です。表名の修飾形式は、命名オプションに依存します。 SQL 命名規則の場合、修飾形式は、スキーマ名の後にピリオド (.) と SQL ID が続きます。システム命名規則の場合、修飾形式は、スキーマ名の後にスラッシュ (/) と SQL ID が続きます。

非修飾形式は、SQL ID です。非修飾形式は、 54 ページの『非修飾オブジェクト名の修飾』で指定された規則に基づいて暗黙的に修飾されます。

表名では、表の名前、または表のシステム・オブジェクトの名前のいずれかを指定することができます。

トリガー名

表に対するトリガーを指定する修飾または非修飾の名前です。トリガー名の修飾形式は、命名オプションに依存します。 SQL 命名規則の場合、修飾形式は、スキーマ名の後にピリオド (.) と システム ID

を付けたものです。 システム命名規則の場合、修飾形式は、スキーマ名の後にスラッシュ (/) と SQL ID が続きます。

非修飾形式は、SQL ID です。非修飾形式は、 54 ページの『非修飾オブジェクト名の修飾』で指定された規則に基づいて暗黙的に修飾されます。

ビュー名

ビューを示す修飾名または非修飾名です。ビュー名の修飾形式は、命名オプションに依存します。 SQL 命名規則の場合、修飾形式は、スキーマ名の後にピリオド (.) と SQL ID が続きます。システム命名規則の場合、修飾形式は、スキーマ名の後にスラッシュ (/) と SQL ID が続きます。

非修飾形式は、SQL ID です。非修飾形式は、 54 ページの『非修飾オブジェクト名の修飾』で指定された規則に基づいて暗黙的に修飾されます。

ビュー名では、ビューの名前、またはビューのシステム・オブジェクトの名前のいずれかを指定することができます。

表 2. IDの長さの制約 (バイト数)

ID のタイプ	最大の長さ
別名	128
権限名	10
相関名	128
カーソル名	18
ホスト ID	64
保管ポイント名	128
サーバー名	18
SQL ラベル	128
ステートメント名	18
非修飾のスキーマ名	10
非修飾の列名	30
非修飾の制約名	128
非修飾の特殊タイプ名	128
非修飾の外部プログラム名 [15]	10
非修飾の関数名	128
非修飾のノード・グループ名	10
非修飾のパッケージ名	10
非修飾のパラメーター名	128
非修飾のプロシージャー名	128
非修飾の特定名	128
非修飾の SQL パラメーター名	128
非修飾の SQL 変数名	128

表 2. IDの長さの制約 (バイト数) (続き)

ID のタイプ	最大の長さ
非修飾のシステム列名	10
非修飾のシステム・オブジェクト名	10
非修飾の表、ビュー、および索引名	128
非修飾のトリガー名	128

非修飾オブジェクト名の修飾

非修飾オブジェクト名は暗黙的に修飾されます。名前を修飾するための規則は、その名前が識別するオブジェクトのタイプによって異なります。

非修飾の別名、制約名、外部プログラム名、索引名、ノード・グループ名、パッケージ名、表名、トリガー名、およびビュー名

非修飾の別名、制約名、外部プログラム名、索引名、ノード・グループ名、パッケージ名、表名、トリガー名、およびビュー名は、以下のように暗黙的に修飾されます。

- 静的 SQL ステートメントの場合
  - DFTRDBCOL パラメーターが CRTSQLxxx コマンド (または、SET OPTION ステートメントで) 指定されている場合は、暗黙的な修飾子はパラメーターに指定されたスキーマ名。
  - その他の場合はすべて、暗黙的な修飾子は命名規則に基づく。
    - SQL 命名規則の場合は、暗黙的な修飾子はそのステートメントの権限ID。
    - システム命名規則の場合は、暗黙的な修飾子は、ジョブ・ライブラリー・リスト (*LIBL)。
- 動的 SQL ステートメントの場合、暗黙的な修飾子はデフォルトのスキーマ名が明示的に指定されているかどうかによって異なります。明示的にこれを指定するメカニズムは、SQL ステートメントを動的に作成し、実行するために使用されるインターフェースにより異なります。
  - デフォルトのスキーマ名が明示的に指定されていない場合
    - SQL 命名規則の場合は、暗黙的な修飾子は実行時のID。
    - システム命名規則の場合は、暗黙的な修飾子は、ジョブ・ライブラリー・リスト (*LIBL)。
  - デフォルトのスキーマ名が明示的に指定されている場合は、暗黙的な修飾子はそのデフォルトのスキーマ。デフォルトのスキーマ名は、以下のインターフェースによって指定可能。

15. REXX プロシージャーでの限界は 33 です。

表 3. デフォルトのスキーマ・インターフェース

SQL インターフェース	指定
組み込み SQL	SQL プログラム作成 (CRTSQLxxx) コマンドおよび SQL パッケージ作成 (CRTSQLPKG) コマンドの DFTRDBCOL パラメーターおよび DYNDFTCOL(*YES)。 DFTRDBCOL および DYNDFTCOL の値の設定に SET OPTION ステートメントも使用可能。 (CRTSQLxxx コマンドの詳細については、「DB2 UDB for iSeries ホスト言語での SQL プログラミング」を参照)。
SQL ステートメント実行	SQL ステートメント実行 (RUNSQLSTM) コマンドの DFTRDBCOL パラメーター。 (RUNSQLSTM コマンドの詳細については、「SQL プログラミング 概念」を参照)。
サーバー上の呼び出しレベル・インターフェース (CLI)	SQL_ATTR_DEFAULT_LIB または SQL_ATTR_DBC_DEFAULT_LIB 環境変数または接続変数。 (CLI の詳細については、「DB2 UDB for iSeries SQL 呼び出しレベル・インターフェース」を参照)。
Developer Kit for Javaを使用したサーバーの JDBC または SQLJ	ライブラリー特性オブジェクト (JDBC および SQLJ の詳細については、iSeries Information Center の IBM Developer Kit for Java トピックを参照)。
iSeries Access ODBC ドライバーを使用したクライアントの ODBC	ODBC セットアップ内の SQL デフォルト・ライブラリー。 (ODBC の詳細については、iSeries Information Center の iSeries Access カテゴリーを参照)。
IBM Toolbox for Java を使用したクライアントの JDBC	JDBC セットアップ内の SQL デフォルト・ライブラリー。 (JDBC の詳細については、iSeries Information Center の iSeries Access カテゴリーを参照)。 (IBM Toolbox for Java の詳細については、iSeries Information Center の IBM Toolbox for Java トピックを参照)。
すべてのインターフェース	SET SCHEMA または QSQCHGDC (動的デフォルト・コレクション変更) API (QSQCHGDC の詳細については、iSeries Information Center のファイル API カテゴリーを参照)。

非修飾の関数名、プロシージャー名、特定名、および特殊タイプ名

データ・タイプ名 (組み込みタイプと特殊タイプの両方とも)、関数名、プロシージャー名、および特定名は、非修飾の名前が使われている SQL ステートメントによって異なります。

- 非修飾名が CREATE、COMMENT、DROP、GRANT、または REVOKE ステートメントのメイン・オブジェクトの場合は、非修飾の表名の修飾と同じ規則を使用

して暗黙的に名前が修飾されます (54 ページの『非修飾の別名、制約名、外部プログラム名、索引名、ノード・グループ名、パッケージ名、表名、トリガー名、およびビュー名』を参照してください)。

- それ以外の場合は、暗黙的なスキーマ名は以下のようにして決められます。
  - 特殊タイプ名の場合、データベース・マネージャーは SQL パスを検索し、そのデータ・タイプが存在するような、パス上の最初のスキーマを選択する。
  - プロシージャー名の場合、データベース・マネージャーは SQL パスを検索し、同じ名前とパラメーター数を持つような、パス上の最初のスキーマを選択する。
  - 関数名とソースとなる関数に指定された特定名の場合、データベース・マネージャーは、 131 ページの『関数解決』で説明しているように、関数解決と連携して SQL パスを使用する。

SQL 名とシステム名 : 特殊な考慮事項

CL コマンドのデータベース・ファイル一時変更 (OVRDBF) を指定すると、ローカル・データ操作用の SQL ステートメントについて、SQL 名またはシステム名を他のオブジェクト名に一時変更することができます。この一時変更は、データ定義用の SQL ステートメントおよびリモートのリレーショナル・データベースで実行されるデータ操作用 SQL ステートメントについては、無視されます。一時変更関数についての詳細は、ファイル管理を参照してください。

スキーマと SQL パス

SQL パスとは、スキーマ名が順に並べられたリストです。データベース・マネージャーは、パスを使用して、CREATE、DROP、COMMENT、GRANT または REVOKE ステートメントのメイン・オブジェクトとして使われるもの以外に、文脈に現れる非修飾特殊タイプ名 (組み込みタイプと特殊タイプの両方) 、関数名、およびプロシージャー名のスキーマ名を解決します。 データベース・マネージャーは、パスを左から右に検索して、同じ非修飾名で同じオブジェクトを持つ、パス上の最初のスキーマ名にオブジェクト名を暗黙的に修飾します。プロシージャーの場合、データベース・マネージャーは、一致したプロシージャー名のうちパラメーター数も同じものだけを選択します。関数の場合、データベース・マネージャーは関数解決と呼ばれるプロセスを使用して、同じ名前の関数がスキーマ内にいくつか存在する可能性があるため、SQL パスと連携して選択すべき関数を決めます。 (詳細については、 131 ページの『関数解決』を参照してください。)

例えば、SQL パスが SMITH、XGRAPHIC、QSYS、QSYS2 で、非修飾の特殊タイプ名 MYTYPE が指定された場合、データベース・マネージャーは MYTYPE を最初にスキーマ SMITH で、次に XGRAPHIC で、その次に QSYS と QSYS2 で探します。

使用されるパスは、以下のようにして決められます。

- 静的 SQL ステートメント (CALL *:ホスト変数* ステートメントを除く) の場合、使用されるパスは CRTSQLxxx コマンドの SQLPATH パラメーターで指定します。 SQLPATH は、SET OPTION ステートメントを使用しても設定することができます。
- 動的 SQL ステートメントの場合 (さらには、CALL *:ホスト変数* ステートメント呼び出しの場合)、使用されるパスは、CURRENT PATH 特殊レジスターで指定されたパスで使用するパスです。 CURRENT PATH 特殊レジスターの詳細については、 111 ページの『CURRENT PATH、CURRENT_PATH、または CURRENT FUNCTION PATH』 を参照してください。

別名

別名 は、表、ビュー、またはデータベース・ファイルのメンバーの代替名と考えてください。別名によって、ファイルの一時変更を避けることができます。別名の方が一時変更よりも都合がよいだけではなく、別名は一度だけ作成すればよい永続オブジェクトでもあります。

名前または表の別名によって、SQL ステートメントで表やビューを参照することができます。 SQL ステートメントで別名を使用した場合には、データベース・ファイルのメンバーだけが参照できます。別名は、同じリレーショナル・データベース内の表、ビュー、またはデータベース・ファイルのメンバーのみを参照することができます。

表名またはビュー名を使用する際にはいつでも別名を使用することができます。ただし、以下の場合を除きます。

- CREATE TABLE または CREATE VIEW ステートメントのように新規の表名またはビュー名の場合は、別名を使用しないでください。例えば、PERSONNEL という別名が作成された場合、CREATE TABLE PERSONNEL のような後続のステートメントはエラーになります。
- データベース・ファイルの個々のメンバーを参照している別名は、選択ステートメント、DELETE、INSERT、SELECT INTO、SET 変数、UPDATE、または VALUES INTO ステートメントでのみ使用することができます。

別名が参照するオブジェクトが存在しない場合でも、別名を作成することができます。ただし、別名を参照するステートメントが実行される時点では、そのオブジェクトは存在している必要があります。オブジェクトが存在しない場合に別名を作成すると、警告が戻されます。ある別名が別の別名を参照することはできません。別名は、同じリレーショナル・データベース内の表、ビュー、またはデータベース・ファイルのメンバーのみを参照することができます。

別名によって表、ビュー、またはデータベース・ファイルのメンバーを参照するというオプションは、明示的に構文図に示されることはありません。また、SQL ステートメントの説明で記述されることもありません。

新規の別名は、既存の表、ビュー、索引、ファイル、または別名と同じ完全修飾名を持つことはできません。

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と似ています。別名は、SQL ステートメントを実行する際には定義しておく必要がありますが、修飾された基本表名、ビュー名、またはデータベース・ファイルのメンバー名によって置き換えられます。例えば、PBIRD.SALES が DSPN014.DIST4_SALES_148 の別名である場合、次のステートメントの実行時には、

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

次のようになります。

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

他の表を参照するために別名を除去し、再作成した場合には、その別名を参照している SQL ステートメントはいずれも、次の実行時には暗黙的に再バインドされま

す。 CREATE VIEW または CREATE INDEX ステートメントが別名を参照している場合、別名を除去し、再作成してもビューや索引に影響はありません。

既存の DB2 UDB (OS/390 および z/OS 版)のアプリケーションで許される構文に関しては、CREATE ALIAS および DROP ALIAS ステートメントで、ALIAS の代わりに SYNONYM を使用することができます。

権限 ID と権限名

権限 ID 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準備処理との間の接続を確立するときに、データベース・マネージャー側で取得する文字ストリングです。権限 ID は、特権の集合を示します。権限 ID がユーザーまたはユーザーのグループを示すこともありますが、データベース・マネージャーでは、権限 ID のこの特性は管理しません。

権限 ID は、データベース・マネージャーで以下のように使用されます。

- 表、ビュー、制約、パッケージ、および索引の名前に関する暗黙の修飾子となる。
- SQL ステートメントの権限を検査する。

権限 ID は、すべての SQL ステートメントに適用されます。暗黙的な修飾は、静的 SQL を使用するか、動的 SQL を使用するかによって異なります。

- 静的 SQL の場合は、暗黙的な修飾子はプログラムの所有者。
- 動的 SQL の場合は、暗黙的な修飾子はプログラムを実行するユーザー。

静的 SQL ステートメントの許可検査に使用される権限 ID は、プリコンパイラーのコマンドで指定された USRPRF の値によって、以下のように異なります。

- USRPRF(*OWNER) が指定される場合、または USRPRF(*NAMING) が指定され、SQL 命名モードが使用される場合は、ステートメントの権限 ID は、非分散 SQL プログラムの所有者です。分散 SQL プログラムの場合は、SQL パッケージの所有者です。
- USRPRF(*USER) が指定される場合、または USRPRF(*NAMING) が指定され、システム命名モードが使用される場合は、ステートメントの権限 ID は、非分散 SQL プログラムを実行するユーザーの権限 ID です。分散 SQL プログラムの場合は、現行サーバーのユーザーの権限 ID です。

動的 SQL ステートメントの許可検査に使用される権限 ID も、そのステートメントの実行される場所と方法によって次のように異なります。

- 非分散プログラムで準備され実行されるステートメントの場合
  - そのプログラムの USRPRF の値が *USER で、DYNUSRPRF の値が *USER である場合は、適用される権限 ID は、その非分散プログラムを実行するユーザーの権限 ID。これは、*実行時権限 ID* と呼ばれる。
  - そのプログラムの USRPRF の値が *OWNER で、DYNUSRPRF の値が *USER である場合は、適用される権限 ID は、その非分散プログラムを実行するユーザーの権限 ID。
  - そのプログラムの USRPRF の値が *OWNER で、DYNUSRPRF の値が *OWNER である場合は、適用される権限 ID は、その非分散プログラムの所有者の権限 ID。
- 分散プログラムで準備および実行されるステートメントの場合
  - その SQL パッケージの USRPRF の値が *USER で、DYNUSRPRF の値が *USER である場合は、適用される権限 ID は、現行サーバーでその SQL パッケージを実行するユーザーの権限 ID。この権限 ID も、実行時権限 ID と呼ばれる。

  - その SQL パッケージの USRPRF の値が *OWNER で、DYNUSRPRF の値が *USER である場合は、適用される権限 ID は、現行サーバーでその SQL パッケージを実行するユーザーの権限 ID。
  - その SQL パッケージの USRPRF の値が *OWNER で、DYNUSRPRF の値が *OWNER である場合は、適用される権限 ID は、現行サーバーのその SQL パッケージの所有者の権限 ID。
- 対話方式で出されるステートメントの場合、SQL 開始 (STRSQL) コマンドを出したユーザーの ID が、適用される権限 ID になります。
- RUNSQLSTM コマンドにより実行されるステートメントの場合、RUNSQLSTM コマンドを出したユーザーの ID が、適用される権限 ID になります。
- ステートメントが REXX から実行される場合、適用される権限 ID は、STRREXPRC コマンドを出したユーザーの ID です。

OS/400 では、実行時権限 ID はジョブのユーザー・プロファイルです。

SQL ステートメントで指定される *権限名* とステートメントの権限 ID を混同してはなりません。権限名は、GRANT や REVOKE ステートメントで使用されるIDで、認可や取り消しの対象を指します。特権 *X* を認可する前提は、*X* がそれらの特権を必要とするステートメントの権限 ID であることです。SQL ステートメントに関する権限を検査するときには、グループ・ユーザー・プロファイルが使用されることもあります。グループ・ユーザー・プロファイルについての詳細は、「iSeries 機密保護解説書」  を参照してください。

例

- SMITH というユーザー ID を持つユーザーがいるとします。このユーザーが次のようなステートメントを対話式で実行する場合は、SMITH が権限 ID となります。

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

  SMITH は、このステートメントの権限 ID です。したがって、このステートメントを実行する権限が SMITH にあるかどうかが検査されます。また、SMITH は、TDEPT の暗黙の修飾子です。

  KEENE は、ステートメントに指定されている権限名です。KEENE には、SMITH.TDEPT に対する SELECT 特権が与えられます。
- SMITH に管理権限があり、以下のステートメントの権限 ID が SMITH であるとします。

  ```
  DROP TABLE TDEPT
  ```

  上記のステートメントによって、表 SMITH.TDEPT が除去されます。

  ```
  DROP TABLE SMITH.TDEPT
  ```

  上記のステートメントによって、表 SMITH.TDEPT が除去されます。

  ```
  DROP TABLE KEENE.TDEPT
  ```

  上記のステートメントによって、表 KEENE.TDEPT が除去されます。

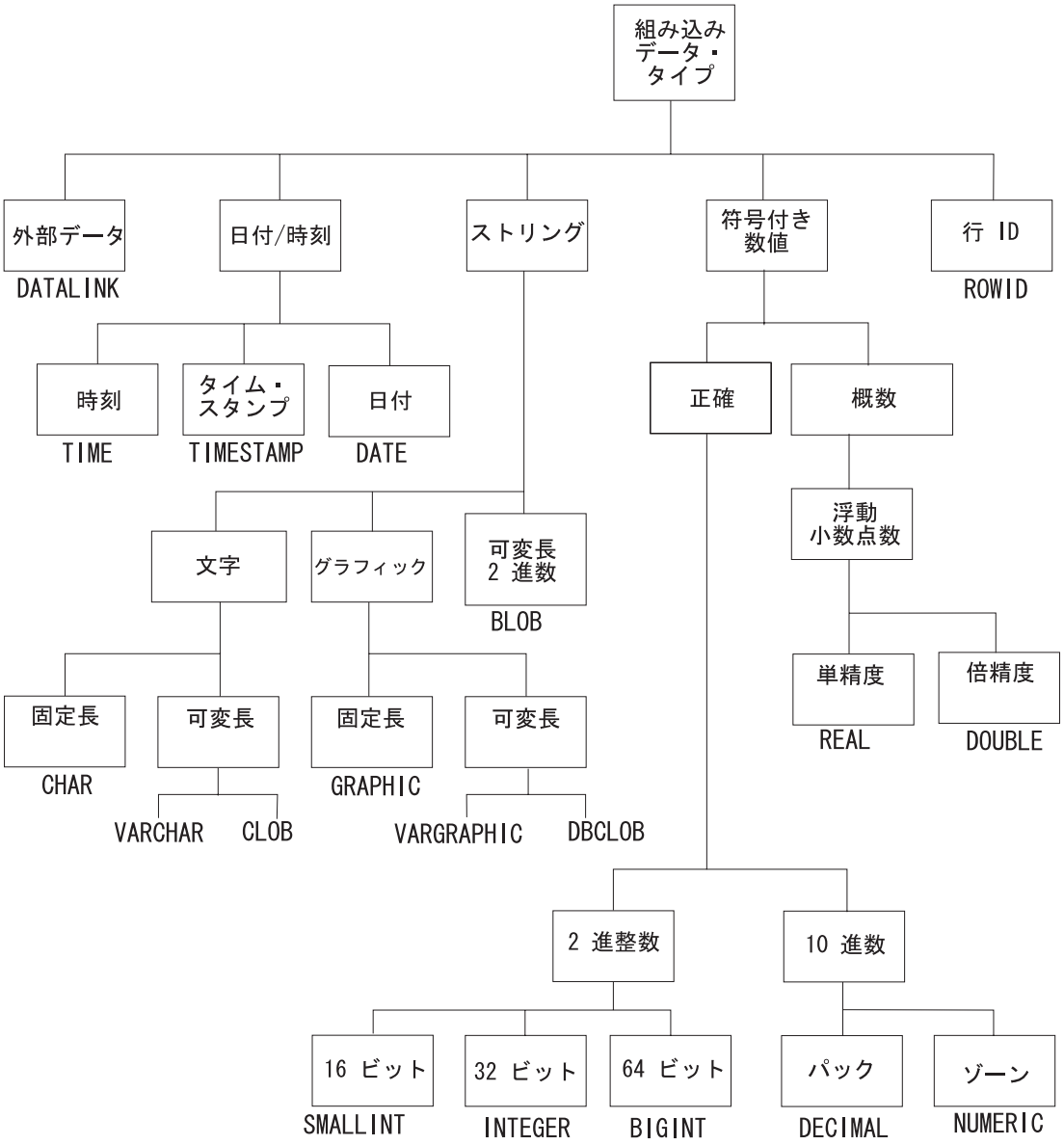
データ・タイプ

SQL で操作できるデータの最小単位を*値* と呼びます。値の解釈は、その値のソースのデータ・タイプに応じて異なります。値のソースには、以下のものがあります。

- 列
- 定数
- 式
- 関数
- ホスト変数
- 特殊レジスター

DB2 は、IBM 提供のデータ・タイプ (組み込みデータ・タイプ) とユーザー定義のデータ・タイプ (特殊タイプ) をサポートしています。このセクションでは、組み込みデータ・タイプについて説明します。特殊データ・タイプの説明については、77 ページの『ユーザー定義タイプ』を参照してください。

次の図には、DB2 UDB for iSeries プログラムでサポートされる種々の組み込みデータ・タイプを示してあります。



ヌル: どのようなデータ・タイプにもヌル値が含まれます。ヌル値は、ヌル以外のすべての値からヌルを区別する特殊な値であり、それによって値 (ヌル以外の) の不在を示します。ヌル値はすべてのデータ・タイプに含まれますが、値のソースによってはヌル値を提供できないものがあります。例えば、定数、NOT NULL として定義されている列、および特殊レジスターには、ヌル値を含めることはできません。また、COUNT 関数および COUNT_BIG 関数はヌル値を戻すことはできません。さらに、照会の結果として ROWID 列にヌル値が戻されることがありますが、ROWID 列にはヌル値を保管することはできません。

データ・タイプの詳細については、以下のトピックを参照してください。

- 64 ページの『2 進ストリング』
- 64 ページの『文字ストリング』
- 65 ページの『文字のサブタイプ』
- 66 ページの『グラフィック・ストリング』

- 67 ページの『グラフィック・サブタイプ』
- 68 ページの『ラージ・オブジェクト (LOB)』
- 69 ページの『数値』
- 70 ページの『日付/時刻の値』
- 76 ページの『データ・リンク値』
- 77 ページの『行 ID の値』
- 77 ページの『ユーザー定義タイプ』

列のデータ・タイプの指定に関する詳しい説明は、 542 ページの『CREATE TABLE』を参照してください。

2 進ストリング

2 進ストリング は、一連のバイトです。 2 進ストリング (BLOB ストリング) の長さとは、そのストリングのバイト数を指します。2 進ストリングは、65535 の CCSID を持っています。

BLOB 列の場合、長さ属性は 1 から 2 147 483 647 バイトまでの範囲内でなければなりません。 BLOB の詳細については、 68 ページの『ラージ・オブジェクト (LOB)』を参照してください。

BLOB ストリング・タイプを持つホスト変数は、REXX、RPG/400、および COBOL/400 を除くすべてのホスト言語で定義することができます。

文字ストリング

文字ストリング は、一連のバイトです。ストリングの長さとは、そのストリングのバイト数を指します。長さがゼロの文字ストリングの値は、*空ストリング* と呼ばれます。この空ストリングとヌル値を混同しないように注意してください。

固定長文字ストリング

固定長の文字ストリングの列の値はすべて、同じ長さを持ちます。この長さは、その列の長さ属性によって決まります。長さ属性は 1 から 32766 までの範囲 (両端を含む) 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のタイプは以下のとおりです。

- VARCHAR (CHAR VARYING および CHARACTER VARYING と同義)
- CLOB (CHAR LARGE OBJECT および CHARACTER LARGE OBJECT と同義)

これらのストリング・タイプの列の値は、異なる長さを持つ場合があります。値の最大長は、列の長さ属性によって決まります。

VARCHAR 列の場合、長さ属性は 1 から 32740 までの範囲 (両端を含む) でなければなりません。 CLOB 列の場合、長さ属性は 1 から 2 147 483 647 までの範囲 (両端を含む) でなければなりません。 CLOB の詳細については、 68 ページの『ラージ・オブジェクト (LOB)』を参照してください。

文字ストリング・ホスト変数

- 固定長文字ストリング変数は、REXX を除くすべてのホスト言語で使用することができます。(C 言語では、固定長文字ストリング変数は、その長さが 1 に限定されます。)
- VARCHAR 可変長文字ストリング変数は、C、COBOL、PL/I、REXX、および RPG で使用することができます。
  - PL/I、REXX、および ILE RPG の場合、可変長文字ストリングのデータ・タイプがあります。
  - COBOL および C では、可変長文字ストリングを構造体として表します。
  - C では、可変長文字ストリング変数は、NUL 終了ストリングによって表すこともできます。
  - RPG/400 では、可変長文字ストリング変数は、外部記述データ構造の結果として組み込まれる VARCHAR 列によってしか表すことができません。
- CLOB 可変長文字ストリング変数は、REXX、RPG/400、および COBOL/400 を除くすべてのホスト言語で定義することができます。
  - ILE RPG では、CLOB 可変長文字ストリングは SQLTYPE キーワードを使用して宣言されます。
  - その他の言語ではすべて、SQL TYPE IS CLOB 文節が使用されます。

文字のサブタイプ

それぞれの文字ストリングは、さらに次のいずれかとして定義されます。

ビット・データ
コード化文字セットに関連付けられていないデータで、変換が行われることのないデータ。ビット・データの CCSID は 65535 です。

SBCS データ
すべての文字が単一バイトで表現されているデータ。SBCS データ文字ストリングはそれぞれ、関連する CCSID を持っています。SBCS データ文字ストリングは、演算での使用に先立って、必要に応じて異なる CCSID を持つ文字ストリングに変換されます。

混合データ
1 バイト文字セット (SBCS) の文字と 2 バイト文字セット (DBCS) の文字を混用することができるデータ。混合データ文字ストリングはそれぞれ、関連の CCSID を持っています。混合データ文字ストリングは、演算に先立って、必要に応じて異なる CCSID を持つ文字ストリングに変換されます。混合データに DBCS 文字が含まれている場合は、SBCS データに変換することはできません。

データベース・マネージャーは、2 バイト文字のサブクラスを認識しません。また、個々の 2 バイト・コードに特定の意味を割り当てることもありません。ただし、混合データの中では、次に示す 2 つの 1 バイト EBCDIC コードに特殊な意味が割り当てられています。

- X'0E' (“シフトアウト”文字)。一連の 2 バイト・コードの始めを示すのに使用されます。
- X'0F' (“シフトイン”文字)。一連の 2 バイト・コードの終わりを示すのに使用されます。

以下の条件に該当する場合に、データベース・マネージャーは、混合データ文字ストリングの 2 バイト文字を認識します。

ストリング中の 2 バイト文字は、シフトアウト文字とシフトイン文字の対で囲まれていなければなりません。

シフトアウト文字とシフトイン文字の対は、ストリングを左から右に読み取ったときに検出されます。 X'0E' というコードは、その後に X'0F' が見つかった場合は、シフトアウト文字として認識されますが、見つからない場合は、そのコードは無効です。X'0E' の後で 2 バイト境界上で見つかった最初の X'0F' を、対のシフトイン文字として扱います。 2 バイト境界上にない X'0F' は、認識されません。

シフトアウト文字とシフトイン文字の間にあるバイトの数は偶数でなければなりません。バイトの各対 (2 バイト) が、それぞれ 1 つの 2 バイト文字であると見なされます。ストリング中には、シフトアウト文字とシフトイン文字の対が複数存在しても構いません。

混合データ文字ストリングの長さとは、その合計バイト数です。2 バイト文字については、それぞれ 2 バイトとして数え、シフトアウト文字またはシフトイン文字については、それぞれ 1 バイトとして数えます。

ジョブの CCSID が、DBCS が使用可能であることを示しており、しかも FOR BIT DATA、FOR SBCS DATA、または SBCS CCSID が指定されていない場合は、CREATE TABLE は、文字の列を DBCS 混用フィールドとして作成します。このような列は、SQL ユーザーからは文字フィールドのように見えますが、システムのデータベースのサポートは、DBCS 混用フィールドとして扱います。 DBCS 混用フィールドの定義については、「DB2 UDB for iSeries データベース・プログラミング」を参照してください。

グラフィック・ストリング

グラフィック・ストリングは一連の 2 バイト文字です。このストリングの長さは、その文字の数になります。文字ストリングと同様に、グラフィック・ストリングも空でも構いません。

固定長グラフィック・ストリング

固定長グラフィック・ストリング列の値はすべて、長さが同じです。この長さは、列の長さ属性によって決まります。長さ属性は 1 から 16383 までの範囲 (両端を含む) でなければなりません。

可変長グラフィック・ストリング

可変長グラフィック・ストリングのタイプは以下のとおりです。

- VARGRAPHIC (GRAPHIC VARYING と同義)
- DBCLOB

これらのストリング・タイプの列の値は、異なる長さを持つ場合があります。値の最大長は、列の長さ属性によって決まります。

VARGRAPHIC 列の場合、長さ属性は 1 から 16370 までの範囲 (両端を含む) でなければなりません。 DBCLOB 列の場合、長さ属性は 1 から 1 073 741 823 まで

の範囲 (両端を含む) でなければなりません。DBCLOB の詳細については、 68 ページの『ラージ・オブジェクト (LOB)』を参照してください。

グラフィック・ストリングのホスト変数

- 固定長グラフィック・ストリングのホスト変数は、C、ILE COBOL、および ILE RPG/400 で定義することができます。(C の場合、固定長グラフィック・ストリングのホスト変数の長さは、1 の長さに限定されます。)

  固定長グラフィック・ストリングのホスト変数は、PL/I、COBOL/400、および RPG/400 では定義できませんが、文字ストリングのホスト変数がファイルの外部記述の GRAPHIC 列からソースに生成された場合は、その文字ストリングのホスト変数は、固定長グラフィック・ストリングのホスト変数と同様に扱われます。
- 可変長グラフィック・ストリングのホスト変数は、C、ILE COBOL、REXX、および ILE RPG で定義することができます。
  - REXX および ILE RPG には、可変長グラフィック・ストリングのデータ・タイプがあります。
  - C および ILE COBOL では、可変長グラフィック・ストリングは構造体として表されます。
  - C の場合、可変長グラフィック・ストリング変数は、NUL 終了グラフィック・ストリングによって表すこともできます。
  - 可変長グラフィック・ストリングのホスト変数は、PL/I、COBOL/400、および RPG/400 では定義できませんが、文字ストリングのホスト変数がファイルの外部記述の VARGRAPHIC 列からソースに生成された場合は、その文字ストリングのホスト変数は可変長グラフィック・ストリングのホスト変数と同様に扱われます。
- DBCLOB 可変長文字ストリング変数は、REXX、RPG/400、および COBOL/400 を除くすべてのホスト言語で定義することができます。
  - ILE RPG では、DBCLOB 可変長文字ストリングは SQLTYPE キーワードを使用して宣言されます。
  - その他の言語ではすべて、SQL TYPE IS DBCLOB 文節が使用されます。

グラフィック・サブタイプ

グラフィック・ストリングはそれぞれ、さらに DBCS データまたは UCS-2 データとして定義されます。

DBCS データ すべての文字がそれぞれ、シフトアウト文字もシフトイン文字も含まない 2 バイト文字セット (DBCS) の文字で表されるデータ。

すべての DBCS グラフィック・ストリングには、2 バイトのコード化文字セットを識別する CCSID があります。DBCS グラフィック・ストリングは、演算での使用に先立って、必要に応じて異なる DBCS CCSID をもつ DBCS グラフィック・ストリングに変換されます。

UCS-2 データ すべての文字がそれぞれ、汎用コード化文字セット (UCS-2) の文字で表されるデータ。

グラフィック・ストリングのホスト変数が明示的に CCSID のタグを付けられていない場合は、ジョブの CCSID の関連する DBCS CCSID が使用されます。関連す

る DBCS CCSID が存在しない場合には、ホスト変数に 65535 のタグが付けられます。グラフィック・ストリングのホスト変数に暗黙に UCS-2 CCSID のタグが付けられることはありません。グラフィックのホスト変数に CCSID のタグを付ける方法については、DECLARE VARIABLE ステートメントを参照してください。

ラージ・オブジェクト (LOB)

ラージ・オブジェクト *(LOB)* という用語は、次のいずれかのデータ・タイプを意味しています。

2 進ラージ・オブジェクト (BLOB) ストリング

2 進ラージ・オブジェクト (BLOB) は、最大長が 2 147 483 647 の可変長ストリングです。 BLOB は、ピクチャー、音声、および混合メディアなど従来にないデータを保管するために考案されたものです。また、BLOB は、特殊タイプおよびユーザー定義の関数が使用する構造化データも保管することができます。 BLOB は 2 進ストリングであると見なされます。

BLOB ストリングと FOR BIT DATA 文字ストリングは同じような目的に使用されることがありますが、この 2 つのデータ・タイプに互換性はありません。 BLOB 関数を使用すると、FOR BIT DATA 文字ストリングを 2 進ストリングに変えることができます。

BLOB の CCSID は 65535 です。

文字ラージ・オブジェクト (CLOB) ストリング

文字ラージ・オブジェクト (CLOB) は、最大長が 2 147 483 647 の可変長ストリングです。 CLOB は、非常に長い文書のような大きな SBCS データまたは混合データを保管するために考案されたものです。例えば、従業員の履歴書、演劇の台本、あるいは小説の原稿などの情報を CLOB に保管することができます。

CLOB の CCSID を 65535 にすることはできません。

2 バイト文字ラージ・オブジェクト (DBCLOB) ストリング

2 バイト文字ラージ・オブジェクト (DBCLOB) は、最大長が 2 バイト文字で 1 073 741 823 の可変長グラフィック・ストリングです。 DBCLOB は、UCS-2 での非常に長い文書のような大きな DBCS データを保管するために考案されたものです。

DBCLOB の CCSID を 65535 にすることはできません。

ロケーターを用いたラージ・オブジェクト (LOB) の操作

アプリケーションで LOB 値全体をホスト変数に移動することを望んでいない場合、アプリケーションでは LOB 値の参照にラージ・オブジェクト・ロケーター (LOB ロケーター) を使用することができます。

LOB ロケーターは、データベース・サーバーにおける単一の LOB 値を表す値を持ったホスト変数です。LOB ロケーターによって、LOB 値全体をホスト変数に保管したり、アプリケーション・プログラムが実行可能なアプリケーション・リクエスター (クライアント) に転送しなくても、非常に大きなオブジェクトをアプリケーション・プログラムで取り扱うことができるようなメカニズムが可能になります。

例えば、LOB 値を選択する場合、アプリケーション・プログラムは、可能であれば LOB 値全体を選択し、それを同じ大きさのホスト変数に入れる (アプリケーション・プログラムが LOB 値全体を一度で処理するのであれば受け入れ可能) か、または、その代わりに LOB 値を選択して LOB ロケーターに入れます。その後、LOB ロケーターを使用すれば、アプリケーション・プログラムはロケーター値を入力として与えることにより、LOB 値上での後続のデータベース操作 (スカラー関数の SUBSTR、CONCAT、VALUE、LENGTH の適用、割り当ての実行、LIKE または POSSTR による LOB の検索、あるいは LOB に対する UDF の適用など) を出すことができます。したがって、例えば、クライアント・ホスト変数に割り当てられたデータ量などのロケーター演算の結果の出力は、一般的には、入力 LOB 値の小さなサブセットになります。

また、LOB ロケーターは、次のような LOB 式を表すことも可能です。

```
SUBSTR((lob1) CONCAT (lob2) CONCAT (lob3), start, length)
```

アプリケーション・プログラムにおける通常のホスト変数の場合、ヌル値がホスト変数に選択されると、標識変数は -1 に設定され、値がヌルであることを示します。しかしながら、LOB ロケーターの場合は、標識変数の意味は若干異なっています。ロケーター・ホスト変数自体は決してヌルになることはないため、負の標識変数値は LOB ロケーターにより表される LOB 値がヌルであることを示しています。標識変数値によって、クライアントに対してヌル情報がローカルに保持されます。サーバーは、有効なロケーターによってヌル値を追跡しません。

LOB ロケーターは値を表しているのであり、行またはデータベースの位置を表しているのではないということを理解することが重要です。いったんロケーターに値が選択されてしまうと、ロケーターが参照している値に影響を及ぼすことになるオリジナルの行や表で実行できる演算はありません。ロケーターに関連した値は、トランザクションが終了するか、あるいはロケーターが明示的に解放されるか、そのいずれかが先に起こるまで有効です。

LOB ロケーターは、トランザクション中に LOB 値を参照するためのメカニズムに過ぎません。したがって、ロケーターが作成されたときのトランザクションを超えてまでも存続することはありません。また、LOB ロケーターはデータベース・タイプでもありません。したがって、データベースに保管されることはなく、その結果、ビュー制約や検査制約に関与することも不可能です。しかしながら、ロケーターは LOB タイプを表しているため、FETCH、OPEN、CALL、および EXECUTE ステートメントで使用される SQLDA 構造内で記述できるような LOB タイプ用の SQLTYPE があります。

数値

すべての数値は、*符号*および*精度*を持ちます。 精度は、符号を除いた 2 進数または 10 進数の合計桁数です。値がゼロの場合、その数値の符号は正となります。

短整数

短整数 は、5 桁の精度を持つ 2 バイト (16 ビット) で構成される 2 進数です。短整数の範囲は -32768 から +32767 までです。

短整数では、10 進数の精度と位取りは、COBOL、RPG、および iSeries のシステム・ファイルによってサポートされます。2 進整数の精度と位取りに関しては、DDS 解説書を参照してください。

長整数

長整数 は、10 桁の精度を持つ 4 バイト (32 ビット) で構成される 2 進数です。長整数の範囲は -2147483648 から +2147483647 までです。

長整数では、10 進数の精度と位取りは、COBOL、RPG、および iSeries のシステム・ファイルによってサポートされます。2 進整数の精度と位取りに関しては、DDS 解説書を参照してください。

64 ビット整数 (BIGINT)

64 ビット整数 は、19 桁の精度を持つ 8 バイト (64 ビット) で構成される 2 進数です。64 ビット整数の範囲は、-9223372036854775808 から +9223372036854775807 までです。

浮動小数点数

単精度浮動小数点 数は、実数を 32 ビットの概数で表したものです。絶対値の範囲は、およそ $1.17549436 \times 10^{-38}$ から $3.40282356 \times 10^{38}$ までです。

倍精度浮動小数点 数は、実数を IEEE 64 ビットの概数で表したものです。絶対値の範囲は、およそ $2.2250738585072014 \times 10^{-308}$ から $1.7976931348623158 \times 10^{308}$ までです。

10 進数

10 進数 の値は、暗黙の小数点を持つパック 10 進数またはゾーン 10 進数を示します。小数点の位置は、その数値の精度および位取りによって決まります。位取り (数値の小数部の桁数) を負の数にしたり、精度より大きい数にすることはできません。最大精度は 31 桁です。

1 つの 10 進数の列にある値は、すべて同一の精度および位取りを持ちます。 10 進変数や 10 進数列の中の数値の範囲は、*-n* から *+n* までです。ここで、*n* の絶対値は、適用可能な精度および範囲で表すことができる最大の数値です。この場合、最大の範囲は、-10^{31} +1 から 10^{31} -1 です。

数値のホスト変数

2 進短整数および 2 進長整数の変数は、すべてのホスト言語で使用することができます。64 ビット整数の変数は、C、C++、ILE COBOL、および ILE RPG のみで使用することができます。浮動小数点変数は、RPG/400 および COBOL/400 を除くすべてのホスト言語で使用することができます。10 進変数は、サポートされているすべてのホスト言語で使用することができます。

日付/時刻の値

日付/時刻の値は、特定の算術演算やストリング演算で使用することができ、特定のストリングと互換性がありますが、ストリングでも数値でもありません。ただし、ストリングで日付/時刻の値を表すことができます。 71 ページの『日付/時刻の値のストリング表現』を参照してください。

日付

日付 は、グレゴリオ暦のもとでの時間を示す 3 つの部分 (年、月、および日) から成る値です。この暦は、紀元 1 年から有効であると想定されています。[16] 年の部分の範囲は、0001 から 9999 までです。日付の形式の *JUL、*MDY、*DMY、および *YMD は、年が 1940 から 2039 までの範囲の日付を表すことができます。月の部分の範囲は、1 から 12 までです。日の部分の範囲は、1 から *x* までです。ここで *x* は、年および月に応じて 28、29、30、または 31 になります。

日付の内部表現は、1 つの整数を含む 4 バイトのストリングです。この整数 (スカリジェル数と呼ばれます) によって、日付を表します。

日付 (DATE) の列の長さは、使用される形式によって、6、8、または 10 バイトのいずれかになります (SQLDA に記述されています)。これらの長さは、値を文字ストリングで表現するのに適した長さです。

時刻

時刻 は、3 つの部分 (時、分、秒) からなる値で、24 時間制を使用して時刻を表します。時の部分の範囲は 0 から 24 まで、分および秒の部分の範囲は 0 から 59 までです。時の値が 24 の場合、分および秒の値は両方ともゼロになります。

時刻の内部表現は、3 バイトのストリングです。それぞれのバイトが、2 つのパック 10 進桁から構成されます。最初のバイトが時、2 番目のバイトが分、3 番目のバイトが秒をそれぞれ表します。

時刻 (TIME) の列の長さは 8 バイトで (SQLDA に記述されています)、この長さは、時刻の文字ストリング表現に適した長さです。

タイム・スタンプ

タイム・スタンプ は、7 つの部分 (年、月、日、時、分、秒、およびマイクロ秒) から成る値で、時刻にマイクロ秒の小数指定があることを除けば、上記で定義したものと同じ日付および時刻を示します。

タイム・スタンプの内部表現は、10 バイトのストリングです。最初の 4 バイトが日付、次の 3 バイトが時刻、最後の 3 バイトがマイクロ秒をそれぞれ表します (最後の 3 バイトには、6 桁のパック化された数字が入っています)。

タイム・スタンプ (TIMESTAMP) の列の長さは 26 バイトで (SQLDA に記述されている)、この長さは、値の文字ストリング表現に適した長さです。

日付時刻ホスト変数

一般に、日付、時刻、およびタイム・スタンプの値を含めるためには文字ストリング・ホスト変数を使用します。ただし、日付、時刻、およびタイム・スタンプ・ホスト変数は、ILE COBOL および ILE RPG でも指定できます。

日付/時刻の値のストリング表現

データ・タイプが DATE (日付)、TIME (時刻)、または TIMESTAMP (タイム・スタンプ) である値は内部形式で表されますが、SQL ユーザーは、この内部形式を意

16. ヒストリー上の日付は、必ずしもグレゴリオ暦に従うとは限らないことに注意してください。 1582-10-04 と 1582-10-15 間の日付は、グレゴリオ暦では存在しませんが、有効な日付として受け入れられます。

識する必要はありません。日付、時刻、およびタイム・スタンプは、文字ストリングまたは UCS-2 グラフィック・ストリングで表すこともできます。SQL のユーザーは、これらの文字ストリング表現を直接扱うことになります。これは、定数には、DATE、TIME、または TIMESTAMP などのデータ・タイプが用意されていないためです。ILE RPG および ILE COBOL のみが日付/時刻変数をサポートしています。検索するには、日付/時刻の値を文字ストリング変数に割り当てることができます。結果のストリングの形式は、ステートメントを作成するときに効力を持っているデフォルトの日付形式およびデフォルトの時刻形式によって異なります。デフォルトの日付および時刻の形式は、日付形式 (DATFMT)、日付区切り文字 (DATSEP)、時刻形式 (TIMFMT)、および時刻区切り文字 (TIMSEP) パラメーターに基づいて設定されます。

日付/時刻の値の有効なストリング表現が内部の日付/時刻の値による演算で使用される場合は、その演算が行われる前に、ストリング表現が日付、時刻、またはタイム・スタンプの内部形式に変換されます。ストリングの CCSID が外部のコード化体系を表している場合 (例えば、ASCII)、そのストリングは、日付/時刻の値の内部形式への変換に先立って、そのデフォルトの CCSID によって示されたコード化文字セットにまず変換されます。

以下の各項では、日付/時刻の値の有効なストリング表現を定義しています。

日付ストリング: 日付のストリング表現は、数字で始まる文字ストリングで、最低 6 文字の長さを持ちます。このストリングには、後書きブランクを付けることができます。IBM SQL の標準形式を使用する場合は、月および日の部分から先行ゼロを省略することができます。IBM SQL 標準形式のそれぞれは、名前によって識別され、関連する省略形 (CHAR 関数で使用される) を含みます。それ以外の形式は、CHAR 関数で使用される省略形を持ちません。年が 2 桁の形式の区切り文字は、日付区切り文字 (DATSEP) パラメーターにより制御されます。日付の有効なストリング形式は、表 4 に示されています。

データベース・マネージャーは、次のいずれかの形式のストリングを日付として認識します。

- デフォルトの日付形式に指定されている形式、または
- IBM SQL 標準日付形式のいずれか、または
- 不定様式の年間通算日形式

表 4. 日付のストリング表現で使用する形式

形式の名前	省略形	日付形式	例
国際標準化機構 (*ISO)	ISO	yyyy-mm-dd	1987-10-12
IBM USA 標準 (*USA)	USA	mm/dd/yyyy	10/12/1987
IBM 欧州標準 (*EUR)	EUR	dd.mm.yyyy	12.10.1987
日本工業規格の西暦 (*JIS)	JIS	yyyy-mm-dd	1987-10-12
不定様式の年間通算日	-	yyyyddd	1987285

表 4. 日付のストリング表現で使用する形式 (続き)

形式の名前	省略形	日付形式	例
年間通算日形式 (*JUL)	-	yy/ddd	87/285
月、日、年 (*MDY)	-	mm/dd/yy	10/12/87
日、月、年 (*DMY)	-	dd/mm/yy	12/10/87
年、月、日 (*YMD)	-	yy/mm/dd	87/12/10

デフォルトの日付形式は、以下のインターフェースを使用して指定できます。

表 5. デフォルト日付形式インターフェース

SQL インターフェース	指定
組み込み SQL	DATFMT および DATSEP パラメーターは、SQL プログラム作成 (CRTSQLxxx) コマンドに指定することができます。 SQL を組み込むプログラム・ソースに DATFMT および DATSEP パラメーターを指定するためには、SET OPTION ステートメントを使用することもできます。 (CRTSQLxxx コマンドの詳細については、「DB2 UDB for iSeries ホスト言語での SQL プログラミング」を参照)
対話式 SQL および SQL 実行ステートメント	SQL 開始 (STRSQL) コマンドで DATFMT および DATSEP パラメーターを指定するか、セッション属性を変更します。あるいは、SQL 実行 (RUNSQLSTM) ステートメントで DATFMT および DATSEP パラメーターを使用します。 (STRSQL および RUNSQLSTM コマンドについての詳細は、「SQL プログラミング 概念」を参照)。
サーバー上の呼び出しレベル・インターフェース (CLI)	SQL_ATTR_DATE_FMT および SQL_ATTR_DATE_SEP 環境変数または接続変数。 (CLI の詳細については、「DB2 UDB for iSeries SQL 呼び出しレベル・インターフェース」を参照)。
Developer Kit for Javaを使用したサーバーの JDBC または SQLJ	「日付形式 (Date Format)」および「日付区切り記号 (Date Separator)」接続プロパティー。 (JDBC および SQLJ の詳細については、iSeries Information Center の IBM Developer Kit for Java トピックを参照)。
iSeries Access ODBC ドライバーを使用したクライアントの ODBC	ODBC セットアップでの「アドバンスト・サーバー・オプション (Advanced Server Options)」の中の「日付形式 (Date Format)」および「日付区切り記号 (Date Separator)」。 (ODBC の詳細については、iSeries Information Center の iSeries Access カテゴリーを参照)。

表 5. デフォルト日付形式インターフェース (続き)

SQL インターフェース	指定
IBM Toolbox for Java を使用したクライアントの JDBC	JDBC セットアップの中の「形式 (Format)」。 (ODBC の詳細については、iSeries Information Center の iSeries Access カテゴリーを参照)。 (IBM Toolbox for Java の詳細については、iSeries Information Center の IBM Toolbox for Java トピックを参照)。

時刻ストリング: 時刻のストリング表現は、数字で始まる文字ストリングで、最低 4 文字の長さを持ちます。このストリングには後書きブランクを付けることができます。時刻の時の部分から先行ゼロを除去することができ、秒の部分全体を除去することができます。ユーザーが秒の除去を選択すると、暗黙にゼロ秒が指定されたことになります。したがって、13.30 は 13.30.00 に等しいことになります。

時刻の有効なストリング形式は、表 6に示されています。IBM SQL 標準形式のそれぞれは、名前によって識別され、関連する省略形 (CHAR 関数で使用される) を含みます。それ以外の形式 (*HMS) は、CHAR 関数によって使用される省略形を持っていません。*HMS 形式の区切り文字は、時刻区切り文字 (TIMSEP) パラメーターにより制御されます。

データベース・マネージャーは、ストリングが次のいずれかの場合に、時刻として認識します。

- デフォルトの時刻形式に指定されている形式、または
- IBM SQL 標準の時刻形式のいずれか

TIMFMT および TIMSEP パラメーターは、CRTSQLxxx、RUNSQLSTM、および STRSQL コマンドに指定します。SET OPTION ステートメントを使用して、組み込み SQL を含むプログラムのソース内に TIMFMT および TIMSEP を指定することができます。

表 6. 時刻のストリング表現で使用する形式

形式の名前	省略形	時刻の形式	例
国際標準化機構 (*ISO)	ISO	hh.mm.ss [17]	13.30.05
IBM USA 標準 (*USA)	USA	hh:mm AM または PM	1:30 PM
IBM 欧州標準 (*EUR)	EUR	hh.mm.ss	13.30.05
日本工業規格の西暦 (*JIS)	JIS	hh:mm:ss	13:30:05
時、分、秒 (*HMS)	-	hh:mm:ss	13:30:05

USA 時刻形式では、時は 12 を超えてはならず、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。24 時間時計を使用して、USA 形式と 24 時間

17. 以前の ISO 形式です。JIS を使用すれば、現行の ISO 形式になります。

時計の間の対応を示すと、次のようになります。

表 7. USA 時刻形式

USA 形式	24 時間時計
12:01 AM ～ 12:59 AM	00.01.00 ～ 00.59.00
01:00 AM ～ 11:59 AM	01:00.00 ～ 11:59.00
12:00 PM (正午) ～ 11:59 PM	12:00.00 ～ 23.59.00
12:00 AM (午前零時)	24.00.00
00:00 AM (午前零時)	00.00.00

USA 形式では、時刻の分の部分と AM または PM との間に 1 つのスペース文字があります。

デフォルトの時刻形式は、以下のインターフェースを使用して指定できます。

表 8. デフォルト時刻形式インターフェース

SQL インターフェース	指定
組み込み SQL	SQL プログラム作成 (CRTSQLxxx) コマンドで、TIMFMT および TIMSEP パラメーターを指定します。 SQL を組み込むプログラム・ソースに TIMFMT および TIMSEP パラメーターを指定するには、SET OPTION ステートメントを使用することもできます。(CRTSQLxxx コマンドの詳細については、「DB2 UDB for iSeries ホスト言語での SQL プログラミング」を参照)
対話式 SQL および SQL 実行ステートメント	SQL 開始 (STRSQL) コマンドで TIMFMT および TIMSEP パラメーターを指定するか、セッション属性を変更します。あるいは、SQL 実行 (RUNSQLSTM) ステートメントで TIMFMT および TIMSEP パラメーターを使用します。 (STRSQL および RUNSQLSTM コマンドについての詳細は、「SQL プログラミング 概念」を参照)。
サーバー上の呼び出しレベル・インターフェース (CLI)	SQL_ATTR_TIME_FMT および SQL_ATTR_TIME_SEP 環境変数または接続変数。 (CLI の詳細については、「DB2 UDB for iSeries SQL 呼び出しレベル・インターフェース」を参照)。
Developer Kit for Javaを使用したサーバーの JDBC または SQLJ	「時刻形式 (Time Format)」および「時刻区切り記号 (Time Separator)」接続プロパティー・オブジェクト。 (JDBC および SQLJ の詳細については、iSeries Information Center の IBM Developer Kit for Java トピックを参照)。
iSeries Access ODBC ドライバーを使用したクライアントの ODBC	ODBC セットアップでの「アドバンスト・サーバー・オプション (Advanced Server Options)」の中の「時刻形式 (Time Format)」および「時刻区切り記号 (Time Separator)」。 (ODBC の詳細については、iSeries Information Center の iSeries Access カテゴリーを参照)。

表 8. デフォルト時刻形式インターフェース (続き)

SQL インターフェース	指定
IBM Toolbox for Java を使用したクライアントの JDBC	JDBC セットアップの中の「形式 (Format)」。 (ODBC の詳細については、iSeries Information Center の iSeries Access カテゴリーを参照)。 (IBM Toolbox for Java の詳細については、iSeries Information Center の IBM Toolbox for Java トピックを参照)。

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現は、数字で始まる文字ストリングで、最低 16 文字の長さを持ちます。タイム・スタンプのストリング表現は、*yyyy-mm-dd-hh.mm.ss.nnnnnn* または *yyyymmddhhmmss* の形式になります。このストリングには、後書きブランクを付けることができます。区切り記号付きのタイム・スタンプ形式を使用しているときは、タイム・スタンプの月、日、時の部分の先行ゼロを省略することができます。マイクロ秒の部分の後続ゼロは、切り捨てたり、全部を除去したりすることができます。ユーザーが、マイクロ秒の部分の数字をすべて除去するように選択すると、その部分には、暗黙のうちに 0 が指定されたことになります。したがって、*1990-3-2-8.30.00.10* は、*1990-03-02-08.30.00.100000* に等しいことになります。

時刻の部分が 24.00.00.000000 であるタイム・スタンプも受け入れられます。

データ・リンク値

データ・リンク値とは、データベースからデータベースの外部に保管されたファイルへの論理的な参照を含むカプセル化された値です。このカプセル化された値の属性は、以下のとおりです。

リンク・タイプ
現在サポートされているリンクのタイプは、URL (Uniform Resource Locator) です。

スキーム
URL の場合、これは HTTP または FILE などの値です。この値は、何が入力されている場合でも、大文字でデータベースに保管されます。

ファイル・サーバー名
ファイル・サーバーの完全なアドレス。この値は、何が入力されている場合でも、大文字でデータベースに保管されます。

ファイル・パス
サーバー内のファイルの識別。この値は、大文字小文字の区別があります。したがって、データベースに保管する場合にも大文字に変換されることはありません。

アクセス制御トークン
必要に応じて、アクセス・トークンがファイル・パスに組み込まれます。これは、動的に生成されるため、データベースに保管されるデータ・リンク値の永続部分ではありません。

コメント
254 バイトまでの記述情報。これは、データのある場所についての詳細や代替の情報など、アプリケーション固有の使用を意図しています。

データ・リンク値で使用する文字は、URL 用に定義されたセットに限定されます。これらの文字には、英大文字 (A から Z) と英小文字 (a から z)、数字 (0 から 9) と特殊文字のサブセット ($、-、_、@、.、&、+、!、*、″、'、(、)、=、 ;、 /、 #、 ?、 :、スペース、およびコンマ) が含まれます。

最初の 4 つの属性はまとめて、リンケージ属性とも言われます。データ・リンク値が、コメントの属性だけを持ち、リンケージ属性をまったく持たないということもあり得ます。そのような値でも列に保管されますが、当然のことながら、そのような列にリンクされるようなファイルはありません。

このようなファイルに対するデータ・リンク参照と、125 ページの『LOB ファイル参照変数の参照』で説明されているような LOB ファイル参照とを識別することが重要です。両方とも、ファイルを表している点は似ています。しかしながら、

- データ・リンクはデータベースに保存されており、リンクされたファイルのリンクとデータの両方とも、データベースにおけるデータの自然な拡張と考えられます。
- ファイル参照変数は一時的に存在しており、ホスト・プログラム・バッファーの代替と考えられます。

データ・リンク値を構築 (DLVALUE) し、データ・リンク値からカプセル化された値を抽出 (DLCOMMENT、DLLINKTYPE、DLURLCOMPLETE、DLURLPATH、DLURLPATHONLY、DLURLSCHEME、DLURLSERVER) するために、組み込みスカラー関数が用意されています。

行 ID の値

行 ID は、表の中の各行を一意的に識別する値です。特定の列またはホスト変数に、行 ID データ・タイプを与えることができます。 ROWID 列を使用することにより、表の中の特定の行まで直接ナビゲートする照会を書くことができます。ROWID 列の中の値はそれぞれ固有のものでなければなりません。表を再編成しても、データベース・マネージャーは永続的にこれらの値を保持します。表に行を挿入するときに、ROWID 列の値を指定しなければ、データベース・マネージャーがその値を生成します。値を指定する場合は、DB2 UDB (OS/390 および z/OS 版) または DB2 UDB for iSeries によりすでに生成されている有効な行 ID 値でなければなりません。

ユーザーは、行 ID 値の内部表現を意識する必要はありません。この値は BIT データを含むものと見なされるため、CCSID 変換を受けることはありません。 ROWID 列の長さ属性は 40 です。

ユーザー定義タイプ

特殊タイプ

特殊タイプ は、その内部表現を組み込みのデータ・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・タイプですが、大部分の演算では別のタイプ、および、互換性のないタイプと考えられます。例えば、ピクチャー・タイ

プ、テキスト・タイプ、およびオーディオ・タイプはすべて、その内部表現に組み込みのデータ・タイプ BLOB を使用していますが、意味体系はまったく異なります。特殊タイプは、SQL ステートメントの CREATE DISTINCT TYPE で作成します。

例えば、次のステートメントによって、AUDIO という名前の特殊タイプが作成されます。

```
CREATE DISTINCT TYPE AUDIO AS BLOB (1M)
```

AUDIO は、組み込みデータ・タイプ BLOB と同じ内部表現を持っていますが、BLOB や他のいずれのタイプとも比較できない、別のタイプと考えられます。このように、AUDIO を他のデータ・タイプと比較することはできないために、AUDIO 専用の関数を作成することが可能になり、そのような関数は他のデータ・タイプには適用不能ということが保証されることになります。

特殊タイプの名前は、スキーマ名で修飾されます。非修飾名の暗黙的なスキーマ名は、特殊タイプが現れる文脈によって異なります。非修飾の特殊タイプ名を使用する場合、次のとおりです。

- CREATE DISTINCT TYPE、あるいは DROP、COMMENT、GRANT、または REVOKE ステートメントのオブジェクトでは、データベース・マネージャーは権限 ID による修飾上の通常のプロセスを使用して、スキーマ名を決めます。 修飾の規則についての詳細は、55 ページの『非修飾の関数名、プロシージャー名、特定名、および特殊タイプ名』を参照してください。
- その他の文脈では、データベース・マネージャーは SQL パスを使用して、スキーマ名を決めます。データベース・マネージャーはパス上を順番に検索し、一致した特殊タイプを持つ最初のスキーマを選択します。 SQL パスの説明については、111 ページの『CURRENT PATH、CURRENT_PATH、または CURRENT FUNCTION PATH』を参照してください。

特殊タイプは、そのソース・タイプの関数と演算子が意味のあるものとは限らないため、それらを自動的に獲得することはありません。 (例えば、AUDIO タイプの LENGTH 関数は、オブジェクトの長さをバイトではなく、秒で戻すかもしれません。) その代わりに、特殊タイプは*強力タイプ* をサポートしています。強力タイプでは、特殊タイプ用に明示的に定義された関数と演算子だけを、その特殊タイプに適用することができるようにしています。ただし、ソース・タイプの関数や演算子は、適切なユーザー定義の関数を作成することによって適用することができます。ユーザー定義の関数は、ソース・タイプをパラメーターとして持つ既存の関数に基づいている必要があります。

特殊タイプは、そのソース・タイプと同じ制約事項に従う必要があります。例えば、1 つの表が持つことができる ROWID 列は 1 つだけです。したがって、ROWID 列を持つ表が、行 ID をソースとする特殊タイプの列を同時に持つことはできません。

データ・リンクに基づいた特殊タイプを除いて、比較演算子が特殊タイプ用に自動的に生成されます。さらに、データベース・マネージャーは、ソース・タイプから特殊タイプへ、および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特殊タイプ用の関数を自動的に生成します。例えば、前に作成された AUDIO タイプの場合、次のようなキャスト関数が作成されます。

```
FUNCTION schema-name.BLOB (schema-name.AUDIO) RETURNS BLOB (1M)

FUNCTION schema-name.AUDIO (BLOB (1M)) RETURNS AUDIO
```

データ・タイプのプロモーション

データ・タイプは、関連したデータ・タイプに分類することができます。そのようなグループ内では、あるデータ・タイプが別のデータ・タイプに優先すると考えられるような優先順位が存在します。この優先順位によって、データベース・マネージャーは、あるデータ・タイプが優先順位では下位の別のデータ・タイプに対して *プロモーション* を可能にしています。例えば、データベース・マネージャーは、データ・タイプ CHAR を VARCHAR に対して、およびデータ・タイプ INTEGER を DOUBLE PRECISION に対してプロモートすることができます。ただし、データベース・マネージャーは CLOB を VARCHAR に対してプロモートすることはできません。

データベース・マネージャーは、以下のようなときにプロモーションを検討します。

- 関数解決を実行するとき (131 ページの『関数解決』参照)
- 特殊タイプをキャストするとき (82 ページの『データ・タイプ間のキャスト』参照)
- 特殊タイプを組み込みデータ・タイプに割り当てるとき (93 ページの『特殊タイプの割り当て』参照)

81 ページの表 9 は、データベース・マネージャーがそれぞれのデータ・タイプをプロモートできる先のデータ・タイプを決める際に使用する優先順位リストを (優先順に) それぞれのデータ・タイプごとに示しています。この表は、最良の選択は同一データ・タイプであり、別のデータ・タイプにプロモーションしないことであることを示しています。また、この表は、プロモーション・プロセスでは同等であると考えられるデータ・タイプも示していることに注意してください。例えば、CHARACTER と GRAPHIC は同等のデータ・タイプであると考えられます。

表 9. データ・タイプの優先順位

データ・タイプ *	データ・タイプ優先順位リスト (高いものから低いものへの順)
CHAR または GRAPHIC	CHAR または GRAPHIC、VARCHAR または VARGRAPHIC、CLOB または DBCLOB
VARCHAR または VARGRAPHIC	VARCHAR または VARGRAPHIC、CLOB または DBCLOB
CLOB または DBCLOB	CLOB または DBCLOB
BLOB	BLOB
SMALLINT	SMALLINT、 INTEGER、 BIGINT、 DECIMAL または NUMERIC、 REAL、 DOUBLE
INTEGER	INTEGER、BIGINT、DECIMAL または NUMERIC、REAL、DOUBLE
BIGINT	BIGINT、DECIMAL または NUMERIC、REAL、DOUBLE
DECIMAL または NUMERIC	DECIMAL または NUMERIC、REAL、DOUBLE
REAL	REAL、DOUBLE
DOUBLE	DOUBLE
DATE	DATE
TIME	TIME
TIMESTAMP	TIMESTAMP
DATALINK	DATALINK
ROWID	ROWID
特殊タイプ	同一特殊タイプ
注: * リストされているデータ・タイプの他の同義語は、リストされている同義語と同じと考えられる。	

データ・タイプ間のキャスト

所定のデータ・タイプを持つ値を、別のデータ・タイプに、あるいは異なる長さ、精度、位取りを持つ同じデータ・タイプにキャストする (変更する) 必要が生じる場合がしばしばあります。 80 ページの『データ・タイプのプロモーション』で説明したように、データ・タイプのプロモーションは、あるデータ・タイプの値を新しいデータ・タイプにキャストする場合の一例です。別のデータ・タイプに変更することができるデータ・タイプは、ソース・データ・タイプからターゲット・データ・タイプへキャスト*可能* です。

あるデータ・タイプから別のデータ・タイプへのキャストは、明示的にも、暗黙的にも起こり得ます。キャスト関数または CAST 指定を使用して、明示的にデータ・タイプをキャストすることができます。データベース・マネージャーは、特殊タイプを含む割り当ての際に、暗黙的にデータ・タイプをキャストします (93 ページの『特殊タイプの割り当て』を参照してください)。さらに、ソースに基づくユーザー定義関数を作成する場合、ソース関数のパラメーターのデータ・タイプは、作成する関数のデータ・タイプに対してキャスト可能である必要があります (446 ページの『CREATE FUNCTION』を参照してください)。

文字ストリングまたはグラフィック・ストリングを別のデータ・タイプにキャストする際に切り捨てが生じた場合は、非ブランクの文字が切り捨てられた場合には、警告が出されます。 この切り捨て動作は、非ブランクの何らかの文字が切り捨てられる場合にエラーが起こった際に、文字ストリングまたはグラフィック・ストリングをターゲットに割り当てることとは似ていません。

キャストする先、あるいはキャストする元のいずれかデータ・タイプとして特殊タイプを含むキャストについて、表 10 はサポートされるキャストを示しています。組み込みデータ・タイプ間のキャストに関しては、 83 ページの表 11 がサポートされるキャストを示しています。

表 10. 特殊タイプが含まれる場合のサポートされるキャスト

データ・タイプ ...	キャスト可能な相手先のデータ・タイプ ...
特殊タイプ DT	特殊タイプ DT のソース・データ・タイプ
特殊タイプ DT のソース・データ・タイプ	特殊タイプ DT
特殊タイプ DT	特殊タイプ DT
データ・タイプ A	特殊タイプ DT (A が特殊タイプ DT のソース・データ・タイプにプロモーション可能な場合) (80 ページの『データ・タイプのプロモーション』参照)
INTEGER	特殊タイプ DT (DT のソース・タイプが SMALLINT の場合)
DOUBLE	特殊タイプ DT (DT のソース・タイプが REAL の場合)
VARCHAR または VARGRAPHIC	特殊タイプ DT (DT のソース・タイプが CHAR または GRAPHIC の場合)

スキーマ名で明示的に修飾されていない特殊タイプがキャストに含まれている場合、データベース・マネージャーは SQL パスを使用してスキーマ名を決めます。データベース・マネージャーは、SQL パス上で、その名前による特殊タイプを持った最初のスキーマの名前を選択します。SQL パスの詳細については、 57 ページの『スキーマと SQL パス』 を参照してください。

データ・タイプ間のキャスト

次の表は、データ・タイプ間でサポートされるキャストを示しています。

表 11. 組み込みデータ・タイプ間でサポートされるキャスト

ソース・データ・タイプ	SMALLINT INTEGER BIGINT	DECIMAL NUMERIC	REAL DOUBLE	CHAR VARCHAR CLOB	GRAPHIC VARGRAPHIC DBCLOB	DATE	TIME	TIME STAMP	BLOB	ROW ID
SMALLINT	Y	Y	Y	Y	--	--	--	--	--	--
INTEGER	Y	Y	Y	Y	--	--	--	--	--	--
BIGINT	Y	Y	Y	Y	--	--	--	--	--	--
DECIMAL	Y	Y	Y	Y	--	--	--	--	--	--
NUMERIC	Y	Y	Y	Y	--	--	--	--	--	--
REAL	Y	Y	Y	Y	--	--	--	--	--	--
DOUBLE	Y	Y	Y	Y	--	--	--	--	--	--
CHAR	Y	Y	Y	Y	*	Y	Y	Y	Y	Y
VARCHAR	Y	Y	Y	Y	*	Y	Y	Y	Y	Y
CLOB	Y	Y	Y	Y	*	--	--	--	Y	--
GRAPHIC	--	--	--	Y*	Y	--	--	--	Y	--
VARGRAPHIC	--	--	--	Y*	Y	--	--	--	Y	--
DBCLOB	--	--	--	Y*	Y	--	--	--	Y	--
DATE	--	--	--	Y**	--	Y	--	Y	--	--
TIME	--	--	--	Y**	--	--	Y	Y	--	--
TIMESTAMP	--	--	--	Y**	--	Y	Y	Y	--	--
BLOB	--	--	--	--	--	--	--	--	Y	--
ROWID	--	--	--	Y	--	--	--	--	Y	Y

注: * 変換は、UCS-2 グラフィックの場合にのみサポートされます。

** DATE、TIME、TIMESTAMP、または ROWID から CLOB へのキャストはサポートされていません。

DATALINK だけを DATALINK タイプにキャストすることができます。

次の表は、データ・タイプへのキャストの規則を示しています。

データ・タイプ間のキャスト

表 12. データ・タイプへのキャストの規則

ターゲット・データ・タイプ	ソース・データ・タイプ	規則
SMALLINT	任意	SMALLINT スカラー関数を参照
INTEGER	任意	INTEGER スカラー関数を参照
BIGINT	任意	BIGINT スカラー関数を参照
DECIMAL	任意	DECIMAL スカラー関数を参照
NUMERIC	任意	ZONED スカラー関数を参照
REAL	任意	REAL スカラー関数を参照
DOUBLE	任意	DOUBLE スカラー関数を参照
CHAR	任意	CHAR スカラー関数を参照
VARCHAR	任意	VARCHAR スカラー関数を参照
CLOB	任意	CLOB スカラー関数を参照
GRAPHIC	任意	ホスト変数へのストリング割り当てを参照
VARGRAPHIC	任意	ホスト変数へのストリング割り当てを参照
DBCLOB	任意	DBCLOB スカラー関数を参照
DATE	任意	DATE スカラー関数を参照
TIME	任意	TIME スカラー関数を参照
TIMESTAMP	CHAR	TIMESTAMP スカラー関数を参照。1 つのオペランドが指定される。
TIMESTAMP	DATE	タイム・スタンプは、指定された日付と、時刻 00:00:00 から構成される。
TIMESTAMP	TIME	タイム・スタンプは、CURRENT_DATE と指定された時刻で構成される。
BLOB	任意	BLOB スカラー関数を参照
DATALINK	DATALINK	データ・リンク割り当てを参照
ROWID	任意	ROWID スカラー関数を参照

割り当ておよび比較

SQL の基本演算は、割り当てと比較です。割り当て演算は、 CALL、 INSERT、 UPDATE、 FETCH、 SELECT、 SET 変数、および VALUE INTO ステートメントの実行時に行われます。比較演算は、述部や他の言語エレメント (MAX、MIN、DISTINCT、GROUP BY、ORDER BY など) が入っているステートメントを実行する過程で行われます。

これらの演算はいずれも、演算に使用するオペランド間でデータ・タイプに互換性がなければならないという基本的な規則があります。この互換性の規則は、UNION、連結、CASE 式、および CONCAT、 VALUE、 COALESCE、 IFNULL、 MIN、 MAX スカラー関数にも適用されます。互換性マトリックスは、次のとおりです。

割り当ておよび比較

表 13. データ・タイプの互換性

オペランド	2 進整数	10 進数[4]	浮動小数点数	文字ストリング	グラフィック・ストリング	2 進ストリング	日付	時刻	タイム・スタンプ	特殊タイプ
2 進整数	あり	あり	あり	なし	なし	なし	なし	なし	なし	2
10 進数	あり	あり	あり	なし	なし	なし	なし	なし	なし	2
浮動小数点数	あり	あり	あり	なし	なし	なし	なし	なし	なし	2
文字ストリング	なし	なし	なし	あり	あり 5	なし 3	1	1	1	2
グラフィック・ストリング	なし	なし	なし	あり 5	あり	なし	なし	なし	なし	2
2 進ストリング	なし	なし	なし	なし 3	なし	あり	なし	なし	なし	2
日付	なし	なし	なし	1	なし	なし	あり	なし	なし	2
時刻	なし	なし	なし	1	なし	なし	なし	あり	なし	2
タイム・スタンプ	なし	なし	なし	1	なし	なし	なし	なし	あり	2
特殊タイプ	2	2	2	2	2	2	2	2	2	2

注:

1. 日付/時刻の値と文字ストリングとの互換性は、割り当て、比較、および VALUE、 COALESCE、 IFNULL、 MIN、 MAX スカラー関数に限定されます。
   - 日付/時刻の値は、 91 ページの『日付/時刻の割り当て』で説明しているように、文字ストリングの列や文字ストリングの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現は、日付の列へ割り当てたり、日付と比較したり、あるいは日付として VALUE、COALESCE、IFNULL、MIN、または MAX スカラー関数で使用することができます。
   - 時刻の有効なストリング表現は、時刻の列へ割り当てたり、時刻と比較したり、あるいは時刻として VALUE、COALESCE、IFNULL、MIN、または MAX スカラー関数で使用することができます。
   - タイム・スタンプの有効な表現は、タイム・スタンプの列へ割り当てたり、タイム・スタンプと比較したり、あるいはタイム・スタンプとして VALUE、COALESCE、IFNULL、MIN、または MAX スカラー関数で使用することができます。
2. 特殊タイプの値は、同じ特殊タイプで定義された値とのみ比較が可能です。データベース・マネージャーは、一般に、特殊タイプの値とそのソース・データ・タイプ間の割り当てをサポートしています。詳細については、 93 ページの『特殊タイプの割り当て』を参照してください。
3. すべての文字ストリングは、FOR BIT DATA サブタイプの場合でも、2 進ストリングとは互換性がありません。
4. 10 進数とは、パック 10 進数とゾーン 10 進数の両方を指します。
5. ビット・データとグラフィック・ストリングは互換性がありません。
6. DATALINK オペランドは、別の DATALINK オペランドに対してのみ割り当てることができます。 DATALINK 値は、列が NO LINK CONTROL で定義されているか、ファイルが存在しており、まだファイル・リンク制御されていない場合にのみ、その列に割り当てることができます。 DATALINK オペランドは、いずれのデータ・タイプとも、直接に比較することができます。
   DLCOMMENT、DLLINKTYPE、DLURLCOMPLETE、DLURLPATH、DLURLPATHONLY、DLURLSCHEME、および DLURLSERVER スカラー関数を使用して、データ・リンクから文字ストリング値を取り出すことができます。これによって、そのデータ・リンクは他のストリングと比較することができるようになります。
7. ROWID オペランドは、別の ROWID オペランドに対してのみ割り当てることができます。

割り当て演算には、ヌル値を入れることのできない列、および関連する標識変数を持たないホスト変数にヌル値を割り当てることはできないという基本的な規則があります。標識変数についての説明は、121 ページの『ホスト変数に対する参照』を参照してください。

数値の割り当て

数値の割り当てには、10 進数の整数部分、または整数の整数部が切り捨てられることはないという基本的な規則があります。10 進数の小数部分は、必要があれば切り捨てることができます。ホスト変数へ数値を割り当てる場合に、SQLCODE に正の値が戻されることがあります。

次のような場合には、エラーが生じます。

- 列への割り当てで、数値の整数部分の切り捨てが起こる。
- 標識変数を持たないホスト変数への割り当てで、数値の整数部分の切り捨てが起こる。

次のような場合には、警告が出されます。

標識変数を持つホスト変数への割り当てで、数値の整数部分の切り捨てが起こる。この場合、数値はホスト変数に割り当てられず、標識変数は -2 にセットされます。

注: 10 進数とは、パック 10 進数とゾーン 10 進数の両方を指します。

注: ファイルから取り出す 10 進数データが SQL CREATE TABLE ステートメントで作成されたものではない 場合は、10 進数フィールドに無効なデータが含まれていることがあります。この場合、そのデータは保管されるのと同じように戻され、警告あるいはエラー・メッセージは出されません。SQL CREATE TABLE ステートメントによって作成された表には、無効な 10 進数データが含まれることを許しません。

浮動小数点に対する 10 進数または整数の割り当て

浮動小数点数は、実数の近似値です。したがって、10 進数または整数を浮動小数点数の列または変数に割り当てた場合に、その結果が元の数値と異なる可能性があります。

受取側の列または変数が、単精度 (32 ビット) ではなく、倍精度 (64 ビット) として定義した方が近似値はより正確になります。

整数に対する浮動小数点数または 10 進数の割り当て

10 進数または浮動小数点数を 2 進整数の列、または変数に割り当てると、必要に応じて、数値の精度および位取りが、ターゲットの精度や位取りに変換されます。ターゲットの位取りがゼロの場合、数値の小数部分は失われます。必要な数の先行ゼロが追加または除去され、その数値の小数部分に、必要な数の後続ゼロが追加されるか、または必要な数の後続数字が除去されます。

10 進数に対する 10 進数の割り当て

10 進数を 10 進数の列または変数に割り当てると、必要に応じて、数値の精度および位取りが、ターゲットの精度や位取りに変換されます。必要な数の先行ゼロが追

加または除去され、その数値の小数部分に、必要な数の後続ゼロが追加されるか、または必要な数の後続数字が除去されます。

10 進数に対する整数の割り当て

整数を 10 進数の列または変数に割り当てると、最初にその整数が一時的な 10 進数に変換され、次に、必要があれば、その一時的な 10 進数の精度および位取りが、ターゲットの精度や位取りに変換されます。整数の位取りがゼロの場合、一時的な 10 進数の精度は 5,0 (短整数の場合)、11,0 (長整数の場合)、または 19,0 (64 ビット整数の場合) になります。

10 進数に対する浮動小数点数の割り当て

浮動小数点数が 10 進数の列または変数に割り当てられる場合、その浮動小数点数は、まず一時的に精度 31 の 10 進数に変換され、その後、必要があれば、その 10 進数の精度および位取りが、ターゲットの精度および位取りに変換されます。この変換では、数値が 31 桁の 10 進数の精度に丸められます (浮動小数点数演算を使用)。その結果、$0.5*10^{-31}$ は 0 に減少します。位取りには、有効数字を失わずにその数値の整数部分を表せる範囲内で最大の値が与えられます。

COBOL および RPG の整数に対する割り当て

COBOL および RPG の短整数または長整数のホスト変数への割り当てでは、そのホスト変数に指定された位取りが考慮されます。ただし、整数のホスト変数への割り当てでは、整数の全桁が使用されます。したがって、COBOL のデータ項目や RPG のフィールドに入っている値が、ホスト変数に関して指定された最大の精度を超える場合があります。

例えば、COL1 に 12345 という値が入っている場合、

```
01  A  PIC  S9999  BINARY.
EXEC SQL SELECT COL1
         INTO :A
         FROM TABLEX
END-EXEC.
```

というステートメントは、A が 4 桁しか定義されていなくても、結果として A には 12345 という値が入ります。

次のような COBOL ステートメントでは、

```
MOVE 12345 TO A.
```

A に 2345 が入るので注意が必要です。

ストリングの割り当て

2 進ストリングの割り当て

2 進ストリングの割り当てには、次の 2 つのタイプがあります。

- *記憶域割り当て*。値が関数またはストアード・プロシージャーの列またはパラメーターに割り当てられた場合です。
- *検索割り当て*。値がホスト変数に割り当てられた場合です。

記憶域割り当て: 基本的な規則では、関数またはプロシージャーの列またはパラメーターに割り当てるストリングの長さが、その列またはパラメーターの長さ属性を

超えてはなりません。ストリングが、その列の長さ属性よりも長い場合には、負の SQLCODE が戻されます。SQLCA の説明については、 865 ページの『付録 B. SQL 連絡域』を参照してください。

検索割り当て: ホスト変数に割り当てるストリングの長さは、そのホスト変数の長さ属性を超えても構いません。変数へのストリングの割り当てで、ストリングの長さがその変数の長さ属性よりも大きい場合、必要な数の文字だけを残してストリングの右側が切り捨てられます。この切り捨てが行われると、SQLCA の SQLWARN1 フィールドに値として 'W' が割り当てられます。

長さ *n* のストリングを、*n* より大きい最大長を持つ可変長ストリングの変数に割り当てた場合は、変数の *n* 番目のバイト以後のバイトは未定義になります。

文字およびグラフィック・ストリングの割り当て

以下の規則は、値のソース (送り側) とターゲット (受け取り側) がいずれもストリングである場合に適用されます。日付/時刻のデータ・タイプについては、 91 ページの『日付/時刻の割り当て』を参照してください。

文字およびグラフィック・ストリングの割り当てには、次の 2 つのタイプがあります。

- *記憶域割り当て*。値が関数またはストアード・プロシージャーの列またはパラメーターに割り当てられた場合です。
- *検索割り当て*。値がホスト変数に割り当てられた場合です。

記憶域割り当て: 基本的な規則では、関数またはプロシージャーの列またはパラメーターに割り当てるストリングの長さが、その列またはパラメーターの長さ属性を超えてはなりません。ストリングが、その列の長さ属性よりも長い場合には、負の SQLCODE が戻されます。(後書きブランクは、通常ストリングの長さに含まれます。ただし、記憶域割り当ての場合には、後続ブランクはそのストリングの長さには含まれません。)

SQLCA の説明については、 865 ページの『付録 B. SQL 連絡域』を参照してください。

固定長ストリングの列またはパラメーターへストリングを割り当てる際に、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には、そのストリングの右側に必要な数の 1 バイト文字、2 バイト文字、または UCS-2 文字のブランクが埋め込まれます。[18]埋め込み文字は、ビット・データの場合でも、常にブランクです。

検索割り当て: ホスト変数に割り当てるストリングの長さは、そのホスト変数の長さ属性を超えても構いません。変数へのストリングの割り当てで、ストリングの長さがその変数の長さ属性よりも大きい場合、必要な数の文字だけを残してストリングの右側が切り捨てられます。この切り捨てが行われると、SQLCA の SQLWARN1 フィールドに値として 'W' が割り当てられます。さらに、変数に標識変数がある場合は、その標識変数にストリングの元の長さがセットされます。 C の NUL 終了ホスト変数で NUL 終了文字だけが切り捨てられ、*NOCNULRQD オプションが、CRTSQLCI または CRTSQLCPPI コマンド (あるいは、SET OPTION ステートメン

18. UCS-2 は、コード・ポイント X'0020' および X'3000' でブランク文字を定義します。データベース・マネージャーは、コード・ポイント X'0020' をブランクで埋め込みます。

トでの CNULRQD(*NO)) で指定されていた場合は、値として 'N' が SQLCA の SQLWARN1 フィールドに入り、NUL 終了文字は変数には入りません。

固定長変数へストリングを割り当てる際に、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には、そのストリングの右側に必要な数の 1 バイト文字、2 バイト文字、または UCS-2 文字のブランクが埋め込まれます。[18]埋め込み文字は、ビット・データの場合でも、常にブランクです。

長さ *n* のストリングを、*n* より大きい最大長を持つ可変長ストリングの変数に割り当てた場合は、変数の *n* 番目以後の文字は未定義になります。

混合ストリングを含む割り当て: ストリングに混合データが入っている場合は、割り当て規則によって、一連の 2 バイト・コードの途中で切り捨てが必要になることがあります。一連の 2 バイト文字の終わりを示すシフトイン文字の消失を防ぐために、ストリングの終わりから追加の文字が切り捨てられ、シフトイン文字が追加されます。この切り捨ての結果、シフトアウト文字とそれに対応するシフトイン文字で囲まれている文字数は、常に偶数バイトになります。

C の NUL 終了ストリングを含む割り当て: 長さ *n* のストリングが、*n+1* を超える長さの C の NUL 終了ストリング変数に割り当てられる場合、

- *CNULRQD オプションが、CRTSQLCI または CRTSQLCPPI コマンド (あるいは、SET OPTION ステートメントでの CNULRQD(*YES)) で指定されていた場合は、そのストリングの右側が x-n-1 個のブランクで埋め込まれます。ここで、x は変数の長さです。埋め込みが行われたストリングが変数に割り当てられ、NUL 終了文字が次の文字位置に入れられます。
- *NOCNULRQD オプションが、CRTSQLCI または CRTSQLCPPI コマンド (あるいは、SET OPTION ステートメントでの CNULRQD(*NO)) で指定されていた場合は、ストリングの右側の埋め込みはありません。そのストリングが変数に割り当てられ、NUL 終了文字が次の文字位置に入れられます。

割り当ての際の変換規則: 列またはホスト変数に割り当てられるストリングは、必要に応じて、最初にターゲットのコード化文字セットに変換されます。文字変換が必要になるのは、以下の条件にすべて該当する場合だけです。

- 両者の CCSID が異なっている。
- どちらの CCSID も 65535 ではない。
- ストリングがヌルでなく、空でもない。
- CCSID 変換選択表に、変換が必要であることが示されている。

次のような場合には、エラーが生じます。

- CCSID 変換選択表が使用されるとき、その CCSID 変換表に CCSID の対に関する情報が入っていなかった。
- 列への割り当てまたは標識変数を持たないホスト変数への割り当て演算で、ストリングの文字を変換できなかった。例えば、2 バイト文字 (DBCS) を、1 バイト文字 (SBCS) の CCSID を持つ列やホスト変数に変換できなかったような場合。

次のような場合には、警告が出されます。

- ストリングの文字が、置換文字に変換された。

- 標識変数を持たないホスト変数への割り当て演算で、ストリングの文字を変換できなかった。例えば、DBCS 文字は、SBCS CCSID をもつホスト変数には変換できません。この場合は、ホスト変数にはストリングが割り当てられず、標準変数に -2 がセットされます。

日付/時刻の割り当て

日付 (DATE) の列に割り当てる値は、日付または日付の有効なストリング表現でなければなりません。日付を割り当てることができるのは、日付 (DATE) の列、文字ストリングの列、文字ストリング変数、または ILE RPG/400 タイム・スタンプ変数だけです。TIME の列に割り当てる値は、時刻、または時刻の有効なストリング表現のいずれかでなければなりません。時刻を割り当てることができるのは、時刻 (TIME) の列、文字ストリングの列、文字ストリング変数、または ILE RPG/400 タイム・スタンプ変数だけです。タイム・スタンプ (TIMESTAMP) の列に割り当てる値は、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現のいずれかでなければなりません。タイム・スタンプを割り当てることができるのは、TIMESTAMP の列、文字ストリングの列、文字ストリング変数、または ILE RPG/400 タイム・スタンプ変数だけです。

日付/時刻の値が文字ストリングの変数や列に割り当てられる場合、その値はストリング表現に変換されます。日付、時刻、タイム・スタンプのどの部分からも、先行ゼロは除去されません。ターゲットの必要な長さは、ストリング表現の形式に応じて変わります。ターゲットの長さが必要な長さよりも長い場合は、変換結果の右側にブランクが埋め込まれます。ターゲットの長さが必要な長さよりも短い場合は、その結果は、関与する日付/時刻の値のタイプとターゲットのタイプによって変わります。

- ターゲットが文字ストリングの列である場合、切り捨ては許されません。以下の規則が適用されます。

  DATE

  日付の形式が、*ISO、USA、*EUR、または *JIS の場合、列の長さ属性は 10 以上でなければなりません。日付の形式が *YMD、*MDY、または *DMY の場合、列の長さ属性は 8 以上でなければなりません。日付の形式が *JUL の場合は、ホスト変数の長さは 6 以上でなければなりません。

  TIME

  ターゲットの列の長さ属性は、8 以上でなければなりません。

  TIMESTAMP

  ターゲットの列の長さ属性は、26 以上でなければなりません。

- ターゲットがホスト変数である場合は、次の規則が適用されます。

  DATE

  日付の形式が *ISO、*USA、*EUR、または *JIS の場合、ホスト変数の長さは 10 以上でなければなりません。日付の形式が *YMD、*MDY、または *DMY の場合、ホスト変数の長さは 8 以上でなければなりません。日付の形式が *JUL の場合、ホスト変数の長さは 6 以上でなければなりません。

  TIME

  – *USA 形式を使用するときは、ホスト変数の長さは 8 以上でなければなりません。この形式には、秒は含まれていません。

– *ISO、*EUR、*JIS、または *HMS の時刻形式を使用している場合は、ホスト変数の長さは、5 以上でなければなりません。変数の長さが 5、6、または 7 の場合、時刻の秒の部分が結果から除去され、SQLWARN1 に 'W' がセットされます。この場合、標識変数があれば、時刻の秒の部分はその標識変数に割り当てられます。また、長さが 6 または 7 の場合には、変数の値が時刻の有効なストリング表現になるように、ブランクの埋め込みが行われます。

TIMESTAMP

ホスト変数の長さは、19 以上でなければなりません。長さが 19 から 25 までの場合、タイム・スタンプはストリングのように切り捨てられ、マイクロ秒の部分の 1 つまたは複数の桁が除去されます。長さが 20 の場合は、変数の値がタイム・スタンプの有効なストリング表現になるように、後書き小数点がブランクに置き換えられます。

データ・リンク割り当て

値をデータ・リンク列に割り当てると、値のリンケージ属性が空であるか、列が NO LINK CONTROL で定義されているのではない限り、ファイルへのリンクが確立します。リンクされた値がすでに列に存在する場合は、そのファイルはリンク解除されます。また、リンクされた値がすでに存在する場合にヌル値を割り当てても、古い値に関連したファイルはリンク解除されます。

列にすでに存在するものと同じデータ位置を、アプリケーションが与えると、そのリンクは保存されます。このことが生じる理由は 2 つあります。

- コメントが変更された。
- 表がリンク保留状態にある場合は、列にあるものと同じリンク属性を与えることによって、表のリンクを復元することができます。

データ・リンク値は、DLVALUE スカラー関数を使用することによって、列に割り当てることができます。 DLVALUE スカラー関数は、後で列に割り当てることができる新しいデータ・リンク値を作成します。値がコメントしか含んでいないか、URL がまったく同じではない限り、割り当ての行動によってファイルへリンクします。

値をデータ・リンク列へ割り当てる際に、次のエラー状態が起こる可能性があります。

- データ・ロケーション (URL) 形式が無効。
- ファイル・サーバーがこのデータベースに登録されていない。
- 無効なリンク・タイプが指定された。
- コメントまたは URL の長さが無効。

  URL パラメーターまたは関数結果のサイズは、入力と出力の両方で同じであり、データ・リンク列の長さに制限されることに注意してください。ただし、場合によっては、戻される URL 値にアクセス・トークンが付加されることがあります。このような可能性がある状態では、出力の場所ではアクセス・トークン用に十分な記憶域とデータ・リンク列に十分な長さが必要になります。したがって、入力で用意される完全に拡張された形式でのコメントと URL の実際の長さを、出力の記憶域に適応するように制限する必要があります。制限された長さを超えた場合には、このエラーが起こります。

割り当てでリンクも作成する場合は、以下のエラーが生じる可能性があります。

- 現在、ファイル・サーバーが使用不能。
- ファイルが存在しない。
- 参照したファイルがリンク用にアクセスできない。
- ファイルがすでに別の列へリンクされている。

  このエラーは、異なるリレーショナル・データベースへのリンクの場合でも生じることに注意してください。

さらに、割り当てで既存のリンクを取り除く場合には、以下のエラーが生じる可能性があります。

- 現在、ファイル・サーバーが使用不能。
- 参照保全制御を伴うファイルが、DB2 UDB 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にない。

データ・リンク値は、スカラー関数 (DLLINKTYPE および DLURLPATH など) を使用することにより、データベースから検索することができます。その後で、これらのスカラー関数の結果をホスト変数に割り当てることができます。

通常は、検索時にファイル・サーバーをアクセスすることは行われないということに注意してください。[19] したがって、後でファイル・システム・コマンドを使用してファイル・サーバーにアクセスしようとすると、失敗する可能性があります。

表はリンク保留状態にあるため、データ・リンク値を検索すると警告が戻されることがあります。

行 ID の割り当て

行 ID の値を割り当てることができるのは、行 ID データ・タイプを持つ列、パラメーター、またはホスト変数のみです。 ROWID 列の値が有効であるためには、その列が GENERATED BY DEFAULT として定義されているか、OVERRIDING SYSTEM VALUE が指定されていることが必要です。 ROWID 列のある各表には、各 ROWID 値をすべて固有の値にするための固有制約が暗黙的に追加されます。この列に指定する値は、DB2 UDB (OS/390 および z/OS 版) または DB2 UDB for iSeries によりすでに生成されている有効な行 ID 値でなければなりません。

特殊タイプの割り当て

特殊タイプ のホスト変数への割り当てに適用される規則は、他のすべての特殊タイプを含んだ割り当ての規則とは異なります。

19. パスに関連したプレフィックス名を決めるためには、ファイル・サーバーへのアクセスが必要になることがあります。これは、ファイル・サーバーのマウント・ポイントを移動するとファイル・サーバー側で変更することができます。サーバーのファイルを最初にアクセスすると、必要とする値をファイル・サーバーから検索し、そのファイル・サーバーのデータ・リンク値を後で検索するために、データベース・サーバーでキャッシュに入れます。ファイル・サーバーをアクセスできない場合には、エラーが戻されます。

ホスト変数への割り当て

特殊タイプのホスト変数への割り当ては、特殊タイプのソース・データ・タイプに基づいています。したがって、特殊タイプのソース・データ・タイプをホスト変数へ割り当て可能な場合にのみ、特殊タイプの値はホスト変数に割り当てることができます。

例: 特殊タイプ AGE が以下の SQL によって作成され、表 STUDENT の STU_AGE 列が特殊タイプで定義されているものと想定します。表 CL_SCHED を使用して、当日後刻に開始される (STARTING) すべてのクラス (CLASS_CODE) を選択します。当日のクラスは、DAY 列に 3 の値を持っています。

```
CREATE DISTINCT TYPE AGE AS SMALLINT WITH COMPARISONS
```

次に、生徒の年齢を、INTEGER データ・タイプを持つホスト変数 HV_AGE に有効に割り当てることを考えます。

```
SELECT  STU_AGE INTO :HV_AGE FROM STUDENTS WHERE STU_NUMBER = 200
```

特殊タイプ (SMALLINT) のソース・データ・タイプがホスト変数 (INTEGER) に割り当て可能なため、特殊タイプの値はホスト変数 HV_AGE に割り当てることができます。特殊タイプ AGE が CHAR(5) のような文字データ・タイプをソースとしていた場合には、文字タイプは整数タイプに割り当てることはできないため、上記の割り当ては無効になります。

ホスト変数以外への割り当て

特殊タイプは、割り当てのソースあるいはターゲットのいずれにもなり得ます。割り当ては、割り当てられる値のデータ・タイプが、ターゲットのデータ・タイプにキャスト可能であるかどうかに基づいています。 82 ページの『データ・タイプ間のキャスト』は、特殊タイプが含まれる場合にサポートされるキャストを示しています。結果としては、特殊タイプの値は、以下の場合にホスト変数以外のいずれのターゲットにも割り当てることができます。

- 割り当てのターゲットが同じ特殊タイプを持っている、あるいは
- 特殊タイプはターゲットのデータ・タイプにキャスト可能である。

以下の場合には、いずれの値も特殊タイプに割り当てることができます。

- 割り当てられる値がターゲットと同じ特殊タイプを持っている、あるいは
- 割り当てられる値のデータ・タイプは、ターゲットの特殊タイプにキャスト可能である。

例: 特殊タイプ AGE のソース・データ・タイプが SMALLINT であるとします。

```
CREATE DISTINCT TYPE AGE AS SMALLINT WITH COMPARISONS
```

次に、2 つの表 TABLE1 および TABLE2 が 4 つの同一の列記述で作成されたとします。

```
AGECOL    AGE
SMINTCOL  SMALLINT
INTCOL    INTEGER
DECCOL    DEC(6,2)
```

次の SQL ステートメントを使用し、X と Y にいろいろな値を代入して、TABLE1 のいろいろな列に TABLE2 から値を挿入する場合、 95 ページの表 14 はその割り当てが有効であるかどうかを示しています。

```
INSERT INTO  TABLE1 (Y) SELECT X FROM TABLE2
```

表 14. いろいろな割り当ての評価 (例えば、INSERT で)

TABLE2.X	TABLE1.Y	有効	理由
AGECOL	AGECOL	はい	ソースとターゲットが同じ特殊タイプ
SMINTCOL	AGECOL	はい	SMALLINT は AGE にキャスト可能 (AGE のソース・タイプは SMALLINT のため)
INTCOL	AGECOL	はい	INTEGER は AGE にキャスト可能 (AGE のソース・タイプは SMALLINT のため)
DECCOL	AGECOL	いいえ	DECIMAL は AGE にキャスト不可能
AGECOL	SMINTCOL	はい	AGE はそのソース・タイプである SMALLINT にキャスト可能
AGECOL	INTCOL	いいえ	AGE は INTEGER にキャスト不可能
AGECOL	DECCOL	いいえ	AGE は DECIMAL にキャスト不可能

数値の比較

数値は代数の場合と同じように比較されます。つまり、符号が考慮されます。例えば、-2 は +1 より小さくなります。

一方の数値が整数で、もう一方の数値が 10 進数の場合、その整数を 10 進数に変換した整数の一時的なコピーを使用して比較が行われます。

10 進数または位取りがゼロ以外の 2 進数を比較するときに、比較する数値の位取りが異なる場合は、一方の数値の一時的なコピー (両者の小数部が同じ桁数になるように、一方の数値の小数部の桁数を後続ゼロによって増やしたもの) を使用して比較が行われます。

一方の数値が浮動小数点数で、もう一方の数値が整数、10 進数、または単精度の浮動小数点数である場合は、2 番目の数値を倍精度の浮動小数点数に変換し、その一時的なコピーを使用して比較が行われます。ただし、単精度浮動小数点の列を定数と比較するときに、その定数が単精度の浮動小数点数で表される場合は、定数の単精度形式を使用して比較が行われます。

2 つの浮動小数点数は、両者の正規形のビット構成が同一である場合にのみ等しくなります。

ストリングの比較

2 進ストリングの比較

2 進ストリングの比較では、ソート順序は常に *HEX を使用し、各ストリングの対応するバイトが比較されます。さらに、2 つのストリングの長さが同一の場合のみ、2 つの 2 進ストリングは等しいと言えます。

文字およびグラフィック・ストリングの比較

文字および UCS-2 グラフィック・ストリングの比較では、実際には、すべての SBCS データと混合データの単一バイト部分についてステートメントが実行される時点で有効なソート順序を使用します。ソート順序が *HEX の場合、各ストリングの対応するバイトが比較されます。その他のソート順序では、各ストリングの対応するバイトの重み付けされた値が比較されます。ストリングの長さが異なる場合は、短い方のストリングの右側にブランクを埋め込んだ一時的コピーを使用して比較します。この埋め込みを行うのは、両方のストリングの長さを等しくするためです。埋め込み文字は、ソート順序に関係なく、常にブランクです。ビット・データの場合も、ブランクを使用します。DBCS グラフィック・データの場合は、埋め込み文字は DBCS のブランク (x'4040') になります。 UCS-2 グラフィック・データの場合は、埋め込み文字は UCS-2 ブランクになります。[20]

2 つのストリングが等しくなるのは、次のいずれかに該当する場合です。

- 両方のストリングが空である。
- *HEX のソート順序を使用した場合、対応するバイトがすべて等しい。
- *HEX 以外のソート順序を使用した場合、対応するバイトの重み付けの値がすべて等しい。

空のストリングは、ブランクのストリングと同じです。等しくない 2 つのストリング間の関係は、それらのストリングの左端から調べて、最初に見つかった等しくないバイトの対の比較によって決定されます。この比較は、そのステートメントが実行される時点で有効なソート順序に従って行われます。

異なる長さを持つ 2 つの可変長ストリングが後書きブランクの数だけが異なる場合には、等しくなります。そのような値の集合から 1 つの値を選択する演算では、選択される値は不定のものになります。このような不定な選択を伴う演算には、DISTINCT、MAX、MIN、UNION、およびグループ化列の参照があります。グループ化列の参照に伴う不定な選択については、GROUP BY の項で詳しく説明されています。

比較の際の変換規則: 2 つのストリングを比較する場合、必要があれば、最初に一方のストリングがもう一方のストリングのコード化文字セットに変換されます。文字変換が必要になるのは、以下の条件にすべて該当する場合だけです。

- 2 つのストリングの CCSID が異なっている。
- どちらの CCSID も 65535 ではない。
- 変換する側として選択されたストリングが、ヌルでも空でもない。
- CCSID 変換選択表に、変換が必要であることが示されている。

コード化体系が異なる 2 つのストリングを比較する場合、オペランドが同一タイプであれば、以下のように、必要な変換が SBCS に適用されます。

20. UCS-2 は、コード・ポイント X'0020' および X'3000' でブランク文字を定義します。データベース・マネージャーは、コード・ポイント X'0020' の位置にあるブランクを埋め込みに使用します。

表 15. 文字変換のためのコード化体系の選択

	第 2 オペランド			
第 1 オペランド	**SBCS データ**	**DBCS データ**	**混合データ**	**UCS-2 データ**
SBCS データ	下記参照	第 2 オペランド	第 2 オペランド	第 2 オペランド
DBCS データ	第 1 オペランド	下記参照	第 2 オペランド	第 2 オペランド
混合データ	第 1 オペランド	第 1 オペランド	下記参照	第 2 オペランド
UCS-2 データ	第 1 オペランド	第 1 オペランド	第 1 オペランド	下記参照

それ以外の場合は、それぞれのオペランドのタイプに応じて、選択されるオペランドが決まります。以下の表は、オペランドのタイプに応じて、どちらのオペランドが変換されるオペランドとして選択されるかを示しています。

表 16. 文字変換するオペランドの選択

	第 2 オペランド				
第 1 オペランド	**列値**	**演算による値**	**特殊レジスター**	**定数**	**ホスト変数**
列値	第 2 オペランド	第 2 オペランド	第 2 オペランド	第 2 オペランド	第 2 オペランド
演算による値	第 1 オペランド	第 2 オペランド	第 2 オペランド	第 2 オペランド	第 2 オペランド
特殊レジスター	第 1 オペランド	第 1 オペランド	第 2 オペランド	第 2 オペランド	第 2 オペランド
定数	第 1 オペランド	第 1 オペランド	第 1 オペランド	第 2 オペランド	第 2 オペランド
ホスト変数	第 1 オペランド	第 1 オペランド	第 1 オペランド	第 1 オペランド	第 2 オペランド

外部コード化体系のデータを含むホスト変数は、何らかの演算で使用される前に、必ず固有のコード化体系に変換されます。上記の規則は、このような変換がすでに行われていることを前提にしています。

ストリングの文字が変換できない場合や、CCSID 変換選択表を使用するときに、表に CCSID の対に関する情報が入っていない場合には、エラーが生じます。ストリングの文字が置換文字に変換された場合は、警告が出されます。

日付/時刻の比較

日付 (DATE)、時刻 (TIME)、またはタイム・スタンプ (TIMESTAMP) の値は、同じデータ・タイプを持つ他の値、または該当するデータ・タイプのストリング表現と比較することができます。比較はすべて日時順に行われます。つまり、0001 年 1 月 1 日からの経過日数 (時間) が多ければ多いほど、より大きな 値になります。

TIME の値および時刻の値のストリング表現を含む比較では、必ず秒も比較されます。ストリング表現で秒の部分を除去している場合は、その部分にゼロ秒があるものとして扱われます。時刻 24:00:00 は、時刻 00:00:00 より大きいものとして比較します。

TIMESTAMP の値に関する比較は、等しいと考えられるような表現であっても、それを考慮せずに日時順で行われます。したがって、次のような述部が成り立ちます。

```
TIMESTAMP('1990-02-23-00.00.00') > '1990-02-22-24.00.00'
```

特殊タイプの比較

特殊タイプの値は、同じ特殊タイプの値とのみ比較が可能です。

例えば、特殊タイプ YOUTH および表 CAMP_DB2_ROSTER が、次の SQL ステートメントを使用して作成されると仮定します。

```
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、AGE および HIGH_SCHOOL_LEVEL が同じ特殊タイプを持っているので有効です。

```
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
```

しかし、AGE と ATTENDEE_NUMBER の比較は、キャスト関数または CAST 指定を使用して、特殊タイプとソース・タイプの間でキャストすることによって、可能になります。以下の比較はすべて有効です。

```
SELECT * FROM CAMP_DB2_ROSTER
  WHERE AGE > YOUTH(ATTENDEE_NUMBER)

SELECT * FROM CAMP_DB2_ROSTER
  WHERE AGE > CAST( ATTENDEE_NUMBER AS YOUTH)

SELECT * FROM CAMP_DB2_ROSTER
  WHERE INTEGER(AGE) > ATTENDEE_NUMBER

SELECT * FROM CAMP_DB2_ROSTER
  WHERE CAST(AGE AS INTEGER) > ATTENDEE_NUMBER
```

結果のデータ・タイプに関する規則

結果のデータ・タイプは、演算のオペランドに適用される規則によって決まります。このセクションでは、この規則について説明します。

この規則は、以下のものに適用されます。

- UNION または UNION ALL 演算の対応する列
- CASE 式の結果の式
- スカラー関数 COALESCE、IFNULL、MAX、MIN、および VALUE の引き数
- IN 述部の IN リストの式の値

結果のデータ・タイプは、オペランドのデータ・タイプによって決まります。最初の 2 つのオペランドのデータ・タイプによって、中間結果のデータ・タイプが決まり、こうして決まったデータ・タイプと次のオペランドのデータ・タイプによって、新しい中間結果のデータ・タイプが決まり、さらに以下同様に続きます。こうして、最後の中間結果のデータ・タイプと最後のオペランドのデータ・タイプによって、結果のデータ・タイプが決まります。データ・タイプの各対ごとに、次の表に要約されている規則を順次適用することによって、結果のデータ・タイプが決まります。

どちらのオペランド列にもヌルが許されない場合は、結果列にもヌルは許されません。それ以外の場合は、結果列にヌルが許されます。オペランド列の記述が結果列の記述と異なる場合は、オペランド列の値が結果列の記述に適合するように変換されます。

この変換操作は、値を結果列に割り当てる場合とまったく同じです。例えば、

- 一方のオペランド列が CHAR(10) で、もう一方のオペランド列が CHAR(5) の場合は、結果列は CHAR(10) になり、CHAR(5) の列から得られた値の右側に 5 個のブランクが埋め込まれます。
- 数値の整数部分を保持できない場合はエラーが生じます。

2 進ストリング・オペランド

2 進ストリング (BLOB) は、他の 2 進ストリング (BLOB) とのみ互換性があります。結果のデータ・タイプは、BLOB です。その他のデータ・タイプは、BLOB スカラー関数を使用してデータ・タイプを BLOB にキャストすることによって、BLOB データ・タイプとして扱うことができます。結果の BLOB の長さは、すべてのデータ・タイプの中で、最も長いものです。

一方のオペランド列	他方のオペランド	結果列のデータ・タイプ
BLOB(x)	BLOB(y)	BLOB(z) (ただし、z = max(x,y))

文字およびグラフィック・ストリングのオペランド

文字およびグラフィック・ストリングは、対応する CCSID 間で変換が決められている場合、他の文字およびグラフィック・ストリングと互換性があります。

結果のデータ・タイプに関する規則

一方のオペランド列	他方のオペランド	結果列のデータ・タイプ
DBCLOB(x)	CHAR(y) または VARCHAR(y) または CLOB(y) または GRAPHIC(y) または VARGRAPHIC(y) または DBCLOB(y)	DBCLOB(z) (ただし、z = max(x,y))
CLOB(x)	GRAPHIC(y) または VARGRAPHIC(y)	DBCLOB(z) (ただし、z = max(x,y))
VARGRAPHIC(x)	VARGRAPHIC(y) または GRAPHIC(y) または VARCHAR(y) または CHAR(y)	VARGRAPHIC(z) (ただし、z = max(x,y))
VARCHAR(x)	GRAPHIC(y)	VARGRAPHIC(z) (ただし、z = max(x,y))
GRAPHIC(x)	GRAPHIC(y) または CHAR(y)	GRAPHIC(z) (ただし、z = max(x,y))
CLOB(x)	CLOB(y) または VARCHAR(y) または CHAR(y)	CLOB(z) (ただし、z = max(x,y))
VARCHAR(x)	VARCHAR(y) または CHAR(y)	VARCHAR(z) (ただし、z = max(x,y))
CHAR(x)	CHAR(y)	CHAR(z) (ただし、z = max(x,y))

結果の CCSID は、次の表に基づいて、結果のサブタイプも決めます。

一方のオペランド列	他方のオペランド	結果列のサブタイプ
UCS-2 データ	DBCS または混合または SBCS データ	UCS-2 データ
DBCS データ	DBCS または混合または SBCS データ	DBCS データ
ビット・データ	混合、SBCS、またはビット・データ	ビット・データ
混合データ	混合または SBCS データ	混合データ
SBCS データ	SBCS データ	SBCS データ

数値オペランド

数値タイプは、他の数値タイプとのみ、互換性があります。

一方のオペランド列	他方のオペランド	結果列のデータ・タイプ
FLOAT (倍精度)	任意の数値タイプ	FLOAT (倍精度)
FLOAT (単精度)	FLOAT (単精度)	FLOAT (単精度)
FLOAT (単精度)	DECIMAL、NUMERIC、BIGINT、INTEGER、または SMALLINT	FLOAT (倍精度)

一方のオペランド列	他方のオペランド	結果列のデータ・タイプ
DECIMAL(w,x)	DECIMAL(y,z) または NUMERIC(y,z,)	DECIMAL(p,s) (ただし、p = min(31, max(x,z)+max(w-x,y-z)) s = max(x,z))
DECIMAL(w,x)	BIGINT	DECIMAL(p,x) (ただし、p = min(31, x+max(w-x,19)))
DECIMAL(w,x)	INTEGER	DECIMAL(p,x) (ただし、p = min(31, x+max(w-x,11)))
DECIMAL(w,x)	SMALLINT	DECIMAL(p,x) (ただし、p = min(31, x+max(w-x,5)))
NUMERIC(w,x)	NUMERIC(y,z)	NUMERIC(p,s) (ただし、p = min(31, max(x,z) + max(w-x, y-z)) s = max(x,z))
NUMERIC(w,x)	BIGINT	NUMERIC(p,x) (ただし、p = min(31, x+max(w-x,19)))
NUMERIC(w,x)	INTEGER	NUMERIC(p,x) (ただし、p = min(31, x + max(w-x,11)))
NUMERIC(w,x)	SMALLINT	NUMERIC(p,x) (ただし、p = min(31, x + max(w-x,5)))
BIGINT	BIGINT	BIGINT
BIGINT	INTEGER	BIGINT
BIGINT	SMALLINT	BIGINT
INTEGER	INTEGER	INTEGER
INTEGER	SMALLINT	INTEGER
SMALLINT	SMALLINT	SMALLINT
NONZERO SCALE BINARY	NONZERO SCALE BINARY	NONZERO SCALE BINARY (どちらかのオペランドが非ゼロの位取りの 2 進数である場合、両方のオペランドとも同じ位取りの 2 進数でなければならない。)

日付/時刻のオペランド

DATE タイプは、別の DATE タイプ、あるいは有効な日付のストリング表現を含む CHAR または VARCHAR 式と互換性があります。結果のデータ・タイプは、DATE です。

TIME タイプは、別の TIME タイプ、あるいは有効な時刻のストリング表現を含む CHAR または VARCHAR 式と互換性があります。結果のデータ・タイプは、TIME です。

TIMESTAMP タイプは、別の TIMESTAMP タイプ、あるいは有効なタイム・スタンプのストリング表現を含む CHAR または VARCHAR 式と互換性があります。結果のデータ・タイプは、TIMESTAMP です。

一方のオペランド列	他方のオペランド	結果列のデータ・タイプ
DATE	DATE	DATE
TIME	TIME	TIME
TIMESTAMP	TIMESTAMP	TIMESTAMP

DATALINK オペランド

データ・リンクは、別のデータ・リンクと互換性があります。ただし、NO LINK CONTROL を伴うデータ・リンクは、他の NO LINK CONTROL を伴うデータ・リンクとのみ互換性があります。FILE LINK CONTROL READ PERMISSION FS を伴うデータ・リンクは、他の FILE LINK CONTROL READ PERMISSION FS を伴うデータ・リンクとのみ互換性があります。FILE LINK CONTROL READ PERMISSION DB を伴うデータ・リンクは、他の FILE LINK CONTROL READ PERMISSION DB を伴うデータ・リンクとのみ互換性があります。結果のデータ・タイプは、DATALINK です。結果の DATALINK の長さは、すべてのデータ・タイプの中で、最も長いものです。

一方のオペランド列	他方のオペランド	結果列のデータ・タイプ
DATALINK(x)	DATALINK(y)	DATALINK(z) (ただし、z = max(x,y))

DISTINCT タイプのオペランド

特殊タイプはそれ自体とのみ互換性があります。結果のデータ・タイプは、特殊タイプです。

一方のオペランド列	他方のオペランド	結果列のデータ・タイプ
特殊タイプ	特殊タイプ	特殊タイプ

ストリングを結合する演算に適用される変換規則

ストリングを結合する演算には、連結、UNION、および UNION ALL があります。(これらの規則は、MAX、MIN、VALUE、COALESCE、IFNULL、および CONCAT スカラー関数と CASE 式にも適用されます。) いずれの場合も、結果の CCSID はバインド時に決まり、演算を実行する際には、その CCSID によって識別されるコード化文字セットにストリングを変換する処理を伴うことがあります。

結果の CCSID は、オペランドの CCSID によって決まります。最初の 2 つのオペランドの CCSID によって中間結果の CCSID が決まり、その中間結果の CCSID と次のオペランドの CCSID によって新しい中間結果の CCSID が決まるというように、常にオペランドの CCSID の組み合わせによって結果の CCSID が決まります。結果のストリングまたは列の CCSID は、その 1 つ前の中間結果の CCSID と最後のオペランドの CCSID によって決まります。以下の規則を順番に適用することによって、CCSID の組み合わせごとに結果の CCSID を判別することができます。

- 両者の CCSID が同じ場合、結果の CCSID もそれと同じになります。
- どちらか一方の CCSID が 65535 である場合、結果の CCSID は 65535 になります。[21]
- 一方の CCSID が、他方の CCSID とは異なるコード化体系でデータを表している場合、結果は次の表によって決まります。

表 17. 中間結果のコード化体系の選択

	第 2 オペランド			
第 1 オペランド	**SBCS データ**	**DBCS データ**	**混合データ**	**UCS-2 データ**
SBCS データ	下記参照	第 2 オペランド	第 2 オペランド	第 2 オペランド
DBCS データ	第 1 オペランド	下記参照	第 2 オペランド	第 2 オペランド
混合データ	第 1 オペランド	第 1 オペランド	下記参照	第 2 オペランド
UCS-2 データ	第 1 オペランド	第 1 オペランド	第 1 オペランド	下記参照

- それ以外の場合、結果の CCSID は次の表によって決定されます。

21. どちらかのオペランドが CLOB または DBCLOB の場合、結果の CCSID は、そのジョブのデフォルト CCSID になります。

表 18. 中間結果の CCSID の選択

	第 2 オペランド				
第 1 オペランド	**列値**	**演算による値**	**定数**	**特殊レジスター**	**ホスト変数**
列値	第 1 オペランド	第 1 オペランド	第 1 オペランド	第 1 オペランド	第 1 オペランド
演算による値	第 2 オペランド	第 1 オペランド	第 1 オペランド	第 1 オペランド	第 1 オペランド
定数	第 2 オペランド	第 2 オペランド	第 1 オペランド	第 1 オペランド	第 1 オペランド
特殊レジスター	第 2 オペランド	第 2 オペランド	第 1 オペランド	第 1 オペランド	第 1 オペランド
ホスト変数	第 2 オペランド	第 2 オペランド	第 2 オペランド	第 2 オペランド	第 1 オペランド

ただし、外部コード化体系のデータを含むホスト変数は、何らかの演算に使用するのに先立って、固有のコード化体系に実際に変換されます。上記の規則は、このような変換がすでに行われていることを前提にしています。

中間結果は、演算による値のオペランドであると見なされることに注意してください。例えば、COLA、COLB、および COLC の CCSID が、それぞれ 37、278、および 500 であるとします。 COLA CONCAT COLB CONCAT COLC の結果の CCSID は、次のように決められます。

1. 最初に COLA CONCAT COLB の結果の CCSID が 37 であると判別されます。これは、両方のオペランドが列なので、第 1 オペランドの CCSID が選択されるからです。
2. ステップ 1 と COLC の連結の結果の CCSID は、500 と判別されます。結果の CCSID の 500 は、第 2 オペランドが演算によって得られた値で、第 2 オペランドが列であるので、第 2 オペランドの CCSID が選択されるからです。

連結演算のオペランド、または MAX、MIN、VALUE、COALESCE、IFNULL、および CONCAT スカラー関数の選択された引き数は、必要ならば、結果のストリングのコード化文字セットに変換されます。UNION または UNION ALL のオペランドの各ストリングは、必要に応じて、結果列のコード化文字セットに変換されます。文字変換が必要になるのは、以下の条件にすべて該当する場合だけです。

- 両者の CCSID が異なっている。
- どちらの CCSID も 65535 ではない。
- ストリングがヌルでなく、空でもない。
- CCSID 変換選択表に、変換が必要であることが示されている。

ストリングの文字が変換できない場合や、CCSID 変換選択表を使用するときに表に CCSID の組み合わせに関する情報が入っていなかった場合は、エラーが起こります。また、ストリングの文字が置換文字に変換された場合には、警告が出されます。

定数

定数 (*リテラル* と呼ばれることもあります) は、ある 1 つの値を指定します。定数は、ストリング定数と数値定数に分類されます。ストリング定数は、さらに文字ストリング定数とグラフィック・ストリング定数に類別されます。数値定数は、さらに整数、浮動小数点数、および 10 進数に類別されます。

定数は、すべて NOT NULL の属性を持ちます。値がゼロの数値定数にある負符号は、無視されます。

整定数

整定数 は、整数を小数点を含まない符号付きまたは無符号の数値 (最高 19 桁) として指定します。整定数のデータ・タイプは、その値が長整数の範囲内であれば、長整数です。整定数のデータ・タイプは、その値が長整数の範囲外であっても、64 ビット整数の範囲内であれば、64 ビット整数です。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。

構文図では、符号を付けてはならない長整数の定数を指す場合に、*整数* という用語を使用しています。

例

```
64      -15     +100        32767     720176     12345678901
```

浮動小数点定数

浮動小数点定数 は、倍精度浮動小数点数を E で区切られた 2 つの数値として指定します。最初の数値には、符号および小数点を付けることができます。2 番目の数値には、符号を付けることはできますが、小数点を付けることはできません。この定数の値は、2 番目の数値によって指定された 10 の累乗に最初の数値を掛けた積です。この値は、浮動小数点数の範囲内でなければなりません。また、この定数内の文字数は、24 文字以下でなければなりません。先行ゼロを含めずに数えて、最初の数値の桁数は 17 桁以下、2 番目の数値の桁数は 3 桁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、10 進数を符号付きまたは無符号の数値 (最高 31 桁) として指定します。この定数は、次のどちらかの条件を満たさなければなりません。

- 小数点を含む、または
- 2147483647 より大きいか、-2147483647 より小さい。

精度は、数字の全桁数 (先行ゼロおよび後書きゼロも含む) であり、小数点の右側にある桁数 (後書きゼロも含む) が、位取りです。

例

```
25.5     1000.     -15.     +37589.3333333333     12345678901
```

2 進ストリング定数

2 進ストリング定数 は、可変長の 2 進ストリングを指定します。2 進ストリングの形式は以下のとおりです。

- X の後に、始めと終わりをストリング区切り文字で囲んだ一連の文字。ストリング区切り文字で囲まれている文字は、偶数桁数の 16 進数字でなければなりません。16 進数字の桁数は、32740 以下でなければなりません。16 進数字とは、数字または A から F までの任意の文字 (大文字または小文字) です。

定数に割り当てられる CCSID は 65535 です。

2 進ストリング定数の構文は、文字定数の 2 番目の構文に等しいことに注意してください。この形式の定数が 2 進ストリング定数として処理されるのは、SET OPTION ステートメントで 2 進ストリング・オプション (SQLCURRULE = *STD) が指定された場合、または CRTSQLxxx コマンドで SQLCURRULE(*STD) パラメーターが指定された場合に限られます。

例

```
X'FFFF'
```

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します。文字ストリング定数には、次のように 2 つの形式があります。

- ストリング区切り文字で始まり、また終了する一連の文字。2 つのストリング区切り文字の間のバイト数は、32740 を超えてはなりません。2 つの連続するストリング区切り文字は、文字ストリング内で 1 つのストリング区切り文字を表す場合に使用します。ストリング内に含まれていない連続する 2 つのストリング区切り文字は、空のストリングを表します。
- X の後に、始めと終わりをストリング区切り文字で囲んだ一連の文字。ストリング区切り文字で囲まれている文字は、偶数桁数の 16 進数字でなければなりません。16 進数字の桁数は、32740 以下でなければなりません。16 進数字とは、数字または A から F までの任意の文字 (大文字または小文字) です。16 進数の表記規則により、1 対の 16 進数字がそれぞれ 1 文字を表します。この形式のストリング定数を使うと、キーボード上にはない文字を指定することができます。

文字ストリング定数として、混合データを使用することができます。ジョブの CCSID が混合データをサポートする場合は、文字ストリング定数に DBCS ストリングが含まれていれば、その文字ストリング定数は混合データとして扱われます。それ以外の場合はすべて、文字ストリング定数は SBCS データとして分類されます。

定数に割り当てられる CCSID は、そのソースが外部コード化体系 (ASCII など) でコード化されている場合を除き、その定数が入っているソースの CCSID です。ホスト変数のデータは、外部コード化体系から現行サーバーのデフォルトの CCSID に変換されます。この場合、定数に割り当てられる CCSID は、現行サーバーのデフォルトの CCSID です。

ソースの CCSID は、アプリケーション・リクエスターによって決められます。ソースの CCSID は、次のようになります。

- STRSQL の場合、アプリケーション・リクエスターのデフォルトの CCSID。
- RUNSQLSTM または STRREXPRC コマンドの場合、指定されたソース・ファイルの CCSID。
- CRTSQLxxx の場合、
  - 静的 SQL では、ソースの CCSID は、CRTSQLxxx コマンドで使用されるソース・ファイルの CCSID です。
  - 動的 SQL では、ソースの CCSID は、PREPARE ステートメントで指定されたホスト変数の CCSID、またはストリング定数が PREPARE ステートメントで指定されている場合は、現行サーバーのデフォルトの CCSID です。

例

```
'Peggy'    '14.12.1990'    '32'    'DON''T CHANGE'    ''    X'FFFF'
```

グラフィック・ストリング定数

DBCS グラフィック・ストリング定数

グラフィック・ストリング定数 は、可変長のグラフィック・ストリングです。指定するストリングの長さは、16370 を超えることはできません。DBCS グラフィック・ストリング定数には、次の 3 つの形式があります。

コンテキスト	グラフィック・ストリング定数	空ストリング	例
すべての コンテキスト	G'SO DBCS ストリング SI'	G'SO SI' G'' g'SO SI' g''	G'SO 元気 SI'
	N'SO DBCS ストリング SI'	N'SO SI' N'' n'SO SI' n''	
PL/I	SO'DBCS ストリング'GSI	SO''GSI	SO'元気'GSI

RV3F000-0

正規形式では、SQL 区切り文字と G または N は、SBCS 文字です。SBCS の ’ は、EBCDIC のアポストロフィ (X’7D’) です。

PL/I 形式では、アポストロフィと G は DBCS 文字です。ストリング内で 1 つのストリング区切り文字を表す場合は、2 つの連続した DBCS ストリング区切り文字を使用します。この PL/I 形式は、PL/I プログラムに組み込まれている静的ステートメントの場合にのみ有効であることに注意してください。

16 進の DBCS グラフィック定数もサポートされます。16 進 DBCS グラフィック定数の形式は、次のとおりです。

GX’ssss’

この定数において、**ssss** は、0 から 32766 までの 16 進数のストリングを表します。ストリング区切り文字に囲まれた文字の数は、4 の偶数倍でなければなりません。4 つの数字のグループのそれぞれが 1 つの DBCS グラフィック文字を表します。シフトインおよびシフトアウトに対応する 16 進数 ('0E'X および '0F'X) は、ストリングの中に含めません。

定数に割り当てられる CCSID は、そのソースが外部コード化体系 (ASCII など) でコード化されている場合を除き、ソースの CCSID に関連する DBCS CCSID です。この場合、定数に割り当てられる CCSID は、その定数を含む SQL ステートメントが準備される時点の現行サーバーのデフォルトの CCSID に関連する DBCS CCSID です。ソースの CCSID に関連する DBCS CCSID がない場合、CCSID は 65535 になります。

関連する DBCS CCSID に関する説明については、iSeries Information Center の グローバリゼーション DBCS CCSID トピックを参照してください。ソースの CCSID に関する説明については、「文字ストリング定数」の項を参照してください。

UCS-2 グラフィック・ストリング定数

16 進 UCS-2 グラフィック定数がサポートされます。 16 進 UCS-2 グラフィック定数の形式は、次のとおりです。

UX'ssss'

この定数において、**ssss** は、0 から 32766 までの 16 進数のストリングを表します。ストリング区切り文字に囲まれた文字の数は、4 の偶数倍でなければなりません。4 つの数字のグループのそれぞれが 1 つの UCS-2 グラフィック文字を表します。

UCS-2 定数の CCSID は 13488 です。

小数点

以下の目的でデフォルトの小数点を指定することができます。

- 数値定数を解釈するため。
- 文字ストリングを数値にキャストするときに使用する小数点文字を決定するため (例えば、DECIMAL、DOUBLE_PRECISION、FLOAT、および REAL スカラー関数および CAST 式で使用される小数点)。
- 数値をストリングにキャストするときに結果の中で使用する小数点文字を決定するため (例えば、CHAR、CLOB、および VARGRAPHIC スカラー関数および CAST 式で使用される小数点)。

デフォルトの小数点は、以下のインターフェースを使用して指定できます。

表 19. デフォルトのデータ形式インターフェース

SQL インターフェース	指定
組み込み SQL	SQL プログラム作成 (CRTSQLxxx) コマンドで、OPTION パラメーターに *JOB、 *PERIOD、*COMMA、または *SYSVAL のいずれかの値を指定します。組み込み SQL を含むプログラムのソースの中で DECMPT パラメーターを指定するには、SET OPTION ステートメントも使用できます。(CRTSQLxxx コマンドの詳細については、DB2 UDB for iSeries ホスト言語での SQL プログラミングを参照)
対話式 SQL および SQL 実行ステートメント	SQL 開始 (STRSQL) コマンドで DECPNT パラメーターを指定するか、セッション属性を変更します。あるいは、SQL 実行 (RUNSQLSTM) ステートメントで DECMPT パラメーターを使用します。(STRSQL および RUNSQLSTM コマンドについての詳細は、「SQL プログラミング 概念」を参照)。
サーバー上の呼び出しレベル・インターフェース (CLI)	SQL_ATTR_DATE_FMT および SQL_ATTR_DATE_SEP 環境変数または接続変数。(CLI の詳細については、「DB2 UDB for iSeries SQL 呼び出しレベル・インターフェース」を参照)。
Developer Kit for Javaを使用したサーバーの JDBC または SQLJ	「小数点 (Decimal Separator)」接続プロパティー。(JDBC および SQLJ の詳細については、iSeries Information Center の IBM Developer Kit for Java トピックを参照)。
iSeries Access ODBC ドライバーを使用したクライアントの ODBC	ODBC セットアップでの「アドバンスト・サーバー・オプション (Advanced Server Options)」の中の「小数点 (Decimal Separator)」。(ODBC の詳細については、iSeries Information Center の iSeries Access カテゴリーを参照)。
IBM Toolbox for Java を使用したクライアントの JDBC	JDBC セットアップの中の「形式 (Format)」。(ODBC の詳細については、iSeries Information Center の iSeries Access カテゴリーを参照)。(IBM Toolbox for Java の詳細については、iSeries Information Center の IBM Toolbox for Java トピックを参照)。

小数点がコンマの場合は、以下の規則が適用されます。

- ピリオドは、小数点としても使用できます。
- 数値定数の区切り記号として使用するコンマの後にはスペースを 1 つ付けなければなりません。
- 小数点として使用するコンマの後にスペースを付けてはなりません。

したがって、小数部を持たない 10 進定数を指定するときには、定数の最後に付くコンマの後に、ブランク以外の文字を入れておく必要があります。 このブランク以外の文字には、次のように、区切り記号のコンマを使用することができます。

```
VALUES(9999999999,, 111)
```

区切り文字

*APOST および *QUOTE は、COBOL ステートメント内でストリング区切り文字を指定する COBOL プリコンパイラー・オプションですが、両方を同時に使用することはできません。*APOST は、アポストロフィ (') をストリング区切り文字として指定し、*QUOTE は、引用符 (") を引用符として指定します。*APOSTSQL および *QUOTESQL は、COBOL プログラムに組み込まれた SQL ステートメントと同様の役割を果たすプリコンパイラー・オプションですが、両方を同時に使用することはできません。*APOSTSQL は、アポストロフィ (') を SQL ストリング区切り文字として指定します。このオプションが使用すると、引用符 (") が SQL エスケープ文字になります。*QUOTESQL 引用符を SQL ストリング区切り文字として指定します。このオプションを使用すると、アポストロフィが SQL エスケープ文字になります。*APOSTSQL および *QUOTESQL の値は、それぞれ *APOST および *QUOTE の値と同じです。

COBOL 以外のホスト言語では、ストリング区切り文字の用法が固定されています。すなわち、ホスト言語および静的 SQL ステートメントのストリング区切り文字にはアポストロフィ (') が使用され、SQL エスケープ文字には引用符 (") が使用されます。

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定義されるアプリケーション・プロセスのための記憶域であり、そこに保管される情報は、SQL ステートメントで参照することができます。特殊レジスターに対する参照は、現行サーバーによって与えられた値に対する参照となります。参照する値がストリングの場合は、その CCSID は、現行サーバーのデフォルトの CCSID となります。DB2 UDB for iSeries には、以下の特殊レジスターがあります。

CURRENT DATE または CURRENT_DATE

CURRENT DATE (現在の日付) 特殊レジスターは、現行サーバーで SQL ステートメントが実行される時点の刻時機構の読み取りに基づく日付を指定します。値はすべて、以下の状態における 1 回の刻時機構の読み取りに基づくものです。

- この特殊レジスターが、1 つの SQL ステートメント内で複数回使用される。
- この特殊レジスターが、1 つのステートメント内で CURRENT TIME または CURRENT TIMESTAMP 特殊レジスターとともに、あるいは CURDATE、CURTIME、または NOW スカラー関数とともに使用される。

例

以下のステートメントは、表 PROJECT 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 をセットしています。

```
UPDATE PROJECT
  SET PRENDATE = CURRENT DATE
  WHERE PROJNO = 'MA2111'
```

CURRENT PATH、CURRENT_PATH、または CURRENT FUNCTION PATH

CURRENT PATH 特殊レジスターは、動的に準備された SQL ステートメントにおける非修飾の特殊タイプ名 (組み込みタイプと特殊タイプの両方)、プロシージャー名、および関数名を解決するために使用する SQL パスを指定します。また、これは、SQL CALL ステートメントのホスト変数 (CALL ホスト変数) として指定された非修飾のプロシージャー名を解決するためにも使用されます。データ・タイプは VARCHAR(3483) です。

CURRENT PATH 特殊レジスターには、1 つまたは複数のスキーマ名のリストが含まれており、そこでは、それぞれのスキーマ名が区切り文字で囲まれ、次のスキーマとはコンマによって区切られています。区切り文字とコンマは、3483 の文字長に含まれています。パス内のスキーマ名の最大数は 268 です。

動的および静的の両方の SQL ステートメントでの非修飾名を解決するために SQL パスを使用する時点、およびその値の効果についての説明は、57 ページの『スキーマと SQL パス』 を参照してください。

活動化グループにおける CURRENT PATH 特殊レジスターの初期値は、実行される最初の SQL ステートメントによって設定されます。

- 活動化グループにおける最初の SQL ステートメントが、SQL プログラムまたは SQL パッケージから実行され、SQLPATH パラメーターが CRTSQLxxx コマン

ドで指定されている場合には、そのパスは SQLPATH パラメーターで指定された値になります。また、SQLPATH 値は、SET OPTION ステートメントを使用しても指定することができます。

- その他の場合は、
  - SQL 名の場合、″QSYS″、″QSYS2″、″ステートメントの*権限 ID の値*″。
  - システム名の場合、″*LIBL″。

SET PATH ステートメントを実行すれば、レジスターの値を変更することができます。このステートメントの詳細については、786 ページの『SET PATH』を参照してください。

例

スキーマ QSYS および QSYS2 (SYSTEM PATH) の前に、スキーマ SMITH を検索するように特殊レジスターを設定します。

```
SET CURRENT PATH SMITH, SYSTEM PATH
```

CURRENT SCHEMA

CURRENT SCHEMA (現行スキーマ) 特殊レジスターは、VARCHAR(128) の値を指定します。この値は、動的に準備された SQL ステートメントの中で、該当する無修飾のデータベース・オブジェクト参照を修飾するために使用するスキーマ名を識別します。[22] CURRENT SCHEMA は、DYNDFTCOL が指定されているプログラムの中の名前を修飾するためには使用されません。プログラム内で DYNDFTCOL が指定されている場合は、そのスキーマ名が CURRENT SCHEMA のスキーマ名の代わりに使用されます。

CURRENT SCHEMA の初期値は、現行セッション・ユーザーの権限 ID です。

静的 SQL ステートメントの場合は、無修飾のデータベース・オブジェクト参照を修飾するために使用するスキーマ名は、DFTRDBCOL キーワードにより制御されます。

例

オブジェクト修飾用のスキーマを 'D123' に設定します。

```
SET CURRENT SCHEMA = 'D123'
```

CURRENT SERVER または CURRENT_SERVER

特殊レジスター CURRENT SERVER (現行サーバー) は、現行のサーバーを識別する VARCHAR(18) (長さ 18 の可変長文字) の値を示します。

CURRENT SERVER は、CONNECT (タイプ 1)、CONNECT (タイプ 2)、または SET CONNECTION ステートメントによって変更できますが、それができるのは特定の条件のもとだけに限られます。425 ページの 『CONNECT (タイプ 1)』、431 ページの 『CONNECT (タイプ 2)』、および 767 ページの 『SET CONNECTION』の説明を参照してください。

22. DB2 UDB (OS/390 および z/OS 版) との互換性を維持するために、特殊レジスター CURRENT SQLID は CURRENT SCHEMA の同義語として扱われます。

CURRENT SERVER を指定できるようにするには、ADDRDBDIRE コマンドまたは WRKRDBDIRE コマンドを使用してリレーショナル・データベース・ディレクトリーに項目を追加することによって、ローカル・リレーショナル・データベースに名前を付けなければなりません。

例

ホスト変数の APPL_SERVE (VARCHAR(18)) を現行サーバーの名前にセットします。

```
SELECT CURRENT SERVER
  INTO :APPL_SERVE
  FROM ROW1_TABLE
```

CURRENT TIME または CURRENT_TIME

CURRENT TIME (現在の時刻) 特殊レジスターは、SQL ステートメントが現行サーバーで実行される時点での刻時機構の読み取りに基づく時刻を指定します。値はすべて、以下の状態における 1 回の刻時機構の読み取りに基づくものです。

- この特殊レジスターが、1 つの SQL ステートメント内で複数回使用される
- この特殊レジスターが、1 つのステートメント内で CURRENT DATE または CURRENT TIMESTAMP 特殊レジスターとともに、あるいは CURDATE、CURTIME、または NOW スカラー関数とともに使用される。

例

表 CL_SCHED を使用して、当日後刻に開始される (STARTING) すべてのクラス (CLASS_CODE) を選択します。当日のクラスは、DAY 列に 3 の値を持っています。

```
SELECT CLASS_CODE FROM CL_SCHED
  WHERE STARTING > CURRENT TIME  AND DAY = 3
```

CURRENT TIMESTAMP または CURRENT_TIMESTAMP

CURRENT TIMESTAMP (現在のタイム・スタンプ) 特殊レジスターは、SQL ステートメントが現行サーバーで実行される刻時機構の読み取りに基づくタイム・スタンプを指定します。値はすべて、以下の状態における 1 回の刻時機構の読み取りに基づくものです。

- この特殊レジスターが、1 つの SQL ステートメント内で複数回使用される
- この特殊レジスターが、1 つのステートメント内で CURRENT DATE または CURRENT TIME 特殊レジスターとともに、あるいは CURDATE、CURTIME、または NOW スカラー関数とともに使用される。

例

以下のステートメントでは、表 IN_TRAY に行を挿入しています。列 RECEIVED には、行が挿入された日時を示すタイム・スタンプの値が入ります。その他の 3 つの列には、ホスト変数 SRC(CHAR(8))、SUB(CHAR(64))、および TXT (VARCHAR(200)) の値が入ります。

```
INSERT INTO IN_TRAY
  VALUES (CURRENT TIMESTAMP, :SRC, :SUB, :TXT)
```

CURRENT TIMEZONE または CURRENT_TIMEZONE

CURRENT TIMEZONE 特殊レジスターは、世界標準時 (UTC)[23] と現行サーバーでの地方時との差を指定します。この時差は時刻期間 (最初の 2 桁が時、次の 2 桁が分、最後の 2 桁が秒を示す 10 進数) で表されます。時を示す数値は、-23 から 23 までです。地方時から CURRENT TIMEZONE を引くと、その地方時が UTC に変換されます。

例

以下のステートメントでは、表の IN_TRAY からすべての行を選択して、その値を UTC に合わせます。

```
SELECT RECEIVED - CURRENT TIMEZONE, SOURCE,
       SUBJECT, NOTE_TEXT FROM IN_TRAY
```

USER

特殊レジスター USER は、現行サーバー側の実行時権限 ID を指定します。この特殊レジスターのデータ・タイプは VARCHAR(18) です。

例

このステートメントは、ユーザーが自分でそこに置いた表 IN_TRAY から、すべての行を選択しています。

```
SELECT * FROM IN_TRAY
  WHERE SOURCE = USER
```

23. 以前はグリニッジ標準時 (GMT) と呼ばれていました。

列名

列名の持つ意味は、その列名が使用されている文脈によって異なります。列名は、次のように使用されることがあります。

- CREATE TABLE ステートメントの中などで列の名前を宣言する。
- CREATE INDEX ステートメントの中などで列を識別する。
- 以下に示すような文脈で列の値を指定する。
  - *列関数* では、その関数を適用するグループまたは中間結果表の 1 つの列のすべての値を、列名によって指定します。グループと中間結果表については、764 ページの『SELECT INTO』の項で説明しています。例えば、MAX(SALARY) は、グループ内の列 SALARY のすべての値に関数 MAX を適用します。
  - *GROUP BY または ORDER BY 文節* では、その文節が適用される中間結果表内のすべての値を、列名によって指定します。例えば、ORDER BY DEPT を指定すると、DEPT という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* では、列名によって、その構造を適用する各行またはグループに対して値を指定します。例えば、検索条件 CODE = 20 がある行に適用される場合、列名 CODE によって指定される値は、それらの行にある列 CODE の値を指定しています。

修飾付き列名

列名の修飾子には、表名、ビュー名、別名、または相関名が可能です。

列名を修飾できるかどうかは、その列名が使用されている文脈によって決まります。

- COMMENT および LABEL ステートメントでは、列名を必ず修飾しなければなりません。
- 列名によって列の値を指定しているところでは、ユーザーのオプションで列名を修飾することができます。
- 上記以外の文脈では、列名を修飾してはなりません。

修飾名が任意指定の個所では、修飾名は 2 つの役目を果たします。詳細は、117 ページの『あいまいさを避けるための列名修飾子』、および 119 ページの『相関参照における列名修飾子』の項を参照してください。

相関名

相関名 は、UPDATE または DELETE ステートメントの最初の文節および照会の FROM 文節で定義することができます。例えば、以下の文節では、X.MYTABLE の相関名として Z を設定しています。

```
FROM X.MYTABLE Z
```

相関名が、表、ビュー、または別名に関連付けられるのは、その相関名が定義されている文脈の中だけです。したがって、同一の相関名を、別のステートメント内で別の目的のために定義したり、同一のステートメント内の別の文節で定義したりすることができます。

相関名を修飾名として使用することによって、あいまいさを避けたり、相関参照を設定したりすることができます。また、相関名を表、ビュー、または別名の短縮名としても使用することができます。上記の例では、単に X.MYTABLE を何度も入力するのを避けるために、相関名 Z を使用しても構いません。

表名、ビュー名、または別名に対して相関名が指定されている場合、その表、ビュー、または別名にある列に対する修飾付き参照では、表名、ビュー名、または別名ではなく、必ず相関名を使用しなければなりません。例えば、以下の例では、EMPLOYEE に対する相関名が指定されているので、EMPLOYEE.PROJECT への参照は誤りとなります。

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

PROJECT に対する修飾付き参照では、以下のように、代わりに相関名 “E” を使用しなければなりません。

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

FROM 文節に指定される名前は、*直接的な* 名前、または*間接的な* 名前のいずれかです。相関名は、常に直接的な名前です。相関名が指定されていない場合は、表名、ビュー名、または別名は、その FROM 文節の中で*直接的* であると言われます。例えば、以下の FROM 文節では、EMPLOYEE には相関名が指定され、DEPARTMENT には相関名が指定されていません。したがって、DEPARTMENT は直接的な名前であり、EMPLOYEE は間接的な名前となります。

```
FROM EMPLOYEE E, DEPARTMENT
```

FROM 文節で直接的な表名、ビュー名、または別名は、その FROM 文節で直接的な他のいかなる表名またはビュー名とも異なっていなければなりません。また、その FROM 文節のいかなる相関名とも異なっていなければなりません。これらの名前は、修飾のない表名またはビュー名を修飾した後で比較されます。

以下に示す最初の 2 つの FROM 文節では、直接的な名前である EMPLOYEE への参照がそれぞれに 1 つしか入っていないので、正しい FROM 文節となります。

1. 次の FROM 文節では、

   ```
   FROM EMPLOYEE E1, EMPLOYEE
   ```

   FROM 文節の 2 番目の EMPLOYEE にある列は、EMPLOYEE.PROJECT などの修飾付き参照によって指示されます。最初の EMPLOYEE に対する修飾付き参照では、相関名 “E1” (E1.PROJECT) を使用しなければなりません。
2. 次の FROM 文節では、

   ```
   FROM EMPLOYEE, EMPLOYEE E2
   ```

   FROM 文節の最初の EMPLOYEE にある列は、EMPLOYEE.PROJECT などの修飾付き参照によって指示されます。2 番目の EMPLOYEE に対する修飾付き参照では、相関名 “E2” (E2.PROJECT) を使用しなければなりません。
3. 次の FROM 文節では、

   ```
   FROM EMPLOYEE, EMPLOYEE                  ***INCORRECT***
   ```

   この文節に含まれている 2 つの表名 (EMPLOYEE と EMPLOYEE) は同じなので、これは許されません。

4. 次のステートメントでは、

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

   FROM 文節内にある 2 つの EMPLOYEE が、両方とも相関名を持っているので、EMPLOYEE.PROJECT という修飾付き参照は誤りです。その代わりに、PROJECT に対する参照は、どちらかの相関名を使って、E1.PROJECT または E2.PROJECT と修飾しなければなりません。

5. 次の FROM 文節では、

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

   2 番目の EMPLOYEE にある列に対する参照では、X.EMPLOYEE(X.EMPLOYEE.PROJECT) を使用しなければなりません。この FROM 文節は、ステートメントの権限 ID が X ではない場合に限って有効です。

FROM 文節で指定する相関名は、以下の名前とは異なっていなければなりません。

- 同じ FROM 文節の他の相関名
- 同じ FROM 文節で直接的な修飾のない表名またはビュー名
- 同じ FROM 文節にある修飾付き表名またはビュー名の 2 番目の SQL ID

例えば、以下のような FROM 文節は誤りです。

```
FROM EMPLOYEE E, EMPLOYEE E
FROM EMPLOYEE DEPARTMENT, DEPARTMENT          ***INCORRECT***
FROM X.T1, EMPLOYEE T1
```

以下のような FROM 文節は、参照が混同される恐れがありますが、構文上は正しい FROM 文節です。

```
FROM EMPLOYEE DEPARTMENT, DEPARTMENT EMPLOYEE
```

また、FROM 文節で相関名を使用すると、結果表の列に関連付ける列名リストを指定する選択も可能になります。相関名と同様に、これらのリストされた列名は、照会の全体にわたって列の参照で使う必要がある直接的な名前になります。列名リストが指定されている場合は、基本表の列名は間接的なものになります。

次の FROM 文節では、

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

D.NUM などの修飾付きの参照は、DEPTNO として表で定義されている DEPARTMENT 表の最初の列を表します。この FROM 文節を用いた D.DEPTNO への参照は、列名 DEPTNO が間接的な列名であるため、誤りです。

列のリストを指定する場合は、そのリストは、表参照で使われる列の数と同じだけの名前からなっている必要があります。それぞれの列名は固有であり、修飾されていない名前でなければなりません。

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、または検索条件の文脈では、いくつかの表またはビュー内の列にある値を列名によって参照します。この列が入って

いる表およびビューのことを、該当する文脈の*目的表* と呼びます。同じ名前の列が、複数の目的表に入っていることもあります。列名を修飾する理由の 1 つは、その列がどの目的表の列であるかを指示するためです。

表指定子

特定の目的表を指示する修飾子のことを*表指定子* と呼びます。目的表を識別する文節では、その目的表を指示する表指定子も設定します。例えば、SELECT 文節の式で使用する目的表は、次のように SELECT 文節の後の FROM 文節で指定します。

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

この例は、FROM 文節で表指定子を設定する方法を示しています。つまり、

- 表名またはビュー名の後に付く名前は、相関名であると同時に表指定子でもあります。したがって、CORZ は表指定子です。CORZ は、選択リスト内の最初の列名を修飾するために使用されています。
- SQL 命名規則では、直接表名やビュー名は表指定子です。したがって、OWNY.MYTABLE は表指定子です。OWNY.MYTABLE は、選択リスト内の 2 番目の列名を修飾するために使用されています。
- システム命名規則では、直接表名や直接ビュー名の表指定子は、修飾のない表名またはビュー名です。次の例で、MYTABLE は、OWNY/MYTABLE の表指定子です。

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

未定義の参照またはあいまいな参照の回避

列名によって列の値を参照する場合は、その名前を持つ列が、必ず 1 つの目的表に入っていなければなりません。次のような場合は、エラーと見なされます。

- 指定した名前を持つ列が入っている目的表が存在しない。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ときに、指定した名前を持つ列が、指定した表に存在しない。この参照も未定義になります。
- 列名が修飾されていないときに、その名前を持つ列が、複数の目的表に存在する。この参照はあいまいになります。

あいまいな参照を避けるには、固有のものとして定義されている表指定子によって列名を修飾します。同じ名前の列が、異なる名前を持ついくつかの目的表に入っている場合は、目的表の名前を表指定子として使うことができます。

文脈内の目的表として、同一の表が何度も指定されることがあります。この場合は、文脈内に現れる個々の表を明確に指示するために、目的表ごとに別々の相関名を使用しなければなりません。例えば、以下の FROM 文節では、最初の表 CORPDATA.EMPLOYEE を参照するために X を定義し、2 番目の表 CORPDATA.EMPLOYEE を参照するために Y を定義しています。

```
FROM CORPDATA.EMPLOYEE X, CORPDATA.EMPLOYEE Y
```

直接的な表名による表指定子で列を修飾する場合は、直接的な表名の修飾形式と非修飾形式のいずれかを使用します。ただし、使用する修飾名および表は、表名またはビュー名と表指定子を完全に修飾したものと同じでなければなりません。

1. ステートメントの権限 ID が CORPDATA である場合は、

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

   このステートメントは、有効なステートメントです。

2. ステートメントの権限 ID が REGION である場合は、

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

   このステートメントは無効です。EMPLOYEE は REGION.EMPLOYEE という表を表していますが、WORKDEPT の修飾子は、CORPDATA.EMPLOYEE という異なる表を表しているからです。

相関参照における列名修飾子

副選択 は、照会の一形式であり、各種の SQL ステートメントのコンポーネントとして使用できます。副照会の詳細については、339 ページの『第 4 章 照会』を参照してください。*副照会* は全選択 1 つで、括弧で囲んだ形式をとります。例えば、*副照会* は検索条件の中で使用することができます。

副照会は、独自の検索条件を含むことができ、これらの検索条件は逆に副照会を含むことができます。したがって、SQL ステートメントは副照会の階層を含むことができます。副照会を含んでいる階層のエレメントは、その階層に含まれている副照会より上位レベルにあるといわれます。

階層の各要素は、1 つの文節を持ち、その文節によって 1 つまたは複数の表指定子を設定します。階層の最上位レベルにある UPDATE または DELETE ステートメントを除いて、この文節は FROM 文節になります。副照会に含まれる検索条件、選択リスト、JOIN 文節、または表関数の引き数では、その階層自身の要素である FROM 文節によって識別されている表の列を参照できるだけでなく、その階層自身の要素から階層の最上位レベルまでのパスに沿って、どのレベルで識別されている表の列も参照することができます。その階層自身より上位のレベルで識別されている表の列への参照を、*相関参照* と呼びます。

Q が表 T に対して定義されている相関名である場合、表 T の列 C に対する相関参照は、C、T.C、または Q.C という形式をとります。以下の説明は、相関参照が常に修飾付きの列名の形式をとり、その修飾名が相関名であることを前提にしています。

Q.C が相関参照となるのは、次の 3 つの条件が満たされている場合だけです。

- 副照会に含まれる検索条件、選択リスト、JOIN 文節、または表関数の引き数で Q.C が使用されている。
- Q が、その副照会の FROM 文節、選択リスト、JOIN 文節、または表関数の引き数で使用されている表を指示していない。
- Q が、上位レベルで使用されている表を指示している。

Q.C によって参照される列 C は、Q を表またはビューの表指定子として使用しているレベルの表またはビューにある列です。同一の表またはビューを複数のレベルから識別する可能性があるので、表指定子には、固有の相関名を使用するようにし

てください。Q を使用して複数のレベルから表を指示した場合、Q.C は、Q.C を使用している副照会の上位にある階層のうち、最下位レベルの階層を参照します。

以下のステートメントでは、Q を T1 および T2 に対する相関名として使用していますが、Q.C は T2 に関連付けられている相関名を参照します。これは、Q.C を使用している副照会の上位にある階層の最下位レベルで、Q が T2 に関連付けられているためです。

```
SELECT *
  FROM T1 Q
  WHERE A < ALL (SELECT B
                   FROM T2 Q
                   WHERE B < ANY (SELECT D
                                    FROM T3
                                    WHERE D = Q.C))
```

修飾されていない列名

次のような場合には、修飾されていない列名も相関参照となります。

- その列が、副照会の検索条件で使用されている。
- その列が、副照会の FROM 文節で使用されている表に含まれていない。
- その列が、上位レベルで使用されている表に含まれている。

修飾されていない相関参照は、SQL ステートメントが分かりにくくなるので、なるべく使用しないでください。列は、ステートメントが準備される時点で、その列が見つかった表によって暗黙的に修飾されます。この暗黙の修飾は、いったん行われると、ステートメントが準備し直されるまで変更されることはありません。修飾されていない相関参照を含む SQL ステートメントが準備、または実行されると、SQL プリコンパイラーはプリコンパイル・リストに警告メッセージを出し、データベース・マネージャーは正の SQLCODE (+12) と SQLSTATE (01545) を出します。

変数に対する参照

SQL ステートメントの中の変数 は、SQL ステートメントの実行時に変化する可能性がある値を指定します。 SQL ステートメントで使用される変数 には幾つかのタイプがあります。

ホスト変数
ホスト変数は、ホスト言語のステートメントによって定義されます。ホスト変数を参照する方法については、121 ページの 『ホスト変数に対する参照』を参照してください。

遷移変数
遷移変数はトリガーの中で定義されるもので、列の古い値または新しい値を参照します。遷移変数を参照する方法については、575 ページの 『CREATE TRIGGER』を参照してください。

SQL 変数
SQL 変数は、SQL 関数、SQL プロシージャー、またはトリガー内の SQL 複合ステートメントによって定義されます。 SQL 変数の詳細については、821 ページの 『SQL パラメーターおよび変数の参照』を参照してください。

SQL パラメーター
SQL パラメーターは、CREATE FUNCTION (SQL スカラー)、CREATE FUNCTION (SQL 表)、または CREATE PROCEDURE (SQL) ステートメントで定義されます。 SQL 変数の詳細については、821 ページの 『SQL パラメーターおよび変数の参照』を参照してください。

パラメーター・マーカー
動的 SQL ステートメントでは、変数を参照することはできません。代わりに、SQLDA の中でパラメーター・マーカーを定義して、使用します。パラメーター・マーカーの詳細については、728 ページの 『パラメーター・マーカー』を参照してください。

本書では、特別に指示がない限り、構文図の中でホスト変数 という用語が使用されている場所では、ホスト変数、遷移変数、SQL 変数、SQL パラメーター、またはパラメーター・マーカーを使用できることを示しています。

ホスト変数に対する参照

ホスト変数 とは、COBOL データ項目、RPG フィールド、または SQL ステートメントで参照される PLI、REXX、C++、あるいは C の変数を指します。ホスト変数は、ホスト言語のステートメントによって定義されます。C、C++、COBOL、PL/I、および RPG のホスト構造の参照方法の詳細については、127 ページの 『C、C++、COBOL、PL/I、および RPG におけるホスト構造』を参照してください。
REXX でのホスト変数の詳細については、「DB2 UDB サーバー (AS/400 版) ホスト言語での SQL プログラミング」を参照してください。

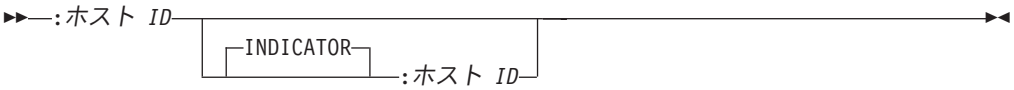
SQL ステートメント中のホスト変数 は、ホスト変数の宣言の規則に従ってプログラム内で記述されたホスト変数でなければなりません。REXX および RPG 以外のすべてのホスト言語では、SQL ステートメントで使用されるホスト変数はすべて、SQL 宣言セクションで宣言されていなければなりません。(REXX では、変数は宣言されている必要はありません。RPG には、宣言セクションはありませんが、ホス

ト変数は、そのプログラム全体にわたって宣言されます。) SQL 宣言セクションで宣言されている変数と同じ名前を持つ変数を、SQL 宣言セクションの外側で宣言してはなりません。 SQL 宣言セクションは、BEGIN DECLARE SECTION で開始し、END DECLARE SECTION で終了します。

ホスト変数の使い方についての詳細は、「SQL プログラミング 概念」を参照してください。

構文図の中で使用されているホスト変数 という用語は、ホスト変数に対する参照を示します。 FETCH、SELECT INTO、SET 変数の INTO 文節または VALUE INTO ステートメントにおけるホスト変数 は、ある行の列の値を割り当てるホスト変数を識別します。 CALL ステートメントまたは EXECUTE ステートメントにおけるホスト変数は、出力パラメーターが割り当てられるホスト変数、また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入力引き数値を指定するホスト変数、あるいはその両方を識別します。それ以外のすべての文脈では、ホスト変数 は、アプリケーション・プログラムから DB2 UDB for iSeries に渡される値を示します。

ホスト変数 参照の一般的な形式は、次のとおりです。



それぞれのホスト *ID* は、ソース・プログラム内で宣言しておく必要があります。2 番目のホスト *ID* によって指定される変数のデータ・タイプは、位取りがゼロの短整数 でなければなりません。

最初のホスト *ID* は主変数 を示し、2 番目のホスト *ID* は主変数の*標識変数* を示します。標識変数は、次のような用途に使用します。

- ヌル値を指示します。標識変数の負の値は、ヌル値を示します。
- 以下のデータ・マッピング・エラーのいずれかを示します。
  - 文字を変換できなかった。
  - 数値変換エラー (アンダーフローまたはオーバーフロー)。
  - 算術式エラー (0 による除算)。
  - 日付またはタイム・スタンプの変換エラー (指定されている日付形式の有効な範囲内にない日付またはタイム・スタンプ)。
  - 日付/時刻の値のストリング表現が正しくない。
  - 混合 (MIXED) データが正しく形成されていない。
  - 数値が無効。
  - スカラー関数 SUBSTR の引き数が範囲外。
- 切り捨てられたストリングの元の長さを記録します。
- ホスト変数に割り当てた時刻が切り捨てられた場合に、その時刻の秒の部分を記録します。

例えば、:V1:V2 というホスト変数参照を使用して挿入値または更新値を指定した場合に、V2 の値が負ならば、指定した値がヌル値であることを示します。V2 が負でない場合、指定した値が V1 の値になります。

同様に、:V1:V2 を CALL、FETCH、または SELECT INTO ステートメントで指定した場合に、戻された値がヌルであれば、V1 は未定義であり、V2 に負の値がセットされます。セットされる負の値は、次のとおりです。

- -1、これは選択された値がヌル値であったことを示します。
- -2、これは、外側の副選択の選択リストのデータ・マッピング・エラーによりヌル値が戻されたことを示します。[24]

参照によって戻された値がヌル値でなければ、その値が V1 に割り当てられ、V2 にはゼロがセットされます (ただし、V1 への割り当ての際に切り捨てが必要だった場合は、V2 にそのストリングの元の長さがセットされます)。また、割り当ての際に、時刻の秒の部分を切り捨てる必要があった場合は、切り捨てられた秒数が V2 にセットされます。

2 番目のホスト ID が省略された場合は、そのホスト変数 は標識変数をもちません。このような場合、ホスト変数 :V1 によって指定された値は常に V1 の値になり、ヌル値をその変数に割り当てることはできません。したがって、INTO 文節では、対応する結果列にヌル値を入れることができない場合以外は、この形式を使用してはなりません。この形式を使用し、しかも列にヌル値が含まれている場合、データベース・マネージャーは、SQLCA の SQLCODE フィールドに負の値 (-407) を入れます。標識変数が用意されていない場合にデータが切り捨てられても、エラー条件とはなりません。

SQL ステートメントでホスト変数を使用する場合は、ホスト変数の前に必ずコロンを付けなければなりません。

C、C++、ILE RPG、および PL/I では、ホスト変数を参照する SQL ステートメントは、そのホスト変数の宣言の有効範囲になければなりません。カーソルに対する SELECT ステートメントでホスト変数を参照する場合、そのホスト変数の宣言の有効範囲内になければならないのは、DECLARE CURSOR ステートメントではなく、OPEN ステートメントです。

ストリング・ホスト変数の CCSID は、次のいずれかです。

- DECLARE VARIABLE ステートメントで指定された CCSID、または
- 該当のホスト変数に対して CCSID 文節を伴う DECLARE VARIABLE の指定がない場合には、そのホスト変数を含む SQL ステートメント実行される時点のアプリケーション・リクエスターのデフォルト CCSID (ただし、ASCII などの外部コード化体系に対する CCSID でない場合のみ)。外部コード体系に対する CCSID である場合には、ホスト変数は、現行サーバーのデフォルトの CCSID に変換されます。

24. 特定のスカラー関数や算術式で、データ・マッピング・エラーのためのヌル値が戻されることがありますが、算術式またはスカラー関数の引き数がヌル可能でない場合は、その結果の列はヌル可能とは見なされません。

例

PROJECT 表を使用して、プロジェクト (PROJNO) 'IF1000' について、ホスト変数 PNAME (VARCHAR(26)) をプロジェクト名 (PROJNAME) に、ホスト変数 STAFF (DECIMAL(5,2)) を平均人員レベル (PRSTAFF) に、そしてホスト変数 MAJPROJ (CHAR(6)) を主プロジェクト (MAJPROJ) に設定します。 PRSTAFF と MAJPROJ の列には空値が入っている可能性があるので、標識変数の STAFF_IND (SMALLINT) と MAJPROJ_IND (SMALLINT) を指定しています。

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

動的 SQL でのホスト変数

動的 SQL ステートメントでは、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 (?) で表し、アプリケーションが値を用意する動的 SQL ステートメントでの位置を表します。すなわち、この位置は、ステートメント・ストリングがもし静的 SQL ステートメントであったならば、ホスト変数が見つかるはずの位置です。次の例は、ホスト変数と、パラメーター・マーカーを使った動的ステートメントを使用する静的 SQL を示しています。

```
INSERT INTO DEPT VALUES( :HV_DEPTNO, :HV_DEPTNAME, :HV_MGRNO, :HV_ADMRDEPT)

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

パラメーター・マーカーの詳細については、 728 ページの『パラメーター・マーカー』を参照してください。

LOB ホスト変数の参照

通常の LOB 変数、LOB ロケーター変数 (125 ページの『LOB ロケーター変数の参照』参照)、および LOB ファイル参照変数 (125 ページの『LOB ファイル参照変数の参照』参照) は、以下のホスト言語で定義することができます。

- C
- C++
- ILE RPG
- ILE COBOL
- PL/I

LOB が許されている場合は、構文図における 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意味していることになります。これらの変数はホスト・プログラム言語での固有のデータ・タイプではないため、SQL 拡張子が使用され、プリコンパイラーはそれぞれの変数を表すために必要なホスト言語構成を生成します。

LOB 値全体を収容できるほどの大きなホスト変数を定義することが可能であり、サーバーからのデータ転送の遅れに関するパフォーマンス上の利点が必要ない場合には、LOB ロケーターは不要です。しかしながら、LOB 値全体を保管するということは、ホスト言語の制約、記憶域の制限、またはパフォーマンス要件により、受け入れられないことがしばしばあります。 LOB 値全体を一時的に保管することが受け入れられない場合、LOB 値は LOB ロケーターによって参照することが可能であ

り、LOB 値の一部を選択してホスト変数に入れたり、LOB 値の一部だけが入っているホスト変数を更新することができます。

他のすべてのホスト変数と同様に、LOB ロケーター変数または LOB ファイル参照変数は、関連した標識変数を持つことができます。 LOB ロケーター変数と LOB ファイル参照変数の標識変数は、他のデータ・タイプの標識変数と同じような働きをします。ヌル値がデータベースから戻されると、標識変数が設定されますがホスト変数は変更ありません。これは、ロケーターがヌル値を指すことはあり得ないということを意味します。

LOB ロケーター変数の参照

LOB *ロケーター変数* は、サーバー上の LOB 値を表すロケーターを含むホスト変数です。これは、以下のホスト言語で定義することができます。

- C
- C++
- ILE RPG
- ILE COBOL
- PL/I

LOB 値を扱うためのロケーターの使用法の詳細については、 68 ページの『ロケーターを用いたラージ・オブジェクト (LOB) の操作』を参照してください。

SQL ステートメントのロケーター変数は、ロケーター変数の宣言の規則に従って、プログラムで記述されている LOB ロケーター変数を識別する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り間接的に行われます。例えば、C の例は次のとおりです。

```
static volatile SQL TYPE IS CLOB_LOCATOR *loc1;
```

構文図で使用されている *ロケーター変数* という用語は、LOB ロケーター変数に対する参照を示します。*メタ変数* *ロケーター変数* を拡張して、*ホスト変数* の場合と同じように、*ホスト ID* を含めることができます。

LOB ロケーターに関連した標識変数がヌルの場合、参照された LOB はヌルです。

ロケーター変数が、現在、いかなる値も表していない場合は、ロケーター変数が参照されたときに、エラーが起こります。

トランザクション・コミットまたはトランザクション終了の場合、そのトランザクションが獲得したすべての LOB ロケーターは解放されます。

アプリケーション・プログラマーは責任を持って、LOB ロケーターが使用されるのは、最初に LOB ロケーターを生成したサーバーで実行される SQL ステートメント内のみであることを保証する必要があります。例えば、LOB ロケーターがあるサーバーから戻されて、LOB ロケーター変数に割り当てられると想定します。この LOB ロケーター変数が、その後、他のサーバーで実行される SQL ステートメントで使用されると、予期しない結果が起こる可能性があります。

LOB ファイル参照変数の参照

LOB *ファイル参照変数* は、LOB の直接ファイル入出力に使用されます。これは、以下のホスト言語で定義することができます。

- C
- C++
- ILE RPG
- ILE COBOL
- PL/I

これらは固有のデータ・タイプではないため、SQL 拡張子が使用され、プリコンパイラーはそれぞれの変数を表すために必要なホスト言語構成を生成します。

ファイル参照変数は、LOB ロケーターが LOB データを含むのではなく、表すのと同じように、ファイルを (含むのではなく) 表します。データベース照会、更新、および挿入では、ファイル参照変数を使用して、単一の列の値を保管したり、検索します。参照されるファイルはアプリケーション・リクエスター内になければなりません。

他のすべてのホスト変数と同様に、ファイル参照変数、関連した標識変数を持つことができます。

ファイル参照変数の長さ属性は、LOB の最大長であると想定されます。

ファイル参照変数は、現在、ルート (/)、QOpenSys、および UDFS ファイル・システムでサポートされています。ファイルが作成されると、ファイルに書き込み中のデータの CCSID が与えられます。現在、混合 CCSID はサポートされていません。ファイル参照変数を用いて作成されたファイルを使用するには、ファイルを 2 進モードでオープンする必要があります。

ファイル参照変数の詳細については、SQL プログラミング 概念を参照してください。

C、C++、COBOL、PL/I、および RPG におけるホスト構造

ホスト構造とは、SQL ステートメントで参照される COBOL のグループ、PL/I の構造、C または C++ の構造体、あるいは RPG のデータ構造を指します。ホスト構造は、ホスト言語のステートメントによって定義されます。これについては、DB2 UDB for iSeries ホスト言語での SQL プログラミングで説明しています。ここで使用する ″ホスト構造″ という用語には、SQLCA や SQLDA は含まれません。

ホスト構造参照の形式は、ホスト変数参照の形式と同じです。:S1:S2 の参照は、S1 がホスト構造を指定している場合は、ホスト構造参照です。S1 がホスト構造を指している場合、S2 は、短整数変数、または短整数変数の配列のいずれかでなければなりません。S1 はホスト構造で、S2 はその標識配列です。

ホスト構造は、ホスト変数のリストを参照できる文脈であれば、どのような文脈でも参照できます。ホスト構造の参照は、その構造に含まれている各ホスト変数を、ホスト言語の構造宣言で定義されている順序にしたがって参照するのと同じことです。標識配列の *n* 番目の変数は、ホスト構造の *n* 番目の変数の標識変数です。

例えば、PL/I で、V1、V2、および V3 が構造 S1 内の変数として宣言されている場合、

```
EXEC SQL FETCH CURSOR1 INTO :S1;
```

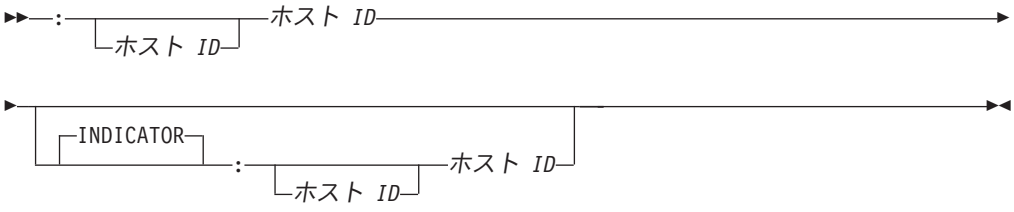
上記のステートメントは、次のステートメントと同等です。

```
EXEC SQL FETCH CURSOR1 INTO :V1, :V2, :V3;
```

ホスト構造に、標識配列より *m 個* だけ多い変数がある場合、ホスト構造の最後の *m 個* の変数は、標識変数を持ちません。ホスト構造に、標識配列より *m 個* だけ少ない変数がある場合、標識配列の最後の *m 個* の変数は無視されます。上記の規則は、ホスト構造への参照に標識変数が含まれる場合、またはホスト変数への参照に標識配列が含まれる場合にも当てはまります。標識配列または標識変数を指定しない場合、ホスト構造の変数はいずれも標識変数を持たないことになります。

構造参照に加えて、ホスト構造内の個々のホスト変数、または標識配列内の個々の標識変数は、修飾名によって参照することができます。この修飾の形式は、ホスト ID の後にピリオドと他のホスト ID を付けたものです。最初のホスト ID は、ホスト構造を指し、2 番目のホスト ID は、そのホスト構造内のホスト変数を指していなければなりません。

次の図は、ホスト変数およびホスト構造に対する参照の構文を示しています。



式の中のホスト変数は、ホスト変数の宣言に関する規則に従ってプログラムで記述されているホスト変数 (構造ではなく) を識別するものでなければなりません。

C、C++、COBOL、PL/I、および RPG におけるホスト構造

ホスト変数は、REXX ではサポートされません。

次の例は、ホスト変数およびホスト構造に対する参照を示しています。

```
:V1    :S1.V1    :S1.V1:V2    :S1.V2:S2.V4
```

C、C++、COBOL、PL/I、および RPG におけるホスト構造配列

PL/I、C++、および C では、ホスト構造配列は、次元属性をもつ構造名です。COBOL の場合は、1 次元の表です。RPG の場合は、オカレンス・データ構造です。ホスト構造配列を参照することができるのは、複数行の取り出しを使用する場合の FETCH ステートメント、またはブロック挿入を使用する場合の INSERT ステートメントだけです。ホスト構造配列は、ホスト言語のステートメントによって定義されます。これについては、DB2 UDB for iSeries ホスト言語での SQL プログラミングで説明しています。

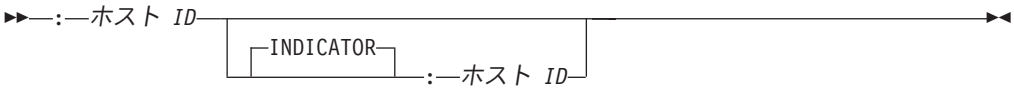
ホスト構造配列の参照の形式は、ホスト変数参照の形式と同じです。:S1:S2 の参照は、S1 がホスト構造配列を指す場合は、ホスト構造配列に対する参照です。S1 がホスト構造を指す場合、S2 は、短整数ホスト変数、短整数ホスト変数の配列、または短整数ホスト変数の 2 次元の配列のいずれかでなければなりません。次の例では、S1 はホスト構造配列であり、S2 はその標識配列です。

```
EXEC  SQL FETCH CURSOR1 FOR 5 ROWS
          INTO :S1:S2;
```

ホスト構造と標識配列の次元は、等しくなければなりません。

ホスト構造に、標識配列より *m 個* だけ多い変数がある場合、ホスト構造の最後の *m 個* の変数は、標識変数を持ちません。ホスト構造に、標識配列より *m 個* だけ少ない変数がある場合、標識配列の最後の *m 個* の変数は無視されます。標識配列または標識変数の指定がない場合は、ホスト構造配列中の変数には標識変数がありません。

次の図は、ホスト構造の配列に対する参照の構文を示しています。



ホスト構造の配列は、REXX ではサポートされません。

関数

関数 とは、関数名の後に、括弧で囲んだ 1 つまたは複数のオペランドを指定することによって実行される演算です。関数は、入力の値のセットと結果の値のセットの間の関係を表しています。関数への入力の値は、*引き数* と呼ばれています。例えば、関数に日時のデータ・タイプを持った 2 つの引き数を渡し、結果としてタイム・スタンプ・データ・タイプの戻り値を渡すことができます。

関数のタイプ

関数を分類する方法は、いくつかあります。その 1 つの方法は、組み込み、ユーザー定義、または特殊タイプ用に生成されたユーザー定義関数として分類することです。

- *組み込み関数* は IBM 提供の関数であり、DB2 UDB for iSeries とともに提供されます。これらの関数は、単一の値の結果を提供します。組み込み関数は、″+″ のような演算子関数、AVG のような列関数、あるいは SUBSTR のようなスカラー関数が含まれています。組み込みの列およびスカラー関数のリストとこれらの関数についての詳細については、169 ページの『第 3 章 組み込み関数』を参照してください。[25]
- *ユーザー定義関数* は、CREATE FUNCTION ステートメントを使用して作成され、カタログ表 QSYS2.SYSROUTINES およびカタログ・ビュー QSYS2.SYSFUNCS で データベース・マネージャーに登録されます。これらの関数により、ユーザー独自の、あるいはサード・パーティーのベンダーの関数定義を追加することによって、データベース・マネージャーの機能を拡張することができます。

  ユーザー定義関数は、*SQL*、*外部*、またはソース のいずれかです。SQL 関数は、SQL ステートメントのみを使用して、データベースに定義されます。外部関数は、関数が呼び出されたときに実行される外部プログラムまたはサービス・プログラムへの参照を伴って、データベースに定義されます。ソース関数は、組み込み関数または別のユーザー定義関数への参照を伴って、データベースに定義されます。ソース関数を使用して、特殊タイプで用いる組み込みの列およびスカラー関数を拡張することができます。

  ユーザー定義関数は、それが作成されたスキーマに常駐します。そのスキーマが、QSYS、QSYS2、または QTEMP ということはあり得ません。
- データベース・マネージャーは、CREATE DISTINCT TYPE ステートメントを使用して特殊タイプが作成されると、いくつかのユーザー定義関数を自動的に生成します。これらの関数は、特殊タイプからソース・タイプへ、さらにソース・タイプから特殊タイプへのキャストをサポートします。特殊タイプはそれ自体とのみしか互換性がないため、データ・タイプ間のキャストの可能性は重要です。

  生成されたキャスト関数は、対象となった特殊タイプと同じスキーマに常駐します。そのスキーマが、QSYS、QSYS2、または QTEMP ということはあり得ません。特殊タイプ用に生成される関数の詳細については、439 ページの『CREATE DISTINCT TYPE』を参照してください。

25. *組み込み関数* は、データベース・マネージャーによって内部的にインプリメントされており、したがって、関連するプログラムやサービス・プログラム・オブジェクトは、*組み込み関数* には存在しません。さらに、カタログには *組み込み関数* についての情報は含まれていません。しかしながら、*組み込み関数* は、あたかも QSYS2 に存在しているように取り扱うことができ、*組み込み関数* 名は QSYS2 で修飾することができます。

関数を分類するもう 1 つの方法では、入力データの値と結果の値によって、列関数、スカラー関数、または表関数として分類します。

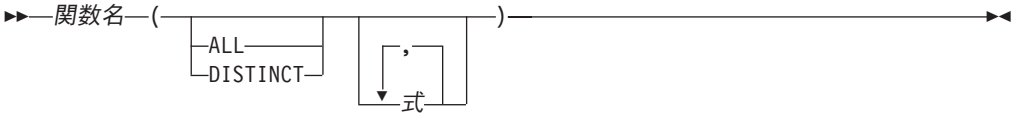
列関数 は、それぞれの引き数ごとに値のセット (列の値など) を受け取り、入力値のセットについて単一の値の結果を戻します。列関数は、しばしば、集約関数と呼ばれます。組み込み関数およびユーザー定義のソース関数は、列関数になり得ます。

スカラー関数 は、それぞれの引き数ごとに単一の値を受け取り、単一の値の結果を戻します。組み込み関数およびユーザー定義関数は、スカラー関数になり得ます。また、特殊タイプ用に作成される関数もスカラー関数です。

表関数 は、受け取った引き数のセットに関する表を戻します。各引き数はそれぞれ単一の値です。表関数は、副選択の FROM 文節の中でのみ参照することができます。表関数は、外部関数または SQL 関数として定義できません (表関数はソース関数となることはできません)。

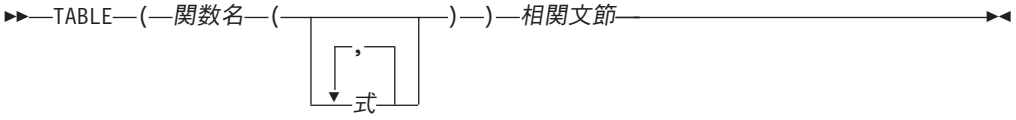
表関数を使用することにより、SQL 言語処理能力を DB2 データ以外のデータに適用すること、またはその種のデータを DB2 表に変換することができます。例えば、表関数により、特定のファイルを表に変換したり、WWW から入手したデータを表にしたり、Lotus Notes データベースにアクセスしてメール・メッセージに関する情報を戻したりすることができます。

スカラー関数または列関数 (組み込みまたはユーザー定義のいずれも) への参照は、次の構文で標準化されています。



ALL または DISTINCT キーワードは、列関数または列関数をソースにしたユーザー定義関数の場合のみ指定することができます。

表関数に対する各参照は、以下の構文に従います。



上記の構文において、式 はスカラー関数または列関数の場合と同じです。表関数を参照する方法についての詳細は、344 ページの『FROM 文節』の FROM 文節に関する説明を参照してください。

関数解決

関数はその関数名によって呼び出されます。関数名は、暗黙的にまたは明示的にスキーマ名で修飾され、その後に括弧で囲まれた関数への引き数が続きます。データベース内では、それぞれの関数はその関数のシグニチャーによって一意的に識別されます。シグニチャーとは、そのスキーマ名、関数名、パラメーター数、およびパ

ラメーターのデータ・タイプのことです。このため、スキーマでは、関数名が同じでも、それぞれパラメーター数が異なるか、パラメーター・データ・タイプが異なるため、複数の関数を持つことができます。あるいは、名前やパラメーター数、パラメーターのタイプが同じ関数でも、別々のスキーマには存在することができます。関数を呼び出すと、データベース・マネージャーは実行すべき関数を決定する必要があります。このプロセスを、関数解決と呼びます。

関数解決は、修飾、または非修飾の関数名で呼び出される関数の場合と似ています。ただし、非修飾名の場合はデータベース・マネージャーは複数のスキーマを検索する必要があるという点は異なります。

修飾された関数解決: 関数が関数名とスキーマ名で呼び出されると、データベース・マネージャーは指定されたスキーマ名だけを検索して、実行する関数を決めます。データベース・マネージャーは、以下のすべての条件が満たされた場合には、適切な関数インスタンスを検出します。

- 関数インスタンスの名前が、関数呼び出しの名前と一致する。
- 関数インスタンスの入力パラメーター数が、関数呼び出しの引き数の数と一致する。
- 関数呼び出しのそれぞれの入力引き数のデータ・タイプが、関数インスタンスの対応するパラメーターのデータ・タイプと一致するか、または*プロモート可能* である。

  このデータ・タイプの比較の結果、最適なものがもたらされ、それが実行用に選択されます (133 ページの『最適検索の方法』を参照してください)。データ・タイプのプロモーションについての詳細は、 80 ページの『データ・タイプのプロモーション』を参照してください。

  このような基準を満たす関数がそのスキーマにない場合には、エラーが生じます。

修飾されない関数解決: 関数が関数名だけで呼び出されると、データベース・マネージャーは実行する関数を決めるためには、複数のスキーマを検索する必要があります。SQL パスは、検索するスキーマのリストを持っています。パス (パスの詳細については、 57 ページの『スキーマと SQL パス』を参照してください) におけるそれぞれのスキーマごとに、データベース・マネージャーは、以下の基準に基づいて候補となる関数を選択します。

- 関数インスタンスの名前が、関数呼び出しの名前と一致する。
- 関数インスタンスの入力パラメーター数が、関数呼び出しの関数の引き数の数と一致する。
- 関数呼び出しのそれぞれの入力引き数のデータ・タイプが、関数インスタンスの対応するパラメーターのデータ・タイプと一致するか、または*プロモート可能* である。

  このデータ・タイプの比較の結果、最適なものがもたらされ、それが実行用に選択されます (133 ページの『最適検索の方法』を参照してください)。データ・タイプのプロモーションについての詳細は、 80 ページの『データ・タイプのプロモーション』を参照してください。

  このような基準を満たす関数がそのスキーマにない場合には、エラーが生じます。

1 つまたは複数の基準が満たされない場合は、そのスキーマでは候補の関数は選択されません。

データベース・マネージャーは、候補の関数を識別した後で、実行する関数として最適のものを選択します (『最適検索の方法』を参照してください)。最適 (関数シグニチャーがスキーマ名を除いて同一) の関数インスタンスが、複数のスキーマにある場合は、データベース・マネージャーは SQL パスで一番早いスキーマの関数を選択します。

関数解決は、組み込み関数を含むすべての関数に適用されます。組み込み関数は、スキーマ QSYS2 に論理的に存在します。スキーマ QSYS2 が SQL パスで明示的に指定されていない場合は、スキーマは暗黙的にそのパスの前にあるものと見なされます。したがって、修飾されない関数名を指定する場合には、必ず意図した関数が選択されるようにパスを指定してください。

最適検索の方法

実行の候補となるような同じ名前の関数が、複数存在する場合があります。そのような場合、データベース・マネージャーは、引き数とパラメーターのデータ・タイプを比較して、呼び出し用に最適な関数を決めます。この決定で考慮されるのは、関数の結果のデータ・タイプでも関数のタイプ (列またはスカラー) でもないことに注意してください。

ある関数のすべてのパラメーターのデータ・タイプが関数呼び出しの引き数のデータ・タイプと同じ場合には、その関数が最適となります。完全な一致が 1 つもない場合には、データベース・マネージャーは以下の方法を使用してパラメーター・リストのデータ・タイプを左から右へ比較します。

1. 関数呼び出しの最初の引き数のデータ・タイプを、それぞれの関数の最初のパラメーターのデータ・タイプと比較します。 (長さ、精度、位取り、および CCSID 属性は、比較の際はいずれも考慮されません。)
2. この引き数について、ある関数のデータ・タイプが他の関数よりも関数呼び出しに合っている場合は、その関数が最適となります。 80 ページの『データ・タイプのプロモーション』のデータ・タイプのプロモーションについての優先順位リストでは、それぞれのデータ・タイプに合うデータ・タイプを、良いものから悪いものへの順に示しています。
3. 最初のパラメーターのデータ・タイプが、関数呼び出しと複数の関数で同じように合っている場合は、関数呼び出しの次の引き数についてこのプロセスを繰り返します。最適なものが見つかるまで、それぞれの引き数について続けます。

以下、関数解決の例を示します。

例 1: MYSCHEMA は 2 つの関数を持っており、両方とも FUNA という名前であると想定します。これらの関数は、次に一部を示す CREATE FUNCTION ステートメントで作成されました。

```
CREATE FUNCTION MYSCHEMA.FUNA (VARCHAR(10), INT, DOUBLE) ...
CREATE FUNCTION MYSCHEMA.FUNA (VARCHAR(10), REAL, DOUBLE) ...
```

さらに、データ・タイプが VARCHAR(10)、SMALLINT、および DECIMAL の 3 つの引き数を持つ関数が、修飾名で呼び出されたものとします。

```
MYSCHEMA.FUNA( VARCHARCOL, SMALLINTCOL, DECIMALCOL  ) ...
```

両方の MYSCHEMA.FUNA 関数とも、 131 ページの『関数解決』で指定された基準を満たしているため、この関数呼び出しの候補となります。スキーマの 2 つの関数インスタンスの最初のパラメーターのデータ・タイプ (この場合は、両方とも VARCHAR) は、関数呼び出しの最初の引き数 (この場合、VARCHAR) と同じように合っています。しかしながら、2 番目のパラメーターについては、最初の関数のデータ・タイプ (INT) の方が、2 番目の関数のデータ・タイプ (REAL) よりも、2 番目の引き数のデータ・タイプ (SMALLINT) とより合っています。したがって、データベース・マネージャーは、MYSCHEMA.FUNA 関数を実行する関数インスタンスとして選択します。

例 2: 次に一部を示す CREATE FUNCTION ステートメントによって、複数の関数が作成されたものと想定します。

```
1. CREATE FUNCTION SMITH.ADDIT (CHAR(5), INT, DOUBLE) ...
2. CREATE FUNCTION SMITH.ADDIT (INT, INT, DOUBLE) ...
3. CREATE FUNCTION SMITH.ADDIT (INT, INT, DOUBLE, INT) ...
4. CREATE FUNCTION JOHNSON.ADDIT (INT, DOUBLE, DOUBLE) ...
5. CREATE FUNCTION JOHNSON.ADDIT (INT, INT, DOUBLE) ...
6. CREATE FUNCTION TODD.ADDIT (REAL) ...
7. CREATE FUNCTION TAYLOR.SUBIT (INT, INT, DECIMAL) ...
```

さらに、アプリケーションが関数を呼び出す際の SQL パスは、 ″TAYLOR″、 ″JOHNSON″、 ″SMITH″ であるものとします。この関数は、次のように、3 つのデータ・タイプ (INT、INT、DECIMAL) で呼び出されています。

```
SELECT  ... ADDIT(INTCOL1, INTCOL2, DECIMALCOL) ...
```

関数 5 が、以下の評価に基づいて、実行する関数インスタンスとして選択されています。

- 関数 6 は、スキーマ TODD が SQL パスにないため、候補から除外します。
- スキーマ TAYLOR の関数 7 は、関数名が正しくないため、候補から除外します。
- スキーマ SMITH の関数 1 は、INT データ・タイプは関数 1 の最初のデータ・タイプである CHAR にプロモートできないため、候補から除外します。
- スキーマ SMITH の関数 3 は、パラメーターの数が間違っているため、候補から除外します。
- 関数 2 はそのパラメーターのデータ・タイプが引き数のデータ・タイプにプロモートできるため、これは候補です。
- スキーマ JOHNSON の関数 4 と 5 は、パラメーターのデータ・タイプが引き数のデータ・タイプに一致するか、プロモートできるため、両方とも候補です。ただし、関数 5 の方がよりよい候補として選択されます。その理由は、両方の関数の最初のパラメーターのデータ・タイプ (INT) は最初の引き数 (INT) と一致していますが、関数 5 の 2 番目のパラメーターのデータ・タイプ (INT) は、関数 4 のデータ・タイプ (DOUBLE) よりも、2 番目の引き数 (INT) により合っているからです。
- 残りの候補である関数 2 と関数 5 では、スキーマ JOHNSON の方がスキーマ SMITH よりも SQL パス上前にあるため、データベース・マネージャーは関数 5 を選択します。

例 3: 次に一部を示す CREATE FUNCTION ステートメントによって、複数の関数が作成されたものと想定します。

```
1. CREATE FUNCTION BESTGEN.MYFUNC (INT, DECIMAL(9,0)) ...
2. CREATE FUNCTION KNAPP.MYFUNC  (INT, NUMERIC(8,0))...
3. CREATE FUNCTION ROMANO.MYFUNC  (INT, FLOAT) ...
```

さらに、アプリケーションが関数を呼び出す際の SQL パスは、 ″ROMANO″、 ″KNAPP″、 ″BESTGEN″ であるとします。この関数は、次のように、2 つのデータ・タイプ (SMALLINT、DECIMAL) で呼び出されています。

```
SELECT  ... MYFUNC(SINTCOL1, DECIMALCOL) ...
```

関数 2 が、以下の評価に基づいて、実行する関数インスタンスとして選択されています。

- 3 つの関数ともすべて、 131 ページの『関数解決』で指定された基準を満たしているため、3 つともこの関数呼び出しの候補です。
- スキーマ ROMANO の関数 3 は、2 番目のパラメーター (FLOAT) が、関数 1 の 2 番目のパラメーター (DECIMAL) や関数 2 (NUMERIC) のいずれよりも、2 番目の引き数 (DECIMAL) に対して適合がよくないため、除外します。
- 関数 1 の 2 番目のパラメーター (DECIMAL) および関数 2 (NUMERIC) の 2 番目の引き数 (DECIMAL) に対する適合は、同じ程度です。
- ″KNAPP″ が ″BESTGEN″ よりも SQL パス上先行するため、関数 2 が最終的に選択されます。

関数の呼び出し

いったん関数が選択されても、まだ、関数の使用が許可されない理由が考えられます。それぞれの関数は、特定のデータ・タイプの結果を戻すように定義されています。結果のデータ・タイプが、関数の呼び出される文脈内で互換性がない場合には、エラーが起こることになります。例えば、次のように、STEP という名前の 2 つの関数が、結果として別のデータ・タイプで定義されていたとします。

```
STEP(SMALLINT) RETURNS CHAR(5)
STEP(DOUBLE) RETURNS INTEGER
```

そして、次の関数参照がありました (ここで、S は SMALLINT 列)。

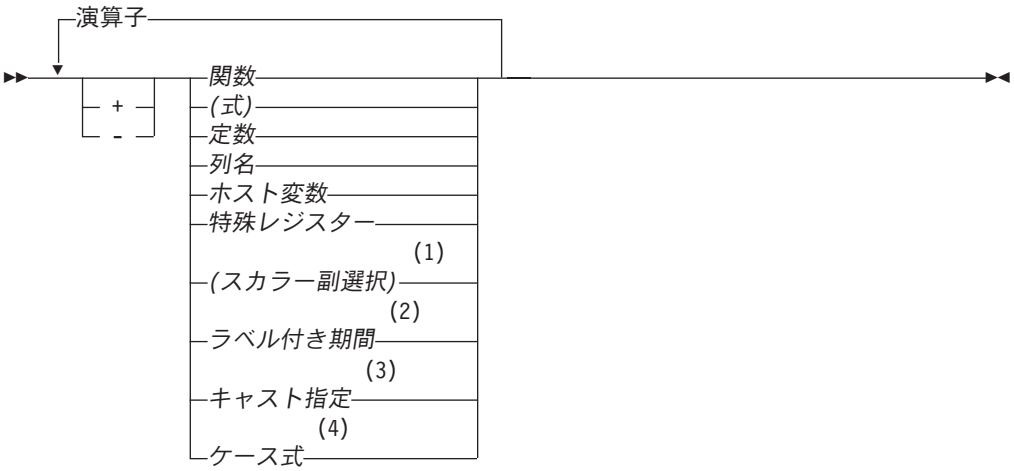
```
SELECT ... 3 +STEP(S)
```

次に、引き数のタイプが完全に一致するため、最初の STEP が選択されます。加算演算子の引き数で要求される数値タイプの代わりに、結果のタイプが CHAR(5) であるため、このステートメントでエラーが起こることになります。

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・タイプに完全に一致しない場合は、列への割り当て (85 ページの『割り当ておよび比較』を参照してください) と同じ規則を使って、引き数は実行時にパラメーターのデータ・タイプに変換されます。これには、精度、位取り、長さ、または CCSID が引き数とパラメーター間で異なっているケースも含まれます。

式

式では、値を指定します。



注:

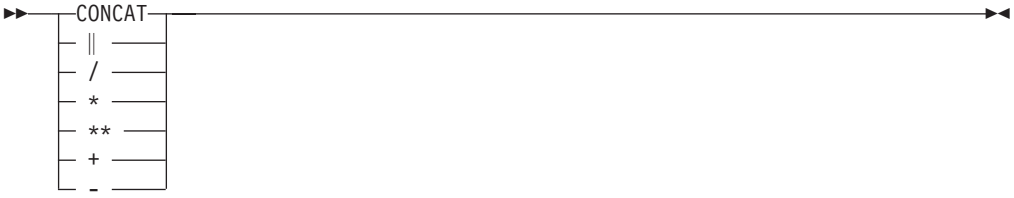
1 詳細については、 141 ページの『スカラー副選択』を参照してください。

2 詳細については、 141 ページの『日付/時刻のオペランドと期間』を参照してください。

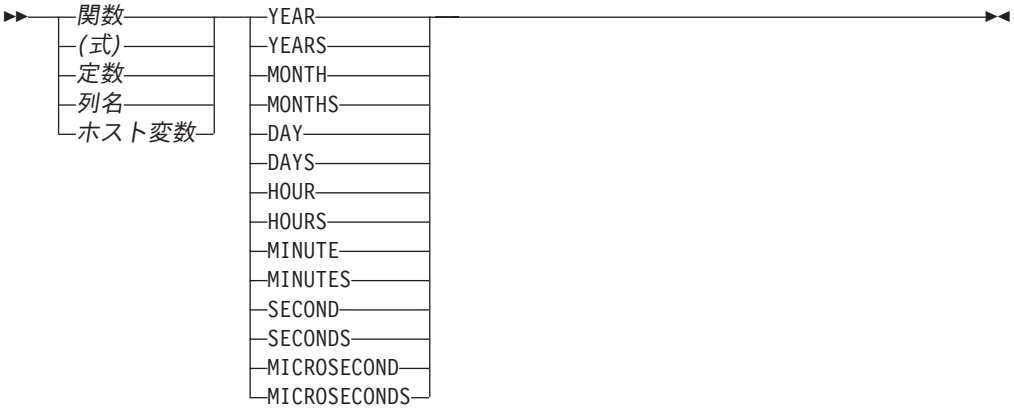
3 詳細については、 149 ページの『CAST の指定』を参照してください。

4 詳細については、 147 ページの『CASE 式』を参照してください。

演算子:



ラベル付き期間:



演算子を使用しない式

演算子を使用しない式では、指定された値が式の結果になります。

例

```
SALARY      :SALARY      'SALARY'      MAX(SALARY)
```

連結演算子を使用する式

連結演算子 (CONCAT または ||) は、2 つのストリングを結合させます。この式の結果は、ストリングになります。

連結のオペランドは、互換性のあるストリングでなければなりません。 2 進ストリングは、他の 2 進ストリングとのみ互換性があります。

結果のデータ・タイプは、データ・タイプによって決まります。結果のデータ・タイプは、次の表に要約されています。

表 20. 連結を用いた結果のデータ・タイプ

一方のオペランド列	他方のオペランド	結果列のデータ・タイプ
DBCLOB(x)	CHAR(y) または VARCHAR(y) または CLOB(y) または GRAPHIC(y) または VARGRAPHIC(y) または DBCLOB(y)	DBCLOB(z) (ただし、z = MIN(x + y, DBCLOB の最大長))
CLOB(x)	GRAPHIC(y) または VARGRAPHIC(y)	DBCLOB(z) (ただし、z = MIN(x + y, DBCLOB の最大長))
VARGRAPHIC(x)	CHAR(y) または VARCHAR(y) または GRAPHIC(y) または VARGRAPHIC(y)	VARGRAPHIC(z) (ただし、z = MIN(x + y, VARGRAPHIC の最大長))
VARCHAR(x)	GRAPHIC(y)	VARGRAPHIC(z) (ただし、z = MIN(x + y, VARGRAPHIC の最大長))
GRAPHIC(x)	CHAR(y) 混合データ	VARGRAPHIC(z) (ただし、z = MIN(x + y, VARGRAPHIC の最大長))
GRAPHIC(x)	CHAR(y) SBCS データまたは GRAPHIC(y)	GRAPHIC(z) (ただし、z = MIN(x + y, GRAPHIC の最大長))
UCS-2 データ	UCS-2 または DBCS または混合 または SBCS データ	UCS-2 データ
DBCS データ	DBCS または混合または SBCS データ	DBCS データ
CLOB(x)	CHAR(y) または VARCHAR(y) または CLOB(y)	CLOB(z) (ただし、z = MIN(x + y, CLOB の最大長))
VARCHAR(x)	CHAR(y) または VARCHAR(y)	VARCHAR(z) (ただし、z = MIN(x + y, VARCHAR の最大長))

表 20. 連結を用いた結果のデータ・タイプ (続き)

一方のオペランド列	他方のオペランド	結果列のデータ・タイプ
CHAR(x) 混合データ	CHAR(y)	VARCHAR(z) (ただし、z = MIN(x + y, VARCHAR の最大長))
CHAR(x) SBCS データ	CHAR(y)	CHAR(z) (ただし、z = MIN(x + y, CHAR の最大長))
ビット・データ	混合または SBCS またはビット・データ	ビット・データ
混合データ	混合または SBCS データ	混合データ
SBCS データ	SBCS データ	SBCS データ
BLOB(x)	BLOB(y)	BLOB(z) (ただし、z = MIN(x + y, BLOB の最大長))

両方のオペランドの合計長が結果のデータ・タイプの最大長属性を超える場合は、次のようになります。

- 結果の長さ属性が結果のデータ・タイプの最大長になります。[26]
- ブランクのみが切り捨てられた場合は、警告もエラーも生じません。
- ブランク以外の文字が切り捨てられた場合は、エラーが起こります。

ヌルになる可能性があるオペランドをどちらか一方に使用した場合は、結果もヌルになる可能性があります。また、どちらか一方がヌルならば、結果はヌル値になります。それ以外の場合、結果は、第 1 オペランドのストリングの後に、第 2 オペランドのストリングが続いたストリングになります。

混合データを連結する場合は、その結果の 『継ぎ目に』 余分なシフト・コードが入ることはありません。したがって、第 1 オペランドのストリングが 『シフトイン』 文字 (X'0F') で終わり、第 2 オペランドの文字ストリングが 『シフトアウト』 文字 (X'0E') で始まっていても、第 1 オペランドのシフトイン文字と第 2 オペランドのシフトアウト文字 (合わせて 2 バイト) は結果から除去されます。

余分なシフト文字が除去された場合を除いて、オペランドの長さの合計が実際の結果の長さになります。余分なシフト文字が除去された場合は、実際の結果の長さは、オペランドの長さの合計よりも 2 だけ小さくなります。

|| 演算子ではなく、CONCAT 演算子を使用することをお勧めします。文字 | のコード・ポイントは、CCSID に応じて変わります。

結果の CCSID は、オペランドの CCSID によって決定されます (これについては、103 ページの『ストリングを結合する演算に適用される変換規則』で説明しています)。これらの規則による結果として、以下の点に注意してください。

- ビット・データのオペランドがあれば、結果はビット・データになります。

26. 該当の式が選択リストに含まれている場合は、長さ属性は、最大レコード・サイズに収まるようにするためにさらに縮小されることがあります。詳しくは、569 ページの『最大行サイズ』を参照してください。

- 一方のオペランドが混合データで、もう一方が SBCS データの場合、結果は混合データになります。ただし、このことは、その結果が、形式が正しい混合データであることを必ずしも意味するわけではありません。

例

ブランクを間に置いて、列 FIRSTNME と列 LASTNAME を連結します。

```
FIRSTNME CONCAT ' ' CONCAT LASTNAME
```

算術演算子を使用する式

算術演算子を使用すると、その演算子をオペランドの値に適用して得られた数値が式の結果となります。

ヌルになる可能性があるオペランドを使用すると、結果もヌルになる可能性があります。どちらか一方のオペランドがヌル値の場合、式の結果はヌル値になります。文字ストリングに対して算術演算子を使用してはなりません。例えば、USER+2 という式は無効です。

接頭演算子として + を使用しても、オペランドは変更されません (これを*単項プラス* と呼びます)。接頭演算子として - を使用すると、ゼロ以外の値を持つオペランドの符号が反転します (これを*単項マイナス* と呼びます)。A のデータ・タイプが短整数の場合、-A のデータ・タイプは長整数になります。接頭演算子の後に続く最初の文字は、正符号や負符号であってはなりません。

挿入演算子 の +、-、*、/、および ** は、それぞれ加算、減算、乗算、除算、および累乗演算を示します。除算の第 2 オペランドの値は、ゼロ以外でなければなりません。

累乗演算演算子 (**) の結果は、倍精度浮動小数点数になります。その他の演算子の結果は、オペランドのタイプによって決まります。

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で、それぞれの位取りがゼロであれば、いずれかの (あるいは両方の) オペランドが 64 ビット整数でない限り、2 進数の演算が実行され、結果は長整数となります。いずれかの (あるいは両方の) オペランドが 64 ビット整数の場合は、結果は 64 ビット整数となります。この場合は、除算で剰余があっても、その剰余は失われます。整数の算術演算 (単項マイナスを含む) の結果は、長整数の値の範囲内になければなりません。どちらかの整数オペランドがゼロ以外の位取りを持つ場合は、そのオペランドが同一の精度および位取りの 10 進数オペランドに変換されます。

整数オペランドと 10 進数オペランド

一方のオペランドが位取りゼロの整数で、もう一方が 10 進数の場合は、整数の一時的なコピー (整数のオペランドを、以下の表に定義されている精度を持つ 10 進数 (位取りはゼロ) に変換したもの) を使用して、10 進数の演算が実行されます。

オペランド	10 進数のコピーの精度
列または変数 : 64 ビット整数	19
列または変数 : 長整数	11

オペランド	10 進数のコピーの精度
列または変数 : 短整数	5
定数 (先行ゼロを含む)	定数の桁数と同じ。

一方のオペランドが位取りがゼロ以外の整数の場合、まず、そのオペランドが同一の精度および位取りを持つ 10 進数オペランドに変換されます。

2 つの 10 進数オペランド

オペランドが両方とも 10 進数の場合は、10 進数の演算が実行されます。10 進算術演算の結果は、必ず 10 進数になります。結果の精度および位取りは、実行された演算とオペランドの精度および位取りによって決まります。加算または減算を実行するときに、2 つのオペランドの位取りが異なっている場合は、一方のオペランドの一時的なコピーを使用して演算が実行されます。この一時的なコピーは、2 つのオペランドの小数部分の桁数が同じになるように、位取りが小さい方のオペランドに後書きゼロを付加して、小数部分の桁数を増やしたものです。

特に指定のないかぎり、10 進数を使用できるすべての関数および演算では、最高 31 桁までの精度が使用できます。10 進演算の結果の精度は、31 桁以下でなければなりません。

SQL における 10 進数演算

SQL における 10 進数演算の結果の精度および位取りは、以下の各式によって定義されます。記号 *p* および *s* は、それぞれ第 1 オペランドの精度と位取りを示し、記号 *p'* および *s'* は、それぞれ第 2 オペランドの精度と位取りを示します。

加算および減算

加算および減算の結果の位取りは、max(*s,s'*) です。精度は、min(31,max(*p-s,p'-s'*) +max(*s,s'*)+1) です。

乗算

乗算の結果の精度は min (31,*p+p'*) であり、位取りは min(31,*s+s'*) です。

除算

除算の結果の精度は 31 です。位取りは、*31-p+s-s'* です。位取りは、正の数でなければなりません。

浮動小数点数オペランド

算術演算子のオペランドのいずれかが浮動小数点数である場合は、演算は浮動小数点数で行われます。必要であれば、オペランドがまず倍精度浮動小数点数に変換されます。したがって、式の要素の中に浮動小数点数がある場合、その式の結果は倍精度の浮動小数点数になります。

浮動小数点数と整数の演算は、その整数を倍精度浮動小数点数に変換した一時的コピーを使用して行われます。浮動小数点数と 10 進数の演算は、10 進数を倍精度浮動小数点数に変換した一時的コピーを使用して行われます。浮動小数点数演算の結果は、浮動小数点数の値の範囲内になければなりません。

浮動小数点オペランドは実数の概数を表すものなので、浮動小数点オペランド (または関数への引き数) が処理される順序によって、結果が少しずつ変わることがあります。オペランドの処理順序は、最適化プログラムにより暗黙的に変更されることがあるので (例えば、最適化プログラムは、どの程度の並列性を使用するか、およびどのアクセス・プランを使用するかなどを決定します)、浮動小数点オペランドを使用する SQL ステートメントを実行する場合は、結果がいつも正確に同じになるという前提でアプリケーションを使用しないようにしてください。

オペランドとしての特殊タイプ

特殊タイプは、そのソース・データ・タイプが数値であっても、算術演算子で使用することはできません。 算術演算を行うには、そのソースとして算術演算子を持つ関数を作成します。例えば、特殊タイプ INCOME および EXPENSES があり、両方とも DECIMAL(8,2) のデータ・タイプを持っている場合、次のユーザー定義関数 REVENUE を使用して、他方から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・タイプを減算するユーザー定義関数を使用して、- (マイナス) 演算子を多重定義する方法があります。

```
CREATE FUNCTION "-" ( INCOME, EXPENSES )
  RETURNS DECIMAL(8,2) SOURCE "-" ( DECIMAL, DECIMAL)
```

スカラー副選択

式の中で使用できるスカラー副選択は、括弧で囲んだ副選択で、単一の列値から成る単一の行を戻します。この副選択が行を戻さない場合は、式の結果はヌル値になります。副選択リスト・エレメントが、単なる列名である式である場合は、その列の名前に基づいて結果の列名が決まります。詳細については、340 ページの『副選択』を参照してください。

日付/時刻のオペランドと期間

日付/時刻の値に対して、増分や減分、および減算を行うことができます。これらの演算では、*期間* と呼ばれる 10 進数を使用することができます。この*期間* は、時間間隔を表す正または負の数値です。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 (125 ページの 図を参照)

ラベル付き期間 とは、数値 (式の結果である場合もある) の後に 7 つの期間キーワードのいずれか 1 つを付けて、特定の時間単位を表すものです。期間キーワードには、YEARS、MONTHS、DAYS、HOURS、MINUTES、SECONDS、および MICROSECONDS があります。[27] 期間キーワードの前に指定される数値は、10 進数 (15,0) に割り当てられた場合と同じように変換されます。ラベル付き期間は、算術演算子のオペランドの一方がデータ・タイプとして DATE、TIME、または

27. これらのキーワードの単数形式も使用できることに注意してください。すなわち、YEAR、 MONTH、 DAY、 HOUR、 MINUTE、 SECOND、および MICROSECOND です。

TIMESTAMP を持つ値である場合にのみ、もう一方のオペランドとして使用することができます。したがって、HIREDATE + 2 MONTHS + 14 DAYS という式は有効ですが、HIREDATE + (2 MONTHS + 14 DAYS) という式は無効です。この 2 つの式で、2 MONTHS および 14 DAYS がラベル付き期間です。

日付期間

日付期間 は、年、月、および日の数を 10 進数 (8,0) の数値として表します。日付期間が正しく解釈されるためには、この数値の形式が、*yyyymmdd* (*yyyy* は年の数、*mm* は月の数、*dd* は日の数) でなければなりません。ある日付の値から別の日付の値を引いた結果 (例えば、式 HIREDATE - BRTHDATE の結果) は、日付期間になります。

時刻期間

時刻期間 は、時、分、および秒の数を 10 進数 (6,0) の数値として表します。時刻期間が正しく解釈されるためには、この数値の形式が、*hhmmss* (*hh* は時の数、*mm* は分の数、*ss* は秒の数) でなければなりません。ある時刻の値から別の時刻の値を引いた結果は、時刻期間になります。

タイム・スタンプ期間

タイム・スタンプ期間 は、年、月、日、時、分、秒、およびマイクロ秒の数を 10 進数 (20,6) の数値として表します。タイム・スタンプ期間が正しく解釈されるためには、この数値の形式が、
yyyymmddhhmmsszzzzzz
(*yyyy*、*mm*、*dd*、*hh*、*mm*、*ss*、および *zzzzzz* は、順に年、月、日、時、分、秒、およびマイクロ秒を表す) でなければなりません。タイム・スタンプ値から別のタイム・スタンプ値を引いた結果は、タイム・スタンプ期間となります。

SQL における日付/時刻の値の演算

日付/時刻の値に対して実行できる算術演算は、加算と減算だけです。日付/時刻の値が加算のオペランドである場合は、もう一方のオペランドは期間でなければなりません。日付/時刻の値に対する加算の演算子の用法に関する特定の規則は、次のとおりです。

- 一方のオペランドが日付の場合、もう一方のオペランドは日付期間、または年、月、日のラベル付き期間のいずれかでなければなりません。
- 一方のオペランドが時刻の場合、もう一方のオペランドは時刻期間、または時、分、秒のラベル付き期間のいずれかでなければなりません。
- 一方のオペランドがタイム・スタンプの場合、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、どのようなタイプの期間でも使用できます。
- 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/時刻の値は、期間から減算することはできず、また日付/時刻の 2 つの値の減算の処理は、日付/時刻の値から期間を減算する処理とは異なるので、日付/時刻の値に対する減算演算子の用法に関する規則は、加算の場合と同じではありません。日付/時刻の値に対する減算演算子の用法に関する特定の規則は、次のとおりです。

- 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現、または年、月、日のラベル付き期間のいずれかでなければなりません。
- 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現のいずれかでなければなりません。
- 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現、または時、分、秒のラベル付き期間のいずれかでなければなりません。
- 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現のいずれかでなければなりません。
- 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、タイム・スタンプのストリング表現、または期間のいずれかでなければなりません。
- 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現のいずれかでなければなりません。
- 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、減算を行えるほかに、増やしたり減らしたりすることができます。

日付の減算: ある日付 (DATE1) から別の日付 (DATE2) を引いた結果は、その 2 つの日付の間にある年、月および日の数を示す日付期間になります。この結果のデータ・タイプは、10 進数 (8,0) です。 DATE1 が DATE2 より大きいか、または両者が等しい場合、DATE1 から DATE2 が引かれます。 DATE1 が DATE2 より小さい場合でも、DATE2 から DATE1 が引かれ、結果の符号が負になります。以下の手順型の記述は、RESULT = DATE1 - DATE2 という演算に伴う各ステップを説明したものです。

```
DAY(DATE2) <= DAY(DATE1) の場合 :
    DAY(RESULT) = DAY(DATE1) - DAY(DATE2)

DAY(DATE2) > DAY(DATE1) の場合 :
    DAY(RESULT) = N + DAY(DATE1) - DAY(DATE2)
      ただし、N = MONTH(DATE2) の最後の日
    MONTH(DATE2) は 1 だけ増やされる

MONTH(DATE2) <= MONTH(DATE1) の場合 :
    MONTH(RESULT) = MONTH(DATE1) - MONTH(DATE2)

MONTH(DATE2) > MONTH(DATE1) の場合 :
    MONTH(RESULT) = 12 + MONTH(DATE1) - MONTH(DATE2)
```

YEAR(DATE2) は 1 だけ増やされる

YEAR(RESULT) = YEAR(DATE1) - YEAR(DATE2)

例えば、DATE('3/15/2000') - '12/31/1999' は 215 (つまり、0 年、2 月、15 日という期間) になります。

日付の増減: 日付に期間を加えた結果や、日付から期間を引いた結果は、それ自身が日付になります。 (この演算の目的に沿って、月はカレンダーのページと同等の意味を持ちます。日付に月を加えるのは、その日付があるカレンダーのページを月の数だけめくることに相当します。) これらの演算の結果は、0001 年 1 月 1 日から 9999 年 12 月 31 日までの間にならなければなりません。日付に対して、年の期間を加えたり引いたりした場合、増減されるのはその日付の年の部分だけです。結果がうるう年以外の年の 2 月 29 日にならないかぎり、月は変更されません。結果がうるう年以外の年の 2 月 29 日になった場合は、日が 28 に変更され、月の最後の日の調整を行ったことを示す 'W' が SQLCA の SQLWARN6 にセットされます。

同様に、日付に対して月の期間を加えたり引いたりした場合も、まず日付の月の部分だけが増減され、必要があれば年の部分が増減されます。結果が無効な日付 (例えば、9 月 31 日など) にならないかぎり、日付の日の部分は変更されません。結果が 9 月 31 日などの無効な日付になった場合は、日の部分が該当する月の最終日に変更され、月の最後の日の調整を行ったことを示す 'W' が SQLCA の SQLWARN6 にセットされます。

日の期間を加えたり引いたりした場合も、まず日付の日の部分が増減され、必要がある場合にだけ月および年の部分が増減されます。DAYS のラベル付き期間を加えた場合は、月の最後の日の調整は行われません。

日付期間 (正または負) も、日付に加えたり、日付から引いたりすることができます。ラベル付き期間を使用すると、結果は有効な日付となります。この場合も、月の終了日の調整が必要ならば、SQLCA 内に警告標識がセットされます。

日付に正の日付期間を加えた場合や、日付から負の日付期間を引いた場合は、日付が年、月、日の順に、指定した数だけ増やされます。DATE1 + X (X は、正の DECIMAL(8,0)) は、次の式と等価です。

```
DATE1 + YEAR(X) YEARS + MONTH(X) MONTHS + DAY(X) DAYS
```

日付から正の日付期間を引いた場合や、日付に負の日付期間を加えた場合は、日付が日、月、年の順に、指定した数だけ減らされます。したがって、DATE1 - X (X は、正の DECIMAL(8,0)) は、次の式と等価です。

```
DATE1 - DAY(X) DAYS - MONTH(X) MONTHS - YEAR(X) YEARS
```

期間を日付に加えるときに、所定の日付に月を 1 つ加えると、一か月後の同じ日付になります。ただし、一か月後に同じ日付が存在しない場合は、例外となります。その場合は、一か月後の月の最終日が日付にセットされます。例えば、1 月 28 日に月を 1 つ加えると、2 月 28 日になります。また、1 月 29、30、または 31 日に月を 1 つ加えると、2 月 28 日 (うるう年以外) と 2 月 29 日 (うるう年) のいずれかになります。

注: 所定の日付に 1 つまたは複数の月を加えた上で、その結果から同数の月を引いても、最終的な日付が元の日付と同じにならない場合があります。

時刻の算術演算

時刻は、減算に加え、増やしたり減らしたりすることができます。

時刻の減算: ある時刻 (TIME1) から別の時刻 (TIME2) を引いた結果は、その 2 つの時刻の間にある時、分、および秒の数を示す時刻期間となります。この結果のデータ・タイプは 10 進数 (6,0) です。TIME1 が TIME2 より大きいか、または両者が等しい場合、TIME1 から TIME2 が引かれます。 TIME1 が TIME2 より小さい場合でも、TIME2 から TIME1 が引かれ、結果の符号が負になります。以下の手順型の記述は、RESULT = TIME1 - TIME2 という演算に伴う各ステップを説明したものです。

SECOND(TIME2) <= SECOND(TIME1) の場合 :
    SECOND(RESULT) = SECOND(TIME1) - SECOND(TIME2)

SECOND(TIME2) > SECOND(TIME1) の場合 :
    SECOND(RESULT) = 60 + SECOND(TIME1) - SECOND(TIME2)
    MINUTE(TIME2) は 1 だけ増やされる

MINUTE(TIME2) <= MINUTE(TIME1) の場合 :
    MINUTE(RESULT) = MINUTE(TIME1) - MINUTE(TIME2)

MINUTE(TIME2) > MINUTE(TIME1) の場合 :
    MINUTE(RESULT) = 60 + MINUTE(TIME1) - MINUTE(TIME2)
    HOUR(TIME2) は 1 だけ増やされる

HOUR(RESULT) = HOUR(TIME1) - HOUR(TIME2)

例えば、TIME('11:02:26') - '00:32:56' の結果は、102930 (10 時間、29 分、30 秒という期間) になります。

時刻の増減: 時刻に期間を加えた結果や、時刻から期間を引いた結果は、それ自身が時刻になります。時のオーバーフローやアンダーフローは破棄されるので、結果は常に時刻となります。時刻に対して、時の期間を加えたり引いたりした場合は、時刻の時の部分だけが増減されます。分および秒の部分は変更されません。

同様に、時刻に対して分の期間を加えたり引いたりした場合は、まず分の部分だけが増減され、必要があれば時の部分が増減されます。時刻の秒の部分は変更されません。

秒の期間を加えたり引いたりした場合も、時刻の秒の部分が増減され、必要があるときだけ分および時の部分が増減されます。

時刻期間 (正または負) を時刻に加えたり、時刻から引いたりすることもできます。この結果は、時刻が時、分、秒の順に、指定した数だけ増やされたり減らされたりしたものになります。TIME1 + X (“X” は 10 進数 (6,0) の数値) は、次の式と同等のものになります。

```
TIME1 + HOUR(X) HOURS + MINUTE(X) MINUTES + SECOND(X) SECONDS
```

タイム・スタンプの算術演算

タイム・スタンプは、減算が行えるほかに、増やしたり減らしたりすることができます。

タイム・スタンプの減算: あるタイム・スタンプ (TS1) から別のタイム・スタンプ (TS2) を引いた結果は、その 2 つのタイム・スタンプの間にある年、月、日、時、分、秒およびマイクロ秒の数を示すタイム・スタンプ期間になります。この結果のデータ・タイプは 10 進数 (20,6) です。TS1 が TS2 より大きいか、または両者が等しければ、TS1 から TS2 が引かれます。TS1 が TS2 より小さい場合でも、TS2 から TS1 が引かれ、結果の符号が負になります。以下の手順型の記述は、RESULT = TS1 - TS2 という演算に伴う各ステップを説明したものです。

MICROSECOND(TS2) <= MICROSECOND(TS1) の場合 :
MICROSECOND(RESULT) = MICROSECOND(TS1) -
MICROSECOND(TS2)

MICROSECOND(TS2) > MICROSECOND(TS1) の場合 :
MICROSECOND(RESULT) = 1000000 +
MICROSECOND(TS1) - MICROSECOND(TS2)
SECOND(TS2) は 1 だけ増やされる

タイム・スタンプの秒および分の部分の減算は、時刻の減算に適用される規則に従って行われます。
HOUR(TS2) <= HOUR(TS1) の場合 :
HOUR(RESULT) = HOUR(TS1) - HOUR(TS2)

HOUR(TS2) > HOUR(TS1) の場合 :
HOUR(RESULT) = 24 + HOUR(TS1) - HOUR(TS2)
DAY(TS2) は 1 だけ増やされる

タイム・スタンプの日の部分の減算は、日付の減算に適用される規則に従って行われます。

タイム・スタンプの増減: タイム・スタンプに期間を加えた結果や、タイム・スタンプから期間を引いた結果は、それ自身がタイム・スタンプになります。時のオーバーフローまたはアンダーフローが、結果の日付部分に桁送りされることを除いて、以前の項で述べたとおりの日付および時刻の算術演算が行われます。この結果は、有効な日付の範囲内になければなりません。マイクロ秒のオーバーフローは、秒に桁送りされます。

演算の優先順位

括弧の中にある式は最初に計算されます。括弧によって計算の順序が指定されていないときは、接頭演算子 (-、単項減算など) の後、かつ乗算および除算の前に、累乗演算を行います。乗算および除算は、加算および減算の前に行います。同じ優先レベルにある演算子は、左から右の順に処理されます。次の表は、すべての演算子

の優先順位を示しています。

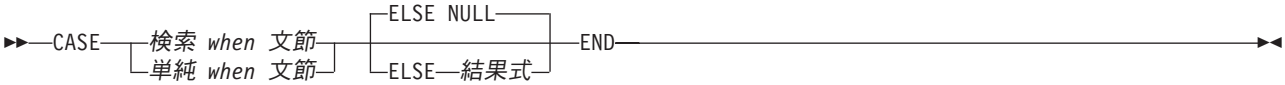
優先順位	演算子
1	+、- (符号付数値に使用する場合)
2	**
3	*、/、CONCAT、‖
4	+、- (2 つのオペランドの間で使用する場合)

例

以下の例では、演算子は 2 行目の番号で示す順序で適用されます。

```
1.10 * (SALARY + BONUS) + SALARY / :VAR3
     2         1        4        3
```

CASE 式



検索 when 文節:

単純 when 文節:

CASE 式により、1 つまたは複数の条件の評価に基づいて式を選択することができます。一般に、case 式の値は、真であることを評価する最初の (左端の) when 文節に続く *結果式* の値です。 when 文節が真であると評価せず、かつ ELSE キーワードが存在する場合は、結果は ELSE *結果式* の値または NULL になります。 when 文節が真であると評価せず、かつ ELSE キーワードが存在しない場合は、結果は NULL になります。when 文節が不明 (ヌルのため) と評価した場合は、when 文節は真ではなく、したがって、偽であると評価する when 文節と同じ方法で扱われます。

単純 when 文節 を使用する場合は、最初の WHEN キーワードの前の式 の値が WHEN キーワードの後の式 の値と等しいかテストされます。最初の WHEN キーワードの前の 式 のデータ・タイプは、WHEN キーワードの後のそれぞれの式 のデータ・タイプと互換性がある必要があります。

結果式 は、THEN または ELSE キーワードの後の式 です。 CASE 式では、少なくとも 1 つの *結果式* がなければなりません (すべてのケースに NULL を指定す

ることはできません)。すべての *結果式* は互換データ・タイプを持っている必要があり、結果の属性は 99 ページの 『結果のデータ・タイプに関する規則』 に基づいて決まります。

CASE の持つ機能のサブセットを扱うように特化された 2 つのスカラー関数、NULLIF および COALESCE があります。次の表は、CASE またはこれらの関数を使用した同等の式を示しています。

表 21. 同等の CASE 式

CASE 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)
CASE WHEN e1 IS NOT NULL THEN e1 ELSE COALESCE(e2,...,eN) END	COALESCE(e1,e2,...,eN)

例

- 部門番号の先頭文字が組織上の部である場合には、CASE 式を使用して、それぞれの従業員が所属する部門の完全な名前をリストすることができます。

```
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
```

- 教育年数が EMPLOYEE 表で使用されており、教育のレベルを示します。 CASE 式を使用して、これらをグループ化し、教育のレベルを示します。

```
SELECT EMPNO, FIRSTNME, MIDINIT, LASTNAME,
  CASE
  WHEN EDLEVEL < 15 THEN 'SECONDARY'
  WHEN EDLEVEL < 19 THEN 'COLLEGE'
  ELSE 'POST GRADUATE'
  END
FROM EMPLOYEE
```

- CASE ステートメントを使用した別の興味ある例として、0 による割り算のエラーの保護があります。例えば、次のコードでは、歩合で収入の 25% 以上を稼ぎながら、歩合の全額を支払われていない従業員を見つけだすものです。

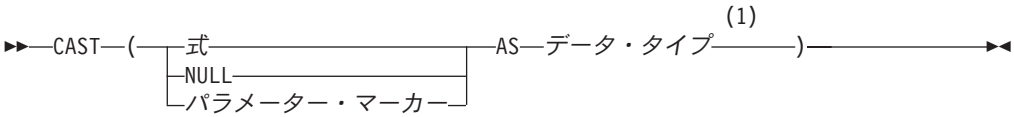
```
SELECT EMPNO, WORKDEPT, SALARY+COMM
FROM EMPLOYEE
WHERE (CASE WHEN SALARY=0 THEN NULL
            ELSE COMM/SALARY
       END) > 0.25
```

- 次の CASE 式は同等のもです。

```
SELECT LASTNAME,
 CASE
 WHEN LASTNAME = 'Haas' THEN 'President'
 ...

SELECT LASTNAME,
 CASE LASTNAME
 WHEN 'Haas' THEN 'President'
 ...
```

CAST の指定

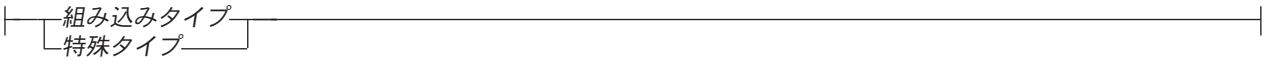


注:

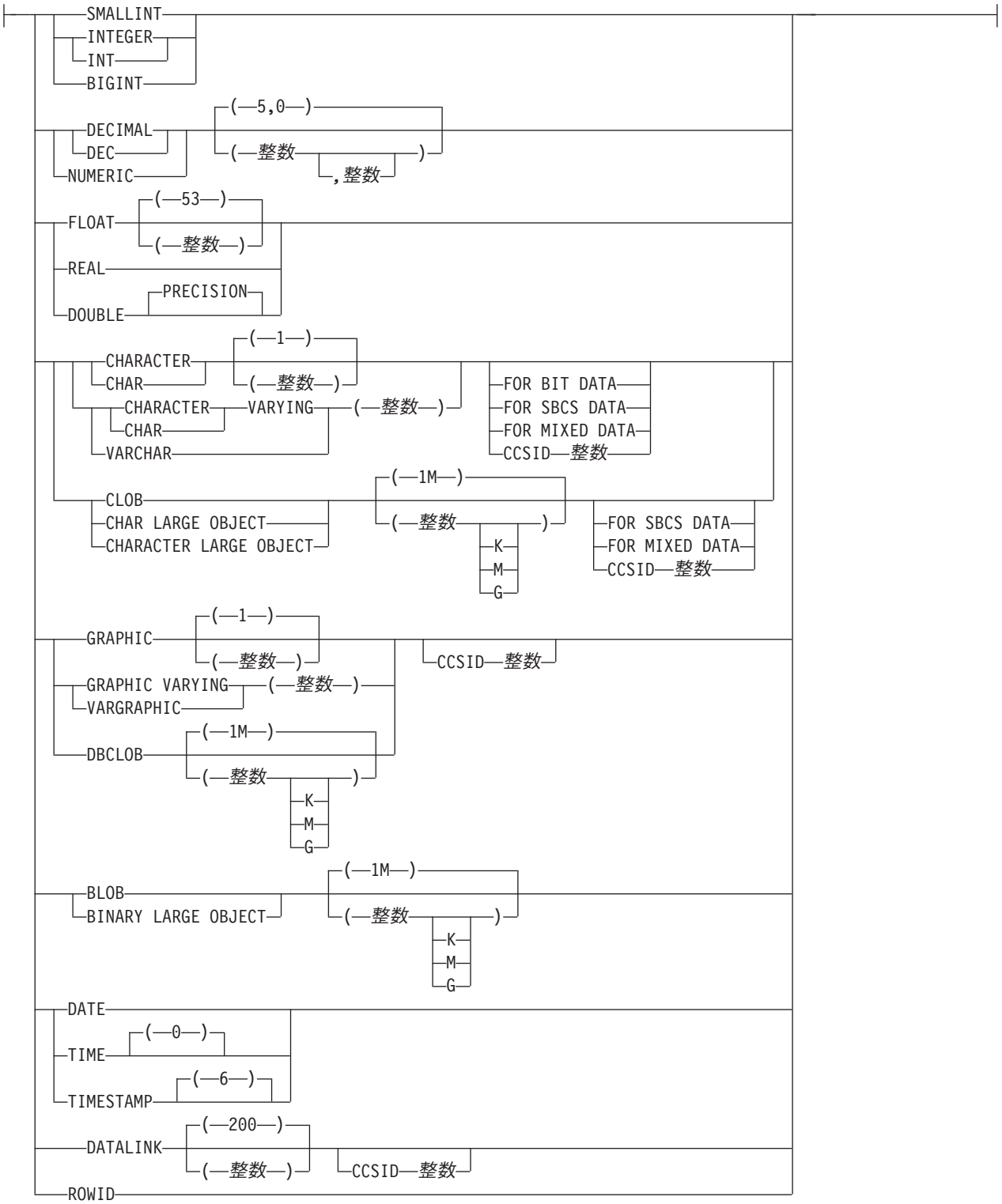
1 データ・タイプ名は修飾することができます。 詳細については、 47 ページの『命名規則』を参照してください。

式

データ・タイプ:



組み込みタイプ:



CAST の指定では、キャストのオペランド (第 1 オペランド) をデータ・タイプ で指定されたタイプにキャストして戻します。いずれかのオペランドのデータ・タイプが特殊タイプの場合は、ステートメントの権限 ID によって保持される特権には、特殊タイプの USAGE 権限が含まれている必要があります。

式 キャスト・オペランドが式の場合 (パラメーター・マーカーまたは NULL 以外の場合)、結果は指定したターゲット・データ・タイプに変換される引き数値です。

サポートされているキャストについては、83 ページの表 11 に示されています。そこでは、最初の列がキャスト・オペランドのデータ・タイプ (ソース・データ・タイプ) を、そして上段のデータ・タイプが CAST 指定のターゲット・データ・タイプを表しています。キャストがサポートされていない場合は、エラーが起こります。

文字またはグラフィック・ストリングを、長さが異なる文字またはグラフィック・ストリングにキャストすると、末尾ブランク以外の切り捨てが生じた場合には、警告が戻されます。

NULL
キャストのオペランドが NULL の場合は、結果は指定したデータ・タイプ を持ったヌル値になります。

パラメーター・マーカー
パラメーター・マーカー (疑問符として指定) は、一般には式であると考えられますが、特別な意味を持つのでこのケースは別に記しています。キャスト・オペランドがパラメーター・マーカー の場合、指定したデータ・タイプ は、置き換えが指定したデータ・タイプに割り当て可能であることの保証であると考えられます (列への割り当てと同じ規則を使用します)。そのようなパラメーター・マーカーは、タイプ・パラメーター・マーカー と考えられます。タイプ・パラメーター・マーカーは、選択リストの DESCRIBE または列の割り当てのために、他のタイプ値と同様に取り扱われることになります。

データ・タイプ
結果のデータ・タイプを指定します。データ・タイプが修飾されていない場合は、適切なデータ・タイプを見つけるために SQL パスを使用します。データ・タイプ の説明については、 542 ページの『CREATE TABLE』を参照してください。

長さ、精度、位取り、または CCSID 属性が指定されている場合は、指定された属性が使用されます。長さ、精度、または位取りが指定されていない場合は、デフォルト値が使用されます。例えば、CHAR のデフォルト値は長さが 1 であり、DECIMAL のデフォルト値は精度が 5 で位取りが 0 です。その他のデータ・タイプのデフォルト属性値については、 542 ページの『CREATE TABLE』を参照してください。(オペレーティング・システム間の移植性のために、浮動小数点データ・タイプを使用する場合は、FLOAT の代わりに REAL または DOUBLE を使用してください。)

CCSID 属性が指定されていない場合は、次のようになります。

- データ・タイプ が BLOB の場合、65535 の CCSID が使用されます。
- 式 が文字ストリングの場合、データ・タイプ は CHAR、VARCHAR、または CLOB で、式 の CCSID が使用されます。

- 式 がグラフィック・ストリングの場合、データ・タイプ は GRAPHIC、VARGRAPHIC、または DBCLOB で、式 の CCSID が使用されます。
- それ以外の場合は、そのデータ・タイプ のデフォルト CCSID が使用されます。

サポートされるデータ・タイプに関する制限は、指定されたキャストのオペランドに基づきます。

- キャストのオペランドが 式 の場合、キャストのオペランドのデータ・タイプに基づいてサポートされているターゲット・データ・タイプに関しては、83 ページの表 11 を参照してください。
- キャストのオペランドがキーワード NULL の場合は、ターゲット・データ・タイプはどのデータ・タイプでも構いません。
- キャストのオペランドがパラメーター・マーカーの場合は、ターゲット・データ・タイプはどのデータ・タイプでも構いません。データ・タイプが特殊タイプの場合、パラメーター・マーカーを使用するアプリケーションは、特殊タイプのソース・データ・タイプを使用することになります。

データ・タイプ間でサポートされるキャスト、およびデータ・タイプへキャストする際の規則についての詳細は、82 ページの『データ・タイプ間のキャスト』を参照してください。

例

- アプリケーションでは、EMPLOYEE 表の SALARY 列 (DECIMAL(9,2) として定義) の整数部分にのみ関与します。次の CAST 指定は、SALARY 列を INTEGER に変換します。

```
SELECT EMPNO, CAST(SALARY AS INTEGER)
  FROM EMPLOYEE
```

- 2 つの特殊タイプが存在するものとします。 T_AGE は、SMALLINT をソースとしており、PERSONNEL 表の AGE 列のデータ・タイプです。 R_YEAR は、INTEGER をソースとしており、同じ表の RETIRE_YEAR 列のデータ・タイプです。次の UPDATE ステートメントを用意しました。

```
UPDATE PERSONNEL SET RETIRE_YEAR = ?
  WHERE AGE = CAST( ? AS T_AGE )
```

述部

述部 は、ある所定の行またはグループについて真、偽、または未知という条件を指示します。以下の規則は、すべてのタイプの述部に適用されます。

- 1 つの述部に指定する値には、すべて互換性がなければなりません。
- 複数のオペランドを持つ述部のオペランドの CCSID 変換は、96 ページの『比較の際の変換規則』に従って行われます。
- データ・リンク値の使用は、NULL 述部に限定されています。

基本述部

注:

1 他の比較演算子もサポートされています。[28]

基本述部 では、2 つの値を比較します。述部のオペランドが SBCS データまたは混合データを含み、そのステートメントが実行される時点で有効なソート順序が *HEX でない場合は、そのオペランドの比較は、オペランドの重み付けされた値を使用して行われます。値の重み付けは、該当のソート順序に基づいています。

一方のオペランドの値がヌルの場合は、述部の結果は未知になります。 それ以外の場合、結果は真または偽のいずれかになります。

2 つの値 *x* および *y* があるとすれば、次のような関係が成り立ちます。

述部　左記の述部が真になる必要十分条件

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、または等しい

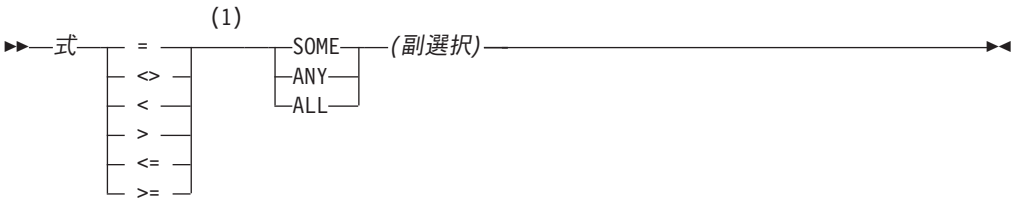
x*<= *y *x* は *y* より小さいか、または等しい

例

```
EMPNO = '528671'
PRTSTAFF <> :VAR1
SALARY + BONUS + COMM < 20000
SALARY > (SELECT AVG(SALARY) FROM EMPLOYEE)
```

28. 基本述部と比較述部では、次の形式の比較演算子もサポートされます。つまり、!=、!<、!>、¬=、¬<、および ¬> がサポートされます。これらのプロダクト特定の形式の比較演算子は、こうした演算子を使用している既存の SQL ステートメントをサポートすることだけが目的であり、新規の SQL ステートメントを作成するときに使用することはお勧めできません。キーボードによっては、NOT (¬) の記号の代わりに 16 進数値を使用しなければなりません。この 16 進数値は、使用するキーボードによって異なります。NOT 記号 (¬) または特定の国でその場所に使う必要がある文字は、あるデータベース・サーバーから別のデータベース・サーバーに渡されるステートメントで構文解析エラーを引き起こすことがあります。問題が起きるのは、ステートメントがある種の組み合わせのソース CCSID とターゲット CCSID を使って文字変換を行う場合です。この問題を避けるために、NOT 記号を含む演算子は、同等の演算子で置き換えてください。例えば、'¬=' は '<>' で、'¬>' は '<=' で、そして '¬<' は '>=' で置き換えます。

多値比較述部



注:

1 他の比較演算子もサポートされています。[28]

多値比較述部 は、ある値と値の集合を比較します。

副選択には、単一の結果列を指定しなければなりません。副選択によって、ヌルか否かに関係なく値がいくつか戻されることがあります。述部のオペランドが SBCS データまたは混合データを含み、そのステートメントが実行される時点で有効なソート順序が *HEX でない場合は、オペランドの重み付けされた値を使用して比較が行われます。値の重み付けは、該当のソート順序に基づいています。

ALL を指定すると、述部の結果は次のようになります。

- 副選択の結果が空の場合、または指定された関係が副選択によって戻されたすべての値について真である場合は、結果は真。
- 副選択によって戻された値の少なくとも 1 つについて指定された関係が偽である場合、結果は偽。
- 副選択によって戻された値の中に指定された関係が偽でないものがあり、しかも、少なくとも 1 つの比較がヌル値により不明になった場合、結果は不明。

SOME または ANY が指定された場合、述部の結果は次のようになります。

- 副選択によって戻された値の少なくとも 1 つについて、指定された関係が真である場合、結果は真。
- 副選択の結果が空の場合、または副選択によって戻されたすべての値について、指定された関係が偽である場合は、結果は偽。
- 副選択によって戻された値の中に指定された関係が不明のものがあり、しかも少なくとも 1 つの比較がヌル値により不明であった場合、結果は不明。

例

下記の例では、次のような表を使用しています。

表 22. 表の内容

TBLA	COLA	TBLB	COLB
	1		2
	2		3
	3		
	4		
	ヌル		

- 次の選択ステートメントの結果は 2,3 となります。副選択は (2,3) を戻します。行 2 および 3 の COLA が、これらの値の少なくとも 1 つと等しくなります。

```
SELECT * FROM TBLA WHERE COLA = ANY(SELECT COLB FROM TBLB)
```

- 次の選択ステートメントの結果は 3,4 となります。副選択は (2,3) を戻します。行 3 および 4 の COLA が、これらの値の少なくとも 1 つよりも大きくなります。

```
SELECT * FROM TBLA WHERE COLA > ANY(SELECT COLB FROM TBLB)
```

- 次の選択ステートメントの結果は 4 となります。副選択は (2,3) を戻します。これらの両方の値よりも大きいのは、行 4 の COLA だけです。

```
SELECT * FROM TBLA WHERE COLA > ALL(SELECT COLB FROM TBLB)
```

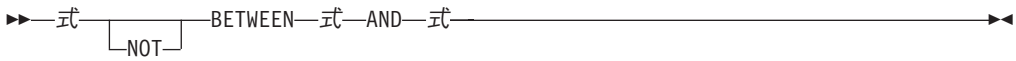
- 次の選択ステートメントの結果は 1、2、3、4、およびヌルとなります。副選択の結果は空です。したがって、述部は TBLA のすべての行について真となります。

```
SELECT * FROM TBLA WHERE COLA > ALL(SELECT COLB FROM TBLB WHERE COLB<0)
```

- 次の選択ステートメントの結果は空集合となります。副選択の結果は空です。したがって、述部は TBLA のすべての行について偽となります。

```
SELECT * FROM TBLA WHERE COLA > ANY(SELECT COLB FROM TBLB WHERE COLB<0)
```

BETWEEN 述部



BETWEEN 述部は、ある値を値の範囲と比較します。ステートメントの実行時点で *HEX 以外のソート順序が有効で、しかも BETWEEN 述部が SBCS データまたは混合データを含む場合は、その値の代わりにストリングの重み付けされた値が比較されます。重み付けされた値は、該当のソート順序に基づいています。

次の BETWEEN 述部は、

```
value1 BETWEEN value2 AND value3
```

論理的に、次の検索条件と等価です。

```
value1 >= value2 AND value1 <= value3
```

次の BETWEEN 述部は、

```
value1 NOT BETWEEN value2 AND value3
```

次の検索条件と等価です。

```
NOT(value1 BETWEEN value2 AND value3);that is,
value1 < value2 OR value1 > value3.
```

BETWEEN 述部のオペランドがそれぞれ異なる CCSID を持つストリングである場合、オペランドは上記の論理的に等価な検索条件が指定されているのと同様に変換されます。

日付/時刻の値と日付/時刻の値のストリング表現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・タイプに変換されます。

例

```
EMPLOYEE.SALARY BETWEEN 20000 AND 40000

SALARY NOT BETWEEN 20000 + :HV1 AND 40000
```

EXISTS 述部

EXISTS 述部は、特定の行が存在するかどうかを検査します。副選択には、列をいくつでも指定できます。

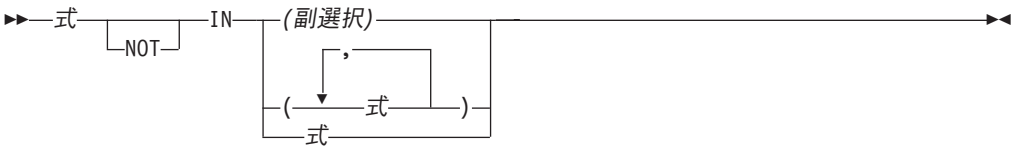
- 結果が真になるのは、副選択によって指定された行の数がゼロでなかった場合だけです。
- 結果が偽になるのは、副選択によって指定された行の数がゼロだった場合だけです。
- 結果が未知になることはありません。

副選択によって戻された値は無視されます。

例

```
EXISTS (SELECT * FROM EMPLOYEE WHERE SALARY > 60000)
```

IN 述部



IN 述部は、ある値と値の集合を比較します。ステートメントの実行時点で *HEX 以外のソート順序が有効で、しかも IN 述部が SBCS データまたは混合データを含む場合には、ストリングの重み付けされた値が、実際の値の代わりに比較されます。値の重み付けは、該当のソート順序に基づいています。

副選択の形式を使用する場合、副選択は単一の結果列を識別するものでなければなりません。また、副選択によっていくつかの値 (ヌルまたはヌル以外) が戻されることがあります。

次の形式の IN 述部は、

式 **IN** (*副選択*)

以下の形式の多値比較述部と同等です。

式 = **ANY** (*副選択*)

次の形式の IN 述部は、

式 **NOT IN** (*副選択*)

以下の形式の多値比較述部と同等です。

式 <> **ALL** (*副選択*)

次の形式の IN 述部は、

式 **IN** 式

以下の形式の基本述部と同等です。

式 = 式

次の形式の IN 述部は、

式 **IN** (*値 1*、*値 2*、...、*値 N*)

以下のものと論理的に等価です。

式 **IN** (**SELECT** * **FROM** R)

1 つの行を持つ表 T を想定します。R は、次の全選択によって形成された一時的表です。

```
SELECT value1 FROM T
  UNION
SELECT value2 FROM T
  UNION
    .
    .
    .
  UNION
SELECT valueN FROM T
```

各ホスト変数は、ホスト構造やホスト変数の宣言規則に従って記述されている構造や変数を識別していなければなりません。

IN 述部のオペランドが異なるデータ・タイプまたは異なる属性をもつ場合は、IN 述部の演算のデータ・タイプの決定に使用される規則は、UNION および UNION ALL の場合に使用される規則です。説明については、99 ページの 『結果のデータ・タイプに関する規則』を参照してください。

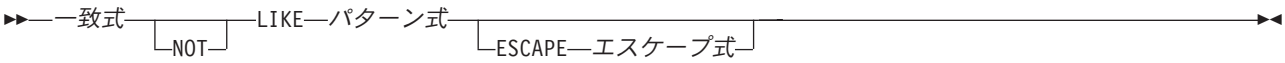
IN 述部のオペランドが異なる CCSID を持つストリングである場合に、変換するオペランドの決定に使用される規則は、ストリングの結合の演算で使用される規則です。説明については、103 ページの 『ストリングを結合する演算に適用される変換規則』を参照してください。

例

```
DEPTNO IN ('D01', 'B01', 'C01')
```

```
EMPNO IN(SELECT EMPNO FROM EMPLOYEE WHERE WORKDEPT = 'E11')
```

LIKE 述部



LIKE 述部は、特定のパターンを持つストリングを検索します。検索するパターンはストリングで指定します。下線およびパーセント記号は、パターンを指定するストリングの中で特別な意味を持ちます。パターン内の後書きブランクは、パターンの一部です。

いずれかの引き数の値がヌルの場合、LIKE 述部の結果は未知になります。

一致式、パターン式、および エスケープ式 は、ストリングを識別しなければなりません。一致式、パターン式、および エスケープ式 の値は、すべてが 2 進ストリングであるか、どれも 2 進ストリングでないか、のどちらかでなければなりません。 この 3 つの引き数には、文字ストリングとグラフィック・ストリングを混合して含めることができます。

どの式も特殊タイプを生成することはできませんが、特殊タイプをソース・タイプにキャストする関数を使用することは可能です。

ステートメントの実行時点で *HEX 以外のソート順序が有効で、しかも LIKE 述部が SBCS データまたは混合データを含む場合は、そのストリングの重み付けされた値が実際の値の代わりに比較されます。値の重み付けは、該当のソート順序に基づいています。

以下の説明における**文字**、**パーセント記号**、および**下線**という用語は、1 バイト文字を指しています。 グラフィック・ストリングの場合は、この用語は 2 バイトまたは UCS-2 文字を指しています。2 進ストリングの場合、この用語は、これらの 1 バイト文字のコード・ポイントを指しています。

一致式
所定の文字パターンに適合するかどうかを調べるストリングを示す式。

LIKE パターン式
突き合わせに使用するストリングを示す式。

単純なパターン記述

このパターンは、一致式の値の適合基準を指定するのに使用されます。ここで、

- 下線記号 (_) は、任意の 1 文字を表します。
- パーセント記号 (%) は、ゼロまたは 1 つ以上の文字から構成されるストリングを表します。
- 上記以外の文字は、すべてその文字自身を表します。

パターン式 に下線またはパーセント記号を含める必要がある場合は、エスケープ式 を使用して、パターン内の下線またはパーセント記号の前に置く 1 文字を指定します。

厳密なパターン記述

この厳密なパターン記述は、エスケープ式 の使用を無視します (後述)。

m は 一致式 の値を表し、*p* は パターン式 の値を表すものとしましょう。 ストリング *p* は、最小数の一連のサブストリング指定子として解釈されるので、*p* の各文字は、厳密に 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、または下線とパーセント記号以外の空でない一連の任意の文字を指します。

m または *p* がヌル値の場合は、述部の結果は不明です。それ以外の場合は、述部の結果は真か偽のいずれかになります。*m* と *p* の両方が空ストリングの場合、あるいは以下のようなサブストリングに *m* を区分化できる場合、結果は真になります。

- *m* のサブストリングがゼロ個または 1 個以上の連続する一連の文字である場合に、*m* の各文字が厳密に 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に、*m* の *n* 番目のサブストリングが任意の 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に、*m* の *n* 番目のサブストリングがゼロ個または 1 個以上の任意の一連の文字である。
- *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に、*m* の *n* 番目のサブストリングが、対応するサブストリング指定子と等しく、かつ対応するサブストリング指定子と同じ長さである。
- *m* のサブストリングの数が、サブストリング指定子の数と同じである。

したがって、*y* が空のストリングで、*m* が空のストリングでない場合は、結果が偽になります。同様に、*m* が空のストリングで、*p* が空のストリングでない場合は、結果が偽になります。

m NOT LIKE *p* という述部は、NOT(*m* LIKE *p*) という検索条件と同等です。

必要な場合、一致式、パターン式、およびエスケープ式 の CCSID は、一致式 とパターン式 の間の互換性のある CCSID に変換されます。

混合データ

式が混合データである場合は、その式には 2 バイト文字が含まれていることがあり、パターンには SBCS 文字と DBCS 文字の両方を含めることができます。この場合、*p* の中の特殊文字は以下のように解釈されます。

- SBCS の下線は、1 つの SBCS 文字を示します。
- DBCS の下線は、1 つの DBCS 文字を示します。
- パーセント記号 (SBCS または DBCS) は、任意のタイプ (SBCS または DBCS) の任意の個数の文字を示します。
- 一致式 とパターン式 の余分なシフト文字は無視されます。

UCS-2 データ

式が UCS-2 グラフィック・データの場合は、パターンには、UCS-2 の下線およびパーセント記号のサポートされているコード・ポイントのいずれか一方または両方を含めることができます。UCS-2 下線のサポートされているコード・ポ

イントは、X'005F' と X'FF3F' です。UCS-2 パーセント記号のサポートされているコード・ポイントは、X'0025' と X'FF05' です。

パラメーター・マーカー

LIKE 述部で指定したパターンがパラメーター・マーカーであり、固定長文字のホスト変数を使用してそのパラメーター・マーカーを置き換えるときは、そのホスト変数の値に正しい長さを指定します。正しい長さで指定しないと、選択によって意図した結果が戻されないことになります。

例えば、ホスト変数が CHAR(10) として定義されている場合、そのホスト変数に WYSE% という値を割り当てると、余白にはブランクが埋め込まれます。使用されるパターンは、次のとおりです。

```
'WYSE%     '
```

このパターンは、WYSE で始まり、5 つのブランク・スペースで終わるすべての値の検索をデータベース・マネージャーに要求します。'WYSE' で始まる値だけを検索したい場合は、ホスト変数に 'WSYE%%%%%%' の値を割り当てる必要があります。

ESCAPE エスケープ式

パターン式の下線 (_) 文字とパーセント (%) 文字の特殊な意味を変更するのに使用する文字を指定する式。これにより、LIKE 述部を使用して、実際にパーセント文字や下線文字を含んでいる値を突き合わせることが可能になります。
ESCAPE 文節と エスケープ式 の使用には、次の規則が適用されます。

- エスケープ式 は、長さが 1 のストリングでなければならない。[29]
- パターン式 には、エスケープ文字の後にエスケープ文字、パーセント記号、または下線が続く場合を除き、エスケープ文字を含めてはならない。例えば、'+' がエスケープ文字の場合、パターン式 に '++' 、'+_'、または '+%' 以外の '+' が入っていると、エラーになります。
- エスケープ式 は、パラメーター・マーカーでもかまいません。

次の例は、連続したエスケープ文字 (この場合は、正符号 (+)) の効果を示しています。

パターン・ストリング	実際のパターン
+%	パーセント記号
++%	正符号の後にゼロ個以上の任意の文字が続く
+++%	正符号の後にパーセント記号が続く

例

例 1: 表 PROJECT の列 PROJNAME のいずれかに、'SYSTEMS' というストリングが入っていないかどうかを検索します。

```
SELECT PROJNAME
  FROM PROJECT
  WHERE PROJECT.PROJNAME LIKE '%SYSTEMS%'
```

29. ヌルで終了する場合は、長さが 2 の C 文字ストリング変数を指定できます。

例 2: 表 EMPLOYEE の列 FIRSTNME に、先頭の文字が 'J' で、長さがちょうど 2 文字のストリングがないかどうかを検索します。

```
SELECT FIRSTNME
     FROM EMPLOYEE
     WHERE EMPLOYEE.FIRSTNME LIKE 'J_'
```

例 3: 表 EMPLOYEE の列 FIRSTNME に、任意の長さで、先頭の文字が 'J' のストリングがないかどうかを検索します。

```
SELECT FIRSTNME
     FROM EMPLOYEE
     WHERE EMPLOYEE.FIRSTNME LIKE 'J%'
```

例 4: この例で、

```
SELECT *
     FROM TABLEY
     WHERE C1 LIKE 'AAAA+%BBB%' ESCAPE '+'
```

'+' はエスケープ文字であり、この検索が、'AAAA%BBB' で始まるストリングの検索であることを示しています。'+%' は、パターンの '%' の 1 つのオカレンスとして解釈されます。

例 5: ソース・データ・タイプが CHAR(5) の ZIP_TYPE という名前の特殊タイプが存在し、ある表 TABLEY にデータ・タイプが ZIP_TYPE の ADDRZIP 列が存在するものと想定します。次のステートメントは、ZIP コード (ADDRZIP) が '9555' で始まっている行を選択します。

```
SELECT *
  FROM TABLEY
  WHERE CHAR(ADDRZIP) LIKE '9555%'
```

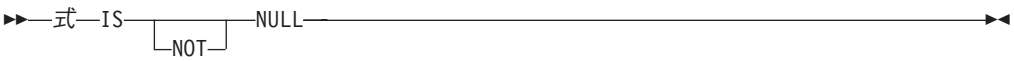
例 6: サンプル表 EMP_RESUME の RESUME 列は、CLOB として定義されています。ホスト変数 LASTNAME の値が 'JONES' である場合、次のステートメントは、列内にストリング JONES が見つかった場合、RESUME 列を選択します。

```
SELECT RESUME
  FROM EMP_RESUME
  WHERE RESUME LIKE '%'||LASTNAME||'%'
```

例 7: 次の EBCDIC における例で、COL1 は混合データであると想定しています。この表は、2 番目の欄の COL1 の値を使用して、最初の欄の述部を実行した場合の結果を示しています。

述部	COL1 値	結果
WHERE COL1 LIKE 'aaa SOAB%CSI'	'aaa SOABDZCSI'	真
WHERE COL1 LIKE 'aaa SOABSI%SOCSI'	'aaa SOABSI dzx SOCSI'	真
WHERE COL1 LIKE 'a% SOCSI'	'a SOCSI'	真
	'ax SOCSI'	真
	'ab SODESI fg SOCSI'	真
WHERE COL1 LIKE 'a _SOCSI'	'a% SOCSI'	真
	'a SOXCSI'	偽
WHERE COL1 LIKE 'a SO__CSI'	'a SOXCSI'	真
	'ax SOCSI'	偽
WHERE COL1 LIKE 'SOSI'	空ストリング	真
WHERE COL1 LIKE 'ab SOCSI_'	'ab SOCSI d'	真
	'ab SOSISOCSI d'	真

NULL 述部



NULL 述部は、ヌル値かどうかを検査します。

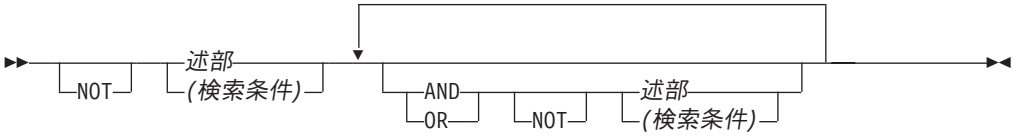
NULL 述部の結果が不明になることはありません。式の値がヌルであれば、結果は真になります。値がヌルでなければ、結果は偽になります。NOT を指定すると、結果が逆になります。

例

```
EMPLOYEE.PHONE IS NULL

SALARY IS NOT NULL
```

検索条件



検索条件 は、ある所定の行またはグループについて、真、偽、または不明の条件を示します。

検索条件の結果は、指定した*論理演算子* (AND、OR、NOT) を、指定したそれぞれの述部の結果に適用することによって求められます。論理演算子を指定しないと、指定した述部の結果が検索条件の結果となります。

AND および OR の定義を以下の表に示します。表の中の P および Q は、任意の述部を示します。

表 23. 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(真) は偽、NOT(偽) は真、NOT(不明) は不明になります。

括弧内の検索条件が最初に評価されます。評価の順序を括弧で指定しなかった場合は、NOT が処理されてから AND が処理され、AND が処理されてから OR が処理されます。同じ優先順位レベルにある演算子が評価されるときの順序は、検索条件が最適化されるため、決まっていません。

例

以下の例では、2 行目の番号が演算子の評価される順序を示しています。

例 1

```
  MAJPROJ = 'MA2100' AND DEPTNO = 'D11'  OR  DEPTNO = 'B03' OR DEPTNO = 'E11'
                     1                   2 or 3            2 or 3
```

例 2

```
MAJPROJ = 'MA2100'  AND  (DEPTNO = 'D11'  OR  DEPTNO = 'B03')  OR  DEPTNO = 'E11'
                    2                      1                    3
```

第 3 章 組み込み関数

組み込み関数 は、DB2 UDB for iSeries に提供されている関数です。組み込み関数は、関数名の後に、括弧で囲んだ 1 つまたは複数のオペランドを指定することによって示されます。関数のオペランドを*引き数* と言います。各引き数は式 の形で指定します。関数の結果は、引き数に対してその関数の演算を適用することによって求められる単一の値です。

組み込み関数 はスキーマ QSYS2 の一部です。*組み込み関数* は、スキーマ名付きまたはスキーマ名なしのどちらの形でも呼び出すことができます。関数名がスキーマ名で修飾されているかどうかに関係なく、データベース・マネージャーは、関数解決を使用してどの関数を使用するかを決定します。関数、および関数解決のプロセスについての詳細は、131 ページの『関数解決』を参照してください。

組み込み関数 は、*列関数* とスカラー*関数* に分類されます。列関数の引き数は、値の集合です。スカラー関数の引き数は、単一の値です。

SQL の構文で*関数* という用語を使用するのは、式の定義の中だけです。したがって、関数を使用できるのは、式を使用できる個所だけに限られます。 列関数を使用する際に適用されるその他の制約事項については、この後の項および 339 ページの『第 4 章 照会』で説明しています。

以下の表は、*組み込み関数* をタイプ別に分類して示しています。

表 24. 列関数

175 ページの『AVG』	数値の集合の平均を戻します。
176 ページの『COUNT』	行または値の集合の中にある行の数または値の数を戻します。
177 ページの『COUNT_BIG』	行または値の集合の中にある行の数または値の数を戻します。この関数は COUNT 関数に類似していますが、COUNT 関数とは異なり、結果の値は整数の最大値より大きい値をとることができます。
179 ページの『MAX』	あるグループの中の値の集合の最大値を戻します。
180 ページの『MIN』	あるグループの中の値の集合の最小値を戻します。
181 ページの『STDDEV または STDDEV_POP』	数値の集合のバイアス標準偏差 (/n) を戻します。
182 ページの『SUM』	数値の集合の合計を戻します。
183 ページの『VAR_POP または VARIANCE または VAR』	数値の集合のバイアス偏差 (/n) を戻します。

表 25. キャスト・スカラー関数

192 ページの『BIGINT』	数値の 64 ビット整数表現を戻します。
193 ページの『BLOB』	任意のタイプのストリングの BLOB 表現を戻します。
196 ページの『CHAR』	値の CHARACTER 表現を戻します。
202 ページの『CLOB』	値の CLOB 表現を戻します。
213 ページの『DATE』	値から DATE を戻します。

表 25. キャスト・スカラー関数 (続き)

221 ページの『DBCLOB』	ストリング式の DBCLOB 表現を戻します。
224 ページの『DECIMAL または DEC』	数値の DECIMAL 表現を戻します。
238 ページの『DOUBLE_PRECISION または DOUBLE』	数値の DOUBLE_PRECISION 表現を戻します。
240 ページの『FLOAT』	数値の FLOAT 表現を戻します。
242 ページの『GRAPHIC』	ストリング式の GRAPHIC 表現を戻します。
254 ページの『INTEGER または INT』	数値の INTEGER 表現を戻します。
292 ページの『REAL』	数値の REAL 表現を戻します。
295 ページの『ROWID』	値から行 ID を戻します。
302 ページの『SMALLINT』	数値の SMALLINT 表現を戻します。
312 ページの『TIME』	値から TIME を戻します。
313 ページの『TIMESTAMP』	1 つまたは 1 対の値から TIMESTAMP を戻します。
326 ページの『VARCHAR』	値の VARCHAR 表現を戻します。
330 ページの『VARGRAPHIC』	ストリング式の VARGRAPHIC 表現を戻します。
337 ページの『ZONED』	数値のゾーン 10 進数表現を戻します。

表 26. データ・リンク・スカラー関数

229 ページの『DLCOMMENT』	データ・リンク値からコメント値を戻します。
230 ページの『DLLINKTYPE』	データ・リンク値からリンク・タイプ値を戻します。
231 ページの『DLURLCOMPLETE』	リンク・タイプ URL のデータ・リンク値から完全な URL 値を戻します。
232 ページの『DLURLPATH』	リンク・タイプ URL のデータ・リンク値から、あるサーバー内のファイルにアクセスするのに必要なパスとファイル名を戻します。該当する場合、戻される値にはファイル・アクセス・トークンが含まれます。
233 ページの『DLURLPATHONLY』	リンク・タイプ URL のデータ・リンク値から、あるサーバー内のファイルにアクセスするのに必要なパスとファイル名を戻します。戻される値には、ファイル・アクセス・トークンは含まれていません。
234 ページの『DLURLSCHEME』	リンク・タイプ URL のデータ・リンク値からそのスキーマを戻します。
235 ページの『DLURLSERVER』	リンク・タイプ URL のデータ・リンク値から、ファイル・サーバーを戻します。
236 ページの『DLVALUE』	データ・リンク値を戻します。

表 27. 日付時刻スカラー関数

211 ページの『CURDATE』	時点の刻時機構の読み取りに基づく日付を戻します。
212 ページの『CURTIME』	時点の刻時機構の読み取りに基づく時刻を戻します。
220 ページの『DAYS』	値の日付の部分を戻します。
216 ページの『DAYOFMONTH』	値の日付の部分を戻します。
217 ページの『DAYOFWEEK』	曜日を表す整数 (1 は日曜日を表し、7 は土曜日を表す) を戻します。
218 ページの『DAYOFWEEK_ISO』	曜日を表す整数 (1 は月曜日を表し、7 は日曜日を表す) を戻します。

表 27. 日付時刻スカラー関数 (続き)

219 ページの『DAYOFYEAR』	その年の日付を表す整数を戻します。
220 ページの『DAYS』	日付の整数表現を戻します。
247 ページの『HOUR』	値の時間の部分を戻します。
256 ページの『JULIAN_DAY』	紀元前 4712 年 1 月 1 日から引き数で指定された日付までの日数を表す整数値を戻します。
272 ページの『MICROSECOND』	値のマイクロ秒の部分を戻します。
273 ページの『MIDNIGHT_SECONDS』	真夜中から指定の時刻値までの間の秒数を表す整数値を戻します。
276 ページの『MINUTE』	値の分の部分を戻します。
279 ページの『MONTH』	値の月の部分を戻します。
282 ページの『NOW』	時点の刻時機構の読み取りに基づくタイム・スタンプを戻します。
289 ページの『QUARTER』	日付が存在する四半期を表す整数を戻します。
298 ページの『SECOND』	値の秒の部分を戻します。
315 ページの『TIMESTAMPDIFF』	2 つのタイム・スタンプの差に基づいて、間隔の見積数を戻します。
333 ページの『WEEK』	年の週を表す整数を戻します。週は日曜日から始まります。
334 ページの『WEEK_ISO』	年の週を表す整数を戻します。週は月曜日から始まります。
336 ページの『YEAR』	値の年の部分を戻します。

表 28. 区分スカラー関数

245 ページの『HASH』	値の集合の区画番号を戻します。
280 ページの『NODENAME』	行が置かれているリレーショナル・データベースの名前を戻します。
281 ページの『NODENUMBER』	行のノード番号を戻します。
284 ページの『PARTITION』	行の区画番号を戻します。

表 29. その他のスカラー関数

206 ページの『COALESCE』	ヌルでない最初の引き数を戻します。
246 ページの『HEX』	値の 16 進数表現を戻します。
248 ページの『IDENTITY_VAL_LOCAL』	識別列の最新割り当て値を戻します。
253 ページの『IFNULL』	ヌルでない最初の引き数を戻します。
261 ページの『LENGTH』	値の長さを戻します。
270 ページの『MAX』	値の集合の最大値を戻します。
274 ページの『MIN』	値の集合の最小値を戻します。
283 ページの『NULLIF』	引き数が等しい場合はヌルを戻します。等しくない場合は、最初の引き数の値を戻します。
296 ページの『RRN』	行の相対レコード番号を戻します。
325 ページの『VALUE』	ヌルでない最初の引き数を戻します。

組み込み関数

表 30. 数値スカラー関数

185 ページの『ABS』	数値の絶対値を戻します。
186 ページの『ACOS』	数値のアークコサイン (逆余弦) をラジアンで戻します。
187 ページの『ANTILOG』	数値の逆対数 (底 10) を戻します。
188 ページの『ASIN』	数値のアークサイン (逆正弦) をラジアンで戻します。
189 ページの『ATAN』	数値のアークタンジェント (逆正接) をラジアンで戻します。
190 ページの『ATANH』	数値の双曲線アークタンジェント (双曲線逆正接) をラジアンで戻します。
191 ページの『ATAN2』	x 座標と y 座標の逆正接を、ラジアンで表された角度として戻します。
195 ページの『CEILING』	数値式より大きいか等しい最小の整数値を戻します。
208 ページの『COS』	数値のコサイン (余弦) を戻します。
209 ページの『COSH』	数値の双曲線コサイン (双曲線余弦) を戻します。
210 ページの『COT』	数値のコタンジェント (余接) を戻します。
226 ページの『DEGREES』	角度の度数を戻します。
228 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。
239 ページの『EXP』	自然対数の底 (e) を引き数の指定だけ累乗した値を戻します。
241 ページの『FLOOR』	数値式に等しいか数値式より小さい最大の整数値を戻します。
263 ページの『LN』	数値の自然対数を戻します。
266 ページの『LOG10』	数値の共通対数 (底 10) を戻します。
277 ページの『MOD』	最初の引き数を 2 番目の引き数で割って、その剰余を戻します。
285 ページの『PI』	PI の値を戻します。
288 ページの『POWER』	最初の引き数を 2 番目の引き数だけ累乗した結果を戻します。
290 ページの『RADIANS』	度で表された引き数に対してラジアン数を戻します。
291 ページの『RAND』	乱数を戻します。
293 ページの『ROUND』	指定の小数部の桁数まで丸められた数値を戻します。
299 ページの『SIGN』	式の符号の標識を戻します。
300 ページの『SIN』	数値のサイン (正弦) を戻します。
301 ページの『SINH』	数値の双曲線サイン (双曲線正弦) を戻します。
305 ページの『SQRT』	数値の平方根を戻します。
310 ページの『TAN』	数値のタンジェント (正接) を戻します。
311 ページの『TANH』	数値の双曲線タンジェント (双曲線正接) を戻します。
321 ページの『TRUNCATE または TRUNC』	指定の小数部の桁数で切り捨てられた数値を戻します。

表 31. ストリング・スカラー関数

201 ページの『CHARACTER_LENGTH』	ストリング式の長さを戻します。
207 ページの『CONCAT』	2 つのストリングを連結します。
258 ページの『LCASE』	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。

表 31. ストリング・スカラー関数 (続き)

227 ページの『DIFFERENCE』	2 つのストリングの音の相違を表す値を戻します。
257 ページの『LAND』	引き数である 2 つのストリングの論理 'AND' であるストリングを戻します。
258 ページの『LCASE』	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。
259 ページの『LEFT』	ストリングから左端の文字を戻します。
264 ページの『LNOT』	引き数ストリングの論理否定 (論理 NOT) であるストリングを戻します。
265 ページの『LOCATE』	別のストリング内のあるストリングの開始位置を戻します。
267 ページの『LOR』	引き数である 2 つのストリングの論理 OR であるストリングを戻します。
268 ページの『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。
269 ページの『LTRIM』	ストリング式の前部からブランクまたは 16 進数ゼロを除去します。
286 ページの『POSITION または POSSTR』	別のストリング内のあるストリングの開始位置を戻します。
306 ページの『STRIP』	ストリング式の後部または前部から、ブランクまたは指定した文字を除去します。
297 ページの『RTRIM』	ストリング式の後部からブランクまたは 16 進数ゼロを除去します。
303 ページの『SOUNDEX』	引き数内のワードの音を表す文字コードを戻します。
304 ページの『SPACE』	引き数で指定されたブランク数からなる文字ストリングを戻します。
307 ページの『SUBSTRING または SUBSTR』	ストリングのサブストリングを戻します。
317 ページの『TRANSLATE』	ストリングの中の 1 文字または複数の文字を変換します。
319 ページの『TRIM』	ストリング式の後部または前部から、ブランクまたは指定した文字を除去します。
323 ページの『UCASE』	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。
323 ページの『UCASE』	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。
324 ページの『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。
335 ページの『XOR』	引き数である 2 つのストリングの論理 XOR であるストリングを戻します。

列関数

以下の説明は、COUNT(*) および COUNT_BIG(*) を除くすべての列関数に適用されます。

- 列関数の引き数は、1 つの式 から得られた値の集合です。式 には列を含めることはできますが、他の列関数を含めることはできません。この集合の有効範囲は、第 4 章『照会』で説明しているグループまたは中間結果表です。
- 照会で GROUP BY 文節が指定され、FROM、WHERE、GROUP BY、および HAVING 文節の中間結果が空集合である場合は、列関数は適用されません。照会の結果は空集合で、SQLCODE は +100 にセットされ、SQLSTATE は '02000' にセットされます。
- 照会で GROUP BY 文節の指定がなく、FROM、WHERE、および HAVING 文節の中間表が空集合の場合は、列関数はその空集合に適用されます。
- 例えば、次の SELECT ステートメントの結果は、部門 D01 の従業員の JOB の固有の値の数です。

```
SELECT COUNT(DISTINCT JOB)
  FROM EMPLOYEE
  WHERE WORKDEPT = 'D01'
```

- キーワード DISTINCT は、この関数の引き数ではなく、この関数の適用に先立って決められる演算の仕様です。DISTINCT が指定される場合は、重複する値は除かれます。ALL が暗黙、または明示的に指定されている場合には、重複する値は除去されません。
- WHERE 文節の中で列関数を使用できるのは、その文節が HAVING 文節の副照会の一部であり、式の中で指定する列名がグループに対する相関参照である場合だけです。式に複数の列名が含まれている場合は、それらの列名は同じグループに対する相関参照でなければなりません。

AVG

AVG 関数は、数値の集合の平均値を戻します。

引き数値は組み込み数値データ・タイプのいずれかでなければならず、また合計は結果のデータ・タイプの範囲内でなければなりません。

結果のデータ・タイプは、原則として引き数の値のデータ・タイプと同じになります。ただし、以下の場合は、結果のデータ・タイプが引き数の値のデータ・タイプとは異なるものになります。

- 引き数値が単精度の浮動小数点数である場合は、結果は倍精度の浮動小数点数になる。
- 引き数値が短整数である場合は、結果は長整数になる。
- 引き数が、10 進数または位取りがゼロ以外の 2 進数で、精度が *p*、位取りが *s* である場合は、結果は、精度が 31 で位取りが 31-*p*+*s* の 10 進数になります。

この関数は、引き数の値からヌル値を除いた値の集合に対して適用されます。DISTINCT を使用すると、重複する値は除かれます。

結果が、ヌルになることもあります。値の集合が空である場合は、結果はヌル値です。それ以外の場合は、結果は集合の値の平均値です。

値が集計される順序は未定義ですが、中間結果はすべて結果のデータ・タイプの範囲内になければなりません。

結果のデータ・タイプが整数である場合、平均値の小数部分は失われます。

例

- 表 PROJECT を使用して、部門 (DEPTNO) 'D11' のプロジェクトの平均要員レベル (列 PRSTAFF の平均値) を、ホスト変数 AVERAGE (DECIMAL(5,2)) にセットします。

```
SELECT AVG(PRSTAFF)
  INTO :AVERAGE
  FROM PROJECT
  WHERE DEPTNO = 'D11'
```

  上記の結果、AVERAGE は 4.25 (つまり 17/4) にセットされます。

- 表 PROJECT を使用して、ホスト変数 ANY_CALC に、部門 (DEPTNO) 'D11' のそれぞれ固有の要員水準の値 (PRSTAFF) の平均値をセットします。

```
SELECT AVG(DISTINCT PRSTAFF)
  INTO :ANY_CALC
  FROM PROJECT
  WHERE DEPTNO = 'D11'
```

  上記の結果、ANY_CALC は 4.66 (つまり、14/3) に設定されます。

COUNT

COUNT 関数は、行または値の集合の中にある行の数または値の数を戻します。

この関数の結果は長整数であり、結果の値は長整数の値の範囲内になければなりません。結果がヌルになることはありません。表が分散表の場合には、結果は DECIMAL(15,0) になります。分散表の詳細については、「DB2 UDB for iSeries マルチ・システム」を参照してください。

COUNT(*) の引き数は、行の集合です。この結果は、その集合の行の数になります。この行の数には、ヌル値しか入っていない行も含まれます。

COUNT(式) の引き数は、値の集合です。この関数は、引き数の値からヌル値を除いた値の集合に適用されます。結果はその集合の値の個数です。

COUNT(DISTINCT 式) の引き数は、値の集合です。引き数値は、長さ属性が 2000 を超える文字ストリング、または長さ属性が 1000 DBCS か UCS-2 文字を超えるグラフィック・ストリング、LOB、またはデータ・リンクを除き、どのような値でも構いません。この関数は、引き数値からヌル値および重複する値を除いて得られる値の集合に適用されます。結果はその集合の値の個数です。

COUNT(DISTINCT 式) を含むステートメントの実行時に *HEX 以外のソート順序が有効で、しかも引き数が SBCS、UCS-2 または混合データを含む場合、結果は、集合の各値の重み付けされた値の比較によって求められます。値の重み付けは、該当のソート順序に基づいています。

例

- 表 EMPLOYEE を使用して、列 SEX の値が 'F' である行の個数をホスト変数 FEMALE (INTEGER) にセットします。

```
SELECT COUNT(*)
  INTO :FEMALE
  FROM EMPLOYEE
  WHERE SEX = 'F'
```

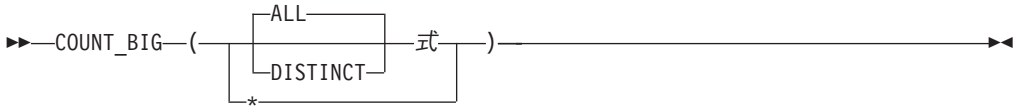
  上記の結果、FEMALE が 13 にセットされます。

- 表 EMPLOYEE を使用して、ホスト変数 FEMALE_IN_DEPT (INTEGER) に、少なくとも一人の女性がいる部門 (WORKDEPT) の数をセットします。

```
SELECT COUNT(DISTINCT WORKDEPT)
  INTO :FEMALE_IN_DEPT
  FROM EMPLOYEE
  WHERE SEX='F'
```

  上記の結果、FEMALE_IN_DEPT は 5 にセットされます (部門 A00、 C01、 D11、 D21、および E11 に、それぞれ女性が少なくとも 1 人はいます)。

COUNT_BIG



COUNT_BIG 関数は、行または値の集合の中にある行の数または値の数を戻します。この関数は COUNT 関数に類似していますが、COUNT 関数とは異なり、結果の値は整数の最大値より大きい値をとることができます。

関数の結果は、精度が 31 で位取りが 0 の 10 進数になります。結果がヌルになることはありません。

COUNT_BIG(*) の引き数は、行の集合です。この結果は、その集合の行の数になります。この行の数には、ヌル値しか入っていない行も含まれます。

COUNT_BIG(式) の引き数は、値の集合です。この関数は、引き数の値からヌル値を除いた値の集合に適用されます。結果はその集合の値の個数です。

COUNT_BIG(DISTINCT 式) の引き数は、値の集合です。引き数値は、長さ属性が 2000 を超える文字ストリング、または長さ属性が 1000 DBCS か UCS-2 文字を超えるグラフィック・ストリング、LOB、またはデータ・リンクを除き、どのような値でも構いません。この関数は、引き数値からヌル値および重複する値を除いて得られる値の集合に適用されます。結果はその集合の値の個数です。

COUNT_BIG(DISTINCT 式) を含むステートメントの実行時に *HEX 以外のソート順序が有効で、しかも引き数が SBCS、UCS-2 または混合データを含む場合、結果は、集合の各値の重み付けされた値の比較によって求められます。 値の重み付けは、該当のソート順序に基づいています。

例

- COUNT の例を参照して、COUNT を COUNT_BIG と読み替えてください。結果のデータ・タイプを除いて、結果は同じです。
- 特定の列上でカウントするためには、ソース化関数は列のタイプを指定しなければなりません。この例では、CREATE FUNCTION ステートメントが、CHAR として定義された任意の列を取るソース化関数を作成し、COUNT_BIG を使用してカウントを実行し、その結果を倍精度の浮動小数点数として戻します。以下の照会は、サンプルの従業員表内で固有の部門数をカウントします。

```
CREATE FUNCTION RICK.COUNT(CHAR()) RETURNS DOUBLE
  SOURCE QSYS2.COUNT_BIG(CHAR());

SET CURRENT PATH RICK, SYSTEM PATH

SELECT COUNT(DISTINCT WORKDEPT FROM EMPLOYEE;
```

新しい関数 (RICK.COUNT) のパラメーター・リストの中にある空の括弧は、この新しい関数の入力パラメーターが、SOURCE 文節に指定されている関数の入力パラメーターと同じタイプであることを意味します。 SOURCE 文節

(COUNT_BIG) の中にある空の括弧は、DB2 が COUNT_BIG 関数を見付けるときに、COUNT_BIG 関数の CHAR パラメーターの長さ属性を無視することを意味します。

MAX

MAX 列関数は、あるグループの中の値の集合の最大値を戻します。

引き数値は、LOB とデータ・リンク値を除く任意の組み込みデータ・タイプとすることができます。

結果のデータ・タイプおよび長さ属性は、引き数の値のデータ・タイプおよび長さ属性と同じになります。引き数がストリングの場合、結果は引き数と同じ CCSID を持ちます。結果が、ヌルになることもあります。

MAX 関数を含むステートメントの実行時に *HEX 以外のソート順序が有効で、しかも引き数が SBCS、UCS-2 または混合データを含む場合、結果は、集合の各値の重み付けされた値の比較によって求められます。値の重み付けは、該当のソート順序に基づいています。

この関数は、引き数の値からヌル値を除いた値の集合に対して適用されます。

この関数を空集合に対して適用すると、結果はヌル値になります。それ以外の場合は、結果は集合の中の最大値になります。

DISTINCT の指定は結果に影響を与えず、またその使用はお勧めできません。

例

- EMPLOYEE 表を使用して、ホスト変数 MAX_SALARY (DECIMAL(7,2)) を最大月給 (SALARY / 12) の値に設定します。

```
SELECT MAX(SALARY) /12
  INTO :MAX_SALARY
  FROM EMPLOYEE
```

  上記の結果、MAX_SALARY は 4395.83 にセットされます。

- 表 PROJECT を使用して、ソート順序で最後になるプロジェクト名 (PROJNAME) をホスト変数 LAST_PROJ (CHAR(24)) にセットします。

```
SELECT MAX(PROJNAME)
  INTO :LAST_PROJ
  FROM PROJECT
```

  上記の結果、LAST_PROJ は 'WELD LINE PLANNING ' にセットされます。

MIN

MIN 列関数は、あるグループの中の値の集合の最小値を戻します。

引き数値は、LOB とデータ・リンク値を除く任意の組み込みデータ・タイプとすることができます。

結果のデータ・タイプおよび長さ属性は、引き数の値のデータ・タイプおよび長さ属性と同じになります。引き数がストリングの場合、結果は引き数と同じ CCSID を持ちます。結果が、ヌルになることもあります。

MIN 関数を含むステートメントの実行時に *HEX 以外のソート順序が有効で、しかも引き数が SBCS、UCS-2 または混合データを含む場合、結果は、集合の各値の重み付けされた値の比較によって求められます。

この関数は、引き数の値からヌル値を除いた値の集合に対して適用されます。

この関数を空集合に対して適用すると、結果はヌル値になります。それ以外の場合は、結果はその集合の中の最小値になります。

DISTINCT の指定は結果に影響を与えず、またその使用はお勧めできません。

例

- 表 EMPLOYEE を使用して、部門 (WORKDEPT) 'D11' の社員の手数料 (COMM) の最大値と最小値の差をホスト変数 COMM_SPREAD (DECIMAL(7,2)) にセットします。

```
SELECT MAX(COMM) - MIN(COMM)
  INTO :COMM_SPREAD
  FROM EMPLOYEE
  WHERE WORKDEPT  = 'D11'
```

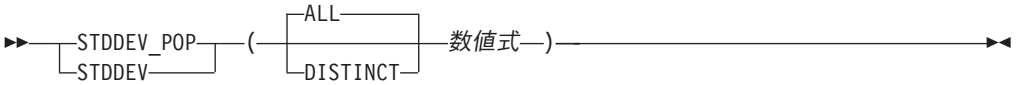
  上記の結果、COMM_SPREAD は 1118 (つまり、2580 - 1462) にセットされます。

- 表 PROJECT を使用して、最も早期に完了が予定されているプロジェクトの予定終了日付 (PRENDATE) を、ホスト変数 FIRST_FINISHED (CHAR(10)) にセットします。

```
SELECT MIN(PRENDATE)
  INTO :FIRST_FINISHED
  FROM PROJECT
```

  上記の結果、FIRST_FINISHED は '1982-09-15' にセットされます。

STDDEV または STDDEV_POP



STDDEV_POP 関数は、数値の集合のバイアス標準偏差 (/n) を戻します。 バイアス標準偏差を計算するための数式は以下のとおりです。

```
STDDEV_POP = SQRT(VAR_POP)
```

ここで、SQRT(VAR_POP) は差の平方根です。

引き数値は組み込み数値データ・タイプのいずれかでなければならず、また合計は結果のデータ・タイプの範囲内でなければなりません。

結果のデータ・タイプは、倍精度の浮動小数点数になります。結果が、ヌルになることもあります。

この関数は、引き数の値からヌル値を除いた値の集合に対して適用されます。DISTINCT が指定される場合は、重複する値は除かれます。

この関数を空集合に対して適用すると、結果はヌル値になります。それ以外の場合は、結果は集合の中の値の標準偏差となります。

値が加算される順序は未定義ですが、中間結果はすべて結果のデータ・タイプの範囲内になければなりません。

STDEV_POP の同義語として STDDEV を指定することもできます。

例

- 表 EMPLOYEE を使用して、部門 A00 の従業員の給与の標準偏差を、ホスト変数 DEV (倍精度浮動小数点数) にセットします。

```
SELECT STDDEV_POP(SALARY)
  INTO :DEV
  FROM EMPLOYEE
  WHERE WORKDEPT = 'A00';
```

  上記の結果、DEV は約 9938.00 にセットされます。

SUM

SUM 関数は、数値の集合の合計値を戻します。

引き数値は組み込み数値データ・タイプのいずれかでなければならず、また合計は結果のデータ・タイプの範囲内でなければなりません。

結果のデータ・タイプは、以下の場合を除いて、引き数値のデータ・タイプと同じです。

- 引き数値が単精度の浮動小数点数の場合は、倍精度の浮動小数点数になります。
- 引き数値が短整数である場合、結果は長整数になります。
- 引き数の値が 2 進数で位取りがゼロ以外の場合、結果は 10 進数になります。

結果が、ヌルになることもあります。

引き数値のデータ・タイプが 10 進数または位取りがゼロ以外の 2 進数の場合は、結果の精度は 31 になり、位取りは引き数値の位取りと同じになります。

この関数は、引き数の値からヌル値を除いた値の集合に対して適用されます。DISTINCT が指定される場合は、重複する値は除かれます。

この関数を空集合に対して適用すると、結果はヌル値になります。それ以外の場合は、結果は集合の中の値の合計値になります。

値が加算される順序は未定義ですが、中間結果はすべて結果のデータ・タイプの範囲内になければなりません。

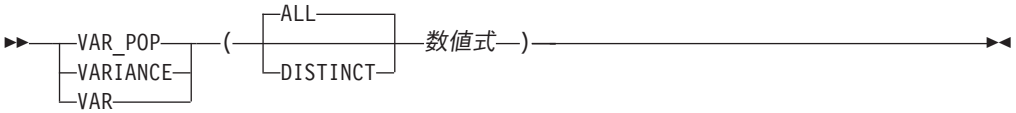
例

- 表 EMPLOYEE を使用して、事務職員 (JOB='CLERK') に支払われる賞与 (BONUS) の総額を、ホスト変数 JOB_BONUS (DECIMAL(9,2)) にセットします。

```
SELECT SUM(BONUS)
  INTO :JOB_BONUS
  FROM EMPLOYEE
  WHERE JOB = 'CLERK'
```

  上記の結果、JOB_BONUS は 2800 にセットされます。

VAR_POP または VARIANCE または VAR



VAR_POP 関数は、数値の集合のバイアス偏差 (/n) を戻します。バイアス偏差を計算するための数式は以下のとおりです。

```
VAR_POP = SUM(X**2)/COUNT(X) - (SUM(X)/COUNT(X))**2
```

引き数値は組み込み数値データ・タイプのいずれかでなければならず、また合計は結果のデータ・タイプの範囲内でなければなりません。

結果のデータ・タイプは、倍精度の浮動小数点数になります。結果が、ヌルになることもあります。

この関数は、引き数の値からヌル値を除いた値の集合に対して適用されます。DISTINCT が指定される場合は、重複する値は除かれます。

この関数を空集合に対して適用すると、結果はヌル値になります。それ以外の場合は、結果は集合の中の値の偏差になります。

値が加算される順序は未定義ですが、中間結果はすべて結果のデータ・タイプの範囲内になければなりません。

VAR_POP の同義語として、VARIANCE および VAR も指定できます。

例

- 表 EMPLOYEE を使用して、部門 A00 の従業員の給与の偏差を、ホスト変数 VARNCE (倍精度浮動小数点数) にセットします。

```
SELECT VAR_POP(SALARY)
  INTO :VARNCE
  FROM EMPLOYEE
  WHERE WORKDEPT = 'A00';
```

  上記の結果、VARNCE は約 98763888.88 にセットされます。

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できます。 スカラー関数は、値の集合ではなく単一のパラメーター値に適用されるものなので、列関数を使用するときの制約事項はスカラー変数には適用されません。スカラー関数では、関数を引き数として使用できます。ただし、スカラー関数の中で式や列関数を使用する場合は、それらの式および列関数の使用法に適用される制約が適用されます。例えば、スカラー関数の引き数に列関数を指定できるのは、そのスカラー関数が使用される文脈で列関数が許される場合だけです。

例

次の SELECT ステートメントの結果は、部門 D01 の従業員数と同じ行数になります。

```
SELECT EMPNO, LASTNAME, YEAR(CURRENT DATE - BIRTHDATE)
  FROM EMPLOYEE
  WHERE WORKDEPT = 'D01'
```

ABS

ABS 関数は、数値の絶対値を戻します。

引き数は、任意の組み込み数値データ・タイプの値を戻す式でなければなりません。

引き数値が短整数の場合は結果が長整数になり、引き数値が単精度浮動小数点数の場合は結果が倍精度浮動小数点数になるという点を除いて、結果のデータ・タイプと長さ属性は、引き数値のデータ・タイプと長さ属性と同じです。

引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

使用上の注意

ABSVAL は ABS の同義語です。これは、DB2 の旧リリースとの互換性を維持するためにのみサポートされています。

例

- ホスト変数 PROFIT は、値が -50000 の長整数であると想定します。

```
SELECT ABS(:PROFIT)
  FROM SYSIBM.SYSDUMMY1
```

  値として 50000 が戻されます。

ACOS

ACOS 関数は、引き数のアークコサイン (逆余弦) を、ラジアンで表した角度で戻します。 ACOS 関数と COS 関数は、逆の演算です。

引き数は、任意の組み込み数値データ・タイプの値を戻す式でなければなりません。 値は、-1 以上で、1 以下でなければなりません。

結果のデータ・タイプは、倍精度の浮動小数点数になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

結果は、0 以上で、PI 以下になります。

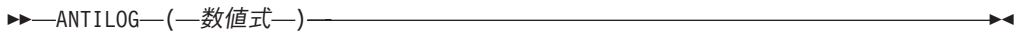
例

- ホスト変数 ACOSINE は、DECIMAL(10,9) で値が 0.070737202 のホスト変数であると想定します。

```
SELECT ACOS(:ACOSINE)
  FROM SYSIBM.SYSDUMMY1
```

  およそ 1.49 の値が戻されます。

ANTILOG



ANTILOG 関数は、数値の逆対数 (底 10) を戻します。ANTILOG 関数と LOG 関数は、逆の演算です。

引き数は、任意の組み込み数値データ・タイプの値を戻す式でなければなりません。

結果のデータ・タイプは、倍精度の浮動小数点数になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

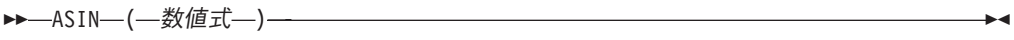
例

- ホスト変数 ALOG は、DECIMAL(10,9) のホスト変数で、値は 1.499961866 であると想定します。

```
SELECT ANTILOG(:ALOG)
  FROM SYSIBM.SYSDUMMY1
```

  およそ 31.62 の値が戻されます。

ASIN



ASIN 関数は、引き数のアークサイン (逆正弦) を、ラジアンで表した角度で戻します。 ASIN 関数と SIN 関数は、逆の演算です。

引き数は、任意の組み込み数値データ・タイプの値を戻す式でなければなりません。 値は、-1 以上で、1 以下でなければなりません。

結果のデータ・タイプは、倍精度の浮動小数点数になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

結果は、-PI /2 以上で、PI /2 以下になります。

例

- ホスト変数 ASINE は、DECIMAL(10,9) のホスト変数で値が 0.997494987 であると想定します。

```
SELECT ASIN(:ASINE)
  FROM SYSIBM.SYSDUMMY1
```

  およそ 1.50 の値が戻されます。

ATAN

ATAN 関数は、引き数のアークタンジェント (逆正接) を、ラジアンで表した角度で戻します。 ATAN 関数と TAN 関数は、逆の演算になります。

引き数は、任意の組み込み数値データ・タイプの値を戻す式でなければなりません。

結果のデータ・タイプは、倍精度の浮動小数点数になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

結果は、-PI/2 以上で、PI/2 以下になります。

例

- ホスト変数 ATANGENT は、DECIMAL(10,8) のホスト変数で値が 14.10141995 であると想定します。

```
SELECT ATAN(:ATANGENT)
  FROM SYSIBM.SYSDUMMY1
```

  およそ 1.50 の値が戻されます。

ATANH

ATANH 関数は、数値の双曲線アークタンジェント (双曲線逆正接) をラジアンで戻します。ATANH 関数と TANH 関数は、逆の演算です。

引き数は、任意の組み込み数値データ・タイプの値を戻す式でなければなりません。 値は、-1 より大きく 1 より小さくなければなりません。

結果のデータ・タイプは、倍精度の浮動小数点数になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

例

- ホスト変数 HATAN が、DECIMAL(10,9) のホスト変数で 0.905148254 の値を持つものと想定します。

```
SELECT ATANH(:HATAN)
  FROM SYSIBM.SYSDUMMY1
```

  およそ 1.50 の値が戻されます。

ATAN2

ATAN2 関数は、x 座標と y 座標のアークタンジェント (逆正接) を、ラジアンで表された角度として戻します。最初の引き数と 2 番目の引き数は、それぞれ x 座標と y 座標を表します。

各引き数は、任意の組み込み数値データ・タイプの値を戻す式です。引き数が両方共 0 であってはなりません。

結果のデータ・タイプは、倍精度の浮動小数点数になります。関数のいずれかの引き数がヌルである可能性がある場合は、結果もヌルである可能性があります。いずれかの引き数がヌルである場合は、結果はヌル値になります。

例

- ホスト変数 HATAN2A と HATAN2B は、それぞれ値が 1 と 2 の DOUBLE ホスト変数であるとします。

```
SELECT ATAN2(:HATAN2A,:HATAN2B)
  FROM SYSIBM.SYSDUMMY1
```

  これは、概略値 1.1071487 の倍精度の浮動小数点数を戻します。

BIGINT

数値から 64 ビット整数へ

文字から 64 ビット整数へ

BIGINT 関数は、次のものの 64 ビット整数表現を戻します。

- 数値
- 10 進数の文字ストリング表現
- 整数の文字ストリング表現
- 浮動小数点数の文字ストリング表現

注: 64 ビット整数値を戻すには、CAST 式も使用することができます。詳しくは、149 ページの『CAST の指定』を参照してください。

数値から 64 ビット整数へ

数値式
任意の組み込み数値データ・タイプの数値を戻す式。

引き数が*数値式* であれば、結果は、その引き数が 64 ビット整数の列または変数に割り当てられた場合に得られるのと同じ数値になります。引き数の整数部が、64 ビット整数の範囲内にない場合は、エラーになります。引き数の小数部は切り捨てられます。

文字から 64 ビット整数へ

文字式
整数の文字ストリング表現である値を戻す式。この式は CLOB であってはなりません。

引き数が*文字式* の場合、結果は、CAST(*文字式* AS BIGINT) で得られる数値と同じです。 先行ブランクと後書きブランクは除去され、結果のストリングは、浮動小数点数、整数、または 10 進数の定数を形成する規則に合致している必要があります。引き数の整数部が、整数の値の範囲内にない場合は、エラーが発生します。引き数の小数部は切り捨てられます。

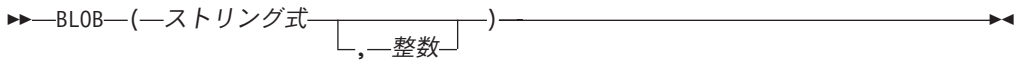
この関数の結果は、64 ビット整数になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

例

- EMPLOYEE 表を使用して、64 ビット整数形式の EMPNO 列を選択し、アプリケーション内の処理をさらに進めます。

```
SELECT BIGINT(SALARY)
  FROM EMPLOYEE
```

BLOB



BLOB 関数は、任意のタイプのストリングの BLOB 表現を戻します。

注: 2 進ストリング値を戻すには、CAST 式も使用できます。詳しくは、149 ページの『CAST の指定』を参照してください。

この関数の結果は、BLOB になります。最初の引き数がヌルである可能性がある場合は、結果もヌルになる可能性があります。最初の引き数がヌルの場合は、結果はヌル値になります。

ストリング式
値が文字ストリング、グラフィック・ストリング、2 進ストリング、または行 ID のいずれかであるストリング式。

整数
結果の 2 進ストリングの長さ属性を指定します。値は 1 〜 2 147 483 647 でなければなりません。

整数 を指定しなかった場合は、以下のようになります。

- ストリング式 が空ストリング定数の場合は、結果の長さ属性は 1。
- 空ストリング定数でない場合は、結果の長さ属性は、引き数がグラフィック・ストリングでない限り、最初の引き数の長さ属性と同じになります。グラフィック・ストリングの場合は、結果の長さ属性は、引き数の長さ属性の 2 倍です。

結果の実際の長さは、結果の長さ属性と式の実際の長さ (または入力がグラフィック・データの場合は式の長さの 2 倍) のいずれか小さい方となります。ストリング式 の長さが結果の長さ属性よりも大きい場合は、切り捨てが行われます。最初の入力引き数が文字ストリングで、切り捨てられた文字がすべてブランクである場合、または最初の入力引き数がグラフィック・ストリングで、切り捨てられた文字がすべて 2 バイト・ブランクである場合以外は、警告 (SQLSTATE 01004) が戻されます。

例

- 次の関数は、ストリング 'This is a BLOB' の BLOB を戻します。

```
SELECT BLOB('This is a BLOB')
  FROM SYSIBM.SYSDUMMY1
```

- 次の関数は、ロケーター myclob_locator が識別するラージ・オブジェクトの BLOB を戻します。

```
SELECT BLOB(:myclob_locator)
  FROM SYSIBM.SYSDUMMY1
```

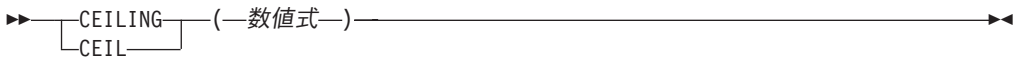
- 表に、TOPOGRAPHIC_MAP という名前の BLOB 列と、MAP_NAME という名前の VARCHAR があるとします。'Pellow Island' というストリングが入っているマップを見つけ、実際のマップの前にマップ名を連結した単一 2 進ストリング

BLOB

を戻すことにします。次の関数は、ロケーター myclob_locator が識別するラージ・オブジェクトの BLOB を戻します。

```
SELECT BLOB( MAP_NAME CONCAT  ':  ' CONCAT  TOPOGRAPHIC_MAP )
  FROM ONTARIO_SERIES_4
  WHERE TOPOGRAPHIC_MAP LIKE '%Pellow Island%'
```

CEILING



CEIL または CEILING 関数は、*数値式* より大きいか等しい最小の整数値を戻します。

引き数は、任意の組み込み数値データ・タイプの値を戻す式です。

この関数の結果のデータ・タイプと長さ属性は引き数と同じになりますが、引き数が DECIMAL または NUMERIC の場合は位取りは 0 になります。例えば、データ・タイプが DECIMAL(5,5) の引き数の場合、結果は DECIMAL(5,0) となります。

引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

例

- 全従業員の中で最も高い月収を検索します。結果を次の整数に切り上げます。SALARY 列は、10 進数のデータ・タイプを持っています。

```
SELECT CEIL(MAX(SALARY)/12
  FROM EMPLOYEE
```

  この例では、4396.00 が戻されます。給与が最も高い従業員は Christine Haas であり、その年収は $52750.00 だからです。CEIL 関数を適用する前の彼女の平均月収は 4395.83 です。

- 正負両方の数値に関して CEILING を使用します。

```
SELECT CEILING( 3.5),
       CEILING( 3.1),
       CEILING(-3.1),
       CEILING(-3.5),
  FROM SYSIBM.SYSDUMMY1
```

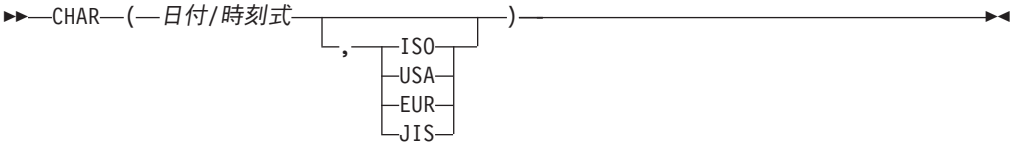
  この例ではそれぞれ、

```
04.   04.  -03.  -03.
```

  が戻されます。

CHAR

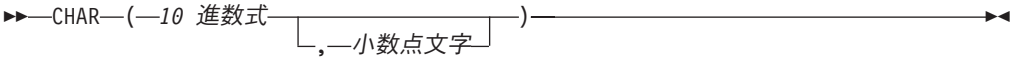
日付/時刻から文字に



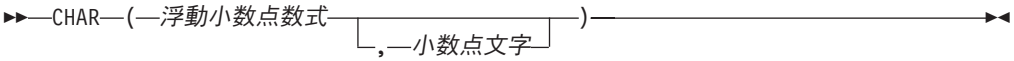
文字から文字に

整数から文字に

10 進数から文字に



浮動小数点数から文字に



CHAR 関数は、次の値の固定長文字ストリング表現を戻します。

- 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)
- 倍精度浮動小数点数 (最初の引き数が DOUBLE または REAL の場合)
- 文字ストリング (最初の引き数が任意のタイプの文字ストリングの場合)
- 日付値 (最初の引き数が DATE の場合)
- 時刻値 (最初の引き数が TIME の場合)
- タイム・スタンプ値 (最初の引き数が TIMESTAMP)
- 行 ID 値 (最初の引き数が ROWID の場合)

注: 固定長文字ストリング値を戻すには、CAST 式も使用できます。詳しくは、149 ページの『CAST の指定』を参照してください。

最初の引き数は、BLOB、GRAPHIC、VARGRAPHIC、または DBCLOB 以外の組み込みデータ・タイプでなければなりません。

この関数の結果は、固定長文字ストリングになります。最初の引き数がヌルである可能性がある場合は、結果もヌルになる可能性があります。最初の引き数がヌルの場合は、結果はヌル値になります。

日付/時刻から文字に

日付/時刻式
次の 3 つの組み込みデータ・タイプのいずれかである式。

日付 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現です。 2 番目の引き数を指定しなかった場合は、デフォルトの日付形式が使用されます。形式として ISO、USA、EUR、または JIS を指定すると、結果の長さは 10 になります。その他の場合は、結果の長さはデフォルトの日付形式の長さになります。詳細については、 71 ページの『日付/時刻の値のストリング表現』を参照してください。

時刻 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現です。 2 番目の引き数を指定しなかった場合は、デフォルトの時刻形式が使用されます。 結果の長さは 8 になります。

タイム・スタンプ
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。

結果は、タイム・スタンプの文字ストリング表現です。結果の長さは 26 になります。

ストリングの CCSID は、現行サーバーにおけるデフォルト SBCS CCSID です。

ISO、EUR、USA、または JIS
結果の文字ストリングの日付形式または時刻形式を指定します。詳しくは、 71 ページの『日付/時刻の値のストリング表現』を参照してください。

文字から文字に

文字式
組み込み文字ストリング・データ・タイプである値を戻す式。

整数
結果の固定長の文字ストリングのための長さ属性を指定します。値は 1 から 32766 (ヌルでもよい場合は、32765) までででなければなりません。最初の引き数が混合データである場合は、2 番目の引き数は 4 より小さくてはなりません。

2 番目の引き数を指定しない場合は、次のようになります。

- 文字式 が空ストリング定数の場合は、結果の長さ属性は 1。
- 空ストリング定数でない場合は、結果の長さ属性は、最初の引き数の長さ属性と同じ。

実際の長さは、結果の長さ属性と同じになります。文字式 の長さが結果の長さより小さい場合は、結果は、結果に指定されている長さまでブランクで埋め込まれます。文字式 の長さが結果の長さ属性よりも大きい場合は、切り捨てが行われます。切り捨てられた文字がすべてブランクであった場合以外は、警告 (SQLSTATE 01004) が戻されます。

ストリングの CCSID は、文字式 の CCSID になります。

整数から文字に

整数式
整数データ・タイプ (SMALLINT、INTEGER、または BIGINT) の値を戻す式。

結果は、引き数を SQL 整数定数の形式で表した固定長文字ストリング表現です。結果は、引き数の値を表す n 文字の有効数字から成ります。引き数が負数の場合は、負符号が前に付きます。結果は左寄せにされます。

- 引き数が短整数の場合は、

  結果の長さは 6 です。結果の文字数が 6 文字より少ない場合は、結果は右側がブランクで埋め込まれます。
- 引き数が長整数の場合は、

  結果の長さは 11 です。結果の文字数が 11 文字より少ない場合は、結果は右側がブランクで埋め込まれます。
- 引き数が 64 ビット整数の場合は、

  結果の長さは 20 です。結果の文字数が 20 文字より少ない場合は、結果は右側がブランクで埋め込まれます。

ストリングの CCSID は、現行サーバーにおけるデフォルト SBCS CCSID です。

10 進数から文字に

10 進数式
組み込み 10 進数データ・タイプ (DECIMAL または NUMERIC) の値を戻す式。 精度や位取りを変えたい場合は、DECIMAL スカラー関数を使用して変更することができます。

小数点文字
結果の文字ストリングにおいて、小数点以下を区切るために使用する 1 バイト文字の定数を指定します。この文字は、ピリオドかコンマとする必要があります。 2 番目の引き数を指定しなかった場合は、デフォルトの小数点が使用されます。詳しくは、 108 ページの『小数点』を参照してください。

結果は、引き数を固定長の文字ストリングで表現したものになります。この結果には、1 文字の小数点文字と *p* 桁までの数字が含まれます。*p* は *10 進数式* の精度で、引き数が負数の場合は負符号が先頭に付きます。先行ゼロは戻されません。後続ゼロは戻されます。

結果の長さは、2+*p* です。*p* は *10 進数式* の精度です。 すなわち、正の値には、1 桁の後書きブランクが常に含まれることになります。

ストリングの CCSID は、現行サーバーにおけるデフォルト SBCS CCSID です。

浮動小数点数から文字に

浮動小数点数式
組み込み浮動小数点数データ・タイプである値を戻す式。

小数点文字
結果の文字ストリングにおいて、小数点以下を区切るために使用する 1 バイト文字の定数を指定します。この文字は、ピリオドかコンマとする必要があります。 2 番目の引き数を指定しなかった場合は、デフォルトの小数点が使用されます。詳しくは、 108 ページの『小数点』を参照してください。

結果は、引き数を浮動小数点定数の形式で表した固定長文字ストリング表現です。結果の長さは 24 です。引き数が負数の場合は、結果の最初の文字は負符号になります。負数以外の場合は、最初の文字は数字です。引き数がゼロであると、結果は

0E0 になります。それ以外の場合の結果には、ゼロ以外の 1 桁の数字と、それに続くピリオド 1 つおよび一連の数字から成る小数部の引き数を表す値に使用される最小文字数が含まれます。

結果の文字数が 24 文字より少ない場合は、結果は右側がブランクで埋め込まれます。

ストリングの CCSID は、現行サーバーにおけるデフォルト SBCS CCSID です。

例

- 列 PRSTDATE には、1988-12-25 に相当する内部値が入っていると想定します。日付形式は *MDY で、日付区切り記号はスラッシュ (/) です。

```
SELECT CHAR(PRSTDATE, USA)
  FROM PROJECT
```

  結果として、'12/25/1988' の値が戻されます。

```
SELECT CHAR(PRSTDATE)
  FROM PROJECT
```

  結果として、'12/25/88' の値が戻されます。

- 列 STARTING 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 (つまり、5 時間) の値を持つ時刻期間であると想定します。

```
SELECT CHAR(STARTING, USA)
  FROM CL_SCHED
```

  結果として、'5:12 PM' の値が戻されます。

```
SELECT CHAR(STARTING + :HOUR_DUR, JIS)
  FROM CL_SCHED
```

  結果として、'10:12:00' の値が戻されます。

- 列 RECEIVED (タイム・スタンプ) には、列 PRSTDATE と列 STARTING を合わせた値に相当する内部値が入っていると想定します。

```
SELECT CHAR(RECEIVED)
  FROM IN_TRAY
```

  この結果、'1988-12-25-17.12.30.000000' の値が戻されます。

- CHAR 関数を使用して、タイプを固定長文字にし、表示桁の長さを EMPLOYEE 表 (VARCHAR(15) と定義) の LASTNAME 列 の 10 文字までに減らすには、次のように指定します。

```
SELECT CHAR(LASTNAME,10)
  FROM EMPLOYEE
```

  LASTNAME を持つ行が 10 文字 (後書きブランクを除く) を超える場合は、値が切り捨てられるという警告 (SQLSTATE 01004) が出ます。

- CHAR 関数を使用して、EDLEVEL (SMALLINT と定義) の値を固定長ストリングとして戻します。

```
SELECT CHAR(EDLEVEL)
  FROM EMPLOYEE
```

EDLEVEL の値が 18 の場合、CHAR(6) では値 '18bbbb' (18 の後ろに 4 つのブランクが続く) が戻されます。

- STAFF 表に、精度 9、位取り 2 の 10 進数の SALARY 列が定義されているとします。現在値は 18357.50、小数点文字はコンマです (18357,50)。

```
SELECT CHAR(SALARY, ',')
  FROM EMPLOYEE
```

結果として、'18357,50bbb' (18357,50 の後に 3 つのブランクを付けたもの) が戻されます。

- 同じ SALARY 列が 20000.25 から減算され、デフォルト値の小数点文字で値が戻されるとします。デフォルト小数点文字はピリオドです。

```
SELECT CHAR(20000.25 - SALARY)
  FROM EMPLOYEE
```

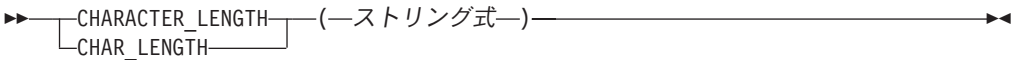
結果として、'-1642.75bbb' (-1642.75 の後に 3 つのブランクを付けたもの) が戻されます。

- ホスト変数 DOUBLE_NUM が倍精度浮動小数点データ・タイプで、値が -987.654321E-35 であるとします。

```
SELECT CHAR(:DOUBLE_NUM)
  FROM SYSIBM.SYSDUMMY1
```

結果は、文字値 '-9.8765432100000002E-33 ' になります。

CHARACTER_LENGTH



CHARACTER_LENGTH または CHAR_LENGTH 関数は、ストリング式の長さを戻します。同様な関数として、 261 ページの『LENGTH』を参照してください。

引き数は、任意の組み込みストリング・データ・タイプの値を戻す式です。

この関数の結果は、長整数になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

結果は、引き数の文字数 (バイト数ではなく) です。1 文字は、SBCS または DBCS 文字です。ストリングの長さには、後書きブランクも含まれます。可変長ストリングを指定した場合に戻る長さは、実際の長さであり、最大長ではありません。

例

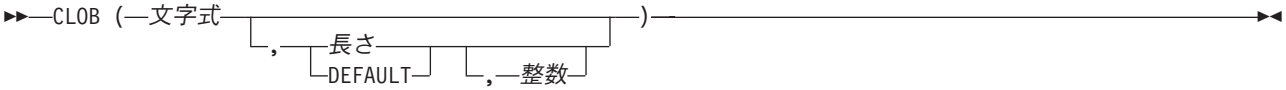
- ホスト変数 ADDRESS は、値が '895 Don Mills Road' の可変長文字ストリングであると想定します。

```
SELECT CHARACTER_LENGTH(:ADDRESS)
  FROM SYSIBM.SYSDUMMY1
```

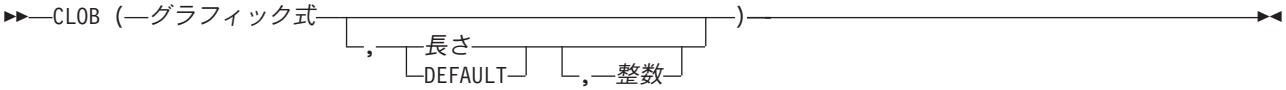
  値 18 が戻されます。

CLOB

文字から CLOB に

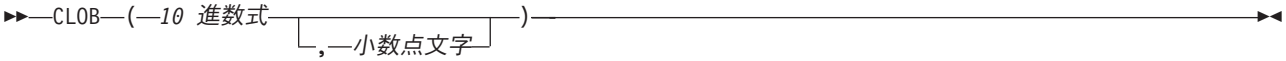


グラフィックから CLOB に



整数から CLOB に

10 進数から CLOB に



浮動小数点数から CLOB に

CLOB 関数は、次のものの文字ストリング表現を戻します。

- 整数 (最初の引き数が SMALLINT、INTEGER、または BIGINT の場合)
- 10 進数 (最初の引き数がパックまたはゾーン 10 進数の場合)
- 倍精度浮動小数点数 (最初の引き数が DOUBLE または REAL の場合)
- 文字ストリング (最初の引き数が任意のタイプの文字ストリングの場合)
- グラフィック・ストリング (最初の引き数が UCS-2 グラフィック・ストリングである場合)

注: 文字ラージ・オブジェクト値を戻すには、CAST 式も使用できます。詳しくは、149 ページの『CAST の指定』を参照してください。

この関数の結果は、CLOB ストリングになります。最初の引き数がヌルである可能性がある場合は、結果もヌルになる可能性があります。最初の引き数がヌルの場合は、結果はヌル値になります。

文字から CLOB に

文字式
組み込み文字ストリング・データ・タイプである値を戻す式。

長さ
結果の可変長文字ストリングのための長さ属性を指定します。値は 1 ～ 2 147 483 647 でなければなりません。 最初の引き数が混合データである場合は、2 番目の引き数は 4 より小さくてはなりません。

2 番目の引き数が指定されないか DEFAULT が指定された場合は、次のようになります。

- 文字式 が空ストリング定数の場合は、結果の長さ属性は 1。
- 空ストリング定数でない場合は、結果の長さ属性は、最初の引き数の長さ属性と同じ。

結果の実際の長さは、結果の長さ属性と 文字式 の実際の長さのいずれか小さい方となります。文字式 の長さが結果の長さ属性よりも大きい場合は、切り捨てが行われます。切り捨てられた文字がすべてブランクであった場合以外は、警告 (SQLSTATE 01004) が戻されます。

整数
結果の CCSID を指定します。これは有効な SBCS CCSID または混合データ CCSID とする必要があります。3 番目の引き数が SBCS CCSID の場合は、結果は SBCS データになります。3 番目の引き数が混合 CCSID の場合は、結果は混合データになります。3 番目の引き数が SBCS CCSID の場合は、最初の引き数が DBCS 択一または DBCS 専用のストリングであることはありません。3 番目の引き数を 65535 とすることはできません。

3 番目の引き数の指定がない場合は、最初の引き数の CCSID は 65535 であってはなりません。

- 最初の引き数がビット・データの場合は、エラーとなる。
- 最初の引き数が SBCS データであれば、結果は SBCS データになる。結果の CCSID は、最初の引き数の CCSID と同一です。
- 最初の引き数が混合データ (DBCS 混用、DBCS 専用、または DBCS 択一) であれば、結果は混合データになる。結果の CCSID は、最初の引き数の CCSID と同一です。

グラフィックから CLOB に

グラフィック式
組み込みグラフィック・ストリング・データ・タイプである値を戻す式。最初の引き数は、DBCS グラフィック・データであってはなりません。

長さ
結果の可変長文字ストリングのための長さ属性を指定します。値は 1 ～ 2 147 483 647 でなければなりません。 結果が混合データの場合は、2 番目の引き数は 4 より小さくてはなりません。

2 番目の引き数が指定されていないか、または DEFAULT が指定されている場合は、結果の長さ属性は、次のように決まります。(ただし、*n* は最初の引き数の長さ属性です。)

- グラフィック式 が空グラフィック・ストリング定数の場合は、結果の長さ属性は 1 になる。
- 結果が SBCS データであれば、結果の長さは *n* になる。
- 結果が混合データであれば、結果の長さは (2.5*(*n* -1)) + 4 になる。

結果の実際の長さは、結果の長さ属性とグラフィック式 の実際の長さのいずれか小さい方となります。グラフィック式 の長さが結果の長さ属性よりも大きい場合は、切り捨てが行われます。切り捨てられた文字がすべてブランクであった場合以外は、警告 (SQLSTATE 01004) が戻されます。

整数
結果の CCSID を指定します。これは有効な SBCS CCSID または混合データ CCSID とする必要があります。3 番目の引き数が SBCS CCSID の場合は、結果は SBCS データになります。3 番目の引き数が混合 CCSID の場合は、結果は混合データになります。3 番目の引き数を 65535 とすることはできません。

3 番目の引き数が指定されていない場合は、結果の CCSID は現行サーバーのデフォルト CCSID になります。デフォルト CCSID が混合データの場合は、結果は混合データになります。デフォルト CCSID が SBCS データの場合は、結果は SBCS データになります。

整数から CLOB に

整数式
組み込み整数データ・タイプ (SMALLINT、INTEGER、または BIGINT) の値を戻す式。

結果は、SQL 整数定数の形式で引き数を表現した可変長文字ストリングです。結果は、引き数の値を表す n 文字の有効数字から成ります。引き数が負数の場合は、負符号が前に付きます。結果は左寄せにされます。

- 引き数が短整数の場合は、結果の長さ属性は 6
- 引き数が長整数の場合は、結果の長さ属性は 11
- 引き数が 64 ビット整数の場合は、結果の長さ属性は 20

結果の実際の長さは、引き数の値を表すために使用できる最小文字数です。先行ゼロは含まれません。引き数が負数の場合は、結果の最初の文字は負符号になります。負数以外の場合は、最初の文字は数字です。

結果の CCSID は、現行サーバーのデフォルト SBCS CCSID になります。

10 進数から CLOB に

10 進数式
組み込み 10 進数データ・タイプ (DECIMAL または NUMERIC) の値を戻す式。 精度や位取りを変えたい場合は、DECIMAL スカラー関数を使用して変更することができます。

小数点文字
結果の文字ストリングにおいて、小数点以下を区切るために使用する 1 バイト文字の定数を指定します。この文字は、ピリオドかコンマとする必要があります。 2 番目の引き数を指定しなかった場合は、デフォルトの小数点が使用されます。詳しくは、 108 ページの『小数点』を参照してください。

結果は、引き数を可変長文字ストリングで表現したものになります。この結果には、1 文字の小数点文字と *p* 桁までの数字が含まれます。*p* は *10 進数式* の精度で、引き数が負数の場合は負符号が先頭に付きます。先行ゼロは戻されません。後続ゼロは戻されます。

結果の長さ属性は、2+*p* です。*p* は *10 進数式* の精度です。結果の実際の長さは、引き数の値を表すために使用できる最小文字数ですが、ただし、後書き文字も含まれます。先行ゼロは含まれません。引き数が負数の場合は、結果の最初の文字は負符号になります。負数でなければ、結果は数字で始まります。

結果の CCSID は、現行サーバーのデフォルト SBCS CCSID になります。

浮動小数点数から CLOB に

浮動小数点数式
組み込み浮動小数点データ・タイプ (DOUBLE または REAL) の値を戻す式。

小数点文字
結果の文字ストリングにおいて、小数点以下を区切るために使用する 1 バイト文字の定数を指定します。この文字は、ピリオドかコンマとする必要があります。 2 番目の引き数を指定しなかった場合は、デフォルトの小数点が使用されます。詳しくは、 108 ページの『小数点』を参照してください。

結果は、浮動小数点定数の形式で引き数を可変長文字ストリングで表現したものになります。

結果の長さ属性は、24 です。結果の実際の長さは、ゼロ以外の 1 桁の数字、その後ろに 1 つの小数点文字 と一連の数字が続く小数部の引き数の値を表す最小文字数です。引き数が負数の場合は、結果の最初の文字は負符号になります。負数でなければ、最初の文字は数字です。引き数がゼロであると、結果は 0E0 になります。

結果の CCSID は、現行サーバーのデフォルト SBCS CCSID になります。

例

- 次の関数は、ストリング 'This is a CLOB' の CLOB を戻します。

```
SELECT CLOB('This is a CLOB')
  FROM SYSIBM.SYSDUMMY1
```

COALESCE

COALESCE 関数は、ヌルでない最初の式の値を戻します。

各引き数には、互換性がなければなりません。文字ストリングの引き数は、日付/時刻の値と互換性があります。データ・タイプの互換性についての詳細は、 85 ページの『割り当ておよび比較』を参照してください。引き数として使用できるのは、組み込みデータ・タイプまたは特殊タイプのいずれかです。[30]

引き数は、指定されている順序にしたがって評価され、ヌルでない最初の引き数がこの関数の結果となります。結果がヌルになる可能性があるのは、指定した引き数がどれもヌルになる可能性がある場合だけです。また、結果が実際にヌルになるのは、すべての引き数がヌルだった場合だけです。

選択された引き数は、必要があれば、結果の属性に変換されます。結果の属性は、99 ページの 『結果のデータ・タイプに関する規則』で説明しているすべてのオペランドを基にして決められます。

例

- 表 DEPARTMENT にあるすべての行からすべての値を選択するときに、部門責任者 (MGRNO) が欠落しているもの (つまり、ヌルのもの) については、'ABSENT' の値を戻します。

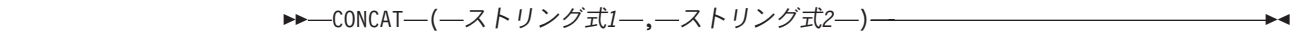
```
SELECT DEPTNO, DEPTNAME, COALESCE(MGRNO, 'ABSENT'), ADMRDEPT
  FROM DEPARTMENT
```

- 表 EMPLOYEE のすべての行から、従業員番号 (EMPNO) および給与 (SALARY) を選択するときに、給与が欠落しているもの (つまり、ヌルのもの) については、値としてゼロを戻します。

```
SELECT EMPNO, COALESCE(SALARY,0)
  FROM EMPLOYEE
```

30. この関数は、ユーザー定義の関数を作成するときのソース関数として使用することはできません。この関数は互換性のあるデータ・タイプを引き数として受け入れるので、特殊タイプをサポートするために追加のシグニチャーを作成する必要はありません。

CONCAT



CONCAT 関数は、2 つのストリング引き数を結合します。 2 つの引き数は互換性のあるストリングでなければなりません。データ・タイプの互換性についての詳細は、 85 ページの『割り当ておよび比較』を参照してください。

この関数の結果は、第 1 のストリングの後に第 2 のストリングを結合したストリングです。引き数のどちらかがヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数のどちらかがヌルである場合は、結果はヌル値になります。

CONCAT 関数は CONCAT 演算子と同等です。詳細は、137 ページの 『連結演算子を使用する式』を参照してください。

例

- 列 FIRSTNME と列 LASTNAME を連結します。

```
SELECT CONCAT(FIRSTNME, LASTNAME)
  FROM EMPLOYEE
  WHERE EMPNO ='000010'
```

  'CHRISTINEHAAS' の値が戻されます。

COS

COS 関数は、引き数のコサイン (余弦) を戻すもので、引き数はラジアンで表された角度です。 COS 関数と ACOS 関数は、逆の演算になります。

引き数は、任意の組み込み数値データ・タイプの値を戻す式でなければなりません。

結果のデータ・タイプは、倍精度の浮動小数点数になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

例

- ホスト変数 COSINE は、値が 1.5 の DECIMAL(2,1) のホスト変数であると想定します。

```
SELECT COS(:COSINE)
  FROM SYSIBM.SYSDUMMY1
```

  およそ 0.07 の値が戻されます。

COSH

COSH 関数は、引き数の双曲線コサイン (双曲線余弦) を戻すもので、引き数はラジアンで表された角度です。

引き数は、任意の組み込み数値データ・タイプの値を戻す式でなければなりません。

結果のデータ・タイプは、倍精度の浮動小数点数になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

例

- ホスト変数 HCOS は、値が 1.5 の DECIMAL(2,1) のホスト変数であると想定します。

```
SELECT COSH(:HCOS)
  FROM SYSIBM.SYSDUMMY1
```

  およそ 2.35 の値が戻されます。

COT

COT 関数は引き数のコタンジェント (余接) を戻すもので、引き数はラジアンで表された角度です。

引き数は、任意の組み込み数値データ・タイプの値を戻す式でなければなりません。

結果のデータ・タイプは、倍精度の浮動小数点数になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

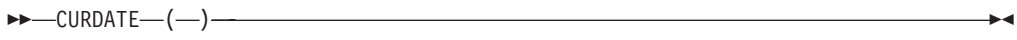
例

- ホスト変数 COTAN は、値が 1.5 の DECIMAL(2,1) のホスト変数であると想定します。

```
SELECT COT(:COTAN)
  FROM SYSIBM.SYSDUMMY1
```

  およそ 0.07 の値が戻されます。

CURDATE



CURDATE 関数は、SQL ステートメントが現行サーバーで実行される時点の刻時機構の読み取りに基づく日付を戻します。CURDATE 関数によって戻される値は、CURRENT DATE 特殊レジスターによって戻される値と同じです。

結果のデータ・タイプは日付になります。結果がヌルになることはありません。

この関数が 1 つの SQL ステートメント内で複数回使用される場合、または 1 つのステートメント内で CURTIME または NOW スカラー関数、あるいは CURRENT DATE、CURRENT TIME、または CURRENT TIMESTAMP 特殊レジスターとともに使用される場合は、値はすべて 1 回の刻時機構読み取りに基づきます。

例

- 刻時機構に基づく現在日付が戻されます。

```
SELECT CURDATE()
  FROM SYSIBM.SYSDUMMY1
```

CURTIME

CURTIME 関数は、SQL ステートメントが現行サーバーで実行される時点の刻時機構の読み取りに基づく時刻を戻します。CURTIME 関数によって戻される値は、CURRENT TIME 特殊レジスターによって戻される値と同じです。

結果のデータ・タイプは時刻です。結果がヌルになることはありません。

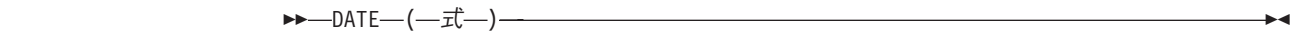
この関数が 1 つの SQL ステートメント内で複数回使用される場合、または 1 つのステートメント内で CURDATE または NOW スカラー関数、あるいは CURRENT DATE、CURRENT TIME、または CURRENT TIMESTAMP 特殊レジスターとともに使用される場合は、値はすべて 1 回の刻時機構読み取りに基づきます。

例

- 刻時機構に基づく現在時刻が戻されます。

```
SELECT CURTIME()
  FROM SYSIBM.SYSDUMMY1
```

DATE



DATE 関数は、指定された値に基づく日付を戻します。

引き数は、日付、タイム・スタンプ、文字ストリング、または数値のいずれかの組み込みデータ・タイプの値を戻す式でなければなりません。

- 式 が文字ストリングである場合は、そのストリングは CLOB であってはならず、値は以下のいずれかでなければなりません。
  - 日付またはタイム・スタンプの有効な文字ストリング表現。日付とタイム・スタンプの有効なストリング表現の形式については、71 ページの 『日付/時刻の値のストリング表現』を参照してください。
  - *yyyynnn* の形式で有効な日付を表す、実際長が 7 の文字ストリング。*yyyy* は年番号を表す数値で、*nnn* は年間通算日を表す 001 ～ 366 の数値です。
- 式 が数値である場合は、その数値は 3652059 以下の正の数でなければなりません。

注: 日付値を戻すには、CAST 式も使用できます。詳しくは、149 ページの『CAST の指定』を参照してください。

関数の結果は、日付になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

その他の規則は、引き数のデータ・タイプに応じて以下のように異なります。

- 引き数がタイム・スタンプの場合 :

  結果はタイム・スタンプの日付の部分になります。
- 引き数が日付の場合 :

  結果は、指定された日付になります。
- 引き数が数値の場合 :

  結果は、0001 年 1 月 1 日の *n*-1 日後の日付になります (*n* は、指定した数値の整数部です)。
- 引き数が文字ストリングの場合 :

  結果は、ストリングが示す日付か、ストリングが示すタイム・スタンプの日付部分です。

  日付のストリング表現が、SBCS データで、その CCSID が SBCS データのデフォルト CCSID とは異なる場合、その値は日付の値として解釈され、変換される前に、SBCS データのデフォルト CCSID を持つように変換されます。

  日付のストリング表現が、混合データで、その CCSID が混合データのデフォルト CCSID とは異なる場合、その値は日付の値として解釈され、変換される前に、混合データのデフォルト CCSID を持つように変換されます。

例

- 列 RECEIVED (TIMESTAMP) 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想定します。

```
SELECT DATE(RECEIVED)
  FROM IN_TRAY
```

  結果は、‘1988-12-25’ の内部表現になります。

- 次の例では、DATE スカラー関数が日付の ISO ストリング表現に適用されています。

```
SELECT DATE('1988-12-25')
  FROM SYSIBM.SYSDUMMY1
```

  結果は、‘1988-12-25’ の内部表現になります。

- 次の例では、DATE スカラー関数が日付の EUR ストリング表現に適用されています。

```
SELECT DATE('25.12.1988')
  FROM SYSIBM.SYSDUMMY1
```

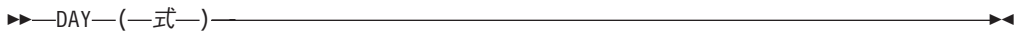
  結果は、‘1988-12-25’ の内部表現になります。

- 次の例では、DATE スカラー関数が正の整数に適用されています。

```
SELECT DATE(35)
  FROM SYSIBM.SYSDUMMY1
```

  結果は、‘0001-02-04’ の内部表現になります。

DAY



DAY 関数は、指定した値の日の部分を戻します。

引き数は、日付、タイム・スタンプ、文字ストリング、または数値データ・タイプのいずれかの組み込みデータ・タイプの値を戻す式でなければなりません。

- 式 が文字ストリングである場合は、そのストリングは CLOB であってはならず、値は日付またはタイム・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。日付とタイム・スタンプの有効なストリング表現の形式については、71 ページの 『日付/時刻の値のストリング表現』を参照してください。
- 式 が数値である場合は、その数値は日付期間またはタイム・スタンプ期間でなければなりません。日付時刻期間の有効な形式については、141 ページの 『日付/時刻のオペランドと期間』を参照してください。

この関数の結果は、長整数になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

その他の規則は、引き数のデータ・タイプ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、日付またはタイム・スタンプの有効な文字ストリング表現である場合は、次のようになります。

  結果は、指定した値の日の部分 (1 から 31 までの整数) 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、指定した値の日の部分 (-99 から 99 までの整数) 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

- 表 PROJECT を使用して、WELD LINE PLANNING プロジェクト (PROJNAME) の停止が予定されている日付 (PRENDATE) の日の部分を END_DAY (SMALLINT) にセットします。

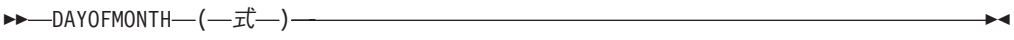
```
SELECT DAY(PRENDATE)
  INTO :END_DAY
  FROM PROJECT
  WHERE PROJNAME = 'WELD LINE PLANNING'
```

  結果として、END_DAY は 15 にセットされます。
- 2 つの日付間の差の日の部分を戻します。

```
SELECT DAY( DATE('2000-03-15') - DATE('1999-12-31') )
  FROM SYSIBM.SYSDUMMY1
```

  結果として、15 の値が戻されます。

DAYOFMONTH



DAYOFMONTH 関数は、月の中の日付を表す 1 ～ 31 の整数を戻します。

引き数は、日付、タイム・スタンプ、または文字ストリングのいずれかの組み込みデータ・タイプの値を戻す式でなければなりません。

式 が文字ストリングである場合は、そのストリングは CLOB であってはならず、値は日付またはタイム・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。日付とタイム・スタンプの有効なストリング表現の形式については、71 ページの 『日付/時刻の値のストリング表現』を参照してください。

この関数の結果は、長整数になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

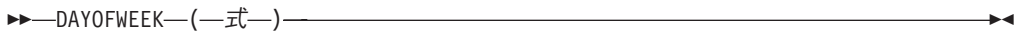
例

- 表 PROJECT を使用して、WELD LINE PLANNING プロジェクト (PROJNAME) の停止が予定されている日付 (PRENDATE) の日の部分を END_DAY (SMALLINT) にセットします。

```
SELECT DAYOFMONTH(PRENDATE)
  INTO :END_DAY
  FROM PROJECT
  WHERE PROJNAME = 'WELD LINE PLANNING'
```

  結果として、END_DAY は 15 にセットされます。

DAYOFWEEK



DAYOFWEEK 関数は、曜日を表す 1 から 7 までの整数 (1 は日曜日を表し、7 は土曜日を表す) を戻します。これに代わる別の方法については、 218 ページの『DAYOFWEEK_ISO』を参照してください。

引き数は、日付、タイム・スタンプ、または文字ストリングのいずれかの組み込みデータ・タイプの値を戻す式でなければなりません。

式 が文字ストリングである場合は、そのストリングは CLOB であってはならず、値は日付またはタイム・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。日付とタイム・スタンプの有効なストリング表現の形式については、71 ページの 『日付/時刻の値のストリング表現』を参照してください。

この関数の結果は、長整数になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

例

- 表 EMPLOYEE を使用して、Christine Haas (EMPNO='000010') の雇用が開始した曜日 (HIREDATE) にホスト変数 DAY_OF_WEEK (INTEGER) をセットします。

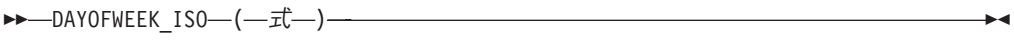
```
SELECT DAYOFWEEK(HIREDATE)
  INTO :DAY_OF_WEEK
  FROM EMPLOYEE
  WHERE EMPNO = '000010'
```

  DAY_OF_WEEK に 6 (金曜日を表す) がセットされる結果になります。

- 次の照会は、4 つの値 (1、2、1、2) を戻します。

```
SELECT DAYOFWEEK(CAST('10/11/1998' AS DATE)),
       DAYOFWEEK(TIMESTAMP('10/12/1998','01.02')),
       DAYOFWEEK(CAST(CAST('10/11/1998' AS DATE)) AS CHAR(20))),
       DAYOFWEEK(CAST(TIMESTAMP('10/12/1998','01.02') AS CHAR(20))),
  FROM SYSIBM.SYSDUMMY1
```

DAYOFWEEK_ISO



DAYOFWEEK_ISO 関数は、曜日を表す 1 から 7 までの整数 (1 は月曜日を表し、7 は日曜日を表す) を戻します。これに代わる別の方法については、217 ページの『DAYOFWEEK』を参照してください。

引き数は、日付、タイム・スタンプ、または文字ストリングのいずれかの組み込みデータ・タイプの値を戻す式でなければなりません。

式 が文字ストリングである場合は、そのストリングは CLOB であってはならず、値は日付またはタイム・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。日付とタイム・スタンプの有効なストリング表現の形式については、71 ページの 『日付/時刻の値のストリング表現』を参照してください。

この関数の結果は、長整数になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

例

- 表 EMPLOYEE を使用して、Christine Haas (EMPNO='000010') の雇用が開始した曜日 (HIREDATE) にホスト変数 DAY_OF_WEEK (INTEGER) をセットします。

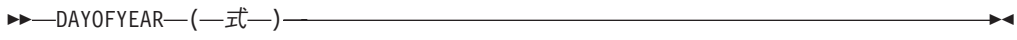
```
SELECT DAYOFWEEK_ISO(HIREDATE)
  INTO :DAY_OF_WEEK
  FROM EMPLOYEE
  WHERE EMPNO = '000010'
```

  DAY_OF_WEEK に 5 (金曜日を表す) がセットされる結果になります。

- 次の照会は、4 つの値、つまり 7、1、7、1 を戻します。

```
SELECT DAYOFWEEK_ISO(CAST('10/11/1998' AS DATE)),
       DAYOFWEEK_ISO(TIMESTAMP('10/12/1998','01.02')),
       DAYOFWEEK_ISO(CAST(CAST('10/11/1998' AS DATE)) AS CHAR(20))),
       DAYOFWEEK_ISO(CAST(TIMESTAMP('10/12/1998','01.02') AS CHAR(20))),
  FROM SYSIBM.SYSDUMMY1
```

DAYOFYEAR



DAYOFYEAR 関数は、年間通算日を表す 1 から 366 までの整数 (1 は 1 月 1 日を表す) を戻します。

引き数は、日付、タイム・スタンプ、または文字ストリングのいずれかの組み込みデータ・タイプの値を戻す式でなければなりません。

式 が文字ストリングである場合は、そのストリングは CLOB であってはならず、値は日付またはタイム・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。日付とタイム・スタンプの有効なストリング表現の形式については、71 ページの 『日付/時刻の値のストリング表現』を参照してください。

この関数の結果は、長整数になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

例

- 表 EMPLOYEE を使用して、従業員の雇用が開始された年間通算日 (HIREDATE) の平均をホスト変数 AVG_DAY_OF_YEAR (INTEGER) にセットします。

```
SELECT AVG(DAYOFYEAR(HIREDATE))
  INTO :AVG_DAY_OF_YEAR
  FROM EMPLOYEE
```

  結果として、AVG_DAY_OF_YEAR は 202 にセットされます。

DAYS

DAYS 関数は、日付の整数表現を戻します。

引き数は、日付、タイム・スタンプ、または文字ストリングのいずれかの組み込みデータ・タイプの値を戻す式でなければなりません。

式 が文字ストリングである場合は、そのストリングは CLOB であってはならず、値は日付またはタイム・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。日付とタイム・スタンプの有効なストリング表現の形式については、71 ページの 『日付/時刻の値のストリング表現』を参照してください。

この関数の結果は、長整数になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

結果は、0001 年 1 月 1 日から *D* までの日数に 1 を加えたものになります (*D* は、同じ引き数を DATE 関数に与えた場合に戻される日付です)。

例

- 表 PROJECT を使用して、プロジェクト (PROJNO) 'IF2000' の終了までに経過する予想日数 (PRENDATE - PRSTDATE) を、ホスト変数 EDUCATION_DAYS (INTEGER) にセットします。

```
SELECT DAYS(PRENDATE) - DAYS(PRSTDATE)
  INTO :EDUCATION_DAYS
  FROM PROJECT
  WHERE PROJNO = 'IF2000'
```

  結果として、EDUCATION_DAYS は 396 にセットされます。

- 表 PROJECT を使用して、部門 (DEPTNO) 'E21' のすべてのプロジェクトについて、予想される経過日数 (PRENDATE - PRSTDATE) の合計を、ホスト変数 TOTAL_DAYS (INTEGER) にセットします。

```
SELECT SUM(DAYS(PRENDATE) - DAYS(PRSTDATE))
  INTO :TOTAL_DAYS
  FROM PROJECT
  WHERE DEPTNO = 'E21'
```

  結果として、TOTAL_DAYS は 1484 にセットされます。

DBCLOB

DBCLOB 関数は、ストリング式の DBCLOB 表現を戻します。

注: 2 バイト文字ラージ・オブジェクト値を戻すには、CAST 式も使用できます。詳しくは、149 ページの『CAST の指定』を参照してください。

ストリング式
文字ストリングまたはグラフィック・ストリングの値を戻す式。 BLOB であってはなりません。CHAR または VARCHAR ビット・データであってはなりません。3 番目の引き数が指定されない限り、CCSID が 65535 の GRAPHIC または VARGRAPHIC であってはなりません。

長さ
結果の可変長文字ストリングの長さ属性を指定します。値は 1 ～ 1 073 741 823 でなければなりません。

2 番目の引き数が指定されないか DEFAULT が指定された場合は、次のようになります。

- 式 が空ストリング定数の場合は、結果の長さ属性は 1。
- 空ストリング定数でない場合は、結果の長さ属性は、最初の引き数の長さ属性と同じ。

整数
結果の可変長グラフィック・ストリングの長さ属性を指定します。これは DBCS または UCS-2 CCSID でなければなりません。CCSID が 65535 であることはできません。

以下の規則では、S は次のいずれかを指します。

- ストリング式が外部コード化スキームのデータを含むホスト変数である場合は、データを固有コード化スキームの CCSID に変換した後の式の結果が S。 (詳細については、34 ページの『文字変換』の項を参照してください。)
- ストリング式が固有コード化スキームのデータである場合は、そのストリング式が S。

3 番目の引き数の指定がなく、最初の引き数が文字である場合は、結果の CCSID は混合 CCSID によって決まります。M でその混合 CCSID を示すことにします。M は次のように決まります。

- S の CCSID が混合 CCSID である場合は、M はその CCSID になる。
- S の CCSID が SBCS CCSID である場合 :
  - S の CCSID が関連する混合 CCSID をもつ場合は、M はその CCSID になる。
  - それ以外の場合は、演算ができない。

次の表には、M をもとにした結果の CCSID を要約してあります。

M	結果の CCSID	説明	DBCS 置換文字
930	300	日本語 EBCDIC	X'FEFE'
933	834	韓国語 EBCDIC	X'FEFE'
935	837	中国語 (簡体字) EBCDIC	X'FEFE'
937	835	中国語 (繁体字) EBCDIC	X'FEFE'
939	300	日本語 EBCDIC	X'FEFE'
5026	4396	日本語 EBCDIC	X'FEFE'
5035	4396	日本語 EBCDIC	X'FEFE'

3 番目の引き数の指定がなく、最初の引き数が文字でない場合は、結果の CCSID は最初の引き数の CCSID と同じになります。

結果の実際の長さは、結果の長さ属性と式 の実際の長さのいずれか小さい方となります。結果の DBCLOB の長さ属性が最初の引き数の実際の長さより小さい場合は、切り捨てが行われ、警告は出ません。

この関数の結果は、DBCLOB ストリングになります。式がヌルである可能性がある場合は、結果もヌルになる可能性があります。式がヌルである場合は、結果はヌル値です。式が空ストリング、または EBCDIC ストリング `X'0E0F'` である場合は、結果は空ストリングになります。

結果が DBCS グラフィック・データの場合は、SBCS と DBCS が等価になるかどうかは M によって決まります。CCSID に関係なく、引き数の中の 2 バイトのコード・ポイントはすべて DBCS 文字と見なされ、引き数の中の 1 バイトのコード・ポイントはすべて SBCS 文字と見なされます (ただし、EBCDIC 混合データのシフト・コード `X'0E'` および `X'0F'` は例外)。

- 引き数の n 番目の文字が DBCS 文字である場合は、結果の n 番目の文字はその DBCS になる。
- 引き数の n 番目の文字が、等価の DBCS 文字をもつ SBCS 文字である場合は、結果の n 番目の文字は、その等価の DBCS 文字になる。
- 引き数の n 番目の文字が、等価の DBCS 文字をもたない SBCS 文字である場合は、結果の n 番目の文字は、DBCS 置換文字になる。

結果が UCS-2 グラフィック・データである場合は、引き数の各文字ごとに結果の 1 文字が決まります。結果の n 番目の文字は、引き数の n 番目の文字と等価の UCS-2 文字になります。

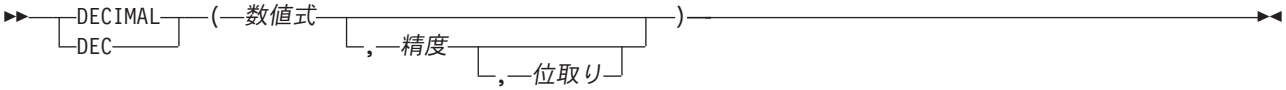
例

- 表 EMPLOYEE を使用して、ホスト変数 VAR_DESC (VARGRAPHIC(24)) を従業員番号 (EMPNO) '000050' に対応する氏名の名 (FIRSTNME) と等価の DBCLOB にセットします。

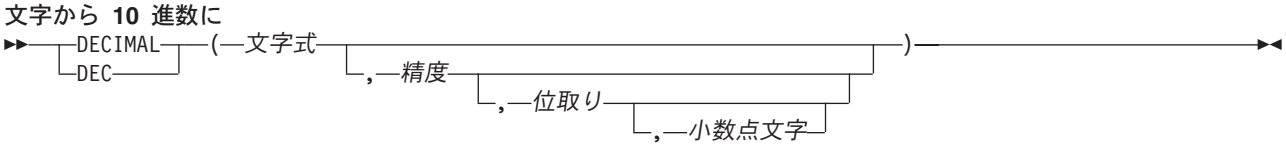
```
SELECT DBCLOB(FIRSTNME)
  INTO :VAR_DESC
  FROM EMPLOYEE
  WHERE EMPNO = '000050'
```

DECIMAL または DEC

数値から 10 進数に



文字から 10 進数に



DECIMAL 関数は、次のものの 10 進数表現を戻します。

- 数値
- 10 進数の文字ストリング表現
- 整数の文字ストリング表現
- 浮動小数点数の文字ストリング表現

注: 10 進数値を戻すには、CAST 式も使用できます。詳しくは、 149 ページの『CAST の指定』を参照してください。

関数の結果は、精度が *p*、位取りが *s* の 10 進数 (*p* は 2 番目の引き数、*s* は 3 番目の引き数) になります。最初の引き数がヌルである可能性がある場合は、結果もヌルになる可能性があります。最初の引き数がヌルの場合は、結果はヌル値になります。

数値から 10 進数に

数値式
任意の組み込み数値データ・タイプの値を戻す式。

精度
1 以上で 31 以下の値を持つ整数定数。

デフォルトの*精度* は、*数値式* のデータ・タイプによって決まります。

- 15 (最初の引き数が浮動小数点数、10 進数、数字、または位取りがゼロ以外の 2 進数の場合)
- 19 (最初の引き数が 64 ビット整数の場合)
- 11 (最初の引き数が長整数の場合)
- 5 (最初の引き数が短整数の場合)

位取り
0 以上で*精度* 以下である整数定数。これを指定しないと、デフォルト値の 0 になります。

結果は、最初の引き数が、精度 *p* で位取り *s* の 10 進数の列または変数に割り当てられた場合に生じる数値と同じになります。数値の整数部を表すのに必要な有効桁数が *p -s* より大きい場合は、エラーになります。

文字から 10 進数に

文字式
数値の文字ストリング表現を含む式。先行ブランクと後書きブランクは除去され、結果のストリングは、整数または 10 進数の定数を形成する規則に合致している必要があります。この式は CLOB であってはなりません。

精度
1 以上で 31 以下である整数定数。これを指定しないと、デフォルト値の 15 になります。

位取り
0 以上で*精度* 以下である整数定数。これを指定しないと、デフォルト値の 0 になります。

小数点文字
数値の整数部分から文字式 の小数桁数を区切るために使用された 1 バイトの文字定数を指定します。この文字は、ピリオドかコンマとする必要があります。 2 番目の引き数を指定しなかった場合は、デフォルトの小数点が使用されます。詳しくは、 108 ページの『小数点』を参照してください。

結果は、CAST(文字式 AS DECIMAL(*p*,*s*)) によって得られる数と同じです。小数点の右側の桁数が位取り *s* より大きい場合は、末尾側から桁が切り捨てられます。文字式 における小数点の左側の有効桁数 (整数部分) が *p-s* より大きい場合は、エラーになります。小数点文字 引き数の指定がある場合は、サブストリング内のデフォルト小数点文字は無効です。

例

- この例では、DECIMAL 関数を使用して表 EMPLOYEE の列 EDLEVEL (データ・タイプ = SMALLINT) に関する選択リストで DECIMAL データ・タイプ (精度が 5 で、位取りが 2) が戻されるようにしています。選択リストには、列 EMPNO も必要です。

```
SELECT EMPNO, DECIMAL(EDLEVEL,5,2)
  FROM EMPLOYEE
```

- 表 PROJECT を使用して、ホスト変数に指定されている期間だけ延ばされている開始日付 (PRSTDATE) を、すべて選択しています。ホスト変数 PERIOD は INTEGER タイプであると想定します。 PERIOD の値を日付期間として使用するためには、PERIOD を「キャスト」して DECIMAL(8,0) にする必要があります。

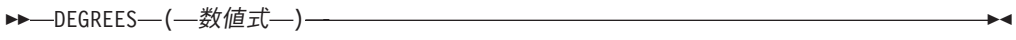
```
SELECT PRSTDATE + DECIMAL(:PERIOD,8)
  FROM PROJECT
```

- SALARY 列への更新が、小数点文字をコンマとして、文字ストリングでウィンドウから入力される (例えば、ユーザーが 21400,50 と入力する) とします。アプリケーションで妥当性検査を受けた後、この値は CHAR(10) と定義されているホスト変数 newsalary に割り当てられます。

```
UPDATE STAFF
  SET SALARY = DECIMAL(:newsalary, 9, 2, ',')
  WHERE ID = :empid
```

SALARY の値は、これで 21400.50 になります。

DEGREES



DEGREES 関数は、引き数の度数をラジアンで表した角度で戻します。

引き数は、任意の組み込み数値データ・タイプの値を戻す式です。

結果のデータ・タイプは、倍精度の浮動小数点数になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

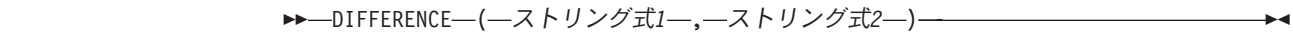
例

- ホスト変数 RAD は、値が 3.142 の DECIMAL(4,3) のホスト変数であると想定します。

```
SELECT DEGREES(:RAD)
  FROM SYSIBM.SYSDUMMY1
```

  およそ 180.0 の値が戻されます。

DIFFERENCE



DIFFERENCE 関数は、ストリングに SOUNDEX 関数を適用し、2 つのストリングの音の相違を表す 0 ～ 4 の値を戻します。値 4 が、音が一致する可能性が最も高くなります。

引き数は、BLOB、CLOB、および DBCLOB を除くいずれかの組み込みデータ・タイプでなければなりません。

結果のデータ・タイプは、INTEGER です。関数のいずれかの引き数がヌルである可能性がある場合は、結果もヌルである可能性があります。いずれかの引き数がヌルである場合は、結果はヌル値になります。

例

- 次のステートメントで、

```
SELECT DIFFERENCE('CONSTRAINT','CONSTANT'),
       SOUNDEX('CONSTRAINT'),
       SOUNDEX('CONSTANT')
  FROM SYSIBM.SYSDUMMY1
```

  4、C523、および C523 が戻されたとします。2 つのストリングが同じ SOUNDEX 値を戻しているので、相違は 4 (可能な最高値) になります。

- 次のステートメントで、

```
SELECT DIFFERENCE('CONSTRAINT','CONTRITE'),
       SOUNDEX('CONSTRAINT'),
       SOUNDEX('CONTRITE')
  FROM SYSIBM.SYSDUMMY1
```

  2、C523、C536 が戻されたとします。この場合、2 つのストリングが異なる SOUNDEX 値を戻しているので、相違値は低くなります。

DIGITS

DIGITS 関数は、数値の絶対値の文字ストリング表現を戻します。

引き数は、SMALLINT、INTEGER、BIGINT、または DECIMAL のいずれかの組み込みデータ・タイプでなければなりません。

引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

この関数の結果は、その位取りには関係なく、引き数の絶対値を表現する固定長の文字ストリングになります。結果には、符号や小数点は含まれません。文字ストリングは数字だけから構成され、必要に応じてストリングは、先行ゼロによって埋め込まれます。ストリングの長さは、次のとおりです。

- 5 (引き数が位取りゼロの短整数の場合)
- 10 (引き数が位取りゼロの長整数の場合)
- 19 (引き数が 64 ビット整数の場合)
- *p* (引き数が 10 進数または位取りがゼロ以外の整数で、精度が *p* の場合)

文字ストリングの CCSID は、現行サーバーにおけるデフォルト SBCS CCSID です。

例

- 表 TABLEX に 10 桁の整数の数値を含む列 INTCOL があるものと想定します。次の例は、列 INTCOL に入っている数値の最初の 4 桁の数字の組み合わせすべてをリストしています。

```
SELECT DISTINCT SUBSTR(DIGITS(INTCOL),1,4)
  FROM TABLEX
```

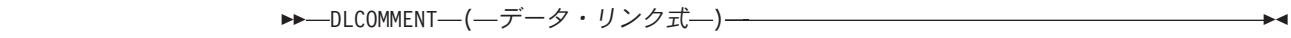
- COLUMNX が DECIMAL(6,2) のデータ・タイプを持ち、その値の 1 つが -6.28 であると想定します。

```
SELECT DIGITS(COLUMNX)
  FROM TABLEX
```

  値として '000628' が戻されます。

  結果は、長さ 6 (該当の列の精度) のストリングになります。この長さになるように先行ゼロが埋め込まれます。符号も小数点も、結果には含まれません。

DLCOMMENT



DLCOMMENT 関数は、データ・リンク値からコメント値を (それが存在する場合) を戻します。

引き数は、結果が DataLink 組み込みデータ・タイプになる式でなければなりません。

引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

この関数の結果は VARCHAR(254) になります。

文字ストリングの CCSID は、データ・リンク式 のものと同じになります。

例

- HOCKEY_GOALS 表の ARTICLES 列へのリンクから、日付、記述、およびコメントを選択するステートメントを準備します。選択する行は、Richard 兄弟 (Maurice か Henri) のいずれかが点を入れたゴールの行です。

```
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 のある行の列 COLA に挿入されているデータ・リンク値があるとします。

```
INSERT INTO TBLA
  VALUES (DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','A comment'))
```

次の関数がこの値に対して実行されると、

```
SELECT DLCOMMENT(COLA)
  FROM TBLA
```

'A comment' という値が戻されます。

DLLINKTYPE

DLLINKTYPE 関数は、データ・リンク値からリンク・タイプ値を戻します。

引き数は、結果が DataLink 組み込みデータ・タイプになる式でなければなりません。

引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

この関数の結果は VARCHAR(4) になります。

文字ストリングの CCSID は、データ・リンク式 のものと同じになります。

例

- スカラー関数を使用して表 TBLA のある行の列 COLA に挿入されているデータ・リンク値があるとします。

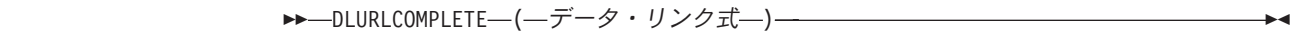
```
INSERT INTO TABLA
  VALUES( DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','A comment') )
```

  次の関数がこの値に対して実行されると、

```
SELECT DLLINKTYPE(COLA)
  FROM TBLA
```

  'URL' という値が戻されます。

DLURLCOMPLETE



DLURLCOMPLETE 関数は、リンク・タイプ URL のデータ・リンク値から完全な URL 値を戻します。この値は、DLURLSCHEME を '://' と、次に DLURLSERVER と、さらに DLURLPATH と連結した結果と同じになります。データ・リンクの属性が FILE LINK CONTROL で、しかも READ PERMISSION DB である場合、値にはファイル・アクセス・トークンが含まれます。

引き数は、結果が DataLink 組み込みデータ・タイプになる式でなければなりません。

引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

この関数の結果は可変長ストリングです。属性の長さは、データ・リンクの属性によって次のように異なります。

- データ・リンクの属性が FILE LINK CONTROL でかつ READ PERMISSION DB の場合は、結果の長さ属性は引き数の長さ属性に 19 を加えたもの
- それ以外の場合は、結果の長さ属性は、引き数の長さ属性

データ・リンク値にコメントしか含まれていない場合は、戻る結果は長さゼロのストリングです。

文字ストリングの CCSID は、データ・リンク式 のものと同じになります。

例

- スカラー関数を使用して表 TBLA のある行の列 COLA に挿入されているデータ・リンク値があるとします。

```
INSERT INTO TABLA
  VALUES( DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','A comment') )
```

次の関数がこの値に対して実行されると、

```
SELECT DLURLCOMPLETE(COLA)
  FROM TBLA
```

'HTTP://DLFS.ALMADEN.IBM.COM/x/y/**********;a.b' という値が戻されます。********** はアクセス・トークンを表します。

DLURLPATH

DLURLPATH 関数は、リンク・タイプ URL のデータ・リンク値から、あるサーバー内のファイルにアクセスするのに必要なパスとファイル名を戻します。該当する場合は、この値にはファイル・アクセス・トークンが含まれます。

引き数は、結果が DataLink 組み込みデータ・タイプになる式でなければなりません。

引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

この関数の結果は可変長ストリングです。属性の長さは、データ・リンクの属性によって次のように異なります。

- データ・リンクの属性が FILE LINK CONTROL でかつ READ PERMISSION DB の場合は、結果の長さ属性は引き数の長さ属性に 19 を加えたもの
- それ以外の場合は、結果の長さ属性は、引き数の長さ属性

データ・リンク値にコメントしか含まれていない場合は、戻る結果は長さゼロのストリングです。

文字ストリングの CCSID は、データ・リンク式 のものと同じになります。

例

- スカラー関数を使用して表 TBLA のある行の列 COLA に挿入されているデータ・リンク値があるとします。

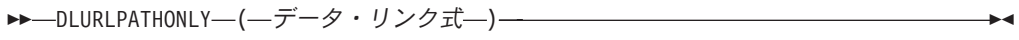
```
INSERT INTO TABLA
  VALUES( DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','A comment') )
```

次の関数がこの値に対して実行されると、

```
SELECT DLURLPATH(COLA)
  FROM TBLA
```

'/x/y/**********;a.b' という値が戻されます。 ********** はアクセス・トークンを表します。

DLURLPATHONLY



DLURLPATHONLY 関数は、リンク・タイプ URL のデータ・リンク値から、あるサーバー内のファイルにアクセスするのに必要なパスとファイル名を戻します。戻される値には、ファイル・アクセス・トークンは含まれていません。

引き数は、結果が DataLink 組み込みデータ・タイプになる式でなければなりません。

引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

この関数の結果は、引き数の長さ属性に等しい長さ属性を持つ可変長ストリングになります。

データ・リンク値にコメントしか含まれていない場合は、戻る結果は長さゼロのストリングです。

文字ストリングの CCSID は、データ・リンク式 のものと同じになります。

例

- スカラー関数を使用して表 TBLA のある行の列 COLA に挿入されているデータ・リンク値があるとします。

```
INSERT INTO TABLA
  VALUES( DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','A comment') )
```

  次の関数がこの値に対して実行されると、

```
SELECT DLURLPATHONLY(COLA)
  FROM TBLA
```

  '/x/y/a.b' という値が戻されます。

DLURLSCHEME

DLURLSCHEME 関数は、リンク・タイプ URL のデータ・リンク値からそのスキームを戻します。この値は常に大文字です。

引き数は、結果が DataLink 組み込みデータ・タイプになる式でなければなりません。

引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

この関数の結果は VARCHAR(20) になります。

データ・リンク値にコメントしか含まれていない場合は、戻る結果は長さゼロのストリングです。

文字ストリングの CCSID は、データ・リンク式 のものと同じになります。

例

- スカラー関数を使用して表 TBLA のある行の列 COLA に挿入されているデータ・リンク値があるとします。

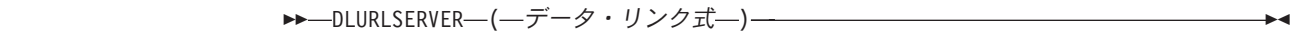
```
INSERT INTO TABLA
  VALUES( DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','A comment') )
```

  次の関数がこの値に対して実行されると、

```
SELECT DLURLSCHEME(COLA)
  FROM TBLA
```

  'HTTP' という値が戻されます。

DLURLSERVER



DLURLSERVER 関数は、リンク・タイプ URL のデータ・リンク値から、ファイル・サーバーを戻します。この値は常に大文字です。

引き数は、結果が DataLink 組み込みデータ・タイプになる式でなければなりません。

引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

この関数の結果は、引き数の長さ属性に等しい長さ属性を持つ可変長ストリングになります。

データ・リンク値にコメントしか含まれていない場合は、戻る結果は長さゼロのストリングです。

文字ストリングの CCSID は、データ・リンク式 のものと同じになります。

例

- スカラー関数を使用して表 TBLA のある行の列 COLA に挿入されているデータ・リンク値があるとします。

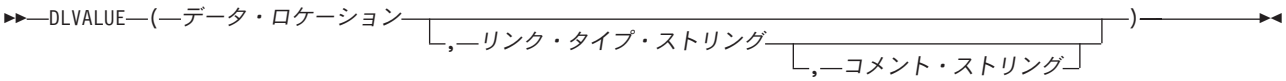
```
INSERT INTO TABLA
  VALUES( DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','A comment') )
```

  次の関数がこの値に対して実行されると、

```
SELECT DLURLSERVER(COLA)
  FROM TBLA
```

  'DLFS.ALMADEN.IBM.COM' という値が戻されます。

DLVALUE



DLVALUE 関数は、データ・リンク値を戻します。この関数を UPDATE ステートメントの SET 文節の右側または INSERT ステートメントの VALUES 文節で使用した場合は、通常はファイルに対するリンクも作成されます。ただし、コメントだけを指定すると (この場合は、*データ・ロケーション* は長さゼロのストリング)、データ・リンク値は空のリンク属性を使用して作成され、したがってファイル・リンクにはなりません。

データ・ロケーション
リンク・タイプが URL の場合は、これは完全な URL 値を含む文字ストリング式です。式が空ストリングではない場合は、この中に URL スキームと URL サーバーを入れる必要があります。文字ストリング式の実際の長さは、32718 文字以下でなければなりません。

リンク・タイプ・ストリング
データ・リンク値のリンク・タイプを指定する任意選択の文字ストリング式。有効な値は 'URL' だけです。

コメント・ストリング
コメント、または追加のロケーション情報を提供する任意選択の文字ストリング式。この文字ストリング式の実際の長さは、254 文字以下でなければなりません。

コメント・ストリング は、ヌル値であってはなりません。*コメント・ストリング* が指定されない場合は、*コメント・ストリング* は空ストリングになります。

最初の引き数がヌルである可能性がある場合は、結果もヌルになる可能性があります。最初の引き数がヌルの場合は、結果はヌル値になります。

この関数の結果は、データ・リンク値になります。

データ・リンクの CCSID は、次の場合を除き、*データ・ロケーション* のものと同じになります。

- *コメント・ストリング* が混合データで、*データ・ロケーション* が混合データではない場合は、結果の CCSID は、*コメント・ストリング* の CCSID になります。[31]
- *データ・ロケーション* が CCSID としてビット・データ (65535)、UCS-2 グラフィック・データ (13488)、トルコ語データ (905 または 1026)、あるいは日本語データ (290、930 または 5026) を持っている場合は、結果の CCSID は次の表のようになります。

31. *コメント・ストリング* の CCSID が 5026 か 930 の場合は、結果の CCSID は 939 になります。

データ・ロケーション の CCSID	*コメント・ストリング* の CCSID	結果の CCSID
65535	65535	ジョブ・デフォルト CCSID
65535	65535 以外	*コメント・ストリング* CCSID (CCSID が 290、930、5026、905、1026 または 13488 の場合を除く。これらの場合は、CCSID は以下に示すように修正される。)
290	任意	4396
930 または 5026	任意	939
905 または 1026	任意	500
13488	任意	500

この関数を使用してデータ・リンク値を定義するときは、値のターゲットの最大長を考慮してください。例えば、DataLink(200) と定義されている列では、*データ・ロケーション* の最大長にコメントを加えたものが 200 バイトになります。

例

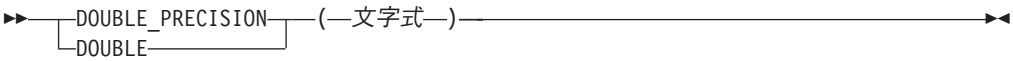
- 表に 1 行を挿入するとします。最初の 2 つのリンクの URL 値は、url_article と url_snapshot という名前の変数に入っています。また、url_snapshot_comment という名前の変数には、スナップショット・リンクに付随するコメントが入っています。ただし、movie のためのリンクはまだありません。url_movie_comment という名前の変数にコメントが入っているだけです。

```
INSERT INTO HOCKEY_GOALS
  VALUES('Maurice Richard',
         'Montreal canadian',
         '?',
         '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_PRECISION または DOUBLE

数値から倍精度へ

文字から倍精度へ



DOUBLE_PRECISION と DOUBLE の関数は、次のものの浮動小数点表現を戻します。

- 数値
- 10 進数の文字ストリング表現
- 整数の文字ストリング表現
- 浮動小数点数の文字ストリング表現

注: 倍精度浮動小数点数値を戻すには、CAST 式も使用できます。詳しくは、 149 ページの『CAST の指定』を参照してください。

数値式
任意の組み込み数値データ・タイプの値を戻す式。

結果は、式が倍精度浮動小数点数の列または変数に割り当てられていた場合に得られるものと同じ数値になります。

文字式
文字ストリング値を戻す式。引き数は CLOB であってはなりません。

結果は、CAST(文字式 AS DOUBLE PRECISION) で得られる数値と同じです。先行ブランクと後書きブランクは除去され、結果のストリングは、浮動小数点数、整数、または 10 進数の定数を形成する規則に合致している必要があります。

この関数の結果は、倍精度浮動小数点数になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

FLOAT は、DOUBLE_PRECISION および DOUBLE の同義語です。

例

- 表 EMPLOYEE を使用して、何らかの手数料を得ている社員について、給与に占める手数料の割合を求めます。給与 (列 SALARY) および手数料 (列 COMM) のデータ・タイプは、DECIMAL (10 進数) です。範囲外の結果が生じる可能性を避けるために、除算が浮動小数点数で行われるように、SALARY に対して DOUBLE-PRECISION が使用されます。

```
SELECT EMPNO, DOUBLE_PRECISION(SALARY)/COMM
  FROM EMPLOYEE
  WHERE COMM > 0
```

EXP

EXP 関数は、自然対数の底 (e) を引き数の指定だけ累乗した値を戻します。EXP 関数と LN 関数は、逆の演算になります。

引き数は、任意の組み込み数値データ・タイプの値を戻す式です。

結果のデータ・タイプは、倍精度の浮動小数点数になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

例

- ホスト変数 E は、値が 3.453789832 の DECIMAL(10、9) ホスト変数であると想定します。

```
SELECT EXP(:E)
  FROM SYSIBM.SYSDUMMY1
```

  およそ 31.62 の値が戻されます。

FLOAT

数値から浮動小数点数に

文字から浮動小数点数に

FLOAT 関数は、数値の浮動小数点数表現を戻します。

FLOAT は、DOUBLE_PRECISION および DOUBLE 関数の同義語です。詳細は、238 ページの 『DOUBLE_PRECISION または DOUBLE』を参照してください。

FLOOR

FLOOR 関数は、*数値式* に等しいか数値式より小さい最大の整数を戻します。

引き数は、任意の組み込み数値データ・タイプの値を戻す式です。

この関数の結果のデータ・タイプと長さ属性は引き数と同じになりますが、引き数が 10 進数の場合は位取りは 0 になります。例えば、データ・タイプが DECIMAL(5,5) の引き数の場合、結果は DECIMAL(5,0) となります。

引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

例

- 小数点の右側の桁をすべて切り捨てるために、FLOOR 関数を使用します。

```
SELECT FLOOR(SALARY)
  FROM EMPLOYEE
```

- 正負両方の数値に関して FLOOR を使用します。

```
SELECT FLOOR( 3.5),
       FLOOR( 3.1),
       FLOOR(-3.1),
       FLOOR(-3.5),
  FROM SYSIBM.SYSDUMMY1
```

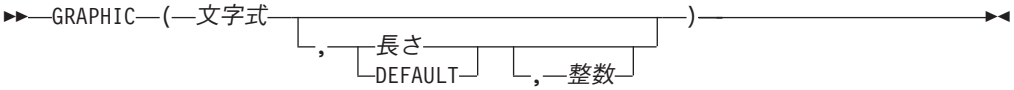
この例ではそれぞれ、

```
3.   3.  -4.  -4.
```

が戻されます。

GRAPHIC

文字からグラフィックに



グラフィックからグラフィックに

GRAPHIC 関数は、ストリング式の固定長グラフィック文字表現を戻します。

注: 固定長グラフィック・ストリング値を戻すには、CAST 式も使用できます。詳しくは、149 ページの『CAST の指定』を参照してください。

この関数の結果は固定長グラフィック・ストリング (GRAPHIC) です。

式がヌルである可能性がある場合は、結果もヌルになる可能性があります。式がヌルである場合は、結果はヌル値です。

文字からグラフィックに

文字式

文字ストリング式を指定します。CHAR または VARCHAR ビット・データであってはなりません。式が空ストリング、または EBCDIC ストリング `X'0E0F'` である場合は、結果は空ストリングになります。

長さ

結果の長さ属性を指定します。これは、最初の引き数がヌル可能でない場合は、1 から 16383 の範囲の整定数でなければならず、最初の引き数がヌル可能である場合は、1 から 16382 の範囲の整定数でなければなりません。文字式 の長さが指定の長さより小さい場合、結果は、結果に指定されている長さまで 2 バイト・ブランクで埋め込まれます。

2 番目の引き数の指定がない場合、または DEFAULT が指定されている場合は、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。

引き数の各文字ごとに、結果の 1 文字が決まります。結果の固定長ストリングの長さ属性が最初の引き数の実際の長さより小さい場合は、切り捨てが行われ、警告が戻されることはありません。

整数

結果の CCSID を指定します。これは DBCS または UCS-2 CCSID でなければなりません。CCSID が 65535 であることはできません。CCSID が UCS-2 グラフィック・データである場合は、引き数の各文字ごとに結果の 1 文字が決まります。結果の n 番目の文字は、引き数の n 番目の文字と等価の UCS-2 文字になります。

整数 が指定されていない場合、結果の CCSID は混合 CCSID によって決まります。M でその混合 CCSID を示すことにします。

以下の規則では、S は次のいずれかを指します。

- ストリング式が外部コード化スキームのデータを含むホスト変数である場合は、データを固有コード化スキームの CCSID に変換した後の式の結果が S。(詳細については、 34 ページの『文字変換』の項を参照してください。)
- ストリング式が固有コード化スキームのデータである場合は、そのストリング式が S。

M は次のように決まります。

- S の CCSID が混合 CCSID である場合は、M はその CCSID になる。
- S の CCSID が SBCS CCSID である場合 :
  - S の CCSID が関連する混合 CCSID をもつ場合は、M はその CCSID になる。
  - それ以外の場合は、演算ができない。

次の表には、M をもとにした結果の CCSID を要約してあります。

M	結果の CCSID	説明	DBCS 置換文字
930	300	日本語 EBCDIC	X'FEFE'
933	834	韓国語 EBCDIC	X'FEFE'
935	837	中国語 (簡体字) EBCDIC	X'FEFE'
937	835	中国語 (繁体字) EBCDIC	X'FEFE'
939	300	日本語 EBCDIC	X'FEFE'
5026	4396	日本語 EBCDIC	X'FEFE'
5035	4396	日本語 EBCDIC	X'FEFE'

SBCS と DBCS が等価になるかどうかは M によって決まります。CCSID に関係なく、引き数の中の 2 バイトのコード・ポイントはすべて DBCS 文字と見なされ、引き数の中の 1 バイトのコード・ポイントはすべて SBCS 文字と見なされます (ただし、EBCDIC 混合データのシフト・コード `X'0E'` および `X'0F'` は例外)。

- 引き数の n 番目の文字が DBCS 文字である場合は、結果の n 番目の文字はその DBCS になる。
- 引き数の n 番目の文字が、等価の DBCS 文字をもつ SBCS 文字である場合は、結果の n 番目の文字は、その等価の DBCS 文字になる。
- 引き数の n 番目の文字が、等価の DBCS 文字をもたない SBCS 文字である場合は、結果の n 番目の文字は、DBCS 置換文字になる。

グラフィックからグラフィックに

グラフィック式
グラフィック・ストリング式を指定します。

長さ
結果の長さ属性を指定します。これは、最初の引き数がヌル可能でない場合は、1 から 16383 の範囲の整定数でなければならず、最初の引き数がヌル可能であ

る場合は、1 から 16382 の範囲の整定数でなければなりません。グラフィック式 の長さが指定の長さより小さい場合、結果は、結果に指定されている長さまで 2 バイト・ブランクで埋め込まれます。

2 番目の引き数の指定がない場合、または DEFAULT が指定されている場合は、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。

グラフィック式 の長さが結果の長さ属性よりも大きい場合は、切り捨てが行われます。 切り捨てられた文字がすべてブランクであった場合以外は、警告 (SQLSTATE 01004) が戻されます。

整数

結果の CCSID を指定します。これは DBCS または UCS-2 CCSID でなければなりません。CCSID が 65535 であることはできません。

整数 が指定されていない場合、結果の CCSID は、最初の引き数の CCSID になります。

例

- 表 EMPLOYEE を使用して、ホスト変数 DESC (GRAPHIC(24)) を従業員番号 (EMPNO) '000050' に対応する氏名の名 (FIRSTNME) と等価の GRAPHIC にセットします。

```
SELECT GRAPHIC( VARGRAPHIC(FIRSTNME))
  INTO :DESC
  FROM EMPLOYEE
  WHERE EMPNO = '000050'
```

HASH

HASH 関数は、値の集合の区画番号を戻します。PARTITION 関数の説明も参照してください。区画番号の詳細については、「DB2 UDB for iSeries マルチ・システム」を参照してください。

引き数には、日付、時刻、タイム・スタンプ、浮動小数点数、またはデータ・リンクの値を除く任意の組み込みデータ・タイプを使用できます。

この関数の結果は、0 から 1023 の値の長整数になります。

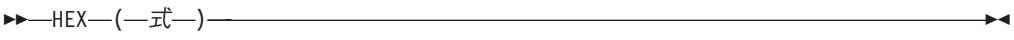
引き数のうちどれかがヌルの場合、その結果はゼロになります。結果がヌルになることはありません。

例

- 区分化キーが EMPNO と LASTNAME から構成されている場合に、HASH 関数を使用して、区画が何であるかを判別します。この照会は、EMPLOYEE の行すべてについての区画番号を戻します。

```
SELECT HASH(EMPNO, LASTNAME)
  FROM EMPLOYEE
```

HEX



HEX 関数は、値の 16 進数表現を戻します。

引き数にはどの組み込みデータ・タイプでも指定できます。

この関数の結果は、文字ストリングになります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

結果は 16 進数字のストリングです。最初の 2 桁が引き数の 1 バイト目を表し、次の 2 桁が引き数の 2 バイト目を表すというように、2 桁一組で引き数の各バイトを順に表します。引き数が日付時刻の値である場合は、結果は引き数の内部形式の 16 進数表現になります。 [32]

引き数が可変長ストリングの場合、結果も可変長ストリングになります。それ以外の場合、結果は固定長ストリングになります。結果の長さ属性は、引き数の記憶域長さ属性の 2 倍になります。記憶域長さ属性の説明については、542 ページの『CREATE TABLE』の項を参照してください。

結果の長さ属性は、最高 32766 まで (結果が固定長の場合)、または最高 32740 まで (結果が可変長の場合) になります。

ストリングの CCSID は、現行サーバーにおけるデフォルト SBCS CCSID です。

例

- HEX 関数を使用して、各従業員の教育レベルを 16 進数表現で戻します。

```
SELECT FIRSTNME, MIDINIT, LASTNAME, HEX(EDLEVEL)
  FROM EMPLOYEE
```

32. DATE、TIMESTAMP、および NUMERIC のデータ・タイプの内部形式は、他のデータベース・プロダクトの場合とは異なるため、これらのデータ・タイプの 16 進数表現も他のデータベース・プロダクトの場合とは異なります。

HOUR

HOUR 関数は、指定した値の時の部分を戻します。

引き数は、時刻、タイム・スタンプ、文字ストリング、または数値データ・タイプのいずれかの組み込みデータ・タイプの値を戻す式でなければなりません。

- 式 が文字ストリングである場合は、そのストリングは CLOB であってはならず、値は時刻またはタイム・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。時刻とタイム・スタンプの有効なストリング表現の形式については、71 ページの 『日付/時刻の値のストリング表現』を参照してください。
- 式 が数値である場合は、その数値は時刻期間またはタイム・スタンプ期間でなければなりません。日付時刻期間の有効な形式については、141 ページの 『日付/時刻のオペランドと期間』を参照してください。

この関数の結果は、長整数になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

その他の規則は、引き数のデータ・タイプ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、時刻またはタイム・スタンプの有効な文字ストリング表現である場合 :

  結果は、指定した値の時の部分 (0 から 24 までの整数) 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、指定した値の時の部分 (-99 から 99 までの整数) 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

- サンプル表 CL_SCHED を使用して、午後に始まるクラスをすべて選択します。

```
SELECT *
  FROM CL_SCHED
  WHERE HOUR(STARTING) BETWEEN 12 AND 17
```

IDENTITY_VAL_LOCAL

IDENTITY_VAL_LOCAL は、識別列に割り当てられた最も新しい値を戻す非決定性関数です。

この関数には入力パラメーターはありません。結果値に対する識別列の実際のデータ・タイプに関係なく、結果は DECIMAL(31,0) です。

戻される値は、識別列を含む表を対象とした最も新しい INSERT ステートメントに指定されている表の識別列に割り当てられた値です。 INSERT ステートメントは同じレベルで発行する必要があります。つまり、現在の値は、次に割り当てられる値で置き換えられるまでは、その現在の値が割り当てられたレベルの中でローカルに使用できる状態になっていることが必要です。新しいレベルが開始されるのは、トリガー、関数、またはストアード・プロシージャーが呼び出されたときです。トリガー状態は、それに関連してトリガーされるアクションと同じレベルにあります。

割り当てられる値には、ユーザーが指定する値 (識別列が GENERATED BY DEFAULT と定義されている場合) と、データベース・マネージャーが生成する識別値があります。

結果が、ヌルになることもあります。結果がヌルになるのは、現行の処理レベルにある識別列を含まない表に対して、INSERT ステートメントを発行した場合です。これには、前挿入トリガーまたは後挿入トリガーの中でこの関数を呼び出した場合も含まれます。

以下のステートメントは、IDENTITY_VAL_LOCAL 関数の結果に影響を与えません。

- 識別列を含まない表に対する INSERT ステートメント
- UPDATE ステートメント
- COMMIT ステートメント
- ROLLBACK ステートメント

使用上の注意

以下の注意事項では、幾つかの異なる状況下でこの関数を呼び出した場合の、この関数の動作を説明します。

INSERT ステートメントの VALUES 文節の中でこの関数を呼び出した場合

INSERT ステートメントのターゲット列に値が割り当てられる前に、INSERT ステートメントの中の式が評価されます。したがって、INSERT ステートメントの中で IDENTITY_VAL_LOCAL を呼び出した場合は、使用される値は、前回の INSERT ステートメント以降に識別列に割り当てられた最も新しい値です。以前に、この IDENTITY_VAL_LOCAL 関数の呼び出しと同じレベルで INSERT ステートメントが実行されていない場合は、この関数はヌル値を戻します。

INSERT ステートメントが失敗した後でこの関数を呼び出した場合

識別列を含む表に対する INSERT ステートメントの実行が失敗した後でこの関数を呼び出した場合は、どのような結果が戻されるかは予測できません。戻される値は、失敗した INSERT の前にこの関数が呼び出されていたとすれば、そのときに戻されているものと予想される値になることもあり、また、INSERT が成功していたとすれば戻されていたであろうと予想される値になることもあります。実際に戻される値は障害の発生時点によって決まるため、予測することはできません。

カーソルの SELECT ステートメントの中でこの関数を呼び出した場合

IDENTITY_VAL_LOCAL 関数の結果は非決定性のものなので、カーソルの SELECT ステートメントの中で IDENTITY_VAL_LOCAL 関数を呼び出した場合の結果は、各 FETCH ステートメントごとに異なる場合があります。

挿入トリガーのトリガー条件の中でこの関数を呼び出した場合

挿入トリガーの条件の中で IDENTITY_VAL_LOCAL 関数を呼び出した場合の結果は、ヌル値になります。

挿入トリガーによりトリガーされたアクションの中でこの関数を呼び出した場合

1 つの表について、複数の前挿入トリガーおよび後挿入トリガーが存在することができます。その場合は、各トリガーはそれぞれ個別に処理され、トリガーされたアクションの中で発行された SQL ステートメントが生成する識別値は、IDENTITY_VAL_LOCAL 関数を使用する他のトリガーされたアクションでは使用できません。これは、トリガーされる複数のアクションが概念的に同じレベルで定義されている場合も同じです。

前挿入トリガーのトリガーされたアクションの中では、IDENTITY_VAL_LOCAL 関数を使用しないでください。前挿入トリガーのトリガーされたアクションの中で IDENTITY_VAL_LOCAL 関数を呼び出した場合の結果は、ヌル値になります。トリガーが定義されている表の識別列の値を、前挿入トリガーのトリガーされたアクションの中で IDENTITY_VAL_LOCAL を呼び出すことによって取得することはできません。ただし、トリガーされたアクションで識別列に対するトリガー遷移変数を参照することにより、この識別列の値を取得することができます。

後挿入トリガーのトリガーされたアクションの中で IDENTITY_VAL_LOCAL 関数を呼び出した場合の結果は、識別列を含む表に対する同じトリガーされたアクションの中で呼び出された最新の INSERT ステートメントで指定されている表の識別列に割り当てられている値になります。 IDENTITY_VAL_LOCAL 関数を呼び出す前に、同じトリガーされたアクションの中で、識別列を含む表に対する INSERT ステートメントが実行されていない場合は、この関数はヌル値を戻します。

トリガーされたアクションを伴う INSERT の後でこの関数を呼び出した場合

トリガーを活動化する INSERT の後で関数を呼び出した場合の結果は、実際に識別列に割り当てられている値 (つまり、以後の SELECT ステートメントで戻されることになる値) になります。この値は、必ずしも、INSERT ステートメントで提供される値、またはデータベース・マネージャーが生成する値とは限りません。割り当てられる値は、識別列に関連したトリガー遷移変数に対する前挿入トリガーのトリガーされたアクションの中で、SET 遷移変数ステートメントに指定されている値の場合もあります。

例

- 変数 IVAR を、EMPLOYEE 表の識別列に割り当てられている値にセットします。 VALUES ステートメントの中でこの関数から戻される値は、1 になります。

```
CREATE TABLE EMPLOYEE
  (EMPNO INTEGER GENERATED ALWAYS AS IDENTITY,
   NAME CHAR(30),
   SALARY DECIMAL(5,2),
   DEPT SMALLINT)

INSERT INTO EMPLOYEE
  (NAME, SALARY, DEPTNO)
  VALUES('Rupert', 989.99, 50)

VALUES IDENTITY_VAL_LOCAL() INTO  :IVAR
```

- T1 および T2 という 2 つの表に、C1 という名前の識別列があるとします。データベース・マネージャーは、表 T1 の C1 列については 値 1、2、3 ... を生成し、表 T2 の C1 列については 値 10、11、12 ... を生成します。

```
CREATE TABLE T1
  (C1 SMALLINT GENERATED ALWAYS AS IDENTITY,
   C2 SMALLINT)

CREATE TABLE T2
  (C1 DECIMAL(15,0) GENERATED BY DEFAULT AS IDENTITY ( START WITH 10 ) ,
   C2 SMALLINT)

INSERT INTO T1 ( C2 ) VALUES(5)

INSERT INTO T1 ( C2 ) VALUES(5)

SELECT  * FROM T1
```

C1	C2
1	5
2	5

```
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。以下の INSERT ステートメントは、T2 に 1 つの行を挿入し、その行の列 C2 には、IDENTITY_VAL_LOCAL 関数が戻す 2 の値が入ります。

```
INSERT INTO T2 ( C2 ) VALUES( IDENTITY_VAL_LOCAL() )

SELECT  * FROM T2
  WHERE C1 = DECIMAL( IDENTITY_VAL_LOCAL(), 15, 0)
```

C1	C2
10	2

この INSERT の後で IDENTITY_VAL_LOCAL 関数を呼び出すと、値 10 が戻されます。これは、データベース・マネージャーが T2 の列 C1 用として生成した値です。ここで、T2 にもう 1 つ行を挿入するものとします。以下の INSERT ステートメントでは、データベース・マネージャーは、列 C1 を識別するために値

13 を割り当て、C2 には IDENTITY_VAL_LOCAL から戻された 値 13 を割り当てます。したがって、C2 には、T2 に挿入された最後の識別値が与えられます。

```
INSERT INTO T2 ( C2, C1 ) VALUES( IDENTITY_VAL_LOCAL(), 13 )

SELECT  * FROM T2
  WHERE C1 = DECIMAL( IDENTITY_VAL_LOCAL(), 15, 0)
```

C1	C2
13	10

- IDENTITY_VAL_LOCAL 関数を呼び出すと同時に、識別値に新しい値を割り当てる INSERT ステートメントの中で、IDENTITY_VAL_LOCAL 関数を呼び出すこともできます。この場合、次に戻される値は、INSERT ステートメントの完了後に IDENTITY_VAL_LOCAL 関数が呼び出された時点で決定されます。例えば、以下の表定義について考えてみてください。

  ```
  CREATE TABLE T3
    (C1 SMALLINT GENERATED BY DEFAULT AS IDENTITY,
     C2 SMALLINT)
  ```

  以下の INSERT ステートメントでは、C2 列用の値として 25 を指定しており、データベース・マネージャーは、C1 (識別列) 用の値として 1 を生成します。その結果、次回の IDENTITY_VAL_LOCAL 関数の呼び出しで戻される値として、1 が設定されます。

  ```
  INSERT INTO T3 ( C2 ) VALUES( 25 )
  ```

  以下の INSERT ステートメントでは、IDENTITY_VAL_LOCAL 関数が呼び出されて、C2 列に入れる値を戻します。値 1 (最初の行の C1 列に割り当てられている識別値) が C2 列に割り当てられ、データベース・マネージャーは C1 (識別列) の値として 2 を生成します。その結果、次回の IDENTITY_VAL_LOCAL 関数の呼び出しで戻される値として、2 が設定されます。

  ```
  INSERT INTO T3 ( C2 ) VALUES( IDENTITY_VAL_LOCAL() )
  ```

  以下の INSERT ステートメントでは、再び IDENTITY_VAL_LOCAL 関数が呼び出されて C2 列に入れる値を戻し、ユーザーが C1 (識別列) 用の値として 11 を指定します。値 2 (2 番目の行の C1 列に割り当てられている識別値) が、C2 列に割り当てられます。 C1 には 11 が割り当てられ、次回の IDENTITY_VAL_LOCAL 関数の呼び出しでは 11 の値が戻されます。

  ```
  INSERT INTO T3 ( C2, C1 ) VALUES( IDENTITY_VAL_LOCAL(), 11 )
  ```

  上記の 3 つの INSERT ステートメントの処理が終わると、表 T3 には以下の値が含まれています。

C1	C2
1	25
2	1
11	2

  T3 の内容は、INSERT ステートメントの列の値が割り当てられる前に、VALUES 文節の中の式が評価されることを示しています。したがって、INSERT ステート

メントの VALUES 文節から IDENTITY_VAL_LOCAL 関数を呼び出すと、前の INSERT ステートメントで識別列に割り当てられている最も新しい値が使用されます。

IFNULL

IFNULL 関数は、ヌルでない最初の式の値を戻します。

IFNULL 関数は、2 つの引き数をもつ COALESCE スカラー関数と同等です。詳しくは、206 ページの 『COALESCE』を参照してください。

例

- 表 EMPLOYEE のすべての行から従業員番号 (EMPNO) および給与 (SALARY) を選択するとき、給与が欠落している (つまり、ヌルである) と、値としてゼロを戻します。

```
SELECT EMPNO, IFNULL(SALARY,0)
  FROM EMPLOYEE
```

INTEGER または INT

数値から整数に

文字から整数に

INTEGER 関数は、次のものの整数表現を戻します。

- 数値
- 10 進数の文字ストリング表現
- 整数の文字ストリング表現
- 浮動小数点数の文字ストリング表現

注: 整数値を戻すには、CAST 式も使用することができます。 詳しくは、 149 ページの『CAST の指定』を参照してください。

数値から整数に

数値式
任意の組み込み数値データ・タイプの数値を戻す式。

引き数が*数値式* の場合、結果は、その引き数が長整数の列または変数に割り当てられたときに得られる数値と同じです。引き数の整数部が、整数の値の範囲内にない場合は、エラーが発生します。引き数の小数部は切り捨てられます。

文字から整数に

文字式
文字ストリング値を戻す式。

整数の文字ストリング表現である値を戻す式。この式は CLOB であってはなりません。

結果は、CAST(文字式 AS INTEGER) から得られる数値と同じです。先行ブランクと後書きブランクは除去され、結果のストリングは、浮動小数点数、整数、または 10 進数の定数を形成する規則に合致している必要があります。引き数の整数部が、整数の値の範囲内にない場合は、エラーが発生します。引き数の小数部は切り捨てられます。

この関数の結果は、長整数になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

例

- 表 EMPLOYEE を使用して、給与 (SALARY) を教育レベル (EDLEVEL) で除算した値が入っているリストを選択します。計算で生じた小数部は、すべて切り捨てられます。このリストには、計算で使用した値と従業員番号 (EMPNO) も入れておきます。

```
SELECT INTEGER(SALARY / EDLEVEL), SALARY, EDLEVEL, EMPNO
  FROM EMPLOYEE
```

JULIAN_DAY

JULIAN_DAY 関数は、紀元前 4712 年 1 月 1 日 (ユリウス暦の開始日) から引き数で指定された日付までの日数を表す整数値を戻します。

引き数は、日付、タイム・スタンプ、または、日付またはタイム・スタンプの有効な文字ストリング表現のいずれかの組み込みデータ・タイプの値を戻す式でなければなりません。文字ストリング・データ・タイプの引き数は CLOB であってはなりません。日付とタイム・スタンプの有効なストリング表現の形式については、71 ページの 『日付/時刻の値のストリング表現』を参照してください。

この関数の結果は、長整数になります。結果はヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

例

- サンプル表 EMPLOYEE を使用して、ホスト変数 JDAY を、Christine Haas (EMPNO = '000010') が雇用された日付 (HIREDATE = '1965-01-01') のユリウス日にセットします。

```
SELECT JULIAN_DAY(HIREDATE)
  INTO :JDAY
  FROM EMPLOYEE
  WHERE EMPNO = '000010'
```

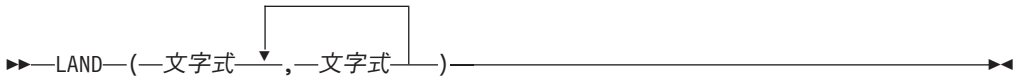
  この結果、JDAY は 2438762 にセットされます。

- 整数ホスト変数 JDAY を、1998 年 1 月 1 日のユリウス日にセットします。

```
SELECT JULIAN_DAY('1998-01-01')
  INTO :JDAY
  FROM SYSIBM.SYSDUMMY1
```

  この結果、JDAY は 2450815 にセットされます。

LAND



LAND 関数は、引き数である 2 つのストリングの論理 'AND' であるストリングを戻します。この関数は、最初の引き数ストリングを取り、次のストリングとの AND 比較を行い、それ以後、次々に前の結果を使用して次の引き数との AND 比較を繰り戻していきます。引き数が前の結果より短い場合は、ブランクが埋め込まれます。

引き数は、文字ストリングでなければなりませんが LOB とすることはできません。引き数は、混合データ文字ストリング、またはグラフィック・ストリングであってはなりません。

必要ならば、引き数は結果の属性に変換されます。結果の属性は、以下のように決められます。

- すべての引き数が固定長ストリングである場合、結果は長さが *n* の固定長ストリングになります (ここで、*n* は最長の引き数の長さ)。
- 引き数の中に可変長ストリングがある場合、結果は長さ属性が *n* の可変長ストリングになります (ここで、*n* は最大の長さ属性を持つ引き数の長さ属性)。結果の実際の長さは *m* です (ここで、*m* は、最長の引き数の実際の長さ)。

引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

結果の CCSID は 65535 です。

例

- ホスト変数 L1 は値が X'A1B1' の CHARACTER(2) のホスト変数、ホスト変数 L2 は値が X'F0F040' の CHARACTER(3) のホスト変数、ホスト変数 L3 は値が X'A1B10040' の CHARACTER(4) のホスト変数であるとします。

  ```
  SELECT LAND(:L1,:L2,:L3)
    FROM SYSIBM.SYSDUMMY1
  ```

  値として X'A0B00000' が戻されます。

- 同様に、

  ```
  SELECT LAND(:L3,:L2,:L1)
    FROM SYSIBM.SYSDUMMY1
  ```

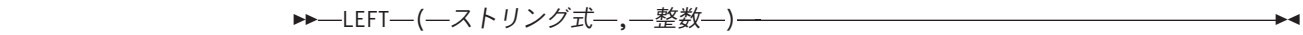
  値として X'A0B00040' が戻されます。この場合は、短い方の引き数にブランク (X'40') を埋め込むため、論理 AND の結果は最初の例とは異なります。

LCASE

LCASE 関数は、すべての文字を引き数の CCSID に基づいて小文字に変換したストリングを戻します。

LCASE 関数は、LOWER 関数と同等です。 詳しくは、 268 ページの『LOWER』を参照してください。

LEFT



LEFT 関数は、*ストリング式* の左端の整数 バイト数を戻します。

ストリング式 が文字ストリングの場合は、結果は文字ストリングで、各文字はそれぞれ 1 バイト文字です。*ストリング式* がグラフィック・ストリングの場合は、結果はグラフィック・ストリングで、各文字はそれぞれ DBCS または UCS-2 文字です。*ストリング式* が 2 進ストリングの場合は、結果は 2 進ストリングで、各文字はそれぞれ 1 バイト文字です。

ストリング式
結果が導き出される元になるストリングを指定する式。*ストリング式* は、文字ストリング、グラフィック・ストリング、または 2 進ストリングのいずれかの組み込みデータ・タイプでなければなりません。

ストリング式 のサブストリングは、*ストリング式* のゼロ個またはそれ以上の連続したバイトです。*ストリング式* がグラフィック・ストリングである場合は、1 文字は DBCS または UCS-2 の 1 文字です。*ストリング式* が文字ストリングまたは 2 進ストリングである場合は、1 文字は 1 バイトです。[33]

整数
結果の長さを指定する式。*整数* は、0 以上で *n* 以下の整数でなければなりません。*n* はストリング式 の長さ属性です。ただし、この整数は整定数の 0 であってはなりません。

ストリング式 は、実際には右側に必要数のブランク文字 (または 2 進ストリングの場合は 16 進数のゼロ) が埋め込まれるため、*ストリング式* の指定されたサブストリングが常に存在しています。

この関数の結果は、*ストリング式* と同じ長さ属性を持つ可変長ストリングで、データ・タイプはストリング式 のデータ・タイプに応じて以下のようになります。

- *ストリング式* が GRAPHIC または VARGRAPHIC である場合は、VARGRAPHIC
- *ストリング式* が CHAR または VARCHAR である場合は、VARCHAR
- *ストリング式* が DBCLOB である場合は、DBCLOB
- *ストリング式* が CLOB である場合は、CLOB
- *ストリング式* が BLOB である場合は、BLOB

整数 が整数定数で、引き数が BLOB、CLOB、または DBCLOB ではない場合は、この関数の結果は固定長ストリングになります。

結果の実際の長さは*整数* です。

33. LEFT 関数は混合データ・ストリングを受け入れます。ただし、LEFT は厳密なバイト・カウントに基づいて演算を行うため、結果は必ずしも適切な形式の混合データ・ストリングにはなりません。

関数のいずれかの引き数がヌルである可能性がある場合は、結果もヌルである可能性があります。いずれかの引き数がヌルである場合は、結果はヌル値になります。

結果の CCSID はストリング式 と同じです。

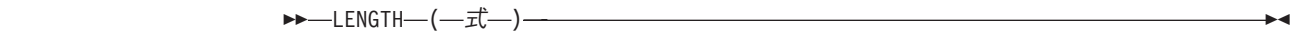
例

- ホスト変数 NAME (VARCHAR(50)) は、'KATIE AUSTIN' という値をもち、ホスト変数 FIRSTNAME_LEN (int) は、5 という値をもつと想定します。

```
SELECT LEFT(:NAME, :FIRSTNAME_LEN)
  FROM SYSIBM.SYSDUMMY1
```

'KATIE' という値が戻されます。

LENGTH



LENGTH 関数は、値の長さを戻します。同様な関数として、 201 ページの『CHARACTER_LENGTH』を参照してください。

引き数は、任意の組み込みデータ・タイプの値を戻す式です。

この関数の結果は、長整数になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

結果は、引き数の長さです。ストリングの長さには、ブランクも含まれます。可変長ストリングの長さは、長さ属性ではなく実際の長さです。

グラフィック・ストリングの長さは、2 バイト文字の数 (バイト数の 2 倍) です。それ以外のすべての値は、値の表現に使用されるバイト数が長さになります。

数値:

- 短整数の場合は 2
- 長整数の場合は 4
- 64 ビット整数の場合は 8
- 精度が *p* のゾーン 10 進数の場合は *p*
- 精度が *p* のパック 10 進数の場合は (*p*/2)+1 の整数部
- 単精度の浮動小数点数の場合は 4
- 倍精度浮動小数点数の場合は 8
- 行 ID の場合は 26

文字ストリング:

- ストリングの長さ

グラフィック・ストリング:

- ストリング内の DBCS または UCS-2 文字の数

日付/時刻の値:

- 時刻の場合は 3
- 日付の場合は 4
- タイム・スタンプの場合は 10

データ・リンク値:

- データ・リンク値を保管するために実際に使用するバイト数 (データ・リンクが FILE LINK CONTROL で、しかも READ PERMISSION DB の場合は、これに 19 を加える)。

例

- ホスト変数 ADDRESS は、値が '895 Don Mills Road' の可変長文字ストリングであると想定します。

```
SELECT LENGTH(:ADDRESS)
  FROM SYSIBM.SYSDUMMY1
```

値 18 が戻されます。

- PRSTDATE が、DATE タイプの列であるとします。

```
SELECT LENGTH(PRSTDATE)
  FROM PROJECT
```

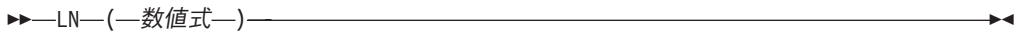
値として 4 が戻されます。

- PRSTDATE が、DATE タイプの列であるとします。

```
SELECT LENGTH(CHAR(PRSTDATE, EUR))
  FROM PROJECT
```

値として 10 が戻されます。

LN



LN 関数は、数値の自然対数を戻します。LN 関数と EXP 関数は、互いに逆の演算になります。

引き数は、任意の組み込み数値データ・タイプの値を戻す式です。引き数の値はゼロより大きくなくてはなりません。

結果のデータ・タイプは、倍精度の浮動小数点数になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

例

- ホスト変数 NATLOG は、値が 31.62 の DECIMAL(4,2) のホスト変数であると想定します。

```
SELECT LN(:NATLOG)
  FROM SYSIBM.SYSDUMMY1
```

  およそ 3.45 の値が戻されます。

LNOT

LNOT 関数は、引き数ストリングの論理否定 (論理 NOT) であるストリングを戻します。

引き数は、文字ストリングでなければなりませんが、LOB とすることはできません。また、引き数には混合文字ストリングやグラフィック・ストリングは使用できません。

結果のデータ・タイプおよび長さ属性は、引き数値のデータ・タイプおよび長さ属性と同じです。引き数が可変長ストリングの場合、結果の実際の長さは引き数値の実際の長さと同じです。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

結果の CCSID は 65535 です。

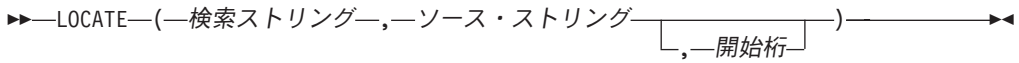
例

- ホスト変数 L1 は、値が X'F0F0' の CHARACTER(2) のホスト変数であると想定します。

```
SELECT LNOT(:L1)
  FROM SYSIBM.SYSDUMMY1
```

  値として X'0F0F' が戻されます。

LOCATE



LOCATE 関数は、ストリング (*ソース・ストリング* と呼ぶ) の中で、あるストリング (*検索ストリング* と呼ぶ) が最初に現れるその開始位置を戻します。*検索ストリング* が検出されないか、どの引き数もヌルの場合は、結果はゼロになります。 *検索ストリング* が検出されると、結果は 1 からソース・ストリング の実際の長さまでの数になります。任意指定の*開始桁* を指定した場合は、ソース・ストリング の中のその文字位置から検索が開始されます。

検索ストリング
: 検索したいストリングを示す式。*検索ストリング* には、文字ストリング、グラフィック・ストリング、または 2 進ストリングの式を指定できます。これは、ソース・ストリング と互換性のあるものでなければなりません。

ソース・ストリング
: 検索を行う相手先のソース・ストリングを指定する式。ソース・ストリング には、文字ストリング、グラフィック・ストリング、または 2 進ストリングの式を指定できます。

開始桁
: ソース・ストリング 内の検索を開始したい位置を指定する式。これは正の整数でなければなりません。

この関数の結果は、長整数になります。引き数のいずれかがヌルになる可能性がある場合は、結果もヌルになる可能性があります。いずれかの引き数がヌルの場合は、結果はヌル値になります。

開始桁 を指定した場合は、この関数は次と同じになります。

```
POSSTR( SUBSTR(source-string,start) , search-string )
```

開始桁 を指定しない場合、この関数は次と同じになります。

```
POSSTR( source-string , search-string )
```

詳細は、286 ページの 『POSITION または POSSTR』を参照してください。

検索ストリング の CCSID がソース・ストリング の CCSID と異なる場合は、ソース・ストリング の CCSID に変換されます。

例

- IN_TRAY 表の全項目から、RECEIVED 列と SUBJECT 列、それに NOTE_TEXT 列の語 'GOOD' の開始位置を選択します。

```
SELECT RECEIVED, SUBJECT, LOCATE('GOOD', NOTE_TEXT)
  FROM IN_TRAY
  WHERE LOCATE('GOOD', NOTE_TEXT) <> 0
```

LOG10

LOG10 関数は、数値の共通対数 (底 10) を戻します。 LOG10 関数と ANTILOG 関数は、逆の演算です。

引き数値には、任意の組み込み数値データ・タイプを使用できます。

結果のデータ・タイプは、倍精度の浮動小数点数になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

LOG は LOG10 の同義語です。これは、DB2 の旧リリースとの互換性を維持するためにのみサポートされています。 データベース・マネージャーおよびアプリケーションによっては、LOG を数値の共通対数ではなく数値の自然対数として設定しているものがあるため、LOG の代わりに LOG10 を使用してください。

例

- ホスト変数 L は、値が 31.62 の DECIMAL(4,2) のホスト変数であると想定します。

```
SELECT LOG10(:L)
  FROM SYSIBM.SYSDUMMY1
```

  およそ 1.49 の値が戻されます。

LOR

LOR 関数は、引き数ストリングの論理和 (論理 OR) のストリングを戻します。この関数は、まず最初の引き数ストリングと次のストリングとの OR 比較を行い、その後次々に得られた結果を使用して次の引き数との OR 比較を行っていきます。引き数が前の結果より短い場合は、ブランクが埋め込まれます。

引き数は、文字ストリングでなければなりませんが LOB とすることはできません。引き数は、混合データ文字ストリング、またはグラフィック・ストリングであってはなりません。

必要ならば、引き数は結果の属性に変換されます。結果の属性は、以下のように決められます。

- すべての引き数が固定長ストリングである場合、結果は長さが *n* の固定長ストリングになります (ここで、*n* は最長の引き数の長さ)。
- 引き数の中に可変長ストリングがある場合、結果は長さ属性が *n* の可変長ストリングになります (ここで、*n* は最大の長さ属性を持つ引き数の長さ属性)。結果の実際の長さは *m* です (ここで、*m* は、最長の引き数の実際の長さ)。

引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

結果の CCSID は 65535 です。

例

- ホスト変数 L1 は値が X'0101' の CHARACTER(2) のホスト変数、ホスト変数 L2 は値が X'F0F000' の CHARACTER(3) のホスト変数、ホスト変数 L3 は値が X'0000000F' の CHARACTER(4) のホスト変数であると想定します。

```
SELECT LOR(:L1,:L2,:L3)
  FROM SYSIBM.SYSDUMMY1
```

  値として X'F1F1000F' が戻されます。

- 同様に、

```
SELECT LOR(:L3,:L2,:L1)
  FROM SYSIBM.SYSDUMMY1
```

  値として X'F1F1404F' が戻されます。この場合、短い方の引き数にブランク (X'40') が埋め込まれるので、論理 OR の結果は最初の例とは異なります。

LOWER

LOWER 関数は、すべての文字を引き数の CCSID に基づいて小文字に変換したストリングを戻します。 SBCS および UCS-2 グラフィック文字だけが変換されます。 A ～ Z の文字は a ～ z に変換され、発音記号がある場合はそれぞれの下段シフトに変換されます。この変換に使用する大文字変換表については、iSeries Information Center のグローバリゼーションのトピックの UCS-2 レベル 1 マッピング・テーブルのセクションを参照してください。

ストリング式
変換するストリングを指定する式。ストリング式は、文字ストリングまたは UCS-2 グラフィック・ストリングでなければなりません。

この関数の結果のデータ・タイプ、長さ属性、実際の長さ、および CCSID は、引き数と同じになります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルである場合は、結果はヌル値です。

LCASE は LOWER の同義語です。

例

- ホスト変数 NAME の値に入っている文字を確実に小文字にしたいとします。NAME はデータ・タイプ VARCHAR(30)、値は 'Christine Smith' です。

```
SELECT LOWER(:NAME)
    FROM SYSIBM.SYSDUMMY1
```

結果は、値 'christine smith' です。

LTRIM

LTRIM 関数は、ストリング式の前部からブランクまたは 16 進数ゼロを除去します。[34]

引き数はストリング式でなければなりません。

- 引き数が 2 進ストリングの場合は、先行 16 進ゼロ (X'00') が除去されます。
- 引き数が DBCS グラフィック・ストリングの場合は、先行 DBCS ブランクが除去されます。
- 最初の引き数が UCS-2 グラフィック・ストリングの場合は、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 SBCS ブランクが除去されます。

結果のデータ・タイプは、式 のデータ・タイプによって異なります。

式 のデータ・タイプ	結果のデータ・タイプ
CHAR または VARCHAR	VARCHAR
GRAPHIC または VARGRAPHIC	VARGRAPHIC
BLOB	BLOB
CLOB	CLOB
DBCLOB	DBCLOB

結果の長さ属性は、*ストリング式* の長さ属性と同じになります。結果の実際の長さは、*ストリング式* の長さから、除去したバイト数を引いた長さになります。すべての文字が除去された場合は、結果は空のストリングになります。

最初の引き数がヌルである可能性がある場合は、結果もヌルになる可能性があります。最初の引き数がヌルの場合は、結果はヌル値になります。

結果の CCSID は、指定したストリングの CCSID と同じになります。

例

- ホスト変数 HELLO (CHAR(9)) には、値として ' Hello' が入っていると想定します。

```
SELECT LTRIM(:HELLO)
  FROM SYSIBM.SYSDUMMY1
```

  結果は 'Hello' になります。

34. LTRIM 関数は、STRIP(式,LEADING) と同じ結果を戻します。

MAX

MAX スカラー関数は、値の集合の中の最大値を戻します。

各引き数には、互換性がなければなりません。文字ストリングの引き数は、日付/時刻の値とは互換性がありますが、グラフィック・ストリングとは互換性がありません。引き数をデータ・リンク値とすることはできません。

この関数の結果は、最大の引き数値になります。結果がヌルになる可能性があるのは、少なくとも 1 つの引き数がヌルになる可能性がある場合です。結果がヌル値になるのは、引き数の 1 つがヌルの場合です。必要があれば、選択された引き数が結果の属性に変換されます。結果の属性は、以下のように決められます。

- 引き数に少なくとも 1 つの日付が含まれており、その他の引き数が日付または日付の有効なストリング表現である場合、結果は日付になります。引き数に少なくとも 1 つの時刻が含まれており、その他の引き数が時刻または時刻の有効なストリング表現である場合、結果は時刻になります。引き数に少なくとも 1 つのタイム・スタンプが含まれており、その他の引き数がタイム・スタンプまたはタイム・スタンプの有効なストリング表現である場合、結果はタイム・スタンプになります。
- 引き数がストリングの場合、結果の CCSID は、各引き数を連結した場合に採用される CCSID になります。 103 ページの『ストリングを結合する演算に適用される変換規則』を参照してください。
- すべての引き数が固定長ストリングである場合、結果は長さが *n* の固定長ストリングになります (ここで、*n* は最長の引き数の長さ)。
- 引き数の中に可変長ストリングがある場合、結果は長さ属性が *n* の可変長ストリングになります (ここで、*n* は最大の長さ属性を持つ引き数の長さ属性)。結果の実際の長さは *m* です (ここで、*m* は、最長の引き数の実際の長さ)。
- 引き数が数値である場合、結果のデータ・タイプはその引き数を加えた場合と同じになります。結果が 10 進数の場合は、以下のようになります。
  - 位取りは *s* (*s* は引き数中の最大の位取り) になります。
  - 精度は、31 と *s+n* (*n* は、引き数の中で精度と位取りとの差が最も大きい数値の桁数) のうち、どちらか小さい方になります。
  - 最大の引き数の整数部分を表すのに必要な桁数は、31-*s* 以下でなければなりません。

ステートメントの実行時に *HEX 以外のソート順序が有効で、しかも SBCS、UCS-2、または混合データが含まれている場合は、そのストリングの重み付けされた値が実際の値の代わりに比較されます。値の重み付けは、該当のソート順序に基づいています。

例

- ホスト変数 M1 は値が 5.5 の DECIMAL(2,1) のホスト変数、ホスト変数 M2 は値が 4.5 の DECIMAL(3,1) のホスト変数、ホスト変数 M3 は値が 6.25 の DECIMAL(3,2) のホスト変数であると想定します。

```
SELECT MAX(:M1,:M2,:M3)
  FROM SYSIBM.SYSDUMMY1
```

  値として 6.25 が戻されます。

- ホスト変数 M1 は値 'AA' の CHARACTER(2) のホスト変数、ホスト変数 M2 は値 'AA ' の CHARACTER(3) のホスト変数、ホスト変数 M3 は値 'AA A' の CHARACTER(4) のホスト変数であると想定します。

```
SELECT MAX(:M1,:M2,:M3)
  FROM SYSIBM.SYSDUMMY1
```

  結果として 'AA A' の値が戻されます。

MICROSECOND

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、文字ストリング、または数値データ・タイプのいずれかの組み込みデータ・タイプの値を戻す式でなければなりません。

- 式 が文字ストリングである場合は、そのストリングは CLOB であってはならず、値はタイム・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。タイム・スタンプの有効なストリング表現の形式については、71 ページの 『日付/時刻の値のストリング表現』を参照してください。
- 式 が数値である場合は、その数値はタイム・スタンプ期間でなければなりません。日付時刻期間の有効な形式については、141 ページの 『日付/時刻のオペランドと期間』を参照してください。

この関数の結果は、長整数になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

その他の規則は、引き数のデータ・タイプ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・スタンプの有効な文字ストリング表現である場合 :

  結果は、値のマイクロ秒の部分 (0 から 999999 までの整数) になります。
- 引き数が期間の場合 :

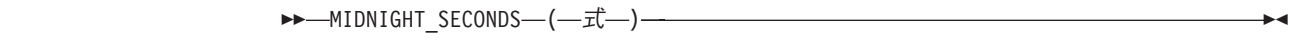
  結果は、値のマイクロ秒の部分 (-999999 から 999999 までの整数) になります。ゼロ以外の結果の符号は、引き数と同じになります。

例

- 表 TABLEA に、TIMESTAMP タイプの列として TS1 と TS2 の 2 つがあると想定します。TS1 のマイクロ秒の部分がゼロ以外で、TS1 と TS2 の秒の部分が一致している行をすべて選択します。

```
SELECT *
  FROM TABLEA
  WHERE MICROSECOND(TS1) <> 0 AND SECOND(TS1) = SECOND(TS2)
```

MIDNIGHT_SECONDS



MIDNIGHT_SECONDS 関数は、真夜中から引き数に指定されている時刻値までの秒数を表す 0 ～ 86 400 の整数値を戻します。

引き数は、時刻、タイム・スタンプ、または、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のいずれかの組み込みデータ・タイプの値を戻す式でなければなりません。文字ストリング・データ・タイプの引き数は CLOB であってはなりません。タイム・スタンプの有効なストリング表現の形式については、71 ページの『日付/時刻の値のストリング表現』を参照してください。

この関数の結果は、長整数になります。 結果はヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

例

- 真夜中から 00:01:00 までの間、および真夜中から 13:10:10 までの間の秒数を調べます。ホスト変数 XTIME1 の値は '00:01:00' であり、XTIME2 の値は '13:10:10' であるものとします。

```
SELECT MIDNIGHT_SECONDS(:XTIME1), MIDNIGHT_SECONDS(:XTIME2)
  FROM SYSIBM.SYSDUMMY1
```

  この例は、60 と 47410 を戻します。1 分は 60 秒、1 時間は 3600 秒なので、00:01:00 は真夜中から 60 秒後 ((60 * 1) + 0)、13:10:10 は 47410 秒後 ((3600 * 13) + (60 * 10) + 10) になります。

- 真夜中から 24:00:00 までの間、および真夜中から 00:00:00 までの間の秒数を調べます。

```
SELECT MIDNIGHT_SECONDS('24:00:00'), MIDNIGHT_SECONDS('00:00:00')
  FROM SYSIBM.SYSDUMMY1
```

  この例は、86400 と 0 を戻します。この 2 つの値は同じ時刻点を表していますが、異なる値が戻されます。

MIN

MIN スカラー関数は、値の集合の中の最小値を戻します。

各引き数には、互換性がなければなりません。文字ストリングの引き数は、日付/時刻の値とは互換性がありますが、グラフィック・ストリングとは互換性がありません。引き数をデータ・リンク値とすることはできません。

この関数の結果は、最小の引き数値になります。結果がヌルになる可能性があるのは、少なくとも 1 つの引き数がヌルになる可能性がある場合です。結果がヌル値になるのは、引き数の 1 つがヌルの場合です。必要があれば、選択された引き数が結果の属性に変換されます。結果の属性は、以下のように決められます。

- 引き数に少なくとも 1 つの日付が含まれており、その他の引き数が日付または日付の有効なストリング表現である場合、結果は日付になります。引き数に少なくとも 1 つの時刻が含まれており、その他の引き数が時刻または時刻の有効なストリング表現である場合、結果は時刻になります。引き数に少なくとも 1 つのタイム・スタンプが含まれており、その他の引き数がタイム・スタンプまたはタイム・スタンプの有効なストリング表現である場合、結果はタイム・スタンプになります。
- 引き数がストリングの場合、結果の CCSID は、各引き数を連結した場合に採用される CCSID になります。 103 ページの『ストリングを結合する演算に適用される変換規則』を参照してください。
- すべての引き数が固定長ストリングである場合、結果は長さが *n* の固定長ストリングになります (ここで、*n* は最長の引き数の長さ)。
- 引き数の中に可変長ストリングがある場合、結果は長さ属性が *n* の可変長ストリングになります (ここで、*n* は最大の長さ属性を持つ引き数の長さ属性)。結果の実際の長さは、*m* (*m* は最小の引き数の実際の長さ) になります。
- 引き数が数値の場合、結果のデータ・タイプは、引き数を加算した結果に採用されるデータ・タイプになります。結果が 10 進数の場合は、以下のようになります。
  - 位取りは *s* (*s* は引き数中の最大の位取り) になります。
  - 精度は、31 と *s*+*n* (*n* は、引き数の中で精度と位取りとの差が最も大きい数値の桁数) のうち、どちらか小さい方になります。
  - 最大の引き数の整数部分を表すのに必要な桁数は、31-*s* 以下でなければなりません。

ステートメントの実行時に *HEX 以外のソート順序が有効で、しかも SBCS、UCS-2、または混合データが含まれている場合は、そのストリングの重み付けされた値が実際の値の代わりに比較されます。値の重み付けは、該当のソート順序に基づいています。

例

- ホスト変数 M1 は値が 5.5 の DECIMAL(2,1) のホスト変数、ホスト変数 M2 は値が 4.5 の DECIMAL(3,1) のホスト変数、ホスト変数 M3 は値が 6.25 の DECIMAL(3,2) のホスト変数であると想定します。

```
SELECT MIN(:M1,:M2,:M3)
  FROM SYSIBM.SYSDUMMY1
```

  値として 4.50 が戻されます。

- ホスト変数 M1 は値 'AA' の CHARACTER(2) のホスト変数、ホスト変数 M2 は値 'AAA' の CHARACTER(3) のホスト変数、ホスト変数 M3 は値 'AAAA' の CHARACTER(4) のホスト変数であると想定します。

```
SELECT MIN(:M1,:M2,:M3)
  FROM SYSIBM.SYSDUMMY1
```

  結果として 'AA ' の値が戻されます。

MINUTE

MINUTE 関数は、指定した値の分の部分を戻します。

引き数は、時刻、タイム・スタンプ、文字ストリング、または数値データ・タイプのいずれかの組み込みデータ・タイプの値を戻す式でなければなりません。

- 式 が文字ストリングである場合は、そのストリングは CLOB であってはならず、値は時刻またはタイム・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。時刻とタイム・スタンプの有効なストリング表現の形式については、71 ページの 『日付/時刻の値のストリング表現』を参照してください。
- 式 が数値である場合は、その数値は時刻期間またはタイム・スタンプ期間でなければなりません。日付時刻期間の有効な形式については、141 ページの 『日付/時刻のオペランドと期間』を参照してください。

この関数の結果は、長整数になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

その他の規則は、引き数のデータ・タイプ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、時刻またはタイム・スタンプの有効な文字ストリング表現である場合 :

  結果は、指定した値の分の部分 (0 から 59 までの整数) 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 :

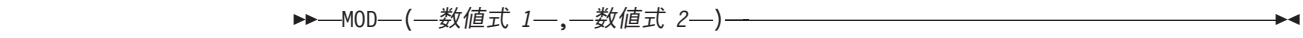
  結果は、指定した値の分の部分 (-99 から 99 までの整数) になります。ゼロ以外の結果の符号は、引き数と同じになります。

例

- サンプル表 CL_SCHED を使用して、期間が 50 分未満のクラスをすべて選択します。

```
SELECT *
  FROM CL_SCHED
  WHERE HOUR(ENDING - STARTING) = 0 AND
      MINUTE(ENDING - STARTING ) < 50
```

MOD



MOD 関数は、最初の引き数を 2 番目の引き数で割って、その剰余を戻します。

剰余の算出には、次の式が使用されます。

```
MOD(x,y) = x - (x/y) * y
```

x/y は、除算の結果を切り捨てた整数です。結果が負の値になるのは、最初の引き数が負の場合だけです。

各引き数は、組み込み数値データ・タイプを戻す式でなければなりません。*数値式 2* はゼロであってはなりません。

引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

結果の属性は、以下のように決められます。

- 両方の引き数が位取りがゼロの長整数または短整数の場合、結果のデータ・タイプは長整数になります。
- 両方の引き数が位取りがゼロの整数であり、少なくとも一方の引き数が 64 ビット整数の場合、結果のデータ・タイプは 64 ビット整数になります。
- 一方の引き数が位取りがゼロの整数で、他方が 10 進数である場合、結果は、10 進数の引き数と同じ精度と位取りの 10 進数になります。
- 両方の引き数が 10 進数または位取りを伴う整数の場合、結果は 10 進数になります。結果の精度は MIN(p-s,p'-s') + MAX(s,s') で、結果の位取りは MAX(s,s') になります。ここで、記号 p と s は第 1 オペランドの精度と位取りを表し、p' と s' は第 2 オペランドの精度と位取りを表します。
- 引き数のいずれかが浮動小数点数である場合、結果のデータ・タイプは倍精度の浮動小数点数になります。

  演算は浮動小数点数で行われます。すなわち、必要なら、オペランドは最初に倍精度浮動小数点数に変換されています。

  浮動小数点数と整数による演算は、倍精度の浮動小数点数に変換されたその整数の一時的なコピーを使用して行われます。浮動小数点数と 10 進数による演算は、倍精度の浮動小数点数に変換されたその 10 進数の一時的なコピーを使用して行われます。浮動小数点数演算の結果は、浮動小数点数の値の範囲内になければなりません。

例

- ホスト変数 M1 は値が 5 の整数のホスト変数であり、ホスト変数 M2 は値が 2 の整数のホスト変数であると想定します。

  ```
  SELECT MOD(:M1,:M2)
    FROM SYSIBM.SYSDUMMY1
  ```

  値として 1 が戻されます。

- ホスト変数 M1 は値が 5 の整数のホスト変数であり、ホスト変数 M2 は値が 2.20 の DECIMAL(3,2) ホスト変数であると想定します。

```
SELECT MOD(:M1,:M2)
  FROM SYSIBM.SYSDUMMY1
```

値として 0.60 が戻されます。

- ホスト変数 M1 は値が 5.50 の DECIMAL(4.2) のホスト変数であり、ホスト変数 M2 は値が 2.0 の DECIMAL(4.1) のホスト変数であると想定します。

```
SELECT MOD(:M1,:M2)
  FROM SYSIBM.SYSDUMMY1
```

値として 1.50 が戻されます。

MONTH

MONTH 関数は、指定した値の月の部分を戻します。

引き数は、日付、タイム・スタンプ、文字ストリング、または数値データ・タイプのいずれかの組み込みデータ・タイプの値を戻す式でなければなりません。

- 式 が文字ストリングである場合は、そのストリングは CLOB であってはならず、値は日付またはタイム・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。日付とタイム・スタンプの有効なストリング表現の形式については、71 ページの 『日付/時刻の値のストリング表現』を参照してください。
- 式 が数値である場合は、その数値は日付期間またはタイム・スタンプ期間でなければなりません。日付時刻期間の有効な形式については、141 ページの 『日付/時刻のオペランドと期間』を参照してください。

この関数の結果は、長整数になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

その他の規則は、引き数のデータ・タイプ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、日付またはタイム・スタンプの有効な文字ストリング表現である場合 :

  結果は、指定した値の月の部分 (1 から 12 までの整数) 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、指定した値の月の部分 (-99 から 99 までの整数) 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

- 表 EMPLOYEE から、誕生日 (BIRTHDATE) が 12 月である社員に関する行をすべて選択します。

```
SELECT *
  FROM EMPLOYEE
  WHERE MONTH(BIRTHDATE) = 12
```

NODENAME

NODENAME 関数は、行が置かれているリレーショナル・データベースの名前を戻します。引き数が非分散表を示している場合は、その CURRENT SERVER 特殊レジスターの値が戻されます。ノードの詳細については、「DB2 マルチ・システム」を参照してください。

引き数は、副選択の表指定子です。表指定子の詳細については、118 ページの 『表指定子』を参照してください。

SQL 命名規則では、表名は修飾できます。システム命名規則では、表名は修飾できません。

引き数がビュー、共通表式、または派生表を示している場合は、この関数は、その基礎表のリレーショナル・データベース名を戻します。引き数が、複数の基礎表から派生したビュー、共通表式、または派生表を示している場合は、この関数は、そのビュー、共通表式、または派生表の外側の副選択内にある最初の表のリレーショナル・データベース名を戻します。

引き数には、外側の副選択に列関数、GROUP BY 文節、HAVING 文節、UNION 文節、または DISTINCT 文節が含まれているようなビュー、共通表式、または派生表を指定してはなりません。その副選択が GROUP BY 文節、または HAVING 文節を含む場合、NODENAME 関数は、WHERE 文節の中か、あるいは列関数のオペランドとしてしか指定することはできません。引き数が相関名である場合は、その相関名が相関参照を示していてはなりません。

この結果のデータ・タイプは、VARCHAR(18) です。結果が、ヌルになることもあります。

結果の CCSID は、現行サーバーのデフォルトの CCSID になります。

例

- EMPLOYEE 表と DEPARTMENT 表を結合し、社員番号 (EMPNO) を選択して、発生した結合に関連する各行からノードを判別します。

```
SELECT EMPNO, NODENAME(X), NODENAME(Y)
  FROM EMPLOYEE X, DEPARTMENT Y
  WHERE X.DEPTNO=Y.DEPTNO
```

NODENUMBER

NODENUMBER 関数は、行のノード番号を戻します。引き数が非分散表を識別している場合、値 0 が戻されます。[35] ノードおよびノード番号の詳細については、「DB2 UDB for iSeries マルチ・システム」を参照してください。

引き数は、副選択の表指定子です。表指定子の詳細については、118 ページの 『表指定子』を参照してください。

SQL 命名規則では、表名は修飾できます。システム命名規則では、表名は修飾できません。

引き数がビュー、共通表式、または派生表を示している場合は、この関数は、その基礎表のノード番号を戻します。引き数が、複数の基礎表から派生したビュー、共通表式、または派生表を示している場合は、この関数は、そのビュー、共通表式、または派生表の外側の副選択内にある最初の表のノード番号を戻します。

引き数には、外側の副選択に列関数、GROUP BY 文節、HAVING 文節、UNION 文節、または DISTINCT 文節が含まれているようなビュー、共通表式、または派生表を指定してはなりません。その副選択が GROUP BY 文節、または HAVING 文節を含む場合、NODENUMBER 関数は、WHERE 文節の中、あるいは列関数のオペランドとしてしか指定することはできません。引き数が相関名である場合は、その相関名が相関参照を示していてはなりません。

結果のデータ・タイプは、長整数です。結果が、ヌルになることもあります。

例

- 表 EMPLOYEE の各行についてのノード番号と従業員名を判別します。 分散表の場合は、その行が存在するノードの番号が戻されます。

```
SELECT NODENUMBER(EMPLOYEE), LASTNAME
  FROM EMPLOYEE
```

35. 引き数が、複数の論理ファイル番号に基づいて DDS が作成した論理ファイルを識別している場合、NODENUMBER は 0 を戻さず、代わりに基になる物理ファイル・メンバーの番号を戻します。

NOW

NOW 関数は、SQL ステートメントが現行サーバーで実行される時点の刻時機構の読み取りに基づくタイム・スタンプを戻します。NOW 関数によって戻される値は、CURRENT TIMESTAMP 特殊レジスターによって戻される値と同じです。この関数が 1 つの SQL ステートメント内で複数回使用される場合、または 1 つのステートメント内で CURDATE または CURTIME スカラー関数、あるいは CURRENT DATE、CURRENT TIME、または CURRENT TIMESTAMP 特殊レジスターとともに使用される場合は、値はすべて 1 回の刻時機構読み取りに基づきます。

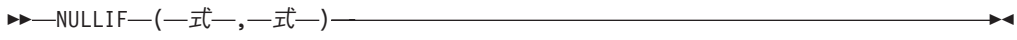
結果のデータ・タイプは、タイム・スタンプになります。結果がヌルになることはありません。

例

- 時点の刻時機構に基づく現在のタイム・スタンプが戻されます。

```
SELECT NOW()
  FROM SYSIBM.SYSDUMMY1
```

NULLIF



NULLIF 関数は、引き数が等しい場合にヌルを戻します。等しくない場合は、最初の引き数の値を戻します。

2 つの引き数は、互換性がありかつ比較可能なデータ・タイプのものでなければなりません。文字ストリングの引き数は、日付/時刻の値と互換性があります。一方のオペランドが特殊タイプである場合は、もう一方のオペランドも同じ 特殊タイプでなければなりません。引き数をデータ・リンク値とすることはできません。

結果の属性は最初の引き数の属性です。結果が、ヌルになることもあります。最初の引き数がヌルか、または両方の引き数が等しい場合は、結果はヌルになります。

NULLIF(e1,e2) を使用した結果は、次の式を使用した結果と同じです。

```
CASE WHEN e1=e2 THEN NULL ELSE e1 END
```

e1=e2 が未知であると評価された (引き数の片方または両方が NULL であったため) 場合は、CASE 式はこれを真ではないと考えます。したがって、この場合は、NULLIF は第 1 オペランドの e1 を戻します。

例

- ホスト変数 PROFIT、CASH および LOSSES は DECIMAL データ・タイプで、値がそれぞれ 4500.00、500.00 および 5000.00 であると想定します。

  ```
  SELECT NULLIF (:PROFIT + :CASH, :LOSSES )
    FROM SYSIBM.SYSDUMMY1
  ```

  結果としてヌル値が戻ります。

PARTITION

PARTITION 関数は、行の区分化キー値にハッシュ関数を適用して取得した、行の区画番号を戻します。HASH 関数の説明も参照してください。引き数が非分散表を示している場合は、値 0 が戻されます。区画番号および区分化キーについての詳細は、「DB2 マルチ・システム」を参照してください。

引き数は、副選択の表指定子です。表指定子の詳細については、118 ページの 『表指定子』を参照してください。

SQL 命名規則では、表名は修飾できます。システム命名規則では、表名は修飾できません。

引き数がビュー、共通表式、または派生表を示している場合は、この関数は、その基礎表の区画番号を戻します。引き数が、複数の基礎表から派生したビュー、共通表式、または派生表を示している場合は、この関数は、そのビュー、共通表式、または派生表の外側の副選択内にある最初の表の区画番号を戻します。

引き数には、外側の副選択に列関数、GROUP BY 文節、HAVING 文節、UNION 文節、または DISTINCT 文節が含まれているようなビュー、共通表式、または派生表を指定してはなりません。その副選択が GROUP BY 文節、または HAVING 文節を含む場合、PARTITION 関数は、WHERE 文節の中、あるいは列関数のオペランドとしてしか指定することはできません。引き数が相関名である場合は、その相関名が相関参照を示していてはなりません。

結果のデータ・タイプは、0 から 1023 の値をもつ長整数になります。結果が、ヌルになることもあります。

例

- 区画番号が 100 である行すべてについて、表 EMPLOYEE から、従業員番号 (EMPNO) を選択します。

```
SELECT EMPNO
  FROM EMPLOYEE
  WHERE PARTITION(EMPLOYEE) = 100
```

PI

PI の値として 3.141592653589793 が戻されます。引き数はありません。

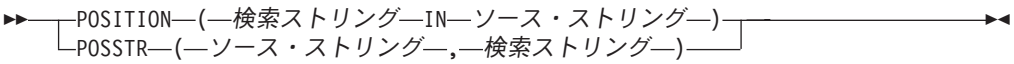
この関数の結果は、倍精度の浮動小数点になります。結果がヌルになることはありません。

例

- 次の関数は、直径 10 の円の円周を戻します。

```
SELECT PI()*10
  FROM SYSIBM.SYSDUMMY1
```

POSITION または POSSTR



POSITION および POSSTR 関数は、あるストリング (ソース・ストリング と呼ぶ) の中で、別のストリング (検索ストリング と呼ぶ) が最初に現れるその開始位置を戻します。 検索ストリング が検出されないか、どの引き数もヌルの場合は、結果はゼロになります。検索ストリング が検出されると、結果は 1 からソース・ストリング の実際の長さまでの数になります。 関連関数については、 265 ページの『LOCATE』を参照してください。

ソース・ストリング
: 検索を行う相手先のソース・ストリングを指定する式。ソース・ストリングは、文字ストリングまたはグラフィック・ストリングの式とすることができます。

検索ストリング
: 検索したいストリングを示す式。検索ストリング は、文字ストリングまたはグラフィック・ストリングの式とすることができます。これは、ソース・ストリング と互換性のあるものでなければなりません。

この関数の結果は、長整数になります。どちらかの引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。どちらかの引き数がヌルの場合は、結果はヌル値になります。

検索ストリング の CCSID がソース・ストリング の CCSID と異なる場合は、ソース・ストリング の CCSID に変換されます。

POSITION 関数は文字単位で実行します。POSSTR 関数は厳密にバイト・カウント単位で実行します。検索ストリング かソース・ストリング の一方が混合データを含んでいる場合は、POSSTR ではなく POSITION を使用してください。POSSTR は厳密にバイト・カウント単位で実行されるため、検索ストリング またはソース・ストリング に混合データが入っている場合は、ソース・ストリング のまったく同じ場所にシフトイン、シフトアウトの文字があった場合だけ、検索ストリング が見つかったことになります。POSITION は文字ストリング単位で実行されるため、シフトイン、シフトアウト文字がまったく同じ場所にある必要がなく、これらの文字は、どの文字が SBCS でどの文字が DBCS であるかを示すためにだけ意味があります。

POSSTR または POSITION 関数を含むステートメントの実行時に *HEX 以外のソート順序が有効で、しかも引き数が SBCS、UCS-2 または混合データを含む場合、結果は、集合の各値の重み付けされた値の比較によって求められます。値の重み付けは、該当のソート順序に基づいています。

検索ストリング の長さがゼロの場合は、この関数が戻す結果は 1 です。その他の場合は、次のようになります。

- ソース・ストリング の長さがゼロの場合は、この関数が戻す結果は 0 です。
- その他の場合は、

- *検索ストリング* の値がソース・ストリング の値の範囲内の連続位置のサブストリングの長さに等しければ、この関数の戻す結果は、ソース・ストリング値内のそのような最初のサブストリングの開始位置です。
- それ以外の場合は、この関数の戻す結果は 0 です。[36]

例

- IN_TRAY 表の全項目から、RECEIVED 列と SUBJECT 列、それに NOTE_TEXT 列の語 'GOOD' の開始位置を選択します。

```
SELECT RECEIVED, SUBJECT, POSSTR(NOTE_TEXT, 'GOOD')
  FROM IN_TRAY
  WHERE POSSTR(NOTE_TEXT, 'GOOD') <> 0
```

36. この中には、*検索ストリング* の方がソース・ストリング よりも長い場合が含まれます。

POWER

POWER 関数は、最初の引き数を 2 番目の引き数だけ累乗した結果を戻します。[37]

各引き数は、任意の組み込み数値データ・タイプの値を戻す式でなければなりません。*数値式 1* の値がゼロである場合は、*数値式 2* はゼロまたはそれより大きい値でなければなりません。引き数がどちらも 0 である場合は、結果は 1 になります。

この関数の結果は、倍精度浮動小数点数になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

例

- ホスト変数 HPOWER は、値が 3 の整数であると想定します。

```
SELECT POWER(2,:HPOWER)
  FROM SYSIBM.SYSDUMMY1
```

  値として 8 が戻されます。

37. POWER 関数の結果は、指数 *数値式 1* ** *数値式 2* の結果とまったく同じです。

QUARTER

QUARTER 関数は、日付が存在する四半期を表す 1 から 4 までの整数を戻します。例えば、1 月、2 月、3 月の日付は、いずれも整数 1 を戻します。

引き数は、日付、タイム・スタンプ、または文字ストリングのいずれかの組み込みデータ・タイプの値を戻す式でなければなりません。 式 が文字ストリングである場合は、そのストリングは CLOB であってはならず、値は日付またはタイム・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。 日付とタイム・スタンプの有効なストリング表現の形式については、71 ページの 『日付/時刻の値のストリング表現』を参照してください。

この関数の結果は、長整数になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

例

- 表 PROJECT を使用して、ホスト変数 QUART (INTEGER) をプロジェクト 'PL2100' が終了した四半期 (PRENDATE) にセットします。

```
SELECT QUARTER(PRENDATE)
  INTO :QUART
  FROM PROJECT
  WHERE PROJNO = 'PL2100'
```

  結果として、QUART は 3 に設定されます。

RADIANS

RADIANS 関数は、度で表された引き数に対してラジアン数を戻します。

引き数は、任意の組み込み数値データ・タイプの値を戻す式です。倍精度の浮動小数点数でない引き数は、関数で処理するために、倍精度の浮動小数点数に変換されます。

結果のデータ・タイプは、倍精度の浮動小数点数になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

例

- ホスト変数 HDEG は、値が 180 の整数であると想定します。次のステートメントは、

```
SELECT RADIANS(:HDEG)
  FROM SYSIBM.SYSDUMMY1
```

  概略値 3.1415926536 の倍精度の浮動小数点数を戻します。

RAND

RAND 関数は、0 と 1 の間の浮動小数点値を戻します。

式が指定されている場合、その式がシード値として使用されます。式は、SMALLINT または INTEGER でなければなりません。

結果のデータ・タイプは、倍精度の浮動小数点数になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

例

- ホスト変数 HRAND は、値が 100 の整数であると想定します。次のステートメントは、

```
SELECT RAND(:HRAND)
  FROM SYSIBM.SYSDUMMY1
```

  0 と 1 の間の乱数の浮動小数点数 (概略値 .0121398 など) を戻します。

- 0 ～ 1 以外の数値間隔の値を生成するには、RAND 関数に必要な間隔の大きさを乗算します。例えば、0 と 10 の間の乱数 (概略値 5.8731398 など) を入手するときは、関数に 10 を乗算します。

```
SELECT RAND(:HRAND) * 10
  FROM SYSIBM.SYSDUMMY1
```

REAL

REAL 関数は、次のものの単精度浮動小数点表現を戻します。

- 数値
- 10 進数の文字ストリング表現
- 整数の文字ストリング表現
- 浮動小数点数の文字ストリング表現

注: 単精度浮動小数点数値を戻すには、CAST 式も使用できます。詳しくは、149 ページの『CAST の指定』を参照してください。

数値式
引き数は、任意の組み込み数値データ・タイプの値を戻す式です。

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ときに得られる数値と同じです。引き数の数値が単精度浮動小数点数の範囲内にない場合は、エラーが起こります。

文字式
文字ストリング値を戻す式。

結果は、CAST(文字式 AS REAL) から得られる数値と同じです。先行ブランクと後書きブランクは除去され、結果のストリングは、浮動小数点数、整数、または 10 進数の定数を形成する規則に合致している必要があります。引き数の数値が単精度浮動小数点数の範囲内にない場合は、エラーが起こります。

この関数の結果は、単精度浮動小数点数になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

例

- 表 EMPLOYEE を使用して、何らかの手数料を得ている社員について、給与に占める手数料の割合を求めます。給与 (列 SALARY) および手数料 (列 COMM) のデータ・タイプは、DECIMAL (10 進数) です。範囲外の結果が生じる可能性を避けるために、除算が浮動小数点数で行われるように、SALARY に対して REAL が使用されます。

```
SELECT EMPNO, REAL(SALARY)/COMM
  FROM EMPLOYEE
  WHERE COMM > 0
```

ROUND

ROUND 関数は、*数値式 1* を、小数点の右側または左側の特定の桁で丸めた値を戻します。

数値式 1
任意の組み込み数値データ・タイプの値を戻す式。

数値式 2
短整数または長整数を戻す式。整数の絶対値は、*数値式 2* が負でなければ、小数点より右の桁数を指定し、*数値式 2* が負であれば、小数点より左の桁数を指定します。

数値式 2 が負でない場合は、*数値式 1* は、小数点の右側の*数値式 2* 桁目で丸められます。その桁の数字が 5 である場合は、次に大きい正の数に切り上げられます。

数値式 2 が負である場合は、*数値式 1* は、小数点の左側の (*数値式 2* + 1) の絶対値に相当する桁で丸められます。その桁の数字が 5 である場合は、次に低い負の数に切り下げられます。*数値式 2* の絶対値が小数点の左側の桁数と同じかそれより大きい場合は、結果は 0 になります。

結果のデータ・タイプおよび長さ属性は、最初の引き数のデータ・タイプおよび長さ属性と同じです。ただし、*数値式 1* が DECIMAL または NUMERIC であり、精度が 31 より小さい場合は、精度は 1 だけ増加します。例えば、データ・タイプが DECIMAL(5,2) の引き数の場合、結果は DECIMAL(6,2) になります。データ・タイプが DECIMAL(31,2) の引き数の場合、結果は DECIMAL(31,2) になります。

どちらかの引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数のどちらかがヌルである場合は、結果はヌル値になります。

例

- 数値 873.726 を、小数点から 2、1、0、-1、-2、-3、-4 の位置で丸めます。

```
SELECT 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)
  FROM SYSIBM.SYSDUMMY1
```

  それぞれ以下の値が戻されます。

```
0873.730   0873.700  0874.000  0870.000  0900.000  1000.000  0000.000
```

- 正負両方の数値を計算します。

```
SELECT ROUND( 3.5, 0),
       ROUND( 3.1, 0),
       ROUND(-3.1, 0),
       ROUND(-3.5, 0)
  FROM SYSIBM.SYSDUMMY1
```

それぞれ以下の値が戻されます。

```
04.0   03.0  -03.0  -04.0
```

ROWID

ROWID 関数は、文字ストリングまたは 2 進ストリングを行 ID にキャストします。

注: 行 ID の値を戻すには、CAST 式も使用できます。詳しくは、149 ページの『CAST の指定』を参照してください。

ストリング式
文字ストリングまたは 2 進ストリングの値を戻す式。*ストリング式* は CLOB であってはなりません。ストリングにはどのような値が含まれていても構いませんが、有効な ROWID 値が戻されるようにするには、DB2 UDB (OS/390 および z/OS 版) または DB2 UDB for iSeries によりすでに生成されている ROWID 値を指定することをお勧めします。例えば、この関数を使用して、CHAR 値にキャストされた ROWID 値を、再び ROWID 値に戻すことができます。

ストリング式 の実際の長さが 40 より小さい場合、結果の埋め込みは行われません。*ストリング式* の実際の長さが 40 より大きい場合は、結果は切り捨てられます。非ブランク文字が切り捨てられた場合は、警告が戻されます。

結果の長さ属性は、40 です。 結果の実際の長さは、*ストリング式* の長さです。

この関数の結果は行 ID です。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

例

- 表 EMPLOYEE に ROWID 列 EMP_ROWID が含まれているとします。また、この表には、X'F0DFD230E3C0D80D81C201AA0A280100000000000203' という行 ID 値で識別される行が含まれているものとします。直接行アクセスを使用して、その行に該当する社員番号を選択します。

```
SELECT EMPNO
  FROM EMPLOYEE
  WHERE EMP_ROWID = ROWID(X'F0DFD230E3C0D80D81C201AA0A280100000000000203')
```

RRN

RRN 関数は、行の相対レコード番号を戻します。

引き数は、副選択の表指定子です。表指定子の詳細については、118 ページの 『表指定子』を参照してください。

SQL 命名規則では、表名は修飾できます。システム命名規則では、表名は修飾できません。

引き数がビュー、共通表式、または派生表を示している場合は、この関数は、その基礎表の相対レコード番号を戻します。引き数が、複数の基礎表から派生したビュー、共通表式、または派生表を示している場合は、この関数は、そのビュー、共通表式、または派生表の外側の副選択内にある最初の表の相対レコード番号を戻します。

引き数が分散表を示している場合、この関数は、その行が位置指定されているノードの行の相対レコード番号を戻します。これは、RRN が分散表の行に固有のものではないということを意味します。

引き数には、外側の副選択に列関数、GROUP BY 文節、HAVING 文節、UNION 文節、または DISTINCT 文節が含まれているようなビュー、共通表式、または派生表を指定してはなりません。副選択が列関数、GROUP BY 文節、または HAVING 文節を含む場合、SELECT 文節に RRN 関数を指定することはできません。引き数が相関名である場合は、その相関名が相関参照を示していてはなりません。

結果のデータ・タイプは、精度が 15 で位取りが 0 の 10 進数です。結果はヌル値の場合もあります。

例

- この例では、表 EMPLOYEE から、部門 20 の社員について、その相対レコード番号と社員名を戻します。

```
SELECT RRN(EMPLOYEE), LASTNAME
  FROM EMPLOYEE
  WHERE DEPTNO = 20
```

RTRIM

RTRIM 関数は、ストリング式の後部からブランクまたは 16 進数ゼロを除去します。[38]

引き数はストリング式でなければなりません。

- 引き数が 2 進ストリングの場合は、後書き 16 進ゼロ (X'00') が除去されます。
- 引き数が DBCS グラフィック・ストリングの場合は、後書き DBCS ブランクが除去されます。
- 最初の引き数が UCS-2 グラフィック・ストリングの場合は、後書き UCS-2 ブランクが除去されます。
- それ以外の場合は、後書き SBCS ブランクが除去されます。

結果のデータ・タイプは、*ストリング式* のデータ・タイプによって異なります。

式 のデータ・タイプ	結果のデータ・タイプ
CHAR または VARCHAR	VARCHAR
GRAPHIC または VARGRAPHIC	VARGRAPHIC
BLOB	BLOB
CLOB	CLOB
DBCLOB	DBCLOB

結果の長さ属性は、*ストリング式* の長さ属性と同じになります。結果の実際の長さは、式の長さから、除去したバイトの数を引いたものになります。すべての文字が除去された場合は、結果は空のストリングになります。

最初の引き数がヌルである可能性がある場合は、結果もヌルになる可能性があります。最初の引き数がヌルの場合は、結果はヌル値になります。

結果の CCSID は、指定したストリングの CCSID と同じになります。

例

- ホスト変数 HELLO (CHAR(9)) には、値として ' Hello' が入っていると想定します。

```
SELECT RTRIM(:HELLO)
    FROM SYSIBM.SYSDUMMY1
```

  結果は 'Hello' になります。

38. RTRIM 関数は、STRIP(式,TRAILING) と同じ結果を戻します。

SECOND

SECOND 関数は、指定した値の秒の部分を戻します。

引き数は、時刻、タイム・スタンプ、文字ストリング、または数値データ・タイプのいずれかの組み込みデータ・タイプの値を戻す式でなければなりません。

- 式 が文字ストリングである場合は、そのストリングは CLOB であってはならず、値は時刻またはタイム・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。時刻とタイム・スタンプの有効なストリング表現の形式については、71 ページの 『日付/時刻の値のストリング表現』を参照してください。
- 式 が数値である場合は、その数値は時刻期間またはタイム・スタンプ期間でなければなりません。日付時刻期間の有効な形式については、141 ページの 『日付/時刻のオペランドと期間』を参照してください。

この関数の結果は、長整数になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

その他の規則は、引き数のデータ・タイプ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、時刻またはタイム・スタンプの有効な文字ストリング表現である場合 :

  結果は、指定した値の秒の部分 (0 から 59 までの整数) 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、指定した値の秒の部分 (-99 から 99 までの整数) 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)) は、値が 153045 であると想定します。

```
SELECT SECOND(:TIME_DUR)
  FROM SYSIBM.SYSDUMMY1
```

  値として 45 が戻されます。
- 列 RECEIVED (TIMESTAMP) 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想定します。

```
SELECT SECOND(RECEIVED)
  FROM IN_TRAY
```

  値として 30 が戻されます。

SIGN

SIGN 関数は式の符号の標識を戻します。戻り値は以下のとおりです。

-1 引き数がゼロ未満の場合

0 引き数がゼロの場合

1 引き数がゼロより大きい場合

引き数は、任意の組み込み数値データ・タイプの値を戻す式です。

結果のデータ・タイプと長さ属性は引き数と同じになります。ただし、引き数が DECIMAL または NUMERIC で、引き数の位取りが精度と同じである場合は、結果の精度は 1 だけ増やされます。例えば、データ・タイプが DECIMAL(5,5) の引き数の場合、結果は DECIMAL(6,5) になります。精度がすでに 31 の場合は、位取りが 1 下がります。例えば、DECIMAL(31,31) の場合、結果は DECIMAL(31,30) になります。

引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

例

- ホスト変数 PROFIT は、値が 50000 の長整数であると想定します。

```
SELECT SIGN(:PROFIT)
  FROM EMPLOYEE
```

  値として 1 が戻されます。

SIN

SIN 関数は引き数のサイン (正弦) を戻すもので、引き数はラジアンで表された角度です。 SIN 関数と ASIN 関数は、逆の演算になります。

引き数は、任意の組み込み数値データ・タイプの値を戻す式です。

結果のデータ・タイプは、倍精度の浮動小数点数になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

例

- ホスト変数 SINE は、値が 1.5 の 10 進数 (2,1) のホスト変数であると想定します。

```
SELECT SIN(:SINE)
  FROM SYSIBM.SYSDUMMY1
```

  およそ 0.99 の値が戻されます。

SINH

SINH 関数は引き数の双曲線サイン (双曲線正弦) を戻すもので、引き数はラジアンで表された角度です。

引き数は、任意の組み込み数値データ・タイプの値を戻す式です。

結果のデータ・タイプは、倍精度の浮動小数点数になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

例

- ホスト変数 HSINE は、値が 1.5 の 10 進数 (2,1) のホスト変数であると想定します。

```
SELECT SINH(:HSINE)
  FROM SYSIBM.SYSDUMMY1
```

  およそ 2.12 の値が戻されます。

SMALLINT

数値から短整数に

文字から短整数に

SMALLINT 関数は、次のものの短整数表現を戻します。

- 数値
- 10 進数の文字ストリング表現
- 整数の文字ストリング表現
- 浮動小数点数の文字ストリング表現

注: 短整数値を戻すには、CAST 式も使用することができます。詳しくは、149 ページの『CAST の指定』を参照してください。

数値から短整数に

数値式
任意の組み込み数値データ・タイプの数値を戻す式。

結果は、引き数が短整数の列または変数に割り当てられたときに得られる数値と同じです。引き数の整数部が、短整数の範囲内にない場合は、エラーになります。引き数の小数部は切り捨てられます。

文字から短整数に

文字式
整数の文字ストリング表現である値を戻す式。 この式は CLOB であってはなりません。

結果は、CAST(文字式 AS SMALLINTL) から得られる数値と同じです。先行ブランクと後書きブランクは除去され、結果のストリングは、浮動小数点数、整数、または 10 進数の定数を形成する規則に合致している必要があります。引き数の整数部が、短整数の範囲内にない場合は、エラーになります。引き数の小数部は切り捨てられます。引き数の小数部は切り捨てられます。

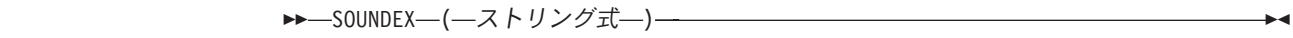
この関数の結果は、短整数になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルである場合は、結果はヌル値です。

例

- 表 EMPLOYEE を使用して、給与 (SALARY) を教育レベル (EDLEVEL) で除算した値が入っているリストを選択します。計算で生じた小数部は、すべて切り捨てられます。このリストには、計算で使用した値と従業員番号 (EMPNO) も入れておきます。

```
SELECT SMALLINT(SALARY / EDLEVEL), SALARY, EDLEVEL, EMPNO
  FROM EMPLOYEE
```

SOUNDEX



SOUNDEX 関数は、引き数内のワードの音を表す 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比較するのに使用できます。

引き数には、BLOB、CLOB、または DBCLOB を除く任意の組み込みストリング・データ・タイプを指定できます。

結果のデータ・タイプは、CHAR(4) です。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

結果の CCSID は、現行サーバーのデフォルトの CCSID になります。

SOUNDEX 関数は、音は分かっているが正確なスペルが分からないストリングを見つけるのに便利です。これは、文字や文字の組み合わせの音に関する想定をして、類似の音をもつワードを検索するのに役立てます。比較は、直接行うことも、ストリングを DIFFERENCE 関数への引き数として渡して行うこともできます。詳しくは、227 ページの 『DIFFERENCE』を参照してください。

例

- EMPLOYEE 表を使用して、姓 が 'Loucesy' のような音をもつ従業員の EMPNO と LASTNAME を検索します。

```
SELECT EMPNO, LASTNAME
       FROM EMPLOYEE
       WHERE SOUNDEX(LASTNAME) = SOUNDEX('Loucesy')
```

以下の行が戻されます。

```
000110 LUCCHESSI
```

SPACE

SPACE 関数は、引き数で指定された SBCS ブランク数からなる文字ストリングを戻します。

引き数は、結果が整数になる式です。整数は、結果の SBCS ブランクの数を示し、0 ～ 32740 でなければなりません。*数値式* が定数の場合、定数 0 であってはなりません。

この関数の結果は、SBCS データを含む可変長文字ストリング (VARCHAR) になります。

数値式 が定数の場合、結果の長さ属性は定数です。 それ以外の場合、結果の長さ属性は 4000 です。結果の実際の長さは、*数値式* の値です。結果の実際の長さは、結果の長さ属性を超えてはなりません。

引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

CCSID は、そのジョブの SBCS データの EBCDIC CCSID です。

例

- 次のステートメントは、5 つのブランクからなる文字ストリングを戻します。

```
SELECT SPACE(5)
  FROM SYSIBM.SYSDUMMY1
```

SQRT

SQRT 関数は、数値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・タイプの値を戻す式です。*数値式* の値は、ゼロまたはそれより大きい値でなければなりません。引き数は、この関数で処理できるように倍精度の浮動小数点数に変換されます。

結果のデータ・タイプは、倍精度の浮動小数点数になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

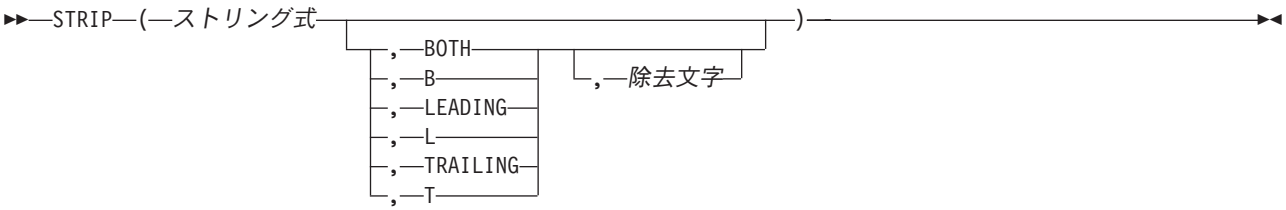
例

- ホスト変数 SQUARE は、値が 9.0 の DECIMAL(2,1) のホスト変数であると想定します。

```
SELECT SQRT(:SQUARE)
  FROM SYSIBM.SYSDUMMY1
```

  およそ 3.00 の値が戻されます。

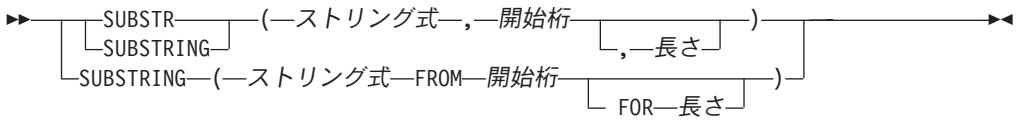
STRIP



STRIP 関数は、ストリング式の後部または前部 (またはその両方) から、ブランクまたは指定した文字を除去します。

STRIP 関数は、TRIM スカラー関数と同等です。詳細は、319 ページの 『TRIM』を参照してください。

SUBSTRING または SUBSTR



SUBSTR 関数および SUBSTRING 関数は、ストリングのサブストリングを戻します。

ストリング式
結果が導き出される元になるストリングを指定する式。

ストリング式 は、文字ストリング、グラフィック・ストリング、または 2 進ストリングでなければなりません。*ストリング式* が文字ストリングの場合は、この関数の結果は文字ストリングになります。ストリング式がグラフィック・ストリングの場合は、関数の結果はグラフィック・ストリングになります。ストリング式が 2 進ストリングの場合は、関数の結果は 2 進ストリングになります。

ストリング式 のサブストリングは、*ストリング式* に含まれるゼロ個以上の連続したバイトです。*ストリング式* がグラフィック・ストリングである場合は、1 文字は DBCS または UCS-2 の 1 文字です。 *ストリング式* が文字ストリングまたは 2 進ストリングである場合は、1 文字は 1 バイトです。 SUBSTR 関数に指定するストリングは、混合データ・ストリングでも構いません。ただし、SUBSTR は厳密なバイト・カウントに基づいて演算を行うので、結果は必ずしも適切な形式の混合データ文字ストリングにはなりません。

開始桁
ストリング式 の中の、結果の最初の文字 (またはバイト) の位置を指定する式。これは 2 進整数でなければなりません。*開始桁* は負またはゼロでも構いません。また、*ストリング式* の長さ属性より大きくても構いません。 (可変長ストリングの長さ属性は、そのストリングの最大長です。)

長さ
結果の長さを指定する式。*長さ* を指定する場合は、2 進整数で指定する必要があります。*長さ* は、0 から *n* までの範囲内の値でなければなりません。*n* は、結果のデータ・タイプの最大長です。

SUBSTR を指定し、*長さ* を明示指定した場合は、実際にはストリング式 の右側に必要数のブランク文字 (2 進ストリングの場合は 16 進数のゼロ) が埋め込まれるため、*ストリング式* の指定したサブストリングは常に存在します。

SUBSTRING を指定し、*長さ* を明示指定した場合は、埋め込みは行われません。

ストリング式 が固定長ストリングの場合は、*長さ* を省略すると、LENGTH(*ストリング式*) - *開始桁* + 1 (*ストリング式* の *開始* 文字 (またはバイト) から最終文字 (またはバイト) までの文字数) が暗黙指定されます。*ストリング式* が可変長ストリングの場合に、*長さ* の指定を省略すると、0 と LENGTH(*ストリング式*) - *開始桁* + 1 のいずれか大きい方が、暗黙の長さの指定として使用されます。結果の長さがゼロの場合は、結果は空ストリングになります。

SUBSTRING または SUBSTR

結果のデータ・タイプは、*ストリング式* のデータ・タイプによって異なり、関数が SUBSTR と SUBSTRING のいずれかによっても異なります。

ストリング式 の データ・タイプ	**SUBSTRING** の 場合の結果の データ・タイプ	**SUBSTR** の場合の結果のデータ・タイプ
CHAR または VARCHAR	VARCHAR	CHAR (*長さ* が整定数で明示的に指定されているか、*長さ* は明示的に指定されていないが、*ストリング式* が固定長ストリングであり、*開始桁* が整定数である場合)。 VARCHAR (それ以外のすべての場合)。
GRAPHIC または VARGRAPHIC	VARGRAPHIC	GRAPHIC (*長さ* が整定数で明示的に指定されているか、*長さ* は明示的に指定されていないが、*ストリング式* が固定長ストリングであり、*開始桁* が整定数である場合)。 VARGRAPHIC (それ以外のすべての場合)。
BLOB	BLOB	BLOB
CLOB	CLOB	CLOB
DBCLOB	DBCLOB	DBCLOB

SUBSTRING 関数を指定すると、結果の長さ属性は、*ストリング式* の長さ属性と同じになります。

SUBSTR 関数を指定して、*ストリング式* が LOB でない場合、結果の長さ属性は、*長さ*、*開始桁*、および*ストリング式* の属性によって決まります。

- *長さ* を整定数で明示的に指定すると、結果の長さ属性は *長さ* になります。
- *長さ* は明示的に指定されていないが、*ストリング式* が固定長ストリングであり、*開始桁* が整定数である場合は、結果の長さ属性が LENGTH(*ストリング式*) - *開始桁* + 1 になります。

それ以外のすべての場合、結果の長さ属性は *ストリング式* の長さ属性と同じになります。 (*ストリング式* の実際の長さが*開始桁* の値より小さい場合は、サブストリングの実際の長さはゼロになります。)

SUBSTR を基にしたソース関数では、結果は常に可変長ストリングになります。

SUBSTR 関数のいずれかの引き数がヌルである可能性がある場合は、結果もヌルである可能性があります。いずれかの引き数がヌルである場合は、結果はヌル値になります。

結果の CCSID は*ストリング式* と同じです。

例

- ホスト変数 NAME (VARCHAR(50)) の値は 'KATIE AUSTIN' で、ホスト変数 SURNAME_POS (INTEGER) の値は 7 であると想定します。

```
SELECT SUBSTR(:NAME, :SURNAME_POS)
  FROM SYSIBM.SYSDUMMY1
```

値として 'AUSTIN' が戻されます。

- 同様に、

```
SELECT SUBSTR(:NAME, :SURNAME_POS, 1)
  FROM SYSIBM.SYSDUMMY1
```

  値として 'A' が戻されます。

- 表 PROJECT から、プロジェクト名 (PROJNAME) が 'OPERATION ' の語で始まっている行をすべて選択します。

```
SELECT *
  FROM PROJECT
  WHERE SUBSTR(PROJNAME,1,10) = 'OPERATION '
```

  'OPERATIONS' などで始まる行を除外したい場合には、定数の最後にスペースを付ける必要があります。

TAN

TAN 関数は引き数のタンジェント (正接) を戻すもので、引き数はラジアンで表された角度です。 TAN 関数と ATAN 関数は、逆の演算になります。

引き数は、任意の組み込み数値データ・タイプの値を戻す式です。

結果のデータ・タイプは、倍精度の浮動小数点数になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

例

- ホスト変数 TANGENT は、値が 1.5 の DECIMAL(2,1) のホスト変数であると想定します。

```
SELECT TAN(:TANGENT)
  FROM SYSIBM.SYSDUMMY1
```

  およそ 14.10 の値が戻されます。

TANH

TANH 関数は引き数の双曲線タンジェント (双曲線正接) を戻すもので、引き数はラジアンで表された角度です。 TANH 関数と ATANH 関数は、逆の演算になります。

引き数は、任意の組み込み数値データ・タイプの値を戻す式です。

結果のデータ・タイプは、倍精度の浮動小数点数になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

例

- ホスト変数 HTANGENT は、値が 1.5 の DECIMAL(2,1) のホスト変数であると想定します。

```
SELECT TANH(:HTANGENT)
  FROM SYSIBM.SYSDUMMY1
```

  およそ 0.90 の値が戻されます。

TIME

TIME 関数は、指定された値から時刻を戻します。

注: 時刻値を戻すには、CAST 式も使用できます。詳しくは、 149 ページの『CAST の指定』を参照してください。

引き数は、時刻、タイム・スタンプ、または文字ストリングのいずれかの組み込みデータ・タイプの値を戻す式でなければなりません。式 が文字ストリングである場合は、そのストリングは CLOB であってはならず、値は日付またはタイム・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。 日付とタイム・スタンプの有効なストリング表現の形式については、71 ページの 『日付/時刻の値のストリング表現』を参照してください。

この関数の結果は、時刻になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

その他の規則は、引き数のデータ・タイプ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です。
- 引き数が文字ストリングの場合 :

  時刻のストリング表現が、SBCS データのデフォルト CCSID 以外の CCSID を持つ SBCS データである場合、その値は、時刻の値として解釈され変換される前に、SBCS データのデフォルト CCSID を持つように変換されます。

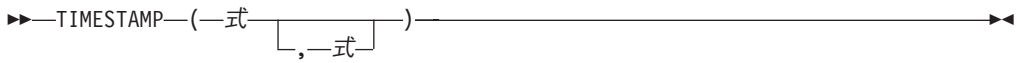
  時刻のストリング表現が、混合データのデフォルト CCSID 以外の CCSID を持つ混合データである場合、その値は、時刻の値として解釈され変換される前に、混合データのデフォルト CCSID を持つように変換されます。

例

- サンプル表 IN_TRAY から、現在の時刻より 1 時間以上あと (日付は問わない) に受け取ったコメントをすべて選択します。

```
SELECT *
  FROM IN_TRAY
  WHERE TIME(RECEIVED) >= CURRENT TIME + 1 HOUR
```

TIMESTAMP



TIMESTAMP 関数は、1 つまたは複数の引き数から導き出されるタイム・スタンプを戻します。

注: タイム・スタンプ値を戻すには、CAST 式も使用できます。詳しくは、 149 ページの『CAST の指定』を参照してください。

引き数に関する規則は、2 番目の引き数を指定するかどうかによって異なります。

- 引き数を 1 つだけ指定する場合 :

  引き数は、タイム・スタンプまたは文字ストリングのいずれかの組み込みデータ・タイプの値を戻す式でなければなりません。

  式 が文字ストリングである場合は、そのストリングは CLOB であってはならず、値は以下のいずれかでなければなりません。

  – 日付またはタイム・スタンプの有効な文字ストリング表現。日付とタイム・スタンプの有効なストリング表現の形式については、71 ページの 『日付/時刻の値のストリング表現』を参照してください。
  – yyyynnn の形式で有効な日付を表す、実際長が 7 の文字ストリング。yyyy は年番号を表す数値で、nnn は年間通算日を表す 001 ～ 366 の数値です。
  – yyyyxxddhhmmss の形式で有効な日付と時刻を表す、実際の長さが 14 の文字ストリング。yyyy は年、xx は月、dd は日、hh は時、mm は分、ss は秒です。

- 引き数を 2 つ指定する場合 :

  最初の引き数は、日付または文字ストリングのいずれかの組み込みデータ・タイプの値を戻す式でなければなりません。 2 番目の引き数は、時刻または文字ストリングのいずれかの組み込みデータ・タイプの値を戻す式でなければなりません。

  最初の式 が文字ストリングである場合は、そのストリングは CLOB であってはならず、値は日付の有効な文字ストリング表現でなければなりません。 2 番目の式 が文字ストリングである場合は、そのストリングは CLOB であってはならず、値は時刻の有効な文字ストリング表現でなければなりません。日付と時刻の有効なストリング表現の形式については、71 ページの 『日付/時刻の値のストリング表現』を参照してください。

この関数の結果は、タイム・スタンプになります。引き数のどちらかがヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数のどちらかがヌルである場合は、結果はヌル値になります。

その他の規則も、2 番目の引き数を指定するかどうかに応じて以下のように異なります。

- 引き数を 2 つ指定する場合 :

結果は、最初の引き数に指定した日付と 2 番目の引き数に指定した時刻から構成されるタイム・スタンプになります。このタイム・スタンプのマイクロ秒の部分は、ゼロになります。

- タイム・スタンプの引き数を 1 つだけ指定した場合 :

  結果は、指定したタイム・スタンプになります。

- 文字ストリングの引き数を 1 つだけ指定した場合 :

  結果は、指定した文字ストリングで表されるタイム・スタンプになります。引き数が長さ 14 の文字ストリングの場合、結果のタイム・スタンプのマイクロ秒の部分は、ゼロになります。

タイム・スタンプのストリング表現が、SBCS データのデフォルト CCSID 以外の CCSID を持つ SBCS データである場合、その値は、タイム・スタンプの値として解釈され変換される前に、SBCS データのデフォルト CCSID を持つように変換されます。

タイム・スタンプのストリング表現が、混合データのデフォルト CCSID 以外の CCSID を持つ混合データである場合、その値は、タイム・スタンプの値として解釈され変換される前に、混合データのデフォルト CCSID を持つように変換されます。

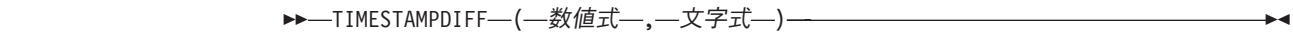
例

- 日付と時刻の値が以下のとおりであるとします。

```
SELECT TIMESTAMP( DATE('1988-12-25'), TIME('17.12.30') )
  FROM SYSIBM.SYSDUMMY1
```

値として、'1988-12-25-17.12.30.000000' が戻されます。

TIMESTAMPDIFF



TIMESTAMPDIFF 関数は、2 つのタイム・スタンプの差に基づいて、最初の引き数によって定義されたタイプの間隔の見積数を戻します。

最初の引き数は、INTEGER または SMALLINT のいずれかの組み込みデータ・タイプでなければなりません。間隔 (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・タイプを減算し、その結果を CHAR(22) に変換したものです。

この関数の結果は、整数になります。引き数のどちらかがヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数のどちらかがヌルである場合は、結果はヌル値になります。

差の見積もりには、次の前提事項を適用できます。

- 1 年は 365 日
- 1 か月は 30 日
- 1 日は 24 時間
- 1 時間は 60 分
- 1 分は 60 秒

これらの前提条件は、2 番目の引き数の情報 (タイム・スタンプ期間) を、最初の引き数で指定された間隔タイプに変換するときに使用します。戻される見積値は、日数が異なることがあります。例えば、タイム・スタンプ '1997-03-01-00.00.00' と '1997-02-01-00.00.00' の差に対して日数 (間隔 16) が要求されている場合、結果は 30 になります。これは、タイム・スタンプの差は 1 か月なので、1 か月は 30 日という前提事項が適用されるからです。

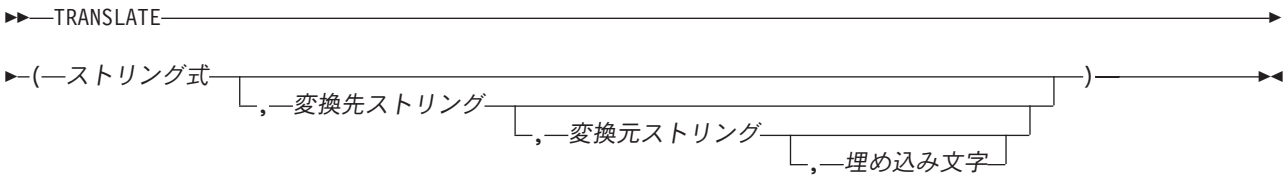
例

- 従業員の年齢を月数で見積もります。

TIMESTAMPDIFF

```
SELECT
    TIMESTAMPDIFF(64,
    CAST(CURRENT_TIMESTAMP-CAST(BIRTHDATE AS TIMESTAMP) AS CHAR(22)))
      AS AGE_IN_MONTHS
  FROM EMPLOYEE
```

TRANSLATE



TRANSLATE 関数は、*ストリング式* の中の 1 つまたは複数の文字を他の文字に変換した値を戻します。

ストリング式
変換されるストリングを指定する式。*ストリング式* は、文字ストリングまたは UCS-2 グラフィック・ストリングでなければなりません。

変換先ストリング
ストリング式 の中の特定の文字をどのような文字に変換するかを指定するストリング。このストリングは*出力変換表* とも呼ばれます。このストリングは文字ストリング定数でなければなりません。文字ストリング引き数の実際の長さは、256 以下でなければなりません。

変換先ストリング の長さ属性が*変換元ストリング* の長さ属性よりも小さい場合は、*埋め込み文字* かブランクのいずれかを使用して、長い方の桁数に合わせて *変換先ストリング* を埋め込みます。*変換先ストリング* の長さ属性が *変換元ストリング* の長さ属性より大きい場合は、*変換先ストリング* にある余分な文字は無視され、警告は出されません。

変換元ストリング
ストリング式 の中のどの文字を変換するのかを指定するストリング。このストリングは*入力変換表* とも呼ばれます。*変換元ストリング* に指定されている文字のいずれかがストリング式 の中に見つかると、その文字は、*変換先ストリング* の中の文字のうち、*変換元ストリング* でのその文字と同じ位置にある文字に変換されます。

このストリングは文字ストリング定数でなければなりません。文字ストリング引き数の実際の長さは、256 以下でなければなりません。

変換元ストリング に重複する文字がある場合は、左からスキャンした最初の文字が使用され、警告は出されません。*変換元ストリング* のデフォルト値は、文字 X'00' で始まり、文字 X'FF' (10 進数 255) で終わるストリングです。

埋め込み文字
変換先ストリング が*変換元ストリング* より短い場合に、変換先ストリングに埋め込む文字を指定するストリング。このストリングは、長さが 1 の文字ストリング定数でなければなりません。デフォルト値は SBCS のブランクです。

最初の引き数が UCS-2 グラフィック・ストリングの場合は、他の引き数を指定することができません。

最初の引き数だけが指定されている場合は、引き数の SBCS 文字は、その引き数の CCSID に基づいて、大文字に変換されます。 SBCS 文字だけが変換されます。 a

～ z の文字は A ～ Z に変換され、発音記号付きの文字はそれぞれの大文字に変換されます。最初の引き数が UCS-2 グラフィック・ストリングの場合は、英字 UCS-2 文字は大文字に変換されます。この変換に使用する大文字変換表については、iSeries Information Center の Globalization トピックの UCS-2 level 1 mapping tables のセクションを参照してください。

複数の引き数を指定した場合は、結果のストリングは、*変換元ストリング* の文字を *変換先ストリング* の対応する文字に変換して、1 文字ずつストリング式 から構築されます。*ストリング式* の中の各文字ごとに、同一文字が*変換元ストリング* で検索されます。その文字が*変換元ストリング* の *n* 番目の文字であることが分かった場合は、結果のストリングには、*変換先ストリング* から *n* 番目の文字が入ります。 *変換先ストリング* が *n* 文字より短い場合は、結果のストリングには埋め込み文字が入ります。その文字が*変換元ストリング* に見つからない場合は、未変換のまま結果のストリングに移されます。

変換はバイト基準で行われ、使用を誤った場合は、結果的に無効の混合ストリングになります。SRTSEQ 属性は、TRANSLATE 関数には適用されません。

この関数の結果のデータ・タイプ、長さ属性、実際の長さ、および CCSID は、引き数と同じになります。最初の引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。 引き数がヌルである場合は、結果はヌル値です。

例

- ストリング 'abcdef' を大文字変換します。

```
SELECT TRANSLATE('abcdef')
  FROM SYSIBM.SYSDUMMY1
```

  'ABCDEF' の値が戻されます。

- 混合文字ストリングを大文字変換します。

```
SELECT TRANSLATE('ab<SO>c<SI>def')
  FROM SYSIBM.SYSDUMMY1
```

  次の値が戻されます。 'AB<SO>C<SI>DEF'

- ホスト変数 SITE が、'Pivabiska Lake Place' という値の可変長文字ストリングである場合

```
SELECT TRANSLATE(:SITE, '$', 'L')
  FROM SYSIBM.SYSDUMMY1
```

  値 'Pivabiska $ake Place' が戻されます。

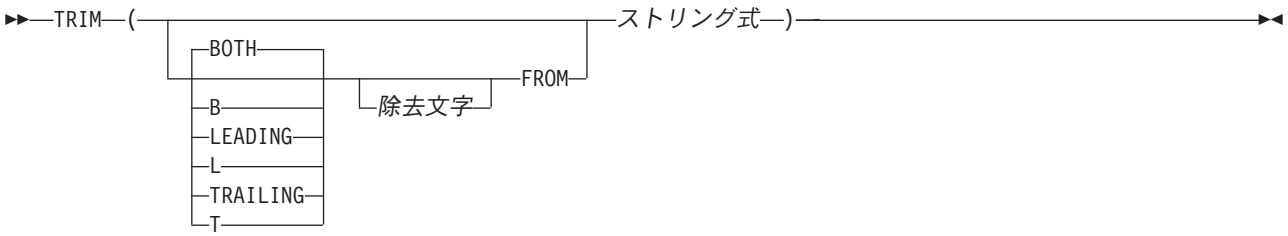
```
SELECT TRANSLATE(:SITE, '$$', 'Ll')
  FROM SYSIBM.SYSDUMMY1
```

  値 'Pivabiska $ake P$ace' が戻されます。

```
SELECT TRANSLATE(:SITE, 'pLA', 'Place', '.')
  FROM SYSIBM.SYSDUMMY1
```

  値 'pivAbiskA LAk. pLA..' が戻されます。.

TRIM



TRIM 関数は、ストリング式の後部または前部から、ブランクまたは指定した文字を除去します。

ストリング式 は、ストリング式である必要があります。

最初の引き数を指定する場合は、ストリングの後部と前部のどちらから文字を除去するのかを指示します。最初の引き数を指定しない場合は、ストリングの前部と後部の両方から文字が除去されます。

2 番目の引き数が指定された場合、除去する 2 進数、SBCS または DBCS 文字を示す 1 文字定数となります。*ストリング式* が 2 進ストリングの場合、2 番目の引き数は 2 進ストリング定数でなければなりません。*ストリング式* が DBCS グラフィック・ストリングまたは DBCS 専用ストリングである場合は、2 番目の引き数は、1 つの DBCS 文字からなるグラフィック定数にする必要があります。 2 番目の引き数を指定しない場合は、次のようになります。

- *ストリング式* が 2 進ストリングの場合、デフォルトの除去文字は 16 進ゼロ (X'00') になる。
- *ストリング式* が DBCS グラフィック・ストリングである場合は、デフォルトの除去文字は DBCS ブランクになる。
- *ストリング式* が UCS-2 グラフィック・ストリングである場合は、デフォルトの除去文字は UCS-2 ブランクになる。
- それ以外の場合は、デフォルトの除去文字は、SBCS ブランクになる。

結果のデータ・タイプは、*ストリング式* のデータ・タイプによって異なります。

式 のデータ・タイプ	結果のデータ・タイプ
CHAR または VARCHAR	VARCHAR
GRAPHIC または VARGRAPHIC	VARGRAPHIC
BLOB	BLOB
CLOB	CLOB
DBCLOB	DBCLOB

結果の長さ属性は、*ストリング式* の長さ属性と同じになります。結果の実際の長さは、式の長さから、除去したバイトの数を引いたものになります。すべての文字が除去された場合は、結果は空のストリングになります。

最初の引き数がヌルである可能性がある場合は、結果もヌルになる可能性があります。最初の引き数がヌルの場合は、結果はヌル値になります。

結果の CCSID は、指定したストリングの CCSID と同じになります。

SRTSEQ 属性は、TRIM 関数には適用されません。

例

- ホスト変数 HELLO (CHAR(9)) には、値として ’ Hello’ が入っていると想定します。

```
SELECT TRIM(:HELLO), TRIM( TRAILING FROM :HELLO)
  FROM SYSIBM.SYSDUMMY1
```

  結果は、それぞれ、’Hello’ および ’ Hello’ になります。

- ホスト変数 BALANCE (CHAR(9)) には、値として ’000345.50’ が入っていると想定します。

```
SELECT TRIM( L '0' FROM :BALANCE )
  FROM SYSIBM.SYSDUMMY1
```

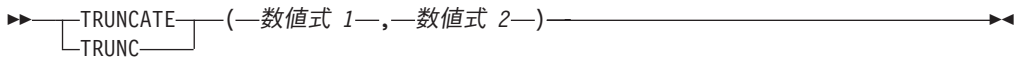
  結果は ’345.50’ になります。

- 除去するストリングの中に、混合データが入っていると想定します。

```
SELECT TRIM( BOTH 'SO SI' FROM 'SO AB C SI' )
    FROM SYSIBM.SYSDUMMY1
```

  結果は次のようになります。 'SO AB C SI'

TRUNCATE または TRUNC



TRUNCATE 関数は、*数値式 1* を、小数点の右側または左側の特定の桁まで切り捨てた値を戻します。

数値式 1
任意の組み込み数値データ・タイプの値を戻す式。

数値式 2
短整数または長整数を戻す式。整数の絶対値は、*数値式 2* が負でなければ、小数点より右の桁数を指定し、*数値式 2* が負であれば、小数点より左の桁数を指定します。

数値式 2 が負でない場合は、*数値式 1* は、小数点の右側の*数値式 2* 桁目まで切り捨てられます。

数値式 2 が負である場合は、*数値式 1* は、小数点の左側の (*数値式 2* + 1) の絶対値に相当する桁まで切り捨てられます。

数値式 2 の絶対値が小数点より左の桁数より大きい場合は、結果は 0 になります。たとえば、TRUNCATE (748.58,-4) = 0 です。

結果のデータ・タイプおよび長さ属性は、最初の引き数のデータ・タイプおよび長さ属性と同じです。

どちらかの引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数のどちらかがヌルである場合は、結果はヌル値になります。

例

- 最も給与の高い従業員の平均月収を計算します。その結果を小数点の右、2 桁目までで切り捨てます。

```
SELECT TRUNCATE(MAX(SALARY/12, 2)
  FROM EMPLOYEE
```

  サンプルの従業員表内で給与の最も高い従業員の年収は $52750.00 なので、この例では 4395.83 の値が戻されます。

- 数値 873.726 を、小数点から 2、1、0、-1、-2、-3 までで切り捨てます。

```
SELECT TRUNCATE(873.726,  2),
       TRUNCATE(873.726,  1),
       TRUNCATE(873.726,  0),
       TRUNCATE(873.726, -1),
       TRUNCATE(873.726, -2),
       TRUNCATE(873.726, -3)
  FROM SYSIBM.SYSDUMMY1
```

  それぞれ以下の値が戻されます。

```
0873.720   0873.700  0873.000  0870.000  0800.000  0000.000
```

- 正負両方の数値を計算します。

TRUNCATE

```
SELECT TRUNCATE( 3.5, 0),
       TRUNCATE( 3.1, 0),
       TRUNCATE(-3.1, 0),
       TRUNCATE(-3.5, 0)
  FROM SYSIBM.SYSDUMMY1
```

それぞれ以下の値が戻されます。

```
3.0   3.0  -3.0  -3.0
```

UCASE

UCASE 関数は、すべての文字を引き数の CCSID に基づいて大文字に変換したストリングを戻します。

UCASE 関数は、UPPER 関数と同等です。 詳しくは、 324 ページの『UPPER』を参照してください。

UPPER

UPPER 関数は、すべての文字を引き数の CCSID に基づいて大文字に変換したストリングを戻します。 SBCS および UCS-2 グラフィック文字だけが変換されます。a ～ z の文字は A ～ Z に変換され、発音記号付きの文字はそれぞれの大文字に変換されます。 この変換に使用する大文字変換表については、iSeries Information Center のグローバリゼーションのトピックの UCS-2 レベル 1 マッピング・テーブルのセクションを参照してください。

ストリング式
変換するストリングを指定する式。ストリング式 は、文字ストリングまたは UCS-2 グラフィック・ストリングでなければなりません。

この関数の結果のデータ・タイプ、長さ属性、実際の長さ、および CCSID は、引き数と同じになります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

UCASE は UPPER の同義語です。

例

- UPPER スカラー関数を使用して、ストリング 'abcdef' を大文字変換します。

```
SELECT UPPER('abcdef')
    FROM SYSIBM.SYSDUMMY1
```

'ABCDEF' の値が戻されます。

- UPPER スカラー関数を使用して、混合文字ストリングを大文字変換します。

次の値が戻されます。 

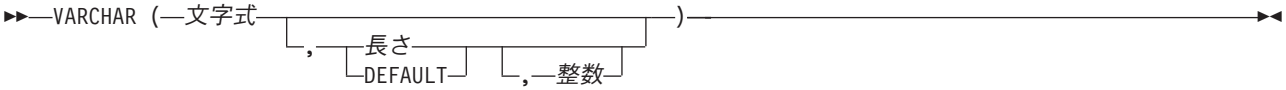
VALUE

VALUE 関数は、ヌルでない最初の式の値を戻します。

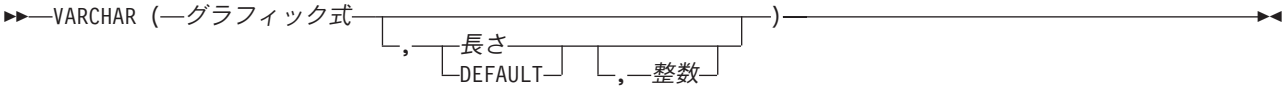
VALUE 関数は、COALESCE スカラー関数と同等です。詳しくは、206 ページの『COALESCE』を参照してください。

VARCHAR

文字から可変長文字に



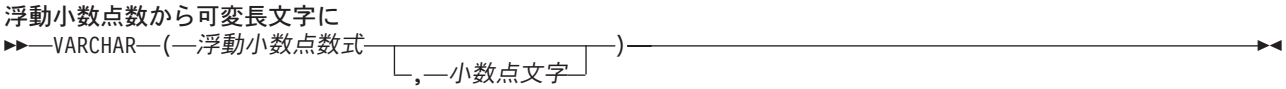
グラフィックから可変長文字に



整数から可変長文字に

10 進数から可変長文字に

浮動小数点数から可変長文字に



VARCHAR 関数は、次のものの文字ストリング表現を戻します。

- 整数 (最初の引き数が SMALLINT、INTEGER、または BIGINT の場合)
- 10 進数 (最初の引き数がパックまたはゾーン 10 進数の場合)
- 倍精度浮動小数点数 (最初の引き数が DOUBLE または REAL の場合)
- 文字ストリング (最初の引き数が任意のタイプの文字ストリングの場合)
- グラフィック・ストリング (最初の引き数が UCS-2 グラフィック・ストリングの場合)

注: 可変長文字ストリング値を戻すには、CAST 式も使用できます。詳しくは、149 ページの『CAST の指定』を参照してください。

この関数の結果は可変長ストリングです。最初の引き数がヌルである可能性がある場合は、結果もヌルになる可能性があります。最初の引き数がヌルの場合は、結果はヌル値になります。

文字から可変長文字に

文字式
CHAR、VARCHAR、または CLOB 組み込みデータ・タイプの値を戻す式。

長さ
結果の可変長文字ストリングのための長さ属性を指定します。値は 1 から 32740 (空でもよい場合は 32739) までででなければなりません。最初の引き数が混合データである場合は、2 番目の引き数は 4 より小さくてはなりません。

2 番目の引き数が指定されないか DEFAULT が指定された場合は、次のようになります。

- ストリング式 が空ストリング定数の場合は、結果の長さ属性は 1。
- 空ストリング定数でない場合は、結果の長さ属性は、最初の引き数の長さ属性と同じ。

結果の実際の長さは、結果の長さ属性と 文字式 の実際の長さのいずれか小さい方となります。文字式の長さが結果の長さ属性よりも大きい場合は、切り捨てが行われます。切り捨てられた文字がすべてブランクであった場合以外は、警告 (SQLSTATE 01004) が戻されます。

整数

結果の CCSID を指定します。これは有効な SBCS CCSID、混合データ CCSID、または 65535 (ビット・データ) とする必要があります。3 番目の引き数が SBCS CCSID の場合は、結果は SBCS データになります。3 番目の引き数が混合 CCSID の場合は、結果は混合データになります。3 番目の引き数が 65535 の場合は、結果はビット・データになります。3 番目の引き数が SBCS CCSID の場合は、最初の引き数が DBCS 択一または DBCS 専用のストリングであることはありません。

3 番目の引き数の指定がない場合は、次のようになります。

- 最初の引き数が SBCS データであれば、結果は SBCS データになる。結果の CCSID は、最初の引き数の CCSID と同一です。
- 最初の引き数が混合データ (DBCS 混用、DBCS 専用、または DBCS 択一) であれば、結果は混合データになる。結果の CCSID は、最初の引き数の CCSID と同一です。

グラフィックから可変長文字に

グラフィック式

GRAPHIC、VARGRAPHIC、または DBCLOB データ・タイプの値を戻す式。最初の引き数は、DBCS グラフィック・データであってはなりません。

長さ

結果の可変長文字ストリングのための長さ属性を指定します。値は 1 から 32740 (空でもよい場合は 32739) までででなければなりません。最初の引き数が混合データを含む場合は、2 番目の引き数は 4 より小さくてはなりません。

2 番目の引き数が指定されていないか、または DEFAULT が指定されている場合は、結果の長さ属性は、次のように決まります。(ただし、*n* は最初の引き数の長さ属性です。)

- グラフィック式 が空グラフィック・ストリング定数の場合は、結果の長さ属性は 1 になる。
- 結果が SBCS データであれば、結果の長さは *n* になる。
- 結果が混合データであれば、結果の長さは (2.5*(*n* -1)) + 4 になる。

結果の実際の長さは、結果の長さ属性と グラフィック式 の実際の長さのいずれか小さい方となります。文字式の長さが結果の長さ属性よりも大きい場合は、切り捨てが行われます。切り捨てられた文字がすべてブランクであった場合以外は、警告 (SQLSTATE 01004) が戻されます。

整数
結果の CCSID を指定します。これは有効な SBCS CCSID または混合データ CCSID とする必要があります。3 番目の引き数が SBCS CCSID の場合は、結果は SBCS データになります。3 番目の引き数が混合 CCSID の場合は、結果は混合データになります。3 番目の引き数を 65535 とすることはできません。

3 番目の引き数が指定されていない場合は、結果の CCSID は現行サーバーのデフォルトの CCSID になります。デフォルトの CCSID が混合データの場合は、結果は混合データになります。デフォルトの CCSID が SBCS データの場合は、結果は SBCS データになります。

整数から可変長文字に

整数式
整数データ・タイプ (SMALLINT、INTEGER、または BIGINT) の値を戻す式。

結果は、SQL 整数定数の形式で引き数を表現した可変長文字ストリングです。結果は、引き数の値を表す n 文字の有効数字から成ります。引き数が負数の場合は、負符号が前に付きます。結果は左寄せされます。

- 引き数が短整数の場合は、結果の長さ属性は 6
- 引き数が長整数の場合は、結果の長さ属性は 11
- 引き数が 64 ビット整数の場合は、結果の長さ属性は 20

結果の実際の長さは、引き数の値を表すために使用できる最小文字数です。先行ゼロは含まれません。引き数が負数の場合は、結果の最初の文字は負符号になります。負数以外の場合は、最初の文字は数字です。

結果の CCSID は、現行サーバーのデフォルトの SBCS CCSID になります。

10 進数から可変長文字に

10 進数式
パックまたはゾーン 10 進数データ・タイプ (DECIMAL または NUMERIC) の値を戻す式。精度や位取りを変えたい場合は、DECIMAL スカラー関数を使用して変更することができます。

小数点文字
結果の文字ストリングにおいて、小数点以下を区切るために使用する 1 バイト文字の定数を指定します。この文字は、ピリオドかコンマとする必要があります。 2 番目の引き数を指定しなかった場合は、デフォルトの小数点が使用されます。詳しくは、 108 ページの『小数点』を参照してください。

結果は、引き数を可変長文字ストリングで表現したものになります。この結果には、1 文字の小数点文字と *p* 桁までの数字が含まれます。*p* は *10 進数式* の精度で、引き数が負数の場合は負符号が先頭に付きます。先行ゼロは戻されません。後続ゼロは戻されます。

結果の長さ属性は、2+*p* です。*p* は *10 進数式* の精度です。結果の実際の長さは、引き数の値を表すために使用できる最小文字数ですが、ただし、後書き文字も含まれます。先行ゼロは含まれません。引き数が負数の場合は、結果の最初の文字は負符号になります。負数でなければ、結果は数字で始まります。

結果の CCSID は、現行サーバーのデフォルト SBCS CCSID になります。

浮動小数点数から可変長文字に

浮動小数点数式
浮動小数点データ・タイプ (DOUBLE または REAL) の値を戻す式。

小数点文字
結果の文字ストリングにおいて、小数点以下を区切るために使用する 1 バイト文字の定数を指定します。この文字は、ピリオドかコンマとする必要があります。 2 番目の引き数を指定しなかった場合は、デフォルトの小数点が使用されます。詳しくは、108 ページの『小数点』を参照してください。

結果は、浮動小数点定数の形式で引き数を可変長文字ストリングで表現したものになります。

結果の長さ属性は、24 です。結果の実際の長さは、ゼロ以外の 1 桁の数字、その後ろに 1 つの小数点文字 と一連の数字が続く小数部の引き数の値を表す最小文字数です。引き数が負数の場合は、結果の最初の文字は負符号になります。負数でなければ、最初の文字は数字です。引き数がゼロであると、結果は 0E0 になります。

結果の CCSID は、現行サーバーのデフォルトの SBCS CCSID になります。

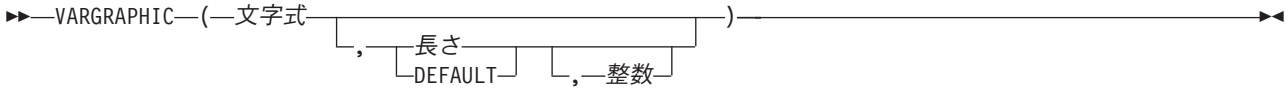
例

- EMPNO を長さ 10 の可変長にします。

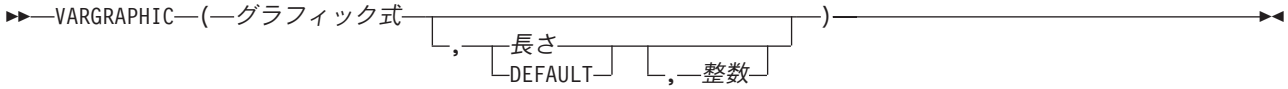
```
SELECT VARCHAR(EMPNO,10)
  INTO :VARHV
  FROM EMPLOYEE
```

VARGRAPHIC

文字からグラフィックに



グラフィックからグラフィックに



VARGRAPHIC 関数は、ストリング式のグラフィック文字表現を戻します。

注: 可変長グラフィック・ストリング値を戻すには、CAST 式も使用できます。詳しくは、 149 ページの『CAST の指定』を参照してください。

この関数の結果は可変長グラフィック・ストリング (VARGRAPHIC) です。

式がヌルである可能性がある場合は、結果もヌルになる可能性があります。式がヌルである場合は、結果はヌル値です。式が空ストリング、または EBCDIC ストリング X'0E0F' である場合は、結果は空ストリングになります。

文字からグラフィックに

文字式
文字ストリング式を指定します。CHAR または VARCHAR ビット・データであってはなりません。

長さ
結果の長さ属性となり、最初の引き数がヌル可能でない場合は、1 から 16370 の範囲の整定数でなければならず、最初の引き数がヌル可能である場合は、1 から 16369 の範囲の整定数でなければなりません。

2 番目の引き数を指定しなかった場合、または DEFAULT を指定した場合は、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。ただし、式が空ストリングまたは EBCDIC ストリング X'0E0F' である場合は結果の長さ属性は 1 です。

結果の実際の長さは、引き数の中の文字数に応じて異なります。引き数の各文字ごとに、結果の 1 文字が決まります。結果の可変長ストリングの長さ属性が最初の引き数の実際の長さより小さい場合は、切り捨てが行われ、警告が戻されることはありません。

整数
結果の CCSID を指定します。これは DBCS または UCS-2 CCSID でなければなりません。CCSID が 65535 であることはできません。CCSID が UCS-2 グラフィック・データである場合は、引き数の各文字ごとに結果の 1 文字が決まります。結果の n 番目の文字は、引き数の n 番目の文字と等価の UCS-2 文字になります。

整数 が指定されていない場合、結果の CCSID は混合 CCSID によって決まります。M でその混合 CCSID を示すことにします。

以下の規則では、S は次のいずれかを指します。

- ストリング式が外部コード化スキームのデータを含むホスト変数である場合は、データを固有コード化スキームの CCSID に変換した後の式の結果が S。(詳細については、 34 ページの『文字変換』の項を参照してください。)
- ストリング式が固有コード化スキームのデータである場合は、そのストリング式が S。

M は次のように決まります。

- S の CCSID が混合 CCSID である場合は、M はその CCSID になる。
- S の CCSID が SBCS CCSID である場合 :
  - S の CCSID が関連する混合 CCSID をもつ場合は、M はその CCSID になる。
  - それ以外の場合は、演算ができない。

次の表には、M をもとにした結果の CCSID を要約してあります。

M	結果の CCSID	説明	DBCS 置換文字
930	300	日本語 EBCDIC	X'FEFE'
933	834	韓国語 EBCDIC	X'FEFE'
935	837	中国語 (簡体字) EBCDIC	X'FEFE'
937	835	中国語 (繁体字) EBCDIC	X'FEFE'
939	300	日本語 EBCDIC	X'FEFE'
5026	4396	日本語 EBCDIC	X'FEFE'
5035	4396	日本語 EBCDIC	X'FEFE'

SBCS と DBCS が等価になるかどうかは M によって決まります。CCSID に関係なく、引き数の中の 2 バイトのコード・ポイントはすべて DBCS 文字と見なされ、引き数の中の 1 バイトのコード・ポイントはすべて SBCS 文字と見なされます (ただし、EBCDIC 混合データのシフト・コード `X'0E'` および `X'0F'` は例外)。

- 引き数の n 番目の文字が DBCS 文字である場合は、結果の n 番目の文字はその DBCS になる。
- 引き数の n 番目の文字が、等価の DBCS 文字をもつ SBCS 文字である場合は、結果の n 番目の文字は、その等価の DBCS 文字になる。
- 引き数の n 番目の文字が、等価の DBCS 文字をもたない SBCS 文字である場合は、結果の n 番目の文字は、DBCS 置換文字になる。

グラフィックからグラフィックに

グラフィック式
グラフィック・ストリング式を指定します。

長さ

結果の長さ属性となり、最初の引き数がヌル可能でない場合は、1 から 16370 の範囲の整定数でなければならず、最初の引き数がヌル可能である場合は、1 から 16369 の範囲の整定数でなければなりません。

2 番目の引き数を指定しなかった場合、または DEFAULT を指定した場合は、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。ただし、式が空ストリングである場合は結果の長さ属性は 1 です。

結果の実際の長さは、引き数の中の文字数に応じて異なります。引き数の各文字ごとに、結果の 1 文字が決まります。結果の可変長ストリングの長さ属性が最初の引き数の実際の長さより小さい場合は、切り捨てが行われ、警告が戻されることはありません。

整数

結果の CCSID を指定します。これは DBCS または UCS-2 CCSID でなければなりません。CCSID が 65535 であることはできません。

整数 が指定されていない場合、結果の CCSID は、最初の引き数の CCSID になります。

例

- 表 EMPLOYEE を使用して、ホスト変数 VAR_DESC (VARGRAPHIC(24)) を従業員番号 (EMPNO) '000050' に対応する氏名の名 (FIRSTNME) と等価の VARGRAPHIC にセットします。

```
SELECT VARGRAPHIC(FIRSTNME)
  INTO :VAR_DESC
  FROM EMPLOYEE
  WHERE EMPNO = '000050'
```

WEEK

WEEK 関数は、年間通算週を表す 1 から 54 までの範囲の整数を戻します。週は日曜日から始まります。1 月 1 日は常に第 1 週に入ります。

引き数は、日付、タイム・スタンプ、または文字ストリングのいずれかの組み込みデータ・タイプの値を戻す式でなければなりません。

式 が文字ストリングである場合は、そのストリングは CLOB であってはならず、値は日付またはタイム・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。日付とタイム・スタンプの有効なストリング表現の形式については、71 ページの 『日付/時刻の値のストリング表現』を参照してください。

この関数の結果は、長整数になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

例

- 表 PROJECT を使用して、ホスト変数 WEEK (INTEGER) をプロジェクト ('PL2100') が終了した週にセットします。

```
SELECT WEEK(PRENDATE)
  INTO :WEEK
  FROM PROJECT
  WHERE PROJNO = 'PL2100'
```

  結果として、WEEK は 38 にセットされます。

- 表 X に DATE_1 という名前の DATE 列があり、以下のリストに示すような日付が入っているとします。

```
SELECT DATE_1, WEEK(DATE_1)
  FROM X
```

  結果として、各日付について WEEK 関数が戻した値を示す以下のようなリストが表示されます。

```
1997-12-28    53
1997-12-31    53
1998-01-01    1
1999-01-01    1
1999-01-04    2
1999-12-31    53
2000-01-01    1
2000-01-03    2
```

WEEK_ISO

WEEK_ISO 関数は、年間通算週を表す 1 から 53 までの範囲の整数を戻します。週は月曜日から始まります。週 1 は、木曜日が含まれるその年の最初の週を表します。つまり、1 月 4 日が含まれる最初の週と同じことです。したがって、年初の最高 3 日間が前年の最後の週と見なされたり、年末の最高 3 日間が来年の最初の週と見なされたりする可能性があります。

引き数は、日付、タイム・スタンプ、または文字ストリングのいずれかの組み込みデータ・タイプの値を戻す式でなければなりません。

式 が文字ストリングである場合は、そのストリングは CLOB であってはならず、値は日付またはタイム・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。日付とタイム・スタンプの有効なストリング表現の形式については、71 ページの 『日付/時刻の値のストリング表現』を参照してください。

この関数の結果は、長整数になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

例

- 表 PROJECT を使用して、ホスト変数 WEEK (INTEGER) をプロジェクト ('AD2100') が終了した週にセットします。

```
SELECT WEEK_ISO(PRENDATE)
  INTO :WEEK
  FROM PROJECT
  WHERE PROJNO = 'AD3100'
```

  結果として、WEEK は 5 にセットされます。

- 表 X に DATE_1 という名前の DATE 列があり、以下のリストに示すような日付が入っているとします。

```
SELECT DATE_1, WEEK_ISO(DATE_1)
  FROM X
```

  結果は以下のようになります。

```
1997-12-28    52
1997-12-31    1
1998-01-01    1
1999-01-01    53
1999-01-04    1
1999-12-31    52
2000-01-01    52
2000-01-03    1
```

XOR

XOR 関数は、引き数ストリングの論理 XOR であるストリングを戻します。この関数は、最初の引き数ストリングを次のストリングと XOR 比較し、それから前の結果を使用して、連続する各引き数との XOR 比較を繰り返します。引き数が前の結果より短い場合は、ブランクが埋め込まれます。

引き数は、文字ストリングでなければなりませんが LOB とすることはできません。引き数は、混合データ文字ストリング、またはグラフィック・ストリングであってはなりません。

必要ならば、引き数は結果の属性に変換されます。結果の属性は、以下のように決められます。

- すべての引き数が固定長ストリングである場合、結果は長さが *n* の固定長ストリングになります (ここで、*n* は最長の引き数の長さ)。
- 引き数の中に可変長ストリングがある場合、結果は長さ属性が *n* の可変長ストリングになります (ここで、*n* は最大の長さ属性を持つ引き数の長さ属性)。結果の実際の長さは *m* です (ここで、*m* は、最長の引き数の実際の長さ)。

引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

結果の CCSID は 65535 です。

例

- ホスト変数 L1 は値が X'0101' の CHARACTER(2) のホスト変数、ホスト変数 L2 は値が X'F0F000' の CHARACTER(3) のホスト変数、ホスト変数 L3 は値が X'0000000F' の CHARACTER(4) のホスト変数であると想定します。

```
SELECT XOR(:L1,:L2,:L3)
  FROM SYSIBM.SYSDUMMY1
```

  値として X'1111404F' が戻されます。この場合、短い方の結果にはブランク (X'40') が埋め込まれるので、論理 XOR の結果は次の例の結果と異なります。

```
SELECT XOR(:L3,:L2,:L1)
  FROM SYSIBM.SYSDUMMY1
```

  値として X'1111400F' が戻されます。

YEAR

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、文字ストリング、または数値データ・タイプのいずれかの組み込みデータ・タイプの値を戻す式でなければなりません。

- 式 が文字ストリングである場合は、そのストリングは CLOB であってはならず、値は日付またはタイム・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。日付とタイム・スタンプの有効なストリング表現の形式については、71 ページの 『日付/時刻の値のストリング表現』を参照してください。
- 式 が数値である場合は、その数値は日付期間またはタイム・スタンプ期間でなければなりません。日付時刻期間の有効な形式については、141 ページの 『日付/時刻のオペランドと期間』を参照してください。

この関数の結果は、長整数になります。引き数がヌルになる可能性がある場合は、結果もヌルになる可能性があります。引き数がヌルの場合は、結果はヌル値になります。

その他の規則は、引き数のデータ・タイプに応じて以下のように異なります。

- 引き数が日付またはタイム・スタンプ、または、日付またはタイム・スタンプの有効な文字ストリング表現である場合 :

  結果は、指定した値の年の部分 (1 から 9999 までの整数) 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 (-9999 から 9999 までの整数) 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

- 表 PROJECT から、開始 (PRSTDATE) と終了 (PRENDATE) が同じ年に予定されているプロジェクトをすべて選択します。

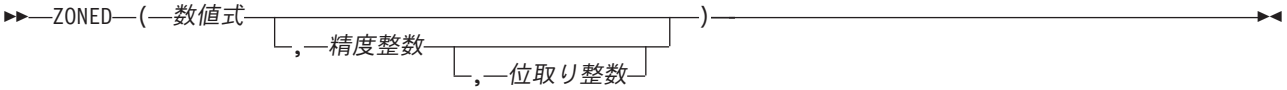
```
SELECT *
  FROM PROJECT
  WHERE YEAR(PRSTDATE) = YEAR(PRENDATE)
```

- 表 PROJECT から、1 年未満で完了するように予定されているプロジェクトをすべて選択します。

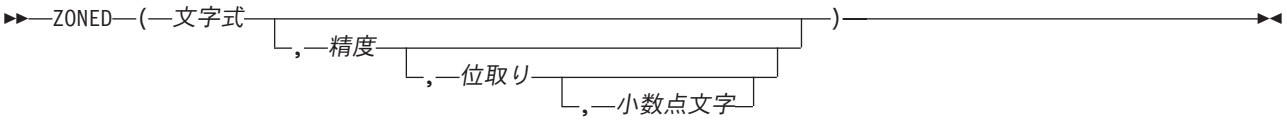
```
SELECT *
  FROM PROJECT
  WHERE YEAR(PRENDATE - PRSTDATE) < 1
```

ZONED

数値からゾーン 10 進数に



文字からゾーン 10 進数に



ZONED 関数は、次のもののゾーン 10 進数表現を戻します。

- 数値
- 整数の文字ストリング表現
- 10 進数の文字ストリング表現
- 浮動小数点数の文字ストリング表現

注: ゾーン10 進数値を戻すには、CAST 式も使用できます。詳しくは、 149 ページの『CAST の指定』を参照してください。

この関数の結果は、精度が *p* および位取りが *s* (*p* と *s* は、それぞれ 2 番目と 3 番目の引き数) のゾーン 10 進数になります。最初の引き数がヌルである可能性がある場合は、結果もヌルになる可能性があります。最初の引き数がヌルの場合は、結果はヌル値になります。

数値からゾーン 10 進数に

数値式
任意の組み込み数値データ・タイプの値を戻す式。

精度
1 以上で 31 以下の値を持つ整数定数。

デフォルトの*精度* は、*数値式* のデータ・タイプによって決まります。

- 15 (最初の引き数が浮動小数点数、10 進数、数字、または位取りがゼロ以外の 2 進数の場合)
- 19 (最初の引き数が 64 ビット整数の場合)
- 11 (最初の引き数が長整数の場合)
- 5 (最初の引き数が短整数の場合)

位取り
0 以上で*精度* 以下である整数定数。これを指定しないと、デフォルト値の 0 になります。

結果は、最初の引き数が、精度 *p* で位取り *s* の 10 進数の列または変数に割り当てられた場合に生じる数値と同じになります。数値の整数部を表すのに必要な有効桁数が *p -s* より大きい場合は、エラーになります。

文字からゾーン 10 進数に

文字式
数値の文字ストリング表現を含む式。先行ブランクと後書きブランクは除去され、結果のストリングは、整数または 10 進数の定数を形成する規則に合致している必要があります。この式は CLOB であってはなりません。

精度
1 以上で 31 以下である整数定数。これを指定しないと、デフォルト値の 15 になります。

位取り
0 以上で*精度* 以下である整数定数。これを指定しないと、デフォルト値の 0 になります。

小数点文字
数値の整数部分から文字式 の小数桁数を区切るために使用された 1 バイトの文字定数を指定します。この文字は、ピリオドかコンマとする必要があります。 2 番目の引き数を指定しなかった場合は、デフォルトの小数点が使用されます。
詳しくは、 108 ページの『小数点』を参照してください。

結果は、CAST(文字式 AS NUMERIC(*p*,*s*)) によって得られる数と同じです。*小数点文字* の右側の桁数が位取り *s* より大きい場合は、末尾の桁が切り捨てられます。文字式 の中の*小数点文字* より左側にある有効桁数 (数値の整数部分) が *p-s* より大きい場合は、エラーになります。*小数点文字* 引き数の指定がある場合は、サブストリング内のデフォルト小数点文字は無効です。

例

- ホスト変数 Z1 は、値が 1.123 の 10 進数ホスト変数であると想定します。

```
SELECT ZONED(:Z1,15,14)
  FROM SYSIBM.SYSDUMMY1
```

  値として 1.12300000000000 が戻されます。

- ホスト変数 Z1 は、値が 1123 の 10 進数ホスト変数であると想定します。

```
SELECT ZONED(:Z1,11,2)
  FROM SYSIBM.SYSDUMMY1
```

  値として 1123.00 が戻されます。

- 同様に、

```
SELECT ZONED(:Z1,4)
  FROM SYSIBM.SYSDUMMY1
```

  値として 1123 が戻されます。

第 4 章 照会

SQL *照会* は、結果表または中間結果表を指定するものです。

照会は、特定の SQL ステートメントのコンポーネントの 1 つになります。照会には、次の 3 つの形式があります。

- *副選択*
- *全選択*
- *選択ステートメント*

764 ページの『SELECT INTO』で説明している別の形式の選択もあります。

『権限』も参照してください。

権限

どのような形式の照会の場合でも、ステートメントの権限 ID によって保持される特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメントで識別されている表やビューのそれぞれについて、
  - 表やビューに対する SELECT 特権、および
  - 表やビューが入っているライブラリーに対する *EXECUTE システム権限
- 管理権限

ステートメントの権限 ID は、以下の場合に表に対する SELECT 特権を持ちます。

- その表の所有者である。
- その表に対する SELECT 特権が認可されているか、または
- その表に対する *OBJOPR および *READ のシステム権限が認可されている。

ステートメントの権限 ID は、以下の場合にビューに対する SELECT 特権を持ちます。

- そのビューの所有者である。
- そのビューに対する SELECT 特権が認可されているか、または
- そのビューに対する *OBJOPR および *READ のシステム権限が認可されており、さらに、そのビューが直接または間接的に依存している表やビューのすべてに対する *READ システム権限が認可されている。すなわち、そのビューの定義で参照されている表やビューのすべて、またそのようなビューが参照される場合、そのビューの定義で参照されている表やビューのすべて、以下同様。

式 に関数が含まれている場合、ステートメントの権限 ID には、各ユーザー定義関数ごとに少なくとも以下の 1 つが含まれていなければなりません。

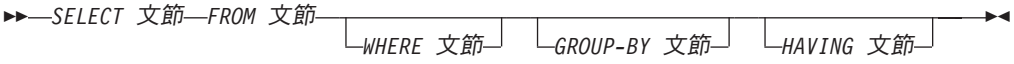
- その関数に対する EXECUTE 特権
- 管理権限

照会

次の場合、ステートメントの権限 ID には、関数に対する EXECUTE 特権が与えられます。

- その関数の所有者である。
- その関数に対する EXECUTE 特権が認可されている、または
- その関数に対するシステム権限の *OBJOPR と *EXECUTE が認可されている。

副選択



副選択 は、全選択、CREATE VIEW ステートメント、および INSERT ステートメントのコンポーネントです。副選択が特定の述部のコンポーネントとなり、その述部がさらに副選択のコンポーネントとなることもあります。

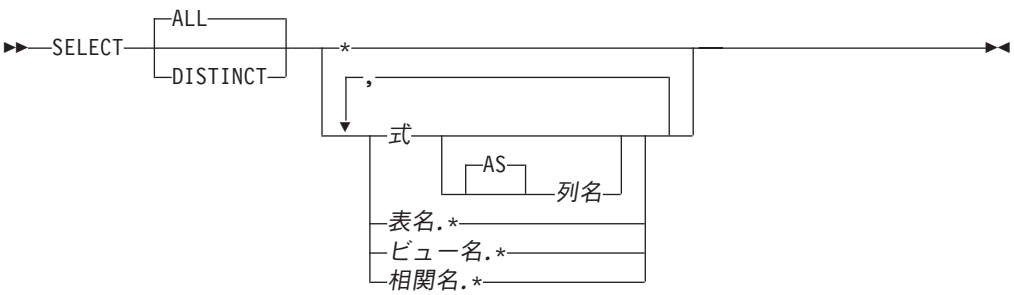
スカラー副選択 は括弧で囲んだ副選択で、単一の結果行および単一の結果列を戻します。副選択の結果に行が含まれない場合、ヌル値が戻されます。結果の中に複数の行がある場合には、エラーが戻されます。

副選択では、FROM 文節で識別されている表またはビューから得られる結果表を指定します。結果表が得られる過程は、各操作の結果が次の操作への入力となるような一連の操作として説明できます。(これは、副選択の説明でだけ使用する考え方です。結果表を得るために、この説明とはまったく異なる方法が使用されることもあります。)

操作の (仮の) 順序は、次のようになります。

1. FROM 文節
2. WHERE 文節
3. GROUP-BY 文節
4. HAVING 文節
5. SELECT 文節

SELECT 文節



SELECT 文節は、最終的な結果表の列を指定します。列の値は、R に対して*選択リスト* を適用することによって作成されます。選択リストは SELECT 文節で指定された名前または式で、R は副選択の前の演算の結果です。例えば、 SELECT、FROM、および WHERE の文節だけを指定した場合、R は WHERE 文節の結果です。

ALL
最終的な結果表のすべての行を選択します。重複行の除去は行いません。これは、デフォルト値です。

DISTINCT
最終的な結果表にある重複行の組から、1 行だけを残して他の行をすべて除去します。

2 つの行が互いに重複したものとして扱われるのは、一方の行にあるそれぞれの値が、もう一方の行の対応する値にすべて等しい場合だけです。(重複行を判別する場合、ヌル値どうしは等しいものとされます。) 除去する値を判別するためには、ソート順序も使用されます。

選択リスト に LOB またはデータ・リンク列が入っている場合、DISTINCT は使用できません。

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。このリストの最初の名前は R の最初の列、2 番目の名前は R の 2 番目の列、以下同様を表します。

名前のリストは、SELECT 文節が入っているステートメントが準備されるときに設定されます。このため、ステートメントが準備された後で表に追加された列があっても、* ではその列を識別しません。

式 136 ページの『式』で説明されているタイプの任意の式を使用できます。式 の中の各*列名* は、R の列を明瞭に識別するものでなければなりません。

列名 または **AS** *列名*
結果の列の名前、または名前の付け直しを指定します。名前は、修飾してはなりません。また固有である必要はありません。

*名前.**
名前 の列を識別する名前のリストを表します。*名前* には、表名、ビュー名、または相関名を指定できます。ここには、FROM 文節で名前を指定した表またはビューを指定しなければなりません。リストの最初の名前は、表またはビューの最初の列を識別し、2 番目の名前は表またはビューの 2 番目の列を識別するというように、対応する列を順に識別します。

名前のリストは、SELECT 文節が入っているステートメントが準備されるときに設定されます。このため、ステートメントが準備された後で表に追加された列があっても、* ではその列を識別しません。

通常、SQL ステートメントが暗黙に再バインド (再準備) される時点で、名前のリストは再確立されません。したがって、ステートメントによって戻される列の数は変わりません。ただし、名前のリストが再度確立され、列の数が変わる場合が 4 つあります。

- SQL プログラムまたは SQL パッケージが保管され、その保管元システムとは異なるリリースの iSeries システム上で復元される場合。

- SQL プログラムまたはパッケージに SQL 命名規則の指定があり、その SQL プログラムまたはパッケージの作成後に、その SQL プログラムの所有者が変更されている場合。
- より最新リリースの OS/400 のインストール後、初めて SQL ステートメントが実行される場合。
- INSERT ステートメントの副選択、または述部の副選択で SELECT * が行われ、しかもその副選択で参照される表またはビューが削除され、他の列によって作成し直されている場合。

SELECT の結果の列の数は、選択リストの実行形式 (つまり、準備 時に確立されたリスト) にある式の数と同じであり、8000 を超えることはできません。副照会を EXISTS 述部で使用している場合を除いて、副照会の結果は単一の式でなければなりません。

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。これらの結果について、個別に説明します。

GROUP BY または HAVING を使用した場合:

- 選択リスト内に*列名* を使用している式の場合は、その式がグループ化式を示すか、または列関数の中で指定されている必要があります。
  - グループ化式が列名の場合、選択リストには列名への加算演算子を適用できます。例えば、グループ化式が列 C1 の場合、選択リストに C1+1 を含めることができます。
  - グループ化式が列名でない場合、選択リストには式への加算演算子を適用できません。例えば、グループ化式が C1+1 の場合、選択リストに C1+1 を含めることはできますが、(C1+1)/8 は含めることができません。
- 選択リスト内で、RRN、PARTITION、NODENAME、および NODENUMBER 関数を指定することはできません。
- 選択リストは R の各グループに適用され、結果には R にあるグループと同じ数の行が含まれます。選択リストが R のグループに適用されるとき、選択リスト内の列関数の引き数はそのグループの中から与えられます。

GROUP BY も HAVING も使用していない場合:

- 選択リスト全体を列関数のリストにする場合以外は、選択リストの中で列関数を使用してはなりません。
- 列関数が入っていない選択リストは、R の各行に適用され、結果には R にある行と同じ数の行が入ります。
- 選択リストを列関数のリストにした場合は、関数の引き数が R から与えられ、選択リストを適用した結果は 1 つの行になります。

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。

結果列のヌル属性

結果列が以下のものから得られた場合は、結果列にヌル値が入ることがあります。

- COUNT および COUNT_BIG を除く列関数
- ヌル値が許される任意の列
- ヌル値のオペランドを使用できるスカラー関数または式
- 標識変数を持つホスト変数
- UNION の結果 (選択リスト内の対応する項目の中に、ヌル可能な項目が少なくとも 1 つある場合)
- 算術式
- スカラー副選択
- ユーザー定義のスカラー関数または表関数

結果列の名前

- AS 文節の指定がある場合、結果列の名前は、AS 文節で指定された名前です。
- 列リストを相関文節で指定した場合は、結果列の名前は、その相関列リストの対応する名前になります。
- AS 文節も相関文節の列リストも指定せず、結果列が単一の列からだけ得られる (関数も演算子もない) 場合は、結果列の名前はその列の修飾の付かない名前になります。
- 上記以外の結果列はすべて、名前を持ちません。

結果列のデータ・タイプ

SELECT の結果の各列のデータ・タイプは、その列を得るときの元になった式から取得されます。

元になった式	結果列のデータ・タイプ
数値の列の名前	その列のデータ・タイプと同じ (10 進数の列については、同じ精度および位取りを持つ)。
整定数	INTEGER または BIGINT (定数の値が INTEGER の範囲外であっても、BIGINT の範囲内である場合)。
10 進数または浮動小数点定数	その定数のデータ・タイプと同じ (10 進定数については同じ精度および位取りを持つ)。
数値のホスト変数の名前	その変数のデータ・タイプと同じ (10 進数の変数については同じ精度および位取りを持つ)。変数のデータ・タイプが、SQL データ・タイプに一致しない (例えば、COBOL の DISPLAY SIGN LEADING SEPARATE など) 場合、結果列は 10 進数になります。
算術式	算術式の結果のデータ・タイプと同じ (10 進数の結果については、同じ精度および位取りを持つ)。算術式の結果については、 136 ページの『式』で説明しています。
任意の関数	関数の結果のデータ・タイプ。組み込み関数の場合の結果のデータ・タイプについては、第 3 章を参照してください。ユーザー定義の関数の場合は、結果のデータ・タイプは、その関数の CREATE FUNCTION ステートメントで定義されているデータ・タイプになります。
ストリング列の名前	その列のデータ・タイプと同じ (長さ属性も同じ)。

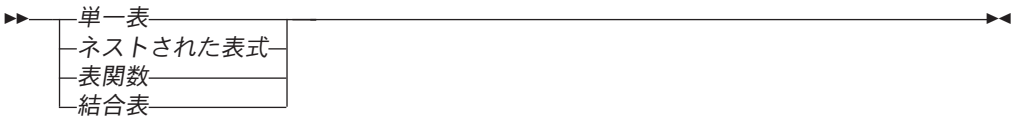
元になった式	結果列のデータ・タイプ
ストリングのホスト変数の名前	その変数のデータ・タイプと同じ (長さ属性もその変数の長さ属性と同じ)。変数のデータ・タイプが、SQL データ・タイプと一致しない (例えば、C の NUL 終了ストリングなど) 場合、結果列は可変長ストリングになります。
長さ *n* の文字ストリング定数	VARCHAR(*n*)
長さ *n* のグラフィック・ストリング定数	VARGRAPHIC(*n*)
日付/時刻の列または ILE RPG コンパイラー または ILE COBOL コンパイラー日付/時刻ホスト変数の名前	その列またはホスト変数のデータ・タイプと同じ。
データ・リンク列の名前。	同じ長さ属性のデータ・リンク。
行 ID 列または行 ID ホスト変数の名前。	ROWID
スカラー副選択	スカラー副選択の選択リスト内の式のデータ・タイプ。

FROM 文節

FROM 文節は、中間結果の表を指定します。

表参照 を 1 つだけ指定した場合は、その*表参照* の結果がそのまま中間結果の表になります。FROM 文節で複数の*表参照* を指定した場合は、中間結果の表は、指定された表参照の行のあらゆる組み合わせ (カルテシアン積) から構成されます。結果の各行は、最初の*表参照* の行に 2 番目の*表参照* の行を連結し、それに 3 番目からの行を連結し、以下同様に行を連結したものです。結果の行数は、個々の*表参照* すべての行数の積です。*表参照* については、『表参照』を参照してください。

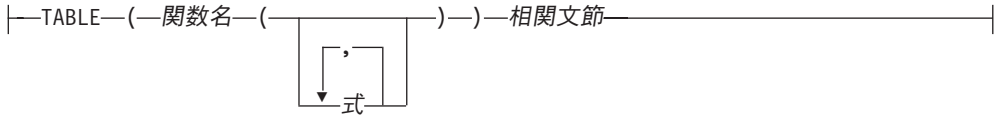
表参照



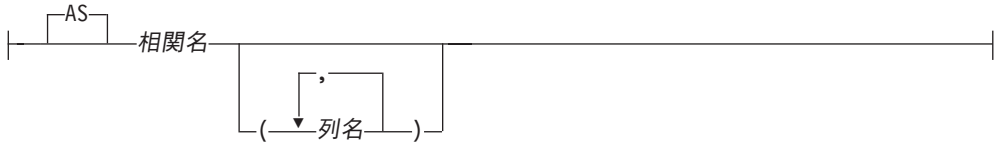
単一表:

ネストされた表式:

表関数:



相関文節:



表参照 は、中間結果の表を指定します。

- 表またはビューを 1 つだけ指定している場合は、その表またはビューが中間結果の表になる。
- WITH 文節の中の括弧内の副選択は、*ネストされた表式* と呼ばれる。[39] ネストされた表式を指定すると、結果表は、そのネストされた表式の結果となります。結果の列には固有の名前は必要ありませんが、固有の名前を持たない列は参照することができません。
- *関数名* を指定した場合は、中間結果の表は、その表関数が戻す行のセットから成る。
- *結合表* を指定した場合は、中間結果の表は、1 つまたは複数の結合演算の結果になる。詳細は、 347 ページの『結合表』を参照してください。

FROM 文節中の名前のリストは、以下の規則に従わなければなりません。

- それぞれの*表名* およびビュー名 は、現行サーバーの既存の表またはその表参照を含む副選択に先立って定義された*共通表式* (356 ページの『共通表式』を参照) の表名とする必要がある。
- 直接名 (exposed name) は固有でなければならない。直接名は、相関名、後に相関名が続かない表名、または後に相関名が続かないビュー名です。

39. *ネストされた表式* は、*派生表* とも呼ばれます。

- 各*関数名* とそれぞれの引き数のタイプを解決した結果が、現行サーバー上に存在する表関数のどれかにならなければならない。関数解決と呼ばれるアルゴリズム (131 ページの『関数解決』の説明を参照) は、関数名と引き数を使用して、どの関数を使用するかを正確に判別します。*相関文節* で列名を指定しなかった場合は、CREATE FUNCTION ステートメントの RETURNS 文節に指定されている列名が、表関数用として使用されます。これは、CREATE TABLE で表の中に定義する列名に似ています。

各*相関名* は、直前の先行表参照によって指定される中間結果表の指定子として定義されます。ネストされた表式および表関数の場合は、相関名を指定する必要があります。

すべての表参照の直接名は固有でなければなりません。直接名には次のものがあります。

- *相関名*
- 後ろに*相関名* が続かない*表名* またはビュー名

表、ビュー、ネストされた表式、派生表、または表関数の列に対する修飾付き参照では、直接名を使用する必要があります。同じ表名またはビュー名を二度使用するときは、少なくとも 1 つの指定の後ろに*相関名* を続けてください。表またはビューの列に対する参照を修飾するためには、*相関名* を使用します。 *相関名* を指定する場合は、列名も同時に指定することによって、*表名*、*ビュー名*、*ネストされた表式*、または*表関数* の列に名前を与えることができます。列リストを指定する場合は、その列リストの中で、表またはビューの各列について、および、*ネストされた表式* または*表関数* の個々の結果列について、それぞれ名前を指定する必要があります。詳細は、 115 ページの『相関名』を参照してください。

一般に、*ネストされた表式* および*表関数* は、どの *FROM 文節* でも指定することができます。ネストされた表式および表関数からの列は、選択リストの中および副選択の後続部分で、相関名 (これは指定する必要があります) 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 文節の他の表名またはビュー名のため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。

- ビューの代わりとして。これはそのビューを作成するのを避けるためです (そのビューの一般的な使用が要求されない場合)。
- 必要な結果表がホスト変数に基づいているとき。

表参照における相関参照: 相関参照は、ネストされた表式で使用することができます。基本的な規則として、相関参照は、副照会階層の高いレベルの*表参照* からの参照である必要があります。この階層には、FROM 文節を左から右に処理することで解決された*表参照* が含まれます。

表関数には、同じ FROM 文節の中にある他の表に対する 1 つまたは複数の相関参照を含めることができます。ただし、これは、FROM 文節内での左から右への表の順序において、参照される表がその参照より前にある場合に限られます。そうでない場合は、副照会の階層内の上位レベルに対する参照のみが許されます。

例 1: 次は正しい例です。

```
SELECT D.DEPTNO, D.DEPTNAME, EMPINFO.AVGSAL, EMPINFO.EMPCOUNT
FROM DEPARTMENT D,
  (SELECT AVG(E.SALARY) AS AVGSAL,COUNT (*) AS EMPCOUNT, E.WORKDEPT AS DEPT
     FROM EMPLOYEE E
     WHERE E.WORKDEPT =
       (SELECT X.DEPTNO
         FROM DEPARTMENT X
         WHERE X.DEPTNO = E.WORKDEPT ) GROUP BY E.WORKDEPT)
AS EMPINFO
WHERE D.DEPTNO = EMPINFO.DEPT
```

次の例は、ネストされた表式 の WHERE 文節において、D.DEPTNO に対する参照が副照会階層の外側にある表を参照しようとしているため、正しくありません。

```
SELECT D.DEPTNO, D.DEPTNAME, EMPINFO.AVGSAL, EMPINFO.EMPCOUNT
FROM DEPARTMENT D,
  (SELECT AVG(E.SALARY) AS AVGSAL,COUNT (*) AS EMPCOUNT
     FROM EMPLOYEE E
     WHERE E.WORKDEPT = D.DEPTNO ) AS EMPINFO
```

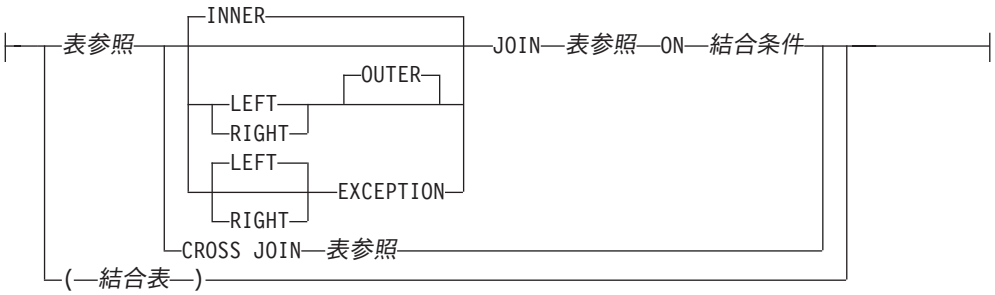
例 2: 次の表関数の例は有効です。

```
SELECT t.c1, z.c5
FROM t, TABLE(tf3 (t.c2 ) ) AS z WHERE t.c3 = z.c4
```

次の例は、t.c2 に対する参照が、FROM 文節内で表関数より右にある表を参照しているため、無効です。

```
SELECT t.c1, z.c5
FROM TABLE(tf6 (t.c2 ) ) AS z, t
WHERE t.c3 = z.c4
```

結合表



結合表 は、内部結合、外部結合、クロス結合、または例外結合のいずれかの結果である中間結果表を指定します。この表は、結合演算子 INNER、 LEFT OUTER、 RIGHT OUTER、 LEFT EXCEPTION、 RIGHT EXCEPTION、または CROSS の 1 つをそのオペランドに適用することによって得られます。

結合演算子を指定しないと、暗黙に INNER になります。複数の結合が行われる順序は、結果に影響する可能性があります。結合の中で、他の結合をネストすることができます。結合の処理順序は、通常は左から右ですが、必須結合条件の位置によって決まります。ネストされた結合の順序を読みやすくするために、括弧を使用することをお勧めします。次の例を見てください。

```
TB1 LEFT JOIN TB2 ON TB1.C1=TB2.C1
    LEFT JOIN TB3 LEFT JOIN TB4 ON TB3.C1=TB4.C1
           ON TB1.C1=TB3.C1
```

これは、次と同じです。

```
(TB1 LEFT JOIN TB2 ON TB1.C1=TB2.C1)
     LEFT JOIN (TB3 LEFT JOIN TB4 ON TB3.C1=TB4.C1)
          ON TB1.C1=TB3.C1
```

内部結合は、結合条件が真の行だけを保持して、左表の各行を右表の各行と結合します。したがって、結果表には、結合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。外部結合は、内部結合によって生成された行に加えて、外部結合のタイプに応じた欠落行が含まれています。例外結合には、次のように、例外結合のタイプに応じた欠落行だけが含まれています。

- *左外部*。 内部結合から欠落した左表からの行が含まれています。
- *右外部*。 内部結合から欠落した右表からの行が含まれています。
- *左例外*。 内部結合から欠落した左表からの行だけが含まれています。
- *右例外*。 内部結合から欠落した右表からの行だけが含まれています。

結合表は、任意の形式の SELECT ステートメントが使用されている、どのような文脈でも使用することができます。ビューまたはカーソルは、その SELECT ステートメントが結合表を含んでいる場合は、読み取り専用です。

結合条件: *結合条件* は、次の規則に適合した*検索条件* です。

- 結合条件には、限定副照会、副選択を伴う IN 述部、または EXISTS 副照会を含めることはできません。基本述部副照会およびスカラー副選択は含めることができます。
- 各列名は、*FROM 文節* の表の 1 つの列を明確に示すものでなければなりません。
- 列関数は、式 で使用することはできません。

どのタイプの結合の場合も、まず 115 ページの『列名』 で指定された列名修飾子を解決するための規則を適用して、結合条件の式の中の列参照を解決した後、列がどの表に属するかに関する規則を適用します。

結合演算: *結合条件* は、T1 と T2 の組み合わせを指定します。ここで、T1 と T2 は、*結合条件* の JOIN 演算子の左右のオペランド表です。T1 と T2 の行の可能な組み合わせに対して、*結合条件* が真の場合、T1 の行と T2 の行が結合されます。T1 の行と T2 の行が結合された場合、結果の行は、T1 のその行の値と T2 のその行の値が連結されたものになります。実行によって、ヌル行が生成されることもあります。列がヌル値を許すかどうかに関係なく、表のヌル行は、表の各列のヌル値から成ります。

INNER JOIN または **JOIN**

T1 INNER JOIN T2 の結果は、それぞれの対にされた行から構成されます。

結合条件 を指定した INNER JOIN 構文を使用すると、FROM 文節にコンマで区切った 2 つの表をリストし、条件を提示するために *WHERE 文節* を使用して、結合を指定した場合と同じ結果が生じます。

LEFT JOIN または **LEFT OUTER JOIN**

T1 LEFT OUTER JOIN T2 の結果は、それぞれの対にされた行と、T1 の対に

されない各行について、その行と T2 のヌル行を連結したものから構成されます。T2 から派生した行はすべてヌル値を許します。

RIGHT JOIN または **RIGHT OUTER JOIN**

T1 RIGHT OUTER JOIN T2 の結果は、それぞれの対にされた行と、T2 の対にされない各行について、その行と T1 のヌル行を連結したものから構成されます。T1 から派生した行はすべてヌル値を許します。

LEFT EXCEPTION JOIN および **EXCEPTION JOIN**

T1 LEFT EXCEPTION JOIN T2 の結果は、T1 の対にされない各行について、その行と T2 のヌル行を連結したものだけから構成されます。T2 から派生した行はすべてヌル値を許します。

RIGHT EXCEPTION JOIN

T1 RIGHT EXCEPTION JOIN T2 の結果は、T2 の対にされない各行について、その行と T1 のヌル行を連結したものだけから構成されます。T1 から派生した行はすべてヌル値を許します。

CROSS JOIN

T1 CROSS JOIN T2 の結果は、T1 の各行が T2 の各行と対にされたものから構成されます。 CROSS JOIN は、カルテシアン積とも呼ばれます。

WHERE 文節

WHERE 文節は、*検索条件* を満たす R の行からなる中間結果表を指定します。 R は、ステートメントの FROM 文節の結果です。

検索条件 は、次の規則を満たすものでなければなりません。

- それぞれの*列名* は、R 内の列を明確に識別するか、または相関参照でなければなりません。*列名* が外側の副選択で識別される表、ビュー、共通表式、または派生表の列を識別する場合、その列名は相関参照になります。
- HAVING 文節の副照会に WHERE 文節が指定され、その関数の引き数がグループに対する相関参照である場合を除いて、列関数を指定することはできません。

WHERE 文節を含むステートメントの実行時に *HEX 以外のソート順序が有効で、しかもその検索条件に SBCS データ、混合データまたは UCS-2 データを持つ述部含まれている場合は、それらの述部の比較は、重み付けされた値を使用して行われます。重み付けされた値は、述部のオペランドに対して該当のソート順序を適用することにより導き出されます。

検索条件 における副照会は、R の各行について有効に実行され、その結果は、R の所定の行に対する*検索条件* の適用で使用されます。副照会が R の各行ごとに実際に実行されるのは、副照会が R の列に対する相関参照を含んでいる場合だけに限られます。実際、相関参照のない副照会は 1 回しか実行されないのに対して、相関参照のある副照会は、各行ごとに 1 回ずつ実行しなければならない場合があります。

GROUP-BY 文節

GROUP BY 文節は、R のグループ化された行で構成される中間結果表を指定します。R は、副選択の直前の文節の結果です。

最も単純な形式では、GROUP BY 文節は 1 つの*グループ化式* を持ちます。*グループ化式* は、R のグループ化の定義で使用される式 です。*グループ化式* に入れる各列名は、R の行を明確に識別する必要があります。*グループ化式* には LOB およびデータ・リンク列は使用できません。各*グループ化式* の長さ属性は、2000、または列がヌル可能な場合は 1999 を超えてはなりません。*グループ化式* には、非決定性の関数、あるいは RRN、PARTITION、NODENAME、または NODENUMBER 関数を入れることはできません。

GROUP BY の結果は、行のグループの集まりになります。複数行を持つグループそれぞれにおいて、各*グループ化式* の値はすべて等しく、*グループ化式* の値の集合が同じ行は、すべて同じグループに入ります。グループ化では、*グループ化式* のヌル値はすべて等しいものと見なされます。

GROUP BY 文節を含むステートメントの実行時に *HEX 以外のソート順序が有効な場合、行は、重み付けされた値を使用してグループ化されます。重み付けされた値は、SBCS データ・*グループ化式*、混合データ・*グループ化式* の SBCS データ、および UCS-2 データ・*グループ化式* にソート順序を適用することによって求められます。

グループ化式 は、HAVING 文節、SELECT 文節、または ORDER BY 文節のソート・キー式 (詳細は ORDER BY 文節を参照) の検索条件で使用することができます。どの場合も、参照は各グループの 1 つの値だけを指定します。これらの文節に指定される*グループ化式* は、GROUP BY 文節の中の*グループ化式* と厳密に一致する必要があります。ただし、ブランクは意味を持ちません。例えば、次の*グループ化式* は、

```
SALARY*.10
```

次の *HAVING 文節* の式に一致します。

```
HAVING SALARY*.10
```

しかし、次とは一致しません。

```
HAVING .10 *SALARY
 または
HAVING (SALARY*.10)+100
```

グループ化式 の中に後書きブランク付きの可変長ストリングがある場合は、後書きブランクの数が異なるために値の長さが同じにならないことがあります。この場合も、*グループ化式* に対する参照では 1 つのグループについて 1 つの値だけを指定

します。ただし、グループ化のために使用する値は、使用可能な値の組から任意に選択されることになります。したがって、選択された値の実際の長さは、予期できないものになります。

GROUP BY 文節には、最高で 120 *グループ化式* または 2000 - n バイト (n は、ヌル値を許すと指定された*グループ化式* の数) を指定することができます。

HAVING 文節

HAVING 文節は、*検索条件* に該当する R のグループからなる中間結果表を指定します。 R は、副選択の直前の文節の結果です。この直前の文節が GROUP BY 文節でない場合、R はグループ化式のない単一グループとして扱われます。

検索条件に*列名* を含むそれぞれの式は、次のいずれかを満たす必要があります。

- R のグループ化式を明確に識別すること。
- 列関数の中に指定されていること。
- 相関参照であること。*列名* が外側の副選択で識別される表、ビュー、共通表式、または派生表の列を識別する場合、その列名は相関参照になります。

RRN、PARTITION、NODENAME、および NODENUMBER 関数は、列関数の内部にある場合を除いて、HAVING 文節に指定することはできません。列関数を使用する際の制約事項については、第 3 章の『関数』を参照してください。

HAVING 文節を含むステートメントの実行時に *HEX 以外のソート順序が有効で、しかもその検索条件に SBCS データ、混合データまたは UCS-2 データを持つ述部が含まれている場合は、それらの述部の比較は、重み付けされた値を使用して行われます。この重み付けされた値は、述部の中のオペランドに対して該当のソート順序を適用して求められます。

検索条件の中のそれぞれの列関数 (引き数が相関参照であるものを除く) には、検索条件が適用される R のグループから引き数が与えられます。

検索条件の中に副照会がある場合は、R のグループに検索条件が適用されるたびにその副照会が実行され、その副照会の結果が、検索条件を適用する際に使用されると考えても構いません。実際には、副照会が各グループごとに実行されるのは、その副照会の中に相関参照がある場合だけです。この相違については、352 ページの『副選択の例』の例 6 および 7 を参照してください。

R のグループに対する相関参照では、グループ化列を識別しなければなりません。グループ化列を識別しない場合は、相関参照を列関数の中に入れる必要があります。

GROUP BY を指定せずに HAVING を使用する場合は、選択リスト内のすべての列名を列関数の中に入れなければなりません。

副選択の例

例 1

表 EMPLOYEE から、すべての列および行を選択します。

```
SELECT * FROM EMPLOYEE
```

例 2

表 EMPPROJACT と EMPLOYEE を結合し、表 EMPPROJACT のすべての行を選択し、結果の各行に表 EMPLOYEE からの社員の姓 (LASTNAME) を加えます。

```
SELECT EMPPROJACT.*, LASTNAME
  FROM EMPPROJACT, EMPLOYEE
  WHERE EMPPROJACT.EMPNO = EMPLOYEE.EMPNO
```

例 3

表 EMPLOYEE と表 DEPARTMENT を結合します。誕生日 (BIRTHDATE) が 1930 年より前であるすべての社員の従業員番号 (EMPNO)、社員のラストネーム (LASTNAME)、部門番号 (表 EMPLOYEE の WORKDEPT と表 DEPARTMENT 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
```

この副選択は、次のようにも書けます。

```
SELECT EMPNO, LASTNAME, WORKDEPT, DEPTNAME
  FROM EMPLOYEE INNER JOIN DEPARTMENT
  ON WORKDEPT = DEPTNO
  WHERE YEAR(BIRTHDATE) < 1930
```

例 4

表 EMPLOYEE において、同一のジョブ・コードを持つ行のグループごとに、ジョブ (JOB) と、給与 (SALARY) の最高額および最低額を選択します。ただし、対象となるグループは、複数の行があり、給与の最高額が 27000 以上であるものに限ります。

```
SELECT JOB, MIN(SALARY), MAX(SALARY)
  FROM EMPLOYEE
  GROUP BY JOB
  HAVING COUNT(*) > 1 AND MAX(SALARY) >= 27000
```

例 5

表 EMPPROJACT から、部門 (WORKDEPT) 'E11' の社員 (EMPNO) に関するすべての行を選択します。(社員の部門番号は、表 EMPLOYEE に示されています。)

```
SELECT * FROM EMPPROJACT
  WHERE EMPNO IN (SELECT EMPNO
                  FROM EMPLOYEE
                  WHERE WORKDEPT = 'E11')
```

例 6

表 EMPLOYEE から、部門別給与 (SALARY) の最高額が全社員の平均給与より少ないすべての部門について、その部門番号 (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 7

表 EMPLOYEE を使用して、部門別給与 (SALARY) 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について、その部門番号 (WORKDEPT) と部門別給与 (SALARY) の最高額を選択します。

```
SELECT WORKDEPT, MAX(SALARY)
  FROM EMPLOYEE EMP_COR
  GROUP BY WORKDEPT
  HAVING MAX(SALARY) < (SELECT AVG(SALARY)
       FROM EMPLOYEE
       WHERE NOT WORKDEPT = EMP_COR.WORKDEPT)
```

例 6 とは対照的に、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。

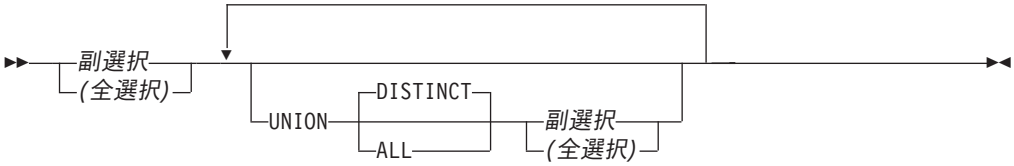
例 8

表 EMPLOYEE と EMPPROJACT を結合し、社員全員とそのプロジェクト番号をすべて選択します。現在割り当てられているプロジェクト番号がない社員も戻します。

```
SELECT EMPLOYEE.EMPNO, PROJNO
  FROM EMPLOYEE LEFT OUTER JOIN EMPPROJACT
  ON EMPLOYEE.EMPNO = EMPPROJACT.EMPNO
```

表 EMPLOYEE 内の社員で、表 EMPPROJACT にプロジェクト番号がない者も、EMPNO の値が入っている結果表の 1 行、および PROJNO の列のヌル値を戻します。

全選択



全選択 は、結果表を指定するものです。UNION を使用しない場合は、指定した副選択の結果が全選択の結果となります。

UNION DISTINCT または **UNION ALL**

他の 2 つの結果表 (R1 および R2) を組み合わせることによって、1 つの結果表を作成します。UNION ALL を指定すると、結果表には R1 および R2 のすべての行が入ります。ALL オプションを付けずに UNION を指定すると、R1

または R2 にあるすべての行の集合から重複行を除去したものが結果表に入ります。ただし、どちらの場合も、UNION 表の各行は R1 または R2 のいずれかから得られた行です。

結果の列は、以下のように命名されます。

- R1 の n 番目の列と R2 の n 番目の列が、同じ結果の列名を持つ場合には、結果表の n 番目の列はその列名を持ちます。
- R1 の n 番目の列と R2 の n 番目の列が同じ名前でない場合は、結果表の n 番目の列には名前が付きません。

2 つの行が重複していると見なされるのは、一方の行のそれぞれの値が、もう一方の行の対応する値にすべて等しい場合です。UNION キーワードを含むステートメントの実行時に *HEX 以外のソート順序が有効で、しかもその結果表に SBCS データ、UCS-2、または混合データを持つ列が含まれている場合は、それらの列の比較は、重み付けされた値を使用して行われます。この重み付けされた値は、それぞれの値に該当のソート順序を適用して求められます。(重複を判別する際には、2 つのヌル値は互いに等しいものと見なされます。)

UNION および UNION ALL は、両方とも結合演算です。1 つの SQL ステートメントの中に UNION ALL 演算子と UNION 演算子を両方とも使用すると、その結果は評価の順序によって異なります。括弧がない場合には、左から右の順に評価されます。括弧がある場合は、最初に括弧で囲まれている副選択が評価され、その後でステートメントの他のコンポーネントが左から右に評価されます。

列に関する規則: R1 と R2 の列の数は同じでなければなりません。また、R1 の n 番目の列のデータ・タイプと R2 の n 番目の列のデータ・タイプには、互換性がなければなりません。文字ストリング値は、日付/時刻の値と互換性がありません。

UNION および UNION ALL の結果の n 番目の列は、R1 および R2 の n 番目の列から得られます。結果列の属性は、結果列に関する規則を使用して決定します。詳細については、99 ページの『結果のデータ・タイプに関する規則』の項を参照してください。

UNION を指定するときは、どの列も LOB またはデータ・リンクであってはなりません。

括弧で囲んだ全選択を*副照会* といいます。例えば、*副照会* は検索条件の中で使用することができます。

全選択の例

例 1

表 EMPLOYEE から、すべての列および行を選択します。

```
SELECT * FROM EMPLOYEE
```

例 2

EMPLOYEE 表で、部門番号 (WORKDEPT) が 'E' で始まる社員、**または** EMPPROJACT 表でプロジェクト番号 (PROJNO) が 'MA2100'、'MA2110'、または 'MA2112' に等しいプロジェクトに参画している社員全員の従業員番号 (EMPNO) をリストします。

```
SELECT EMPNO FROM EMPLOYEE
  WHERE WORKDEPT LIKE 'E%'
UNION
SELECT EMPNO FROM EMPPROJACT
  WHERE PROJNO IN('MA2100', 'MA2110', 'MA2112')
```

例 3

例 2 と同じ照会を行うのに加えて、EMPLOYEE 表からの行に 'emp'、EMPPROJACT 表からの行に 'empprojact' の「タグ」を付けます。例 2 からの結果とは異なり、この照会は、関連の「タグ」 によって表を識別して、同じ EMPNO を 2 度以上戻すことがあります。

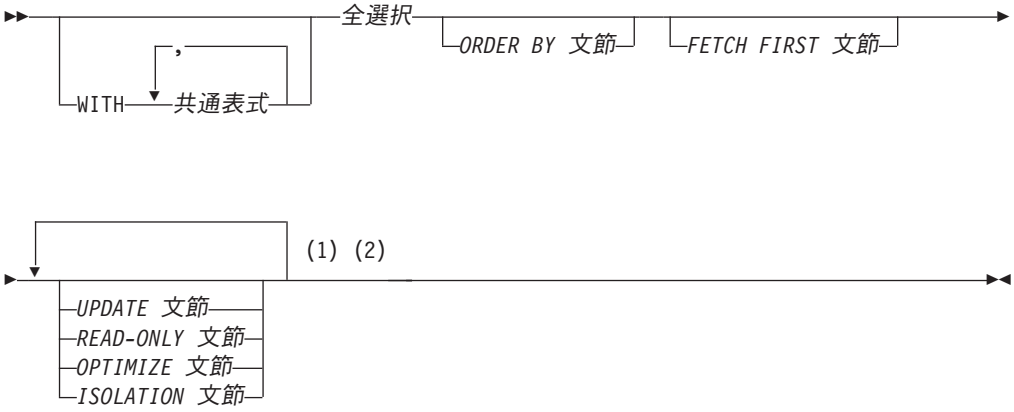
```
SELECT EMPNO, 'emp' FROM EMPLOYEE
  WHERE WORKDEPT LIKE 'E%'
UNION
SELECT EMPNO, 'empprojact' FROM EMPPROJACT
  WHERE PROJNO IN('MA2100', 'MA2110', 'MA2112')
```

例 4

例 2 と同じ照会を行いますが、重複行が除去されないように、UNION ALL を使用しています。

```
SELECT EMPNO FROM EMPLOYEE
  WHERE WORKDEPT LIKE 'E%'
UNION ALL
SELECT EMPNO FROM EMPPROJACT
  WHERE PROJNO IN('MA2100', 'MA2110', 'MA2112')
```

選択ステートメント



注:

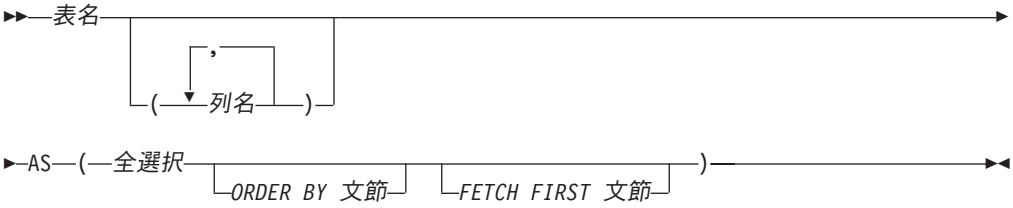
1 UPDATE 文節と READ-ONLY 文節は、同一の選択ステートメントで両方をともに指定することはできません。

2 各文節はそれぞれ 1 回だけ指定できます。

*選択*ステートメント は、照会の 1 つの形式で、DECLARE CURSOR ステートメントに直接指定するか、または準備を行い、その後で DECLARE CURSOR ステートメントで参照することができます。このステートメントは、対話式機能 (STRSQL

コマンド) を使用して対話式に行うこともできます。この場合、結果表はユーザーのワークステーションに表示されます。どちらの場合も、*選択ステートメント* で指定した表が、全選択の結果となります。

共通表式



共通表式 を使用すると、次に続く全選択の FROM 文節で表として指定できる*表名* を持つ結果表を定義することができます。*表名* には修飾をしてはなりません。 1 つの WITH キーワードに続けて、複数の共通表式を指定することができます。指定された各共通表式は、後続の共通表式の FROM 文節でその名前を参照することもできます。

列名のリストを指定する場合は、そのリストは、全選択の結果表にある列の数と同じ数の列名で構成されている必要があります。それぞれの*列名* は固有でなければならず、修飾は付けられません。これらの列名を指定しない場合は、列名は、共通表式を定義するために使用される副選択の選択リストから取得されます。

共通表式の*表名* は、同じステートメント内の他の共通表式の表名とは異なっている必要があります。共通表式の*表名* は、全選択におけるどの FROM 文節の表名として指定してもかまいません。共通表式の*表名* は、同じ修飾名を持つ既存の表、ビュー、または別名 (カタログ内の) を上書きします。

同一ステートメントの中で複数の共通表式を定義した場合、それらが共通表式間で相互に参照し合うことはできません。*相互参照* が起きるのは、2 つの共通表式の *dt1* と *dt2* が、*dt1* は *dt2* を参照し、また、*dt2* は *dt1* を参照するように作成されている場合です。

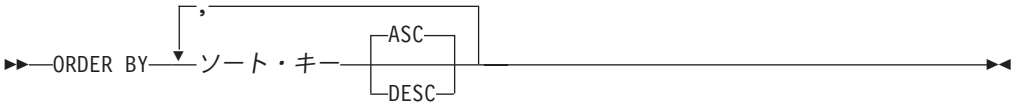
共通表式 は、CREATE VIEW および INSERT ステートメントでも全選択の前に任意に指定することができます。

共通表式 は、次の場合に使用できます。

- ビューの代わりとして。これはそのビューを作成するのを避けるためです (そのビューの一般的な使用が要求されず、定位置更新または削除が使用されない場合)。
- 必要な結果表がホスト変数に基づいているとき
- 同じ結果表を*全選択* で共用するとき

FROM 文節内で、共通表式の*全選択* がそれ自身に対する参照を含んでいる場合は、その共通表式は*反復表式* です。反復共通表式は、DB2 UDB for iSeries ではサポートされません。

ORDER BY 文節



ソート・キー:

ORDER BY 文節は、結果表の行の順序付けを指定します。1 つのソート・キー を指定した場合は、行はそのソート・キー の値によって順に並べられます。複数のソート・キー を指定した場合は、行は、最初に指定されたソート・キー の値によって、次に 2 番目に指定されたソート・キー の値によって、以下同様の値によって順に並べられます。

ORDER BY 文節を含むステートメントの実行時に *HEX 以外のソート順序が有効で、しかもその ORDER BY 文節に SBCS データ、UCS-2、または混合データを持つソート・キー が含まれている場合は、それらのソート・キー の比較は、重み付けされた値を使用して行われます。この重み付けされた値は、ソート・キー の値に該当のソート順序を適用して求められます。

選択リスト内の名前のある列は、*単純整数* または*単純列名* のソート・キー で識別することができます。選択リスト内の名前のない列は、*単純整数* またはソート・キー式 で識別することができます。選択リストに AS 文節の指定がない場合、および列が定数、演算子を伴う式、または関数によって得られる場合は、列には名前がありません。UNION 演算子を含む全選択の場合に、全選択における列の名前についての規則は、353 ページの 『全選択』を参照してください。

単純列名

通常、結果表の 1 つの列を識別します。その場合、*単純列名* は、選択リスト内の名前付き列の名前でなければなりません。列は、LOB 列や DATALINK 列にすることはできません。 全選択に UNION または UNION ALL が含まれる場合は、列名は修飾できません。

照会が副選択である場合は、*単純列名* は、FROM 文節に指定されている表、ビュー、またはネストされた表の列名も識別することができます。副選択の結果がグループであるのに、*単純列名* が*グループ化式* でない場合は、エラーになります。

整数

0 より大きく、結果表の列の数以下の値を指定しなければなりません。整数 *n* は、結果表の *n* 番目の列を識別します。この識別された列は、LOB またはデータ・リンクであってはなりません。

ORDER BY 文節

ソート・キー式
単純な 1 つの列名または無符号の整数定数ではない式。順序付けを適用する照会では、この形式のソート・キー を使用するために副選択にする必要があります。

全選択 に UNION または UNION ALL が含まれる場合は、*ソート・キー式* に RRN、PARTITION、NODENAME、または NODENUMBER を含めることはできません。*ソート・キー式* の結果は、LOB またはデータ・リンクであってはなりません。

副選択をグループ化する場合、*ソート・キー式* は、次のようにすることができます。

- 副選択の選択リスト内の式とする
- 副選択の GROUP BY 文節からの*グループ化式* を含む

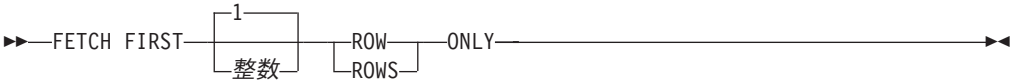
ASC
列の値を昇順に使用します。これは、デフォルト値です。

DESC
列の値を降順に使用します。

順序付けは、第 2 章で説明した比較規則にしたがって行われます。ヌル値は、他のどの値よりも上位に置かれます。ユーザーの順序付けの指定によって完全な順序が判別できない場合は、最後に指定されたソート・キー の値が重複する行は、順序が不定になります。また、ORDER BY 文節を指定しなかった場合は、結果表の行の順序が不定になります。

ORDER BY 文節には、最高で 10000-n ソート・キー または 10000-n バイト (n は、ヌル値を許すと指定されたソート・キー の数) を指定できます。

FETCH FIRST 文節



FETCH FIRST 文節 は、取り出すことができる行の最大数を設定します。データベース・マネージャーは、この文節が指定されていない場合に結果表に何行入るかに関係なく、アプリケーションが*整数* 行より多く行を取り出すことを望んでいないものと判断します。*整数* 行を超えて取り出そうとすると、通常のデータの終わりと同じ方法で処理されます。 *整数* の値は、正整数 (ゼロでない) でなければなりません。

結果表を最初の*整数* 行に制限すると、パフォーマンスが向上します。最初の*整数* 行を判別すると、データベース・マネージャーは照会の処理をやめます。

SELECT ステートメントで *FETCH FIRST 文節* を指定すると、結果表は読み取り専用になります。読み取り専用の結果表は、UPDATE または DELETE ステートメントで参照してはなりません。

FETCH FIRST 文節 は、UPDATE 文節が入っているステートメント中では使用できません。

FETCH FIRST 文節 と *ORDER BY 文節* の両方が指定されている場合、最初の整数行を戻す前に、結果セット全体に対する順序付けが行われます。

UPDATE 文節

UPDATE 文節は、以後の位置指定 UPDATE ステートメントで更新することができる列を識別します。各*列名* は、それぞれ修飾のない名前でなければならず、全選択の最初の FROM 文節で識別されている表またはビューの 1 つの列を識別するものでなければなりません。 列名を指定しないで UPDATE 文節を指定した場合は、全選択の最初の FROM 文節で識別されている表またはビューの更新可能な列がすべて含まれます。この文節は、付随する OPTIMIZE 文節の前にあっても後ろにあっても構いません。

全選択の結果表が読み取り専用である場合、(詳細は、 597 ページの『DECLARE CURSOR』を参照)、FOR READ ONLY 文節が使用されている場合、または DECLARE CURSOR ステートメントで、DYNAMIC キーワードを指定しないで SCROLL キーワードを指定した場合には、FOR UPDATE OF 文節を指定してはなりません。

選択ステートメントに関連するカーソルを識別する位置指定 UPDATE ステートメントは、以下の場合に更新可能なすべての列を更新することができます。

- 選択ステートメントに、次の中の 1 つが含まれていない。
  - UPDATE 文節
  - FOR READ ONLY 文節
  - ORDER BY 文節
- DECLARE CURSOR ステートメントに DYNAMIC キーワードを伴わない SCROLL キーワードが含まれていない。

UPDATE 文節は、ISO/ANSI SQL には含まれていないパフォーマンス・オプションです。

READ-ONLY 文節

READ-ONLY 文節

FOR READ ONLY または FOR FETCH ONLY 文節は、結果表が読み取り専用であることを示します。例えば、そのカーソルは、位置指定 DELETE または UPDATE ステートメントでは使用されません。

結果表によっては、本来、読み取り専用の表があります (例えば、読み取り専用のビューに基づく表)。そのような表についても FOR READ ONLY を指定することはできますが、この指定は効力をもちません。

更新や削除が許されている結果表の場合は、FOR READ ONLY を指定すると、データベース・マネージャーによるブロッキングが可能になり、排他的なロックが回避されるので、FETCH 操作のパフォーマンスが向上することがあります。例えば、FOR READ ONLY 文節または ORDER BY 文節のない動的 SQL ステートメントを含むプログラムでは、データベース・マネージャーは、DYNAMIC キーワードの指定のない SCROLL が指定されていないカーソルを、FOR UPDATE OF 文節が指定されている場合のようにオープンすることになります。

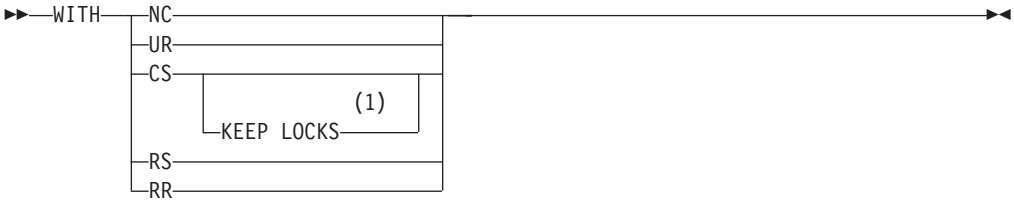
本来読み取り専用であるか、または FOR READ ONLY として指定されているかには関係なく、読み取り専用の結果表は、UPDATE または DELETE ステートメントで参照してはなりません。

FOR READ ONLY 文節は、UPDATE 文節が入っているステートメント中では使用できません。この文節は、付随する OPTIMIZE 文節の前にあっても後ろにあっても構いません。

OPTIMIZE 文節

OPTIMIZE 文節は、プログラムが整数 で指定された行数を超えて結果表から検索を行う意図はないことと想定するように、データベース・マネージャーに伝えます。この文節がない場合、またはキーワード ALL の指定がある場合は、データベース・マネージャーは、結果表の行をすべて検索するものと想定し、この方針に沿って最適化します。取り出される行の数を指定の整数 行に最適化する、すなわち最小値にすると、パフォーマンスが向上する可能性があります。この文節によって、結果表や行を取り出す順序が変更されるわけではありません。任意の数の行を取り出すことができますが、指定の整数 回の取り出しの後は、パフォーマンスが下がる可能性があります。整数 の値は、正整数 (ゼロでない) でなければなりません。この文節は、付随する UPDATE 文節または READ-ONLY 文節の前でも後でもかまいません。

ISOLATION 文節



注:

1 KEEP LOCKS 文節は、DECLARE CURSOR、SELECT INTO および全選択ステートメントでのみ使用することができます。

ISOLATION 文節は、SELECT、SELECT INTO、INSERT、UPDATE、DELETE、および DECLARE CURSOR ステートメントで分離レベルを指定します。この分離レベルが有効なのは、ISOLATION 文節を含むステートメントを実行する場合だけです。分離レベルの詳細については、分離レベルを参照してください。

KEEP LOCKS 文節は、獲得したどの読み取りロックも長期間保持することを指定するものです。通常は、読み取りロックは、次の行が読み取られると解除されます。ISOLATION 文節がカーソルに関連付けられている場合は、ロックは、そのカーソルがクローズされるか、または COMMIT か ROLLBACK ステートメントが実行されるまで保持されます。関連付けられていない場合は、ロックは、この SQL ステートメントの完了まで保持されます。KEEP LOCKS 文節は、SQL SELECT、SELECT INTO または DECLARE CURSOR ステートメントで使用できます。この文節は、カーソルが更新可能な場合は使用できません。

同義のキーワード : 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- キーワード NONE を NC の同義語として使用することができます。
- キーワード CHG を UR の同義語として使用することができます。
- キーワード ALL を RS の同義語として使用することができます。

選択ステートメントの例

例 1

表 EMPLOYEE から、すべての列および行を選択します。

```
SELECT * FROM EMPLOYEE
```

例 2

表 PROJECT から、プロジェクト名 (PROJNAME)、開始日付 (PRSTDATE)、および終了日付 (PRENDATE) を選択します。結果表を終了日付によって順序付けして、終了日付の最も早いプロジェクトが先頭になるようにしています。

```
SELECT PROJNAME, PRSTDATE, PRENDATE
  FROM PROJECT
  ORDER BY PRENDATE DESC
```

例 3

表 EMPLOYEE のすべての部門について、部門番号 (WORKDEPT) および給与 (SALARY) の部門別平均額を選択します。結果表は、部門別平均給与によって昇順に並べます。

```
SELECT WORKDEPT, AVG(SALARY)
  FROM EMPLOYEE
  GROUP BY WORKDEPT
  ORDER BY 2
```

例 4

UP_CUR という名前のカーソルを C プログラムで使用するように宣言します。これによって、PROJECT 表内の開始日付 (PRSTDATE) と終了日付 (PRENDATE) の列が更新されます。このプログラムでは、行ごとのプロジェクト番号 (PROJNO) の値と一緒に PRSTDATE と PRENDATE の両方の値を受け取らなければなりません。宣言では、照会のアクセス・パスを最大 2 行の検索に最適化するように指定しています。このように最適化しても、プログラムは結果表から 3 行以上検索することができます。ただし、3 行以上検索すると、パフォーマンスが下がる可能性があります。

```
EXEC SQL  DECLARE UP_CUR CURSOR FOR
            SELECT PROJNO, PRSTDATE, PRENDATE
              FROM PROJECT
              FOR UPDATE OF PRSTDATE, PRENDATE
              OPTIMIZE FOR 2 ROWS ;
```

例 5

この例は、式 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

販売担当者の従業員数と給与、およびそれぞれの部門の平均給与と人数を調べます。平均給与が最高の部門の平均給与もリストします。

この場合、共通表式を使用すれば、DINFO ビューを正規ビューとして作成するオーバーヘッドを節約できます。全選択の残りのコンテキストから、ビューでは、販売担当者の部門の行だけを考慮すれば済みます。

```
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

分離レベルが反復可能読取り (RS, ALL) の表から項目を選択します。

```
SELECT NAME, SALARY
  FROM PAYROLL
  WHERE DEPT = 704
  WITH RS
```

第 5 章 ステートメント

この章には、以下の表にリストしている SQL ステートメントの構文図、意味の説明、規則、および使用例を示しています。

表 32. SQL スキーマ・ステートメント

SQL ステートメント	説明	ページ
ALTER TABLE	表の記述を変更します。	376
COMMENT	別名、列、関数、索引、パッケージ、パラメーター、プロシージャー、表、タイプ、またはビューの記述に対してコメントの置換や追加を行います。	412
CREATE ALIAS	別名を作成します。	436
CREATE DISTINCT TYPE	特殊タイプを作成します。	439
CREATE FUNCTION	ユーザー定義の関数を作成します。	446
CREATE FUNCTION (外部スカラー)	外部スカラー関数を作成します。	450
CREATE FUNCTION (外部表)	外部表関数を作成します。	467
CREATE FUNCTION (ソース化)	別の既存のスカラー関数や列関数にもとづいて、ユーザー定義の関数を作成します。	482
CREATE FUNCTION (SQL スカラー)	SQL スカラー関数を作成します。	490
CREATE FUNCTION (SQL 表)	SQL 表関数を作成します。	499
CREATE INDEX	表上の索引を作成します。	508
CREATE PROCEDURE	プロシージャーを作成します。	512
CREATE PROCEDURE (外部)	外部プロシージャーを作成します。	514
CREATE PROCEDURE (SQL)	SQL プロシージャーを作成します。	527
CREATE SCHEMA	スキーマ、およびそのスキーマ内の一連のオブジェクトを作成します。	537

表 32. SQL スキーマ・ステートメント (続き)

SQL ステートメント	説明	ページ
CREATE TABLE	表を作成します。	542
CREATE TRIGGER	トリガーを作成します。	575
CREATE VIEW	1 つまたは複数の表あるいはビューの、1 つのビューを作成します。	589
DROP	別名、関数、索引、パッケージ、プロシージャー、スキーマ、表、トリガー、タイプ、またはビューを除去します。	653
GRANT (特殊タイプ特権)	特殊タイプに対する特権を認可します。	681
GRANT (関数またはプロシージャー特権)	関数やプロシージャーに対する特権を付与します。	684
GRANT (パッケージ特権)	パッケージに関する特権を認可します。	692
GRANT (表特権)	表またはビューに関する特権を認可します。	695
LABEL	別名、列、パッケージ、表、またはビューの記述に対して、ラベルの置換や追加を行います。	714
RENAME	表、ビュー、または索引の名前を変更します。	739
REVOKE (特殊タイプ特権)	特殊タイプを使用する特権を取り消します。	742
REVOKE (関数またはプロシージャー特権)	関数やプロシージャーに対する特権を取り消します。	744
REVOKE (パッケージ特権)	パッケージ内でステートメントを実行する特権を取り消します。	751
REVOKE (表特権)	表またはビューに関する特権を取り消します。	753

表 33. SQL データ変更ステートメント

SQL ステートメント	説明	ページ
DELETE	表から 1 つまたは複数の行を削除します。	636
INSERT	表に 1 行または複数行を挿入します。	705
UPDATE	表の 1 つまたは複数の行にある、1 つまたは複数の列の値を更新します。	801

表 34. SQL データ・ステートメント

SQL ステートメント	説明	ページ
	すべての SQL データ変更ステートメント	366 ページの表 33
CLOSE	カーソルをクローズします。	410
DECLARE CURSOR	SQL カーソルを定義します。	597
FETCH	結果表の行にカーソルを位置付けます。また、結果表の 1 行または複数行の値をホスト変数に割り当てることもできます。	672
FREE LOCATOR	LOB ロケーター変数とその値との関連を除去します。	680
HOLD LOCATOR	作業単位が変わっても、LOB ロケーター変数が値との関連を保持できるようにします。	701
LOCK TABLE	同時に実行されているプロセスによる表の変更を防止するか、または表の使用を防止します。	718
OPEN	カーソルをオープンします。	720
SELECT	照会を実行します。	763
SELECT INTO	ホスト変数に値を割り当てます。	764
SET 遷移変数	遷移変数に値を割り当てます。	796
SET 変数	ホスト変数に値を割り当てます。	798
VALUES	トリガーからユーザー定義関数を呼び出す方式を提供します。	811
VALUES INTO	1 行だけで構成される結果表を指定し、それらの値をホスト変数に割り当てます。	813

表 35. SQL トランザクション・ステートメント

SQL ステートメント	説明	ページ
COMMIT	作業単位を終了させ、その作業単位によって行われたデータベースの変更をコミットします。	422
RELEASE SAVEPOINT	作業単位内の保管ポイントを解放します。	738
ROLLBACK	作業単位を終了させ、その作業単位によって行われたデータベースの変更をバックアウトします。	757
SAVEPOINT	作業単位内に保管ポイントをセットします。	761
SET TRANSACTION	現行作業単位の分離レベルを変更します。	793

ステートメント

表 36. SQL 接続ステートメント

SQL ステートメント	説明	ページ
CONNECT (タイプ 1)	サーバーに接続し、リモート作業単位のルール (規則) を確立します。	425
CONNECT (タイプ 2)	サーバーに接続し、アプリケーション指向の分散作業単位のルール (規則) を確立します。	431
DISCONNECT	1 つまたは複数の接続をただちに終了させます。	650
RELEASE	1 つまたは複数の接続を解放保留状態にします。	736
SET CONNECTION	既存の接続の 1 つを識別して、そのプロセスのサーバーを確立します。	767

表 37. SQL 動的ステートメント

SQL ステートメント	説明	ページ
DESCRIBE	準備済みステートメントの結果列を記述します。	642
DESCRIBE TABLE	表またはビューに関する情報を取得します。	647
EXECUTE	準備済みの SQL ステートメントを実行します。	667
EXECUTE IMMEDIATE	SQL ステートメントを準備して、実行します。	670
PREPARE	SQL ステートメントを実行できるように準備します。	725

表 38. SQL セッション・ステートメント

SQL ステートメント	説明	ページ
DECLARE GLOBAL TEMPORARY TABLE	宣言済みのグローバル一時表を定義します。	605
SET PATH	CURRENT PATH 特殊レジスターに値を割り当てます。	786
SET SCHEMA	CURRENT SCHEMA 特殊レジスターに値を割り当てます。	791

表 39. SQL 組み込みホスト言語ステートメント

SQL ステートメント	説明	ページ
BEGIN DECLARE SECTION	SQL 宣言セクションの開始を示します。	402
DECLARE PROCEDURE	外部プロシージャーを定義します。	621
DECLARE STATEMENT	準備済み SQL ステートメントを識別するための名前を宣言します。	631
DECLARE VARIABLE	ホスト変数に対して、デフォルト値以外のサブタイプまたは CCSID を宣言します。	633
END DECLARE SECTION	SQL 宣言セクションの終わりを示します。	665
INCLUDE	ソース・プログラムに宣言を挿入します。	703
SET RESULT SET	プロシージャーの中の結果セットを識別します。	788
SET OPTION	SQL ステートメントの処理に関するオプションを設定します。	770
WHENEVER	SQL 戻りコードに基づいて、行うべきアクションを定義します。	816

表 40. SQL 制御ステートメント

SQL ステートメント	説明	ページ
割り当てステートメント	出力パラメーターまたはローカル変数に値を割り当てます。	823
CALL	プロシージャーを呼び出します。	404
CASE	複数の条件に基づいて実行パスを選択します。	826
複合 (compound) ステートメント	他のステートメントを 1 つの SQL ルーチンの中にグループとしてまとめます。	828
FOR	表のそれぞれの行ごとにステートメントを実行します。	835
GET DIAGNOSTICS	直前に実行された SQL ステートメントに関する情報を取得します。	837
GOTO	SQL ルーチンまたはトリガーの中のユーザー定義のラベルに分岐します。	840
IF	条件の真理値に基づいて条件付きで実行します。	842
ITERATE	制御のフローをラベル付きループの先頭に戻します。	844

表 40. SQL 制御ステートメント (続き)

SQL ステートメント	説明	ページ
LEAVE	ブロックまたはループを終了することによって実行を継続します。	845
LOOP	ステートメントの実行を繰り返します。	847
REPEAT	ステートメントの実行を繰り返します。	849
RESIGNAL	エラー条件または警告条件を再通知します。	851
RETURN	ルーチンから戻ります。	854
SIGNAL	エラー条件または警告条件を通知します。	856
WHILE	指定された条件が真である間、ステートメントの実行を繰り返します。	859

以下の項も参照してください。

- 『SQL ステートメントの呼び出し方法』
- 373 ページの『SQL の戻りコード』
- 375 ページの『SQL のコメント』

SQL ステートメントの呼び出し方法

この章で説明する SQL ステートメントは、*実行可能* ステートメントと*実行不能* ステートメントに類別されます。各ステートメントの説明の*呼び出し* の項に、そのステートメントが実行可能か否かを示しています。

実行可能ステートメント は、次のいずれかの方法で呼び出すことができます。

- アプリケーション・プログラムの中に組み込む。
- 動的に準備して実行する。
- 対話方式で呼び出す。

注: REXX に組み込まれたステートメントや RUNSQLSTM を使用して処理されるステートメントは、動的に準備され、実行されます。

ステートメントによって、上記の方法のいくつか、またはすべてを使用できるステートメントがあります。各ステートメントの説明の*呼び出し* の項には、呼び出しにどのような方法を使用できるかを示しています。

実行不能ステートメント は、アプリケーション・プログラムの中に組み込むことだけが可能です。

SQL ステートメントには、この章で説明されているステートメントのほかにもう 1 つ、*選択ステートメント* というステートメント構成があります。 355 ページの『選択ステートメント』を参照してください。このステートメントに関しては、他のステートメントと使用方法が異なるため、この章で説明していません。

選択ステートメント には、以下の 3 種類の呼び出し方法があります。

- DECLARE CURSOR に組み込んで、OPEN によって暗黙に実行されるようにする。
- 動的に準備し、DECLARE CURSOR で参照して、OPEN によって暗黙に実行されるようにする。
- 対話方式で呼び出す。

最初の 2 つの方式は、それぞれ*選択ステートメント* の*静的* 呼び出しおよび*動的* 呼び出しと呼ばれます。

アプリケーション・プログラムへのステートメントの組み込み

SQL ステートメントは、 CRTSQLCBL、 CRTSQLCBLI、 CRTSQLCI、 CRTSQLFTN、 CRTSQLCPPI、 CRTSQLPLI、 CRTSQLRPG、または CRTSQLRPGI コマンドを使用することでプリコンパイラーに実行依頼されるソース・プログラムの中に組み込むことができます。 このようなステートメントは、プログラムに*組み込まれた* と表現されます。組み込みステートメントは、ホスト言語のステートメントを使用できる環境であれば、プログラム中のどこにでも入れることができます。各組み込みステートメントそれぞれの前には、キーワード EXEC と SQL がなければなりません。

実行可能ステートメント

アプリケーション・プログラムに組み込まれた実行可能ステートメントは、同じ場所にホスト言語のステートメントが指定されていればそのホスト言語ステートメントが実行されることになるすべての時点で、実行されます。つまり、ループの中にあるステートメントは、そのループが実行されるたびに実行されます。また、条件付き構造の中にあるステートメントは、その条件が満たされた場合にのみ実行されます。

組み込みステートメントでは、ホスト変数を参照することができます。組み込みステートメントでは、以下の 2 つの目的でホスト変数を参照します。

- 入力として使用する (ステートメントの実行時にホスト変数の現行値を使用する)。
- 出力として使用する (ステートメントの実行結果として、ホスト変数に新しい値を割り当てる)。

特に、式や述部内のホスト変数の参照は、すべて、それらの変数の現行値によって実際に置換されます。つまり、それらの変数は入力として使用されます。その他の参照の扱い方については、ステートメントごとに個別に説明します。

実行可能ステートメントの後では、必ず SQL 戻りコードのテストを行わなければなりません。代わりに、WHENEVER ステートメント (このステートメント自体は実行不能ステートメント) を使用して、組み込みステートメントの実行直後の制御の流れを変えることができます。

SQL ステートメントで参照しているオブジェクトは、ステートメントを準備する時点では存在していなくても構いません。

実行不能ステートメント

組み込み実行不能ステートメントは、プリコンパイラーによってのみ処理されます。このようなステートメント内で検出されたエラーは、すべて、プリコンパイラーによって報告されます。実行不能ステートメントは、*決して実行されることはありません*。アプリケーション・プログラムの実行可能ステートメントの中に、実行不能ステートメントが入っている場合、その実行不能ステートメントはノー・オペレーションとして扱われます。したがって、そのようなステートメントに続けて、SQL 戻りコードのテストを行ってはなりません。

動的な準備と実行

アプリケーション・プログラムは、ホスト変数に入れられた文字ストリングの形式で動的に SQL ステートメントを構築することができます。通常、ステートメントは、プログラムで使用できるデータ (例えば、ワークステーションからの入力) から構築されます。このようなステートメントは、(組み込まれた) ステートメント PREPARE を使用して実行の準備を行うことができ、(組み込まれた) ステートメント EXECUTE によって実行することができます。代わりに、(組み込まれた) ステートメント EXECUTE IMMEDIATE を使用して、1 つのステップでステートメントの準備と実行を行うことができます。

動的に準備するステートメントには、ホスト変数に対する参照が含まれていてはなりません。代わりに、パラメーター・マーカーを入れることができます。パラメーター・マーカーに関する規則については、725 ページの『PREPARE』を参照してください。準備されたステートメントが実行されると、パラメーター・マーカーは、EXECUTE ステートメントで指定されたホスト変数の現行値によって事実上置き換えられます。この置き換えに関する規則については、667 ページの『EXECUTE』を参照してください。準備済みのステートメントは、ホスト変数の異なる値を使用して、何回でも実行することができます。EXECUTE IMMEDIATE では、パラメーター・マーカーを使用することはできません。

ステートメントが正常に実行されたか否かは、その EXECUTE (または EXECUTE IMMEDIATE) ステートメントの実行後に、SQLCA に設定される SQL 戻りコードによって示されます。この SQL 戻りコードは、組み込みステートメントに関する上記の説明に従って検査してください。詳細については、373 ページの 『SQL の戻りコード』の節を参照してください。

選択ステートメントの静的呼び出し

選択ステートメントは、(実行不能) ステートメント DECLARE CURSOR の一環として組み込むことができます。このようなステートメントは、該当のカーソルが、(組み込まれた) ステートメント OPEN によってオープンされるたびに実行されます。カーソルのオープン後、結果表は、FETCH ステートメントを繰り返して実行することにより 1 回に 1 行ずつ、または複数行 FETCH ステートメントを使用して、1 回に複数行を検索することができます。

このような方法で使用する場合、*選択ステートメント* にはホスト変数への参照を含めることができます。このような参照は、OPEN の実行時点における該当の変数の値によって事実上置き換えられます。

選択ステートメントの動的呼び出し

アプリケーション・プログラムは、ホスト変数に入れられた文字ストリングの形式で*選択ステートメント* を動的に構築することができます。一般的に、このようなステートメントはそのプログラムで使用可能なデータ (例えば、ワークステーションから得られる照会) から構築されます。構築されたステートメントは、(組み込まれた) ステートメント OPEN によってそのカーソルがオープンされるたびに実行されます。カーソルのオープン後、結果表は、FETCH ステートメントを繰り返して実行することにより 1 回に 1 行ずつ、または複数行 FETCH ステートメントを使用して、1 回に複数行を検索することができます。

このような方法で使用する場合、*選択ステートメント* には、ホスト変数に対する参照を含めてはなりません。ホスト変数に対する参照の代わりに、パラメーター・マーカーを入れることができます。パラメーター・マーカーに関する規則については、725 ページの『PREPARE』を参照してください。パラメーター・マーカーは、実際には OPEN ステートメントで指定されているホスト変数の値に置き換えられます。この置き換えに関する規則については、720 ページの『OPEN』を参照してください。

対話式呼び出し

データベース・マネージャーのアーキテクチャーの一環として、ワークステーションから SQL ステートメントを入力する機能があります。DB2 UDB for iSeries ライセンス・プログラムには、この機能に対応する構造化照会言語開始 (STRSQL) コマンド、照会管理機能開始 (STRQM) コマンド、および iSeries ナビゲーター の SQL スクリプト・サポートが備わっています。また、他のプロダクトを使用して SQL ステートメントを入力することもできます。このように入力するステートメントを、対話式に出すステートメントと呼びます。動的に準備することができないステートメントは、接続管理のステートメント (CONNECT、 DISCONNECT、 RELEASE、および SET CONNECTION) を除いて、対話式に出すことはできません。

パラメーター・マーカーやホスト変数に対する参照は、アプリケーション・プログラムの文脈中でのみ意味を持つので、対話式に出すステートメントには、パラメーター・マーカーやホスト変数への参照を含めてはなりません。

SQL の戻りコード

実行可能 SQL ステートメントを含むアプリケーション・プログラムは、少なくとも次のいずれか 1 つを用意しなければなりません。

- 名前が SQLCA の構造体
- 名前が SQLCODE の独立型の整数変数
- 名前が SQLSTATE の独立型の CHAR(5) (C の場合は CHAR(6)) の変数

独立型の SQLCODE と SQLSTATE の両者を用意しても構いません。SQLCA を用意する場合には、独立型の SQLCODE または SQLSTATE はいずれも用意することはできません。独立型の SQLCODE または SQLSTATE は、ホスト構造体中で宣言されていてはなりません。

REXX および RPG の場合には、SQLCA が自動的に用意されます。これら以外の言語では、INCLUDE SQLCA ステートメントの使用により、SQLCA を入手することができます。独立型の SQLCODE または SQLSTATE を用意する場合は、INCLUDE SQLCA を使用してはなりません。SQLCA には、名前が SQLCODE (RPG の場合は SQLCOD) の整数変数、および名前が SQLSTATE (RPG の場合は SQLSTT) の文字ストリングが含まれています。

SQLCA の代わりに独立型の SQLSTATE を用意するオプションは、ISO/ANSI SQL 規格に対する適合を意図しています。独立型の SQLSTATE の代わりに、独立型の SQLCODE を用意するオプションは、ISO/ANSI SQL 規格でその使用が望まれていない機能です。ISO/ANSI SQL 規格との適合が必要になる場合には、独立型の SQLSTATE を使用しなければなりません。

SQLCODE

アプリケーション・プログラムが SQLCA 変数と独立型変数のいずれを提供しているかには関係なく、SQLCODE は、SQL ステートメントが実行されるたびにデータベース・マネージャーによって設定されます。 DB2 UDB for iSeriesは、次のように、ISO/ANSI SQL 規格に準拠しています。

- SQLCODE = 0 で、しかも SQLWARN0 がブランクの場合、実行は正しく行われました。
- SQLCODE = 100 の場合、データが見つかりませんでした。例えば、カーソルが結果表の最後の行より後にあることが原因で、FETCH ステートメントがデータを戻さない場合など。
- SQLCODE > 0 で、しかも 100 ではない場合、警告が出されましたが、実行は正常に行われました。
- SQLCODE = 0 で、しかも SQLWARN0 = 'W' の場合、警告が出されましたが、実行は正常に行われました。
- SQLCODE < 0 の場合、実行は失敗しました。

DB2 UDB for iSeries SQLCODE とそれに対応している SQLSTATE の全リストについては、iSeries Information Center の SQL メッセージおよびコードを参照してください。

SQLSTATE

アプリケーション・プログラムが SQLCA または独立型の変数のどちらを用意しているかに関係なく、SQL ステートメントが実行されるたびに、データベース・マネージャーは、必ず SQLSTATE もセットします。したがって、アプリケーション・プログラムでは、SQLCODE の代わりに SQLSTATE をテストすることによって、SQL ステートメントの実行の成否を検査することができます。

SQLSTATE はアプリケーション・プログラムに、共通エラー条件を示す共通コードを戻します。さらに、SQLSTATE は、アプリケーション・プログラムで特定のエラーまたはエラーのクラスをテストできるように設計されています。この方式は、すべてのデータベース・マネージャーで同一であり、審議中の ISO/ANSI の規格案に基づいています。SQLSTATE クラス、ならびに、それぞれの SQLCODE に関連付けられている SQLSTATE の全リストについては、iSeries Information Center の SQL メッセージおよびコードを参照してください。

SQL のコメント

静的 SQL ステートメントの中では、ホスト言語または SQL のコメントを使用することができます。動的 SQL ステートメントの中に SQL コメントを組み込むことができます。SQL コメントには、次の 2 つのタイプがあります。

単純コメント
単純コメントの前には、2 つのハイフンを連続で付けます。

ブラケット付きコメント
ブラケット付きコメントは、/* と */ で囲みます。

単純コメントは、次の規則に従って使用してください。

- 2 つのハイフンは同一行に置く必要があります。また、ハイフンとハイフンの間にスペースを入れることはできません。
- 単純コメントは、スペースを入れることができるところならば、どこからでも開始できます (ただし、区切りトークン内や 'EXEC' と 'SQL' との間は除きます)。
- 単純コメントは、次の行に続けることはできません。
- COBOL では、2 つのハイフンの前にスペースを 1 つ置く必要があります。

ブラケット付きコメントは、次の規則に従って使用してください。

- /* は、同一行に置く必要があります。また、間にスペースを入れることはできません。
- */ は、同一行に置く必要があります。また、間にスペースを入れることはできません。
- ブラケット付きコメントは、スペースを入れることができるところならば、どこからでも開始できます (ただし、区切りトークン内や 'EXEC' と 'SQL' との間は除きます)。
- ブラケット付きコメントは、次の行に続けることができます。
- ブラケット付きコメントを、別のブラケット付きコメントの中にネストすることができます。

例

この例では、ステートメントの中に単純コメントを組み込む方法を示します。

```
CREATE VIEW PRJ_MAXPER         -- PROJECTS WITH MOST SUPPORT PERSONNEL
   AS SELECT PROJNO, PROJNAME  -- NUMBER AND NAME OF PROJECT
      FROM PROJECT
      WHERE DEPTNO = 'E21'     -- SYSTEMS SUPPORT DEPT CODE
      AND PRSTAFF > 1
```

この例では、ステートメントの中にブラケット付きコメントを組み込む方法を示します。

```
CREATE VIEW PRJ_MAXPER         /* PROJECTS WITH MOST SUPPORT
                                      PERSONNEL               */
   AS SELECT PROJNO, PROJNAME  /* NUMBER AND NAME OF PROJECT   */
      FROM PROJECT
      WHERE DEPTNO = 'E21'     /* SYSTEMS SUPPORT DEPT CODE    */
      AND PRSTAFF > 1
```

ALTER TABLE

ALTER TABLE ステートメントは表の定義を変更します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。このステートメントは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメントで識別される表に対して、
  - その表の ALTER 特権、および
  - その表が入っているライブラリーに対する *EXECUTE システム権限
- 管理権限

次のいずれかに該当する場合、ステートメントの権限 ID は、その表に対する ALTER 特権を持っています。

- その表の所有者である。
- その表に対する ALTER 特権が認可されている。
- その表に対する *OBJALTER または *OBJMGT のいずれかのシステム権限が認可されている。

外部キーを定義する場合、ステートメントの権限 ID が保持する特権には、親表に関して少なくとも次の 1 つが含まれていなければなりません。

- 該当の表に対する REFERENCES 特権またはオブジェクト管理権限。
- 指定された親キーの各列に対する REFERENCES 特権。
- その表の所有権。
- 管理権限。

次のいずれかに該当する場合は、ステートメントの権限 ID には、表に対する REFERENCES 特権があります。

- その表の所有者である。
- その表に対する REFERENCES 特権が認可されている。
- その表に対する *OBJREF か *OBJMGT のどちらかのシステム権限が認可されている。

特殊タイプを参照する場合は、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- ステートメント内に識別されているそれぞれの 特殊タイプ に対しては次のもの。
  - その 特殊タイプに対する USAGE 特権。および
  - その 特殊タイプ が入っているライブラリーに対する *EXECUTE システム権限

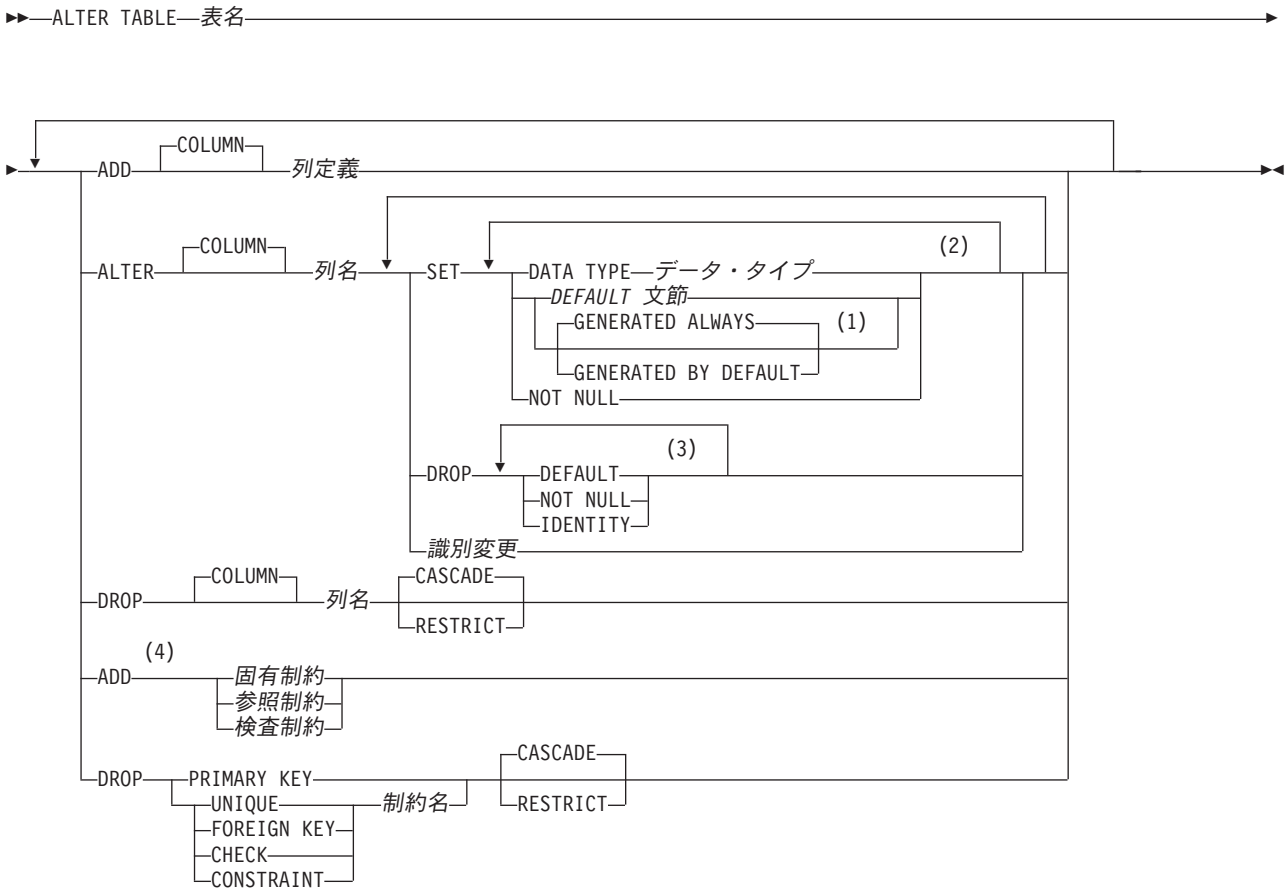
- 管理権限

次のいずれかに該当する場合、ステートメントの権限 ID には、特殊タイプ に対する USAGE 特権が与えられます。

- その 特殊タイプ の所有者である。
- その 特殊タイプ に対する USAGE 特権が付与されている。
- その 特殊タイプ に対するシステム権限の *OBJOPR と *EXECUTE が付与されている。

ALTER TABLE

構文



注:

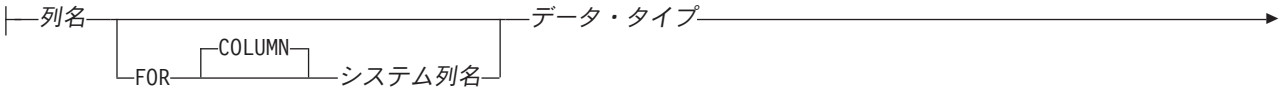
1 GENERATED を指定できるのは、列のデータ・ タイプが ROWID (または ROWID データ・タイプに基づく特殊タイプ) であるか、または列が識別列である場合のみです。

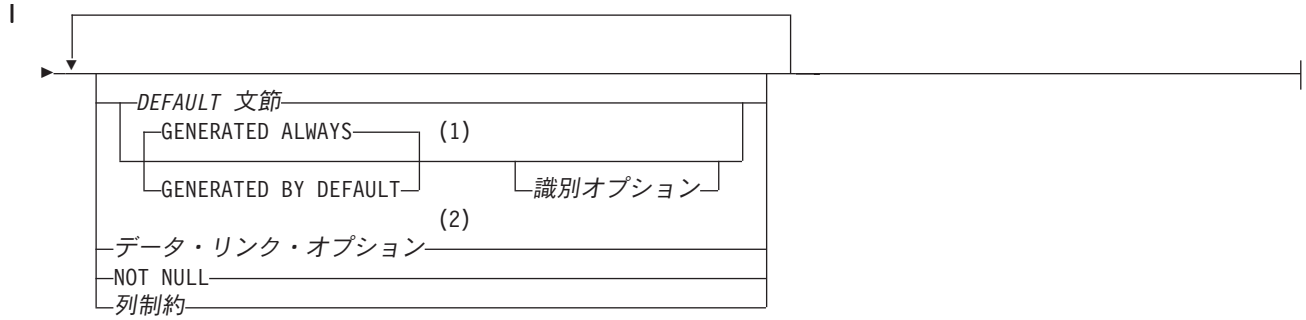
2 各文節はそれぞれ 1 回のみ指定できます。 DATA TYPE を指定する場合は、最初に指定する必要があります。

3 各文節はそれぞれ 1 回のみ指定できます。

4 これが ALTER TABLE ステートメントの最初の文節である場合は、ADD キーワードは任意指定ですが、指定することを強くお勧めします。それ以外の場合は、ADD キーワードは必須です。

列定義:





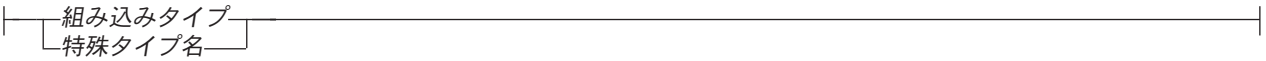
注:

1 GENERATED を指定できるのは、列のデータ・タイプ が ROWID (または ROWID データ・タイプに基づく特殊タイプ) であるか、また は列が識別列である場合のみです。

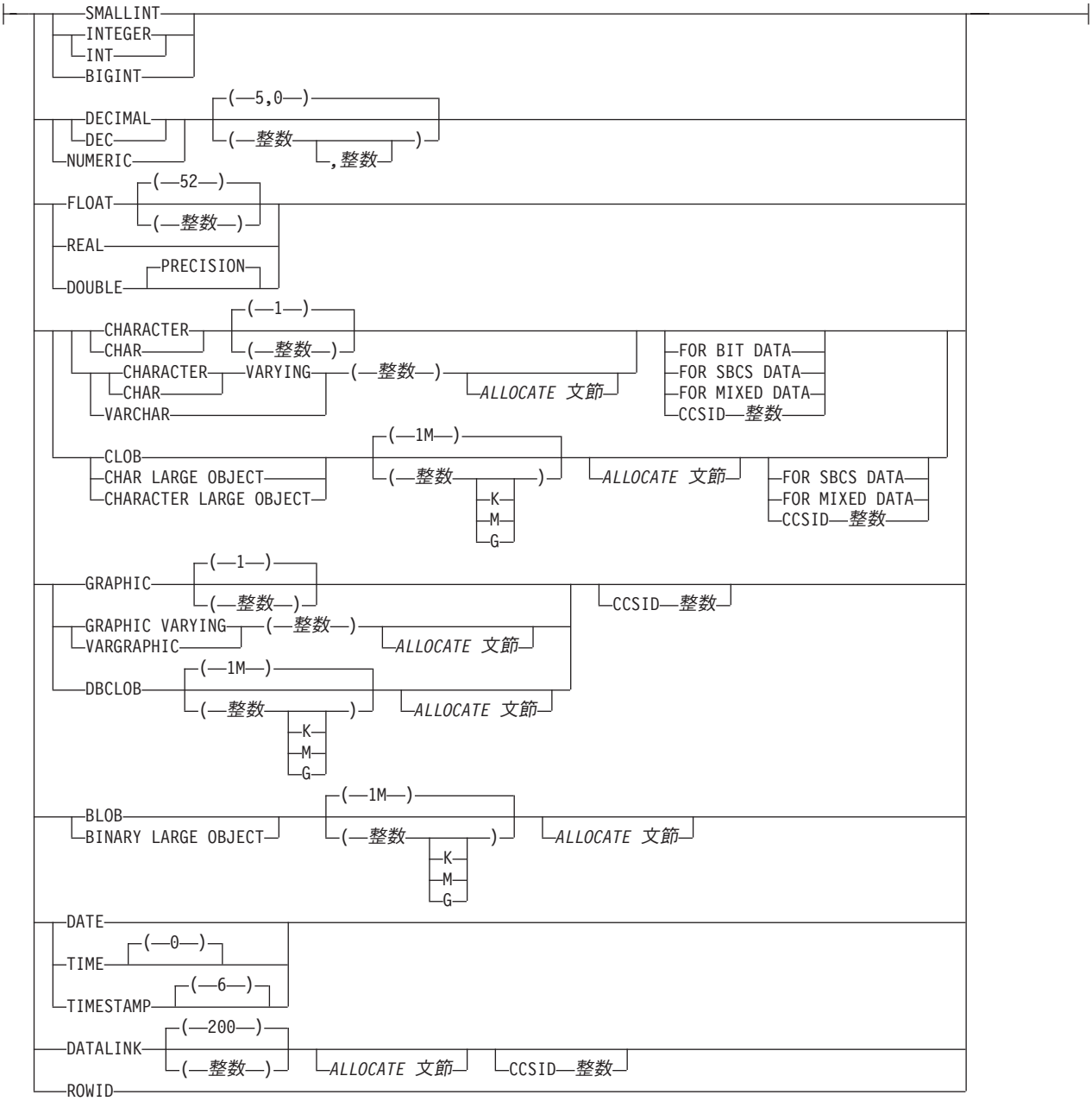
2 データ・リンク・オプションは、DATALINK、および DATALINK を ソースとする特殊タイプに対してのみ指定することができます。

ALTER TABLE

データ・タイプ:

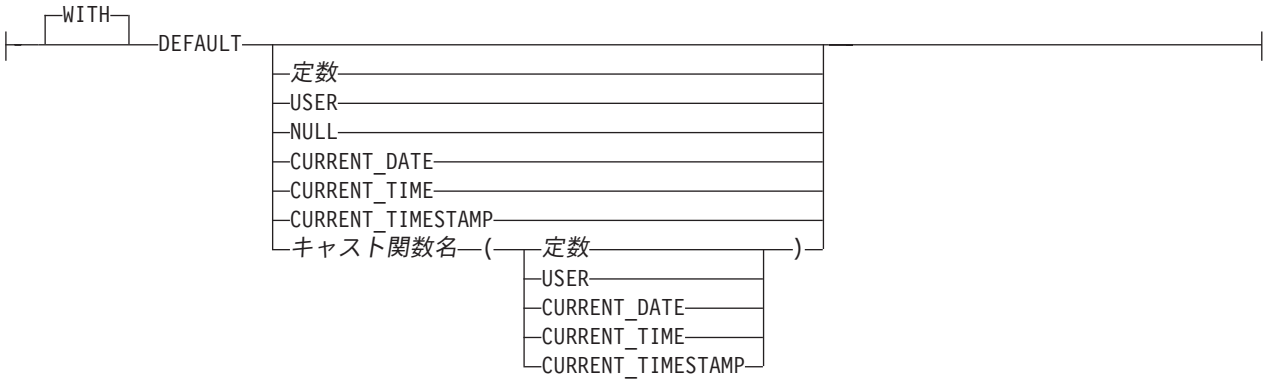


組み込みタイプ:

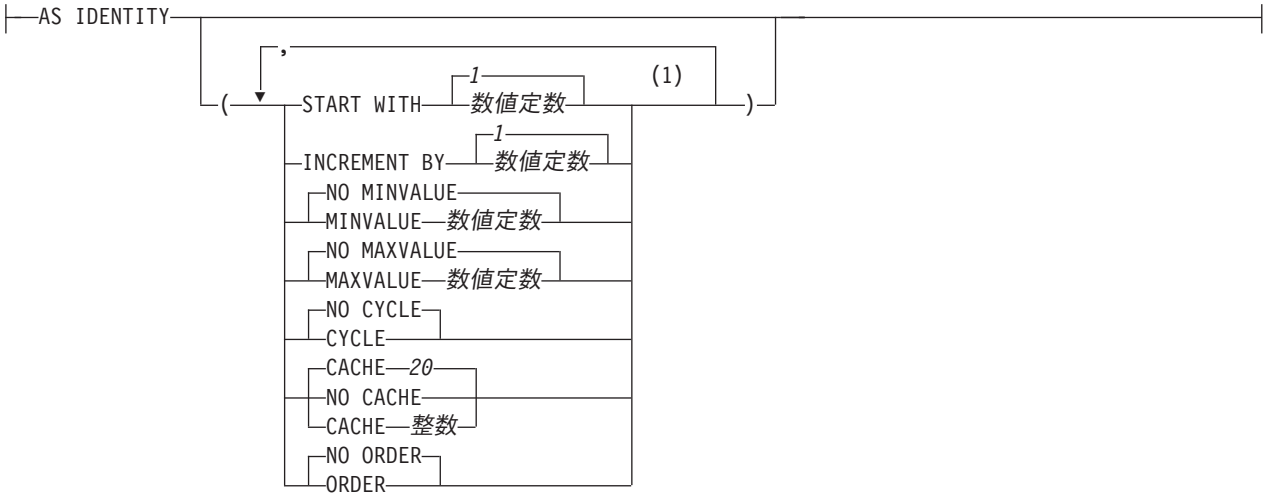


ALLOCATE 文節:

DEFAULT 文節:



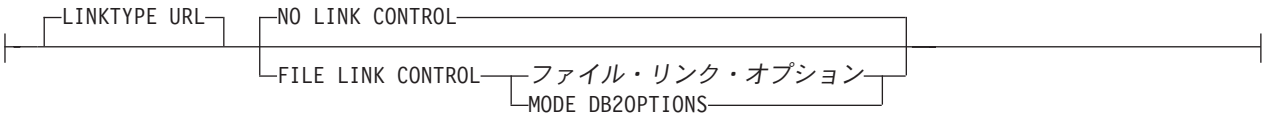
識別オプション:



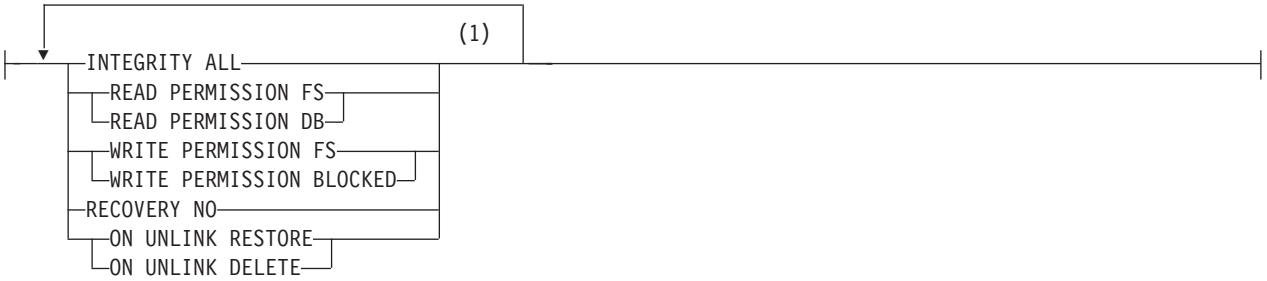
注:

1 各文節はそれぞれ 1 回のみ指定できます。

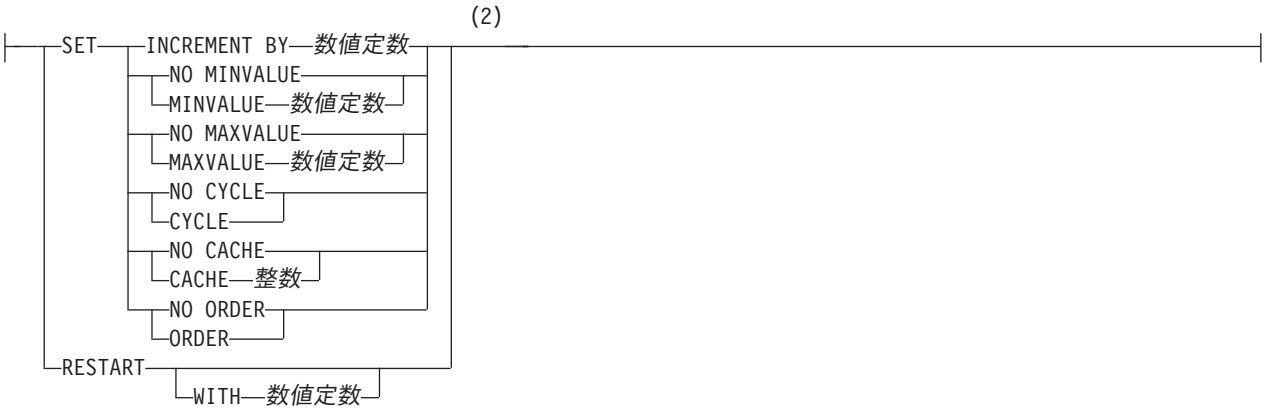
データ・リンク・オプション:



ファイル・リンク・オプション:



識別変更:

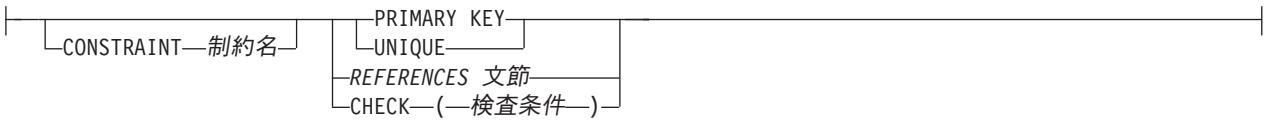


注:

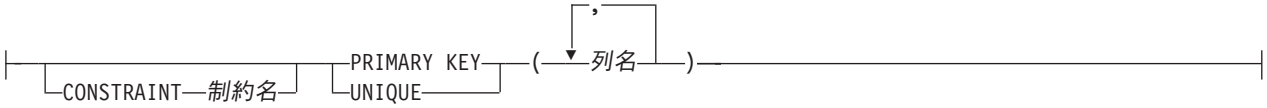
1 5 つのファイル・リンク・オプション をすべて指定する必要がありますが、指定する順序は任意です。

2 各文節はそれぞれ 1 回のみ指定できます。

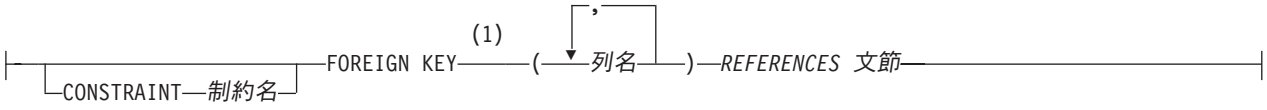
列制約:



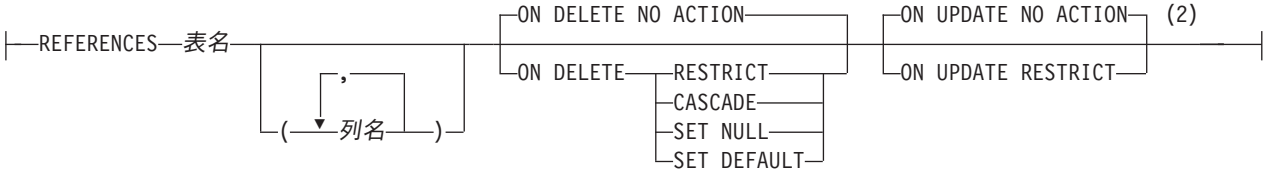
固有制約:



参照制約:



REFERENCES 文節:



検査制約:

注:

1 他の製品との互換性を保つために、列名 (CONSTRAINT キーワードがない) を FOREIGN KEY の後に指定することもあります。

2 ON DELETE と ON UPDATE 文節は、どの順序で指定しても構いません。

説明

表名
変更したい表を識別します。*表名* は、 現行サーバーに存在している表を示すものでなければなりません。この表は、ビュー、カタログ表、またはグローバル一時表であってはなりません。

ADD COLUMN

列定義
表に列を追加します。表に行がある場合は、その表の列の値はそれぞれデフォルト値に設定されます。ただし、列が ROWID 列または識別列 (AS IDENTITY として定義されている列) である場合を除きます。 ROWID 列および識別列の

デフォルト値は、データベース・マネージャーが生成します。表にすでに *n* 個の列がある場合、新規の列の順番は *n*+1 になります。 *n* の値は 8000 以下でなければなりません。

1 つの表は ROWID 列を 1 つだけ持つことができます。すでに識別列がある表に、さらに識別列を追加することはできません。

新しい列を追加した結果、すべての列の行バッファー・バイト・カウントの合計が、32766 (VARCHAR 列または VARGRAPHIC 列を指定する場合は 32740) を超えてしまう場合は、列の追加はできません。さらに、LOB を指定してある場合は、挿入または更新の時点で、すべての列のバイト・カウントの合計が 3 758 096 383 を超えていてはなりません。データ・タイプごとの列のバイト・カウントについては、 569 ページの『使用上の注意』を参照してください。

列名
表に追加したい列の名前を指定します。表の複数の列や、表のシステム列名に同じ名前を使用してはなりません。*列名* は修飾しません。

FOR COLUMN *システム列名*
列の OS/400 名を指定します。表の複数の列名またはシステム列名に、同じ名前を使用してはなりません。

システム列名が指定されず、また列名が有効なシステム列名でない場合には、システム列名が生成されます。システム列名の生成方法に関する詳細については、572 ページの 『列名の生成の規則』を参照してください。

データ・タイプ
列のデータ・タイプを指定します。

組み込みタイプ
組み込みデータ・タイプを指定します。*組み込みタイプ* の説明については、 542 ページの『CREATE TABLE』を参照してください。

FILE LINK CONTROL を指定した DATALINK 列を CASCADE の削除規則を伴う参照制約を受ける表に追加することはできません。

DEFAULT
列のデフォルト値を指定します。この文節は、*列定義* で複数回指定することはできません。 ROWID 列または識別列 (AS IDENTITY と定義されている列) については、DEFAULT を指定することはできません。 ROWID 列および識別列のデフォルト値は、データベース・マネージャーが生成します。 DEFAULT キーワードの後に値が指定されていない場合は、次のようになります。

- 列がヌル可能の場合、デフォルト値はヌル値になります。
- 列がヌル可能でない場合、デフォルト値は列のデータ・タイプによって決まります。

データ・タイプ	デフォルト値
数値	0
固定長ストリング	ブランク
可変長ストリング	0 のストリング長

日付	既存の行の場合は、1 月 1 日 0001 に対応する日付。追加した行の場合は、現在の日付。
時刻	既存の行の場合は、0 時、0 分、0 秒に対応する時刻。追加した行の場合は、現在の時刻。
タイム・スタンプ	既存の行の場合は、1 月 1 日 0001 に対応する日付、および 0 時、0 分、0 秒、0 マイクロ秒に対応する時刻。追加した行の場合は、現在のタイム・スタンプ。
データ・リンク	DLVALUE(’’,’URL’,’’) に対応する値。
特殊タイプ	特殊タイプの対応するソース・タイプのデフォルト値

NOT NULL および DEFAULT を*列の定義* から省いた場合、DEFAULT NULL の暗黙の指定が取られます。

定数
その列のデフォルト値としての定数を指定します。これは、85 ページの 『割り当ておよび比較』で説明している割り当て規則に従って、その列に割り当てることができる値を表す定数にする必要があります。浮動小数点定数は、SMALLINT、INTEGER、BIGINT、DECIMAL、または NUMERIC 列に使用してはなりません。10 進定数には、小数点より右方に、その列に指定された位取りより多くの桁を含めてはなりません。

USER
INSERT 時や UPDATE 時に USER 特殊レジスターの値をその列のデフォルト値として指定します。列のデータ・タイプは、USER 特殊レジスターの長さ属性と同じかそれより大きい長さ属性を持つ CHAR または VARCHAR でなければなりません。既存の行の場合、値は ALTER TABLE ステートメントの処理時の USER 特殊レジスターの値になります。

NULL
その列のデフォルト値としてヌルを指定します。NOT NULL を指定する場合は、同じ*列の定義* 内で DEFAULT NULL を指定してはなりません。

CURRENT_DATE
現在の日付を列のデフォルト値として指定します。CURRENT_DATE を指定する場合は、列のデータ・タイプは DATE または DATE に基づく特殊タイプでなければなりません。

CURRENT_TIME
現在の時刻を列のデフォルト値として指定します。CURRENT_TIME を指定する場合は、列のデータ・タイプは TIME または TIME に基づく特殊タイプでなければなりません。

CURRENT_TIMESTAMP
現在のタイム・スタンプを列のデフォルト値として指定します。

CURRENT_TIMESTAMP を指定する場合は、列のデータ・タイプは TIMESTAMP または TIMESTAMP に基づく特殊タイプでなければなりません。

キャスト関数名
この形式のデフォルト値は、特殊タイプやデータ・タイプ BLOB、CLOB、 DBCLOB、 DATE、 TIME、または TIMESTAMP として定義された列でのみ使用することができます。次の表は、これらのキャスト関数 の許可されている使用法を示します。

データ・タイプ	キャスト関数名
BLOB、CLOB、または DBCLOB に基づく特殊タイプ N	BLOB、CLOB、または DBCLOB *
DATE、TIME、または TIMESTAMP に基づく特殊タイプ N	N (N の作成時に生成されたユーザー定義のキャスト関数) ** あるいは DATE、TIME、または TIMESTAMP *
他のデータ・タイプに基づく特殊タイプ	N (N の作成時に生成されたユーザー定義のキャスト関数) **
BLOB、CLOB、または DBCLOB	BLOB、CLOB、または DBCLOB *
DATE、TIME、または TIMESTAMP	DATE、TIME、または TIMESTAMP *

注:

* 関数には、QSYS2 の暗黙的または明示的スキーマ名のデータ・タイプ (または、特殊タイプのソース・タイプ) の名前と一致する名前を指定する必要があります。

** 関数には、列の特殊タイプの名前と一致する名前を指定する必要があります。スキーマ名で修飾する場合は、特殊タイプのスキーマ名と同じ名前を指定する必要があります。修飾しない場合は、関数の解析から得られるスキーマ名は、特殊タイプのスキーマ名と同じ名前にする必要があります。

定数
定数を引き数として指定します。この定数は、特殊タイプのソース・タイプの定数、あるいは、特殊タイプでない場合は、データ・タイプの定数の規則に準拠する必要があります。BLOB、 CLOB、 DBCLOB、 DATE、 TIME、および TIMESTAMP 関数の場合は、この定数をストリング定数にする必要があります。

USER
INSERT 時や UPDATE 時に USER 特殊レジスターの値をその列のデフォルト値として指定します。列の特殊タイプのソース・タイプのデータ・タイプは、USER の長さ属性と同じかそれより大きい長さ属性を持つ CHAR または VARCHAR でなければなりません。既存の行の場合、値は ALTER TABLE ステートメントの処理時の USER 特殊レジスターの値になります。

CURRENT_DATE
現在の日付を列のデフォルト値として指定します。
CURRENT_DATE を指定する場合、列の特殊タイプのソース・タイプのデータ・タイプは、DATE にする必要があります。

CURRENT_TIME
現在の時刻を列のデフォルト値として指定します。

CURRENT_TIME を指定する場合、列の特殊タイプのソース・タイプのデータ・タイプは、TIME にする必要があります。

CURRENT_TIMESTAMP
現在のタイム・スタンプを列のデフォルト値として指定します。CURRENT_TIMESTAMP を指定する場合、列の特殊タイプのソース・タイプのデータ・タイプは、TIMESTAMP にする必要があります。

GENERATED
列の値をデータベース・マネージャーが生成することを指定します。列が識別列 (AS IDENTITY 文節で定義されたもの) と見なされる場合は、GENERATED を指定する必要があります。また、列のデータ・タイプが ROWID (または ROWID に基づく特殊タイプ) である場合も、GENERATED を指定できます。その他の場合は、GENERATED を指定してはなりません。

ALWAYS
表に行が 1 つ挿入されるたびに、常にデータベース・マネージャーが列の値を生成することを指定します。 ALWAYS が推奨値です。

BY DEFAULT
行が挿入されたときに、列の値が指定されていない場合のみ、データベース・マネージャーが列の値を生成することを指定します。値が指定されている場合は、データベース・マネージャーはその値を使用します。

ROWID 列の場合は、データベース・マネージャーは指定された値を使用しますが、その値は、すでに DB2 UDB (OS/390 および z/OS 版) または DB2 UDB for iSeries により生成されている有効な固有の行 ID でなければなりません。

識別列の場合は、データベース・マネージャーは指定された値を挿入しますが、その識別列が固有制約を持っているか、またはその識別列を単独で指定する固有索引を持っている場合を除き、その値がその列の固有の値であるかどうかの検査は行いません。

AS IDENTITY
列が表の識別列であることを指定します。 1 つの表は識別列を 1 つだけ持つことができます。 AS IDENTITY を指定できるのは、列のデータ・タイプが、厳密に位取りがゼロの数値タイプ (SMALLINT、 INTEGER、 BIGINT、 DECIMAL、または位取りがゼロの NUMERIC、またはこれらのタイプに基づく特殊タイプ) である場合だけです。

識別列は、暗黙的に NOT NULL になります。

START WITH *数値定数*
識別列について生成される最初の値を指定します。小数点の右側にゼロ以外の数字がないことを条件として、この列に割り当てることのできる任意の正または負の値を指定できます。

識別列を定義するときに値を明示的に指定していない場合、デフォルト値は、昇順の場合は MINVALUE で降順の場合は MAXVALUE です。この値は、シーケンスが最大値または最小値に達した後で、シーケンスの循環により到達する値になるとは限りません。 START WITH 文節を

使用することにより、この循環に使用される値の範囲外の値からシーケンスを開始することができます。循環に使用する範囲は、MINVALUE および MAXVALUE で定義します。

INCREMENT BY *数値定数*
識別列の連続した値の間隔を指定します。この値には、0 でなく、長整数定数の値を超過せず、かつ小数点の右側にゼロ以外の数字がないことを条件として、この列に割り当てることのできる任意の正または負の値を指定できます。デフォルト値は 1 です。

この値が正である場合は、識別列の値のシーケンスは昇順になります。この値が負の場合は、値のシーケンスは降順になります。

MAXVALUE *数値定数*
この識別列用として生成される最大値を示す数値定数を指定します。この値には、この列に割り当てることのできる任意の正または負の値を指定できますが、最小値より大きい値でなければなりません。

識別列を定義するときにこの値を明示的に指定していない場合は、この値は、昇順シーケンスの場合は該当データ・タイプ (および、DECIMAL の場合は精度) の最大値になります。降順シーケンスの場合は、この値は START WITH の値ですが、START WITH を指定していなければ -1 です。

MINVALUE *数値定数*
この識別列用として生成される最小値を示す数値定数を指定します。この値には、この列に割り当てることのできる任意の正または負の値を指定できますが、最大値より小さい値でなければなりません。

識別列を定義するときにこの値を明示的に指定していない場合は、この値は、昇順シーケンスの場合は START WITH の値ですが、START WITH を指定していなければ 1 です。降順シーケンスの場合は、この値は、該当データ・タイプ (および、DECIMAL の場合は精度) の最小値です。

CACHE または NO CACHE
事前割り振りの値をメモリー内に保持するかどうかを指定します。値を事前に割り振ってキャッシュに保管しておくと、表に行を挿入するときのパフォーマンスが向上します。

CACHE *整数*
データベース・マネージャーが事前割り振りしてメモリー内に保持する、識別列シーケンスの値の数を指定します。指定できる最小の値は 2 で、最大の値は、1 つの整数で表せる最大の値です。デフォルト値は 20 です。

システム障害が発生すると、キャッシュに保管されていてまだ割り当てられていない識別列の値はすべて失われ、使用不能になります。したがって、CACHE に指定する値は、システム障害が発生したときに失われる可能性のある識別列の値の最大数でもあります。

NO CACHE
識別列の値を事前割り振りしないことを指定します。

CYCLE または NO CYCLE
シーケンスの最大値または最小値に達した後も、この識別列について値を生成し続けるかどうかを指定します。

CYCLE
最大値または最小値に達した後も、この列の値を生成し続けることを指定します。このオプションを使用した場合は、昇順シーケンスがシーケンスの最大値に達した後では、最小値が生成されます。降順シーケンスがシーケンスの最小値に達した後は、最大値が生成されます。列の最大値と最小値によって、循環に使用される範囲が決まります。

CYCLE が効力を持っている場合は、データベース・マネージャーは 1 つの識別列について重複する値を生成することがあります。対象の識別列について固有制約または固有索引が存在する場合は、固有でない値が生成されるとエラーが起こります。

NO CYCLE
シーケンスの最大値または最小値に達した後は、この識別列について値を生成しないことを指定します。これは、デフォルト値です。

ORDER または NO ORDER
識別値を要求された順序で生成するかどうかを指定します。

ORDER
識別値を要求された順序で生成することを指定します。

NO ORDER
値を要求された順序で生成する必要がないことを指定します。これは、デフォルト値です。

データ・リンク・オプション
DATALINK 列に関連したオプションを指定します。データ・リンク・オプション については、542 ページの『CREATE TABLE』を参照してください。

NOT NULL
列にヌル値が入るのを防止します。NOT NULL を指定しないと、列にヌル値が入ってもよいことを暗黙指定することになります。NOT NULL を指定する場合は、DEFAULT も指定する必要があります。

列制約

CONSTRAINT *制約名*
制約の名前を指定します。制約名は、現行サーバーにすでに存在している制約を識別するものであってはなりません。

この文節の指定がない場合、固有制約の名前がデータベース・マネージャーによって生成されます。

PRIMARY KEY
これは、1 つの列からなる基本キーを定義する簡便な手段です。列 C の定義に PRIMARY KEY を指定した場合、その効果は、別個の文節として PRIMARY KEY(C) 文節を指定したのと同一です。

この文節は、複数の*列定義* で指定してはなりません。また列定義に UNIQUE 文節の指定がある場合には、この文節を指定してはなりません。列は、LOB 列や DATALINK 列にすることはできません。

基本キーを追加すると、CHECK 制約が暗黙的に追加され、その基本キーを構成する列で NULL を使用することはできないという規則が適用されます。

UNIQUE
これは、1 つの列からなる固有キーを定義する簡便な手段です。列 C の定義に UNIQUE の指定がある場合、その効果は、別個の文節として UNIQUE(C) 文節が指定された場合と同一です。

この文節は、1 つの列定義で複数回指定することはできません。また、*列定義* で PRIMARY KEY が指定されている場合には、この文節を指定してはなりません。列は、LOB 列や DATALINK 列にすることはできません。

REFERENCES 文節
列定義の*REFERENCES 文節* は、1 つの列からなる外部キーを定義する簡便な手段です。列 C の定義に REFERENCES 文節の指定がある場合、その効果は、C が識別された唯一の列である FOREIGN KEY 文節の一環としてその REFERENCES 文節が指定されている場合と同一です。

CHECK(*検査条件***)**
これを指定すると、単一の列だけを参照するという*検査条件* の検査制約を簡略式で定義することができます。したがって、列 C の列定義で CHECK を指定した場合、検査制約の*検査条件* では、C 以外の列を参照することができなくなります。結果は、検査制約を別個の文節として指定した場合と同じです。

CHECK 制約の中で、FILE LINK CONTROL 列を持つ ROWID または DATALINK を参照することはできません。その他の制限事項については、395 ページの『ADD 検査制約』を参照してください。

ALTER COLUMN

既存の列の定義を変更します。指定した属性だけが変更されます。それ以外のものは、未変更のままになります。

列名
変更したい列を識別します。列名は修飾してはなりません。この名前は、指定した表の列を識別するものでなければなりません。この名前で識別する列は、この ALTER TABLE ステートメントで追加または除去しようとしている列であってはなりません。

SET DATA TYPE *データ・タイプ*
変更したい列の新しいデータ・タイプを指定します。新しいデータ・タイプには、その列の既存のデータ・タイプとの互換性を保持させる必要があります。データ・タイプの互換性に関する詳細については、85 ページの 『割り当ておよび比較』を参照してください。ただし、一般規則には次の 2 つの例外があります。

- 文字と UCS-2 グラフィックとの間のデータ・タイプの変更は許されます。

- 日付/時刻のデータ・タイプから文字へのデータ・タイプの変更は許されません。

指定した長さ、精度、および位取りは、既存の長さ、精度、および位取りに比較して、大きい場合も、小さい場合も、同じである場合もあります。ただし、新しい長さ、精度、または位取りの方が小さい場合は、切り捨てまたは数値変換エラーが起こる場合があります。

指定した列にデフォルト値が入れられており、新しいデフォルト値を指定しない場合は、既存のデフォルト値で、85 ページの 『割り当ておよび比較』で説明している割り当て規則に従ってその列に割り当てることができる値を表す必要があります。

列を固有キー、基本キー、または外部キーで指定する場合は、それらのキーの列の長さの新しい合計は 2000-n を超えてはなりません。ここで、n はヌルになることができる、指定した列の数です。

属性を変更すると、列への割り当てに関する規則に従って、列内の既存の値が新しい列属性に変換されます。ただし、ストリング値は切り捨てられるという例外を伴います。

SET *DEFAULT 文節*
変更したい列の新しいデフォルト値を指定します。指定するデフォルト値は、85 ページの 『割り当ておよび比較』で説明している割り当て規則に従って、その列に割り当てることができる値を表す必要があります。

SET NOT NULL
列にヌル値を含めることはできないことを指定します。表の既存の行にあるこの列の値は、すべてヌル以外でなければなりません。指定した列にデフォルト値があり、新しいデフォルト値を指定しない場合は、既存のデフォルト値は NULL であってはなりません。SET NULL の DELETE 規則を伴う参照制約の外部キーで列が識別され、その外部キーに他のヌル可能列がない場合は、SET NOT NULL は使用できません。

SET GENERATED ALWAYS または GENERATED BY DEFAULT
列の値をデータベース・マネージャーが生成することを指定します。列が識別列 (AS IDENTITY 文節で定義されている列) と見なされる場合、または列のデータ・タイプが ROWID (または ROWID に基づく特殊タイプ) である場合は、GENERATED を指定する必要があります。その他の場合は、GENERATED を指定してはなりません。

DROP DEFAULT
列の現行デフォルト値を除去します。指定した列にはデフォルト値がなければならず、ヌル属性として NOT NULL があってはなりません。新しいデフォルト値は、ヌル値になります。

DROP NOT NULL
列の NOT NULL 属性を除去し、その列がヌル値をもてるようにします。デフォルト値の指定がない場合、またはデフォルト値がまだ存在していない場合は、新しいデフォルト値はヌル値になります。表の基本キーで列を指定する場合は、DROP NOT NULL は使用できません。

DROP IDENTITY
列の識別属性を除去して、列を単純な数値データ・タイプ列にします。列が識別列でない場合は、DROP IDENTITY は使用できません。

識別変更
列の識別属性を変更します。この列は識別列でなければなりません。属性の説明については、387を参照してください。

RESTART
識別列に入れる次の値を指定します。 WITH 数値定数を指定していない場合は、この識別列を最初に作成したときに暗黙的または明示的に指定されている開始値から、シーケンスが再開始されます。この列は識別列でなければなりません。

WITH *数値定数*
この列に入れる次の値として*数値定数* を使用することを指定します。*数値定数* は、厳密に数値定数でなければなりません。この値には、小数点の右側にゼロ以外の数字がないことを条件として、この列に割り当てることのできる任意の正または負の値を指定できます。

DROP COLUMN

識別された列を表から除去します。

列名
除去したい列を識別します。列名は修飾してはなりません。この名前は、指定した表の列を識別するものでなければなりません。この名前で識別する列は、この ALTER TABLE ステートメントですでに追加または除去した列であってはなりません。この名前で表の唯一の列を識別してはなりません。

CASCADE
除去される列に従属しているビュー、索引、トリガー、または制約もすべて除去されることを指定します。[40]

RESTRICT
列にビュー、索引、トリガー、または制約が従属している場合は、その列は除去できないことを指定します。[40]

ある制約で参照されている列がすべて同一の ALTER TABLE ステートメントで除去される場合は、その除去が RESTRICT で妨げられることはありません。

ADD 固有制約

CONSTRAINT *制約名*
制約の名前を指定します。*制約名* は、現行サーバーにすでに存在している制約を識別するものであってはなりません。*制約名*は、スキーマ内で固有でなければなりません。

指定しない場合は、固有の制約名がデータベース・マネージャーによって生成されます。

UNIQUE(*列名***、...)**
識別された列で構成される固有キーを定義します。それぞれの*列名* は、該当の

40. トリガーは、UPDATE OF 列リストまたはトリガー・アクション内の任意の場所で参照されている場合、列に従属します。

表の列を識別する非修飾の名前でなければなりません。同じ列を複数回指定することはできません。この列は、LOB 列または DATALINK 列であってはなりません。指定できる列の数は 120 を超えてはならず、その長さの合計は 2000-n を超えてはなりません。ここで、n はヌルが許される列の数です。指定する列は、その表の他の UNIQUE 制約または PRIMARY KEY に指定されている列と同じであってはなりません。例えば、UNIQUE(A,B) は、UNIQUE(B,A) あるいは PRIMARY KEY(A,B) が該当の表にすでに存在する場合には許されません。一連の列に指定されている既存のどの非空値も固有の値にする必要があります。複数のヌル値が許されます。

識別された列に固有索引がすでに存在する場合は、その索引が固有索引として指定されます。それ以外の場合は、固有キーの固有性をサポートするために固有索引が作成されます。固有索引は、別個のシステム論理ファイルとしてではなく、システム物理ファイルの一部として作成されます。

PRIMARY KEY(*列名***、...)**

指定した列で構成される基本キーを定義します。それぞれの *列名* は、該当の表の列を識別する非修飾の名前でなければなりません。同じ列を複数回指定することはできません。この列は、LOB 列または DATALINK 列であってはなりません。 指定する列の数は 120 を超えてはならず、それらの列の長さの合計は 2000 を超えてはなりません。該当の表にすでに基本キーが存在していてはなりません。指定する列は、その表の他の UNIQUE 制約に指定されている列と同じであってはなりません。例えば、PRIMARY KEY(A,B) は、その表に UNIQUE(B,A) がすでに存在する場合には許されません。指定した一連の列の既存の値は、固有でなければなりません。基本キーを追加すると、CHECK 制約が暗黙的に追加され、その基本キーを構成するどの列でも NULL を使用することはできないという規則が適用されます。

指定した列に固有索引がすでに存在している場合には、その索引は、基本索引として扱われます。このような場合以外は、基本キーの固有性をサポートする基本索引が作成されます。固有索引は、別個のシステム論理ファイルとしてではなく、システム物理ファイルの一部として作成されます。

ADD 参照制約

CONSTRAINT *制約名*

制約の名前を指定します。*制約名* は、現行サーバーにすでに存在している制約を識別するものであってはなりません。

指定しない場合は、固有の制約名がデータベース・マネージャーによって生成されます。

FOREIGN KEY

参照制約を定義します。

以下の説明で T1 は、変更したい表を表しています。

(*列名***、...)**

参照制約の外部キーは、指定した列で構成されます。各 *列名* は、T1 の列を識別する非修飾の名前でなければなりません。同じ列を複数回指定することはできません。この列は、LOB 列または DATALINK 列であってはなりま

せん。 指定する列の数は 120 を超えてはならず、それらの列の長さの合計は 2000-n を超えてはなりません。ここで、n は指定した列でヌルが許される列の数です。

REFERENCES *表名*

REFERENCES 文節で指定する表名 は、現行サーバー上に存在する基本表を示すものでなければなりませんが、カタログ表またはグローバル一時表を示すものであってはなりません。この表は、制約関係における親表と呼ばれます。

参照制約の外部キー、親キー、および親表が既存の参照制約の外部キー、親キー、および親表と同じである場合は、その参照制約は重複 します。 重複する参照制約は許されますが、お勧めできません。

以下の説明で、T2 は親表を示しています。

(*列名***、...)**

参照制約の親キーは、ここで指定する列によって構成されます。各*列名* は T2 の列を識別する非修飾の名前でなければなりません。同じ列を複数回指定することはできません。この列は、LOB 列または DATALINK 列であってはなりません。 指定できる列の数は 120 を超えてはならず、その長さの合計は 2000-n を超えてはなりません。ここで、n はヌルが許される列の数です。

この列名のリストは、T2 の基本キーまたは T2 に存在する UNIQUE 制約の列名のリストと同一でなければなりません。名前はどのような順序で指定しても構いません。例えば、(A,B) を指定すると、UNIQUE(B,A) として定義された固有制約はこの要件を満たします。列名のリストの指定がない場合、T2 は基本キーを持たなければなりません。列名のリストの省略は、基本キーの列の暗黙の指定を意味しています。

指定した外部キーは、T2 の親キーと同じ数の列を持たなければなりません。外部キーの *n* 番目の列と親キーの *n* 番目の列の記述は、同一のデータ・タイプと長さを持つ必要があります。

表が空である場合を除き、表の使用に先立って、外部キーの値の妥当性を検査する必要があります。外部キーの値は、ALTER TABLE ステートメントの実行中に妥当性を検査されます。したがって、外部キーのヌル以外の値はすべて、T2 の親キーの値と一致している必要があります。

FOREIGN KEY 文節によって指定する参照制約は、T2 が親で、T1 が従属の関係を定義します。

ON DELETE

親表の行が削除される時点で、従属表について行うアクションを指定します。可能なアクションには以下の 5 つがあります。

- NO ACTION (デフォルト)
- RESTRICT
- CASCADE
- SET NULL
- SET DEFAULT

外部キーの列にヌルが許される列がある場合を除いて、SET NULL を指定してはなりません。

T1 が削除トリガーを持つ場合には、CASCADE を指定してはなりません。T1 が更新トリガーを持つ場合には、SET NULL および SET DEFAULT を指定してはなりません。

FILE LINK CONTROL を指定した DATALINK 列が T1 に含まれる場合には、CASCADE を指定してはなりません。

削除規則は、T2 の行が DELETE または波及削除操作の対象で、しかもその行が T1 に従属する行を持っている場合に適用されます。以下の説明で、*p* はそのような T2 の行を表しています。

- RESTRICT または NO ACTION を指定した場合、エラーが生じ、行の削除は行われません。
- CASCADE を指定した場合、削除操作は、T1 の *p* の従属行に波及します。
- SET NULL を指定した場合、T1 の *p* の各従属行の外部キーのヌル可能な各列は、ヌルに設定されます。
- SET DEFAULT を指定した場合、T1 の *p* の各従属行の外部キーの各列は、そのデフォルト値に設定されます。

ON UPDATE

親表の行が更新される時点で、従属表で行うアクションを指定します。

更新規則は、T2 の行が UPDATE または波及更新操作の対象で、しかもその行が T1 に従属行を持つ場合に適用されます。以下の説明で、*p* はそのような T2 の行を表しています。

- RESTRICT または NO ACTION を指定した場合、エラーが生じ、行の更新は行われません。

ADD 検査制約

CONSTRAINT *制約名*

制約の名前を指定します。*制約名* は、現行サーバーにすでに存在している制約を識別するものであってはなりません。*制約名*は、スキーマ内で固有でなければなりません。

指定しない場合は、固有の制約名がデータベース・マネージャーによって生成されます。

CHECK(*検査条件***)**

検査制約を定義します。表のどの行についても、*検査条件* は真または不明のいずれかでなければなりません。

検査条件 は、*検索条件* です。ただし、下記の条件は除きます。

- 表の列だけを参照することができます。
- FILE LINK CONTROL 列を持つ ROWID または DATALINK を参照することはできません。
- 次のいずれも含めることはできません。
  - 副照会

- スカラー副選択
- 列関数
- ホスト変数
- パラメーター・マーカー
- LOB を含む複合式 (連結など)
- CURRENT TIMEZONE、CURRENT SCHEMA、CURRENT SERVER、CURRENT PATH、および USER 特殊レジスター
- NOW、CURDATE、および CURTIME スカラー関数
- NODENAME スカラー関数
- LOB を含む式
- 特殊タイプの作成に伴って暗黙に生成された関数以外のユーザー定義関数
- ATAN2、DIFFERENCE、RAND、RADIANS、および SOUNDEX スカラー関数
- スカラー関数 DLVALUE、DLURLPATH、DLURLPATHONLY、DLURLSERVER、または DLURLSCHEME
- スカラー関数 DLURLCOMPLETE (FILE LINK CONTROL と READ PERMISSION DB の属性が指定されたデータ・リンクの場合)

検索条件の詳細については、 167 ページの『検索条件』を参照してください。

DROP

PRIMARY KEY
基本キーの定義、およびその基本キーが親キーである参照制約すべてを除去します。該当の表は、基本キーをもっていなければなりません。

FOREIGN KEY *制約名*
制約名 で指定された参照制約を除去します。*制約名* は、該当の表が従属している参照制約を識別していなければなりません。

UNIQUE *制約名*
制約名 で指定した固有制約、およびその固有キーが親キーである参照制約すべてを除去します。*制約名* は、該当の表の固有制約を識別していなければなりません。DROP UNIQUE は、PRIMARY KEY 固有制約を除去することはありません。

CHECK *制約名*
検査制約*制約名* を除去します。*制約名* では、表の検査制約を識別する必要があります。

CONSTRAINT *制約名*
制約名 で指定した制約を除去します。*制約名*では、表の検査制約、固有制約、または参照制約を識別する必要があります。この制約が、PRIMARY KEY または UNIQUE の制約である場合、その基本キーまたは固有キーが親キーである参照制約もすべて除去されます。

CASCADE
固有制約に関して、除去される制約に従属している参照制約があれば、それもすべて除去されることを指定します。

RESTRICT
固有制約に関して、それに従属している参照制約がある場合は、その固有制約は除去できないことを指定します。

使用上の注意

ALTER TABLE ステートメントの ADD、ALTER、または DROP COLUMN 文節では、1 つの列を *1 回* だけ参照できます。ただし、ALTER TABLE ステートメントで制約を追加また除去する場合は、この同じ列を複数回参照することができます。

ALTER TABLE ステートメント内での操作の順序は、次のとおりです。

- 制約の除去
- RESTRICT オプションが指定された列の除去
- 他のすべての列定義の変更
  - CASCADE オプションが指定された列の除去
  - 列変更属性の追加
  - 列の追加
- 制約の追加

上記の各段階内で、ユーザーが文節を指定する順序は、文節が実行される順序になります。ただし、これには例外が 1 つあります。列のいずれかが除去される場合は、操作は列定義の追加または変更が行われる前に論理的に実行されます。

ビューまたは別のジョブの QTEMP 内の論理ファイルが、変更される表に従属している場合は、それらはいずれも ALTER TABLE ステートメントの結果として除去されます。

権限検査が行われるのは、除去される表に対してだけです。それ以外のオブジェクトには ALTER TABLE ステートメントでアクセスできますが、それらのオブジェクトに対する権限はまったく必要ありません。例えば、除去される表に存在しているビューに対しても、除去される表を参照制約によって参照する従属表に対しても、権限はまったく必要ありません。

表を変更するにあたっては、あらかじめその表と従属ビュー、および論理ファイルの現行バックアップをとっておくことをぜひお勧めします。

次のパフォーマンスに関する考慮事項は、表に対して列の追加、変更、または除去を行う際に ALTER TABLE ステートメントに適用されます。

- 表内のデータはコピーできます。[41]

  列を追加および除去する場合は、データをコピーする必要があります。

41. 記憶域が不足していて完全なコピーを作成できない場合は、必要な空き記憶域が約 16 ～ 32 メガバイトだけで済むという特殊コピーが実行されます。

列を変更する場合は、通常、データをコピーする必要があります。ただし、変更の内容が単に次のような場合は、データをコピーする必要はありません。

- VARCHAR 列の長さ属性が増やされる場合に、現行の長さ属性が 20 よりも大きい。
- VARGRAPHIC 列の長さ属性が増やされる場合に、現行の長さ属性が 10 よりも大きい。
- VARCHAR 列の割り振られた長さが変更される場合に、現行および新規の割り振られた長さが共に 20 より小さいか同じである。
- VARGRAPHIC 列の割り振られた長さが変更される場合に、現行および新規の割り振られた長さが共に 10 より小さいか同じである。
- 列の CCSID が変更される場合に、古い CCSID と新規の CCSID との間で変換が不要である。例えば、1 つの CCSID が 65535 である場合、データ変換は不要。
- デフォルト値が変更される場合に、デフォルト値の長さが現行の割り振られた長さより大きくない。
- DROP DEFAULT が指定されている。
- DROP NOT NULL が指定されているのに、表の変更が完了した後も、少なくとも 1 つのヌル可能列がその表の中に依然として存在している。

• 索引の再作成が必要になる可能性があります。[42]

  列が表に追加される場合や列が除去または変更されるときにそれらの列が索引キー内で参照されない場合は、索引を再作成する必要はありません。

  索引や制約のキー内で使用される列を変更する場合は、通常、その索引を再作成する必要があります。ただし、次のような場合は、索引の再作成は不要です。

  - VARCHAR キーや VARGRAPHIC キーの長さ属性が増やされる場合。
  - 列の CCSID が変更される場合に、古い CCSID と新規の CCSID との間で変換が不要である。例えば、1 つの CCSID が 65535 である場合。

カスケード効果

列を追加しても、SQL ビューや大部分の論理ファイルに対するカスケード効果はありません。[43] 例えば、表に列を追加しても、どの従属ビューにも列は追加されません。これは、それらのビューが SELECT * 文節を使用して作成されたビューである場合にも該当します。

列を除去または変更した場合は、数種類のカスケード効果が生じる可能性があります。表 41 に列を除去した場合のカスケード効果を示します。

表 41. 列の除去によるカスケード効果

操作	RESTRICT 効果	CASCADE 効果
ビューによって参照した列の除去	列の除去は許されません。	ビューおよびそのビューに従属しているビューすべてが除去されます。

42. 再作成が必要な索引は、すべて、データベース・サーバー・ジョブによって非同期で再作成されます。

43. 列を物理ファイルに追加する場合、列は、その物理ファイルの形式を共用する論理ファイルにも追加されます (ただし、その形式がその論理ファイルにおいて別のベースオン・ファイルで再使用されない場合に限ります)。

表 41. 列の除去によるカスケード効果 (続き)

操作	RESTRICT 効果	CASCADE 効果
非ビュー論理ファイルで参照する列の除去	この除去は可能で、次の場合に列が論理ファイルから除去されます。 • 論理ファイルが、変更しようとしているファイルと形式を共用する。また、 • 除去された列が、キー・フィールドとして使用されなかったり、選択/省略仕様で使用されない。また、 • 形式が、論理ファイルにおいて別のベースオン・ファイルで再使用されない。 それ以外の場合は、列の除去は許されません。	この除去は可能で、次の場合に列が論理ファイルから除去されます。 • 論理ファイルが、変更しようとしているファイルと形式を共用する。また、 • 除去された列が、キー・フィールドとして使用されなかったり、選択仕様や省略仕様で使用されない。また、 • 形式が、論理ファイルにおいて別のベースオン・ファイルで再使用されない。 それ以外の場合、論理ファイルは除去されます。
索引のキーで参照される列の除去	索引の除去は不可能です。	索引は除去されます。
固有制約の中で参照した列の除去	固有制約の中で参照した列がすべて同じ ALTER COLUMN ステートメントで除去され、しかもその固有制約が参照制約によって参照されていない場合は、それらの列および制約は除去されます。(したがって、この制約を満たすために使用された索引もすべて除去されます。) 例えば、列 A が除去され、固有制約 UNIQUE(A) または PRIMARY KEY(A) が存在し、この固有制約を参照する参照制約がない場合は、この操作が許されます。 それ以外の場合は、列の除去は許されません。	固有制約は、その固有制約を参照する参照制約と同じように、除去されます。(したがって、それらの制約によって使用された索引もすべて除去されます。)
参照制約の中で参照した列の除去	参照制約の中で参照した列がすべて同時に除去される場合は、それらの列および制約は除去されます。(したがって、外部キーによって使用された索引も除去されます。) 例えば、列 B が削除され、参照制約 FOREIGN KEY (A) が存在する場合は、この操作が許されます。 それ以外の場合は、列の除去は許されません。	参照制約は除去されます。(したがって、外部キーによって使用された索引も除去されます。)

400 ページの表 42 に列の変更によるカスケード効果をリストしてあります。(次の表で列の変更という場合は、データ・タイプ、精度、位取り、長さ、またはヌル可能性特性の変更を意味します。)

表 42. 列の変更によるカスケード効果

操作	効果
ビューによって参照した列の変更	変更が許されます。 表に従属しているビューが再作成されます。ビューの再作成時には、新しい列属性が使用されます。
非ビュー論理ファイルで参照する列の変更	変更が許されます。 表に従属している非ビュー論理ファイルが再作成されます。論理ファイルが、変更しようとしているファイルと形式を共用する場合、および、形式が論理ファイルにおいて別のベースオン・ファイルで再使用されない場合は、論理ファイルの再作成時に新規の列属性が使用されます。 それ以外の場合は、論理ファイルの再作成時に新規の列属性は使用されません。代わりに、現行の論理ファイル属性が使用されます。
索引のキーで参照される列の変更	変更が許されます。(したがって、通常、索引は再作成されます。)
固有制約の中で参照した列の変更	変更が許されます。(したがって、通常、索引は再作成されます。) 固有制約が参照制約によって参照される場合は、外部キーの属性は、その固有キーの属性とは一致しなくなっています。その制約は定義済み検査保留状態に置かれます。
参照制約の中で参照した列の変更	変更が許されます。 • 参照制約が定義済み検査保留状態にある場合は、変更が許され、その制約を使用可能状態に置く試みがなされます。(したがって、通常、固有制約を満たすために使用した索引は再作成されます。) • 参照制約が使用可能状態にある場合は、その制約は定義済み検査保留状態に置かれます。

例

例 1

次の列を持つ新たな表 EQUIPMENT が作成されていると想定します。

列名	データ・タイプ
EQUIP_NO	INT
EQUIP_DESC	VARCHAR(50)
LOCATION	VARCHAR(50)
EQUIP_OWNER	CHAR(3)

表 EQUIPMENT に参照制約を追加して、所有者 (EQUIP_OWNER) が、表 DEPARTMENT に存在する部門番号 (DEPTNO) でなければならないようにします。ある部門が表 DEPARTMENT から除去された場合は、その部門が所有していた備品すべての所有者 (EQUIP_OWNER) の値は、所有者未定 (またはヌル) に設定される必要があります。制約に、名前 DEPTQUIP を指定しています。表 DEPARTMENT は、(DEPTNO) として定義された基本キーを持つものと想定しています。

```
ALTER TABLE EQUIPMENT
  ADD CONSTRAINT DEPTQUIP
   FOREIGN KEY (EQUIP_OWNER)
   REFERENCES DEPARTMENT
   ON DELETE SET NULL
```

例 2

例 1 の場合と同じ表の存在を想定します。

- 各備品番号ごとの在庫数量が入る列を表 EQUIPMENT に追加します。列 QUANTITY を呼び出します。

```
ALTER TABLE EQUIPMENT
  ADD COLUMN QUANTITY INT
```

- 列 EQUIP_OWNER のデフォルト値を 'ABC' に変更します。

```
ALTER TABLE EQUIPMENT
  ALTER COLUMN EQUIP_OWNER
  SET DEFAULT 'ABC'
```

- 列 LOCATION を除去します。ビュー、索引、または制約がその列に対して構築されている場合は、それらもすべて除去します。

```
ALTER TABLE EQUIPMENT
  DROP COLUMN LOCATION CASCADE
```

- SUPPLIER と呼ばれる新しい列が追加され、LOCATION と呼ばれる既存の列が除去され、新しい列 SUPPLIER に対する固有制約が追加され、基本キーが既存の列 EQUIP_NO に対して構築されるように、表を変更します。

```
ALTER TABLE EQUIPMENT
  ADD COLUMN SUPPLIER INT
  DROP COLUMN LOCATION
  ADD UNIQUE SUPPLIER
  ADD PRIMARY KEY EQUIP_NO
```

- 列 EQUIP_DESC が可変長列であることに注意してください。25 という長さが割り振られている場合、次の ALTER TABLE ステートメントはその割り振られた長さを変更しません。

```
ALTER TABLE EQUIPMENT
  ALTER COLUMN EQUIP_DESC
  SET DATA TYPE VARCHAR(60)
```

BEGIN DECLARE SECTION

BEGIN DECLARE SECTION ステートメントは、SQL 宣言セクションの開始を示します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込んで使用します。それ以外の使用法はありません。このステートメントは、実行可能ステートメントではありません。Java、RPG または REXX では指定できません。

権限

権限は不要です。

構文

説明

BEGIN DECLARE SECTION ステートメントは、ホスト言語の規則に従って変数宣言を置ける場所であれば、アプリケーション・プログラム内のどこにでもコーディングすることができます。ホスト構造体の宣言の内部に、コーディングすることはできません。このステートメントを使用して、SQL 宣言セクションの始まりを示します。

SQL 宣言セクションは、END DECLARE SECTION ステートメントで終了します。END DECLARE SECTION ステートメントの詳細については、665 ページの『END DECLARE SECTION』を参照してください。

BEGIN DECLARE SECTION と END DECLARE SECTION ステートメントは、対にして使用しなければなりません。また、これらのステートメントをネストすることはできません。

DECLARE VARIABLE および INCLUDE ステートメント以外の SQL ステートメントは、宣言セクション内にコーディングしてはなりません。

プログラムに SQL 宣言セクションの指定がある場合は、その SQL 宣言セクションで宣言されている変数だけがホスト変数として使用できます。プログラムに SQL 宣言セクションの指定がない場合は、そのプログラムの中の変数はすべてがホスト変数として使用できます。

RPG および REXX 以外のホスト言語では、そのソース (原始) プログラムが SQL の IBM SQL 規格に適合するように、SQL 宣言セクションを指定する必要があります。SQL 宣言セクションは、C++ のすべてのホスト変数に必須です。SQL 宣言セクションは、変数に対する最初の参照よりも前にコーディングされている必要があります。 JAVA および RPG では、このようなステートメントを使用せずにホスト変数の宣言が行われ、また REXX ではホスト変数の宣言はまったく行われません。

SQL 宣言セクションの外側で宣言されている変数の名前は、SQL 宣言セクション内で宣言されている変数と同じ名前であってはなりません。

複数の SQL 宣言セクションを、プログラムに指定することができます。

例

例 1

C プログラムで、ホスト変数の hv_smint (SMALLINT)、hv_vchar24 (VARCHAR(24))、および hv_double (FLOAT) を定義します。

```
EXEC SQL  BEGIN DECLARE SECTION;
  static short                          hv_smint;
  static struct {
           short hv_vchar24_len;
           char  hv_vchar24_value[24];
         }                              hv_vchar24;
  static double                         hv_double;
EXEC SQL  END DECLARE SECTION;
```

例 2

COBOL プログラムで、ホスト変数 HV-SMINT (SMALLINT)、HV-VCHAR24 (VARCHAR(24))、および HV-DEC72 (DECIMAL(7,2)) を定義します。

```
WORKING-STORAGE SECTION.
EXEC SQL BEGIN DECLARE SECTION  END-EXEC.
  01  HV-SMINT              PIC S9(4)       BINARY.
  01  HV-VCHAR24.
      49 HV-VCHAR24-LENGTH  PIC S9(4)       BINARY.
      49 HV-VCHAR24-VALUE   PIC X(24).
  01  HV-DEC72              PIC S9(5)V9(2)  PACKED-DECIMAL.
EXEC SQL END DECLARE SECTION  END-EXEC.
```

CALL

CALL ステートメントはプロシージャーを呼び出します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。このステートメントは、動的に準備できる実行可能ステートメントです。

権限

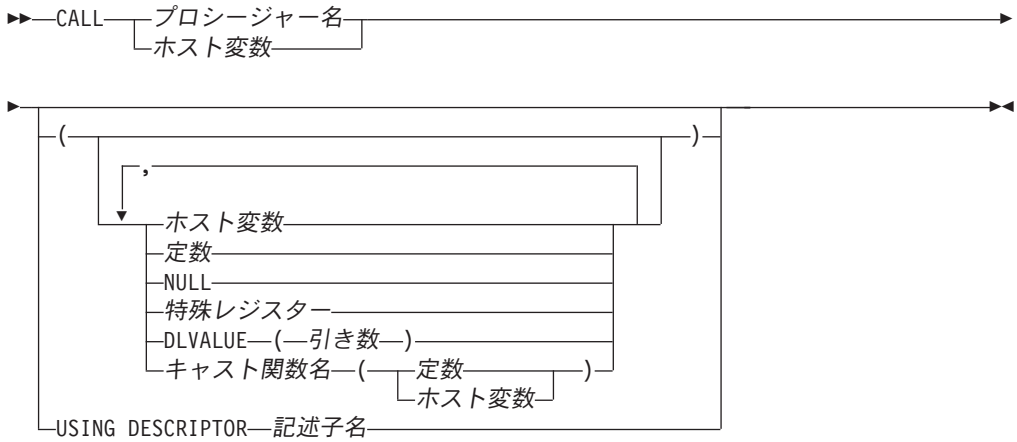
このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- 呼び出すプロシージャーが 外部 REXX プロシージャーである場合
  - そのプロシージャーに関連するソース・ファイルに対する *OBJOPR、*READ、および *EXECUTE システム権限
  - そのソース・ファイルを含むライブラリーに対する *EXECUTE システム権限、および
  - CL コマンドに対する *USE システム権限
- 呼び出すプロシージャーが Java 外部プロシージャーである場合
  - Java クラスを含む統合ファイル・システム・ファイルに対する読み取り権限 (*R)
  - 統合ファイル・システム・ファイルを検出するためにアクセスする必要があるすべてのディレクトリーに対する読み取りおよび実行権限 (*RX)
- 呼び出すプロシージャーが外部プロシージャーではあるが、REXX または Java 外部プロシージャーではない場合
  - そのプロシージャーに関連するプログラムに対する *EXECUTE システム権限、および
  - そのプロシージャーに関連するプログラムを含むライブラリーに対する *EXECUTE システム権限
- 呼び出すプロシージャーが SQL プロシージャーである場合
  - そのプロシージャーに対する EXECUTE 特権。および、
  - その SQL プロシージャーを含むライブラリーに対する *EXECUTE システム権限
- 管理権限

次の場合、ステートメントの権限 ID には、プロシージャーに対する EXECUTE 特権が与えられます。

- そのプロシージャーの所有者である。
- そのプロシージャーに対する EXECUTE 特権が認可されている、または
- そのプロシージャーに対するシステム権限 *OBJOPR および *EXECUTE が認可されている。

構文



説明

プロシージャー名 または*ホスト変数*

指定したプロシージャー名、またはホスト変数に含まれているプロシージャー名によって、呼び出したいプロシージャーを識別します。このプロシージャー名は、現行サーバーに存在しているプロシージャーを識別するものでなければなりません。ホスト変数を指定する場合、

- ホスト変数は文字ストリング変数または UCS-2 グラフィック・ストリングでなければならず、標識変数を含んでいてはなりません。
- ホスト変数内に含まれるプロシージャー名は左寄せでなければならず、その長さがホスト変数の長さより短い場合は、右側にブランクを埋め込まなければなりません。
- プロシージャーの名前は、区切り文字付きの名前でないかぎり、大文字でなければなりません。

プロシージャー名は、修飾されなかった場合、パラメーターのパスと番号に基づいて暗黙的に修飾されます。詳細については、54 ページの『非修飾オブジェクト名の修飾』を参照してください。

プロシージャー名が、CREATE PROCEDURE ステートメントや DECLARE PROCEDURE ステートメントによって定義されたプロシージャーを識別し、かつ、現行サーバーが DB2 UDB for iSeries サーバーである場合は、次のようになります。

- その CREATE PROCEDURE または DECLARE PROCEDURE ステートメントによって、外部プログラム、言語、および呼び出し規則の名前が決まります。
- そのプロシージャーのパラメーターの属性が、CREATE PROCEDURE または DECLARE PROCEDURE ステートメントによって定義されます。

上記以外の場合、

- 現行サーバーが外部プログラム、言語、および呼び出し規則の名前を決定します。

- 現行サーバーが DB2 UDB for iSeries である場合、
  - 外部プログラム名は、外部プロシージャー名と同じであると想定されます。
  - そのプログラムのプログラム属性情報が認識可能な言語を示している場合には、その言語が使用されます。それ以外の場合は、言語は C であると見なされます。
  - 呼び出し規則は GENERAL と見なされます。
- アプリケーション・リクエスターは、ホスト変数またはパラメーター・マーカーであるパラメーターはすべて INOUT であると想定します。ホスト変数以外のパラメーターは、すべて IN であると見なされます。
- パラメーターの実際の属性は、現行サーバーによって決定されます。

  現行サーバーが DB2 UDB for iSeries である場合、パラメーターの属性は、CALL ステートメント上に指定された引き数の属性と同じになります。[44]

ホスト変数 または *定数* または **NULL** または *特殊レジスター*
パラメーターとしてプロシージャーに渡される値のリストを識別します。n 番目の値は、プロシージャーの n 番目のパラメーターに対応付けられます。

CALL ステートメントが実行されると、その各パラメーターの値は、プロシージャーの対応するパラメーターに割り当てられます。 制御は、ホスト言語の呼び出し規則に従って、プロシージャーに渡されます。 プロシージャーの実行が完了すると、プロシージャーの各パラメーターの値は、OUT または INOUT として定義されている CALL ステートメントの対応するパラメーターに割り当てられます。 パラメーターの割り当てに使用される規則について詳しくは、88 ページの『ストリングの割り当て』 を参照してください。[45]

DLVALUE(*引き数***)**
パラメーターの値は、DLVALUE スカラー関数の結果の値になることを指定します。DLVALUE スカラー関数は、DataLink パラメーターにしか指定できません。 DLVALUE 関数は、(体系、サーバー、およびパス/ファイルの) 挿入時にリンク値を必要とします。 DLVALUE の最初の引き数は定数、ホスト変数、またはタイプされるパラメーター・マーカー (CAST(? AS データ・タイプ)) にする必要があります。 DLVALUE の 2 番目と 3 番目の引き数は、定数かホスト変数にする必要があります。

キャスト関数名
この形式の引き数は、特殊タイプやデータ・タイプ BLOB、 CLOB、 DBCLOB、 DATE、 TIME、または TIMESTAMP として定義されたパラメーターでのみ使用することができます。次の表は、これらのキャスト*関数* の許可されている使用法を示します。

---

44. 10 進定数の場合、先行ゼロは、引き数の属性を決定する際に有効になります。通常は、先行ゼロは有効数字ではありません。

45. CALL ステートメントが準備された後、組み込み SQL EXECUTE ステートメントによって実行された場合、OUT および INOUT パラメーターは割り当てられません。

パラメーター・タイプ	キャスト関数名
BLOB、CLOB、または DBCLOB に基づく特殊タイプ N	BLOB、CLOB、または DBCLOB *
DATE、TIME、または TIMESTAMP に基づく特殊タイプ N	DATE、TIME、または TIMESTAMP *
BLOB、CLOB、または DBCLOB	BLOB、CLOB、または DBCLOB *
DATE、TIME、または TIMESTAMP	DATE、TIME、または TIMESTAMP *

注:

* 関数には、QSYS2 の暗黙的または明示的スキーマ名のデータ・タイプ (または、特殊タイプのソース・タイプ) の名前と一致する名前を指定する必要があります。

定数
定数を引き数として指定します。この定数は、特殊タイプのソース・タイプの定数、あるいは、特殊タイプでない場合は、データ・タイプの定数の規則に準拠する必要があります。BLOB、 CLOB、 DBCLOB、 DATE、 TIME、および TIMESTAMP 関数の場合は、この定数をストリング定数にする必要があります。

ホスト変数
ホスト変数を引き数として指定します。このホスト変数は、特殊タイプのソース・タイプの定数、あるいは、特殊タイプでない場合は、データ・タイプの定数の規則に準拠する必要があります。

USING DESCRIPTOR *記述子名*
SQLDA を識別します。この SQLDA には、ホスト変数の有効な記述が入っていなければなりません。

CALL ステートメントの処理に先立って、SQLDA の以下のフィールドをセットしておく必要があります。 (REXX の場合は、規則が異なります。詳細については、DB2 UDB for iSeries ホスト言語での SQL プログラミング を参照してください。)

- SQLN (SQLDA に用意する SQLVAR のオカレンスの数を示します。)
- SQLDABC (SQLDA 用に割り振る記憶域のバイト数を示します。)
- SQLD (ステートメントを処理するときに、SQLDA で使用する変数の個数を指示します。)
- SQLVAR の各オカレンス (変数の属性を指示します。)

SQLDA の記憶域は、SQLVAR のオカレンスをすべて収容するのに十分な大きさで割り振らなければなりません。したがって、SQLDABC の値は、16 + SQLN*(80) よりも大きいか、または等しくなければなりません。ここで、80 は SQLVAR の 1 つのオカレンスの長さです。 LOB または特殊タイプが指定された場合には、各パラメーター・マーカーごとに 2 つの SQLVAR 項目が必要であり、SQLN はパラメーター・マーカー数の 2 倍にセットしなければなりません。

SQLD には、ゼロ以上で SQLN 以下の値をセットしなければなりません。 この値は、CALL ステートメント内のパラメーター・マーカー数と同じでなければなりません。SQLDA によって記述されている n 番目の変数は、準備済みス

テートメントの n 番目のパラメーター・マーカーに対応します。(SQLDA の説明については、 877 ページの『付録 C. SQL 記述子域 (SQLDA)』を参照してください。)

RPG/400 には、ポインターを設定する機能が用意されていないことに注意してください。SQLDA はポインターを使用して適切なホスト変数を見つけるので、ユーザーは、RPG/400 アプリケーションの外側でそのようなポインターを設定する必要があります。

使用上の注意

プロシージャー名 が CREATE PROCEDURE または DECLARE PROCEDURE ステートメントによって定義されたプロシージャーを識別している場合は、OUT または INOUT の各パラメーターは、ホスト変数として指定する必要があります。

プロシージャー名 が CREATE PROCEDURE または DECLARE PROCEDURE ステートメントによって定義されたプロシージャーを識別している場合は、指定する引き数の数は、その CREATE PROCEDURE または DECLARE PROCEDURE ステートメントによって定義されているパラメーターの数と同じでなければなりません。

定数 とホスト変数 の詳細については、105 ページの 『定数』と 121 ページの『ホスト変数に対する参照』を参照してください。*特殊レジスター* の詳細については、111 ページの 『特殊レジスター』を参照してください。NULL はヌル値を指定します。

呼び出したい外部プロシージャーが REXX プロシージャーである場合、そのプロシージャーは、CREATE PROCEDURE または DECLARE PROCEDURE ステートメントを使用して宣言する必要があります。

ホスト変数は、REXX プロシージャー内では CALL ステートメントに使用することはできません。その代わりとして、CALL は、パラメーター・マーカーを使用して PREPARE と EXECUTE のオブジェクトにする必要があります。

SQL や外部プロシージャーが呼び出されると、そのプロシージャーの作成時に定義された SQL データ・アクセスに対して属性が設定されます。それらの属性に使用できる値は、次のとおりです。

```
NONE
CONTAINS
READS
MODIFIES
```

次のような場合に、2 番目のプロシージャーが現行プロシージャーの実行中に呼び出されるとエラーが出されます。

- 呼び出されたプロシージャーに SQL が使用されているが、呼び出しプロシージャーで SQL が許可されていない。
- 呼び出されたプロシージャーが SQL データを読み取っているが、呼び出しプロシージャーで SQL データの読み取りを許可していない。
- 呼び出されたプロシージャーが SQL データを変更したが、呼び出しプロシージャーで SQL データの変更を許可していない。

結果セットがプロシージャーから戻されるのは、そのプロシージャーが、iSeries Access ODBC ドライバーを使用するクライアント、または iSeries Access 最適化 SQL API を使用するクライアント、SQL 呼び出しレベル・インターフェース、または JDBC から呼び出される場合だけです。 プロシージャーから結果セットを戻す方法には、次の 3 つがあります。

- SET RESULT SETS ステートメントがプロシージャーで実行される場合は、その SET RESULT SETS ステートメントが結果セットを識別します。結果セットは、SET RESULT SETS ステートメントで指定した順序で戻されます。
- SET RESULT SETS ステートメントがプロシージャーで実行されない場合
  - WITH RETURN 文節でカーソルが指定されていない場合、プロシージャーがオープンし、オープン状態のまま戻す各カーソルが、それぞれ 1 つの結果セットを識別します。結果セットは、カーソルがオープンされた順序で戻されます。
  - WITH RETURN 文節でカーソルが指定されている場合、WITH RETURN 文節で定義されたカーソルのうち、プロシージャーがオープンし、オープン状態のまま戻す各カーソルが、それぞれ 1 つの結果セットを識別します。 結果セットは、カーソルがオープンされた順序で戻されます。

オープン・カーソルを使用して結果セットが戻される場合、現行カーソル位置から始まる行が戻されます。

CALL ステートメントのネスト

プロシージャーとして実行しているプログラム、ユーザー定義の関数、またはトリガーで CALL ステートメントを出すことができます。プロシージャー、ユーザー定義の関数、またはトリガーでプロシージャー、ユーザー定義の関数、またはトリガーを呼び出すと、その呼び出しはネストされるものであると見なされます。プロシージャーと関数のネストのレベル数には制限は設けられていませんが、トリガーの場合は、最大 300 レベルだけしかネストできません。

プロシージャーが何らかの照会の結果セットを戻す場合、その結果セットは、プロシージャーの呼び出し元に戻されます。 SQL CALL ステートメントがネストされた場合、その結果セットは、直前のネスト・レベルのプログラムだけに表示されます。例えば、クライアント・プログラムがプロシージャー PROCA を呼び出すと、そのプロシージャーは、プロシージャー PROCB を呼び出します。 PROCA だけが PROCB によって戻された結果セットをアクセスすることができます。クライアント・プログラムは、照会の結果セットをアクセスすることはできません。

例

プロシージャー PGM1 を呼び出し、2 つのパラメーターを渡します。

```
CALL PGM1 (:hv1,:hv2)
```

CLOSE

CLOSE ステートメントは、カーソルをクローズします。カーソルのオープン時に結果表が作成された場合は、その表は壊されます。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込んで使用します。それ以外の使用法はありません。このステートメントは、動的には準備できない実行可能ステートメントです。 JAVA ではこのステートメントは指定できません。

権限

権限は不要です。カーソルを使用するために必要な権限については、 597 ページの『DECLARE CURSOR』を参照してください。

構文

説明

カーソル名

クローズするカーソルを識別します。*カーソル名* は、DECLARE CURSOR ステートメントの項の説明に従って宣言されているカーソルを識別しなければなりません。CLOSE ステートメントは、オープン状態にあるカーソルに対して実行しなければなりません。

使用上の注意

以下の時点では、プログラム内のすべてのカーソルはクローズ状態にあります。

- プログラムが呼び出されたとき。
  - CLOSQLCSR(*ENDPGM) が指定されている場合、プログラムが呼び出されるたびに、すべてのカーソルがクローズ状態になります。
  - CLOSQLCSR(*ENDSQL) が使用されている場合、1 つの SQL プログラムが呼び出しスタックに残っている間は、プログラムが初めて呼び出されるときに限って、すべてのカーソルがクローズ状態になります。
  - CLOSQLCSR(*ENDJOB) が指定されている場合、ジョブ内でプログラムが最初に呼び出されたときに限って、すべてのカーソルがクローズ状態になります。
  - CLOSQLCSR(*ENDMOD) が指定されている場合、モジュールが開始されるたびに、すべてのカーソルがクローズ状態になります。
  - CLOSQLCSR(*ENDACTGRP) が指定されている場合、活動化グループ内で、プログラム中のモジュールが最初に開始されたときに、すべてのカーソルがクローズ状態になります。
- HOLD オプションの指定がない COMMIT または ROLLBACK ステートメントを実行して、プログラムから新しい作業単位を開始したとき。HOLD オプションを指定して宣言されたカーソルは、COMMIT ステートメントではクローズされません。

注: DB2 UDB for iSeries データベース・マネージャーは、照会をインプリメントするためにファイルをオープンします。このファイルのクローズは、SQL CLOSE ステートメントとは別に行うことができます。詳細については、SQL プログラミング 概念を参照してください。

カーソルを、できるだけ早い時機に明示的にクローズすることによって、パフォーマンスを向上させることができます。

例

COBOL プログラムでカーソル C1 を使用し、EMPPROJACT 表の最初の 4 つの列から一度に 1 行ずつ値を取り出して、その値を次のホスト変数に入れます。

- EMP (CHAR(6))
- PRJ (CHAR(6))
- ACT (SMALLINT)
- TIM (DECIMAL(5,2))

最後にカーソルをクローズします。

```
EXEC SQL  BEGIN DECLARE SECTION  END-EXEC.
  77  EMP              PIC X(6).
  77  PRJ              PIC X(6).
  77  ACT              PIC S9(4) BINARY.
  77  TIM              PIC S9(3)V9(2) PACKED-DECIMAL.
EXEC SQL  END DECLARE SECTION  END-EXEC.
.
.
.

EXEC SQL  DECLARE C1 CURSOR FOR
            SELECT EMPNO, PROJNO, ACTNO, EMPTIME
              FROM EMPPROJACT                        END-EXEC.

EXEC SQL  OPEN C1  END-EXEC.

EXEC SQL  FETCH C1 INTO :EMP, :PRJ, :ACT, :TIM  END-EXEC.

IF SQLSTATE = '02000'
  PERFORM DATA-NOT-FOUND
ELSE
  PERFORM GET-REST UNTIL SQLSTATE IS NOT EQUAL TO '00000'.

EXEC SQL  CLOSE C1  END-EXEC.

GET-REST
EXEC SQL  FETCH C1 INTO :EMP, :PRJ, :ACT, :TIM  END-EXEC.
.
.
.
```

COMMENT

COMMENT ステートメントは、種々のデータベース・オブジェクトのカタログ記述にコメントを追加したり、置換したりします。

呼び出し

このステートメントは、アプリケーション・プログラムに埋め込むことも、対話式に呼び出すこともできます。このステートメントは、動的に準備できる実行可能ステートメントです。

権限

表、ビュー、別名、索引、列、特殊タイプ、またはパッケージに対してコメントを付けるには、ステートメントの権限 ID が保持する特権に、次のうち少なくともいずれか 1 つを含める必要があります。

- ステートメント内に示されている表、ビュー、別名、索引、特殊タイプ、またはパッケージの場合
  - その表、ビュー、別名、索引、特殊タイプ、またはパッケージに対する ALTER 特権、および
  - その表、ビュー、別名、索引、特殊タイプ、またはパッケージが収められているライブラリーに対するシステム権限の *EXECUTE
- 管理権限

次の場合、ステートメントの権限 ID には、表、ビュー、別名、索引、特殊タイプ、またはパッケージに対する ALTER 特権が与えられます。

- その表、ビュー、別名、索引、特殊タイプ、またはパッケージの所有者である。
- その表、ビュー、別名、特殊タイプ、またはパッケージに対する ALTER 特権が与えられている。あるいは、
- その表、ビュー、別名、索引、特殊タイプ、またはパッケージに対する *OBJALTER または *OBJMGT のいずれかのシステム権限が与えられている。

トリガーに対してコメントを付けるには、ステートメントの権限 ID が保持する特権に、次のうち少なくともいずれか 1 つを含める必要があります。

- ステートメント内のトリガーの対象表に対して、
  - 対象表に対する ALTER 特権
  - 対象表が入っているライブラリーに対するシステム権限の *EXECUTE
- 管理権限

関数に対してコメントを付けるには、ステートメントの権限 ID が保持する特権に、次のうち少なくともいずれか 1 つを含める必要があります。

- SYSFUNCS カタログ・ビューの場合
  - ビューに対する UPDATE 特権
  - QSYS2 ライブラリーに対する *EXECUTE システム権限
- 管理権限

プロシージャーに対してコメントを付けるには、ステートメントの権限 ID が保持する特権に、次のうち少なくともいずれか 1 つを含める必要があります。

- SYSPROCS カタログ・ビューの場合
  - ビューに対する UPDATE 特権、および
  - QSYS2 ライブラリーに対する *EXECUTE システム権限
- 管理権限

パラメーターに対してコメントを付けるには、ステートメントの権限 ID が保持する特権に、次のうち少なくともいずれか 1 つを含める必要があります。

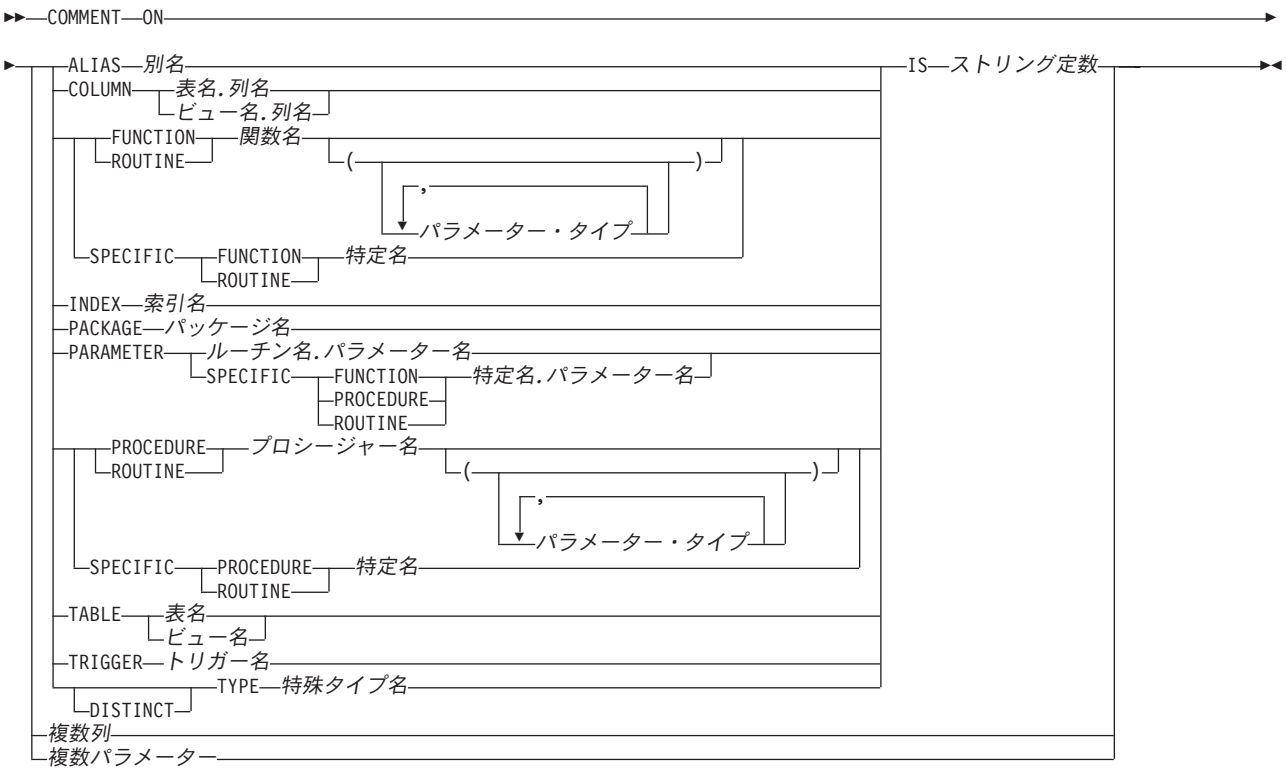
- SYSPARMS カタログ表の場合
  - その表に対する UPDATE 特権、および
  - QSYS2 ライブラリーに対する *EXECUTE システム権限
- 管理権限

次のいずれかが真の場合、ステートメントの権限 ID は、表に対する UPDATE 特権を持っています。

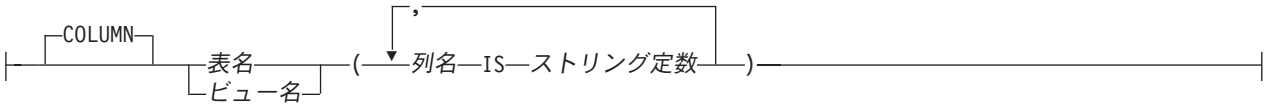
- その表の所有者である。
- その表に対する UPDATE 特権が認可されている
- その表に対する *OBJOPR および *UPD のシステム権限が認可されている。

COMMENT

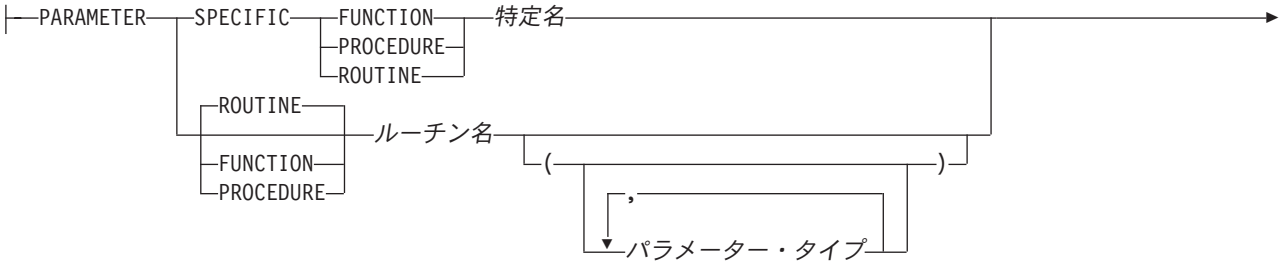
構文


COMMENT ON
ALIAS 別名
COLUMN 表名.列名
ビュー名.列名
FUNCTION
ROUTINE
関数名
(
,
パラメーター・タイプ
)
SPECIFIC
FUNCTION
ROUTINE
特定名
INDEX 索引名
PACKAGE パッケージ名
PARAMETER ルーチン名.パラメーター名
SPECIFIC
FUNCTION
PROCEDURE
ROUTINE
特定名.パラメーター名
PROCEDURE
ROUTINE
プロシージャー名
(
,
パラメーター・タイプ
)
SPECIFIC
PROCEDURE
ROUTINE
特定名
TABLE 表名
ビュー名
TRIGGER トリガー名
TYPE 特殊タイプ名
DISTINCT
IS ストリング定数
複数列
複数パラメーター

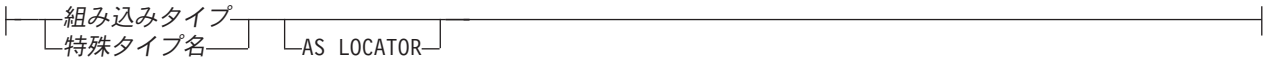
複数列:



複数パラメーター:

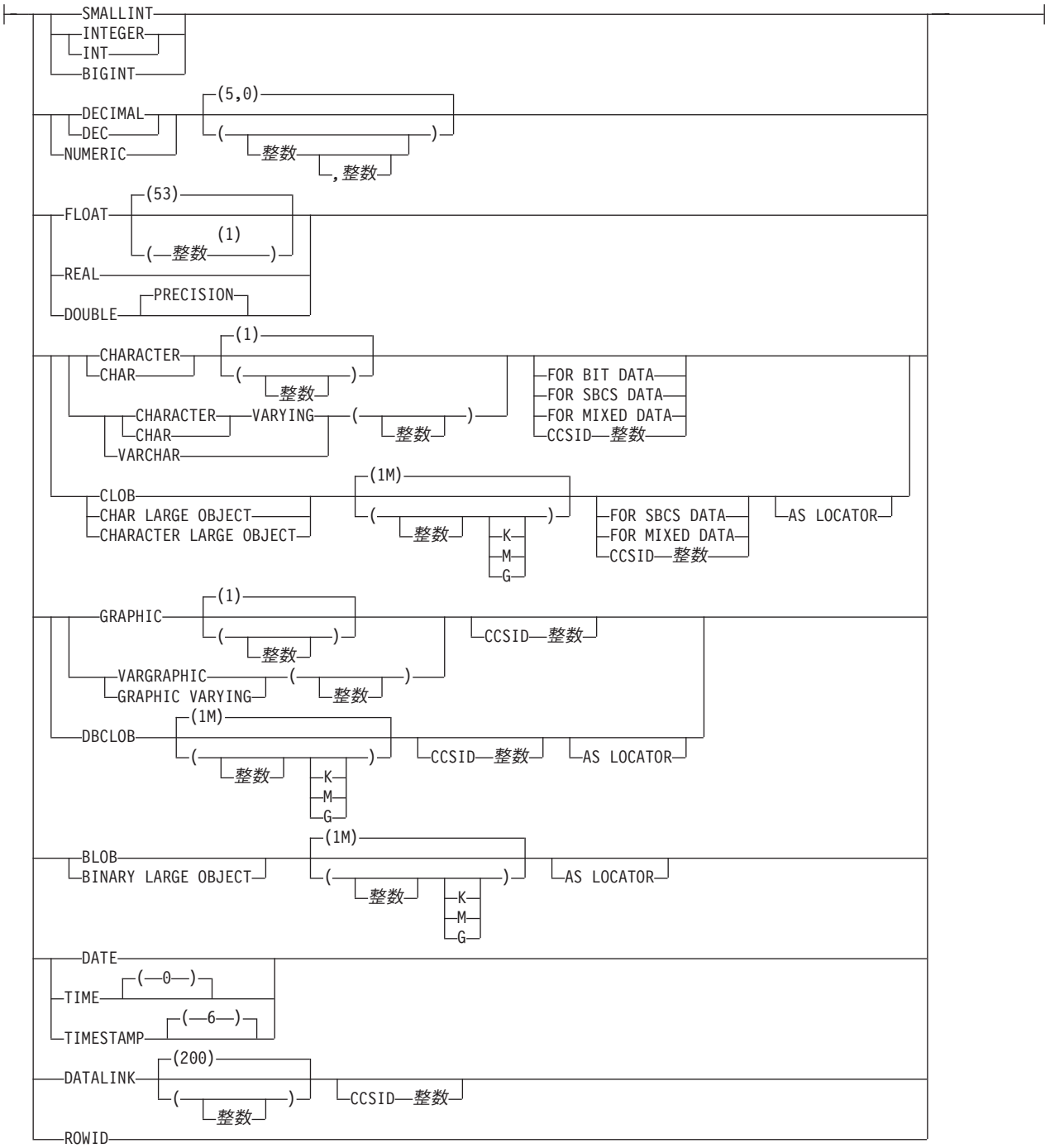


パラメーター・タイプ:



COMMENT

組み込みタイプ:



注:

1 突き合わせは、データ・タイプ (REAL または DOUBLE) に基づいて行われるので、精度に指定された値は、関数の作成時に 指定された値に一致している必要はありません。

説明

ALIAS *別名*
コメントの適用対象である別名を識別します。この名前によって、現行サーバーに存在している別名を識別する必要があります。

COLUMN
列に対してコメントの追加、またはコメントの置き換えを行うことを指定します。

表名.列名 または*ビュー名.列名*
コメントの追加、または置き換えを行う列を識別します。*表名* または*ビュー名* は、現行サーバーにある表またはビューを示すものでなければなりませんが、グローバル一時表を示すものであってはなりません。*列名* は、その表またはビューの列を識別するものでなければなりません。

DISTINCT TYPE *特殊タイプ名*
コメントが適用される特殊タイプを指定します。この名前は、現行サーバーに存在する特殊タイプを示すものでなければなりません。

FUNCTION
関数に対してコメントの追加や置換を行うことを指定します。コメントの適用対象である関数を識別します。関数は、それぞれその名前、関数シグニチャー、あるいは特定名によって識別することができます。 関数解決 (および SQL パス) に関する規則は、使用されません。

FUNCTION *関数名*
関数名 では、現行サーバーに存在している厳密に 1 つの関数を識別する必要があります。この関数には、パラメーターをいくつでも定義することができます。指定されたスキーマまたは暗黙のスキーマの中に、指定された名前の関数が複数ある場合、エラーが戻されます。

FUNCTION *関数名 (パラメーター・タイプ、...)*
関数名 (パラメーター・タイプ、...) は、現行サーバーに存在していて、指定された関数シグニチャーを持つ関数を識別する必要があります。 指定されたパラメーターは、CREATE FUNCTION ステートメント上の対応する位置に指定されたデータ・タイプと一致していなければなりません。データ・タイプの数、ならびにそれらのデータ・タイプの論理連結を使用して、コメントの適用対象である特定の関数インスタンスを識別します。*関数名 ()* を指定する場合、識別される関数にパラメーターを使用することはできません。

関数名
関数の名前を識別します。

(パラメーター・タイプ、...)
関数のパラメーターを識別します。

非修飾の特殊タイプ名を指定する場合、データベース・マネージャーはその特殊タイプのスキーマ名を解決するための SQL パスを検索します。

長さ属性、精度属性、あるいは位取り属性があるデータ・タイプの場合、値を指定することも、1 組の空の括弧を使用することもできます。

- 中が空の括弧は、データ・タイプが一致しているか否かの判別時にデータベース・マネージャー によって属性が無視されることを示します。
- 長さ属性、精度属性、あるいは位取り属性に特定の値を使用する場合、その値は、CREATE FUNCTION ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。
- 長さ属性、精度属性、または位取り属性が明示的に指定されておらず、空の括弧も指定されていない場合、該当のデータ・タイプのデフォルト属性が暗黙指定されます。次の例を見てください。

```
CHAR      CHAR(1)
GRAPHIC   GRAPHIC(1)
DECIMAL   DECIMAL(5,0)
FLOAT     DOUBLE (length of 8)
```

暗黙の長さは、CREATE FUNCTION ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。データ・タイプのデフォルト長さの全リストは、542 ページの『CREATE TABLE』 を参照してください。

サブタイプまたは CCSID 属性のあるデータ・タイプの場合は、FOR DATA 文節または CCSID 文節の指定は任意選択です。どちらか一方の文節を省略すると、データ・タイプが一致しているか否かの判別時にデータベース・マネージャー によって属性が無視されることが指示されます。 どちらか一方の文節を指定する場合は、CREATE FUNCTION ステートメントに暗黙的または明示的に指定されている値と一致させる必要があります。

SPECIFIC FUNCTION *特定名*

特定名 では、現行サーバーに存在している特定関数を識別する必要があります。

INDEX

索引についてコメントの追加または置き換えを行うことを指定します。

索引名

コメントが適用される索引を指定します。この名前は、現行サーバーに存在する索引を示すものでなければなりません。

PACKAGE

パッケージに対してコメントの追加や置換を行うことを指定します。

パッケージ名

コメントを付けるパッケージを識別します。この名前は、現行サーバーに存在しているパッケージを識別していなければなりません。

PARAMETER

パラメーターに対してコメントの追加や置換を行うことを指定します。

ルーチン名.パラメーター名

コメントが適用されるパラメーターを識別します。このパラメーターの指定対象には、プロシージャーや関数があります。*ルーチン名* では、現行サーバーに存在しているプロシージャーや関数を識別し、*パラメーター名* では、そのプロシージャーや関数のパラメーターを識別する必要があります。

特定名.パラメーター名
コメントが適用されるパラメーターを識別します。このパラメーターの指定対象には、プロシージャーや関数があります。*特定名* では、現行サーバーに存在しているプロシージャーや関数を識別し、*パラメーター名* では、そのプロシージャーや関数のパラメーターを識別する必要があります。

PROCEDURE
プロシージャーに対してコメントの追加や置換を行うことを指定します。コメントが適用されるプロシージャーを識別します。プロシージャーは、それぞれその名前、プロシージャー・シグニチャー、あるいは特定名によって識別することができます。 プロシージャー解析 (および SQL パス) に関する規則は、使用されません。

PROCEDURE *プロシージャー名*
プロシージャー名 は、現行サーバーに存在しているただ 1 つのプロシージャーを識別していなければなりません。このプロシージャーには、パラメーターをいくつでも定義することができます。指定されたスキーマまたは暗黙のスキーマの中に、指定された名前のプロシージャーが複数ある場合、エラーが戻されます。

PROCEDURE *プロシージャー名 (パラメーター・タイプ、...)*
このプロシージャー名 (パラメーター・タイプ、...) は、現行サーバーに存在していて、指定されたプロシージャー・シグニチャーを持つプロシージャーを識別する必要があります。指定されたパラメーターは、CREATE PROCEDURE ステートメント上の対応する位置に指定されたデータ・タイプと一致していなければなりません。除去するプロシージャー・インスタンスを識別する場合、データ・タイプの数とデータ・タイプの論理連結が使用されます。*プロシージャー名 ()* を指定する場合、識別されるプロシージャーにパラメーターを使用することはできません。

プロシージャー名
プロシージャーの名前を識別します。

(パラメーター・タイプ、...)
プロシージャーのパラメーターを識別します。

非修飾の特殊タイプ名を指定する場合、データベース・マネージャーはその特殊タイプのスキーマ名を解決するための SQL パスを検索します。

長さ属性、精度属性、あるいは位取り属性があるデータ・タイプの場合、値を指定することも、1 組の空の括弧を使用することもできます。

- 中が空の括弧は、データ・タイプが一致しているか否かの判別時にデータベース・マネージャー によって属性が無視されることを示します。
- 長さ属性、精度属性、あるいは位取り属性に特定の値を使用する場合、その値は、CREATE PROCEDURE ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。
- 長さ属性、精度属性、または位取り属性が明示的に指定されておらず、空の括弧も指定されていない場合、該当のデータ・タイプのデフォルト属性が暗黙指定されます。次の例を見てください。

```
CHAR      CHAR(1)
GRAPHIC   GRAPHIC(1)
DECIMAL   DECIMAL(5,0)
FLOAT     DOUBLE (length of 8)
```

暗黙の長さは、CREATE PROCEDURE ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。データ・タイプのデフォルト長さの全リストは、542 ページの『CREATE TABLE』 を参照してください。

サブタイプまたは CCSID 属性のあるデータ・タイプの場合は、FOR DATA 文節または CCSID 文節の指定は任意選択です。どちらか一方の文節を省略すると、データ・タイプが一致しているか否かの判別時にデータベース・マネージャー によって属性が無視されることが指示されます。 どちらか一方の文節を指定する場合は、CREATE PROCEDURE ステートメントに暗黙的または明示的に指定されている値と一致させる必要があります。

SPECIFIC PROCEDURE *特定名*

特定名 は、現行サーバーに存在している特定のプロシージャーを識別していなければなりません。

TABLE *表名または ビュー名*

コメントを付ける表またはビューを識別します。名前は、現行サーバーに存在する表またはビューを示している必要がありますが、グローバル一時表を示すものであってはなりません。

TRIGGER *トリガー名*

コメントが適用されるトリガーを識別します。この名前は、現行サーバーに存在しているトリガーを識別していなければなりません。

IS

この後に、付加するコメントを指定します。

ストリング定数

2000 文字以内であれば、どんな文字ストリング定数でも構いません。この定数には、SBCS 文字または DBCS 文字を含むことができます。

複数列

表またはビューの複数の列に対してコメントを行うには、その表またはビューを指定してから、以下の形式のリストを括弧で囲んで指定します。

```
列名 IS ストリング定数,
列名 IS ストリング定数, ...
```

列名は修飾してはなりません。それぞれの名前は、指定した表またはビューの列を識別しなければならず、その表またはビューは、現行サーバーに存在していなければなりません。

複数パラメーター

プロシージャーや関数の複数のパラメーターに対してコメントを付けるには、プロシージャー名、関数名、または特定名を指定してから、次の形式でリストを括弧で囲んで指定します。

```
パラメーター名 IS ストリング定数,
パラメーター名 IS ストリング定数, ...
```

パラメーター名は修飾することはできません。また、それぞれの名前では、指定されたプロシージャーや関数のパラメーターを識別し、しかも、そのプロシージャーや関数は現行サーバーに入れておく必要があります。

同義のキーワード

以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- PACKAGE の同義語として、キーワード PROGRAM を使用することができます。
- キーワード DATA を DISTINCT の同義語として使用することができます。

例

例 1

表 EMPLOYEE に関するコメントを挿入します。

```
COMMENT ON TABLE EMPLOYEE
  IS 'Reflects first quarter 1981 reorganization'
```

例 2

ビュー EMP_VIEW1 に関するコメントを挿入します。

```
COMMENT ON TABLE EMP_VIEW1
  IS 'View of the EMPLOYEE table without salary information'
```

例 3

表 EMPLOYEE の列 EMPNO に関するコメントを挿入します。

```
COMMENT ON COLUMN EMPLOYEE.EMPN
  IS 'Highest grade level passed in school'
```

例 4

DEPARTMENT 表内の 2 列に対してコメントを入力します。

```
COMMENT ON DEPARTMENT
  (MGRNO IS 'EMPLOYEE NUMBER OF DEPARTMENT MANAGER',
   ADMRDEPT IS 'DEPARTMENT NUMBER OF ADMINISTERING DEPARTMENT')
```

例 5

CORPDATA.PAYROLL パッケージに関するコメントを挿入します。

```
COMMENT ON PACKAGE CORPDATA.PAYROLL
  IS 'This package is used for distributed payroll processing.'
```

COMMIT

COMMIT ステートメントは、作業単位を終了させ、その作業単位によって行われたデータベースの変更をコミットします。

呼び出し

このステートメントは、アプリケーション・プログラムに埋め込むことも、対話式に呼び出すこともできます。このステートメントは、動的に準備できる実行可能ステートメントです。Java では指定できません。

トリガー・プログラムとその対象となるプログラムが同じコミットメント定義のもとで実行される場合、トリガーでは COMMIT は許されません。リモート・サーバーで呼び出されるプロシージャーの場合は、COMMIT をそのプロシージャーで使用することはできません。

権限

権限は不要です。

構文

説明

COMMIT ステートメントは、そのステートメントが実行される作業単位を終了させ、新たな作業単位を開始します。このステートメントは、対象の作業単位の中で SQL スキーマ・ステートメント (DROP SCHEMA を除く) および SQL データ変更ステートメントにより行われたすべての変更をコミットします。 SQL スキーマ・ステートメントおよび SQL データ変更ステートメントについては、365 ページの表 32 および 366 ページの表 33 を参照してください。

解放保留状態の接続は終了します。

WORK
COMMIT WORK は、COMMIT と同じ効果を持ちます。

HOLD
リソースを保持するように指示します。 これを指定すると、現在オープンされているカーソルはクローズされず、その作業単位の途中で獲得されたすべてのリソースは保持されます。特定の行およびオブジェクトについて、その作業単位の中で暗黙的に獲得されたロックは、解放されます。

HOLD を省略すると、

- この作業単位のコミットメント定義のもとでオープンされたカーソルは、WITH HOLD 文節を指定して宣言されている場合を除いてクローズされます。
- この作業単位のコミットメント定義のもとで LOCK TABLE ステートメントによって確立された表のロックは、解放されます。

クローズされないカーソルに必要なオブジェクト・レベルのロックを除き、暗黙に獲得されたロックは、すべて解放されます。

保留されていないロケーターはすべて解放されます。保留状態のロケーターについての詳細は、701 ページの『HOLD LOCATOR』を参照してください。

使用上の注意

条件によっては、暗黙の COMMIT が行われる場合があります。ただし、以下に示すような理由から、アプリケーションが終了する前に、明示的に COMMIT または ROLLBACK を出すことをお勧めします。

- デフォルトの活動化グループの場合
  - デフォルトの活動化グループのもとで実行されているアプリケーションが終了する時点で、暗黙の COMMIT は実行されません。デフォルトの活動化グループのもとで実行されるプログラムには、対話式 SQL、Query Manager、および非 ILE プログラムなどがあります。
  - 作業をコミットするには、COMMIT を出す必要があります。
- デフォルト以外の活動化グループで、コミットメント定義の有効範囲がその活動化グループの場合
  - その活動化グループが正常終了する時点で、そのコミットメント定義は暗黙のうちにコミットされます。
  - その活動化グループが異常終了する場合、コミットメント定義は暗黙のうちにロールバックされます。
- 活動化グループがどのようなタイプであっても、コミットメント定義の有効範囲がジョブであれば、暗黙のコミットが行われることはありません。

1 つの作業単位には、最大 4,000,000 行までの処理を組み込むことができます。これには、SELECT や FETCH ステートメントの過程で検索された行[46]、ならびに、INSERT、DELETE、および UPDATE ステートメントの一部として挿入、削除、または更新された行も含まれます。[47]

コミットおよびロールバック操作が DROP SCHEMA ステートメントに影響することはありません。したがって、このステートメントは、 COMMIT(*CHG)、COMMIT(*CS)、 COMMIT(*ALL)、または COMMIT(*RR) も指定しているアプリケーション・プログラムでは使用できません。

SQL で使用されるコミットメント定義は、次のように決められます。

- SQL を呼び出すプログラムの活動化グループがすでに活動化グループ・レベルのコミットメント定義を使用している場合、SQL はそのコミットメント定義を使用します。

---

46. この制限には、次のものも含まれます。
    - 高水準言語のファイル処理機能によるコミットメント制御のもとでオープンされたファイルに基づいてアクセスまたは変更された行。
    - トリガー、または CASCADE、SET NULL、あるいは SET DEFAULT 参照保全削除規則の結果として削除、更新、または挿入された行。

47. COMMIT(*CHG) または COMMIT(*CS) を指定した場合は例外で、これらの行は行数の合計には含まれません。

- SQL を呼び出すプログラムの活動化グループがジョブ・レベルのコミットメント定義を使用している場合、SQL はそのジョブ・レベルのコミットメント定義を使用します。
- SQL を呼び出すプログラムの活動化グループがその時点でコミットメント定義を使用していない場合、ジョブ・コミットメント定義が開始されていれば、SQL はそのジョブ・コミットメント定義を使用します。
- SQL を呼び出すプログラムの活動化グループがその時点でコミットメント定義を使用しておらず、しかもジョブ・コミットメント定義が開始されていない場合、SQL は暗黙的にコミットメント定義を開始します。SQL は、以下の指定を伴うコミットメント制御開始 (STRCMTCTL) コマンドを使用します。
  - CMTSCOPE(*ACTGRP) パラメーター
  - CRTSQLxxx、STRSQL、または RUNSQLSTM のいずれかのコマンドで指定された COMMIT オプションに基づく LCKLVL パラメーター REXX では、LCKLVL パラメーターは、SET OPTION ステートメントのコミット・オプションに基づきます。

例

C プログラム (COMMIT(*CHG) を使用してコンパイルされる) の中で、EMPLOYEE 表のある従業員 (EMPNO) から別の従業員へ、ある金額の手数料 (COMM) を移します。一方の行から手数料の金額を引いた上で、その金額をもう一方の行に加えます。この両方の操作が正常に完了した場合に、データベースへの永続的な変更を行うように、COMMIT WORK ステートメントを使用しています。

```
void main ()
  {

     EXEC SQL  BEGIN DECLARE SECTION;
     decimal(5,2) AMOUNT;
     char FROM_EMPNO[7];
     char TO_EMPNO[7];
     EXEC SQL  END DECLARE SECTION;
     EXEC SQL  INCLUDE SQLCA;
     EXEC SQL  WHENEVER SQLERROR GOTO SQLERR;
     ...
     EXEC SQL  UPDATE EMPLOYEE
               SET COMM = COMM - :AMOUNT
               WHERE EMPNO = :FROM_EMPNO;
     EXEC SQL  UPDATE EMPLOYEE
               SET COMM = COMM + :AMOUNT
               WHERE EMPNO = :TO_EMPNO;
   FINISHED:
     EXEC SQL  COMMIT WORK;
     return;

   SQLERR:
      ...
     EXEC SQL  WHENEVER SQLERROR CONTINUE;  /* continue if error on rollback */
     EXEC SQL  ROLLBACK WORK;
     return;
 }
```

CONNECT (タイプ 1)

CONNECT (タイプ 1) ステートメントは、リモート作業単位の規則を使用して、アプリケーション・プロセス内の活動化グループを識別されたサーバーに接続します。次に、このサーバーはその活動化グループの現行サーバーになります。このタイプの CONNECT ステートメントが使用されるのは、RDBCNNMTH(*RUW) が CRTSQLxxx コマンドで指定されている場合です。これら 2 つのタイプのステートメントの相違点については、926 ページの 『CONNECT (タイプ 1) と CONNECT (タイプ 2) の相違点』を参照してください。接続状態の詳細については、31 ページの 『アプリケーション指向の分散作業単位』を参照してください。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことと、対話式に呼び出すことだけが可能です。このステートメントは、動的には準備できない実行可能ステートメントです。Java または REXX では指定できません。

CONNECT は、トリガー、関数、または、リモート・サーバーで呼び出されるプロシージャーでは、使用できません。

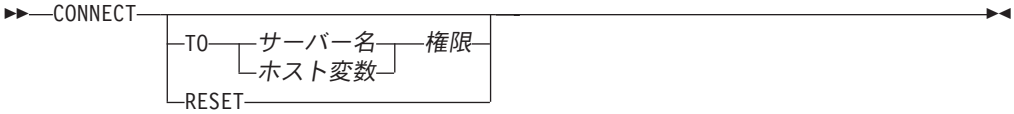
権限

このステートメントの権限 ID によって保持される特権には、通信レベルのセキュリティーが含まれていなければなりません。(資料 分散データベース・プログラミング のセキュリティーに関する節を参照してください。)

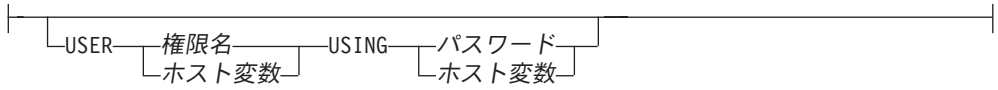
サーバーが DB2 UDB for iSeries である場合は、次の場合に該当しないかぎり、ステートメントの発行者のユーザー・プロファイルを、サーバー・システム上でも有効なユーザー・プロファイルにする必要があります。

- ユーザーが指定されている。この場合は、USER 文節で、サーバー・システム上の有効なユーザー・プロファイルを指定する必要があります。
- TCP/IP が、サーバーのサーバー権限記入項目で使用されている。この場合は、サーバー権限記入項目で、サーバー・システム上の有効なユーザー・プロファイルを指定する必要があります。

構文



権限:



説明

TO サーバー名 またはホスト変数
指定したサーバー名、または指定したホスト変数に入っているサーバー名によってサーバーを識別します。ホスト変数を指定する場合、

- その変数は、文字ストリング変数でなければなりません。
- 標識変数を伴っていてはなりません。
- サーバー名は、その変数内で左寄せし、通常 ID の形成の規則に従っていなければなりません。
- サーバー名の長さが、ホスト変数の長さよりも短い場合、右側をブランクで埋めなければなりません。

サーバー名がローカル・リレーショナル・データベースの場合、指定する権限名は、ジョブの権限名でなければなりません。 指定された権限名がジョブのものと異なっていると、エラーが発生し、アプリケーションは接続されない状態のままになります。

この CONNECT ステートメントが実行される時点で、指定したサーバー名またはホスト変数に入っているサーバー名は、ローカル・ディレクトリーに記述されているサーバーを識別していなければなりません。また、その活動化グループは、接続可能状態でなければなりません。

RESET
CONNECT RESET は、CONNECT TO x と同等です。ここで、x はローカル・サーバー名です。

CONNECT オペランドの指定なし
この形式の CONNECT ステートメントは、現行サーバーについての情報を戻し、接続状態、オープン・カーソル、準備されたステートメント、またはロックに対してまったく影響を与えません。情報は、前述のように、SQLCA に入れられます。

USER *権限名またはホスト変数*
指定された*権限名*、またはリモート・ジョブの開始に使用される権限名が入っている*ホスト変数* によって、権限名を識別します。

ホスト変数 を指定する場合、

- 文字ストリング変数でなければなりません。
- 標識変数を伴っていてはなりません。
- 権限名は、そのホスト変数に左寄せして入れ、権限名の命名規則に従っていなければなりません。
- 権限名の長さが、ホスト変数の長さよりも短い場合には、右側をブランクで埋めなければなりません。

USING *パスワードまたはホスト変数*
指定されたパスワード、またはリモート・ジョブの開始に使用される権限名のパスワードが入っているホスト変数 によってパスワードを識別します。

パスワードをリテラルとして指定する場合、そのリテラルは文字ストリングでなければなりません。最大長は 128 文字です。左寄せしなければなりません。

ホスト変数 を指定する場合、

- その変数は、文字ストリング変数でなければなりません。
- 標識変数を伴っていてはなりません。
- パスワードは、ホスト変数に左寄せして入れなければなりません。
- パスワードの長さがホスト変数の長さよりも短い場合には、右側をブランクで埋めなければなりません。

使用上の注意

接続成功: CONNECT ステートメントが正常に完了した場合 :

- オープン・カーソルはすべてクローズされ、準備されたステートメントはすべて破棄され、すべてのロックは現行接続から解放されます。
- その活動化グループは、現行および休止の接続 (接続されている場合) すべてから切り離され、指定されたサーバーに接続されます。
- 接続したサーバーの名前が、特殊レジスター CURRENT SERVER に入れられます。
- サーバーに関する情報が、SQLCA のフィールド SQLERRP および SQLERRD(4) に入れられます。そのサーバーが IBM リレーショナル・データベース・プロダクトである場合は、フィールド SQLERRP の中の情報は、*pppvvrrm* の形式をとります。ただし、
  - *ppp* は、次のようにプロダクトを識別します。
    ARI (DB2 USB サーバー (VM および VSE 版) の場合)
    DSN (DB2 UDB (OS/390 および z/OS 版) の場合)
    QSQ (DB2 UDB for iSeries の場合)
    他のすべての DB2 USB プロダクトの場合は SQL
  - *vv* は、2 桁のバージョン ID です (例えば、'07' など)。
  - *rr* は、2 桁のリリースIDです (例えば、'01' など)。
  - *m* は、1 桁のモディフィケーション・レベルを示します (例えば、'0' など)。

  例えば、サーバーが DB2 UDB (OS/390 および z/OS 版) のバージョン 7 であれば、SQLERRP の値は 'DSN07010' になります。

  SQLCA の SQLERRD(4) フィールドには、サーバーがコミット可能な更新の実行を許可するか否かを示す値が入ります。 CONNECT (タイプ 1) ステートメントの場合、SQLERRD(4) には、常に値 1 が入ります。値 1 は、コミット可能な更新を行うことができること、ならびに、接続が次のようなものであることを示します。
  - 無保護会話を使用する。[48] または
  - アプリケーション・リクエスターのドライバー・プログラムとの接続に *RUW 接続方式を使用する、または
  - *RUW 接続方式を使用するローカル接続である。
- 接続についての追加の情報は SQLCA のフィールド SQLERRMC に入れられます。865 ページの 『付録 B. SQL 連絡域』を参照してください。

48. ネットワーク接続と SQL 接続との間で混同が生じる可能性を減少させるため、本書では、TCP/IP ならびに APPC を介するネットワーク接続に適用させる場合に「会話」という用語を使用します。ただし、正式には、これは APPC 接続だけに適用されます。

CONNECT (タイプ 1)

接続失敗: この CONNECT ステートメントの実行が成功しなかった場合、SQLCA の SQLERRP フィールドには、エラーを検出したアプリケーション・リクエスターのモジュールの名前が入れられます。モジュール名の最初の 3 文字は、プロダクトを識別しています。例えば、アプリケーション・リクエスターが DB2 UDB UWO (NT 版) である場合、最初の 3 文字は 'SQL' になります。

活動化グループが接続可能状態でないために、CONNECT ステートメントが正常に実行されない場合は、その活動化グループの接続状態は変更されません。CONNECT ステートメントが他の何らかの理由で正常に実行されない場合は、次のようになります。

- その活動化グループは接続可能でありながら、未接続状態のままです。
- オープン・カーソルはすべてクローズされ、準備されたステートメントはすべて破棄され、すべてのロックは現行または休止の接続すべてから解放されます。

接続可能で未接続状態のアプリケーションのみが、CONNECT または SET CONNECTION ステートメントを実行できます。

暗黙接続:

- デフォルト活動化グループで実行される場合、次の状態にあれば、SQL プログラムは、暗黙のうちにリモートのリレーショナル・データベースに接続します。
  - その活動化グループが接続可能状態にある。
  - プログラム・スタックの最初の SQL プログラムの最初の SQL ステートメントが実行される。
- デフォルト以外の活動化グループで実行されている SQL プログラムは、その活動化グループに関する最初の SQL プログラムの最初の SQL ステートメントを実行する時に、リモートのリレーショナル・データベースへ暗黙に接続します。

注: 活動化グループにより実行される最初の SQL ステートメントを CONNECT ステートメントにするのは、望ましい方法です。

APPC を RDB との接続に使用している場合、暗黙の接続では、常にアプリケーション・リクエスター・ジョブの*権限名*を送信しますが、パスワードは送信しません。 サーバー・ジョブの*権限名*が異なる場合や、パスワードを送る必要がある場合は、明示的な接続ステートメントを使用することが必要です。

TCP/IP を RDB との接続に使用している場合、暗黙の接続は上記の制約に拘束されることはありません。 ADDSVRAUTE コマンドと他の -SVRAUTE コマンドを使用すると、暗黙 (または明示) の CONNECT の実行元である所定のユーザーに対して、該当の RDB との接続で使用されるリモート権限名とパスワードを指定することが可能になります。

パスワードが ADDSVRAUTE コマンドや CHGSVRAUTE コマンドで保管されるようにするには、QRETSVRSEC システム値をデフォルト値の '0' ではなく、'1' に設定する必要があります。これらのコマンドを DRDA 接続に使用している場合に非常に重要なことは、SERVER パラメーターに RDB 名の値を英大文字で入力しなければならないと認識しておくことです。詳細については、タイプ 2 の CONNECT の節の例 2 を参照してください。

暗黙の接続の詳細については、SQL プログラミング 概念を参照してください。ユーザー・プロファイルとリレーショナル・データベースとの接続が一度確立されると、同じユーザー・プロファイルと同じリレーショナル・データベースとの以後の接続では、パスワード (指定がある場合) の妥当性が再検証されない場合があります。パスワードの妥当性の再検証は、会話がまだ活動状態であるかどうかによって異なります。詳細に関しては、分散データベース・プログラミング を参照してください。

接続状態: 接続状態については、29 ページの 『リモート作業単位の接続管理』を参照してください。 CONNECT ステートメントは、連続して使用しても正常に実行されます。これは、CONNECT ステートメントは接続可能状態からその活動化グループを除去しないからです。

アプリケーション・グループの現行または休止接続に対する CONNECT は、次のように行われます。

- サーバー名によって識別された接続が、CONNECT (タイプ 1) ステートメントを使用して確立されていた場合には、どのようなアクションも取られません。カーソルはクローズされず、準備されたステートメントは破棄されず、またロックは解放されません。
- サーバー名によって識別された接続が、CONNECT (タイプ 2) ステートメントを使用して確立されていた場合、その CONNECT ステートメントは、他の CONNECT ステートメントと同様に実行されます。

CONNECT の前に CONNECT、 COMMIT、 DISCONNECT、 SET CONNECTION、 RELEASE、または ROLLBACK 以外の SQL ステートメントがある場合は、その CONNECT は正常に実行できません。エラーを避けるために、CONNECT ステートメントを実行する前に、コミットまたはロールバック操作を実行してください。

前の現行または休止接続が、保護会話を使用して確立されていた場合には、CONNECT (タイプ 1) ステートメントは失敗します。CONNECT (タイプ 2) ステートメントを使用するか、または保護会話を使用したその接続を解放し、コミットを正しく行うことによって終了しなければなりません。

リモート・リレーショナル・データベースとの接続、ならびに、ローカル・ディレクトリーの詳細については、SQL プログラミング 概念および分散データベース・プログラミングを参照してください。

例

DRDA と共に TCP/IP を使用した接続を行う際に使用可能であり、融通性がさらに備わっている例については、CONNECT タイプ 2 の節の例 2 を参照してください。この例は、CONNECT タイプ 1 にも適用されます。

例 1

C プログラムにおいて、ユーザー JOE をサーバー TOROLAB3 に接続します。JOE のパスワードは XYZ1 です。接続が正常に完了した後で、接続したサーバーの 3 文字のプロダクトID を、ホスト変数 *product* にコピーします。

CONNECT (タイプ 1)

```
void main ()
  {
   char product[4] = "   ";
   EXEC SQL BEGIN DECLARE SECTION ;
   char username[11];
   char userpass[129];
   EXEC SQL END DECLARE SECTION ;
   EXEC SQL INCLUDE SQLCA ;

   strcpy(username,"JOE");
   strcpy(userpass,"XYZ1";
   EXEC SQL CONNECT TO  TOROLAB3
              USER :username USING :userpass;
   if (strncmp(SQLSTATE, "00000", 5) )
     { strncpy(product,sqlca.sqlerrp,3); }
   ...
   return;
  }
```

CONNECT (タイプ 2)

CONNECT (タイプ 2) ステートメントは、アプリケーション指向分散作業単位の規則を使用して、アプリケーション・プロセス内の活動化グループを識別されたサーバーに接続します。次に、このサーバーはその活動化グループの現行サーバーになります。このタイプの CONNECT ステートメントは、CRTSQLxxx コマンドに RDBCNNMTH(*DUW) の指定があった場合に使用します。これら 2 つのタイプのステートメントの相違点については、926 ページの 『CONNECT (タイプ 1) と CONNECT (タイプ 2) の相違点』を参照してください。接続状態の詳細については、31 ページの 『アプリケーション指向の分散作業単位』を参照してください。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むか、または対話式に呼び出すことができます。このステートメントは、動的には準備できない実行可能ステートメントです。Java または REXX では指定できません。

CONNECT は、トリガー、関数、または、リモート・サーバーで呼び出されるプロシージャーでは、使用できません。

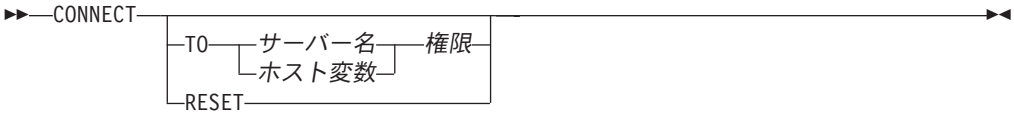
権限

このステートメントの権限 ID によって保持される特権には、通信レベルのセキュリティーが含まれていなければなりません。(分散データベース・プログラミングのセキュリティーに関する節を参照してください。)

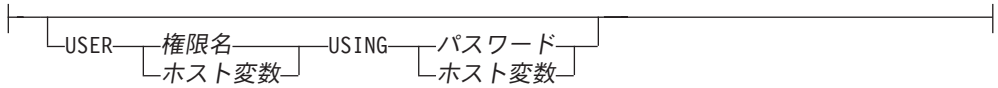
サーバーが DB2 UDB for iSeries である場合は、次の場合に該当しないかぎり、ステートメントの発行者のプロファイル ID を、サーバー・システム上でも有効なユーザー・プロファイルにする必要があります。

- USER が指定されている。USER が指定されている場合は、USER 文節で、サーバー・システム上の有効なユーザー・プロファイルを指定する必要があります。
- TCP/IP が、サーバーのサーバー権限記入項目で使用されている。この場合は、サーバー権限記入項目で、サーバー・サーバー・システム上の有効なユーザー・プロファイルを指定する必要があります。

構文



権限:



説明

TO *サーバー名 またはホスト変数*
指定したサーバー名、または指定したホスト変数に入っているサーバー名によってサーバーを識別します。ホスト変数を指定する場合、

- その変数は、文字ストリング変数でなければなりません。
- 標識変数を伴っていてはなりません。
- サーバー名は、その変数内で左寄せし、通常 ID の形成の規則に従っていなければなりません。
- サーバー名の長さが、ホスト変数の長さよりも短い場合、右側をブランクで埋めなければなりません。

この CONNECT ステートメントが実行される時点で、指定したサーバー名またはホスト変数に入っているサーバー名は、ローカル・ディレクトリーに記述されているサーバーを識別していなければなりません。

以下の説明で S は、指定したサーバー名またはホスト変数に入っているサーバー名を表しています。S は、該当のアプリケーション・プロセスの既存の接続を識別していてはなりません。

RESET
CONNECT RESET は、CONNECT TO *x* と同等です。ここで、*x* はローカル・サーバー名です。

CONNECT (オペランドの指定なし)
この形式の CONNECT ステートメントは、現行サーバーについての情報を戻し、接続状態、オープン・カーソル、準備されたステートメント、またはロックに対してまったく影響を与えません。情報は、前述のように、SQLCA のフィールドに入れられます。

さらに、SQLCA のフィールド SQLERRD(3) には、該当の作業単位の接続の状況を示す値が入れられます。次の値のいずれかが入れられます。

- 1 - この作業単位の接続では、コミット可能な更新を行うことができる。
- 2 - この作業単位の接続では、コミット可能な更新を行うことはできない。

USER *権限名またはホスト変数*
指定された *権限名*、またはリモート・ジョブの開始に使用される権限名が入っているホスト変数 によって、権限名を識別します。

ホスト変数 を指定する場合、

- 文字ストリング変数でなければなりません。
- 標識変数を伴っていてはなりません。権限名は、そのホスト変数に左寄せして入れ、権限名の命名規則に従っていなければなりません。
- 権限名の長さが、ホスト変数の長さよりも短い場合には、右側をブランクで埋めなければなりません。

USING *パスワードまたはホスト変数*
指定されたパスワード、またはリモート・ジョブの開始に使用される権限名のパスワードが入っているホスト変数 によってパスワードを識別します。

パスワードをリテラルとして指定する場合、そのリテラルは文字ストリングでなければなりません。最大長は 128 文字です。左寄せしなければなりません。

ホスト変数 を指定する場合、

- その変数は、文字ストリング変数でなければなりません。
- 標識変数を伴っていてはなりません。
- パスワードは、ホスト変数に左寄せして入れなければなりません。
- パスワードの長さがホスト変数の長さよりも短い場合には、右側をブランクで埋めなければなりません。

使用上の注意

接続成功: CONNECT ステートメントが正常に完了した場合 :

- サーバー S への接続が作成され、現行および保留状態に置かれます。それ以前の接続は (存在する場合)、休止状態になります。
- S が特殊レジスター CURRENT SERVER に入れられます。
- サーバー S に関する情報が、SQLCA のフィールド SQLERRP および SQLERRD(4) に入れられます。そのサーバーが IBM リレーショナル・データベース・プロダクトである場合は、フィールド SQLERRP の中の情報は、*pppvvrrm* の形式をとります。ただし、
  - *ppp* は、次のようにプロダクトを識別します。
    ARI (DB2 USB サーバー (VM および VSE 版) の場合)
    DSN (DB2 UDB (OS/390 および z/OS 版) の場合)
    QSQ (DB2 UDB for iSeries の場合)
    他のすべての DB2 USB プロダクトの場合は SQL
  - *vv* は、2 桁のバージョン ID です (例えば、'07' など)。
  - *rr* は、2 桁のリリースIDです (例えば、'01' など)。
  - *m* は、1 桁のモディフィケーション・レベルを示します (例えば、'0' など)。

  例えば、サーバーが DB2 UDB (OS/390 および z/OS 版) のバージョン 7 であれば、SQLERRP の値は 'DSN07010' になります。

  SQLCA の SQLERRD(4) フィールドには、サーバー S がコミット可能な更新の実行を許可するか否かを示す値が入ります。以下は、この CONNECT に関して SQLCA のフィールド SQLERRD(4) に入れられる値とその意味を示しています。
  - 1 - コミット可能な更新を行うことができる。会話は、無保護会話です。[48]
  - 2 - コミット可能な更新は行うことができない。会話は、無保護会話です。
  - 3 - コミット可能な更新を行うことができるか否かは不明である。会話は、保護会話です。
  - 4 - コミット可能な更新を行うことができるか否かは不明である。会話は、無保護会話です。
  - 5 - コミット可能な更新を行うことができるか否かは不明である。接続は、ローカル接続、またはアプリケーション・リクエスターのドライバー・プログラムとの接続です。
- 接続についての追加の情報は SQLCA のフィールド SQLERRMC に入れられます。865 ページの 『付録 B. SQL 連絡域』を参照してください。

接続失敗: CONNECT ステートメントが正常に実行されなかった場合は、その活動化グループの接続状態、およびその接続の状態は変わりません。

暗黙接続: 暗黙接続では、常にアプリケーション・リクエスター・ジョブの*権限名*が送信され、パスワードは送信されません。サーバー・ジョブの*権限名*が異なる場合や、パスワードを送る必要がある場合は、明示的な接続ステートメントを使用することが必要です。

TCP/IP を RDB との接続に使用している場合、暗黙の接続は上記の制約に拘束されることはありません。 ADDSVRAUTE コマンドと他の -SVRAUTE コマンドを使用すると、暗黙 (または明示) の CONNECT の実行元である所定のユーザーに対して、該当の RDB との接続で使用されるリモート権限名とパスワードを指定することが可能になります。

パスワードが ADDSVRAUTE コマンドや CHGSVRAUTE コマンドで保管されるようにするには、QRETSVRSEC システム値をデフォルト値の '0' ではなく、'1' に設定する必要があります。これらのコマンドを DRDA 接続に使用している場合に非常に重要なことは、SERVER パラメーターに RDB 名の値を英大文字で入力しなければならないと認識しておくことです。詳細については、タイプ 2 の CONNECT の節の例 2 を参照してください。

暗黙の接続の詳細については、SQL プログラミング 概念を参照してください。ユーザー・プロファイルとリレーショナル・データベースとの接続が一度確立されると、同じユーザー・プロファイルと同じリレーショナル・データベースとの以後の接続では、パスワード (指定がある場合) の妥当性が再検証されない場合があります。パスワードの妥当性の再検証は、会話がまだ活動状態であるかどうかによって異なります。詳細に関しては、分散データベース・プログラミングを参照してください。

例

例 1

TOROLAB1 および TOROLAB2 で SQL ステートメントを実行します。最初の CONNECT ステートメントは TOROLAB1 との接続を確立し、2 番目の CONNECT ステートメントはその接続を休止状態にします。

```
EXEC SQL  CONNECT TO TOROLAB1;

(TOROLAB1 にあるオブジェクトを参照するステートメントを実行)

EXEC SQL  CONNECT TO TOROLAB2;

(TOROLAB2 にあるオブジェクトを参照するステートメントを実行)
```

例 2

ユーザー JOE は、パスワードが SHIBBOLETH のユーザー ID ANONYMOUS によって TOROLAB3 に接続し、SQL ステートメントを実行したいとします。TOROLAB3 の RDB ディレクトリー項目では、その接続タイプに *IP を指定します。

アプリケーションを実行する前に、ある種のセットアップを行う必要があります。

このコマンドは、前にまだ実行していないならば、サーバーのセキュリティー情報を OS/400 に保存できるようにするために必須です。

```
CHGSYSVAL SYSVAL(QRETSVRSEC) VALUE('1')
```

このコマンドによって、必須のサーバー権限項目が追加されます。

```
ADDSVRAUTE USRPRF(JOE) SERVER(TOROLAB3) USRID(ANONYMOUS) +
           PASSWORD(SHIBBOLETH)
```

JOE のユーザー・プロファイルのもとで実行されるこのステートメントは、この時点で、必要な接続を確立します。

```
EXEC SQL CONNECT TO TOROLAB3;
(TOROLAB3 にあるオブジェクトを参照するステートメントを実行)
```

CREATE ALIAS

CREATE ALIAS ステートメントは、現行サーバーにあるデータベース・ファイルの表、ビュー、またはメンバーの別名を定義します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことも、対話式に呼び出すこともできます。このステートメントは、動的に準備できる実行可能ステートメントです。

権限

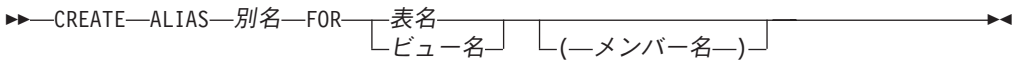
このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- 次のシステム権限
  - DDM ファイル作成 (CRTDDMF) コマンドに対する *USE
  - 別名が作成されるライブラリーに対する *EXECUTE、*READ、および *ADD
- 管理権限

SQL 名が指定され、別名が作成されるライブラリーと同じ名前のユーザー・プロファイルが存在し、しかも、その名前がステートメントの権限 ID と異なっている場合、ステートメントの権限 ID が保持している特権には、少なくとも次のいずれか 1 つを含める必要があります。

- その名前を持つユーザー・プロファイルに対する *ADD システム権限
- 管理権限

構文



説明

別名

別名を指定します。暗黙的または明示的修飾子も含め、この名前は、現行サーバーにすでに存在している索引、表、ビュー、別名、またはファイルと同じ名前にすることはできません。

SQL 名が指定されている場合、別名は、暗黙的または明示的修飾子で指定しているスキーマ内に作成されます。

システム名が指定されている場合、別名は、修飾子で指定しているスキーマ内に作成されます。修飾されていない場合、別名は、その別名の作成の対象である表かビューと同じスキーマ内に作成されます。表が修飾されず、しかも別名の作成時に存在していなかった場合、別名は、現行ライブラリー (*CURLIB) 内に作成されます。

別名が有効なシステム名でない場合、DB2 UDB for iSeries は、システム名を生成します。名前の生成に関する規則については、572 ページの 『表名の生成の規則』を参照してください。

FOR *表名* または *ビュー名*
別名の定義の対象である現行サーバーの表やビューを識別します。 別名を指定することはできません (別名は、別の別名を参照することはできません)。

表名 やビュー名 では、別名の作成時に存在していた表やビューを識別する必要はありません。表やビューが別名の作成時に存在していない場合は、警告が戻されます。別名の使用時に表やビューが存在していない場合は、エラーが戻されます。

SQL 名が指定されており、*表名* またはビュー名 が修飾されていない場合の修飾子は、暗黙の修飾子です。詳細については、47 ページの 『命名規則』を参照してください。

システム名が指定されており、*表名* やビュー名 が修飾されておらず、しかも別名の作成時に存在していない場合、*表名* やビュー名 は、別名が作成されるライブラリーによって修飾されます。

メンバー名
データベース・ファイルのメンバーを識別します。

メンバーが指定されている場合、別名は、データ操作 (DML) SQL ステートメント内でのみ使用することができます。メンバー名が指定されていない場合は、*FIRST が使用されます。

使用上の注意

データベース・ファイル・オーバーライド (OVRDBF) CL コマンドを使用すれば、データベース・マネージャーは、データベース・ファイルの個々のメンバーを処理することができるようになります。しかし、データベース・ファイルのメンバーに対する別名を作成する方が、オーバーライドを実行する必要がなくなるため、簡単でしかもパフォーマンスも向上します。

別名は、システム名または SQL 名を参照するように定義することができます。しかし、システム名は作成処理時に生成されるものなので、SQL 名を指定する方をお勧めします。

別名の属性 : 別名は特殊形式の DDM ファイルとして作成されます。

分散表を介して作成される別名は、現行サーバー上でのみ作成されます。分散表の詳細については、DB2 UDB for iSeries マルチ・システムを参照してください。

別名の所有権 : SQL 名を指定した場合は、別名の*所有者* は、作成した別名が入れられるスキーマと同じ名前のユーザー・プロファイルです。その他の場合は、別名の*所有者* は、このステートメントを実行しているジョブのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

システム名を指定した場合は、別名の*所有者* は、このステートメントを実行しているジョブのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

別名の権限 : SQL 名を使用する場合は、別名は、*PUBLIC に対するシステム権限 *EXCLUDE を使用して作成されます。システム名を使用する場合、別名は、スキーマの作成権限 (CRTAUT) パラメーターによって決められる *PUBLIC に対する権限を使用して作成されます。

別名の所有者がグループ・プロファイルのメンバー (GRPPRF キーワード) であり、グループ権限が指定されている (GRPAUT キーワード) 場合は、そのグループ・プロファイルにも、その別名に対する権限が与えられます。

同義のキーワード: 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。 これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- キーワード SYNONYM は、ALIAS の同義語として使用することができます。

例

例 1

PROJECT 表に対して、CURRENT_PROJECTS という別名を作成します。

```
CREATE ALIAS CURRENT_PROJECTS
  FOR PROJECT
```

例 2

SALES 表の JANUARY メンバーに対して、SALES_JANUARY という別名を作成します。SALES 表には、12 メンバー (1 年のそれぞれの月ごとに 1 つずつ) があります。

```
CREATE ALIAS SALES_JANUARY
  FOR SALES(JANUARY)
```

CREATE DISTINCT TYPE

CREATE DISTINCT TYPE ステートメントは、現行サーバー上に特殊タイプを定義します。特殊タイプは、常に組み込みデータ・タイプの 1 つをソースとして作成されます。ステートメントの実行が正常に完了すると、特殊タイプとソース・タイプとの間でキャストする関数も生成され、しかも、その特殊タイプで使用するための比較演算子 (データ・リンクを除く) のサポートが生成されます。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことも、対話式に呼び出すこともできます。このステートメントは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- 特殊タイプが作成されるライブラリーに対するシステム権限
  *EXECUTE、*READ および *ADD。
- 管理権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- SYSTYPES カタログ表の場合
  - 該当の表に対する INSERT 特権、および
  - QSYS2 ライブラリーに対する *EXECUTE システム権限
- 管理権限

ステートメントの権限 ID は、次の場合に表についての INSERT 特権を持ちます。

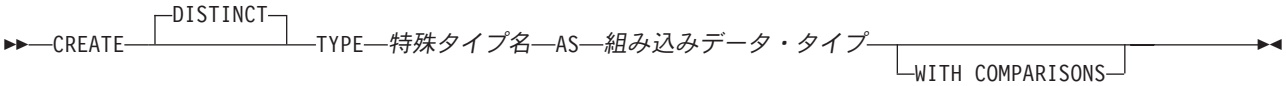
- その表の所有者である。
- その表についての INSERT 特権を認可されている、または
- その表についての *OBJOPR および *ADD システム権限が認可されている。

SQL 名が指定され、特殊タイプが作成されるライブラリーと同じ名前のユーザー・プロファイルが存在し、しかも、その名前がステートメントの権限 ID と異なっている場合、ステートメントの権限 ID が保持している特権には、少なくとも次のいずれか 1 つを含める必要があります。

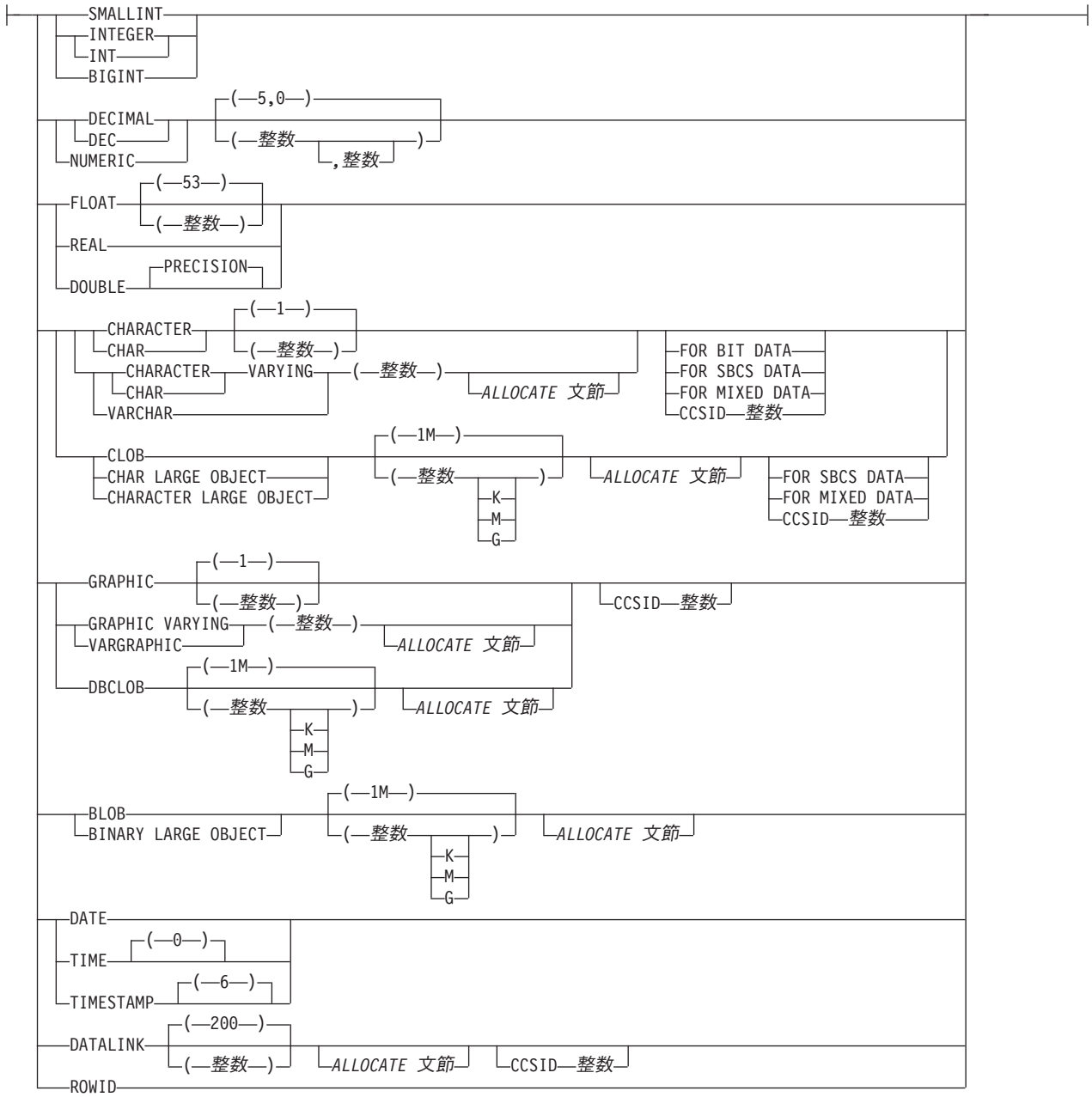
- その名前を持つユーザー・プロファイルに対する *ADD システム権限
- 管理権限

構文

CREATE DISTINCT TYPE



組み込みデータ・タイプ:



説明

特殊タイプ名

特殊タイプの名前を指定します。暗黙的または明示的修飾子も含め、この名前は、現行サーバーにすでに存在している特殊タイプと同じ名前にすることはできません。

SQL 名が指定されている場合、特殊タイプは、暗黙的または明示的修飾子で指定しているスキーマ内に作成されます。

システム名が指定されている場合、特殊タイプは、修飾子で指定しているスキーマ内に作成されます。修飾されない場合の特殊タイプは、現行ライブラリー (*CURLIB) 内に作成されます。

特殊タイプが有効なシステム名でない場合、DB2 UDB for iSeries は、システム名を生成します。名前の生成に関する規則については、572 ページの 『表名の生成の規則』を参照してください。

特殊タイプ名 は、組み込みデータ・タイプの名前にすることはできません。また、下記のシステム予約のどのキーワードにすることもできません。これは、それらのキーワードを区切り文字付き ID として指定している場合にも該当します。

=	<	>	>=
<=	<>	¬=	¬<
¬<	!=	!<	!>
ALL	FALSE	ONLY	TABLE
AND	FOR	OR	THEN
ANY	FROM	OVERLAPS	TRIM
BETWEEN	IN	PARTITION	TRUE
BOOLEAN	IS	POSITION	TYPE
CASE	LIKE	RRN	UNIQUE
CAST	MATCH	SELECT	UNKNOWN
CHECK	NODENAME	SIMILAR	WHEN
DISTINCT	NODENUMBER	SOME	
EXCEPT	NOT	STRIP	
EXISTS	NULL	SUBSTRING	

修飾付きの*特殊タイプ名* を指定した場合は、スキーマ名は QSYS、 QSYS2、QTEMP、または SYSIBM であってはなりません。

組み込みデータ・タイプ
特殊タイプの内部表示のベースとして使用されるデータ・タイプを指定します。このデータ・タイプは、組み込みデータ・タイプにする必要があります。LONG VARCHAR や LONG VARGRAPHIC を除き、CREATE TABLE ステートメントに対して許可されている組み込みデータ・タイプのどれを使用しても構いません。

長さ属性、精度属性、または位取り属性が明示的に指定されていない場合、該当のデータ・タイプのデフォルト属性が暗黙指定されます。次の例を参照してください。

```
CHAR      CHAR(1)
GRAPHIC   GRAPHIC(1)
DECIMAL   DECIMAL(5,0)
FLOAT     DOUBLE (length of 8)
```

特殊タイプのソースがストリング・データ・タイプの場合、CCSID は、その特殊タイプの作成時の特殊データ・タイプに関連しています。データ・タイプの詳細については、 542 ページの『CREATE TABLE』 を参照してください。

WITH COMPARISONS
これを指定すると、特殊タイプの 2 つのインスタンスを比較するために、シス

テム生成の比較関数が作成されます。 WITH COMPARISONS はデフォルト値です。WITH COMPARISONS の指定の有無にかかわらず、DATALINK を除くすべてのソース・タイプについて比較関数が生成されます。[49] 他の DB2 USB プロダクトとの互換性を保つために、WITH COMPARISONS を指定する必要があります。

比較関数は、LIKE 述部をサポートしません。特殊タイプ に対して LIKE 述部を使用するためには、それを組み込みタイプにキャストする必要があります。

使用上の注意

CREATE DISTINCT TYPE ステートメントの実行が正常に完了すると、データベース・マネージャーは、次のキャスト関数を生成します。

- 特殊タイプからソース・タイプに変換する 1 つの関数
- ソース・タイプから特殊タイプに変換する 1 つの関数
- ソース・タイプが SMALLINT の場合は、INTEGER から特殊タイプに変換する 1 つの関数
- ソース・タイプが REAL の場合は、DOUBLE から特殊タイプに変換する 1 つの関数
- ソース・タイプが CHAR の場合は、VARCHAR から特殊タイプに変換する 1 つの関数
- ソース・タイプが GRAPHIC の場合は、VARGRAPHIC から特殊タイプに変換する 1 つの関数

キャスト関数は、次のステートメントを実行した場合と同様に作成されます (ただし、サービス・プログラムは作成されないので、キャスト関数に対する特権を認可したり取り消したりすることはできません)。

```
CREATE FUNCTION 特殊タイプ名 (ソース・タイプ名)
  RETURNS 特殊タイプ名

CREATE FUNCTION ソース・タイプ名 (特殊タイプ名)
  RETURNS ソース・タイプ名
```

CREATE DISTINCT TYPE ステートメントのソース・データ・タイプに長さ、精度、または位取りを指定した場合でも、特殊タイプからソース・タイプに変換するキャスト関数の名前は、単に、そのソース・データ・タイプの名前です。キャスト関数が戻す値のデータ・タイプには、ソース・データ・タイプに指定した長さ、精度、または位取りの値がすべて組み込まれます。 (443 ページの表 43を参照してください。)

ソース・タイプから特殊タイプに変換するキャスト関数の名前は、その特殊タイプの名前です。キャスト関数の入力パラメーターには、長さ、精度、および位取りも含め、ソース・データ・タイプと同じデータ・タイプが指定されます。

生成されるキャスト関数は、特殊タイプと同じスキーマで作成されます。 同じ名前と同じ関数シグニチャーをもつ関数が、現行サーバー内にすでに存在していてはなりません。

---

49. これらの比較関数については、サービス・プログラムは生成されません。これらの比較関数は、SYSROUTINES カタログ表には登録されません。

例えば、T_SHOESIZE という名前の特殊タイプが、次のステートメントを使用して作成されると仮定します。

```
CREATE DISTINCT TYPE CLAIRE.T_SHOESIZE AS VARCHAR(2) WITH COMPARISONS
```

このステートメントが実行されると、データベース・マネージャーは、次のキャスト関数も生成します。 VARCHAR は、特殊タイプからソース・タイプに変換し、T_SHOESIZE は、ソース・タイプから特殊タイプに変換します。

```
FUNCTION CLAIRE.VARCHAR (CLAIRE.T_SHOESIZE) RETURNS VARCHAR(2)

FUNCTION CLAIRE.T_SHOESIZE (VARCHAR(2) RETURNS CLAIRE.T_SHOESIZE
```

関数 VARCHAR は、データ・タイプ VARCHAR(2) と一緒に値を戻し、関数 T_SHOESIZE には、データ・タイプ VARCHAR(2) と一緒に入力パラメーターが指定されます。

生成されたキャスト関数を明示的に除去することはできません。特殊タイプに対して生成されるキャスト関数は、その特殊タイプが DROP ステートメントで除去される時点で暗黙に除去されます。

次の表は、特殊タイプに対してソース・データ・タイプにすることができる組み込みデータ・タイプごとに、生成されたキャスト関数の名前、入力パラメーターのデータ・タイプ、およびそれらの関数が戻す値のデータ・タイプを示します。

表 43. 特殊タイプに対するキャスト関数

ソース・タイプ名	関数名	パラメーター・タイプ	戻りタイプ
SMALLINT	*特殊タイプ名*	SMALLINT	*特殊タイプ名*
	特殊タイプ名	INTEGER	*特殊タイプ名*
	SMALLINT	*特殊タイプ名*	SMALLINT
INTEGER	*特殊タイプ名*	INTEGER	*特殊タイプ名*
	INTEGER	*特殊タイプ名*	INTEGER
BIGINT	*特殊タイプ名*	BIGINT	*特殊タイプ名*
	BIGINT	*特殊タイプ名*	BIGINT
DECIMAL	*特殊タイプ名*	DECIMAL(p,s)	*特殊タイプ名*
	DECIMAL	*特殊タイプ名*	DECIMAL(p,s)
NUMERIC	*特殊タイプ名*	NUMERIC(p,s)	*特殊タイプ名*
	NUMERIC	*特殊タイプ名*	NUMERIC(p,s)
REAL	*特殊タイプ名*	REAL	*特殊タイプ名*
	特殊タイプ名	DOUBLE	*特殊タイプ名*
	REAL	*特殊タイプ名*	REAL
FLOAT(n) (ここで、n <= 24)	*特殊タイプ名*	REAL	*特殊タイプ名*
	特殊タイプ名	DOUBLE	*特殊タイプ名*
	REAL	*特殊タイプ名*	REAL
FLOAT(n) (ここで、n > 24)	*特殊タイプ名*	DOUBLE	*特殊タイプ名*
	DOUBLE	*特殊タイプ名*	DOUBLE

表 43. 特殊タイプに対するキャスト関数 (続き)

ソース・タイプ名	関数名	パラメーター・タイプ	戻りタイプ
DOUBLE または DOUBLE PRECISION	*特殊タイプ名*	DOUBLE	*特殊タイプ名*
	DOUBLE	*特殊タイプ名*	DOUBLE
CHAR	*特殊タイプ名*	CHAR(n)	*特殊タイプ名*
	特殊タイプ名	VARCHAR(n)	*特殊タイプ名*
	CHAR	*特殊タイプ名*	CHAR(n)
VARCHAR	*特殊タイプ名*	VARCHAR(n)	*特殊タイプ名*
	VARCHAR	*特殊タイプ名*	VARCHAR(n)
CLOB	*特殊タイプ名*	CLOB(n)	*特殊タイプ名*
	CLOB	*特殊タイプ名*	CLOB(n)
GRAPHIC	*特殊タイプ名*	GRAPHIC (n)	*特殊タイプ名*
	特殊タイプ名	VARGRAPHIC (n)	*特殊タイプ名*
	GRAPHIC	*特殊タイプ名*	GRAPHIC (n)
VARGRAPHIC	*特殊タイプ名*	VARGRAPHIC (n)	*特殊タイプ名*
	VARGRAPHIC	*特殊タイプ名*	VARGRAPHIC (n)
DBCLOB	*特殊タイプ名*	DBCLOB(n)	*特殊タイプ名*
	DBCLOB	*特殊タイプ名*	DBCLOB(n)
BLOB	*特殊タイプ名*	BLOB(n)	*特殊タイプ名*
	BLOB	*特殊タイプ名*	BLOB(n)
DATE	*特殊タイプ名*	DATE	*特殊タイプ名*
	DATE	*特殊タイプ名*	DATE
TIME	*特殊タイプ名*	TIME	*特殊タイプ名*
	TIME	*特殊タイプ名*	TIME
TIMESTAMP	*特殊タイプ名*	TIMESTAMP	*特殊タイプ名*
	TIMESTAMP	*特殊タイプ名*	TIMESTAMP
DATALINK	*特殊タイプ名*	DATALINK	*特殊タイプ名*
	DATALINK	*特殊タイプ名*	DATALINK
ROWID	*特殊タイプ名*	ROWID	*特殊タイプ名*
	ROWID	*特殊タイプ名*	ROWID
注: * 変換は、UCS-2 グラフィックの場合にのみサポートされます。 DATALINK だけを DATALINK タイプにキャストすることができます。			

可搬性のあるアプリケーションに対して特殊タイプを作成する場合、NUMERIC と FLOAT はお勧めできません。それらの代わりに、DECIMAL と DOUBLE を使用するようにしてください。

特殊タイプの属性 : 特殊タイプは *SQLUDT オブジェクトとして作成されます。

特殊タイプの所有権 : SQL 名を指定した場合は、特殊タイプの*所有者* は、作成した特殊タイプが入れられるスキーマと同じ名前のユーザー・プロファイルです。その他の場合は、特殊タイプの*所有者* は、このステートメントを実行しているジョブのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

システム名を指定した場合は、特殊タイプの*所有者* は、このステートメントを実行しているジョブのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

特殊タイプの権限 : SQL 名を使用する場合は、特殊タイプは、*PUBLIC に対するシステム権限 *EXCLUDE を使用して作成されます。システム名を使用する場合、特殊タイプタイプは、スキーマの作成権限 (CRTAUT) パラメーターによって決められる *PUBLIC に対する権限を使用して作成されます。

特殊タイプの所有者がグループ・プロファイルのメンバー (GRPPRF キーワード) であり、グループ権限が指定されている (GRPAUT キーワード) 場合は、そのグループ・プロファイルにも、その特殊タイプに対する権限が与えられます。

例

例 1

INTEGER データ・タイプをソースとする SHOESIZE という名前の特殊タイプを作成します。

```
CREATE DISTINCT TYPE SHOESIZE AS INTEGER WITH COMPARISONS
```

このステートメントの実行が正常に完了すると、2 つのキャスト関数も生成されます。関数 INTEGER(SHOESIZE) は、データ・タイプ INTEGER が指定された値を戻し、関数 SHOESIZE(INTEGER) は、特殊タイプ SHOESIZE が指定された値を戻します。

例 2

DOUBLE データ・タイプをソースとする MILES という名前の特殊タイプを作成します。

```
CREATE DISTINCT TYPE MILES AS DOUBLE WITH COMPARISONS
```

このステートメントの実行が正常に完了すると、2 つのキャスト関数も生成されます。関数 DOUBLE(MILES) は、データ・タイプ DOUBLE が指定された値を戻し、関数 MILES(DOUBLE) は、特殊タイプ MILES が指定された値を戻します。

CREATE FUNCTION

CREATE FUNCTION ステートメントを使用すると、現行サーバーに登録されるユーザー定義の関数を作成することができます。

定義できる関数のタイプは以下のとおりです。

- 外部スカラー

  このタイプの関数は、C または Java などのプログラミング言語で書かれ、スカラー値を戻します。この外部プログラムは、現行サーバーで定義されている関数により、その関数の各種属性に基づいて参照されます。 450 ページの『CREATE FUNCTION (外部スカラー)』を参照してください。

- 外部表

  このタイプの関数は、C または Java などのプログラミング言語で書かれ、一組の行を戻します。この外部プログラムは、現行サーバーで定義されている関数により、その関数の各種属性に基づいて参照されます。 467 ページの『CREATE FUNCTION (外部表)』を参照してください。

- ソース化

  このタイプの関数は、すでに現行サーバーに存在している他の関数 (組み込み、外部、ソース化、または SQL) を呼び出すことによりインプリメントされます。ソース化関数は、スカラーの結果、または列関数の結果を戻します。 482 ページの『CREATE FUNCTION (ソース化)』を参照してください。 この関数は、基礎となっているソース関数の属性を継承します。

- SQL スカラー

  このタイプの関数は SQL のみで書かれるもので、スカラー値を戻します。関数本体は、関数の各種属性と一緒に現行サーバーで定義されます。 490 ページの『CREATE FUNCTION (SQL スカラー)』 を参照してください。

- SQL 表

  このタイプの関数は SQL のみで書かれるもので、一組の行を戻します。関数本体は、関数の各種属性と一緒に現行サーバーで定義されます。 499 ページの『CREATE FUNCTION (SQL 表)』を参照してください。

使用上の注意

関数名の選択

修飾付き関数名を指定する場合は、スキーマ名は QSYS2、QSYS、QTEMP、または SYSIBM であってはなりません。

以下の名前はシステムが使用するために予約されているので、関数名には使用できません。

=	<	>	>=
<=	<>	¬=	¬<
¬<	!=	!>	!<
ALL	FALSE	ONLY	TABLE
AND	FOR	OR	THEN
ANY	FROM	OVERLAPS	TRIM
BETWEEN	IN	PARTITION	TRUE

BOOLEAN	IS	POSITION	TYPE
CASE	LIKE	RRN	UNIQUE
CAST	MATCH	SELECT	UNKNOWN
CHECK	NODENAME	SIMILAR	WHEN
DISTINCT	NODENUMBER	SOME	
EXCEPT	NOT	STRIP	
EXISTS	NULL	SUBSTRING	

入力パラメーターのデータ・タイプの選択

関数の入力パラメーターのデータ・タイプを選択する場合は、それらの入力パラメーターの値に影響を与える可能性のあるプロモーションの規則を考慮してください (80 ページの『データ・タイプのプロモーション』を参照してください)。例えば、関数の入力引き数の 1 つである定数には、その関数で予期していたデータ・タイプとは別の組み込みデータ・タイプが指定される場合があり、さらに重要なことに、その定数は、予期されていたデータ・タイプにプロモートできない可能性があります。プロモーションの規則に従って、パラメーターには、次のデータ・タイプを使用することをお勧めします。

- SMALLINT に代わって INTEGER
- REAL に代わって DOUBLE
- CHAR に代わって VARCHAR
- GRAPHIC に代わって VARGRAPHIC

DB2 UDB for iSeries 以外のプラットフォーム間における関数の可搬性を得るには、次のデータ・タイプを使用しないでください。これらのデータ・タイプの表示方法は、プラットフォームに応じてそれぞれに異なる可能性があります。

- FLOAT。この代わりに、DOUBLE や REAL を使用すること。
- NUMERIC。この代わりに、DECIMAL を使用すること。

パラメーターに AS LOCATOR を指定

値の代わりにロケーターを渡すことにより、関数との間で受け渡しするバイト数を削減できることがあります。これは、パラメーターの値が非常に大きい場合に便利です。 AS LOCATOR 文節は、実際の値の代わりにパラメーターの値へのロケーターを渡すことを指定します。 AS LOCATOR は、LOB データ・タイプまたは LOB データ・タイプに基づく特殊タイプのパラメーターの場合に限り使用するようにしてください。

AS LOCATOR 文節によって、データ・タイプがプロモート可能かどうかの決定が影響を受けることも、関数解決に使用される関数シグニチャーが影響を受けることもありません。

SQL 関数には、AS LOCATOR は指定できません。

スキーマ内の関数の固有性の決定

現行サーバーでは、それぞれの関数シグニチャーを固有のものにする必要があります。関数のシグニチャーは、入力パラメーターの数およびデータ・タイプと結合された修飾関数名です。つまり、異なる 2 つのスキーマのそれぞれに、それらに対応しているすべてのデータ・タイプに同じデータ・タイプが指定されている同じ名前

の 1 つの関数を入れることができるということです。ただし、それらに対応しているすべてのデータ・タイプに、同じデータ・タイプが指定されている同じ名前の関数を 2 つ入れることはできません。

対応しているデータ・タイプが一致しているか否かの判別時に、データベース・マネージャー は、比較において、長さ、精度、位取り、または CCSID の属性はどれも考慮に入れません。 データベース・マネージャー は、データ・タイプの同義語 (REAL と FLOAT、ならびに DOUBLE と FLOAT) を一致と見なします。したがって、CHAR(8) と CHAR(35) は同じであると見なされ、同様に、DECIMAL(11,2) と DECIMAL(4,3) は同じであると見なされます。

次のステートメントを実行して、同じスキーマ内に 4 つの関数を作成すると想定します。 2 番目と 4 番目のステートメントは、1 番目と 3 番目のステートメントが作成した関数と重複している関数を作成するので失敗します。

```
CREATE FUNCTION PART (INT, CHAR(15) ...
CREATE FUNCTION PART (INTEGER, CHAR(40) ...

CREATE FUNCTION ANGLE (DECIMAL(12,2)) ...
CREATE FUNCTION ANGLE (DEC(10,7)) ...
```

関数の特定名

名前もスキーマも同じである (ただしパラメーター・リストは異なる) 複数の関数を定義するときは、特定名も指定することをお勧めします。関数のソース化、除去、関数に対する権限の認可または取り消し、または関数へのコメントの付加を行うときに、特定名を使用して、その関数を一意的に識別することができます。ただし、この関数をその特定名で呼び出すことはできません。

SPECIFIC 文節を指定しなかった場合は、特定名が生成されます。

組み込み関数の拡張とオーバーライド

組み込み関数の拡張またはオーバーライドが必要な場合を除き、ユーザー定義の関数に組み込み関数と同じ名前を付けることはお勧めできません。

既存の組み込み関数の機能の拡張: 組み込み関数と同じ名前の新しいユーザー定義の関数と、固有の関数シグニチャーを作成します。例えば、組み込み数値タイプの代わりに特殊タイプ MONEY を入力として受け入れる、組み込み関数 ROUND に似たユーザー定義の関数が必要になったとします。

この場合、ROUND という名前の新しいユーザー定義関数のシグニチャーは、組み込み関数 ROUND がサポートするどの関数シグニチャーとも異なるものになります。

組み込み関数をオーバーライドする: 既存の組み込み関数のいずれかと名前もシグニチャーも同じである新しいユーザー定義の関数を作成します。この新しい関数は、その組み込み関数の対応するパラメーターと同じ名前およびデータ・タイプを持ちますが、インプリメントされるロジックが異なります。例えば、組み込み関数 ROUND に類似しているが、丸めの規則が異なるユーザー定義の関数が必要になったとします。

この場合、ROUND という名前の新しいユーザー定義関数のシグニチャーは、組み込み関数 ROUND がサポートするシグニチャーのどれかと同じになります。

組み込み関数をオーバーライドした場合、非修飾関数名を使用しているアプリケーションが、前回はその名前の組み込み関数を使用して正常に実行されたのに、今回は失敗するという状況が生じることがあります。さらに事態が悪化すると、一見して正常に実行されたように見えるのに、実際には、データベース・マネージャーがその組み込み関数ではなくユーザー定義の関数の方を選択したため、意図しない結果が生じるという状況が発生することもあります。

CREATE FUNCTION (外部スカラー)

CREATE FUNCTION (外部スカラー) ステートメントは、現行サーバー上に外部スカラー関数を作成します。この関数は、単一の結果を戻します。

呼び出し

このステートメントは、アプリケーション・プログラムの中に組み込んだり、あるいは、対話式に出すことができます。このステートメントは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも次のいずれか 1 つを含める必要があります。

- SYSFUNCS カタログ・ビューと SYSPARMS カタログ表の場合
  - 該当の表に対する INSERT 特権、および
  - QSYS2 ライブラリーに対する *EXECUTE システム権限
- 管理権限

ステートメントの権限 ID は、次の場合に表についての INSERT 特権を持ちます。

- その表の所有者である。
- その表についての INSERT 特権を認可されている、または
- その表についての *OBJOPR および *ADD システム権限が認可されている。

外部プログラムやサービス・プログラムが存在している場合、このステートメントの権限 ID が保持する特権には、少なくとも次のいずれか 1 つを含める必要があります。

- SQL ステートメントで参照された外部プログラムやサービス・プログラムの場合
  - その外部プログラムやサービス・プログラムが入っているライブラリーに対するシステム権限の *EXECUTE。
  - その外部プログラムやサービス・プログラムに対するシステム権限の *EXECUTE。
  - そのプログラムやサービス・プログラムに対するシステム権限の *CHANGE。システムには、プログラム・オブジェクトを更新し、関数を別のシステムに保管 / 復元するために必要な情報を入れる場合にこの権限が必要となります。ユーザーにこの権限が与えられていない場合、関数は同じように作成されますが、プログラム・オブジェクトは更新されません。
- 管理権限

SQL 名が指定され、関数が作成されるライブラリーと同じ名前のユーザー・プロファイルが存在し、しかも、その名前がステートメントの権限 ID と異なっている場合、ステートメントの権限 ID が保持している特権には、少なくとも次のいずれか 1 つを含める必要があります。

- その名前を持つユーザー・プロファイルに対する *ADD システム権限
- 管理権限

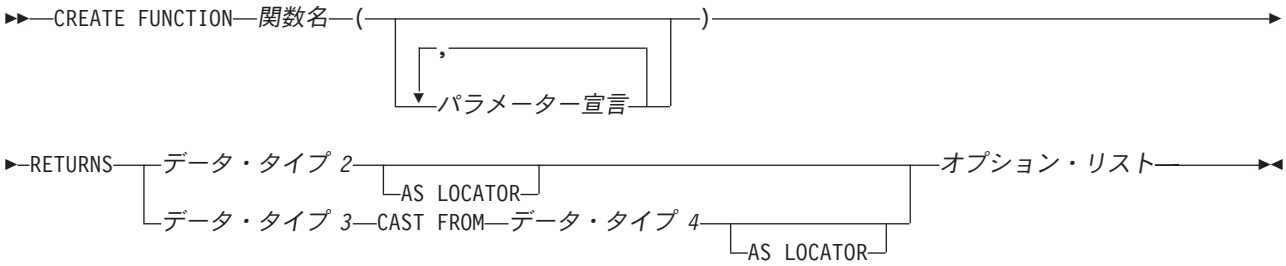
特殊タイプを参照する場合は、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- ステートメント内に識別されているそれぞれの 特殊タイプ に対しては次のもの。
  - その 特殊タイプ に対する USAGE 特権。および
  - その 特殊タイプ が入っているライブラリーに対する *EXECUTE システム権限
- 管理権限

次のいずれかに該当する場合、ステートメントの権限 ID には、特殊タイプ に対する USAGE 特権が与えられます。

- その 特殊タイプ の所有者である。
- その 特殊タイプ に対する USAGE 特権が付与されている。
- その 特殊タイプ に対するシステム権限の *OBJOPR と *EXECUTE が付与されている。

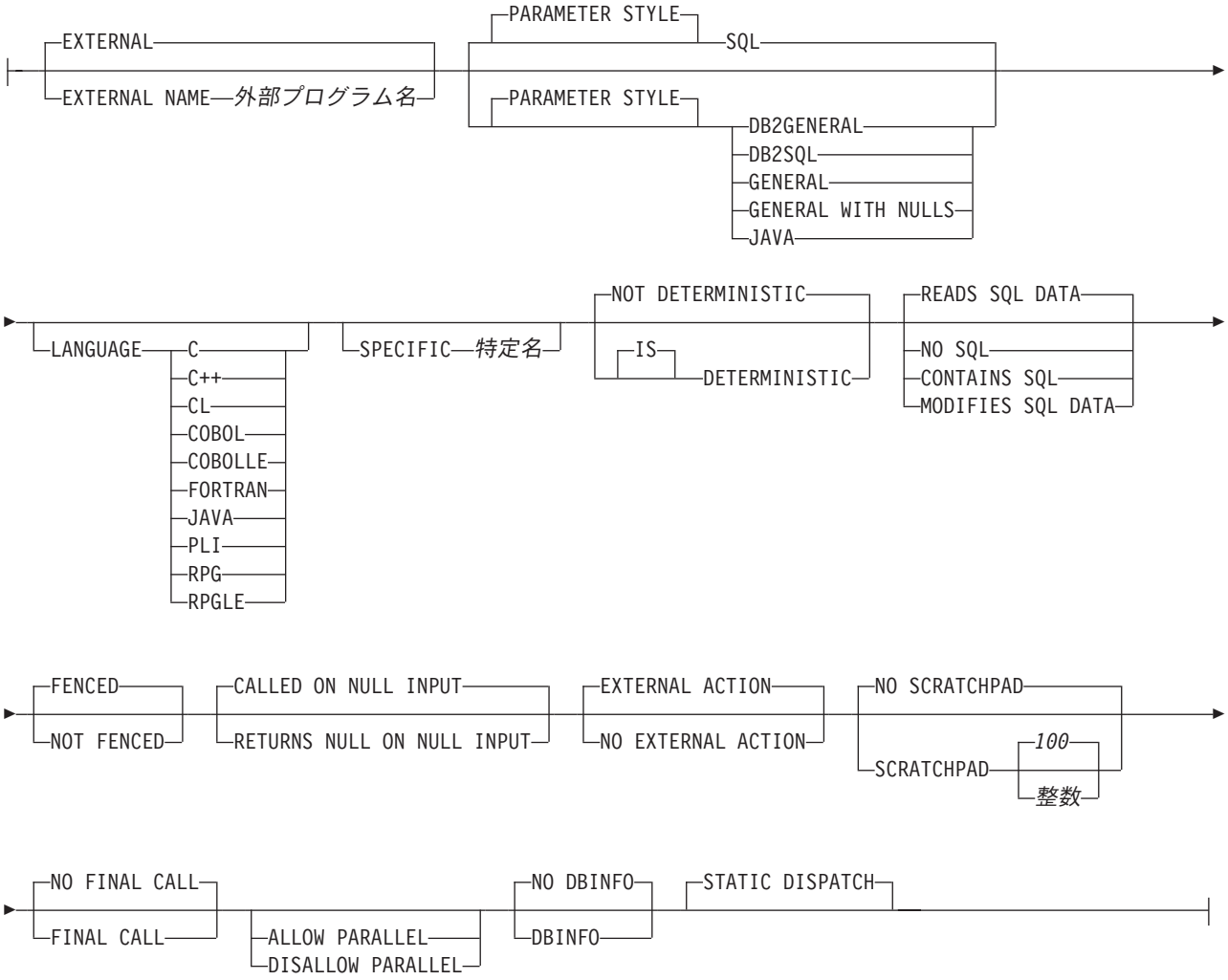
構文



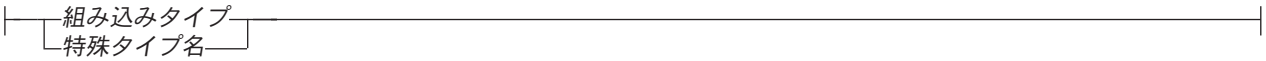
パラメーター宣言:

CREATE FUNCTION (外部スカラー)

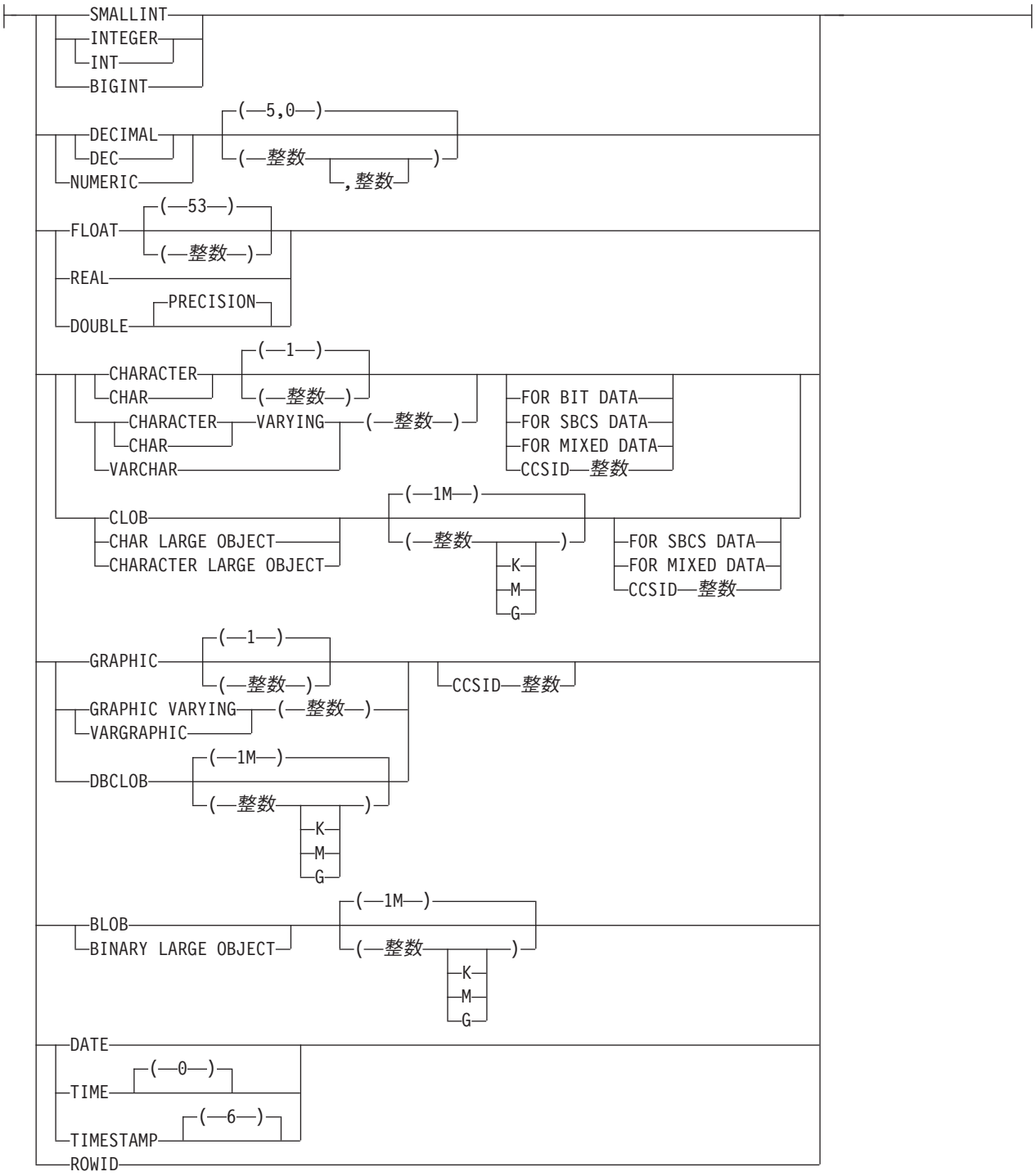
オプション・リスト:



データ・タイプ:



組み込みタイプ:



説明

関数名

ユーザー定義の関数の名前を指定します。名前、スキーマ名、パラメーターの数、およびそれぞれのパラメーターのデータ・タイプ (データ・タイプの長さ、精度、位取り、または CCSID の属性に関係なく) の組み合わせで、現行サーバー上に存在しているユーザー定義の関数を識別してはなりません。

SQL 命名の場合、関数は、暗黙または明示修飾子で指定されたスキーマ内に作成されます。

システム命名の場合、関数は、修飾子で指定されたスキーマ内に作成されます。修飾子を指定しない場合、関数は、現行ライブラリー (*CURLIB) 内に作成されます。現行ライブラリーがない場合、関数は QGPL 内に作成されます。

通常、それぞれの関数の関数シグニチャーが固有であれば、複数の関数に同じ名前を指定することができます。

| 一部の関数名は、システムが使用するために予約されています。詳細については、446 ページの『関数名の選択』を参照してください。

(パラメーター宣言,...)

関数のパラメーターの数とそれぞれのパラメーターのデータ・タイプを指定します。 それぞれのパラメーターに名前を指定することができますが、これは必須ではありません。

CREATE FUNCTION で許容されているパラメーターの最大数は 90 です。PARAMETER STYLE SQL で作成された外部関数の場合は、指定された入力パラメーターと結果パラメーターに加え、標識、SQLSTATE、関数名、特定名、およびメッセージ・テキストの暗黙のパラメーター、ならびに、あらゆるオプション・パラメーターが含まれます。パラメーターの最大数は、外部プログラムのコンパイルに使用されるライセンス・プログラムで許容されているパラメーターの最大数によっても制約されます。

パラメーター名

入力パラメーターの名前を指定します。同じ名前を何度も指定することはできません。

データ・タイプ 1

関数の入力パラメーターの数とそれぞれの入力パラメーターのデータ・タイプを指定します。関数のパラメーターはすべて入力パラメーターです。関数が受信を予期しているそれぞれのパラメーターのリストには、記入項目を 1 つ設ける必要があります。それぞれのパラメーターに名前を指定することができますが、これは必須ではありません。

| PARAMETER STYLE JAVA を指定する場合は、ラージ・オブジェクト (LOB) データ・タイプのパラメーターはサポートされません。

関数にはパラメーターを指定しなくても構いません。この場合は、次のように、中が空の 1 組の括弧をコーディングする必要があります。

```
CREATE FUNCTION WOOFER()
```

CCSID が指定されている場合、関数に渡される前に、パラメーターはその CCSID に変換されます。CCSID が指定されていない場合は、CCSID は、関数の呼び出し時点における現行サーバーのデフォルトの CCSID によって決まります。

AS LOCATOR
これを指定すると、入力パラメーターは、実際の値ではなく、値のロケーターになります。 AS LOCATOR は、入力パラメーターに LOB データ・タイプや LOB データ・タイプをベースとする特殊タイプが指定されている場合にのみ、指定することができます。

RETURNS
関数の出力を指定します。

データ・タイプ 2
出力のデータ・タイプと属性を指定します。

あらゆる組み込みデータ・タイプ (ただし、LONG VARCHAR、LONG VARGRAPHIC、または DataLink は除く) や特殊タイプ (データ・リンクをベースとしていない) を指定することができます。

CCSID が指定されている場合

- AS LOCATOR が指定されていない場合、戻される結果はその CCSID でコード化されていると想定されます。
- AS LOCATOR が指定され、ロケーターが指しているデータの CCSID が異なる CCSID でコード化されている場合、データは指定された CCSID に変換されます。

CCSID が指定されていない場合

- AS LOCATOR が指定されていない場合、戻される結果は、ジョブの CCSID (グラフィック・ストリング戻り値の場合は、ジョブに関連したグラフィック CCSID) でコード化されていると想定されます。
- AS LOCATOR が指定され、ロケーターが指しているデータの CCSID が異なる CCSID でコード化されている場合、ロケーターが指しているデータは、ジョブの CCSID に変換されます。変換時に文字が失われるのを防ぐために、関数から戻される文字をすべて表現できる CCSID を明示的に指定することを考慮してください。 これは、データ・タイプがグラフィック・ストリング・データの場合に特に重要です。 この場合、CCSID 13488 (UCS-2 グラフィック・ストリング・データ) を使用することを考慮してください。

データ・タイプ 3 **CAST FROM** *データ・タイプ 4*
出力のデータ・タイプと属性 (*データ・タイプ 4*)、および、その出力が呼び出しステートメントに戻されるデータ・タイプ (*データ・タイプ 3*) を指定します。これら 2 つのデータ・タイプは、異なっていても構いません。例えば、次の定義の場合、関数は値 CHAR(10) を戻したら、データベース・マネージャー は、その値を DATE 値に変換してから、関数を呼び出したステートメントにそれを渡します。

```
CREATE FUNCTION GET_HIRE_DATE (CHAR6)
  RETURNS DATE CAST FROM CHAR(10)
```

データ・タイプ *4* の値は、特殊タイプにすることはできません。これは、データ・タイプ *3* にキャストできるようにする必要があります。データ・タイプ *3* の値は、任意の組み込みデータ・タイプや特殊タイプにすることができます。 (データ・タイプのキャストについては、 82 ページの『データ・タイプ間のキャスト』を参照してください。)

CCSID については、上記のデータ・タイプ *2* の説明を参照してください。

AS LOCATOR
これを指定すると、関数は、実際の値ではなく、値のロケーターを戻します。AS LOCATOR は、関数の出力に LOB データ・タイプや LOB データ・タイプをベースとする特殊タイプが指定されている場合にのみ、指定することができます。

SPECIFIC *特定名*
関数の固有名を指定します。この名前は、暗黙的または明示的にスキーマ名で修飾されます。 スキーマ名も含め、この名前では、現行サーバーに存在している別の関数またはプロシージャーの特定名を識別することはできません。 修飾されない場合の暗黙の修飾子は、関数名の修飾子と同じです。修飾される場合の修飾子は、関数名の修飾子と同じものにする必要があります。

特定名を指定しなかった場合、その特定名は、関数名に設定されます。この特定名の関数やプロシージャーがすでに存在している場合は、固有表名の生成に使用される規則にほぼ準拠した固有名が生成されます。

LANGUAGE (言語文節)
言語文節は、外部プログラムの言語を指定します。

LANGUAGE を指定しなかった場合、言語は、関数の作成時に外部プログラムと関連したプログラム属性情報から決定されます。次の場合、プログラムの言語は C であると想定されます。

- プログラムに関連したプログラム属性情報で、認識可能な言語を識別しない。
- プログラムが見つからない。

C
外部プログラムは C で作成されます。

C++
外部プログラムは C++ で作成されます。

CL
外部プログラムは CL または ILE CL で作成されます。

COBOL
外部プログラムは COBOL で作成されます。

COBOLLE
外部プログラムは ILE COBOL で作成されます。

FORTRAN
外部プログラムは FORTRAN で作成されます。

JAVA
外部プログラムは JAVA で作成されます。このユーザー定義の関数はデー

タベース・マネージャーにより呼び出されます。この関数は、指定された Java クラスの共通静的メソッドでなければなりません。

PLI
外部プログラムは PL/I で作成されます。

RPG
外部プログラムは RPG で作成されます。

RPGLE
外部プログラムは ILE RPG で作成されます。

DETERMINISTIC または **NOT DETERMINISTIC**
関数が決定的であるか否かを指定します。

NOT DETERMINISTIC
これを指定すると、関数は、必ずしも、同一の入力引き数が指定された連続関数呼び出しから同じ結果を戻すとは限らなくなります。 NOT DETERMINISTIC は、特殊レジスターまたは非決定的関数に対する参照がこの関数に含まれている場合に指定してください。

DETERMINISTIC
これを指定すると、関数は、必ず、同一の入力引き数が指定された連続呼び出しから同じ結果を戻します。

CONTAINS SQL、**READS SQL DATA**、**MODIFIES SQL DATA**、または **NO SQL**
この関数がなんらかの SQL ステートメントを実行できるかどうか、および実行できる場合にどのようなタイプのステートメントを実行できるかを指定します。データベース・マネージャーは、この関数が発行する SQLがこの条件を満たしているかどうかを検査します。各データ・アクセス指示の下で実行できる SQL ステートメントの詳細なリストについては、913 ページの『付録 F. SQL ステートメントの特性』を参照してください。

CONTAINS SQL
この関数は、データを読み取りまたは変更する SQL ステートメントを実行しません。

NO SQL
この関数は SQL ステートメントを実行しません。

READS SQL DATA
この関数は、データを変更する SQL ステートメントを実行しません。

MODIFIES SQL DATA
この関数は、どの関数でもサポートされないステートメントを除くすべての SQL ステートメントを実行できます。

FENCED または **NOT FENCED**
この関数を、呼び出し元の SQL ステートメントと同じスレッド内で実行するか、別のスレッドで実行するかを指定します。

FENCED
この関数は別のスレッドで実行されます。

NOT FENCED
この関数は、呼び出し元の SQL ステートメントと同じスレッド内で実行で

きます。 NOT FENCED を指定すると、この関数の呼び出し間でカーソルをオープン状態のままにすることができます。カーソルをオープン状態のままにしておくことができるため、関数の呼び出し間でカーソル位置も同じに維持されます。

NULL INPUT
入力パラメーターが NULL である場合に、関数を呼び出す必要があるか否かを指定します。

CALLED ON NULL INPUT
常に関数を呼び出します。

RETURNS NULL ON NULL INPUT
ヌル値が渡され、関数の出力が NULL になる場合は、関数を呼び出す必要はありません。

EXTERNAL ACTION または **NO EXTERNAL ACTION**
関数に外部アクションが含まれているかどうかを指定します。

EXTERNAL ACTION
関数は、なんらかの外部アクション (関数プログラムの有効範囲外のアクション) を行います。したがって、関数は、それぞれの連続関数呼び出しで呼び出す必要があります。 EXTERNAL ACTION は、この関数に、外部アクションを持つ他の関数に対する参照が含まれている場合に、指定してください。

NO EXTERNAL ACTION
関数は外部アクションを行いません。この関数は、連続した各関数呼び出しごとに呼び出す必要はありません。

SCRATCHPAD
関数に、静的メモリー域が必要か否かを指定します。

SCRATCHPAD *整数*
これを指定すると、関数には、持続メモリー域の長さ整数が必要となります。 この整数に指定できる範囲は、1 ～ 16,000,000 です。メモリー域を指定しなかった場合、その区域のサイズは 100 バイトになります。パラメーター・スタイル DB2SQL を指定すると、ポインターは、静的記憶域を指す必須パラメーターに続いて渡されます。 PARALLEL を指定すると、メモリー域は、ステートメント内のそれぞれのユーザー定義の関数参照に割り振られます。DISALLOW PARALLEL を指定すると、1 つのメモリー域だけが関数に割り振られます。

スクラッチパッドの有効範囲は SQL です。SQL ステートメント内の関数の参照ごとに、1 つのスクラッチパッドが存在します。例えば、関数 UDFX が SCRATCHPAD キーワードによって定義されていると想定した場合、次の SQL ステートメント内では、UDFX の 3 つの参照に対して 3 つのスクラッチパッドが割り振られます。

```
SELECT A, UDFX(A)
  FROM TABLEB
  WHERE UDFX(A) > 103 OR UDFX(A) < 19
```

関数が並列タスクのもとで実行されている場合、SQL ステートメント内の関数のそれぞれの参照の並列タスクごとに、1 つのスクラッチパッドが割り振られます。これによって、予測不能の結果が生じる可能性があります。例

えば、関数で、呼び出される回数をカウントするためにスクラッチパッドを使用した場合、そのカウントは、SQL ステートメントではなく、並列タスクによって行われた呼び出し回数を反映します。並列性を使用して正しく動作しない関数には、DISALLOW PARALLEL 文節を指定してください。

SCRATCHPAD は、PARAMETER STYLE DB2SQL または PARAMETER STYLE DB2GENERAL でのみ許可されます。

NO SCRATCHPAD
これを指定すると、関数では、持続メモリー域を必要としなくなります。

FINAL CALL
関数が特殊呼び出し指示を必要とするかどうかを指定します。 PARAMETER STYLE DB2SQL を指定してある場合に、FINAL CALL を指定すると、初回呼び出しか、通常呼び出しか、または最終呼び出しかを示す追加パラメーターが、この関数に渡されます。

NO FINAL CALL
関数に対する最終呼び出しを行わないことを指定します。

FINAL CALL
関数に対する最終呼び出しを行うことを指定します。この関数は、最終呼び出しとその他の呼び出しを区別するために、呼び出しのタイプを示す追加の引き数を受け取ります。

FINAL CALL は、PARAMETER STYLE DB2SQL または PARAMETER STYLE DB2GENERAL でのみ許可されます。

呼び出しには以下のタイプがあります。

First Call (初回呼び出し)
この SQL ステートメントでの関数に対するこの参照についての、関数に対する初回呼び出しを示します。初回呼び出しは通常呼び出しです。 SQL 引き数が渡され、関数は結果を戻すものと見なされます。

Normal Call (通常呼び出し)
SQL 引き数が渡され、関数が結果を戻すことを指定します。

Final Call (最終呼び出し)
関数に対する最後の呼び出しであり、これにより関数はリソースを解放できることを指定します。最終呼び出しは通常呼び出しではありません。エラーが起きた場合は、データベース・マネージャーは最終呼び出しを行おうとします。

最終呼び出しが発生するのは以下の場合です。

- *ステートメントの終わり :* カーソル指向ステートメント用のカーソルがクローズされたとき、またはステートメントの実行が完了したとき。
- *並列タスクの終わり :* 並列タスクにより関数が実行されたとき。
- *トランザクションの終わり :* ステートメント処理の正常終了が発生しなかったとき。例えば、何らかの理由によりアプリケーションのロジックがカーソルのクローズをバイパスした場合。

WITH HOLD として定義されているカーソルがオープン状態にあるときにコミット操作が発生した場合は、カーソルがクローズされるかアプリケーションが終了した時点で、最終呼び出しが行われます。並列タスクの終わりにコミットが発生した場合は、WITH HOLD として定義されているカーソルがオープン状態にあってもなくても、最終呼び出しが行われます。

FINAL CALL ではコミット可能な操作は行ってはなりません。FINAL CALL は、COMMIT 操作の一部として呼び出されたクローズ中に実行される可能性があるからです。

PARALLEL

関数を並列で実行できるかどうかを指定します。

ALLOW PARALLEL

これを指定すると、関数は並列で実行できるようになります。

DISALLOW PARALLEL

これを指定すると、関数は並列で実行できなくなります。

以下の文節の 1 つまたは複数を指定した場合、デフォルトは DISALLOW PARALLEL になります。

- NOT DETERMINISTIC
- EXTERNAL ACTION
- FINAL CALL
- MODIFIES SQL DATA
- SCRATCHPAD

それ以外の場合は、ALLOW PARALLEL がデフォルトです。

DBINFO

関数にデータベース情報を渡す必要があるかどうかを指定します。

DBINFO

データベース・マネージャーは、状況情報が入っている構造体を関数に渡す必要があることを指定します。 表 44 は、DBINFO 構造体の説明を示しています。 DBINFO 構造体についての詳しい情報は、QSYSINC.H 内の組み込みファイル SQLUDF に入っています。

DBINFO は、PARAMETER STYLE DB2SQL または PARAMETER STYLE DB2GENERAL でのみ許可されます。

表 44. DBINFO フィールド

フィールド	データ・タイプ	説明
リレーショナル・データベース	VARCHAR(128)	現行サーバーの名前
権限 ID	VARCHAR(128)	実行時権限 ID

表 44. DBINFO フィールド (続き)

フィールド	データ・タイプ	説明
CCSID 情報	INTEGER INTEGER INTEGER INTEGER CHAR(8)	ジョブの CCSID 情報。CCSID を識別する情報は、次のとおりです。 • SBCS CCSID • DBCS CCSID • 混合 CCSID • 最初の 3 つの CCSID のどれに該当するかの指示。 • 予約済み CREATE FUNCTION ステートメントのパラメーターの 1 つとして明示的に CCSID を指定していない場合は、入力ストリングは、関数の実行時にジョブの CCSID でコード化されるものと見なされます。入力ストリングの CCSID がパラメーターの CCSID と同じではない場合は、この外部関数に渡される入力ストリングは、外部プログラムの呼び出しの前に変換されます。
ターゲット列	VARCHAR(128) VARCHAR(128) VARCHAR(128)	ユーザー定義の関数が UPDATE ステートメントの SET 文節の右側に指定されている場合、次の情報がターゲット列を識別します。 • スキーマ名 • 基礎表名 • 列名 ユーザー定義の関数が UPDATE ステートメントの SET 文節の右側に指定されなかった場合、これらのフィールドはブランクになります。
バージョンとリリース	CHAR(8)	データベース・マネージャーのバージョン、リリース、および修正レベル。
プラットフォーム	INTEGER	サーバーのプラットフォーム・タイプ。

NO DBINFO
これを指定すると、関数では、渡されるデータベース情報を必要としなくなります。

STATIC DISPATCH
関数を静的にディスパッチすることを指定します。すべての関数が静的にディスパッチされます。

EXTERNAL NAME *外部プログラム名*
SQL ステートメント内でこの関数が呼び出されたときに実行するプログラム、サービス・プログラム、または Java クラスを指定します。この名前は、関数が呼び出される時点でサーバー上に存在しているプログラム、サービス・プログラム、または Java クラスを示すものでなければなりません。命名オプションが *SYS であり、名前が修飾されていなければ、関数の呼び出し時に現行パスを使用して該当のプログラムやサービス・プログラムを検索します。

この名前の妥当性は、サーバーで検査されます。名前の形式が正しくない場合、エラーが戻されます。

この外部プログラム名の指定がなければ、その外部プログラム名は、関数名と同じであると想定されます。

このプログラム、サービス・プログラム、または Java クラスは、関数の作成時に存在している必要はありませんが、関数の呼び出し時には存在している必要があります。

CONNECT、 SET CONNECTION、 RELEASE、 DISCONNECT、 COMMIT、 ROLLBACK および SET TRANSACTION ステートメントは、関数の外部プログラム内で使用することはできません。

PARAMETER STYLE

関数にパラメーターを渡し、関数から値を戻すために使用する規則を指定します。

SQL

適用可能なパラメーターはすべて渡されます。これらのパラメーターは、次の順序で配列されるように定義されます。

- 最初の N 個のパラメーターは、CREATE FUNCTION ステートメント上に指定される入力パラメーターです。
- 関数の結果を表すパラメーター。
- 入力パラメーターの標識変数を表す N 個のパラメーター。
- 結果の標識変数を表すパラメーター。
- SQLSTATE の CHAR(5) 出力パラメーター。戻される SQLSTATE は、関数が成功したかどうかを示します。戻される SQLSTATE は、外部プログラムによって割り当てられた SQLSTATE です。

  ユーザーは、関数からエラーまたは警告を戻すために、外部プログラム内で SQLSTATE を任意の有効な値にセットすることができます。
- 完全修飾関数名の VARCHAR(517) 入力パラメーター。
- 特定の名前の VARCHAR(128) 入力パラメーター。
- メッセージ・テキストの VARCHAR(70) 出力パラメーター。

  制御が呼び出し側プログラムに戻された場合、SQLCA の SQLERRMC フィールドの 6 番目のトークンにメッセージ・テキストが入っています。入手できるのは、メッセージ・テキストの一部分だけです。SQLERRMC 内のメッセージ・データのレイアウトについては、メッセージ・ファイル QSQLMSG 内のメッセージ SQL0443 の置換データ記述を参照してください。

渡されるパラメーターについての詳細は、該当するソース・ファイル内の組み込み sqludf を参照してください。例えば、C の場合、sqludf は QSYSINC/H で見つかります。

DB2GENERAL

このパラメーター・スタイルは、Java クラスでメソッドとして定義されている外部関数にパラメーターを渡し、外部関数から値を戻すための変換を指定するのに使用します。適用可能なパラメーターはすべて渡されます。これらのパラメーターは、次の順序で配列されるように定義されます。

- 最初の N 個のパラメーターは、CREATE FUNCTION ステートメント上に指定される入力パラメーターです。
- 関数の結果を表すパラメーター。

DB2GENERAL は、LANGUAGE が JAVA の場合にのみ許されます。

GENERAL

適用可能なパラメーターはすべて渡されます。これらのパラメーターは、次の順序で配列されるように定義されます。

- 最初の N 個のパラメーターは、CREATE FUNCTION ステートメント上に指定される入力パラメーターです。

結果は、関数を戻す値の値として戻されることに注意してください。次の例を見てください。

```
return_val func(parameter-1, parameter-2, ...)
```

GENERAL は、EXTERNAL NAME でサービス・プログラムを識別する場合にのみ許可されます。

GENERAL WITH NULLS

適用可能なパラメーターはすべて渡されます。これらのパラメーターは、次の順序で配列されるように定義されます。

- 最初の N 個のパラメーターは、CREATE FUNCTION ステートメント上に指定される入力パラメーターです。
- 標識変数配列に追加の引き数が渡されます。
- 結果の標識変数を表すパラメーター。

結果は、関数を戻す値の値として戻されることに注意してください。次の例を見てください。

```
return_val func(parameter-1, parameter-2, ...)
```

GENERAL WITH NULLS は、EXTERNAL NAME でサービス・プログラムを識別する場合にのみ許可されます。

JAVA

このパラメーター・スタイルは、Java 言語および SQLJ ルーチン仕様に準拠するパラメーターの引き渡し規則を指定します。適用可能なパラメーターはすべて渡されます。これらのパラメーターは、次の順序で配列されるように定義されます。

- 最初の N 個のパラメーターは、CREATE FUNCTION ステートメント上に指定される入力パラメーターです。

結果は、関数を戻す値の値として戻されることに注意してください。次の例を見てください。

```
return_val func(parameter-1, parameter-2, ...)
```

JAVA は、LANGUAGE が JAVA の場合にのみ許されます。

DB2SQL

適用可能なパラメーターはすべて渡されます。これらのパラメーターは、次の順序で配列されるように定義されます。

- 最初の N 個のパラメーターは、CREATE FUNCTION ステートメント上に指定される入力パラメーターです。
- 関数の結果を表すパラメーター。
- 入力パラメーターの標識変数を表す N 個のパラメーター。
- 結果の標識変数を表すパラメーター。

- SQLSTATE の CHAR(5) 出力パラメーター。戻される SQLSTATE は、関数が成功したかどうかを示します。戻される SQLSTATE は、以下のいずれかです。
  - 外部プログラムで実行された最後の SQL ステートメントからの SQLSTATE
  - 外部プログラムによって割り当てられた SQLSTATE

    ユーザーは、関数からエラーまたは警告を戻すために、外部プログラム内で SQLSTATE を任意の有効な値にセットすることができます。
- 完全修飾関数名の VARCHAR(517) 入力パラメーター。
- 特定の名前の VARCHAR(128) 入力パラメーター。
- メッセージ・テキストの VARCHAR(70) 出力パラメーター。
- 0 ～ 3 個のオプション・パラメーター
  - CREATE FUNCTION ステートメントで SCRATCH PAD を指定した場合、スクラッチパッドの VARCHAR(n) 入出力パラメーター。
  - CREATE FUNCTION ステートメントで FINAL CALL を指定した場合、呼び出しタイプの INTEGER 入力パラメーター。
  - CREATE FUNCTION ステートメントで DBINFO を指定した場合、dbinfo 構造体の構造。

渡されるパラメーターについての詳細は、該当するソース・ファイル内の組み込み sqludf を参照してください。例えば、C の場合、sqludf は QSYSINC/H で見つかります。

パラメーターを渡す方法は、外部関数の言語によって決まります。たとえば、C では、VARCHAR または CHAR パラメーターはヌル文字で終了するストリングとして渡されます。詳細については、SQL プログラミング 概念を参照してください。

使用上の注意

関数の作成

ILE 外部プログラムまたはサービス・プログラムに関連した外部関数が作成されると、その関数に関連したプログラムやサービス・プログラムのオブジェクトへの関数属性の保管が試行されます。*PGM オブジェクトや *SRVPGM オブジェクトが保管された上でこのシステムや別のシステムに復元されると、カタログは、それらの属性を使用して自動的に更新されます。

外部関数の場合は、次の制約の範囲内で属性を保管することができます。

- 外部プログラム・ライブラリーは、SYSIBM、QSYS、または QSYS2 であってはなりません。
- 外部プログラムは、CREATE FUNCTION ステートメントの発行時に存在していなければなりません。
- 外部プログラムは、ILE *PGM オブジェクトか *SRVPGM オブジェクトにする必要があります。
- 外部プログラムやサービス・プログラムには、少なくとも 1 つの SQL ステートメントを含める必要があります。

オブジェクトを更新できない場合でも、関数は作成されます。

関数の復元時には、次のような動作が生じます。

- 関数が初めに作成される時点で特定名が指定されており、しかもその名前が固有でない場合は、エラーが出されます。
- 特定名が指定されていない場合は、必要に応じて固有名が生成されます。
- シグニチャーが固有でない場合は、関数を登録することは不可能で、エラーが出されます。

関数の呼び出し

外部関数が呼び出されると、その関数は、外部プログラムやサービス・プログラムの作成時に指定された活動化グループであれば、どの活動化グループ内でも実行します。ただし、通常は、関数が呼び出しプログラムと同じ活動化グループ内で実行するように ACTGRP(*CALLER) を使用します。 ACTGRP(*NEW) は使用できません。

Java 関数の使用上の注意

Java 関数を実行するためには、システムに Developer Kit for Java (5722-JV1) をインストールしておく必要があります。インストールされていないと、SQLCODE -443 が戻され、CPDB521 メッセージがジョブ・ログに入ります。

Java プロシージャーの実行中にエラーが発生すると、SQLCODE -443 が戻されます。 エラーによっては、プロシージャーが実行されていたジョブのジョブ・ログに他のメッセージが入っている場合があります。

同義のキーワード

以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- キーワード VARIANT と NOT VARIANT は、NOT DETERMINISTIC と DETERMINISTIC の同義語として使用することができます。
- キーワード NULL CALL と NOT NULL CALL は、CALLED ON NULL INPUT と RETURNS NULL ON NULL INPUT の同義語として使用できます。
- キーワード SIMPLE CALL は、GENERAL の同義語として使用できます。
- DB2GENERAL の同義語として、値 DB2GENRL を使用できます。

例 1

次の論理をインプリメントする外部関数サービス・プログラムを C で書き込むと想定します。

```
output = 2 * input - 4
```

入力引き数の 1 つがヌルである場合にのみ、関数はヌル値を戻す必要があります。関数呼び出しを回避し、入力値がヌルの場合にヌルの結果を得るための最も簡単な方法は、CREATE FUNCTION ステートメント上に RETURNS NULL ON NULL INPUT と指定することです。ただし、次の例には、入力パラメーターがヌルである場合にヌルを戻すためのコードが組み込まれています。特定名 MINENULL1 を使用して、関数の作成に必要なステートメントを書き込みます。

```
CREATE FUNCTION NTEST1 (INTEGER)
  RETURNS INTEGER
  EXTERNAL NAME 'MYLIB/NTESTMOD(nudft1)'
  SPECIFIC MINENULL1
  LANGUAGE C
  DETERMINISTIC
  NO SQL
  PARAMETER STYLE DB2SQL
  CALLED ON NULL INPUT
  NO EXTERNAL ACTION
```

プログラム・コードは次のとおりです。

```
  void nudft1
    (int *input,              /* ptr to input arg             */
     int *output,             /* ptr to output arg            */
     short *input_ind,          /* ptr to input indicator       */
     short *output_ind,         /* ptr to output indicator      */
     char sqlstate[6], /* sqlstate                      */
     char fname[140],  /* fully qualified function name */
     char finst[129],  /* function specific name       */
     char msgtext[71]) /* msg text buffer              */
  {
    if (*input_ind == -1)
      *output_ind = -1;
    else
  {
      *output = 2*(*input)-4;
      *output_ind = 0;
    }
    return;
  }
```

例 2

ユーザー McBride (管理権限が与えられているユーザー) が SMITH スキーマ内に CENTER という名前の外部関数を作成すると想定します。 McBride は、この関数に特定名 FOCUS98 を指定しようとしています。関数プログラムでは、ある種の初期設定を実行し、結果を保管するためにスクラッチパッドを使用します。関数プログラムでは、FLOAT データ・タイプが指定された値を戻します。関数の作成に McBride が必要としているステートメントを書き込み、関数の呼び出し時に、DECIMAL(8,4) というデータ・タイプが指定された値が戻されるようにします。

```
CREATE FUNCTION SMITH.CENTER (FLOAT, FLOAT, FLOAT)
  RETURNS DECIMAL(8,4) CAST FROM FLOAT
  EXTERNAL NAME CMOD
  SPECIFIC FOCUS98
  LANGUAGE C
  DETERMINISTIC
  NO SQL
  PARAMETER STYLE DB2SQL
  RETURNS NULL ON NULL INPUT
  NO EXTERNAL ACTION
  SCRATCHPAD
  NO FINAL CALL
```

CREATE FUNCTION (外部表)

CREATE FUNCTION (外部表) ステートメントは、現行サーバー上に外部表関数を作成します。この外部表関数は、結果表を戻します。

表関数 を SELECT の FROM 文節の中で使用することにより、SELECT に表を戻すことができます (一度に 1 行ずつ戻されます)。

呼び出し

このステートメントは、アプリケーション・プログラムの中に組み込んだり、あるいは、対話式に出すことができます。このステートメントは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも次のいずれか 1 つを含める必要があります。

- SYSFUNCS カタログ・ビューと SYSPARMS カタログ表の場合
  - 該当の表に対する INSERT 特権、および
  - QSYS2 ライブラリーに対する *EXECUTE システム権限
- 管理権限

ステートメントの権限 ID は、次の場合に表についての INSERT 特権を持ちます。

- その表の所有者である。
- その表についての INSERT 特権を認可されている、または
- その表についての *OBJOPR および *ADD システム権限が認可されている。

外部プログラムやサービス・プログラムが存在している場合、このステートメントの権限 ID が保持する特権には、少なくとも次のいずれか 1 つを含める必要があります。

- SQL ステートメントで参照された外部プログラムやサービス・プログラムの場合
  - その外部プログラムやサービス・プログラムが入っているライブラリーに対するシステム権限の *EXECUTE。
  - その外部プログラムやサービス・プログラムに対するシステム権限の *EXECUTE。
  - そのプログラムやサービス・プログラムに対するシステム権限の *CHANGE。システムには、プログラム・オブジェクトを更新し、関数を別のシステムに保管/復元するために必要な情報を入れる場合にこの権限が必要となります。ユーザーにこの権限が与えられていない場合、関数は同じように作成されますが、プログラム・オブジェクトは更新されません。
- 管理権限

SQL 名が指定され、関数が作成されるライブラリーと同じ名前のユーザー・プロファイルが存在し、しかも、その名前がステートメントの権限 ID と異なっている場合、ステートメントの権限 ID が保持している特権には、少なくとも次のいずれか 1 つを含める必要があります。

- その名前を持つユーザー・プロファイルに対する *ADD システム権限

- 管理権限

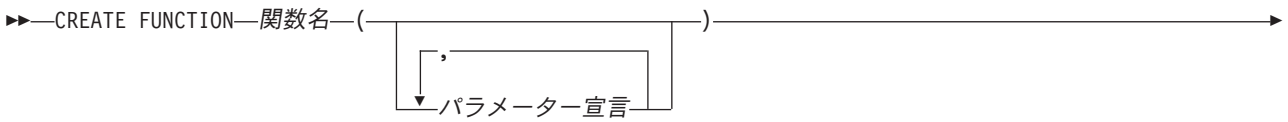
特殊タイプを参照する場合は、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- ステートメント内に識別されているそれぞれの 特殊タイプ に対しては次のもの。
  - その 特殊タイプ に対する USAGE 特権。および
  - その 特殊タイプ が入っているライブラリーに対する *EXECUTE システム権限
- 管理権限

次のいずれかに該当する場合、ステートメントの権限 ID には、特殊タイプ に対する USAGE 特権が与えられます。

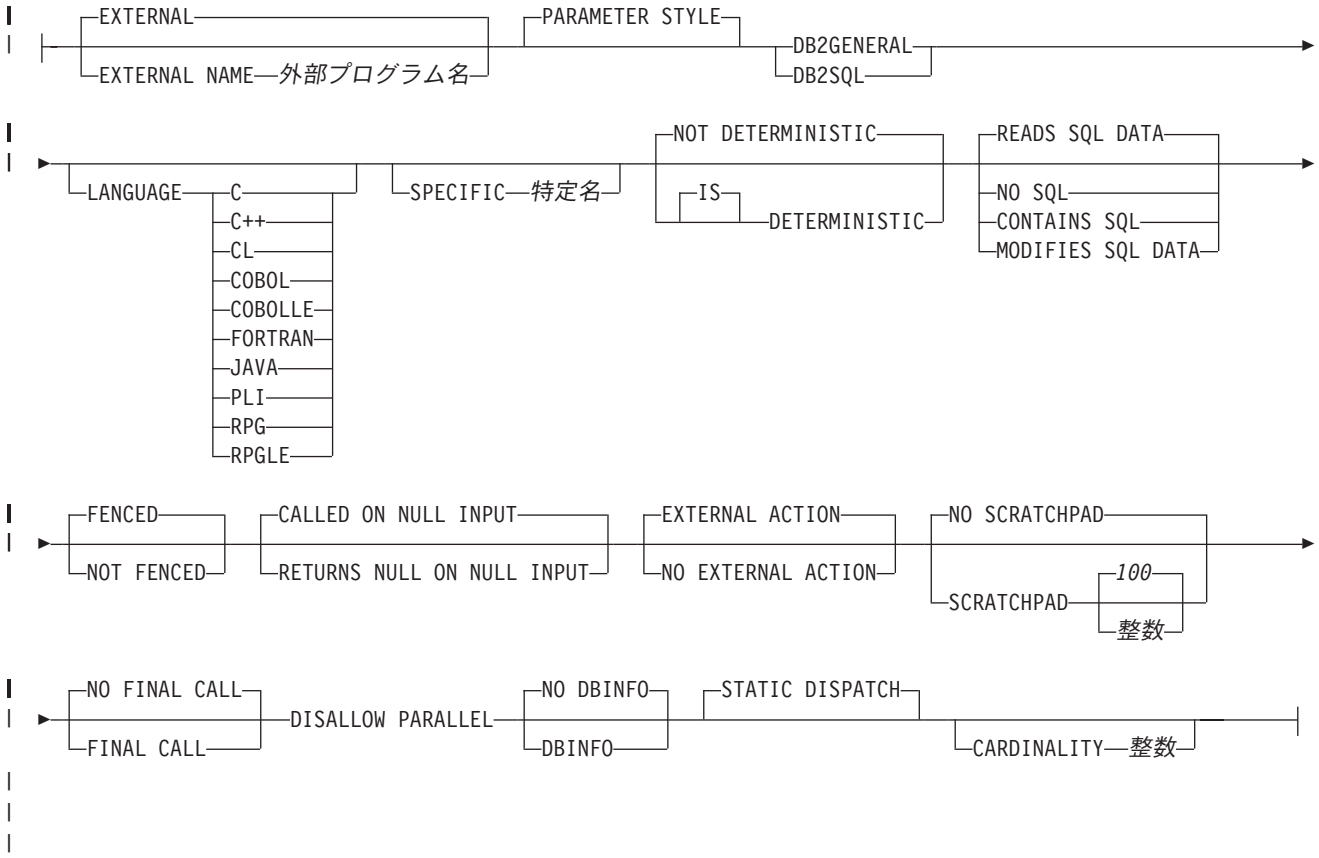
- その 特殊タイプ の所有者である。
- その 特殊タイプ に対する USAGE 特権が付与されている。
- その 特殊タイプ に対するシステム権限の *OBJOPR と *EXECUTE が付与されている。

構文

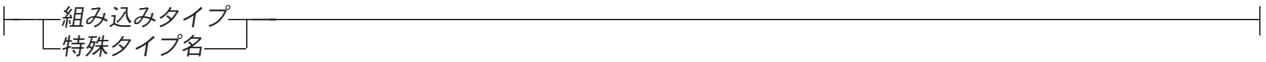


パラメーター宣言:

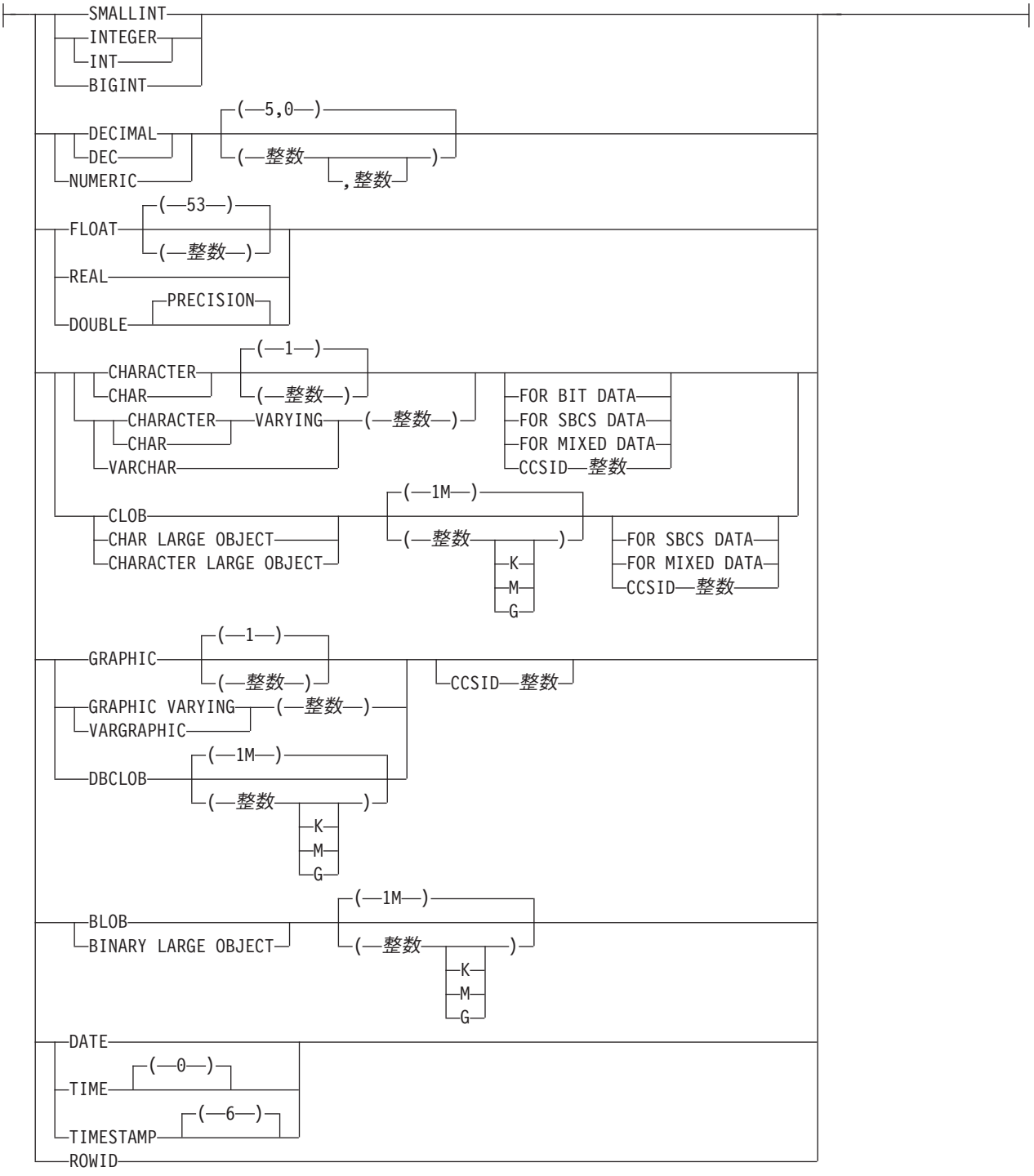
オプション・リスト:


EXTERNAL
EXTERNAL NAME—外部プログラム名
PARAMETER STYLE
DB2GENERAL
DB2SQL
LANGUAGE
C
C++
CL
COBOL
COBOLLE
FORTRAN
JAVA
PLI
RPG
RPGLE
SPECIFIC—特定名
NOT DETERMINISTIC
IS
DETERMINISTIC
READS SQL DATA
NO SQL
CONTAINS SQL
MODIFIES SQL DATA
FENCED
NOT FENCED
CALLED ON NULL INPUT
RETURNS NULL ON NULL INPUT
EXTERNAL ACTION
NO EXTERNAL ACTION
NO SCRATCHPAD
SCRATCHPAD
100
整数
NO FINAL CALL
FINAL CALL
DISALLOW PARALLEL
NO DBINFO
DBINFO
STATIC DISPATCH
CARDINALITY—整数

データ・タイプ:



組み込みタイプ:



説明

関数名
ユーザー定義の関数の名前を指定します。名前、スキーマ名、パラメーターの数、およびそれぞれのパラメーターのデータ・タイプ (データ・タイプの長さ、精度、位取り、または CCSID の属性に関係なく) の組み合わせで、現行サーバー上に存在しているユーザー定義の関数を識別してはなりません。

SQL 命名の場合、関数は、暗黙または明示修飾子で指定されたスキーマ内に作成されます。

システム命名の場合、関数は、修飾子で指定されたスキーマ内に作成されます。修飾子を指定しない場合、関数は、現行ライブラリー (*CURLIB) 内に作成されます。現行ライブラリーがない場合、関数は QGPL 内に作成されます。

通常、それぞれの関数の関数シグニチャーが固有であれば、複数の関数に同じ名前を指定することができます。

一部の関数名は、システムが使用するために予約されています。 詳細については、446 ページの『関数名の選択』を参照してください。

(パラメーター宣言,...)
関数のパラメーターの数とそれぞれのパラメーターのデータ・タイプを指定します。 それぞれのパラメーターに名前を指定することができますが、これは必須ではありません。

CREATE FUNCTION (外部表) で使用できるパラメーターの最大数は 90 です。さらに、パラメーターの最大数は、外部プログラムのコンパイルに使用されるライセンス・プログラムで許容されているパラメーターの最大数によっても制約されます。

パラメーター名
入力パラメーターの名前を指定します。同じ名前を何度も指定することはできません。

データ・タイプ 1
関数の入力パラメーターの数とそれぞれの入力パラメーターのデータ・タイプを指定します。関数のパラメーターはすべて入力パラメーターです。関数が受信を予期しているそれぞれのパラメーターについて、リスト内に記入項目を 1 つ設ける必要があります。それぞれのパラメーターに名前を指定することができますが、これは必須ではありません。

PARAMETER STYLE JAVA を指定する場合は、ラージ・オブジェクト (LOB) データ・タイプのパラメーターはサポートされません。

関数にはパラメーターを指定しなくても構いません。この場合は、次のように、中が空の 1 組の括弧をコーディングする必要があります。

```
CREATE FUNCTION WOOFER()
```

CCSID が指定されている場合、関数に渡される前に、パラメーターはその CCSID に変換されます。CCSID が指定されていない場合は、CCSID は、関数の呼び出し時点における現行サーバーのデフォルトの CCSID によって決まります。

AS LOCATOR
これを指定すると、入力パラメーターは、実際の値ではなく、値のロケータ

ーになります。 AS LOCATOR は、入力パラメーターに LOB データ・タイプや LOB データ・タイプをベースとする特殊タイプが指定されている場合にのみ、指定することができます。

RETURNS TABLE
関数の出力表を指定します。

パラメーターの数が N であるとすると、PARAMETER STYLE DB2GENERAL の場合、列の数は (255-(N*2))/2 以下でなければなりません。 PARAMETER STYLE DB2SQL の場合は、列の数は (247-(N*2))/2 以下でなければなりません。

列名
出力表の列の名前を指定します。同じ名前を何度も指定することはできません。

データ・タイプ 2
出力のデータ・タイプと属性を指定します。

あらゆる組み込みデータ・タイプ (ただし、LONG VARCHAR、LONG VARGRAPHIC、または DataLink は除く) や特殊タイプ (データ・リンクをベースとしていない) を指定することができます。

DATE または TIME を指定した場合は、表関数は ISO 形式の日付または時刻を戻す必要があります。

CCSID が指定されている場合

- AS LOCATOR が指定されていない場合、戻される結果はその CCSID でコード化されていると想定されます。
- AS LOCATOR が指定され、ロケーターが指しているデータの CCSID が異なる CCSID でコード化されている場合、データは指定された CCSID に変換されます。

CCSID が指定されていない場合

- AS LOCATOR が指定されていない場合、戻される結果は、ジョブの CCSID (グラフィック・ストリング戻り値の場合は、ジョブに関連したグラフィック CCSID) でコード化されていると想定されます。
- AS LOCATOR が指定され、ロケーターが指しているデータの CCSID が異なる CCSID でコード化されている場合、ロケーターが指しているデータは、ジョブの CCSID に変換されます。変換時に文字が失われるのを防ぐために、関数から戻される文字をすべて表現できる CCSID を明示的に指定することを考慮してください。 これは、データ・タイプがグラフィック・ストリング・データの場合に特に重要です。 この場合、CCSID 13488 (UCS-2 グラフィック・ストリング・データ) を使用することを考慮してください。

AS LOCATOR
これを指定すると、関数は、実際の値ではなく、該当の列の値に対するロケーターを戻します。 AS LOCATOR を指定できるのは、LOB データ・タイプか、LOB データ・タイプに基づく特殊タイプの場合だけです。

SPECIFIC *特定名*
関数の固有名を指定します。この名前は、暗黙的または明示的にスキーマ名で修

飾されます。 スキーマ名も含め、この名前は、現行サーバーに存在している別の関数またはプロシージャーの特定名を示すものであってはなりません。 修飾されない場合の暗黙の修飾子は、関数名の修飾子と同じです。修飾される場合の修飾子は、関数名の修飾子と同じものにする必要があります。

特定名を指定しなかった場合、その特定名は、関数名に設定されます。この特定名の関数やプロシージャーがすでに存在している場合は、固有表名の生成に使用される規則にほぼ準拠した固有名が生成されます。

LANGUAGE (言語文節)

言語文節は、外部プログラムの言語を指定します。

LANGUAGE を指定しなかった場合、その LANGUAGE は、関数の作成時に、外部プログラムと関連したプログラム属性情報から決定されます。次の場合、プログラムの言語は C であると想定されます。

- プログラムに関連したプログラム属性情報で、認識可能な言語を識別しない。
- プログラムが見つからない。

C
外部プログラムは C で作成されます。

C++
外部プログラムは C++ で作成されます。

CL
外部プログラムは CL または ILE CL で作成されます。

COBOL
外部プログラムは COBOL で作成されます。

COBOLLE
外部プログラムは ILE COBOL で作成されます。

FORTRAN
外部プログラムは FORTRAN で作成されます。

JAVA
外部プログラムは JAVA で作成されます。データベース・マネージャーは、ユーザー定義関数を、Java クラス内のメソッドとして呼び出します。

PLI
外部プログラムは PL/I で作成されます。

RPG
外部プログラムは RPG で作成されます。

RPGLE
外部プログラムは ILE RPG で作成されます。

DETERMINISTIC または **NOT DETERMINISTIC**

関数が決定的であるか否かを指定します。

NOT DETERMINISTIC
これを指定すると、関数は、必ずしも、同一の入力引き数が指定された連続関数呼び出しから同じ結果を戻すとは限らなくなります。 NOT

DETERMINISTIC は、特殊レジスターまたは非決定的関数に対する参照がこの関数に含まれている場合に指定してください。

DETERMINISTIC
これを指定すると、関数は、必ず、同一の入力引き数が指定された連続呼び出しから同じ結果を戻します。

CONTAINS SQL、READS SQL DATA、MODIFIES SQL DATA、または NO SQL
関数がなんらかの SQL ステートメントを実行できるかどうか、および実行できる場合にどのようなタイプのステートメントを実行できるかを指定します。データベース・マネージャーは、この関数が発行する SQLがこの条件を満たしているかどうかを検査します。各データ・アクセス指示の下で実行できる SQL ステートメントの詳細なリストについては、913 ページの『付録 F. SQL ステートメントの特性』を参照してください。

CONTAINS SQL
関数は、データを読み取りまたは変更する SQL ステートメントを実行しません。

NO SQL
関数は SQL ステートメントを実行しません。

READS SQL DATA
関数は、データを変更する SQL ステートメントを実行しません。

MODIFIES SQL DATA
関数は、どの関数でもサポートされないステートメントを除くすべての SQL ステートメントを実行できます。

FENCED または NOT FENCED
この関数を、呼び出し元の SQL ステートメントと同じスレッド内で実行するか、別のスレッドで実行するかを指定します。

FENCED
関数は別のスレッドで実行されます。同じ SQL ステートメント内で同じ関数を複数回呼び出すと、互いに競合することがあるので、関数に SQL カーソルが含まれている場合は、FENCED を選択するのが最も安全です。

NOT FENCED
関数は、呼び出し元の SQL ステートメントと同じスレッド内で実行できます。 NOT FENCED を指定すると、この関数の呼び出し間でカーソルをオープン状態のままにすることができます。カーソルをオープン状態のままにしておくことができるため、関数の呼び出し間でカーソル位置も同じに維持されます。

NULL INPUT
入力パラメーターが NULL である場合に、関数を呼び出す必要があるか否かを指定します。

CALLED ON NULL INPUT
常に関数を呼び出します。

RETURNS NULL ON NULL INPUT
表関数のオープン時に、その関数の引き数のどれかがヌルである場合は、そのユーザー定義の表関数は呼び出されず、関数出力は空の表 (行のない表) になります。

EXTERNAL ACTION または **NO EXTERNAL ACTION**
関数に外部アクションが含まれているかどうかを指定します。

EXTERNAL ACTION
関数は、なんらかの外部アクション (関数プログラムの有効範囲外のアクション) を行います。したがって、関数は、それぞれの連続関数呼び出しで呼び出す必要があります。 EXTERNAL ACTION は、この関数に、外部アクションを持つ他の関数に対する参照が含まれている場合に、指定してください。

NO EXTERNAL ACTION
関数は外部アクションを行いません。この関数は、連続した各関数呼び出しごとに呼び出す必要はありません。

SCRATCHPAD
関数に、静的メモリー域が必要か否かを指定します。

SCRATCHPAD *整数*
これを指定すると、関数には、持続メモリー域の長さ整数が必要となります。 この整数に指定できる範囲は、1 ～ 16,000,000 です。メモリー域を指定しなかった場合、その区域のサイズは 100 バイトになります。パラメーター・スタイル DB2SQL を指定すると、ポインターは、静的記憶域を指す必須パラメーターに続いて渡されます。この関数には、メモリー域が 1 つだけ割り振られます。

スクラッチパッドの有効範囲は SQL です。SQL ステートメント内の関数の参照ごとに、1 つのスクラッチパッドが存在します。例えば、関数 UDFX が SCRATCHPAD キーワードによって定義されていると想定した場合、次の SQL ステートメント内では、UDFX の 2 つの参照に対して 2 つのスクラッチパッドが割り振られます。

```
SELECT A.C1, B.C1
  FROM TABLE(UDFX(:hv1)) AS A, TABLE(UDFX(:hv1)) AS B
```

NO SCRATCHPAD
これを指定すると、関数では、持続メモリー域を必要としなくなります。

FINAL CALL
関数が、最終呼び出し (および独立した初回呼び出し) を必要とするかどうかを指定します。表関数の場合は、FINAL CALL オプションを選択するかどうかに関係なく、呼び出しタイプ引き数が常に存在します。呼び出しタイプ引き数は、初回呼び出し、オープン呼び出し、フェッチ呼び出し、クローズ呼び出し、または最終呼び出しのいずれであるかを示します。

FINAL CALL
関数が、最終呼び出し (および独立した初回呼び出し) を必要とすることを指定します。これは、スクラッチパッドをどの時点で再初期化するかも制御します。 NO FINAL CALL を指定した場合は、表関数に対してデータベース・マネージャーが行うことのできる呼び出しのタイプは、オープン呼び出し、フェッチ呼び出し、およびクローズ呼び出しの 3 つだけです。これに

対して、FINAL CALL を指定した場合は、オープン、フェッチ、およびクローズに加えて、表関数に対する初回呼び出しと最終呼び出しも行うことができます。これは、関数に対する最終呼び出しを行うことを指定します。この関数は、最終呼び出しとその他の呼び出しを区別するために、呼び出しのタイプを示す追加の引き数を受け取ります。

呼び出しには以下のタイプがあります。

First Call (初回呼び出し)
この SQL ステートメントでのこの関数に対するこの参照についての、この関数に対する初回呼び出しを示します。

Open Call (オープン呼び出し)
この SQL での表関数結果をオープンするための呼び出しを指定します。

Fetch Call (フェッチ呼び出し)
この SQL ステートメントで表関数から行をフェッチするための呼び出しを指定します。

Close Call (クローズ呼び出し)
この SQL での表関数結果をクローズするための呼び出しを指定します。

Final Call (最終呼び出し)
この関数に対する最後の呼び出しであり、これにより関数はリソースを解放できることを指定します。エラーが起きた場合は、データベース・マネージャーは最終呼び出しを行おうとします。

最終呼び出しが発生するのは以下の場合です。

- *ステートメントの終わり :* カーソル指向ステートメント用のカーソルがクローズされたとき、またはステートメントの実行が完了したとき。
- *トランザクションの終わり :* ステートメント処理の正常終了が発生しなかったとき。例えば、何らかの理由によりアプリケーションのロジックがカーソルのクローズをバイパスした場合。

WITH HOLD として定義されているカーソルがオープン状態にあるときにコミット操作が発生した場合は、カーソルがクローズされるかアプリケーションが終了した時点で、最終呼び出しが行われます。

FINAL CALL ではコミット可能な操作は行ってはなりません。FINAL CALL は、COMMIT 操作の一部として呼び出されたクローズ中に実行される可能性があるからです。

NO FINAL CALL
関数が、最終呼び出し (および独立した初回呼び出し) を必要としないことを指定します。ただし、オープン呼び出し、フェッチ呼び出し、およびクローズ呼び出しは行われます。

DISALLOW PARALLEL
これを指定すると、関数は並列で実行できなくなります。 表関数は並列では実行できません。

DBINFO

関数にデータベース情報を渡す必要があるかどうかを指定します。

DBINFO

データベース・マネージャーは、状況情報が入っている構造体を関数に渡す必要があることを指定します。 表 45 は、DBINFO 構造体の説明を示しています。 DBINFO 構造体についての詳しい情報は、QSYSINC.H 内の組み込みファイル SQLUDF に入っています。

表 45. DBINFO フィールド

フィールド	データ・タイプ	説明
リレーショナル・データベース	VARCHAR(128)	現行サーバーの名前
権限 ID	VARCHAR(128)	実行時権限 ID
CCSID 情報	INTEGER INTEGER INTEGER INTEGER CHAR(8)	ジョブの CCSID 情報。CCSID を識別する情報は、次のとおりです。 • SBCS CCSID • DBCS CCSID • 混合 CCSID • 最初の 3 つの CCSID のどれに該当するかの指示。 • 予約済み CREATE FUNCTION ステートメントのパラメーターの 1 つとして明示的に CCSID を指定していない場合は、入力ストリングは、関数の実行時にジョブの CCSID でコード化されるものと見なされます。入力ストリングの CCSID がパラメーターの CCSID と同じではない場合は、この外部関数に渡される入力ストリングは、外部プログラムの呼び出しの前に変換されます。
ターゲット列	VARCHAR(128) VARCHAR(128) VARCHAR(128)	ユーザー定義の関数が UPDATE ステートメントの SET 文節の右側に指定されている場合、次の情報がターゲット列を識別します。 • スキーマ名 • 基礎表名 • 列名 ユーザー定義の関数が UPDATE ステートメントの SET 文節の右側に指定されなかった場合、これらのフィールドはブランクになります。
バージョンとリリース	CHAR(8)	データベース・マネージャーのバージョン、リリース、および修正レベル。
プラットフォーム	INTEGER	サーバーのプラットフォーム・タイプ。
表関数の列リスト項目の数。	SMALLINT	下記の「表関数の列リスト」フィールドに指定されている表関数列リスト内のゼロでない項目の数。
予約済み	CHAR(24)	将来の利用のため予約済み。

表 45. DBINFO フィールド (続き)

フィールド	データ・タイプ	説明
表関数の列リスト	ポインター (16 バイト)	このフィールドは、データベース・マネージャーが動的に割り振る短精度整数の配列を指すポインターです。「表関数の列リスト項目の数」フィールドに n を指定した場合、最初の n 個の項目のみが対象になります。n は、0 またはそれより大きく、この関数の RETURNS TABLE 文節で定義した結果列の数と同じかまたはそれより小さい値です。個々の値は、このステートメントが表関数から取得する必要がある列の順序番号に対応しています。つまり、1 の値は最初に定義された結果列を意味し、2 は 2 番目に定義された結果列を意味します (以下同様)。これらの値はどのような順序になっていても構いません。 SELECT COUNT(*) FROM TABLE(TF(...)) AS QQ のように、実際の列値を必要としない照会を指定するステートメントの場合は、n はゼロになることがあるという点に注意してください。 この配列により最適化が可能になる場合があります。それは、この関数が、表関数のすべての結果列のすべての値を戻す必要がなくなるからです。特定のコンテキストでは一部の値しか必要とされないことがあり、配列内で番号により識別されている列がこれらの値に相当します。この最適化により関数ロジックが複雑になることもあるので、定義されているすべての列を戻すことを選択することもできます。

NO DBINFO
関数にデータベース情報を渡す必要がないことを指定します。

CARDINALITY *整数*
この任意選択の文節は、最適化を目的として、この関数が戻すものとして予期される行数の見積もりを指定します。整数の有効な値の範囲は、0 ～ 2 147 483 647 です。

表関数について CARDINALITY 文節を指定しなかった場合は、データベース・マネージャーは、デフォルトに基づき特定の有限値が想定されます。

重要 : ある関数が事実上無限のカーディナリティーを持つ場合、つまり、その関数が呼び出されるたびに行を戻し、永遠に表終わり条件を戻さない場合は、表終わり条件を必要とする照会も無限に実行され続けることになるため、割り込みが必要になります。例えば、GROUP BY や ORDER BY を含む照会などがこれに該当します。このような UDF は書かないようにしてください。

STATIC DISPATCH
関数を静的にディスパッチすることを指定します。すべての関数が静的にディスパッチされます。

EXTERNAL NAME *外部プログラム名*
SQL ステートメント内でこの関数が呼び出されたときに実行するプログラム、サービス・プログラム、または Java クラスを指定します。この名前は、関数が呼び出される時点でサーバー上に存在しているプログラム、サービス・プログラム、または Java クラスを示すものでなければなりません。命名オプションが *SYS であり、名前が修飾されていなければ、関数の呼び出し時に現行パスを使用して該当のプログラムやサービス・プログラムを検索します。

この名前の妥当性は、サーバーで検査されます。名前の形式が正しくない場合、エラーが戻されます。

この外部プログラム名の指定がなければ、その外部プログラム名は、関数名と同じであると想定されます。

このプログラム、サービス・プログラム、または Java クラスは、関数の作成時に存在している必要はありませんが、関数の呼び出し時には存在している必要があります。

CONNECT、 SET CONNECTION、 RELEASE、 DISCONNECT、 COMMIT、 ROLLBACK および SET TRANSACTION ステートメントは、関数の外部プログラム内で使用することはできません。

PARAMETER STYLE

関数にパラメーターを渡し、関数から値を戻すために使用する規則を指定します。

DB2GENERAL

このパラメーター・スタイルは、Java クラスでメソッドとして定義されている外部関数にパラメーターを渡し、外部関数から値を戻すための変換を指定するのに使用します。適用可能なパラメーターはすべて渡されます。これらのパラメーターは、次の順序で配列されるように定義されます。

- 最初の N 個のパラメーターは、CREATE FUNCTION ステートメント上に指定される入力パラメーターです。
- その後の M 個のパラメーターは、RETURNS TABLE 文節に指定されている、この関数の結果列です。

DB2GENERAL は、LANGUAGE が JAVA の場合にのみ許されます。

DB2SQL

適用可能なパラメーターはすべて渡されます。これらのパラメーターは、次の順序で配列されるように定義されます。

- 最初の N 個のパラメーターは、CREATE FUNCTION ステートメント上に指定される入力パラメーターです。
- その後の M 個のパラメーターは、RETURNS TABLE 文節に指定されている、この関数の結果列です。
- 入力パラメーターの標識変数を表す N 個のパラメーター。
- RETURNS TABLE 文節に指定されているこの関数の結果列の標識変数を表す M 個のパラメーター。
- SQLSTATE の CHAR(5) 出力パラメーター。戻される SQLSTATE は、関数が成功したかどうかを示します。戻される SQLSTATE は、以下のいずれかです。
  - 外部プログラムで実行された最後の SQL ステートメントからの SQLSTATE
  - 外部プログラムによって割り当てられた SQLSTATE

    ユーザーは、関数からエラーまたは警告を戻すために、外部プログラム内で SQLSTATE を任意の有効な値にセットすることができます。
- 完全修飾関数名の VARCHAR(517) 入力パラメーター。
- 特定の名前の VARCHAR(128) 入力パラメーター。
- メッセージ・テキストの VARCHAR(70) 出力パラメーター。

- CREATE FUNCTION ステートメントで SCRATCH PAD を指定した場合、スクラッチパッドの VARCHAR(n) 入出力パラメーター。
- 呼び出しタイプを示す INTEGER 入力パラメーター。
- CREATE FUNCTION ステートメントで DBINFO を指定した場合、dbinfo 構造体の構造。

渡されるパラメーターについての詳細は、該当するソース・ファイル内の組み込み sqludf を参照してください。例えば、C の場合、sqludf は QSYSINC/H で見つかります。

パラメーターを渡す方法は、外部関数の言語によって決まります。たとえば、C では、VARCHAR または CHAR パラメーターはヌル文字で終了するストリングとして渡されます。詳細については、SQL プログラミング 概念を参照してください。

使用上の注意

関数の作成

ILE 外部プログラムまたはサービス・プログラムに関連した外部関数が作成されると、その関数に関連したプログラムやサービス・プログラムのオブジェクトへの関数属性の保管が試行されます。*PGM オブジェクトや *SRVPGM オブジェクトが保管された上でこのシステムや別のシステムに復元されると、カタログは、それらの属性を使用して自動的に更新されます。

外部関数の場合は、次の制約の範囲内で属性を保管することができます。

- 外部プログラム・ライブラリーは、SYSIBM、QSYS、または QSYS2 であってはなりません。
- 外部プログラムは、CREATE FUNCTION ステートメントの発行時に存在していなければなりません。
- 外部プログラムは、ILE *PGM オブジェクトか *SRVPGM オブジェクトにする必要があります。
- 外部プログラムやサービス・プログラムには、少なくとも 1 つの SQL ステートメントを含める必要があります。

オブジェクトを更新できない場合でも、関数は作成されます。

関数の復元時には、次のような動作が生じます。

- 関数が初めに作成される時点で特定名が指定されており、しかもその名前が固有でない場合は、エラーが出されます。
- 特定名が指定されていない場合は、必要に応じて固有名が生成されます。
- シグニチャーが固有でない場合は、関数を登録することは不可能で、エラーが出されます。

関数の呼び出し

外部関数が呼び出されると、その関数は、外部プログラムやサービス・プログラムの作成時に指定された活動化グループであれば、どの活動化グループ内でも実行し

ます。ただし、通常は、関数が呼び出しプログラムと同じ活動化グループ内で実行するように ACTGRP(*CALLER) を使用します。 ACTGRP(*NEW) は使用できません。

Java 関数の使用上の注意

Java 関数を実行するためには、システムに Developer Kit for Java (5722-JV1) をインストールしておく必要があります。 インストールされていないと、SQLCODE -443 が戻され、CPDB521 メッセージがジョブ・ログに入ります。

Java プロシージャーの実行中にエラーが発生すると、SQLCODE -443 が戻されます。 エラーによっては、プロシージャーが実行されていたジョブのジョブ・ログに他のメッセージが入っている場合があります。

同義のキーワード

以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- キーワード VARIANT と NOT VARIANT は、NOT DETERMINISTIC と DETERMINISTIC の同義語として使用することができます。
- キーワード NULL CALL と NOT NULL CALL は、CALLED ON NULL INPUT と RETURNS NULL ON NULL INPUT の同義語として使用できます。
- DB2GENERAL の同義語として、値 DB2GENRL を使用できます。

例 1

以下の例で作成する表関数は、テキスト管理システム内にある既知の各文書を示す単一の文書 ID が入った行を 1 つずつ戻します。最初のパラメーターは特定のサブジェクト・エリアに対応し、2 番目のパラメーターには特定のストリングが入ります。

単一セッションのコンテキストでは、この UDF は常に同じ表を戻すので、DETERMINISTIC として定義されています。 RETURNS が DOCMATCH からの出力を定義している点に注意してください。各表関数について、FINAL CALL を指定する必要があります。さらに、表関数は並列では実行できないため、DISALLOW PARALLEL キーワードが追加されています。 DOCMATCH の場合の出力のサイズは大きく変化しますが、代表的な値は CARDINALITY 20 なので、最適化プログラムを支援するためにこの値が指定されています。

```
CREATE FUNCTION DOCMATCH (VARCHAR(30), VARCHAR(255))
  RETURNS TABLE (DOCID CHAR(16))
  EXTERNAL NAME 'MYLIB/RAJIV(UDFMATCH)'
  LANGUAGE C
  PARAMETER STYLE DB2SQL
  NO SQL
  DETERMINISTIC
  NO EXTERNAL ACTION
  NOT FENCED
  SCRATCHPAD
  FINAL CALL
  DISALLOW PARALLEL
  CARDINALITY 20
```

CREATE FUNCTION (ソース化)

この CREATE FUNCTION ステートメントは、現行サーバーで、他の既存のスカラー関数または列関数に基づいてユーザー定義の関数を作成するために使用します。

呼び出し

このステートメントは、アプリケーション・プログラムの中に組み込んだり、あるいは、対話式に出すことができます。このステートメントは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも次のいずれか 1 つを含める必要があります。

- SYSFUNCS カタログ・ビューと SYSPARMS カタログ表の場合
  - 該当の表に対する INSERT 特権、および
  - QSYS2 ライブラリーに対する *EXECUTE システム権限
- 管理権限

ステートメントの権限 ID は、次の場合に表についての INSERT 特権を持ちます。

- その表の所有者である。
- その表についての INSERT 特権を認可されている、または
- その表についての *OBJOPR および *ADD システム権限が認可されている。

ソース関数がユーザー定義の関数である場合は、そのソース関数に対して、このステートメントの権限 ID に、少なくとも次のいずれか 1 つを含める必要があります。

- その関数に対する EXECUTE 特権
- 管理権限

次の場合、ステートメントの権限 ID には、関数に対する EXECUTE 特権が与えられます。

- その関数の所有者である。
- その関数に対する EXECUTE 特権が認可されている、または
- その関数に対するシステム権限の *OBJOPR と *EXECUTE が認可されている。

ソース化関数を作成するには、ステートメントの権限 ID が保持する特権に、少なくとも次のいずれか 1 つを含める必要があります。

- 次のシステム権限
  - サービス・プログラム作成 (CRTSRVPGM) コマンドに対する *USE
  - プログラム作成 (CRTPGM) コマンドに対する *USE
  - 関数が作成されるライブラリーに対する *EXECUTE と *ADD
- 管理権限

SQL 名が指定され、関数が作成されるライブラリーと同じ名前のユーザー・プロファイルが存在し、しかも、その名前がステートメントの権限 ID と異なっている場合、ステートメントの権限 ID が保持している特権には、少なくとも次のいずれか 1 つを含める必要があります。

- その名前を持つユーザー・プロファイルに対する *ADD システム権限
- 管理権限

特殊タイプを参照する場合は、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

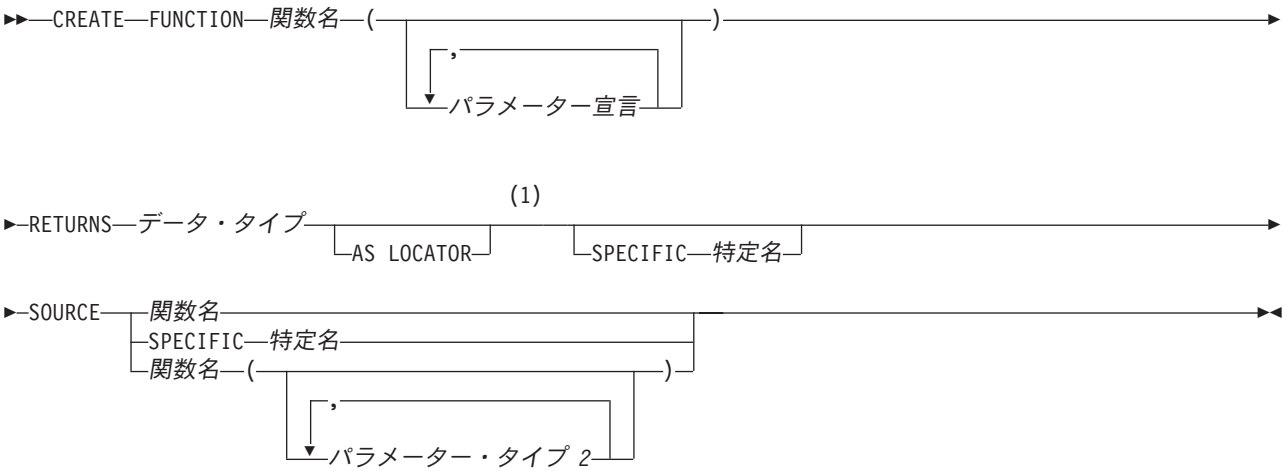
- ステートメント内に識別されているそれぞれの 特殊タイプ に対しては次のもの。
  - その 特殊タイプ に対する USAGE 特権。および
  - その 特殊タイプ が入っているライブラリーに対する *EXECUTE システム権限
- 管理権限

次のいずれかに該当する場合、ステートメントの権限 ID には、特殊タイプ に対する USAGE 特権が与えられます。

- その 特殊タイプ の所有者である。
- その 特殊タイプ に対する USAGE 特権が付与されている。
- その 特殊タイプ に対するシステム権限の *OBJOPR と *EXECUTE が付与されている。

CREATE FUNCTION (ソース化)

構文



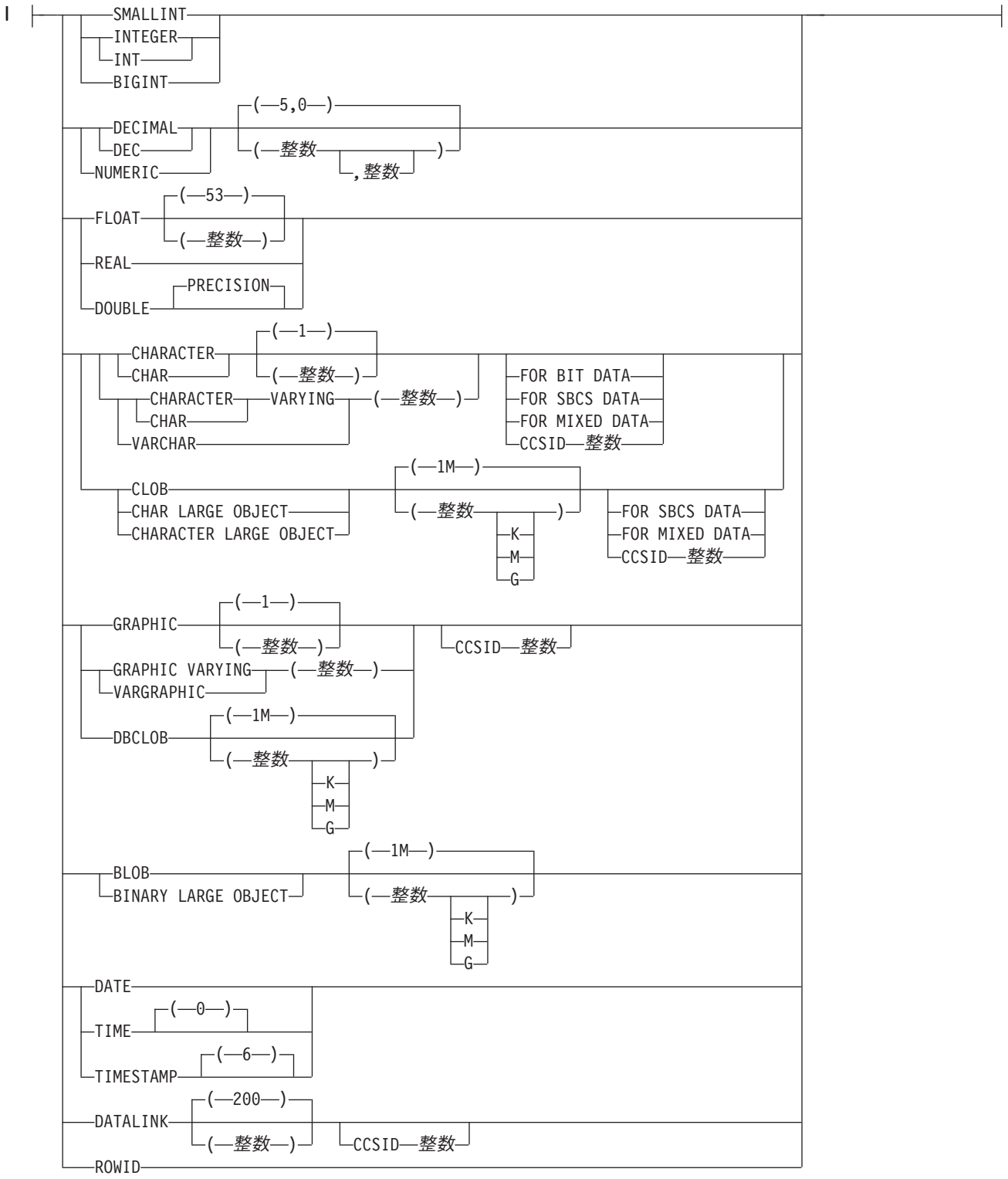
パラメーター宣言:

データ・タイプ:

注:

1 RETURNS、SPECIFIC、および SOURCE 文節は、どのような順序で指定しても構いません。

組み込みタイプ:



説明

関数名

ユーザー定義の関数の名前を指定します。名前、スキーマ名、パラメーターの数、およびそれぞれのパラメーターのデータ・タイプ (データ・タイプの長さ、

精度、位取り、または CCSID の属性に関係なく) の組み合わせで、現行サーバー上に存在しているユーザー定義の関数を識別してはなりません。

SQL 命名の場合、関数は、暗黙または明示修飾子で指定されたスキーマ内に作成されます。

システム命名の場合、関数は、修飾子で指定されたスキーマ内に作成されます。修飾子を指定しない場合、関数は、現行ライブラリー (*CURLIB) 内に作成されます。現行ライブラリーがない場合、関数は QGPL 内に作成されます。

特殊タイプを指定した既存関数の使用を可能にするために、関数がその既存関数をソースとして作成される場合、その名前は、その既存関数と同じ名前にすることができます。通常、それぞれの関数の関数シグニチャーが固有であれば、複数の関数に同じ名前を指定することができます。

一部の関数名は、システムが使用するために予約されています。 詳細については、446 ページの『関数名の選択』を参照してください。

(パラメーター宣言,...)

関数のパラメーターの数とそれぞれのパラメーターのデータ・タイプを指定します。 それぞれのパラメーターに名前を指定することができますが、これは必須ではありません。

パラメーター名

入力パラメーターの名前を指定します。同じ名前を何度も指定することはできません。

データ・タイプ 1

関数の入力パラメーターの数とそれぞれの入力パラメーターのデータ・タイプを指定します。関数のパラメーターはすべて入力パラメーターです。関数が受信を予期しているそれぞれのパラメーターについて、リスト内に記入項目を 1 つ設ける必要があります。それぞれのパラメーターに名前を指定することができますが、これは必須ではありません。外部関数には、データ・リンクは使用できません。

関数にはパラメーターを指定しなくても構いません。この場合は、次のように、中が空の 1 組の括弧をコーディングする必要があります。

```
CREATE FUNCTION WOOFER()
```

CCSID が指定されている場合、関数に渡される前に、パラメーターはその CCSID に変換されます。CCSID が指定されていない場合は、CCSID は、関数の呼び出し時点における現行サーバーのデフォルトの CCSID によって決まります。

AS LOCATOR

これを指定すると、入力パラメーターは、実際の値ではなく、値のロケーターになります。 AS LOCATOR は、入力パラメーターに LOB データ・タイプや LOB データ・タイプをベースとする特殊タイプが指定されている場合にのみ、指定することができます。

RETURNS

関数の出力を指定します。

データ・タイプ

出力のデータ・タイプと属性を指定します。

あらゆる組み込みデータ・タイプ (ただし、LONG VARCHAR、LONG VARGRAPHIC、または DataLink は除く) や特殊タイプ (データ・リンクをベースとしていない) を指定することができます。ただし、それが結果タイプのソース関数からキャストできる場合に限ります。(データ・タイプのキャストについては、82 ページの『データ・タイプ間のキャスト』を参照してください。)

AS LOCATOR
これを指定すると、関数は、実際の値ではなく、値のロケーターを戻します。AS LOCATOR は、関数の出力に LOB データ・タイプや LOB データ・タイプをベースとする特殊タイプが指定されている場合にのみ、指定することができます。 AS LOCATOR 文節は、SQL 関数をソースとする関数に使用することはできません。

SPECIFIC *特定名*
関数の固有名を指定します。この名前は、暗黙的または明示的にスキーマ名で修飾されます。 スキーマ名も含め、この名前は、現行サーバーに存在している別の関数またはプロシージャーの特定名を示すものであってはなりません。 修飾されない場合の暗黙の修飾子は、関数名の修飾子と同じです。修飾される場合の修飾子は、関数名の修飾子と同じものにする必要があります。

特定名を指定しなかった場合、その特定名は、関数名に設定されます。この特定名の関数やプロシージャーがすでに存在している場合は、固有表名の生成に使用される規則にほぼ準拠した固有名が生成されます。

SOURCE
これを指定すると、作成する関数はソース化関数になります。ソース化関数は、別の関数 (ソース関数) によってインプリメントされます。ソース関数は、COALESCE、HASH、IFNULL、LAND、LOR、MAX、MIN、NODENAME、NODENUMBER、NULLIF、PARTITION、POSITION、RRN、STRIP、SUBSTRING、TRIM、VALUE、および XOR を除く組み込みスカラー関数または列関数、または前に作成したユーザー定義の関数のどれであっても構いません。 システム生成のユーザー定義関数 (特殊タイプ の作成時に生成された関数) の場合もあります。

引き数が 1 つ指定された場合、ソース関数は以下の組み込み関数のいずれでもかまいません。BLOB、CHAR、CLOB、DBCLOB、DECIMAL、GRAPHIC、TRANSLATE、VARCHAR、VARGRAPHIC、および ZONED。

ソース化関数をスカラー関数をもとにして直接的または間接的に作成する場合、そのソース化関数は、そのスカラー関数の属性を継承します。これには、ソース化関数のいくつかの層が含まれる場合があります。例えば、関数 A が関数 B をソースとし、関数 B は関数 C をソースとしているとします。また、関数 C はスカラー関数であるとします。関数 A と B は、関数 C の CREATE FUNCTION ステートメント上に指定されているすべての属性を継承します。

修飾されていない場合は、デフォルトのスキーマを使用して関数が検出されます。

FUNCTION *関数名*
関数名 では、現行サーバーに存在している厳密に 1 つの関数を識別する必要があります。この関数には、パラメーターをいくつでも定義することがで

きます。指定されたスキーマまたは暗黙のスキーマの中に、指定された名前の関数が複数ある場合、エラーが戻されます。

ほとんどの組み込み関数は、複数のデータ・タイプを入力パラメーターとして受け入れるように定義されています。これらの 1 つをソースとして使用するには、パラメーター・タイプを指定する必要があります。

FUNCTION *関数名 (パラメーター・タイプ 2, ...)*

関数名 (パラメーター・タイプ 2, ...) では、現行サーバーに存在していて、指定された関数シグニチャーを持つ関数を識別する必要があります。 指定されたパラメーターは、CREATE FUNCTION ステートメント上の対応する位置に指定されたデータ・タイプと一致していなければなりません。データ・タイプの数、ならびにそれらのデータ・タイプの論理連結の数を使用して、ソース関数として使用する特定の関数インスタンスを識別します。

非修飾の特殊タイプ名を指定する場合、データベース・マネージャー はその特殊タイプのスキーマ名を解決するための SQL パスを検索します。

長さ属性、精度属性、または位取り属性が指定されたデータ・タイプの場合は、値を指定するか、あるいは、1 組の中が空の括弧を使用することができます。

- 中が空の括弧は、データ・タイプが一致しているか否かの判別時に データベース・マネージャー によって属性が無視されることを示します。
- 長さ属性、精度属性、あるいは位取り属性に特定の値を使用する場合、その値は、CREATE FUNCTION ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。
- 長さ、精度、または位取りが明示的には指定されておらず、しかも中が空の括弧も指定されていない場合は、そのデータ・タイプのデフォルト属性が暗黙指定されます。デフォルト属性の詳細については、542 ページの『CREATE TABLE』を参照してください。

サブタイプまたは CCSID 属性のあるデータ・タイプの場合は、FOR DATA 文節または CCSID 文節の指定は任意選択です。どちらか一方の文節を省略すると、データ・タイプが一致しているか否かの判別時に データベース・マネージャー によって属性が無視されることが指示されます。 どちらか一方の文節を指定する場合は、CREATE FUNCTION ステートメントに暗黙的または明示的に指定されている値と一致させる必要があります。

SPECIFIC *特定名*

特定名 では、現行サーバーに存在している特定関数を識別する必要があります。

作成しようとしている関数の入力パラメーターの数は、ソース関数のパラメーターの数と同じにする必要があります。それぞれの入力パラメーターのデータ・タイプが、ソース関数の対応パラメーターと同じでなかったり、その対応パラメーターにキャストできない場合は、エラーが発生します。ソース関数の最終結果のタイプは、ソース化関数の結果と一致させるか、あるいは、その結果にキャストできるようにする必要があります。

CCSID が指定され、戻りデータの CCSID が異なる CCSID でコード化されている場合、データは指定された CCSID に変換されます。

CCSID が指定されていない場合、戻りデータの CCSID が異なる CCSID でコード化されている場合には、戻りデータはジョブの CCSID (グラフィック・ストリング戻り値の場合は、ジョブに関連したグラフィック CCSID) に変換されます。変換時に文字が失われるのを防ぐために、関数から戻される文字をすべて表現できる CCSID を明示的に指定することを考慮してください。 これは、データ・タイプがグラフィック・ストリング・データの場合に特に重要です。 この場合、CCSID 13488 (UCS-2 グラフィック・ストリング・データ) を使用することを考慮してください。

使用上の注意

ソース化関数が作成されると、その関数を表示する小型のサービス・プログラム・オブジェクトが作成されます。このサービス・プログラムが別のシステムに保管および復元されると、CREATE FUNCTION ステートメントの属性は自動的にそのシステム上のカタログに追加されます。

ソース関数上で指定された属性はすべて新しいソース化関数に伝搬されます (例えば、PARAMETER STYLE、STATIC DISPATCH など)。

例 1

組み込み INTEGER データ・タイプをベースにした特殊タイプ HATSIZE を作成したと想定します。異なる部門の平均の帽子サイズを計算するために、AVG 関数を作成します。

```
EXEC SQL
   CREATE FUNCTION AVG (HATSIZE) RETURNS HATSIZE
   SOURCE AVG (INTEGER);
```

例 2

Smith が外部スカラー関数 CENTER を自分のスキーマに登録した後で、この関数の使用が必要であることを決定します。ただし、この関数に、1 つの INTEGER 引き数と 1 つの FLOAT 引き数ではなく、2 つの INTEGER 引き数を受け入れさせるとします。CENTER をベースにしたソース化関数を作成します。

```
EXEC SQL
   CREATE FUNCTION MYCENTERG (INTEGER, INTEGER)
     RETURNS FLOAT
     SOURCE SMITH.CENTER (INTEGER, FLOAT);
```

CREATE FUNCTION (SQL スカラー)

CREATE FUNCTION (SQL スカラー) ステートメントは、現行サーバー上に SQL 関数を作成します。この関数は、単一の結果を戻します。

呼び出し

このステートメントは、アプリケーション・プログラムの中に組み込んだり、あるいは、対話式に出すことができます。このステートメントは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも次のいずれか 1 つを含める必要があります。

- SYSFUNCS カタログ・ビューと SYSPARMS カタログ表の場合
  - 該当の表に対する INSERT 特権、および
  - QSYS2 ライブラリーに対する *EXECUTE システム権限
- 管理権限

ステートメントの権限 ID は、次の場合に表についての INSERT 特権を持ちます。

- その表の所有者である。
- その表についての INSERT 特権を認可されている、または
- その表についての *OBJOPR および *ADD システム権限が認可されている。

このステートメントの権限 ID によって保持される特権には、少なくとも次の 1 つが含まれていなければなりません。

- 次のシステム権限
  - サービス・プログラム作成 (CRTSRVPGM) コマンドに対する *USE
  - 関数が作成されるライブラリーに対する *EXECUTE と *ADD
- 管理権限

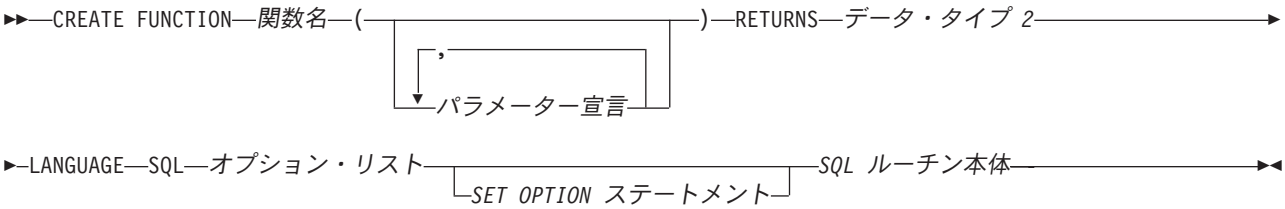
特殊タイプを参照する場合は、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- ステートメント内に識別されているそれぞれの 特殊タイプ に対しては次のもの。
  - その 特殊タイプ に対する USAGE 特権。および
  - その 特殊タイプ が入っているライブラリーに対する *EXECUTE システム権限
- 管理権限

次のいずれかに該当する場合、ステートメントの権限 ID には、特殊タイプ に対する USAGE 特権が与えられます。

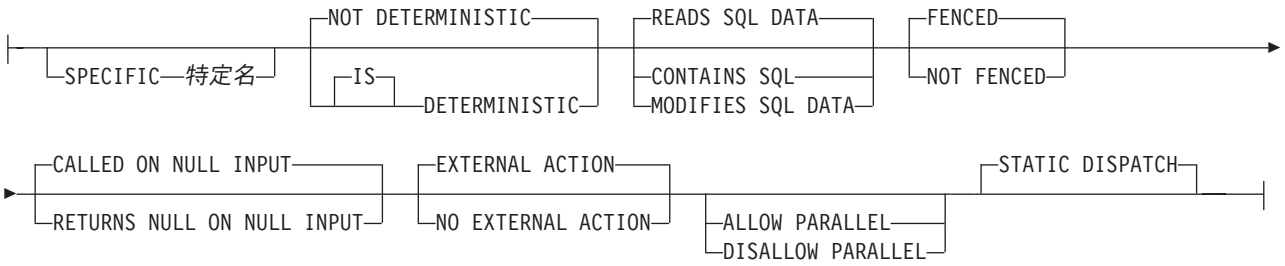
- その 特殊タイプ の所有者である。
- その 特殊タイプ に対する USAGE 特権が付与されている。
- その 特殊タイプ に対するシステム権限の *OBJOPR と *EXECUTE が付与されている。

構文



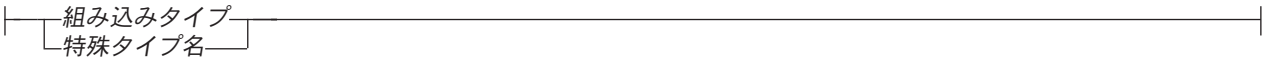
パラメーター宣言:

オプション・リスト:

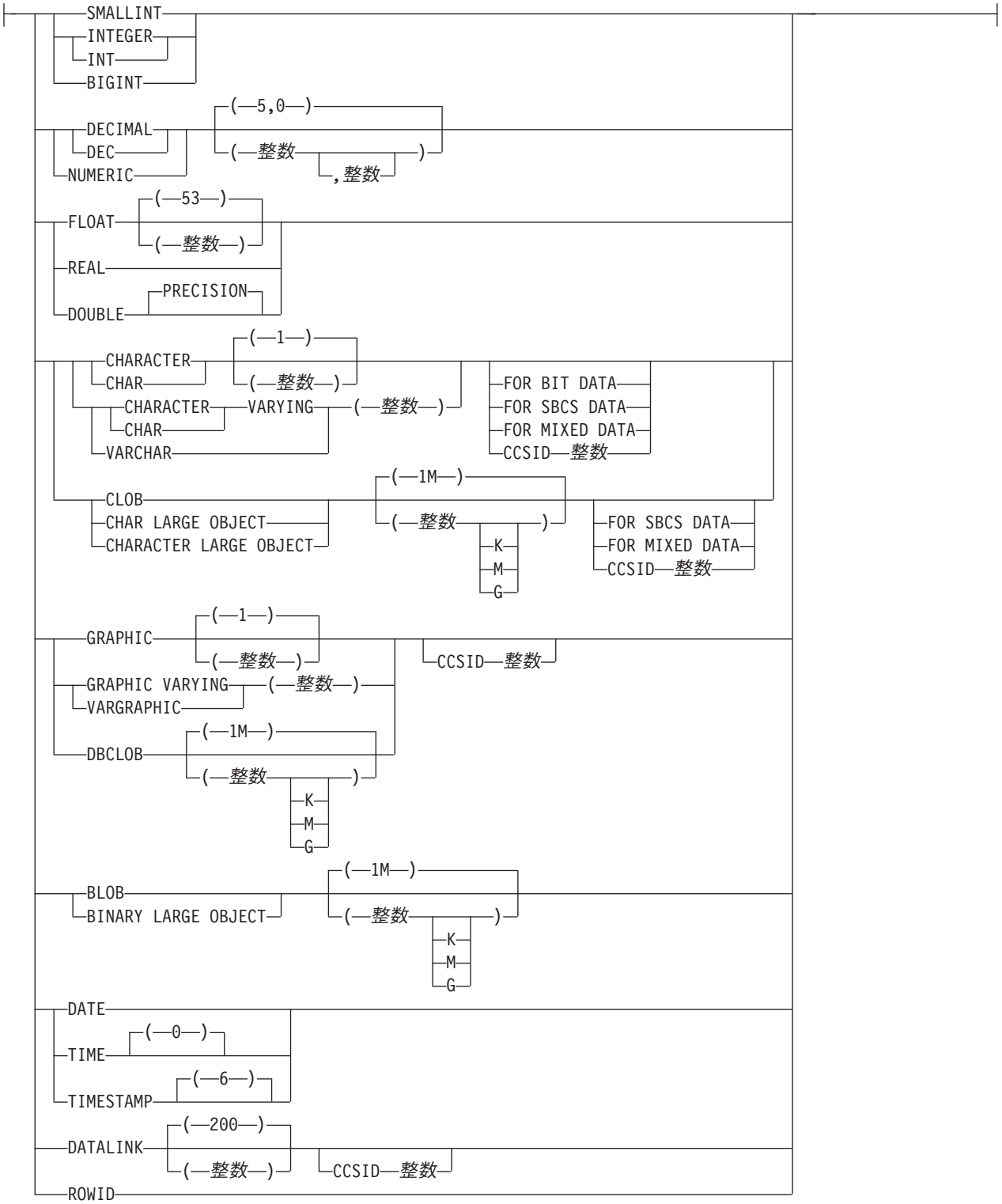


SQL ルーチン本体:

データ・タイプ:



組み込みタイプ:



説明

関数名
ユーザー定義の関数の名前を指定します。名前、スキーマ名、パラメーターの数、およびそれぞれのパラメーターのデータ・タイプ (データ・タイプの長さ、精度、位取り、または CCSID の属性に関係なく) の組み合わせで、現行サーバー上に存在しているユーザー定義の関数を識別してはなりません。

SQL 命名の場合、関数は、暗黙または明示修飾子で指定されたスキーマ内に作成されます。

システム命名の場合、関数は、修飾子で指定されたスキーマ内に作成されます。修飾子を指定しない場合、関数は、現行ライブラリー (*CURLIB) 内に作成されます。現行ライブラリーがない場合、関数は QGPL 内に作成されます。

通常、それぞれの関数の関数シグニチャーが固有であれば、複数の関数に同じ名前を指定することができます。

一部の関数名は、システムが使用するために予約されています。 詳細については、446 ページの『関数名の選択』を参照してください。

(パラメーター宣言,...)
関数のパラメーターの数とそれぞれのパラメーターのデータ・タイプを指定します。 それぞれのパラメーターに名前を指定することができますが、これは必須ではありません。

指定できるパラメーターの最大数は 90 です。

パラメーター名
入力パラメーターの名前を指定します。同じ名前を何度も指定することはできません。パラメーター名は、SQL 関数のパラメーターに対して指定する必要があります。

データ・タイプ 1
関数の入力パラメーターの数とそれぞれの入力パラメーターのデータ・タイプを指定します。関数のパラメーターはすべて入力パラメーターです。関数が受信を予期しているそれぞれのパラメーターについて、リスト内に記入項目を 1 つ設ける必要があります。

関数にはパラメーターを指定しなくても構いません。この場合は、次のように、中が空の 1 組の括弧をコーディングする必要があります。

```
CREATE FUNCTION WOOFER()
```

CCSID が指定されている場合、関数に渡される前に、パラメーターはその CCSID に変換されます。CCSID が指定されていない場合は、CCSID は、関数の呼び出し時点における現行サーバーのデフォルトの CCSID によって決まります。

RETURNS
関数の出力を指定します。

データ・タイプ 2
出力のデータ・タイプと属性を指定します。

あらゆる組み込みデータ・タイプ (ただし、LONG VARCHAR または LONG VARGRAPHIC は除く) や特殊タイプを指定することができます。

CCSID が指定され、戻りデータの CCSID が異なる CCSID でコード化されている場合、データは指定された CCSID に変換されます。

CCSID が指定されていない場合、戻りデータの CCSID が異なる CCSID でコード化されている場合には、戻りデータはジョブの CCSID (グラフィック・ストリング戻り値の場合は、ジョブに関連したグラフィック CCSID) に変換されます。 変換時に文字が失われるのを防ぐために、関数から戻される文字をすべて表現できる CCSID を明示的に指定することを考慮してください。 これは、データ・タイプがグラフィック・ストリング・データの場合に特に重要です。 この場合、CCSID 13488 (UCS-2 グラフィック・ストリング・データ) を使用することを考慮してください。

SPECIFIC *特定名*

関数の固有名を指定します。この名前は、暗黙的または明示的にスキーマ名で修飾されます。 スキーマ名も含め、この名前は、現行サーバーに存在している別の関数またはプロシージャーの特定名を示すものであってはなりません。 修飾されない場合の暗黙の修飾子は、関数名の修飾子と同じです。修飾される場合の修飾子は、関数名の修飾子と同じものにする必要があります。

特定名を指定しなかった場合、その特定名は、関数名に設定されます。この特定名の関数やプロシージャーがすでに存在している場合は、固有表名の生成に使用される規則にほぼ準拠した固有名が生成されます。

LANGUAGE SQL

これは SQL 関数であることを指定します。

DETERMINISTIC または **NOT DETERMINISTIC**

関数が決定的であるか否かを指定します。

NOT DETERMINISTIC

これを指定すると、関数は、必ずしも、同一の入力引き数が指定された連続関数呼び出しから同じ結果を戻すとは限らなくなります。 NOT DETERMINISTIC は、特殊レジスターまたは非決定的関数に対する参照がこの関数に含まれている場合に指定してください。

DETERMINISTIC

これを指定すると、関数は、必ず、同一の入力引き数が指定された連続呼び出しから同じ結果を戻します。

CONTAINS SQL、**READS SQL DATA**、または **MODIFIES SQL DATA**

この関数がなんらかの SQL ステートメントを実行できるかどうか、および実行できる場合にどのようなタイプのステートメントを実行できるかを指定します。データベース・マネージャーは、この関数が発行する SQLがこの条件を満たしているかどうかを検査します。各データ・アクセス指示の下で実行できる SQL ステートメントの詳細なリストについては、913 ページの『付録 F. SQL ステートメントの特性』を参照してください。

CONTAINS SQL

この関数は、データを読み取りまたは変更する SQL ステートメントを実行しません。

READS SQL DATA

この関数は、データを変更する SQL ステートメントを実行しません。

MODIFIES SQL DATA
この関数は、どの関数でもサポートされないステートメントを除くすべての SQL ステートメントを実行できます。

FENCED または **NOT FENCED**
この関数を、呼び出し元の SQL ステートメントと同じスレッド内で実行するか、別のスレッドで実行するかを指定します。

FENCED
この関数は別のスレッドで実行されます。同じ SQL ステートメント内で同じ関数を複数回呼び出すと、互いに競合することがあるので、関数に SQL カーソルが含まれている場合は、FENCED を選択するのが最も安全です。

NOT FENCED
この関数は、呼び出し元の SQL ステートメントと同じスレッド内で実行できます。 NOT FENCED を指定すると、この関数の呼び出し間でカーソルをオープン状態のままにすることができます。カーソルをオープン状態のままにしておくことができるため、関数の呼び出し間でカーソル位置も同じに維持されます。

NULL INPUT
入力パラメーターが NULL である場合に、関数を呼び出す必要があるか否かを指定します。

CALLED ON NULL INPUT
常に関数を呼び出します。

RETURNS NULL ON NULL INPUT
ヌル値が渡され、関数の出力が NULL になる場合は、関数を呼び出す必要はありません。

EXTERNAL ACTION または **NO EXTERNAL ACTION**
関数に外部アクションが含まれているかどうかを指定します。

EXTERNAL ACTION
関数は、なんらかの外部アクション (関数プログラムの有効範囲外のアクション) を行います。したがって、関数は、それぞれの連続関数呼び出しで呼び出す必要があります。 EXTERNAL ACTION は、この関数に、外部アクションを持つ他の関数に対する参照が含まれている場合に、指定してください。

NO EXTERNAL ACTION
関数は外部アクションを行いません。この関数は、連続した各関数呼び出しごとに呼び出す必要はありません。

PARALLEL
関数を並列で実行できるかどうかを指定します。

ALLOW PARALLEL
これを指定すると、関数は並列で実行できるようになります。

DISALLOW PARALLEL
これを指定すると、関数は並列で実行できなくなります。

以下の文節の 1 つまたは複数を指定した場合、デフォルトは DISALLOW PARALLEL になります。

- NOT DETERMINISTIC
- EXTERNAL ACTION
- MODIFIES SQL DATA

それ以外の場合は、ALLOW PARALLEL がデフォルトです。

STATIC DISPATCH
関数を静的にディスパッチすることを指定します。すべての関数が静的にディスパッチされます。

SET OPTION ステートメント
関数を作成するときに使用するオプションを指定します。例えば、デバッグ可能な関数を作成するときは、次のステートメントを含めることができます。

```
SET OPTION DBGVIEW = *STMT
```

詳しくは、 770 ページの『SET OPTION』を参照してください。

オプション CLOSQLCSR、 CNULRQD、 DFTRDBCOL、 DYNDFTCOL、 NAMING は、CREATE FUNCTION ステートメントでは使用できません。

SQL ルーチン本体
複合ステートメントも含め、単一の SQL ステートメントを指定します。SQL 関数の定義についての詳細は、 819 ページの『第 6 章 SQL 制御ステートメント』を参照してください。

CONNECT、 SET CONNECTION、 RELEASE、 DISCONNECT、 COMMIT、 ROLLBACK、 SET TRANSACTION ステートメントを実行するプロシージャーへの呼び出しは、関数内では使用できません。

SQL ルーチン本体 が複合ステートメントである場合は、そのステートメントには RETURN ステートメントが少なくとも 1 つは含まれていなければならず、関数の呼び出し時に RETURN ステートメントが 1 つ実行される必要があります。

使用上の注意

関数の所有権 : SQL 名を指定した場合は、関数の*所有者* は、作成した関数が入れられるスキーマと同じ名前のユーザー・プロファイルです。その他の場合は、関数の*所有者* は、このステートメントを実行しているジョブのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

システム名を指定した場合は、関数の*所有者* は、このステートメントを実行しているジョブのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

関数の権限 : SQL 名を使用する場合は、関数は、*PUBLIC に対するシステム権限 *EXCLUDE を使用して作成されます。システム名を使用する場合、関数は、スキーマの作成権限 (CRTAUT) パラメーターによって決められる *PUBLIC に対する権限を使用して作成されます。

関数の所有者がグループ・プロファイルのメンバー (GRPPRF キーワード) であり、グループ権限が指定されている (GRPAUT キーワード) 場合は、そのグループ・プロファイルにも、その関数に対する権限が与えられます。

関数の作成

SQL 関数が作成される場合、データベース・マネージャー は、組み込み SQL ステートメントと一緒に C ソース・コードが収められる一時ソース・ファイルを作成します。次いで、CRTSRVPGM コマンドを使用して、*SRVPGM オブジェクトが作成されます。サービス・プログラムの作成に使用される SQL オプションは、CREATE FUNCTION ステートメントの実行時に有効なオプションです。サービス・プログラムは、ACTGRP(*CALLER) を使用して作成します。

ソース・ファイル・メンバーと *SRVPGM オブジェクトの判別には、特定名が使用されます。特定名が有効なシステム名ならば、その特定名がメンバーやプログラムの名前として使用されます。メンバーは、すでに存在している場合、オーバーレイされます。指定されたライブラリー内にプログラムがすでに存在している場合は、システム表名の生成に関する規則を使用して固有名が生成されます。特定名が有効なシステム名でない場合は、システム表名の生成に関する規則を使用して固有名が生成されます。

関数の属性は、関連したサービス・プログラム・オブジェクトに保管されます。*SRVPGM オブジェクトが保管された後、このシステムや別のシステム上に復元すると、カタログはそれらの属性を使用して自動的に更新されます。

関数の復元時には、次のような動作が生じます。

- 関数が初めに作成される時点で特定名が指定されており、しかもその名前が固有でない場合は、エラーが出されます。
- 特定名が指定されていない場合は、必要に応じて固有名が生成されます。
- シグニチャーが固有でない場合は、関数を登録することは不可能で、エラーが出されます。

ID の解決

ルーチン本体内に指定された表が存在している場合、SQL ルーチンの作成時に特定の列、SQL パラメーター、または SQL 変数を識別するために SQL ルーチン本体内のすべての参照が解決されます。表が存在しない場合は、関数の作成時に変数やパラメーターを識別するために、SQL 変数またはパラメーターとして存在しているすべての名前が解決されます。残りの名前は、関数の呼び出し時に表に結合される列であると想定されます。

列、ならびに、SQL 変数とパラメーターに重複名が使用された場合は、列に表指定子、パラメーターに関数名、また、SQL 変数にラベル名を使用してその重複名を修飾します。

関数の呼び出し

SQL 関数が呼び出されると、その関数は呼び出しプログラムの活動化グループ内で実行します。

SELECT ステートメントの選択リストで関数が指定され、その関数が EXTERNAL ACTION または MODIFIES SQL DATA を指定している場合、関数は、戻される各行に対してだけ呼び出されます。それ以外の場合は、選択されていない行に対して UDF が呼び出されることもあります。

同義のキーワード

以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- キーワード VARIANT と NOT VARIANT は、NOT DETERMINISTIC と DETERMINISTIC の同義語として使用することができます。
- キーワード NULL CALL と NOT NULL CALL は、CALLED ON NULL INPUT と RETURNS NULL ON NULL INPUT の同義語として使用できます。

例 1

規則をインプリメントするための SQL 関数 NTEST1 を作成します。

```
output = 2 * input - 4
```

```
CREATE FUNCTION NTEST1 (p_input INTEGER)
  RETURNS INTEGER
  LANGUAGE SQL
  SPECIFIC MINENULL1
  func1_lab:
    BEGIN
      DECLARE p_output INT;
      IF p_input IS NULL THEN
        SET p_output = NULL;
      ELSE
        SET p_output = 2 * p_input - 4;
     END IF;
     RETURN p_output;
    END
```

CREATE FUNCTION (SQL 表)

CREATE FUNCTION (SQL 表) ステートメントは、現行サーバー上に SQL 表関数を作成します。その関数は単一の結果表を戻します。

呼び出し

このステートメントは、アプリケーション・プログラムの中に組み込んだり、あるいは、対話式に出すことができます。このステートメントは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも次のいずれか 1 つを含める必要があります。

- SYSFUNCS カタログ・ビューと SYSPARMS カタログ表の場合
  - 該当の表に対する INSERT 特権、および
  - QSYS2 ライブラリーに対する *EXECUTE システム権限
- 管理権限

ステートメントの権限 ID は、次の場合に表についての INSERT 特権を持ちます。

- その表の所有者である。
- その表についての INSERT 特権を認可されている、または
- その表についての *OBJOPR および *ADD システム権限が認可されている。

このステートメントの権限 ID によって保持される特権には、少なくとも次の 1 つが含まれていなければなりません。

- 次のシステム権限
  - サービス・プログラム作成 (CRTSRVPGM) コマンドに対する *USE
  - 関数が作成されるライブラリーに対する *EXECUTE と *ADD
- 管理権限

特殊タイプを参照する場合は、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

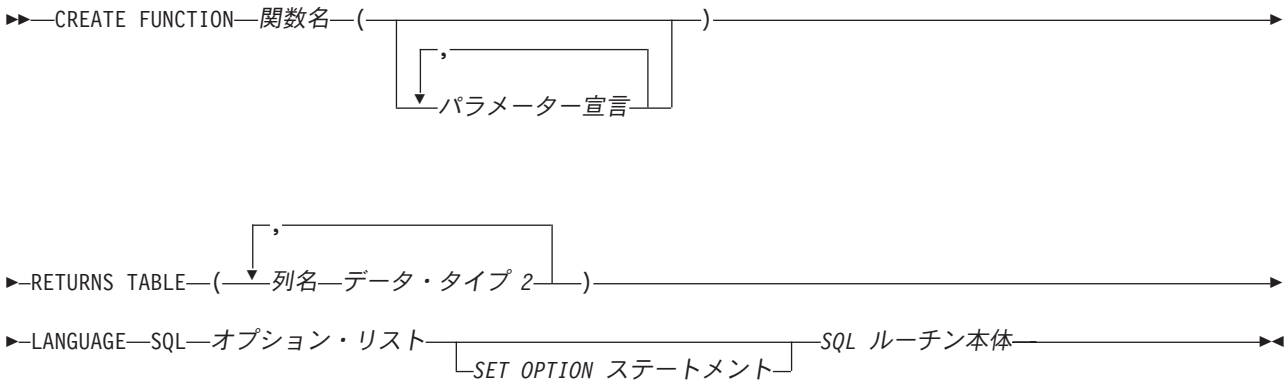
- ステートメント内に識別されているそれぞれの 特殊タイプ に対しては次のもの。
  - その 特殊タイプ に対する USAGE 特権。および
  - その 特殊タイプ が入っているライブラリーに対する *EXECUTE システム権限
- 管理権限

次のいずれかに該当する場合、ステートメントの権限 ID には、特殊タイプ に対する USAGE 特権が与えられます。

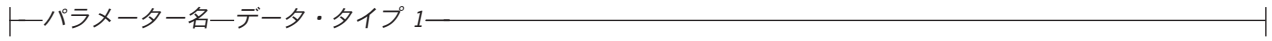
- その 特殊タイプ の所有者である。
- その 特殊タイプ に対する USAGE 特権が付与されている。
- その 特殊タイプ に対するシステム権限の *OBJOPR と *EXECUTE が付与されている。

CREATE FUNCTION (SQL 表)

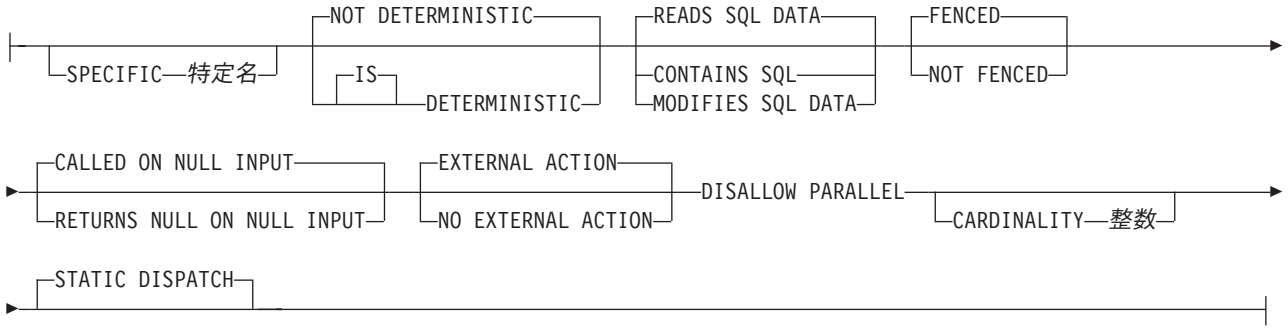
構文



パラメーター宣言:

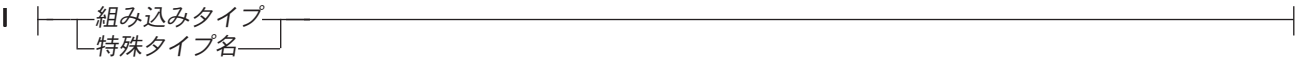


オプション・リスト:

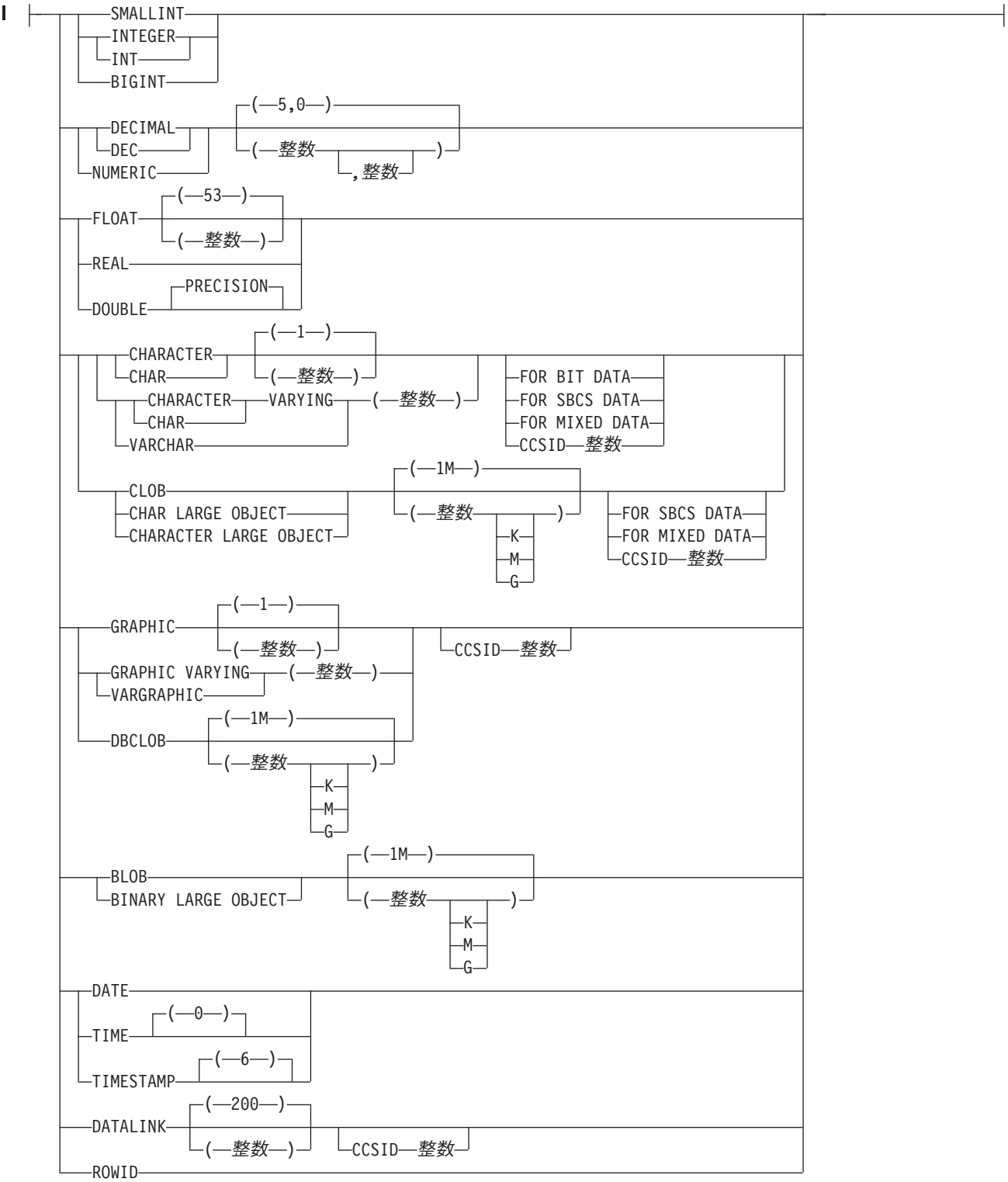


SQL ルーチン本体:

データ・タイプ:



組み込みタイプ:



説明

関数名

ユーザー定義の関数の名前を指定します。名前、スキーマ名、パラメーターの数、およびそれぞれのパラメーターのデータ・タイプ (データ・タイプの長さ、精度、位取り、または CCSID の属性に関係なく) の組み合わせで、現行サーバー上に存在しているユーザー定義の関数を識別してはなりません。

SQL 命名の場合、関数は、暗黙または明示修飾子で指定されたスキーマ内に作成されます。

システム命名の場合、関数は、修飾子で指定されたスキーマ内に作成されます。修飾子を指定しない場合、関数は、現行ライブラリー (*CURLIB) 内に作成されます。現行ライブラリーがない場合、関数は QGPL 内に作成されます。

通常、それぞれの関数の関数シグニチャーが固有であれば、複数の関数に同じ名前を指定することができます。

一部の関数名は、システムが使用するために予約されています。 詳細については、446 ページの『関数名の選択』を参照してください。

(パラメーター宣言,...)

関数のパラメーターの数とそれぞれのパラメーターのデータ・タイプを指定します。 それぞれのパラメーターに名前を指定することができますが、これは必須ではありません。

指定できるパラメーターの最大数は 90 です。

パラメーター名

入力パラメーターの名前を指定します。同じ名前を何度も指定することはできません。パラメーター名は、SQL 関数のパラメーターに対して指定する必要があります。

データ・タイプ 1

関数の入力パラメーターの数とそれぞれの入力パラメーターのデータ・タイプを指定します。関数のパラメーターはすべて入力パラメーターです。関数が受信を予期しているそれぞれのパラメーターについて、リスト内に記入項目を 1 つ設ける必要があります。

関数にはパラメーターを指定しなくても構いません。この場合は、次のように、中が空の 1 組の括弧をコーディングする必要があります。

```
CREATE FUNCTION WOOFER()
```

CCSID が指定されている場合、関数に渡される前に、パラメーターはその CCSID に変換されます。CCSID が指定されていない場合は、CCSID は、関数の呼び出し時点における現行サーバーのデフォルトの CCSID によって決まります。

RETURNS TABLE

関数の出力表を指定します。

パラメーターの数が N であるとすると、列の数は (247-(N*2))/2 以下でなければなりません。

列名
出力表の列の名前を指定します。同じ名前を何度も指定することはできません。

データ・タイプ 2
出力のデータ・タイプと属性を指定します。

あらゆる組み込みデータ・タイプ (ただし、LONG VARCHAR または LONG VARGRAPHIC は除く) や特殊タイプを指定することができます。

CCSID が指定され、戻りデータの CCSID が異なる CCSID でコード化されている場合、データは指定された CCSID に変換されます。

CCSID が指定されていない場合、戻りデータの CCSID が異なる CCSID でコード化されている場合には、戻りデータはジョブの CCSID (グラフィック・ストリング戻り値の場合は、ジョブに関連したグラフィック CCSID) に変換されます。 変換時に文字が失われるのを防ぐために、関数から戻される文字をすべて表現できる CCSID を明示的に指定することを考慮してください。 これは、データ・タイプがグラフィック・ストリング・データの場合に特に重要です。 この場合、CCSID 13488 (UCS-2 グラフィック・ストリング・データ) を使用することを考慮してください。

SPECIFIC *特定名*
関数の固有名を指定します。この名前は、暗黙的または明示的にスキーマ名で修飾されます。 スキーマ名も含め、この名前は、現行サーバーに存在している別の関数またはプロシージャーの特定名を示すものであってはなりません。 修飾されない場合の暗黙の修飾子は、関数名の修飾子と同じです。修飾される場合の修飾子は、関数名の修飾子と同じものにする必要があります。

特定名を指定しなかった場合、その特定名は、関数名に設定されます。この特定名の関数やプロシージャーがすでに存在している場合は、固有表名の生成に使用される規則にほぼ準拠した固有名が生成されます。

LANGUAGE SQL
これは SQL 関数であることを指定します。

DETERMINISTIC または **NOT DETERMINISTIC**
関数が決定的であるか否かを指定します。

NOT DETERMINISTIC
これを指定すると、関数は、必ずしも、同一の入力引き数が指定された連続関数呼び出しから同じ結果を戻すとは限らなくなります。 NOT DETERMINISTIC は、特殊レジスターまたは非決定的関数に対する参照がこの関数に含まれている場合に指定してください。

DETERMINISTIC
これを指定すると、関数は、必ず、同一の入力引き数が指定された連続呼び出しから同じ結果を戻します。

CONTAINS SQL、**READS SQL DATA**、または **MODIFIES SQL DATA**
この関数がなんらかの SQL ステートメントを実行できるかどうか、および実行できる場合にどのようなタイプのステートメントを実行できるかを指定します。データベース・マネージャーは、この関数が発行する SQLがこの条件を満たしているかどうかを検査します。各データ・アクセス指示の下で実行できる SQL

ステートメントの詳細なリストについては、 913 ページの『付録 F. SQL ステートメントの特性』を参照してください。

CONTAINS SQL
この関数は、データを読み取りまたは変更する SQL ステートメントを実行しません。

READS SQL DATA
この関数は、データを変更する SQL ステートメントを実行しません。

MODIFIES SQL DATA
この関数は、どの関数でもサポートされないステートメントを除くすべての SQL ステートメントを実行できます。

FENCED または **NOT FENCED**
この関数を、呼び出し元の SQL ステートメントと同じスレッド内で実行するか、別のスレッドで実行するかを指定します。

FENCED
この関数は別のスレッドで実行されます。同じ SQL ステートメント内で同じ関数を複数回呼び出すと、互いに競合することがあるので、関数に SQL カーソルが含まれている場合は、FENCED を選択するのが最も安全です。

NOT FENCED
この関数は、呼び出し元の SQL ステートメントと同じスレッド内で実行できます。 NOT FENCED を指定すると、この関数の呼び出し間でカーソルをオープン状態のままにすることができます。カーソルをオープン状態のままにしておくことができるため、関数の呼び出し間でカーソル位置も同じに維持されます。

NULL INPUT
入力パラメーターが NULL である場合に、関数を呼び出す必要があるか否かを指定します。

CALLED ON NULL INPUT
常に関数を呼び出します。

RETURNS NULL ON NULL INPUT
表関数のオープン時に、その関数の引き数のどれかがヌルである場合は、そのユーザー定義の表関数は呼び出されず、関数出力は空の表 (行のない表) になります。

EXTERNAL ACTION または **NO EXTERNAL ACTION**
関数に外部アクションが含まれているかどうかを指定します。

EXTERNAL ACTION
関数は、なんらかの外部アクション (関数プログラムの有効範囲外のアクション) を行います。したがって、関数は、それぞれの連続関数呼び出しで呼び出す必要があります。 EXTERNAL ACTION は、この関数に、外部アクションを持つ他の関数に対する参照が含まれている場合に、指定してください。

NO EXTERNAL ACTION
関数は外部アクションを行いません。この関数は、連続した各関数呼び出しごとに呼び出す必要はありません。

DISALLOW PARALLEL
これを指定すると、関数は並列で実行できなくなります。 表関数は並列では実行できません。

CARDINALITY *整数*
この任意選択の文節は、最適化を目的として、この関数が戻すものとして予期される行数の見積もりを指定します。整数の有効な値の範囲は、0 ～ 2 147 483 647 です。

表関数について CARDINALITY 文節を指定しなかった場合は、データベース・マネージャーは、デフォルトに基づき特定の有限値を想定します。

STATIC DISPATCH
関数を静的にディスパッチすることを指定します。すべての関数が静的にディスパッチされます。

SET OPTION ステートメント
関数を作成するときに使用するオプションを指定します。例えば、デバッグ可能な関数を作成するときは、次のステートメントを含めることができます。

```
SET OPTION DBGVIEW = *STMT
```

詳しくは、 770 ページの『SET OPTION』を参照してください。

オプション CLOSQLCSR、 CNULRQD、 DFTRDBCOL、 DYNDFTCOL、 NAMING は、CREATE FUNCTION ステートメントでは使用できません。

SQL ルーチン本体
複合ステートメントも含め、単一の SQL ステートメントを指定します。SQL 関数の定義についての詳細は、 819 ページの『第 6 章 SQL 制御ステートメント』を参照してください。

CONNECT、 SET CONNECTION、 RELEASE、 DISCONNECT、 COMMIT、 ROLLBACK、 SET TRANSACTION ステートメントを実行するプロシージャーへの呼び出しは、関数内では使用できません。

SQL ルーチン本体 が複合ステートメントである場合は、そのステートメントには RETURN ステートメントが 1 つだけ含まれていなければならず、関数の呼び出し時にその RETURN ステートメントが実行される必要があります。

使用上の注意

関数の所有権 : SQL 名を指定した場合は、関数の*所有者* は、作成した関数が入れられるスキーマと同じ名前のユーザー・プロファイルです。 その他の場合は、関数の*所有者* は、このステートメントを実行しているジョブのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

システム名を指定した場合は、関数の*所有者* は、このステートメントを実行しているジョブのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

関数の権限 : SQL 名を使用する場合は、関数は、*PUBLIC に対するシステム権限 *EXCLUDE を使用して作成されます。 システム名を使用する場合、関数は、スキーマの作成権限 (CRTAUT) パラメーターによって決められる *PUBLIC に対する権限を使用して作成されます。

関数の所有者がグループ・プロファイルのメンバー (GRPPRF キーワード) であり、グループ権限が指定されている (GRPAUT キーワード) 場合は、そのグループ・プロファイルにも、その関数に対する権限が与えられます。

関数の作成

SQL 関数が作成される場合、データベース・マネージャー は、組み込み SQL ステートメントと一緒に C ソース・コードが収められる一時ソース・ファイルを作成します。次いで、CRTSRVPGM コマンドを使用して、*SRVPGM オブジェクトが作成されます。 サービス・プログラムの作成に使用される SQL オプションは、CREATE FUNCTION ステートメントの実行時に有効なオプションです。サービス・プログラムは、ACTGRP(*CALLER) を使用して作成します。

ソース・ファイル・メンバーと *SRVPGM オブジェクトの判別には、特定名が使用されます。特定名が有効なシステム名ならば、その特定名がメンバーやプログラムの名前として使用されます。メンバーは、すでに存在している場合、オーバーレイされます。指定されたライブラリー内にプログラムがすでに存在している場合は、システム表名の生成に関する規則を使用して固有名が生成されます。特定名が有効なシステム名でない場合は、システム表名の生成に関する規則を使用して固有名が生成されます。

関数の属性は、関連したサービス・プログラム・オブジェクトに保管されます。*SRVPGM オブジェクトが保管された後、このシステムや別のシステム上に復元すると、カタログはそれらの属性を使用して自動的に更新されます。

関数の復元時には、次のような動作が生じます。

- 関数が初めに作成される時点で特定名が指定されており、しかもその名前が固有でない場合は、エラーが出されます。
- 特定名が指定されていない場合は、必要に応じて固有名が生成されます。
- シグニチャーが固有でない場合は、関数を登録することは不可能で、エラーが出されます。

ID の解決

ルーチン本体内に指定された表が存在している場合、SQL ルーチンの作成時に特定の列、SQL パラメーター、または SQL 変数を識別するために SQL ルーチン本体内のすべての参照が解決されます。表が存在しない場合は、関数の作成時に変数やパラメーターを識別するために、SQL 変数またはパラメーターとして存在しているすべての名前が解決されます。残りの名前は、関数の呼び出し時に表に結合される列であると想定されます。

列、ならびに、SQL 変数とパラメーターに重複名が使用された場合は、列に表指定子、パラメーターに関数名、また、SQL 変数にラベル名を使用してその重複名を修飾します。

関数の呼び出し

SQL 関数が呼び出されると、その関数は呼び出しプログラムの活動化グループ内で実行します。

同義のキーワード

以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- キーワード VARIANT と NOT VARIANT は、NOT DETERMINISTIC と DETERMINISTIC の同義語として使用することができます。
- キーワード NULL CALL と NOT NULL CALL は、CALLED ON NULL INPUT と RETURNS NULL ON NULL INPUT の同義語として使用できます。

例

指定した部門番号に該当する社員を戻す表関数を定義します。

```
CREATE FUNCTION DEPTEMPLOYEES (DEPTNO CHAR(3))
  RETURNS TABLE (EMPNO CHAR(6),
                 LASTNAME VARCHAR(15),
                 FIRSTNAME VARCHAR(12))
  LANGUAGE SQL
  READS SQL DATA
  NO EXTERNAL ACTION
  DETERMINISTIC
  DISALLOW PARALLEL
  RETURN
      SELECT EMPNO,LASTNAME,FIRSTNME
        FROM EMPLOYEE
        WHERE EMPLOYEE.WORKDEPT =DEPTEMPLOYEES.DEPTNO
```

CREATE INDEX

CREATE INDEX ステートメントは、現行サーバーで表の索引を作成します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことも、対話式に呼び出すこともできます。このステートメントは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- 次のシステム権限
  - 論理ファイル作成 (CRTLF) コマンドに対する *USE 権限。
  - 索引が作成されるライブラリーに対する *EXECUTE および *ADD 権限
  - データ・ディクショナリーに対する *CHANGE 権限。ただし、索引が作成されるライブラリーが、データ・ディクショナリーをもつ SQL スキーマの場合。
- 管理権限

このステートメントの権限 ID によって保持される特権には、少なくとも次の 1 つが含まれていなければなりません。

- 該当の表に対する INDEX 特権。
- 管理権限。

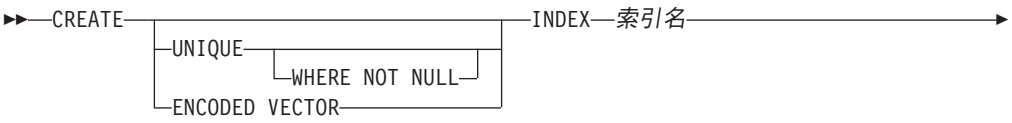
次の場合に、このステートメントの権限 ID は INDEX 特権を持ちます。

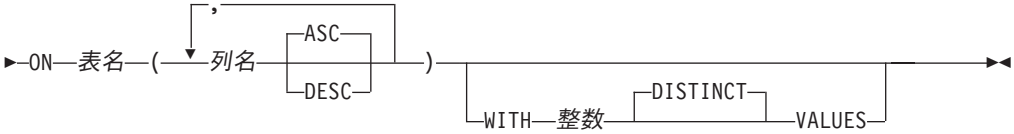
- その表の所有者である。
- その表に対する INDEX または ALTER 特権が認可されている。
- その表に対する *OBJALTER または *OBJMGT システム権限のいずれかが認可されている。

SQL 名が指定され、該当の表が作成されるライブラリーの名前と同じ名前のユーザー・プロファイルが存在し、しかもその名前がステートメントの権限 ID と異なっている場合、ステートメントの権限 ID によって保持される特権には、少なくとも次の 1 つが含まれていなければなりません。

- その名前を持つユーザー・プロファイルに対する *ADD システム権限
- 管理権限

構文





説明

UNIQUE
表に、同一の索引キーの値を持つ行が複数入るのを防止します。この制約が適用されるのは、表の行を更新するときと、新しい行を挿入するときです。

CREATE INDEX ステートメントの実行時にも、この制約が検査されます。重複するキーの値を持つ行がすでに表に入っている場合、索引は作成されません。

UNIQUE を使用した場合は、ヌル値も他のすべての値と同じように扱われます。例えば、ヌル値を入れることができる単一の列をキーにすると、その列には、ヌル値が 1 つしか入らなくなります。

UNIQUE WHERE NOT NULL
索引キーに非ヌルの同一の値をもつ複数の行が表に入るのを防止します。複数のヌル値は使用できます。その他の点では、UNIQUE と同等です。

ENCODED VECTOR
これを指定すると、結果の索引は、コード化ベクトル索引 (EVI) になります。

コード化ベクトル索引を使用して、行の順序を保証することはできません。これは、データベース・マネージャーが照会のパフォーマンスを向上させる場合に使用します。詳細については、データベース・パフォーマンスおよび Query 最適化を参照してください。

索引名
索引の名前を指定します。暗黙的または明示的修飾子も含め、この名前は、現行サーバーにすでに存在している索引、表、ビュー、別名、またはファイルと同じ名前にすることはできません。

SQL 名が指定されている場合、索引は、暗黙的または明示的修飾子で指定しているスキーマ内に作成されます。

システム名が指定されている場合、索引名は、修飾子で指定しているスキーマ内に作成されます。修飾されていない場合、索引名は、その索引の作成に使用した表と同じスキーマ内に作成されます。

索引名が有効なシステム名でない場合、DB2 UDB for iSeries はシステム名を生成します。名前の生成に関する規則については、572 ページの 『表名の生成の規則』を参照してください。

ON *表名*
その索引を作成したい表を指定します。この*表名* は、現行サーバーに存在している基礎表 (ビューではなく) を識別するものでなくてはなりません。

(列名, ...)
索引キーを構成する列のリストを識別します。

それぞれの*列名* は、該当の表の列を識別する非修飾の名前でなければなりません。 同じ列を複数回指定することができます。*列名* では、LOB 列、

DATALINK 列、または LOB 列や datalink 列に基づく特殊タイプを識別することはできません。指定する列の数は 120 を超えてはならず、それらの列の合計バイト数は 2000-n を超えてはなりません。ここで、n は指定した列のうちヌルが許される列の数です。

ASC
索引項目を列の値にしたがって昇順に並べます。デフォルト値は ASC です。

DESC
索引項目を列の値にしたがって降順に並べます。

WITH *整数* **DISTINCT VALUES**
特殊キー値の見積数を指定します。 この文節は、あらゆるタイプの索引に対して指定することができます。

コード化ベクトル索引の場合は、これを使用し、それぞれの特殊キー値に割り当てられるコードの初期サイズを決定します。デフォルト値は 256 です。

非コード化ベクトル索引の場合、これは、最適化プログラムへのヒントとして使用されます。

使用上の注意

指定した表にすでにデータが入っていれば、CREATE INDEX によって、そのデータに関する索引項目が作成されます。表にまだデータが入っていない場合、CREATE INDEX は、索引の記述を作成します。 (索引項目は、表にデータが挿入されたときに作成されます)。索引には、常に表の最新の状態が反映されています。

ソート順序 : SBCS または混合データを含む列に対して作成される索引は、このステートメントの実行時点で有効なソート順序に従って作成されます。ソート順序が *HEX 以外の場合は、SBCS データまたは混合データのキーは、該当のソート順序に基づいてキーが重み付けされた値です。

索引の属性 : 索引はキー付き論理ファイルとして作成されます。索引が作成される場合、ファイル待ち時間とレコード待ち時間の属性は、論理ファイル作成 (CRTLF) コマンドの WAITFILE キーワードと WAITRCD キーワード上に指定されたデフォルト値に設定されます。

分散表に対して作成される索引は、この表が配布されるサーバーのすべてで作成されます。分散表の詳細については、DB2 UDB for iSeries マルチ・システムを参照してください。

索引の所有権 : SQL 名を指定した場合は、索引の*所有者* は、作成した索引が入れられるスキーマと同じ名前のユーザー・プロファイルです。その他の場合は、索引の*所有者* は、このステートメントを実行しているジョブのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

システム名を指定した場合は、索引の*所有者* は、このステートメントを実行しているジョブのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

索引の権限 : SQL 名を使用する場合は、索引は、*PUBLIC に対するシステム権限 *EXCLUDE を使用して作成されます。 システム名を使用する場合、索引は、スキーマの作成権限 (CRTAUT) パラメーターによって決められる *PUBLIC に対する権限を使用して作成されます。

索引の所有者がグループ・プロファイルのメンバー (GRPPRF キーワード) であり、グループ権限が指定されている (GRPAUT キーワード) 場合は、そのグループ・プロファイルにも、その索引に対する権限が与えられます。

例

例 1

PROJECT 表に UNIQUE_NAM という名前の索引を作成します。この索引の目的は、表内に、同じプロジェクト名 (PROJNAME) が重複して入ることがないようにすることです。索引項目は昇順になります。

```
CREATE UNIQUE INDEX UNIQUE_NAME
  ON PROJECT(PROJNAME)
```

例 2

EMPLOYEE 表に JOB_BY_DPT という名前の索引を作成します。索引項目は、各部門 (WORKDEPT) ごとにジョブ・タイトル (JOB) にしたがって昇順に並べます。

```
CREATE INDEX JOB_BY_DPT
  ON EMPLOYEE (WORKDEPT, JOB)
```

CREATE PROCEDURE

CREATE PROCEDURE ステートメントは、現行サーバーでプロシージャーを定義します。

定義できるプロシージャーのタイプは以下のとおりです。

- 外部

  このタイプのプロシージャー・プログラムは、C、COBOL、Java などのプログラミング言語で書かれます。この外部実行ファイルは、現行サーバーで定義されているプロシージャーにより、プロシージャーの各種属性に基づいて参照されます。 514 ページの『CREATE PROCEDURE (外部)』 を参照してください。

- SQL

  このタイプのプロシージャーは SQL のみで書かれます。プロシージャー本体は、プロシージャーの各種属性と一緒に現行サーバーで定義されます。 527 ページの『CREATE PROCEDURE (SQL)』 を参照してください。

使用上の注意

パラメーターのデータ・タイプの選択

DB2 UDB for iSeries 以外のプラットフォーム間におけるプロシージャーの可搬性を得るには、次のデータ・タイプを使用しないでください。これらのデータ・タイプの表示方法は、プラットフォームに応じてそれぞれに異なる可能性があります。

- FLOAT。この代わりに、DOUBLE や REAL を使用すること。
- NUMERIC。この代わりに、DECIMAL を使用すること。

パラメーターに AS LOCATOR を指定

値の代わりにロケーターを渡すことにより、プロシージャーとの間で受け渡しするバイト数を削減できることがあります。これは、パラメーターの値が非常に大きい場合に便利です。 AS LOCATOR 文節は、実際の値の代わりにパラメーターの値へのロケーターを渡すことを指定します。 AS LOCATOR は、LOB データ・タイプまたは LOB データ・タイプに基づく特殊タイプのパラメーターの場合に限り使用するようにしてください。

SQL プロシージャーには、AS LOCATOR は指定できません。

スキーマ内でのプロシージャーの固有性の決定

現行サーバーでは、それぞれのプロシージャー・シグニチャーを固有のものにする必要があります。 プロシージャーのシグニチャーは、修飾プロシージャー名と、入力パラメーターの数を組み合わせたものです (パラメーターのデータ・タイプはプロシージャーのシグニチャーの一部ではありません)。これは、2 つの異なるスキーマに、名前が同じでパラメーター数も同じであるプロシージャーが含まれていてもよいということを意味します。ただし、1 つのスキーマに、名前もパラメーター数も同じである 2 つのプロシージャーを含めることはできません。

プロシージャーの特定名

名前もスキーマも同じである (ただしパラメーター数は異なる) 複数のプロシージャーを定義するときは、特定名も指定することをお勧めします。プロシージャーの除

去、プロシージャーに対する権限の認可または取り消し、またはプロシージャーへのコメントの付加を行うときに、特定名を使用して、そのプロシージャーを一意的に識別することができます。

SPECIFIC 文節を指定しなかった場合は、特定名が生成されます。

CREATE PROCEDURE (外部)

CREATE PROCEDURE (外部) ステートメントは、現行サーバーで外部プロシージャーを作成します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。このステートメントは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- SYSPROCS カタログ・ビューと SYSPARMS カタログ表の場合
  - 該当の表に対する INSERT 特権、および
  - QSYS2 ライブラリーに対する *EXECUTE システム権限
- 管理権限

ステートメントの権限 ID は、次の場合に表についての INSERT 特権を持ちます。

- その表の所有者である。
- その表についての INSERT 特権を認可されている、または
- その表についての *OBJOPR および *ADD システム権限が認可されている。

外部プログラムが存在している場合、このステートメントの権限 ID が保持する特権には、少なくとも次のいずれか 1 つが含まれていなければなりません。

- SQL ステートメントで参照された外部プログラムの場合
  - その外部プログラムに対する *EXECUTE システム権限、および
  - その外部プログラムが入っているライブラリーに対する *EXECUTE システム権限
- 管理権限

特殊タイプを参照する場合は、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

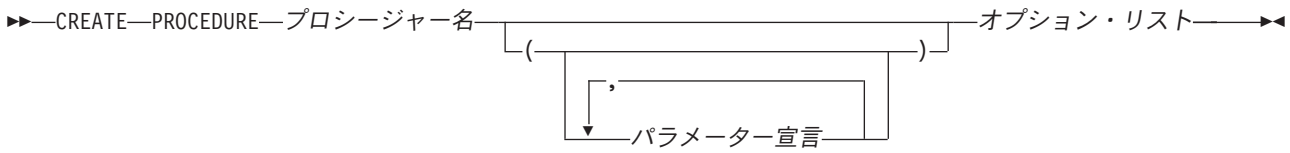
- ステートメント内に識別されているそれぞれの 特殊タイプ に対しては次のもの。
  - その 特殊タイプ に対する USAGE 特権。および
  - その 特殊タイプ が入っているライブラリーに対する *EXECUTE システム権限
- 管理権限。

次のいずれかに該当する場合、ステートメントの権限 ID には、特殊タイプ に対する USAGE 特権が与えられます。

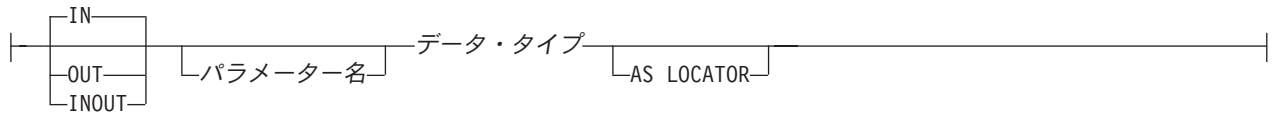
- その特殊タイプの所有者である。
- その特殊タイプに対する USAGE 特権が付与されている。

- その特殊タイプに対するシステム権限の *OBJOPR と *EXECUTE が付与されている。

構文

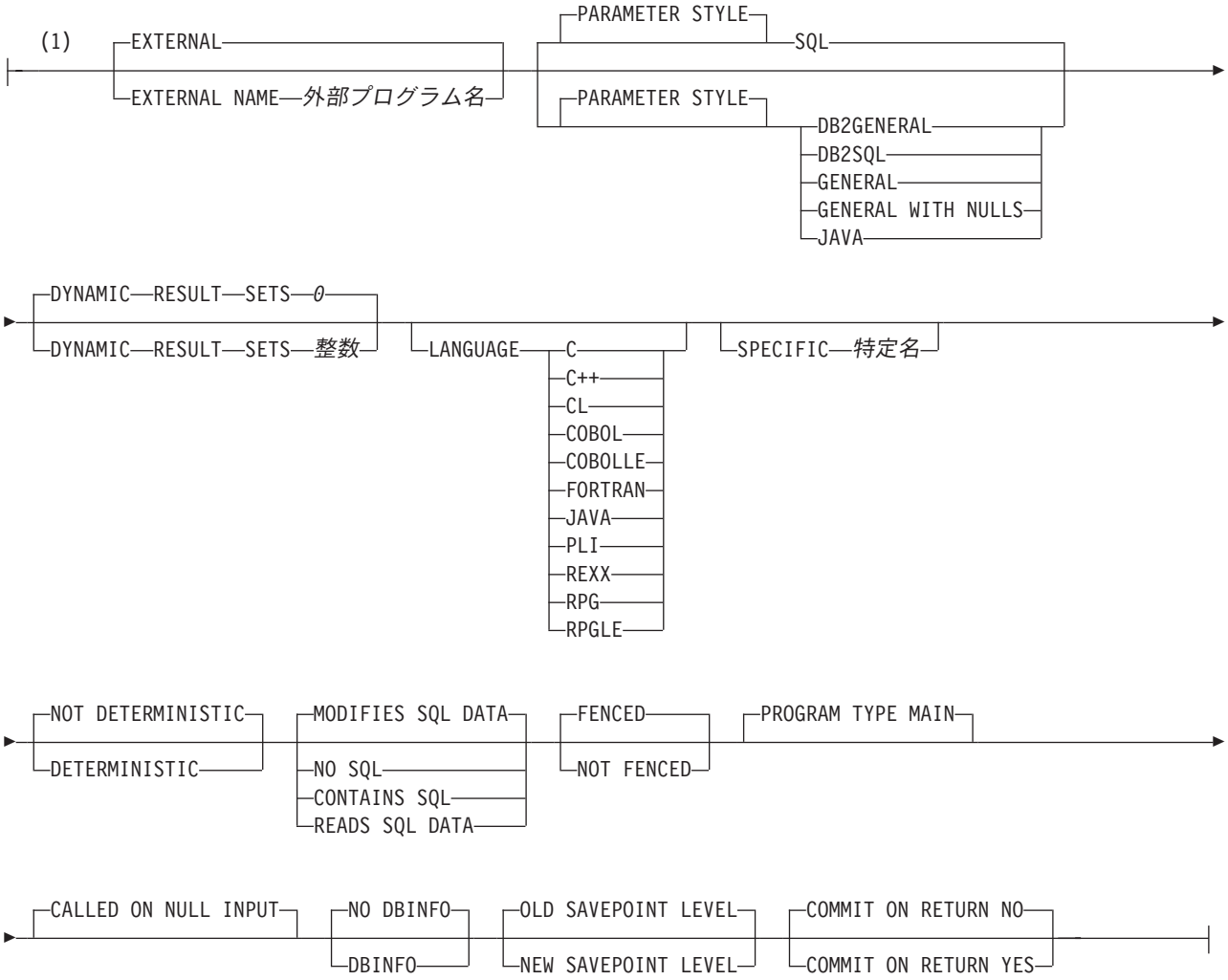


パラメーター宣言:



CREATE PROCEDURE (外部)

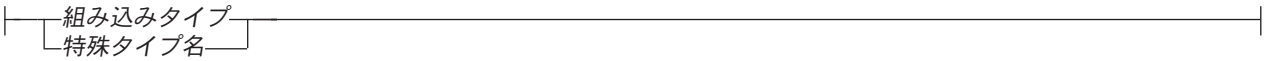
オプション・リスト:



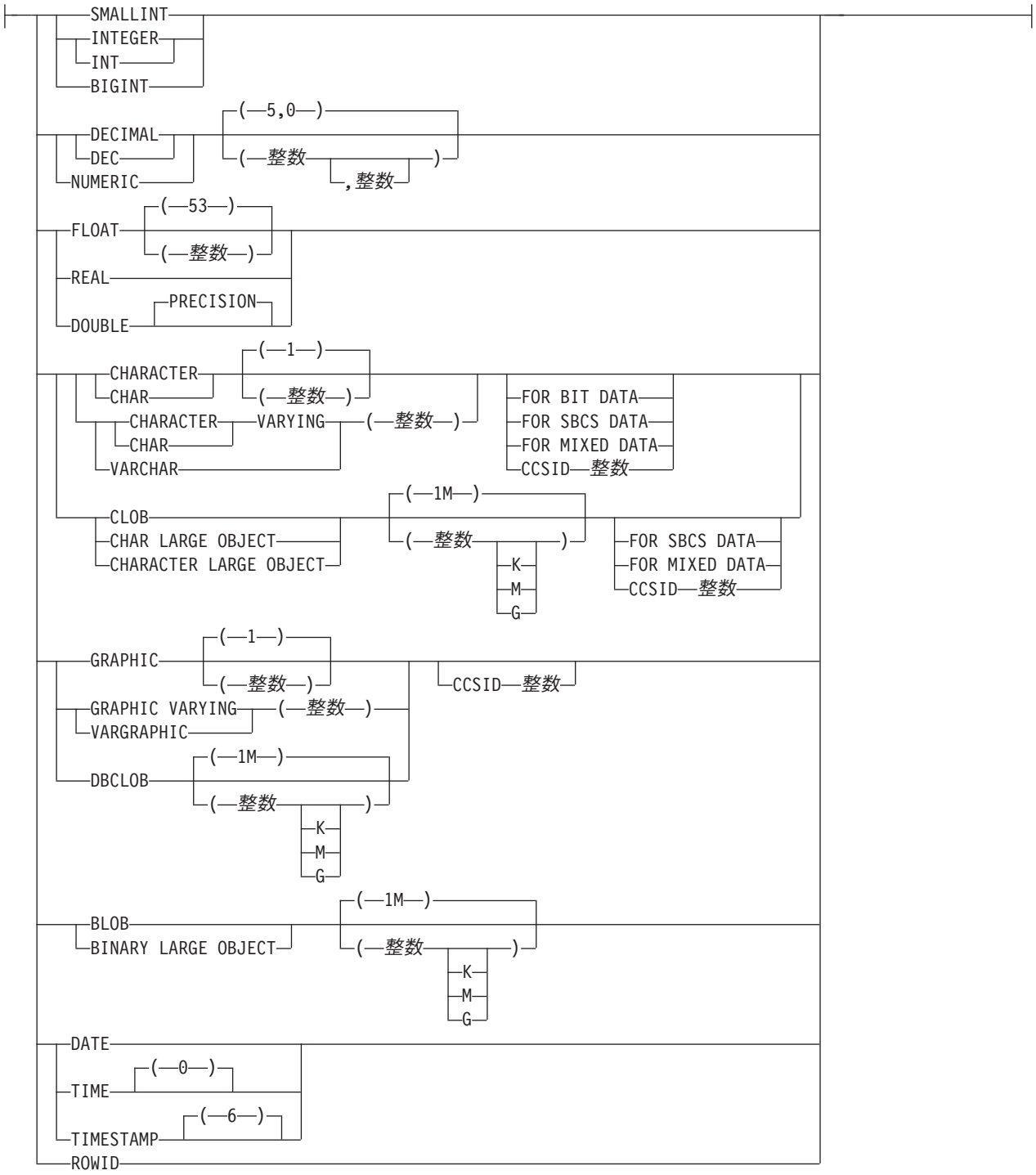
注:

1 オプション文節は、別の順序で指定することができます。

データ・タイプ:



組み込みタイプ:



説明

プロシージャー名
プロシージャーを指定します。名前、スキーマ名、パラメーターの数の組み合わせで、現行サーバーに存在しているプロシージャーを識別してはなりません。

SQL 命名の場合、プロシージャーは、暗黙または明示修飾子で指定されたスキーマ内に作成されます。

システム命名の場合、プロシージャーは、修飾子によって指定されたスキーマ内に作成されます。 修飾子の指定がない場合は、プロシージャーは現行ライブラリー (*CURLIB) に作成されます。

(パラメーター宣言,...)
プロシージャーのパラメーターの数とそれぞれのパラメーターのデータ・タイプを指定します。 プロシージャーに関するパラメーターは、入力専用、出力専用、または入出力両用に使用できます。それぞれのパラメーターに名前を指定することができますが、これは必須ではありません。

CREATE PROCEDURE で使用できるパラメーターの最大数は 255 です。GENERAL WITH NULLS を指定する場合は、最高 254 です。パラメーター・スタイル SQL を指定した場合のパラメーターの許容数は、90 だけです。パラメーターの数の最大数は、その外部プログラムのコンパイルに使用されるライセンス・プログラムで許されるパラメーターの最大の数によっても制約されます。

IN パラメーターが、プロシージャーへの入力パラメーターであることを指定します。プロシージャー内でパラメーターに対する変更が行われても、制御が戻った後で、呼び出し元の SQL アプリケーションがその変更内容を使用することはできません。[50]

OUT
パラメーターが、プロシージャーから戻される出力パラメーターであることを示します。

データ・リンクやデータ・リンクをベースとした特殊タイプは、出力パラメーターとして指定することはできません。

INOUT
パラメーターが、このプロシージャー用の入出力両方のパラメーターであることを指定します。

データ・リンクやデータ・リンクをベースとした特殊タイプは、入出力パラメーターとして指定することはできません。

パラメーター名
パラメーター名を指定します。この名前は、このプロシージャー用の他のパラメーター名 と同じものであってはなりません。

データ・タイプ
パラメーターのデータ・タイプを指定します。

指定するデータ・タイプは LANGUAGE 文節で指定する言語にとって有効なものでなければなりません。データ・リンクは、外部プロシージャーには

50. 言語タイプが REXX の場合、パラメーターは、すべて、入力パラメーターでなければなりません。

無効です。データ・タイプの詳細については、542 ページの 『CREATE TABLE』 および SQL プログラミング 概念を参照してください。

PARAMETER STYLE JAVA を指定する場合は、ラージ・オブジェクト (LOB) データ・タイプのパラメーターはサポートされません。

CCSID が指定されている場合、プロシージャーに渡される前に、パラメーターはその CCSID に変換されます。CCSID が指定されていない場合は、CCSID は、プロシージャーの呼び出し時点における現行サーバーのデフォルトの CCSID によって決まります。

AS LOCATOR
これを指定すると、入力パラメーターは、実際の値ではなく、値のロケーターになります。 AS LOCATOR は、入力パラメーターに LOB データ・タイプや LOB データ・タイプをベースとする特殊タイプが指定されている場合にのみ、指定することができます。

LANGUAGE
その外部プログラムの作成に使用されている言語を指定します。この文節は、外部プログラムが REXX プロシージャーである場合に必要です。

LANGUAGE の指定がない場合は、プロシージャーの作成時点で、該当の外部プログラムに関連するプログラム属性情報から、LANGUAGE を決定します。該当のプログラムに関連するプログラム属性情報では認識可能な言語が識別されない場合、または該当のプログラムが見つからない場合は、言語は C であると見なされます。

C 外部プログラムは C で作成されます。

C++
外部プログラムは C++ で作成されます。

CL
外部プログラムは CL で作成されます。

COBOL
外部プログラムは COBOL で作成されます。

COBOLLE
外部プログラムは ILE COBOL で作成されます。

FORTRAN
外部プログラムは FORTRAN で作成されます。

JAVA
外部プログラムは JAVA で作成されます。

PLI
外部プログラムは PL/I で作成されます。

REXX
外部プログラムは REXX プロシージャーです。

RPG
外部プログラムは RPG で作成されます。

RPGLE
外部プログラムは ILE RPG で作成されます。

PARAMETER STYLE

プロシージャーにパラメーターを渡し、プロシージャーから値を戻すために使用する規則を指定します。

SQL

CALL ステートメントに指定されているパラメーターに加えて、幾つかの追加パラメーターをプロシージャーに渡すことを指定します。これらのパラメーターは、次の順序で配列されるように定義されます。

- 最初の N 個のパラメーターは、CREATE PROCEDURE ステートメント上に指定されるパラメーターです。
- パラメーターの標識変数を表す N 個のパラメーター。
- SQLSTATE の CHAR(5) 出力パラメーター。戻される SQLSTATE は、プロシージャーが成功したかどうかを示します。戻される SQLSTATE は、外部プログラムによって割り当てられたものです。

  ユーザーは、関数からエラーまたは警告を戻すために、外部プログラム内で SQLSTATE を任意の有効な値にセットすることができます。
- 完全修飾プロシージャー名の VARCHAR(517) 入力パラメーター。
- 特定の名前の VARCHAR(128) 入力パラメーター。
- メッセージ・テキストの VARCHAR(70) 出力パラメーター。

渡されるパラメーターについての詳細は、該当するソース・ファイル内の組み込み sqludf を参照してください。例えば、C の場合、sqludf は QSYSINC/H で見つかります。

LANGUAGE JAVA を指定した場合は、PARAMETER STYLE SQL は使用できません。

DB2GENERAL

このプロシージャーに、Java メソッド用として定義されているパラメーター引き渡し規則を使用することを指定します。

PARAMETER STYLE DB2GENERAL を指定できるのは、LANGUAGE JAVA を指定した場合だけです。 Java でのパラメーター引き渡しの詳細については、「Developer Kit for Java」を参照してください。

DB2SQL

CALL ステートメントに指定されているパラメーターに加えて、幾つかの追加パラメーターをプロシージャーに渡すことを指定します。 DB2SQL は、以下の追加パラメーターを最後のパラメーターとして渡すことができるという点以外は、PARAMETER STYLE SQL と同じです。

- DBINFO が CREATE PROCEDURE ステートメント上に指定されている場合は、dbinfo 構造体のパラメーター。

渡されるパラメーターについての詳細は、該当するソース・ファイル内の組み込み sqludf を参照してください。例えば、C の場合、sqludf は QSYSINC/H で見つかります。

LANGUAGE JAVA を指定した場合は、PARAMETER STYLE DB2SQL は使用できません。

GENERAL

このプロシージャーが CALL に指定されているパラメーターを受け取るよ

うなパラメーター引き渡しメカニズムを使用することを指定します。標識変数用の追加の引き数が渡されることはありません。

LANGUAGE JAVA を指定した場合は、PARAMETER STYLE GENERAL は使用できません。

GENERAL WITH NULLS

CALL ステートメントで GENERAL に指定されているパラメーターに加えて、他の引き数もプロシージャーに渡すことを指定します。この追加の引き数には、CALL ステートメントの各パラメーターについてそれぞれ 1 つずつエレメントがある標識配列が含まれています。 C では、これは多くの場合短精度整数の配列です。標識の処理方法に関する詳細については、SQL プログラミング 概念を参照してください。

LANGUAGE JAVA を指定した場合は、PARAMETER STYLE GENERAL は使用できません。

JAVA

このプロシージャーで、Java 言語および SQLJ ルーチンの仕様に準拠するパラメーター引き渡し規則を使用することを指定します。 INOUT および OUT パラメーターは、値を戻しやすくするために、単一項目配列として渡されます。移植性を高めるためには、PARAMETER STYLE JAVA 規則を使用する Java プロシージャーを書く必要があります。

PARAMETER STYLE JAVA を指定できるのは、LANGUAGE JAVA を指定した場合だけです。 Java でのパラメーター引き渡しの詳細については、「Developer Kit for Java」を参照してください。

パラメーターを渡す方法は、外部プロシージャーの言語によって決まります。たとえば、C では、VARCHAR または CHAR パラメーターはヌル文字で終了するストリングとして渡されます。詳細については、SQL プログラミング 概念を参照してください。

EXTERNAL NAME *外部プログラム名*

該当のプロシージャーが CALL ステートメントによって呼び出される時点で実行されるプログラムを指定します。このプログラム名は、プロシージャーの呼び出し時点で該当のサーバーに存在しているプログラムを識別するものでなければなりません。命名オプションが *SYS で、プログラム名が修飾されていない場合は、プロシージャーの呼び出し時点で、ライブラリー・リストを使用して該当のプログラムを検索します。このプログラムは、ILE サービス・プログラムであってはなりません。

この名前の妥当性は、サーバーで検査されます。名前の形式が正しくない場合、エラーが戻されます。

外部プログラム名の指定がない場合、外部プログラム名は該当のプロシージャー名と同じであると見なされます。

この外部プログラムは、プロシージャーの作成時点に存在している必要はありませんが、プロシージャーの呼び出し時点には存在していなければなりません。

CONNECT、 SET CONNECTION、 RELEASE、 DISCONNECT、 COMMIT、 ROLLBACK および SET TRANSACTION ステートメントは、リモート・サー

バー上で実行中のプロシージャー内で使用することはできません。COMMIT および ROLLBACK ステートメントは、ATOMIC SQL プロシージャー内で使用することはできません。

DYNAMIC RESULT SETS *整数*

プロシージャーから戻すことのできる結果セットの最大数を指定します。整数には、ゼロより大か等しい値を指定する必要があります。ゼロを指定すると、結果セットは戻されません。プロシージャーには、任意の数の結果セットを指定することができますが、取り出しを待機中の結果セットを指定できるのは、一度に 100 のプロシージャーだけです。 SET RESULT SETS ステートメントを発行した場合は、戻される結果の数は、このキーワードに指定した結果セットの数と、SET RESULTS SET ステートメントに指定した結果セットの数のいずれか少ない方です。

結果セットはスクロール可能です。結果セットを戻すのにカーソルが使用された場合、結果セットはカーソル位置から始まります。つまり、5 つの FETCH NEXT 操作が実行された後、プロシージャーから戻った場合、結果セットは、結果セットの 6 行目から始まります。

結果セットが戻されるのは、次の両方の条件を満たしている場合に限られます。

- プロシージャーが、iSeries Access の ODBC または JDBC ドライバー、iSeries サーバー上の JDBC、または SQL 呼び出しレベル・インターフェースから呼び出された場合。
- 外部プログラムが ACTGRP(*NEW) の属性を持っていない。

結果セットの詳細については、788 ページの 『SET RESULT SETS』 を参照してください。

SPECIFIC *特定名*

プロシージャーの固有名を指定します。この名前は、暗黙的または明示的にスキーマ名で修飾されます。 スキーマ名も含め、この名前では、現行サーバーに存在している別のプロシージャーまたは関数の特定名を識別することはできません。 修飾されない場合の暗黙の修飾子は、プロシージャー名の修飾子と同じです。 修飾される場合の修飾子は、プロシージャー名の修飾子と同じものにする必要があります。

特定名 を指定しなかった場合、その特定名は、プロシージャー名と同じ名前になります。この特定名の関数やプロシージャーがすでに存在している場合は、固有表名の生成に使用される規則にほぼ準拠した固有名が生成されます。

DETERMINISTIC または **NOT DETERMINISTIC**

このプロシージャーが、同じ IN 引き数および INOUT 引き数を指定して呼び出された場合に、常に同じ結果を戻すかどうかを指定します。

NOT DETERMINISTIC

このプロシージャーは、同じ IN 引き数および INOUT 引き数を指定して呼び出された場合に、データベース内の参照先データが変更されていない限り、常に同じ結果を戻します。

DETERMINISTIC

このプロシージャーは、同じ IN 引き数および INOUT 引き数を指定して呼び出された場合に、データベース内の参照先データが変更されていなくても、必ずしも同じ結果を戻すとは限りません。

CONTAINS SQL、READS SQL DATA、MODIFIES SQL DATA、または NO SQL

SQL ステートメントがある場合に、このプロシージャーまたはこのプロシージャーから呼び出されたルーチンの中で、どの SQL ステートメントを実行できるかを指定します。各データ・アクセス指示の下で実行できる SQL ステートメントの詳細なリストについては、913 ページの『付録 F. SQL ステートメントの特性』を参照してください。

CONTAINS SQL

このプロシージャーで、SQL データの読み取りも変更も行わない SQL ステートメントを実行できることを指定します。

NO SQL

このプロシージャーではどの SQL ステートメントも実行できないことを指定します。

READS SQL DATA

このプロシージャーに、SQL データを変更しない SQL ステートメントを組み込めることを指定します。

MODIFIES SQL DATA

このプロシージャーで、どのプロシージャーでもサポートされないステートメントを除くすべての SQL ステートメントを実行できることを指定します。

CALLED ON NULL INPUT

パラメーター値のどれかがヌルである場合に、このプロシージャーを呼び出すことを指定します。

FENCED または **NOT FENCED**

このパラメーターは、他のプロダクトとの互換性を保持するために許可されており、DB2 UDB for iSeries で使用されることはありません。

PROGRAM TYPE MAIN

このプロシージャーをメイン・ルーチンとして実行することを指定します。

DBINFO

データベース・マネージャーは、状況情報が入っている構造体をプロシージャーに渡す必要があることを指定します。 表 46 は、DBINFO 構造体の説明を示しています。 DBINFO 構造体についての詳しい情報は、QSYSINC.H 内の組み込みファイル SQLUDF に入っています。

DBINFO は、PARAMETER STYLE DB2SQL を指定した場合のみ指定できます。

表 46. DBINFO フィールド

フィールド	データ・タイプ	説明
リレーショナル・データベース	VARCHAR(128)	現行サーバーの名前
権限 ID	VARCHAR(128)	実行時権限 ID

表 46. DBINFO フィールド (続き)

フィールド	データ・タイプ	説明
CCSID 情報	INTEGER INTEGER INTEGER INTEGER CHAR(8)	ジョブの CCSID 情報。CCSID を識別する情報は、次のとおりです。 • SBCS CCSID • DBCS CCSID • 混合 CCSID • 最初の 3 つの CCSID のどれに該当するかの指示。 • 予約済み CREATE PROCEDURE ステートメントのパラメーターの 1 つとして明示的に CCSID を指定していない場合は、入力ストリングは、関数の実行時にジョブの CCSID でコード化されるものと見なされます。入力ストリングの CCSID がパラメーターの CCSID と同じではない場合は、この外部関数に渡される入力ストリングは、外部プログラムの呼び出しの前に変換されます。
ターゲット列	VARCHAR(128) VARCHAR(128) VARCHAR(128)	プロシージャーへの呼び出しには適用されません。
バージョンとリリース	CHAR(8)	データベース・マネージャーのバージョン、リリース、および修正レベル。
プラットフォーム	INTEGER	サーバーのプラットフォーム・タイプ。

OLD SAVEPOINT LEVEL または **NEW SAVEPOINT LEVEL**
このプロシージャーに入ったときに、新しい保管ポイント・レベルを作成するかどうかを指定します。

OLD SAVEPOINT LEVEL
新しい保管ポイント・レベルを作成しません。このプロシージャー内で、OLD SAVEPOINT LEVEL が暗黙的または明示的に指定された SAVEPOINT ステートメントが発行された場合、SAVEPOINT ステートメントはプロシージャーの呼び出し元と同じ保管ポイント・レベルで作成されます。これは、デフォルト値です。

NEW SAVEPOINT LEVEL
このプロシージャーに入ったときに、新しい保管ポイント・レベルが作成されます。プロシージャー内に設定されているすべての保管ポイントは、このプロシージャーが呼び出されたレベルより深くネストされた保管ポイント・レベルで作成されます。したがって、プロシージャー内のどの新規保管ポイントも、同じ名前を持つ上位の保管ポイント・レベル (例えば呼び出し側プログラムの保管ポイント・レベル) で設定されている既存の保管ポイントと競合することはありません。

COMMIT ON RETURN
データベース・マネージャーが、プロシージャーからの戻りと同時にトランザクションをコミットするかどうかを指定します。

NO
データベース・マネージャーは、プロシージャーから戻ったときにコミットを行いません。 NO はデフォルトです。

YES

データベース・マネージャーは、プロシージャーから正常に戻った場合にコミットを行います。プロシージャーの戻り時にエラーがあった場合は、コミットは行われません。

コミット操作の対象には、呼び出し側アプリケーション・プロセスおよびこのプロシージャーが行う作業が含まれます。[51]

プロシージャーが結果セットを戻す場合に、結果セットに関連したカーソルをコミット後に使用できるようにするには、カーソルを WITH HOLD として定義しておく必要があります。

使用上の注意

プロシージャーの作成

ILE 外部プログラムに関連した外部プロシージャーが作成されると、そのプロシージャーの属性を関連のプログラム・オブジェクトに保管しようと試みられます。*PGM オブジェクトが保管された後、このシステムや別のシステムに復元すると、カタログはそれらの属性を使用して自動的に更新されます。

外部プロシージャーの場合は、次の制約の範囲内で属性を保管することができます。

- 外部プログラム・ライブラリーは、SYSIBM、QSYS、または QSYS2 であってはなりません。
- 外部プログラムは、CREATE PROCEDURE ステートメントの発行時に存在していなければなりません。
- 外部プログラムは、ILE *PGM オブジェクトでなければなりません。
- 外部プログラムには、少なくとも 1 つの SQL ステートメントが含まれていることが必要です。

オブジェクトを更新できない場合でも、それにかかわらず、プロシージャーは作成されます。

プロシージャーの復元時には、次のような動作が生じます。

- プロシージャーが初めに作成される時点で特定名が指定されており、しかもその名前が固有でない場合は、エラーが出されます。
- 特定名が指定されていない場合は、必要に応じて固有名が生成されます。
- プロシージャー名とパラメーター数が固有でない場合は、プロシージャーを登録することは不可能で、エラーが出されます。

プロシージャーの呼び出し

DECLARE PROCEDURE ステートメントで、作成されたプロシージャーと同じ名前のプロシージャーを定義し、そのプロシージャー名がホスト変数によって識別されていない静的 CALL ステートメントが、同じソース・プログラムから実行される場合は、CREATE PROCEDURE ステートメントの属性ではなく、DECLARE PROCEDURE ステートメントの属性が使用されます。

---

51. 外部プログラムが ACTGRP(*NEW) を指定して作成されており、ジョブ・コミットメント定義を使用しない場合は、プロシージャーにより行われた作業は、活動化グループ終了に伴ってコミットまたはロールバックされます。

CREATE PROCEDURE ステートメントが適用されるのは、静的および動的 CALL ステートメント、ならびにそのプロシージャー名がホスト変数によって識別されている CALL ステートメントです。

外部プロシージャーが呼び出されると、そのプロシージャーは、外部プログラムの作成時に指定された活動化グループであれば、どの活動化グループ内でも実行します。ただし、通常は、プロシージャーが呼び出しプログラムと同じ活動化グループ内で実行するように ACTGRP(*CALLER) を使用する必要があります。

Java プロシージャーの使用上の注意

Java プロシージャーを実行するためには、システムに Developer Kit for Java (5722-JV1) をインストールしておく必要があります。 インストールされていないと、SQLCODE -443 が戻され、CPDB521 メッセージがジョブ・ログに入ります。

Java プロシージャーの実行中にエラーが発生すると、SQLCODE -443 が戻されます。 エラーによっては、プロシージャーが実行されていたジョブのジョブ・ログに他のメッセージが入っている場合があります。

同義のキーワード

以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- キーワード VARIANT と NOT VARIANT は、NOT DETERMINISTIC と DETERMINISTIC の同義語として使用することができます。
- キーワード NULL CALL と NOT NULL CALL は、CALLED ON NULL INPUT と RETURNS NULL ON NULL INPUT の同義語として使用できます。
- キーワード SIMPLE CALL は、GENERAL の同義語として使用できます。
- DB2GENERAL の同義語として、値 DB2GENRL を使用できます。
- DYNAMIC RESULT SET、RESULT SETS、および RESULT SET は、DYNAMIC RESULT SETS の同義語として使用できます。

例

COBOL プログラムの中に外部プロシージャー PROC1 を作成します。CALL ステートメントを使用してこのプロシージャーを呼び出すと、LIB1 ライブラリーの中の PGM1 という名前の COBOL プログラムが呼び出されます。

```
EXEC SQL
  CREATE PROCEDURE PROC1
          (CHAR(10), CHAR(10))
          EXTERNAL NAME LIB1.PGM1
          LANGUAGE COBOL GENERAL;

EXEC SQL
   CALL PROC1 ('FIRSTNAME ','LASTNAME  ');
```

CREATE PROCEDURE (SQL)

CREATE PROCEDURE (SQL) ステートメントは、現行サーバーで SQL プロシージャーを作成します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。このステートメントは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- SYSPROCS カタログ・ビューと SYSPARMS カタログ表の場合
  - 該当の表に対する INSERT 特権、および
  - QSYS2 ライブラリーに対する *EXECUTE システム権限
- 管理権限

ステートメントの権限 ID は、次の場合に表についての INSERT 特権を持ちます。

- その表の所有者である。
- その表についての INSERT 特権を認可されている、または
- その表についての *OBJOPR および *ADD システム権限が認可されている。

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- 次のシステム権限
  - プログラム作成 (CRTPGM) コマンドに対する *USE
  - プロシージャーが作成されるライブラリーに対する *EXECUTE と *ADD
- 管理権限

SQL 名が指定され、該当のプロシージャーが作成されるライブラリーの名前と同じ名前のユーザー・プロファイルが存在し、しかもその名前がステートメントの権限 ID と異なっている場合、ステートメントの権限 ID によって保持される特権には、少なくとも次の 1 つが含まれていなければなりません。

- その名前を持つユーザー・プロファイルに対する *ADD システム権限
- 管理権限

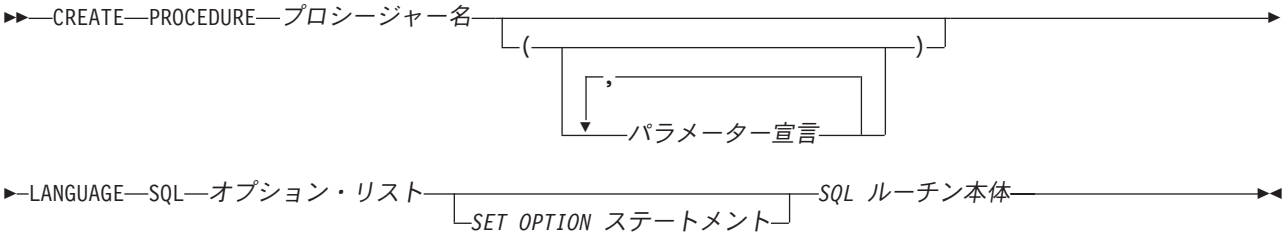
特殊タイプを参照する場合は、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- ステートメント内に識別されているそれぞれの特殊タイプに対しては次のもの。
  - その 特殊タイプ に対する USAGE 特権。および
  - その 特殊タイプ が入っているライブラリーに対する *EXECUTE システム権限
- 管理権限

次のいずれかに該当する場合、ステートメントの権限 ID には、特殊タイプに対する USAGE 特権が与えられます。

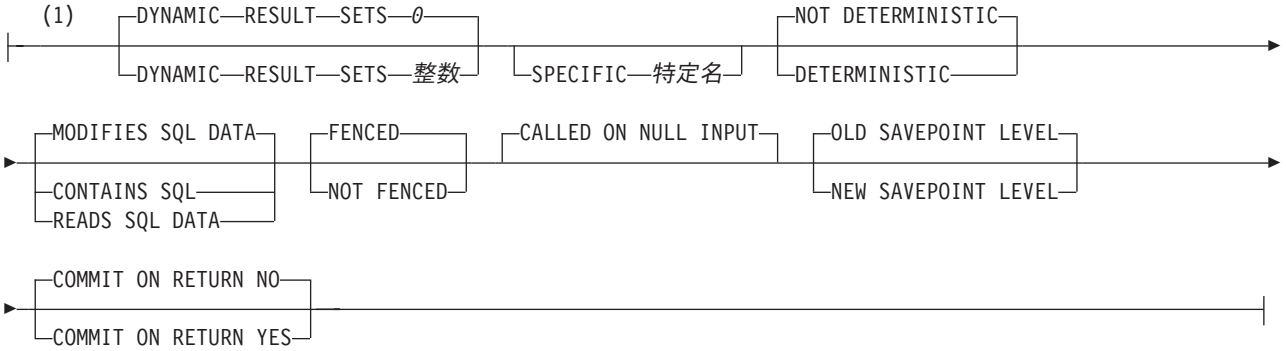
- その特殊タイプの所有者である。
- その特殊タイプに対する USAGE 特権が付与されている。
- その特殊タイプに対するシステム権限の *OBJOPR と *EXECUTE が付与されている。

構文



パラメーター宣言:

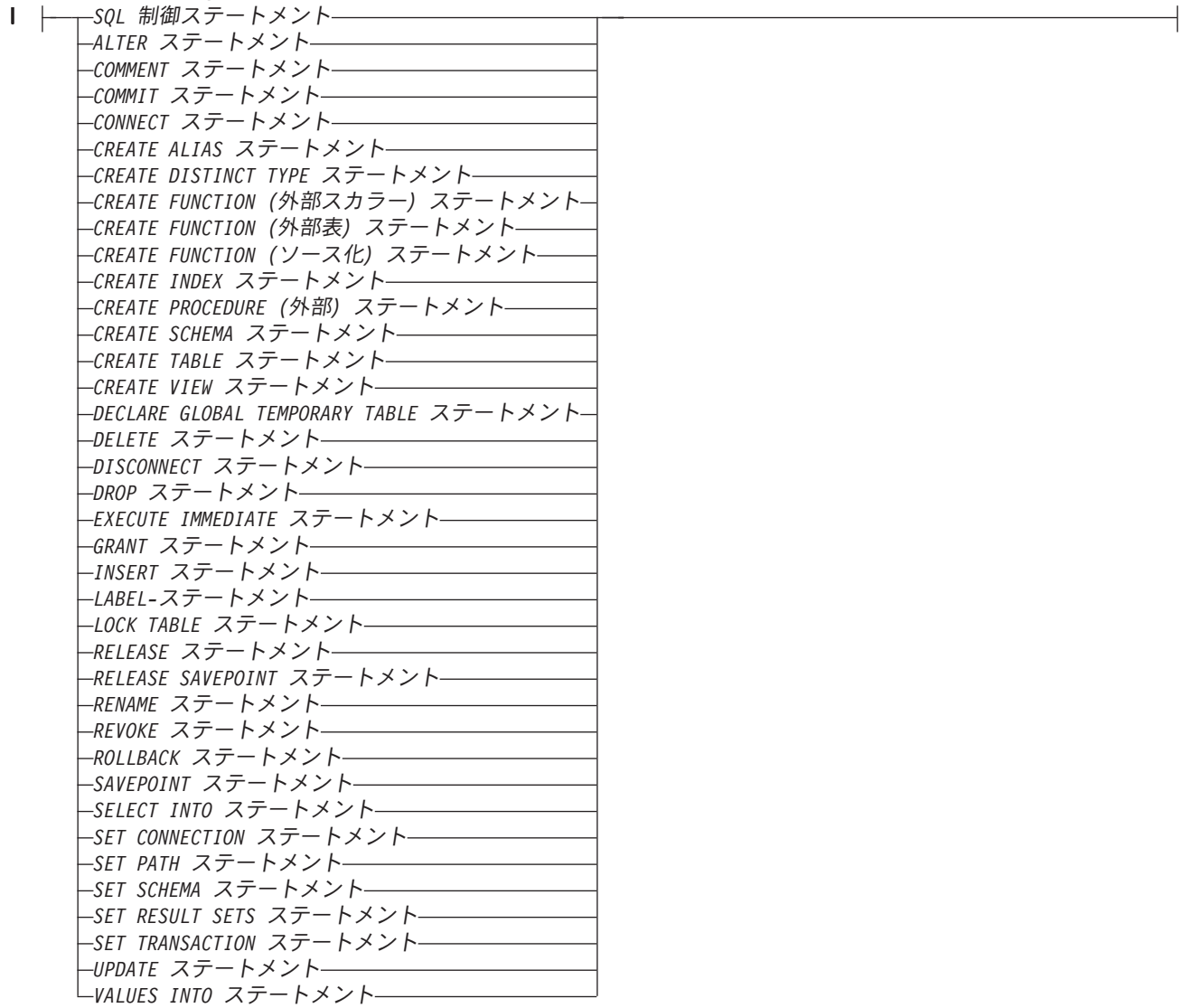
オプション・リスト:



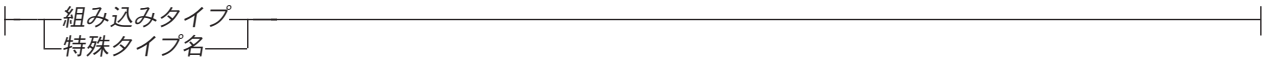
注:

1 オプション文節は、別の順序で指定することができます。

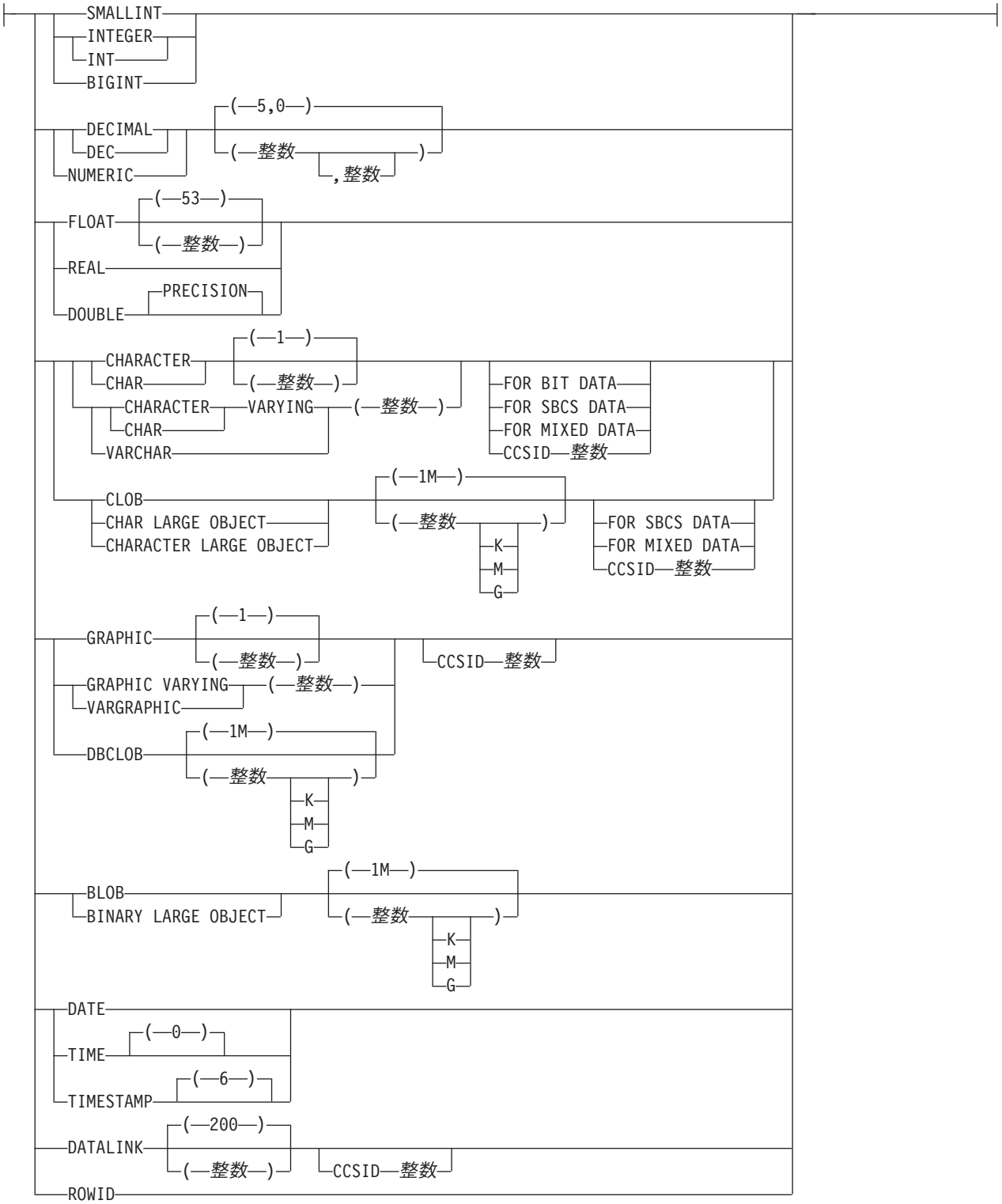
SQL ルーチン本体:


SQL 制御ステートメント
ALTER ステートメント
COMMENT ステートメント
COMMIT ステートメント
CONNECT ステートメント
CREATE ALIAS ステートメント
CREATE DISTINCT TYPE ステートメント
CREATE FUNCTION (外部スカラー) ステートメント
CREATE FUNCTION (外部表) ステートメント
CREATE FUNCTION (ソース化) ステートメント
CREATE INDEX ステートメント
CREATE PROCEDURE (外部) ステートメント
CREATE SCHEMA ステートメント
CREATE TABLE ステートメント
CREATE VIEW ステートメント
DECLARE GLOBAL TEMPORARY TABLE ステートメント
DELETE ステートメント
DISCONNECT ステートメント
DROP ステートメント
EXECUTE IMMEDIATE ステートメント
GRANT ステートメント
INSERT ステートメント
LABEL-ステートメント
LOCK TABLE ステートメント
RELEASE ステートメント
RELEASE SAVEPOINT ステートメント
RENAME ステートメント
REVOKE ステートメント
ROLLBACK ステートメント
SAVEPOINT ステートメント
SELECT INTO ステートメント
SET CONNECTION ステートメント
SET PATH ステートメント
SET SCHEMA ステートメント
SET RESULT SETS ステートメント
SET TRANSACTION ステートメント
UPDATE ステートメント
VALUES INTO ステートメント

データ・タイプ:



組み込みタイプ:



説明

プロシージャー名
プロシージャーを指定します。名前、スキーマ名、パラメーターの数の組み合わせで、現行サーバーに存在しているプロシージャーを識別してはなりません。

SQL 命名の場合、プロシージャーは、暗黙または明示修飾子で指定されたスキーマ内に作成されます。

システム命名の場合、プロシージャーは、修飾子によって指定されたスキーマ内に作成されます。 修飾子の指定がない場合は、プロシージャーは現行ライブラリー (*CURLIB) に作成されます。 現行ライブラリーがない場合、プロシージャーは QGPL 内に作成されます。

(パラメーター宣言,...)
プロシージャーのパラメーターの数とそれぞれのパラメーターのデータ・タイプを指定します。 プロシージャーに関するパラメーターは、入力専用、出力専用、または入出力両用に使用できます。それぞれのパラメーターに名前を指定することができますが、これは必須ではありません。

SQL プロシージャーに許されるパラメーターの最大数は 253 です。

IN パラメーターが、プロシージャーへの入力パラメーターであることを指定します。プロシージャー内でパラメーターに対する変更が行われても、制御が戻った後で、呼び出し元の SQL アプリケーションがその変更内容を使用することはできません。

OUT
パラメーターが、プロシージャーから戻される出力パラメーターであることを示します。 プロシージャー内でこのパラメーターが設定されていない場合は、ヌル値が戻されます。

INOUT
パラメーターが、このプロシージャー用の入出力両方のパラメーターであることを指定します。

パラメーター名
パラメーター名を指定します。この名前は、このプロシージャー用の他のパラメーター名 と同じものであってはなりません。

データ・タイプ
パラメーターのデータ・タイプを指定します。

指定するデータ・タイプは LANGUAGE 文節で指定する言語にとって有効なものでなければなりません。データ・タイプの詳細については、542 ページの 『CREATE TABLE』 および SQL プログラミング 概念 を参照してください。

CCSID が指定されている場合、プロシージャーに渡される前に、パラメーターはその CCSID に変換されます。CCSID が指定されていない場合は、CCSID は、プロシージャーの呼び出し時点における現行サーバーのデフォルトの CCSID によって決まります。

LANGUAGE SQL
これは SQL プロシージャーであることを指定します。

DYNAMIC RESULT SETS *整数*

プロシージャーから戻すことのできる結果セットの最大数を指定します。整数には、ゼロより大か等しい値を指定する必要があります。ゼロを指定すると、結果セットは戻されません。プロシージャーには、任意の数の結果セットを指定することができますが、取り出しを待機中の結果セットを指定できるのは、一度に 100 のプロシージャーだけです。 SET RESULT SETS ステートメントを発行した場合は、戻される結果の数は、このキーワードに指定した結果セットの数と、SET RESULTS SET ステートメントに指定した結果セットの数のいずれか少ない方です。

結果セットはスクロール可能です。結果セットを戻すのにカーソルが使用された場合、結果セットはカーソル位置から始まります。つまり、5 つの FETCH NEXT 操作が実行された後、プロシージャーから戻った場合、結果セットは、結果セットの 6 行目から始まります。

結果セットが戻されるのは、次の条件を満たしている場合に限られます。

- プロシージャーが、iSeries Access の ODBC または JDBC ドライバー、iSeries サーバー上の JDBC、または SQL 呼び出しレベル・インターフェースから呼び出された場合。

結果セットの詳細については、788 ページの 『SET RESULT SETS』を参照してください。

SPECIFIC *特定名*

プロシージャーの固有名を指定します。この名前は、暗黙的または明示的にスキーマ名で修飾されます。 スキーマ名も含め、この名前では、現行サーバーに存在している別のプロシージャーまたは関数の特定名を識別することはできません。 修飾されない場合の暗黙の修飾子は、プロシージャー名の修飾子と同じです。 修飾される場合の修飾子は、プロシージャー名の修飾子と同じものにする必要があります。

特定名 を指定しなかった場合、その特定名は、プロシージャー名と同じ名前になります。この特定名の関数やプロシージャーがすでに存在している場合は、固有表名の生成に使用される規則にほぼ準拠した固有名が生成されます。

DETERMINISTIC または **NOT DETERMINISTIC**

このプロシージャーが、同じ IN 引き数および INOUT 引き数を指定して呼び出された場合に、常に同じ結果を戻すかどうかを指定します。

NOT DETERMINISTIC

このプロシージャーは、同じ IN 引き数および INOUT 引き数を指定して呼び出された場合に、データベース内の参照先データが変更されていない限り、常に同じ結果を戻します。

DETERMINISTIC

このプロシージャーは、同じ IN 引き数および INOUT 引き数を指定して呼び出された場合に、データベース内の参照先データが変更されていなくても、必ずしも同じ結果を戻すとは限りません。

CONTAINS SQL、**READS SQL DATA**、または **MODIFIES SQL DATA**

このプロシージャーまたはこのプロシージャーから呼び出されたルーチンの中で、どの SQL ステートメントを実行できるかを指定します。各データ・アクセス指示の下で実行できる SQL ステートメントの詳細なリストについては、913 ページの『付録 F. SQL ステートメントの特性』を参照してください。

CONTAINS SQL
このプロシージャーで、SQL データの読み取りも変更も行わない SQL ステートメントを実行できることを指定します。

READS SQL DATA
このプロシージャーに、SQL データを変更しない SQL ステートメントを組み込めることを指定します。

MODIFIES SQL DATA
このプロシージャーで、どのプロシージャーでもサポートされないステートメントを除くすべての SQL ステートメントを実行できることを指定します。

CALLED ON NULL INPUT
パラメーター値のどれかがヌルである場合に、このプロシージャーを呼び出すことを指定します。

FENCED または **NOT FENCED**
このパラメーターは、他のプロダクトとの互換性を保持するために許可されており、DB2 UDB for iSeries で使用されることはありません。

SET OPTION ステートメント
プロシージャーを作成するときに使用するオプションを指定します。例えば、デバッグ可能なプロシージャーを作成するときは、次のステートメントを含めることができます。

```
SET OPTION DBGVIEW = *STMT
```

詳しくは、 770 ページの『SET OPTION』を参照してください。

オプション CLOSQLCSR、 CNULRQD、 DFTRDBCOL、 DYNDFTCOL、 NAMING は、CREATE PROCEDURE ステートメントでは使用できません。

OLD SAVEPOINT LEVEL または **NEW SAVEPOINT LEVEL**
このプロシージャーに入ったときに、新しい保管ポイント・レベルを作成するかどうかを指定します。

OLD SAVEPOINT LEVEL
新しい保管ポイント・レベルを作成しません。このプロシージャー内で、OLD SAVEPOINT LEVEL が暗黙的または明示的に指定された SAVEPOINT ステートメントが発行された場合、SAVEPOINT ステートメントはプロシージャーの呼び出し元と同じ保管ポイント・レベルで作成されます。これは、デフォルト値です。

NEW SAVEPOINT LEVEL
このプロシージャーに入ったときに、新しい保管ポイント・レベルが作成されます。プロシージャー内に設定されているすべての保管ポイントは、このプロシージャーが呼び出されたレベルより深くネストされた保管ポイント・レベルで作成されます。したがって、プロシージャー内のどの新規保管ポイントも、同じ名前を持つ上位の保管ポイント・レベル (例えば呼び出し側プログラムの保管ポイント・レベル) で設定されている既存の保管ポイントと競合することはありません。

COMMIT ON RETURN
データベース・マネージャーが、プロシージャーからの戻りと同時にトランザクションをコミットするかどうかを指定します。

NO
データベース・マネージャーは、プロシージャーから戻ったときにコミットを行いません。 NO はデフォルトです。

YES
データベース・マネージャーは、プロシージャーから正常に戻った場合にコミットを行います。プロシージャーの戻り時にエラーがあった場合は、コミットは行われません。

コミット操作の対象には、呼び出し側のアプリケーション・プロセスとこのプロシージャーが行う作業が含まれます。

プロシージャーが結果セットを戻す場合に、結果セットに関連したカーソルをコミット後に使用できるようにするには、カーソルを WITH HOLD として定義しておく必要があります。

SQL ルーチン本体
複合ステートメントも含め、単一の SQL ステートメントを指定します。SQL プロシージャーの定義に関する詳細については、819 ページの『第 6 章 SQL 制御ステートメント』 を参照してください。

CONNECT、 SET CONNECTION、 RELEASE、 DISCONNECT、 COMMIT、 ROLLBACK および SET TRANSACTION ステートメントは、リモート・サーバー上で実行中のプロシージャー内で使用することはできません。COMMIT および ROLLBACK ステートメントは、ATOMIC SQL プロシージャー内で使用することはできません。

使用上の注意

プロシージャーの所有権 : SQL 名を指定した場合は、プロシージャーの*所有者*は、作成したプロシージャーが入れられるスキーマと同じ名前のユーザー・プロファイルです。それ以外の場合は、このステートメントを実行するジョブのユーザー・プロファイルまたはグループ・ユーザー・プロファイルが、プロシージャーの*所有者* になります。

システム名を指定した場合は、プロシージャーの*所有者* は、このステートメントを実行しているジョブのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

プロシージャーの権限 : SQL 名を使用する場合は、プロシージャーは、*PUBLIC に対するシステム権限 *EXCLUDE を使用して作成されます。システム名を使用する場合、プロシージャーは、スキーマの作成権限 (CRTAUT) パラメーターによって決められる *PUBLIC に対する権限を使用して作成されます。

プロシージャーの所有者がグループ・プロファイルのメンバー (GRPPRF キーワード) であり、グループ権限が指定されている (GRPAUT キーワード) 場合は、そのグループ・プロファイルにも、そのプロシージャーに対する権限が与えられます。

プロシージャーの作成

SQL プロシージャーが作成される場合、SQL は、組み込み SQL ステートメントと一緒に C ソース・コードが収められる一時ソース・ファイルを作成します。次いで、CRTPGM コマンドを使用して、プログラム・オブジェクトを作成します。 プログラムの作成に使用される SQL オプションは、CREATE PROCEDURE ステートメントの実行時に有効なオプションです。プログラムは、ACTGRP(*CALLER) を使用して作成します。

SQL プロシージャーが作成された場合、プロシージャーの属性は、作成されたプログラム・オブジェクトに保管されます。*PGM オブジェクトが保管された後、このシステムや別のシステムに復元すると、カタログはそれらの属性を使用して自動的に更新されます。

プロシージャーの復元時には、次のような動作が生じます。

- プロシージャーが初めに作成される時点で特定名が指定されており、しかもその名前が固有でない場合は、エラーが出されます。
- 特定名が指定されていない場合は、必要に応じて固有名が生成されます。
- プロシージャー名とパラメーター数が固有でない場合は、プロシージャーを登録することは不可能で、エラーが出されます。

プロシージャー名は、それが有効なシステム名である場合は、ソース・ファイルのメンバーの名前、ならびに、プログラム・オブジェクトの名前として使用されます。プロシージャー名が有効なシステム名でない場合は、固有名が生成されます。同じ名前のソース・ファイル・メンバーがすでに存在している場合は、そのメンバーがオーバーレイされます。同じ名前のモジュールやプログラムがすでに存在している場合、オブジェクトはオーバーレイされずに、固有名が生成されます。これらの固有名は、システム表名の生成に関する規則に従って生成されます。

プロシージャーの呼び出し

DECLARE PROCEDURE ステートメントで、作成されたプロシージャーと同じ名前のプロシージャーを定義し、そのプロシージャー名がホスト変数によって識別されていない静的 CALL ステートメントが、同じソース・プログラムから実行される場合は、CREATE PROCEDURE ステートメントの属性ではなく、DECLARE PROCEDURE ステートメントの属性が使用されます。

CREATE PROCEDURE ステートメントが適用されるのは、静的および動的 CALL ステートメント、ならびにそのプロシージャー名がホスト変数によって識別されている CALL ステートメントです。

SQL プロシージャーは、SQL CALL ステートメントを使用して呼び出す必要があります。SQL プロシージャーは、呼び出されると、呼び出しプログラムの活動化グループ内で実行します。

同義のキーワード

以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- キーワード VARIANT と NOT VARIANT は、NOT DETERMINISTIC と DETERMINISTIC の同義語として使用することができます。

- キーワード NULL CALL と NOT NULL CALL は、CALLED ON NULL INPUT と RETURNS NULL ON NULL INPUT の同義語として使用できます。
- DYNAMIC RESULT SET、RESULT SETS、および RESULT SET は、DYNAMIC RESULT SETS の同義語として使用できます。

例

SQL プロシージャーの定義を作成します。このプロシージャーは、従業員番号と昇給の乗数を、入力として受け入れます。プロシージャー本体では、次のタスクが実行されます。

- 従業員の新規の給与を計算する。
- 新規の給与値を使用して、従業員表を更新する。

```
EXEC SQL
  CREATE PROCEDURE UPDATE_SALARY_1
          (IN EMPLOYEE_NUMBER CHAR(10),
           IN RATE DECIMAL(6,2))
          LANGUAGE SQL
          MODIFIES SQL DATA
          UPDATE EMP
            SET SALARY = SALARY + RATE
            WHERE EMPNO = EMPLOYEE_NUMBER
```

CREATE SCHEMA

CREATE SCHEMA ステートメントは、現行サーバーにスキーマを定義し、オプションとして、表、ビュー、別名、索引、および特殊タイプを作成します。 コメントとラベルは、表、ビュー、別名、索引、列、および特殊タイプのカタログ記述内に加えることができます。表、ビュー、および特殊タイプの特権をユーザーに与えることが可能です。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。このステートメントは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- 次の CL コマンドに対する *USE システム権限
  - ライブラリー作成 (CRTLIB)
  - WITH DATA DICTIONARY を指定する場合は、データ・ディクショナリー作成 (CRTDTADCT)
- 管理権限

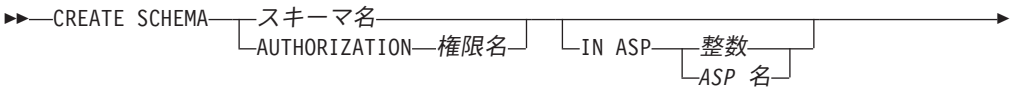
このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

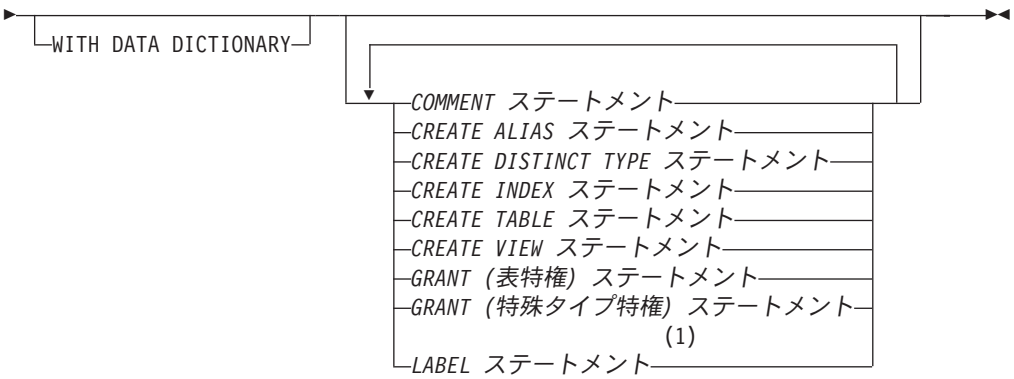
- CREATE SCHEMA ステートメントに含まれている各 SQL ステートメントについて定義されている特権
- 管理権限

AUTHORIZATION 文節の指定がある場合、そのステートメントの権限 ID によって保持される特権には、少なくとも次の 1 つが含まれていなければなりません。

- 権限名によって識別されるユーザー・プロファイルに対する *ADD システム権限
- 管理権限

構文





注:

1 パッケージ、プロシージャー、関数、およびパラメーターに対するラベルとコメントは、CREATE SCHEMA ステートメントではサポ ートされていません。

説明

スキーマ名
スキーマの名前を指定します。スキーマは、この名前で作成されます。スキーマ名を指定すると、このステートメントの権限 ID は、実行時権限 ID になります。この名前は、現行サーバーにある既存のスキーマの名前と同じであってはなりません。 スキーマの*所有者* は、ステートメントを実行するジョブのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

スキーマの所有者がグループ・プロファイルのメンバー (GRPPRF キーワード) であり、グループ権限が指定されている (GRPAUT キーワード) 場合は、そのグループ・プロファイルにも、そのビューに対する権限が与えられます。

権限名
ステートメントの権限 ID を示します。この権限名は、スキーマ名でもあります。この名前は、現行サーバーにある既存のスキーマの名前と同じであってはなりません。

IN ASP *整数*
スキーマを作成したい補助記憶域プール (ASP) を指定します。1 から 32 までの整数を指定します。 1 を指定した場合、スキーマはシステム ASP に作成されます。この文節を省略すると、1 の ASP が想定されます。

IN ASP *ASP 名*
スキーマを作成したい補助記憶域プール (ASP) を指定します。この名前は、現行サーバーに存在する補助記憶域プールを示すものでなければなりません。

WITH DATA DICTIONARY
この文節を指定した場合は、IDDU データ・ディクショナリーがスキーマに作成されます。

データ・ディクショナリーを指定して作成されたスキーマには、LOB 列や DATALINK 列を含む表を入れることはできません。

COMMENT ステートメント
表、ビュー、または列のカタログ記述へのコメントの付加または置き換えを行い

ます。パッケージについてのコメントは許されません。412 ページの『COMMENT』の COMMENT ステートメントの節を参照してください。

CREATE ALIAS ステートメント
別名を作成してスキーマ内に入れます。436 ページの CREATE ALIAS ステートメントの節を参照してください。

CREATE DISTINCT TYPE ステートメント
ユーザー定義の特殊タイプを作成してスキーマ内に入れます。439 ページの『CREATE DISTINCT TYPE』の CREATE DISTINCT TYPE ステートメントの節を参照してください。

CREATE INDEX ステートメント
索引を作成してスキーマ内に入れます。508 ページの CREATE INDEX ステートメントの節を参照してください。

CREATE TABLE ステートメント
表を作成してスキーマ内に入れます。542 ページの CREATE TABLE ステートメントの節を参照してください。

CREATE VIEW ステートメント
ビューを作成してスキーマ内に入れます。589 ページの CREATE VIEW ステートメントの節を参照してください。

GRANT (表特権) ステートメント
このスキーマの表およびビューに対する特権を与えます。 695 ページの『GRANT (表特権)』の GRANT ステートメントの節を参照してください。

GRANT (特殊タイプ特権) ステートメント
スキーマ内の特殊タイプに対する特権を与えます。 681 ページの 『GRANT (特殊タイプ特権)』の GRANT ステートメントの節を参照してください。

LABEL ステートメント
該当のスキーマの中の表、ビュー、または列のカタログ記述のラベルの付加または置き換えを行います。パッケージについてのラベルは許されません。 714 ページの LABEL ステートメントの項を参照してください。

使用上の注意

スキーマの属性 : スキーマは以下のものとして作成されます。

- ライブラリー: ライブラリーは、関連オブジェクトをグループ化し、名前でオブジェクトを見つけることができるようにします。
- カタログ: カタログには、そのスキーマの表、ビュー、索引、およびパッケージの記述が入ります。カタログは、一連のビュー、および WITH DATA DICTIONARY の指定がある場合には IDDU データ・ディクショナリーから構成されます。詳細については、SQL プログラミング 概念 を参照してください。
- ジャーナルおよびジャーナル・レシーバー: ジャーナル QSQJRN とジャーナル・レシーバー QSQJRN0001 がスキーマ内に作成され、以後にスキーマ内に作成されるすべての表への変更を記録するのに使用されます。詳細については、iSeries Information Center のジャーナル管理トピックを参照してください。

分散表に対して作成される索引は、この表が配布されるサーバーのすべてで作成されます。分散表の詳細については、DB2 UDB for iSeries マルチ・システムを参照してください。

オブジェクトの所有権 : 作成したオブジェクトの所有権は以下のように決定されます。

- AUTHORIZATION 文節が指定されている場合は、指定された権限 ID が、このステートメントによって作成されたすべてのオブジェクトを所有します。
- AUTHORIZATION 文節の指定がなく、SQL 名が指定されている場合は、このステートメントによって作成されたすべてのオブジェクトの*所有者*は、スキーマ名と同じ名前を持つユーザー・プロファイル (その名前のユーザー・プロファイルが存在している場合) になります。
- これら以外の場合、このステートメントによって作成されるすべてのオブジェクトの*所有者*は、このステートメントを実行するジョブのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

オブジェクトの権限 : SQL 名を使用する場合は、スキーマおよびその他のオブジェクトは、*PUBLIC に対するシステム権限 *EXCLUDE を使用して作成され、作成権限パラメーター CRTAUT(*EXCLUDE) を指定してライブラリーが作成されます。スキーマに関する権限を持つユーザーは、スキーマの所有者だけです。他のユーザーがそのスキーマに対する権限を必要とする場合、スキーマの所有者は、CL コマンドのオブジェクト権限付与 (GRTOBJAUT) を使用して、作成したオブジェクトに対する権限を与えることができます。

システム名を使用する場合は、スキーマおよびその他のオブジェクトの作成時に *PUBLIC に与えられるシステム権限は、システム値 QCRTAUT によって決まり、ライブラリーは CRTAUT(*SYSVAL) を指定して作成されます。システム・セキュリティーについての詳細は、「iSeries 機密保護解説書」  、および「SQL プログラミングの概念」を参照してください。

スキーマの所有者がグループ・プロファイルのメンバー (GRPPRF キーワード) であり、グループ権限が指定されている (GRPAUT キーワード) 場合は、そのグループ・プロファイルにも、そのスキーマに対する権限が与えられます。

オブジェクト名 : CREATE TABLE、CREATE INDEX、CREATE ALIAS、CREATE DISTINCT TYPE、または CREATE VIEW ステートメントに、作成中の表、索引、別名、特殊タイプ、またはビューの修飾名を指定する場合は、その修飾名の中のスキーマ名には、作成中のスキーマと同じ名前を指定する必要があります。スキーマ定義で参照されるその他の表またはビュー名は、どのようなスキーマ名で修飾されていても構いません。どの SQL ステートメントにおいても、非修飾の表、索引、別名、特殊タイプ、またはビューの名前は、作成されるスキーマの名前で暗黙的に修飾されます。

SQL ステートメント相互間には、区切り記号を使用しません。

SQL ステートメント長 : CREATE SCHEMA ステートメント内の個々の CREATE TABLE、 CREATE INDEX、 CREATE DISTINCT TYPE、 CREATE VIEW、 COMMENT、 LABEL、または GRANT ステートメントの最大長は、どれも 65536 です。

キーワードの同義語 : 旧リリースとの互換性を維持するために、SCHEMA の同義語として COLLECTION キーワードを使用できます。これは標準キーワードではないので、使用しないようにしてください。

例

例 1

在庫部品表と部品番号に関する索引をもつスキーマを作成します。ユーザー・プロファイル JONES に、スキーマに対する権限を与えています。

```
CREATE SCHEMA INVENTORY

CREATE TABLE PART (PARTNO SMALLINT NOT NULL,
                   DESCR  VARCHAR(24),
                   QUANTITY INT)

CREATE INDEX PARTIND ON PART (PARTNO)

GRANT ALL ON PART TO JONES
```

例 2

SMITH の権限 ID を使用してスキーマを作成します。学生番号列に、コメントを付け、学生表を作成します。

```
 CREATE SCHEMA AUTHORIZATION SMITH

CREATE TABLE SMITH.STUDENT (STUDNBR SMALLINT NOT NULL UNIQUE,
                            LASTNAME CHAR(20),
                            FIRSTNAME CHAR(20),
                            ADDRESS CHAR(50))

COMMENT ON STUDENT (STUDNBR IS 'THIS IS A UNIQUE ID#')
```

CREATE TABLE

CREATE TABLE ステートメントは、現行サーバーで表を定義します。この定義には、その表の名前、およびその表の列の名前と属性を含める必要があります。この定義には、基本キーなど、表の他の属性も含めることができます。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。このステートメントは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- 次のシステム権限
  - 物理ファイル作成 (CRTPF) コマンドに対する *USE 権限
  - その表が作成されるライブラリーに対する *EXECUTE および *ADD 権限
  - ジャーナルに対する *OBJOPR および *OBJMGT 権限
  - データ・ディクショナリー に対する *CHANGE 権限。ただし、表が作成されるライブラリーが、データ・ディクショナリーをもつ SQL スキーマの場合。
- 管理権限

SQL 名が指定され、該当の表が作成されるライブラリーの名前と同じ名前のユーザー・プロファイルが存在し、しかもその名前がステートメントの権限 ID と異なっている場合、ステートメントの権限 ID によって保持される特権には、少なくとも次の 1 つが含まれていなければなりません。

- その名前を持つユーザー・プロファイルに対する *ADD システム権限
- 管理権限

外部キーを定義する場合、ステートメントの権限 ID が保持する特権には、親表に関して少なくとも次の 1 つが含まれていなければなりません。

- 該当の表に対する REFERENCES 特権またはオブジェクト管理権限。
- 指定された親キーの各列に対する REFERENCES 特権。
- その表の所有権。
- 管理権限。

次のいずれかに該当する場合は、ステートメントの権限 ID には、表に対する REFERENCES 特権があります。

- その表の所有者である。
- その表に対する REFERENCES 特権が認可されている。
- その表に対する *OBJREF か *OBJMGT のどちらかのシステム権限が認可されている。

次のいずれかに該当する場合は、ステートメントの権限 ID に、表の列に対する REFERENCES 特権が与えられます。

- その表の所有者である。
- その列に対する REFERENCES 特権が与えられている。
- その列に対するシステム権限の *OBJREF が与えられているか、あるいは、その表に対するシステム権限の *OBJMGT が与えられている。

LIKE 文節または AS 選択ステートメントを指定する場合は、このステートメントの権限 ID が保持する特権には、これらの文節で指定する表またはビューに対する次の特権の少なくとも 1 つが含まれていなければなりません。

- 表またはビューに対する SELECT 特権
- 表またはビューの所有権
- 管理権限

ステートメントの権限 ID は、以下の場合に表に対する SELECT 特権を持ちます。

- その表の所有者である。
- その表に対する SELECT 特権が認可されているか、または
- その表についての *OBJOPR および *READ システム権限が認可されている。

ステートメントの権限 ID は、以下の場合にビューに対する SELECT 特権を持ちます。

- そのビューの所有者である。
- そのビューに対する SELECT 特権が認可されているか、または
- そのビューについての *OBJOPR および *READ システム権限、およびそのビューが直接または間接に従属しているすべての表やビューについての *READ システム権限が認可されている。すなわち、そのビューの定義で参照されている表やビューのすべて、またそのようなビューが参照される場合、そのビューの定義で参照されている表やビューのすべて、以下同様。

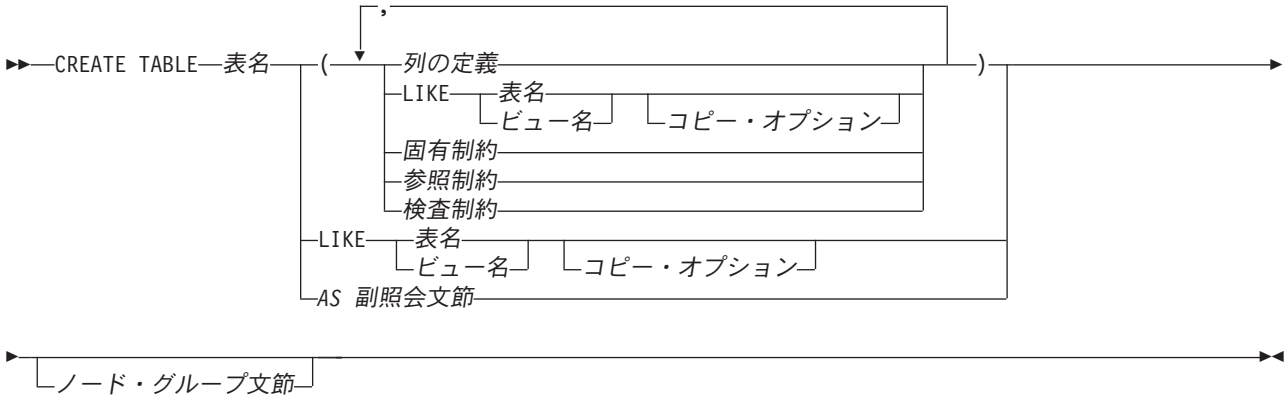
特殊タイプを参照する場合は、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- ステートメント内に識別されているそれぞれの 特殊タイプ に対しては次のもの。
  - その 特殊タイプ に対する USAGE 特権。および
  - その 特殊タイプ が入っているライブラリーに対する *EXECUTE システム権限
- 管理権限

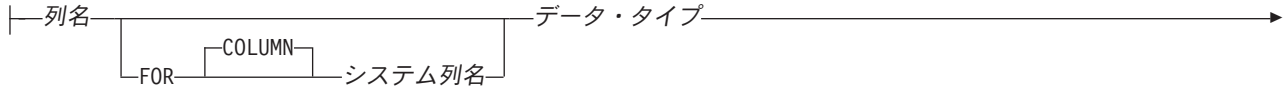
次のいずれかに該当する場合、ステートメントの権限 ID には、特殊タイプ に対する USAGE 特権が与えられます。

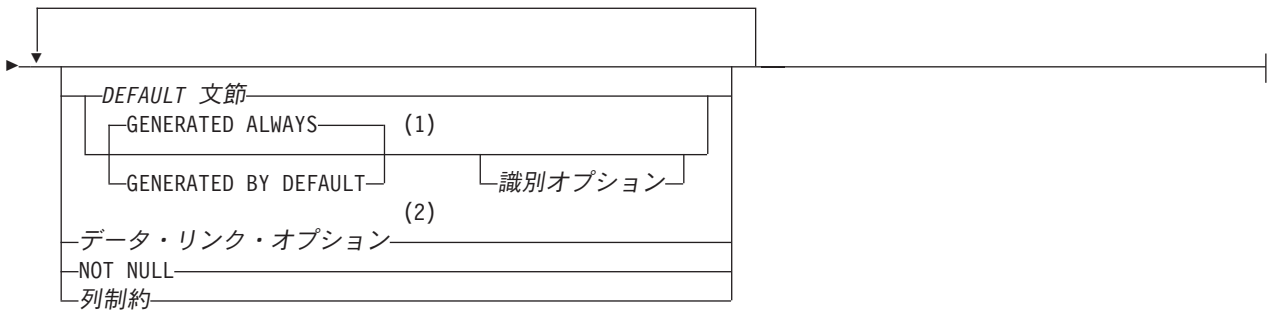
- その 特殊タイプ の所有者である。
- その 特殊タイプ に対する USAGE 特権が付与されている。
- その 特殊タイプ に対するシステム権限の *OBJOPR と *EXECUTE が付与されている。

構文



列定義:



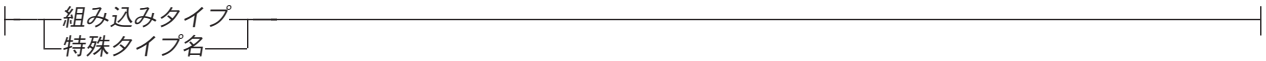


注:

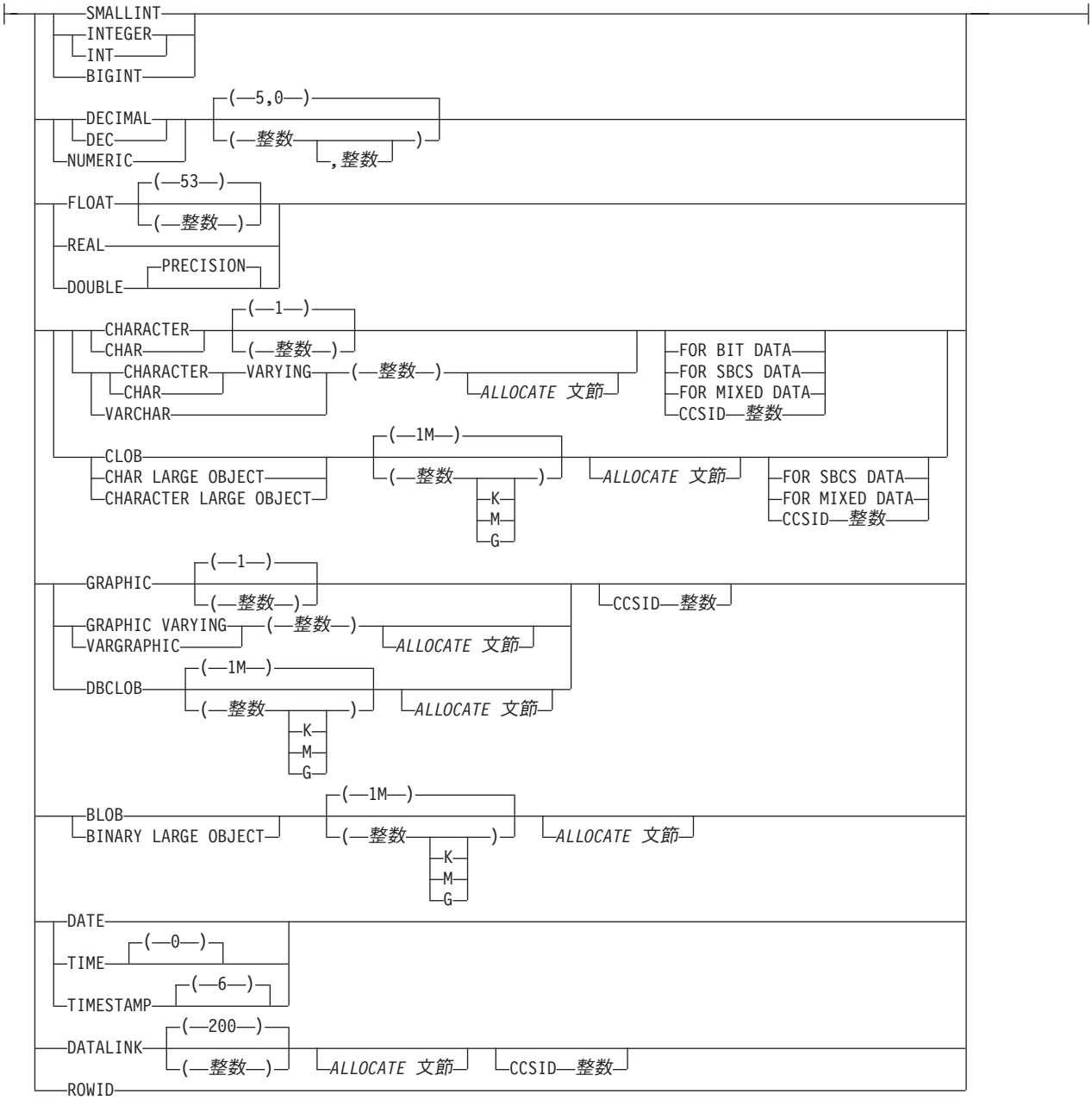
1 GENERATED を指定できるのは、列のデータ・タイプが ROWID (また は ROWID データ・タイプに基づく特殊タイプ) であるか、または列が識別列である場合のみです。

2 データ・リンク・オプションは、DATALINK、および DATALINK をソ ースとする特殊タイプに対してのみ指定することができます。

データ・タイプ:

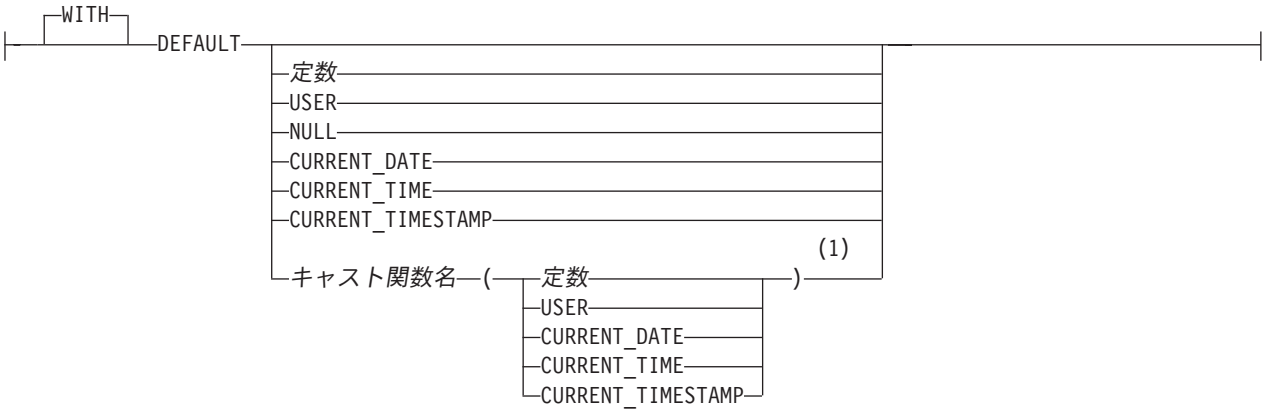


組み込みタイプ:

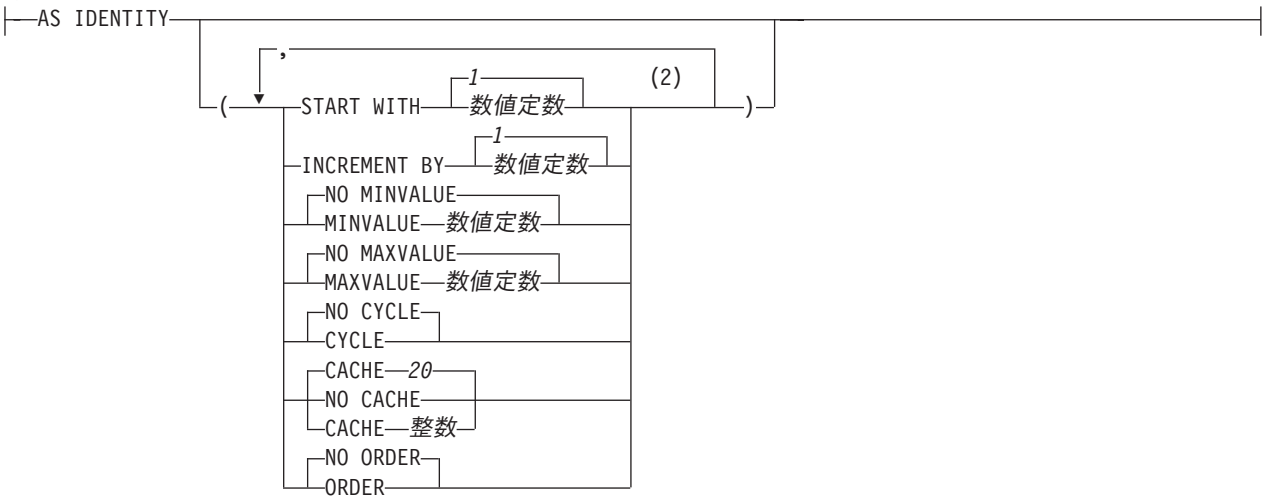


ALLOCATE 文節:

DEFAULT 文節:



識別オプション:

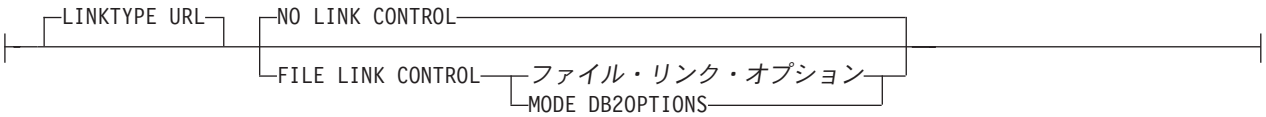


注:

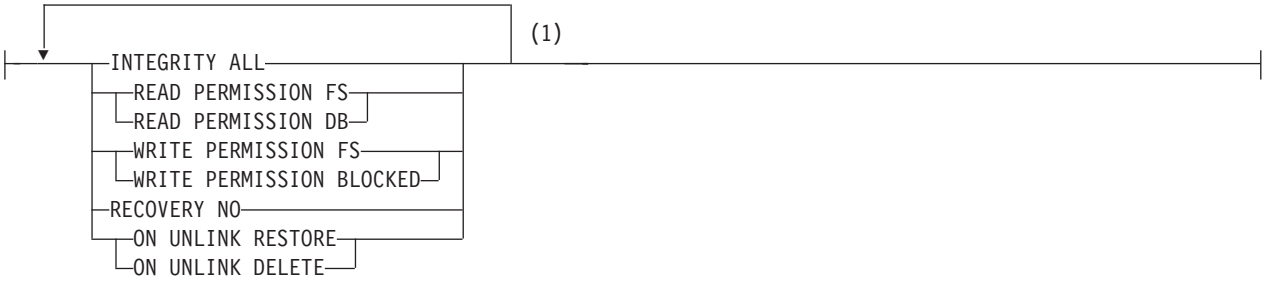
1 この形式の DEFAULT 値は、特殊タイプとして定義されている列だけでしか使用できません。

2 各文節はそれぞれ 1 回のみ指定できます。

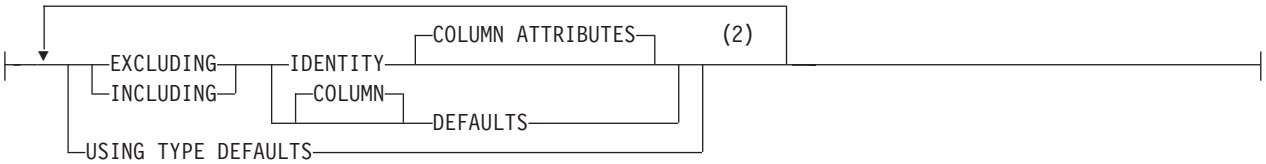
データ・リンク・オプション:



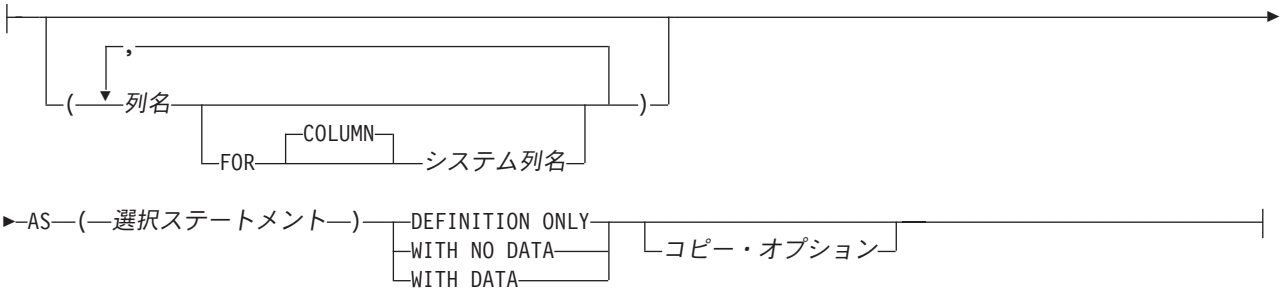
ファイル・リンク・オプション:



コピー・オプション:



AS 副照会文節:



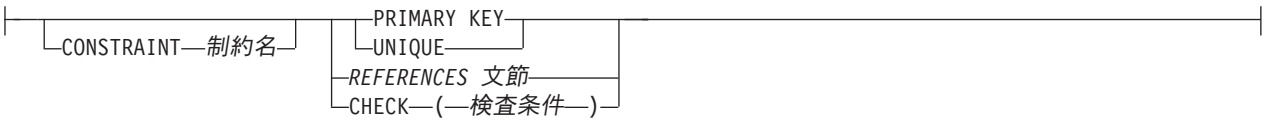
注:

1 5 つのファイル・リンク・オプション をすべて指定する必要がありますが、指定する順序は任意です。

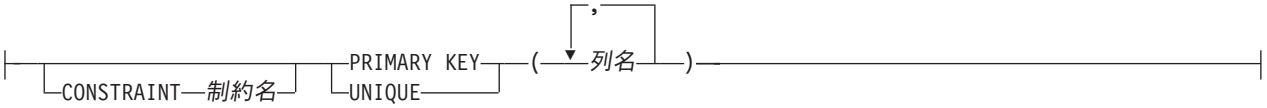
2 各文節はそれぞれ 1 回のみ指定できます。

CREATE TABLE

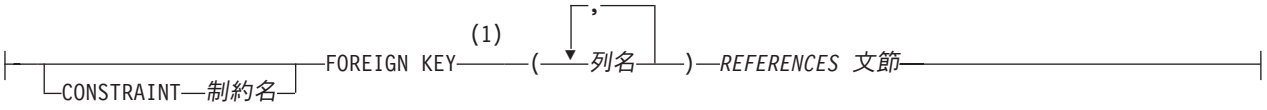
列制約:



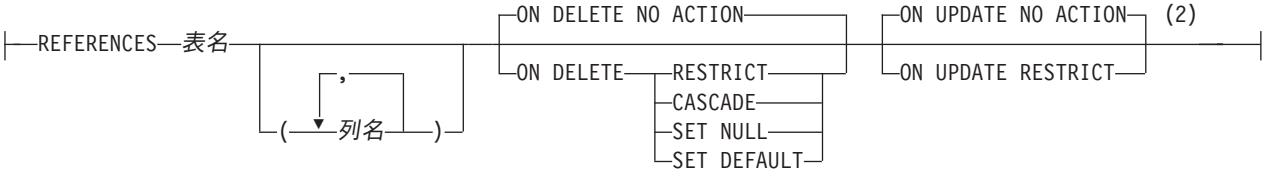
固有制約:



参照制約:

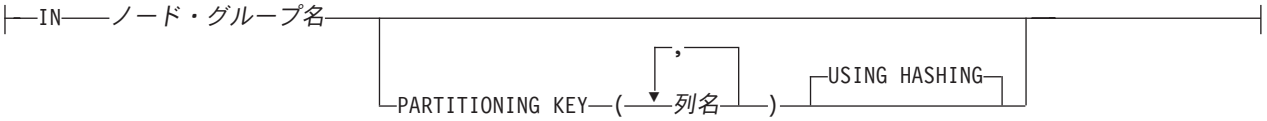


REFERENCES 文節:



検査制約:

ノード・グループ文節:



注:

1 他のプロダクトとの互換性を保持するために、FOREIGN KEY の後に*制約名* (CONSTRAINT キーワードなし) を指定することもできます。

2 ON DELETE と ON UPDATE 文節は、どの順序で指定しても構いません。

説明

表名

表の名前を指定します。 暗黙的または明示的修飾子も含め、この名前は、現行サーバーにすでに存在している索引、表、ビュー、別名、またはファイルと同じ名前にすることはできません。

SQL 名が指定されている場合、表は、暗黙的または明示的修飾子で指定しているスキーマ内に作成されます。

システム名が指定されている場合、表名は、修飾子で指定しているスキーマ内に作成されます。修飾されない場合の表は、現行ライブラリー (*CURLIB) 内に作成されます。現行ライブラリーがない場合、表は QGPL 内に作成されます。

列の定義

列の属性を定義します。少なくとも 1 つ以上で、8000 を超えない列の定義がなければなりません。

列の行バッファー・バイト・カウントの合計は、32766 (VARCHAR 列または VARGRAPHIC 列が指定されている場合は 32740) 以下でなければなりません。さらに、LOB を指定してある場合は、挿入または更新の時点で、すべての列の行データ・バイト・カウントの合計が 3 758 096 383 を超えていてはなりません。 データ・タイプごとの列のバイト・カウントについては、 569 ページの『使用上の注意』を参照してください。

列名
表を構成する列の名前を指定します。*列名* は修飾できません。表の複数の列や表のシステム列名に同じ名前を使用することもできません。

FOR COLUMN *システム列名*
列の OS/400 名を指定します。表の複数の列やシステム列名に対して、同じ名前を使用してはなりません。

システム列名が指定されず、また列名が有効なシステム列名でない場合には、システム列名が生成されます。システム列名の生成方法に関する詳細については、572 ページの 『列名の生成の規則』を参照してください。

データ・タイプ
列のデータ・タイプを指定します。

組み込みタイプ
組み込みタイプ には以下のいずれかを使用します。

SMALLINT
短整数を示します。

INTEGER または **INT**
長整数を示します。

BIGINT
64 ビット整数を示します。

DECIMAL(*整数***,***整数***)** または **DEC(***整数***,***整数***)**
DECIMAL(*整数***)** または **DEC(***整数***)**
DECIMAL または **DEC**
パック 10 進数を示します。最初の整数は数値の精度、つまり総桁数であり、1 から 31 までの範囲で指定できます。2 番目の整数は、数値の位取り (小数点の右側に置く桁数) です。位取りは、0 からその数値の精度までの範囲で指定できます。

DECIMAL(*p*,0) は、DECIMAL(*p*) と指定でき、また DECIMAL(5,0) は、DECIMAL と指定できます。

NUMERIC(*整数***,***整数***)**
NUMERIC(*整数***)**

NUMERIC
ゾーン 10 進数を示します。最初の整数は数値の精度、つまり総桁数であり、1 から 31 までの範囲で指定できます。2 番目の整数は、数値の位取り (小数点の右側に置く桁数) です。位取りは、0 からその数値の精度までの範囲で指定できます。

NUMERIC(*p*) を NUMERIC(*p*,0) の代わりに、NUMERIC を NUMERIC(5,0) の代わりに使用しても構いません。

FLOAT
倍精度の浮動小数点数を示します。

FLOAT(*整数***)**
指定する整数の値によって、単精度、または倍精度の浮動小数点数を示します。整数の値は、1 から 53 までの範囲になければなりません。1 から 24 までの値は単精度、25 から 53 までの値は倍精度を示します。デフォルト値は 53 です。

REAL
単精度の浮動小数点数を示します。

DOUBLE PRECISION または **DOUBLE**
倍精度の浮動小数点数を示します。

CHARACTER(*整数***)** または**CHAR(***整数***)**
CHARACTERまたは **CHAR**
整数 で指定した長さの固定長文字ストリングを示します。整数として指定できる値の範囲は、1 から 32766 (ヌル可能な場合には 32765) までです。FOR MIXED DATA または混合データの CCSID を指定する場合は、4 から 32766 (ヌル可能な場合は 32765) までが範囲になります。長さ指定を省略した場合は、1 文字の長さが指定されたものと見なされます。

CHARACTER VARYING (*整数***)** または **CHAR VARYING (***整数***)** または **VARCHAR(***整数***)**
最大長が整数 の可変長文字ストリングを示します。指定できる範囲は 1 から 32740 (ヌル可能な場合は 32739) までです。FOR MIXED DATA または混合データの CCSID を指定する場合は、4 から 32740 (ヌル可能な場合は 32739) までが範囲になります。

CLOB(*整数***[K|M|G])** または **CHAR LARGE OBJECT(***整数***[K|M|G])** or **CHARACTER LARGE OBJECT(***整数***[K|M|G])**
CLOB または **CHAR LARGE OBJECT** または **CHARACTER LARGE OBJECT**
指定された最大長の文字ラージ・オブジェクト・ストリングを示します。最大長は、1 ～ 2 147 483 647 の範囲の値でなければなりません。FOR MIXED DATA や混合データ CCSID を指定する場合は、4 ～ 2 147 483 647 の範囲の値を指定します。長さ指定を省略すると、1 メガバイトの長さが想定されます。CLOB は、分散表内で使用することはできません。

52. 他のプロダクトとの互換性を備えるために、このオプションが用意されています。ただし、代わりに VARCHAR(整数) または VARGRAPHIC(整数) の指定をお勧めします。

整数
整数の最大値は 2 147 483 647 です。*整数* は、ストリングの最大長を表します。

整数 **K**
整数の最大値は 2 097 152 です。 ストリングの最大長は、*整数* の 1024 倍です。

整数 **M**
整数の最大値は 2 048 です。 ストリングの最大長は、*整数* の 1 048 576 倍です。

整数 **G**
整数の最大値は 2 です。ストリングの最大長は、*整数* の 1 073 741 824 倍です。

GRAPHIC(*整数***)**
GRAPHIC
整数 で指定された長さを持つ固定長グラフィック・ストリングを示します。この整数は、1 から 16383 (ヌルが使用可能な場合は 16382) までの範囲です。長さ指定を省略した場合は、1 文字の長さが指定されたものと見なされます。

VARGRAPHIC(*整数***)** または **GRAPHIC VARYING(***整数***)**
最大長が*整数* の可変長グラフィック・ストリングを示します。指定できる範囲は 1 から 16370 (ヌル可能な場合は 16369) までです。

DBCLOB(*整数***[K|M|G])**
DBCLOB
指定された最大長の 2 バイト文字ラージ・オブジェクト・ストリングを示します。

最大長は、1 ～ 1 073 741 823 の範囲の値でなければなりません。長さ指定を省略すると、1 メガバイトの長さが想定されます。DBCLOB は、分散表内で使用することはできません。

整数
整数の最大値は 1 073 741 823 です。 *整数* は、ストリングの最大長を表します。

整数 **K**
整数の最大値は 1 028 576 です。 ストリングの最大長は、*整数* の 1024 倍です。

整数 **M**
整数の最大値は 1 024 です。 ストリングの最大長は、*整数* の 1 048 576 倍です。

整数 **G**
整数の最大値は 1 です。ストリングの最大長は、*整数* の 1 073 741 824 倍です。

BLOB(*整数***[K|M|G])** または **BINARY LARGE OBJECT(***整数***[K|M|G])**
BLOB または **BINARY LARGE OBJECT**
指定された最大長の 2 進ラージ・オブジェクト・ストリングを示します。

最大長は、1 ～ 2 147 483 647 の範囲の値でなければなりません。長さ指定を省略すると、1 メガバイトの長さが想定されます。BLOB は、分散表内で使用することはできません。

整数
整数の最大値は 2 147 483 647 です。*整数* は、ストリングの最大長を表します。

整数 **K**
整数の最大値は 2 097 152 です。 ストリングの最大長は、*整数* の 1024 倍です。

整数 **M**
整数の最大値は 2 048 です。 ストリングの最大長は、*整数* の 1 048 576 倍です。

整数 **G**
整数の最大値は 2 です。ストリングの最大長は、*整数* の 1 073 741 824 倍です。

DATE
日付を示します。

TIME
時刻を示します。

TIMESTAMP
タイム・スタンプを示します。

DATALINK(*整数***)** または **DATALINK**
指定された最大長のデータ・リンクを示します。最大長は、1 ～ 32717 の範囲の値でなければなりません。FOR MIXED DATA や混合データ CCSID を指定する場合は、4 ～ 32717 の範囲の値を指定します。予期される最大の URL と、何らかのデータ・リンク・コメントが収まるだけの充分な長さを指定する必要があります。長さ指定を省略すると、200 という長さが想定されます。DATALINK は、分散表内で使用することはできません。

DATALINK 値は、1 組の組み込みスカラー関数でカプセル化される値です。DLVALUE 関数で DATALINK 値を作成します。次の関数を使用すれば、DATALINK 値から属性を抽出することができます。

- DLCOMMENT
- DLLINKTYPE
- DLURLCOMPLETE
- DLURLPATH
- DLURLPATHONLY
- DLURLSCHEME
- DLURLSERVER

データ・リンクは、どの索引にもその一部として含めることはできません。したがって、基本キー、外部キー、または固有制約の 1 つの列としてデータ・リンクを組み込むことは不可能です。

ROWID
行 ID を示します。 ROWID 列は、1 つの表に 1 つだけ設けることができます。

特殊タイプ名
列のデータ・タイプが、特殊タイプ (ユーザー定義のデータ・タイプ) であることを指定します。この列の長さ、精度、および位取りは、それぞれ、特殊タイプのソースとなっているタイプの長さ、精度、および位取りと同じになります。スキーマ名なしの特殊タイプを指定すると、その特殊タイプ名は、SQL パス上のスキーマを検索することで解決されます。

ALLOCATE(*整数*)
VARCHAR、VARGRAPHIC、および LOB タイプに関して、それぞれの行の該当列に対して予約するスペースを指定します。長さが割り振られた値以下の列の値は、行の固定長部分に保管されます。長さが割り振られた値より長い列の値は、行の可変長部分に保管され、これを検索するためには、余分の入出力操作が必要になります。割り振ることのできる値の範囲は、1 からストリングの最大長までですが、最大行バッファー・サイズにより制限されます。最大行バッファー・サイズについては、569 ページの『最大行サイズ』を参照してください。 FOR MIXED または混合データの CCSID を指定する場合は、4 からストリングの最大長までが範囲になります。割り振られる長さを指定しなかった場合は、割り振られる長さに 0 を指定したものと見なされます。VARGRAPHIC の場合は、整数は DBCS 文字または UCS-2 文字の数になります。デフォルト値として定数が指定され、ALLOCATE の長さがそのデフォルト値の長さより小さい場合は、ALLOCATE の長さはそのデフォルト値の長さであると見なされます。

FOR BIT DATA
列の値が、コード化文字セットと関連付けられていないこと、および変換されないことを指定します。 FOR BIT DATA は、CHARACTER または VARCHAR の列にのみ有効です。FOR BIT DATA を指定した列の CCSID は、65535 です。FOR BIT DATA は、CLOB 列には使用できません。

FOR SBCS DATA
列の値に、SBCS (1 バイト文字セット) データが入ることを指定します。FOR SBCS DATA が、CHAR、VARCHAR、および CLOB 列のデフォルト値になるのは、表の作成時における現行サーバーのデフォルトの CCSID が DBCS 対応でない場合、あるいは、それらの列の長さが 4 よりも小さい場合です。FOR SBCS DATA は、CHARACTER、VARCHAR、または CLOB 列のみに有効です。FOR SBCS DATA の CCSID は、表作成時点における現行サーバーのデフォルトの CCSID によって決まります。

FOR MIXED DATA
列の値に、SBCS データと DBCS データの両方が入ることを指定します。FOR MIXED DATA が、CHAR、VARCHAR、および CLOB 列のデフォルト値になるのは、表の作成時における現行サーバーのデフォルトの CCSID が DBCS 対応でなく、しかも、それらの列の長さが 3 よりも大きい場合です。どの FOR MIXED DATA 列も DBCS 混用データベース・フィールドです。FOR MIXED DATA は、CHARACTER、VARCHAR、または CLOB 列のみに有効です。FOR MIXED DATA の CCSID は、表作成時点における現行サーバーのデフォルトの CCSID によって決まります。

CCSID 整数
整数で指定した CCSID を持つデータが、列の値に入ることを指定します。その整数が SBCS CCSID である場合、その列のデータは SBCS データです。整数が混合データ CCSID である場合、その列は混合データとなり、その列の長さには、3 よりも大きい値を指定する必要があります。文字列の場合、CCSID は SBCS CCSID や 混合データ CCSID にする必要があります。グラフィックの列の場合は、CCSID は DBCS または UCS-2 CCSID でなければなりません。 CCSID がグラフィックの列に指定されていない場合は、CCSID は、表の作成時点における現行サーバーのデフォルトの CCSID によって決まります。有効な CCSID のリストについては、897 ページの『付録 E. CCSID の値』の項を参照してください。

DEFAULT
列のデフォルト値を指定します。この文節は、1 つの*列定義* の中で複数回指定することはできません。 ROWID 列または識別列 (AS IDENTITY と定義されている列) については、DEFAULT を指定することはできません。 ROWID 列および識別列のデフォルト値は、データベース・マネージャーが生成します。DEFAULT キーワードの後に値が指定されていない場合は、次のようになります。

- 列がヌル可能の場合、デフォルト値はヌル値になります。
- 列がヌル可能でない場合、デフォルト値は列のデータ・タイプによって決まります。

データ・タイプ	デフォルト値
数値	0
固定長ストリング	ブランク
可変長ストリング	0 のストリング長
日付	INSERT の時点の当日の日付
時刻	INSERT の時点の当日の時刻
タイム・スタンプ	INSERT の時点の当日のタイム・スタンプ
データ・リンク	DLVALUE(’’,’URL’,’’) に対応する値
特殊タイプ	特殊タイプの対応するソース・タイプのデフォルト値

NOT NULL および DEFAULT を*列の定義* から省いた場合、DEFAULT NULL の暗黙の指定が取られます。

定数
その列のデフォルト値としての定数を指定します。これは、85 ページの『割り当ておよび比較』で説明している割り当て規則に従って、その列に割り当てることができる値を表す定数にする必要があります。浮動小数点定数は、SMALLINT、INTEGER、DECIMAL、または NUMERIC 列に使用してはなりません。10 進定数には、小数点より右方に、その列に指定された位取りより多くの桁を含めてはなりません。

USER
INSERT または UPDATE の時点での USER 特殊レジスターの値を、その

列のデフォルト値として指定します。列のデータ・タイプは、USER 特殊レジスターの長さ属性と同じかそれより大きい長さ属性を持つ CHAR または VARCHAR でなければなりません。

NULL
その列のデフォルト値としてヌルを指定します。NOT NULL を指定する場合は、同じ列の定義内で DEFAULT NULL を指定してはなりません。

CURRENT_DATE
現在の日付を列のデフォルト値として指定します。CURRENT_DATE を指定する場合は、列のデータ・タイプは DATE または DATE に基づく特殊タイプでなければなりません。

CURRENT_TIME
現在の時刻を列のデフォルト値として指定します。CURRENT_TIME を指定する場合は、列のデータ・タイプは TIME または TIME に基づく特殊タイプでなければなりません。

CURRENT_TIMESTAMP
現在のタイム・スタンプを列のデフォルト値として指定します。CURRENT_TIMESTAMP を指定する場合は、列のデータ・タイプは TIMESTAMP または TIMESTAMP に基づく特殊タイプでなければなりません。

キャスト関数名
この形式のデフォルト値は、特殊タイプやデータ・タイプ BLOB、CLOB、 DBCLOB、 DATE、 TIME、または TIMESTAMP として定義された列でのみ使用することができます。次の表は、これらのキャスト関数 の許可されている使用法を示します。

データ・タイプ	キャスト関数名
BLOB、CLOB、または DBCLOB に基づく特殊タイプ N	BLOB、CLOB、または DBCLOB *
DATE、TIME、または TIMESTAMP に基づく特殊タイプ N	N (N の作成時に生成されたユーザー定義のキャスト関数) ** あるいは DATE、TIME、または TIMESTAMP *
他のデータ・タイプに基づく特殊タイプ	N (N の作成時に生成されたユーザー定義のキャスト関数) **
BLOB、CLOB、または DBCLOB	BLOB、CLOB、または DBCLOB *
DATE、TIME、または TIMESTAMP	DATE、TIME、または TIMESTAMP *

注:

* 関数には、QSYS2 の暗黙的または明示的スキーマ名のデータ・タイプ (または、特殊タイプのソース・タイプ) の名前と一致する名前を指定する必要があります。

** 関数には、列の特殊タイプの名前と一致する名前を指定する必要があります。スキーマ名で修飾する場合は、特殊タイプのスキーマ名と同じ名前を指定する必要があります。修飾しない場合は、関数の解析から得られるスキーマ名は、特殊タイプのスキーマ名と同じ名前にする必要があります。

定数
定数を引き数として指定します。この定数は、特殊タイプのソース・タ

イプの定数、あるいは、特殊タイプでない場合は、データ・タイプの定数の規則に準拠する必要があります。BLOB、 CLOB、DBCLOB、DATE、 TIME、および TIMESTAMP 関数の場合は、この定数をストリング定数にする必要があります。

USER
INSERT または UPDATE の時点での USER 特殊レジスターの値をその列のデフォルト値として指定します。この列の特殊タイプのソース・タイプのデータ・タイプは、USER 特殊レジスターの長さ属性と同じかそれより大きい長さ属性を持つ CHAR または VARCHAR でなければなりません。

CURRENT_DATE
現在の日付を列のデフォルト値として指定します。CURRENT_DATE を指定する場合、列の特殊タイプのソース・タイプのデータ・タイプは、DATE にする必要があります。

CURRENT_TIME
現在の時刻を列のデフォルト値として指定します。CURRENT_TIME を指定する場合、列の特殊タイプのソース・タイプのデータ・タイプは、TIME にする必要があります。

CURRENT_TIMESTAMP
現在のタイム・スタンプを列のデフォルト値として指定します。CURRENT_TIMESTAMP を指定する場合、列の特殊タイプのソース・タイプのデータ・タイプは、TIMESTAMP にする必要があります。

GENERATED
列の値をデータベース・マネージャーが生成することを指定します。列が識別列 (AS IDENTITY 文節で定義されたもの) と見なされる場合は、GENERATED を指定する必要があります。また、列のデータ・タイプが ROWID (または ROWID に基づく特殊タイプ) である場合も、GENERATED を指定できます。その他の場合は、GENERATED を指定してはなりません。

ALWAYS
表に行が 1 つ挿入されるたびに、常にデータベース・マネージャーが列の値を生成することを指定します。 ALWAYS は推奨値です。

BY DEFAULT
行が挿入されたときに、列の値が指定されていない場合のみ、データベース・マネージャーが列の値を生成することを指定します。値が指定されている場合は、データベース・マネージャーはその値を使用します。

ROWID 列の場合は、データベース・マネージャーは指定された値を使用しますが、その値は、すでに DB2 UDB (OS/390 および z/OS 版) または DB2 UDB for iSeries により生成されている有効な固有の行 ID でなければなりません。

識別列の場合は、データベース・マネージャーは指定された値を挿入しますが、その識別列が固有制約を持っているか、またはその識別列を単独で指定する固有索引を持っている場合を除き、その値がその列の固有の値であるかどうかの検査は行いません。

AS IDENTITY

列が表の識別列であることを指定します。 1 つの表は識別列を 1 つだけ持つことができます。 AS IDENTITY を指定できるのは、列のデータ・タイプが、厳密に位取りがゼロの数値タイプ (SMALLINT、 INTEGER、 BIGINT、 DECIMAL、または位取りがゼロの NUMERIC、またはこれらのタイプに基づく特殊タイプ) である場合だけです。

識別列は、暗黙的に NOT NULL になります。

START WITH *数値定数*

識別列について生成される最初の値を指定します。小数点の右側にゼロ以外の数字がないことを条件として、この列に割り当てることのできる任意の正または負の値を指定できます。

識別列を定義するときに値を明示的に指定していない場合のデフォルト値は、昇順の場合は MINVALUE で、降順の場合は MAXVALUE です。この値は、シーケンスが最大値または最小値に達した後で、シーケンスの循環により到達する値になるとは限りません。 START WITH 文節を使用することにより、この循環に使用される値の範囲外の値からシーケンスを開始することができます。循環に使用する範囲は、MINVALUE および MAXVALUE で定義します。

INCREMENT BY *数値定数*

識別列の連続した値の間隔を指定します。この値には、0 でなく、長整数定数の値を超過せず、かつ小数点の右側にゼロ以外の数字がないことを条件として、この列に割り当てることのできる任意の正または負の値を指定できます。デフォルト値は 1 です。

この値が正である場合は、識別列の値のシーケンスは昇順になります。この値が負の場合は、値のシーケンスは降順になります。

MAXVALUE *数値定数*

この識別列用として生成される最大値を示す数値定数を指定します。この値には、この列に割り当てることのできる任意の正または負の値を指定できますが、最小値より大きい値でなければなりません。

識別列を定義するときにこの値を明示的に指定していない場合は、この値は、昇順シーケンスの場合は該当データ・タイプ (および、DECIMAL の場合は精度) の最大値になります。降順シーケンスの場合は、この値は START WITH の値ですが、START WITH を指定していなければ -1 です。

MINVALUE *数値定数*

この識別列用として生成される最小値を示す数値定数を指定します。この値には、この列に割り当てることのできる任意の正または負の値を指定できますが、最大値より小さい値でなければなりません。

識別列を定義するときにこの値を明示的に指定していない場合は、この値は、昇順シーケンスの場合は START WITH の値ですが、START WITH を指定していなければ 1 です。降順シーケンスの場合は、この値は、該当データ・タイプ (および、DECIMAL の場合は精度) の最小値です。

CACHE または NO CACHE
事前割り振りの値をメモリー内に保持するかどうかを指定します。値を事前に割り振ってキャッシュに保管しておくと、表に行を挿入するときのパフォーマンスが向上します。

CACHE *整数*
データベース・マネージャーが事前割り振りしてメモリー内に保持する、識別列シーケンスの値の数を指定します。指定できる最小の値は 2 で、最大の値は、1 つの整数で表せる最大の値です。デフォルト値は 20 です。

システム障害が発生すると、キャッシュに保管されていてまだ割り当てられていない識別列の値はすべて失われ、使用不能になります。したがって、CACHE に指定する値は、システム障害が発生したときに失われる可能性のある識別列の値の最大数でもあります。

NO CACHE
識別列の値を事前割り振りしないことを指定します。

CYCLE または NO CYCLE
シーケンスの最大値または最小値に達した後も、この識別列について値を生成し続けるかどうかを指定します。

CYCLE
最大値または最小値に達した後も、この列の値を生成し続けることを指定します。このオプションを使用した場合は、昇順シーケンスがシーケンスの最大値に達した後では、最小値が生成されます。降順シーケンスがシーケンスの最小値に達した後は、最大値が生成されます。列の最大値と最小値によって、循環に使用される範囲が決まります。

CYCLE が効力を持っている場合は、データベース・マネージャーは 1 つの識別列について重複する値を生成することがあります。対象の識別列について固有制約または固有索引が存在する場合は、固有でない値が生成されるとエラーが起こります。

NO CYCLE
シーケンスの最大値または最小値に達した後は、この識別列について値を生成しないことを指定します。これは、デフォルト値です。

ORDER または NO ORDER
識別値を要求された順序で生成するかどうかを指定します。

ORDER
識別値を要求された順序で生成することを指定します。

NO ORDER
値を要求された順序で生成する必要がないことを指定します。これは、デフォルト値です。

データ・リンク・オプション
DATALINK データ・タイプに関連したオプションを指定します。

LINKTYPE URL
リンクのタイプを URL として定義します。

NO LINK CONTROL
これを指定すると、リンク済みファイルが存在するか否かを判別するための検査は行われなくなります。URL の構文だけが検査されます。リンク済みファイルに関してデータベース・マネージャー制御は行われません。

FILE LINK CONTROL
これを指定すると、リンク済みファイルの存在を確かめるための検査を行う必要が生じます。追加のオプションを使用して、データベース・マネージャーにリンク済みファイルに対するより強力な制御権を与えることが可能です。

FILE LINK CONTROL が指定されると、各ファイルは一度だけリンクすることができます。つまり、その URL を指定できるのは単一の表内の単一の FILE LINK CONTROL 列内だけです。

ファイル・リンク・オプション
リンク済みファイルのデータベース・マネージャー制御のレベルを定義する追加のオプションです。

INTEGRITY
DATALINK 値と実ファイルとの間のリンクの整合性レベルを指定します。

ALL
これを指定すると、DATALINK 値として指定されたどのファイルもデータベース・マネージャーに制御されるようになります。また、標準ファイル・システムのプログラミング・インターフェースを使用してそれらのファイルの削除または名前変更は行うことができなくなります。

READ PERMISSION
DATALINK 値に指定されたファイルの読み取り許可を決定する方法を指定します。

FS
これを指定すると、読み取りアクセス許可は、ファイル・システム許可によって決定されるようになります。このようなファイルには、列からファイル名を検索しなくてもアクセスできます。

DB
これを指定すると、読み取りアクセス許可は、データベースによって決定されるようになります。このファイルへのアクセスは、オープン操作において、表から DATALINK 値の検索時に戻される有効なファイル・アクセス・トークンを渡すことでしか許可されません。READ PERMISSION DB を指定する場合は、WRITE PERMISSION BLOCKED も指定する必要があります。

WRITE PERMISSION
DATALINK 値に指定されたファイルの書き込み許可を決定する方法を指定します。

FS
これを指定すると、書き込みアクセス許可は、ファイル・システム

許可によって決定されるようになります。このようなファイルには、列からファイル名を検索しなくてもアクセスできます。

BLOCKED
これを指定すると、書き込みアクセスがブロックされます。ファイルは、どのインターフェースを介しても直接更新することはできません。情報に対して更新を実行するには、代替メカニズムを使用する必要があります。例えば、ファイルをコピーし、そのコピーを更新してから、その DATALINK 値を更新することで、そのファイルの新しいコピーを指し示します。

RECOVERY
この列の値で参照されるファイルの特定時点回復をデータベース・マネージャーがサポートするか否かを指定します。

NO
これを指定すると、特定時点回復はサポートされません。

ON UNLINK
DATALINK 値の変更または削除 (リンク解除) 時にファイルに対して講じるアクションを指定します。これは、WRITE PERMISSION FS を使用している場合には適用できないことに注意してください。

RESTORE
これを指定すると、ファイルのリンクが解除されたら、データ・リンク・ファイル・マネージャーは、そのファイルを、それがリンクされた時点で存在していた許可と一緒に所有者に戻そうとします。その所有者がすでにファイル・サーバーへの登録を解除されている場合、この結果は、それらのファイルが収められているファイル・システムによって異なります。それらのファイルが AIX ファイル・システムにある場合の所有者は「dfmunknown」です。IFS にある場合の所有者は QDLFM です。これは、INTEGRITY ALL と WRITE PERMISSION BLOCKED も指定されている場合にのみ指定することができます。

DELETE
これを指定すると、ファイルは、リンク解除の時点で削除されます。これは、READ PERMISSION DB と WRITE PERMISSION BLOCKED も指定されている場合にのみ指定することができます。

MODE DB2OPTIONS
このモードは、1 組のデフォルト・ファイル・リンク・オプションを定義します。DB2OPTIONS によって定義されるデフォルト値は、次のとおりです。

- INTEGRITY ALL
- READ PERMISSION FS
- WRITE PERMISSION FS
- RECOVERY NO

NOT NULL
列にヌル値が入るのを防止します。NOT NULL を指定しないことは、その列がヌルであってもよいことを意味します。

列制約

CONSTRAINT *制約名*
制約の名前を指定します。*制約名* は、すでに CREATE TABLE ステートメントで指定され、かつすでに現行サーバーに存在している制約を示すものであってはなりません。

この文節の指定がない場合、固有制約の名前がデータベース・マネージャーによって生成されます。

PRIMARY KEY
これは、1 つの列からなる基本キーを定義する簡便な手段です。列 C の定義に PRIMARY KEY を指定した場合、その効果は、別個の文節として PRIMARY KEY(C) 文節を指定したのと同一です。

この文節は複数の列の定義に指定してはなりません。また列の定義に UNIQUE 文節の指定がある場合には、この文節を指定してはなりません。この列は、LOB 列または DATALINK 列であってはなりません。

基本キーを追加すると、CHECK 制約が暗黙的に追加され、その基本キーを構成する列で NULL を使用することはできないという規則が適用されます。

UNIQUE
これは、1 つの列からなる固有キーを定義する簡便な手段です。列 C の定義に UNIQUE の指定がある場合、その効果は、別個の文節として UNIQUE(C) 文節が指定された場合と同一です。

この文節は、1 つの列定義で複数回指定することはできません。また、列定義で PRIMARY KEY が指定されている場合は、この文節を指定してはなりません。この列は、LOB 列または DATALINK 列であってはなりません。

REFERENCES 文節
列定義 の*REFERENCES 文節* は、単一の列から成る外部キーを定義するための簡便な手段です。列 C の定義に参照文節の指定がある場合、その効果は、C が識別された唯一の列である FOREIGN KEY 文節の一環としてその参照文節が指定されている場合と同一です。

CHECK(*検査条件***)**
列定義 の CHECK(*検査条件*) は、単一の列のみを参照する*検査条件* を持つ検査制約を定義するための簡便な手段です。したがって、列 C の列定義で CHECK を指定した場合、検査制約の*検査条件* では、C 以外の列を参照することができなくなります。結果は、検査制約を別個の文節として指定した場合と同じです。

CHECK 制約の中で、FILE LINK CONTROL 列を持つ ROWID または DATALINK を参照することはできません。その他の制限事項については、567 ページの『検査制約』を参照してください。

LIKE

表名 または *ビュー名*
指定の表またはビューに定義されている列をこの表に含めることを指定します。指定する*表名* やビュー名 は、すでにサーバー上に存在する表またはビューを識別していなければなりません。

LIKE の使用は、n 列 (n は、識別された表またはビュー内の列数) を暗黙的に定義したことになります。この暗黙の定義には、n 列の以下の属性が含まれます (そのデータ・タイプに該当する場合)

- 列名 (および、システム列名)
- データ・タイプ、長さ、精度、および位取り
- CCSID

表名 の直後に LIKE 文節を指定し、括弧で囲まなかった場合は、以下の列属性も含まれます。その他の場合は、これらの属性は含まれません (デフォルト値および識別属性は、*コピー・オプション* を使用して制御することもできます)。

- デフォルト値 (*表名* が指定され、*ビュー名* は指定されていない場合)
- ヌル可能性
- 識別属性
- 列の見出しとテキスト (714 ページの『LABEL』を参照)

暗黙の定義には、識別された表またはビューのその他のオプション属性は含まれません。例えば、新規の表には、基本キー、外部キー、またはトリガーは自動的には組み込まれません。新規の表にこうしたオプション属性が組み込まれるのは、オプション文節を明示的に指定した場合に限られます。

指定された表またはビューが非 SQL 作成の物理ファイルまたは論理ファイルの場合、非 SQL 属性は除去されます。例えば、日付と時刻の形式は ISO 形式に変更されます。

AS 副照会文節

列名
表を構成する列の名前を指定します。*列名* は修飾できません。表の複数の列や表のシステム列名に同じ名前を使用することもできません。

FOR COLUMN *システム列名*
列の OS/400 名を指定します。表の複数の列やシステム列名に対して、同じ名前を使用してはなりません。

システム列名が指定されず、また列名が有効なシステム列名でない場合には、システム列名が生成されます。システム列名の生成方法に関する詳細については、572 ページの 『列名の生成の規則』を参照してください。

選択ステートメント
表の列の名前および記述が、選択ステートメントを実行した場合に選択ステートメントの派生結果表に現れる列と同じになるようにすることを指定します。 AS 選択ステートメントを使用すると、この表について n 個の列を暗黙的に定義したことになります。n は、選択ステートメントの結果として発生する列の数です。この暗黙の定義には、n 列の以下の属性が含まれます (そのデータ・タイプに該当する場合)

- 列名 (および、システム列名)
- データ・タイプ、長さ、精度、および位取り
- CCSID
- ヌル可能性

- 列の見出しとテキスト (714 ページの『LABEL』を参照)

以下の属性は組み込まれません (デフォルト値と識別属性は、*コピー・オプション* を使用して組み込むことができます)。

- デフォルト値
- 識別属性

暗黙の定義には、識別された表またはビューのその他のオプション属性は含まれません。例えば、新規の表には、表からの基本キーや外部キーは自動的には組み込まれません。新規の表にこうしたオプション属性が組み込まれるのは、オプション文節を明示的に指定した場合に限られます。

暗黙的に定義された列は、*選択ステートメント* の結果表の列の名前を継承します。したがって、すべての結果列について、*選択ステートメント* または列名リストの中で列名を指定する必要があります。式、定数、および関数から派生する結果列については、*選択ステートメント* で結果列の直後に AS 列名文節を指定するか、*選択ステートメント* の前の列リスト内に名前を指定する必要があります。

選択ステートメント は、ホスト変数または組み込みパラメーター・マーカー (疑問符) を参照するものであってはなりません。

WITH DATA
選択ステートメント を実行することを指定します。表の作成後に、*選択ステートメント* の結果表の行が自動的に表に挿入されます。

WITH NO DATA または DEFINITION ONLY
選択ステートメント を実行しないことを指定します。したがって、自動的に表に挿入される行のセットを持つ結果表はありません。

コピー・オプション

INCLUDING IDENTITY COLUMN ATTRIBUTES
この表が、*選択ステートメント*、*表名*、または*ビュー名* の結果として生じる列の識別属性 (もしあれば) を継承することを指定します。一般に、識別属性がコピーされるのは、表、ビュー、または選択ステートメントの中の対応する列のエレメントが、識別属性を持つ基礎表列の名前に直接または間接にマップされる表列またはビュー列の名前である場合です。

INCLUDING IDENTITY COLUMN ATTRIBUTES 文節と AS *選択ステートメント* 文節を指定してあるときは、以下の場合には新規の表の列は識別属性を継承しません。

- *選択ステートメント* の選択リストに、識別列名の複数のインスタンスが含まれている (つまり同じ列を複数回選択している) 場合。
- *選択ステートメント* の選択リストに、複数の識別列が含まれている (つまり結合が含まれている) 場合。
- 選択リスト内の式のどれかに識別列が含まれている場合。
- *選択ステートメント* に一組の演算 (共用体) が含まれている場合。

INCLUDING IDENTITY を指定しなかった場合は、表には識別列は含まれません。

EXCLUDING IDENTITY COLUMN ATTRIBUTES
この表が、*選択ステートメント*、*表*、または*ビュー名* の結果として生じる列の識別属性を継承しないことを指定します。

INCLUDING COLUMN DEFAULTS
この表が、*選択ステートメント*、*表名*、または*ビュー名* から生じる列のデフォルト値を継承することを指定します。デフォルト値は、INSERT で値が指定されていない場合に、列に割り当てられる値です。

USING TYPE DEFAULTS を指定する場合は、INCLUDING COLUMN DEFAULTS を指定しないでください。

INCLUDING COLUMN DEFAULTS を指定しなかった場合は、デフォルト値は継承されません。

EXCLUDING COLUMN DEFAULTS
この表が、*選択ステートメント*、*表名*、または*ビュー名* から生じる列のデフォルト値を継承しないことを指定します。

USING TYPE DEFAULTS
この表のデフォルト値が、*選択ステートメント*、*表名*、または*ビュー名* から生じる列のデータ・タイプに応じて決まることを指定します。その列がヌル可能である場合は、デフォルト値はヌル値です。その他の場合は、デフォルト値は以下のようになります。

データ・タイプ	デフォルト値
数値	0
固定長ストリング	ブランク
可変長ストリング	0 のストリング長
日付	INSERT の時点の当日の日付
時刻	INSERT の時点の当日の時刻
タイム・スタンプ	INSERT の時点の当日のタイム・スタンプ
データ・リンク	DLVALUE(’’,’URL’,’’) に対応する値
特殊タイプ	特殊タイプの対応するソース・タイプのデフォルト値

INCLUDING COLUMN DEFAULTS を指定する場合は、USING TYPE DEFAULTS は指定しないでください。

固有制約

CONSTRAINT *制約名*
制約の名前を指定します。*制約名* は、すでに CREATE TABLE ステートメントで指定され、かつすでに現行サーバーに存在している制約を示すものであってはなりません。

この文節の指定がない場合、固有制約の名前がデータベース・マネージャーによって生成されます。

PRIMARY KEY(*列名***、...)**
指定した列で構成される基本キーを定義します。表は基本キーを 1 つだけ持つことができます。したがって、この文節は複数回指定することはできず、またこの簡便な手法が表の基本キーを定義するのに使用されていた場合には、指定する

ことはできません。指定する列は、その CREATE TABLE ステートメントで前に指定した他の UNIQUE 制約で指定されている列と同じであってはなりません。例えば、UNIQUE(B,A) がすでに指定されている場合、PRIMARY KEY(A,B) の指定は許されません。

それぞれの*列名* は、該当の表の列を識別する非修飾の名前でなければなりません。 同じ列を複数回指定することはできません。この列は、LOB 列または DATALINK 列であってはなりません。 指定できる列の数は 120 を超えてはならず、それぞれのバイト数の合計は 2000-n を超えてはなりません。ここで、n はヌルが許される列の数です。バイト数については、 570 ページの表 47 を参照してください。

固有索引は、別個のシステム論理ファイルとしてではなく、システム物理ファイルの一部として作成されます。基本キーを追加すると、CHECK 制約が暗黙的に追加され、その基本キーを構成するどの列でも NULL を使用することはできないという規則が適用されます。

UNIQUE(*列名***、...)**
識別された列で構成される固有キーを定義します。UNIQUE 文節は複数回指定しても構いません。指定する列は、その CREATE TABLE ステートメントで前に指定した他の UNIQUE 制約や PRIMARY KEY で指定されている列と同じであってはなりません。ある固有制約が他の制約の指定と同一であるか否かを判別するには、その列のリストを対比します。例えば、UNIQUE(A,B) は UNIQUE(B,A) と同一です。

それぞれの列名は、該当の表の列を識別する非修飾の名前でなければなりません。同じ列を複数回指定することはできません。この列は、LOB 列または DATALINK 列であってはなりません。 指定できる列の数は 120 を超えてはならず、それぞれのバイト数の合計は 2000-n を超えてはなりません。ここで、n はヌルが許される列の数です。バイト数については、 570 ページの表 47を参照してください。

指定された列に基づく固有索引が、その CREATE TABLE ステートメントの実行過程で作成されます。固有索引は、別個のシステム論理ファイルとしてではなく、システム物理ファイルの一部として作成されます。

参照制約

CONSTRAINT *制約名*
制約の名前を指定します。*制約名* は、すでに CREATE TABLE ステートメントで指定され、かつすでに現行サーバーに存在している制約を示すものであってはなりません。

この文節の指定がない場合、固有制約の名前がデータベース・マネージャーによって生成されます。

FOREIGN KEY
FOREIGN KEY 文節の各指定は、1 つの参照制約を定義します。

(*列名***、...)**
参照制約の外部キーは、指定した列で構成されます。それぞれの*列名* は、該当の表の列を識別する非修飾の名前でなければなりません。 同じ列を複数回指定することはできません。この列は、LOB 列または DATALINK 列

であってはなりません。 指定できる列の数は 120 を超えてはならず、その長さの合計は 2000-n を超えてはなりません。ここで、n はヌルが許される列の数です。

REFERENCES *表名*

REFERENCES 文節に指定する表名 は、作成しようとしている表、またはサーバーに既存の基礎表を識別していなければならず、またカタログ表またはグローバル一時表を識別していてはなりません。

参照制約の外部キー、親キー、および親表が、前に指定した参照制約の外部キー、親キー、および親表と同一である場合は、その参照制約は*重複* します。重複する参照制約は許されますが、お勧めできません。

以下の説明で、T1 は作成される表を指し、T2 は識別された親表を表しています。

指定した外部キーは、T2 の親キーと同じ数の列を持たなければなりません。外部キーの *n* 番目の列の記述とその親キーの *n* 番目の列の記述は、同一のデータ・タイプと長さを持たなければなりません。

(*列名***、...)**

参照制約の親キーは、ここで指定する列によって構成されます。各*列名* は T2 の列を識別する非修飾の名前でなければなりません。同じ列を複数回指定することはできません。この列は、LOB 列または DATALINK 列であってはなりません。 指定できる列の数は 120 を超えてはならず、それぞれのバイト数の合計は 2000-n を超えてはなりません。ここで、n はヌルが許される列の数です。 バイト数については、 570 ページの表 47 を参照してください。

この列名のリストは、T2 の基本キーまたは T2 に存在する UNIQUE 制約の列名のリストと同一でなければなりません。名前は、基本キーと同じ順序で指定する必要はありません。しかし、*外部キー* 文節の列のリストに対応する順序で指定する必要があります。列名のリストの指定がない場合、T2 は基本キーを持たなければなりません。列名のリストの省略は、基本キーの列の暗黙の指定を意味しています。

FOREIGN KEY 文節によって指定される参照制約は、T2 が親表で、T1 がその従属表である関係を定義します。

ON DELETE

親表の行が削除される時点で、従属表について行うアクションを指定します。可能なアクションには以下の 5 つがあります。

- NO ACTION (デフォルト)
- RESTRICT
- CASCADE
- SET NULL
- SET DEFAULT

外部キーの列にヌルが許される列がある場合を除いて、SET NULL を指定してはなりません。

FILE LINK CONTROL を指定した DATALINK 列が T1 に含まれる場合には、CASCADE を指定してはなりません。

削除規則は、T2 の行が DELETE または波及削除操作の対象で、しかもその行が T1 に従属する行を持っている場合に適用されます。以下の説明で、*p* はそのような T2 の行を表しています。

- RESTRICT または NO ACTION を指定した場合、エラーが生じ、行の削除は行われません。
- CASCADE を指定した場合、削除操作は、T1 の *p* の従属行に波及します。
- SET NULL を指定した場合、T1 の *p* の各従属行の外部キーのヌル可能な各列は、ヌルに設定されます。
- SET DEFAULT を指定した場合、T1 の *p* の各従属行の外部キーの各列は、そのデフォルト値に設定されます。

ON UPDATE
親表の行が更新される時点で、従属表で行うアクションを指定します。

更新規則は、T2 の行が UPDATE または波及更新操作の対象で、しかもその行が T1 に従属行を持つ場合に適用されます。以下の説明で、*p* はそのような T2 の行を表しています。

- RESTRICT または NO ACTION を指定した場合、エラーが生じ、行の更新は行われません。

検査制約

CONSTRAINT *制約名*
検査制約の名前を指定します。*制約名* は、すでに CREATE TABLE ステートメントで指定され、かつすでに現行サーバーに存在している制約を示すものであってはなりません。

この文節の指定がない場合、固有制約の名前がデータベース・マネージャーによって生成されます。

CHECK(*検査条件***)**
検査制約を定義します。どのような場合も、*検査条件* は、表の行ごとに真か不明にする必要があります。

検査条件 は、*検索条件* です。ただし、次の条件は除きます。

- 表の列だけを参照することができます。
- FILE LINK CONTROL 列を持つ ROWID または DATALINK を参照することはできません。
- 次のいずれも含めることはできません。
  - 副照会
  - スカラー副選択
  - 列関数
  - ホスト変数
  - パラメーター・マーカー
  - LOB を含む複合式 (連結など)

- CURRENT TIMEZONE、CURRENT SCHEMA、CURRENT SERVER、CURRENT PATH、および USER 特殊レジスター
- NOW、CURDATE、および CURTIME スカラー関数
- NODENAME スカラー関数
- 特殊タイプの作成に伴って暗黙に生成された関数以外のユーザー定義関数
- ATAN2、DIFFERENCE、RADIANS、RAND、および SOUNDEX スカラー関数
- スカラー関数 DLVALUE、DLURLPATH、DLURLPATHONLY、DLURLSERVER、または DLURLSCHEME
- スカラー関数 DLURLCOMPLETE (FILE LINK CONTROL と READ PERMISSION DB の属性が指定されたデータ・リンクの場合)

検索条件の詳細については、167 ページの『検索条件』を参照してください。LOB データ・タイプおよび式が含まれる検査制約の詳細については、DB2 UDB for iSeries データベース・プログラミングを参照してください。

ノード・グループ文節

IN *ノード・グループ名*

この表のデータが分配されるノード・グループを指定します。この名前は、現行サーバーに存在するノード・グループを示すものでなければなりません。この文節を指定すると、表は、そのノード・グループのシステムすべてにわたる分散表として作成されます。

LOB 列や DATALINK 列は、分散表内で使用することはできません。

分散表を作成するには、DB2 マルチ・システム・プロダクトをインストールする必要があります。分散表の詳細については、DB2 UDB for iSeries マルチ・システムを参照してください。

PARTITIONING KEY(*列名***,...)**

区分化キーを指定します。区分化キーは、ノード・グループのどのノードに行を置くかを判別するために使用します。それぞれの*列名* は、該当の表の列を識別する非修飾の名前でなければなりません。同じ列を複数回指定することはできません。 PARTITIONING KEY 文節の指定がない場合、基本キーの最初の列が、区分化キーとして使用されます。基本キーがない場合は、表の最初の列で、浮動小数点、日付、時刻、あるいはタイム・スタンプではない列が、区分化キーとして使用されます。

区分化キーを構成する列は、その表に対して固有の制約を構成する列のサブセットでなければなりません。浮動小数点、日付、時刻、およびタイム・スタンプの列は、区分化キーには使用できません。

USING HASHING

ノード・グループ内の対応するサーバーに適切に行を分散するために、区分化キーのデータをハッシュすることを指定します。

使用上の注意

表の属性 : 表は物理ファイルとして作成されます。表が作成される場合、ファイル待ち時間とレコード待ち時間の属性は、物理ファイル作成 (CRTLF) コマンドの WAITFILE キーワードと WAITRCD キーワード上に指定されたデフォルト値に設定されます。

SQL 表は、削除済みの行で使用していたスペースがその後の挿入要求で再利用されるように作成されます。この属性は、コマンドの *CHGPF*、および *REUSEDLT(*NO)* パラメーターの指定によって変更することができます。*CHGPF* コマンドの詳細については、iSeries Information Center の**プログラミング**・カテゴリーの CL 解説書情報を参照してください。

表を作成すると、スキーマの中のジャーナル QSQJRN に対して、自動的にジャーナル処理が開始されます。

分散表は、その表が配布されるすべてのサーバーで作成されます。分散表の詳細については、DB2 UDB for iSeries マルチ・システムを参照してください。

表の所有権 : SQL 名を指定した場合は、表の*所有者* は、作成した表が入れられるスキーマと同じ名前のユーザー・プロファイルです。 それ以外の場合は、ステートメントを実行するジョブのユーザー・プロファイルまたはグループ・ユーザー・プロファイルが表の*所有者* になります。

システム名を指定した場合は、表の*所有者* は、このステートメントを実行しているジョブのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

表の権限 : SQL 名を使用する場合は、表は、*PUBLIC に対するシステム権限 *EXCLUDE を使用して作成されます。システム名を使用する場合、表は、スキーマの作成権限 (CRTAUT) パラメーターによって決められる *PUBLIC に対する権限を使用して作成されます。

表の所有者がグループ・プロファイルのメンバー (GRPPRF キーワード) であり、グループ権限が指定されている (GRPAUT キーワード) 場合は、そのグループ・プロファイルにも、その表に対する権限が与えられます。

最大行サイズ

最大行サイズについては、*列定義* の記述で参照される制約事項が 2 つあります。

- 最大行バッファー・サイズは 32766、あるいは VARCHAR、VARGRAPHIC、または LOB 列が指定されている場合は 32740 です。
- LOB が指定されている場合の最大行サイズは、3 758 096 383 です。 LOB が指定されていない場合の最大行データ・サイズは 32766 で、VARCHAR 列や VARGRAPHIC 列が指定されている場合の最大行データ・サイズは 32740 です。

行バッファーまたは行データ (あるいはその両方) の長さを決定するには、そのデータ・タイプのバイト・カウントに基づいて、その行のそれぞれの列の該当の長さを加算します。

次の表は、ヌル値が使用できない列に関して、データ・タイプごとの列のバイト・カウントを示します。ヌル値が許される列であればどのような列であっても、8 つの列ごとに 1 バイトが必要になります。

表 47. データ・タイプ別のバイト・カウント

データ・タイプ	行バッファー・ バイト・カウント	行データ・バイト・カウント
SMALLINT	2	2
INTEGER	4	4
BIGINT	8	8
DECIMAL(*p*, *s*)	(*p*/2) + 1 の整数部	(*p*/2) + 1 の整数部
NUMERIC(*p*, *s*)	*p*	*p*
FLOAT (単精度)	4	4
FLOAT (倍精度)	8	8
CHAR(*n*)	*n*	*n*
VARCHAR(*n*)	*n*+2	*n*+2
CLOB(*n*)	29+*埋め込み*	*n*+29
GRAPHIC(*n*)	*n**2	*n**2
VARGRAPHIC (n)	*n**2+2	*n**2+2
DBCLOB(*n*)	29+*埋め込み*	*n**2+29
BLOB(*n*)	29+*埋め込み*	*n*+29
DATE	10	4
TIME	8	3
TIMESTAMP	26	10
DATALINK(*n*)	*n*+24	*n*+24
ROWID	42	28
特殊タイプ	ソース・タイプのバイト・カウント	ソース・タイプのバイト・カウント
注: *埋め込み* は、境界合わせに必要な 1 ～ 15 の値です。		

データベースに記述される精度

- 浮動小数点フィールドは、ビット精度ではなく、10 進精度で iSeries データベース内に定義されます。ビット数による精度を 10 進精度に変換するには、*10 進精度 = CEILING(n/3.31)* (*n* は、変換するビット数) という算式を使用します。10 進精度は、対話式 SQL を使用した場合に、表示される数値の桁数を決めるのに使用されます。
- SMALLINT (短整数) フィールドは、10 進精度 (4,0) で保管されます。
- INTEGER (整数) フィールドは、10 進精度 (9,0) で保管されます。
- BIGINT フィールドは、10 進精度 (19,0) で保管されます。

LONG VARCHAR と LONG VARGRAPHIC

非標準構文である LONG VARCHAR および LONG VARGRAPHIC がサポートされていますが、これは使用しないようにしてください。標準構文である VARCHAR(整数) および VARGRAPHIC(整数) の方が優先されます。したがって、VARCHAR(整数) および VARGRAPHIC(整数) を使用することをお勧めします。CREATE TABLE ステートメントの処理後、データベース・マネージャーは、

LONG VARCHAR 列を VARCHAR、そして LONG VARGRAPHIC 列を VARGRAPHIC と見なして処理を進めます。最大長は、移植不能な製品固有の方式で計算されます。

LONG VARCHAR [52]
行内で使用可能なスペースの量によって最大長が決まる可変長文字ストリングを示します。

LONG VARGRAPHIC [52]
行内で使用可能なスペースの量によって最大長が決まる可変長グラフィック・ストリングを示します。

LONG 列の最大長は、次のようにして決まります。 ただし、

- m は、最大行サイズとする。
- i は、表のすべての列 (ただし、LONG VARCHAR でも LONG VARGRAPHIC でもない) のバイト数の合計とする。
- j は、表の LONG VARCHAR および LONG VARGRAPHIC の列の数とする。
- k は、該当の行でヌルが使用可能な列の数とする。

LONG VARCHAR 列それぞれの長さは、INTEGER((m-24-i-((k+7)/8))/j) になります。

それぞれの LONG VARGRAPHIC 列の長さは、LONG VARCHAR 列に関して計算した長さを 2 で割って決定します。この結果の整数部が長さになります。

識別列の使用

表に識別列がある場合は、データベース・マネージャーは、表に行が挿入されたときに、その列の順次数値を自動的に生成することができます。したがって、識別列は基本キーとして最適です。識別列と ROWID 列は、どちらにもデータベース・マネージャーが生成する値が含まれるという点で同じです。 ROWID は、直接行アクセスに使用すると便利です。 ROWID 列には、ROWID データ・タイプの値が入ります。これは、規則的に昇順または降順にはならない 40 バイトの VARCHAR 値を戻します。したがって、ROWID データ値は、社員番号や製品番号の生成など、多くのアプリケーション用途にはあまり適していません。直接行アクセスを必要としないデータの場合は、一般に識別列を使用する方が効果的です。なぜなら、識別列には既存の数値データ・タイプが含まれており、ROWID 値には不向きなさまざまな用途に利用できるからです。

表が特定時点の状態に回復されたときに (RMVJRNCHG を使用)、識別列について生成される値のシーケンスに大きなギャップが生じることがあります。例えば、増分値を 1 とする識別列を持つ表があり、時点 T1 において最後に生成された値が 100 であり、以後データベース・マネージャーが最大 1000 までの値を生成するものとします。さらに、この表が時点 T1 にさかのぼって回復されたとします。この場合、回復の完了後最初に挿入される行の識別列の値は 1001 になり、識別列の値に 100 から 1001 というギャップが生じます。

CYCLE を指定した場合は、列に対して固有制約または固有索引が定義されていない限り、その列が GENERATED ALWAYS であっても、その列について重複値が生成されることがあります。

システム名の生成規則

システムがシステム表、ビュー、索引、または列名を生成する場合は、特定の方法があります。以下の各項では、これらの方法およびシステム名生成規則について説明します。

列名の生成の規則

システム列名は、表またはビューの作成時点でそのシステム列名の指定がなく、しかも、列名が有効なシステム列名でない場合に生成されます。

列名に特殊文字が入っておらず、しかもその長さが 10 桁を超える場合には、10 桁のシステム列名が次のように生成されます。

- その名前の最初の 5 文字
- 5 桁の固有の番号

次の例を見てください。

```
LONGCOLUMNNAME のシステム列名は LONGC00001
```

列名が区切られている場合には、

- 区切り文字と区切り文字の範囲内にある文字から、最初の 5 文字がシステム列名の最初の 5 文字として使用されます。その範囲内の文字の数が、5 文字以下の場合には、その名前の右側は、下線 (_) で埋められます。小文字は、大文字に変換されます。システム列名に使用できる有効な文字は、A ～ Z、0 ～ 9、@、#、\、および _ だけです。これら以外の文字は、いずれも下線 (_) 文字に変えられます。この結果、最初の文字が下線になる場合、最初の文字は文字 Q に変えられます。
- 上記の 5 桁の文字に 5 桁の固有の番号が付加されます。

次の例を見てください。

```
"abc" のシステム列名は ABC__00001
"COL2.NAME" のシステム列名は COL2_00001
"C 3" のシステム列名は C_3__00001
"??" のシステム列名は Q____00001
"*column1" のシステム列名は QCOLU00001
```

表名の生成の規則

表、ビュー、別名、または索引が次のいずれかの名前で作成される場合に、システム名が生成されます。

- 長さが 10 桁を超える名前
- システム名での使用が有効でない文字を含む名前

SQL 名、または対応するシステム名はいずれも、SQL ステートメントで使用して、作成済みの該当ファイルのアクセスに用いることができます。ただし、SQL 名を識別するのは、DB2 UDB for iSeries だけであり、他の環境では、システム名を使用する必要があります。

名前に特殊文字が含まれておらず、その長さが 10 桁を超えている場合には、次のように 10 桁のシステム名が生成されます。

- その名前の最初の 5 文字
- 5 桁の固有の番号

次の例を見てください。

```
LONGTABLENAME のシステム名は LONGT00001
```

その SQL 名に特殊文字が入っている場合、システム名の生成は次のようになります。

- その名前の最初の 4 文字
- 4 桁の固有の番号

さらに、

- 特殊文字は、すべて下線 (_) に置き換えられます。
- 後書きブランクは、すべてその名前から除去されます。
- その名前を有効なシステム名にする上で区切り文字が必要になる場合には、その名前は二重の引用符 (″) によって区切られます。

次の例を見てください。

```
"??" のシステム名は "__0001"
"longtablename" のシステム名は "long0001"
"LONGTableName" のシステム名は LONG0001
"A b   " のシステム名は "A_b0001"
```

SQL は相互参照ファイルを検索して、システム名が固有であることを保証します。ある名前がすでに相互参照ファイルに存在している場合、その名前が固有の名前になるまで、その番号を増やします。

上記の規則を使用しても固有名を決められない場合は、名前の番号の桁数を 1 桁追加して、固有名になるまで、または範囲の限界に達するまで、番号を増分します。例えば、″longtablename″ を作成しているときに、″long0001″ 〜 ″long9999″ がすでに存在する場合、名前は ″lon00001″ になります。

例

例 1

管理権限を持っているものとして、‘ROSSITER.INVENTORY’ という名前の表を作成します。この表は、次のような列から構成されます。

- 部品番号: 短整数、ヌル不可
- 品名 : 0 〜 24 の文字、ヌル可
- 在庫数量: 整数、ヌル可

```
CREATE TABLE ROSSITER.INVENTORY
            (PARTNO         SMALLINT     NOT NULL,
             DESCR          VARCHAR(24 ),
             QONHAND        INT )
```

例 2

DEPARTMENT という名前の表を作成します。この表は、次のような列から構成されます。

- 部門番号 : 3 文字長で、ヌルは使用できない。
- 部門名 : 0 〜 36 文字長で、ヌルは使用できない。
- 管理者番号 : 6 文字長でヌルを使用できる。

- 管理部門 : 3 文字長で、ヌルは使用できない。
- 場所 : 16 文字長でヌルを使用できる。

```
CREATE TABLE DEPARTMENT
            (DEPTNO    CHAR(3)     NOT NULL,
             DEPTNAME  VARCHAR(36) NOT NULL,
             MGRNO     CHAR(6),
             ADMRDEPT  CHAR(3)     NOT NULL,
             LOCATION  CHAR(16),
             PRIMARY KEY(DEPTNO) )
```

例 3

ビュー PRJ_LEADER の列と同じ列定義に従って、REORG_PROJECTS という名前の表を作成します。

```
CREATE TABLE REORG_PROJECTS
  LIKE PRJ_LEADER
```

CREATE TRIGGER

CREATE TRIGGER ステートメントは、現行サーバーでトリガーを定義します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。このステートメントは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメントで識別された表に対する権限とトリガー名修飾子
  - 対象表を含むライブラリーに対する *EXECUTE
  - 対象表に対する ALTER(*OBJALTER)、または WITH GRANT OPTION 特権 (*OBJMGT)
  - 対象表に対する SELECT(*OBJOPR および *READ)
  - BEFORE UPDATE トリガーに NEW 相関変数を変更する SET ステートメントが含まれている場合、対象表に対する UPDATE(*UPD および *OBJOPR)
  - トリガー名ライブラリーに対する SELECT(*OBJOPR および *READ) 特権と INSERT(*OBJOPR および *ADD) 特権
  - 物理ファイル・トリガー追加 (ADDPFTRG) コマンドに対する *USE
  - SQL 命名規則が有効で、トリガー名のスキーマ修飾子に一致するユーザー・プロファイルが存在し、その名前がステートメントのスキーマ修飾子と異なる場合には、*ALLOBJ および *SECADM 特殊権限が必要です。
- 管理権限

ステートメントの権限 ID は、以下の場合に表に対する ALTER 特権を持ちます。

- その表の所有者である。
- その表に対する ALTER 特権が認可されている。
- その表に対する *OBJALTER または *OBJMGT システム権限のいずれかが認可されている。

ステートメントの権限 ID は、以下の場合に表に対する SELECT 特権を持ちます。

- その表の所有者である。
- その表に対する SELECT 特権が認可されているか、または
- その表についての *OBJOPR および *READ システム権限が認可されている。

ステートメントの権限 ID は、次の場合に表についての UPDATE 特権を持ちます。

- その表の所有者である。
- その表またはその表の列に対する UPDATE 特権が認可されている。
- その表に対する *OBJOPR および *UPD のシステム権限が認可されている。

さらに、このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

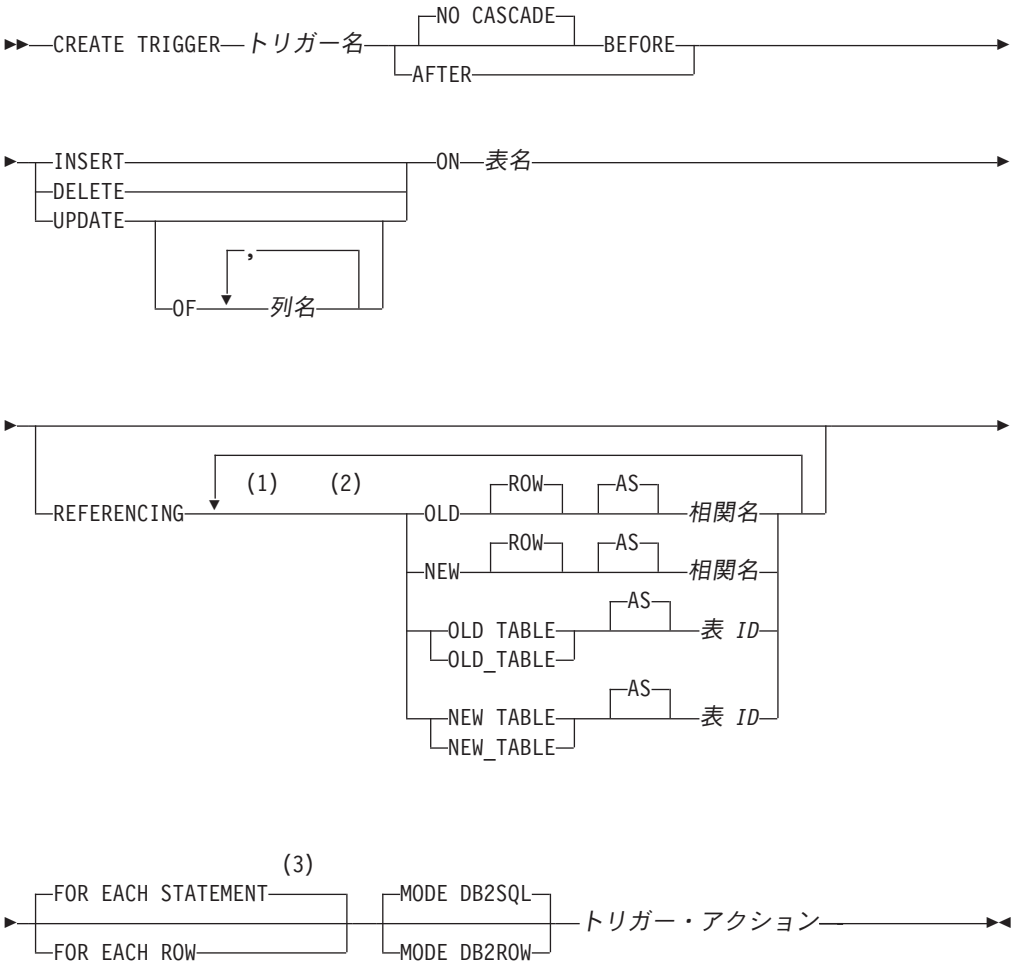
- 次のシステム権限
  - プログラム作成 (CRTPGM) コマンドに対する *USE
- 管理権限

SQL 名が指定され、該当のトリガーが作成されるライブラリーの名前と同じ名前のユーザー・プロファイルが存在し、しかもその名前がステートメントの権限 ID と異なっている場合、ステートメントの権限 ID が保持する特権には、少なくとも次の 1 つが含まれていなければなりません。

- *ALLOBJ および *SECADM 特殊権限
- 管理権限

SQL トリガー本体 内の SQL ステートメントで識別されている各表またはビューの場合、権限 ID が保持する特権には、その SQL ステートメントを実行するのに必要な特権が含まれていなければなりません。

構文



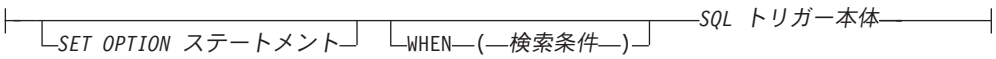
注:

1 同じ文節を複数回指定することはできません。

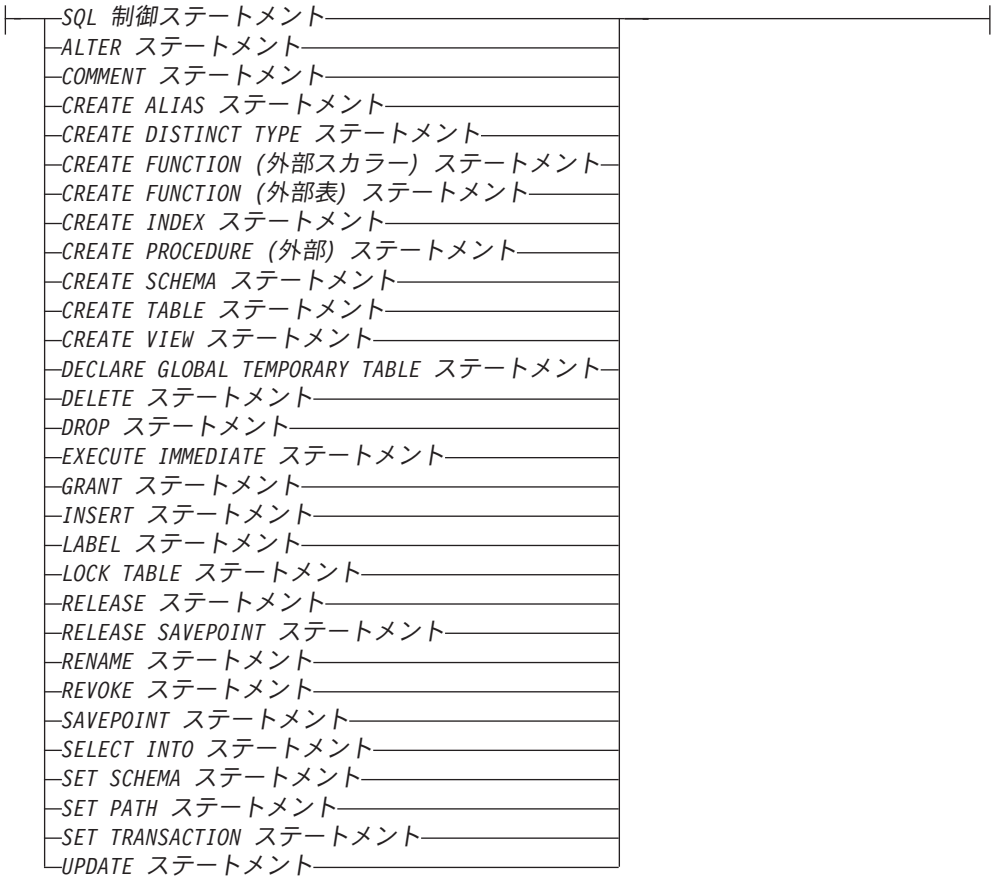
2 OLD TABLE と NEW TABLE は、それぞれ一度だけ、しかも AFTER トリガーの場合にだけ指定できます。

3 FOR EACH STATEMENT は、BEFORE トリガーの場合は指定してはなりません。

トリガー・アクション:



SQL トリガー本体:



説明

トリガー名

トリガーの名前を指定します。暗黙的または明示的修飾子も含め、この名前は、現行サーバーにすでに存在しているトリガーと同じ名前にすることはできません。QTEMP は、*トリガー名* スキーマ修飾子として使用することはできません。

SQL 名が指定されている場合、トリガーは、暗黙的または明示的修飾子で指定しているスキーマ内に作成されます。

システム名が指定されている場合、トリガーは、修飾子で指定しているスキーマ内に作成されます。修飾されていない場合、トリガーは、対象表と同じスキーマ内に作成されます。

トリガー名が有効なシステム名でない場合、または同じ名前のプログラムがすでに存在する場合、データベース・マネージャーはシステム名を生成します。名前の生成に関する規則については、572 ページの 『表名の生成の規則』 を参照してください。

NO CASCADE
NO CASCADE は、他のプロダクトとの互換性を保持するために許可されており、DB2 UDB for iSeries で使用されることはありません。

BEFORE
トリガーが前トリガーであることを指定します。データベース・マネージャーは、対象表に対する挿入、削除、または更新操作による変更を適用する前に、*トリガー・アクション* を実行します。前トリガーの*トリガー・アクション* には更新を含めることができないので、この*トリガー・アクション* は別のトリガーを起動しないことも指定します。

AFTER
トリガーが後トリガーであることを指定します。データベース・マネージャーは、対象表に対する挿入、削除、または更新操作による変更を適用した後で、*トリガー・アクション* を実行します。

INSERT
トリガーが挿入トリガーであることを指定します。データベース・マネージャーは、対象表に対する挿入操作のたびに、*トリガー・アクション* を実行します。

DELETE
トリガーが削除トリガーであることを指定します。データベース・マネージャーは、対象表に対する削除操作のたびに、*トリガー・アクション* を実行します。

UPDATE
トリガーが更新トリガーであることを指定します。データベース・マネージャーは、対象表に対する更新操作のたびに、*トリガー・アクション* を実行します。

明示的な*列名* リストが指定されていない場合、後で ALTER TABLE ステートメントによって追加される列も含めて、対象表の列に対する更新操作はすべて*トリガー・アクション* を起動します。

OF *列名*, **...**
指定する各*列名* は、対象表の列でなければならず、リストには一度しか表示できません。リストされた列に対する更新操作はすべて*トリガー・アクション* を起動します。

ON *表名*
トリガー定義の対象表を識別します。この名前は、現行サーバー上に存在する基礎表を示すものでなければならず、カタログ表、QTEMP 内の表、またはグローバル一時表を示すものであってはなりません。

REFERENCING
遷移変数 の相関名と*遷移表* の表名を指定します。*相関名* は、トリガー SQL 操作の影響を受ける行セット内の特定行を識別します。*表 ID* は、影響を受ける行セット全体を識別します。

次のように*相関名* を指定して列を修飾することにより、トリガー SQL 操作の影響を受ける各行が、*トリガー・アクション* に対して使用可能になります。

OLD ROW AS *相関名*
トリガー SQL 操作の前の行の値を識別する相関名を指定します。

NEW ROW AS *相関名*
トリガー SQL 操作とすでに実行された BEFORE トリガー内の SET ステートメントによって変更された行の値を識別する相関名を指定します。

次のように一時表名を指定することにより、トリガー SQL 操作によって影響を受ける全セット全体が、*トリガー・アクション* に対して使用可能になります。

OLD TABLE AS *表 ID*
トリガー SQL 操作の前の、影響を受ける行セット全体の値を識別する一時表の名前を指定します。現行トリガーが、あるトリガーの *SQL トリガー本体* 内のステートメントによって起動された場合、OLD TABLE には、そのトリガーによって影響を受けた行も含まれます。

NEW TABLE AS *表 ID*
トリガー SQL 操作とすでに実行された BEFORE トリガー内の SET ステートメントによって変更された、影響を受ける行セット全体の状態を識別する一時表の名前を指定します。

トリガー定義には、最大で、2 つの相関名 (OLD ROW および NEW ROW) と 2 つの表名 (OLD TABLE および NEW TABLE) を含めることができます。名前はすべて相互に固有でなければなりません。

OLD ROW *相関名* および OLD TABLE *表 ID* は、トリガー・イベントが DELETE 操作または UPDATE 操作の場合にのみ有効です。DELETE 操作の場合、OLD ROW *相関名* は、削除された行内の列の値を取り込み、OLD TABLE *表 ID* は、削除された行セット内の値を取り込みます。 UPDATE 操作の場合、OLD ROW *相関名* は、UPDATE 操作前の行の列の値を取り込み、OLD TABLE *表 ID* は、更新された行セット内の値を取り込みます。

NEW ROW *相関名* および NEW TABLE *表 ID* は、トリガー・イベントが INSERT 操作または UPDATE 操作の場合にのみ有効です。どちらの操作の場合も、NEW ROW *相関名* は、挿入または更新された行内の列の値を取り込み、NEW TABLE *表 ID* は、挿入または更新された行セット内の値を取り込みます。BEFORE トリガーの場合、更新された行の値には、BEFORE トリガーの *トリガー・アクション* 内の SET ステートメントからの変更が含まれます。

OLD TABLE と NEW TABLE は、BEFORE トリガーまたは MODE DB2ROW の場合は指定できません。

OLD ROW と NEW ROW は、FOR EACH STATEMENT トリガーの場合は指定できません。

OLD ROW および NEW ROW *相関名* 変数は、AFTER トリガー内では変更できません。

下表は、相関変数と遷移表の可能な組み合わせを要約しています。

細分性 : **FOR EACH ROW**

モード	起動時	トリガー操作	許される相関変数	許される遷移表
DB2ROW	BEFORE	DELETE	OLD	NONE
		INSERT	NEW	
		UPDATE	OLD, NEW	
	AFTER	DELETE	OLD	
		INSERT	NEW	
		UPDATE	OLD, NEW	
DB2SQL	BEFORE	DELETE	OLD	
		INSERT	NEW	
		UPDATE	OLD, NEW	
	AFTER	DELETE	OLD	OLD TABLE
		INSERT	NEW	NEW TABLE
		UPDATE	OLD, NEW	OLD TABLE, NEW TABLE

細分性 : **FOR EACH STATEMENT**

モード	起動時	トリガー操作	許される相関変数	許される遷移表
DB2SQL	AFTER	DELETE	NONE	OLD TABLE
		INSERT		NEW TABLE
		UPDATE		OLD TABLE, NEW TABLE

文字データ・タイプの遷移変数は、対象表の列の CCSID を継承します。*トリガー・アクション* の実行時に、遷移変数はホスト変数のように扱われます。したがって、文字変換が行われる可能性があります。

一時遷移表は、読み取り専用です。これは変更できません。

各*相関名* と各表 *ID* の効力範囲は、そのトリガー定義全体です。

FOR EACH ROW
データベース・マネージャーは、トリガー操作が変更する対象表の各行ごとに*トリガー・アクション* を実行することを指定します。そのトリガー操作がどの行も変更しない場合には、*トリガー・アクション* は実行されません。

FOR EACH STATEMENT
データベース・マネージャーは、トリガー操作につき一度だけ、*トリガー・アクション* を実行することを指定します。UPDATE または DELETE ステートメントのトリガーによって、どの行も影響を受けない場合でも、UPDATE または DELETE FOR EACH STATEMENT トリガーが起動されます。

FOR EACH STATEMENT は、BEFORE トリガーに対しては指定できません。

FOR EACH STATEMENT は、MODE DB2ROW トリガーに対しては指定できません。

MODE DB2SQL
MODE DB2SQL トリガーは、すべての行操作が完了した後で起動されます。

MODE DB2ROW
MODE DB2ROW トリガーは、各行の操作時に起動されます。

MODE DB2ROW は、BEFORE と AFTER 起動時の両方に有効です。

トリガー・アクション
トリガーの起動時に実行するアクションを指定します。*トリガー・アクション* は、1 つまたは複数の SQL ステートメントと、ステートメントを実行するかどうかを制御するオプション条件から構成されます。

SET OPTION ステートメント
トリガーを作成するときに使用するオプションを指定します。例えば、デバッグ可能トリガーを作成する場合は、次のステートメントを含めることができます。

```
SET OPTION DBGVIEW = *LIST
```

詳しくは、770 ページの『SET OPTION』を参照してください。

オプション CLOSQLCSR、 CNULRQD、 DFTRDBCOL、 DYNDFTCOL、および NAMING は、CREATE TRIGGER ステートメントでは使用できません。

オプション DATFMT、DATSEP、TIMFMT、および TIMSEP は、OLD ROW または NEW ROW が指定されている場合は使用できません。

WHEN (*検索条件***)**
真、偽、または不明として評価される条件を指定します。起動された SQL ステートメントは、*検索条件* が真と評価された場合にのみ実行されます。WHEN 文節を省略した場合は、関連の SQL ステートメントは常に実行されます。

SQL トリガー本体
複合ステートメントも含め、単一の SQL ステートメントを指定します。SQL トリガーの定義についての詳細は、819 ページの『第 6 章 SQL 制御ステートメント』を参照してください。

CONNECT、 SET CONNECTION、 RELEASE、 DISCONNECT、COMMIT、 ROLLBACK、 SET TRANSACTION、および SET RESULT SETS ステートメントを実行するプロシージャーへの呼び出しは、トリガーの*トリガー・アクション* 内では使用できません。

トリガーが BEFORE トリガーの場合、*SQL トリガー本体* には、INSERT、UPDATE、 DELETE、 ALTER TABLE、 COMMENT 、すべての CREATE ステートメント、DROP、すべての GRANT ステートメント、LABEL、RENAME、すべての REVOKE ステートメントを含めてはなりません。 また、SQL データを変更するプロシージャーまたは関数に対する参照を含めることもできません。

UNDO ハンドラーは、トリガーでは使用できません。

トリガー・アクション 内で参照される表、ビュー、別名、特殊タイプ、ユーザー定義関数、およびプロシージャーはすべて、そのトリガーが作成された時点での現行サーバーに存在していることが必要です。 別名が参照している表やビューも、トリガーの作成時に存在していなければなりません。これには、ライブラリー QTEMP 内のオブジェクトも含まれます。QTEMP 内のオブジェクトは*トリガー・アクション* で参照できますが、QTEMP 内のこれらのオブジェクトを除去しても、トリガーは除去されません。

トリガーの作成時に、*トリガー・アクション* は、CREATE トリガー・ステートメントの結果として、次のように変更されます。

- 命名方式が SQL 命名に切り替えられます。
- 未修飾のオブジェクト参照子は、すべて明示的に修飾されます。
- すべての暗黙の列リスト (例えば、SELECT *、列リストのない INSERT、UPDATE SET ROW) は、実際の列名リストに展開されます。

変更された*トリガー・アクション* は、カタログに保管されます。

トリガー・アクション 内のステートメントは、プロシージャーまたはユーザー定義関数が異なる活動化グループ内で実行される場合、現行サーバー以外のサーバーにアクセスできるプロシージャーまたはユーザー定義関数を呼び出すことができます。

使用上の注意

トリガーの起動

トリガーを起動できるのは、挿入、削除、更新の操作だけです。参照制約の結果として生じる削除操作は、トリガーを起動しません。したがって、次のようになります。

- DELETE トリガー・イベントを含むトリガーは、ON DELETE CASCADE 参照制約を含む表には追加できません。
- UPDATE トリガー・イベントを含むトリガーは、ON DELETE SET NULL または ON DELETE SET DEFAULT 参照制約を含む表には追加できません。

トリガーの起動によって、*トリガー・カスケード* が生じることがあります。これは、あるトリガーの起動によって、SQL ステートメントが実行され、その実行によって別のトリガーが起動されたり、同じトリガーが再度起動されたりする結果起きるものです。トリガー・アクションでは、最初の変更の結果として更新が行われ、その結果としてさらにトリガーが起動されるといったことも起こります。トリガー・カスケードを使用すると、有効なトリガー・チェーンを起動することが可能で、単一の削除、挿入、または更新ステートメントによって、データベースに対する多数の変更を行うことができます。カスケードのレベル数は、200 またはジョブやプロセスに許容される最大記憶量のいずれか先に達する値に制限されます。

制約を適用するためのトリガーの追加

すでに行が含まれている表に対してトリガーを追加しても、トリガー・アクションは実行されません。したがって、表内のデータに対して制約を適用するためのトリガーを設計した場合、既存の行内のデータは、それらの制約を満たしていない可能性があります。

複数のトリガー

1 つの表に対してトリガー SQL 操作と起動時が同一の複数のトリガーを定義できます。トリガーは、作成された順序で起動されます。最初に作成されたトリガーが最初に実行され、2 番目に作成されたトリガーが 2 番目に実行されるというようになります。

ソース表には、最大 300 のトリガーを追加できます。

対照表またはトリガー・アクション内で参照されている表への列の追加

トリガーを定義した後で対照表に列を追加する場合は、次の規則が適用されます。

- 列リストが明示的に定義されていない UPDATE トリガーの場合、新規の列の更新時にトリガーが起動されます。
- *トリガー・アクション* 内の SQL ステートメントがトリガー表を参照している場合、トリガーを再作成するまでは、新規の列は SQL ステートメントに関連付けられません。
- OLD_TABLE および NEW_TABLE 遷移表には新規の列は含まれますが、トリガーを再作成しない限り、その列を参照することはできません。

トリガー・アクション内の SQL ステートメントによって参照されている表に列を追加した場合、トリガーを再作成するまでは、新規の列は SQL ステートメントに関連付けられません。

トリガー・アクションで参照されている表の名前変更または移動

トリガー・アクション 内で参照されている表はすべて (対象表を含む) 移動や名前変更が可能です。ただし、*トリガー・アクション* は、引き続き古い名前やスキーマを参照します。*トリガー・アクション* の実行時に参照された表が見つからないと、エラーになります。そのため、トリガーをいったん除去した後、トリガーを再作成して、名前変更または移動した表を参照するようにする必要があります。

従属オブジェクト

トリガーを作成する時点で、参照されるすべてのオブジェクトが存在していることが必要です。

日時に関する考慮事項

OLD ROW または NEW ROW が指定されている場合、日付または時刻定数、あるいは*トリガー・アクション* 内の SQL ステートメントで使用されている変数内の日付と時刻のストリング表現は、ISO、EUR 、JIS、USA 形式であるか、または DDS および CRTPF CL コマンドを使用して表を作成した場合は、その表の作成時に指定された日時形式に一致していなければなりません。DDS 指定に複数の異なる日時形式が含まれている場合、トリガーは作成できません。

トリガー・アクションで参照されている表に対する特権の廃棄または取り消し

トリガー・アクション の中で参照されている表、ビュー、別名などのオブジェクトを除去すると、トリガーの起動時に、そのオブジェクトを参照しているステートメントのアクセス・プランが再作成されます。その時点でそのオブジェクトが存在していない場合は、対象表についての対応する INSERT、UPDATE、または DELETE 操作は失敗します。

トリガーの作成者がトリガー実行のために必要としている特権が取り消された場合は、トリガーの起動時に、そのオブジェクトを参照しているステートメントの

アクセス・プランが再作成されます。その時点でその特権が存在していない場合は、対象表についての対応する INSERT、UPDATE、または DELETE 操作は失敗します。

トリガーを無効化する操作

無効トリガー とは、もう起動して利用できなくなったトリガーをいいます。トリガーが無効になると、対象表に対する INSERT、UPDATE、または DELETE 操作は行えません。トリガーが無効になるのは、次のような場合です。

- *トリガー・アクション* 内の SQL ステートメントが対象表を参照しており、トリガーが自己参照トリガーであるときに、その表をシステム CRTDUPOBJ CL コマンドを使用して複写した場合。
- *トリガー・アクション* 内の SQL ステートメントが from ライブラリー内の表またはビューを参照しており、システム CRTDUPOBJ CL コマンドを使用して表を複写したときに、オブジェクトが新規ライブラリー内で見つからない場合。
- システム RSTOBJ または RSTLIB CL コマンドを使用して表を新規ライブラリーに復元したときに、*トリガー・アクション* が対象表を参照しており、トリガーが自己参照トリガーである場合。

無効トリガーは、最初にそれを除去してから、CREATE TRIGGER ステートメントを使用して再作成しなければなりません。 対象表に対してトリガー操作および起動時が同一の複数のトリガーが定義されている場合、トリガーの除去と再作成は、トリガーの起動順序に影響を与えるので注意が必要です。

トリガー実行時のエラー

SQL トリガー本体 ステートメントの実行中に発生したエラーは、SQLCODE -723 および SQLSTATE 09000 を使用して戻されます。

SQL トリガー本体 ステートメントには、SIGNAL ステートメントを含めることができます。SIGNAL ステートメントで指定された SQLCODE -438 および SQLSTATE が戻されます。

トリガー・プログラム・オブジェクト

トリガーが作成されるときに、SQL は、組み込み SQL ステートメントを含む C ソース・コードが入った一時ソース・ファイルを作成します。次いで、CRTPGM コマンドを使用して、プログラム・オブジェクトを作成します。 プログラムの作成に使用される SQL オプションは、CREATE TRIGGER ステートメントの実行時に有効なオプションです。プログラムは、ACTGRP(*CALLER) を使用して作成します。

トリガーは、トリガーの*所有者* の借用権限を使用して実行されます。

トリガーの所有権

SQL 名を指定した場合は、トリガーの*所有者* は、作成したトリガーが入れられるスキーマと同じ名前のユーザー・プロファイルです。それ以外の場合は、トリガーの*所有者*は、ステートメントを実行するジョブのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

システム名を指定した場合は、トリガーの *所有者* は、このステートメントを実行しているジョブのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

トリガーの権限

トリガー・プログラム・オブジェクトの権限は、次のとおりです。

- SQL 命名が有効の場合、トリガー・プログラムは *EXCLUDE 共通権限によって作成され、その名前のユーザー・プロファイルが存在する場合は、トリガー名のスキーマ修飾子から権限を借用します。スキーマ修飾子のユーザー・プロファイルが存在しない場合には、トリガー・プログラムの所有者は、スキーマ修飾子のユーザー・プロファイルになります。スキーマ修飾子と同じ名前のユーザー・プロファイルが存在し、その名前がステートメントの権限 ID と異なっている場合、スキーマ修飾子ライブラリー内にトリガー・プログラム・オブジェクトを作成するには、特殊権限の *ALLOBJ と *SECADM が必要です。スキーマ修飾子のユーザー・プロファイルが存在しない場合は、トリガー・プログラムの所有者は、SQL CREATE TRIGGER ステートメントを実行するジョブのユーザー・プロファイルまたはグループ・ユーザー・プロファイルになります。グループ・ユーザー・プロファイルがトリガー・プログラム・オブジェクトの所有者になるのは、ステートメントを実行するユーザーのプロファイルで OWNER(*GRPPRF) が指定された場合に限られます。トリガー・プログラムの所有者がグループ・プロファイルのメンバーであり、ユーザーのプロファイルで OWNER(*GRPPRF) が指定された場合、プログラムは、グループ・プロファイルの借用権限を使用して実行されます。
- システム命名が有効の場合、トリガー・プログラムは *EXCLUDE 共通権限によって作成され、SQL CREATE TRIGGER ステートメントを実行するジョブのユーザー・プロファイルまたはグループ・ユーザー・プロファイルから権限を借用します。

トランザクションの分離

どのトリガーも、起動されると、SET TRANSACTION ステートメントを実行するので、トリガーによる操作は、すべてそのトリガーを起動させたアプリケーション・プログラムと同じ分離レベルで実行されます。ただし、ユーザーは独自の SET TRANSACTION ステートメントを、トリガーの *SQL トリガー本体* 内の *SQL 制御ステートメント* に組み込むことができます。ユーザーが SET TRANSACTION ステートメントをトリガーの *SQL トリガー本体* に組み込んだ場合、トリガーは、そのトリガーを起動させたアプリケーション・プログラムの分離レベルではなく、SET TRANSACTION ステートメントで指定された分離レベルで実行されます。

トリガーを起動させたアプリケーション・プログラムが No Commit (COMMIT(*NONE) または COMMIT(*NC)) 以外の分離レベルで実行されている場合、トリガー内部の操作はコミットメント制御下で実行され、アプリケーションが現行作業単位をコミットするまでは、コミットやロールバックは行われません。トリガーの *SQL トリガー本体* で ATOMIC が指定され、ATOMIC トリガーを起動させたアプリケーション・プログラムが分離レベル No Commit (COMMIT(*NONE) または COMMIT(*NC)) で実行されている場合、トリガー内部の操作はコミットメント制御下では実行されません。トリガーを起動させたアプリケーションが分離レベル No Commit (COMMIT(*NONE) または COMMIT(*NC)) で実行されている場合、トリガーの操作は即時にデータベースに書き込まれ、ロールバックすることはできません。

ある表に対して、物理ファイルトリガー追加 (ADDPFTRG) CL コマンドによって定義されたシステム・トリガーと、CREATE TRIGGER ステートメントによ

って定義された SQL トリガーの両方が定義されている場合、システム・トリガーが SET TRANSACTION ステートメントを実行して、トリガーを起動した元のアプリケーションと同じ分離レベルで実行されるようにすることをお勧めします。また、システム・トリガーは、呼び出し元アプリケーションの活動化グループ内で実行することもお勧めします。システム・トリガーを別の活動化グループ (ACTGRP(*NEW)) で実行すると、それらのシステム・トリガーは、呼び出し元アプリケーションの作業単位に参加せず、SQL トリガーの作業単位にも参加しません。別の活動化グループで実行されるシステム・トリガーは、コミットメント制御下で実行したデータベース操作をコミットまたはロールバックする責任があります。 CREATE TRIGGER ステートメントによって定義された SQL トリガーは、常に呼び出し元の活動化グループ内で実行されます。

トリガー・アプリケーションがコミットメント制御を使用して実行されている場合、SQL トリガーの操作およびカスケード SQL トリガーは、副作業単位に取り込まれます。トリガーの操作およびカスケード・トリガーが成功した場合、副作業単位に取り込まれた操作は、トリガー・アプリケーションが現行の作業単位をコミットまたはロールバックする時点で、コミットまたはロールバックされます。呼び出し元と同じ活動化グループ内で実行され、呼び出し元の分離レベルで SET TRANSACTION を実行するシステム・トリガーも、副作業単位に参加します。トリガー・アプリケーションがコミットメント制御を使用せずに実行されている場合、SQL トリガーの操作もコミットメント制御を使用せずに実行されます。

トリガーを起動させたアプリケーションが分離レベル No Commit (COMMIT(*NONE) または COMMIT(*NC)) で実行されており、INSERT、UPDATE、または DELETE ステートメントを実行して、ステートメントの実行中にエラーが発生した場合、その操作のエラーの後は、他のシステム・トリガーや SQL トリガーは起動されません。ただし、いくつかの変更はすでに行われています。トリガー・アプリケーションがコミットメント制御を使用して実行されている場合、副作業単位に取り込まれたトリガーの操作は、最初のエラーが検出された時点でロールバックされ、現行の INSERT、UPDATE、または DELETE ステートメントの追加のトリガーは起動されません。

例

例 1

会社が管理する従業員の数を追跡する 2 つのトリガーを作成します。トリガー表は EMPLOYEE 表で、トリガーは COMPANY_STATS 表の総従業員数の列を増分または減分します。COMPANY_STATS 表のプロパティーは、次のとおりです。

```
CREATE TABLE COMPANY_STATS
  (NBEMP INTEGER,
   NBPRODUCT INTEGER,
   REVENUE DECIMAL(15,0))
```

この例は、行トリガーを使用して、要約データを別表に保守します。

最初のトリガー NEW_HIRE を作成して、新しい従業員が雇用されるたびに従業員数を増分するようにします。つまり、新しい行が EMPLOYEE 表に挿入されるたびに、表 COMPANY_STATS の列 NBEMP を 1 だけ増分します。

```
CREATE TRIGGER NEW_HIRE
  AFTER INSERT ON EMPLOYEE
  FOR EACH ROW MODE DB2SQL
  BEGIN ATOMIC
    UPDATE COMPANY_STATS SET NBEMP = NBEMP + 1;
  END
```

2 番目のトリガー FORM_EMP を作成して、従業員が会社を辞めるたびに従業員数を減分するようにします。つまり、表 EMPLOYEE から行が削除されるたびに、表 COMPANY_STATS の列 NBEMP を 1 だけ減分します。

```
CREATE TRIGGER FORM_EMP
  AFTER DELETE ON EMPLOYEE
  FOR EACH ROW MODE DB2SQL
  BEGIN ATOMIC
    UPDATE COMPANY_STATS SET NBEMP = NBEMP - 1;
  END
```

例 2

トリガー REORDER を作成します。これは、部品レコードが更新され、該当部品の手持ちの数量が最大在庫量の 10% を下回るたびに出荷要求を出すために、ユーザー定義関数 ISSUE_SHIP_REQUEST を呼び出します。ユーザー定義関数 ISSUE_SHIP_REQUEST は、その部品の最大在庫量から手持ちの数量を差し引いた数量の部品を発注します。この関数は、要求が適切な供給業者に送信されることも保証します。

部品行は PARTS 表に入っています。 この表には、その他の列も含まれていますが、トリガーは列 ON_HAND または MAX_STOCKED が更新された場合にのみ起動されます。

```
CREATE TRIGGER REORDER
  AFTER UPDATE OF ON_HAND, MAX_STOCKED ON PARTS
  REFERENCING NEW AS NROW
  FOR EACH ROW MODE DB2SQL
  WHEN (NROW.ON_HAND < 0.10 * NROW.MAX_STOCKED)
  BEGIN ATOMIC
    VALUES(ISSUE_SHIP_REQUEST(NROW.MAX_STOCKED - NROW.ON_HAND, NROW.PARTNO));
  END
```

例 3

例 2 のシナリオと同じですが、ユーザー定義関数を呼び出すのに、VALUES ステートメントの代わりに全選択を使用します。この例は、行トリガーではなく、ステートメント・トリガーとしてトリガーを定義する方法も示しています。WHERE 文節で真と評価された遷移表内の各行について、その部品の出荷要求が出されます。

```
CREATE TRIGGER REORDER
  AFTER UPDATE OF ON_HAND, MAX_STOCKED ON PARTS
  REFERENCING NEW TABLE AS NTABLE
  FOR EACH STATEMENT MODE DB2SQL
  BEGIN ATOMIC
    SELECT ISSUE_SHIP_REQUEST(MAX_STOCKED - ON_HAND, PARTNO)
    FROM NTABLE
    WHERE ON_HAND < 0.10 * MAX_STOCKED;
  END
```

CREATE VIEW

CREATE VIEW ステートメントは、現行サーバーに 1 つまたは複数の表またはビューに関するビューを作成します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことも、対話式に呼び出すこともできます。このステートメントは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- 次のシステム権限
  - 論理ファイル作成 (CRTLF) CL コマンドに対する *USE 権限
  - 該当のビューが作成されるライブラリーに対する *EXECUTE および *ADD 権限
  - データ・ディクショナリーに対する *CHANGE 権限。ただし、ビューが作成されるライブラリーが、データ・ディクショナリーをもつ SQL スキーマの場合。
- 管理権限

このステートメントの権限 ID によって保持される特権には、少なくとも次の 1 つが含まれていなければなりません。

- 全選択を介して直接的に参照されたり、あるいは、全選択で参照されるビューを介して間接的に参照されるそれぞれの表とビューに対する次の特権。
  - 表やビューに対する SELECT 特権、および
  - 表やビューが入っているライブラリーに対する *EXECUTE システム権限
- 管理権限

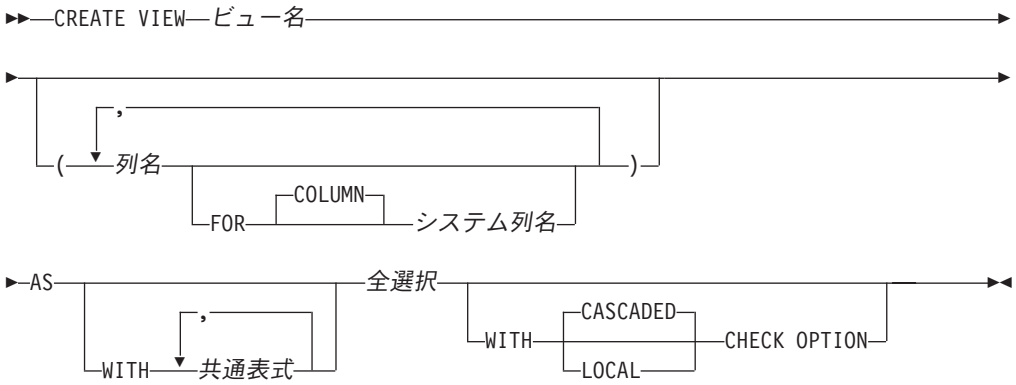
ステートメントの権限 ID は、以下の場合に表に対する SELECT 特権を持ちます。

- その表の所有者である。
- その表に対する SELECT 特権が認可されているか、または
- その表についての *OBJOPR および *READ システム権限が認可されている。

ステートメントの権限 ID は、以下の場合にビューに対する SELECT 特権を持ちます。

- そのビューの所有者である。
- そのビューに対する SELECT 特権が認可されているか、または
- そのビューについての *OBJOPR および *READ システム権限、およびそのビューが直接または間接に従属しているすべての表やビューについての *READ システム権限が認可されている。すなわち、そのビューの定義で参照されている表やビューのすべて、またそのようなビューが参照される場合、そのビューの定義で参照されている表やビューのすべて、以下同様。

構文



説明

ビュー名

ビューの名前を指定します。暗黙的または明示的修飾子も含め、この名前は、サーバーにすでに存在している表、ビュー、索引、別名、またはファイルと同じ名前にすることはできません。

SQL 名が指定されている場合、ビューは、暗黙的または明示的修飾子で指定しているスキーマ内に作成されます。

システム名が指定されている場合、ビューは、修飾子で指定しているスキーマ内に作成されます。修飾されていない場合、ビュー名は、最初の FROM 文節上に指定されている、最初の表と同じスキーマ内に作成されます (任意の共通表式またはネストされた表式の FROM 文節を含む) 。

ビュー名が有効なシステム名でない場合、DB2 UDB for iSeries SQL は、システム名を生成します。名前の生成に関する規則については、572 ページの 『表名の生成の規則』を参照してください。

(列名, ...)

このビューの列の名前を指定します。列名のリストを指定する場合は、そのリストは、全選択の結果表にある列の数と同じ数の列名で構成されている必要があります。 それぞれの*列名* およびシステム*列名* は固有でなければならず、修飾は付けられません。列名のリストを指定しなかった場合は、ビューの列は、全選択の結果表の列名および列のシステム名を継承します。

副選択の結果表に、重複する列名、重複するシステム列名、または名前なしの列がある場合は、列名 (およびシステム列名) のリストを指定する必要があります。名前が指定されない列についての詳細は、343 ページの 『結果列の名前』を参照してください。

FOR COLUMN *システム列名*

列の OS/400 名を指定します。ビューの複数の列またはビューの列名に、同じ名前を使用してはなりません。

システム列名が指定されず、また列名が有効なシステム列名でない場合には、システム列名が生成されます。システム列名の生成方法に関する詳細については、572 ページの 『列名の生成の規則』を参照してください。

AS *全選択*

ビューを定義します。ビューは、常に、全選択が実行された場合に結果として生じる行から構成されます。

共通表式 は、後に続く全選択で使用するための共通表式を定義します。 詳細については、356 ページの『共通表式』を参照してください。

全選択 では、ホスト変数を参照することはできません。 *全選択* の説明については、353 ページの 『全選択』を参照してください。

WITH CASCADED CHECK OPTION

このビューを介して挿入または更新される行は、すべてがこのビューの定義に適合しなければならないことを指定します。このビューの定義に適合しない行は、このビューを使用して検索することができない行です。

以下の場合は、WITH CHECK OPTION は指定できません。

- ビューが読み取り専用である。
- ビューの定義に副照会が含まれている。
- 定義の中の WHERE 文節にスカラー副選択が含まれている。
- ビューの定義に非決定的関数が含まれている。

挿入を許さない更新可能ビューに関して WITH CHECK OPTION を指定した場合は、検査オプションは更新のみに適用されます。

WITH CHECK OPTION を指定しない場合は、ビューの定義は、そのビューを使用するいずれの挿入または更新操作のチェックにも使用されません。ただし、そのビューが WITH CHECK OPTION を伴う他のビューに直接または間接に従属している場合には、挿入または更新の操作の過程でなおチェックが行われます。そのビューの定義は使用されないので、そのビューの定義に適合しない行がそのビューを介して挿入、または更新されることがあります。

ビュー V に関する WITH CHECK OPTION は、V に直接または間接的に従属している更新可能などのビューにも継承されます。したがって、更新可能なビューが V 上に定義されている場合は、そのビューに対して WITH CHECK OPTION が指定されていなくても、V に関する検査オプションはそのビューにも適用されます。例えば、次のような更新可能なビューを想定します。

```
CREATE VIEW V1 AS SELECT COL1 FROM T1 WHERE COL1 > 10

CREATE VIEW V2 AS SELECT COL1 FROM V1 WITH CHECK OPTION

CREATE VIEW V3 AS SELECT COL1 FROM V2 WHERE COL1 < 100
```

V1 には WITH CHECK OPTION の指定がなく、また V1 は WITH CHECK OPTION の指定を持つ他のビューに従属していないので、V1 を使用する次の INSERT ステートメントは、正しく実行されます。

```
INSERT INTO V1 VALUES(5)
```

V2 には WITH CHECK OPTION の指定があり、挿入は V2 の定義に適合しない行を生成する可能性があるので、V2 を使用する次の INSERT ステートメントは、エラーになるはずです。

```
INSERT INTO V2 VALUES(5)
```

V3 を使用する次の INSERT ステートメントは、V3 には WITH CHECK OPTION の指定がなくても、V3 が WITH CHECK OPTION の指定を伴う V2 に従属しているので、エラーになります。

```
INSERT INTO V3 VALUES(5)
```

V3 を使用する次の INSERT ステートメントは、V3 の定義には適合していなくても (V3 には WITH CHECK OPTION の指定がない)、V2 の定義には適合しているので (WITH CHECK OPTION ので指定がある)、正常に実行されます。

```
INSERT INTO V3 VALUES(200)
```

WITH LOCAL CHECK OPTION

WITH LOCAL CHECK OPTION は、次の点を除いて、WITH CASCADED CHECK OPTION と同等です。すなわち、WITH LOCAL CHECK OPTION を指定して定義されたビューでは、行の更新によってその行がそのビューの定義に適合しなくなる場合でも、なおその行の更新が可能である点が異なります。 これは、そのようなビューが、その定義に WITH CASCADED CHECK OPTION または WITH LOCAL CHECK OPTION のどちらの文節も指定されていないビューに直接、または間接に従属している場合にのみ起こります。

WITH LOCAL CHECK OPTION は、行を挿入または更新するときに、WITH LOCAL CHECK OPTION または WITH CASCADED CHECK OPTION を備えている従属ビューだけの検索条件を検査するよう指定します。これに対して、WITH CASCADED CHECK OPTION は、行の挿入または更新の時点で、すべての従属ビューの検索条件がチェックされることを指定します。

CASCADED と LOCAL の相違点を例によって示します。次のような更新可能なビューについて考えます。この場合の x と y は、LOCAL か CASCADED のどちらかを表します。

- V1 は T0 で定義されている。
- V2 は V1 WITH x CHECK OPTION を指定して定義されている。
- V3 は V2 で定義されている。
- V4 は V3 WITH y CHECK OPTION を指定して定義されている。
- V5 は V4 で定義されている。

次の表は、INSERT または UPDATE の操作中に、どのビューの検索条件がチェックされるかを示しています。

表 48. INSERT および UPDATE の過程でその検索条件がチェックされるビュー

INSERT または UPDATE で 使用されるビュー	**x = LOCAL y = LOCAL**	**x = CASCADED y = CASCADED**	**x = LOCAL y = CASCADED**	**x = CASCADED y = LOCAL**
V1	なし	なし	なし	なし
V2	V2	V2 V1	V2	V2 V1
V3	V2	V2 V1	V2	V2 V1
V4	V4 V2	V4 V3 V2 V1	V4 V3 V2 V1	V4 V2 V1
V5	V4 V2	V4 V3 V2 V1	V4 V3 V2 V1	V4 V2 V1

使用上の注意

削除可能なビュー : 以下の条件がすべて満たされている場合は、カーソルは*削除可能* です。

- 外側の全選択が、1 つの基礎表または削除可能ビューのみを示している。
- 外側の全選択に、GROUP BY 文節または HAVING 文節が含まれていない。
- 外側の全選択の選択リストに列関数が含まれていない。
- 外側の全選択に UNION 演算子または UNION ALL 演算子が含まれていない。
- 外側の全選択に DISTINCT 文節が含まれていない。

更新可能なビュー : 以下の条件がすべて満たされている場合は、ビューの列は*更新可能* です。

- ビューが削除可能である。
- その列が、表の 1 つの列または他のビューの更新可能な列からのみ派生したものである。すなわち、結果の列は、演算子、スカラー関数、定数、またはそれ自体がそのような式から取り出された列を含む式から取り出される列であってはなりません。

ビューは、UPDATE ステートメントのオブジェクト表にはなれません。ただし、最初の SELECT 文節に、1 つの列からだけ取り出される結果の列が少なくとも 1 つ含まれている場合は、その限りではありません。すなわち、結果の列は、演算子、スカラー関数、定数、またはそれ自体がそのような式から取り出された列を含む式から取り出される列であってはなりません。

ビューが*削除可能* になるのは、ビュー内のすべての列が*削除可能* な場合です。

挿入可能なビュー : ビューの列が 1 つでも更新可能であれば、そのビューは*挿入可能* です。

読み取り専用のビュー : 削除可能でないビューは*読み取り専用* です。読み取り専用ビューに対して INSERT、UPDATE、または DELETE ステートメントを実行することはできません。

削除可能でないカーソルは*読み取り専用* です。

ソート順序 : ビューは、CREATE VIEW ステートメントの実行時に効力を持っているソート順序に従って作成されます。ビューのソート順序は、そのビューの全選択における SBCS データおよび混合データに関連するすべての比較で使用されます。照会にビューが含まれる場合は、そのビューの全選択から中間結果表が作成されます。照会を実行するときに有効なソート順序が、その照会で指定される選択すべてに適用されます。

ビューの属性 : ビューは、キーのない論理ファイルとして作成されます。ビューが作成される場合、ファイル待ち時間とレコード待ち時間の属性は、論理ファイル作成 (CRTLF) コマンドの WAITFILE キーワードと WAITRCD キーワード上に指定されたデフォルト値に設定されます。

分散表を介して作成されるビューは、その表が配布されるすべてのシステム上で作成されます。ビューが複数の分散表に作成され、その表が同一のノード・グループを使用して配布されない場合は、そのビューは、CREATE VIEW ステートメントを

実行するシステムにのみ作成されます。分散表の詳細については、DB2 UDB for iSeries マルチ・システムを参照してください。

ビューの所有権 : SQL 名を指定した場合は、ビューの*所有者* は、作成したビューが入れられるスキーマと同じ名前のユーザー・プロファイルです。 それ以外の場合は、ステートメントを実行するジョブのユーザー・プロファイルまたはグループ・ユーザー・プロファイルがビューの*所有者* になります。

システム名を指定した場合は、ビューの *所有者* は、このステートメントを実行しているジョブのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

ビューの権限 : SQL 名を使用する場合は、ビューは、*PUBLIC に対するシステム権限 *EXCLUDE を使用して作成されます。システム名を使用する場合、ビューは、スキーマの作成権限 (CRTAUT) パラメーターによって決められる *PUBLIC に対する権限を使用して作成されます。

ビューの所有者がグループ・プロファイルのメンバー (GRPPRF キーワード) であり、グループ権限が指定されている (GRPAUT キーワード) 場合は、そのグループ・プロファイルにも、そのビューに対する権限が与えられます。

所有者は、所有するビューについての SELECT 特権や除去する権限を必ず獲得します。 SELECT 特権は、所有者が全選択で識別された表やビューすべてについての SELECT 特権を認可する権限も持っている場合にのみ、他のユーザーに認可することができます。

また、所有者はそのビューについての INSERT、UPDATE、および DELETE 特権も入手することがあります。ビューが読み取り専用でない場合、全選択の最初の FROM 文節で識別された表やビューに対して所有者が持つ特権と同じ特権を新たなビューについても獲得することになります。そのような特権が認可できるのは、それらの元となっている特権も認可できる場合だけに限られます。

識別列 : 列が識別列と見なされるのは、ビュー定義の全選択の中の対応する列のエレメントが、表の識別列の名前であるか、基礎表の識別列の名前に直接または間接的にマップされるビューの列の名前である場合です。その他の場合は、ビューの列には識別プロパティーは与えられません。例えば、

- ビュー定義の選択リストに、識別列の名前の複数のインスタンスが含まれている (つまり同じ列を複数回選択している) 場合。
- ビュー定義に結合が含まれている場合。
- ビュー定義の中の列のいずれかに、識別列を参照する式が含まれている場合。
- ビュー定義に UNION が含まれている場合。

ビューの制限 : ビューは、UPDATE ステートメントのオブジェクト表にはなれません。ただし、最初の SELECT 文節に、1 つの列からだけ取り出される結果の列が少なくとも 1 つ含まれている場合は、その限りではありません。すなわち、結果の列は、演算子、スカラー関数、定数、またはそれ自体がそのような式から取り出された列を含む式から取り出される列であってはなりません。

1 つのビューから参照できる実表は、そのビューの基礎となっているビューから参照している実表を含めて 32 個までです。

1 つのビューに指定できる列の数は 8000 までです。この数は、そのビューで参照している表の数、列名の長さ、および WHERE 文節の長さによっても減少します。

ビュー定義のテスト : ユーザーは、SELECT * FROM ビュー名 を実行することによって、定義したビューが持つ意味をテストすることができます。

例

例 1

表 PROJECT をもとに、MA_PROJ という名前のビューを作成します。この表には PROJNO (プロジェクト番号) が 'MA' という文字で始まっている行だけを入れます。

```
CREATE VIEW MA_PROJ
  AS SELECT * FROM PROJECT
        WHERE SUBSTR(PROJNO, 1, 2) = 'MA'
```

例 2

例 1 と同じようにビューを作成します。ただし、このビューでは、PROJNO (プロジェクト番号)、PROJNAME (プロジェクト名)、および RESPEMP (プロジェクトに関与している従業員) の各列だけを選択します。

```
CREATE VIEW MA_PROJ2
  AS SELECT PROJNO, PROJNAME, RESPEMP FROM PROJECT
       WHERE SUBSTR(PROJNO, 1, 2) = 'MA'
```

例 3

例 2 と同じようにビューを作成します。ただし、このビューでは、プロジェクト IN_CHARGE に関与している従業員について列を呼び出します。

```
CREATE VIEW MA_PROJ (PROJNO, PROJNAME, IN_CHARGE)
  AS SELECT PROJNO, PROJNAME, RESPEMP FROM PROJECT
       WHERE SUBSTR(PROJNO, 1, 2) = 'MA'
```

注: 列名を 1 つだけ変更する場合でも、MA_PROJ の後の括弧内に、ビューを構成する 3 つの列の名前をすべて指定しなければなりません。

例 4

PRJ_LEADER という名前のビューを作成します。このビューには、PROJECT 表の最初の 4 つの列 (PROJNO, PROJNAME, DEPTNO, RESPEMP) と、そのプロジェクトの責任者 (RESPEMP) の名字 (LASTNAME) を合わせて入れます。名前は、表 EMPLOYEE 内の EMPNO と表 PROJECT 内の RESEMP を突き合わせることによって、表 EMPLOYEE から取得しています。

```
CREATE VIEW PRJ_LEADER
  AS SELECT PROJNO, PROJNAME, DEPTNO, RESPEMP, LASTNAME
       FROM PROJECT, EMPLOYEE
       WHERE RESPEMP = EMPNO
```

例 5

例 4 と同じようにビューを作成します。ただし、このビューには、PROJNO、PROJNAME、DEPTNO、RESEMP、および LASTNAME の各列に加えて、責任者の給与総額 (SALARY + BONUS +COMM) を入れます。さらに、このビューでは、PRSTAFF (平均人員数) が 1 より大きいプロジェクトだけを選択しています。

```
CREATE VIEW PRJ_LEADER (PROJNO, PROJNAME, DEPTNO, RESPEMP, LASTNAME, TOTAL_PAY)
  AS SELECT PROJNO, PROJNAME, DEPTNO, RESPEMP, LASTNAME, SALARY+BONUS+COMM
       FROM PROJECT, EMPLOYEE
       WHERE RESPEMP = EMPNO AND PRSTAFF > 1
```

DECLARE CURSOR

DECLARE CURSOR ステートメントは、カーソルを定義します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込んで使用します。それ以外の使用法はありません。このステートメントは、実行可能ステートメントではありません。Java では指定できません。

権限

このステートメントを使用するための権限は不要です。ただし、カーソルに関して OPEN または FETCH を使用するには、ステートメントの権限 ID によって保持される特権に、少なくとも次の 1 つが含まれていなければなりません。

- 該当のカーソルの SELECT ステートメントで識別される各表またはビューに対して、
  - 表やビューに対する SELECT 特権、および
  - 表やビューが入っているライブラリーに対する *EXECUTE システム権限
- 管理権限

ステートメントの権限 ID は、以下の場合に表に対する SELECT 特権を持ちます。

- その表の所有者である。
- その表に対する SELECT 特権が認可されているか、または
- その表に対する *OBJOPR および *READ のシステム権限が認可されている。

ステートメントの権限 ID は、以下の場合にビューに対する SELECT 特権を持ちます。

- そのビューの所有者である。
- そのビューに対する SELECT 特権が認可されているか、または
- そのビューについての *OBJOPR および *READ システム権限、およびそのビューが直接または間接に従属しているすべての表やビューについての *READ システム権限が認可されている。すなわち、そのビューの定義で参照されている表やビューのすべて、またそのようなビューが参照される場合、そのビューの定義で参照されている表やビューのすべて、以下同様。

カーソルの SELECT ステートメントは、次のいずれかです。

- *ステートメント名* によって識別される準備済み選択ステートメント。
- 指定した*選択ステートメント*。

ステートメント名 を指定した場合は、

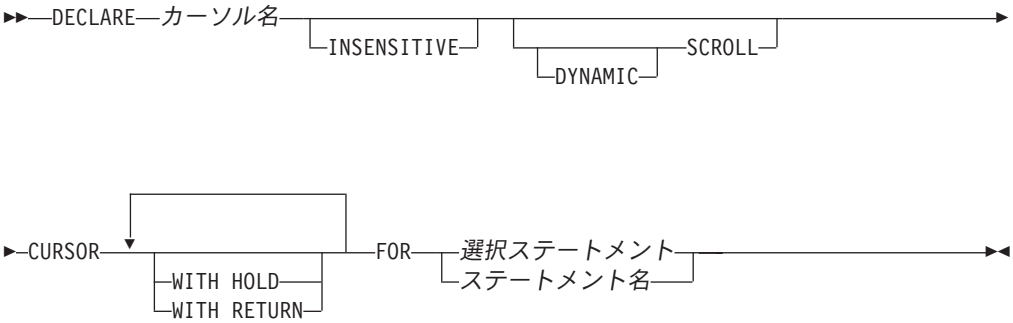
- プログラムの作成時点の CRTSQLxxx コマンドに DYNUSRPRF(*OWNER) が指定されていた場合を除き、ステートメントの権限 ID は、実行時の権限 ID です。詳細は、60 ページの 『権限 ID と権限名』を参照してください。
- CRTSQLxxx コマンドで DLYPRP(*YES) が指定されていない場合には、選択ステートメントを準備するときに権限検査が行われます。

- DLYPRP (*YES) パラメーターを使用してコンパイルされているプログラムについては、カーソルをオープンするときに権限検査が行われます。

選択ステートメント を指定した場合は、

- SQL 命名を指定した USRPRF(*OWNER) または USRPRF(*NAMING) が、CRTSQLxxx コマンドで指定された場合は、ステートメントの権限 ID は、その SQL プログラムまたはパッケージの所有者です。
- システム命名を指定した USRPRF(*USER) または USRPRF(*NAMING) が、CRTSQLxxx コマンドで指定された場合は、ステートメントの権限 ID は、実行時権限 ID です。
- REXX では、ステートメントの権限 ID は、実行時権限 ID です。
- カーソルがオープンされるときには、権限検査が実行されます。

構文



説明

カーソル名
カーソルの名前を指定します。ソース・プログラムで宣言されている、他のカーソルと同じ名前を指定してはなりません。

INSENSITIVE
これを指定すると、カーソルは、そのオープン後は、この活動化グループや他の活動化グループで実行する挿入、更新、または削除を検知しなくなります。INSENSITIVE を指定すると、カーソルは読み取り専用になり、カーソルのオープン時に一時結果が作成されます。さらに、SELECT ステートメントに FOR UPDATE 文節を使用することができなくなり、しかも、アプリケーションでは、データ (ALWCPYDTA(*OPTIMIZE) または ALWCPYDTA(*YES)) のコピーを許可する必要が生じます。

SCROLL
カーソルがスクロール可能であることを指定します。カーソルは、他の活動化グループによって行われる挿入、更新、および削除をただちに検知する場合とそうでない場合があります。DYNAMIC の指定がない場合、カーソルは読み取り専用になります。さらに、SELECT ステートメントには FOR UPDATE 文節を使用できません。

DYNAMIC SCROLL
結果表が更新可能であればカーソルが更新可能であること、またカーソルは、他

のアプリケーション処理が行った挿入、更新、および削除に通常は即座に検知することを指定します。ただし、次の場合には、キーワード DYNAMIC は無視され、カーソルは挿入、更新、および削除に即応しません。

- 一時的結果表として組み込まれている照会。一時的結果表は、下記の場合に作成されます。
  - INSENSITIVE が指定された場合。
  - ORDER BY 文節に指定した列に対する記憶域の合計長が 2000 バイトを超えている場合。
  - ORDER BY 文節と GROUP BY 文節の指定する列が異なる場合、または列の指定の順序が異なる場合。
  - ORDER BY 文節と GROUP BY 文節に、ユーザー定義の関数、あるいは、スカラー関数である DLVALUE、DLURLPATH、DLURLPATHONLY、DLURLSERVER、DLURLSCHEME、または、属性が FILE LINK CONTROL と READ PERMISSION DB のデータ・リンクの場合は DLURLCOMPLETE のいずれかが含まれている場合。
  - UNION 文節または DISTINCT 文節が指定されている場合。
  - ORDER BY 文節または GROUP BY 文節で指定されている列が、すべて同じ表のものでない場合。
  - JOINDFT データ記述仕様 (DDS) キーワードによって定義された論理ファイルが、別のファイルに結合されている場合。
  - 複数のデータベース・ファイルのメンバーに基づく論理ファイルが指定されている場合。
  - DECLARE CURSOR の選択ステートメントが GROUP BY 文節を使用しているときに、FETCH ステートメントで CURRENT または RELATIVE スクロール・オプションが指定されている場合。
- 次のような副照会が組み込まれている照会。
  - 最外部の照会が内部の副選択に相関値を提供しない場合。
  - 最外部の照会で、IN、= ANY、= SOME、または <> ALL 副照会を参照しない場合。

WITH HOLD

コミット操作の結果として、カーソルがクローズされるのを防止します。WITH HOLD 文節を使用して宣言されたカーソルがコミット時点で暗黙にクローズするのは、そのカーソルに関連する接続がコミット操作中に終了する場合だけです。

WITH HOLD の指定がある場合、コミット操作はその時点の作業単位における変更をすべてコミットし、そのカーソルを維持する上で必要としないロックだけを解放します。後で、位置指定 UPDATE または DELETE ステートメントを実行するのに先立って、FETCH ステートメントが必要になります。

カーソルはすべて、CONNECT (タイプ 1) またはロールバック操作によって暗黙にクローズされます。ある接続に関連するカーソルはすべて、その接続の切り離しによって暗黙にクローズされます。カーソルは、WITH HOLD が指定されていない場合、あるいは、そのカーソルに関連した接続が解放保留状態にある場合にも、コミット操作によって暗黙にクローズされます。

カーソルがコミット操作の前にクローズされた場合、その効果は、そのカーソルが WITH HOLD オプションを指定せずに宣言されたのと同一です。

WITH RETURN
この文節は、カーソルはプロシージャーからの結果セットとして使用することを示します。 WITH RETURN は、プロシージャーのソース・コードに DECLARE CURSOR ステートメントが含まれている場合にだけ有効です。 それ以外の場合、プリコンパイラーはこの文節を受け入れますが、無効です。

SQL プロシージャー内では、SQL プロシージャーの終了時にまだオープンされている、WITH RETURN 文節を使用して宣言されたカーソルは、SQL プロシージャーからの結果セットを定義しています。SQL プロシージャー内のその他のオープン・カーソルは、SQL プロシージャーの終了時にクローズされます。外部プロシージャー (LANGUAGE SQL を使用して定義されたもの以外) では、WITH RETURN 文節は無効であり、外部プロシージャーの終了時にオープンしているカーソルはすべて結果セットと見なされます。

結果セットには、現行カーソル位置から、プロシージャーが呼び出し元に戻る時点での結果セットの最後までのすべての行が含まれます。

選択ステートメント
カーソルの SELECT ステートメントを指定します。詳細については、355 ページの『選択ステートメント』を参照してください。

選択ステートメント にはパラメーター・マーカーを含めてはなりません (REXX は例外) が、ホスト変数への参照は含めることができます。
RPG、PL/I、および REXX 以外のホスト言語では、ソース・プログラムにおけるホスト変数の宣言は、DECLARE CURSOR ステートメントよりも前になければなりません。RPG と PL/I の場合、ホスト変数宣言は、DECLARE CURSOR ステートメントの後に指定することができます。REXX の場合は、ホスト変数の代わりにパラメーター・マーカーを使用する必要があり、しかも、ステートメントを準備しておく必要があります。

ステートメント名
カーソルの SELECT ステートメントは、そのカーソルのオープンの時点でステートメント名 によって識別される準備済み選択ステートメントです。このステートメント名 は、ソース・プログラムの他の DECLARE CURSOR ステートメントで指定されているステートメント名 と同じであってはなりません。準備済みステートメントについての詳細は、725 ページの『PREPARE』を参照してください。

使用上の注意

DECLARE CURSOR ステートメントは、該当するカーソルを明示的に参照するどのステートメントよりも前に置かなければなりません。

カーソルの結果表 : オープン状態にあるカーソルは、*結果表* と、その結果表の行に対する相対的な位置を指示します。指示される表は、該当するカーソルの SELECT ステートメントで指定されている結果表です。

以下の条件がすべて満たされている場合は、カーソルは*削除可能* です。
- 外側の全選択が、1 つの基礎表または削除可能ビューのみを示している。
- 外側の全選択に、GROUP BY 文節または HAVING 文節が含まれていない。

- 外側の全選択の選択リストに列関数が含まれていない。
- 外側の全選択に UNION 演算子または UNION ALL 演算子が含まれていない。
- 外側の全選択に DISTINCT 文節が含まれていない。
- 選択ステートメントに ORDER BY 文節が含まれていて、FOR UPDATE OF 文節または DYNAMIC SCROLL が指定されている。
- 選択ステートメントに FOR READ ONLY 文節が含まれていない。
- 選択ステートメントに FETCH FIRST n ROWS ONLY 文節が含まれていない。
- 外側の全選択の結果に一時表が使用されていない。
- DYNAMIC キーワードが指定されていない場合に、選択ステートメントに SCROLL キーワードが含まれていない。
- FOR UPDATE OF 文節が指定されていない場合に、選択リストに DATALINK 列が含まれていない。

以下のすべての条件が満たされている場合は、カーソルに関連した外側の全選択の選択リスト内の結果列は*更新可能* です。

- カーソルが削除可能である。
- 結果列が、表の 1 つの列またはビューの更新可能な列からのみ派生したものである。 すなわち、結果の列は、演算子、スカラー関数、定数、またはそれ自体がそのような式から取り出された列を含む式から取り出される列であってはなりません。

削除可能でないカーソルは*読み取り専用* です。

ORDER BY と FOR UPDATE OF を指定する場合、FOR UPDATE OF 文節中の列は、ORDER BY 文節に指定するどの列とも重複してはなりません。

FOR UPDATE OF 文節を指定しない場合は、副選択の SELECT 文節の中の列のうち、更新できるものだけが変更できます。

カーソルの有効範囲 : *カーソル名* の有効範囲は、そのカーソル名が定義されているソース・プログラム、すなわち、プリコンパイラーに渡されるプログラムです。したがって、カーソルを参照できるステートメントは、そのカーソル宣言と一緒にプリコンパイルされたステートメントだけです。例えば、別個にコンパイルされた他のプログラムから呼び出されるプログラムでは、その呼び出しプログラムでオープンされているカーソルを使用することはできません。

また、カーソル名の有効範囲は、カーソルが収められているプログラムの実行場所であるスレッドに限定されます。例えば、同一ジョブ内の 2 つの別個のスレッドで同じプログラムが実行しているとした場合、2 番目のスレッドは、最初のスレッドでオープンされたカーソルを使用することはできません。

CRTSQLxxx コマンドに CLOSQLCSR(*ENDJOB)、CLOSQLCSR(*ENDSQL)、または CLOSQLCSR(*ENDACTGRP) が指定されている場合を除いて、カーソルを参照できるのは、プログラム・スタック内の該当するプログラムの同じインスタンスに限られます。

- CLOSQLCSR(*ENDJOB) が指定されている場合は、プログラム・スタックにある該当するプログラムのどのインスタンスでもカーソルを参照することができます。

- CLOSQLCSR(*ENDSQL) が指定されている場合は、プログラム・スタック上の最後の SQL プログラムが終了するまでは、そのプログラム・スタックにある該当するプログラムのどのインスタンスでもカーソルを参照することができます。
- CLOSQLCSR(*ENDACTGRP) が指定されている場合は、活動化グループが終了するまでは、その活動化グループ内のモジュールのすべてのインスタンスでカーソルを参照することができます。

カーソルの有効範囲は、そのカーソルが宣言されているプログラムですが、そのプログラムから作成された各パッケージは、そのカーソルの別個のインスタンスを含み、実行時に複数のカーソルが存在することがあります。例えば、CONNECT (タイプ 2) ステートメントを使用して、次の順序でロケーション X とロケーション Y に接続するプログラムを想定します。

```
EXEC SQL DECLARE C CURSOR FOR...
EXEC SQL CONNECT TO X;
EXEC SQL OPEN C;
EXEC SQL FETCH C INTO...
EXEC SQL CONNECT TO Y;
EXEC SQL OPEN C;
EXEC SQL FETCH C INTO...
```

2 番目の OPEN C ステートメントは、カーソル C の別個のインスタンスを参照しているので、エラーにはなりません。

SELECT ステートメントは、カーソルがオープンされた時点で評価されます。同一のカーソルをいったんオープンし、クローズした後で、再びオープンした場合には、結果が異なる可能性があります。同一の SELECT ステートメントを使用する、複数のカーソルを同時にオープンすることができます。この場合、それぞれのカーソルの活動は、独立したものとして扱われます。

データのブロック化 : データの処理効率を高めるために、データベース・マネージャーは読み取り専用カーソル用のデータをブロック化することができます。カーソルを位置指定 UPDATE または DELETE ステートメントの中で使用することを予定していない場合は、カーソルを FOR READ ONLY として宣言してください。

カーソルの感応性 : DYNAMIC SCROLL カーソルの場合は、ALWCPYDTA プリコンパイル・オプションは無視されます。挿入、更新、および削除に対する感応性を維持しなければならない場合には、照会の実行に一時的結果が必要でない限り、データの一時コピーは作成されません。

REXX カーソル : REXX プロシージャー内の DECLARE CURSOR ステートメントでホスト変数を使用する場合は、その DECLARE CURSOR は PREPARE および EXECUTE の対象でなければなりません。

例

例 1

表 DEPARTMENT からデータを検索するための照会のカーソルとして、C1 を宣言します。 DECLARE CURSOR ステートメントにこの照会自体が含まれています。

```
EXEC SQL  DECLARE C1 CURSOR FOR
          SELECT DEPTNO, DEPTNAME, MGRNO
            FROM DEPARTMENT
            WHERE WHERE ADMRDEPT = 'A00';
```

例 2

STMT2 という名前のステートメント用のカーソルとして、C2 を宣言します。

```
EXEC SQL  DECLARE C2 CURSOR FOR STMT2;
```

例 3

表 EMPLOYEE の位置指定更新に使用する照会用のカーソルとして、C3 を宣言します。更新が完了するたびに、カーソルをクローズせずに更新がコミットされるようにします。

```
EXEC SQL  DECLARE C3 CURSOR WITH HOLD FOR
            SELECT *
              FROM EMPLOYEE
              FOR UPDATE OF WORKDEPT, PHONENO, JOB, EDLEVEL, SALARY;
```

更新する列を明示的に指定する代わりに、列を指定せずに FOR UPDATE 文節を使用することもできます。この方法で、表の更新可能な列をすべて更新することができます。このカーソルは更新可能なので、このカーソルを使用して表から行を削除することもできます。

例 4

PL/I プログラムにおいて、カーソル C1 を使用して、指定したプロジェクト (PROJNO) に関する値を、表 EMPPROJACT の最初の 4 列から一度に 1 つずつ取り出して、それらの値を EMP (CHARACTER(6))、PRJ (CHARACTER(6))、ACT (SMALLINT)、および TIM (DECIMAL(5,2)) というホスト変数に入れています。 検索するプロジェクトの値は、ホスト変数 SEARCH_PRJ (CHARACTER(6)) から入手します。

```
EXEC SQL  BEGIN DECLARE SECTION;
  DCL  EMP              CHAR(6);
  DCL  PRJ              CHAR(6);
  DCL  SEARCH_PRJ       CHAR(6);
  DCL  ACT              BINARY     FIXED(15);
  DCL  TIM              DEC        FIXED(5,2);
  DCL  SELECT_STMT      CHAR(200)  VARYING;
EXEC SQL  END DECLARE SECTION;


SELECT_STMT = 'SELECT EMPNO, PROJNO, ACTNO, EMPTIME ' ||
                'FROM EMPPROJACT ' ||
                'WHERE PROJNO = ?';
.
.
.
EXEC SQL  PREPARE SELECT_PRJ FROM :SELECT_STMT;

EXEC SQL  DECLARE C1 CURSOR FOR SELECT_PRJ;

EXEC SQL  OPEN C1 USING :SEARCH_PRJ;

EXEC SQL  FETCH C1 INTO :EMP, :PRJ, :ACT, :TIM;

IF SQLSTATE = '02000' THEN
  CALL DATA_NOT_FOUND;
ELSE
  DO WHILE (SUBSTR(SQLSTATE,1,2) = '00'
  | SUBSTR(SQLSTATE,1,2) = '01');
    EXEC SQL  FETCH C1 INTO :EMP, :PRJ, :ACT, :TIM;
  END;
```

```
EXEC SQL  CLOSE C1;
.
.
.
```

例 6

DECLARE CURSOR ステートメントによって、カーソル名 C1 を SELECT の結果に関連付けます。C1 は、更新可能、スクロール可能なカーソルです。

```
EXEC SQL DECLARE C1 DYNAMIC SCROLL CURSOR FOR
    SELECT DEPTNO, DEPTNAME, MGRNO
    FROM CORPDATA.TDEPT
    WHERE ADMRDEPT = 'A00';
```

例 7

4 つの列から値を取り出し、逐次化可能 (RR) 分離レベルを使用して、それらの値をホスト変数に割り当てるために、カーソルを宣言します。

```
DECLARE CURSOR1 CURSOR FOR
  SELECT COL1, COL2, COL3, COL4
  FROM TBLNAME WHERE COL1 = :varname
  WITH RR
```

DECLARE GLOBAL TEMPORARY TABLE

DECLARE GLOBAL TEMPORARY TABLE ステートメントは、現行アプリケーション・プロセス用の宣言済み一時表を定義します。宣言済み一時表記述は、システム・カタログには含まれません。この記述は持続性のあるものではなく、複数のセッション間で共用することはできません。複数のセッションで同名の宣言済みグローバル一時表が定義されていても、その一時表の記述はそれぞれのセッションごとに固有のものです。アプリケーション・プロセスが終了すると、一時表は除去されます。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。このステートメントは、動的に準備できる実行可能ステートメントです。

権限

LIKE または AS 選択ステートメント文節を指定する場合は、このステートメントの権限 ID が保持する特権には、その LIKE 文節または AS 副照会文節で指定するすべての表またはビューに対して、少なくとも次の 1 つが含まれていなければなりません。

- 表またはビューに対する SELECT 特権
- 表またはビューの所有権
- 管理権限

特殊タイプを参照する場合は、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

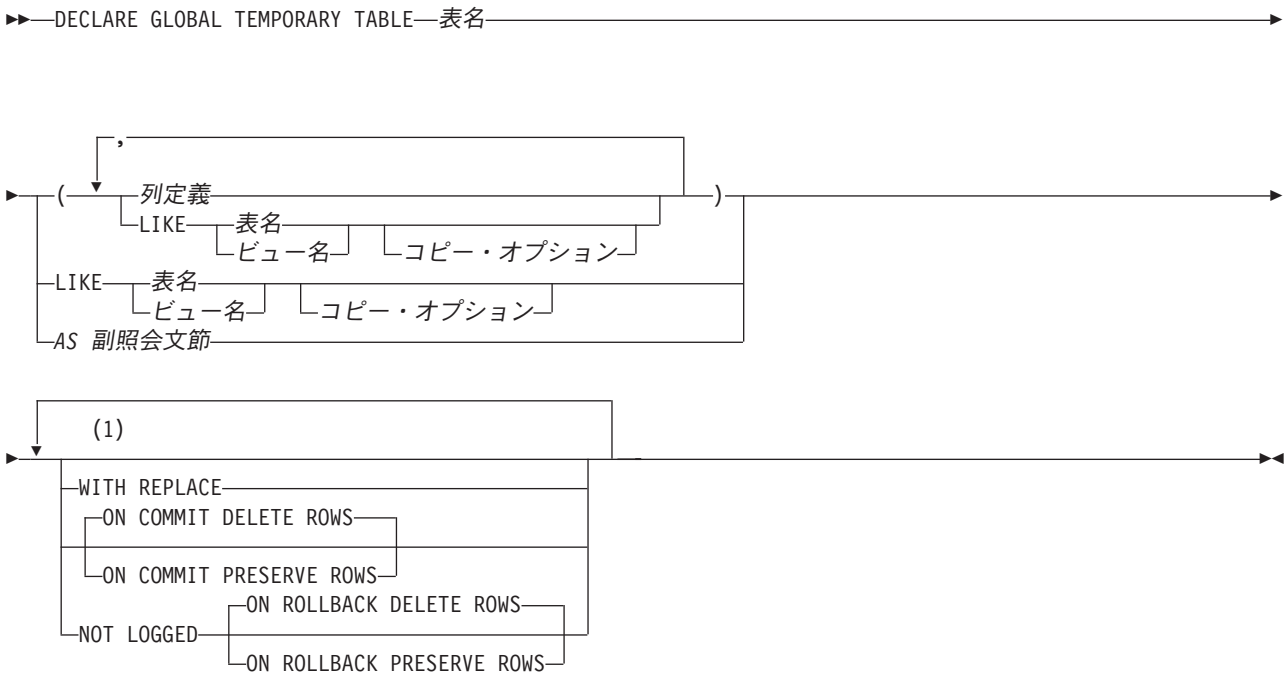
- ステートメント内に識別されているそれぞれの特殊タイプに対しては次のもの。
  - その 特殊タイプ に対する USAGE 特権。および
  - その 特殊タイプ が入っているライブラリーに対する *EXECUTE システム権限
- 管理権限

次のいずれかに該当する場合、ステートメントの権限 ID には、特殊タイプ に対する USAGE 特権が与えられます。

- その 特殊タイプ の所有者である。
- その 特殊タイプ に対する USAGE 特権が付与されている。
- その 特殊タイプ に対するシステム権限の *OBJOPR と *EXECUTE が付与されている。

DECLARE GLOBAL TEMPORARY TABLE

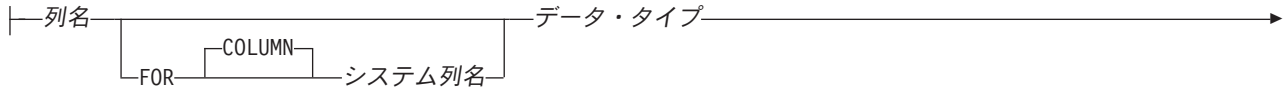
構文

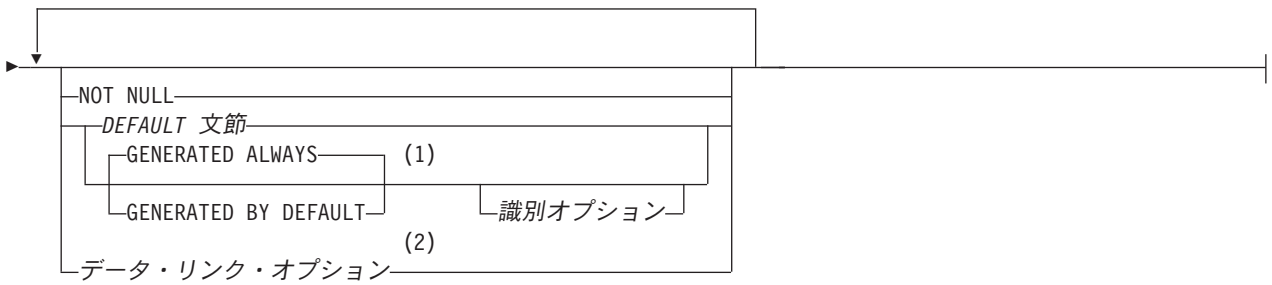


注:

1 各文節はそれぞれ 1 回のみ指定できます。

列定義:



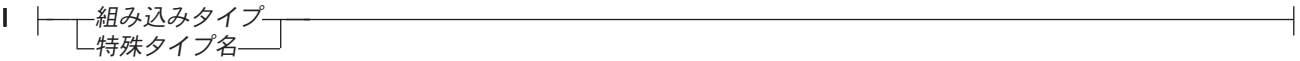


注:

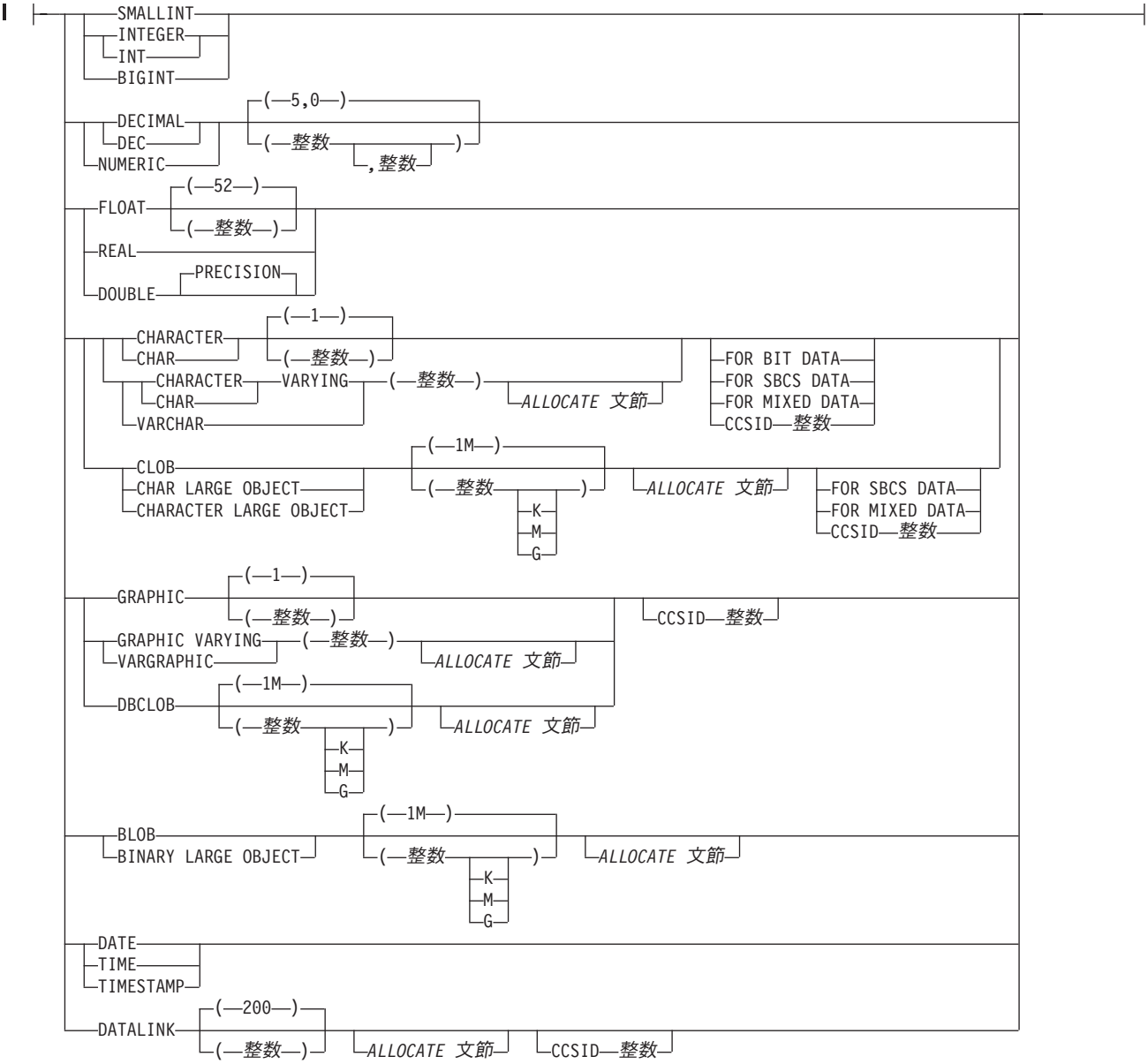
1 GENERATED を指定できるのは、対象の列が識別列である場合だけです。

2 データ・リンク・オプションは、DATALINK、および DATALINK をソ ースとする特殊タイプに対してのみ指定することができます。

データ・タイプ:

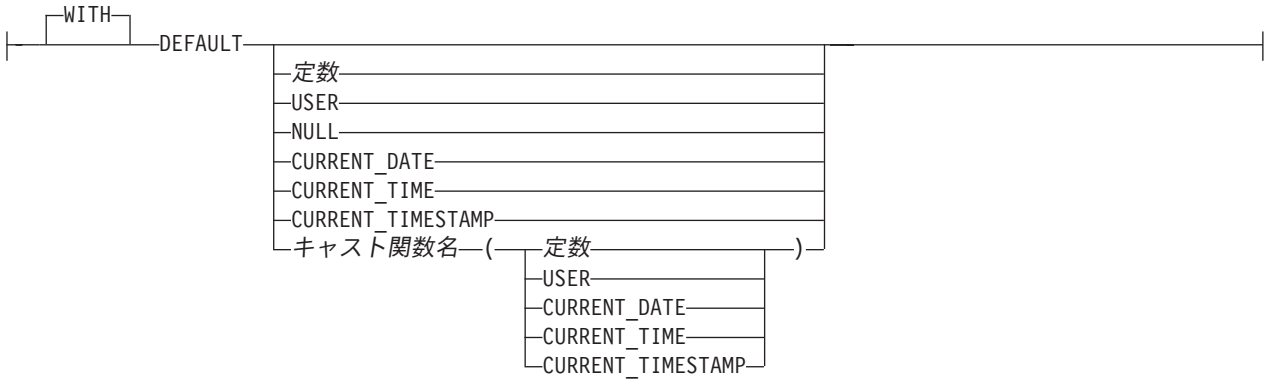


組み込みタイプ:

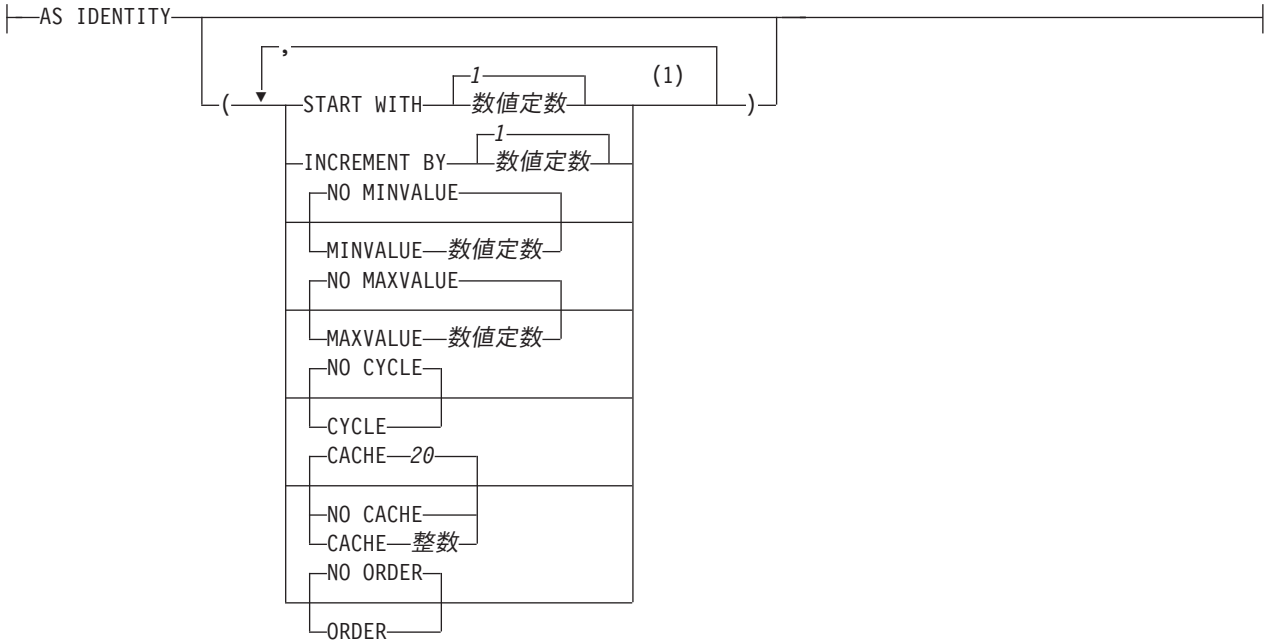


ALLOCATE 文節:

DEFAULT 文節:



識別オプション:

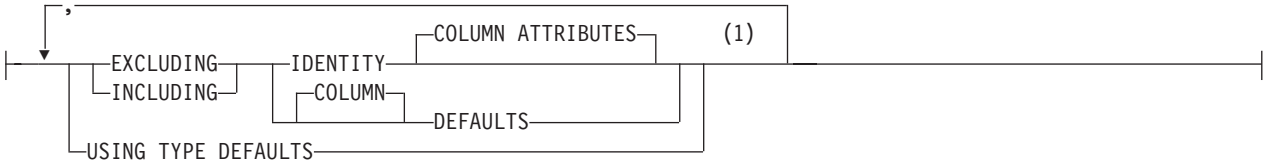


注:

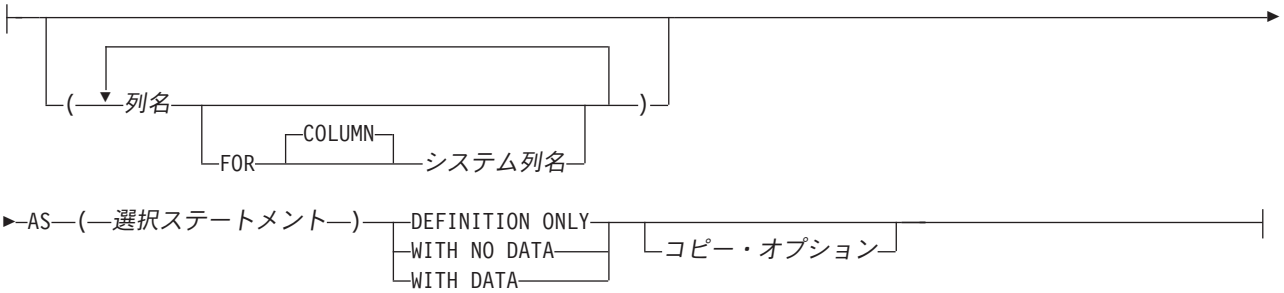
1 各文節はそれぞれ 1 回のみ指定できます。

データ・リンク・オプション:

コピー・オプション:



AS 副照会文節:



注:

1 各文節はそれぞれ 1 回のみ指定できます。

説明

表名

一時表の名前を指定します。 修飾子を明示指定する場合は、SESSION でなければなりません。指定しなかった場合は、修飾子は暗黙的に SESSION に設定されます。 SESSION という名前の永続ライブラリーの中に同名の表、ビュー、索引、または別名がすでに存在している場合は、以下のアクションがとられます。

- 宣言した一時表は、SESSION.table-name の名前で定義されます。宣言済み一時表の解決には永続ライブラリーは含まれないので、エラーは起こりません。
- SESSION.table-name に対する参照は、SESSION.table-name の名前を持つ永続的な表、ビュー、索引、または別名ではなく、この宣言済み一時表に解決されます。

この表は QTEMP ライブラリー内に作成されます。

列定義

列の属性を定義します。少なくとも 1 つ以上で、8000 を超えない列の定義がなければなりません。

列の行バッファー・バイト・カウントの合計は、32766 (VARCHAR 列または VARGRAPHIC 列が指定されている場合は 32740) 以下でなければなりません。さらに、LOB を指定してある場合は、すべての列の行データ・バイト・カウントの合計が 3.5 GB を超えていてはなりません。 データ・タイプごとの列のバイト・カウントについては、569 ページの『使用上の注意』を参照してください。

列名
表を構成する列の名前を指定します。*列名* は修飾できません。表の複数の列や表のシステム列名に同じ名前を使用することもできません。

FOR COLUMN *システム列名*
列の OS/400 名を指定します。表の複数の列やシステム列名に対して、同じ名前を使用してはなりません。

システム列名が指定されず、また列名が有効なシステム列名でない場合には、システム列名が生成されます。システム列名の生成方法に関する詳細については、572 ページの 『列名の生成の規則』を参照してください。

データ・タイプ
列のデータ・タイプを指定します。

組み込みタイプ
組み込みデータ・タイプを指定します。 *組み込みタイプ* の説明については、542 ページの『CREATE TABLE』を参照してください。

グローバル一時表については、ROWID 列、または FILE LINK CONTROL を伴う DATALINK 列は指定できません。

NOT NULL
列にヌル値が入るのを防止します。NOT NULL を指定しないことは、その列がヌルであってもよいことを意味します。

DEFAULT
列のデフォルト値を指定します。この文節は、1 つの*列定義* の中で複数回指定することはできません。識別列 (AS IDENTITY と定義されている列) については、DEFAULT を指定することはできません。識別列のデフォルト値は、データベース・マネージャーが生成します。 DEFAULT キーワードの後に値が指定されていない場合は、次のようになります。

- 列がヌル可能の場合、デフォルト値はヌル値になります。
- 列がヌル可能でない場合、デフォルト値は列のデータ・タイプによって決まります。

データ・タイプ	デフォルト値
数値	0
固定長ストリング	ブランク
可変長ストリング	0 のストリング長
日付	INSERT の時点の当日の日付
時刻	INSERT の時点の当日の時刻
タイム・スタンプ	INSERT の時点の当日のタイム・スタンプ
データ・リンク	DLVALUE(’’,’URL’,’’) に対応する値
特殊タイプ	特殊タイプの対応するソース・タイプのデフォルト値

NOT NULL および DEFAULT を*列の定義* から省いた場合、DEFAULT NULL の暗黙の指定が取られます。

定数
その列のデフォルト値としての定数を指定します。これは、85 ページの『割り当ておよび比較』で説明している割り当て規則に従って、その列に割り当てることができる値を表す定数にする必要があります。浮動小数点定数は、SMALLINT、INTEGER、DECIMAL、または NUMERIC 列に使用してはなりません。10 進定数には、小数点より右方に、その列に指定された位取りより多くの桁を含めてはなりません。

USER
INSERT または UPDATE の時点での USER 特殊レジスターの値を、その列のデフォルト値として指定します。この列のデータ・タイプは、USER 特殊レジスターの長さ属性と同じかそれより大きい長さ属性を持つ CHAR または VARCHAR でなければなりません。

NULL
その列のデフォルト値としてヌルを指定します。NOT NULL を指定する場合は、同じ列の定義内で DEFAULT NULL を指定してはなりません。

CURRENT_DATE
現在の日付を列のデフォルト値として指定します。CURRENT_DATE を指定する場合は、列のデータ・タイプは DATE または DATE に基づく特殊タイプでなければなりません。

CURRENT_TIME
現在の時刻を列のデフォルト値として指定します。CURRENT_TIME を指定する場合は、列のデータ・タイプは TIME または TIME に基づく特殊タイプでなければなりません。

CURRENT_TIMESTAMP
現在のタイム・スタンプを列のデフォルト値として指定します。CURRENT_TIMESTAMP を指定する場合は、列のデータ・タイプは TIMESTAMP または TIMESTAMP に基づく特殊タイプでなければなりません。

キャスト関数名
この形式のデフォルト値は、特殊タイプやデータ・タイプ BLOB、CLOB、 DBCLOB、 DATE、 TIME、または TIMESTAMP として定義された列でのみ使用することができます。次の表は、これらのキャスト関数 の許可されている使用法を示します。

データ・タイプ	キャスト関数名
BLOB、CLOB、または DBCLOB に基づく特殊タイプ N	BLOB、CLOB、または DBCLOB *
DATE、TIME、または TIMESTAMP に基づく特殊タイプ N	N (N の作成時に生成されたユーザー定義のキャスト関数) ** あるいは DATE、TIME、または TIMESTAMP *
他のデータ・タイプに基づく特殊タイプ	N (N の作成時に生成されたユーザー定義のキャスト関数) **
BLOB、CLOB、または DBCLOB	BLOB、CLOB、または DBCLOB *
DATE、TIME、または TIMESTAMP	DATE、TIME、または TIMESTAMP *

注:

* 関数には、QSYS2 の暗黙的または明示的スキーマ名のデータ・タイプ (または、特殊タイプのソース・タイプ) の名前と一致する名前を指定する必要があります。

** 関数には、列の特殊タイプの名前と一致する名前を指定する必要があります。スキーマ名で修飾する場合は、特殊タイプのスキーマ名と同じ名前を指定する必要があります。修飾しない場合は、関数の解析から得られるスキーマ名は、特殊タイプのスキーマ名と同じ名前にする必要があります。

定数
定数を引き数として指定します。この定数は、特殊タイプのソース・タイプの定数、あるいは、特殊タイプでない場合は、データ・タイプの定数の規則に準拠する必要があります。BLOB、 CLOB、 DBCLOB、 DATE、 TIME、および TIMESTAMP 関数の場合は、この定数をストリング定数にする必要があります。

USER
INSERT または UPDATE の時点での USER 特殊レジスターの値をその列のデフォルト値として指定します。この列の特殊タイプのソース・タイプのデータ・タイプは、USER 特殊レジスターの長さ属性と同じかそれより大きい長さ属性を持つ CHAR または VARCHAR でなければなりません。

CURRENT_DATE
現在の日付を列のデフォルト値として指定します。CURRENT_DATE を指定する場合、列の特殊タイプのソース・タイプのデータ・タイプは、DATE にする必要があります。

CURRENT_TIME
現在の時刻を列のデフォルト値として指定します。CURRENT_TIME を指定する場合、列の特殊タイプのソース・タイプのデータ・タイプは、TIME にする必要があります。

CURRENT_TIMESTAMP
現在のタイム・スタンプを列のデフォルト値として指定します。CURRENT_TIMESTAMP を指定する場合、列の特殊タイプのソース・タイプのデータ・タイプは、TIMESTAMP にする必要があります。

GENERATED
列の値をデータベース・マネージャーが生成することを指定します。列が識別列 (AS IDENTITY 文節で定義されたもの) と見なされる場合は、GENERATED を指定する必要があります。

ALWAYS
表に行が 1 つ挿入されるたびに、常にデータベース・マネージャーが列の値を生成することを指定します。 ALWAYS は推奨値です。

BY DEFAULT
行が挿入されたときに、列の値が指定されていない場合のみ、データベース・マネージャーが列の値を生成することを指定します。値が指定されている場合は、データベース・マネージャーはその値を使用します。

識別列の場合は、データベース・マネージャーは指定された値を挿入しますが、その識別列が固有制約を持っているか、またはその識別列を単独で指定する固有索引を持っている場合を除き、その値がその列の固有の値であるかどうかの検査は行いません。

AS IDENTITY
列が表の識別列であることを指定します。 1 つの表は識別列を 1 つだけ持つことができます。 AS IDENTITY を指定できるのは、列のデータ・タイプが、厳密に位取りがゼロの数値タイプ (SMALLINT、 INTEGER、 BIGINT、 DECIMAL、または位取りがゼロの NUMERIC、またはこれらのタイプに基づく特殊タイプ) である場合だけです。

識別列は、暗黙的に NOT NULL になります。

START WITH *数値定数*
識別列について生成される最初の値を指定します。小数点の右側にゼロ以外の数字がないことを条件として、この列に割り当てることのできる任意の正または負の値を指定できます。

識別列を定義するときに値を明示的に指定していない場合のデフォルト値は、昇順の場合は MINVALUE で、降順の場合は MAXVALUE です。この値は、シーケンスが最大値または最小値に達した後で、シーケンスの循環により到達する値になるとは限りません。 START WITH 文節を使用することにより、この循環に使用される値の範囲外の値からシーケンスを開始することができます。循環に使用する範囲は、MINVALUE および MAXVALUE で定義します。

INCREMENT BY *数値定数*
識別列の連続した値の間隔を指定します。この値には、0 でなく、長整数定数の値を超過せず、かつ小数点の右側にゼロ以外の数字がないことを条件として、この列に割り当てることのできる任意の正または負の値を指定できます。デフォルト値は 1 です。

この値が正である場合は、識別列の値のシーケンスは昇順になります。この値が負の場合は、値のシーケンスは降順になります。

MAXVALUE *数値定数*
この識別列用として生成される最大値を示す数値定数を指定します。この値には、この列に割り当てることのできる任意の正または負の値を指定できますが、最小値より大きい値でなければなりません。

識別列を定義するときにこの値を明示的に指定していない場合は、この値は、昇順シーケンスの場合は該当データ・タイプ (および、DECIMAL の場合は精度) の最大値になります。降順シーケンスの場合は、この値は START WITH の値ですが、START WITH を指定していなければ -1 です。

MINVALUE *数値定数*
この識別列用として生成される最小値を示す数値定数を指定します。この値には、この列に割り当てることのできる任意の正または負の値を指定できますが、最大値より小さい値でなければなりません。

識別列を定義するときにこの値を明示的に指定していない場合は、この値は、昇順シーケンスの場合は START WITH の値ですが、START WITH を指定していなければ 1 です。降順シーケンスの場合は、この値は、該当データ・タイプ (および、DECIMAL の場合は精度) の最小値です。

CACHE または NO CACHE
事前割り振りの値をメモリー内に保持するかどうかを指定します。値を事前に割り振ってキャッシュに保管しておくと、表に行を挿入するときのパフォーマンスが向上します。

CACHE *整数*
データベース・マネージャーが事前割り振りしてメモリー内に保持する、識別列シーケンスの値の数を指定します。指定できる最小の値は 2 で、最大の値は、1 つの整数で表せる最大の値です。デフォルト値は 20 です。

システム障害が発生すると、キャッシュに保管されていてまだ割り当てられていない識別列の値はすべて失われ、使用不能になります。したがって、CACHE に指定する値は、システム障害が発生したときに失われる可能性のある識別列の値の最大数でもあります。

NO CACHE
識別列の値を事前割り振りしないことを指定します。

CYCLE または NO CYCLE
シーケンスの最大値または最小値に達した後も、この識別列について値を生成し続けるかどうかを指定します。

CYCLE
最大値または最小値に達した後も、この列の値を生成し続けることを指定します。このオプションを使用した場合は、昇順シーケンスがシーケンスの最大値に達した後では、最小値が生成されます。降順シーケンスがシーケンスの最小値に達した後は、最大値が生成されます。列の最大値と最小値によって、循環に使用される範囲が決まります。

CYCLE が効力を持っている場合は、データベース・マネージャーは 1 つの識別列について重複する値を生成することがあります。対象の識別列について固有制約または固有索引が存在する場合は、固有でない値が生成されるとエラーが起こります。

NO CYCLE
シーケンスの最大値または最小値に達した後は、この識別列について値を生成しないことを指定します。これは、デフォルト値です。

ORDER または NO ORDER
識別値を要求された順序で生成するかどうかを指定します。

ORDER
識別値を要求された順序で生成することを指定します。

NO ORDER
値を要求された順序で生成する必要がないことを指定します。これは、デフォルト値です。

データ・リンク・オプション
DATALINK データ・タイプに関連したオプションを指定します。

LINKTYPE URL
リンクのタイプを URL として定義します。

NO LINK CONTROL
これを指定すると、リンク済みファイルが存在するか否かを判別するための検査は行われなくなります。URL の構文だけが検査されます。リンク済みファイルに関してデータベース・マネージャー制御は行われません。

LIKE

*表名*または*ビュー名*
指定の表またはビューに定義されている列をこの表に含めることを指定します。LIKE 文節で指定する*表名* や*ビュー名* は、すでにサーバー上に存在する表またはビューを識別していなければなりません。

LIKE の使用は、n 列 (n は、識別された表またはビュー内の列数) を暗黙的に定義したことになります。この暗黙の定義には、n 列の以下の属性が含まれます (そのデータ・タイプに該当する場合)

- 列名 (および、システム列名)
- データ・タイプ、長さ、精度、および位取り
- CCSID

表名 の直後に LIKE 文節を指定し、括弧で囲まなかった場合は、以下の列属性も含まれます。その他の場合は、これらの属性は含まれません (デフォルト値および識別属性は、*コピー・オプション* を使用して制御することもできます)。

- デフォルト値 (*表名* が指定され、*ビュー名* は指定されていない場合)
- 識別属性
- ヌル可能性
- 列の見出しとテキスト (714 ページの『LABEL』を参照)

指定された表またはビューが非 SQL 作成の物理ファイルまたは論理ファイルの場合、非 SQL 属性は除去されます。例えば、日付と時刻の形式は ISO 形式に変更されます。

暗黙の定義には、識別された表またはビューのその他のオプション属性は含まれません。例えば、新規の表には、表からの基本キーや外部キーは自動的には組み込まれません。新規の表にこうしたオプション属性が組み込まれるのは、オプション文節を明示的に指定した場合に限られます。

AS 副照会文節

列名

表を構成する列の名前を指定します。*列名* は修飾できません。表の複数の列や表のシステム列名に同じ名前を使用することもできません。

FOR COLUMN *システム列名*

列の OS/400 名を指定します。表の複数の列やシステム列名に対して、同じ名前を使用してはなりません。

システム列名が指定されず、また列名が有効なシステム列名でない場合には、システム列名が生成されます。システム列名の生成方法に関する詳細については、572 ページの 『列名の生成の規則』を参照してください。

選択ステートメント

表の列の名前および記述が、選択ステートメントを実行した場合に選択ステートメントの派生結果表に現れる列と同じになるようにすることを指定します。 AS 選択ステートメントを使用すると、この表について n 個の列を暗黙的に定義したことになります。n は、選択ステートメントの結果として発生する列の数です。この暗黙の定義には、n 列の以下の属性が含まれます (そのデータ・タイプに該当する場合)

- 列名 (および、システム列名)
- データ・タイプ、長さ、精度、および位取り
- CCSID
- ヌル可能性
- 列の見出しとテキスト (714 ページの『LABEL』を参照)

以下の属性は組み込まれません (デフォルト値と識別属性は、*コピー・オプション* を使用して組み込むことができます)。

- デフォルト値
- 識別属性

暗黙の定義には、識別された表またはビューのその他のオプション属性は含まれません。例えば、新規の表には、表からの基本キーや外部キーは自動的には組み込まれません。新規の表にこうしたオプション属性が組み込まれるのは、オプション文節を明示的に指定した場合に限られます。

暗黙的に定義された列は、*選択ステートメント* の結果表の列の名前を継承します。したがって、すべての結果列について、*選択ステートメント* または列名リストの中で列名を指定する必要があります。式、定数、および関数から派生する結果列については、*選択ステートメント* で結果列の直後に AS 列名文節を指定するか、*選択ステートメント* の前の列リスト内に名前を指定する必要があります。

選択ステートメント は、ホスト変数または組み込みパラメーター・マーカー (疑問符) を参照するものであってはなりません。

WITH DATA

選択ステートメント を実行することを指定します。表の作成後に、*選択ステートメント* の結果表の行が自動的に表に挿入されます。

WITH NO DATA または DEFINITION ONLY
選択ステートメント を実行しないことを指定します。したがって、自動的に表に挿入される行のセットを持つ結果表はありません。

コピー・オプション

INCLUDING IDENTITY COLUMN ATTRIBUTES
この表が、*選択ステートメント*、*表名*、または *ビュー名* の結果として生じる列の識別属性 (もしあれば) を継承することを指定します。一般に、識別属性がコピーされるのは、表、ビュー、または選択ステートメントの中の対応する列のエレメントが、識別属性を持つ基礎表列の名前に直接または間接にマップされる表列またはビュー列の名前である場合です。

INCLUDING IDENTITY COLUMN ATTRIBUTES 文節と AS *選択ステートメント* 文節を指定してあるときは、以下の場合には新規の表の列は識別属性を継承しません。

- *選択ステートメント* の選択リストに、識別列名の複数のインスタンスが含まれている (つまり同じ列を複数回選択している) 場合。
- *選択ステートメント* の選択リストに、複数の識別列が含まれている (つまり結合が含まれている) 場合。
- 選択リスト内の式のいずれかに識別列が含まれている場合。
- *選択ステートメント* に一組の演算 (共用体) が含まれている場合。

INCLUDING IDENTITY を指定しなかった場合は、表には識別列は含まれません。

EXCLUDING IDENTITY COLUMN ATTRIBUTES
この表が、*選択ステートメント*、*表*、または*ビュー名* の結果として生じる列の識別属性を継承しないことを指定します。

INCLUDING COLUMN DEFAULTS
この表が、*選択ステートメント*、*表名*、または*ビュー名* から生じる列のデフォルト値を継承することを指定します。デフォルト値は、INSERT で値が指定されていない場合に、列に割り当てられる値です。

USING TYPE DEFAULTS を指定する場合は、INCLUDING COLUMN DEFAULTS を指定しないでください。

INCLUDING COLUMN DEFAULTS を指定しなかった場合は、表はデフォルト値を継承しません。

EXCLUDING COLUMN DEFAULTS
この表が、*選択ステートメント*、*表名*、または*ビュー名* から生じる列のデフォルト値を継承しないことを指定します。

USING TYPE DEFAULTS
この表のデフォルト値が、*選択ステートメント*、*表名*、または*ビュー名* から生じる列のデータ・タイプに応じて決まることを指定します。その列がヌル可能である場合は、デフォルト値はヌル値です。その他の場合は、デフォルト値は以下のようになります。

データ・タイプ	デフォルト値
数値	0
固定長ストリング	ブランク
可変長ストリング	0 のストリング長
日付	INSERT の時点の当日の日付
時刻	INSERT の時点の当日の時刻
タイム・スタンプ	INSERT の時点の当日のタイム・スタンプ
データ・リンク	DLVALUE(’’,’URL’,’’) に対応する値
特殊タイプ	特殊タイプの対応するソース・タイプのデフォルト値

INCLUDING COLUMN DEFAULTS を指定する場合は、USING TYPE DEFAULTS は指定しないでください。

WITH REPLACE
指定した名前の宣言済みグローバル一時表がすでに存在している場合に、その既存の表を、このステートメントで定義した一時表で置き換える (そして既存の表のすべての行を削除する) ことを指定します。

WITH REPLACE を指定しない場合は、現行セッション内にすでに存在しているどの宣言済みグローバル一時表とも異なる名前を指定する必要があります。

ON COMMIT
COMMIT 操作が行われるときにこのグローバル一時表に対して行うアクションを指定します。

この宣言済みグローバル一時表が No Commit (NC) 分離レベルでオープンされた場合、または COMMIT HOLD 操作が行われる場合は、ON COMMIT 文節は適用されません。

DELETE ROWS
表について WITH HOLD カーソルがオープンされていない場合は、表のすべての行が削除されます。これは、デフォルト値です。

PRESERVE ROWS
表の行は保存されます。

NOT LOGGED
表に対する変更 (表の作成も含む) をログに記録しないことを指定します。ROLLBACK (または ROLLBACK TO SAVEPOINT) 操作を行うときに、この作業単位 (または保管ポイント) で表が変更されていても、その変更はロールバックされません。この作業単位 (または保管ポイント) の中で表を作成した場合、その表は除去されます。この作業単位 (または保管ポイント) の中で表を除去した場合は、その表は復元されますが、行は含まれていません。

ON ROLLBACK
ROLLBACK 操作が行われるときにこのグローバル一時表に対して行うアクションを指定します。

この宣言済みグローバル一時表が No Commit (NC) 分離レベルでオープンされた場合、または ROLLBACK HOLD 操作が行われる場合は、ON ROLLBACK 文節は適用されません。

DELETE ROWS
表のすべての行が削除されます。これは、デフォルト値です。

PRESERVE ROWS
表の行は保存されます。

使用上の注意

- **インスタンス化、有効範囲、および終了 :** P はアプリケーション・プロセスを表し、T は、P の中のアプリケーション・プログラム内の宣言済み一時表であるものとします。
  - P の中のプログラムが DECLARE GLOBAL TEMPORARY TABLE ステートメントを発行すると、T の空のインスタンスが作成されます。
  - P の中のどのプログラムも T を参照でき、それらの参照はどれも T の同じインスタンスに対する参照です。(SQL 関数、SQL プロシージャー、またはトリガーの複合ステートメント内で DECLARE GLOBAL TEMPORARY ステートメントを指定した場合は、宣言済み一時表の有効範囲は、その複合ステートメントではなくアプリケーション・プロセスです。)

    T がリモート・サーバーで宣言されたものである場合は、T に対する参照では、T を宣言するときに使用したものと同じ接続を使用する必要があり、その接続は T の宣言後に終了していてはなりません。 T が宣言されたデータベース・サーバーへの接続が終了すると、T は除去され、T のインスタンス化された行は破棄されます。
  - T の定義に ON COMMIT DELETE ROWS 文節が含まれている場合は、コミット操作により P の中の 1 つの作業単位が終了した時点で、P の中のどのプログラムについても T に依存する WITH HOLD カーソルがオープン状態にないときは、すべての行が削除されます。
  - T の定義に ON ROLLBACK DELETE ROWS 文節が含まれている場合は、ロールバック操作により P の中の 1 つの作業単位が終了した時点で、すべての行が削除されます。
  - T を宣言したアプリケーション・プロセスが終了すると、T は除去され、T の行は破棄されます。
- **一時表の所有権 :** この表の *所有者* は、このステートメントを実行しているジョブのユーザー・プロファイルです。
- **一時表の権限 :** 宣言済み一時表を定義するときに、その表に関するすべての表特権、およびその表を除去する権限が、PUBLIC に対して暗黙に認可されます。
- **他の SQL ステートメント内での宣言済み一時表の参照 :** 多くの SQL ステートメントが宣言済み一時表をサポートしています。 DECLARE GLOBAL TEMPORARY TABLE 以外の SQL ステートメントの中で一時表を参照するには、その表を暗黙的または明示的に SESSION で修飾する必要があります。

  表名の修飾子として SESSION を使用しても、アプリケーション・プロセスに、その表名についての DECLARE GLOBAL TEMPORARY TABLE ステートメントが含まれていない場合は、データベース・マネージャーは、宣言済み一時表が参照されているのではないと判断します。そして、データベース・マネージャーは、このような表参照を永続表に解決します。
- **宣言済み一時表の使用に関する制限 :**

- ALTER TABLE、 COMMENT、 CREATE TRIGGER、 GRANT、 LABEL、 LOCK、 RENAME、または REVOKE ステートメントでは、宣言済み一時表は指定できません。
- 宣言済み一時表は、参照制約の中で親表として指定することはできません。
- CREATE INDEX または CREATE VIEW ステートメントで宣言済み一時表を参照する場合は、該当の索引またはビューは、SESSION (または QTEMP ライブラリー) の中に作成する必要があります。

例

例 1

社員番号、給与、歩合給、および賞与に関する列定義のある宣言済み一時表を定義します。

```
DECLARE GLOBAL TEMPORARY TABLE SESSION.TEMP_EMP
  (EMPNO    CHAR(6)     NOT NULL,
             SALARY     DECIMAL(9, 2),
             BONUS      DECIMAL(9, 2),
             COMM       DECIMAL(9, 2))
  ON COMMIT PRESERVE ROWS
```

例 2

USER1.EMPTAB という基礎表に 3 つの列があり、その 1 つが識別列であるとします。この基礎表を同じ列名および属性 (識別属性も含む) を持つ一時表を宣言します。

```
DECLARE GLOBAL TEMPORARY TABLE TEMPTAB1
  LIKE USER1.EMPTAB
  INCLUDING IDENTITY
  ON COMMIT PRESERVE ROWS
```

上記の例では、データベース・マネージャー は TEMPTAB1 の暗黙修飾子として SESSION を使用します。

DECLARE PROCEDURE

DECLARE PROCEDURE ステートメントは、外部プロシージャーを定義します。

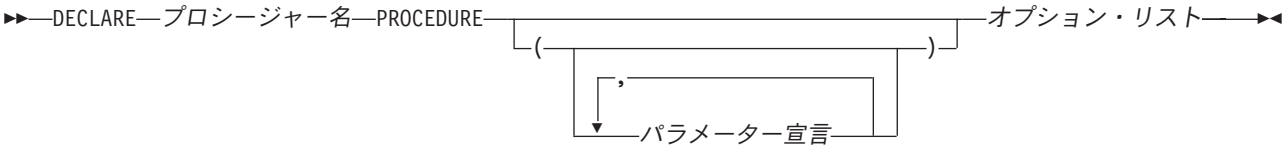
呼び出し

このステートメントは、アプリケーション・プログラムに組み込んで使用します。それ以外の使用法はありません。このステートメントは、実行可能ステートメントではありません。REXX で指定してはなりません。

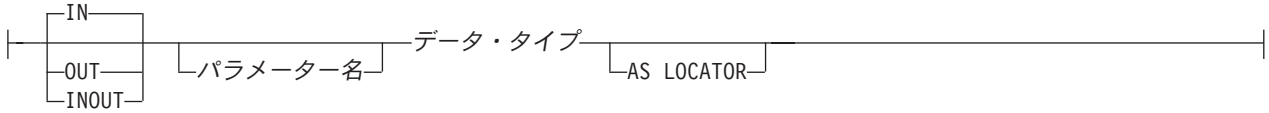
権限

不要です。

構文

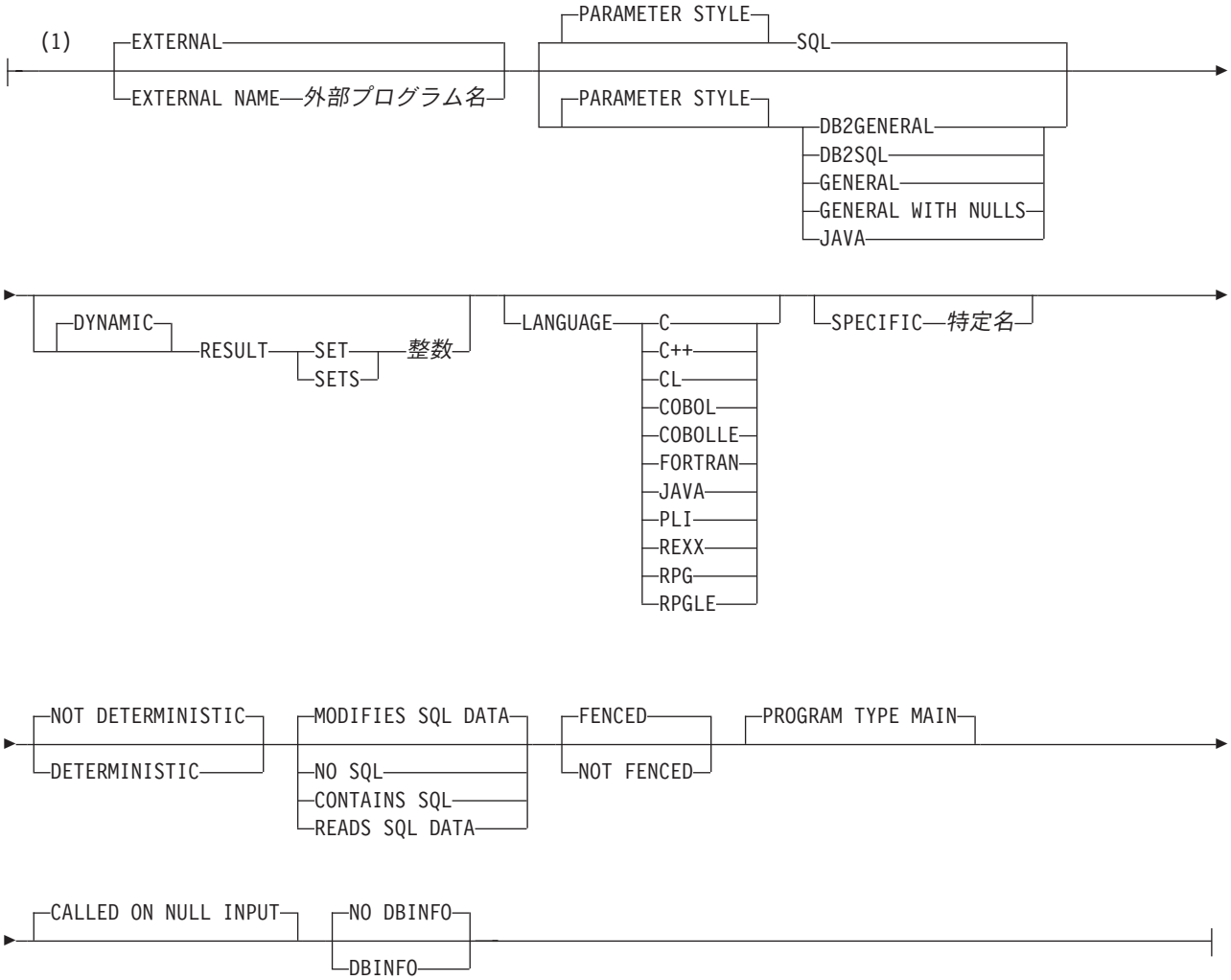


パラメーター宣言:



DECLARE PROCEDURE

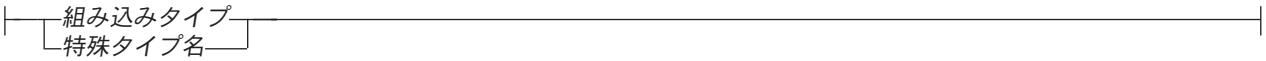
オプション・リスト:



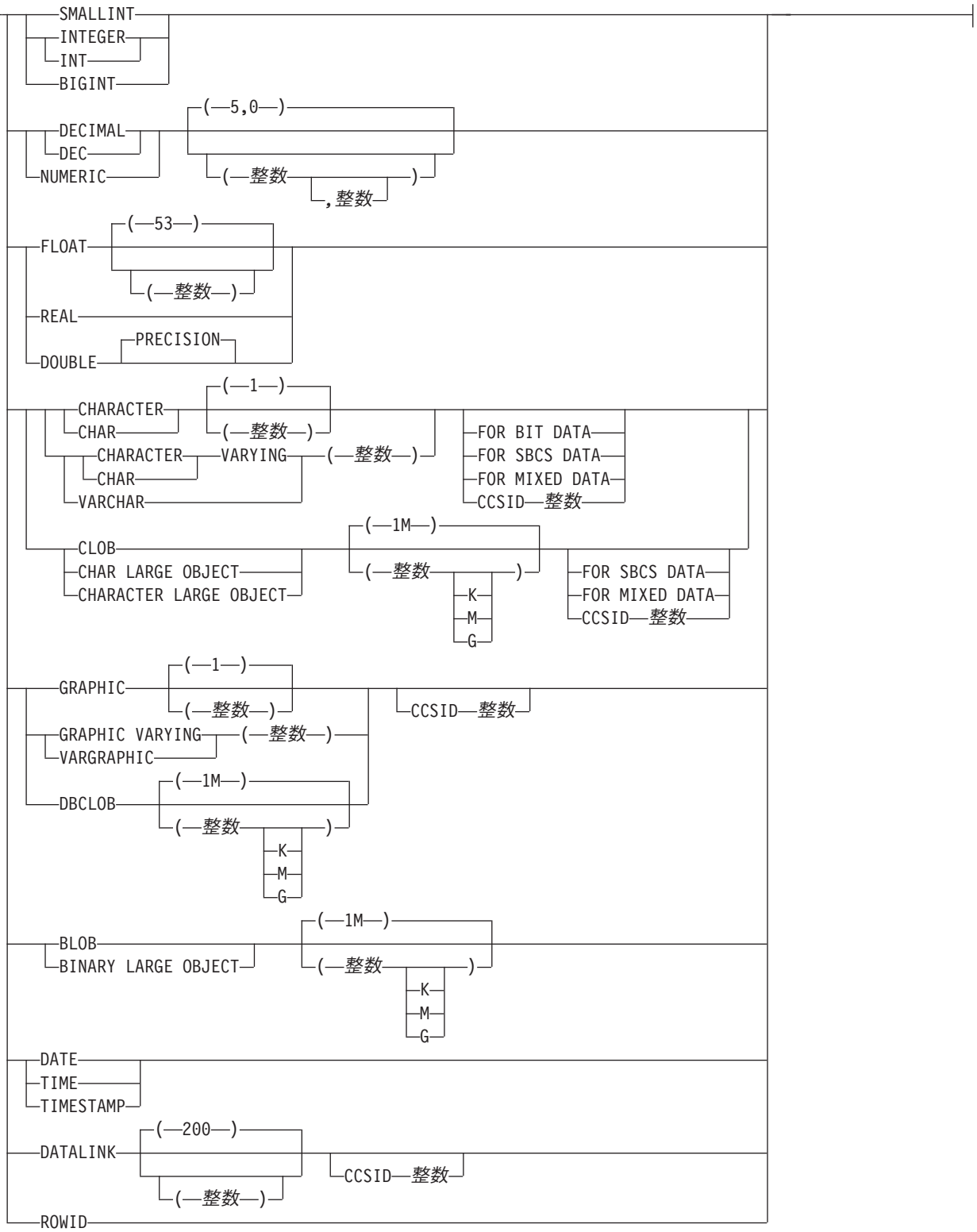
注:

1 オプション文節は、別の順序で指定することができます。

データ・タイプ:



組み込みタイプ:



説明

プロシージャー名
プロシージャーを指定します。この名前は、そのソース・プログラムで宣言されている他のプロシージャーの名前と同じであってはなりません。

(パラメーター宣言,...)
プロシージャーのパラメーターの数とそれぞれのパラメーターのデータ・タイプを指定します。 プロシージャーに関するパラメーターは、入力専用、出力専用、または入出力両用に使用できます。 それぞれのパラメーターに名前を指定することができますが、これは必須ではありません。

SQL プロシージャーに許されるパラメーターの最大数は 255 です。

IN
パラメーターが、プロシージャーへの入力パラメーターであることを指定します。プロシージャー内でパラメーターに対する変更が行われても、制御が戻った後で、呼び出し元の SQL アプリケーションがその変更内容を使用することはできません。[53]

OUT
パラメーターが、プロシージャーから戻される出力パラメーターであることを示します。

データ・リンクやデータ・リンクをベースとした特殊タイプは、出力パラメーターとして指定することはできません。

INOUT
パラメーターが、このプロシージャー用の入出力両方のパラメーターであることを指定します。

データ・リンクやデータ・リンクをベースとした特殊タイプは、入出力パラメーターとして指定することはできません。

パラメーター名
パラメーター名を指定します。この名前は、このプロシージャー用の他のパラメーター名 と同じものであってはなりません。

データ・タイプ
パラメーターのデータ・タイプを指定します。

指定するデータ・タイプは LANGUAGE 文節で指定する言語にとって有効なものでなければなりません。すべてのデータ・タイプが SQL プロシージャーに有効です。データ・リンクは、外部プロシージャーには無効です。データ・タイプの詳細については、542 ページの 『CREATE TABLE』 および SQL プログラミング 概念 を参照してください。

CCSID が指定されている場合、プロシージャーに渡される前に、パラメーターはその CCSID に変換されます。CCSID が指定されていない場合は、CCSID は、プロシージャーの呼び出し時点における現行サーバーのデフォルトの CCSID によって決まります。

AS LOCATOR
これを指定すると、入力パラメーターは、実際の値ではなく、値のロケータ

53. 言語タイプが REXX の場合は、パラメーターはすべて入力パラメーターでなければなりません。

ーになります。 AS LOCATOR は、入力パラメーターに LOB データ・タイプや LOB データ・タイプをベースとする特殊タイプが指定されている場合にのみ、指定することができます。

DYNAMIC RESULT SETS *整数*

プロシージャーから戻すことのできる結果セットの最大数を指定します。*整数* には、ゼロより大か等しい値を指定する必要があります。ゼロを指定すると、結果セットは戻されません。プロシージャーには、任意の数の結果セットを指定することができますが、取り出しを待機中の結果セットを指定できるのは、一度に 100 のプロシージャーだけです。 SET RESULT SETS ステートメントを発行した場合は、戻される結果の数は、このキーワードに指定した結果セットの数と、SET RESULTS SET ステートメントに指定した結果セットの数のいずれか少ない方です。

結果セットは、プロシージャーが iSeries Access のクライアントや SQL 呼び出しレベル・インターフェースから呼び出される場合にのみ戻されます。結果セットの詳細については、788 ページの 『SET RESULT SETS』を参照してください。

LANGUAGE

その外部プログラムの作成に使用されている言語を指定します。この文節は、外部プログラムが REXX プロシージャーである場合に必要です。

LANGUAGE の指定がない場合は、その LANGUAGE は該当の外部プログラムに関連するプログラム属性情報によって決定されます。該当のプログラムに関連するプログラム属性情報によっては認識可能な言語を特定できない場合には、言語は C であると見なされます。

C
外部プログラムは C で作成されます。

C++
外部プログラムは C++ で作成されます。

CL
外部プログラムは CL で作成されます。

COBOL
外部プログラムは COBOL で作成されます。

COBOLLE
外部プログラムは ILE COBOL で作成されます。

FORTRAN
外部プログラムは FORTRAN で作成されます。

JAVA
外部プログラムは JAVA で作成されます。

PLI
外部プログラムは PL/I で作成されます。

REXX
外部プログラムは REXX プロシージャーです。

RPG
外部プログラムは RPG で作成されます。

RPGLE
外部プログラムは ILE RPG で作成されます。

SPECIFIC *特定名*
プロシージャーを一意に識別する修飾または非修飾名を指定します。暗黙的または明示的修飾子も含め、この*特定名* には、*プロシージャー名* と同じ名前を指定する必要があります。

修飾子を指定しないと、*プロシージャー名* の暗黙的または明示的修飾子が使用されます。修飾子を指定する場合、その修飾子は、*プロシージャー名* の明示的または暗黙的修飾子と同じものにする必要があります。

特定名 を指定しなかった場合、その特定名は、プロシージャー名と同じ名前になります。

DETERMINISTIC または **NOT DETERMINISTIC**
このプロシージャーが、同じ IN 引き数および INOUT 引き数を指定して呼び出された場合に、常に同じ結果を戻すかどうかを指定します。

NOT DETERMINISTIC
このプロシージャーは、同じ IN 引き数および INOUT 引き数を指定して呼び出された場合に、データベース内の参照先データが変更されていない限り、常に同じ結果を戻します。

DETERMINISTIC
このプロシージャーは、同じ IN 引き数および INOUT 引き数を指定して呼び出された場合に、データベース内の参照先データが変更されていなくても、必ずしも同じ結果を戻すとは限りません。

CONTAINS SQL、**READS SQL DATA**、**MODIFIES SQL DATA**、または **NO SQL**
SQL ステートメントがある場合に、このプロシージャーまたはこのプロシージャーから呼び出されたルーチンの中で、どの SQL ステートメントを実行できるかを指定します。各データ・アクセス指示の下で実行できる SQL ステートメントの詳細なリストについては、913 ページの『付録 F. SQL ステートメントの特性』を参照してください。

CONTAINS SQL
このプロシージャーで、SQL データの読み取りも変更も行わない SQL ステートメントを実行できることを指定します。

NO SQL
このプロシージャーではどの SQL ステートメントも実行できないことを指定します。

READS SQL DATA
このプロシージャーに、SQL データを変更しない SQL ステートメントを組み込めることを指定します。

MODIFIES SQL DATA
このプロシージャーで、どのプロシージャーでもサポートされないステートメントを除くすべての SQL ステートメントを実行できることを指定します。

CALLED ON NULL INPUT
パラメーター値のどれかがヌルである場合に、このプロシージャーを呼び出すことを指定します。

FENCED または **NOT FENCED**
このパラメーターは、他のプロダクトとの互換性を保持するために許可されており、DB2 UDB for iSeries で使用されることはありません。

PROGRAM TYPE MAIN
このプロシージャーをメイン・ルーチンとして実行することを指定します。

DBINFO
データベース・マネージャーは、状況情報が入っている構造体をプロシージャーに渡す必要があることを指定します。 表 49 は、DBINFO 構造体の説明を示しています。 DBINFO 構造体についての詳しい情報は、QSYSINC.H 内の組み込みファイル SQLUDF に入っています。

DBINFO は、PARAMETER STYLE DB2SQL でのみ許可されます。

表 49. DBINFO フィールド

フィールド	データ・タイプ	説明
リレーショナル・データベース	VARCHAR(128)	現行サーバーの名前
権限 ID	VARCHAR(128)	実行時権限 ID
CCSID 情報	INTEGER INTEGER INTEGER INTEGER CHAR(8)	ジョブの CCSID 情報。CCSID を識別する情報は、次のとおりです。 • SBCS CCSID • DBCS CCSID • 混合 CCSID • 最初の 3 つの CCSID のどれに該当するかの指示。 • 予約済み DECLARE PROCEDURE ステートメントのパラメーターの 1 つとして明示的に CCSID を指定していない場合は、入力ストリングは、関数の実行時にジョブの CCSID でコード化されるものと見なされます。入力ストリングの CCSID がパラメーターの CCSID と同じではない場合は、この外部関数に渡される入力ストリングは、外部プログラムの呼び出しの前に変換されます。
ターゲット列	VARCHAR(128) VARCHAR(128) VARCHAR(128)	プロシージャーへの呼び出しには適用されません。
バージョンとリリース	CHAR(8)	データベース・マネージャーのバージョン、リリース、および修正レベル。
プラットフォーム	INTEGER	サーバーのプラットフォーム・タイプ。

EXTERNAL NAME *外部プログラム名*
該当のプロシージャーが CALL ステートメントによって呼び出される時点で実行されるプログラムを指定します。このプログラム名は、サーバーに存在するプログラムを識別していなければなりません。このプログラムは、ILE サービス・プログラムであってはなりません。

この名前の妥当性は、サーバーで検査されます。名前の形式が正しくない場合、エラーが戻されます。

外部プログラム名の指定がない場合、外部プログラム名は該当のプロシージャー名と同じであると見なされます。

PARAMETER STYLE

プロシージャーにパラメーターを渡し、プロシージャーから値を戻すために使用する規則を指定します。

SQL

CALL ステートメントに指定されているパラメーターに加えて、幾つかの追加パラメーターをプロシージャーに渡すことを指定します。これらのパラメーターは、次の順序で配列されるように定義されます。

- 最初の N 個のパラメーターは、DECLARE PROCEDURE ステートメント上に指定されるパラメーターです。
- パラメーターの標識変数を表す N 個のパラメーター。
- SQLSTATE の CHAR(5) 出力パラメーター。戻される SQLSTATE は、プロシージャーが成功したかどうかを示します。戻される SQLSTATE は、外部プログラムによって割り当てられたものです。

  ユーザーは、関数からエラーまたは警告を戻すために、外部プログラム内で SQLSTATE を任意の有効な値にセットすることができます。
- 完全修飾プロシージャー名の VARCHAR(517) 入力パラメーター。
- 特定の名前の VARCHAR(128) 入力パラメーター。
- メッセージ・テキストの VARCHAR(70) 出力パラメーター。

渡されるパラメーターについての詳細は、該当するソース・ファイル内の組み込み sqludf を参照してください。例えば、C の場合、sqludf は QSYSINC/H で見つかります。

LANGUAGE JAVA を指定した場合は、PARAMETER STYLE SQL は使用できません。

DB2GENERAL

このプロシージャーに、Java メソッド用として定義されているパラメーター引き渡し規則を使用することを指定します。

PARAMETER STYLE DB2GENERAL を指定できるのは、LANGUAGE JAVA を指定した場合のみです。 Java でのパラメーター引き渡しの詳細については、「Developer Kit for Java」を参照してください。

DB2SQL

CALL ステートメントに指定されているパラメーターに加えて、幾つかの追加パラメーターをプロシージャーに渡すことを指定します。 DB2SQL は、以下の追加パラメーターを最後のパラメーターとして渡すことができるという点以外は、PARAMETER STYLE SQL と同じです。

- DBINFO が DECLARE PROCEDURE ステートメント上に指定されている場合は、dbinfo 構造体のパラメーター。

渡されるパラメーターについての詳細は、該当するソース・ファイル内の組み込み sqludf を参照してください。例えば、C の場合、sqludf は QSYSINC/H で見つかります。

LANGUAGE JAVA を指定した場合は、PARAMETER STYLE DB2SQL は使用できません。

GENERAL
このプロシージャーが CALL に指定されているパラメーターを受け取るようなパラメーター引き渡しメカニズムを使用することを指定します。標識変数に対し、引き数がさらに渡されることはありません。

LANGUAGE JAVA を指定した場合は、PARAMETER STYLE GENERAL は使用できません。

GENERAL WITH NULLS
CALL ステートメントで GENERAL に指定されているパラメーターに加えて、他の引き数もプロシージャーに渡すことを指定します。この追加の引き数には、CALL ステートメントの各パラメーターについてそれぞれ 1 つずつエレメントがある標識配列が含まれています。 C では、これは多くの場合短精度整数の配列です。標識の処理方法に関する詳細については、SQL プログラミング 概念を参照してください。

LANGUAGE JAVA を指定した場合は、PARAMETER STYLE GENERAL は使用できません。

JAVA
このプロシージャーで、Java 言語および SQLJ ルーチンの仕様に準拠するパラメーター引き渡し規則を使用することを指定します。 INOUT および OUT パラメーターは、値を戻しやすくするために、単一項目配列として渡されます。移植性を高めるためには、PARAMETER STYLE JAVA 規則を使用する Java プロシージャーを書く必要があります。

PARAMETER STYLE JAVA を指定できるのは、LANGUAGE JAVA を指定した場合だけです。 Java でのパラメーター引き渡しの詳細については、「Developer Kit for Java」を参照してください。

パラメーターを渡す方法は、外部プロシージャーの言語によって決まります。たとえば、C では、VARCHAR または CHAR パラメーターはヌル文字で終了するストリングとして渡されます。詳細については、SQL プログラミング 概念を参照してください。

使用上の注意

プロシージャー名 の有効範囲は、それが定義されているソース・プログラムです。すなわち、プリコンパイラーに実行依頼されるプログラムです。したがって、別個にコンパイルされた他のプログラムやモジュールから呼び出されるプログラムは、呼び出しプログラムの DECLARE PROCEDURE ステートメントからの属性を使用しません。

DECLARE PROCEDURE ステートメントは、そのプロシージャーを参照するすべての CALL ステートメントよりも前に入れなければなりません。

DECLARE PROCEDURE ステートメントで許されるパラメーターの数は、最高 255 です。 GENERAL WITH NULLS を指定する場合は、最高 254 です。パラメーター・スタイル SQL を指定した場合のパラメーターの許容数は、90 のみです。パラ

メーターの数の最大数は、その外部プログラムのコンパイルに使用されるライセンス・プログラムで許されるパラメーターの最大の数によっても制約されます。

DECLARE PROCEDURE ステートメントが適用されるのは、静的 CALL ステートメントだけです。動的に準備された CALL ステートメント、またはプロシージャー名がホスト変数によって識別されている CALL ステートメントには適用されません。

同義のキーワード

以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- キーワード VARIANT と NOT VARIANT は、NOT DETERMINISTIC と DETERMINISTIC の同義語として使用することができます。
- キーワード NULL CALL と NOT NULL CALL は、CALLED ON NULL INPUT と RETURNS NULL ON NULL INPUT の同義語として使用できます。
- キーワード SIMPLE CALL は、GENERAL の同義語として使用できます。
- DB2GENERAL の同義語として、値 DB2GENRL を使用できます。

例

外部プロシージャー PROC1 を、C プログラムの中で宣言します。CALL ステートメントを使用してこのプロシージャーを呼び出すと、LIB1 ライブラリーの中の PGM1 という名前の COBOL プログラムが呼び出されます。

```
EXEC SQL
DECLARE PROC1 PROCEDURE
        (CHAR(10), CHAR(10))
        EXTERNAL NAME LIB1.PGM1
        LANGUAGE COBOL GENERAL;

EXEC SQL
CALL PROC1 ('FIRSTNAME ','LASTNAME  ');
```

DECLARE STATEMENT

DECLARE STATEMENT ステートメントは、プログラムの文書化の目的に使用します。このステートメントは、準備される SQL ステートメントを識別するのに使用する名前を宣言します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込んで使用します。それ以外の使用法はありません。このステートメントは、実行可能ステートメントではありません。このステートメントは、Java または REXX では使用できません。

権限

権限は不要です。

構文

説明

ステートメント名
準備される SQL ステートメントを識別するために、プログラムで使用される 1 つまたは複数の名前をリストします。

例

この例は、C プログラムにおける DECLARE STATEMENT ステートメントの使用法を示しています。

```
EXEC SQL INCLUDE SQLDA;
void main ()
  {
   EXEC SQL BEGIN DECLARE SECTION ;
   char src_stmt[32000];
   char sqlda[32000]
   EXEC SQL END DECLARE SECTION ;
   EXEC SQL INCLUDE SQLCA ;

   strcpy(src_stmt,"SELECT DEPTNO, DEPTNAME, MGRNO \
     FROM DEPARTMENT \
     WHERE ADMRDEPT = 'A00'");

   EXEC SQL DECLARE OBJ_STMT STATEMENT;

   (Allocate storage from SQLDA)

   EXEC SQL DECLARE C1 CURSOR FOR OBJ_STMT;

   EXEC SQL PREPARE OBJ_STMT FROM :src_stmt;
   EXEC SQL DESCRIBE OBJ_STMT INTO :sqlda;

   (Examine SQLDA)  (Set SQLDATA pointer addresses)
```

```
EXEC SQL OPEN C1;

while (strncmp(SQLSTATE, "00000", 5) )
  {
  EXEC SQL FETCH C1 USING DESCRIPTOR :sqlda;

  (Print results)

  }

EXEC SQL CLOSE C1;
return;
}
```

DECLARE VARIABLE

DECLARE VARIABLE ステートメントは、ホスト変数に対して、デフォルト値以外のサブタイプまたは CCSID を割り当てるのに使用します。

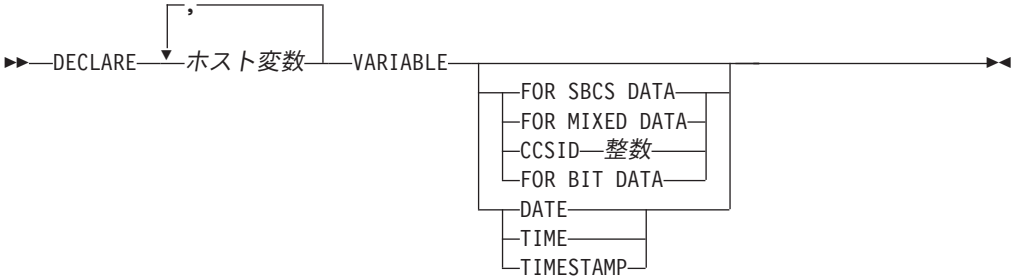
呼び出し

このステートメントは、アプリケーション・プログラムに組み込んで使用します。それ以外の使用法はありません。このステートメントは、実行可能ステートメントではありません。Java または REXX では指定できません。

権限

権限は不要です。

構文



説明

ホスト変数

該当のプログラムで定義されている文字またはグラフィックのストリングのホスト変数の名前を指定します。このホスト変数には、標識変数は指定できません。そのホスト変数の定義は、その変数を参照する DECLARE VARIABLE ステートメントの前でも後でも構いません。

FOR BIT DATA

ホスト変数の値が、コード化文字セットに関連付けられていないことを指定します。したがって、変換は行われません。 FOR BIT DATA が指定されたホスト変数の CCSID は 65535 です。FOR BIT DATA は、グラフィックのホスト変数には指定できません。

FOR SBCS DATA

ホスト変数の値に、SBCS (1 バイト文字セット) データが入ることを指定します。アプリケーション・リクエスターにおけるジョブの CCSID 属性が DBCS 対応でない場合や、ホスト変数の長さが 4 より小さい場合は、FOR SBCS DATA がデフォルト値となります。FOR SBCS DATA の CCSID は、アプリケーション・リクエスターにおけるジョブの CCSID 属性によって決まります。FOR SBCS DATA は、グラフィックのホスト変数には指定できません。

FOR MIXED DATA

ホスト変数の値に、SBCS データと DBCS データが両方とも入るように指示します。 アプリケーション・リクエスターにおけるジョブの CCSID 属性が

DBCS 対応であり、ホスト変数の長さが 3 より大きい場合は、FOR MIXED DATA がデフォルト値となります。FOR DBCS DATA の CCSID は、アプリケーション・リクエスターにおけるジョブの CCSID 属性によって決まります。FOR MIXED DATA は、グラフィックのホスト変数には指定できません。

CCSID *整数*
ホスト変数の値に、CCSID 整数のデータが入ることを指定します。この整数が SBCS の CCSID である場合は、ホスト変数は SBCS データです。この整数が混合データの CCSID である場合は、ホスト変数は混合データです。文字ホスト変数の場合は、指定する CCSID は SBCS または混合 CCSID でなければなりません。

この変数がグラフィック・データ・タイプのものである場合は、指定する CCSID は、DBCS または UCS-2 の CCSID でなければなりません。有効な CCSID のリストについては、897 ページの『付録 E. CCSID の値』の項を参照してください。 UCS-2 データを指示するには、CCSID 13488 を指定するようにしてください。 CCSID が指定されていない場合は、グラフィック・ストリング変数の CCSID は、そのジョブに関連付けられている DBCS CCSID です。

DATE
このホスト変数の値に、日付を示すデータが入ることを指定します。

TIME
このホスト変数の値に、時刻を示すデータが入ることを指定します。

TIMESTAMP
このホスト変数の値に、タイム・スタンプを示すデータが入ることを指定します。

使用上の注意

DECLARE VARIABLE ステートメントは、アプリケーションのうち、SQL ステートメントが有効であればどの位置にでも指定することができます。ただし、次のような例外があります。

- ホスト言語が COBOL または RPG の場合は、DECLARE VARIABLE ステートメントは、その DECLARE VARIABLE ステートメントで指定されるホスト変数を参照する SQL ステートメントより前でなければなりません。
- DATE、TIME、または TIMESTAMP をヌル文字で終了する文字ストリングに対して C で指定すると、その C 宣言の長さは 1 だけ減らされます。

以下の場合は、プリコンパイル時にエラー・メッセージが出されます。

- 存在しないホスト変数を参照している。
- 数値変数が参照されている。
- すでに参照されている変数を参照している。
- 固有でないホスト変数が参照されている。
- SQL ステートメントと DECLARE VARIABLE ステートメントが同一のホスト変数を参照しているときに、DECLARE VARIABLE ステートメントの方がその SQL ステートメントより後に置かれている。
- グラフィックのホスト変数に関して、FOR BIT DATA、FOR SBCS DATA、または FOR MIXED DATA 文節が指定されている。

- グラフィックのホスト変数に関して、SBCS または混合の CCSID が指定されている。
- 文字ホスト変数に関して、DBCS または UCS-2 の CCSID が指定されている。
- DATE、TIME、または TIMESTAMP が文字でないホスト変数に対して指定されている。
- DATE、TIME、または TIMESTAMP に使用されるホスト変数の長さが不足していて、最小の日付、時刻、またはタイム・スタンプの値を指定することができない。

例

この例では、C プログラムの変数 *fred* および *pete* を混合データとして、また *jean* および *dave* を CCSID が 37 の SBCS データとして宣言しています。

```
void main ()
  {
     EXEC SQL BEGIN DECLARE SECTION;
     char fred[10];
     EXEC SQL DECLARE :fred VARIABLE  FOR MIXED DATA;

     decimal(6,0) mary;
     char pete[4];
     EXEC SQL DECLARE :pete VARIABLE FOR MIXED DATA;

     char jean[30];
     char dave[9];
     EXEC SQL DECLARE :jean, :dave VARIABLE  CCSID 37;
     EXEC SQL END DECLARE SECTION;
     EXEC SQL INCLUDE SQLCA;
    ...

  }
```

DELETE

DELETE ステートメントは、表またはビューから行を削除します。ビューから行を削除すると、そのビューの元になっている表から行が削除されます。

このステートメントには、以下の 2 つの形式があります。

- *検索* DELETE 形式。この形式は、1 つまたは複数の行を削除する場合に使用します。必要に応じて、削除される行を検索条件によって限定することができます。
- *位置指定* DELETE 形式。この形式は、1 行だけ削除する場合に使用します。削除される行は、カーソルの現在位置によって決まります。

呼び出し

検索 DELETE ステートメントは、アプリケーション・プログラムに組み込めるほか、対話式に呼び出すこともできます。位置指定 DELETE ステートメントは、必ずアプリケーション・プログラムに組み込まなければなりません。検索 DELETE と位置指定 DELETE は、どちらも動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメントに指定された表またはビューに対して、
  - その表またはビューに対する DELETE 特権
  - 表やビューが入っているライブラリーに対する *EXECUTE システム権限
- 管理権限

ステートメントの権限 ID は、次の場合に表についての DELETE 特権を持ちます。

- その表の所有者である。
- その表に対する DELETE 特権を認可されている、または
- その表に対する *OBJOPR および *DLT のシステム権限を認可されている。

次の場合、ステートメントの権限 ID には、ビューに対する DELETE 特権が与えられます。[54]

- そのビューについての DELETE 特権を認可されている、または
- そのビューに対する *OBJOPR および *DLT のシステム権限を認可されており、しかもそのビューが直接または間接に従属する最初の表またはビューに対する *DLT システム権限を認可されている。すなわち、そのビューの定義で参照されている最初の表またはビュー、およびあるビューが参照されている場合、そのビューの定義で参照されている最初の表またはビュー、以下同様です。

54. ビューが作成される際に、その所有者は、必ずしもそのビューに対する DELETE 特権を獲得するとは限りません。所有者が DELETE 特権を獲得するのは、そのビューが削除を許されており、しかも所有者が副選択で参照されている最初の表に対しても DELETE 特権をもっている場合だけです。

検索 DELETE の*検索条件* に、その表またはビューの列の参照が含まれている場合、そのステートメントの権限 ID によって保持される特権には、以下の 1 つも含まれていなければなりません。

- その表またはビューについての SELECT 特権
- 管理権限

検索条件 に副照会が含まれている場合、ステートメントの権限 ID によって保持される特権には、少なくとも以下の 1 つも含まれていなければなりません。

- その副照会で識別されている各表またはビューについて、
  - 表やビューに対する SELECT 特権、および
  - 表やビューが入っているライブラリーに対する *EXECUTE システム権限
- 管理権限

ステートメントの権限 ID は、以下の場合に表に対する SELECT 特権を持ちます。

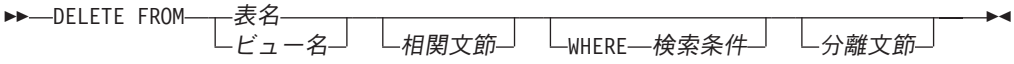
- その表の所有者である。
- その表に対する SELECT 特権が認可されているか、または
- その表についての *OBJOPR および *READ システム権限が認可されている。

ステートメントの権限 ID は、以下の場合にビューに対する SELECT 特権を持ちます。

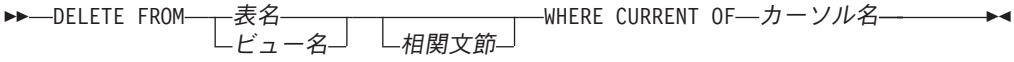
- そのビューの所有者である。
- そのビューに対する SELECT 特権が認可されているか、または
- そのビューについての *OBJOPR および *READ システム権限、およびそのビューが直接または間接に従属しているすべての表やビューについての *READ システム権限が認可されている。すなわち、そのビューの定義で参照されている表やビューのすべて、またそのようなビューが参照される場合、そのビューの定義で参照されている表やビューのすべて、以下同様。

構文

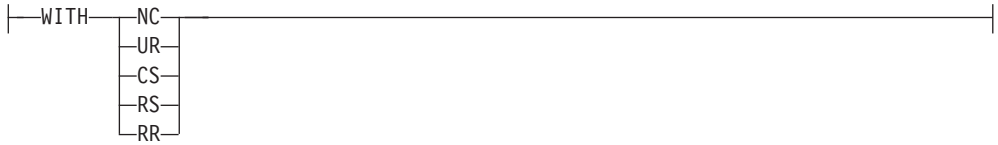
検索 DELETE:



位置指定 DELETE:



分離文節:



説明

FROM *表名* またはビュー名
行を削除する表またはビューを識別します。この名前では、サーバーに存在している表やビューを識別する必要がありますが、カタログ表、カタログ表のビュー、または読み取り専用ビューを識別することはできません。読み取り専用のビューに関する説明については、593 ページの『使用上の注意』の項を参照してください。

相関文節
検索条件 の中で使用して、表やビュー、ならびに、その表やビューの列名を指定することができます。*相関文節* の詳細については、339 ページの『第 4 章 照会』を参照してください。*相関名* の説明については、115 ページの『相関名』を参照してください。

WHERE
削除する行を指定します。この文節は、任意指定です。検索条件を与えたり、カーソル名を指定したりすることができます。この文節を指定しなかった場合は、表またはビューにあるすべての行が削除されます。

検索条件
167 ページの 『検索条件』で説明している任意の検索条件です。副照会以外の*検索条件*で指定する*列名* は、該当する表またはビューの列を識別するものでなければなりません。

検索条件 は、表またはビューの各行に適用されます。削除される行は、*検索条件* の結果が真になった行です。

*検索条件*に副照会が含まれている場合は、行に*検索条件* が適用されるたびにその副照会が実行され、*検索条件* を適用する際に副照会の結果が使用されるたびに、その副照会が実行されると考えることができます。 実際に、相関参照のない副照会は一度しか実行されないことがありますが、相関参照を伴う副照会は各行ごとに一度実行する必要がある場合があります。

副照会が DELETE ステートメントのオブジェクト表、あるいは、CASCADE、SET NULL、または SET DEFAULT の削除規則を使用して従属表を参照する場合、その副照会は、行が削除される前に、完全に評価されます。

CURRENT OF *カーソル名*
削除操作で使用するカーソルを識別します。この*カーソル名* は、宣言されているカーソルを識別しなければなりません。カーソルの宣言については、DECLARE CURSOR ステートメントの「使用上の注意」の項を参照してください。

表またはビューの名前は、このカーソルの SELECT ステートメントの FROM 文節の中でも指定しなければなりません。また、カーソルの結果表は、読み取り専用であってはなりません。読み取り専用の結果表の説明については、597 ページの『DECLARE CURSOR』を参照してください。

DELETE ステートメントが実行される時点では、カーソルはある 1 行 (削除される行) に位置付けられていなければなりません。行が削除されると、カーソルは、結果表にあるその次の行の前に位置付けられます。次の行が存在しない場合は、カーソルは最後の行の後に位置付けられます。

分離文節
このステートメントに関して使用する分離レベルを指定します。*分離文節* を指定しなかった場合は、デフォルトの分離レベルが使用されます。*分離文節* の詳細については、ISOLATION 文節の項を参照してください。

DELETE の規則

トリガー
識別された表または識別されたビューの基礎表が削除トリガーを持つ場合、トリガーが起動します。トリガーが起動された結果、他のステートメントが実行されたり、削除される値に基づいてエラー条件が発生したりすることがあります。

参照保全
識別された表、または識別された表の基礎表が親表である場合、選択された行は、RESTRICT または NO ACTION の削除規則との関係において従属関係をもってはなりません。また、その DELETE は、RESTRICT または NO ACTION の削除規則との関係において従属関係をもつ下層行に波及してはなりません。

削除操作が、RESTRICT または NO ACTION 削除規則によって防止されない場合には、選択された行は削除されます。選択された行に従属する行は、いずれも影響を受けます。

- SET NULL の削除規則に関連する行の従属行の外部キーのヌル可能な列は、ヌル値に設定されます。
- SET DEFAULT の削除規則に関連する行の従属行の外部キーの列は、対応するデフォルト値に設定されます。
- CASCADE の削除規則に関連する行の従属行はいずれも削除され、それらの行についても上記の規則が適用されます。

参照制約 (RESTRICT 削除規則を伴う参照制約以外の) は、ステートメントの終わりで実際上チェックされます。複数行削除の場合、これは、すべての行が削除され、何らかの関連トリガーが起動された後で起こります。

検査制約
検査制約を指定することにより、親表と従属表の関係について SET NULL または SET DEFAULT の削除規則が設定されている場合に、親表の行が削除されないようにすることができます。親表の行のどれかを削除したときに、従属表の中の列がヌルまたはデフォルト値に設定されることになり、そのヌルまたはデフォルト値が原因で検査制約の検索条件の評価結果が偽になる場合は、その行は削除されません。

使用上の注意

なんらかの削除操作を実行しているときにエラーが起きた場合は、このステートメントから変更されたアクション、参照制約、およびトリガーされた SQL ステートメントはロールバックされます (ただし、このステートメントまたは他のトリガーされた SQL ステートメントの分離レベルが NC である場合を除きます)。

適切なロックがすでに存在している場合を除き、正常な DELETE ステートメントの実行の過程で、1 つまたは複数の排他ロックが獲得されます。コミット操作またはロールバック操作によりロックが解除されるまでは、DELETE 操作の効果を認識できるのは以下のものだけです。

- 削除を行ったアプリケーション・プロセス
- 分離列 UR または NC を使用している他のアプリケーション・プロセス

ロックは、他のアプリケーション・プロセスがその表の操作を行うのを防止します。ロックの詳細については、COMMIT、ROLLBACK、および LOCK TABLE の各ステートメントの説明、および 24 ページの『分離レベル』を参照してください。

アプリケーション・プロセスが、その非更新可能なカーソルのいずれかが位置づけられている行を削除する場合、それらのカーソルはそれらの結果表の次の行の前に位置付けられます。次の行 R の前に位置付けられるカーソルが C であると想定します (OPEN、C を介した DELETE、他のカーソルを介した DELETE、または検索 DELETE の結果として)。R の派生元である基礎表に作用する INSERT、UPDATE、および DELETE 操作が発生すると、C を参照する次の FETCH 操作では、必ずしも、R 上に C を置くとは限りません。例えば、この操作で C を R' 上に置く可能性があります。この場合の R' とは、その時点で結果表の次の行になる新しい行です。

DELETE ステートメントが完了すると、削除された行の数が、SQLCA の SQLERRD(3) に戻されます。SQLERRD(3) の値には、CASCADE 削除規則またはトリガーの結果として削除された行の数は含まれません。

SQLCA の SQLERRD(5) の値は、参照制約によって影響を受けた行の数を示します。この値には、CASCADE 削除規則の結果として削除された行の数、および SET NULL または SET DEFAULT 削除規則の結果として、その外部キーが NULL またはデフォルト値に設定された行の数が含まれます。

SQLCA の説明については、 865 ページの『付録 B. SQL 連絡域』を参照してください。

COMMIT(*RR)、COMMIT(*ALL)、COMMIT(*CS)、または COMMIT(*CHG) を指定している場合、1 つの DELETE ステートメントで削除または変更できる行の最大数は 4000000 です。変更された行の数には、トリガー、CASCADE、SET NULL、または SET DEFAULT 参照保全削除規則の結果として、同じコミットメント定義のもとで挿入、更新、または削除された行はいずれも含まれます。

ホスト変数は、REXX プロシージャー内では DELETE ステートメントに使用することはできません。その代わりとして、DELETE は、パラメーター・マーカーを使用して PREPARE と EXECUTE のオブジェクトにする必要があります。

同義のキーワード： 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- キーワード NONE を NC の同義語として使用することができます。
- キーワード CHG を UR の同義語として使用することができます。
- キーワード ALL を RS の同義語として使用することができます。

例

例 1

表 DEPARTMENT から、部門 (DEPTNO) 'D11' を削除します。

```
DELETE FROM DEPARTMENT
  WHERE DEPTNO = 'D11'
```

例 2

表 DEPARTMENT から、すべての部門を削除します (つまり、表を空にします)。

```
DELETE FROM DEPARTMENT
```

例 3

Java プログラム・ステートメントを使用して、接続コンテキスト 'ctx' 上の PROJECT 表から、部門 (DEPTNO) がホスト変数 HOSTDEPT の値 (java.lang.String) に等しいすべてのサブプロジェクト (MAJPROJ が NULL のもの) を削除します。

```
#sql [ctx] { DELETE FROM PROJECT
               WHERE DEPTNO = :HOSTDEPT AND MAJPROJ IS NULL };
```

例 4

以下の例に示す Java プログラム (一部分) は、退職した社員 (JOB) を表示し、要求があれば、接続コンテキスト 'ctx' 上にある EMPLOYEE 表から特定の社員を削除します。

```
#sql iterator empIterator implements sqlj.runtime.ForUpdate
     ( ... );
empIterator C1;

#sql [ctx] C1 = { SELECT * FROM EMPLOYEE
                    WHERE JOB = 'RETIRED' };

#sql {  FETCH C1 INTO ...     };
while ( !C1.endFetch() )  {
   System.out.println( ... );
                ...
   if ( condition for deleting row ) {
       #sql [ctx] { DELETE FROM EMPLOYEE
                      WHERE CURRENT OF C1  };
   }
   #sql { FETCH C1 INTO ...     };
}
C1.close();
```

DESCRIBE

DESCRIBE ステートメントは、準備済みステートメントに関する情報の入手に使用します。準備済みステートメントの説明については、725 ページの『PREPARE』を参照してください。

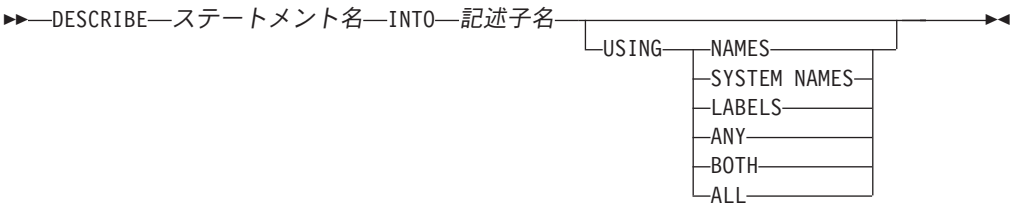
呼び出し

このステートメントは、アプリケーション・プログラムに組み込んで使用します。それ以外の使用法はありません。このステートメントは、動的には準備できない実行可能ステートメントです。

権限

権限は不要です。準備済みステートメントを作成するために必要な権限については、725 ページの『PREPARE』を参照してください。

構文



説明

ステートメント名
その情報を入手したいステートメントを識別します。DESCRIBE ステートメントが実行される時点で、この名前はサーバー側で準備済みステートメントを識別していなければなりません。

INTO *記述子名*
SQL 記述子域 (SQLDA) を識別します。SQLDA については、877 ページの『付録 C. SQL 記述子域 (SQLDA)』で説明しています。DESCRIBE ステートメントを実行する前に、SQLDA の以下の変数をセットしておく必要があります。 (REXX の場合は、規則が異なります。詳細については、DB2 UDB for iSeries ホスト言語での SQL プログラミングを参照してください。)

SQLN SQLDA で用意される SQLVAR オカレンスの数を指定します。DESCRIBE ステートメントを実行する前に、SQLN にゼロ以上の値をセットしておく必要があります。必要なオカレンスの数を決定する手法については、880 ページの『必要な SQLVAR オカレンスの数の決定』を参照してください。

DESCRIBE ステートメントが実行されると、データベース・マネージャーでは、SQLDA の各変数に次のような値を割り当てます。

変数	データベース・マネージャーが戻す情報

SQLDAID 最初の 6 バイトは 'SQLDA ' (つまり、5 文字の後にスペース文字) に設定されます。

7 番目のバイト (SQLDOUBLED と呼ばれる) は、SQLDA の各選択リスト項目 (または、結果表の列) に 2 つ、3 つ、または 4 つの SQLVAR 項目が入っている場合、それぞれ '2'、'3'、または '4' に設定されます。 この技法は、LOB、特殊タイプ、ラベル、およびシステム名に対応するために使用されています。それ以外の場合、SQLDOUBLED はスペース文字に設定されます。

SQLDA 内に DESCRIBE 応答全体を含める余地がない場合、二重フラグはスペースに設定されます。

8 番目のバイトはスペース文字に設定されます。

SQLDABC SQLDA の長さ。

SQLD 準備済みステートメントが SELECT の場合は、その結果表にある列の数であり、それ以外の場合は、0 が割り当てられます。

SQLVAR SQLD の値が 0 の場合、または SQLD の値が SQLN の値より大きい場合は、SQLVAR のオカレンスには値が割り当てられません。

SQLD の値が *n* (ただし、*n* は 0 より大きいが、SQLN の値より小さいか、または等しい値) の場合は、値が SQLVAR の最初の *n* 個のオカレンスに割り当てられ、その結果、SQLVAR の最初のオカレンスには結果表の最初の列の記述が入り、SQLVAR の 2 番目のオカレンスには結果表の 2 番目の列の記述が入ります。以下同様です。SQLVAR オカレンスに割り当てられる値については、878 ページの『SQLVAR のオカレンスのフィールドの説明』を参照してください。

USING

SQLDA のそれぞれの SQLNAME 変数に、どのような値を割り当てるかを指定します。要求した値が存在しない場合は、SQLNAME の長さは 0 にセットされます。

NAMES

列の名前を割り当てます。これは、デフォルト値です。選択リストに名前が明示的にリストされている準備済みステートメントについての DESCRIBE の場合、指定されたその名前が戻されます。

SYSTEM NAMES

列のシステム列名を割り当てます。

LABELS

列のラベルを割り当てます。(列のラベルは、LABEL ステートメントによって定義されます。) ラベルの最初の 20 バイトだけが戻されます。

ANY

列のラベルを割り当てます。列にラベルがない場合は、代わりに列の名前が割り当てられます。

BOTH

列のラベルと名前の両方を割り当てます。この場合、追加情報に応じるために、1 つの列ごとに SQLVAR の 2 ～ 3 つのオカレンスが必要になります

が、その数は、結果セットに特殊タイプが入っているか否かによって決まります。この拡張の SQLVAR 配列を指定するには、SQLN を 2*n か 3*n (この場合の n は、表やビュー内の列数) に設定します。SQLVAR の最初の n 個のオカレンスには、列の名前が入り、2 番目または 3 番目の n オカレンスには、列のラベルが含まれます。 特殊タイプがない場合、SQLVAR 項目の 2 番目のセットにそのラベルが戻されます。 それ以外の場合、ラベルは、SQLVAR の 3 番目のセット内に戻されます。

ALL
ラベル、列名、およびシステム列名を割り当てます。この場合、追加情報に応じるために、1 つの列ごとに SQLVAR の 3 ～ 4 つのオカレンスが必要になりますが、その数は、結果セットに特殊タイプが入っているか否かによって決まります。この拡張の SQLVAR 配列を指定するには、SQLN を 3*n か 4*n (この場合の n は、結果表内の列数) に設定します。SQLVAR の最初の n オカレンスには、システム列名が入ります。 2 番目または 3 番目の n オカレンスには、列のラベルが含まれます。3 番目または 4 番目の n オカレンスには、列名が含まれます。特殊タイプが指定されていない場合、ラベルは、SQLVAR 記入項目の 2 番目のセット内に戻され、列名は、SQLVAR 記入項目の 3 番目のセット内に戻されます。それ以外の場合、ラベルは、SQLVAR 記入項目の 3 番目のセット内に戻され、列名は、SQLVAR 記入項目の 4 番目のセット内に戻されます。

使用上の注意

準備済みステートメントに関する情報は、PREPARE ステートメントの INTO 文節を使用しても入手することができます。

SQLDA の割り振り

DESCRIBE または PREPARE INTO ステートメントを実行する前に、SQLN にゼロ以上の値をセットして、SQLDA に用意する SQLVAR のオカレンスの数を指示するとともに、SQLN の各オカレンスを収容するのに十分な記憶域を割り振っておく必要があります。(REXX では、SQLDA に関して記憶域を割り振る必要はありません。) 準備済み SELECT ステートメントの結果表にある列の記述を入手する場合は、SQLVAR のオカレンスの数を、結果表にある列の数以上にしなければなりません。さらに、USING BOTH や USING ALL を指定している場合、あるいは、列に LOB や特殊タイプを指定している場合は、SQLVAR のオカレンス数として、列数の 2、3、または 4 倍の数値を指定する必要があります。詳細については、880 ページの『必要な SQLVAR オカレンスの数の決定』を参照してください。

提供されるオカレンスが不足していて、オカレンスのすべてのセットを戻せるとは限らない場合、SQLN は、すべての情報を戻すのに必要なオカレンスの合計数に設定されます。それ以外の場合、SQLN は、列数に設定されます。

SQLDA を割り振る際に考えられる方法を、以下に 3 つ示します。

方法 1: アプリケーションで処理する必要があるどの選択リストにも対応できるように、SQLVAR のオカレンス数を十分にとって SQLDA を割り振ります。極端なことをいえば、SQLVAR の数を、結果表で許容されている列の最大数の 4 倍にすることも可能です。いったん割り振りが終了すれば、アプリケーションでは、この SQLDA を繰り返し使用することができます。

この方法は、大量の記憶域を使用します。また、ある特定の選択リストで、その記憶域のごく一部しか使用しないとしても、記憶域の割り振りが解除されることはありません。

方法 2: 選択リストを処理するたびに、以下の 3 つのステップを繰り返し実行します。

1. SQLVAR のオカレンスがない SQLDA、つまり SQLN がゼロの SQLDA を使用して、DESCRIBE ステートメントを実行します。SQLD に戻される値は、結果表の列数です。これは、SQLVAR のオカレンスの必要数か、必要数の 1/2、1/3、または 1/4 のいずれかです。SQLVAR 項目がないので、警告が出されます。その警告で示された SQLSTATE が 01005 の場合、SQLVAR 項目数は、SQLD に戻された値の 2 倍、3 倍、または 4 倍のはずです。詳細については、880 ページの『必要な SQLVAR オカレンスの数の決定』を参照してください。
2. SQLD に戻された値を使用して、SQLVAR の十分なオカレンスを指定して、SQLDA を割り振ります。
3. 次に、この新しい SQLDA を使用して、再び DESCRIBE ステートメントを実行します。

この方法をとると、方法 1 より記憶域管理が向上しますが、DESCRIBE ステートメントの数が 2 倍になります。

方法 3: 選択リストの大半 (ほとんど全部) を扱うのに十分な範囲で、なるべく小さな SQLDA を割り振ります。SQLDA が小さ過ぎることが原因で、DESCRIBE ステートメントの実行に失敗した場合は、より大きな SQLDA を割り振って、再び DESCRIBE ステートメントを実行します。新しい SQLDA では、最初の DESCRIBE の実行で SQLD に戻された値を使用して、SQLVAR のオカレンス数を指定してください。

この方法は、方法 1 と方法 2 を組み合わせたものです。方法 3 の効果は、最初の SQLDA のサイズを適切に選定できるかどうかによって左右されます。

例

C プログラムの中で、SQLVAR のオカレンスなしの SQLDA を使用して DESCRIBE ステートメントを実行します。SQLD がゼロより大きければ、その値を使用して、SQLVAR に必要なオカレンス数を指定した SQLDA を割り振ります。その後で、新しい SQLDA を使用して DESCRIBE ステートメントを実行します。

```
EXEC SQL   BEGIN DECLARE SECTION;
  char stmt1_str [200];
EXEC SQL  END DECLARE SECTION;
EXEC SQL  INCLUDE SQLDA;
EXEC SQL  DECLARE DYN_CURSOR  CURSOR FOR STMT1_NAME;

... /* code to prompt user for a query, then to generate    */
    /* a select-statement in the stmt1_str                  */
EXEC SQL  PREPARE STMT1_NAME FROM :stmt1_str;

... /* code to set SQLN to zero and to allocate the SQLDA   */
EXEC SQL  DESCRIBE STMT1_NAME INTO :sqlda;

... /* code to check that SQLD is greater than zero, to set */
    /* SQLN to SQLD, then to re-allocate the SQLDA          */
```

DESCRIBE

```
EXEC SQL  DESCRIBE STMT1_NAME INTO :sqlda;

... /* code to prepare for the use of the SQLDA            */
EXEC SQL  OPEN DYN_CURSOR;

... /* loop to fetch rows from result table               */
EXEC SQL  FETCH DYN_CURSOR USING DESCRIPTOR :sqlda;
.
.
.
```

DESCRIBE TABLE

DESCRIBE TABLE ステートメントは、表またはビューに関する情報を入手します。

呼び出し

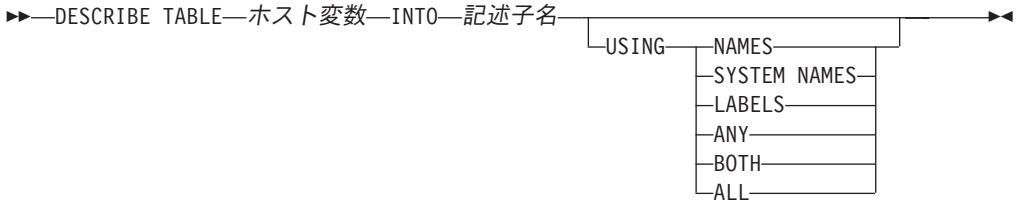
このステートメントは、アプリケーション・プログラムに組み込んで使用します。それ以外の使用法はありません。このステートメントは、動的には準備できない実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメントに指定された表またはビューに対して、
  - その表またはビューに対する *OBJOPR システム権限
  - 表やビューが入っているライブラリーに対する *EXECUTE システム権限
- 管理権限

構文



説明

ホスト変数

その情報を入手したい表またはビューを識別します。DESCRIBE TABLE ステートメントを実行する時点で、

- この名前は、サーバーにある表またはビューを識別するものでなければなりません。
- ホスト変数は文字ストリング変数または UCS-2 グラフィック・ストリング変数でなければならず、標識変数を含んでいてはなりません。
- ホスト変数内に入れる表名の長さがホスト変数の長さより短い場合は、左そろえにして右側にブランクを埋め込まなければなりません。
- 表の名前は、区切り文字付の名前でない限り、大文字にしなければなりません。

DESCRIBE TABLE ステートメントを実行すると、データベース・マネージャーによって、SQLDA の各変数には次のような値が割り当てられます。

INTO *記述子名*

SQL 記述子域 (SQLDA) を識別します。SQLDA については、 877 ページの

『付録 C. SQL 記述子域 (SQLDA)』で説明しています。DESCRIBE TABLE ステートメントを実行する前に、SQLDA の以下の変数をセットしておく必要があります。 (REXX の場合は、規則が異なります。詳細については、DB2 UDB for iSeries ホスト言語での SQL プログラミングを参照してください。)

SQLN SQLDA で用意される SQLVAR オカレンスの数を指定します。DESCRIBE TABLE ステートメントを実行する前に、SQLN をゼロ以上の値にセットしておく必要があります。必要なオカレンスの数を決定する手法については、880 ページの『必要な SQLVAR オカレンスの数の決定』を参照してください。

変数	データベース・マネージャーが戻す情報
SQLDAID	最初の 6 バイトは 'SQLDA ' (つまり、5 文字の後にスペース文字) に設定されます。 7 番目のバイト (SQLDOUBLED と呼ばれる) は、SQLDA の各選択リスト項目 (または、結果表の列) に 2 つ、3 つ、または 4 つの SQLVAR 項目が入っている場合、それぞれ '2'、'3'、または '4' に設定されます。 この技法は、LOB、特殊タイプ、ラベル、およびシステム名に対応するために使用されています。それ以外の場合、SQLDOUBLED はスペース文字に設定されます。 SQLDA 内に DESCRIBE 応答全体を含める余地がない場合、二重フラグはスペースに設定されます。
SQLDABC	SQLDA の長さ。
SQLD	参照している表またはビューの列の個数。
SQLVAR	SQLD の値が 0 の場合、または SQLD の値が SQLN の値より大きい場合は、SQLVAR のオカレンスには値が割り当てられません。 SQLD の値が *n* (*n* は 0 より大きく、SQLN の値より小さいか等しい値) の場合、SQLVAR の最初の *n* 個のオカレンスに値が割り当てられます。SQLVAR の最初のオカレンスには、表またはビューの最初の列に関する記述が入り、SQLVAR の 2 番目のオカレンスには、表またはビューの 2 番目の列に関する記述が入るというように、表またはビューの対応する列に関する記述が順に入ります。SQLVAR オカレンスに割り当てられる値については、878 ページの『SQLVAR のオカレンスのフィールドの説明』を参照してください。

USING

SQLDA のそれぞれの SQLNAME 変数に、どのような値を割り当てるかを指定します。要求した値が存在しない場合は、SQLNAME の長さは 0 にセットされます。

NAMES

列の名前を割り当てます。これは、デフォルト値です。

SYSTEM NAMES

列のシステム列名を割り当てます。

LABELS
列のラベルを割り当てます。(列のラベルは、LABEL ステートメントによって定義されます。) ラベルの最初の 20 バイトだけが戻されます。

ANY
列のラベルを割り当てます。列にラベルがない場合は、代わりに列の名前が割り当てられます。

BOTH
列のラベルと名前の両方を割り当てます。この場合、追加情報に応じるために、1 つの列ごとに SQLVAR の 2 ～ 3 つのオカレンスが必要になりますが、その数は、結果セットに特殊タイプが入っているか否かによって決まります。この拡張の SQLVAR 配列を指定するには、SQLN を 2*n* か 3*n* (この場合の *n* は、表やビュー内の列数) に設定します。SQLVAR の最初の *n* 個のオカレンスには、列の名前が入り、2 番目または 3 番目の *n* オカレンスには、列のラベルが含まれます。 特殊タイプがない場合、SQLVAR 項目の 2 番目のセットにそのラベルが戻されます。 それ以外の場合、ラベルは、SQLVAR の 3 番目のセット内に戻されます。

ALL
ラベル、列名、およびシステム列名を割り当てます。この場合、追加情報に応じるために、1 つの列ごとに SQLVAR の 3 ～ 4 つのオカレンスが必要になりますが、その数は、結果セットに特殊タイプが入っているか否かによって決まります。この拡張の SQLVAR 配列を指定するには、SQLN を 3*n* か 4*n* (この場合の *n* は、結果表内の列数) に設定します。SQLVAR の最初の *n* オカレンスには、システム列名が入ります。 2 番目または 3 番目の *n* オカレンスには、列のラベルが含まれます。3 番目または 4 番目の *n* オカレンスには、列名が含まれます。特殊タイプが指定されていない場合、ラベルは、SQLVAR 記入項目の 2 番目のセット内に戻され、列名は、SQLVAR 記入項目の 3 番目のセット内に戻されます。それ以外の場合、ラベルは、SQLVAR 記入項目の 3 番目のセット内に戻され、列名は、SQLVAR 記入項目の 4 番目のセット内に戻されます。

使用上の注意

DESCRIBE TABLE ステートメントを実行する前に、SQLN にゼロ以上の値をセットして、SQLDA に用意する SQLVAR のオカレンスの数を指示するとともに、SQLN の各オカレンスを収容するのに十分な記憶域を割り振っておく必要があります。表またはビューの列の記述を取得するには、SQLVAR のオカレンスの数が列の数以上でなければなりません。さらに、USING BOTH や USING ALL を指定している場合、あるいは、列に LOB や特殊タイプを指定している場合は、SQLVAR のオカレンス数として、列数の 2、3、または 4 倍の数値を指定する必要があります。詳細については、880 ページの『必要な SQLVAR オカレンスの数の決定』を参照してください。

提供されるオカレンスが不足していて、オカレンスのすべてのセットを戻せるとは限らない場合、SQLN は、すべての情報を戻すのに必要なオカレンスの合計数に設定されます。それ以外の場合、SQLN は、列数に設定されます。

SQLDA の割り振りのために使用する方法については、644 ページの『SQLDA の割り振り』を参照してください。

例

C プログラムの中で、SQLVAR のオカレンスなしの SQLDA を使用して DESCRIBE ステートメントを実行します。SQLD がゼロより大きければ、その値を使用して、SQLVAR に必要なオカレンス数を指定した SQLDA を割り振ります。その後で、新しい SQLDA を使用して DESCRIBE ステートメントを実行します。

```
EXEC SQL   BEGIN DECLARE SECTION;
  char table_name[201];
EXEC SQL  END DECLARE SECTION;
EXEC SQL  INCLUDE SQLDA;
EXEC SQL  DECLARE DYN_CURSOR CURSOR FOR STMT1_NAME;

.../*code to prompt user for a table or view */
.../*code to set SQLN to zero and to allocate the SQLDA */
EXEC SQL  DESCRIBE TABLE :table_name INTO :sqlda;

... /* code to check that SQLD is greater than zero, to set */
    /* SQLN to SQLD, then to re-allocate the SQLDA          */
EXEC SQL  DESCRIBE TABLE :table_name INTO :sqlda;

.
.
.
```

DISCONNECT

DISCONNECT ステートメントは、無保護会話の 1 つまたは複数の接続を終了させます。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むか、または対話式に呼び出すことができます。このステートメントは、動的には準備できない実行可能ステートメントです。Java または REXX では指定できません。

DISCONNECT はトリガーでは使用できません。リモート・サーバーで外部プロシージャーを呼び出す場合、その外部プロシージャーでは DISCONNECT は使用できません。

権限

権限は不要です。

構文

説明

サーバー名 または *ホスト変数*
指定したサーバー名、または指定したホスト変数に入っているサーバー名によってサーバーを識別します。ホスト変数を指定する場合、

- その変数は、文字ストリング変数でなければなりません。
- 標識変数を伴っていてはなりません。
- サーバー名は、その変数内で左寄せし、通常 ID の形成の規則に従っていなければなりません。
- サーバー名の長さが、ホスト変数の長さよりも短い場合、右側をブランクで埋めなければなりません。

DISCONNECT ステートメントが実行される時点で、指定したサーバー名またはホスト変数に入っているサーバー名は、その活動化グループの既存の休止接続、または現行接続を識別していなければなりません。識別された接続は、保護会話を使用できません。

CURRENT
活動化グループの現行接続を識別します。活動化グループは接続状態でなければなりません。その現行接続は、保護会話を使用してはなりません。

ALL または **ALL SQL**
活動化グループの既存のすべての接続 (ローカルおよびリモートの接続の両方) を識別します。このステートメントの実行時に接続が存在しない場合、エラーや警告は起こりません。接続は、いずれも保護会話を使用することはできません。

使用上の注意

識別される接続は、現行作業単位の過程で SQL ステートメントを実行するのに使用された接続であってはならず、また保護会話の接続であってはなりません。保護会話の接続を終了するには、RELEASE ステートメントを使用します。ローカル接続は、保護会話と見なされることはありません。

DISCONNECT ステートメントが正常に実行された場合は、識別された接続はそれぞれ終了します。現行接続が遮断されると、その活動化グループは未接続状態になります。

DISCONNECT ステートメントが不成功の場合、活動化グループの接続状態およびその接続の状態は変わりません。

CONNECT (タイプ 1) の使用は、DISCONNECT の使用を妨げることはありません。

DISCONNECT は、カーソルをクローズし、リソースを解放し、その接続のそれ以上の使用を防止します。

ROLLBACK は、DISCONNECT によって終了した接続を再接続することはありません。

リモートの接続を作成し、維持するにはリソースが必要になります。したがって、再使用の予定がないリモート接続は、できるだけ早く終了するようにする必要があり、再使用の予定があるリモート接続は、遮断されないようにする必要があります。

DISCONNECT ステートメントは、コミット操作の直後に実行しなければなりません。DISCONNECT が現行接続の終了に使用される場合、次に実行される SQL ステートメントは、CONNECT または SET CONNECTION でなければなりません。

DISCONNECT ALL は、ローカル・サーバーとの接続を終了させます。接続に、WITH HOLD 文節を指定して定義されたオープン・カーソルがある場合でも、接続は終了します。

例

例 1 : TOROLAB1 との接続は不要になりました。次のステートメントは、コミット操作の後で実行されます。

```
EXEC SQL DISCONNECT TOROLAB1;
```

例 2 : 現行接続は不要になりました。次のステートメントは、コミット操作の後で実行されます。

```
EXEC SQL DISCONNECT CURRENT;
```

例 3 : 既存の接続は不要になりました。次のステートメントは、コミット操作の後で実行されます。

```
EXEC SQL DISCONNECT ALL;
```

DROP

DROP ステートメントは、オブジェクトを削除します。削除されるオブジェクトに直接または間接的に依存しているオブジェクトがあれば、それも削除されます。オブジェクトを削除すると、そのオブジェクトの記述も必ずカタログから削除されます。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことも、対話式に呼び出すこともできます。このステートメントは、動的に準備できる実行可能ステートメントです。

権限

表、ビュー、索引、別名またはパッケージを除去するには、ステートメントの権限 ID によって保持される特権には、少なくとも次の 1 つが含まれていなければなりません。

- 次のシステム権限
  - 除去したいオブジェクトについての *OBJOPR および *OBJEXIST システム権限。
  - 該当のオブジェクトが表またはビューである場合は、その表またはビューに従属しているビュー、索引、および論理ファイルに対する *OBJOPR および *OBJEXIST システム権限。
  - 除去したいオブジェクトが入っているライブラリーについての *EXECUTE システム権限。
- 管理権限。

スキーマを除去するには、ステートメントの権限 ID によって保持される特権には、少なくとも次の 1 つが含まれていなければなりません。

- 次のシステム権限
  - 除去したいライブラリーに対する *OBJEXIST、*OBJOPR、*EXECUTE、および *READ システム権限。
  - そのスキーマのすべてのオブジェクトについての *OBJOPR および *OBJEXIST システム権限、およびそのスキーマの中の表やビューに従属するビュー、索引、および論理ファイルについての *OBJOPR および *OBJEXIST システム権限。
  - そのスキーマの中のその他のオブジェクト・タイプの削除に必要な追加の権限。スキーマにデータ・ディクショナリーが入っている場合における、そのデータ・ディクショナリーに対する *OBJMGT、およびジャーナル・レシーバーに対する一部のシステム・データ権限がその例です。詳しくは、iSeries 機密保護解説書  を参照してください。
- 管理権限

特殊タイプ を除去するためには、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- 次のシステム権限

  - 除去したい 特殊タイプ についての *OBJOPR および *OBJEXIST システム権限。
  - SYSTYPES、SYSPARMS、および SYSROUTINES カタログ表に対する DELETE 特権
  - QSYS2 ライブラリーに対する *EXECUTE システム権限
- 管理権限

関数を除去するためには、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- 次のシステム権限
  - SQL 関数の場合、その関数に関連したプログラム・オブジェクトに対する *OBJEXIST システム権限
  - SYSFUNCS および SYSPARMS カタログ表に対する DELETE 特権
  - QSYS2 ライブラリーに対する *EXECUTE システム権限
- 管理権限

プロシージャーを除去するためには、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- 次のシステム権限
  - SQL プロシージャーの場合、そのプロシージャーに関連したプログラム・オブジェクトに対する *OBJEXIST システム権限
  - SYSPROCS および SYSPARMS カタログ表に対する DELETE 特権
  - QSYS2 ライブラリーに対する *EXECUTE システム権限
- 管理権限

ステートメントの権限 ID は、次の場合に表についての DELETE 特権を持ちます。

- その表の所有者である。
- その表に対する DELETE 特権を認可されている、または
- その表に対する *OBJOPR および *DLT のシステム権限を認可されている。

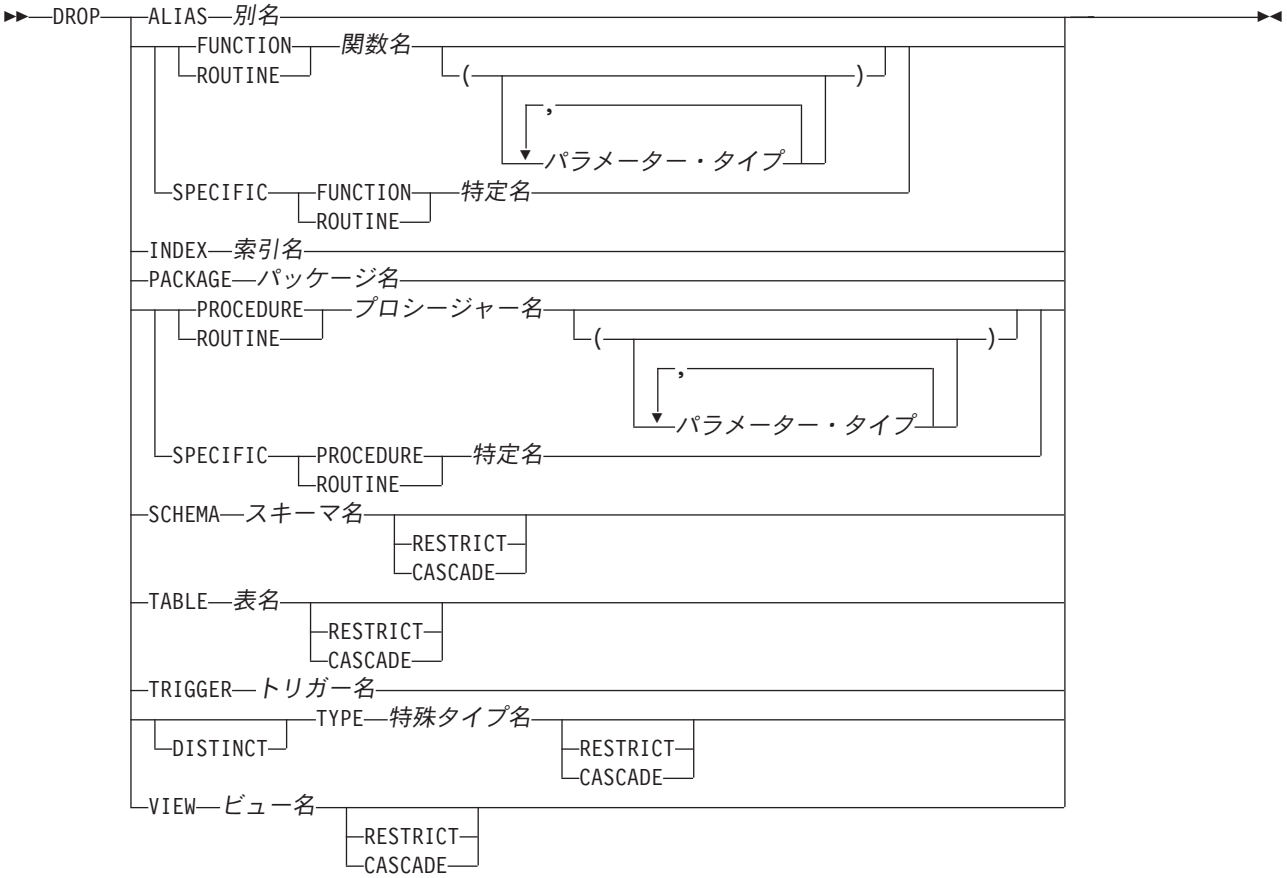
トリガーを除去するには、ステートメントの権限 ID が保持する特権に、次のうち少なくともいずれか 1 つを含める必要があります。

- 次の権限
  - 物理ファイル削除トリガー (RMVPFTRG) コマンドに対する *USE システム権限
  - トリガーの対象表に対する権限
    - 対象表に対する ALTER 特権
    - 対象表が入っているライブラリーに対する *EXECUTE システム権限
  - 削除されるトリガーが SQL トリガーの場合
    - トリガー・プログラム・オブジェクトに対する *OBJEXIST システム権限
    - トリガーが入っているライブラリーに対する *EXECUTE システム権限
- 管理権限

次のいずれかに該当する場合、ステートメントの権限 ID は、その表に対する ALTER 特権を持っています。

- その表の所有者である。
- その表に対する ALTER 特権が認可されている。
- その表に対する *OBJALTER または *OBJMGT のいずれかのシステム権限が認可されている。

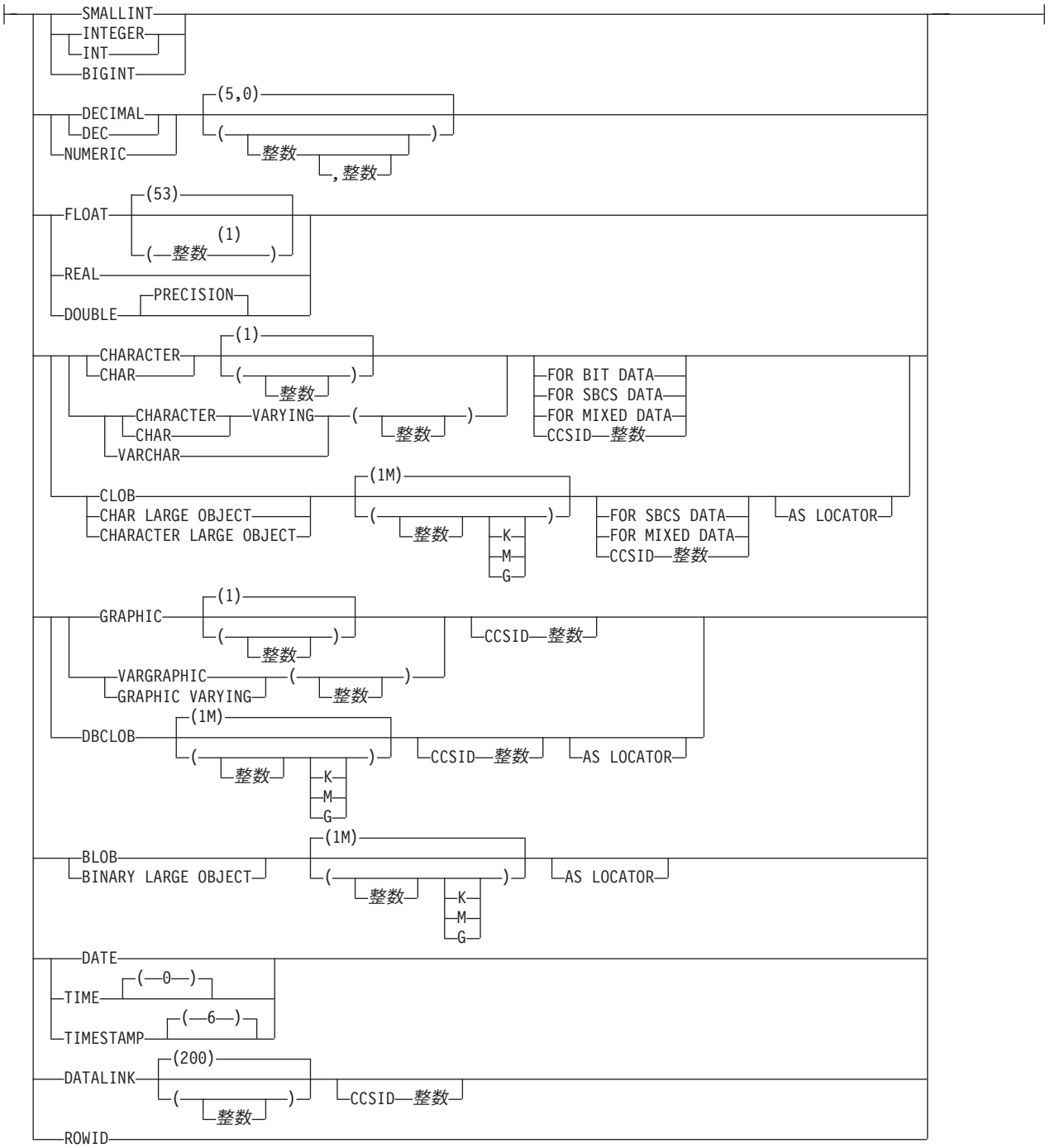
構文



パラメーター・タイプ:

DROP

組み込みタイプ:



注:

1 突き合わせは、データ・タイプ (REAL または DOUBLE) に基づいて行われるので、精度に指定された値は、関数の作成時に 指定された値に一致している必要はありません。

説明

ALIAS *別名*

除去する別名を識別します。この*別名* は、現行サーバーに存在している別名を示すものでなければなりません。指定した別名は、スキーマから削除されます。

別名を除去しても、その別名を使用して定義された制約やビューには影響を与えません。別名は、関数、パッケージ、プロシージャー、プログラム、またはトリガーで参照されているかどうかに関係なく、除去できます。別名を参照するアクセス・プランは、そのアクセス・プランが次回に使用されるときに、暗黙的に再準備されます。そのときにその別名が存在しないと、エラーが戻されます。

FUNCTION

除去する関数を識別します。除去する関数は、それぞれその名前、関数シグニチャー、あるいは特定名によって識別することができます。関数解決の規則 (およびパス) は使用されません。指定した関数は、スキーマから削除されます。これが、SQL 関数の場合、またはソース化関数の場合、その関数に関連したサービス・プログラム (*SRVPGM) も削除されます。これが外部関数の場合、CREATE FUNCTION ステートメントに指定されているプログラムまたはサービス・プログラム内に保管されている情報も、そのオブジェクトから削除されます。その関数に対する特権も、すべて除去されます。

CREATE DISTINCT TYPE ステートメントによって暗黙的に生成された関数は、除去できません。

関数は、別の関数がそれに従属している場合は、除去できません。関数が別の関数に従属するのは、CREATE FUNCTION ステートメントの SOURCE 文節でそれが識別されている場合です。関数は、関数、パッケージ、プロシージャー、プログラム、またはトリガーで参照されているかどうかに関係なく、除去できます。関数を参照するアクセス・プランは、そのアクセス・プランが次回に使用されるときに、暗黙的に再準備されます。そのときにその関数が存在しないと、エラーが戻されます。

FUNCTION *関数名*

関数名 では、現行サーバーに存在している厳密に 1 つの関数を識別する必要があります。指定されたスキーマまたは暗黙のスキーマの中に、指定された名前の関数が複数ある場合、エラーが戻されます。

FUNCTION *関数名 (パラメーター・タイプ, ...)*

関数名 (パラメーター・タイプ, ...) は、現行サーバーに存在していて、指定された関数シグニチャーを持つ関数を識別する必要があります。 指定されたパラメーターは、CREATE FUNCTION ステートメント上の対応する位置に指定されたデータ・タイプと一致していなければなりません。除去する関数インスタンスを識別する場合、データ・タイプの数とデータ・タイプの論理連結が使用されます。*関数名 ()* を指定する場合、識別される関数にパラメーターを使用することはできません。

関数名
関数の名前を識別します。

(パラメーター・タイプ, ...)
関数のパラメーターを識別します。

非修飾の特殊タイプ名を指定する場合、データベース・マネージャーはその特殊タイプのスキーマ名を解決するための SQL パスを検索します。

長さ属性、精度属性、あるいは位取り属性があるデータ・タイプの場合、値を指定することも、1 組の空の括弧を使用することもできます。

- 中が空の括弧は、データ・タイプが一致しているか否かの判別時にデータベース・マネージャー によって属性が無視されることを示します。
- 長さ属性、精度属性、あるいは位取り属性に特定の値を使用する場合、その値は、CREATE FUNCTION ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。
- 長さ属性、精度属性、または位取り属性が明示的に指定されておらず、空の括弧も指定されていない場合、該当のデータ・タイプのデフォルト属性が暗黙指定されます。例えば以下のようになります。

```
CHAR      CHAR(1)
GRAPHIC   GRAPHIC(1)
DECIMAL   DECIMAL(5,0)
FLOAT     DOUBLE (length of 8)
```

暗黙の長さは、CREATE FUNCTION ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。データ・タイプのデフォルト長さの全リストは、542 ページの『CREATE TABLE』 を参照してください。

サブタイプまたは CCSID 属性のあるデータ・タイプの場合は、FOR DATA 文節または CCSID 文節の指定は任意選択です。どちらか一方の文節を省略すると、データ・タイプが一致しているか否かの判別時にデータベース・マネージャー によって属性が無視されることが指示されます。 どちらか一方の文節を指定する場合は、CREATE FUNCTION ステートメントに暗黙的または明示的に指定されている値と一致させる必要があります。

SPECIFIC FUNCTION *特定名*
この*特定名* は、現行サーバーに存在している特定の関数を識別していなければなりません。

INDEX *索引名*
除去する索引を識別します。この*索引名* は、現行サーバーに存在している索引を示すものでなければなりません。指定した索引は、スキーマから削除されます。

索引は、関数、パッケージ、プロシージャー、プログラム、またはトリガーで参照されているかどうかに関係なく、削除できます。索引を参照するアクセス・プランは、そのアクセス・プランが次回に使用されるときに、暗黙的に再準備されます。

PACKAGE *パッケージ名*
除去するパッケージを識別します。このパッケージ名 は、現行サーバーに存在しているパッケージを識別していなければなりません。指定したパッケージは、スキーマから削除されます。パッケージに対する特権も、すべて削除されます。

パッケージは、関数、パッケージ、プロシージャー、プログラム、またはトリガーで参照されているかどうかに関係なく、削除できます。このパッケージを参照するアクセス・プランは、そのアクセス・プランが次回に使用されるときに、暗黙的に再準備されます。そのときにそのパッケージが存在しないと、エラーが戻されます。

PROCEDURE

除去したいプロシージャーを識別します。除去するプロシージャーは、それぞれその名前、プロシージャー・シグニチャー、あるいは特定名によって識別することができます。プロシージャー解決の規則 (およびパス) は使用されません。

プロシージャーは、関数、パッケージ、プロシージャー、プログラム、またはトリガーで参照されているかどうかに関係なく、削除できます。プロシージャーを参照するアクセス・プランは、そのアクセス・プランが次回に使用されるときに、暗黙的に再準備されます。そのときにそのプロシージャーが存在しないと、エラーが戻されます。

PROCEDURE *プロシージャー名*

プロシージャー名 は、現行サーバーに存在しているただ 1 つのプロシージャーを識別していなければなりません。このプロシージャーには、パラメーターをいくつでも定義することができます。指定されたスキーマまたは暗黙のスキーマの中に、指定された名前のプロシージャーが複数ある場合、エラーが戻されます。

PROCEDURE *プロシージャー名 (パラメーター・タイプ, ...)*

プロシージャー名 (パラメーター・タイプ, ...) では、現行サーバーに存在していて、指定されたプロシージャー・シグニチャーを持つプロシージャーを識別する必要があります。指定されたパラメーターは、CREATE PROCEDURE ステートメント上の対応する位置に指定されたデータ・タイプと一致していなければなりません。除去するプロシージャー・インスタンスを識別する場合、データ・タイプの数とデータ・タイプの論理連結が使用されます。*プロシージャー名 ()* を指定する場合、識別されるプロシージャーにパラメーターを使用することはできません。

プロシージャー名

プロシージャーの名前を識別します。

(パラメーター・タイプ, ...)

プロシージャーのパラメーターを識別します。

非修飾の特殊タイプ名を指定する場合、データベース・マネージャーはその特殊タイプのスキーマ名を解決するための SQL パスを検索します。

長さ属性、精度属性、あるいは位取り属性があるデータ・タイプの場合、値を指定することも、1 組の空の括弧を使用することもできます。

- 中が空の括弧は、データ・タイプが一致しているか否かの判別時にデータベース・マネージャー によって属性が無視されることを示します。
- 長さ属性、精度属性、あるいは位取り属性に特定の値を使用する場合、その値は、CREATE PROCEDURE ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。

- 長さ属性、精度属性、または位取り属性が明示的に指定されておらず、空の括弧も指定されていない場合、該当のデータ・タイプのデフォルト属性が暗黙指定されます。例えば以下のようになります。

```
CHAR      CHAR(1)
GRAPHIC   GRAPHIC(1)
DECIMAL   DECIMAL(5,0)
FLOAT     DOUBLE (length of 8)
```

  暗黙の長さは、CREATE PROCEDURE ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。データ・タイプのデフォルト長さの全リストは、542 ページの『CREATE TABLE』 を参照してください。

サブタイプまたは CCSID 属性のあるデータ・タイプの場合は、FOR DATA 文節または CCSID 文節の指定は任意選択です。どちらか一方の文節を省略すると、データ・タイプが一致しているか否かの判別時にデータベース・マネージャー によって属性が無視されることが指示されます。 どちらか一方の文節を指定する場合は、CREATE PROCEDURE ステートメントに暗黙的または明示的に指定されている値と一致させる必要があります。

SPECIFIC PROCEDURE *特定名*

特定名 は、現行サーバーに存在している特定のプロシージャーを識別していなければなりません。

指定したプロシージャーは、カタログ表 SYSPROCS および SYSPARMS から削除されます。これが SQL プロシージャーの場合、その SQL プロシージャーに関連したプログラム (*PGM) オブジェクトも削除されます。そのプロシージャーに対する特権も、すべて削除されます。

SCHEMA *スキーマ名*

除去するスキーマを識別します。*スキーマ名* は、現行サーバーに存在しているスキーマを識別していなければなりません。指定したスキーマは削除されます。スキーマ内の各オブジェクトは、指定の削除オプション (CASCADE、RESTRICT、または neither) を使用して該当する DROP ステートメント実行した場合と同様に除去されます。これらのオブジェクトに従属するオブジェクトの扱いについては、これらのオブジェクト・タイプの DROP の説明を参照してください。

DROP SCHEMA は、コミット・レベルが *NONE の場合にのみ有効です。

Neither CASCADE nor RESTRICT

スキーマが別のスキーマ内の関数、パッケージ、プロシージャー、プログラム、表、またはトリガーで参照されている場合も、スキーマは除去されることを指定します。そのスキーマを参照するアクセス・プランは、アクセス・プランが次回に使用されるときに、暗黙的に再準備されます。そのときにそのスキーマが存在しないと、エラーが戻されます。

CASCADE

このスキーマを参照しているトリガーをすべて除去することを指定します。そのスキーマが別のスキーマ内の関数、パッケージ、プロシージャー、またはプログラムで参照されている場合、そのスキーマを参照するアクセス・プ

ランは、アクセス・プランが次回に使用されるときに、暗黙的に再準備されます。そのときにそのスキーマが存在しないと、エラーが戻されます。

RESTRICT
スキーマが別のスキーマ内の SQL トリガーで参照されている場合、そのスキーマは除去できないことを指定します。そのスキーマが別のスキーマ内の関数、パッケージ、プロシージャー、またはプログラムで参照されている場合、そのスキーマを参照するアクセス・プランは、アクセス・プランが次回に使用されるときに、暗黙的に再準備されます。そのときにそのスキーマが存在しないと、エラーが戻されます。

TABLE *表名*
除去する表を識別します。この*表名* は、現行サーバーに存在している基礎表を識別していなければなりませんが、カタログ表を識別するものであってはなりません。指定した表は、スキーマから削除されます。その表に関する特権、制約、およびトリガーもすべて除去されます。

Neither CASCADE nor RESTRICT
表が制約、索引、トリガー、またはビューで参照されていても、表は除去されることを指定します。 その表を参照する索引やビューは、すべて除去されます。表が関数、パッケージ、プロシージャー、プログラム、またはトリガーで参照されている場合、その表を参照するアクセス・プランは、アクセス・プランが次回に使用されるときに、暗黙的に再準備されます。そのときにその表が存在しないと、エラーが戻されます。

CASCADE
表が制約、索引、トリガー、またはビューで参照されていても、表は除去されることを指定します。 その表を参照する制約、索引、トリガー、およびビューは、すべて除去されます。表が関数、パッケージ、プロシージャー、またはプログラムで参照されている場合、その表を参照するアクセス・プランは、アクセス・プランが次回に使用されるときに、暗黙的に再準備されます。 そのときにその表が存在しないと、エラーが戻されます。

RESTRICT
表が制約、索引、トリガー、またはビューで参照されている場合、表は除去できないことを指定します。表が関数、パッケージ、プロシージャー、またはプログラムで参照されている場合、その表を参照するアクセス・プランは、アクセス・プランが次回に使用されるときに、暗黙的に再準備されます。 そのときにその表が存在しないと、エラーが戻されます。

TRIGGER *トリガー名*
除去するトリガーを識別します。*トリガー名* は、現行サーバーに存在しているトリガーを識別していなければなりません。指定したトリガーは、スキーマから削除されます。トリガーが SQL トリガーの場合、そのトリガーに関連したプログラム・オブジェクトもスキーマから削除されます。

DISTINCT TYPE *特殊タイプ名*
除去する特殊タイプを指定します。*特殊タイプ名*は、現行サーバーに存在する特殊タイプを示すものでなければなりません。指定したタイプは、スキーマから削除されます。

Neither CASCADE nor RESTRICT
制約、索引、表、およびビューがタイプを参照している場合、そのタイプは除去できないことを指定します。

次の DROP ステートメントは、除去されるタイプのパラメーターや戻り値を持つ、または除去されるタイプを参照する、すべてのプロシージャーまたは関数 R で、有効に実行されます。

```
DROP ROUTINE R
```

次の DROP ステートメントは、除去されるタイプを参照するすべてのトリガー T で、有効に実行されます。

```
DROP TRIGGER T
```

このステートメントをカスケードにして、従属する関数やプロシージャーを除去することも可能です。これらの関数やプロシージャーのすべてが、特殊タイプと従属関係にあるために除去されるリストに入っている場合、特殊タイプの除去は正常に行われます。タイプがパッケージまたはプログラムで参照されている場合、そのタイプを参照するアクセス・プランは、アクセス・プランが次回に使用されるときに、暗黙的に再準備されます。 そのときにそのタイプが存在しないと、エラーが戻されます。

CASCADE
タイプが制約、関数、索引、プロシージャー、表、トリガー、またはビューで参照されていても、タイプは除去されることを指定します。そのタイプを参照する制約、関数、索引、プロシージャー、表、トリガー、およびビューは、すべて除去されます。タイプがパッケージまたはプログラムで参照されている場合、そのタイプを参照するアクセス・プランは、アクセス・プランが次回に使用されるときに、暗黙的に再準備されます。 そのときにそのタイプが存在しないと、エラーが戻されます。

RESTRICT
タイプが制約、関数 (そのタイプの作成時に作成された関数以外の)、索引、プロシージャー、表、トリガー、またはビューで参照されている場合、そのタイプは除去できないことを指定します。 タイプがパッケージまたはプログラムで参照されている場合、そのタイプを参照するアクセス・プランは、アクセス・プランが次回に使用されるときに、暗黙的に再準備されます。 そのときにそのタイプが存在しないと、エラーが戻されます。

VIEW *ビュー名*
除去するビューを識別します。この *ビュー名* は、現行サーバーに存在しているビューを識別していなければなりませんが、カタログ・ビューを識別するものであってはなりません。指定したビューは、スキーマから削除されます。ビューが削除されると、そのビューに関する特権はすべて削除されます。

Neither CASCADE nor RESTRICT
トリガーまたは別のビューで参照されていても、ビューは除去されることを指定します。 そのビューを参照するビューはすべて除去されます。ビューが関数、パッケージ、プロシージャー、プログラム、またはトリガーで参照されている場合、そのビューを参照するアクセス・プランは、アクセス・プランが次回に使用されるときに、暗黙的に再準備されます。そのときにそのビューが存在しないと、エラーが戻されます。

CASCADE
トリガーまたは別のビューで参照されていても、ビューは除去されることを指定します。 そのビューを参照するトリガーおよびビューはすべて除去されます。ビューが関数、パッケージ、プロシージャー、またはプログラムで参照されている場合、そのビューを参照するアクセス・プランは、アクセス・プランが次回に使用されるときに、暗黙的に再準備されます。そのときにそのビューが存在しないと、エラーが戻されます。

RESTRICT
トリガーまたは別のビューで参照されている場合、ビューは除去できないことを指定します。ビューが関数、パッケージ、プロシージャー、またはプログラムで参照されている場合、そのビューを参照するアクセス・プランは、アクセス・プランが次回に使用されるときに、暗黙的に再準備されます。そのときにそのビューが存在しないと、エラーが戻されます。

使用上の注意

同義のキーワード： 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。 これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- キーワード SYNONYM は、ALIAS の同義語として使用することができます。
- キーワード DATA を DISTINCT の同義語として使用することができます。
- PACKAGE の同義語として、キーワード PROGRAM を使用することができます。
- キーワード COLLECTION を SCHEMA の同義語として使用できます。

例

例 1
MY_IN_TRAY という名前の表を削除します。この表に関して作成されているビューまたは索引がある場合は、この除去はできません。

```
DROP TABLE MY_IN_TRAY RESTRICT
```

例 2
MA_PROJ という名前のビューを削除します。

```
DROP VIEW MA_PROJ
```

例 3
PERS.PACKA という名前のパッケージを削除します。

```
DROP PACKAGE PERS.PACKA
```

例 4
特殊タイプ DOCUMENT が現在使用されていなければ、それを除去します。

```
DROP DISTINCT TYPE DOCUMENT RESTRICT
```

例 5
自分が SMITH であり、ATOMIC_WEIGHT が、スキーマ CHEM 内でその名前を持つ唯一の関数であると想定します。ATOMIC_WEIGHT を除去します。

```
DROP FUNCTION CHEM.ATOMIC_WEIGHT RESTRICT
```

例 6

自分が SMITH であり、スキーマ SMITH の中に関数 CENTER を作成したと想定します。除去する関数インスタンスを識別するために関数シグニチャーを使用して、CENTER を除去します。

```
DROP FUNCTION CENTER (INTEGER, FLOAT) RESTRICT
```

例 7

自分が SMITH であり、CENTER という名前の別の関数を作成したと想定し、さらに、スキーマ JOHNSON の中で、その関数に特定名 FOCUS97 を付けたとします。除去する関数インスタンスを識別するために特定名を使用して、CENTER を除去します。

```
DROP SPECIFIC FUNCTION JOHNSON.FOCUS97
```

例 8

自分が SMITH であり、ストアード・プロシージャー OSMOSIS がスキーマ BIOLOGY 内にあると想定します。OSMOSIS を除去します。

```
DROP PROCEDURE BIOLOGY.OSMOSIS
```

例 9

自分が SMITH であり、トリガー BONUS がスキーマ内にあると想定します。BONUS を除去します。

```
DROP TRIGGER BONUS
```

END DECLARE SECTION

END DECLARE SECTION ステートメントは、SQL 宣言セクションの終わりを示します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込んで使用します。それ以外の使用法はありません。このステートメントは、実行可能ステートメントではありません。 RPG、Java、または REXX では指定できません。

権限

権限は不要です。

構文

説明

END DECLARE SECTION ステートメントは、ホスト言語の規則に従って宣言を置ける場所であれば、アプリケーション・プログラム内のどこにでもコーディングできます。このステートメントは、SQL 宣言セクションの終わりを示すのに使用されます。SQL 宣言セクションは、BEGIN DECLARE SECTION ステートメントで開始します。BEGIN DECLARE SECTION ステートメントの詳細については、402 ページの『BEGIN DECLARE SECTION』を参照してください。

BEGIN DECLARE SECTION と END DECLARE SECTION ステートメントは、対にして使用しなければなりません。また、これらのステートメントをネストすることはできません。

DECLARE VARIABLE および INCLUDE ステートメント以外の SQL ステートメントは、宣言セクションの中に入れてはなりません。

プログラムに SQL 宣言セクションの指定がある場合は、その SQL 宣言セクションで宣言されている変数だけがホスト変数として使用できます。SQL 宣言セクションの指定がないときは、プログラムの中の変数はすべてホスト変数として使用できます。

RPG および REXX 以外のホスト言語では、SQL 宣言セクションを指定して、ソース・プログラムが IBM SQL 規格に適合するようにする必要があります。SQL 宣言セクションは、変数に対する最初の参照よりも前にコーディングされている必要があります。RPG の場合には、このようなステートメントを使用しなくてもホスト変数の宣言が行われ、また REXX では、ホスト変数の宣言は、全く行われません。

SQL 宣言セクションの外側で宣言されている変数の名前は、SQL 宣言セクション内で宣言されている変数と同じ名前であってはなりません。

複数の SQL 宣言セクションを、プログラムに指定することができます。

END DECLARE SECTION

例

END DECLARE SECTION ステートメントの使用例については、 402 ページの『BEGIN DECLARE SECTION』を参照してください。

EXECUTE

EXECUTE ステートメントは、準備済み SQL ステートメントを実行します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込んで使用します。それ以外の使用法はありません。このステートメントは、動的には準備できない実行可能ステートメントです。

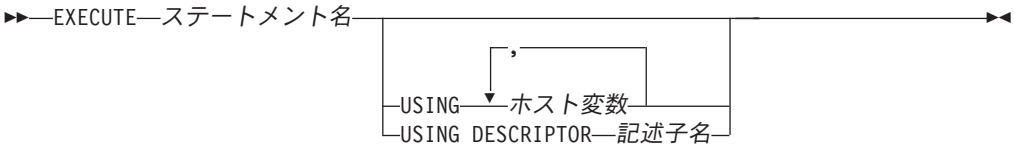
権限

権限の規則は、EXECUTE によって指定される SQL ステートメントに対して定義されている規則が適用されます。例えば、

EXECUTE を使用して INSERT ステートメントを実行する場合に適用される権限規則については、INSERT の説明を参照してください。

プログラムの作成時点の CRTSQLxxx コマンドに DYNUSRPRF(*OWNER) が指定されていた場合を除き、ステートメントの権限 ID は、実行時の権限 ID です。詳細は、60 ページの 『権限 ID と権限名』を参照してください。

構文



説明

ステートメント名

実行する準備済みステートメントを識別します。ステートメント名は、それ以前に準備したステートメントを識別していなければなりません。準備済みステートメントには、SELECT ステートメントを指定してはなりません。

USING

この次に、ホスト変数のリストを指定することを示します。準備済みステートメント内のパラメーター・マーカー (疑問符) は、このキーワードの次に指定したホスト変数の値に置き換えられます。パラメーター・マーカーの説明については、725 ページの『PREPARE』を参照してください。準備済みステートメントにパラメーター・マーカーが含まれている場合は、必ず USING 文節を使用してください。ステートメントにパラメーター・マーカーが入っていなければ、USING は無視されます。

ホスト変数,...

1 つまたは複数のホスト構造体、あるいはホスト変数を指定します。それらの構造体や変数は、ホスト構造体および変数の宣言の規則に従ってプログラムで宣言されていなければなりません。ホスト構造体に対する参照は、その個々の変数に対する参照に置き換えられます。リストされた変数の数は、準備済みステートメントのパラメーター・マーカーの数と同じでなければなり

ません。リスト中の *n* 番目の変数は、準備済みステートメントの *n* 番目のパラメーター・マーカーに対応します。

DESCRIPTOR *記述子名*

SQLDA を識別します。この SQLDA には、ホスト変数の有効な記述が入っていなければなりません。

EXECUTE ステートメントの処理に先立って、ユーザーは SQLDA の以下のフィールドをセットしておく必要があります。 (REXX の場合は、規則が異なります。詳細については、DB2 UDB for iSeries ホスト言語での SQL プログラミング を参照してください。)

- SQLN (SQLDA に用意する SQLVAR のオカレンスの数を示します。)
- SQLDABC (SQLDA 用に割り振る記憶域のバイト数を示します。)
- SQLD (ステートメントを処理するときに SQLDA で使用する変数の個数を指示します。)
- SQLVAR の各オカレンス (変数の属性を指示します。)

SQLDA の記憶域は、SQLVAR のオカレンスをすべて収容するのに十分な大きさで割り振らなければなりません。 LOB または特殊タイプが結果の中に存在する場合、各パラメーター・マーカーごとに追加の SQLVAR 項目が必要です。 SQLDA の詳細については、SQLVAR の説明や SQLVAR のオカレンス数の判別方法の説明も含めて、 877 ページの『付録 C. SQL 記述子域 (SQLDA)』を参照してください。

SQLD には、ゼロ以上で SQLN 以下の値をセットしなければなりません。この値は、準備済みステートメント内のパラメーター・マーカーの個数と同じでなければなりません。SQLDA で記述されている *n* 番目の変数が、準備済みステートメントの *n* 番目のパラメーター・マーカーに対応します。

RPG/400 には、ポインターを設定する機能が用意されていないことに注意してください。SQLDA はポインターを使用して適切なホスト変数を見つけるので、ユーザーは、RPG/400 アプリケーションの外側でそのようなポインターを設定する必要があります。

使用上の注意

パラメーター・マーカーの置き換え

準備済みステートメントのパラメーター・マーカーは、実際には、そのステートメントを実行する前に対応するホスト変数によって置き換えられます。パラメーター・マーカーの置き換えは、ホスト変数の値をソースとし、データベース・マネージャー内部の変数をターゲットとする割り当て演算によって処理されます。 タイプ付きパラメーター・マーカーの場合、ターゲット変数の属性は、CAST によって指定されたものになります。タイプ無しパラメーター・マーカーの場合、ターゲット変数の属性は、パラメーター・マーカーのコンテキストによって決まります。パラメーター・マーカーに適用される規則については、 729 ページの表 57 を参照してください。

V が、パラメーター・マーカー P に対応するホスト変数を指すものとします。V の値は、値を列に割り当てる場合の規則に従って、P のターゲットの変数に割り当てられます。したがって、次のことがいえます。

- V は、ターゲットと互換性のあるものでなければなりません。
- V が数値ならば、V の整数部の絶対値は、ターゲットの整数部の絶対値の最大を超えてはなりません。
- V の属性がターゲットの属性と一致しない場合は、ターゲットの属性に合わせて値が変換されます。
- ターゲットにヌルを入れることができない場合は、V の値はヌルであってはなりません。

ただし、値を列に割り当てる場合の規則とは、以下の点が異なります。

- V がストリングで、その長さがターゲットの長さ属性より大きければ、V の値は途中で切り捨てられます (エラーは出されません)。

準備されたステートメントを実行するときに P の代わりに使用される値は、P のターゲット変数の値です。例えば、V が CHAR(6) で、ターゲットが CHAR(8) の場合、P の代わりに使用される値は、V の値に 2 つのブランクが埋め込まれた値になります。

例

この例は、COBOL プログラムの一部です。パラメーター・マーカーが指定されている INSERT ステートメントが、どのように準備され、実行されるかを示しています。

```
EXEC SQL  BEGIN DECLARE SECTION  END-EXEC.
  77 EMP              PIC X(6).
  77 PRJ              PIC X(6).
  77 ACT              PIC S9(4) COMP-4.
  77 TIM              PIC S9(3)V9(2).
  01 HOLDER.
     49  HOLDER-LENGTH    PIC S9(4) COMP-4.
     49  HOLDER-VALUE     PIC X(80).
EXEC SQL  END DECLARE SECTION  END-EXEC.
.
.
.
MOVE 70 TO HOLDER-LENGTH.
MOVE "INSERT INTO EMPPROJACT (EMPNO, PROJNO, ACTNO, EMPTIME)
-              "VALUES (?, ?, ?, ?)" TO HOLDER-VALUE.
EXEC SQL  PREPARE MYINSERT FROM :HOLDER  END-EXEC.

IF SQLCODE = 0
  PERFORM DO-INSERT THRU END-DO-INSERT
ELSE
  PERFORM ERROR-CONDITION.

DO-INSERT.
  MOVE "000010" TO EMP.
  MOVE "AD3100" TO PRJ.
  MOVE 160      TO ACT.
  MOVE .50      TO TIM.
  EXEC SQL  EXECUTE MYINSERT USING :EMP, :PRJ, :ACT, :TIM  END-EXEC.
END-DO-INSERT.
.
.
.
```

EXECUTE IMMEDIATE

EXECUTE IMMEDIATE ステートメントは、以下の処理を行います。

- 文字ストリング形式の SQL ステートメントをもとにして、そのステートメントの実行可能形式を準備する
- その SQL ステートメントを実行する

EXECUTE IMMEDIATE ステートメントは、PREPARE ステートメントと EXECUTE ステートメントの基本機能を結合したものです。このステートメントは、ホスト変数もパラメーター・マーカーも含まない SQL ステートメントを準備し、実行するのに使用することができます。

呼び出し

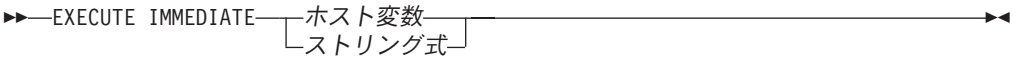
このステートメントは、アプリケーション・プログラムに組み込んで使用します。それ以外の使用法はありません。このステートメントは、動的には準備できない実行可能ステートメントです。

権限

権限の規則は、EXECUTE IMMEDIATE により指定される SQL ステートメントに対する規則が適用されます。例えば、EXECUTE IMMEDIATE を使用して INSERT ステートメントを実行する場合に適用される権限規則については、705 ページの『INSERT』を参照してください。

プログラムの作成時点の CRTSQLxxx コマンドに DYNUSRPRF(*OWNER) が指定されていた場合を除き、ステートメントの権限 ID は、実行時の権限 ID です。詳細は、60 ページの 『権限 ID と権限名』を参照してください。

構文



説明

ホスト変数
ホスト変数を指定します。この変数は、文字ストリングまたは UCS-2 グラフィックのホスト変数を宣言する規則に従って宣言されていなければなりません。ホスト変数は、CLOB または DBCLOB データ・タイプを持っていてはならず、標識変数を指定してはなりません。

ストリング式
ストリング式 は、結果が文字ストリングになる PL/I のストリング式 です。文字ストリングを生み出す SQL 式は許されません。ストリング式 は、PL/I でのみ許されます。

指定されたホスト変数またはストリング式の値は、ステートメント・ストリング と呼ばれます。

ステートメント・ストリングは、以下の SQL ステートメントのいずれかでなければなりません。[55]

ALTER	DROP	REVOKE
CALL	GRANT	ROLLBACK
COMMENT	INSERT	SET PATH
COMMIT	LABEL	SET TRANSACTION
CREATE	LOCK TABLE	UPDATE
DELETE	RENAME	

ステートメント・ストリングは、次のようなストリングであってはなりません。

- EXEC SQL で始まり、END-EXEC またはセミコロン (;) で終わるストリング。
- ホスト変数への参照を含むストリング。
- パラメーター・マーカーを含むストリング。

EXECUTE IMMEDIATE ステートメントを実行すると、指定したステートメント・ストリングが解析され、エラーの有無が検査されます。SQL ステートメントとして正しくない場合は、そのステートメントは実行されず、実行を妨げているエラー条件が SQLCA で報告されます。SQL ステートメントとして正しくても、ステートメントの実行時にエラーが発生すると、そのエラー条件が SQLCA に報告されます。

使用上の注意

同一の SQL ステートメントを何回も実行する場合は、EXECUTE IMMEDIATE ステートメントを使用するよりは、PREPARE および EXECUTE ステートメントを使用した方が効率がよくなります。

例

C を使用して、ホスト変数 Qstring 内の SQL ステートメントを実行します。

```
void main ()
  {

    EXEC SQL  BEGIN DECLARE SECTION  END-EXEC.

    char Qstring[100] = "INSERT INTO WORK_TABLE SELECT * FROM EMPPROJACT
     WHERE ACTNO >= 100";

    EXEC SQL  END DECLARE SECTION  END-EXEC.
    EXEC SQL  INCLUDE SQLCA;
  .
  .
  .
    EXEC SQL EXECUTE IMMEDIATE :Qstring;

    return;
  }
```

55. SELECT ステートメントは使用できません。SELECT ステートメントを動的に処理するには、PREPARE、DECLARE CURSOR、および OPEN ステートメントを使用します。

FETCH

FETCH ステートメントは、カーソルを結果表の行に位置付けます。FETCH ステートメントは、ゼロ、1 つ、または複数の行を戻すこともでき、戻された行の値をホスト変数に割り当てます。

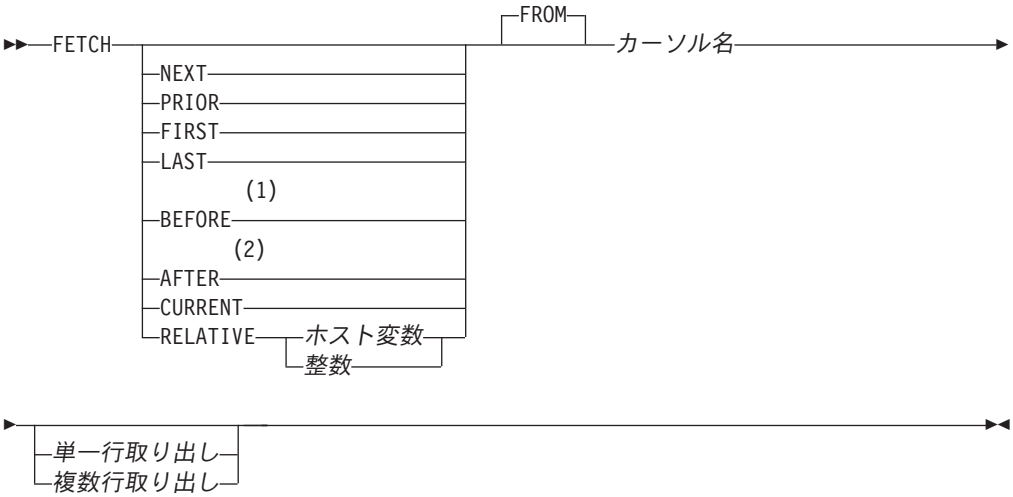
呼び出し

このステートメントは、アプリケーション・プログラムに組み込んで使用します。それ以外の使用法はありません。このステートメントは、動的には準備できない実行可能ステートメントです。複数行の取り出し (FETCH) は、REXX プロシージャーでは許されません。

権限

カーソルを使用する場合に必要な権限についての説明は、597 ページの『DECLARE CURSOR』を参照してください。

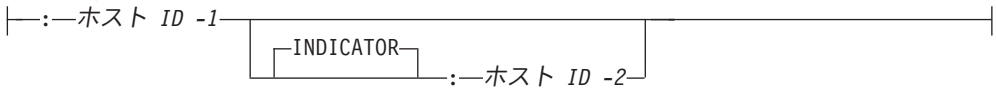
構文



単一行取り出し:

複数行取り出し:

行記憶域:



注:

1 BEFORE が指定されている場合には、単一取り出し文節、または複数行取り出し文節を指定することはできません。

2 AFTER が指定されている場合には、単一取り出し文節、または複数行取り出し文節を指定することはできません。

説明

NEXT、PRIOR、FIRST、LAST、BEFORE、AFTER、CURRENT、および RELATIVE の各キーワードは、カーソルの新しい位置を指定します。これらのキーワードのうち、SCROLL が宣言されていないカーソルに使用できるのは、NEXT のみです。

NEXT
現行カーソル位置から見て、結果表の次の行にカーソルを位置付けます。他のカーソルの方向付けが指定されていない場合には、NEXT がデフォルト値になります。

PRIOR
現行カーソル位置から見て、結果表の前の行にカーソルを位置付けます。

FIRST
結果表の先頭の行にカーソルを位置付けます。

LAST
結果表の最終行にカーソルを位置付けます。

BEFORE
結果表の先頭行の前にカーソルを位置付けます。

AFTER
結果表の最終行の後にカーソルを位置付けます。

CURRENT
カーソルの位置は変えずに、現行カーソル位置を維持します。カーソルが DYNAMIC SCROLL として宣言されている場合、現在行が変更された結果、結果表のソート順序の中での位置が変化していると、エラーが戻されます。

RELATIVE
ホスト変数 または整数 が、整数値 *k* に割り当てられます。RELATIVE によって決められるカーソルの結果表内の行の位置は、*k*>0 の場合は現在行の *k* 行後、*k*<0 の場合は現在行の *k* 行前になります。ホスト変数 を指定する場合には、位取りがゼロの数値変数を指定しなければならず、また標識変数を入れることはできません。

表 50. 同義のスクロール指定

指定	代替
RELATIVE +1	NEXT
RELATIVE -1	PRIOR
RELATIVE 0	CURRENT

FROM
このキーワードは、文脈を分かりやすくするために使用するものです。スクロー

ル位置オプションを指定する場合には、このキーワードが必要です。スクロール・オプションを指定しない場合には、FROM キーワードは任意選択です。

カーソル名
取り出し操作で使用するカーソルを識別します。このカーソル名 は、DECLARE CURSOR ステートメントに関する 598 ページの 『説明』で説明している宣言されたカーソルを識別していなければなりません。FETCH ステートメントの実行時点で、指定したカーソルはオープン状態でなければなりません。

単一行取り出し文節または複数行取り出し文節が指定されていない場合、データはユーザーに戻されません。ただし、カーソルは位置付けられ、行ロックが掛けられます。 ロックの詳細については、24 ページの 『分離レベル』を参照してください。

単一行取り出し

INTO *ホスト変数*,...
1 つまたは複数のホスト構造体またはホスト変数を識別しますが、それらはホスト構造体およびホスト変数の宣言に関する規則に従って宣言されているものでなければなりません。INTO の操作形式では、ホスト構造体は、その個々の変数に対する参照に置き換えられます。結果の行の最初の値がリストの最初のホスト変数に割り当てられ、2 番目の値が 2 番目のホスト変数に割り当てられます。以下同様です。

INTO DESCRIPTOR *記述子名*
ゼロまたは 1 つのホスト変数の有効な記述が入っている SQLDA を識別します。

ユーザーは、FETCH ステートメントを処理する前に、SQLDA の以下のフィールドをセットしておく必要があります。 (REXX の場合は、規則が異なります。詳細については、DB2 UDB for iSeries ホスト言語での SQL プログラミングを参照してください。)

- SQLN (SQLDA に用意する SQLVAR のオカレンスの数を示します。)
- SQLDABC (SQLDA 用に割り振る記憶域のバイト数を示します。)
- SQLD (ステートメントを処理するときに、SQLDA で使用する変数の個数を指示します。)
- SQLVAR の各オカレンス (変数の属性を指示します。)

SQLDA の記憶域は、SQLVAR のオカレンスをすべて収容するのに十分な大きさで割り振らなければなりません。したがって、SQLDABC の値は、16 + SQLN*(80) よりも大きいか、または等しくなければなりません。ここで、80 は SQLVAR の 1 つのオカレンスの長さです。

SQLD には、ゼロ以上で SQLN 以下の値をセットしなければなりません。 詳細は、 877 ページの『付録 C. SQL 記述子域 (SQLDA)』を参照してください。

複数行取り出し

FOR *k* **ROWS**
整数値 *k* に対してホスト変数 または整数 を評価します。ホスト変数 を指定する場合は、位取りがゼロの数値ホスト変数である必要があり、また標識変数を含

めることはできません。*k* は 1 ～ 32767 の範囲でなければなりません。カーソルは、方向付けキーワード (例えば NEXT) の指定する行に置かれ、その行が取り出されます。それから、次の *k*-1 行が取り出され (表内で順方向に移動)、カーソルの終わりに達するまでこれが繰り返されます。取り出し操作の後、カーソルは最後に取り出された行に置かれます。

例えば、FETCH PRIOR FROM C1 FOR 3 ROWS を実行すると、前の行、現在行、および次の行が、この順序で戻されます。 カーソルは次の行に置かれます。FETCH RELATIVE -1 FROM C1 FOR 3 ROWS でも、同じ結果が戻されます。 これに対して FETCH FIRST FROM C1 FOR :x ROWS では、先頭の *x* 行が戻され、カーソルは番号 *x* の行に置かれたままとなります。

複数行取り出しが正しく実行されると、次のように 3 つの変数が SQLCA に設定されます。

- SQLERRD(3) には、取り出された行の数を示す値が設定されます。
- SQLERRD(4) には、取り出された行の長さを示す値が設定されます。
- SQLERRD(5) には、取り出された行が最後の行である場合、+100 が設定されます。[56]

INTO *ホスト構造体配列*

ホスト構造体配列 は、ホスト構造体の宣言に関する規則に従って定義されているホスト構造体の配列を識別します。

すなわち、配列の最初の構造は最初の行に対応し、配列の 2 番目の構造は 2 行目に対応しています。以下も同じです。さらに、行の最初の値が構造体内の最初の項目に対応し、行の 2 番目の値が構造体の 2 番目の項目に対応するというように、これも順番に対応しています。取り出す行数は、ホスト構造体配列の次元以下でなければなりません。

USING DESCRIPTOR *記述子名*

SQLDA を識別しますが、行記憶域の中の行の形式を記述するゼロまたはそれ以上のホスト変数の有効な記述が入っているものでなければなりません。

ユーザーは、FETCH ステートメントを処理する前に、SQLDA の以下のフィールドをセットしなければなりません。

- SQLN (SQLDA に用意する SQLVAR のオカレンスの数を示します。)
- SQLDABC (SQLDA 用に割り振る記憶域のバイト数を示します。)
- SQLD (ステートメントを処理するときに SQLDA で使用する変数の個数を指示します。)
- SQLVAR オカレンス (ホスト変数の属性を指定します。)

SQLDA の他のフィールド (SQLNAME など) の値は、FETCH ステートメントを実行した後に定義することはできず、また、使用するべきではありません。

SQLDA の記憶域は、SQLVAR のオカレンスをすべて収容するのに十分な大きさで割り振らなければなりません。したがって、SQLDABC の値は、16 + SQLN*(80) よりも大きいか、または等しくなければなりません。ここで、80 は SQLVAR の 1 つのオカレンスの長さです。 LOB または特殊タイプが指定され

56. 戻された行の数が要求した行の数に等しい場合は、行終わりの警告は発生せず、SQLERRD(5) は +100 に設定されません。

た場合には、各パラメーター・マーカーごとに 2 つの SQLVAR 項目が必要であり、SQLN はパラメーター・マーカー数の 2 倍にセットしなければなりません。

SQLD には、ゼロ以上で SQLN 以下の値をセットしなければなりません。 詳細は、 877 ページの『付録 C. SQL 記述子域 (SQLDA)』を参照してください。

FETCH が完了すると、最初の SQLVAR 項目の SQLDATA ポインターは、最初の行の割り振り済み記憶域内の最初の列に対して戻された値をアドレス指定し、2 番目の SQLVAR 項目の SQLDATA ポインターは、最初の行の割り振り済み記憶域内の 2 番目の列に対して戻された値を、というように順にアドレス指定します。ヌル可能な最初の SQLVAR 項目の SQLIND ポインターは最初の標識値をアドレス指定し、2 番目のヌル可能な SQLVAR 項目の SQLIND ポインターは 2 番目の標識値を、というように順にアドレス指定します。SQLDA は、16 バイト境界上に割り振らなければなりません。

INTO *行記憶域*

ホスト変数を使用して指定されたホスト *ID -1* は、行を戻す記憶域の割り振りを指定します。行は、SQLDA によって記述された形式でその記憶域に戻されます。ホスト *ID -1* は、要求された行をすべて保持できるだけの十分な大きさが必要です。

ホスト *ID -2* は、任意選択の標識区域を識別します。SQLVAR オカレンスのいずれかの SQLTYPE がヌル可能な場合は、必ずこれを指定してください。この標識は、短整数として戻されます。ホスト *ID -2* は、戻される各行についてヌル可能な各値ごとに標識を入れられるだけの十分な大きさが必要です。

INTO 文節によって識別されているか、または SQLDA で記述されている *n 番目* のホスト変数は、カーソルの結果表の *n 番目* の列に対応します。各ホスト変数のデータ・タイプは、それぞれに対応する列と互換性がなければなりません。

変数への割り当ては、それぞれ 41 ページの『第 2 章 言語エレメント』で説明されている規則に従って行われます。変数の数が行の中の値の数より少ない場合、SQLCA の SQLWARN3 フィールドに 'W' がセットされます。結果の列の数よりも変数の数が多い場合には、警告は出されない点に注意してください。値がヌルの場合は、標識変数が用意されている必要があります。割り当てでエラーが起こった場合、その値は変数に割り当てられず、それ以後の変数への値の割り当ては行われません。ただし、変数にすでに割り当てられている値があれば、その値は割り当てられたままです。

外側の SELECT ステートメントの SELECT リストの算術式の結果としてエラーが起こった場合 (ゼロによる除算やオーバーフローなど)、または文字変換エラーが起こった場合には、結果はヌル値になります。他のヌル値の場合と同様に、標識変数を用意しなければなりません。該当のホスト変数の値は、未定義になります。ただし、この場合、標識変数は -2 にセットされます。ステートメントの処理は、エラーが発生しなかった場合と同様に継続されます。 (ただし、このエラーで正の SQLCODE になります。) 標識変数を用意していない場合は、SQLCA のフィールド SQLCODE に負の値が戻されます。エラーが生じた時点で、すでにいくつかの値がホスト変数に割り当てられていることがあり、それらの値は割り当てられたままになります。

結果列に LOB が含まれている場合、または現行接続がリモート・サーバーへの接続の場合、複数行取り出しは許されません。

使用上の注意

オープン状態のカーソルの位置として、次の 3 つの位置が考えられます。

- 行の前
- 行
- 最終行の後

カーソルがある行に位置付けられている場合、その行をカーソルの現在行と呼びます。UPDATE または DELETE ステートメントで参照するカーソルは、行に位置付けられていなければなりません。カーソルは、FETCH ステートメントの結果としてのみ、行に位置付けることができます。

エラーの発生によって、カーソルの状態が予期できないものになることがあります。

ホスト変数として文字変数を指定し、その変数が、結果を収容するのに十分な大きさを持っていない場合には、SQLCA の SQLWARN1 に 'W' が割り当てられます。標識変数が用意されている場合、結果の実際の長さは、そのホスト変数に関連する標識変数に戻されます。

ホスト変数として C の NUL で終了するホスト変数を指定し、その変数が、結果および NUL 終了文字を入れられるだけの十分な大きさをもっていない場合は、以下のようになります。

- CRTSQLCI コマンドまたは CRTSQLCPPI コマンドに *CNULRQD オプションを指定した場合 (または SET OPTION ステートメントに CNULRQD(*YES) を指定した場合)、以下のようになります。
  - 結果が切り捨てられます。
  - 最後の文字は NUL 終了文字になります。
  - SQLCA の SQLWARN1 に ‘W’ が割り当てられます。
- CRTSQLCI コマンドまたは CRTSQLCPPI コマンドに *NOCNULRQD オプションを指定した場合 (または SET OPTION ステートメントに CNULRQD(*NO) を指定した場合)、以下のようになります。
  - NUL 終了文字は戻されません。
  - SQLCA の SQLWARN1 に ‘N’ が割り当てられます。

同義のキーワード : 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- USING DESCRIPTOR は、単一取り出し文節で INTO DESCRIPTOR の同義語として使用することができます。

例

2 つの表 (FORUM と ARCHIVE) があり、それぞれの表には下記の列があります。

名前	FORUM	RECEIVED	SOURCE	TOPIC	ENTRY_TEXT
タイプ:	CHAR(8) NOT NULL	タイム・スタンプ NOT NULL	CHAR(8) NOT NULL	CHAR(64) NOT NULL	VARCHAR(4000) NOT NULL
説明:	フォーラムの名前	項目を受け取った日時	項目を追加するユーザーのユーザー ID	フォーラム内のトピック	この項目表に追加するテキスト

表 FORUM には、名前を持つ多くのフォーラムが入っています。各フォーラムには、1 つまたは複数のトピックが含まれており、さらに各トピックには、1 つまたは複数の項目が含まれています。トピックが古くなると、そのトピックの項目は削除されるか、または表 ARCHIVE に移されます。

以下の PL/I プログラムは、表 FORUM を保守するのに使用するプログラムです。ユーザーは、このプログラムを呼び出すときに、3 つのコマンドのいずれか 1 つを使用します。各コマンドを出すときには、テキストのストリングを一緒に指定します。このストリングは、所定のトピックに関する列 TOPIC 内に入っている値 (TOPIC の値全体である必要はありません) です。コマンドは、以下の 3 つです。

- 1 (該当するトピックの項目すべてについて TOPIC の値の内容を変更します)
- 2 (該当するトピックに関する項目をすべて表 ARCHIVE に移します)
- 3 (該当するトピックに関する項目を保管せずに すべて削除します)

```
CLEANUP:  PROC OPTIONS(MAIN);
  DCL NOT_END BIT(1);
EXEC SQL  BEGIN DECLARE SECTION;
  DCL  ACTION       BINARY FIXED(15);   /* 1=chg-topic   2=archive   3=delete */
  DCL  SRCH_FORUM   CHAR(8);
  DCL  SRCH_TOPIC   CHAR(66) VARYING;
  DCL  NEW_TOPIC    CHAR(64) VARYING;
  DCL  FORUM        CHAR(8);
  DCL  TSTMP        CHAR(26);
  DCL  PERSON       CHAR(8);
  DCL  TOPIC        CHAR(64) VARYING;
  DCL  TXT          CHAR(2000) VARYING;
EXEC SQL  END DECLARE SECTION;
EXEC SQL  INCLUDE SQLCA;
EXEC SQL  WHENEVER NOT FOUND CONTINUE;
EXEC SQL  WHENEVER SQLWARNING CONTINUE;
EXEC SQL  WHENEVER SQLERROR GOTO ERRCHK;

EXEC SQL  CONNECT TO TOROLAB3;
GET LIST (ACTION, SRCH_FORUM, SRCH_TOPIC, NEW_TOPIC);
SRCH_TOPIC = '%' || SRCH_TOPIC || '%';
EXEC SQL  DECLARE CUR CURSOR FOR
             SELECT * FROM FORUM
                 WHERE FORUM = :SRCH_FORUM AND TOPIC LIKE :SRCH_TOPIC
                 FOR UPDATE OF TOPIC;
EXEC SQL  OPEN CUR;

NOT_END = '1'B;
DO WHILE (NOT_END);
  EXEC SQL  FETCH CUR INTO :FORUM, :TSTMP, :PERSON, :TOPIC, :TXT;
  IF SQLSTATE = '02000' THEN
    NOT_END = '0'B;
  ELSE DO;
    SELECT;
      WHEN (ACTION = 1)               /* change topic value */
        EXEC SQL  UPDATE FORUM
                    SET TOPIC = :NEW_TOPIC
                    WHERE CURRENT OF CUR;
      WHEN (ACTION = 2)               /* archive entry to another table */
        DO;
          EXEC SQL  INSERT INTO ARCHIVE
                      VALUES (:FORUM, :TSTMP, :PERSON, :TOPIC, :TXT);
          EXEC SQL  DELETE FROM FORUM WHERE CURRENT OF CUR;
        END;
      WHEN (ACTION = 3)               /* delete topic */
        EXEC SQL  DELETE FROM FORUM WHERE CURRENT OF CUR;
    END; /* select */
  END;  /* else do */
END;  /* do while */

FINISHED:
  EXEC SQL  CLOSE CUR;
  EXEC SQL  COMMIT WORK;
  RETURN;
ERRCHK:
  DISPLAY ('Unexpected Error -changes will be backed out');
  PUT SKIP LIST (SQLCA);
  EXEC SQL  WHENEVER SQLERROR CONTINUE;  /* continue if error on rollback */
  EXEC SQL  ROLLBACK WORK;
  RETURN;
END;  /* CLEANUP */
```

FREE LOCATOR

FREE LOCATOR ステートメントは、ロケーター変数とその値の間の関連を除去します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込んで使用します。それ以外の使用法はありません。これを対話式に発行することはできません。このステートメントは、動的に準備できる実行可能ステートメントです。ただし、準備済みステートメントを実行するには、USING 文節を指定した EXECUTE ステートメントを使用しなければなりません。FREE LOCATOR は、EXECUTE IMMEDIATE ステートメントと併用することはできません。

権限

権限は不要です。

構文

説明

ホスト変数、...

1 つまたは複数のホスト変数を指定します。これらのホスト変数は、ロケーター変数の宣言の規則に従って宣言する必要があります。この変数に、標識変数を指定してはなりません。ロケーター変数のタイプは、バイナリー・ラージ・オブジェクト・ロケーター、文字ラージ・オブジェクト・ロケーター、2 バイト文字ラージ・オブジェクト・ロケーターのいずれかでなければなりません。

このホスト変数には、現在ロケーターが割り当てられている必要があります。つまり、この作業単位中に (CALL、FETCH、SELECT INTO、SET 変数、または VALUES INTO ステートメントによって) ロケーターが割り当てられていなければならず、それ以降そのロケーターが (FREE LOCATOR ステートメントによって) 解放されていてはならない、ということです。そうでない場合には、エラーが発生します。

FREE LOCATOR ステートメントに複数のホスト変数が指定されていて、ロケーターの 1 つでエラーが発生した場合、どのロケーターも解放されることはありません。

例

従業員表に列 RESUME、HISTORY、および PICTURE が含まれていて、それらの列値を表すためにロケーターが確立されていると想定します。COBOL プログラムでは、CLOB ロケーター変数 LOCRES と LOCHIST、および BLOB ロケーター変数 LOCPIC を解放します。

```
EXEC SQL FREE LOCATOR :LOCRES, :LOCHIST, :LOCPIC END-EXEC.
```

GRANT (特殊タイプ特権)

この形式の GRANT ステートメントは、特殊タイプ に対する特権を認可します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことも、対話式に呼び出すこともできます。このステートメントは、動的に準備できる実行可能ステートメントです。

権限

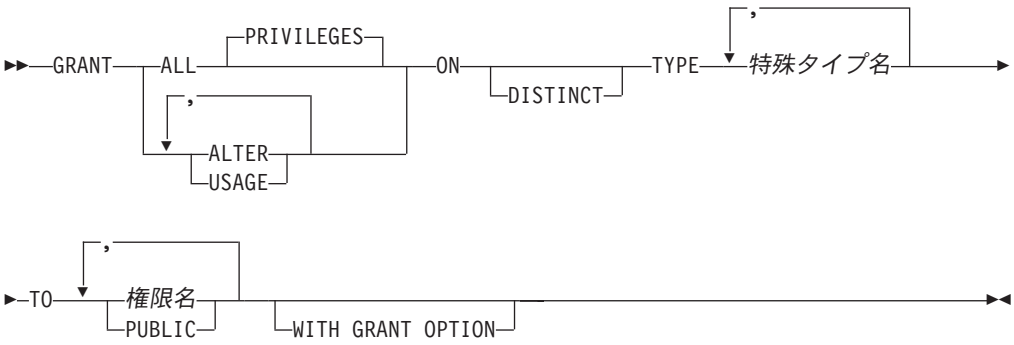
このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメント内に識別されているそれぞれの 特殊タイプ に対しては次のもの。
  - このステートメントで指定されるすべての特権
  - その 特殊タイプ に対する *OBJMGT システム権限。
  - その 特殊タイプ が入っているライブラリーに対する *EXECUTE システム権限
- 管理権限

WITH GRANT OPTION を指定する場合、ステートメントの権限 ID によって保持される特権には、少なくとも次の 1 つが含まれていなければなりません。

- 特殊タイプ の所有権
- 管理権限

構文



説明

ALL または **ALL PRIVILEGES**

1 つまたは複数の特権を認可します。認可される特権は、指定された 特殊タイプに対してステートメントの権限 ID がもっている認可可能な特権のすべてで

す。 特殊タイプに対する ALL PRIVILEGES を認可することは、*ALL システム権限を認可するのと同じではないことに注意する必要があります。

ALL を使用しない場合には、以下にリストしたキーワードの 1 つまたは複数を使用する必要があります。各キーワードは、そこで説明している特権を認可します。

ALTER
COMMENT ステートメントを使用するための特権を認可します。

USAGE
表、関数、またはプロシージャーの中で特殊タイプを使用する特権を認可します。

ON DISTINCT TYPE *特殊タイプ名*
特権を認可する特殊タイプを指定します。*特殊タイプ名*は、現行サーバーに存在する特殊タイプを示すものでなければなりません。

TO
特権を認可するユーザーを指定します。

権限名、...
1 つまたは複数の権限 ID をリストします。同じ*権限名* は、複数回指定してはなりません。

PUBLIC
ユーザー (権限 ID) の集合に対して特権を認可します。

この集合は、該当の特殊タイプに対して私的に認可された特権をもっていないユーザーで構成されます。 例えば、ALTER が PUBLIC に認可されている場合に、USAGE が HERNANDEZ に認可されると、この私的な認可により、HERNANDEZ は ALTER 特権を持つことができなくなります。

WITH GRANT OPTION
指定した*権限名* が、ON 文節で指定されている特殊タイプに対する特権を他のユーザーに認可できるようにします。

WITH GRANT OPTION の指定がない場合は、指定した*権限名* は、ON 文節で指定されている 特殊タイプ に対する特権を、別のユーザーに認可することができません。ただし、指定した権限名が、他の何らかの方法で認可できる権限を入手した場合 (例えば、*OBJMGT システム権限の認可) を除きます。

使用上の注意

GRANT および REVOKE ステートメントは、SQL オブジェクトに対するシステム権限の割り当ておよび除去を行います。次の表は、SQL 特権に対応するシステム権限を示しています。

表 51. 特殊タイプ に対して認可または取り消しされる特権

SQL の特権	特殊タイプ に対する認可または取り消しに対応するシステム権限
ALL (ALL の認可または取り消しは、ステートメントの権限 ID がもつ特権のみを認可または取り消します。)	*OBJALTER *OBJOPR *EXECUTE *OBJMGT (取り消しのみ)

表 51. 特殊タイプ に対して認可または取り消しされる特権 (続き)

SQL の特権	特殊タイプ に対する認可または取り消しに対応するシステム権限
ALTER	*OBJALTER
USAGE	*EXECUTE *OBJOPR
WITH GRANT OPTION	*OBJMGT

同義のキーワード : 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- キーワード DATA を DISTINCT の同義語として使用することができます。

例

特殊タイプ SHOE_SIZE に関する USAGE 特権をユーザー JONES に認可します。この GRANT ステートメントでは、JONES に、特殊タイプ SHOE_SIZE に関連するキャスト関数を実行する特権は与えません。

```
GRANT USAGE
  ON DISTINCT TYPE SHOE_SIZE
  TO JONES
```

GRANT (関数またはプロシージャー特権)

この形式の GRANT ステートメントは、関数またはプロシージャーに対する特権を認可します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことも、対話式に呼び出すこともできます。このステートメントは、動的に準備できる実行可能ステートメントです。

権限

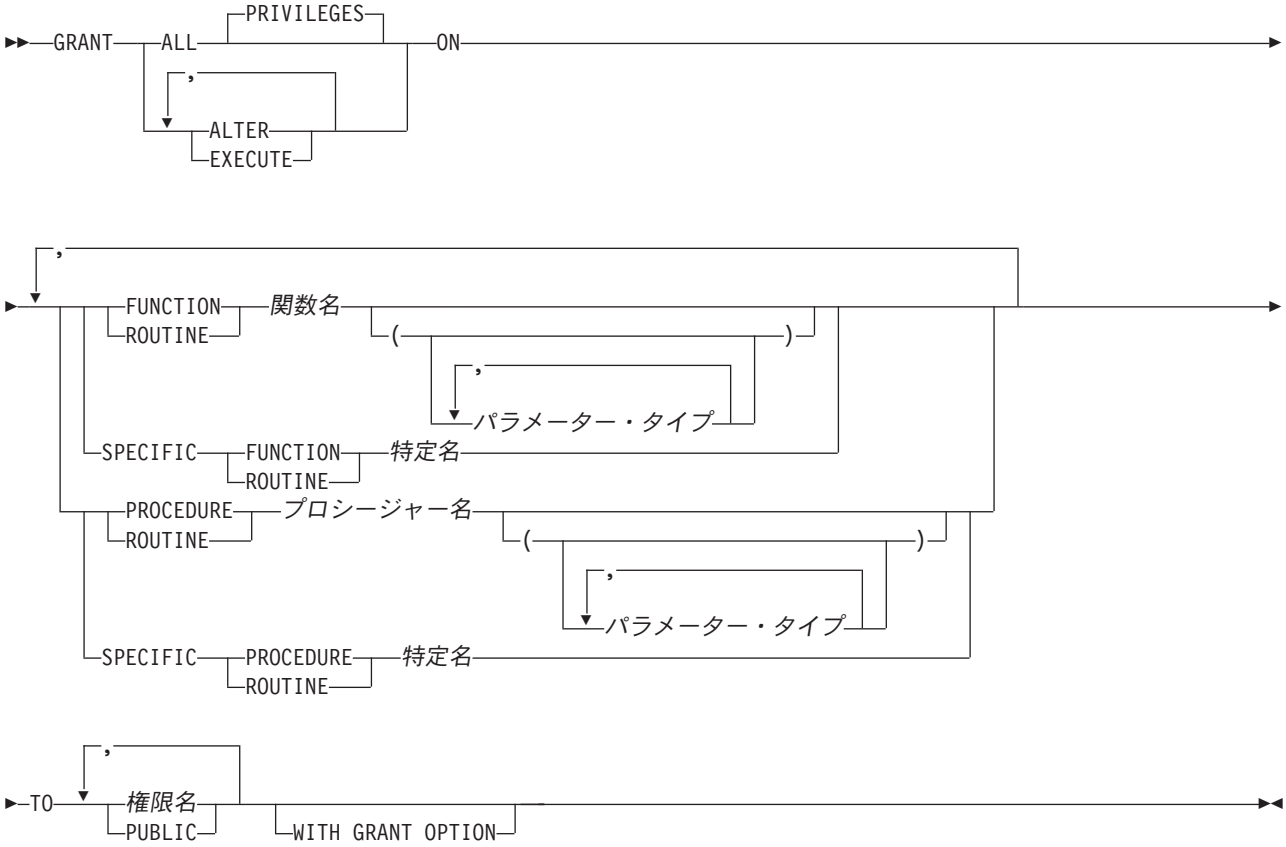
このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメント内で識別された、それぞれの関数またはプロシージャーごとに、
  - このステートメントで指定されるすべての特権
  - その関数またはプロシージャーに対する *OBJMGT システム権限。
  - その関数またはプロシージャーが入っているライブラリー (これが Java ルーチンの場合は、ディレクトリー) に対する *EXECUTE システム権限。
- 管理権限

WITH GRANT OPTION を指定する場合、ステートメントの権限 ID によって保持される特権には、少なくとも次の 1 つが含まれていなければなりません。

- その関数またはプロシージャーの所有権
- 管理権限

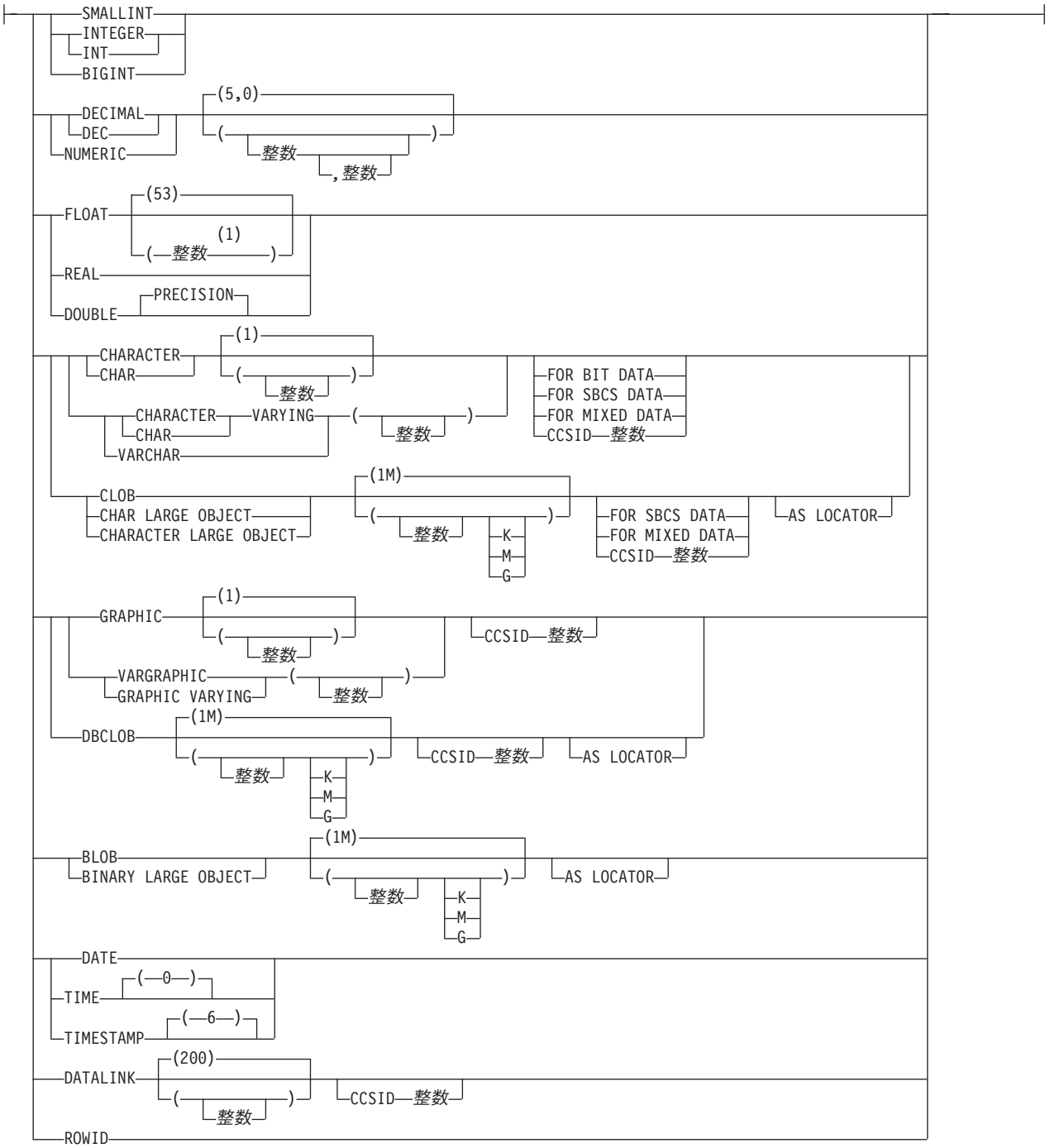
構文



パラメーター・タイプ:

GRANT (関数またはプロシージャー特権)

組み込みタイプ:



注:

1 突き合わせは、データ・タイプ (REAL または DOUBLE) に基づいて行われるので、精度に指定された値は、関数の作成時に指定された値に一致している必要はありません。

説明

ALL または **ALL PRIVILEGES**

1 つまたは複数の特権を認可します。認可される特権は、指定された関数またはプロシージャーに対してステートメントの権限 ID がもっている認可可能な特

権のすべてです。関数またはプロシージャーに対する ALL PRIVILEGES を認可するのは、*ALL システム権限を認可するのと同じではないことに注意する必要があります。

ALL を使用しない場合には、以下にリストしたキーワードの 1 つまたは複数を使用する必要があります。各キーワードは、そこで説明している特権を認可します。

ALTER
COMMENT ステートメントを使用するための特権を認可します。

EXECUTE
関数またはプロシージャーを実行するための特権を認可します。

FUNCTION
特権を認可する対象の関数を識別します。関数は、それぞれその名前、関数シグニチャー、あるいは特定名によって識別することができます。 関数解決の規則 (およびパス) は使用されません。

FUNCTION *関数名*
関数名 では、現行サーバーに存在している厳密に 1 つの関数を識別する必要があります。この関数には、パラメーターをいくつでも定義することができます。指定されたスキーマまたは暗黙のスキーマの中に、指定された名前の関数が複数ある場合、エラーが戻されます。

FUNCTION *関数名 (パラメーター・タイプ, ...)*
関数名 (パラメーター・タイプ, ...) は、現行サーバーに存在していて、指定された関数シグニチャーを持つ関数を識別する必要があります。 指定されたパラメーターは、CREATE FUNCTION ステートメント上の対応する位置に指定されたデータ・タイプと一致していなければなりません。認可する関数インスタンスを識別する場合、データ・タイプの数とデータ・タイプの論理連結が使用されます。*関数名 ()* を指定する場合、識別される関数にパラメーターを使用することはできません。

関数名
関数の名前を識別します。

(パラメーター・タイプ, ...)
関数のパラメーターを識別します。

非修飾の特殊タイプ名を指定する場合、データベース・マネージャーはその特殊タイプのスキーマ名を解決するための SQL パスを検索します。

長さ属性、精度属性、あるいは位取り属性があるデータ・タイプの場合、値を指定することも、1 組の空の括弧を使用することもできます。

- 中が空の括弧は、データ・タイプが一致しているか否かの判別時にデータベース・マネージャーによって属性が無視されることを示します。
- 長さ属性、精度属性、あるいは位取り属性に特定の値を使用する場合、その値は、CREATE FUNCTION ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。

- 長さ属性、精度属性、または位取り属性が明示的に指定されておらず、空の括弧も指定されていない場合、該当のデータ・タイプのデフォルト属性が暗黙指定されます。例えば以下のようになります。

```
CHAR      CHAR(1)
GRAPHIC   GRAPHIC(1)
DECIMAL   DECIMAL(5,0)
FLOAT     DOUBLE (length of 8)
```

暗黙の長さは、CREATE FUNCTION ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。データ・タイプのデフォルト長さの全リストは、542 ページの『CREATE TABLE』 を参照してください。

サブタイプまたは CCSID 属性のあるデータ・タイプの場合は、FOR DATA 文節または CCSID 文節の指定は任意選択です。どちらか一方の文節を省略すると、データ・タイプが一致しているか否かの判別時にデータベース・マネージャーによって属性が無視されることが指示されます。 どちらか一方の文節を指定する場合は、CREATE FUNCTION ステートメントに暗黙的または明示的に指定されている値と一致させる必要があります。

SPECIFIC FUNCTION *特定名*

この *特定名* は、現行サーバーに存在している特定の関数を識別していなければなりません。

PROCEDURE

特権を認可する対象のプロシージャーを識別します。プロシージャーは、それぞれその名前、プロシージャー・シグニチャー、あるいは特定名によって識別することができます。 プロシージャー解決の規則 (およびパス) は使用されません。

PROCEDURE *プロシージャー名*

プロシージャー名 は、現行サーバーに存在しているただ 1 つのプロシージャーを識別していなければなりません。このプロシージャーには、パラメーターをいくつでも定義することができます。指定されたスキーマまたは暗黙のスキーマの中に、指定された名前のプロシージャーが複数ある場合、エラーが戻されます。

PROCEDURE *プロシージャー名 (パラメーター・タイプ, ...)*

プロシージャー名 (パラメーター・タイプ, ...) では、現行サーバーに存在していて、指定されたプロシージャー・シグニチャーを持つプロシージャーを識別する必要があります。指定されたパラメーターは、CREATE PROCEDURE ステートメント上の対応する位置に指定されたデータ・タイプと一致していなければなりません。認可するプロシージャー・インスタンスを識別する場合、データ・タイプの数とデータ・タイプの論理連結が使用されます。*プロシージャー名 ()* を指定する場合、識別されるプロシージャーにパラメーターを使用することはできません。

プロシージャー名

プロシージャーの名前を識別します。

(パラメーター・タイプ, ...)

プロシージャーのパラメーターを識別します。

非修飾の特殊タイプ名を指定する場合、データベース・マネージャーはその特殊タイプのスキーマ名を解決するための SQL パスを検索します。

長さ属性、精度属性、あるいは位取り属性があるデータ・タイプの場合、値を指定することも、1 組の空の括弧を使用することもできます。

- 中が空の括弧は、データ・タイプが一致しているか否かの判別時にデータベース・マネージャー によって属性が無視されることを示します。
- 長さ属性、精度属性、あるいは位取り属性に特定の値を使用する場合、その値は、CREATE PROCEDURE ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。
- 長さ属性、精度属性、または位取り属性が明示的に指定されておらず、空の括弧も指定されていない場合、該当のデータ・タイプのデフォルト属性が暗黙指定されます。例えば以下のようになります。

```
CHAR      CHAR(1)
GRAPHIC   GRAPHIC(1)
DECIMAL   DECIMAL(5,0)
FLOAT     DOUBLE (length of 8)
```

暗黙の長さは、CREATE PROCEDURE ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。データ・タイプのデフォルト長さの全リストは、542 ページの『CREATE TABLE』 を参照してください。

サブタイプまたは CCSID 属性のあるデータ・タイプの場合は、FOR DATA 文節または CCSID 文節の指定は任意選択です。どちらか一方の文節を省略すると、データ・タイプが一致しているか否かの判別時にデータベース・マネージャーによって属性が無視されることが指示されます。 どちらか一方の文節を指定する場合は、CREATE PROCEDURE ステートメントに暗黙的または明示的に指定されている値と一致させる必要があります。

SPECIFIC PROCEDURE *特定名*

特定名 は、現行サーバーに存在している特定のプロシージャーを識別していなければなりません。

TO

特権を認可するユーザーを指定します。

権限名、...

1 つまたは複数の権限 ID をリストします。同じ*権限名* は、複数回指定してはなりません。

PUBLIC

ユーザー (権限 ID) の集合に対して特権を認可します。

この集合は、該当の関数またはプロシージャーに対して私的に認可された特権をもっていないユーザーで構成されます。例えば、ALTER が PUBLIC に認可されてから、EXECUTE が HERNANDEZ に認可されると、この私的な認可によって、HERNANDEZ は ALTER 特権をもてなくなります。

WITH GRANT OPTION

指定した*権限名* が、ON 文節で指定されている関数またはプロシージャーに対する特権を他のユーザーに認可できるようにします。

WITH GRANT OPTION の指定がない場合は、指定した*権限名* は、ON 文節で指定されている関数またはプロシージャーに対する特権を別のユーザーに認可することができません。ただし、指定した権限名が、他の何らかの方法で認可できる権限を入手した場合 (例えば、*OBJMGT システム権限の認可) を除きます。

使用上の注意

SQL または外部関数か外部プロシージャーに認可された特権は、その関連のプログラム (*PGM) オブジェクトまたはサービス・プログラム (*SRVPGM) オブジェクトに認可されます。Java 外部関数またはプロシージャーに認可された特権は、関連のクラス・ファイルまたは jar ファイルに対して認可されます。

GRANT および REVOKE ステートメントは、SQL オブジェクトに対するシステム権限の割り当ておよび除去を行います。次の表は、SQL 特権に対応するシステム権限を示しています。

表 52. 非 Java 関数またはプロシージャーに対して認可や取り消しが行われる特権

SQL の特権	関数またはプロシージャーに対する認可や取り消しに対応するシステム権限
ALL (ALL の認可または取り消しは、ステートメントの権限 ID がもつ特権のみを認可または取り消します。)	*OBJALTER *OBJOPR *EXECUTE *OBJMGT (取り消しのみ)
ALTER	*OBJALTER
EXECUTE	*EXECUTE *OBJOPR
WITH GRANT OPTION	*OBJMGT

表 53. Java 関数またはプロシージャーに対して認可や取り消しが行われる特権

SQL の特権	Java 関数またはプロシージャーに対する認可や取り消しに対応するデータ権限	Java 関数またはプロシージャーに対する認可や取り消しに対応するオブジェクト権限
ALL (ALL の認可または取り消しは、ステートメントの権限 ID がもつ特権のみを認可または取り消します。)	*RWX	*OBJEXIST *OBJALTER *OBJMGT (取り消しのみ)
ALTER	*R	*OBJALTER
EXECUTE	*RX	*EXECUTE
WITH GRANT OPTION	*RWX	*OBJMGT

同義のキーワード : 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- EXECUTE の同義語としてキーワード RUN を使用することもできます。

例

プロシージャー CORPDATA.PROCA に関する EXECUTE 特権を、PUBLIC に対して認可します。

```
GRANT EXECUTE
  ON PROCEDURE CORPDATA.PROCA
  TO PUBLIC
```

GRANT (パッケージ特権)

この形式の GRANT ステートメントは、パッケージに対する特権を認可します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことも、対話式に呼び出すこともできます。このステートメントは、動的に準備できる実行可能ステートメントです。

権限

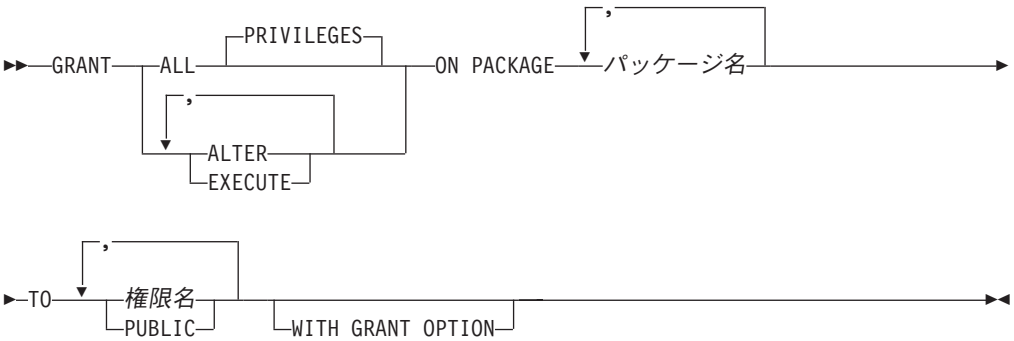
このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメント内で識別された、それぞれのパッケージごとに、
  - このステートメントで指定されるすべての特権
  - パッケージに対する *OBJMGT システム権限
  - パッケージが入っているライブラリーについての *EXECUTE システム権限
- 管理権限

WITH GRANT OPTION を指定する場合、ステートメントの権限 ID によって保持される特権には、少なくとも次の 1 つが含まれていなければなりません。

- そのパッケージの所有権
- 管理権限

構文



説明

ALL または **ALL PRIVILEGES**

1 つまたは複数の特権を認可します。認可される特権は、指定されたパッケージに対してステートメントの権限 ID がもっている認可可能な特権のすべてです。パッケージに対する ALL PRIVILEGES を認可することは、*ALL システム権限を認可するのと同じではないことに注意する必要があります。

ALL を使用しない場合には、以下にリストしたキーワードの 1 つまたは複数を使用する必要があります。各キーワードは、そこで説明している特権を認可します。

ALTER
COMMENT ステートメントを使用するための特権を認可します。

EXECUTE
パッケージのステートメントを実行する特権を認可します。

ON PACKAGE *パッケージ名*
特権を認可する対象のパッケージを識別します。このパッケージ名 は、現行サーバーに存在しているパッケージを識別していなければなりません。

TO
特権を認可するユーザーを指定します。

権限名、...
1 つまたは複数の権限 ID をリストします。同じ*権限名* は、複数回指定してはなりません。

PUBLIC
ユーザー (権限 ID) の集合に対して特権を認可します。

この集合は、該当のパッケージに対して私的に認可された特権をもっていないユーザーで構成されます。例えば、ALTER が PUBLIC に認可されてから、EXECUTE が HERNANDEZ に認可されると、この私的な認可によって、HERNANDEZ は ALTER 特権をもてなくなります。

WITH GRANT OPTION
指定した*権限名* が、ON 文節で指定されているパッケージに対する特権を他のユーザーに認可できるようにします。

WITH GRANT OPTION の指定がない場合は、指定した*権限名* は、ON 文節で指定されているパッケージに対する特権を別のユーザーに認可することができません。ただし、指定した権限名が、他の何らかの方法で認可できる権限を入手した場合 (例えば、*OBJMGT システム権限の認可) を除きます。

使用上の注意

GRANT および REVOKE ステートメントは、SQL オブジェクトに対するシステム権限の割り当ておよび除去を行います。次の表は、SQL 特権に対応するシステム権限を示しています。

表 54. パッケージについて、認可または取り消しの対象となる特権

SQL の特権	パッケージに対する認可または取り消しに対応するシステム権限
ALL (ALL の認可または取り消しは、ステートメントの権限 ID がもつ特権のみを認可または取り消します。)	*OBJALTER *OBJOPR *EXECUTE *OBJMGT (取り消しのみ)
ALTER	*OBJALTER
EXECUTE	*EXECUTE *OBJOPR

表 54. パッケージについて、認可または取り消しの対象となる特権 (続き)

SQL の特権	パッケージに対する認可または取り消しに対応するシステム権限
WITH GRANT OPTION	*OBJMGT

同義のキーワード : 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- EXECUTE の同義語としてキーワード RUN を使用することができます。
- PACKAGE の同義語として、キーワード PROGRAM を使用することができます。

例

パッケージ CORPDATA.PKGA に関する EXECUTE 特権を PUBLIC に対して認可します。

```
GRANT EXECUTE
  ON PACKAGE CORPDATA.PKGA
  TO PUBLIC
```

GRANT (表特権)

この形式の GRANT ステートメントは、表またはビューに対する特権を認可します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことも、対話式に呼び出すこともできます。このステートメントは、動的に準備できる実行可能ステートメントです。

権限

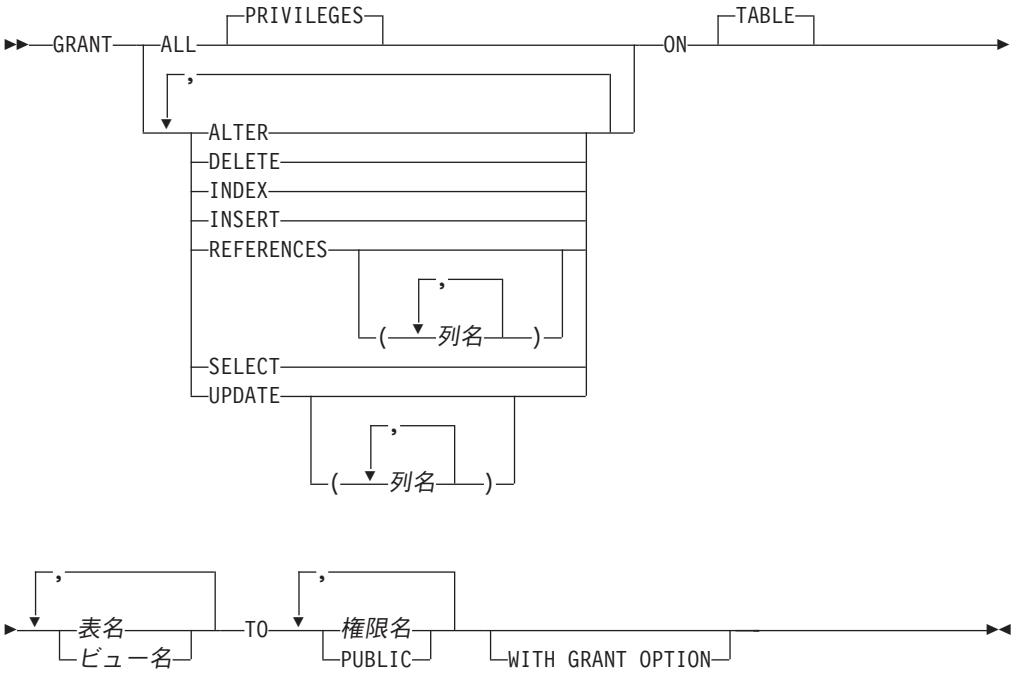
このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメント内で識別された、それぞれの表またはビューごとに、
  - このステートメントで指定されるすべての特権
  - その表またはビューに対する *OBJMGT システム権限
  - 表やビューが入っているライブラリーに対する *EXECUTE システム権限
- 管理権限

WITH GRANT OPTION を指定する場合、ステートメントの権限 ID によって保持される特権には、少なくとも次の 1 つが含まれていなければなりません。

- その表の所有権
- 管理権限

構文



説明

ALL または **ALL PRIVILEGES**
1 つまたは複数の特権を認可します。認可される特権は、指定された表またはビューに対してステートメントの権限 ID がもっている認可可能な特権のすべてです。表またはビューに対する ALL PRIVILEGES を認可することは、*ALL システム権限を認可するのと同じではないことに注意する必要があります。

ALL を使用しない場合には、以下にリストしたキーワードの 1 つまたは複数を使用する必要があります。各キーワードはそこで説明している特権を認可しますが、ON 文節で指定した表やビューに適用されるものだけです。例えば、UPDATE、DELETE、および INSERT 特権は、読み取り専用のビューには適用されません。

ALTER
表に対する ALTER TABLE、CREATE TRIGGER、および DROP TRIGGER ステートメントを使用する特権を認可します。表およびビューに対する COMMENT および LABEL ステートメントを使用する特権を認可します。

DELETE
DELETE ステートメントを使用するための特権を認可します。DELETE は、読み取り専用のビューには認可することはできません。

INDEX
CREATE INDEX ステートメントを使用するための特権を認可します。この特権は、ビューに対して認可することはできません。

INSERT
INSERT ステートメントを使用するための特権を認可します。INSERT は、挿入を認めないビューには認可することはできません。

REFERENCES
表が親である場合に、参照制約を追加する特権を認可します。列のリストが指定されていない場合、または ALL PRIVILEGES を指定することによって REFERENCES が表またはビューのすべての列に対して認可されている場合、被認可者は、親キーとして ON 文節内に指定されている各表のすべての列を使用して、参照制約を追加することができます。これは、ALTER TABLE ステートメントによって後から追加された列も対象となります。この特権は、ビューの場合も認可できますが、ビューに対してはこの特権は使用されません。

REFERENCES (*列名***,...)**
親キーとして列リストに指定されている列のみを使用して、参照制約を追加する特権を認可します。それぞれの *列名* は、ON 文節に指定されている各表の列を識別する非修飾の名前でなければなりません。この特権は、ビューの列の場合も認可できますが、ビューについてはこの特権は使用されません。

SELECT
SELECT または CREATE VIEW ステートメントを使用するための特権を認可します。

UPDATE
UPDATE ステートメントを使用するための特権を認可します。列のリストが指定されていない場合、または ALL PRIVILEGES を指定することによって、UPDATE が表またはビューのすべての列に対して認可されている場合、被認可

者は、ON 文節に指定されている各表のすべての更新可能列を更新することができます。これは ALTER TABLE ステートメントによって後から追加された列も対象となります。UPDATE は、更新が許されないビューには認可できません。

UPDATE (*列名*,...)
列リストに示されている列のみを更新するために、UPDATE ステートメントを使用する特権を認可します。それぞれの*列名*は、ON 文節に指定されている各表およびビューの列を識別する非修飾の名前でなければなりません。UPDATE は、更新が許されないビューの列には認可できません。

ON *表名* または *ビュー名* ,...
特権を認可したい表またはビューを識別します。表名またはビュー名は、現行サーバーにある表またはビューを示すものでなければなりませんが、グローバル一時表を示すものであってはなりません。

TO
特権を認可するユーザーを指定します。

権限名、...
1 つまたは複数の権限 ID をリストします。同じ*権限名* は、複数回指定してはなりません。

PUBLIC
ユーザー (権限 ID) の集合に対して特権を認可します。

この集合は、該当の表またはビューに対して私的に認可された特権をもっていないユーザーで構成されます。例えば、SELECT が PUBLIC に認可されている場合に、UPDATE が HERNANDEZ に認可されると、この私的な認可により、HERNANDEZ は SELECT 特権を持つのを妨げられます。

WITH GRANT OPTION
指定した*権限名* が、ON 文節で指定されている表およびビューに対する特権を他のユーザーに認可できるようにします。

WITH GRANT OPTION の指定がない場合は、指定した*権限名* は、ON 文節で指定されている表およびビューに対する特権を別のユーザーに認可することができません。ただし、指定した権限名が、他の何らかの方法で認可できる権限を入手した場合 (例えば、*OBJMGT システム権限の認可) を除きます。

使用上の注意

GRANT および REVOKE ステートメントは、SQL オブジェクトに対するシステム権限の割り当ておよび除去を行います。次の表は、表に対して認可される SQL 特権に対応するシステム権限を示しています。左側の欄は、SQL 特権をリストしています。右側の欄は、認可または取り消しされる同等のシステム権限をリストしています。

表 55. 表に対して認可または取り消しされる特権

SQL の特権	表に対する認可または取り消しに対応するシステム権限
ALL (ALL の GRANT または取り消しは、ステートメントの権限 ID がもつ特権のみを認可または取り消します。)	*OBJALTER [57] *OBJMGT (取り消しのみ) *OBJOPR *OBJREF *ADD *DLT *READ*UPD
ALTER	*OBJALTER[58]
DELETE	*OBJOPR *DLT
INDEX	*OBJALTER[58]
INSERT	*OBJOPR *ADD
REFERENCES	*OBJREF[58]
SELECT	*OBJOPR *READ
UPDATE	*OBJOPR *UPD
WITH GRANT OPTION	*OBJMGT

次の表は、ビューに対して認可される SQL 特権に対応するシステム権限を示しています。左側の欄は、SQL 特権をリストしています。中央の欄は、ビュー自体に対して認可または取り消しされる同等のシステム権限をリストしています。右側の欄にリストされているシステム権限は、ビューの定義内で参照さているすべての表およびビューに対して認可され、ビューが参照されている場合は、その定義で参照されているすべての表およびビューのすべてに対して認可される、というようになります。[59]

ビューが複数の表やビューを参照している場合、*DLT、*ADD、および *UPD システム権限は、そのビューの定義の副選択の最初の表、またはビューについてのみ認可されます。*READ システム権限は、そのビューの定義で参照されているすべての表およびビューに関して認可されます。

ある SQL 特権に対応して、複数のシステム権限が認可される場合に、それらの権限のいずれか 1 つを認可することができないと、警告が出され、その特権に対応する権限はいずれも認可されません。GRANT とは異なり、REVOKE はビューに関するシステム権限を取り消すだけです。参照される表やビューからシステム権限が取り消されることはありません。

---

57. SQL の INDEX および ALTER 特権は、同じシステム権限 *OBJALTER に対応しています。 INDEX と ALTER の両方を認可しても、ユーザーに与えられる権限が増加するわけではありません。

58. WITH GRANT OPTION が与えられたユーザーは、ALTER および REFERENCES 権限によって与えられる機能も実行することができます。

59. 指定した権限が、ビューの定義で参照されている表およびビューに認可されるのは、権限の認可対象のユーザーが別の権限ソースからそのような権限 (例えば共通認可の権限) を入手していない場合だけに限られます。

表 56. ビューに対して認可または取り消しされる特権

SQL の特権	ビューに対する認可または取り消しに対応するシステム権限	参照された表およびビューに対する認可または取り消しに対応するシステム権限
ALL (ALL の GRANT または REVOKE は、ステートメントの権限 ID がもつ特権のみを認可または取り消します。)	*OBJALTER *OBJMGT (取り消しのみ) *OBJOPR *OBJREF *ADD *DLT *READ*UPD	*ADD *DLT *READ*UPD
ALTER	*OBJALTER[60]	なし
DELETE	*OBJOPR *DLT	*DLT
INDEX	該当しない	該当しない
INSERT	*OBJOPR *ADD	*ADD
REFERENCES	*OBJREF[60]	なし
SELECT	*OBJOPR *READ	*READ
UPDATE	*OBJOPR *UPD	*UPD
WITH GRANT OPTION	*OBJMGT	なし

例

例 1

ユーザーが権限を持っている場合、自分が表 WESTERN_CR (スキーマ KATHLEEN にある) に関して所有している全特権を PUBLIC に認可します。

```
GRANT ALL ON KATHLEEN.WESTERN_CR
TO PUBLIC
```

例 2

表 CALENDAR に関する適切な特権を認可して、ROANNA および EMMA が表 CALENDAR を読み取って、新しい項目を挿入できるようにします。ROANNA および EMMA には、既存の項目の変更や削除は許しません。

```
GRANT SELECT, INSERT ON CALENDAR
TO ROANNA, EMMA
```

例 3

TABLE1 および VIEW1 関する列特権を FRED に認可します。この GRANT ステートメントの中に指定されている列が両方とも、TABLE1 と VIEW1 のどちらにもなければなりません。

60. WITH GRANT OPTION がユーザーに付与された場合は、そのユーザーは、ALTER および REFERENCES 権限によって付与される機能も行うことができます。

```
GRANT UPDATE(column_1, column_2)
  ON  TABLE1, VIEW1
  TO FRED WITH GRANT OPTION
```

HOLD LOCATOR

HOLD LOCATOR ステートメントは、作業単位が変わっても LOB ロケーター変数が特定の値との関連を維持できるようにします。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込んで使用します。それ以外の使用法はありません。これを対話式に発行することはできません。このステートメントは、動的に準備できる実行可能ステートメントです。ただし、準備済みステートメントを実行するには、USING 文節を指定した EXECUTE ステートメントを使用しなければなりません。HOLD LOCATOR は、EXECUTE IMMEDIATE ステートメントと併用することはできません。

権限

権限は不要です。

構文

説明

ホスト変数, ...
ホスト変数を指定します。この変数は、ホスト変数のロケーター変数を宣言する規則に従って宣言されていなければなりません。この変数に、標識変数を指定してはなりません。ロケーター変数のタイプは、バイナリー・ラージ・オブジェクト・ロケーター、文字ラージ・オブジェクト・ロケーター、2 バイト文字ラージ・オブジェクト・ロケーターのいずれかでなければなりません。

HOLD LOCATOR ステートメントが実行された後は、ホスト変数リスト内の各ロケーター変数は保持プロパティーを持つことになります。

このホスト変数には、現在ロケーターが割り当てられている必要があります。つまり、この作業単位中に (CALL、FETCH、SELECT INTO、SET 変数、または VALUES INTO ステートメントによって) ロケーターが割り当てられていなければならず、それ以降そのロケーターが (FREE LOCATOR ステートメントによって) 解放されていてはならない、ということです。そうでない場合には、エラーが発生します。

HOLD LOCATOR ステートメントに複数のホスト変数が指定されていて、ロケーターの 1 つでエラーが発生した場合、どのロケーターも保持されません。

使用上の注意

保持プロパティーを持っているホスト変数 LOB ロケーター変数が解放される (つまり変数と値の間の関連が除去される) のは、以下の場合です。

- そのロケーター変数を対象とする SQL FREE LOCATOR ステートメントが実行されたとき。

- SQL ROLLBACK ステートメントが実行されたとき。
- SQL セッションが終了したとき。

例

従業員表に列 RESUME、HISTORY、および PICTURE が含まれていて、それらの列の値を表すためのロケーターがプログラムの中で確立されていると想定します。CLOB ロケーター変数 LOCRES および LOCHIST、および BLOB ロケーター変数 LOCPIC に、保持プロパティーを与えます。

```
HOLD LOCATOR :LOCRES,:LOCHIST,:LOCPIC
```

INCLUDE

INCLUDE ステートメントは、宣言またはステートメントをソース・プログラムに組み込みます。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込んで使用します。それ以外の使用法はありません。このステートメントは、実行可能ステートメントではありません。Java または REXX では指定できません。

権限

このステートメントの権限 ID は、メンバーを含むファイルについて、*OBJOPR および *READ のシステム権限を持つ必要があります。

構文

説明

SQLCA

SQL 連絡域 (SQLCA) の記述を組み込むことを指定します。 INCLUDE SQLCA は、1 つのプログラムで一度しか指定できません。プログラムに独立型の SQLCODE または独立型の SQLSTATE を組み込む場合は、INCLUDE SQLCA を指定してはなりません。

SQLCA は C、COBOL、および PL/I で指定できます。SQLCA を指定しない場合は、変数 SQLCODE または SQLSTATE をプログラムで使用する必要があります。詳細は、373 ページの 『SQL の戻りコード』を参照してください。

RPG プログラムでは、SQLCA を指定してはなりません。RPG プログラムの場合、プリコンパイラーによって自動的に SQLCA が組み込まれます。

SQLCA の説明については、 865 ページの『付録 B. SQL 連絡域』を参照してください。

SQLDA

SQL 記述子域 (SQLDA) の記述を組み込むことを指定します。 INCLUDE SQLDA は、C、COBOL、PL/I、および ILE RPG/400 で指定することができます。

SQLDA についての詳細は、 877 ページの『付録 C. SQL 記述子域 (SQLDA)』を参照してください。

メンバー名

CRTSQL*xxx* コマンドの INCFILE パラメーターに指定されているファイルから組み込むメンバーを識別します。

このメンバーには、いずれかのホスト言語ステートメント、および INCLUDE ステートメント以外の SQL ステートメントを入れることができます。COBOL の場合、DATA DIVISION または PROCEDURE DIVISION 以外で INCLUDE メンバー名 を指定してはなりません。

INCLUDE ステートメントは、プログラムをプリコンパイルすると、ソース・ステートメントに置き換えられます。

このため、プログラムで INCLUDE ステートメントを指定する場合は、置き換え後のソース・ステートメントがコンパイラーに受け入れられるような場所に置かなければなりません。

使用上の注意

SRCFILE パラメーターで指定されたソース・ファイルの CCSID が INCFILE パラメーターで指定されたソース・ファイルの CCSID と異なる場合は、INCLUDE ステートメントからのソースはソース・ファイルの CCSID に変換されます。

例

SQL 連絡域を C プログラムに組み込みます。

```
EXEC SQL INCLUDE SQLCA;
```

INSERT

INSERT ステートメントは、表またはビューに行を挿入します。ビューに行を挿入すると、そのビューの基礎になっている表にも行が挿入されます。

このステートメントには、次の 3 つの形式があります。

- *VALUES を使用した INSERT* の形式は、提供された値または参照された値を使用して、表またはビューに単一行を挿入する時に使用します。
- *SELECT を使用する INSERT* の形式は、他の表やビューからの値を使用して、表またはビューに 1 つ以上の行を挿入する時に使用します。
- *ROWS を使用する INSERT* 形式は、ホスト構造体配列で用意されている値を使用して、表またはビューに複数の行を挿入する場合に使用します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは動的に準備できる実行可能なステートメントです。ただし、ROWS の形式は例外で、アプリケーション・プログラムに組み込まれる静的ステートメントでなければなりません。n ROWS の形式は、REXX プロシージャーでは許されません。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメントに指定された表またはビューに対して、
  - その表やビューについての INSERT 特権、および
  - 表やビューが入っているライブラリーに対する *EXECUTE システム権限
- 管理権限

ステートメントの権限 ID は、次の場合に表についての INSERT 特権を持ちます。

- その表の所有者である。
- その表についての INSERT 特権を認可されている、または
- その表についての *OBJOPR および *ADD システム権限が認可されている。

ステートメントの権限 ID は、次の場合にビューに対して INSERT 特権をもちます。[61]

- そのビューについての INSERT 特権を認可されている。または、
- そのビューに対する *OBJOPR および *ADD システム権限が認可されており、さらに、そのビューの定義の最初の FROM 文節の最初の表またはビューに対する *ADD システム権限が認可されている。また、これがビューである場合は、そのビューの定義の最初の FROM 文節の最初の表またはビューに対する *ADD システム権限が認可されている。以下同様。

61. ビューの作成時に、所有者は必ずしもそのビューに対して INSERT 特権を獲得するとは限りません。所有者が INSERT 特権を獲得するのは、そのビューが挿入を許し、しかも所有者が副選択で参照されている最初の表に対して INSERT 特権を持っている場合のみに限られます。

副選択が指定されている場合、ステートメントの権限 ID によって保持される特権には次の 1 つが含まれていなければなりません。

- その副選択で識別された、それぞれの表またはビューごとに、
  - 表やビューに対する SELECT 特権、および
  - 表やビューが入っているライブラリーに対する *EXECUTE システム権限
- 管理権限

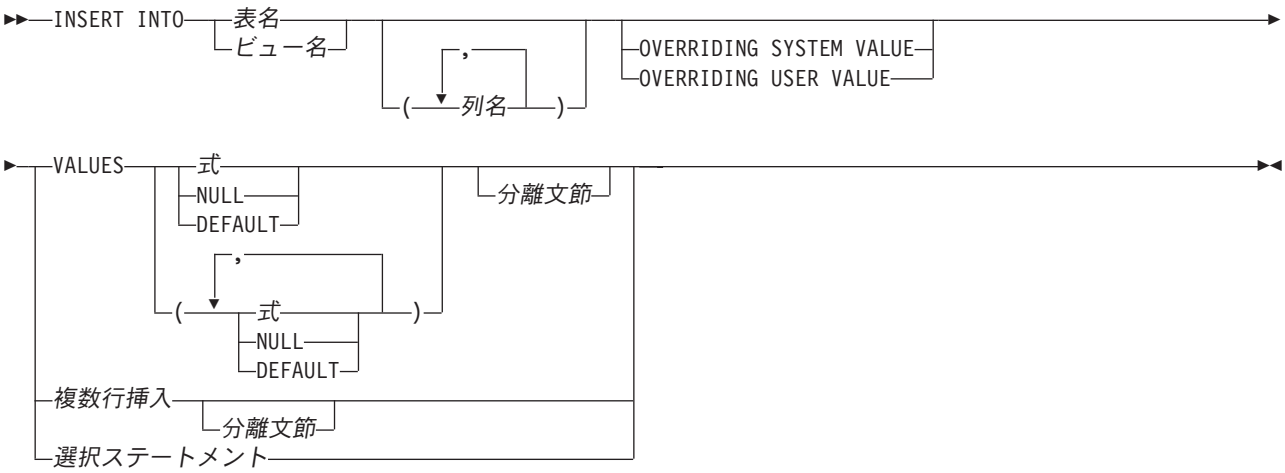
ステートメントの権限 ID は、以下の場合に表に対する SELECT 特権を持ちます。

- その表の所有者である。
- その表に対する SELECT 特権が認可されているか、または
- その表に対する *OBJOPR および *READ のシステム権限が認可されている。

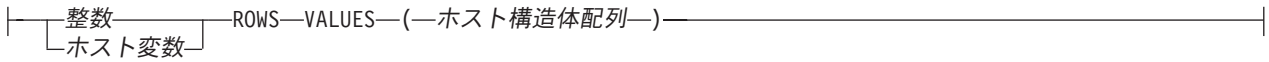
ステートメントの権限 ID は、以下の場合にビューに対する SELECT 特権を持ちます。

- そのビューの所有者である。
- そのビューに対する SELECT 特権が認可されているか、または
- そのビューについての *OBJOPR および *READ システム権限、およびそのビューが直接または間接に従属しているすべての表やビューについての *READ システム権限が認可されている。すなわち、そのビューの定義で参照されている表やビューのすべて、またそのようなビューが参照される場合、そのビューの定義で参照されている表やビューのすべて、以下同様。

構文



複数行挿入:



分離文節:

説明

INTO *表名* または *ビュー名*

挿入操作の対象となるものを識別します。この名前は、現行サーバーに存在している表またはビューを識別していなければなりませんが、カタログ表、カタログ表のビュー、または読み取り専用のビューを識別するものであってはなりません。

以下のようなビューの列には、値を挿入することはできません。

- 定数、式、またはスカラー関数から得られた列。
- そのビューの他の列と同じ基礎表の列から得られた列。

挿入操作の対象となるビューに上記のような列がある場合は、列名のリストを指定しなければなりません。この列名のリストから、値を挿入できない列の名前を除外する必要があります。

(*列名***、...)**

値を挿入する列を指定します。それぞれの名前は、表またはビューの列を識別す

る修飾のない名前でなければなりません。同じ列を複数回指定することはできません。また、ビューの列を指定する場合に、値を挿入できない列を指定してはなりません。

列名のリストを指定しなかった場合は、該当する表またはビューにあるすべての列を左から右の順序で指定したものと見なされます。このリストは、ステートメントを準備するときに確立されるので、ステートメントを準備した後で表に追加した列をリストに指定してはなりません。

INSERT ステートメントがアプリケーションに組み込まれ、参照している表またはビューがプログラムの作成時に存在している場合には、そのステートメントはプログラムの作成時に準備されます。これ以外の場合には、INSERT ステートメントは、そのステートメントの最初の正常な実行時に準備されます。

OVERRIDING SYSTEM VALUE または OVERRIDING USER VALUE
特定の ROWID または識別列についてシステムが生成した値またはユーザーが指定した値を使用するかどうかを指定します。 OVERRIDING SYSTEM VALUE を指定する場合は、INSERT ステートメントの対象とする暗黙または明示的な列リストに、GENERATED ALWAYS として定義された列が含まれていることが必要です。 OVERRIDING USER VALUE を指定する場合は、INSERT ステートメントの対象とする暗黙または明示的な列リストに、GENERATED ALWAYS または GENERATED BY DEFAULT として定義された列が含まれていることが必要です。

OVERRIDING SYSTEM VALUE
GENERATED ALWAYS として定義されている列について、VALUES 文節に指定されている値または全選択の結果として得られた値を使用することを指定します。システム生成の値は挿入されません。

OVERRIDING USER VALUE
GENERATED ALWAYS または GENERATED BY DEFAULT として定義されている列について、VALUES 文節に指定されている値または全選択の結果として得られた値を無視することを指定します。代わりにシステム生成の値が挿入され、ユーザー指定の値はオーバーライドされます。

OVERRIDING SYSTEM VALUE と OVERRIDING USER VALUE のどちらも指定しない場合は、以下のようになります。

- GENERATED ALWAYS として定義されている ROWID または識別列については、値を指定することはできません。
- GENERATED BY DEFAULT として定義されている ROWID または識別列については、値を指定することができます。値を指定した場合は、この列にその値が割り当てられます。ただし、BY DEFAULT として定義された ROWID 列に値を挿入できるのは、その値が、DB2 UDB (OS/390 および z/OS 版) または DB2 UDB for iSeries によりすでに生成されている有効な行 ID 値である場合に限られます。 BY DEFAULT として定義された識別列に値を挿入した場合は、その識別列が固有制約または固有索引内の唯一のキーである場合以外は、データベース・マネージャーはその指定された値が該当の列についての固有な値であるかどうかを検査しません。固有制約も固有索引もない場合は、データベース・マネージャーは、NO CYCLE が有効である場合に限り、システム生成の値のセットの中でのみ各値の固有性を保証します。

値が指定されていない場合は、データベース・マネージャーは新しい値を生成します。

VALUES
値のリストの形式で 1 つの新しい行を指定します。

VALUES 文節の値の数は、列のリストの列名の数と同じでなければなりません。リストの最初の列には VALUES 文節の最初の値が挿入され、リストの 2 番目の列には VALUES 文節の 2 番目の値が挿入されるというように、指定した列に対応する値が順に挿入されます。

式 列の値を、式から割り当てることを指定します。式 は、136 ページの『式』で説明しているタイプの任意の式です。式には、列関数や列名を含めることはできません。

式 が単一ホスト変数の場合、そのホスト変数は構造体を識別できます。この文節に指定する各ホスト変数は、ホスト構造体やホスト変数の宣言の規則に従って宣言されているホスト構造体またはホスト変数を識別していなければなりません。このステートメントの操作形式では、ホスト構造体に対する参照は、その個々の変数それぞれに対する参照によって置き換えられます。ホスト変数 の説明については、第 2 章を参照してください。

NULL
列の値をヌル値にすることを指定します。NULL は、ヌル可能列にのみ指定してください。

DEFAULT
列にデフォルト値を割り当てることを指定します。挿入する値は、次のように、列がどのように定義されたかによって異なります。

- WITH DEFAULT 文節が使用される場合、挿入されるデフォルト値は、その列に関して定義されている値になります (542 ページの『CREATE TABLE』の *列定義* の *デフォルト文節* を参照してください)。
- WITH DEFAULT 文節または NOT NULL 文節が使用されない場合、挿入される値は NULL です。
- NOT NULL 文節が使用されていて、WITH DEFAULT 文節が使用されていないか、DEFAULT NULL が使用されている場合、その列については DEFAULT キーワードは指定できません。
- 列が ROWID または識別列である場合は、データベース・マネージャーは新しい値を生成します。

GENERATED ALWAYS として定義されている ROWID または識別列については、OVERRIDING USER VALUE 文節を指定した場合、つまりユーザー指定の値を無視してシステム生成の固有値を挿入することを指示した場合以外は、DEFAULT を指定する必要があります。

選択ステートメント
選択ステートメントの結果表の形式で、新しい一連の行を指定します。FOR READ ONLY 文節、FOR UPDATE 文節、および OPTIMIZE 文節は、挿入により使用される選択ステートメントでは無効です。選択ステートメントで ORDER BY 文節を指定すると、その ORDER BY 文節によって識別される列の値にしたがって行が挿入されます。選択ステートメントの説明については、355 ページの 『選択ステートメント』の項を参照してください。

選択ステートメントの使用により、1 行、複数行、またはゼロ行の行を挿入することができます。挿入される行がない場合、SQLCODE は +100 にセットされ、SQLSTATE は '02000' にセットされます。

INSERT の基本オブジェクトと、選択ステートメント内のいずれかの副選択の基本オブジェクトが同じ表であるとき、その選択ステートメントは、行が挿入される前にすべて評価されます。

結果表の列の数と、列のリストに指定した列名の数は同じでなければなりません。リストの最初の列には、結果表の最初の列の値が挿入され、リストの 2 番目の列には、結果表の 2 番目の列が挿入されるというように、対応する列の値が順に挿入されます。

分離文節

INSERT ステートメントに関して、使用したい分離レベルを指定します。分離文節の説明については、360 ページの 『ISOLATION 文節』を参照してください。

複数行挿入

整数 または *ホスト変数* **ROWS**

挿入する行数を指定します。ホスト変数を指定する場合、そのホスト変数は位取りゼロの数値でなければならず、また標識変数を含むことはできません。

VALUES (*ホスト構造体配列***)**

ホスト構造体の配列の形式で新しい一連の行を指定します。ホスト構造体配列は、その宣言の規則に従ってプログラムで宣言されていなければなりません。ホスト構造体配列名の代わりに、パラメーター・マーカーを使用することができます。

ホスト構造体の変数の数は、列のリストの名前の数に等しくなければなりません。配列の最初のホスト構造体は最初の行に対応し、配列の 2 番目のホスト構造体は 2 番目の行に対応します。以下同様です。さらに、ホスト構造体の最初の変数は、該当の行の最初の列に対応し、ホスト構造体の 2 番目の変数は、該当の行の 2 番目の列に対応します。以下同様です。

ホスト構造体の配列についての説明は、
129 ページの 『C、C++、COBOL、PL/I、および RPG におけるホスト構造配列』を参照してください。

挿入値に LOB が含まれている場合、または現行接続が iSeries 以外のリモート・サーバーへの接続の場合、複数行挿入は許されません。

INSERT の規則

デフォルト値

列のリストに指定されていない列には、その列のデフォルト値が挿入されます。デフォルト値を持たない列は、必ず列のリストに指定しなければなりません。同様に、ビューに値を挿入する場合に、そのビューに含まれていない基礎表の列があれば、基礎表の該当する列には、デフォルト値が挿入されます。したがって、ビューにない基礎表の列は、すべてデフォルト値をもっていなければなりません。

割り当て

挿入する値は、第 2 章で説明されている割り当て規則に従って、列に割り当てられます。

妥当性

識別された表、または識別されたビューの基礎表が、1 つまたは複数の固有索引あるいは固有制約を持つ場合、表に挿入される各行は、それらの索引によって課せられた制約に適合していなければなりません。

固有索引および固有制約は、COMMIT(*NONE) の指定がある場合を除き、そのステートメントの終わりでチェックされます。複数行挿入の場合、これはすべての行が挿入され、関連のトリガーが起動された後で行われます。
COMMIT(*NONE) が指定されている場合は、各行が挿入されるごとにチェックが行われます。

識別された表、または識別されたビューの基礎表が、1 つまたは複数の検査制約を持つ場合、表に挿入される各行ごとに、検査制約は真または不明でなければなりません。

検査制約は、ステートメントの終わりで必ずチェックされます。複数行挿入の場合、このチェックはすべての行が挿入された後に行われます。

ビューが識別される場合は、挿入される行は、適用される WITH CHECK OPTION に適合しなければなりません。詳細は、589 ページの 『CREATE VIEW』を参照してください。

トリガー

識別された表または識別されたビューの基礎表が挿入トリガーを持つ場合、トリガーが起動されます。トリガーが起動された結果、挿入する値に応じて、他のステートメントが実行されたり、エラー条件が発生したりすることがあります。

参照保全

外部キーの非ヌルの挿入値は、関連の親表の親キーの値のいずれかに等しくなければなりません。

参照制約 (RESTRICT 削除規則を伴う参照制約以外の) は、ステートメントの終わりで実際上チェックされます。複数行挿入の場合、これはすべての行が挿入され、関連のトリガーが起動された後で行われます。

使用上の注意

挿入値がいずれかの制約に違反した場合、またはその他のエラーが INSERT ステートメントの実行中に発生し、しかも COMMIT(*NONE) が指定されていた場合は、そのステートメントの実行中に行われた変更はすべて撤回されます。ただし、エラーが発生する前に、その作業単位の中で行われていたその他の変更は撤回されません。COMMIT(*NONE) が指定されていれば、変更が撤回されることはありません。

INSERT ステートメントの実行後、SQLCA の SQLERRD(3) の値には、データベース・マネージャーが挿入した行の数がセットされます。 SQLERRD(3) の値には、トリガーの結果として挿入された行の数は含まれません。

COMMIT(*RR)、COMMIT(*ALL)、COMMIT(*CS)、または COMMIT(*CHG) が指定されている場合は、正常に実行される INSERT ステートメントの実行中に、1 つま

たは複数の排他的ロックが掛けられます。そのようなロックがコミットまたはロールバック操作によって解放されるまで、挿入された行は、以下によってのみアクセスすることができます。

- その挿入を行ったアプリケーション・プロセス
- 読み取り専用カーソル、SELECT INTO ステートメント、または副照会を介して、COMMIT(*NONE) または COMMIT(*CHG) を使用する別のアプリケーション・プロセス

ロックは、他のアプリケーション・プロセスがその表の操作を行うのを防止します。ロックの詳細については、COMMIT、ROLLBACK、および LOCK TABLE の各ステートメントの説明を参照してください。また 24 ページの『分離レベル』および DB2 UDB for iSeries データベース・プログラミング を参照してください。

COMMIT(*RR)、COMMIT(*ALL)、COMMIT(*CS)、または COMMIT(*CHG) を指定した場合は、1 つの INSERT ステートメントで最高 500 000 000 行を挿入または変更することができます。変更される行の数には、トリガーの結果として同じコミットメント定義のもとで挿入、更新、または削除される行が含まれます。

ホスト変数は、REXX プロシージャー内の INSERT ステートメントでは使用できません。INSERT を使用する場合は、必ず、パラメーター・マーカーを使用する PREPARE および EXECUTE の対象として使用してください。

同義のキーワード： 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- キーワード NONE を NC の同義語として使用することができます。
- キーワード CHG を UR の同義語として使用することができます。
- キーワード ALL を RS の同義語として使用することができます。

例

例 1

以下の仕様を持つ新しい部門を、表 DEPARTMENT に挿入します。

- 部門番号 (DEPTNO) は、'E31'
- 部門名 (DEPTNAME) は、'ARCHITECTURE'
- 管理担当者の従業員番号 (MGRNO) は、'00390'
- 報告先の部門 (ADMRDEPT) は、部門 'E01'

```
INSERT INTO DEPARTMENT
  VALUES ('E31', 'ARCHITECTURE', '00390', 'E01')
```

例 2

例 1 と同じように新しい部門を表 DEPARTMENT に挿入します。ただし、この新しい部門には管理担当者を割り当てません。

```
INSERT INTO DEPARTMENT (DEPTNO, DEPTNAME, ADMRDEPT)
  VALUES ('E31', 'ARCHITECTURE', 'E01')
```

例 3

EMPPROJACT 表と同じ列構成で、表 MA_EMPPROJACT を作成します。そこで、プロジェクト番号 (PROJNO) が文字 'MA' で始まっている行を、EMPPROJACT 表から MA_EMPPROJACT にロードします。

```
CREATE TABLE MA_EMPPROJACT
          LIKE EMPPROJACT
INSERT INTO MA_EMPPROJACT
  SELECT * FROM EMPPROJACT
    WHERE SUBSTR(PROJNO, 1, 2) = 'MA'
```

例 4

C プログラムのステートメントを使用して、表 PROJECT にプロジェクトの骨組みを追加します。プロジェクト番号 (PROJNO)、プロジェクト名 (PROJNAME)、部門番号 (DEPTNO)、および管理担当者 (RESPEMP) の値は、ホスト変数から入手します。プロジェクト開始日付 (PRSTDATE) には、現在の日付を使用します。表内のその他の列には、NULL 値を割り当てておきます。

```
EXEC SQL  INSERT INTO PROJECT  (PROJNO, PROJNAME, DEPTNO, RESPEMP, PRSTDATE)
            VALUES (:PRJNO, :PRJNM, :DPTNO, :REMP, CURRENT DATE);
```

例 5

PL/I プログラムで、ブロック化挿入を使用して、表 DEPARTMENT に 10 行を追加します。挿入するデータは、ホスト構造体配列 DEPT に入っています。

```
DCL  1  DEPT(10),
        3  DEPT  CHAR(3),
        3  LASTNAME  CHAR(29)  VARYING,
        3  WORKDEPT  CHAR(6),
        3  JOB  CHAR(3);

EXEC SQL INSERT INTO DEPARTMENT 10 ROWS VALUES (:DEPT);
```

例 6

READ UNCOMMITTED (UR, CHG) オプションを使用して、EMPPROJACT 表に新しいプロジェクトを挿入します。

```
INSERT INTO EMPPROJACT
  VALUES ('000140', 'PL2100', 30)
  WITH CHG
```

LABEL

LABEL ステートメントは、表、ビュー、別名、パッケージ、または列のカタログ記述のラベルの付加または置き換えを行います。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことも、対話式に呼び出すこともできます。このステートメントは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメントで識別されている表、ビュー、別名、またはパッケージの場合、
  - その表、ビュー、別名、またはパッケージに対する ALTER 特権
  - その表、ビュー、別名、またはパッケージが入っているライブラリーに対する *EXECUTE システム権限
- 管理権限

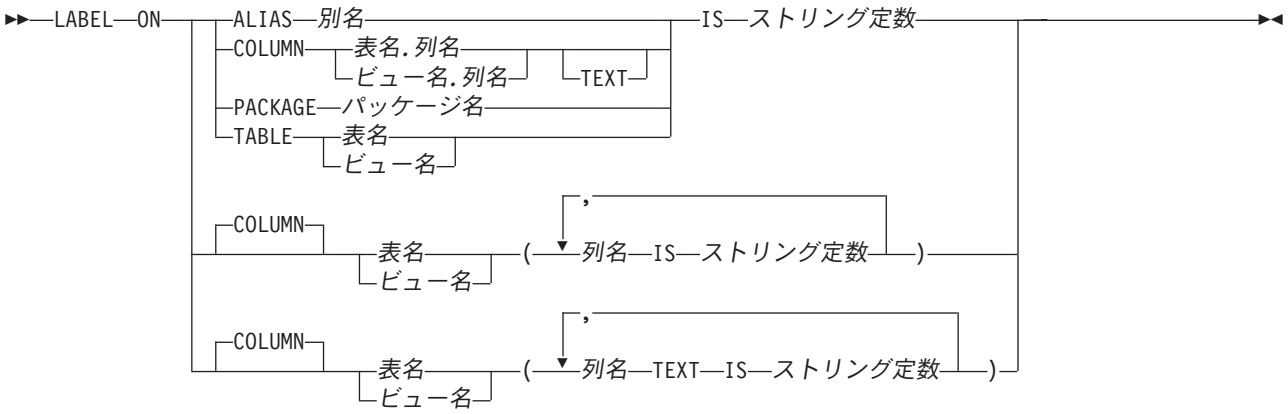
ステートメントの権限 ID は、次の場合に表、ビューまたはパッケージに対する ALTER 権限を保有します。

- その表、ビュー、またはパッケージの所有者である場合。
- その表、ビュー、またはパッケージに対する ALTER 特権が認可されている場合。
- その表、ビュー、またはパッケージに対する *OBJALTER または *OBJMGT のシステム権限のいずれかが認可されている場合。

ステートメントの権限 ID は、次の場合に別名に対する ALTER 権限を保有します。

- その別名の所有者である場合。
- その別名の *OBJALTER または *OBJMGT のいずれかのシステム権限が認可されている場合。

構文



説明

ALIAS
別名のラベルであることを指定します。別名のラベルは、システム・オブジェクト・テキストとして付加されます。

別名
ラベルを適用する別名を識別します。この名前は、現行サーバーに存在している別名を識別していなければなりません。

COLUMN
列のラベルであることを指定します。列のラベルは、システム列見出しまたは列テキストとして付加されます。列見出しは、照会の結果を表示または印刷する場合に使用されます。

表名.列名 または*ビュー名.列名*
ラベルを適用する列を指定します。*表名* または*ビュー名* は、現行サーバーにある表またはビューを示すものでなければなりませんが、グローバル一時表を示すものであってはなりません。*列名* は、その表またはビューの列を識別するものでなければなりません。

TEXT
OS/400 の列テキストを使用することを指定します。 TEXT を省略すると、列見出しが指定されます。

PACKAGE
パッケージのラベルであることを指定します。パッケージのラベルは、システム・オブジェクト・テキストとして付加されます。

パッケージ名
ラベルを適用するパッケージを識別します。この名前は、現行サーバーに存在しているパッケージを識別していなければなりません。

LABEL

TABLE

表またはビューのラベルであることを指定します。表またはビューのラベルは、システム・オブジェクト・テキストとして付加されます。

表名 または *ビュー名*

表またはビューを識別します。ここで指定した表またはビューに関するラベルが追加されます。*表名* または *ビュー名* は、現行サーバーにある表またはビューを示すものでなければなりませんが、グローバル一時表を示すものであってはなりません。

IS

この後に、付加したいラベルを指定します。

ストリング定数

表、ビュー、別名、SQL パッケージ、または列のテキストの場合は、50 バイトまでの長さ、列の見出しの場合は、60 バイトまでの長さであればどのような SQL 文字ストリング定数でも構いません。この定数には、1 バイト文字および 2 バイト文字を入れることができます。

列見出しとしてのラベルは、20 バイトまでの 3 つの部分 (セグメント) から構成されています。対話式 SQL、Query/400 プログラム、DB2 Query Manager and SQL Development Kit for iSeries、およびその他のプロダクトによって、各 20 バイトのセグメントをそれぞれ別の行に表示または印刷することができます。列のラベルに混合データが使用される場合、20 バイトのそれぞれのセグメントは、有効な混合データ文字ストリングでなければなりません。シフト文字は、20 バイトのセグメントの中で対になっていなければなりません。

使用上の注意

列見出しは、照会の結果を表示または印刷する場合に使用されます。 最初の列見出しは最初の行に、2 番目の列見出しは 2 行目に、3 番目の列見出しは 3 行目に、それぞれ表示または印刷されます。列見出しは最高 60 バイトまでですが、最初の 20 バイトが最初の列見出し、2 番目の 20 バイトが 2 番目の列見出し、3 番目の 20 バイトが 3 番目の列見出しになります。ブランクは、それぞれの 20 バイトの列見出しの終わりから削除されます。

列見出し情報の 60 バイトすべてがカタログ・ビュー SYSCOLUMNS で使用可能です。ただし、DESCRIBE または DESCRIBE TABLE ステートメントの SQLDA で戻されるのは、最初の列見出しだけです。

DESCRIBE または DESCRIBE TABLE ステートメントでは、列テキストが戻されません。データベース・マネージャーが、共用されているレコード様式の記述の列見出しの情報を変更すると、その変更は、その様式の記述を共用するすべてのファイルに反映されます。ファイルが他のファイルと様式を共用しているかどうかを調べるには、CL コマンドのデータベース関係表示 (DSPDBR) の RCDFMT パラメーターを使用します。

同義のキーワード： 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- PACKAGE の同義語として、キーワード PROGRAM を使用することもできます。

例

- 表 DEPARTMENT の DEPTNO 列にラベルを付けます。

```
LABEL ON COLUMN DEPARTMENT.DEPTNO
  IS 'DEPARTMENT NUMBER'
```

- 列見出しが 2 行に表示されている、表 DEPARTMENT の列 DEPTNO にラベルを付けます。

```
LABEL ON COLUMN DEPARTMENT.DEPTNO
  IS 'Department          Number'
```

- パッケージ PAYROLL にラベルを付けます。

```
LABEL ON PACKAGE CORPDATA.PAYROLL
  IS 'Payroll Package'
```

LOCK TABLE

LOCK TABLE ステートメントは、並行して実行されるアプリケーション・プロセスによる表の変更や表の使用を防止します。

呼び出し

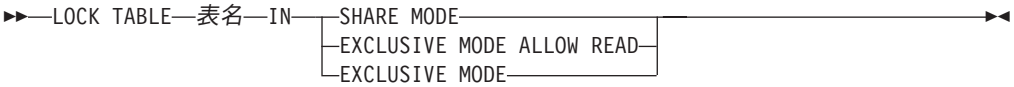
このステートメントは、アプリケーション・プログラムに組み込むことも、対話式に呼び出すこともできます。このステートメントは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメントで識別される表に対して、
  - その表についての *OBJOPR システム権限、および
  - その表が入っているライブラリーに対する *EXECUTE システム権限
- 管理権限

構文



説明

表名

ロックする表を識別します。この表名は、現行サーバー上に存在する基礎表を示すものでなければならず、カタログ表またはグローバル一時表を示すものであってはなりません。

IN SHARE MODE

並行して実行されているアプリケーション・プロセスが、この表に対して読み取り専用操作以外の操作を実行できないようにします。このステートメントが実行されるアプリケーション・プロセス用の共用ロック (*SHRNUP) が確立されます。他のアプリケーション・プロセスが共用ロック (*SHRNUP) を確立することもあり、その場合、このアプリケーション・プロセスが読み取り専用以外の操作を実行できないようにします。

IN EXCLUSIVE MODE ALLOW READ

並行して実行されているアプリケーション・プロセスが、この表に対して読み取り専用操作以外の操作を実行できないようにします。このステートメントが実行されるアプリケーション・プロセス用の排他読み取り許可ロック (*EXCLRD) が確立されます。他のアプリケーション・プロセスは共用ロック (*SHRNUP) を確立できず、その場合、このアプリケーション・プロセスが該当の表に対して更新、削除、および挿入を実行できないようにすることは不可能です。

IN EXCLUSIVE MODE

並行して実行されるアプリケーション・プロセスが、この表に対していかなる操

作も実行できないようにします。このステートメントが実行されるアプリケーション・プロセス用の排他ロック (*EXCL) が確立されます。

ロックは、LOCK TABLE ステートメントが実行されたときに確立されます。

ロックは、以下の時点で解放されます。

- 作業単位が終了したとき。ただし、作業単位が COMMIT HOLD または ROLLBACK HOLD によって終了した場合を除きます。
- プログラム・スタックの中の最初の SQL プログラムが終了したとき。ただし、CLOSQLCSR(*ENDJOB) または CLOSQLCSR(*ENDACTGRP) が CRTSQLxxx コマンドで指定されていた場合を除きます。
- 活動化グループが終了したとき。
- 接続が CONNECT (タイプ 1) ステートメントの使用により変更されたとき。
- そのロックに関連する接続が DISCONNECT ステートメントの使用により切り離されたとき。
- 接続が解放保留状態にあり、正常な COMMIT が行われたとき。

また、オブジェクト割り振り解除 (DLCOBJ) コマンドを出して、表をアンロックすることもできます。

同期ステートメントなので、アプリケーション・プロセスから LOCK TABLE ステートメントを出したときに、競合するロックをすでに他のアプリケーション・プロセスが保持していると、ステートメントを出したアプリケーション・プロセスは、最大でデフォルトの待ち時間だけ待ち状態になります。

例

表 DEPARTMENT をロックします。表 DEPARTMENT がロックされている間は、他のアプリケーション・プロセスによるこの表の更新および読み取りを許しません。

```
LOCK TABLE DEPARTMENT IN EXCLUSIVE MODE
```

OPEN

OPEN ステートメントは、カーソルをオープンします。

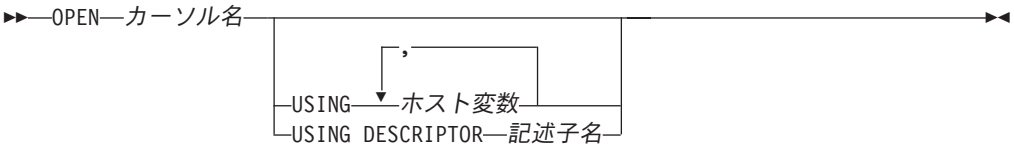
呼び出し

このステートメントは、アプリケーション・プログラムに組み込んで使用します。それ以外の使用法はありません。このステートメントは、動的には準備できない実行可能ステートメントです。

権限

カーソルを使用するために必要な権限については、 597 ページの『DECLARE CURSOR』を参照してください。

構文



説明

カーソル名

オープンするカーソルを識別します。このカーソル名 は、宣言されているカーソルを識別しなければなりません。カーソルの宣言については、DECLARE CURSOR ステートメントの「使用上の注意」の項を参照してください。このカーソルは、OPEN ステートメントを実行するときには、クローズ状態になければなりません。

カーソルに関連付けられる SELECT ステートメントは、以下のいずれかです。

- DECLARE CURSOR ステートメントで指定した*選択*ステートメント、または
- DECLARE CURSOR ステートメントで指定したステートメント名 によって識別される準備済み*選択*ステートメント。このステートメントが正しく準備されていない場合や、*選択*ステートメント でない場合は、カーソルを正常にオープンすることはできません。

カーソルの対象となる結果表は、SELECT ステートメントを評価することによって得られます。SELECT ステートメントを評価するときには、SELECT ステートメントに特殊レジスターが指定されていれば、その特殊レジスターの現行値が使用され、OPEN ステートメントの USING 文節または SELECT ステートメントにホスト変数が指定されていれば、そのホスト変数の現行値が使用されます。結果表の行は、OPEN ステートメントの実行時に取得され、一時表に保持される場合と、後続の FETCH ステートメントの実行時に取得される場合があります。どちらの場合も、カーソルはオープン状態になり、結果表の最初の行の前に位置付けられます。ただし、表が空であれば、実際のカーソルの位置は “最終行の後” になります。

USING

この後にホスト変数のリストを指定します。準備済みステートメントのパラメーター・マーカー (疑問符) は、ここに指定したホスト変数の値によって置き換えられます。パラメーター・マーカーの説明については、725 ページの『PREPARE』を参照してください。DECLARE CURSOR ステートメントでパラメーター・マーカーの入った準備済みステートメントを指定している場合は、USING を使用する必要があります。準備済みステートメントにパラメーター・マーカーが入っていない場合は、USING は無視されます。

ホスト変数,...
: ホスト構造体またはホスト変数を指定します。指定するホスト構造体または変数はそれらの宣言の規則に従ってプログラムで宣言されていなければなりません。ホスト構造体に対する参照は、その個々の変数に対する参照に置き換えられます。リストされた変数の数は、準備済みステートメントのパラメーター・マーカーの数と同じでなければなりません。リスト中の *n* 番目の変数は、準備済みステートメントの *n* 番目のパラメーター・マーカーに対応します。

DESCRIPTOR *記述子名*
: SQLDA を識別します。この SQLDA には、ホスト変数の有効な記述が入っていなければなりません。

  OPEN ステートメントを処理する前に、ユーザーは SQLDA の以下のフィールドをセットしておく必要があります。 (REXX の場合は、規則が異なります。詳細については、DB2 UDB for iSeries ホスト言語での SQL プログラミングを参照してください。)

  - SQLN (SQLDA に用意する SQLVAR のオカレンスの数を示します。)
  - SQLDABC (SQLDA 用に割り振る記憶域のバイト数を示します。)
  - SQLD (ステートメントを処理するときに、SQLDA で使用する変数の個数を指示します。)
  - SQLVAR の各オカレンス (変数の属性を指示します。)

  SQLDA の記憶域は、SQLVAR のオカレンスをすべて収容するのに十分な大きさで割り振らなければなりません。 LOB または特殊タイプが結果の中に存在する場合、各パラメーター・マーカーごとに追加の SQLVAR 項目が必要です。 SQLDA の詳細については、SQLVAR の説明や SQLVAR のオカレンス数の判別方法の説明も含めて、877 ページの『付録 C. SQL 記述子域 (SQLDA)』を参照してください。

  SQLD には、ゼロ以上で SQLN 以下の値をセットしなければなりません。この値は、準備済みステートメント内のパラメーター・マーカーの個数と同じでなければなりません。SQLDA で記述されている *n* 番目の変数が、準備済みステートメントの *n* 番目のパラメーター・マーカーに対応します。

RPG/400 はポインターを設定する機能を用意しておらず、SQLDA はポインターを使用して、適切なホスト変数を見つけるため、ユーザーは、RPG/400 アプリケーションの外側でポインターを設定しなければならないことに注意する必要があります。

パラメーター・マーカーの置き換え

ステートメント内にある各パラメーター・マーカーは、実際には、カーソルに関連する SELECT ステートメントが評価されるときに、対応するホスト変数の値に置き換えられます。パラメーター・マーカーの置き換えは、ホスト変数の値をソースとし、データベース・マネージャー内部の変数をターゲットとする割り当て演算によって処理されます。 タイプ付きパラメーター・マーカーの場合、ターゲット変数の属性は、CAST によって指定されたものになります。タイプ無しパラメーター・マーカーの場合、ターゲット変数の属性は、パラメーター・マーカーのコンテキストによって決まります。パラメーター・マーカーに適用される規則については、 729 ページの表 57 を参照してください。

V が、パラメーター・マーカー P に対応するホスト変数を指すものとします。V の値は、値を列に割り当てる場合の規則に従って、P のターゲットの変数に割り当てられます。したがって、次のことがいえます。

- V は、ターゲットと互換性のあるものでなければなりません。
- V が数値ならば、V の整数部の絶対値は、ターゲットの整数部の絶対値の最大を超えてはなりません。
- V の属性がターゲットの属性と一致しない場合は、ターゲットの属性に合わせて値が変換されます。
- ターゲットにヌルを入れることができない場合は、V の値はヌルであってはなりません。

ただし、値を列に割り当てる場合の規則とは、以下の点が異なります。

- V がストリングで、その長さがターゲットの長さ属性より大きければ、V の値は途中で切り捨てられます (エラーは出されません)。

カーソルの SELECT ステートメントが評価されるときに、P の代わりに使用される値は、P のターゲット変数の値です。例えば、V が CHAR(6) で、ターゲットが CHAR(8) の場合は、P の代わりに使用される値は、V の値に 2 つのブランクが埋め込まれた値になります。

USING 文節は、パラメーター・マーカーが入っている準備済み SELECT ステートメントを対象としたものです。しかし、カーソルに関連する SELECT ステートメントが、DECLARE CURSOR ステートメントの一部として入っているときにも、USING 文節を使用することができます。この場合、OPEN ステートメントは、SELECT ステートメント内のホスト変数の属性がターゲットの変数の属性と同じであることを除けば、SELECT ステートメント内のホスト変数がすべてパラメーター・マーカーである場合と同じように実行されます。このため、カーソルに関連する SELECT ステートメントにあるホスト変数の値は、USING 文節内のホスト変数の値に変更されることになります。

使用上の注意

クローズ状態のカーソル

以下の時点では、プログラム内のすべてのカーソルはクローズ状態にあります。

- プログラムが呼び出されたとき。

  - CLOSQLCSR(*ENDPGM) が指定されている場合、プログラムが呼び出されるたびに、すべてのカーソルがクローズ状態になります。
  - CLOSQLCSR(*ENDSQL) が使用されている場合、1 つの SQL プログラムが呼び出しスタックに残っている間は、プログラムが初めて呼び出されるときに限って、すべてのカーソルがクローズ状態になります。
  - CLOSQLCSR(*ENDJOB) が指定されている場合は、ジョブが活動状態である限りは、プログラムが初めて呼び出された時に限って、すべてのカーソルがクローズ状態になります。
  - CLOSQLCSR(*ENDMOD) が指定されている場合、モジュールが開始されるたびに、すべてのカーソルがクローズ状態になります。
  - CLOSQLCSR(*ENDACTGRP) が指定されている場合は、プログラム内のモジュールが活動化グループ内で最初に開始された時に限って、すべてのカーソルがクローズ状態になります。
- HOLD オプションの指定がない COMMIT または ROLLBACK ステートメントを実行して、プログラムから新しい作業単位を開始したとき。HOLD オプションを指定して宣言されたカーソルは、COMMIT ステートメントではクローズされません。
- CONNECT (タイプ 1) ステートメントが実行されたとき。

また、次の場合に、カーソルがクローズ状態になることもあります。

- CLOSE ステートメントが実行されたとき。
- DISCONNECT ステートメントによって、そのカーソルが関連する接続が切り離されたとき。
- そのカーソルが関連した接続が解放保留状態にあり、正常な COMMIT が行われたとき。

カーソルの結果表から行を取り出すには、カーソルがオープンされているときに、FETCH ステートメントを実行しなければなりません。クローズ状態のカーソルをオープン状態に変更する方法は、OPEN ステートメントを実行する以外にはありません。

一時表の効果

カーソルの結果表が読み取り専用でなければ、その結果表の行は後続の FETCH ステートメントを実行したときに取得されます。これと同じ方式は、読み取り専用の結果表にも使用されます。ただし、結果表が読み取り専用である場合は、DB2 UDB for iSeriesがこの方式に代えて一時表方式の使用を選択することがあります。一時表を使用する方式では、OPEN ステートメントの実行時に、結果表全体が一時表に挿入されます。一時表を使用した場合は、プログラムの結果が以下の 2 つの点で異なります。

- 通常は、後続の FETCH ステートメントが実行されるまでは発生しないエラーが、OPEN ステートメントの実行時に発生する可能性がある。
- カーソルがオープンされている時に INSERT、UPDATE、および DELETE ステートメントを実行すると、結果表に影響を与えない可能性がある。

逆に、一時表を使用しない場合は、カーソルがオープンされている間に INSERT、UPDATE、および DELETE ステートメントを実行すると、結果表に影響を与える可能性があります。このような操作の影響は、常に予測できるとは限りま

せん。例えば、SELECT * FROM T として定義されている結果表の行にカーソル C が位置付けられているときに、T に対して行を挿入した場合は、その行の順序が定っていないことから、その挿入が結果表に与える影響は予測できないものになります。後続の FETCH C で、T の新しい行が取り出されることも、取り出されないこともあります。

例

例 1

COBOL プログラム内に、以下のような処理を行う組み込みステートメントを書きます。

1. カーソル C1 を定義します。このカーソルは、管理部門 (ADMRDEPT) 'A00' によって管理されている部門の行を、表 DEPARTMENT からすべて取り出すために使用します。
2. 最初に取り出す行の前に、カーソル C1 を位置付けます。

```
EXEC SQL  DECLARE C1 CURSOR FOR
           SELECT DEPTNO, DEPTNAME, MGRNO FROM DEPARTMENT
              WHERE ADMRDEPT = 'A00'  END-EXEC.

EXEC SQL  OPEN C1  END-EXEC.
```

例 2

C プログラムに OPEN ステートメントをコーディングして、カーソル DYN_CURSOR を、動的に定義される選択ステートメントに関連付けます。準備済み選択ステートメントの選択リストには、すでに 2 つの項目が定義されていているものとします。最初の項目のデータ・タイプは整数で、2 番目の項目のデータ・タイプは VARCHAR(64) です。 (以下の例には、関連するホスト変数の定義、PREPARE ステートメント、および DECLARE CURSOR ステートメントも示しています。)

```
EXEC SQL  BEGIN DECLARE SECTION;
   static short hv_int;
   char hv_vchar64[64];
   char stmt1_str[200];
EXEC SQL  END DECLARE SECTION;

EXEC SQL  PREPARE STMT1_NAME FROM :stmt1_str;

EXEC SQL  DECLARE DYN_CURSOR CURSOR FOR STMT1_NAME;

EXEC SQL  OPEN DYN_CURSOR USING :hv_int, :hv_vchar64;
```

例 3

例 2 と同じような OPEN ステートメントをコーディングします。ただし、この例では、選択ステートメント内の項目の数とデータ・タイプは分かっていません。

```
EXEC SQL  BEGIN DECLARE SECTION;
   char stmt1_str[200];
EXEC SQL  END DECLARE SECTION;
EXEC SQL  INCLUDE SQLDA;

EXEC SQL  PREPARE STMT1_NAME FROM :stmt1_str;
EXEC SQL  DECLARE DYN_CURSOR  CURSOR FOR STMT1_NAME;

EXEC SQL  OPEN DYN_CURSOR USING DESCRIPTOR :sqlda;
```

PREPARE

PREPARE ステートメントは、文字ストリング形式のステートメントから実行可能な形式の SQL ステートメントを作成します。このような文字ストリング形式は、*ステートメント・ストリング* と呼ばれ、実行可能な形式は、*準備済みステートメント* と呼ばれます。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込んで使用します。それ以外の使用法はありません。このステートメントは、動的には準備できない実行可能ステートメントです。

権限

権限の規則は、その PREPARE ステートメントに指定された SQL ステートメントに対して定義されている規則と同じです。例えば、SELECT ステートメントを準備する場合に適用される権限規則については、355 ページの 『選択ステートメント』を参照してください。

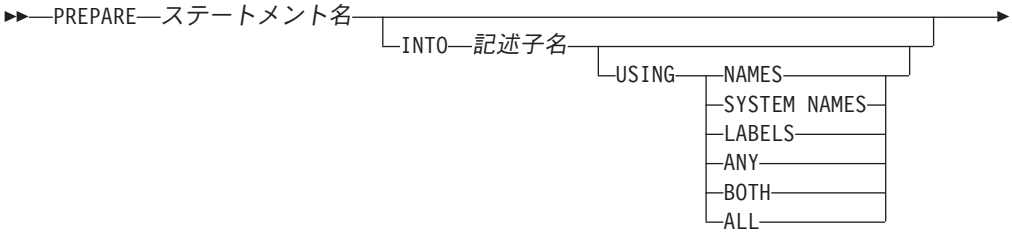
CRTSQLxxx コマンドに DLYPRP(*NO) が指定されていると、以下の場合を除き、権限の検査は該当のステートメントが準備される時点で行われます。

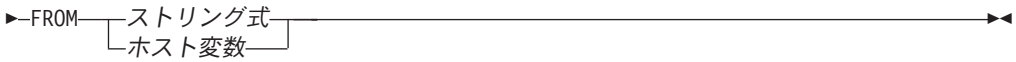
- DROP SCHEMA ステートメントを準備する場合、該当のスキーマのすべてのオブジェクトについての *OBJEXIST システム権限は、そのステートメントの実行時まで検査されません。
- DROP TABLE ステートメントを準備する場合、該当の表を参照するビュー、索引、および論理ファイルのすべてについての *OBJEXIST システム権限は、そのステートメントの実行時点まで検査されません。
- DROP VIEW ステートメントを準備する場合、該当のビューを参照するすべてのビューについての *OBJEXIST システム権限は、そのステートメントの実行時点まで検査されません。

CRTSQLxxx コマンドに DLYPRP(*YES) が指定されている場合、該当のステートメントが実行されるか、または OPEN ステートメントで使用されるまで、権限の検査はすべて据え置かれます。

プログラムの作成時点の CRTSQLxxx コマンドに DYNUSRPRF(*OWNER) が指定されていた場合を除き、ステートメントの権限 ID は、実行時の権限 ID です。詳細は、60 ページの 『権限 ID と権限名』を参照してください。

構文





説明

ステートメント名

準備済みステートメントの名前を指定します。この名前に、既存の準備済みステートメントを指定すると、その準備済みステートメントは次の場合に破棄されます。

- そのステートメントが同じプログラムの同じインスタンス内で準備された場合。
- 両方のステートメントに関連した CRTSQLxxx コマンドに CLOSQLCSR(*ENDJOB)、CLOSQLCSR(*ENDACTGRP)、または CLOSQLCSR(*ENDSQL) が指定されている場合。

この名前に、プログラムの同じインスタンスの中のオープン・カーソルに関連する SELECT ステートメントを指定してはなりません。

INTO

INTO を使用すると、PREPARE ステートメントが正常に実行されたときに、準備済みステートメントに関する情報が、記述子名で指定した SQLDA 内に入ります。したがって、次の PREPARE ステートメントは、

```
EXEC SQL PREPARE S1 INTO :SQLDA FROM :V1;
```

上記のステートメントは、次のステートメントと同等です。

```
EXEC SQL PREPARE S1 FROM :V1;
EXEC SQL DESCRIBE S1 INTO :SQLDA;
```

記述子名

SQL 記述子域 (SQLDA) を識別します。SQLDA については、877 ページの 『付録 C. SQL 記述子域 (SQLDA)』で説明しています。PREPARE ステートメントを実行する前に、SQLDA に次の変数をセットしておく必要があります。 (REXX の規則とは異なります。詳細については、DB2 UDB for iSeries ホスト言語での SQL プログラミングを参照してください。)

SQLN

SQLVAR によって表される変数の個数を指定します。 (SQLN によって、SQLVAR 配列の大きさ (エレメント数) が指定されます。) SQLN は PREPARE ステートメントの実行に先立ってゼロよりも大きいか、または等しい値に設定しなければなりません。必要なオカレンスの数を決定する手法については、 880 ページの『必要な SQLVAR オカレンスの数の決定』 を参照してください。

SQLDA に入れられる情報の説明については、 642 ページの『DESCRIBE』を参照してください。

USING

SQLDA のそれぞれの SQLNAME 変数に、どのような値を割り当てるかを指定します。要求した値が存在しない場合、SQLNAME の長さは 0 にセットされます。

NAMES
列の名前を割り当てます。これは、デフォルト値です。準備されたステートメントで名前がその選択リストに明示的に指定されている場合、指定されたそれらの名前が戻されます。

SYSTEM NAMES
列のシステム列名を割り当てます。

LABELS
列のラベルを割り当てます。(列のラベルは、LABEL ステートメントによって定義されます。) ラベルの最初の 20 バイトだけが戻されます。

ANY
列のラベルを割り当てます。列がラベルを持たない場合、ラベルとして列名が使用されます。

BOTH
列のラベルと名前の両方を割り当てます。この場合、追加情報に応じるために、1 つの列ごとに SQLVAR の 2 ～ 3 つのオカレンスが必要になりますが、その数は、結果セットに特殊タイプが入っているか否かによって決まります。この拡張の SQLVAR 配列を指定するには、SQLN を 2*n* か 3*n* (この場合の *n* は、表やビュー内の列数) に設定します。SQLVAR の最初の *n* 個のオカレンスには、列の名前が入り、2 番目または 3 番目の *n* オカレンスには、列のラベルが含まれます。 特殊タイプがない場合、SQLVAR 項目の 2 番目のセットにそのラベルが戻されます。 特殊タイプがある場合には、SQLVAR 項目の 3 番目のセットにそのラベルが戻されます。

同じ SQLDA を以後の FETCH ステートメントで使用する場合には、その PREPARE が完了したあと、SQLN を *n* に設定してください。

ALL
ラベル、列名、およびシステム列名を割り当てます。この場合、追加情報に応じるために、1 つの列ごとに SQLVAR の 3 ～ 4 つのオカレンスが必要になりますが、その数は、結果セットに特殊タイプが入っているか否かによって決まります。この拡張の SQLVAR 配列を指定するには、SQLN を 3*n* か 4*n* (この場合の *n* は、結果表内の列数) に設定します。SQLVAR の最初の *n* オカレンスには、システム列名が入ります。 2 番目または 3 番目の *n* オカレンスには、列のラベルが含まれます。3 番目または 4 番目の *n* オカレンスには、列名が含まれます。特殊タイプが指定されていない場合、ラベルは、SQLVAR 記入項目の 2 番目のセット内に戻され、列名は、SQLVAR 記入項目の 3 番目のセット内に戻されます。それ以外の場合、ラベルは、SQLVAR 記入項目の 3 番目のセット内に戻され、列名は、SQLVAR 記入項目の 4 番目のセット内に戻されます。

同じ SQLDA を以後の FETCH ステートメントで使用する場合には、その PREPARE が完了したあと、SQLN を *n* に設定してください。

FROM
ステートメント・ストリングを指定します。このステートメント・ストリングは、指定したストリング式 またはホスト変数 の値です。

ストリング式
ストリング式 は、結果が文字ストリングになる PL/I のストリング式 です。文字ストリングを生み出す SQL 式は許されません。ストリング式は、PL/I でのみ許されます。

ホスト変数
ホスト変数を指定します。この変数は、文字ストリングまたは UCS-2 グラフィックのホスト変数を宣言する規則に従ってプログラム内で宣言されます。ホスト変数は、CLOB または DBCLOB データ・タイプを持っていてはならず、標識変数を指定してはなりません。

ステートメント・ストリングは、以下の SQL ステートメントのいずれかでなければなりません。

ALTER	GRANT	SAVEPOINT
CALL	HOLD LOCATOR	選択ステートメント
COMMENT	INSERT	SET PATH
COMMIT	LABEL	SET SCHEMA
CREATE	LOCK TABLE	SET TRANSACTION
DECLARE GLOBAL TEMPORARY TABLE	RELEASE SAVEPOINT	UPDATE
DELETE	RENAME	VALUES INTO
DROP	REVOKE	
FREE LOCATOR	ROLLBACK	

ステートメント・ストリングは、次のようなストリングであってはなりません。

- EXEC SQL で始まり、END-EXEC またはセミコロン (;) で終わるストリング。
- ホスト変数への参照を含むストリング。

パラメーター・マーカー

ステートメント・ストリングには、ホスト変数への参照を入れることはできないが、パラメーター・マーカー を含めることはできます。パラメーター・マーカーは、準備されたステートメントの実行時点で、ホスト変数の値によって置き換えられます。パラメーター・マーカーは疑問符 (?) で表し、そのステートメント・ストリングが静的 SQL ステートメントであった場合にホスト変数を使用することができる個所で使用します。パラメーター・マーカーが、値によってどのように置き換えられるかについては、 720 ページの『OPEN』および 667 ページの『EXECUTE』を参照してください。

パラメーター・マーカーには、次の 2 つのタイプがあります。

タイプ付きパラメーター・マーカー
ターゲット・データ・タイプが一緒に指定されているパラメーター・マーカー。その汎用形式は、次のとおりです。

```
CAST(? AS データ・タイプ)
```

この表記は関数呼び出しではありませんが、実行時のそのパラメーターのタイプは、指定のデータ・タイプになるか、指定のデータ・タイプに変換できるデータ・タイプになることを “約束” します。例えば、次の場合、

```
UPDATE EMPLOYEE
  SET LASTNAME = TRANSLATE(CAST(? AS VARCHAR(12)))
 WHERE EMPNO = ?
```

TRANSLATE 関数の引き数の値は、実行時に提供されます。その値のデータ・タイプは、VARCHAR(12)、あるいは VARCHAR(12) に変換できるデータ・タイプになります。詳細については、149 ページの『CAST の指定』を参照してください。

タイプ無しパラメーター・マーカー
ターゲット・データ・タイプが指定されていないパラメーター・マーカー。その形式は、単一の疑問符 (?) です。タイプ無しパラメーター・マーカーのデータ・タイプは、コンテキストによって提供されます。例えば、上記の更新ステートメントの述部にあるタイプ無しパラメーター・マーカーは、EMPNO 列のデータ・タイプと同じになります。

タイプ付きパラメーター・マーカーは、ホスト変数がサポートされており、データ・タイプが CAST 関数の約束に基づいていれば、動的 SQL ステートメント内のどこでも使用できます。

タイプ無しパラメーター・マーカーは、ホスト変数がサポートされている場合、動的 SQL ステートメント内の選択された場所で使用できます。使用する場所とその結果のデータ・タイプを、表 57 にまとめます。この表では、タイプ無しパラメーター・マーカーを適用できるかどうかが分かりやすいように、使用する場所は、式、述部、関数別にグループ分けしてあります。

表 57. タイプ無しパラメーター・マーカーの使用法

タイプ無しパラメーター・マーカーの場所	データ・タイプ
式 (選択リスト、CASE、VALUES を含む)	
副照会内でない選択リストで単独で使用	エラー
EXISTS 副照会内の選択リストで単独で使用	エラー
副照会内の選択リストで単独で使用	副照会の他のオペランドのデータ・タイプ。[62]
INSERT ステートメントの選択ステートメント内の選択リストで単独で使用	ターゲット表の関連の列のデータ・タイプ。[62]
単一算術演算子の両方のオペランド (演算子優先順位と演算順序の規則を考慮して) 次の場合も含まれます。 `? + ? + 10`	エラー
算術式 (日時式は除く) の単一演算子の一方のオペランド 次の場合も含まれます。 `? + ? * 10`	他方のオペランドのデータ・タイプ。
日時式のラベル付き期間 (ラベル付き期間のうち、単位のタイプを示す部分はパラメーター・マーカーにできません)	DECIMAL(15,0)
日時式のその他のオペランド (例えば、'timecol + ?' や '? - datecol')	エラー

表 57. タイプ無しパラメーター・マーカーの使用法 (続き)

タイプ無しパラメーター・マーカーの場所	データ・タイプ
CONCAT 演算子のオペランド	エラー
UPDATE ステートメントの SET 文節の右側の値として	列のデータ・タイプ。その列がユーザー定義特殊タイプとして定義されている場合は、ユーザー定義特殊タイプのソース・データ・タイプ。[62]
CASE 式の CASE キーワードの後の式	エラー
CASE 式 (単純および検索) 内の結果式の少なくとも 1 つ。残りの結果式は、タイプ無しパラメーター・マーカーか NULL のどちらかである場合。	エラー
単純 CASE 式内の WHEN の後の任意または全部の式	CASE の後の式と、タイプ無しパラメーター・マーカーでない WHEN の後の式に対して、 99 ページの『結果のデータ・タイプに関する規則』 を適用した結果。
CASE 式 (単純と検索の両方) 内の結果式。その式の少なくとも 1 つの結果式が、非 NULL で、タイプ無しパラメーター・マーカーでもない場合。	NULL またはタイプ無しパラメーター・マーカー以外のすべての結果式に対して 99 ページの『結果のデータ・タイプに関する規則』を適用した結果。
INSERT ステートメント以外の単一行 VALUES 文節で列式として単独で使用	エラー
INSERT ステートメント内の単一行 VALUES 文節で列式として単独で使用	列のデータ・タイプ。その列がユーザー定義特殊タイプとして定義されている場合は、ユーザー定義特殊タイプのソース・データ・タイプ。[62]
SET 特殊レジスター・ステートメントの右側の値として	特殊レジスターのデータ・タイプ。
VALUES INTO ステートメントの INTO 文節内の値として	関連式のデータ・タイプ。[62]
FREE LOCATOR または HOLD LOCATOR ステートメントの中の値として	ロケーター
述部	
比較演算子の両方のオペランド	エラー
比較演算子の一方のオペランド。他方のオペランドが、タイプ無しパラメーター・マーカーまたは特殊タイプ以外の場合。	他方のオペランドのデータ・タイプ。[62]
比較演算子の一方のオペランド。他方のオペランドが 特殊タイプ の場合。	エラー
BETWEEN 述部のすべてのオペランド	エラー
BETWEEN 述部の 2 つのオペランド (第 1 と第 2、第 1 と第 3 のいずれか)	唯一の非パラメーター・マーカーのデータ・タイプと同じ。
BETWEEN 述部の 1 つのみのオペランド	タイプ無しパラメーター・マーカー以外のすべてのオペランドに 99 ページの『結果のデータ・タイプに関する規則』 を適用した結果。ただし、CCSID 属性は、実行時に指定された値の CCSID になります。

表 57. タイプ無しパラメーター・マーカーの使用法 (続き)

タイプ無しパラメーター・マーカーの場所	データ・タイプ
IN 述部のすべてのオペランド。例えば、? IN (?,?,?)	エラー
IN 述部の第 1 オペランド。右側が副選択の場合 (例えば、? IN (副選択))	選択された列のデータ・タイプ。
IN 述部の第 1 オペランド。右側が副選択でない場合 (例えば、? IN (?,A,B) または ? IN (A,?,B,?))	IN リスト内の、タイプ無しパラメーター・マーカー以外のすべてのオペランド (IN キーワードの右側のオペランド) に対して 99 ページの『結果のデータ・タイプに関する規則』 を適用した結果。ただし、CCSID 属性は、実行時に指定された値の CCSID になります。
IN 述部の IN リストの任意または全部のオペランド (例えば、IN (?,B,?))	IN 述部の、タイプ無しパラメーター・マーカー以外のすべてのオペランド (IN 述部の左右のオペランド) に対して 99 ページの『結果のデータ・タイプに関する規則』 を適用した結果。ただし、CCSID 属性は、実行時に指定された値の CCSID になります。
LIKE 述部の 3 つのオペランドすべて	エラー
LIKE 述部の一致式	エラー
LIKE 述部のパターン式	一致式のデータ・タイプによって、VARCHAR(32740)、VARGRAPHIC(16370)、BLOB(32740) のいずれか。 パターンの値の固定長ホスト変数の使用法については、 161 ページの『LIKE 述部』 ページを参照してください。
LIKE 述部のエスケープ式	一致式のデータ・タイプによって、VARCHAR(1)、VARGRAPHIC(1)、または BLOB(1) のいずれか。
NULL 述部のオペランド	エラー
関数	
COALESCE、IFNULL、LAND、LOR、MIN、MAX、NULLIF、VALUE、または XOR のすべてのオペランド	エラー
COALESCE、IFNULL、LAND、LOR、MIN、MAX、NULLIF、VALUE、または XOR の任意のオペランド。少なくとも第 1 オペランドがタイプ無しパラメーター・マーカー以外の場合。	タイプ無しパラメーター・マーカー以外のすべてのオペランドに 99 ページの『結果のデータ・タイプに関する規則』 を適用した結果。
POSITION の第 1 オペランドまたは POSSTR の第 2 オペランド	他方のオペランドのデータ・タイプによって、VARCHAR(32740)、VARGRAPHIC(16370)、BLOB(32740) のいずれか。
他のすべてのスカラー関数 (ユーザー定義関数を含む) の他のすべてのオペランド	エラー
列関数のオペランド	エラー

使用上の注意

エラーの検査

PREPARE ステートメントが実行されると、ステートメント・ストリングが解析され、エラーがないか検査されます。ステートメント・ストリングが無効な場合は、準備済みステートメントは作成されず、準備済みステートメントを作成できない原因となったエラー条件が SQLCA に報告されます。

ローカルおよびリモート処理では、DLYPREP(*YES) オプションを指定すると、一部の SQL ステートメントで「遅延」エラーを受け取ることがあります。例えば、DESCRIBE、EXECUTE、および OPEN で、通常は PREPARE 処理中に出される SQLCODE を受け取ることがあります。

参照および実行の規則

準備済みステートメントは、以下のようなステートメントから参照できます (ただし、ステートメントによっては、参照できる準備済みステートメントが制約されることがあります)。

ステートメント	準備済みステートメントの制約事項
DESCRIBE	なし
DECLARE CURSOR	カーソルがオープンされているときは SELECT する必要がある。
EXECUTE	SELECT してはならない。

準備済みステートメントは、何度でも実行することができます。準備済みステートメントを一度しか実行せず、ステートメントの中でパラメーター・マーカーも使用しない場合は、PREPARE ステートメントと EXECUTE ステートメントを使用するより、EXECUTE IMMEDIATE ステートメントを使用した方が効率的です。

準備済みステートメントの永続性

準備済みステートメントはすべて、次の場合に破棄されます。[63]

- CONNECT (タイプ 1) ステートメントが実行された場合。
- 準備済みステートメントが関連する接続が、DISCONNECT ステートメントにより切り離された場合。
- 準備済みステートメントが解放保留の接続に関連し、正常なコミットが行われた場合。
- SQL ステートメントに関連した有効範囲 (ジョブ、活動化グループ、またはプログラム) が終了した場合。

ステートメントの有効範囲

ステートメント名 の有効範囲は、それが定義されているソース・プログラムです。準備済みステートメントを他の SQL ステートメントから参照できるのは、その SQL ステートメントが PREPARE ステートメントによってプリコンパイルされたも

62. データ・タイプが DATE、TIME、TIMESTAMP の場合は、VARCHAR(32740) が使用されます。

63. 準備済みステートメントは、キャッシュに入れて、実際に破棄しないことも可能です。ただし、キャッシュに入れたステートメントは、同一のステートメントを再度準備するときにしか使用できません。

のである場合だけです。例えば、別にコンパイルされた他のプログラムから呼び出されたプログラムは、呼び出しプログラムによって作成された準備済みステートメントを使用することができません。

また、ステートメント名の有効範囲は、そのステートメントを含むプログラムが実行しているスレッドに限定されます。例えば、同じジョブの中にある別々の 2 つのスレッドで同じプログラムが実行している場合、2 番目のスレッドは、最初のスレッドが準備したステートメントを使用することができません。

ステートメントが定義されているプログラムがそのステートメントの有効範囲ですが、そのプログラムから作成された各パッケージはそれぞれ準備されたステートメントの別個のインスタンスを含み、実行時に準備されたステートメントが複数存在することがあります。例えば、CONNECT (タイプ 2) ステートメントを使用して、次の順序でロケーション X とロケーション Y に接続するプログラムを想定します。

```
EXEC SQL CONNECT TO X;
EXEC SQL PREPARE S FROM :hv1;
EXEC SQL EXECUTE S;
.
.
.
EXEC SQL CONNECT TO Y;
EXEC SQL PREPARE S FROM :hv1;
EXEC SQL EXECUTE S;
```

S の 2 番目の準備は、Y で S の別個のインスタンスを準備します。

CRTSQLxxx コマンドに CLOSQLCSR(*ENDJOB)、CLOSQLCSR(*ENDACTGRP)、または CLOSQLCSR(*ENDSQL) が指定されていない場合、準備済みステートメントを参照できるのは、プログラム・スタックにあるプログラムの同一のインスタンスに制限されます。

- CLOSQLCSR(*ENDJOB) が指定されている場合、プログラム・スタックにあるそのプログラム (ステートメントを準備したプログラム) のどのインスタンスでも準備済みステートメントを参照できます。この場合、準備済みステートメントはジョブの終わりに破棄されます。
- CLOSQLCSR(*ENDSQL) が指定されている場合、プログラム・スタック内の最後の SQL プログラムが終了するまでの間、プログラム・スタックにあるそのプログラム (ステートメントを準備したプログラム) のどのインスタンスでも、準備済みステートメントを参照できます。この場合、準備済みステートメントはプログラム・スタックの最後の SQL プログラムが終了すると破棄されます。
- CLOSQLCSR(*ENDACTGRP) が指定されている場合、その活動化グループが終了するまでは、そのステートメントを準備したプログラムのモジュールのすべてのインスタンスで、その準備済みステートメントを参照できます。この場合、準備済みステートメントは活動化グループが終了すると破棄されます。

例

例 1

COBOL プログラムで、選択ステートメント以外のステートメントを準備して実行します。このステートメントは、ホスト変数 HOLDER に入っているものとします。プログラムでは、ユーザーからの何らかの指示に基づいて、ホスト変数

HOLDER にステートメント・ストリングを入れます。準備するステートメントには、パラメーター・マーカーは入っていません。

```
EXEC SQL  PREPARE STMT_NAME FROM :HOLDER  END-EXEC.

EXEC SQL  EXECUTE STMT_NAME  END-EXEC.
```

例 2

例 1 と同様に、選択ステートメント以外のステートメントを準備して実行しますが、準備するステートメントには、任意の数のパラメーター・マーカーを含めることができるものとします。

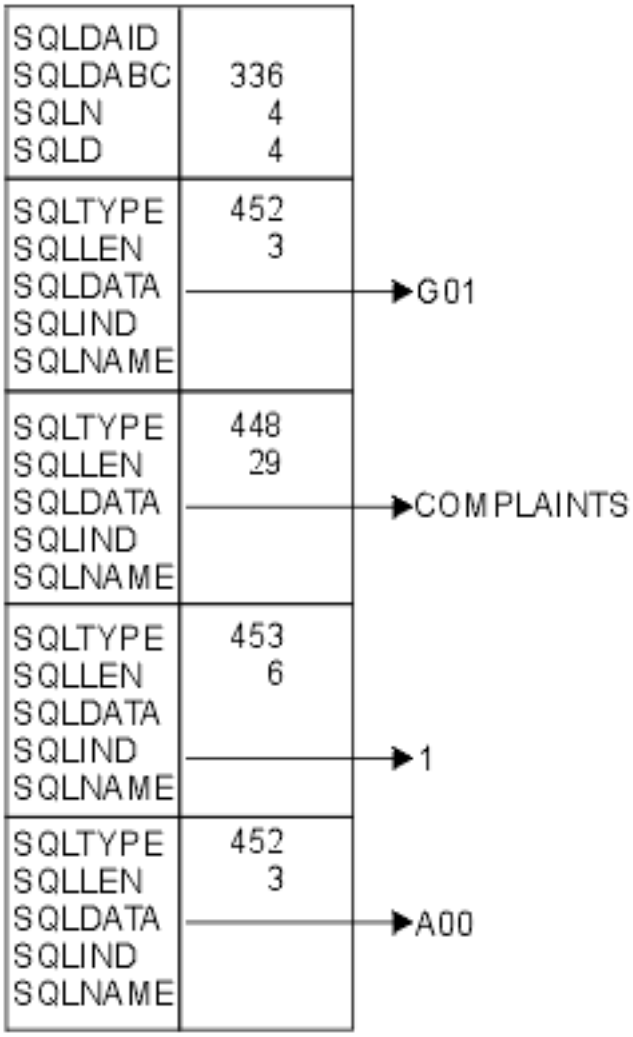
```
EXEC SQL  PREPARE STMT_NAME FROM :HOLDER  END-EXEC.

EXEC SQL  EXECUTE STMT_NAME USING DESCRIPTOR :INSERT_DA  END-EXEC.
```

以下のステートメントを準備するものとします。

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

部門番号 G01、部門名 COMPLAINTS、管理者なし、報告先部門は部門 A00 という行を挿入するとすれば、EXECUTE ステートメントを実行する前に、構造体 INSERT_DA には次のような値が入っていなければなりません。


SQLDAID
SQLDABC
SQLN
SQLD
336
4
4
SQLTYPE
SQLLEN
SQLDATA
SQLIND
SQLNAME
452
3
G01
SQLTYPE
SQLLEN
SQLDATA
SQLIND
SQLNAME
448
29
COMPLAINTS
SQLTYPE
SQLLEN
SQLDATA
SQLIND
SQLNAME
453
6
1
SQLTYPE
SQLLEN
SQLDATA
SQLIND
SQLNAME
452
3
A00

RELEASE

RELEASE ステートメントは、1 つまたは複数の接続を解放保留状態にします。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことと、対話式に呼び出すことだけが可能です。このステートメントは、動的には準備できない実行可能ステートメントです。Java または REXX では指定できません。

RELEASE はトリガーでは使用できません。リモート・サーバーで外部プロシージャーを呼び出す場合、その外部プロシージャーでは RELEASE は使用できません。

権限

権限は不要です。

構文

説明

サーバー名 または ホスト変数
指定したサーバー名、または指定したホスト変数に入っているサーバー名によってサーバーを識別します。ホスト変数を指定する場合、

- その変数は、文字ストリング変数でなければなりません。
- 標識変数を伴っていてはなりません。
- サーバー名は、その変数内で左寄せし、通常 ID の形成の規則に従っていなければなりません。
- サーバー名の長さが、ホスト変数の長さよりも短い場合、右側をブランクで埋めなければなりません。

RELEASE ステートメントが実行される時点で、指定したサーバー名、または指定のホスト変数に入っているサーバー名は、活動化グループの既存の接続を識別していなければなりません。

CURRENT
活動化グループの現行接続を識別します。活動化グループは接続状態でなければなりません。

ALL または **ALL SQL**
活動化グループの既存のすべての接続 (ローカルおよびリモートの接続の両方) を識別します。

このステートメントの実行時に接続が存在しない場合、エラーや警告は起こりません。

RELEASE ステートメントが正常に実行された場合は、識別されている各接続は解放保留状態になり、したがって、次のコミット操作中に終了することになります。RELEASE ステートメントが不成功の場合には、その活動化グループの接続状態およびその接続の状態は変わりません。

使用上の注意

CONNECT (タイプ 1) の使用は、RELEASE の使用を妨げることはありません。

RELEASE は、カーソルをクローズしません。また、どのようなリソースも解放しません。さらに、該当の接続をさらに使用するのを妨げることはありません。

ROLLBACK は、接続の状態を解放保留から保留にリセットすることはありません。

リモートの接続を作成し、維持するにはリソースが必要になります。したがって、再使用の予定がないリモート接続は、解放保留状態にする必要があり、再使用の予定があるリモート接続は、解放保留状態にしてはなりません。

コミット操作が行われる時点で現行接続が解放保留状態にある場合は、その接続は終了し、その活動化グループは未接続状態になります。この場合は、次に実行される SQL ステートメントは、CONNECT または SET CONNECTION であることが必要です。

RELEASE ALL は、ローカル・サーバーとの接続を解放保留状態にします。解放保留状態の接続は、WITH HOLD 文節を指定して定義したオープン・カーソルを持つ場合でも、コミット操作中に終了します。

例

例 1 : 次の作業単位では、TOROLAB1 との接続は必要としません。次のステートメントは、次のコミット操作の過程で既存の接続を終了させます。

```
EXEC SQL RELEASE TOROLAB1;
```

例 2 : 次の作業単位では、現行接続は必要としません。次のステートメントは、次のコミット操作の過程で既存の接続を終了させます。

```
EXEC SQL  RELEASE CURRENT;
```

例 3 : 次の作業単位では、既存の接続はいずれも必要としません。次のステートメントは、次のコミット操作の過程で既存の接続を終了させます。

```
EXEC SQL RELEASE ALL;
```

RELEASE SAVEPOINT

RELEASE SAVEPOINT ステートメントは、1 つの作業単位内で、指定された保管ポイントとそれ以降に確立されたすべての保管ポイントを解放します。

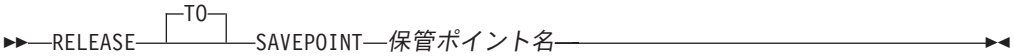
呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。このステートメントは、動的に準備できる実行可能ステートメントです。

権限

権限は不要です。

構文



説明

保管ポイント名
解放する保管ポイントを指定します。指定した名前の保管ポイントが存在しない場合は、エラーが起こります。指定した保管ポイントと、この作業単位内でそれ以降に確立されているすべての保管ポイントが解放されます。解放された後は、その保管ポイントは維持されないので、その保管ポイントまでのロールバックはできなくなります。

使用上の注意

解放した保管ポイントの名前は、別の SAVEPOINT ステートメントで再使用することができます。同じ保管ポイント名が指定されている前の SAVEPOINT ステートメントで、UNIQUE キーワードが指定されていても構いません。

対象の活動化グループについてコミットメント制御が活動状態にない場合は、RELEASE SAVEPOINT ステートメントは使用できません。どのコミットメント定義が使用されているかを判別する方法については、423 ページの『使用上の注意』を参照してください。

例

あるメイン・ルーチンが、保管ポイント A を設定した後で、保管ポイント B および C を設定するサブルーチンを呼び出すものとします。メイン・ルーチンに制御が戻ると、メイン・ルーチンは、保管ポイント A とそれ以降に設定されたすべての保管ポイントを解放します。つまり、A のほかに、サブルーチンが設定した保管ポイント B および C が解放されます。

```
RELEASE SAVEPOINT A
```

RENAME

RENAME ステートメントは、表、ビュー、または索引の名前を変更します。表、ビュー、または索引の名前またはシステム・オブジェクト名 (あるいは、その両方) を変更できます。

呼び出し

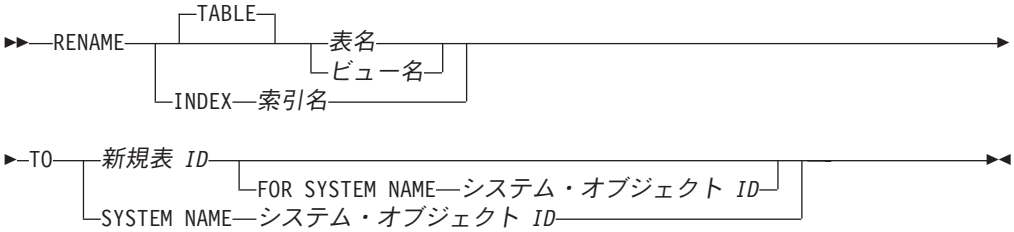
このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。このステートメントは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- 次のシステム権限
  - オブジェクト名を変更する場合、
    - 名前を変更する表、ビュー、または索引に対する *OBJMGT システム権限。
    - 名前を変更する表、ビュー、または索引が入っているライブラリーに対する *EXECUTE システム権限。
  - オブジェクトのシステム名を変更する場合、
    - 名前を変更する表、ビュー、または索引に対する *OBJMGT システム権限。
    - 名前を変更する表、ビュー、または索引が入っているライブラリーに対する *EXECUTE および *UPD システム権限。
- 管理権限。

構文



説明

TABLE *表名*または *ビュー名*

名前の変更をする表またはビューを示します。*表名* またはビュー名 は、現行サーバーにある表またはビューを示すものでなければなりません。しかし、カタログ表またはグローバル一時表を示すものであってはなりません。指定された名前が別名であっても構いません。指定した表またはビューは、新しい名前に変更さ

れます。その表あるいはビューに関する特権、制約、索引、トリガー、ビュー、および論理ファイルはすべてそのままの状態に保持されます。

該当の表あるいはビューを参照するアクセス・プランはいずれも、そのアクセス・プランを使用するプログラムが次回実行される時に再度暗黙的に準備されます。プログラムは元の名前で表あるいはビューを参照するので、その時に元の名前の表あるいはビューが存在しない場合、SQLCA の SQLCODE フィールドには負の値が戻されます。

INDEX *索引名*
名前を変更する索引を示します。この*索引名* は、現行サーバーに存在している索引を示すものでなければなりません。指定した索引は、新しい名前に変更されます。

この索引を参照するアクセス・プランは、名前変更によって影響は受けません。

新規表 ID
それぞれ、表、ビュー、索引の新しい表名、ビュー名、*索引名* を示します。*新規表 ID* は、現行サーバーにすでに存在する表、ビュー、別名、または索引と同一であってはなりません。*新規表 ID* は、非修飾 SQL ID でなければなりません。

SYSTEM NAME *システム・オブジェクト ID*
それぞれ、表、ビュー、索引の、新しいシステム・オブジェクト *ID* を示します。システム・オブジェクト *ID* は、現行サーバーにすでに存在する表、ビュー、別名、または索引と同一であってはなりません。システム・オブジェクト *ID* は、非修飾システム ID でなければなりません。

オブジェクトの名前とオブジェクトのシステム名が同一であり、*名前* が指定されていない場合、システム・オブジェクト *ID* を指定すると、それが新規の名前およびシステム・オブジェクト名になります。それ以外の場合には、システム・オブジェクト *ID* の指定は、オブジェクトのシステム名だけに影響を与え、オブジェクトの名前には影響を与えません。

新規表 ID とシステム・オブジェクト *ID* の両方が指定されている場合、両方を有効なシステム・オブジェクト名にすることはできません。

使用上の注意

名前変更の操作は、指定された新しい名前に応じて実行されます。

- 新しい名前が有効なシステム ID の場合、
  - 代替名がある場合は、それが除去されます。
  - システム・オブジェクト名は、新しい名前に変更されます。
- 新しい名前が有効な ID でない場合、
  - 代替名が追加されるか、新しい名前に変更されます。
  - システム・オブジェクト名 (表またはビューの) が、名前を変更する表、ビューまたは索引として指定された場合、新しいシステム・オブジェクト名が生成されます。表名の生成規則についての詳細は、572 ページの 『表名の生成の規則』を参照してください。

表名 の別名が指定されている場合は、その別名は現行サーバーに存在していなければならず、そしてその別名により識別される表が名前変更されます。その別名自体

の名前は変更されないので、名前変更の後も引き続き古い表を参照することになります。別名の名前を変更するためのサポートはありません。

例

例 1

MY_IN_TRAY という名前の表を MY_IN_TRAY_94 に変更します。システム・オブジェクト名は、そのまま変更されません (MY_IN_TRAY)。

```
RENAME TABLE MY_IN_TRAY TO MY_IN_TRAY_94
  FOR SYSTEM NAME MY_IN_TRAY
```

例 2

MA_PROJ という名前の表を、MA_PROJ_94 に変更します。

```
RENAME TABLE MA_PROJ
  TO SYSTEM NAME MA_PROJ_94
```

REVOKE (特殊タイプ特権)

この形式の REVOKE ステートメントは、特殊タイプに対する特権を除去します。

呼び出し

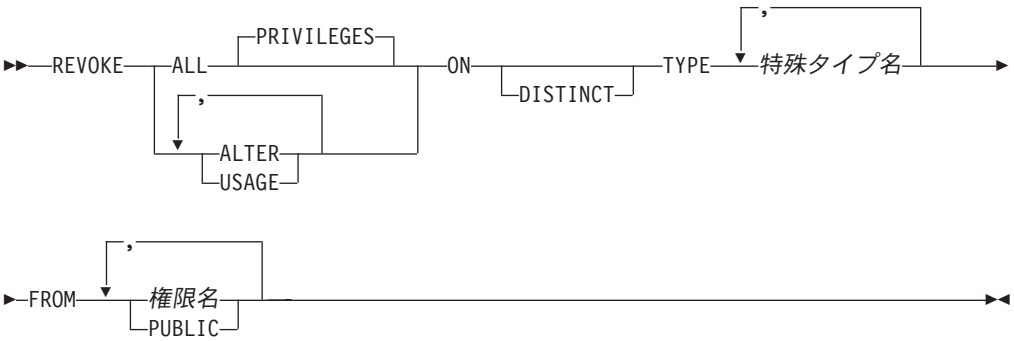
このステートメントは、アプリケーション・プログラムに組み込むことも、対話式に呼び出すこともできます。このステートメントは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメント内に識別されているそれぞれの 特殊タイプ に対しては次のもの。
  - このステートメントで指定されるすべての特権
  - その 特殊タイプ に対する *OBJMGT システム権限
  - その 特殊タイプ が入っているライブラリーに対する *EXECUTE システム権限
- 管理権限

構文



説明

ALL または **ALL PRIVILEGES**

各*権限名* から 1 つまたは複数の特殊タイプ特権を取り消します。取り消される特権は、識別された特殊タイプに関して、*権限名* に認可されていた特権です。特殊タイプ に対する ALL PRIVILEGES を取り消すのは、*ALL システム権限を取り消すのと同じではないことに注意する必要があります。

ALL を使用しない場合には、以下にリストしたキーワードの 1 つまたは複数を使用する必要があります。各キーワードは、そこで説明されている特権を取り消します。

ALTER

COMMENT ステートメントを使用するための特権を取り消します。

USAGE
表、関数、プロシージャー内で、あるいは CREATE DISTINCT TYPE ステートメントの中のソース・タイプとして特殊タイプを使用する特権を取り消します。

ON DISTINCT TYPE *特殊タイプ名*
特権を取り消したい特殊タイプを指定します。 *特殊タイプ名*は、現行サーバーに存在する 特殊タイプを示すものでなければなりません。

FROM
特権を取り消すユーザーを識別します。

権限名、...
1 つまたは複数の権限 ID をリストします。同じ*権限名* を複数回指定することはできません。

PUBLIC
指定した特権を、PUBLIC (共通認可) から取り消します。

使用上の注意

特殊タイプに対する特権を取り消した場合は、どのユーザーが認可を行ったかには関係なく、その特殊タイプに対する特権の認可はすべて無効になります。

特殊タイプ 特権を取り消すと、対応するシステム権限が取り消されます。 SQL 特権に対応するシステム権限の説明については、681 ページの 『GRANT (特殊タイプ特権)』を参照してください。

同義のキーワード：以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- キーワード DATA を DISTINCT の同義語として使用することができます。

例

特殊タイプ SHOESIZE に関する USAGE 特権を、ユーザー JONES から取り消します。

```
REVOKE USAGE
  ON DISTINCT TYPE SHOESIZE
  FROM JONES
```

REVOKE (関数またはプロシージャー特権)

この形式の REVOKE ステートメントは、関数またはプロシージャーに対する特権を除去します。

呼び出し

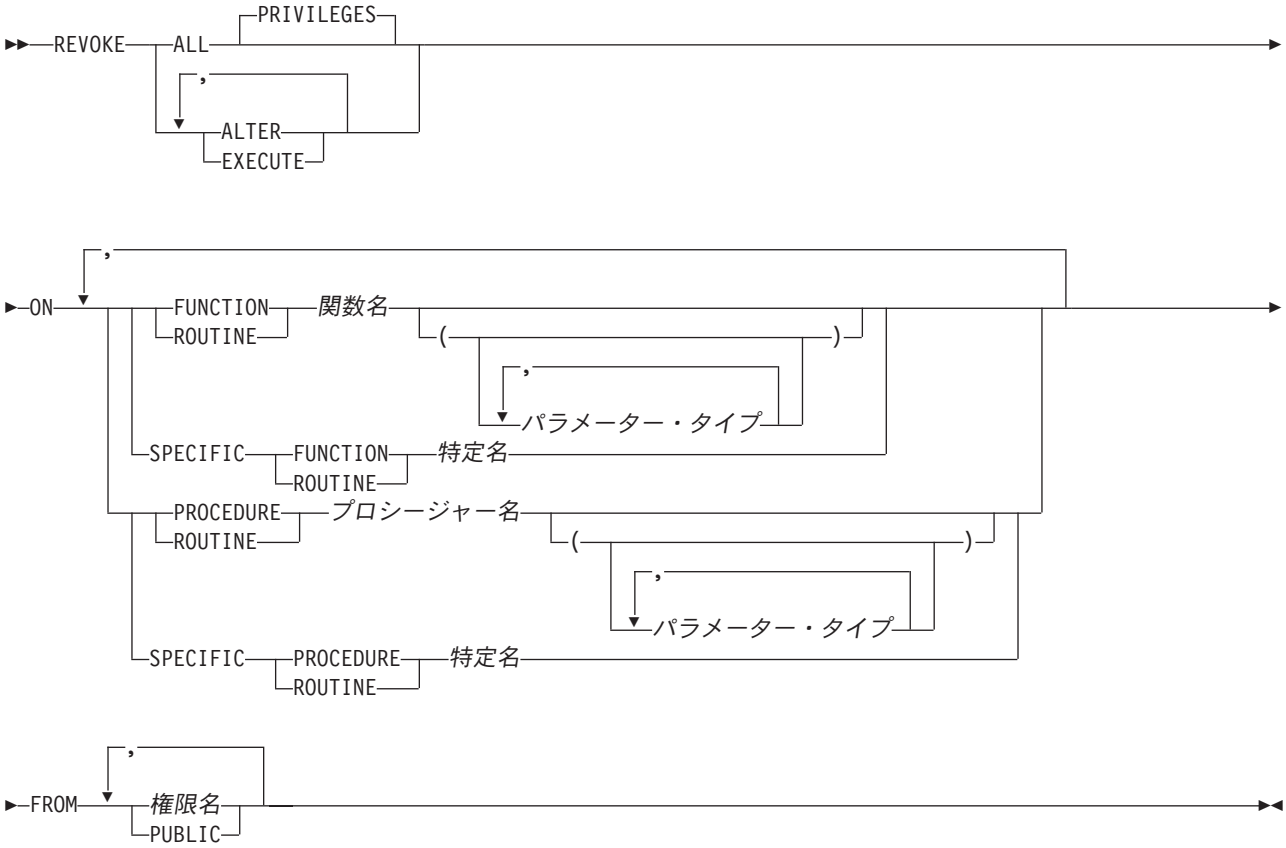
このステートメントは、アプリケーション・プログラムに組み込むことも、対話式に呼び出すこともできます。このステートメントは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメント内で識別された、それぞれの関数またはプロシージャーごとに、
  - このステートメントで指定されるすべての特権
  - その関数またはプロシージャーに対する *OBJMGT システム権限
  - その関数またはプロシージャーが入っているライブラリー (これが Java ルーチンの場合は、ディレクトリー) に対する *EXECUTE システム権限
- 管理権限

構文

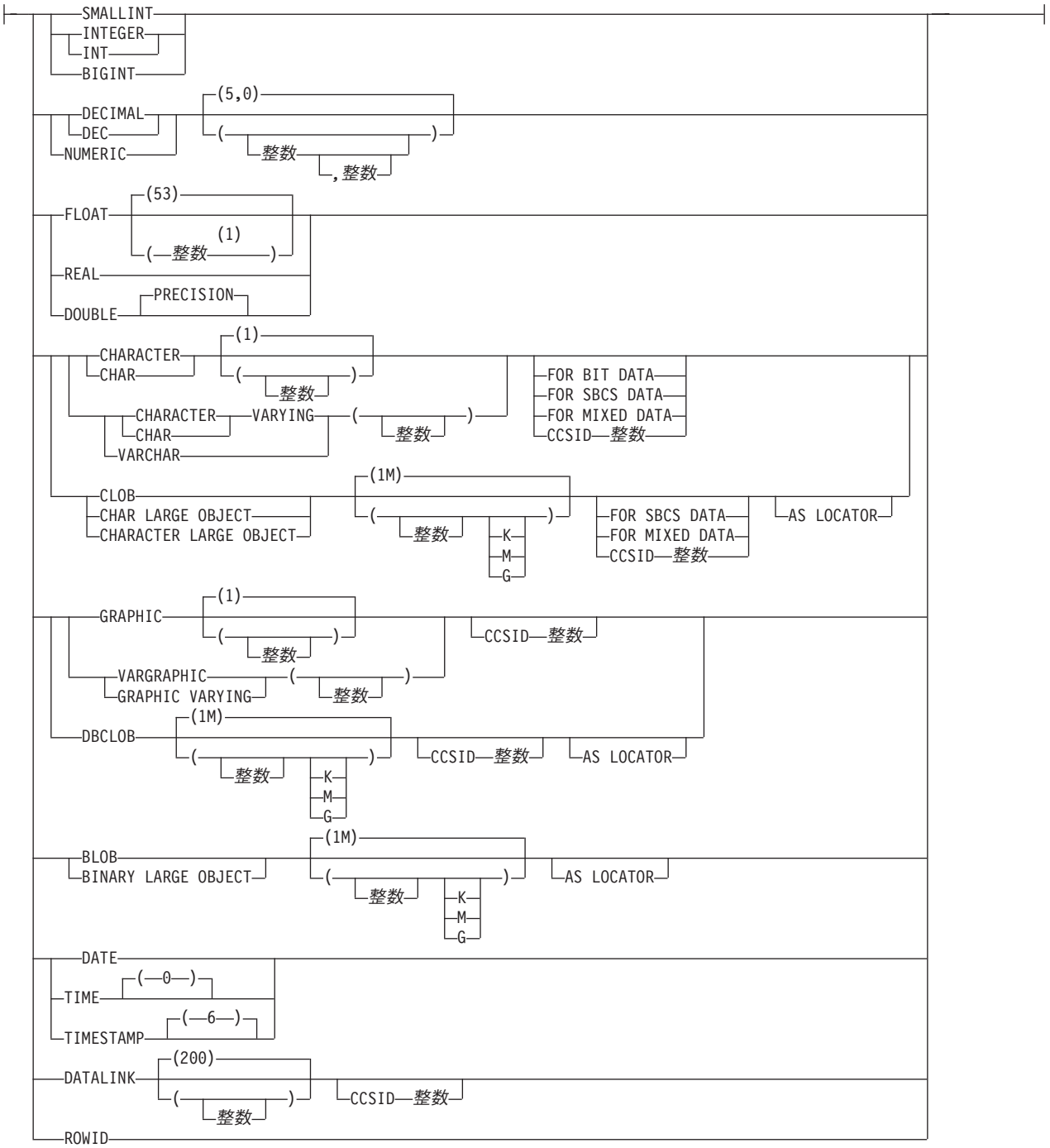

REVOKE
ALL
PRIVILEGES
ALTER
EXECUTE
ON
FUNCTION
ROUTINE
関数名
パラメーター・タイプ
SPECIFIC
FUNCTION
ROUTINE
特定名
PROCEDURE
ROUTINE
プロシージャー名
パラメーター・タイプ
SPECIFIC
PROCEDURE
ROUTINE
特定名
FROM
権限名
PUBLIC

パラメーター・タイプ:


組み込みタイプ
特殊タイプ名
AS LOCATOR

REVOKE (関数またはプロシージャー特権)

組み込みタイプ:



注:

1 突き合わせは、データ・タイプ (REAL または DOUBLE) に基づいて行われるので、精度に指定された値は、関数の作成時に 指定された値に一致している必要はありません。

説明

ALL または **ALL PRIVILEGES**
各*権限名* から 1 つまたは複数の関数またはプロシージャー特権を取り消します。取り消される特権は、識別された関数またはプロシージャーに関して、*権限名* に認可されていた特権です。関数またはプロシージャーに対する ALL PRIVILEGES を取り消すのは、*ALL システム権限を取り消すのと同じではないことに注意する必要があります。

ALL を使用しない場合には、以下にリストしたキーワードの 1 つまたは複数を使用する必要があります。各キーワードは、そこで説明されている特権を取り消します。

ALTER
COMMENT ステートメントを使用するための特権を取り消します。

EXECUTE
関数またはプロシージャーを実行するための特権を取り消します。

FUNCTION
特権を取り消す対象の関数を識別します。関数は、それぞれその名前、関数シグニチャー、あるいは特定名によって識別することができます。 関数解決の規則 (およびパス) は使用されません。

FUNCTION *関数名*
関数名 では、現行サーバーに存在している厳密に 1 つの関数を識別する必要があります。この関数には、パラメーターをいくつでも定義することができます。指定されたスキーマまたは暗黙のスキーマの中に、指定された名前の関数が複数ある場合、エラーが戻されます。

FUNCTION *関数名 (パラメーター・タイプ, ...)*
関数名 (パラメーター・タイプ, ...) は、現行サーバーに存在していて、指定された関数シグニチャーを持つ関数を識別する必要があります。 指定されたパラメーターは、CREATE FUNCTION ステートメント上の対応する位置に指定されたデータ・タイプと一致していなければなりません。取り消す関数インスタンスを識別する場合、データ・タイプの数とデータ・タイプの論理連結が使用されます。*関数名 ()* を指定する場合、識別される関数にパラメーターを使用することはできません。

関数名
関数の名前を識別します。

(パラメーター・タイプ, ...)
関数のパラメーターを識別します。

非修飾の特殊タイプ名を指定する場合、データベース・マネージャーはその特殊タイプのスキーマ名を解決するための SQL パスを検索します。

長さ属性、精度属性、あるいは位取り属性があるデータ・タイプの場合、値を指定することも、1 組の空の括弧を使用することもできます。

- 中が空の括弧は、データ・タイプが一致しているか否かの判別時にデータベース・マネージャー によって属性が無視されることを示します。

- 長さ属性、精度属性、あるいは位取り属性に特定の値を使用する場合、その値は、CREATE FUNCTION ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。
- 長さ属性、精度属性、または位取り属性が明示的に指定されておらず、空の括弧も指定されていない場合、該当のデータ・タイプのデフォルト属性が暗黙指定されます。例えば以下のようになります。

```
CHAR      CHAR(1)
GRAPHIC   GRAPHIC(1)
DECIMAL   DECIMAL(5,0)
FLOAT     DOUBLE (length of 8)
```

暗黙の長さは、CREATE FUNCTION ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。データ・タイプのデフォルト長さの全リストは、542 ページの『CREATE TABLE』 を参照してください。

サブタイプまたは CCSID 属性のあるデータ・タイプの場合は、FOR DATA 文節または CCSID 文節の指定は任意選択です。どちらか一方の文節を省略すると、データ・タイプが一致しているか否かの判別時にデータベース・マネージャー によって属性が無視されることが指示されます。 どちらか一方の文節を指定する場合は、CREATE FUNCTION ステートメントに暗黙的または明示的に指定されている値と一致させる必要があります。

SPECIFIC FUNCTION *特定名*

この*特定名* は、現行サーバーに存在している特定の関数を識別していなければなりません。

PROCEDURE

特権を取り消す対象のプロシージャーを識別します。プロシージャーは、それぞれその名前、プロシージャー・シグニチャー、あるいは特定名によって識別することができます。 プロシージャー解決の規則 (およびパス) は使用されません。

PROCEDURE *プロシージャー名*

プロシージャー名 は、現行サーバーに存在しているただ 1 つのプロシージャーを識別していなければなりません。このプロシージャーには、パラメーターをいくつでも定義することができます。指定されたスキーマまたは暗黙のスキーマの中に、指定された名前のプロシージャーが複数ある場合、エラーが戻されます。

PROCEDURE *プロシージャー名 (パラメーター・タイプ, ...)*

プロシージャー名 (パラメーター・タイプ, ...) では、現行サーバーに存在していて、指定されたプロシージャー・シグニチャーを持つプロシージャーを識別する必要があります。指定されたパラメーターは、CREATE PROCEDURE ステートメント上の対応する位置に指定されたデータ・タイプと一致していなければなりません。除去するプロシージャー・インスタンスを識別する場合、データ・タイプの数とデータ・タイプの論理連結が使用されます。*プロシージャー名 ()* を指定する場合、識別されるプロシージャーにパラメーターを使用することはできません。

プロシージャー名

プロシージャーの名前を識別します。

(パラメーター・タイプ, ...)
プロシージャーのパラメーターを識別します。

非修飾の特殊タイプ名を指定する場合、データベース・マネージャーはその特殊タイプのスキーマ名を解決するための SQL パスを検索します。

長さ属性、精度属性、あるいは位取り属性があるデータ・タイプの場合、値を指定することも、1 組の空の括弧を使用することもできます。

- 中が空の括弧は、データ・タイプが一致しているか否かの判別時にデータベース・マネージャーによって属性が無視されることを示します。
- 長さ属性、精度属性、あるいは位取り属性に特定の値を使用する場合、その値は、CREATE PROCEDURE ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。
- 長さ属性、精度属性、または位取り属性が明示的に指定されておらず、空の括弧も指定されていない場合、該当のデータ・タイプのデフォルト属性が暗黙指定されます。例えば以下のようになります。

```
CHAR      CHAR(1)
GRAPHIC   GRAPHIC(1)
DECIMAL   DECIMAL(5,0)
FLOAT     DOUBLE (length of 8)
```

暗黙の長さは、CREATE PROCEDURE ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。データ・タイプのデフォルト長さの全リストは、542 ページの『CREATE TABLE』 を参照してください。

サブタイプまたは CCSID 属性のあるデータ・タイプの場合は、FOR DATA 文節または CCSID 文節の指定は任意選択です。どちらか一方の文節を省略すると、データ・タイプが一致しているか否かの判別時にデータベース・マネージャーによって属性が無視されることが指示されます。 どちらか一方の文節を指定する場合は、CREATE PROCEDURE ステートメントに暗黙的または明示的に指定されている値と一致させる必要があります。

SPECIFIC PROCEDURE *特定名*
特定名 は、現行サーバーに存在している特定のプロシージャーを識別していなければなりません。

FROM
特権を取り消すユーザーを識別します。

権限名,...
1 つまたは複数の権限 ID をリストします。同じ*権限名* を複数回指定することはできません。

PUBLIC
指定した特権を、PUBLIC (共通認可) から取り消します。

使用上の注意

関数またはプロシージャーに対する特権を取り消した場合は、どのユーザーが認可を行ったかには関係なく、その関数またはプロシージャーに対する特権の認可はすべて無効になります。

SQL、または外部関数か外部プロシージャーに関して取り消された特権は、その関連のプログラム (*PGM) オブジェクトまたはサービス・プログラム (*SRVPGM) オブジェクトについて取り消されます。

関数またはプロシージャー特権を取り消すと、対応するシステム権限が取り消されます。SQL 特権に対応するシステム権限の説明は、684 ページの 『GRANT (関数またはプロシージャー特権)』を参照してください。

同義のキーワード : 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- EXECUTE の同義語としてキーワード RUN を使用することもできます。

例

PUBLIC に対するプロシージャー CORPDATA.PROCA に関する EXECUTE 特権を取り消します。

```
REVOKE EXECUTE
  ON PROCEDURE CORPDATA.PROCA
  FROM PUBLIC
```

REVOKE (パッケージ特権)

この形式の REVOKE ステートメントは、パッケージに対する特権を除去します。

呼び出し

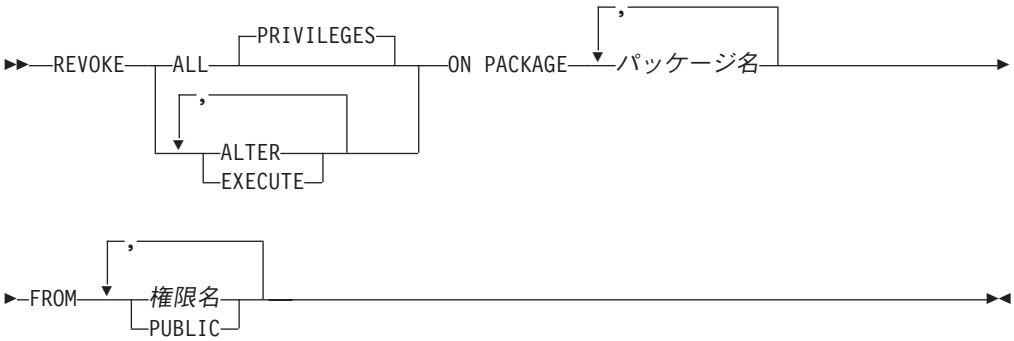
このステートメントは、アプリケーション・プログラムに組み込むことも、対話式に呼び出すこともできます。このステートメントは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメント内で識別された、それぞれのパッケージごとに、
  - このステートメントで指定されるすべての特権
  - パッケージに対する *OBJMGT システム権限
  - パッケージが入っているライブラリーについての *EXECUTE システム権限
- 管理権限

構文



説明

ALL または **ALL PRIVILEGES**
各*権限名* から 1 つまたは複数のパッケージ特権を取り消します。取り消される特権は、識別されたパッケージに関して、*権限名* に認可されていた特権です。パッケージに対する ALL PRIVILEGES を取り消すのは、*ALL システム権限を取り消すのと同じではないことに注意する必要があります。

ALL を使用しない場合には、以下にリストしたキーワードの 1 つまたは複数を使用する必要があります。各キーワードは、そこで説明されている特権を取り消します。

ALTER
COMMENT および LABEL ステートメントを使用する特権を取り消します。

EXECUTE
パッケージ中のステートメントを実行する特権を取り消します。

ON PACKAGE *パッケージ名*
特権を取り消したいパッケージを指定します。このパッケージ名 は、現行サーバーに存在しているパッケージを識別していなければなりません。

FROM
特権を取り消すユーザーを識別します。

権限名,...
1 つまたは複数の権限 ID をリストします。同じ*権限名* を複数回指定することはできません。

PUBLIC
指定した特権を、PUBLIC (共通認可) から取り消します。

使用上の注意

パッケージに対する特権を取り消した場合は、どのユーザーが認可を行ったかには関係なく、そのパッケージに対する特権の認可はすべて無効になります。

パッケージ特権を取り消すと、対応するシステム権限が取り消されます。SQL 特権に対応するシステム権限の説明については、692 ページの 『GRANT (パッケージ特権)』を参照してください。

同義のキーワード : 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- EXECUTE の同義語としてキーワード RUN を使用することができます。
- PACKAGE の同義語として、キーワード PROGRAM を使用することができます。

例

PUBLIC から、パッケージ CORPDATA.PKGA に関する EXECUTE 特権を取り消します。

```
REVOKE EXECUTE
  ON PACKAGE CORPDATA.PKGA
  FROM PUBLIC
```

REVOKE (表特権)

この形式の REVOKE ステートメントは、表またはビューに対する特権を除去します。

呼び出し

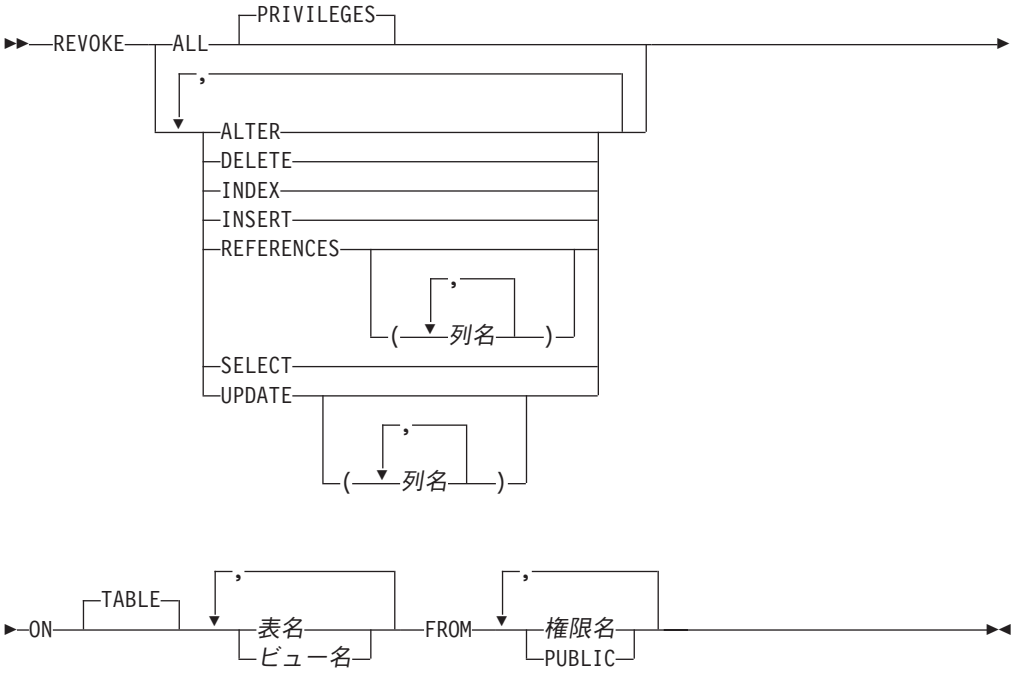
このステートメントは、アプリケーション・プログラムに組み込むことも、対話式に呼び出すこともできます。このステートメントは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメント内で識別された、それぞれの表またはビューごとに、
  - このステートメントで指定されるすべての特権
  - その表またはビューに対する *OBJMGT システム権限
  - 表やビューが入っているライブラリーに対する *EXECUTE システム権限
- 管理権限

構文



説明

ALL または **ALL PRIVILEGES**

各*権限名* から 1 つまたは複数の特権を取り消します。取り消される特権は、識別された表およびビューに関して、*権限名* に認可されていた特権です。表また

はビューに対する ALL PRIVILEGES を取り消すのは、*ALL システム権限を取り消すのと同じではないことに注意する必要があります。

ALL を使用しない場合には、以下にリストしたキーワードの 1 つまたは複数を使用する必要があります。各キーワードはそこで説明されている特権を取り消しますが、ON 文節で指定された表およびビューに当てはまる特権だけが取り消されます。

ALTER
表に対して ALTER TABLE ステートメントを使用する特権を取り消します。表およびビューに対して、COMMENT および LABEL ステートメントを使用する特権を取り消します。

DELETE
DELETE ステートメントを使用する特権を取り消します。

INDEX
CREATE INDEX ステートメントを使用する特権を取り消します。

INSERT
INSERT ステートメントを使用する特権を取り消します。

REFERENCES
その表が親になる参照制約を追加する特権を取り消します。

REFERENCES (*列名*,...)
親キーで指定された列を使用して参照制約を追加する特権を取り消します。それぞれの列名は、ON 文節に指定されている各表の列を識別する非修飾の名前でなければなりません。

SELECT
SELECT または CREATE VIEW ステートメントを使用する特権を取り消します。

UPDATE
UPDATE ステートメントを使用する特権を取り消します。

UPDATE (*列名*,...)
指定された列を更新する特権を取り消します。それぞれの列名は、ON 文節に指定されている各表の列を識別する非修飾の名前でなければなりません。

ON *表名* またはビュー名 ,...
特権を取り消す対象の表またはビューを識別します。*表名* またはビュー名 は、現行サーバーにある表またはビューを示すものでなければなりませんが、グローバル一時表を示すものであってはなりません。

FROM
特権を取り消すユーザーを識別します。

権限名、...
1 つまたは複数の権限 ID をリストします。同じ*権限名* は、複数回指定してはなりません。

PUBLIC
指定した特権を、PUBLIC (共通認可) から取り消します。

使用上の注意

システム権限

INDEX または ALTER 特権のいずれかを取り消すと、システム権限 *OBJALTER が取り消されます。

表またはビュー特権を取り消すと、対応するシステム権限が取り消されます。ただし、以下の例外があります。

- 表またはビューに対する特権を取り消した際に、*OBJOPR が取り消されるのは、*ADD、*DLT、*READ、および *UPD もすべて取り消された場合だけです。
- ビューに対する権限を取り消す場合、そのビューの定義の副選択で参照されている、どのような表やビューからも権限は取り消されません。

複数のシステム権限を 1 つの SQL 特権の取り消しで取り消す場合、それらのシステム権限の中に 1 つでも取り消すことができないものがあると、警告が出され、その特権の取り消しでは権限は取り消されません。

SQL 特権に対応するシステム権限の説明については、 695 ページの『GRANT (表特権)』を参照してください。

複数回の認可

同じ特権が同じユーザーに対して複数回認可されている場合は、そのユーザーからその特権を取り消すと、それらの認可はすべて無効になります。

ある特権を取り消すと、その特権がどのようなユーザーに認可されているかには関係なく、その特権の認可がすべて取り消されます。

WITH GRANT OPTION を取り消す唯一の方法は、ALL を指定して取り消すことです。

例

例 1

表 EMPLOYEE に対する SELECT 特権を、ユーザー PULASKI から取り消します。

```
REVOKE SELECT
  ON EMPLOYEE
  FROM PULASKI
```

例 2

以前にはすべてのローカル・ユーザーに認可していた表 EMPLOYEE に対する更新特権を取り消します。 特定のユーザーに対する認可には、影響を与えないことに注意してください。

```
REVOKE UPDATE
  ON EMPLOYEE
  FROM PUBLIC
```

例 3

表 EMPLOYEE に対するすべての特権を、ユーザー KWAN および THOMPSON から取り消します。

```
REVOKE ALL
  ON EMPLOYEE
  FROM KWAN,THOMPSON
```

例 4

VIEW1 の column_1 を更新する特権を、FRED から取り消します。

```
REVOKE UPDATE(column_1)
  ON VIEW1
  FROM FRED
```

ROLLBACK

ROLLBACK ステートメントは次の目的に使用できます。

- 作業単位を終了させ、その作業単位でリレーショナル・データベースに対して行われたすべての変更をバックアウトする。アプリケーション・プロセスが使用しているリカバリー可能リソースがリレーショナル・データベースだけである場合は、ROLLBACK は作業単位も終了します。
- 作業単位を終了させずに、その作業単位内に設定された保管ポイント以降に行われた変更のみをバックアウトする。保管ポイントまでのロールバックにより、選択した変更を取り消すことができます。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことも、対話式に呼び出すこともできます。このステートメントは、動的に準備できる実行可能ステートメントです。

トリガー・プログラムとその対象となるプログラムが同じコミットメント定義のもとで実行される場合、トリガーでは ROLLBACK は使用できません。ROLLBACK が外部プロシージャーで使用できないのは、その外部プロシージャーと、CALL ステートメントを出したプログラムが、同じコミットメント定義のもとで実行される場合です。

権限

権限は不要です。

構文

説明

SAVEPOINT 文節なしの ROLLBACK を使用すると、このステートメントが実行される作業単位が終了し、新たな作業単位が開始されます。この作業単位で実行された ALTER、 CALL、 COMMENT、 CREATE、 DECLARE GLOBAL TEMPORARY TABLE、 DELETE、DROP (DROP SCHEMA を除く)、 GRANT、 INSERT、 LABEL、 RENAME、 REVOKE、および UPDATE ステートメントにより行われたすべての変更がバックアウトされます。

ただし、以下のステートメントはトランザクション制御下にはないので、ROLLBACK を発行してもこれらのステートメントにより行われた変更は取り消されません。

- CONNECT
- DISCONNECT
- RELEASE CONNECTION

- SET CONNECTION
- SET PATH
- SET SCHEMA

宣言済みグローバル一時表に対する ROLLBACK または ROLLBACK TO SAVEPOINT の影響は、DECLARE GLOBAL TEMPORARY TABLE ステートメントの ON ROLLBACK 文節の設定によって決まります。

WORK
ROLLBACK WORK と ROLLBACK の効果は同じです。

HOLD
リソースを保持するように指示します。 HOLD を指定すると、現在オープンされているカーソルはクローズされず、その作業単位の過程で獲得したリソース (表の行に対するロックは除く) はすべて保持されます。ただし、その作業単位の過程で特定の行に対して暗黙に掛けられたロックは解放されます。

HOLD を省略した場合、TO SAVEPOINT 文節なしの ROLLBACK では、この作業単位のコミットメント定義のもとで以下のことが行われます。

- この作業単位のコミットメント定義のもとでオープンされたカーソルは、クローズされます。
- この作業単位のコミットメント定義のもとで LOCK TABLE ステートメントによって確立された表のロックは、解放されます。
- すべての LOB ロケーター (保持されているものも含む) が解放されます。

カーソルを含むプログラムまたはルーチンの作成時に ALWBLK(*ALLREAD) が指定されなかった場合、ROLLBACK HOLD が終了したときのカーソルの位置は、該当する作業単位を開始したときと同じになります。

TO SAVEPOINT
作業単位を終了せずに、部分ロールバック (保管ポイントまで) のみを行うことを指定します。保管ポイント名を指定しなかった場合は、最後の活動保管ポイントまでのロールバックが行われます。例えば、ある作業単位の中で保管ポイント A、B、および C がこの順序で設定されているときに、C が解放されたとすれば、ROLLBACK TO SAVEPOINT により保管ポイント B までのロールバックが行われます。

保管ポイント名
どの保管ポイントまでロールバックするかを指定します。指定した名前の保管ポイントが存在しない場合は、エラーが起こります。

ROLLBACK TO SAVEPOINT が正常に完了した後も、保管ポイントは存続します。

保管ポイントが設定された後で行われたすべてのデータベース変更 (宣言済み一時表に対する変更も含む) がバックアウトされます。ロックおよび LOB ロケーターはすべて保持されます。

ROLLBACK TO SAVEPOINT によるカーソルへの影響は、保管ポイントに含まれるステートメントによって決まります。

- 保管ポイントに、カーソルが依存している DDL が含まれている場合は、そのカーソルはクローズされます。 ROLLBACK TO SAVEPOINT の後でこのようなカーソルを使用しようとすると、エラーが起こります。
- その他の場合は、カーソルは ROLLBACK TO SAVEPOINT の影響を受けません (オープンされ位置付けされたままの状態を維持します)。

ロールバックの対象となった保管ポイントより後で設定された保管ポイントは、すべて解放されます。ロールバックの対象となった保管ポイントは解放されません。

使用上の注意

デフォルトの活動化グループが終了すると、暗黙のロールバックが行われます。したがって、明示的な COMMIT または ROLLBACK ステートメントは、デフォルトの活動化グループが終了する前に出しておかなければなりません。

次のような場合は、ROLLBACK が自動的に実行されます。

1. デフォルトの活動化グループが最後に COMMIT を出さずに終了した場合。
2. 活動化グループの作業の完了を妨げるような障害 (例えば、電源障害など) が発生した場合。

   障害が起こった時点で COMMIT が進行中であったためにその作業単位が準備状態である場合、ロールバックは行われません。代わりに、その作業単位に関連するすべての接続の再同期化が行われます。詳しくは、コミットメント制御トピックを参照してください。
3. サーバーとの接続が失われるような障害 (例えば、通信回線の障害など) が発生した場合。

   障害が起こった時点で COMMIT が進行中であったためにその作業単位が準備状態である場合、ロールバックは行われません。代わりに、その作業単位に関連するすべての接続の再同期化が行われます。詳しくは、コミットメント制御トピックを参照してください。
4. デフォルト活動化グループ以外の活動化グループが異常終了した場合。

1 つの作業単位には、最高 400 万までの行の処理を含めることができますが、これには、SELECT INTO または FETCH ステートメント[64]の過程で取り出された行、および INSERT、DELETE、および UPDATE 操作の一環として挿入、削除、または更新されたものも含まれます。[65]

コミットおよびロールバック操作が DROP SCHEMA ステートメントに影響することはありません。したがって、このステートメントは、

---

64. COMMIT(*CHG) または COMMIT(*CS) を指定した場合は例外で、これらの行は行数の合計には含まれません。

65. この制約には以下も含まれます。

- 高水準言語のファイル処理機能によるコミットメント制御のもとでオープンされたファイルに基づいてアクセスまたは変更された行。
- トリガー、または CASCADE、SET NULL、あるいは SET DEFAULT 参照保全削除規則の結果として削除、更新、または挿入された行。

COMMIT(*CHG)、COMMIT(*CS)、COMMIT(*ALL)、または COMMIT(*RR) も指定しているアプリケーション・プログラムでは使用できません。

対象の活動化グループについてコミットメント制御が活動状態にない場合は、ROLLBACK ステートメントは使用できません。どのコミットメント定義が使用されているかを判別する方法については、COMMIT ステートメントの項の コミットメント定義に関する説明を参照してください。

破棄される準備済みステートメントに関連するカーソルは、そのステートメントが再度準備されるまで、オープンすることはできません。ROLLBACK は、接続の状態に影響を与えません。

1 つの作業単位の中で、CLOSE の後に ROLLBACK を実行すると、その作業単位の中で行われた変更はすべてバックアウトされます。ただし、CLOSE 自体はバックアウトされないので、ファイルが再オープンされることはありません。

例

例 1

ROLLBACK ステートメントを使用した例については、422ページの COMMIT の項を参照してください。

例 2

あるリカバリー単位の開始後に、A、B、C の 3 つの保管ポイントが設定され、その後 C が解放されたとします。

```
SAVEPOINT A ON ROLLBACK RETAIN CURSORS;
...
SAVEPOINT B ON ROLLBACK RETAIN CURSORS;
....
SAVEPOINT C ON ROLLBACK RETAIN CURSORS;
...
RELEASE SAVEPOINT C
```

保管ポイント A までの範囲内ですべての DB2 データベース変更をロールバックします。

```
ROLLBACK WORK TO SAVEPOINT A
```

上記で保管ポイント名を指定しなかったとすれば (つまり、ROLLBACK WORK TO SAVEPOINT を指定した場合)、設定されている最後の活動保管ポイント (つまり B) までロールバックが行われることになります。

SAVEPOINT

SAVEPOINT ステートメントは、作業単位内に保管ポイントを設定します。保管ポイントは作業単位内の特定時点を表すもので、リレーショナル・データベースに対する変更をその時点までロールバックすることができます。

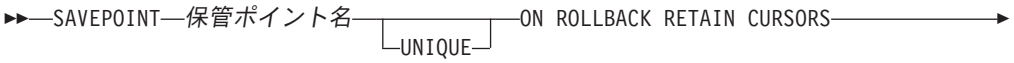
呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。このステートメントは、動的に準備できる実行可能ステートメントです。

権限

権限は不要です。

構文



注:

1 ROLLBACK のオプションは、どのような順序で指定しても構いません。

説明

保管ポイント名
新しい保管ポイントを指定します。

UNIQUE
アプリケーション・プログラムが、この作業単位内でこの保管ポイント名を再使用できないことを指定します。この作業単位内に、すでに *保管ポイント名* と同じ名前が存在している場合は、エラーが起こります。

UNIQUE を省略した場合は、アプリケーションが作業単位内でこの保管ポイント名を再使用できることを示します。*保管ポイント名* がこの作業単位内の既存の保管ポイントのどれかと同じであっても、その既存の保管ポイントの作成時に UNIQUE オプションが指定されていなければ、その既存の保管ポイントは破棄され、新しい保管ポイントが作成されます。保管ポイントを破棄して、その名前を他の保管ポイントに再使用することは、保管ポイントを解放することとは異なります。 1 つの保管ポイント名を再使用した場合、破棄される保管ポイントは 1 つだけです。しかし、RELEASE SAVEPOINT ステートメントを使用して保管ポイントの 1 つを解放すると、その保管ポイント以降に設定されているすべての保管ポイントが解放されます。

ON ROLLBACK RETAIN CURSORS
この保管ポイントへのロールバックが行われたときに、この保管ポイントの設定後にオープンされたカーソルをクローズしないことを指定します。

- 保管ポイントに、カーソルが依存している DDL が含まれている場合は、そのカーソルはクローズされます。 ROLLBACK TO SAVEPOINT の後でこのようなカーソルを使用しようとすると、エラーが起こります。
- その他の場合は、カーソルは ROLLBACK TO SAVEPOINT の影響を受けません (オープンされ位置付けされたままの状態を維持します)。

上記のカーソルは、保管ポイントへのロールバックの後もオープン状態のままになっていますが、使用不能になることもあります。例えば、保管ポイントへのロールバックが原因で、カーソルが位置付けされている行の挿入がロールバックされることになった場合、カーソルを使用してその行を更新または削除しようとすると、エラーが起こります。

ON ROLLBACK RETAIN LOCKS
この保管ポイントへのロールバックが行われたときに、保管ポイントの設定後に獲得されたロックをどれも解放しないことを指定します。

使用上の注意

アプリケーションでは、挿入操作がバッファーに入れられることがあります。SAVEPOINT、ROLLBACK、または RELEASE TO SAVEPOINT ステートメントを実行すると、バッファーはフラッシュされます。

対象の活動化グループについてコミットメント制御が活動状態にない場合は、SAVEPOINT ステートメントは使用できません。どのコミットメント定義が使用されているかを判別する方法については、 423 ページの『使用上の注意』を参照してください。

例

ある作業単位内の 3 箇所に保管ポイントを設定したいとします。第 1 の保管ポイントには A と命名し、この保管ポイント名は再使用できるようにします。第 2 の保管ポイントには B と命名し、この名前は再使用できないようにします。第 3 の保管ポイントが設定可能な状態になった時点では保管ポイント A は不要になるので、第 3 の保管ポイントの名前として A を再使用します。

```
SAVEPOINT A ON ROLLBACK RETAIN CURSORS;
.
.
.
SAVEPOINT B UNIQUE ON ROLLBACK RETAIN CURSORS;
.
.
.
SAVEPOINT A ON ROLLBACK RETAIN CURSORS;
```

SELECT

SELECT ステートメントは、照会の形式の 1 つです。このステートメントは対話式でのみ使用することができます。詳しくは、355 ページの『選択ステートメント』および 339 ページの『第 4 章 照会』を参照してください。

SELECT INTO

SELECT INTO ステートメントは、1 行以内で構成される結果表を作成し、その行の値をホスト変数に割り当てます。表が空である場合は、このステートメントは +100 を SQLCODE に、'02000' を SQLSTATE に割り当て、ホスト変数には値を割り当てません。検索条件に該当する行が複数ある場合、ステートメントの処理は終了し、エラーが起こります。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込んで使用します。それ以外の使用法はありません。このステートメントは、動的には準備できない実行可能ステートメントです。REXX で指定してはなりません。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメントで識別されている表やビューのそれぞれについて、
  - 表やビューに対する SELECT 特権、および
  - 表やビューが入っているライブラリーに対する *EXECUTE システム権限
- 管理権限

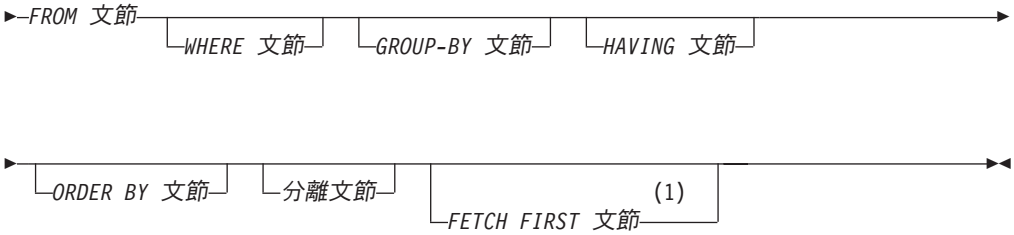
ステートメントの権限 ID は、以下の場合に表に対する SELECT 特権を持ちます。

- その表の所有者である。
- その表に対する SELECT 特権が認可されているか、または
- その表についての *OBJOPR および *READ システム権限が認可されている。

ステートメントの権限 ID は、以下の場合にビューに対する SELECT 特権を持ちます。

- そのビューの所有者である。
- そのビューに対する SELECT 特権が認可されているか、または
- そのビューについての *OBJOPR および *READ システム権限、およびそのビューが直接または間接に従属しているすべての表やビューについての *READ システム権限が認可されている。すなわち、そのビューの定義で参照されている表やビューのすべて、またそのようなビューが参照される場合、そのビューの定義で参照されている表やビューのすべて、以下同様。

構文



注:

1 FETCH FIRST 文節で指定できる行は 1 つだけです。

説明

結果表は、*分離文節*、*FROM 文節*、*WHERE 文節*、*GROUP BY 文節*、*HAVING 文節*、*SELECT 文節*、*ORDER BY 文節*、および *FETCH FIRST 文節* をこの順序で評価することによって求められます。

SELECT 文節、*FROM 文節*、*WHERE 文節*、*GROUP BY 文節*、*HAVING 文節*、*ORDER BY 文節* 、*FETCH FIRST 文節*、および*分離 文節* の説明については、339 ページの『第 4 章 照会』を参照してください。

GROUP BY 文節 でグループ化を指定すると、複数行から成る結果表を作成するという意味合いが強くなるので、ほとんどの場合、結果表の行を 1 行だけにするために *HAVING 文節* が必要になることに注意してください。

INTO *ホスト変数*,...

1 つまたは複数のホスト構造体、あるいはホスト変数を指定します。これらのホスト構造体や変数は、その宣言の規則に従ってプログラムで宣言する必要があります。INTO 文節の操作形式では、ホスト構造体に対する参照は、その個々の変数に対する参照によって置き換えられます。結果の行の最初の値がリストの最初のホスト変数に割り当てられ、2 番目の値が 2 番目のホスト変数に割り当てられます。以下同様です。各ホスト変数のデータ・タイプは、それぞれに対応する列と互換性がなければなりません。

ホスト変数への割り当てはそれぞれ、第 2 章で説明している規則に従って行われます。変数の数が行の中の値の数より少ない場合、SQLCA の SQLWARN3 フィールドに 'W' がセットされます。結果の列の数よりもホスト変数の数が多い場合、警告は出されないので注意してください。値がヌルの場合は、標識変数が用意されている必要があります。割り当てエラーが発生した場合は、該当のホスト変数の値、および後に続くホスト変数の値は予期できなくなります。ただし、変数にすでに割り当てられている値があれば、その値は割り当てられたままです。

SELECT 文節の結果の列の評価で、次のデータ・マッピング・エラーのいずれかが生じた場合は、結果はヌル値になります。

- 文字を変換できなかった。
- 数値変換エラー (アンダーフローまたはオーバーフロー)。
- 算術式エラー (0 による除算)。

- 日付またはタイム・スタンプの変換エラー (指定されている日付形式の有効な範囲内にない日付またはタイム・スタンプ)。
- 日付/時刻の値のストリング表現が正しくない。
- 混合 (MIXED) データが正しく形成されていない。
- 数値が無効だった。
- スカラー関数 SUBSTR の引き数が範囲外。

他のヌル値の場合と同様に、標識変数を用意しなければなりません。 該当のホスト変数の値は、未定義になります。ただし、この場合、標識変数は -2 にセットされます。ステートメントの処理は、エラーが発生しなかった場合と同様に継続されます。 (ただし、このエラーで正の SQLCODE になります。) 標識変数を用意していない場合は、SQLCA のフィールド SQLCODE に負の値が戻されます。エラーが発生した変数およびそれ以降のあらゆる変数の値は、予期できないものになります。ただし、変数にすでに割り当てられている値があれば、その値は割り当てられたままです。

結果表に複数の行があることが原因でエラーが発生した場合 (SQLSTATE 21000) でも、すべてのホスト変数に値が割り当てられますが、その値がどの行から得られるかは不定であり、予期できません。

例

例 1

COBOL プログラムのステートメントを使用して、表 EMPLOYEE の給与 (SALARY) 最高額を、分離レベルが非コミット読み取り (CS) のホスト変数 MAX-SALARY (DECIMAL(9,2)) に入れます。

```
EXEC SQL  SELECT MAX(SALARY)
            INTO :MAX-SALARY
            FROM EMPLOYEE WITH CS
END-EXEC.
```

例 2

Java プログラムのステートメントを使用して、接続コンテキスト 'ctx' にある EMPLOYEE 表から、従業員番号 (EMPNO) の値がホスト変数 HOST_EMP に保管されている値 (java.lang.String) と同じである行を選択します。さらに、選択した行にある名字 (LASTNAME) および教育レベル (EDLEVEL) を、それぞれホスト変数 HOST_NAME (ストリング) および HOST_EDUCATE (整数) に入れます。

```
#sql [ctx] {  SELECT LASTNAME, EDLEVEL
              INTO :HOST_NAME, :HOST_EDUCATE
              FROM EMPLOYEE
              WHERE EMPNO = :HOST_EMP   };
```

SET CONNECTION

SET CONNECTION ステートメントは、既存の接続の 1 つを識別することによって、活動化グループの現行サーバーを確立します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことと、対話式に呼び出すことだけが可能です。このステートメントは、動的には準備できない実行可能ステートメントです。Java または REXX では指定できません。

SET CONNECTION はトリガーでは使用できません。リモート・サーバーで外部プロシージャーを呼び出す場合、その外部プロシージャーでは SET CONNECTION は使用できません。

権限

権限は不要です。

構文

説明

サーバー名 または ホスト変数
指定したサーバー名、または指定したホスト変数に入っているサーバー名によって接続を識別します。ホスト変数を指定する場合、

- その変数は、文字ストリング変数でなければなりません。
- 標識変数を伴っていてはなりません。
- サーバー名は、その変数内で左寄せし、通常 ID の形成の規則に従っていなければなりません。
- サーバー名の長さが、ホスト変数の長さよりも短い場合、右側をブランクで埋めなければなりません。

以下の説明で S は、指定したサーバー名またはホスト変数に入っているサーバー名を表しています。S は該当のアプリケーション・プロセスの既存の接続を識別していなければなりません。S が現行の接続を識別している場合は、S の状態およびアプリケーション・プロセスの他の接続すべての状態は変わりませんが、S に関する情報が SQLCA のフィールド SQLERRP に入れられます。S が休止接続を識別している場合、次の規則が適用されます。

SET CONNECTION ステートメントが成功した場合、

- 接続 S は、現行状態になります。
- S が特殊レジスター CURRENT SERVER に入れられます。
- サーバー S に関する情報が、SQLCA のフィールド SQLERRP に入れられます。サーバーが IBM リレーショナル・データベースのプロダクトである場合は、その情報は *pppvvrrm* の形式をとります。ここで、

– *ppp* は、次のようにプロダクトを識別します。
  ARI (DB2 UDB サーバー (VSE および VM 版) の場合)
  DSN (DB2 UDB (OS/390 および z/OS 版) の場合)
  QSQ (DB2 UDB for iSeries の場合)
  SQL (他のすべての DB2 プロダクトの場合)
– *vv* は、2 桁のバージョン ID です (例えば、'04' など)。
– *rr* は、2 桁のリリース ID です (例えば、'01' など)。
– *m* は、1 桁のモディフィケーション・レベルを示します (例えば、'0' など)。

例えば、サーバーが DB2 UDB (OS/390 および z/OS 版) のバージョン 4 であれば、SQLERRP の値は 'DSN04010' になります。

- 接続に関する追加の情報が SQLCA のフィールド SQLERRD(4) に入れられます。SQLERRD(4) には、該当のサーバーがコミット可能な更新を行うのを許すかどうかを示す値が入ります。以下は、この CONNECT に関して SQLCA のフィールド SQLERRD(4) に入れられる値とその意味を示しています。
  – 1 - コミット可能な更新を行うことができ、また、接続は無保護会話を使用し、CONNECT (タイプ 1) ステートメントを使用するアプリケーション・リクエスター・ドライバー・プログラムに対して確立された接続であるか、または CONNECT (タイプ 1) を使用して確立されたローカル接続のいずれかです。
  – 2 - コミット可能な更新は行うことができません。会話は無保護です。
  – 3 - コミット可能な更新を行うことができるか否かは不明です。会話は保護会話です。
  – 4 - コミット可能な更新を行うことができるか否かは不明です。会話は無保護です。
  – 5 - コミット可能な更新を行うことができるかどうかは不明で、その接続は、CONNECT (タイプ 2) ステートメントを使用して確立されたローカル接続、または CONNECT (タイプ 2) ステートメントを使用して確立されたアプリケーション・リクエスター・ドライバー・プログラムとの接続です。
- 接続についての追加の情報は SQLCA のフィールド SQLERRMC に入れられます。フィールド SQLERRMC の中の情報の説明については、付録 B、「SQL 連絡域」を参照してください。
- それ以前の現行接続は、休止状態になります。

SET CONNECTION ステートメントが不成功であった場合、該当の活動化グループの接続状態およびその接続の状態は変わりません。

使用上の注意

同一の作業単位の中で接続が使用され、休止状態になり、その後で現行状態に復元された場合は、その接続に関するロック、カーソル、および準備済みステートメントの状況は、その活動化グループによる最後の使用を反映しています。

| 現行の独立補助記憶域プール (IASP) ネーム・スペースがローカル接続のリレーショナル・データベースに一致しない場合は、そのローカル接続に対する SET CONNECTION は失敗します。

例

TOROLAB1 で SQL ステートメントを実行し、次に TOROLAB2 で SQL ステートメントを実行し、最後に TOROLAB1 でさらに SQL ステートメントを実行します。

```
EXEC SQL CONNECT TO TOROLAB1;

 (TOROLAB1 のオブジェクトを参照するステートメントを実行する)

EXEC SQL CONNECT TO TOROLAB2;

 (TOROLAB2 のオブジェクトを参照するステートメントを実行する)


EXEC SQL SET CONNECTION TOROLAB1;

 (TOROLAB1 のオブジェクトを参照するステートメントを実行する)
```

最初の CONNECT ステートメントは、TOROLAB1 との接続を確立し、2 番目の CONNECT ステートメントはその接続を休止状態にし、SET CONNECTION ステートメントはその接続を現行状態に戻します。

SET OPTION

SET OPTION ステートメントは、SQL ステートメントで使用される処理オプションを設定します。

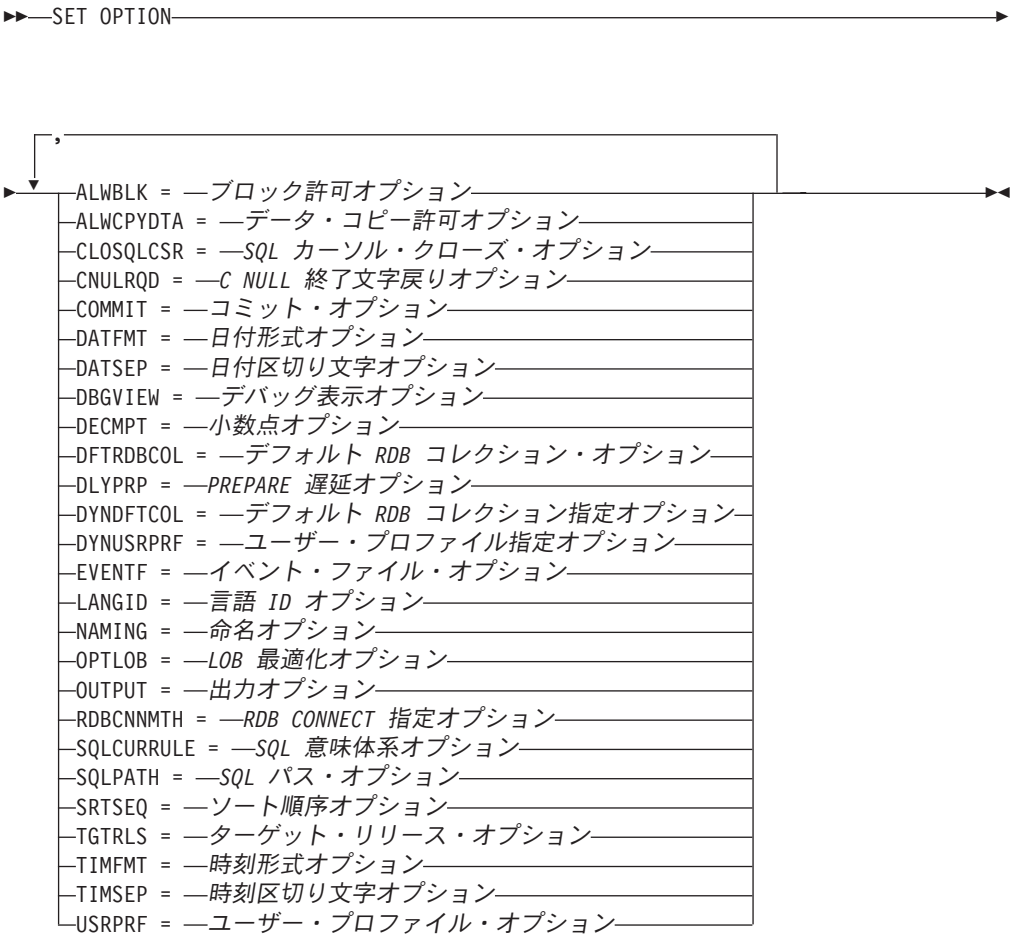
呼び出し

このステートメントは、REXX プロシージャーで使用、またはアプリケーション・プログラムに組み込むことができます。REXX プロシージャーで使用する場合、このステートメントは実行可能ステートメントです。アプリケーション・プログラムに組み込む場合、このステートメントは実行可能ではなく、他のどの SQL ステートメントよりも先に行う必要があります。このステートメントは、動的に準備することができません。

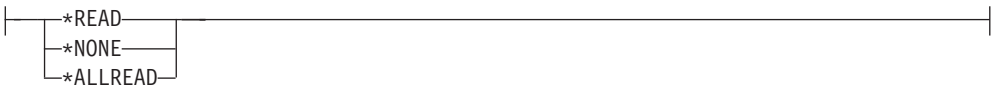
権限

権限は不要です。

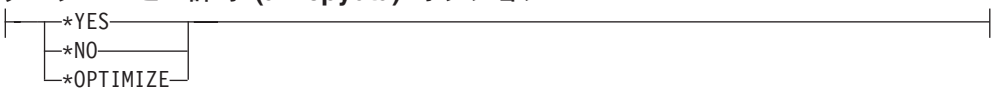
構文



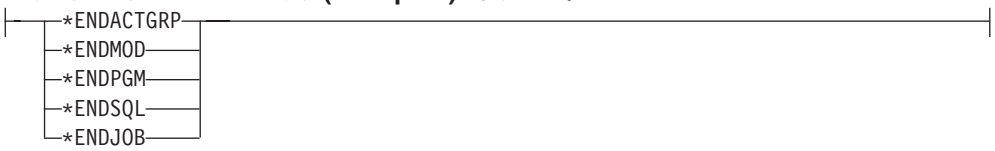
ブロック許可 (alwblk) オプション:



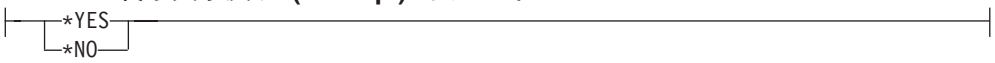
データ・コピー許可 **(alwcpydta)** オプション:



SQL カーソル・クローズ **(closqlcsr)** オプション:



C NULL 終了文字戻り **(cnulrqd)** オプション:



コミット **(commit)** オプション:

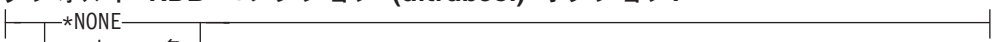
日付形式 **(datfmt)** オプション:

日付区切り文字 **(datsep)** オプション:

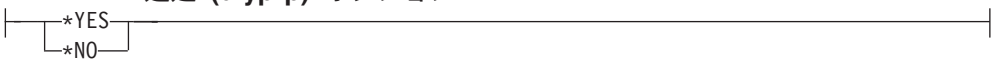
小数点 (decmpt) オプション:

デバッグ表示 (dbgview) オプション:

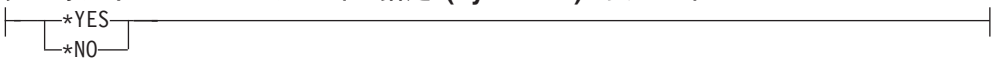
デフォルト RDB コレクション (dftrdbcol) オプション:



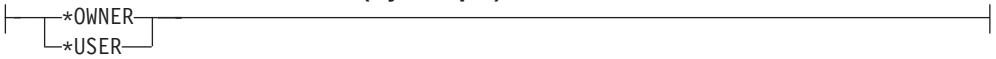
PREPARE 遅延 (dlyprp) オプション:



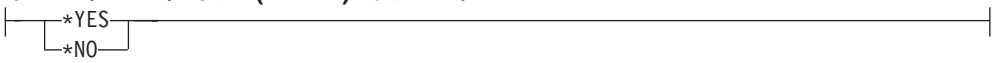
デフォルト RDB コレクション指定 (dyndftcol) オプション:



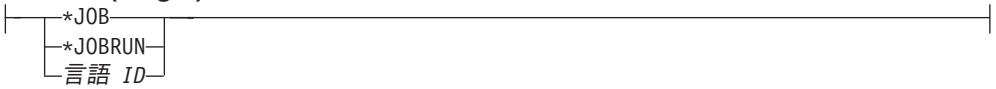
ユーザー・プロファイル指定 (dynusrprf) オプション:



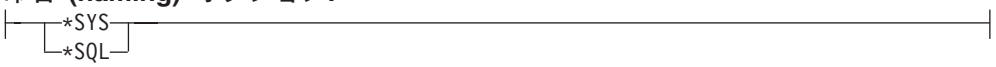
イベント・ファイル (eventf) オプション:



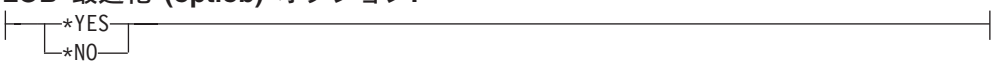
言語 ID (langid) オプション:



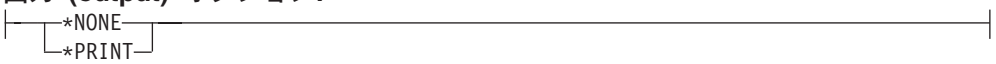
命名 (naming) オプション:



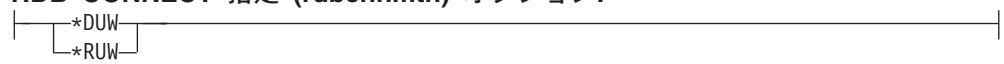
LOB 最適化 (optlob) オプション:



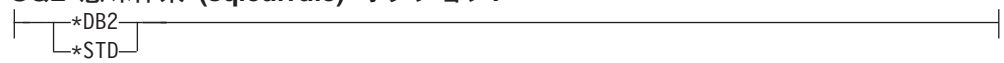
出力 (output) オプション:



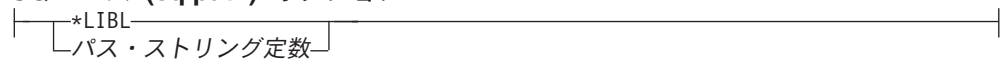
RDB CONNECT 指定 (rdbcnnmth) オプション:



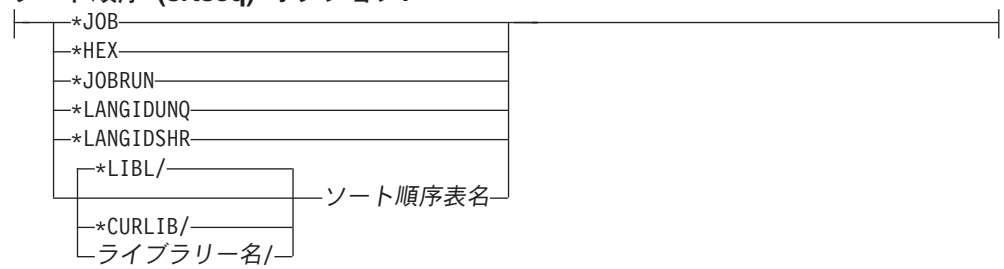
SQL 意味体系 (sqlcurrule) オプション:



SQL パス (sqlpath) オプション:



ソート順序 (srtseq) オプション:

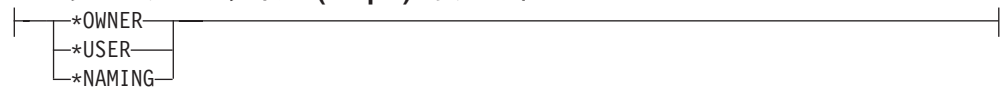


ターゲット・リリース (tgtrls) オプション:

時刻形式 (timfmt) オプション:

時刻区切り文字 (timsep) オプション:

ユーザー・プロファイル (usrprf) オプション:



説明

ALWBLK

データベース・マネージャーが行ブロッキングを使用できるかどうか、およびブロッキングを読み取り専用カーソルに使用できる範囲を指定します。このオプションは、REXX では無視されます。

***ALLREAD**

COMMIT が *NONE または *CHG の場合、読み取り専用カーソルの場合に行がブロックされます。プログラム内にある、明示的に更新できないカーソルはすべて EXECUTE または EXECUTE IMMEDIATE ステートメントがそのプログラム内にある可能性があっても、読み取り専用処理用にオープンされます。

*ALLREAD を指定すると、

- *READ で許可されているブロッキングに加えて、コミットメント制御レベル *CHG のもとで行ブロッキングが可能になります。
- プログラム内のほとんどすべての読み取り専用カーソルのパフォーマンスを上げることができますが、以下のやり方で照会が制限されます。
  - ロールバック (ROLLBACK) コマンド、ホスト言語での ROLLBACK ステートメント、または ROLLBACK HOLD SQL ステートメントは、*ALLREAD が指定されると、読み取り専用カーソルの位置変更をしません。
  - 位置指定 UPDATE または DELETE ステートメントを動的に実行 (例えば、EXECUTE IMMEDIATE を使用して) しても、カーソルの DECLARE ステートメントに FOR UPDATE 文節が含まれていない場合は、そのカーソル内の行を更新することはできません。

***NONE**

カーソルに関するデータの検索のために、行はブロックされません。

*NONE を指定すると、

- 検索されるデータが必ず現行のデータになります。
- 照会用のデータの最初の行を検索するために要する時間が短縮される場合があります。
- 照会がクローズする前に、その照会の最初の数行しか検索されないときは、データベース・マネージャーがプログラムによって使用されないデータ行のブロックを検索するのを、取り止めるようにします。
- 多数の行を検索する照会の場合、その照会全体のパフォーマンスを低下させる場合があります。

***READ**

次の場合に、カーソルに関するデータの読み取り専用検索で、行がブロックされます。

- COMMIT パラメーターに *NONE が指定され、コミットメント制御が使用されないことが指示されたとき。
- FOR READ ONLY 文節によってカーソルが宣言されたとき、またはカーソルに関して位置指定 UPDATE ステートメントまたは DELETE ステートメントを実行できる動的ステートメントがないとき。

*READ を指定すると、上記の条件を満たし、かつ大量の行を検索する照会の全体のパフォーマンスを上げることができます。

ALWCPYDTA

データのコピーを SELECT ステートメント内で使用できるかどうかを指定します。このオプションは、REXX では無視されます。

***OPTIMIZE**

システムが、データベースから直接検索されたデータを使用するか、そのデータのコピーを使用するかを決定します。この決定は、どちらの方法が最高のパフォーマンスを発揮するかに基づいて行われます。COMMIT が *CHG または *CS で、ALWBLK が *ALLREAD でない場合、または COMMIT が *ALL または *RR の場合には、照会の実行が必要な場合に限りデータのコピーが使用されます。

***YES**

データのコピーは、必要な場合にだけ使用されます。

***NO**

データのコピーを使用することはできません。そのデータの一時コピーが照会の実行に必要な場合、エラー・メッセージが戻されます。

CLOSQLCSR

SQL カーソルが暗黙的にクローズされるとき、SQL 準備済みステートメントが暗黙的に廃棄され、LOCK TABLE ロックが解除されることを指定します。SQL カーソルは、CLOSE、COMMIT、または ROLLBACK (HOLD はなし) の各 SQL ステートメントを出すと、明示的にクローズされます。このオプションは、REXX では無視されます。*ENDACTGRP および *ENDMOD は、ILE プログラムおよびモジュールが使用するためのものです。
*ENDPGM、*ENDSQL、および *ENDJOB は、非 ILE プログラムが使用します。

このオプションは、SQL 関数、SQL プロシージャー、または SQL トリガーでは使用できません。

***ENDACTGRP**

活動化グループが終了すると、SQL カーソルはクローズし、SQL 準備済みステートメントは暗黙的に廃棄され、LOCK TABLE ロックは解除されます。

***ENDMOD**

モジュールが終了すると、SQL カーソルはクローズし、SQL 準備済みステートメントは暗黙的に廃棄されます。LOCK TABLE ロックは、呼び出しスタックの最初の SQL プログラムが終了すると解除されます。

***ENDPGM**

プログラムが終了すると、SQL カーソルはクローズし、SQL 準備済みステートメントは廃棄されます。LOCK TABLE ロックは、呼び出しスタックの最初の SQL プログラムが終了すると解除されます。

***ENDSQL**

SQL カーソルは、呼び出しから次の呼び出しまでの間もオープンしたままで、新たに SQL OPEN を実行しなくても取り出すことができます。この場合、呼び出しスタック上で高位にあるプログラムの 1 つが、少なくとも 1

個の SQL ステートメントを実行していなければなりません。呼び出しスタックの最初の SQL プログラムが終了すると、SQL カーソルはクローズし、SQL 準備済みステートメントは廃棄され、LOCK TABLE ロックは解除されます。最初に呼び出された SQL プログラム (呼び出しスタック上の最初の SQL プログラム) に *ENDSQL が指定されると、そのプログラムは *ENDPGM が指定された場合と同様に扱われます。

***ENDJOB**
SQL カーソルは、呼び出しから次の呼び出しまでの間もオープンしたままで、新たに SQL OPEN 実行しなくても取り出すことができます。呼び出しスタック上で高位にあるプログラムが、SQL ステートメントを実行している必要はありません。呼び出しスタックの最初の SQL プログラムが終了しても、SQL カーソルはオープンしたままで、SQL 準備済みステートメントは保存され、LOCK TABLE ロックは保持されます。ジョブが終了すると、SQL カーソルはクローズされ、SQL 準備済みステートメントは廃棄され、LOCK TABLE ロックは解除されます。

CNULRQD
文字およびグラフィック・ホスト変数に NUL 終了文字を戻すかどうかを指定します。このオプションは、C および C++ プログラム内の SQL ステートメントにしか使用されません。

このオプションは、SQL 関数、SQL プロシージャー、または SQL トリガーでは使用できません。

***YES**
出力の文字およびグラフィック・ホスト変数に、常に NUL 終了文字が含まれます。NUL 終了文字用のスペースが不足している場合、データが切り捨てられ、NUL 終了文字が追加されます。入力の文字およびグラフィック・ホスト変数には、NUL 終了文字が必須です。

***NO**
出力の文字およびグラフィック・ホスト変数の場合、ホスト変数の長さがデータとまったく同じ場合、NUL 終了文字は戻されません。入力の文字およびグラフィック・ホスト変数には、NUL 終了文字は必要ありません。

COMMIT
使用される分離レベルを指定します。REXX では、ソースで参照されるファイルはこのオプションの影響を受けません。SQL ステートメントで参照される表、ビュー、およびパッケージだけが影響を受けます。分離レベルの詳細については、24 ページの 『分離レベル』を参照してください。

***CHG**
非コミット読み取りの分離レベルを指定します。

***NONE**
コミットなしの分離レベルを指定します。REXX プロシージャーに DROP SCHEMA ステートメントが入っている場合、*NONE を使用する必要があります。

***CS**
カーソル固定の分離レベルを指定します。

***ALL**
読み取り固定の分離レベルを指定します。

***RR**
反復可能読み取りの分離レベルを指定します。

DATFMT
日付結果列にアクセスするときに使用する形式を指定します。日付の出力フィールドは、すべて指定した形式で戻されます。入力日付ストリングのときは、日付が有効な形式で指定されたかどうかを判別するために、指定した値が使用されます。

注: *USA、*ISO、*EUR、または *JIS の形式を使用する入力日付ストリングは、常に有効です。

***JOB:**
ジョブに指定された形式が使用されます。ジョブの現行日付形式を決定するには、ジョブ表示 (DSPJOB) コマンドを使用してください。

***ISO**
国際標準化機構 (ISO) の日付形式 (yyyy-mm-dd) が使用されます。

***EUR**
欧州の日付形式 (dd.mm.yyyy) が使用されます。

***USA**
米国の日付形式 (mm/dd/yyyy) が使用されます。

***JIS**
日本工業規格 (JIS) の日付形式 (yyyy-mm-dd) が使用されます。

***MDY**
日付形式 (mm/dd/yy) が使用されます。

***DMY**
日付形式 (dd/mm/yy) が使用されます。

***YMD**
日付形式 (yy/mm/dd) が使用されます。

***JUL**
年間通算日形式 (yy/ddd) が使用されます。

DATSEP
日付の結果列にアクセスする場合に使用される、区切り文字を指定します。

注: このパラメーターは、*JOB、*MDY、*DMY、*YMD、または *JUL が DATFMT パラメーターで指定されたときだけ適用されます。

***JOB**
そのジョブで指定されている日付区切り文字が使用されます。ジョブ表示 (DSPJOB) コマンドを使用すると、ジョブの現在の値を確かめることができます。

***SLASH** または **'/'**
スラッシュ (/) が使用されます。

***PERIOD** または **'.'**
ピリオド (.) が使用されます。

***COMMA** または **','**
コンマ (,) が使用されます。

***DASH** または **'-'**
ダッシュ (-) が使用されます。

***BLANK** または **' '**
ブランク ( ) が使用されます。

DBGVIEW
コンパイラーが提供するデバッグ情報のタイプを指定します。DBGVIEW パラメーターは、SQL 関数、プロシージャー、およびトリガーの本体でのみ指定できます。選択可能な項目は、次のとおりです。

***NONE**
デバッグ表示は生成されません。

***SOURCE**
SQL ステートメント・ソースを使用して、コンパイル済みモジュール・オブジェクトをデバッグできます。

***STMT**
プログラム・ステートメント番号と記号 ID を使用して、コンパイル済みモジュール・オブジェクトをデバッグできます。

***LIST**
コンパイル済みモジュール・オブジェクトのデバッグのリスト表示を生成します。

DECMPT
小数点を表すのに使用する記号を指定します。選択可能な項目は、次のとおりです。

***PERIOD**
小数点を表すのにピリオドを使用します。

***COMMA**
小数点を表すのにコンマを使用します。

***SYSVAL**
小数点の表現は、システム値 (QDECFMT) に従います。

***JOB**
小数点を表すのに、ジョブ値 (DECFMT) を使用します。

DFTRDBCOL
表、ビュー、索引、および SQL パッケージの非修飾名に使用するスキーマ名を指定します。このパラメーターは、静的 SQL ステートメントにのみ適用します。このオプションは、REXX では無視されます。

このオプションは、SQL 関数、SQL プロシージャー、または SQL トリガーでは使用できません。

***NONE**
OPTION プリコンパイル・パラメーターに指定された、または SET OPTION NAMING オプションによって指定された命名規則が使用されます。

スキーマ名
スキーマの名前を指定します。この値は、OPTION プリコンパイル・パラメーターに指定された、または SET OPTION NAMING オプションによって指定された命名規則の代わりに使用されます。

DLYPRP
PREPARE ステートメントの動的ステートメント妥当性検査を、 OPEN、EXECUTE、または DESCRIBE ステートメントが実行されるまで遅らせるかどうかを指定します。妥当性検査を遅らせると、冗長妥当性検査が行われなくなるのでパフォーマンスが上がります。このオプションは、REXX では無視されます。

***NO**
動的ステートメント妥当性検査は遅れません。動的ステートメントが準備されると、アクセス・プランの妥当性検査が行われます。動的ステートメントが OPEN または EXECUTE ステートメントで使用されると、アクセス・プランの妥当性検査が再度行われます。動的ステートメントによって参照されるオブジェクトの権限または存在は変わる可能性があるため、OPEN または EXECUTE ステートメントを出した後でも SQLCODE または SQLSTATE をチェックして、その動的ステートメントがまだ有効であるか確認する必要があります。

***YES**
動的ステートメントの妥当性検査は、その動的ステートメントが OPEN 、EXECUTE、または DESCRIBE SQL ステートメントで使用されるまで遅れます。動的ステートメントが使用された時点で、妥当性検査は完了し、アクセス・プランが作成されます。*YES を指定する場合、OPEN、EXECUTE、または DESCRIBE ステートメントを実行した後に SQLCODE および SQLSTATE をチェックして、その動的ステートメントが有効であるか確認する必要があります。

注: *YES を指定すると、PREPARE ステートメントで INTO 文節が使用された場合、または DESCRIBE ステートメントで、そのステートメントに OPEN が出される前に動的ステートメントが使用された場合、パフォーマンスは上がりません。

DYNDFTCOL
DFTRDBCOL パラメーターに指定されたスキーマ名が、動的ステートメントにも使用されることを指定します。このオプションは、REXX では無視されます。

このオプションは、SQL 関数、SQL プロシージャー、または SQL トリガーでは使用できません。

***NO**
表、ビュー、索引、および SQL パッケージの非修飾名として DFTRDBCOL に指定された値を、動的 SQL ステートメントには使用しま

せん。OPTION プリコンパイル・パラメーターに指定された、または SET OPTION NAMING オプションによって指定された命名規則が使用されます。

***YES**
DFTRDBCOL に指定されたスキーマ名が、動的 SQL ステートメントの中で、表、ビュー、索引、および SQL パッケージの非修飾名として使用されます。

DYNUSRPRF
動的 SQL ステートメントにユーザー・プロファイルを使用することを指定します。このオプションは、REXX では無視されます。

***USER**
ローカル動的 SQL ステートメントが、ジョブのユーザー・プロファイルのもとで実行されます。分散動的 SQL ステートメントは、サーバー・ジョブのユーザー・プロファイルのもとで実行されます。

***OWNER**
ローカル動的 SQL ステートメントが、プログラムの所有者のユーザー・プロファイルのもとで実行されます。分散動的 SQL ステートメントは、SQL パッケージの所有者のユーザー・プロファイルのもとで実行されます。

EVENTF
イベント・ファイルを生成するかどうかを指定します。連携開発環境/400 (CoOperative Development Environment/400: CODE/400) は、イベント・ファイルを使用して、CODE/400 エディターと統合されたエラー・フィードバックを提供します。

***YES**
コンパイラーは、連携開発環境/400 (CODE/400) が使用するイベント・ファイルを生成します。

***NO**
コンパイラーは、連携開発環境/400 (CODE/400) が使用するイベント・ファイルを生成しません。

LANGID
SRTSEQ(*LANGIDUNQ) または SRTSEQ(*LANGIDSHR) が指定されているときに使用される、言語 ID を指定します。

***JOB** または ***JOBRUN**
そのジョブの LANGID の値が使用されます。

分散アプリケーションの場合、LANGID(*JOBRUN) が有効なのは、SRTSEQ(*JOBRUN) も指定されている場合だけです。

言語 ID
使用したい言語の ID を指定します。言語 ID として使用できる値についての説明は、iSeries Information Center の言語 ID トピックを参照してください。

NAMING
SQL 命名規則とシステム命名規則のどちらを使用するかを指定します。このオプションは、SQL 関数、SQL プロシージャー、または SQL トリガーでは使用できません。

選択可能な項目は、次のとおりです。

***SYS**
システム命名規則が使用されます。

***SQL**
SQL 命名規則が使用されます。

OPTLOB
LOB へのアクセスが、DRDA を介してアクセスする場合に最適化できるかどうかを指定します。選択可能な項目は、次のとおりです。

***YES**
LOB アクセスは最適化されます。カーソルの最初の FETCH によって、それ以降のすべての FETCH においてそのカーソルが LOB でどのように使用されるかが決定されます。このオプションは、そのカーソルがクローズされるまで有効です。

最初の FETCH で LOB 列にアクセスするために LOB ロケーターを使用すると、それ以降、そのカーソルの FETCH で、その LOB 列を LOB ホスト変数内に取り出すことはできません。

最初の FETCH で LOB ホスト変数内に LOB 列が置かれると、それ以降、そのカーソルの FETCH でその列用に LOB ロケーターを使用することができません。

***NO**
LOB アクセスは最適化されません。列を LOB ロケーター内に取り出すか、LOB ホスト変数内に取り出すかについての制約はありません。このオプションによって、パフォーマンスが低下する場合があります。

OUTPUT
プリコンパイラーおよびコンパイラー・リストを生成するかどうかを指定します。OUTPUT パラメーターは、SQL 関数、プロシージャー、およびトリガーの本体でのみ指定できます。選択可能な項目は、次のとおりです。

***NONE**
プリコンパイラーおよびコンパイラー・リストは生成されません。

***PRINT**
プリコンパイラーおよびコンパイラー・リストが生成されます。

RDBCNNMTH
CONNECT ステートメントに使用する意味体系を指定します。このオプションは、REXX では無視されます。

***DUW**
CONNECT (タイプ 2) の意味体系は、分散作業単位をサポートするのに使用されます。追加のリレーショナル・データベースに対して CONNECT ステートメントを連続して使用しても、それ以前の接続は切断されません。

***RUW**
CONNECT (タイプ 1) の意味体系は、リモート作業単位をサポートするのに使用されます。連続する CONNECT ステートメントによって、新しい接続が確立される前に直前の接続が切断されることになります。

SQLCURRULE
SQL ステートメントに使用する意味体系を指定します。

***DB2**
すべての SQL ステートメントの意味体系は、デフォルトにより、DB2 UDB 用に設定された規則に従います。以下の意味体系は、このオプションによって制御されます。

- 16 進定数が文字データとして処理されます。

***STD**
すべての SQL ステートメントの意味体系は、デフォルトにより、ISO および ANSI SQL の規格用に設定された規則に従います。以下の意味体系は、このオプションによって制御されます。

- 16 進定数が 2 進データとして処理されます。

SQLPATH
静的 SQL ステートメント内で、プロシージャー、関数、およびユーザー定義タイプを見つけるために使用するパスを指定します。このオプションは、REXX では無視されます。

***LIBL**
使用されるパスは、実行時のライブラリー・リストです。

文字ストリング
コンマで区切られた 1 つまたは複数のスキーマ名をもつ文字定数。

SRTSEQ
SQL ステートメントの中のストリング比較に使用されるソート順序表を指定します。

注: *HEX を指定しなければならないのは、REXX プロシージャーが接続されるサーバーが DB2 UDB for iSeries ではないか、または iSeries システムのリリース・レベルが V2R3M0 より前の場合です。

***JOB** または ***JOBRUN**
そのジョブの SRTSEQ の値が使用されます。

***HEX**
ソート順序表は使用しません。ソート順序を決定するには、文字の 16 進値を使用します。

***LANGIDUNQ**
ソート順序表には、コード・ページの各文字ごとに固有の重み付けが含まれていなければなりません。

***LANGIDSHR**
指定された LANGID の共用重みづけソート表が使用されます。

ソート順序表名
そのプログラムで使用したいソート順序表の名前を指定します。ソート順序表の名前は、次のライブラリーの値のいずれかによって修飾できます。

***LIBL**
そのジョブのライブラリー・リストのユーザーおよびシステム部分のすべてのライブラリーが検索され、見つかった最初の表が使用されます。

***CURLIB**
ジョブ用の現行ライブラリーが検索されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

ライブラリー名
検索したいライブラリーの名前を指定します。

TGTRLS
ユーザーが作成するオブジェクトを使用するオペレーティング・システムのリリースを指定します。TGTRLS パラメーターは、SQL 関数、プロシージャー、およびトリガーの本体でのみ指定できます。選択可能な項目は、次のとおりです。

VxRxMx
VxRxMx 形式でリリースを指定します。ここで、Vx はバージョン、Rx はリリース、Mx は修正レベルを表します。例えば、V5R1M0 は、バージョン 5、リリース 1、修正レベル 0 です。オブジェクトは、指定のリリースまたはそれ以降のリリースのオペレーティング・システムがインストールされたシステム上で使用できます。

有効な値は、現行のバージョン、リリース、および修正レベルによって決まり、新規リリースのたびに変更されます。データベース・マネージャーによってサポートされる最も古いリリース・レベルより前のリリース・レベルを指定すると、サポートされる最も古いリリースを示したエラー・メッセージが送られます。

TGTRLS オプションは、SQL 関数、SQL プロシージャー、およびトリガーに対してのみ指定できます。

TIMFMT
時刻の結果列にアクセスするときに使用する形式を指定します。時刻の出力フィールドはすべて指定した形式で戻されます。入力時刻ストリングの場合は、指定された値を使用して、時刻が有効な形式で指定されているかどうかを判別します。

注: *USA、*ISO、*EUR、または *JIS の形式を使用する入力時刻ストリングは、常に有効です。

***HMS**
形式 (hh:mm:ss) が使用されます。

***ISO**
国際標準化機構 (ISO) の時刻形式 (hh.mm.ss) が使用されます。

***EUR**
欧州の時刻形式 (hh.mm.ss) が使用されます。

***USA**
米国の時刻形式 (hh:mm *xx*) が使用されます。ここで、*xx* は AM または PM です。

***JIS**
日本工業規格 (JIS) の時刻形式 (hh:mm:ss) が使用されます。

TIMSEP
時刻の結果列にアクセスする場合に使用される区切り文字を指定します。

注: このパラメーターは、TIMFMT パラメーターで *HMS が指定されたときだけ適用されます。

***JOB**
そのジョブに指定されている時刻の区切り文字が使用されます。ジョブ表示 (DSPJOB) コマンドを使用すると、ジョブの現在の値を確かめることができます。

***COLON** または **':'**
コロン (:) が使用されます。

***PERIOD** または **'.'**
ピリオド (.) が使用されます。

***COMMA** または **','**
コンマ (,) が使用されます。

***BLANK** または **' '**
ブランク ( ) が使用されます。

USRPRF
コンパイル済みプログラム・オブジェクトが実行される際に使用されるユーザー・プロファイル (そのプログラム・オブジェクトが静的 SQL ステートメント内の各オブジェクトごとに保有する権限も含む) を指定します。プログラムの所有者またはプログラム・ユーザーのいずれかのプロファイルが、プログラム・オブジェクトがどのオブジェクトを使用できるかを制御するために使用されます。このオプションは、REXX では無視されます。

***NAMING**
ユーザー・プロファイルは、命名規則によって決定されます。命名規則が *SQL の場合、USRPRF(*OWNER) が使用されます。命名規則が *SYS の場合、USRPRF(*USER) が使用されます。

***USER**
プログラム・オブジェクトを実行しているユーザーのプロファイルが使用されます。

***OWNER**
プログラムの所有者とプログラム・ユーザーの両方のユーザー・プロファイルが、プログラムの実行時に使用されます。

使用上の注意

REXX プロシージャーの開始時に、オプションはそれぞれのデフォルト値に設定されます。各オプションのデフォルト値は、構文図に最初にリストされている値です。オプションが SET OPTION ステートメントによって変更されると、新しい値は、そのオプションが再度変更されるか、またはその REXX プロシージャーが終了するまで有効です。

アプリケーション・プログラムでは、処理オプションは、最初に CRTSQLxxx コマンドで指定した値に設定されます。各オプションは、SET OPTION ステートメントを検出すると更新されます。 SET OPTION ステートメントはすべて、他のどの組み込み SQL よりも先に置く必要があります。

同義のキーワード：以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- *UR は *CHG の同義語として使用できます。
- *NC は *NONE の同義語として使用できます。
- *RS は *ALL の同義語として使用できます。

例

例 1：分離レベルを *ALL に、命名モードを SQL 名に設定します。

```
EXEC SQL SET OPTION COMMIT =*ALL, NAMING =*SQL
```

例 2：日付形式を欧州形式に、分離レベルを *CS に、小数点をコンマに設定します。

```
EXEC SQL SET OPTION DATFMT = *EUR, COMMIT = *CS, DECMPT = *COMMA
```

SET PATH

SET PATH ステートメントは、CURRENT PATH 特殊レジスターの値を変更します。

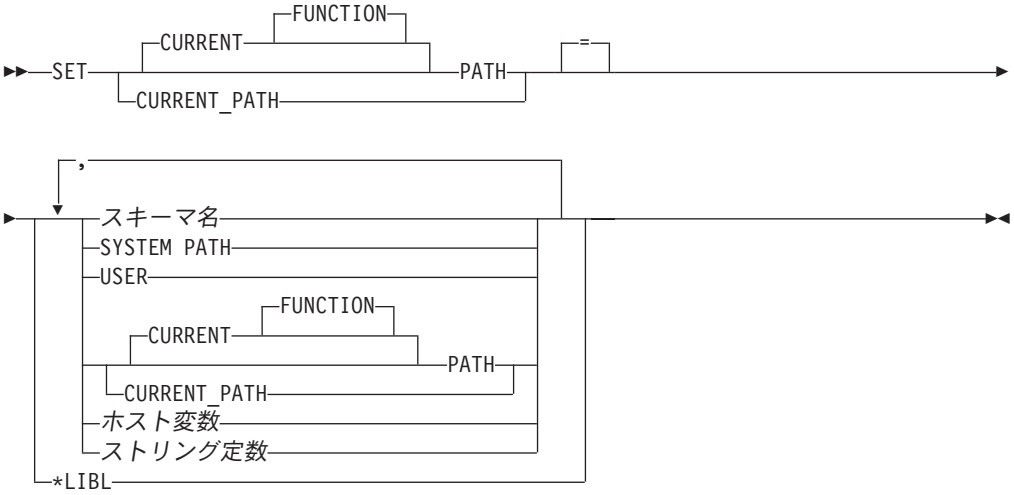
呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。このステートメントは、動的に準備できる実行可能ステートメントです。

権限

このステートメントを実行するための権限は、不要です。

構文



説明

CURRENT PATH 特殊レジスターの値は、指定された値によって置き換えられます。

スキーマ名
スキーマを識別します。パスの設定時には、そのスキーマが存在するかどうかについての検査は行われません。

SYSTEM PATH
この値は、スキーマ名 ″QSYS″、″QSYS2″ を指定する場合と同じです。

USER
この値は USER 特殊レジスターです。

CURRENT PATH
このステートメントの実行前の CURRENT PATH 特殊レジスターの値。

ホスト変数
コンマで区切られた、1 つまたは複数のスキーマ名を含むホスト変数。

ホスト変数は、次の条件に合っていなければなりません。

- 文字ストリング変数である。
- 標識変数を伴っていない。
- 左寄せされているスキーマ名を含み、通常 ID の形成の規則に従っている。
- 右側はブランクで埋め込まれている。
- ヌル値ではない。

ストリング定数
コンマで区切られた 1 つまたは複数のスキーマ名をもつ文字定数。

使用上の注意

スキーマ名は、パス内で複数回使用できません。

SET PATH ステートメントは、コミット可能操作ではありません。ROLLBACK は、CURRENT PATH には影響を与えません。

指定可能なスキーマの数は、CURRENT PATH 特殊レジスターの合計長によって制限されます。特殊レジスター・ストリングは、指定された各スキーマ名から末尾ブランクを除去し、二重引用符によって区切り、コンマで各スキーマ名を区切って作成します。作成されたストリングの長さが 3483 バイトを超えると、エラーが戻されます。パスには最大 268 のスキーマ名を表示できます。

CURRENT PATH 特殊レジスターの初期値は、活動化グループ内で実行された最初の SQL ステートメントにシステム命名が使用された場合は、*LIBL になります。最初の SQL ステートメントに SQL 命名が使用された場合、初期値は ″QSYS″、″QSYS2″、″X″ (X は USER 特殊レジスターの値) になります。

スキーマの QSYS と QSYS2 を指定する必要はありません。これらは、パスに含まれない場合、最初のスキーマとして暗黙的に想定されます (この場合、CURRENT PATH 特殊レジスターに含まれていません)。

CURRENT PATH 特殊レジスターは、動的 SQL ステートメント内でユーザー定義特殊タイプおよび関数を解決するために使用されます。詳細については、57 ページの『スキーマと SQL パス』を参照してください。

例

次のステートメントは、CURRENT PATH 特殊レジスターを設定します。

```
SET PATH = FUNC_XYZ, "NewFun98", QSYS2
```

SET RESULT SETS

SET RESULT SETS ステートメントは、外部プロシージャーが iSeries Access クライアントまたは SQL 呼び出しレベル・インターフェースによって呼び出されたとき、または DRDA を使用してリモート・システムからアクセスされたときに、そのプロシージャーから戻される可能性がある 1 つまたは複数の結果セットを識別します。

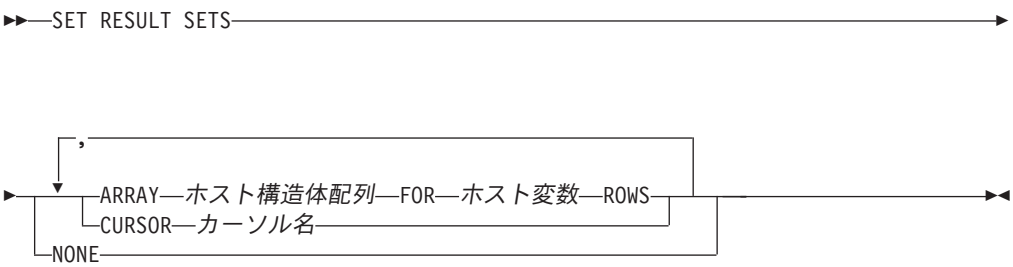
呼び出し

このステートメントは、アプリケーション・プログラムに組み込んで使用します。それ以外の使用法はありません。このステートメントは、動的には準備できない実行可能ステートメントです。 Java または REXX プロシージャーでは使用できません。

権限

権限は不要です。

構文



説明

CURSOR *カーソル名*

プロシージャーから戻される可能性がある結果セットを定義するために使用するカーソルを識別します。この*カーソル名* は、DECLARE CURSOR ステートメントに関する 598 ページの 『説明』で説明している宣言されたカーソルを識別していなければなりません。SET RESULT SETS ステートメントの実行時点では、カーソルはオープン状態であることが必要です。

ARRAY *ホスト構造体配列*

ホスト構造体配列 は、ホスト構造体の宣言に関する規則に従って定義されているホスト構造体の配列を識別します。この配列には、C の NUL で終了するホスト変数を入れることはできません。

配列の最初の構造体が最初の行に対応し、配列の 2 番目の構造体が 2 番目の行に対応するというように、順番に対応しています。さらに、行の最初の値が構造体内の最初の項目に対応し、行の 2 番目の値が構造体の 2 番目の項目に対応するというように、これも順番に対応しています。

DRDA を使用している場合、LOB を配列に入れて戻すことはできません。

1 つの SET RESULT SETS ステートメントで指定できるのは、1 つの配列だけです。

FOR *ホスト変数* **ROWS**
結果セットの行数を指定します。ホスト変数 は、位取りがゼロの数値ホスト変数であることが必要であり、標識変数を含んでいてはなりません。指定する行数は 0 から 32767 までの範囲になければならず、ホスト構造体配列のディメンション以下でなければなりません。

NONE
結果セットを戻さないことを指定をします。プロシージャーが終了した際に、オープン状態のカーソルは戻されません。

使用上の注意

結果セットがプロシージャーから戻されるのは、そのプロシージャーが、iSeries Access ODBC ドライバーを使用するクライアント、または iSeries Access 最適化 SQL API を使用するクライアント、SQL 呼び出しレベル・インターフェース、または JDBC から呼び出される場合だけです。 非 iSeries クライアントが、分散リレーショナル・データベース・アーキテクチャー (DRDA) 接続を使用して、サーバーとしての iSeries にアクセスした場合も、結果セットが戻されます。

外部プロシージャー : 外部プロシージャーから結果セットを戻すには、次の 3 つの方法があります。

- SET RESULT SETS ステートメントがプロシージャーで実行される場合は、その SET RESULT SETS ステートメントが結果セットを識別します。結果セットは、SET RESULT SETS ステートメントで指定した順序で戻されます。
- SET RESULT SETS ステートメントがプロシージャーで実行されない場合
  - WITH RETURN 文節でカーソルが指定されていない場合、プロシージャーがオープンし、オープン状態のまま戻す各カーソルが、それぞれ 1 つの結果セットを識別します。結果セットは、カーソルがオープンされた順序で戻されます。
  - WITH RETURN 文節でカーソルが指定されている場合、WITH RETURN 文節で定義されたカーソルのうち、プロシージャーがオープンし、オープン状態のまま戻す各カーソルが、それぞれ 1 つの結果セットを識別します。 結果セットは、カーソルがオープンされた順序で戻されます。

オープン・カーソルを使用して結果セットが戻される場合、現行カーソル位置から始まる行が戻されます。

プロシージャーから結果セットを戻すには、CREATE PROCEDURE (外部) ステートメントまたは DECLARE PROCEDURE ステートメントで RESULT SETS 文節を指定してください。戻される結果セットの最大数は、CREATE PROCEDURE (外部) ステートメントまたは DECLARE PROCEDURE ステートメントに指定した数を超えることはできません。

SQL プロシージャー : SQL プロシージャーから結果セットを戻すためには、RESULT SETS 文節を指定してプロシージャーを作成する必要があります。 WITH RETURN 文節で定義されたカーソルのうち、プロシージャーがオープンし、オープン状態のまま戻す各カーソルが、それぞれ 1 つの結果セットを識別します。

- プロシージャー内で SET RESULT SETS ステートメントが実行される場合は、その SET RESULT SETS ステートメントに指定されている結果セットが戻されます。結果セットは、SET RESULT SETS ステートメントで指定した順序で戻されます。
- プロシージャー内で SET RESULT SETS ステートメントが実行されない場合は、結果セットはカーソルがオープンされた順序で戻されます。

オープン・カーソルを使用して結果セットが戻される場合、現行カーソル位置から始まる行が戻されます。

SQL プロシージャーからなんらかの結果セットを戻すには、CREATE PROCEDURE (SQL) ステートメントで RESULT SETS 文節を指定する必要があります。戻される結果セットの最大数は、CREATE PROCEDURE ステートメントに指定した数を超えることはできません。

例

次の SET RESULT SETS ステートメントは、カーソル X を、プロシージャーが呼び出されるときに戻される結果セットとして指定します。 ODBC クライアントからの結果セットの使用についての詳細な説明と例については、iSeries Information Center の iSeries Access カテゴリーを参照してください。

```
EXEC SQL SET RESULT SETS CURSOR X;
```

SET SCHEMA

SET SCHEMA ステートメントは、CURRENT SCHEMA 特殊レジスターの値を変更します。

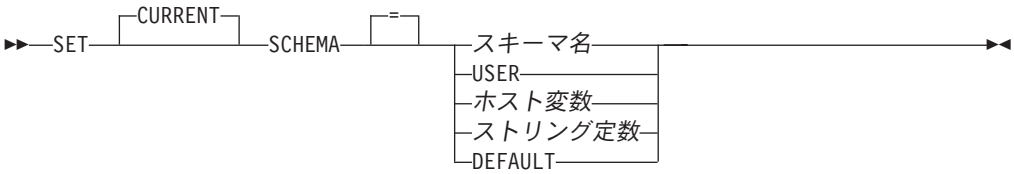
呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。このステートメントは、動的に準備できる実行可能ステートメントです。

権限

このステートメントを実行するための権限は、不要です。

構文



説明

CURRENT SCHEMA 特殊レジスターの値が、指定した値で置き換えられます。

スキーマ名
スキーマを識別します。CURRENT SCHEMA を設定する時点では、このスキーマが存在するかどうかの検査は行われません。

USER
この値は USER 特殊レジスターです。

ホスト変数
スキーマ名を含むホスト変数。

ホスト変数は、次の条件に合っていなければなりません。

- 文字ストリング変数である。
- 標識変数を伴っていない。
- 左寄せされているスキーマ名を含み、通常 ID の形成の規則に従っている。
- 右側はブランクで埋め込まれている。
- ヌル値ではない。

ストリング定数
スキーマ名を含む文字変数。

DEFAULT
CURRENT SCHEMA は初期値に設定されます。 SQL 命名規則の場合の初期値は USER です。システム命名規則の場合の初期値は *LIBL です。

使用上の注意

CURRENT SCHEMA 特殊レジスターの値は、DYNDFTCOL が指定されているプログラムの場合を除き、すべての動的 SQL ステートメントの中のすべての非修飾名の修飾子として使用されます。プログラムの中で DYNDFTCOL が指定されている場合は、CURRENT SCHEMA が示すスキーマ名の代わりに、そのプログラムのスキーマ名が使用されます。

SET SCHEMA ステートメントは、コミット可能操作ではありません。ROLLBACK は、CURRENT SCHEMA には影響を与えません。

SQL 命名規則の場合は、CURRENT SCHEMA 特殊レジスターの初期値は USER になります。システム命名規則の場合は、CURRENT SCHEMA 特殊レジスターの初期値は '*LIBL' です。

CURRENT SCHEMA 特殊レジスターの設定により、CURRENT PATH 特殊レジスターが影響を受けることはありません。したがって、SQL パスには CURRENT SCHEMA は組み込まれないため、関数、プロシージャー、および特殊タイプの解決でこれらのオブジェクトが見つからないことがあります。現行スキーマの値を SQL パスに組み込むには、SET SCHEMA ステートメントを発行するときに、必ず、SET SCHEMA ステートメントからのスキーマ名を含む SET PATH ステートメントも発行するようにしてください。

CURRENT SCHEMA の同義語として、CURRENT SQLID を使用できます。 SET CURRENT SQLID ステートメントの効果は、SET CURRENT SCHEMA ステートメントと同じです。他の効果 (ステートメント権限の変更など) は発生しません。

SET SCHEMA を使用することは、QSQCHGDC API を呼び出すことと同じです。

例

例 1

次のステートメントは、CURRENT SCHEMA 特殊レジスターを設定します。

```
SET SCHEMA = RICK
```

例 2

次の例では、CURRENT SCHEMA 特殊レジスターの現行値を検索して、CURSCHEMA というホスト変数に入れます。

```
EXEC SQL VALUES(CURRENT SCHEMA) INTO :CURSCHEMA
```

値は、例えば例 1 で設定された RICK です。

SET TRANSACTION

SET TRANSACTION ステートメントは、現行の作業単位の分離レベルと読み取り専用属性を設定します。

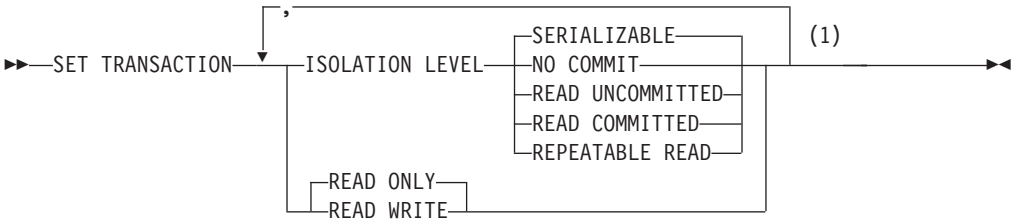
呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことも、対話式に呼び出すことも可能です。このステートメントは、動的に準備できる実行可能ステートメントです。

権限

権限は不要です。

構文



注:

1 ISOLATION LEVEL 文節は 1 つだけ指定でき、また READ WRITE または READ ONLY 文節はどちらか一方を 1 つだけ指定できます。

説明

ISOLATION LEVEL
トランザクションの分離レベルを指定します。 ISOLATION LEVEL 文節を指定しなかった場合は、ISOLATION LEVEL SERIALIZABLE が暗黙指定されます。

NO COMMIT
分離レベル NC (COMMIT(*NONE)) を指定します。

READ UNCOMMITTED
分離レベル UR (COMMIT(*CHG)) を指定します。

READ COMMITTED
分離レベル CS (COMMIT(*CS)) を指定します。

REPEATABLE READ [66]
分離レベル RS (COMMIT(*ALL)) を指定します。

66. REPEATABLE READ は ISO および ANS の標準用語で、DB2 UDB for iSeriesの *ALL の分離レベル、および IBM SQL の読み取り固定 (RS) の分離レベルに対応するものです。SERIALIZABLE は ISO および ANS の標準で、IBM SQL の反復可能読み取り (RR) に対応する用語です。

SERIALIZABLE
分離レベル RR (COMMIT(*RR)) を指定します。

READ WRITE または READ ONLY
このトランザクションでデータ変更操作が許されるかどうかを指定します。

READ WRITE
すべての SQL 操作が許されることを指定します。 ISOLATION LEVEL READ UNCOMMITTED を指定した場合以外は、これがデフォルト値です。

READ ONLY
SQL データを変更しない SQL 操作のみが許されることを指定します。 ISOLATION LEVEL READ UNCOMMITTED を指定した場合は、これがデフォルト値です。

使用上の注意

SET TRANSACTION ステートメントは、そのプロセスの現行活動化グループの SQL ステートメントの分離レベルを設定します。その活動化グループのコミットメント制御の有効範囲がそのジョブの範囲である場合、SET TRANSACTION ステートメントは同一のジョブ・コミット有効範囲を持つ他の活動化グループすべての分離レベルを設定します。

SET TRANSACTION ステートメントは、トリガーで実行される場合を除き、作業単位の最初の SQL ステートメントである場合にのみ実行することができます。トリガー・プログラムでは、READ ONLY を指定した SET TRANSACTION はコミット境界でのみ使用できます。トリガーでは、いつでも SET TRANSACTION ステートメントを実行することができますが、そのトリガーの最初のステートメントとして実行することをお勧めします。SET TRANSACTION ステートメントがトリガー内で役立つのは、トリガーの中の SQL ステートメントの分離レベルを、そのトリガーを起動させたアプリケーションと同じレベルに設定する場合です。

現行接続がリモート・サーバーとの接続である場合は、SET TRANSACTION ステートメントは、現行サーバーのトリガーに入っていないかぎり、使用できません。SET TRANSACTION ステートメントが実行されると、該当の作業単位がコミットまたはロールバックされるまで、CONNECT および SET CONNECTION ステートメントは使用できません。

SET TRANSACTION ステートメントの有効範囲は、そのステートメントが実行される文脈に基づいています。トリガーで SET TRANSACTION ステートメントが実行される場合は、指定した分離レベルは、別の SET TRANSACTION ステートメントが実行されるか、またはそのトリガーが終了するかのいずれかが起こるまで、後続のすべての SQL ステートメントに適用されます。 SET TRANSACTION ステートメントがトリガーの外部で実行される場合は、指定した分離レベルは、COMMIT または ROLLBACK 操作が行われるまで、後続のすべての SQL ステートメントに適用されます (ただし、トリガー内にあって、SET TRANSACTION の後で実行されるステートメントを除きます)。

SET TRANSACTION ステートメントは、SET TRANSACTION ステートメントの実行時にまだオープンされている WITH HOLD カーソルに対しては無効です。

分離レベルの詳細については、24 ページの 『分離レベル』を参照してください。

同義のキーワード

以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- NO COMMIT の同義語として、キーワード NC または NONE を使用できます。
- READ UNCOMMITTED の同義語として、UR および CHG を使用できます。
- READ COMMITTED の同義語として、キーワード CS を使用することができます。
- REPEATABLE READ の同義語として、キーワード RS または ALL を使用できます。
- SERIALIZABLE の同義語として、キーワード RR を使用できます。

例

例 1

次の SET TRANSACTION ステートメントは、分離レベルを NONE に設定します (SQL プリコンパイラーのコマンドで *NONE を指定するのと同等です)。

```
EXEC SQL SET TRANSACTION ISOLATION LEVEL NO COMMIT;
```

例 2

次の SET TRANSACTION は、分離レベルを SERIALIZABLE に設定します。

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
```

SET 遷移変数

SET 遷移変数ステートメントは、新しい*遷移変数* に値を割り当てます。

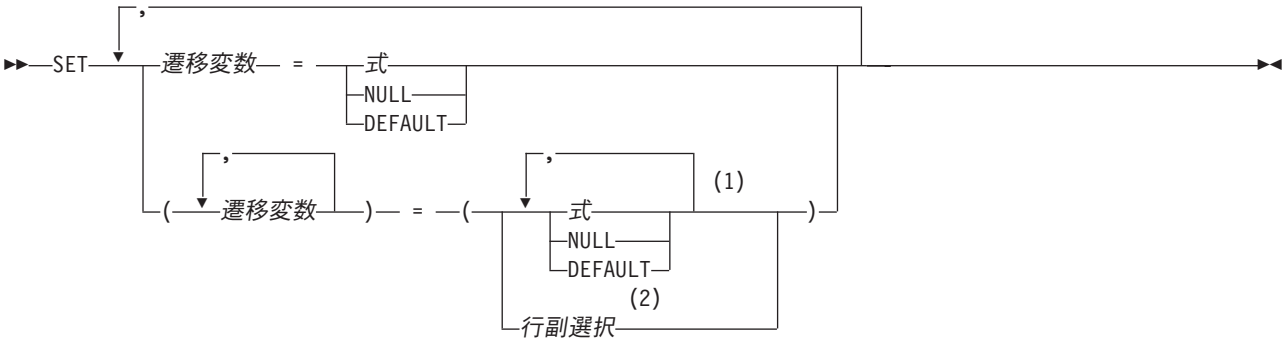
呼び出し

このステートメントは、BEFORE トリガー内の SQL ステートメントとしてのみ使用できます。このステートメントは、動的には準備できない実行可能ステートメントです。

権限

行副選択 が指定されている場合、339 ページの『第 4 章 照会』で各副選択に必要な権限についての説明を参照してください。

構文



注:

1 式、NULL、および DEFAULT の数は、*遷移変数* の数と同じでなければなりません。

2 選択リストの中の列の数は、*遷移変数* の数と同じでなければなりません。

説明

遷移変数
新しい行の中のどの列を更新するかを指定します。*遷移変数* は、トリガーの対象表の中の列を示すものでなければならず、必要に応じて、新しい値を示す相関名により修飾することができます。 OLD *遷移変数* を指定してはなりません。

各*遷移変数* のデータ・タイプは、それぞれに対応する結果列と互換性のあるものでなければなりません。値は、列への割り当ての規則に従って*遷移変数* に割り当てられます。詳しくは、85 ページの『割り当ておよび比較』を参照してください。

式 *遷移変数* の新しい値を指定します。式 は、136 ページの 『式』で説明しているタイプの任意の式です。式には列関数を含めることはできません。

式 には、OLD および NEW *遷移変数* に対する参照を含めることができます。CREATE TRIGGER ステートメントに OLD 文節と NEW 文節の両方が含まれている場合は、*遷移変数* を*相関名* で修飾して、どちらの*遷移変数* かを明示する必要があります。

NULL
ヌル値を指定します。NULL は、ヌル可能列に対してのみ指定できます。

DEFAULT
遷移変数 に関連した列のデフォルト値を使用することを指定します。この列が IDENTITY 列であるか、ROWID データ・タイプの列である場合は、データベース・マネージャーが値を生成します。

行副選択
1 つの結果行を戻す副選択。結果列の値が、それぞれ対応する*遷移変数* に割り当てられます。副選択の結果に行が含まれない場合、ヌル値が割り当てられます。結果の中に複数の行がある場合には、エラーが戻されます。

使用上の注意

複数の割り当て
同じ SET 遷移変数ステートメントに複数の割り当てが含まれている場合は、割り当てを行う前にすべての式 が評価されます。したがって、式の中での*遷移変数* に対する参照は、常に、この SET ステートメントで割り当てが行われる前の*遷移変数* の値です。

例

例 1

給与列の値が 50000 を超えないようにします。新しい値が 50000 より大きい場合は、50000 に設定します。

```
CREATE TRIGGER LIMIT_SALARY
  BEFORE INSERT ON EMPLOYEE
  REFERENCING NEW AS NEW_VAR
  FOR EACH ROW MODE DB2SQL
  WHEN (NEW_VAR.SALARY > 50000)
    BEGIN ATOMIC
    SET NEW_VAR.SALARY = 50000;
  END
```

例 2

職名が更新されたときに、新しい職名に基づいて給与が増額されるようにします。そして、その地位での年数を 0 に設定します。

```
CREATE TRIGGER SET_SALARY
  BEFORE UPDATE OF JOB ON STAFF
  REFERENCING OLD AS OLD_VAR
              NEW AS NEW_VAR
  FOR EACH ROW MODE DB2SQL
  BEGIN ATOMIC
    SET (NEW_VAR.SALARY, NEW_VAR.YEARS) =
        (OLD_VAR.SALARY * CASE NEW_VAR.JOB
            WHEN 'Sales' THEN 1.1
            WHEN 'Mgr'   THEN 1.05
            ELSE 1 END ,0);
  END
```

SET 変数

SET 変数ステートメントは、1 行以内で構成される結果表を作成し、その行の値をホスト変数に割り当てます。

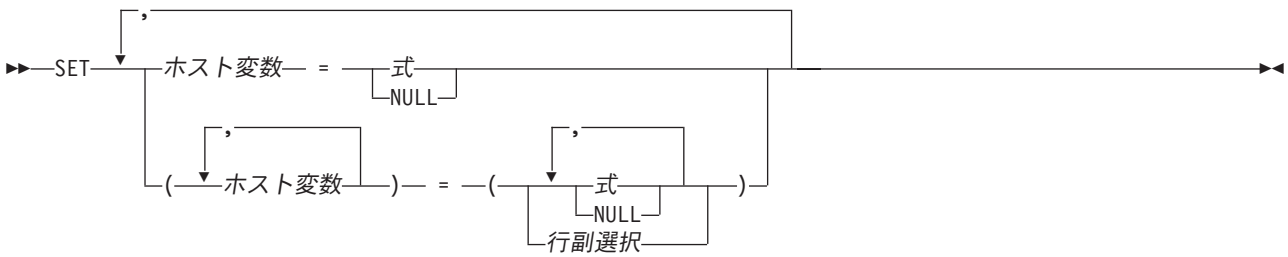
呼び出し

このステートメントは、アプリケーション・プログラムに組み込んで使用します。それ以外の使用法はありません。このステートメントは、動的には準備できない実行可能ステートメントです。

権限

行副選択 が指定されている場合、 339 ページの『第 4 章 照会』で各副選択に必要な権限についての説明を参照してください。

構文



説明

ホスト変数、...

1 つまたは複数のホスト変数、あるいはホスト構造体を指定します。これらは、ホスト変数の宣言に関する規則に従って宣言されていなければなりません。(121 ページの『ホスト変数に対する参照』を参照)。ホスト構造体は、ホスト構造体の各エレメントを表すホスト変数のリストによって、論理的に置き換えられます。

各ホスト変数 に割り当てられる値は、ホスト変数 の直後に指定することができます。例えば、ホスト変数 = 式、ホスト変数 = 式 のように指定します。 または、対になっている括弧を使用すると、すべてのホスト変数 とすべての値を指定することができます。つまり、(ホスト変数、ホスト変数) = (式、式) などのように指定します。

各ホスト変数のデータ・タイプは、それぞれに対応する結果列と互換性がなければなりません。割り当ては、それぞれ 85 ページの『割り当ておよび比較』で説明されている規則に従って行われます。等号演算子の左側に指定するホスト変数の数は、それに対応して等号演算子の右側に指定されている結果の値の数と同じでなければなりません。値がヌルの場合は、標識変数が用意されている必要があります。割り当てでエラーが起こった場合、その値は変数に割り当てられず、そ

れ以後の変数への値の割り当ては行われません。ただし、変数にすでに割り当てられている値があれば、その値は割り当てられたままです。

副選択の式 あるいは SELECT リストの算術式の結果、エラーが発生した (ゼロによる除算やオーバーフローなど) 場合、または文字変換エラーが起こった場合、結果はヌル値になります。他のヌル値の場合と同様に、標識変数を用意しなければなりません。 該当のホスト変数の値は、未定義になります。ただし、この場合、標識変数は -2 にセットされます。ステートメントの処理は、エラーが発生しなかった場合と同様に継続されます。 (ただし、このエラーで正の SQLCODE になります。) 標識変数を用意していない場合は、SQLCA のフィールド SQLCODE に負の値が戻されます。エラーが生じた時点で、すでにいくつかの値がホスト変数に割り当てられていることがあり、それらの値は割り当てられたままになります。

式 ホスト変数の新しい値を指定します。式 は、136 ページの 『式』で説明しているタイプの任意の式です。この式の中で列名を使用してはなりません。

NULL
ホスト変数の新しい値をヌル値にすることを指定します。

行副選択
1 つの結果行を戻す副選択。結果列の値は、対応する各ホスト変数 に割り当てられます。副選択の結果に行が含まれない場合、ヌル値が割り当てられます。結果の中に複数の行がある場合には、エラーが戻されます。

使用上の注意

ホスト変数として文字変数を指定し、その変数が、結果を収容するのに十分な大きさ持っていない場合には、SQLCA の SQLWARN1 に 'W' が割り当てられます。標識変数が用意されている場合、結果の実際の長さは、そのホスト変数に関連する標識変数に戻されます。

ホスト変数として C の NUL で終了するホスト変数を指定し、その変数が、結果および NUL 終了文字を入れられるだけの十分な大きさをもっていない場合は、以下のようになります。

- CRTSQLCI コマンドまたは CRTSQLCPPI コマンドに *CNULRQD オプションを指定した場合 (または SET OPTION ステートメントに CNULRQD(*YES) を指定した場合)、以下のようになります。
  - 結果が切り捨てられます。
  - 最後の文字は NUL 終了文字になります。
  - SQLCA の SQLWARN1 に ‘W’ が割り当てられます。
- CRTSQLCI コマンドまたは CRTSQLCPPI コマンドに *NOCNULRQD オプションを指定した場合 (または SET OPTION ステートメントに CNULRQD(*NO) を指定した場合)、以下のようになります。
  - NUL 終了文字は戻されません。
  - SQLCA の SQLWARN1 に ‘N’ が割り当てられます。

例

例 1

CURRENT PATH 特殊レジスターの値を、ホスト変数 HV1 に割り当てます。

```
EXEC SQL SET :HV1 = CURRENT PATH;
```

例 2

LOB ロケーター LOB1 が CLOB 値と関連していると想定します。CLOB 値の一部を、LOB ロケーターを使用してホスト変数 DETAILS に割り当てます。

```
EXEC SQL SET :DETAILS = SUBSTR(:LOB1,1,35);
```

UPDATE

UPDATE ステートメントは、表またはビューの行の指定した列の値を更新します。ビューの行を更新すると、そのビューの基本表の行が更新されます。

このステートメントには、以下の 2 つの形式があります。

- *検索* UPDATE 形式。この形式は、1 つまたは複数の行を更新するために使用します (必要に応じて、更新する行を検索条件によって限定することができます)。
- *位置指定* UPDATE 形式。この形式は、1 つの行だけを更新するために使用します (更新される行は、カーソルの現在位置によって決まります)。

呼び出し

検索 UPDATE ステートメントは、アプリケーションに組み込むか、または対話式に呼び出すことができます。位置指定 UPDATE ステートメントは、アプリケーション・プログラムに組み込んで使用しなければなりません。どちらの形式も、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメントに指定された表またはビューに対して、
  - 表やビューに対する UPDATE 特権、または
  - 更新する各列に対する UPDATE 特権、または
  - その表の所有権、および
  - 表やビューが入っているライブラリーに対する *EXECUTE システム権限
- 管理権限

ステートメントの権限 ID は、次のいずれかの場合に表 (または表の指定された列) に対する UPDATE 特権を保有します。

- その表の所有者である。
- その表またはその表の列に対する UPDATE 特権が認可されている。
- その表に対する *OBJOPR および *UPD のシステム権限が認可されている。

ステートメントの権限 ID は、次のいずれかの場合にビューに対する UPDATE 特権を保有します。[67]

- そのビューまたはそのビューの列に対する UPDATE 特権が認可されている。
- そのビューに対する *OBJOPR および *UPD システム権限が認可されており、さらに、そのビューの定義の最初の FROM 文節の最初の表またはビューに対する *UPD システム権限が認可されている。また、これがビューである場合は、そのビューの定義の最初の FROM 文節の最初の表またはビューに対する *UPD システム権限が認可されている。以下同様。

67. ビューが作成される時点で、所有者は必ずしもそのビューに対して UPDATE 特権を獲得するとは限りません。所有者が UPDATE 特権を獲得するのは、そのビューが更新を許可し、しかも所有者が副選択で参照されている最初の表に対して UPDATE 特権を保有している場合だけに限られます。

割り当て文節 の式 にその表またはビューの列に対する参照が含まれている場合、または検索 UPDATE の*検索条件* にその表またはビューの列に対する参照が含まれている場合は、ステートメントの権限 ID が保持する特権には、以下のいずれか 1 つも含まれていなければなりません。

- その表またはビューについての SELECT 特権
- 管理権限

ステートメントの権限 ID は、以下の場合に表に対する SELECT 特権を持ちます。

- その表の所有者である。
- その表に対する SELECT 特権が認可されているか、または
- その表についての *OBJOPR および *READ システム権限が認可されている。

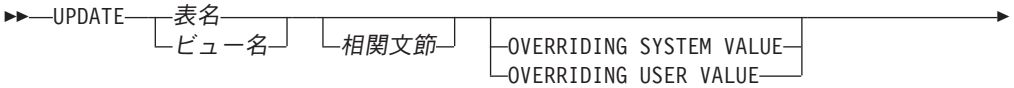
ステートメントの権限 ID は、以下の場合にビューに対する SELECT 特権を持ちます。

- そのビューの所有者である。
- そのビューに対する SELECT 特権が認可されているか、または
- そのビューについての *OBJOPR および *READ システム権限、およびそのビューが直接または間接に従属しているすべての表やビューについての *READ システム権限が認可されている。すなわち、そのビューの定義で参照されている表やビューのすべて、またそのようなビューが参照される場合、そのビューの定義で参照されている表やビューのすべて、以下同様。

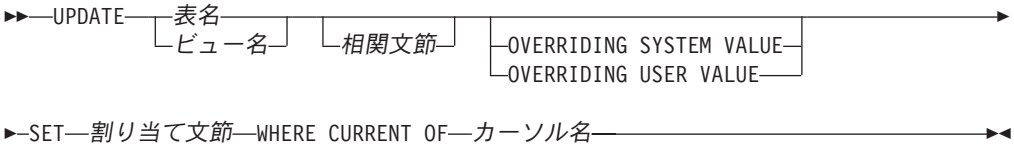
検索条件 に副照会が含まれている場合、または*割り当て文節* に*スカラー副選択* か*行副選択* が含まれている場合、339 ページの『第 4 章 照会』で、各副選択に必要な権限の説明を参照してください。

構文

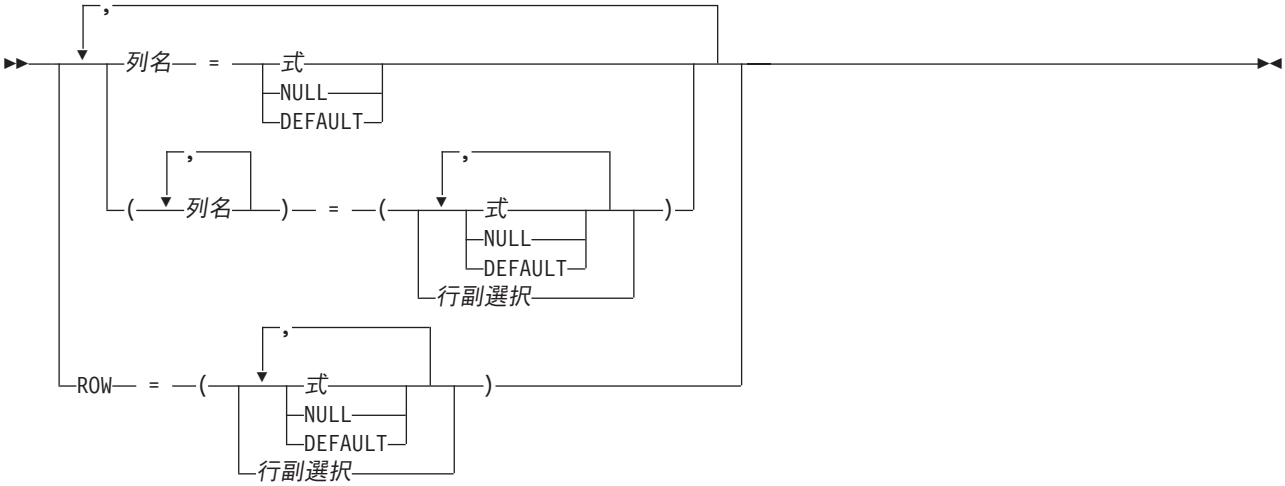
検索 UPDATE:



位置指定 UPDATE:



割り当て文節 :



分離文節:

説明

表名 または*ビュー名*

更新する表またはビューを指定します。この名前は、現行サーバーに存在している表またはビューを識別していなければなりませんが、カタログ表、カタログ表のビュー、または読み取り専用のビューを識別するものであってはなりません。読み取り専用ビューおよび更新可能ビューについての説明は、589 ページの『CREATE VIEW』を参照してください。

相関文節

検索条件 または*割り当て文節* の中で、表またはビューを指定するために使用できます。*相関文節* の詳細については、 344 ページの『表参照』を参照してください。*相関名* の説明については、 115 ページの『相関名』を参照してください。

OVERRIDING SYSTEM VALUE または OVERRIDING USER VALUE

特定の ROWID または識別列についてシステムが生成した値またはユーザーが指定した値を使用するかどうかを指定します。 OVERRIDING SYSTEM VALUE を指定する場合は、SET 文節内の暗黙または明示的な列リストに、GENERATED ALWAYS として定義された列が含まれていることが必要です。OVERRIDING USER VALUE を指定する場合は、INSERT ステートメントの対象とする暗黙または明示的な列リストに、GENERATED ALWAYS または GENERATED BY DEFAULT として定義された列が含まれていることが必要です。

OVERRIDING SYSTEM VALUE

GENERATED ALWAYS として定義されている列について、SET 文節に指定されている値を使用することを指定します。システム生成の値は使用されません。

OVERRIDING USER VALUE

GENERATED ALWAYS または GENERATED BY DEFAULT として定義されている列について、SET 文節に指定されている値を無視することを指定します。代わりにシステム生成の値が使用され、ユーザー指定の値はオーバーライドされます。

OVERRIDING SYSTEM VALUE と OVERRIDING USER VALUE のどちらも指定しない場合は、以下のようになります。

- GENERATED ALWAYS として定義されている ROWID または識別列については、値を指定することはできません。
- GENERATED BY DEFAULT として定義されている ROWID または識別列については、値を指定することができます。値を指定した場合は、この列にその値が割り当てられます。ただし、BY DEFAULT として定義された ROWID 列の値を更新できるのは、指定した値が、DB2 UDB (OS/390 および z/OS 版) または DB2 UDB for iSeries によりすでに生成されている有効な行 ID の値である場合に限られます。 BY DEFAULT として定義された識別列に値を更新した場合は、その識別列が固有制約または固有索引内の唯一のキーである場合以外は、データベース・マネージャーはその指定された値が該当の列についての固有な値であるかどうかを検査しません。固有制約も固有索引もない場合は、データベース・マネージャーは、NO CYCLE が有効である場合に限り、システム生成の値のセットの中でのみ各値の固有性を保証します。

  値が指定されていない場合は、データベース・マネージャーは新しい値を生成します。

SET

列名への値の割り当てを指定します。

列名

更新する列を識別します。*列名* は、指定した表またはビューの列を識別し

なければなりません。ただし、スカラー関数、定数、または式から得られるビューの列を指定してはなりません。列を複数回指定することはできません。

位置指定 UPDATE の場合 :

- UPDATE 文節をカーソルに関する SELECT ステートメントに指定する場合は、SET リストのそれぞれの列名を、UPDATE 文節にも指定しなければなりません。
- UPDATE 文節をカーソルに関する SELECT ステートメントに指定しない場合は、更新可能な任意の列の名前を指定することができます。

詳しくは、 359 ページの『UPDATE 文節』を参照してください。

1 つのビューに同じ列から得られる 2 つの列がある場合、その列の値を更新することは可能ですが、その 2 つの列を同一の UPDATE ステートメントで更新することはできません。

列名 のリストを指定する場合は、式、NULL、および DEFAULT の数が*列名* の数に一致していなければなりません。

ROW
指定された表またはビューのすべての列を識別します。ビューが指定されている場合、そのビューの列がまったく、スカラー関数、定数、または式から派生していない場合があります。

式、NULL、および DEFAULT の数 (または*行副選択* からの結果列の数) は、行の列の数と一致していなければなりません。

位置指定 UPDATE の場合、カーソルの SELECT ステートメント内に UPDATE 文節が指定されている場合、その UPDATE 文節にも表またはビューの各列を指定しなければなりません。詳細については、358ページの「UPDATE 文節」を参照してください。

ビューに、そのビューの別の列から派生したビュー列が含まれている場合、そのビューに ROW を指定することはできません。これは、両方の列を同じ UPDATE ステートメント内で更新できないためです。

式 列の新しい値を指定します。 式 は、136 ページの 『式』で説明しているタイプの任意の式です。この式の中で、列関数を使用してはなりません。

式の中の*列名* は、指定した表またはビューの列の名前を指定するものでなければなりません。行が更新されるたびに、その行の列の値 (行を更新する前の値) がこの式の列の値になります。

NULL
列の新しい値をヌル値にすることを指定します。NULL は、ヌル可能列にのみ指定してください。

DEFAULT
列にデフォルト値を割り当てることを指定します。使用される値は、次のように、列がどのように定義されたかによって異なります。

- WITH DEFAULT 文節が使用される場合、使用されるデフォルト値は、その列に関して定義されている値になります (542 ページの『CREATE TABLE』の*列定義* の*デフォルト文節* を参照してください)。

- WITH DEFAULT 文節または NOT NULL 文節が使用されない場合、使用される値は NULL です。
- NOT NULL 文節が使用されていて、WITH DEFAULT 文節が使用されていないか、DEFAULT NULL が使用されている場合、その列については DEFAULT キーワードは指定できません。

行副選択

1 つの結果行を戻す副選択。選択リスト内の結果列の数は、割り当てのために指定された*列名* の数 (または ROW が指定されている場合は、その行の列の数) と一致していなければなりません。結果列の値は、対応する各*列名*に割り当てられます。副選択の結果に行が含まれない場合、ヌル値が割り当てられます。結果の中に複数の行がある場合には、エラーが戻されます。

行副選択 には、UPDATE ステートメントのターゲット表の列に対する参照が含まれている場合があります。行が更新されるたびに、その行の列の値 (行を更新する前の値) がこの式の列の値になります。

WHERE

更新する行を指定します。この文節を省略することも、*検索条件* を指定することも、特定のカーソルを指定することもできます。この文節を指定しなかった場合は、指定した表またはビューのすべての行が更新されます。

検索条件

167 ページの 『検索条件』で説明している、いずれかの検索条件を指定します。検索条件の中のそれぞれの*列名* (副照会の中は除く) は、指定した表またはビューにある列の名前を指定するものでなければなりません。UPDATE と副照会の基本オブジェクトが両方とも同じ表になる場合に、検索条件に副照会が含まれるときは、その副照会の評価が完了してから行が更新されます。

検索条件 は、表またはビューの各行に適用されます。更新された行は、*検索条件* の結果が真であるものです。

*検索条件*に副照会が含まれている場合は、いずれかの行に*検索条件* が適用されるたびにその副照会が実行され、副照会の結果が*検索条件* の適用に使用されると考えることができます。実際には、相関参照のない副照会は一度しか実行されないことがあります。各行ごとに 1 回ずつ実行する必要があるのは、相関参照がある副照会です。

CURRENT OF *カーソル名*

更新操作で使用するカーソルを識別します。*カーソル名* は、597 ページの『DECLARE CURSOR』の説明にしたがって宣言されているカーソルを識別しなければなりません。

更新を指定した表またはビューは、このカーソルに関する SELECT ステートメントの FROM 文節でも指定されていなければなりません。また、このカーソルの結果表が読み取り専用であってはなりません。読み取り専用の結果表の説明については、597 ページの『DECLARE CURSOR』を参照してください。

UPDATE ステートメントの実行時点では、カーソルはある行に位置付けられていなければなりません。その行が更新されます。

分離文節

このステートメントに関して使用する分離レベルを指定します。*分離文節* の詳細については、ISOLATION 文節の項を参照してください。

UPDATE の規則

割り当て

更新値は、第 2 章で説明されている割り当て規則に従って、列に割り当てられます。

妥当性

識別された表、または識別されたビューの基礎表が 1 つまたは複数の固有索引または固有制約をもつ場合は、その表の更新される各行は、それらの固有索引によって課せられる制約に適合しなければなりません。

固有索引および固有制約は、COMMIT(*NONE) の指定がある場合を除き、そのステートメントの終わりでチェックされます。複数行更新の場合、これはすべての行が更新された後で行われます。COMMIT(*NONE) が指定されている場合には、各行が更新されるごとにチェックが行われます。

識別された表、または識別されたビューの基礎表が、1 つまたは複数の検査制約を持つ場合、更新された表の各行ごとの、それぞれの検査制約は真または不明でなければなりません。

検査制約は、ステートメントの終わりで必ずチェックされます。複数行更新の場合、これはすべての行が更新された後で行われます。

ビューが識別されている場合は、更新された行は適用される WITH CHECK OPTION に適合しなければなりません。詳細は、589 ページの 『CREATE VIEW』を参照してください。

トリガー

識別された表または識別されたビューの基礎表が更新トリガーを持つ場合、トリガーが起動されます。トリガーが起動された結果、更新される値に応じて、他のステートメントが実行されたり、エラー条件が発生したりすることがあります。

参照保全

親行の親キーの値は変更できません。

更新値がヌル値以外の外部キーを生成する場合、その外部キーは関連する親表の親キーの何らかの値に等しくなければなりません。

参照制約 (RESTRICT 削除規則を伴う参照制約以外の) は、ステートメントの終わりで実際上チェックされます。複数行更新の場合、これはすべての行が更新された後で行われます。

使用上の注意

更新値が何らかの制約に違反している場合、またはその他のエラーが UPDATE ステートメントの実行中に発生し、しかも COMMIT(*NONE) の指定がなかった場合、そのステートメントの実行中に行われた変更はすべて撤回されます。ただし、エラーが発生する前に、その作業単位の中で行われていたその他の変更は撤回されません。COMMIT(*NONE) が指定されていれば、変更が撤回されることはありません。

エラーの発生によって、カーソルの状態が予期できないものになることがあります。

UPDATE ステートメントの実行が完了すると、SQLCA の SQLERRD(3) の値には、更新された行の数がセットされます。SQLCA の説明については、 865 ページの『付録 B. SQL 連絡域』を参照してください。

UPDATE ステートメントが正しく実行されると、該当するロックがすでに存在する場合を除いて、1 つまたは複数の排他ロックが確立されます。これらのロックがコミットまたはロールバックの操作によって解放されるまで、更新された行へのアクセスは、以下に限定されます。

- その更新を行ったアプリケーション・プロセス
- 読み取り専用カーソル、SELECT INTO ステートメント、または副照会を介して、COMMIT(*NONE) または COMMIT(*CHG) を使用する別のアプリケーション・プロセス

ロックは、他のアプリケーション・プロセスがその表の操作を行うのを防止します。ロックについての詳細は、COMMIT、ROLLBACK、および LOCK TABLE ステートメント、および 24 ページの 『分離レベル』の分離レベルの項を参照してください。また、DB2 UDB for iSeries データベース・プログラミングも参照してください。

COMMIT(*RR)、COMMIT(*ALL)、COMMIT(*CS)、または COMMIT(*CHG) が指定されている場合は、1 つの UPDATE ステートメントで、最高 500 000 000 行を更新または変更することができます。変更される行の数には、トリガーの結果として同じコミットメント定義のもとで挿入、更新、または削除される行が含まれます。

ホスト変数は、REXX プロシージャー内の UPDATE ステートメントでは使用できません。UPDATE を使用する場合は、必ず、パラメーター・マーカーを使用する PREPARE および EXECUTE の対象として使用してください。

DATALINK 列の URL 値を更新した場合、それは古い DATALINK 値を削除して新しい値を挿入するのと同じ結果になります。まず、古い値がいずれかのファイルにリンクしていた場合は、そのファイルへのリンクが解除されます。次に、その DATALINK 値のリンケージ属性が空であれば、指定したファイルがその列にリンクされます。

DATALINK 列のコメント値は、URL パスとして (例えば DLVALUE スカラー関数のデータ位置引き数として) 空のストリングを指定するか、または新しい値に古い値と同じ値を指定することにより、ファイルに再リンクせずに更新することができます。 DATALINK 列をヌル値で更新した場合は、既存の DATALINK 値を削除した場合と同じ結果になります。

既存の値または新しい値のいずれかのファイル・サーバーがデータベース・サーバーに登録されていない場合は、DATALINK 値を更新しようとしたときにエラーが起きることがあります。

同義のキーワード : 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- キーワード NONE を NC の同義語として使用することができます。
- キーワード CHG を UR の同義語として使用することができます。
- キーワード ALL を RS の同義語として使用することができます。

例

例 1

表 EMPLOYEE の従業員番号 (EMPNO) が '000290' のジョブ (JOB) を、'LABORER' に変更します。

```
UPDATE EMPLOYEE
  SET JOB = 'LABORER'
  WHERE EMPNO = '000290'
```

例 2

表 PROJECT で、部門 'D21' が担当しているすべてのプロジェクトのプロジェクト人員数 (PRSTAFF) を 1.5 増やします。

```
UPDATE PROJECT
  SET PRSTAFF = PRSTAFF + 1.5
  WHERE DEPTNO = 'D21'
```

例 3

部門 (WORKDEPT) 'E21' の管理担当者を除くすべての従業員が、一時的に解雇されました。これを示すために、表 EMPLOYEE の対象従業員のジョブ (JOB) を NULL に、給与 (SALARY、BONUS、COMM) の値をゼロに変更します。

```
UPDATE EMPLOYEE
  SET JOB=NULL, SALARY=0, BONUS=0, COMM=0
  WHERE WORKDEPT = 'E21' AND JOB <> 'MANAGER'
```

例 4

C プログラムで、表 EMPLOYEE にある行を表示した上で、要求があれば、特定の従業員のジョブ (JOB) を入力された新しいジョブに変更します。

```
void main ()
  {
   EXEC SQL BEGIN DECLARE SECTION ;
   char change[4];
   char newjob[20];
   EXEC SQL END DECLARE SECTION ;
   EXEC SQL INCLUDE SQLCA ;

   EXEC SQL  DECLARE C1 CURSOR FOR
               SELECT *
                 FROM EMPLOYEE
                 FOR UPDATE OF JOB;

   EXEC SQL  OPEN C1;

   EXEC SQL  FETCH C1 INTO ...     ;

   getlist(change);
   if (strcmp(change, "YES") )
     {
       EXEC SQL  UPDATE EMPLOYEE
                   SET JOB = :newjob
                   WHERE CURRENT OF C1;
     }
```

```
  EXEC SQL  CLOSE C1;
  return;
}
```

VALUES

VALUES ステートメントは、トリガーからユーザー定義関数を呼び出す方式を提供します。遷移変数をユーザー定義関数に渡すことができます。

呼び出し

このステートメントは、トリガーのトリガー・アクション内でのみ使用できます。

権限

行副選択 が指定されている場合、 339 ページの『第 4 章 照会』で各副選択に必要な権限についての説明を参照してください。

構文

説明

VALUES
1 つ以上の列から成る 1 行を導き出します。

式
136 ページの 『式』で説明しているタイプの任意の式です。ホスト変数を含めてはなりません。

NULL
ヌル値を指定します。

行副選択
1 つの結果行を戻す副選択。副選択の結果に行が含まれない場合、ヌル値が戻されます。結果の中に複数の行がある場合には、エラーが戻されます。

使用上の注意

式は評価されますが、結果値は廃棄され、出力変数に割り当てられません。式の一部としてユーザー定義関数が指定されている場合、ユーザー定義関数が呼び出されます。関数が呼び出されたときに、負の SQLCODE が戻された場合、データベース・マネージャー はトリガーの実行を停止し、実行されたトリガー・アクションをすべてロールバックします (トリガーが、分離レベル *NONE で実行されていない場合)。

例

例

トリガーが起動されたときにユーザー定義関数 NEWEMP を呼び出す、後トリガー EMPISRT1 を作成します。表 EMP に対する挿入操作は、トリガーを起動します。新規の従業員番号、ラストネーム、およびファーストネームの遷移変数を、ユーザー定義関数に渡します。

```
CREATE TRIGGER EMPISRT1
   AFTER INSERT ON EMPLOYEE
   REFERENCING NEW AS N
   FOR EACH ROW
   MODE DB2SQL
   BEGIN ATOMIC
     VALUES( NEWEMP(N.EMPNO, N.LASTNAME, N.FIRSTNAME));
   END
```

VALUES INTO

VALUES INTO ステートメントは、1 行以内で構成される結果表を作成し、その行の値をホスト変数に割り当てます。

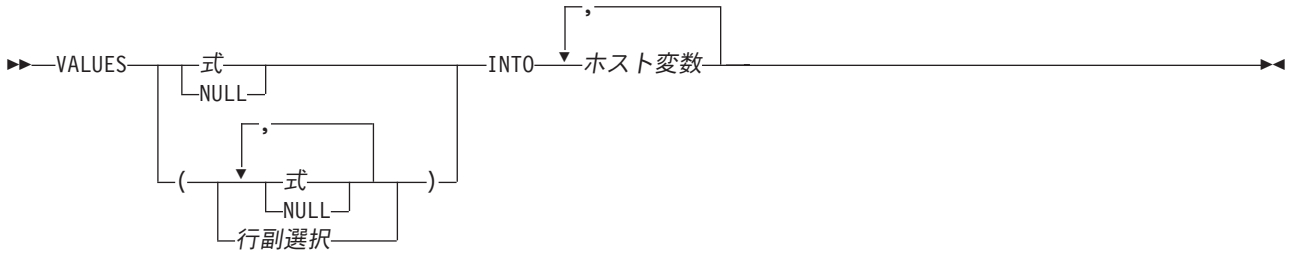
呼び出し

このステートメントは、アプリケーション・プログラムに組み込んで使用します。それ以外の使用法はありません。このステートメントは、動的に準備できる実行可能ステートメントです。

権限

行副選択 が指定されている場合、339 ページの『第 4 章 照会』で各副選択に必要な権限についての説明を参照してください。

構文



説明

VALUES

1 つ以上の列から成る 1 行を導き出します。

式 ホスト変数の新しい値を指定します。式 は、136 ページの 『式』で説明しているタイプの任意の式です。この式の中で列名を使用してはなりません。ホスト構造体はサポートされません。

NULL

ホスト変数の新しい値をヌル値にすることを指定します。

行副選択

1 つの結果行を戻す副選択。結果列の値は、対応する各ホスト変数 に割り当てられます。副選択の結果に行が含まれない場合、ヌル値が割り当てられます。結果の中に複数の行がある場合には、エラーが戻されます。

INTO

ホスト変数およびホスト構造体のリストを指定します。結果の行の最初の値がリストの最初のホスト変数に割り当てられ、2 番目の値が 2 番目のホスト変数に割り当てられます。以下同様です。割り当ては、それぞれ 85 ページの『割り当ておよび比較』で説明されている規則に従って行われます。

値よりホスト変数の数が少ない場合、SQLCA の SQLWARN3 フィールドに 'W' という値が割り当てられます。(865 ページの『付録 B. SQL 連絡域』を参照してください。) 結果の列の数よりも変数の数が多い場合には、警告は出されない点に注意してください。値がヌルの場合は、標識変数が用意されている必要があります。割り当てでエラーが起こった場合、その値は変数に割り当てられず、それ以後の変数への値の割り当ては行われません。ただし、変数にすでに割り当てられている値があれば、その値は割り当てられたままです。

副選択の式 あるいは SELECT リストの算術式の結果、エラーが発生した (ゼロによる除算やオーバーフローなど) 場合、または文字変換エラーが起こった場合、結果はヌル値になります。他のヌル値の場合と同様に、標識変数を用意しなければなりません。 該当のホスト変数の値は、未定義になります。ただし、この場合、標識変数は -2 にセットされます。ステートメントの処理は、エラーが発生しなかった場合と同様に継続されます。 (ただし、このエラーで正の SQLCODE になります。) 標識変数を用意していない場合は、SQLCA のフィールド SQLCODE に負の値が戻されます。エラーが生じた時点で、すでにいくつかの値がホスト変数に割り当てられていることがあり、それらの値は割り当てられたままになります。

ホスト変数, ...

1 つまたは複数のホスト構造体、あるいはホスト変数を指定します。これらのホスト構造体や変数は、その宣言の規則に従って宣言する必要があります。 121 ページの『ホスト変数に対する参照』を参照してください。INTO の操作形式では、ホスト構造体は、その個々の変数に対する参照に置き換えられます。

使用上の注意

エラーが発生した場合、現行のホスト変数に値は割り当てられません。ただし、LOB 値が含まれている場合、対応するホスト変数が変更されている可能性がありますが、その変数の内容は予測できません。

ホスト変数として文字変数を指定し、その変数が、結果を収容するのに十分な大きさ持っていない場合には、SQLCA の SQLWARN1 に 'W' が割り当てられます。標識変数が用意されている場合、結果の実際の長さは、そのホスト変数に関連する標識変数に戻されます。

ホスト変数として C の NUL で終了するホスト変数を指定し、その変数が、結果および NUL 終了文字を入れられるだけの十分な大きさをもっていない場合は、以下のようになります。

- CRTSQLCI コマンドまたは CRTSQLCPPI コマンドに *CNULRQD オプションを指定した場合 (または SET OPTION ステートメントに CNULRQD(*YES) を指定した場合)、以下のようになります。
  - 結果が切り捨てられます。
  - 最後の文字は NUL 終了文字になります。
  - SQLCA の SQLWARN1 に 'W' が割り当てられます。
- CRTSQLCI コマンドまたは CRTSQLCPPI コマンドに *NOCNULRQD オプションを指定した場合 (または SET OPTION ステートメントに CNULRQD(*NO) を指定した場合)、以下のようになります。

- NUL 終了文字は戻されません。
- SQLCA の SQLWARN1 に 'N' が割り当てられます。

例

例 1

CURRENT PATH 特殊レジスターの値を、ホスト変数 HV1 に割り当てます。

```
EXEC SQL VALUES CURRENT PATH INTO :HV1;
```

例 2

LOB ロケーター LOB1 が CLOB 値と関連していると想定します。CLOB 値の一部を、LOB ロケーターを使用してホスト変数 DETAILS に割り当てます。

```
EXEC SQL VALUES (SUBSTR(:LOB1,1,35)) INTO  :DETAILS;
```

WHENEVER

WHENEVER ステートメントは、指定した例外条件が発生した場合にとるべきアクションを指定します。

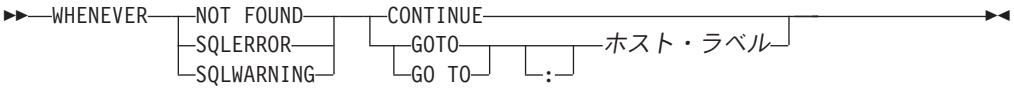
呼び出し

このステートメントは、アプリケーション・プログラムに組み込んで使用します。それ以外の使用法はありません。このステートメントは、実行可能ステートメントではありません。Java または REXX では指定できません。 REXX におけるエラー処理の説明に関しては、DB2 UDB for iSeries ホスト言語での SQL プログラミングを参照してください。

権限

権限は不要です。

構文



説明

NOT FOUND、SQLERROR、または SQLWARNING 文節は、例外条件のタイプを識別するために使用します。

NOT FOUND
SQLCODE が +100 に、SQLSTATE が '02000' になる結果をもたらす、すべての条件を識別します。

SQLERROR
SQLCODE に負の値が戻されるすべての条件を識別します。

SQLWARNING
警告状態 (SQLWARN0 の値が 'W')、または +100 以外の正の SQLCODE、あるいはクラス・コード 01 の SQLSTATE をもたらすすべての条件を識別します。

CONTINUE または GO TO 文節は、指定したタイプの例外条件が存在した場合に、次に実行するステートメントを指定するのに使用します。

CONTINUE
ソース・プログラムにある、次の命令が実行されます。

GOTO または **GO TO** *ホスト・ラベル*
ホスト・ラベル によって識別されるステートメントを実行するように指定します。ホスト・ラベルの部分には、単独のトークンを指定します (必要に応じて、トークン前にコロンを付けます)。このトークンの形式は、ホスト言語によって異なります。例えば、COBOL プログラムでは、*セクション名* または修飾のない*段落名* を指定します。

使用上の注意

WHENEVER ステートメントには、以下の 3 つのタイプがあります。

WHENEVER NOT FOUND
WHENEVER SQLERROR
WHENEVER SQLWARNING

プログラム内のそれぞれの実行可能 SQL ステートメントは、どれか 1 つのタイプの暗黙または明示的な WHENEVER ステートメントの有効範囲内に含まれます。WHENEVER ステートメントの有効範囲は、プログラム内のステートメントのリスト順によって決まり、実行順序には関連がありません。

SQL ステートメントは、ソース・プログラム内でそのステートメントの前に指定されている最後の WHENEVER ステートメント (上記の 3 つのタイプのどれか) の有効範囲内に含まれます。SQL ステートメントの前に、いずれのタイプの WHENEVER ステートメントも指定されていなければ、その SQL ステートメントは、CONTINUE が指定されているタイプの暗黙的 WHENEVER ステートメントの有効範囲に含まれます。

SQL は、COBOL、C、および RPG でのネストされたプログラムをサポートします。しかし、SQL は通常の COBOL、C、または RPG の有効範囲規則に従いません。つまり、ネストされたプロシージャーよりも前にプログラム・ソースで指定された最後の WHENEVER ステートメントが、まだ、そのネストされたプロシージャーについては有効です。 WHENEVER ステートメントで参照されるラベルは、その内部プログラムで複写されたものである必要があります。他に、その内部プログラムで新しい WHENEVER ステートメントを指定することもできます。

FORTRAN では、WHENEVER ステートメントの有効範囲は、同じサブプログラム内の SQL ステートメントに限定されます。

例

以下の目的で、COBOL プログラムに組み込む必要があるステートメントを書きます。

1. ステートメントでエラーが発生した場合は、必ずラベル HANDLER に進む。

```
EXEC SQL  WHENEVER SQLERROR GOTO HANDLER  END-EXEC.
```

2. どのステートメントで警告が発生しても処理は続行する。

```
EXEC SQL  WHENEVER SQLWARNING CONTINUE END-EXEC.
```

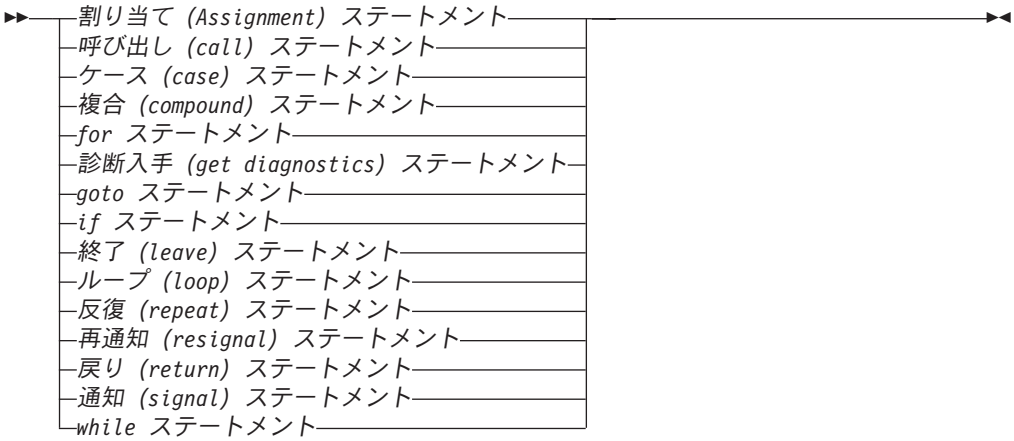
3. データを戻すべきステートメントがデータを戻さなかった場合は、ラベル ENDDATA に進む。

```
EXEC SQL  WHENEVER NOT FOUND GOTO ENDDATA  END-EXEC.
```

第 6 章 SQL 制御ステートメント

制御ステートメントは SQL ステートメントの一種で、これを使用することで、構造化プログラミング言語でプログラムを書く場合と同じような方法で SQL が使用できるようになります。 SQL 制御ステートメントは、ロジック・フローを制御し、変数の宣言と設定をし、警告および例外を処理する能力を提供します。一部の SQL 制御ステートメントには、ネストされた他の SQL ステートメントが組み込まれることもあります。

構文



制御ステートメントは、SQL プロシージャー、SQL 関数、および SQL トリガーでサポートされています。

SQL プロシージャーは、CREATE PROCEDURE ステートメントに LANGUAGE SQL および SQL ルーチン本体を指定して作成します。 SQL 関数は、CREATE FUNCTION ステートメントに LANGUAGE SQL および SQL ルーチン本体を指定して作成します。 SQL ルーチンは、SQL プロシージャーまたは SQL 関数です。SQL トリガーは、CREATE TRIGGER ステートメントに SQL ルーチン本体を指定して作成します。

SQL ルーチン本体は、単一の SQL プロシージャー・ステートメントでなければならず、SQL 制御ステートメントであっても構いません。

SQL ルーチン本体は、プロシージャー、関数、またはトリガーの実行可能部分で、データベース・マネージャーによってプログラムまたはサービス・プログラムに変換されます。 SQL ルーチンまたはトリガーが作成されるときに、SQL は、組み込み SQL ステートメントを含む C ソース・コードが入った一時ソース・ファイル (QTEMP/QSQLSRC) を作成します。DBGVIEW(*SOURCE) が指定されていた場合は、SQL は、ルーチンまたはトリガー用のルート・ソースを一時ソース・ファイル QTEMP/QSQDSRC の中に作成します。

SQL 制御ステートメント

SQL プロシージャーまたは SQL トリガーは、CRTPGM コマンドを使用してプログラム (*PGM) オブジェクトとして作成されます。 SQL 関数は、CRTSRVPGM コマンドを使用してサービス・プログラム (*SRVPGM) オブジェクトとして作成されます。 このプログラムまたはサービス・プログラムは、プロシージャー名、関数名、またはトリガー名の暗黙的または明示的な修飾子となるライブラリー内に作成されます。

プログラムまたはサービス・プログラムが作成されると、制御ステートメント以外の SQL ステートメントは、そのプログラムまたはサービス・プログラム内の組み込み SQL ステートメントになります。

指定されたプロシージャーまたは関数は、SYSROUTINES および SYSPARMS カタログ表内で登録され、プログラムに対する内部リンクが SYSROUTINES から作成されます。プロシージャーが SQL CALL ステートメントを使用して呼び出されるとき、あるいは関数が SQL ステートメント内で呼び出されるときに、そのルーチンに関連したプログラムが呼び出されます。指定された SQL トリガーは、SYSTRIGGER カタログ表内で登録されます。

この章の以後の部分では制御ステートメントについて説明します。説明には、構文図、意味の説明、使用上の注意、および SQL ルーチン本体を構成するステートメントの使用例が含まれています。

821 ページの『SQL パラメーターおよび変数の参照』には、SQL パラメーターおよび変数の参照に関するセクションもあります。特定の SQL 制御ステートメントを記述するときに使用される、2 つの共通エレメントがあります。次の 2 つです。

- SQL 制御ステートメント (前の説明を参照)。
- 822 ページの『SQL プロシージャー・ステートメント』

SQL 制御ステートメントの構文および追加情報については、次のトピックを参照してください。

- 823 ページの『割り当て (Assignment) ステートメント』
- 825 ページの『呼び出し (call) ステートメント』
- 826 ページの『ケース (case) ステートメント』
- 828 ページの『複合 (compound) ステートメント』
- 842 ページの『IF ステートメント』
- 835 ページの『FOR ステートメント』
- 837 ページの『診断入手 (get diagnostics) ステートメント』
- 840 ページの『GOTO ステートメント』
- 844 ページの『ITERATE ステートメント』
- 845 ページの『終了 (leave) ステートメント』
- 847 ページの『ループ (loop) ステートメント』
- 849 ページの『反復 (repeat) ステートメント』
- 851 ページの『再通知 (resignal) ステートメント』
- 854 ページの『戻り (return) ステートメント』
- 856 ページの『通知 (signal) ステートメント』
- 859 ページの『WHILE ステートメント』

SQL パラメーターおよび変数の参照

SQL パラメーターおよび SQL 変数は、ホスト変数 が指定できる SQL プロシージャー・ステートメント内の任意の場所で参照することができます。

SQL パラメーターは、ルーチン内の任意の場所で参照でき、そのルーチン名で修飾することができます。SQL 変数は、それらの変数が宣言されている復号ステートメント内の任意の場所で参照することができ、その復号ステートメントの先頭に指定されているラベル名で修飾することができます。

SQL パラメーターと SQL 変数はすべてヌル可能と見なされます。SQL 変数は、NOT NULL として明示的に宣言できます。SQL ルーチン内の SQL パラメーターまたは SQL 変数の名前は、そのルーチン内で参照される表またはビュー内の列の名前と同じにすることができます。この場合、その名前を明示的に修飾して、それが列であるか、SQL 変数であるか、SQL パラメーターであるかを指示する必要があります。

その名前を修飾しない場合、次の規則によって、その名前が列を参照するのか、あるいは SQL 変数または SQL パラメーターを参照するのかが記述されます。

- SQL ルーチン本体内に指定されている表またはビューが、ルーチンの作成時に存在している場合、その名前は最初に列名としてチェックされます。列として見つからなければ、その名前は、複合内の SQL 変数名としてチェックされ、次に SQL パラメーター名としてチェックされます。
- 参照された表またはビューがルーチンの作成時に存在していない場合、その名前は最初に SQL 変数名としてチェックされ、次に SQL パラメーター名としてチェックされます。 それで見つからなければ、名前は列名と見なされます。

SQL ルーチン内の SQL パラメーターまたは SQL 変数の名前は、特定の SQL ステートメントで使用される ID の名前と同じにすることができます。その名前を修飾しない場合、次の規則によって、その名前が ID を参照するのか、SQL パラメーターまたは SQL 変数を参照するのかが記述されます。

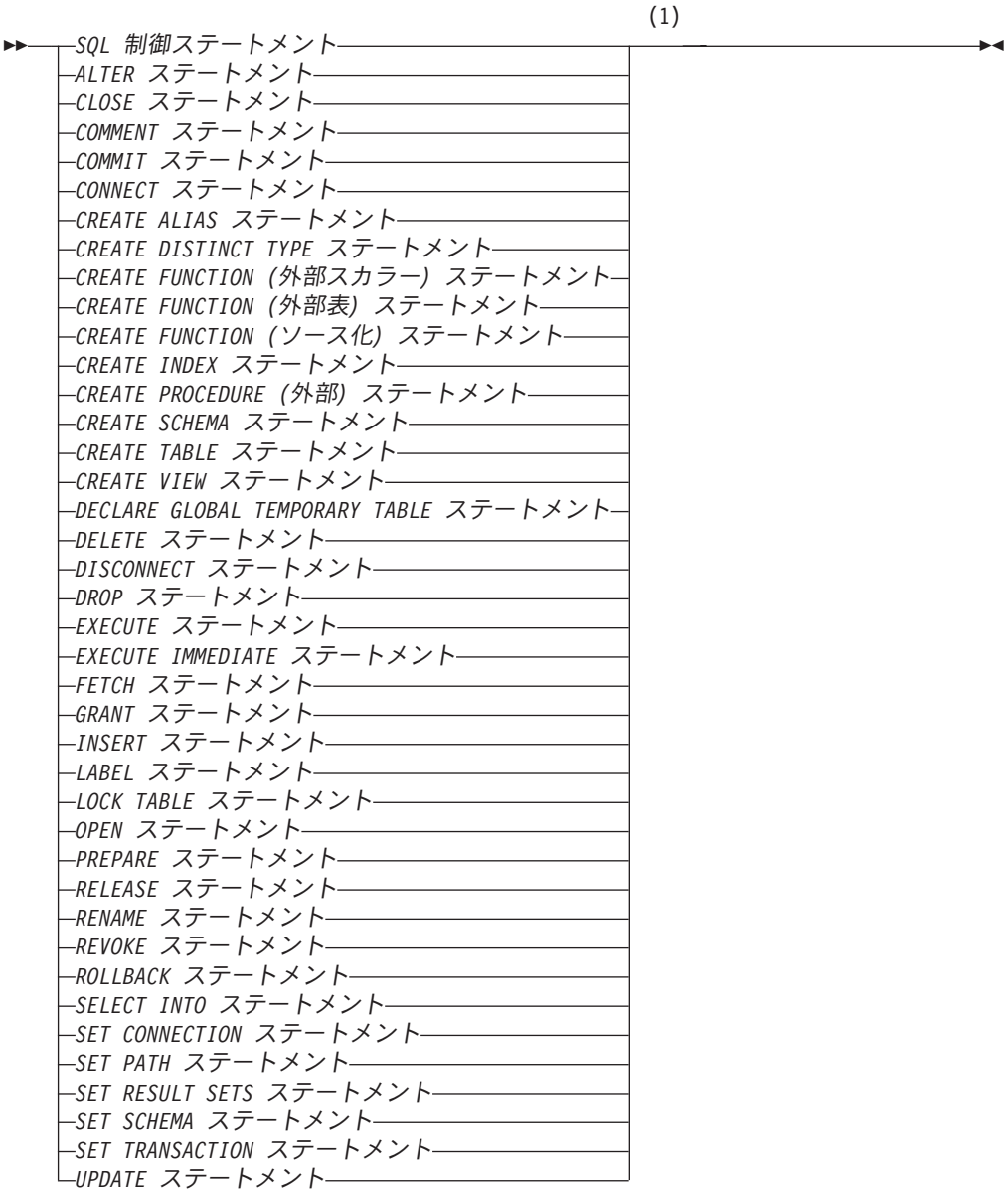
- SET PATH ステートメントおよび SET SCHEMA ステートメントでは、その名前は SQL パラメーター名または SQL 変数名としてチェックされます。 SQL 変数名または SQL パラメーター名として見つからなければ、その名前は ID として使用されます。
- CONNECT ステートメントでは、その名前は ID として使用されます。

SQL プロシージャー、関数、トリガー、SQL パラメーター、および SQL 変数に使用する名前の先頭に、'SQL' を付けてはなりません。

SQL プロシージャー・ステートメント

SQL 制御ステートメントでは、その SQL 制御ステートメントの内部に、複数の SQL ステートメントを指定することができます。これらのステートメントは、SQL プロシージャー・ステートメントとして定義されます。

構文



注:

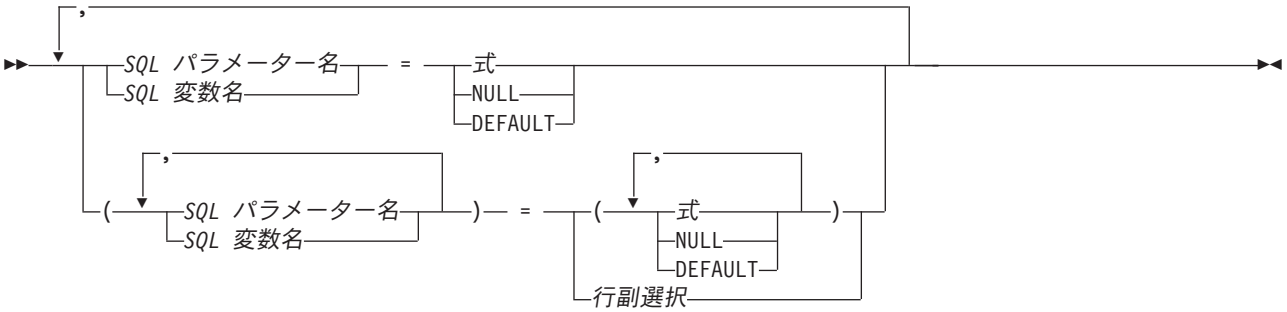
1 COMMIT、ROLLBACK、CONNECT、DISCONNECT、SET CONNECTION、および SET RESULT SETS ステートメントは、SQL プロシージャーでしか使用できません。 SET TRANSACTION ステートメントは、SQL プロシージャーとトリガーで使用できます。

割り当て (Assignment) ステートメント

割り当てステートメントは、SQL パラメーターまたは SQL 変数に値を割り当てます。

構文

割り当て文節 :



説明

ラベル
割り当てステートメントのラベルを指定します。ラベルは、その SQL 関数、SQL プロシージャー、または SQL トリガー内部で固有のものでなければならず、そのラベルが使用されている SQL 関数、SQL プロシージャー、または SQL トリガーの名前と同じであってはなりません。

SQL パラメーター名
割り当てのターゲットの SQL パラメーターを識別します。この SQL パラメーターは、CREATE PROCEDURE ステートメントまたは CREATE FUNCTION ステートメントのパラメーター*宣言* に指定しておく必要があります。

SQL 変数名
割り当てのターゲットの SQL 変数を識別します。SQL 変数は、複合ステートメントまたは遷移変数で定義することができます。

式 または **NULL**
割り当てのソースの式または値を指定します。

DEFAULT
遷移変数に関連付けられた列のデフォルト値を使用することを指定します。これは SQL トリガーの遷移変数に対してのみ指定できます。

行副選択
1 つの結果行を戻す副選択。結果列の値は、対応する SQL 変数またはパラメーターに割り当てられます。副選択の結果に行が含まれない場合、ヌル値が割り当てられます。結果の中に複数の行がある場合には、エラーが戻されます。

使用上の注意

割り当てステートメントは、SQL 割り当て規則に従っている必要があります。 割り当て規則については、 85 ページの『割り当ておよび比較』を参照してください。

ターゲットとソースのそれぞれのデータ・タイプには、互換性がなければなりません。

固定長変数へのストリングの割り当てで、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は、そのストリングの右側に必要な数の 1 バイト文字、2 バイト文字、または UCS-2 文字のブランクが埋め込まれます。

変数へのストリングの割り当てで、ストリングの長さが、その変数の長さ属性より長い場合には、負の SQLCODE が設定されます。

変数に割り当てられるストリングは、必要に応じて、最初にターゲットのコード化文字セットに変換されます。

数値変数への割り当ての際に数値の整数部分の切り捨てが行われると、負の SQLCODE が設定されます。

特殊レジスターの名前 (PATH など) に一致する ID を使用して変数を宣言した場合は、その変数を区切り文字で囲んで、特殊レジスターに対する割り当てと区別する必要があります (例えば、PATH という名前の変数を整数として宣言する場合は、″PATH″ = 1)。

割り当てのターゲットが変数で、ソースが変数または定数の場合、割り当てはインラインで行うことができます。この場合、SQLCODE と SQLSTATE はリセットされません。

SQL パラメーターの割り当て規則 : IN パラメーターは、割り当てステートメントの左側または右側に指定することができます。制御が呼び出し元に戻るときには、IN パラメーターのオリジナル値が保存されています。 OUT パラメーターは、割り当てステートメントの左側または右側に指定することができます。最初の指定時に値を割り当てておかなかった場合、値は未定義になります。制御が呼び出し元に戻るときに、OUT パラメーターに最後に割り当てられた値が呼び出し元に戻されます。 INOUT パラメーターの場合、このパラメーターの最初の値は呼び出し元により決定され、パラメーターに最後に割り当てられた値が呼び出し元に戻されます。

例

SQL 変数の p_salary を 10% 増やします。

```
SET p_salary = p_salary + (p_salary * .10)
```

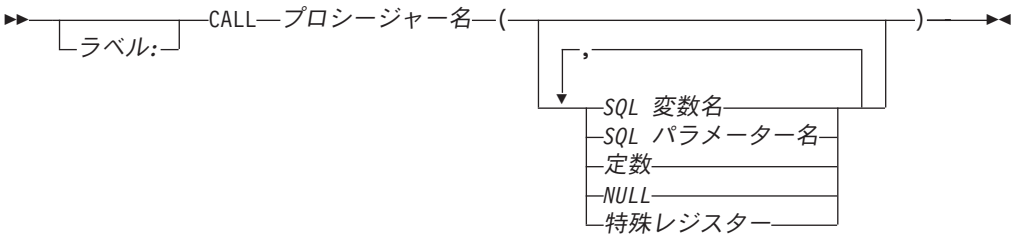
SQL 変数の p_salary をヌル値に設定します。

```
SET p_salary = NULL
```

呼び出し (call) ステートメント

CALL ステートメントは、プロシージャーを呼び出します。 404 ページの『CALL』を参照してください。

構文



説明

ラベル
CALL ステートメントのラベルを指定します。ラベルは、その SQL 関数、SQL プロシージャー、または SQL トリガー内部で固有のものでなければならず、そのラベルが使用されている SQL 関数、SQL プロシージャー、または SQL トリガーの名前と同じであってはなりません。

プロシージャー名
呼び出すプロシージャーを識別します。このプロシージャー名 は、現行サーバーに存在しているプロシージャーを識別していなければなりません。

SQL 変数名 または *SQL パラメーター名* または定数または *NULL* または特殊文字
パラメーターとしてプロシージャーに渡す値のリストを識別します。

各 OUT または INOUT パラメーターは、SQL パラメーターまたは SQL 変数として指定する必要があります。

使用上の注意

指定された引き数の数は、そのプロシージャーによって定義されたパラメーター数と同じでなければなりません。

プロシージャー内の特殊レジスターの初期値は、そのプロシージャーの呼び出し元から継承されます。プロシージャー内で特殊レジスターに割り当てられた値は、その SQL プロシージャー全体で使用され、そのプロシージャーから呼び出される後続のすべてのプロシージャーで継承されます。プロシージャーがその呼び出し元に戻るときは、特殊レジスターは呼び出し元のオリジナルの値に復元されます。

詳細については、 404 ページの『CALL』を参照してください。

例

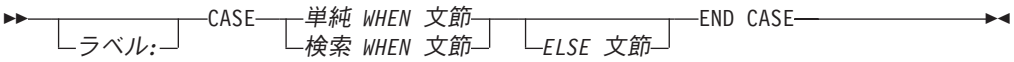
プロシージャー *proc1* を呼び出し、SQL 変数をパラメーターとして渡します。

```
CALL proc1(v_empno, v_salary)
```

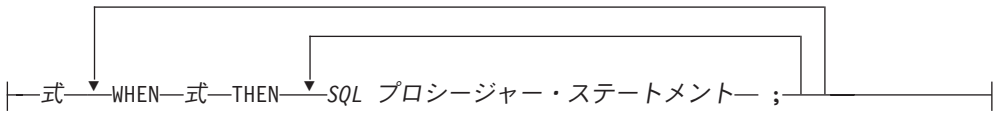
ケース (case) ステートメント

CASE ステートメントは、複数の条件に基づいて実行パスを選択します。

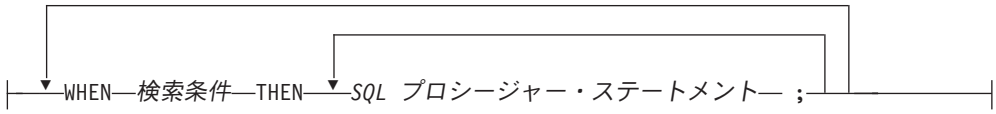
構文



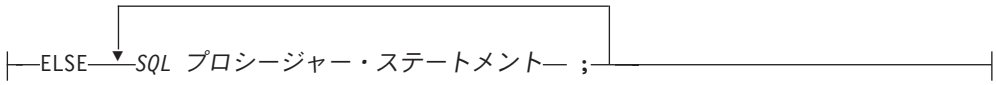
単純 WHEN 文節:



検索 WHEN 文節:



ELSE 文節:



説明

ラベル

CASE ステートメントのラベルを指定します。ラベルは、その SQL 関数、SQL プロシージャー、または SQL トリガー内部で固有のものでなければならず、そのラベルが使用されている SQL 関数、SQL プロシージャー、または SQL トリガーの名前と同じであってはなりません。

単純 WHEN 文節

最初の WHEN キーワードの前の式 の値が、WHEN キーワードの後のそれぞれの式 の値と等しいかどうかテストされます。その比較が真であれば、THEN ステートメントが実行されます。比較の結果が不明または偽であれば、処理は次の比較から続けられます。結果が比較のどれにも一致せず、ELSE 文節が指定されている場合には、その ELSE 文節の中のステートメントが処理されます。

検索 WHEN 文節

WHEN キーワードの後にある*検索条件* が評価されます。評価の結果が真であれば、関連した THEN 文節の中のステートメントが処理されます。評価の結果が偽、または不明であれば、次の*検索条件* が評価されます。評価結果が真になる*検索条件* がまったくなくて、ELSE 文節が指定されている場合は、その ELSE 文節の中のステートメントが処理されます。

ELSE 文節

単純 WHEN 文節 または*検索 WHEN 文節* に指定された条件がいずれも真でない場合は、*ELSE 文節* 内のステートメントが実行されます。

WHEN の中で指定された条件がいずれも真でない場合に ELSE が指定されていなければ、実行時にエラーが出され、CASE ステートメントの実行は終了します (SQLSTATE 20000)。

SQL プロシージャー・ステートメント
実行するステートメントを指定します。 822 ページの『SQL プロシージャー・ステートメント』 を参照してください。

使用上の注意

CASE ステートメントが可能なすべての実行条件を網羅していることを確認してください。

CASE ステートメントのネスト : *単純 WHEN 文節* を使用する CASE ステートメントは、最大 3 レベルまでネストすることができます。*検索 WHEN 文節* を使用する CASE ステートメントでは、ネスト・レベル数に制限はありません。

例

SQL 変数 v_workdept の値に応じて、表 DEPARTMENT 内の列 DEPTNAME を該当の名前で更新します。

次の例は、*単純 WHEN 文節* の構文を使用してこれを行う方法を示しています。

```
CASE v_workdept
  WHEN 'A00'
    THEN UPDATE department SET
                deptname = 'DATA ACCESS 1';
  WHEN 'B01'
    THEN UPDATE department SET
                deptname = 'DATA ACCESS 2';
    ELSE UPDATE department SET
                deptname = 'DATA ACCESS 3';
END CASE
```

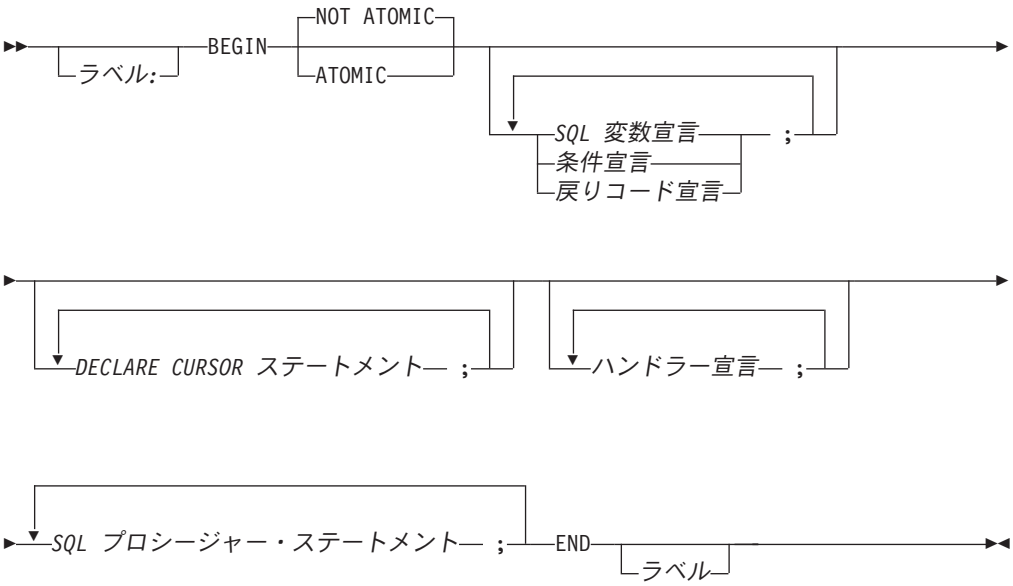
次の例は、*検索 WHEN 文節* の構文を使用してこれを行う方法を示しています。

```
CASE
  WHEN v_workdept = 'A00'
    THEN UPDATE department SET
                deptname = 'DATA ACCESS 1';
  WHEN v_workdept = 'B01'
    THEN UPDATE department SET
                deptname = 'DATA ACCESS 2';
    ELSE UPDATE department SET
                deptname = 'DATA ACCESS 3';
END CASE
```

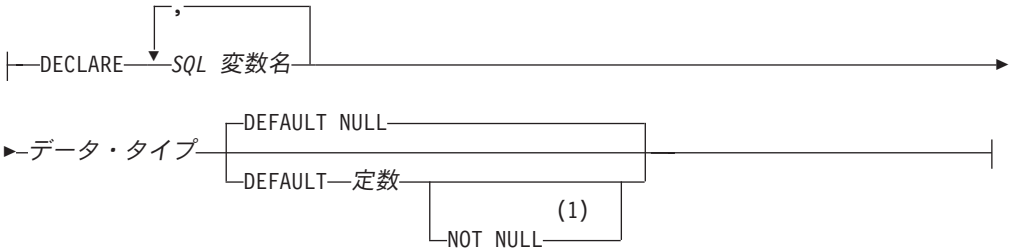
複合 (compound) ステートメント

複合ステートメントは、他のステートメントを 1 つの SQL ルーチンの中にグループとしてまとめます。複合ステートメント内では、SQL 変数、カーソル、およびハンドラーを宣言することができます。

構文

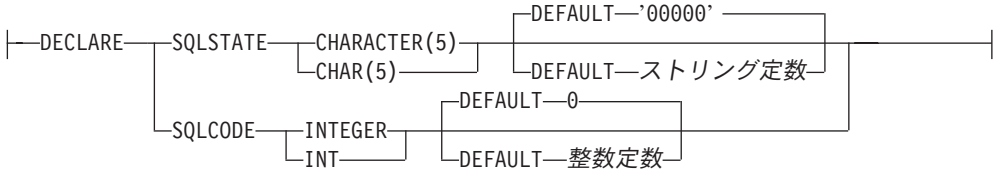


SQL 変数宣言:

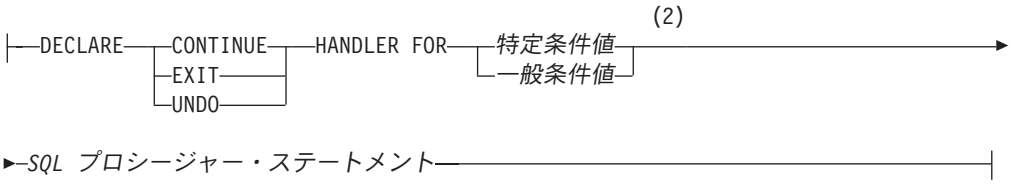


条件宣言:

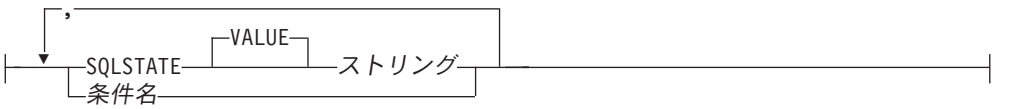
戻りコード宣言:



ハンドラー宣言:



特定条件値:



一般条件値:

データ・タイプ:

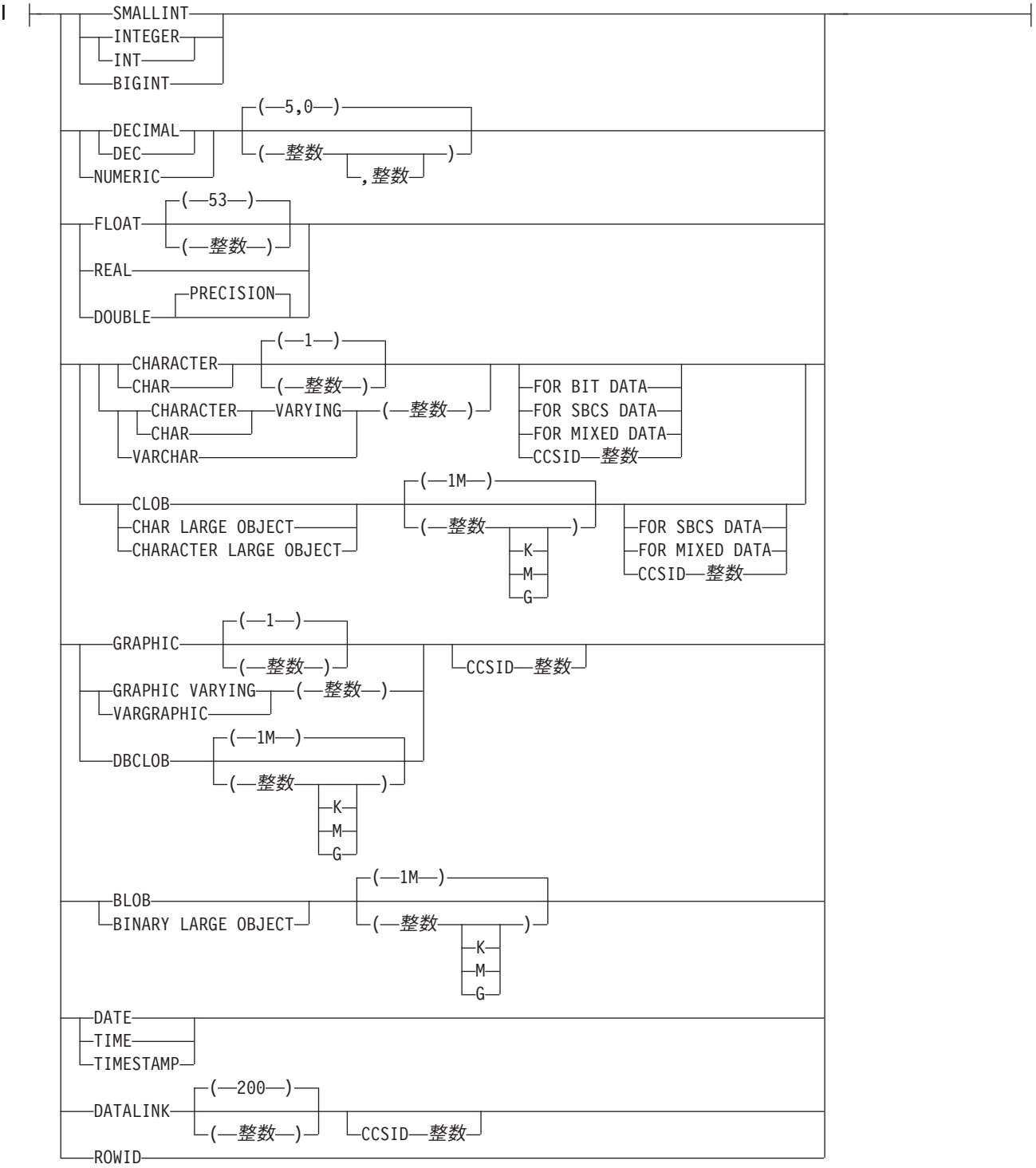
注:

1 DEFAULT 文節と NOT NULL 文節は、どちらの順序で指定しても構いません。

2 *特定条件値* と*一般条件値* を同一の ハンドラー宣言に同時に指定することはできません。

複合 (compound) ステートメント

組み込みタイプ:



説明

ラベル

コード・ブロックのラベルを定義します。ラベルは、その SQL 関数、SQL プロシージャー、または SQL トリガー内部で固有のものでなければならず、その

ラベルが使用されている SQL 関数、SQL プロシージャー、または SQL トリガーの名前と同じであってはなりません。 開始ラベルを指定すると、複合ステートメント内で宣言されている SQL 変数を修飾するためにそれを使用することができ、LEAVE ステートメント上でもそれを指定できます。

終了ラベルを指定する場合、開始ラベルと同じにしなければなりません。

ATOMIC または **NOT ATOMIC**

ATOMIC は、複合ステートメント内でエラーが起こった場合、その複合ステートメント内のすべてのステートメントをロールバックすることを示します。 ATOMIC を指定した場合は、複合ステートメント内で COMMIT または ROLLBACK ステートメントを指定することはできません (ROLLBACK TO SAVEPOINT を指定できます)。

NOT ATOMIC は、複合ステートメント内でエラーが起こっても、その複合ステートメントはロールバックされないことを示します。 NOT ATOMIC は、SQL トリガーの最外部の複合ステートメント内に指定されている場合は、ATOMIC として処理されます。

SQL 変数宣言

複合ステートメントに対してローカルな変数を宣言します。

SQL 変数名

ローカル変数の名前を定義します。データベース・マネージャーは、区切り文字のない SQL 変数名はすべて大文字に変換します。 *SQL 変数名* は、*複合ステートメント* 内部で固有の名前でなければなりません (ただし、*複合ステートメント* の内部にネストされている*複合ステートメント* の場合を除きます)。 SQL 変数名は、列名または SQL パラメーター名と同じであってはなりません。同一ステートメント内に同名の列が複数個ある場合に、SQL 変数名がどのように解決されるかについては、821 ページの『SQL パラメーターおよび変数の参照』を参照してください。変数名は 'SQL' で始まっていてはなりません。

SQL 変数名 は、それが宣言されている*複合ステートメント* の内部でのみ参照することができます (これは、その*複合ステートメント* 内部にネストされている個々の*複合ステートメント* に関しても同様です)。

データ・タイプ

変数のデータ・タイプを指定します。データ・タイプの説明については、542 ページの『CREATE TABLE』を参照してください。

データ・タイプ がグラフィック・ストリング・データ・タイプの場合は、UCS-2 データを指す CCSID 13488 を指定してください。 CCSID が指定されていない場合は、グラフィック・ストリング変数の CCSID は、そのジョブに関連付けられている DBCS CCSID です。

DEFAULT *定数* または **NULL**

SQL 変数のデフォルト値を定義します。この変数は、SQL プロシージャー、SQL 関数、または SQL トリガーが呼び出されるときに初期化されます。デフォルト値の指定がない場合は、SQL 変数は NULL に初期化されます。

NOT NULL
SQL 変数にヌル値が入るのを防ぎます。NOT NULL を指定しないことは、その列がヌルであってもよいことを意味します。

条件宣言
条件名と対応する SQLSTATE 値を宣言します。

条件名
条件の名前を指定します。条件名は、*複合ステートメント* 内部で固有の名前でなければなりません (ただし、*複合ステートメント* の内部にネストされている*複合ステートメント* の場合を除きます)。

条件名 は、それが宣言されている*複合ステートメント* の内部でのみ参照することができます (これは、その*複合ステートメント* 内部にネストされている個々の*複合ステートメント* に関しても同様です)。

FOR SQLSTATE *ストリング定数*
この条件に関連する SQLSTATE を指定します。 ストリング定数は、5 文字で指定しなければなりませんが、'00000' にすることはできません。

戻りコード宣言
SQLSTATE および SQLCODE という特殊変数を宣言します。これらの変数は、SQL ステートメントの実行後に戻される SQL 戻りコードに自動的に設定されます。 SQLSTATE 変数および SQLCODE 変数はどちらも、SQL プロシージャー、SQL 関数、または SQL トリガーの最外部の*複合ステートメント* の中でのみ宣言されます。

これらの変数に値を割り当てることは、禁止されてはいません。ただし、割り当てた値は次の SQL ステートメントによって置換されるので、割り当ててもあまり意味がありません。 SQLCODE 変数および SQLSTATE 変数は NULL に設定することはできません。

SQLCODE 変数および SQLSTATE 変数の値を使用する意図がある場合は、これらの変数の値をすぐに別の SQL 変数に保存する必要があります。SQLSTATE 用のハンドラーがある場合は、この割り当てをハンドラー内の最初のステートメントとして指定することによって、次の SQL プロシージャー・ステートメントで値が置換されるのを防ぐ必要があります。

カーソル宣言ステートメント
カーソルを宣言します。カーソル名は、*複合ステートメント* 内部で固有の名前でなければなりません (ただし、*複合ステートメント* の内部にネストされている*複合ステートメント* の場合を除きます)。

カーソル名 は、それが宣言されている*複合ステートメント* の内部でのみ参照することができます (これは、その*複合ステートメント* 内部にネストされている個々の*複合ステートメント* に関しても同様です)。

このカーソルをオープンするには OPEN ステートメントを使用し、このカーソルを使用して行を読み取るには FETCH ステートメントを使用します。*カーソル宣言ステートメント* が SQL プロシージャーの中にあり、CLOSE ステートメントが指定されておらず、そのプロシージャーの作成時に RESULT SET が指定されていなかった場合は、*複合ステートメント* の終わりでこのカーソルはクローズされます。詳しくは、597 ページの『DECLARE CURSOR』を参照してください。

ハンドラー宣言

ハンドラー、つまり複合ステートメント内で例外条件または完了条件が発生したときに実行される SQL プロシージャー・ステートメントを指定します。条件ハンドラーには以下の 3 つのタイプがあります。

CONTINUE
このハンドラーの呼び出しが正常に行われると、例外が起こったステートメントの後の SQL ステートメントに制御が戻されます。 IF、 CASE、 FOR、 WHILE、または REPEAT の中で比較を実行しているときにエラーが発生すると、それに対応する END IF、 END CASE、 END FOR、 END WHILE、または END REPEAT の後のステートメントに制御が戻されます。

EXIT
このハンドラーの呼び出しが正常に行われると、ハンドラーを宣言した複合ステートメントの終了点に制御が戻されます。

UNDO
*複合ステートメント*によって行われた変更を ROLLBACK し、ハンドラーを呼び出します。 このハンドラーの呼び出しが正常に行われると、*複合ステートメント*の終了点に制御が戻されます。 UNDO を指定する場合は、ATOMIC を指定する必要があります。

UNDO は、SQL 関数または SQL トリガーの最外部の*複合ステートメント*の中には指定できません。

このハンドラーが活動化される条件は以下のとおりです。

SQLSTATE *ストリング*
この特定の SQLSTATE 条件が発生したときにハンドラーを呼び出すことを指定します。 SQLSTATE は、'00000' と指定することはできません。

条件名
条件名で指定した条件が発生したときにハンドラーを呼び出すことを指定します。この条件名は、*条件宣言*の中ですでに定義されていなければなりません。

SQLEXCEPTION
SQLEXCEPTION が発生したときにハンドラーを呼び出すことを指定します。 SQLEXCEPTION は、クラス値が ″00″、″01″、および ″02″ 以外の SQLSTATE 値に相当します。

SQLWARNING
SQLWARNING が発生したときにハンドラーを呼び出すことを指定します。 SQLWARNING は、クラス値が ″01″ の SQLSTATE 値に相当します。

NOT FOUND
NOT FOUND 条件が発生したときにハンドラーを呼び出すことを指定します。 NOT FOUND は、クラス値 が ″02″ の SQLSTATE 値に相当します。

同一条件を*ハンドラー宣言* の中で複数回指定することはできません。

使用上の注意

ハンドラー宣言 に関する規則

- 同じ複合ステートメント内の複数のハンドラー宣言に、同じ条件を重複して含めることはできません。
- ハンドラー宣言には、同じ条件値または SQLSTATE 値を複数回入れることはできず、また、1 つの SQLSTATE 値と、それと同じ SQLSTATE 値を表す条件名とを含めることもできません。詳しい説明と SQLSTATE 値のリストに関しては、SQL プログラミング 概念を参照してください。
- ハンドラーは、例外条件または完了条件に最も適したハンドラーである場合に活動化されます。 最も適したハンドラーとは、該当の例外条件または完了条件の SQLSTATE に最も一致度の高い*複合ステートメント* に定義されている (例外条件または完了条件用の) ハンドラーです。例えば、SQLSTATE 22001 用と SQLEXCEPTION 用の両方のハンドラーが存在していれば、SQLSTATE 22001 に対する通知が出た場合には、SQLSTATE 22001 用のハンドラーが最も適したハンドラーということになります。例外が発生し、その例外のためのハンドラーがない場合は、*複合ステートメント* の実行は終了します。警告または検出不能条件が発生し、そのためのハンドラーがない場合は、次のステートメントに移って処理が続けられます。

例

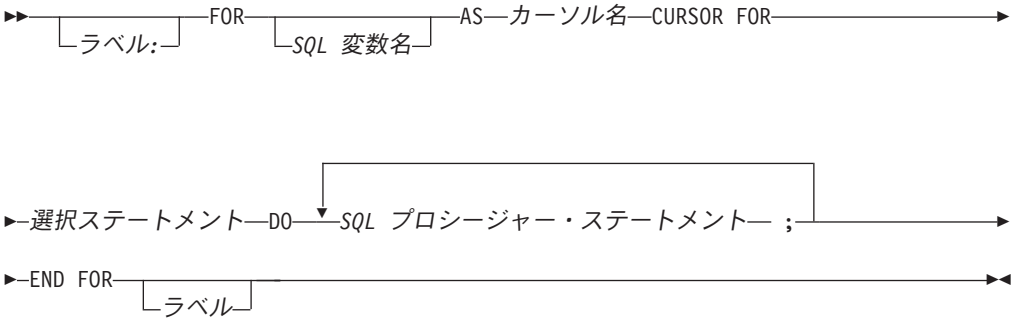
以下の例では SQL プロシージャー PROC1 が作成されます。このプロシージャーのルーチン本体は複合ステートメントです。この複合ステートメントは、SQL 変数、SQLSTATE '02000' に関する条件、継続ハンドラー、およびカーソル宣言ステートメントを宣言します。WHILE ステートメントは、FETCH ステートメントでループします。'02000' (ファイルの終わり) 条件が戻されると、ハンドラーが呼び出され、SQL 変数 at_end は 1 に設定されます。at_end は 0 でなくなるので、制御がハンドラーから戻され、WHILE ループは終了します。

```
CREATE PROCEDURE PROC1 () LANGUAGE SQL
BEGIN
DECLARE v_firstnme VARCHAR(12);
DECLARE v_midinit CHAR(1);
DECLARE v_lastname VARCHAR(15);
DECLARE v_edlevel SMALLINT;
DECLARE v_salary DECIMAL(9,2);
DECLARE at_end INT DEFAULT 0;
DECLARE not_found
       CONDITION FOR SQLSTATE '02000';
DECLARE c1 CURSOR FOR
  SELECT firstnme, midinit, lastname,
                    edlevel, salary
  FROM employee;
DECLARE CONTINUE HANDLER FOR not_found
            SET at_end = 1;
OPEN c1;
FETCH c1 INTO v_firstnme, v_midinit,
            v_lastname, v_edlevel, v_salary;
WHILE at_end = 0 DO
   FETCH c1 INTO
              v_firstname, v_midinit,
              v_lastname, v_edlevel, v_salary;
END WHILE;
CLOSE c1;
END
```

FOR ステートメント

FOR ステートメントは、表のそれぞれの行ごとにステートメントを実行します。

構文



説明

ラベル

コード・ブロックのラベルを定義します。ラベルは、その SQL 関数、SQL プロシージャー、または SQL トリガー内部で固有のものでなければならず、そのラベルが使用されている SQL 関数、SQL プロシージャー、または SQL トリガーの名前と同じであってはなりません。

終了ラベルを指定する場合、開始ラベルと同じにしなければなりません。

SQL 変数名

SQL 変数名 を使用すると、ステートメント内の変数を修飾することができます。 *SQL 変数名* は、その SQL 関数、SQL プロシージャー、または SQL トリガー内のどのラベルとも異なっていなければならず、その *SQL 変数名* が使用されている SQL 関数、SQL プロシージャー、または SQL トリガーの名前と同じであってはなりません。 *SQL 変数名* またはラベル のどちらか一方を使用して、ステートメント内の他の SQL 変数を修飾することができます。

SQL 変数名 を指定した場合は、SQL 関数、SQL プロシージャー、または SQL トリガーをデバッグするときに、そのステートメントの他のすべての SQL 変数名の修飾にも、この SQL 変数名を使用する必要があります。

カーソル名

カーソルの名前を指定します。これを指定しない場合、固有のカーソル名が生成されます。

選択ステートメント

カーソルの選択ステートメントを指定します。

選択リスト内のそれぞれの式には名前が必要です。式の名前が単純な列名ではない場合、その式の名前を指定するには AS 文節を使用しなければなりません。AS 文節を指定すると、その名前は変数に使用されますが、必ず固有の名前でなければなりません。

SQL プロシージャー・ステートメント
表のそれぞれの行ごとに実行される SQL ステートメント。この SQL ステートメントには、FOR ステートメント上のラベルを指定する LEAVE ステートメントを含めることはできず、FOR ステートメントのカーソル名を指定する OPEN、FETCH、または CLOSE が含まれていてはなりません。

使用上の注意

FOR ステートメントは、表内の行ごとに、それぞれ 1 つまたは複数のステートメントを実行します。 カーソルは、選択された列および行を記述する選択リストを指定することによって定義されます。FOR ステートメント内のステートメントは、選択されたそれぞれの行ごとに実行されます。

選択リストの内容は固有の列名でなければならず、選択リストに指定されている表が、関数、プロシージャー、またはトリガーの作成時には存在していなければなりません。

FOR ステートメントに指定されているカーソルは、その FOR ステートメント以外の場所で参照することはできず、OPEN、FETCH、または CLOSE ステートメントに指定できません。

例

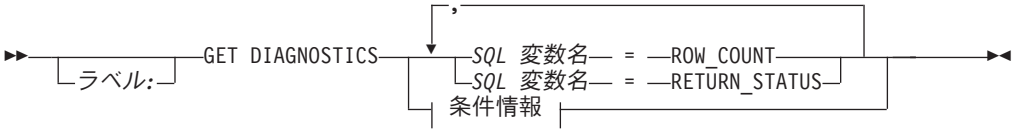
この例では、FOR ステートメントが、employee 表から 3 列を選択するカーソルを指定するために使用されています。 選択されたすべての行について、SQL 変数 *fullname* の設定が、ラストネーム、コンマ、ファーストネーム、ブランク、ミドルネームのイニシャル、の順で行われます。*fullname* のそれぞれの値が、表 TNAMES に挿入されます。

```
BEGIN
  DECLARE fullname CHAR(40);
  FOR vl AS
      c1 CURSOR FOR
    SELECT firstnme, midinit, lastname FROM employee
     DO
       SET fullname =
          lastname || ', ' || firstnme ||' ' || midinit;
       INSERT INTO TNAMES VALUE ( fullname );
     END FOR;
END;
```

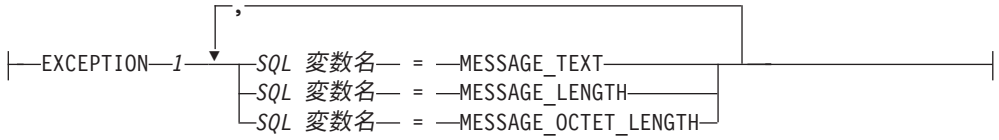
診断入手 (get diagnostics) ステートメント

GET DIAGNOSTICS ステートメントは、直前に実行された SQL ステートメントに関する情報を取得します。

構文



条件情報:



説明

ラベル

GET DIAGNOSTIC ステートメントのラベルを指定します。ラベルは、その SQL 関数、SQL プロシージャー、または SQL トリガー内部で固有のものでなければならず、そのラベルが使用されている SQL 関数、SQL プロシージャー、または SQL トリガーの名前と同じであってはなりません。

SQL 変数名

割り当てのターゲットの SQL 変数または SQL パラメーターを識別します。MESSAGE_TEXT が指定されている場合、この変数は、データ・タイプが CHAR または VARCHAR でなければなりません。それ以外の場合、この SQL 変数は整変数でなければなりません。

ROW_COUNT

直前に実行された SQL ステートメントに関連付けられた行数を識別します。直前の SQL ステートメントが DELETE、INSERT、または UPDATE ステートメントの場合、ROW_COUNT は、そのステートメントによって削除、挿入、または更新された行数を識別します (ただし、トリガーまたは参照保全制約の影響を受けている行は除きます)。直前のステートメントが PREPARE ステートメントの場合、ROW_COUNT は、準備済みステートメントの結果行の見積行数を識別します。

RETURN_STATUS

直前の SQL CALL ステートメントから戻された状況値を識別します。直前のステートメントが CALL ステートメントでない場合は、戻される値は意味がなく、予測不能です。詳細については、854 ページの 『戻り (return) ステートメント』 を参照してください。

条件情報

直前の SQL ステートメントに関してエラー情報を戻すか、警告情報を戻すかを指定します。

エラーに関する情報が必要な場合、GET DIAGNOSTICS ステートメントは、エラーを処理するハンドラーに指定された最初のステートメントでなければなりません。

警告に関する情報が必要な場合は、次のようにします。

- ハンドラーがその警告条件に対する制御を取得する場合、GET DIAGNOSTICS ステートメントは、そのハンドラーに指定された最初のステートメントでなければなりません。
- ハンドラーがその警告条件に対する制御を取得しない場合、GET DIAGNOSTICS ステートメントは、その直前のステートメントの次に実行されるステートメントでなければなりません。

MESSAGE_TEXT
直前に実行された SQL ステートメントから戻されたエラーまたは警告のメッセージ・テキストを識別します。直前の SQL ステートメントが SQLCODE ゼロで完了した場合、空ストリングまたはブランクが戻されます。メッセージ・テキストが *SQL 変数名* の長さ属性より長い場合は、警告は戻されません。

MESSAGE_LENGTH または MESSAGE_OCTET_LENGTH
直前に実行された SQL ステートメントから戻されたエラーまたは警告のメッセージ・テキストの長さを識別します。直前の SQL ステートメントが SQLCODE ゼロで完了した場合、長さ 0 が戻されます。

使用上の注意

GET DIAGNOSTICS ステートメントは、診断エリア (SQLCA) の内容を変更することはありません。 SQL プロシージャー、SQL 関数、または SQL トリガーの中で SQLSTATE 特殊変数または SQLCODE 特殊変数が宣言されている場合、これらの特殊変数は、GET DIAGNOSTICS ステートメントの発行後に戻された SQLSTATE または SQLCODE に設定されます。

例

SQL プロシージャーにおいて、GET DIAGNOSTICS ステートメントを実行して、更新された行数を判別します。

```
CREATE PROCEDURE sqlprocg (IN deptnbr VARCHAR(3)) LANGUAGE SQL
  BEGIN
    DECLARE SQLSTATE CHAR(5);
    DECLARE rcount INTEGER;
    UPDATE CORPDATA.PROJECT
      SET PRSTAFF = PRSTAFF + 1.5
      WHERE DEPTNO = deptnbr;
    GET DIAGNOSTICS rcount = ROW_COUNT;
    /* At this point, rcount contains the number of rows that were updated. */
  END;
```

SQL プロシージャー内部で、TRYIT というストアード・プロシージャーの呼び出しから戻された状況値を処理します。 TRYIT で RETURN ステートメントを使用して状況値を明示的に戻すこともでき、データベース・マネージャーから状況値が暗黙的に戻されることもあります。このプロシージャーは、正常に実行されると、値ゼロを戻します。

```
CREATE PROCEDURE TESTIT ()
  LANGUAGE SQL
  A1: BEGIN
      DECLARE RETVAL INTEGER DEFAULT 0;
      ...
      CALL TRYIT
      GET DIAGNOSTICS RETVAL = RETURN_STATUS;
      IF RETVAL <> 0 THEN
         ...
         LEAVE A1;
      ELSE
         ...
      END IF;
  END A1
```

SQL プロシージャーで、GET DIAGNOSTICS ステートメントを実行して、エラーのメッセージ・テキストを取り出します。

```
CREATE PROCEDURE divide2 ( IN numerator INTEGER,
                           IN denominator INTEGER,
                          OUT divide_result INTEGER,
                              OUT divide_error
VARCHAR(70) )
  LANGUAGE SQL
    BEGIN
       DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
          GET DIAGNOSTICS EXCEPTION 1
            divide_error = MESSAGE_TEXT;
       SET divide_result = numerator / denominator;
    END;
```

GOTO ステートメント

GOTO ステートメントは、SQL ルーチンまたは SQL トリガー内部のユーザー定義のラベルに分岐します。

構文

説明

ラベル 1

コード・ブロックのラベルを定義します。ラベルは、その SQL 関数、SQL プロシージャー、または SQL トリガー内部で固有のものでなければならず、そのラベルが使用されている SQL 関数、SQL プロシージャー、または SQL トリガーの名前と同じであってはなりません。

ラベル 2

処理を継続するラベル付きステートメントを指定します。ラベル付きステートメントと GOTO ステートメントは、両方とも同じ効力範囲内に存在しなければなりません。

- GOTO ステートメントが FOR ステートメントで定義されている場合、*ラベル* は、同じ FOR ステートメント内で定義されていなければなりません (ただし、ネストされた FOR ステートメントは除きます)。
- GOTO ステートメントが FOR ステートメントの外部で定義されている場合、*ラベル* は、FOR ステートメントの内部で定義されていてはなりません。
- GOTO ステートメントがハンドラーで定義されている場合、*ラベル* は、同じハンドラー内で定義されていなければなりません。
- GOTO ステートメントがハンドラーの外部で定義されている場合、*ラベル* は、ハンドラーの内部で定義されていてはなりません。

ラベル 2 が GOTO ステートメントで到達可能な有効範囲内に定義されていない場合は、エラーが戻されます。

使用上の注意

GOTO ステートメントの使用は控えめにする必要があります。GOTO ステートメントは通常の処理順序を妨げるので、ルーチンが読みにくくなり、保守もしにくくなるからです。 IF や LEAVE などの別のステートメントを使用すれば、GOTO ステートメントを使わなくて済むことがあります。

例

次のステートメントでは、パラメーター *rating* と *v_empno* がプロシージャーに渡されます。サービスの時間が、出力パラメーター *return_parm* で日付期間として戻されます。会社のサービスの時間が 6 か月未満の場合、GOTO ステートメントは制御をプロシージャーの最後に移動し、*new_salary* は変更されないままになります。

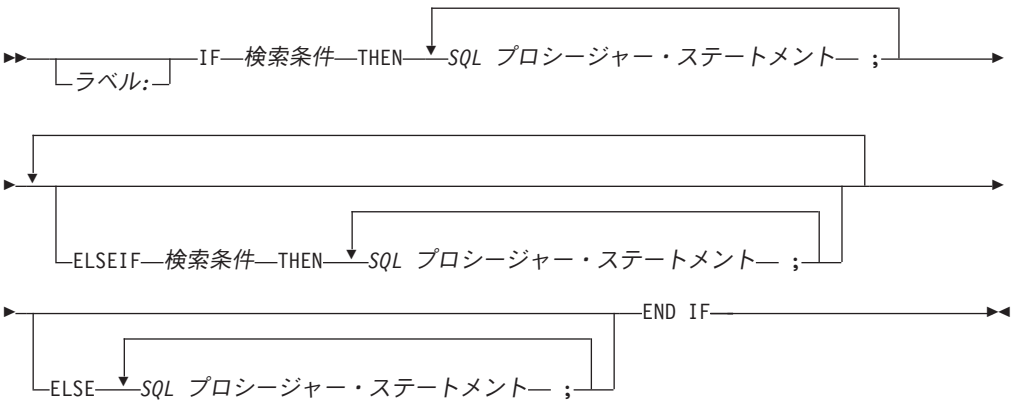
```
CREATE PROCEDURE adjust_salary (IN v_empno CHAR(6),
                                IN rating INTEGER,
                                OUT return_parm DECIMAL(8,2))
  LANGUAGE SQL
  MODIFIES SQL DATA
    BEGIN
       DECLARE new_salary DECIMAL(9,2);
       DECLARE service DECIMAL(8,2);
            SELECT salary, current_date - hiredate
               INTO new_salary, service
               FROM employee
               WHERE empno = v_empno;
            IF service < 600
              THEN GOTO exit1;
            END IF;
            IF rating = 1
             THEN SET new_salary =
                          new_salary + (new_salary * .10);
             ELSEIF rating = 2
             THEN SET new_salary =
                          new_salary + (new_salary * .05);
            END IF;
            UPDATE employee
               SET salary = new_salary
               WHERE empno = v_empno;

      exit1: SET return_parm = service;
    END
```

IF ステートメント

IF ステートメントは、検索条件の結果に基づいて、異なるセットの SQL ステートメントを実行します。

構文



説明

ラベル
IF ステートメントのラベルを指定します。ラベルは、その SQL 関数、SQL プロシージャー、または SQL トリガー内部で固有のものでなければならず、そのラベルが使用されている SQL 関数、 SQL プロシージャー、または SQL トリガーの名前と同じであってはなりません。

検索条件
SQL ステートメントを実行することが必要になる*検索条件* を指定します。この条件が不明または偽であれば、処理は次の検索条件または ELSE 文節から続けられます。

SQL プロシージャー・ステートメント
前の*検索条件* が真であった場合に実行しなければならない SQL ステートメントを指定します。

例

次の SQL プロシージャーは、2 つの IN パラメーター (従業員番号と従業員考課) を受け入れます。 *rating* (考課) の値に応じて、従業員表が更新され、給与および賞与の列に新しい値が入ります。

```
CREATE PROCEDURE UPDATE_SALARY_IF
   (IN employee_number CHAR(6), INOUT rating SMALLINT)
   LANGUAGE SQL
   MODIFIES SQL DATA
   BEGIN
      DECLARE not_found CONDITION FOR SQLSTATE '02000';
      DECLARE EXIT HANDLER FOR not_found
         SET rating = -1;
      IF rating = 1
         THEN UPDATE employee
```

```
      SET salary = salary * 1.10, bonus = 1000
      WHERE empno = employee_number;
   ELSEIF rating = 2
      THEN UPDATE employee
      SET salary = salary * 1.05, bonus = 500
      WHERE empno = employee_number;
   ELSE UPDATE employee
      SET salary = salary * 1.03, bonus = 0
      WHERE empno = employee_number;
   END IF;
END
```

ITERATE ステートメント

ITERATE ステートメントは、制御のフローをラベル付きループの先頭に戻します。

構文

説明

ラベル 1

コード・ブロックのラベルを定義します。ラベルは、その SQL 関数、SQL プロシージャー、または SQL トリガー内部で固有のものでなければならず、そのラベルが使用されている SQL 関数、SQL プロシージャー、または SQL トリガーの名前と同じであってはなりません。

ラベル 2

データベース・マネージャーが制御のフローを渡す先の FOR、 LOOP、REPEAT、または WHILE ステートメントのラベルを指定します。

例

この例では、カーソルを使用して、新規部門に関する情報を戻します。 *not_found* 条件ハンドラーが呼び出された場合は、制御のフローはループの外部に移ります。 *v_dept* の値が 'D11' の場合は、ITERATE ステートメントは制御のフローを LOOP ステートメントの先頭に戻します。それ以外の場合は、新規の列が DEPARTMENT 表に挿入されます。

```
CREATE PROCEDURE ITERATOR ()
   LANGUAGE SQL
   MODIFIES SQL DATA
   BEGIN
      DECLARE v_dept CHAR(3);
      DECLARE v_deptname VARCHAR(29);
      DECLARE v_admdept CHAR(3);
      DECLARE at_end INTEGER DEFAULT 0;
      DECLARE not_found CONDITION FOR SQLSTATE '02000';
      DECLARE c1 CURSOR FOR
        SELECT deptno,deptname,admrdept
        FROM department
        ORDER BY deptno;
      DECLARE CONTINUE HANDLER FOR not_found
        SET at_end = 1;
      OPEN c1;
      ins_loop:
      LOOP
         FETCH c1 INTO v_dept, v_deptname, v_admdept;
         IF at_end = 1 THEN
            LEAVE ins_loop;
         ELSEIF v_dept ='D11' THEN
            ITERATE ins_loop;
         END IF;
         INSERT INTO department (deptno,deptname,admrdept)
            VALUES('NEW', v_deptname, v_admdept);
      END LOOP;
      CLOSE c1;
   END
```

終了 (leave) ステートメント

LEAVE ステートメントは、ブロックまたはループを終了することによって実行を継続します。

構文

説明

ラベル 1
コード・ブロックのラベルを定義します。ラベルは、その SQL 関数、SQL プロシージャー、または SQL トリガー内部で固有のものでなければならず、そのラベルが使用されている SQL 関数、SQL プロシージャー、または SQL トリガーの名前と同じであってはなりません。

ラベル 2
終了する複合、FOR、LOOP、REPEAT、または WHILE ステートメントのラベルを指定します。

使用上の注意

LEAVE ステートメントが複合ステートメントの外部に制御を転送するときは、結果セットを戻すために使用されるカーソルを除き、その複合ステートメント内のすべてのオープン・カーソルがクローズされます。

例

この例には、カーソル *c1* のデータを取り出すループが含まれています。 SQL 変数 *at_end* の値がゼロでない場合、LEAVE はループの外部に制御を転送します。

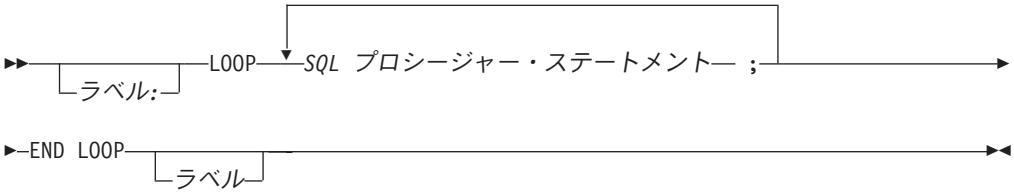
```
CREATE PROCEDURE LEAVE_LOOP (OUT COUNTER INTEGER)
   LANGUAGE SQL
   BEGIN
      DECLARE v_counter INTEGER;
      DECLARE v_firstnme VARCHAR(12);
      DECLARE v_midinit CHAR(1);
      DECLARE v_lastname VARCHAR(15);
      DECLARE at_end SMALLINT DEFAULT 0;
      DECLARE not_found CONDITION FOR SQLSTATE '02000';
      DECLARE c1 CURSOR FOR
         SELECT firstnme, midinit, lastname
         FROM employee;
      DECLARE CONTINUE HANDLER FOR not_found
         SET at_end = 1;
      SET v_counter = 0;
      OPEN c1;
      fetch_loop:
      LOOP
         FETCH c1 INTO v_firstnme, v_midinit, v_lastname;
         IF at_end <> 0 THEN
            LEAVE fetch_loop;
         END IF;
         SET v_counter = v_counter + 1;
```

```
    END LOOP fetch_loop;
    SET counter = v_counter;
    CLOSE c1;
END
```

ループ (loop) ステートメント

LOOP ステートメントは、1 つのステートメントまたは 1 グループのステートメントの実行を繰り返します。

構文



説明

ラベル
LOOP ステートメントの名前を指定します。ラベルは、その SQL 関数、SQL プロシージャー、または SQL トリガー内部で固有のものでなければならず、そのラベルが使用されている SQL 関数、SQL プロシージャー、または SQL トリガーの名前と同じであってはなりません。 開始ラベルを指定すると、そのラベルを LEAVE ステートメントに指定することができます。

終了ラベルを指定する場合、開始ラベルと同じにしなければなりません。

SQL プロシージャー・ステートメント
ループで実行する SQL ステートメントを指定します。

例

このプロシージャーでは、LOOP ステートメントを使用して従業員表から値を取り出します。ループが繰り返されるたびに、OUT パラメーターの *counter* が増分し、*v_midinit* の値がチェックされて、その値がシングル・スペース (' ') でないかどうかが確認されます。 *v_midinit* がシングル・スペースの場合は、LEAVE ステートメントは制御のフローをループの外部に渡します。

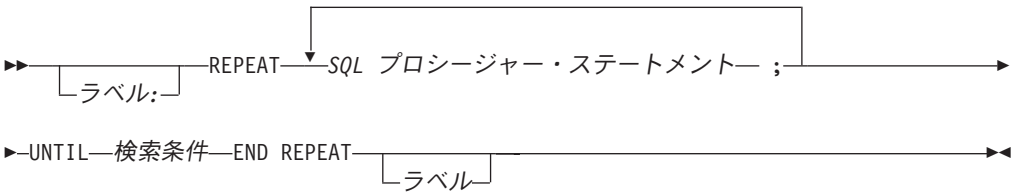
```
CREATE PROCEDURE LOOP_UNTIL_SPACE (OUT COUNTER INTEGER)
   LANGUAGE SQL
   BEGIN
      DECLARE v_counter INTEGER DEFAULT 0;
      DECLARE v_firstnme VARCHAR(12);
      DECLARE v_midinit CHAR(1);
      DECLARE v_lastname VARCHAR(15);
      DECLARE c1 CURSOR FOR
         SELECT firstnme, midinit, lastname
         FROM employee;
      DECLARE CONTINUE HANDLER FOR NOT FOUND
         SET counter = -1;
      OPEN c1;
      fetch_loop:
      LOOP
         FETCH c1 INTO v_firstnme, v_midinit, v_lastname;
         IF v_midinit = ' ' THEN
            LEAVE fetch_loop;
```

```
      END IF;
      SET v_counter = v_counter + 1;
   END LOOP fetch_loop;
   SET counter = v_counter;
   CLOSE c1;
END
```

反復 (repeat) ステートメント

REPEAT ステートメントは、検索条件が真になるまで、1 つのステートメントまたは 1 グループのステートメントの実行を繰り返します。

構文



説明

ラベル

REPEAT ステートメントの名前を指定します。ラベルは、その SQL 関数、SQL プロシージャー、または SQL トリガー内部で固有のものでなければならず、そのラベルが使用されている SQL 関数、SQL プロシージャー、または SQL トリガーの名前と同じであってはなりません。 開始ラベルを指定すると、そのラベルを LEAVE ステートメントに指定することができます。

終了ラベルを指定する場合、開始ラベルと同じにしなければなりません。

SQL プロシージャー・ステートメント

REPEAT ループで実行する SQL ステートメントを指定します。

検索条件

REPEAT ループが実行されるつど、その後に*検索条件* が評価されます。この条件が真であれば、REPEAT ループは終了します。この条件が不明または偽であれば、ループは継続します。

例

not_found 条件ハンドラーが呼び出されるまで、REPEAT ステートメントで表から列を取り出します。

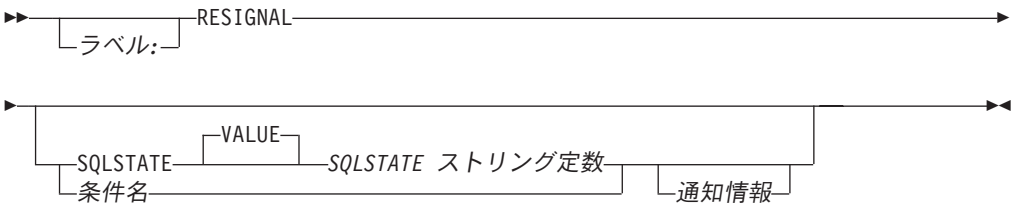
```
CREATE PROCEDURE REPEAT_STMT (OUT COUNTER INTEGER)
   LANGUAGE SQL
   BEGIN
      DECLARE v_counter INTEGER DEFAULT 0;
      DECLARE v_firstnme VARCHAR(12);
      DECLARE v_midinit CHAR(1);
      DECLARE v_lastname VARCHAR(15);
      DECLARE at_end SMALLINT DEFAULT 0;
      DECLARE not_found CONDITION FOR SQLSTATE '02000';
      DECLARE c1 CURSOR FOR
         SELECT firstnme, midinit, lastname
         FROM employee;
      DECLARE CONTINUE HANDLER FOR not_found
         SET at_end = 1;
      OPEN c1;
      fetch_loop:
```

```
   REPEAT
      FETCH c1 INTO v_firstnme, v_midinit, v_lastname;
      SET v_counter = v_counter + 1;
      UNTIL at_end > 0
   END REPEAT fetch_loop;
   SET counter = v_counter;
   CLOSE c1;
END
```

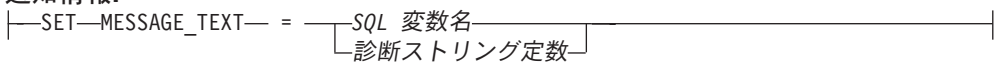
再通知 (resignal) ステートメント

RESIGNAL ステートメントは、エラー条件または警告条件を戻すために、ハンドラー内部で使用されます。

構文



通知情報:



説明

ラベル

RESIGNAL ステートメントのラベルを指定します。ラベルは、その SQL 関数、SQL プロシージャー、または SQL トリガー内部で固有のものでなければならず、そのラベルが使用されている SQL 関数、SQL プロシージャー、または SQL トリガーの名前と同じであってはなりません。

SQLSTATE VALUE *SQLSTATE ストリング定数*

戻される SQLSTATE エラー・コードを指定します。このストリングは、厳密に 5 文字の文字ストリング定数で、かつ次の 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' ～ '9') またはアクセント記号なしの英大文字 ('A' ～ 'Z') でなければなりません。
- SQLSTATE クラス (最初の 2 文字) は、'00' であってはなりません (これは正常終了を表します)。

SQLSTATE がこの規則に従っていないと、エラーが戻されます。

条件名

戻される条件の名前を指定します。*条件名*は、複合ステートメントの中で宣言する必要があります。

MESSAGE_TEXT

エラーまたは警告を説明するストリングを指定します。このストリングは、SQLCA の SQLERRMC フィールドに戻されます。ストリングの実際の長さが 70 バイトを超える場合、警告せずに切り捨てられます。

SQL 変数名

複合ステートメントの中で宣言する必要がある SQL 変数を識別します。SQL 変数は、CHAR または VARCHAR データ・タイプとして定義しなければなりません。

診断ストリング定数
メッセージ・テキストを入れる文字ストリング定数を指定します。

使用上の注意

RESIGNAL ステートメントには、任意の有効な SQLSTATE 値を使用できますが、プログラマーは、アプリケーション用に予約されている範囲に基づいて、新規の SQLSTATE を定義することをお勧めします。そうすれば、使用する SQLSTATE 値が今後リリースされるデータベース・マネージャーで定義される値と重なってしまう可能性を防止できます。 SQLSTATE の詳細については、iSeries Information Center の SQL メッセージおよびコードを参照してください。

RESIGNAL ステートメントを SQLSTATE 文節や*条件名*なしで指定する場合、RESIGNAL ステートメントはハンドラー内になければなりません。 SQL ルーチンは、ハンドラーを呼び出したのと同一の条件で、呼び出し元に戻ります。

RESIGNAL ステートメントが発行され、SQLSTATE または*条件名* が指定されている場合は、SQLCA で戻される SQLCODE は、次のように SQLSTATE 値に基づいて設定されます。

- 指定された SQLSTATE クラスが '01' または '02' の場合、警告または NOT FOUND が通知され、SQLCODE は +438 に設定されます。
- それ以外の場合、例外が戻され、SQLCODE は -438 に設定されます。

RESIGNAL ステートメントが実行され、SQLSTATE 値も*条件名* も指定されていない場合、SQLCODE は変更されません。

SQLSTATE または条件が、例外が通知されたこと (SQLSTATE クラスが '01' または '02' 以外) を示している場合は、次のようになります。

- その RESIGNAL ステートメントと同じ複合ステートメント内にハンドラーが存在し、しかも*複合ステートメント* の中に SQLEXCEPTION、あるいは指定の SQLSTATE または条件に対するハンドラーが含まれている場合は、例外は処理され、制御はそのハンドラーに渡されます。
- *複合ステートメント* がネストされていて、外側の*複合ステートメント* の中に SQLEXCEPTION あるいは指定の SQLSTATE または条件に対するハンドラーが含まれている場合は、例外は処理され、制御はそのハンドラーに渡されます。
- それ以外の場合は、例外は処理されず、制御は即時に複合ステートメントの終了点に戻されます。

SQLSTATE または条件が、警告 (SQLSTATE クラス '01') または NOT FOUND (SQLSTATE クラス '02') が通知されたことを示している場合は、次のようになります。

- その RESIGNAL ステートメントと同じ複合ステートメント内にハンドラーが存在し、しかも*複合ステートメント* の中に SQLWARNING (SQLSTATE クラスが '01' の場合)、NOT FOUND (SQLSTATE クラスが '02' の場合)、または指定の SQLSTATE または条件に対するハンドラーが含まれている場合は、警告または NOT FOUND 条件は処理され、制御はそのハンドラーに渡されます。
- *複合ステートメント* がネストされていて、外側の複合ステートメントの中に SQLWARNING (SQLSTATE クラスが '01' の場合)、NOT FOUND (SQLSTATE

クラスが '02' の場合)、あるいは指定の SQLSTATE または条件に対するハンドラーが含まれている場合は、警告または NOT FOUND 条件は処理され、制御はそのハンドラーに戻されます。

- それ以外の場合、警告は処理されず、次のステートメントから処理を継続します。

SQLSTATE 値は、2 文字のクラス・コード値と、それに続く 3 文字のサブクラス・コード値から構成されます。クラス・コード値は、成否の実行条件のクラスを表します。

例

この例は、ゼロによる割り算 (ゼロ除算) エラーを検出します。IF ステートメントは、SIGNAL ステートメントを使用して、オーバーフロー条件ハンドラーを呼び出します。オーバーフロー条件ハンドラーは、RESIGNAL ステートメントを使用して、異なる SQLSTATE 値をクライアント・アプリケーションに戻します。

```
CREATE PROCEDURE divide ( IN numerator INTEGER,
                          IN denominator INTEGER,
                          OUT divide_result INTEGER )
  LANGUAGE SQL
    BEGIN
       DECLARE overflow CONDITION FOR '22003';
       DECLARE CONTINUE HANDLER FOR overflow
           RESIGNAL SQLSTATE '22375';
       IF denominator = 0 THEN
           SIGNAL overflow;
       ELSE
           SET divide_result = numerator / denominator;
       END IF;
     END;
```

戻り (return) ステートメント

RETURN ステートメントは、ルーチンから戻るための処理を実行します。 SQL 関数の場合は、関数の結果を戻します。SQL プロシージャーの場合は、オプションで整数の状況値を戻します。 SQL 表関数の場合は、関数の結果を反映した表を戻します。

構文

QUERY 式:

説明

ラベル

RETURN ステートメントのラベルを指定します。ラベルは、その SQL 関数、SQL プロシージャー、または SQL トリガー内部で固有のものでなければならず、そのラベルが使用されている SQL 関数、SQL プロシージャー、または SQL トリガーの名前と同じであってはなりません。

式 ルーチンから戻される値を指定します。

- ルーチンが関数の場合、式 の指定は必須です。この値は、CREATE FUNCTION ステートメントの RETURNS 文節に指定されているデータ・タイプと互換性がなければなりません。
- ルーチンがプロシージャーの場合、式 のデータ・タイプは INTEGER でなければなりません。式がヌル値の場合、値 0 が戻されます。

NULL

ヌル値は、SQL 関数から戻されます。SQL プロシージャーでは、NULL は許されません。

QUERY 式

SQL 表関数から戻される表を識別します。*全選択* は、SQL スカラー関数および SQL プロシージャーでは使用できません。

共通表式

全選択の場合に使用する共通表式を指定します。

全選択

表関数で戻される行を指定します。全選択の列の数は、関数結果の列の数と一致していなければなりません。全選択では、1 つ以上の列が含まれた 1 つ以上の行が戻されることもあり、1 行も戻されないこともあります。

使用上の注意

プロシージャーから値が戻された場合、呼び出し元は次の方法で値にアクセスできます。

- SQL プロシージャーが別の SQL プロシージャーまたは SQL 関数から呼び出された場合、GET DIAGNOSTICS ステートメントを使用して RETURN_STATUS を取り出します。
- ODBC アプリケーションのエスケープ文節 CALL 構文 (?=CALL...) で、戻り値パラメーター・マーカーを宛先とするパラメーターを使用します。
- SQLCODE がゼロ未満ではない場合、SQL プロシージャーの CALL 処理によって戻された SQLCA から直接 sqlerrd[0] の値を取り出します。 SQLCODE がゼロ未満の場合は、sqlerrd[0] は設定されず、アプリケーションは RETURN_STATUS 値が -1 であるものと想定します。

プロシージャーから戻るのに RETURN ステートメントが使用されなかった場合、または RETURN ステートメントで値が指定されていない場合は、次のようになります。

- プロシージャーがゼロ以上の SQLCODE で戻る場合、RETURN_STATUS は 0 に設定されます。
- プロシージャーがゼロ未満の SQLCODE で戻る場合、RETURN_STATUS は -1 に設定されます。

プロシージャーから戻るのに戻り値が指定された RETURN ステートメントが使用された場合、SQLCA 内の SQLCODE、SQLSTATE、およびメッセージ・テキストはゼロに初期化されます。エラーは呼び出し元に戻されません。

RETURN は、SQL トリガーでは使用できません。

SQL 表関数ステートメントのルーチン本体に指定できる RETURN ステートメントは、1 つだけです。

例

RETURN ステートメントを使用して SQL プロシージャーから戻り、成功したときは状況値 0 、失敗したときは状況値 -200 を戻します。

```
BEGIN
  ...
  GOTO fail;
  ...
  success: RETURN 0
  failure: RETURN -200
...
END
```

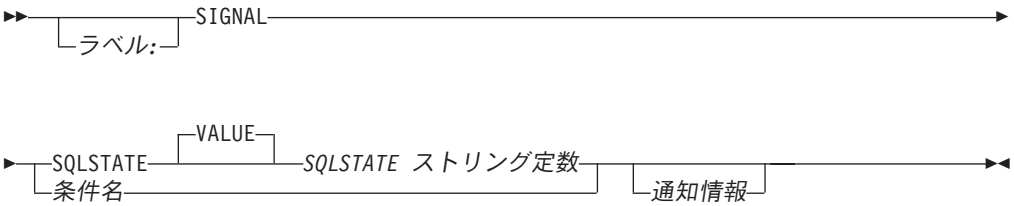
既存のサイン (正弦) 関数とコサイン (余弦) 関数を使用して、値のタンジェント (正接) を戻すスカラー関数を定義します。

```
CREATE FUNCTION mytan (x DOUBLE)
  RETURNS DOUBLE
  LANGUAGE SQL
  CONTAINS SQL
  NO EXTERNAL ACTION
  DETERMINISTIC
  RETURN SIN(x)/COS(x)
```

通知 (signal) ステートメント

SIGNAL ステートメントは、エラー条件または警告条件を通知します。これは、指定の SQLSTATE とオプションのメッセージ・テキストを使用して、エラーまたは警告を戻します。

構文



説明

ラベル

SIGNAL ステートメントのラベルを指定します。ラベルは、その SQL 関数、SQL プロシージャー、または SQL トリガー内部で固有のものでなければならず、そのラベルが使用されている SQL 関数、SQL プロシージャー、または SQL トリガーの名前と同じであってはなりません。

SQLSTATE VALUE *SQLSTATE ストリング定数*

通知する SQLSTATE を指定します。 このストリングは、厳密に 5 文字の文字ストリング定数で、かつ次の 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' ～ '9') またはアクセント記号なしの英大文字 ('A' ～ 'Z') でなければなりません。
- SQLSTATE クラス (最初の 2 文字) は、'00' であってはなりません (これは正常終了を表します)。

SQLSTATE がこの規則に従っていないと、エラーが戻されます。

条件名

通知される条件の名前を指定します。*条件名*は、複合ステートメントの中で宣言する必要があります。

MESSAGE_TEXT

エラーまたは警告を説明するストリングを指定します。このストリングは、SQLCA の SQLERRMC フィールドで戻されます。ストリングの実際の長さが 70 バイトを超える場合、警告せずに切り捨てられます。

SQL 変数名
複合ステートメントの中で宣言する必要がある SQL 変数を識別します。SQL 変数は、CHAR または VARCHAR データ・タイプとして定義しなければなりません。

診断ストリング定数
メッセージ・テキストを入れる文字ストリング定数を指定します。

(診断ストリング定数)
メッセージ・テキストを入れるストリング定数を指定します。この形式は、SQL トリガーの本体でのみサポートされます。 ANS および ISO 規格に準拠するためには、この形式は使用してはなりません。これは、他のプロダクトとの互換性のために提供されています。

使用上の注意

SIGNAL ステートメントには、任意の有効な SQLSTATE 値を使用できますが、プログラマーは、アプリケーション用に予約されている範囲に基づいて、新規の SQLSTATE を定義することをお勧めします。そうすれば、使用する SQLSTATE 値が今後リリースされるデータベース・マネージャーで定義される値と重なってしまう可能性を防止できます。 SQLSTATE の詳細については、iSeries Information Center の SQL メッセージおよびコードを参照してください。

SIGNAL ステートメントが実行された場合、SQLCA で戻される SQLCODE は、次のように SQLSTATE 値に基づいて設定されます。

- 指定された SQLSTATE クラスが '01' または '02' の場合、警告または NOT FOUND が通知され、SQLCODE は +438 に設定されます。
- それ以外の場合、例外が通知され、SQLCODE は -438 に設定されます。

SQLSTATE または条件が、例外 (SQLSTATE クラスが '01' または '02' 以外) が通知されたことを示している場合は、次のようになります。

- その SIGNAL ステートメントと同じ複合ステートメント内にハンドラーが存在し、しかもその複合ステートメントの中に SQLEXCEPTION あるいは指定の SQLSTATE または条件に対するハンドラーが含まれている場合、例外は処理され、制御はそのハンドラーに渡されます。
- それ以外の場合は、例外は処理されず、制御は即時に複合ステートメントの終了点に戻されます。

SQLSTATE または条件が、警告 (SQLSTATE クラス '01') または NOT FOUND (SQLSTATE クラス '02') が通知されたことを示している場合は、次のようになります。

- その SIGNAL ステートメントと同じ複合ステートメント内にハンドラーが存在し、しかもその複合ステートメントの中に SQLWARNING (SQLSTATE クラスが '01'の場合)、NOT FOUND (SQLSTATE クラスが '02' の場合)、または指定の SQLSTATE または条件に対するハンドラーが含まれている場合、警告または NOT FOUND 条件は処理され、制御はそのハンドラーに渡されます。
- それ以外の場合、警告は処理されず、次のステートメントから処理を継続します。

SQLSTATE 値は、2 文字のクラス・コード値と、それに続く 3 文字のサブクラス・コード値から構成されます。クラス・コード値は、成否の実行条件のクラスを表します。

SIGNAL ステートメントには、任意の有効な SQLSTATE 値を使用できますが、プログラマーは、アプリケーション用に予約されている範囲に基づいて、新規の SQLSTATE を定義することをお勧めします。そうすれば、使用する SQLSTATE 値が今後リリースされるデータベース・マネージャーで定義される値と重なってしまう可能性を防止できます。

- 文字 '7' ～ '9' または 'I' ～ 'Z' で始まる SQLSTATE クラスは、定義しても構いません。これらのクラス内では、任意のサブクラスを定義できます。
- 文字 '0' ～ '6' または 'A' ～ 'H' で始まる SQLSTATE クラスは、データベース・マネージャー用に予約されています。これらのクラス内では、文字 '0' ～ 'H' で始まるサブクラスは、データベース・マネージャー用に予約されています。文字 'I' ～ 'Z' で始まるサブクラスは、定義しても構いません。

SQLSTATE の詳細については、iSeries Information Center の SQL メッセージおよびコードを参照してください。

例

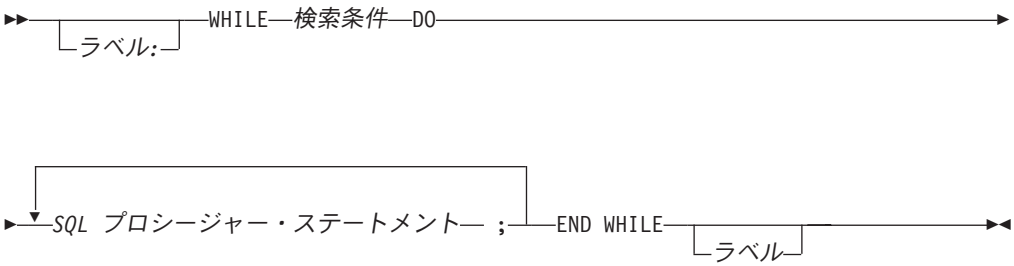
次の複合ステートメントでは、パラメーター*rating* がプロシージャーに渡されます。会社のサービスの時間が 6 か月未満の場合、例外 II001 が即時に呼び出し元に戻されます。

```
CREATE PROCEDURE raise ( IN rating INTEGER )
  LANGUAGE SQL
    BEGIN
       DECLARE new_salary DECIMAL(9,2);
       DECLARE service DECIMAL(8,0);
       DECLARE v_empno CHAR(6) DEFAULT '123456';
            SELECT salary, current_date - hiredate
               INTO new_salary, service
               FROM employee
               WHERE empno = v_empno;
            IF service < 600
              THEN SIGNAL SQLSTATE 'II001'
                      SET MESSAGE_TEXT = 'Insufficient time in service.';
            END IF;
            IF rating = 1
             THEN SET new_salary =
                         new_salary + (new_salary * .10);
             ELSEIF rating = 2
             THEN SET new_salary =
                         new_salary + (new_salary * .05);
            END IF;
            UPDATE employee
               SET salary = new_salary
               WHERE empno = v_empno;
     END;
```

WHILE ステートメント

WHILE ステートメントは、指定された条件が真である間、ステートメントの実行を繰り返します。

構文



説明

ラベル

コード・ブロックのラベルを定義します。ラベルは、その SQL 関数、SQL プロシージャー、または SQL トリガー内部で固有のものでなければならず、そのラベルが使用されている SQL 関数、SQL プロシージャー、または SQL トリガーの名前と同じであってはなりません。 開始ラベルを指定すると、LEAVE ステートメント上でそれを指定することができます。

終了ラベルを指定する場合、開始ラベルと同じにしなければなりません。

検索条件

WHILE ループの実行の前に、必ず*検索条件* が評価されます。この条件が真であれば、WHILE ループ内の *SQL プロシージャー・ステートメント* が実行されます。

SQL プロシージャー・ステートメント

WHILE ループで実行する SQL ステートメントを指定します。

例

この例では、WHILE ステートメントを使用して FETCH ステートメントおよび SET ステートメントを繰り返し実行します。 SQL 変数 *v_counter* の値が IN パラメーター *deptNumber* で識別される部門の従業員数の半分未満である間は、WHILE ステートメントは、FETCH および SET ステートメントを実行し続けます。条件が真にならなくなった時点で、制御のフローは WHILE ステートメントから離れ、カーソルがクローズされます。

```
CREATE PROCEDURE dept_median
   (IN deptNumber SMALLINT, OUT medianSalary DECIMAL(7,2))
    LANGUAGE SQL
    BEGIN
       DECLARE v_numRecords INTEGER DEFAULT 1;
       DECLARE v_counter INTEGER DEFAULT 0;
       DECLARE c1 CURSOR FOR
          SELECT salary
          FROM staff
          WHERE dept = deptNumber
          ORDER BY salary;
```

```
   DECLARE EXIT HANDLER FOR NOT FOUND
      SET medianSalary = 6666;
   SET medianSalary = 0;
   SELECT COUNT(*) INTO v_numRecords
      FROM staff
      WHERE dept = deptNumber;
   OPEN c1;
   WHILE v_counter < (v_numRecords/2 + 1) DO
      FETCH c1 INTO medianSalary;
      SET v_counter = v_counter +1;
   END WHILE;
   CLOSE c1;
END
```

付録 A. SQL の制約

以下の表には、DB2 UDB for iSeries のデータベース・マネージャーによる制約を記載してあります。

表 58. IDの長さに関する制約

制限されるID	DB2 UDB for iSeries の制限
別名の最大長	128
権限名の最大長	10
列ラベルの最大長	60
相関名の最大長	128
カーソル名の最大長	18
ホスト ID の最大長	64
保管ポイント名の最大長	128
サーバー名の最大長	18
SQL ルーチン・ラベルの最大長	128
ステートメント名の最大長	18
表、パッケージ、または別名ラベルの最大長	50
非修飾のスキーマ名の最大長	10
非修飾の列名の最大長	30
非修飾の制約名の最大長	128
非修飾のデータ・タイプ名の最大長	128
非修飾の外部プログラム名の最大長[68]	10
非修飾の関数名の最大長	128
非修飾の非グループ名の最大長	10
非修飾のパッケージ名の最大長	10
非修飾のプロシージャー名の最大長	128
非修飾の特定名の最大長	128
非修飾の SQL パラメーター名の最大長	128
非修飾の SQL 変数名の最大長	128
非修飾形式の表名、ビュー名、および索引名の最大長	128
非修飾のトリガー名の最大長	128
非修飾のシステム列名	10
非修飾のシステム表、ビュー、および索引名	10

68. サービス・プログラムの入り口点名の場合、限界は 279 になります。 REXX プロシージャーの場合は、限界は 33 になります。

表 59. 数値の制約

数値の制約	DB2 UDB for iSeries の制限
BIGINT (長整数) の最小値	-9 223 372 036 854 775 808
BIGINT (長整数) の最大値	+9 223 372 036 854 775 807
INTEGER (整数) の最小値	-2 147 483 648
INTEGER (整数) の最大値	+2 147 483 647
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
FLOAT (浮動小数点数) の最小値[70]	-1.79×10^{308}
FLOAT (浮動小数点数) の最大値[70]	$+1.79 \times 10^{308}$
FLOAT (浮動小数点数) の正の最小値[70]	$+2.23 \times 10^{-308}$
FLOAT (浮動小数点数) の負の最大値[70]	-2.23×10^{-308}
REAL (実数) の最小値[70]	-3.4×10^{38}
REAL (実数) の最大値[70]	$+3.4 \times 10^{38}$
REAL (実数) の正の最小値[70]	$+1.18 \times 10^{-38}$
REAL (実数) の負の最大値[70]	-1.18×10^{-38}

表 60. ストリングの制約

ストリングの制約	DB2 UDB for iSeries の制限
BLOB の最大長	2 147 483 647
CHAR (文字) の最大長[71]	32765
VARCHAR (可変長文字) の最大長[71]	32739
CLOB の最大長	2 147 483 647
C の NUL 終了ストリングの最大長[71]	32739
GRAPHIC (グラフィック) の最大長[71]	16382
VARGRAPHIC (可変グラフィック) の最大長[71]	16369
DBCLOB の最大長	1 073 741 823
C の NUL 終了グラフィックの最大長[71]	16369
文字定数の最大長	32740
グラフィック定数の最大長	16370
連結文字ストリングの最大長[71]	32765
連結グラフィック・ストリングの最大長[71]	16369

表 61. 日付/時刻の制約

日付/時刻の制約	DB2 UDB for iSeries の制限
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 62. データ・リンクの制約

データ・リンクの制約	DB2 UDB for iSeries の制限
DATALINK の最大長	32718
DATALINK コメントの最大長	254

表 63. データベース・マネージャーの制約

データベース・マネージャーの制約	DB2 UDB for iSeries の制限
1 つの表の列の最大数	8000
1 つのビューの列の最大数	8000
関数内のパラメーターの最大数	90
プロシージャー内のパラメーターの最大数	254[73]
LOB のない行の最大長 (すべてのオーバーヘッドを含む)	32766
LOB がある行の最大長 (すべてのオーバーヘッドを含む)	3 758 096 383
表の最大サイズ	1 テラバイト
索引の最大サイズ	1 テラバイト
1 つの表の行の最大数	4 294 967 288
索引キーの最大長	2000
索引キーの列の最大数	120
1 つの表の索引の最大数	約 4000
1 つの SQL ステートメントから参照する表の最大数	256
1 つの SQL ビューから参照する表の最大数	32
プリコンパイルのプログラムのホスト変数宣言の最大数[69]	記憶域のサイズによる
1 つの SQL ステートメント内のホスト変数および定数の最大数	4096[74]
挿入または更新に使用するホスト変数の最大長	32766
SQL ステートメントの最大長	65535
1 つの選択リストのエレメントの最大数[72]	約 8000
1 つの WHERE または HAVING 文節の述部の最大数	4690
1 つの GROUP BY 文節の列の最大数	120
1 つの GROUP BY 文節の列の最大合計長	2000
1 つの ORDER BY 文節の列の最大数	10000
1 つの ORDER BY 文節の列の最大合計長	10000
SQLDA の最大サイズ	16 777 215
準備済みステートメントの最大数	記憶域のサイズによる
1 つのプログラムで宣言できるカーソルの最大数	記憶域のサイズによる
一度にオープンできるカーソルの最大数	記憶域のサイズによる
リレーショナル・データベース内の最大表数	記憶域のサイズによる
1 つの表の制約の最大数	300
副選択で許される最高レベル	32
コメントの最大長	2000
パスの最大長	3483

表 63. データベース・マネージャーの制約 (続き)

データベース・マネージャーの制約	DB2 UDB for iSeries の制限
パス内のスキーマの最大数	268
1 つの作業単位で変更される行の最大数	500 000 000
表上のトリガーの最大数	300
ネストされたトリガー呼び出しの最大数	200
取り出しを待つ結果セットを伴うプロシージャーの最大数	100
パスワードの最大長	128
トランザクション内のロケーターの最大数	250 000
一度にアクティブにできる保管ポイントの最大数	記憶域のサイズによる
1 つのプロセス内で同時に割り振りできる CLI ハンドルの最大数	80 000

69. RPG/400 および PL/I のプログラムで、パラメーターを渡す手法として従来の手法が使用されている場合は、上限は約 4000 になります。この制限は、プログラムで使用できるポインターの数によるものです。その他の場合はすべて、上限はご使用のオペレーティング・システム内部の制約に基づいています。

70. この値は、概算値です。

71. 列が NOT NULL の場合は、最大値は 1 大きくなります。

72. 上限は、解析された SQL ステートメントについて生成される内部構造のサイズに基づいています。

73. PARAMETER STYLE SQL を含むプロシージャーのパラメーター数は 90 に制限されています。 PARAMETER STYLE GENERAL を含む SQL プロシージャーは、253 に制限されています。 PARAMETER STYLE GENERAL WITH NULLS を含むプロシージャーは、254 に制限されています。 PARAMETER STYLE GENERAL を含む外部プロシージャーは、255 に制限されています。パラメーターの数の最大数は、その外部プログラムのコンパイルに使用されるライセンス・プログラムで許されるパラメーターの最大の数によっても制約されます。

74. ステートメントが読み取り専用ではない場合は、上限は 2048 です。上限は概算値なので、非常に大きいストリング定数またはストリング変数を使用する場合は、示されている値より小さくなることもあります。

75. DRDA 接続 1 つ当たりに割り振られるハンドルの最大数は 500 です。

付録 B. SQL 連絡域

SQLCA は一組の変数で、各 SQL ステートメントの実行の終了時に更新されます。実行可能な SQL ステートメントが入っているプログラムは、正確に 1 つの SQLCA を用意する必要があります (ただし、代わりに独立型の SQLCODE または独立型の SQLSTATE 変数を使用する場合を除く)。

RPG および REXX を除くすべてのホスト言語では、SQL INCLUDE ステートメントを使用して SQLCA の宣言を用意することができます。REXX プロシージャーにおける SQLCA の使用法については、DB2 UDB for iSeries ホスト言語での SQL プログラミングを参照してください。

C、COBOL、FORTRAN、および PL/I では、この記憶域の名前は、SQLCA でなければなりません。PL/I および C では、該当する構造体の名前を SQLCA としなければなりません。すべての SQL ステートメントは、必ず SQLCA の宣言の有効範囲内になければなりません。

プログラムで独立型の SQLCODE を指定している場合は、SQLCA を組み込んではなりません。この場合、プリコンパイラーは、変数 SQLCODE の名前を SQLCADE に変更した (または、変数 SQLCOD の名前を SQLCAD に変更した) SQLCA を組み込みます。プリコンパイラーは、独立型の SQLCODE に正しい値が入るように、プログラムにステートメントを追加します。

プログラムで独立型の SQLSTATE を指定している場合には、SQLCA を組み込んではなりません。プリコンパイラーは、変数 SQLSTATE の名前を SQLSTATE に変更して SQLCA を組み込みます。プリコンパイラーは、独立型の SQLSTATE に正しい値が入るように、プログラムにステートメントを追加します。

RPG および REXX では、独立型の SQLCODE および独立型の SQLSTATE を指定してはなりません。

フィールドの説明

以下の表に示している名前は、SQL の INCLUDE ステートメントによって指定されている名前です。大部分については、C (および C++)、COBOL、FORTRAN および PL/I では同じ名前を使用します。RPG/400 では、名前が 6 文字までに制限されているため、RPG では異なる名前を使用します。PL/I の名前と COBOL の名前が異なっている 1 つの事例に注意してください。

表 64. SQL の INCLUDE ステートメントによって組み込まれる名前

C の名前、COBOL および PL/I の名前	FORTRAN[1] の名前	RPG の名前	フィールド・データ・タイプ	フィールドの値
SQLCAID sqlcaid	使用不可 SQLCAID	SQLAID	CHAR(8)	記憶ダンプのための「目印」として、'SQLCA' が入ります。
SQLCABC sqlcabc	使用不可 SQLCABC	SQLABC	INTEGER	SQLCA の長さ 136 が入ります。

表 64. SQL の INCLUDE ステートメントによって組み込まれる名前 (続き)

C の名前、COBOL および PL/I の名前	FORTRAN[1] の名前	RPG の名前	フィールド・データ・タイプ	フィールドの値
SQLCODE sqlcode	SQLCOD SQLCODE	SQLCOD	INTEGER	SQL 戻りコードが入ります。 **コード　意味** **0**　正常に実行された。ただし、SQLWARN 標識がセットされている場合がある。 **正の値**　正常に実行された (ただし、警告条件があった)。 **負の値**　エラー状態。
SQLERRML[2] sqlerrml	SQLTXL SQLERRML	SQLERL	SMALLINT	SQLERRMC の長さ標識 (0 から 70 までの範囲)。0 は SQLERRMC の値が無関係であることを示します。
SQLERRMC[2] sqlerrmc	SQLTXT SQLERRMC	SQLERM	CHAR (70)	SQLCODE に関連するメッセージ置換テキストが入ります。CONNECT および SET CONNECTION の場合、この SQLERRMC フィールドにはその接続に関する情報が入ります。置換テキストについての説明は、 871 ページの表 67 を参照してください。
SQLERRP sqlerrp	SQLERP SQLERRP	SQLERP	CHAR(8)	エラーを戻したプロダクトおよびモジュールの名前が入ります。最初の 3 文字は、次のようにプロダクトを識別します。 ARI (DB2 (VM および VSE 版) の場合) DSN (DB2 UDB (OS/390 および z/OS 版) の場合) QSQ (DB2 UDB for iSeries の場合) 他のすべての DB2 プロダクトの場合は SQL 詳細については、425 ページの『CONNECT (タイプ 1)』または 431 ページの 『CONNECT (タイプ 2)』を参照してください。
SQLERRD sqlerrd	SQLERR SQLERRD	SQLERR[3]	配列	診断情報が提供される 6 つの INTEGER 変数が入ります。診断情報の説明については、 868 ページの表 66 を参照してください。
SQLWARN sqlwarn	SQLWRN SQLWARN	SQLWRN[4]	CHAR(11)	11 個の CHAR(1) の警告標識です。それぞれにブランク、'W'、または 'N' のいずれかが入ります。
SQLSTATE sqlstate	SQLSTT SQLSTATE	SQLSTT	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です。

表 64. SQL の INCLUDE ステートメントによって組み込まれる名前 (続き)

C の名前、COBOL および PL/I の名前	FORTRAN[1] の名前	RPG の名前	フィールド・データ・タイプ	フィールドの値
注:				
[1] 最初の名前は、FORTRAN SQLCA についての IBM SQL SQLCA 名を示しています。2 番目の名前は、FORTRAN での SQLCA の DB2 UDB for iSeries を設定するために使用可能な代替名を示しています。				
[2] COBOL では、SQLERRM には SQLERRML と SQLERRMC が含まれています。PL/I では、可変長ストリング SQLERRM は、SQLERRMC にプレフィックス SQLERRML が付いたものと同じです。				
[3] RPG/400 および ILE RPG/400 では、SQLERR は 24 文字 (配列ではなく) として定義されます。これらの文字は、SQLER1 から SQLER6 までのフィールドによって再定義されます。各フィールドは、フルワード 2 進数です。ILE RPG/400 の場合には、SQLERR は配列としても再定義されています。この名前の配列は SQLERRD です。				
[4] 11 文字 (配列ではなく) として定義されます。				

表 65. SQLWARN 診断情報

C の名前 COBOL および PL/I の名前	FORTRAN[1] の名前	RPG の名前	フィールドの値
SQLWARN0 sqlwarn[0]	SQLWRN(0) SQLWARN(1:1)	SQLWN0	他の標識がすべてブランクの場合は、ブランクが入ります。'W' または 'N' が入っている他の標識が 1 つでもあれば、'W' が入ります。
SQLWARN1 sqlwarn[1]	SQLWRN(1) SQLWARN(2:2)	SQLWN1	ストリング列の値をホスト変数に割り当てたときに、値が途中で切り捨てられると、ここに 'W' が入ります。*NOCNULRQD が CRTSQLCI または CRTSQLCPPI コマンド (または SET OPTION ステートメントの CNULRQD(*NO)) で指定され、ストリング列の値が C NUL 終了ホスト変数に割り当てられ、しかもそのホスト変数が結果を入れるには十分な大きさであるが、NUL 終了文字を入れるには十分な大きさでない場合は、ここに 'N' が入ります。
SQLWARN2 sqlwarn[2]	SQLWRN(2) SQLWARN(3:3)	SQLWN2	関数の引き数からヌル値が除去された場合に 'W' が入ります。MIN 関数の場合は、必ずしも 'W' にセットされません。この場合は、その関数の結果がヌル値の除去に左右されないからです。
SQLWARN3 sqlwarn[3]	SQLWRN(3) SQLWARN(4:4)	SQLWN3	列の数がホスト変数の数よりも多い場合、'W' が入ります。
SQLWARN4 sqlwarn[4]	SQLWRN(4) SQLWARN(5:5)	SQLWN4	準備済みの 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 sqlwarn[5]	SQLWRN(5) SQLWARN(6:6)	SQLWN5	予約済み

表 65. SQLWARN 診断情報 (続き)

C の名前 COBOL および PL/I の名前	FORTRAN[1] の名前	RPG の名前	フィールドの値
SQLWARN6 sqlwarn[6]	SQLWRN(6) SQLWARN(7:7)	SQLWN6	日付演算の結果、月の終わりの調整となった場合に、'W' が入ります。
SQLWARN7 sqlwarn[7]	SQLWRN(7) SQLWARN(8:8)	SQLWN7	予約済み
SQLWARN8 sqlwarn[8]	SQLWRX(1) SQLWARN(9:9)	SQLWN8	文字変換の結果に置換文字が含まれている場合に、'W' が入ります。
SQLWARN9 sqlwarn[9]	SQLWRX(2) SQLWARN(10:10)	SQLWN9	予約済み
SQLWARNA sqlwarn[10]	SQLWRX(3) SQLWARN(11:11)	SQLWNA	予約済み

表 66. SQLERRD の診断情報

C の名前 COBOL および PL/I の名前	FORTRAN[1] の名前	RPG の名前	フィールドの値
SQLERRD(1) sqlerrd[0]	SQLERR(1)	SQLER1	SQLCODE が 0 より小さい場合に、CPF エスケープ・メッセージの最後の 4 文字が入ります。例えば、メッセージが CPF5715 であれば、X'F5F7F1F5' が SQLERRD(1) に入れられます。[1] プロシージャーの呼び出しの場合、RETURN ステートメントで指定された戻り状況値が入ります。RETURN ステートメントで戻り状況値が指定されていない場合は、次のようになります。 • 呼び出しステートメントが成功した場合、0 が戻されます。 • 呼び出しステートメントが成功しなかった場合、-200 が戻されます。
SQLERRD(2) sqlerrd[1]	SQLERR(2)	SQLER2	SQL コードが 0 より小さい場合に、CPD 診断メッセージの最後の 4 文字が入ります。[1] CALL ステートメントの場合は、SQLERRD(2) には結果セットの数が入ります。

表 66. SQLERRD の診断情報 (続き)

C の名前 COBOL および PL/I の名前	FORTRAN[1] の名前	RPG の名前	フィールドの値
SQLERRD(3) sqlerrd[2]	SQLERR(3)	SQLER3	状況ステートメントの CONNECT の場合は、SQLERRD(3) には接続状況に関する情報が入ります。詳しくは、431 ページの『CONNECT (タイプ 2)』を参照してください。 INSERT、UPDATE、および DELETE の場合に、影響を受ける行の数を示します。 FETCH ステートメントの場合は、SQLERRD(3) には取り出された行の数が入ります。 PREPARE ステートメントの場合は、選択された行の見積数が入ります。行数が 2 147 483 647 より多い場合は、2 147 483 647 が戻されます。

表 66. SQLERRD の診断情報 (続き)

C の名前 COBOL および PL/I の名前	FORTRAN[1] の名前	RPG の名前	フィールドの値
SQLERRD(4) sqlerrd[3]	SQLERR(4)	SQLER4	PREPARE ステートメントの場合は、すべての実行で必要なリソースの相対的な数の見積もりが入ります。この数は、索引、ファイル・サイズ、または CPU モデルなどの現在の可用性によって異なります。これは、DB2 UDB for iSeries Query Optimizer によって選択されたアクセス・プランの見積コストです。 CONNECT および SET CONNECTION ステートメントの場合は、SQLERRD(4) には会話のタイプが入り、コミット可能な更新を行うことができるかどうかが示されます。詳しくは、431 ページの 『CONNECT (タイプ 2)』 を参照してください。 CALL ステートメントの場合、SQLERRD(4) には、プロシージャーが正常に実行されなかった原因となった、エラーのメッセージ・キーが入ります。QMHRTVPM API は、そのメッセージ・キーのメッセージ記述を戻すために使用できます。 DELETE、INSERT または UPDATE ステートメント内のトリガー・エラーの場合、SQLERRD(4) には、トリガー・プログラムからシグナルで通知されたエラーのメッセージ・キーが入ります。QMHRTVPM API は、そのメッセージ・キーのメッセージ記述を戻すために使用できます。 FETCH ステートメントの場合は、SQLERRD(4) には取り出された行の長さが入ります。

表 66. SQLERRD の診断情報 (続き)

C の名前 COBOL および PL/I の名前	FORTRAN[1] の名前	RPG の名前	フィールドの値
SQLERRD(5) sqlerrd[4]	SQLERR(5)	SQLER5	CALL ステートメントの場合は、SQLERRD(5) にはプロシージャーから戻された結果セットの数が入ります。 CONNECT または SET CONNECTION ステートメントでは、SQLERRD(5) には次の値が入ります。 • 接続が未接続である場合には -1 • 接続がローカルである場合には 0 • 接続がリモートである場合には 1 DELETE ステートメントの場合は、参照制約による影響を受けた行の数を示します。 EXECUTE IMMEDIATE または PREPARE ステートメントの場合、構文エラーの位置が入っていることがあります。 複数行の FETCH ステートメントでは、現在表にある最後の行が取り出された場合は、SQLERRD(5) には +100 が入ります。 PREPARE ステートメントの場合、SQLERRD(5) には、準備済みステートメント内のパラメーター・マーカーの数が入ります。
SQLERRD(6) sqlerrd[5]	SQLERR(6)	SQLER6	SQLCODE が 0 の場合に、SQL 完了メッセージの ID が入ります。 それ以外の場合、この値は未定義になります。

注:

[1] SQLERRD(1) および SQLERRD(2) が設定されるのは、設定の条件に該当する場合、および現行サーバーが DB2 UDB for iSeries である場合のみに限られます。

表 67. CONNECT および SET CONNECTION の場合の SQLERRMC の置換テキスト

記述	データ・タイプ
リレーショナル・データベース名	CHAR(18)
プロダクト識別 (SQLERRP と同じ)	CHAR(8)
サーバー・ジョブのユーザー ID	CHAR(10)
接続方式 (*DUW または *RUW)	CHAR(10)

表 67. CONNECT および SET CONNECTION の場合の SQLERRMC の置換テキスト (続き)

記述		データ・タイプ
DDM サーバー・クラス名		CHAR(10)
QAS	DB2 UDB for iSeries	
QDB2	DB2 UDB (OS/390 および z/OS 版)	
QDB2/2	DB2 UDB サーバー (OS/2 版)	
QDB2/6000	DB2 UDB サーバー (AIX/6000 版)	
QDB2/HPUX	DB2 UDB サーバー (HP-UX** 版)	
QDB2/NT	DB2 UDB サーバー (NT 版)	
QDB2/SUN	DB2 UDB サーバー (SUN Solaris** 版)	
QSQLDS/VM	DB2 (VM および VSE 版)	
QSQLDS/VSE	DB2 (VM および VSE 版)	
接続タイプ (SQLERRD(4) と同じ)		SMALLINT

INCLUDE SQLCA の宣言

C および C++ の場合、INCLUDE SQLCA 宣言は以下のステートメントと同等です。

```
#ifndef    SQLCODE
struct     sqlca
{
           unsigned char  sqlcaid[8];
           long           sqlcabc;
           long           sqlcode;
           short          sqlerrml;
           unsigned char  sqlerrmc[70];
           unsigned char  sqlerrp[8];
           long           sqlerrd[6];
           unsigned char  sqlwarn[11];
           unsigned char  sqlstate[5];
};
#define      SQLCODE   sqlca.sqlcode
#define      SQLWARN0  sqlca.sqlwarn[0]
#define      SQLWARN1  sqlca.sqlwarn[1]
#define      SQLWARN2  sqlca.sqlwarn[2]
#define      SQLWARN3  sqlca.sqlwarn[3]
#define      SQLWARN4  sqlca.sqlwarn[4]
#define      SQLWARN5  sqlca.sqlwarn[5]
#define      SQLWARN6  sqlca.sqlwarn[6]
#define      SQLWARN7  sqlca.sqlwarn[7]
#define      SQLWARN8  sqlca.sqlwarn[8]
#define      SQLWARN9  sqlca.sqlwarn[9]
#define      SQLWARNA  sqlca.sqlwarn[10]
#define      SQLSTATE  sqlca.sqlstate
#endif
struct sqlca sqlca;
```

COBOL の場合: INCLUDE SQLCA の宣言は、以下のステートメントと同等です。

```
01 SQLCA.
   05 SQLCAID      PIC X(8).
   05 SQLCABC      PIC S9(9) BINARY.
   05 SQLCODE      PIC S9(9) BINARY.
   05 SQLERRM.
      49 SQLERRML  PIC S9(4) BINARY.
      49 SQLERRMC  PIC X(70).
   05 SQLERRP      PIC X(8).
   05 SQLERRD      OCCURS 6 TIMES
                   PIC S9(9) BINARY.
   05 SQLWARN.
      10 SQLWARN0  PIC X(1).
      10 SQLWARN1  PIC X(1).
      10 SQLWARN2  PIC X(1).
      10 SQLWARN3  PIC X(1).
      10 SQLWARN4  PIC X(1).
      10 SQLWARN5  PIC X(1).
      10 SQLWARN6  PIC X(1).
      10 SQLWARN7  PIC X(1).
      10 SQLWARN8  PIC X(1).
      10 SQLWARN9  PIC X(1).
      10 SQLWARNA  PIC X(1).
   05 SQLSTATE     PIC X(5).
```

注: COBOL では、作業記憶域セクション (WORKING STORAGE SECTION) の外側で INCLUDE SQLCA を指定してはなりません。

FORTRAN の場合: INCLUDE SQLCA 宣言は、以下のステートメントと同等です。

```
 CHARACTER SQLCA(136)
 CHARACTER SQLCAID*8
 INTEGER*4 SQLCABC
 INTEGER*4 SQLCODE
 INTEGER*2 SQLERRML
 CHARACTER SQLERRMC*70
 CHARACTER SQLERRP*8
 INTEGER*4 SQLERRD(6)
 CHARACTER SQLWARN*11
 CHARACTER SQLSTOTE*5
 EQUIVALENCE (SQLCA(  1), SQLCAID)
 EQUIVALENCE (SQLCA(  9), SQLCABC)
 EQUIVALENCE (SQLCA( 13), SQLCODE)
 EQUIVALENCE (SQLCA( 17), SQLERRML)
 EQUIVALENCE (SQLCA( 19), SQLERRMC)
 EQUIVALENCE (SQLCA( 89), SQLERRP)
 EQUIVALENCE (SQLCA( 97), SQLERRD)
 EQUIVALENCE (SQLCA(121), SQLWARN)
 EQUIVALENCE (SQLCA(132), SQLSTOTE)

 INTEGER*4 SQLCOD,
C           SQLERR(6)
 INTEGER*2 SQLTXL
 CHARACTER  SQLERP*8,
C           SQLWRN(0:7)*1,
C           SQLWRX(1:3)*1,
C           SQLTXT*70,
C           SQLSTT*5,
C           SQLWRNWK*8,
C           SQLWRXWK*3,
C           SQLERRWK*24,
C           SQLERRDWK*24
 EQUIVALENCE (SQLWRN(1), SQLWRNWK)
 EQUIVALENCE (SQLWRX(1), SQLWRXWK)
 EQUIVALENCE (SQLCA(97), SQLERRDWK)
```

```
 EQUIVALENCE (SQLERR(1), SQLERRWK)
 COMMON /SQLCA1/SQLCOD,SQLERR,SQLTXL
 COMMON /SQLCA2/SQLERP,SQLWRN,SQLTXT,SQLWRX,SQLSTT
```

PL/I の場合: INCLUDE SQLCA の宣言は、以下のステートメントと同等です。

```
DCL 1 SQLCA,
      2 SQLCAID      CHAR(8),
      2 SQLCABC      BIN FIXED(31),
      2 SQLCODE      BIN FIXED(31),
      2 SQLERRM      CHAR(70) VAR,
      2 SQLERRP      CHAR(8),
      2 SQLERRD(6)   BIN FIXED(31),
      2 SQLWARN,
        3 SQLWARN0   CHAR(1),
        3 SQLWARN1   CHAR(1),
        3 SQLWARN2   CHAR(1),
        3 SQLWARN3   CHAR(1),
        3 SQLWARN4   CHAR(1),
        3 SQLWARN5   CHAR(1),
        3 SQLWARN6   CHAR(1),
        3 SQLWARN7   CHAR(1),
        3 SQLWARN8   CHAR(1),
        3 SQLWARN9   CHAR(1),
        3 SQLWARNA   CHAR(1),
      2 SQLSTATE     CHAR(5);
```

RPG/400 の場合: SQLCA の宣言は、以下のステートメントと同等です。

```
ISQLCA          DS
I                                       1   8 SQLAID            SQL
I                                    B  9  120SQLABC            SQL
I                                    B 13  160SQLCOD            SQL
I                                    B 17  180SQLERL            SQL
I                                      19  88 SQLERM            SQL
I                                      89  96 SQLERP            SQL
I                                      97 120 SQLERR            SQL
I                                    B 97 1000SQLER1            SQL
I                                    B101 1040SQLER2            SQL
I                                    B105 1080SQLER3            SQL
I                                    B109 1120SQLER4            SQL
I                                    B113 1160SQLER5            SQL
I                                    B117 1200SQLER6            SQL
I                                      121 131 SQLWRN           SQL
I                                      121 121 SQLWN0           SQL
I                                      122 122 SQLWN1           SQL
I                                      123 123 SQLWN2           SQL
I                                      124 124 SQLWN3           SQL
I                                      125 125 SQLWN4           SQL
I                                      126 126 SQLWN5           SQL
I                                      127 127 SQLWN6           SQL
I                                      128 128 SQLWN7           SQL
I                                      129 129 SQLWN8           SQL
I                                      130 130 SQLWN9           SQL
I                                      131 131 SQLWNA           SQL
I                                      132 136 SQLSTT           SQL
```

ILE RPG/400 の場合: SQLCA の宣言は、以下のステートメント同等です。

```
D*      SQL Communications area
D SQLCA           DS
D  SQLAID                 1      8A
D  SQLABC                 9     12B 0
D  SQLCOD                13     16B 0
D  SQLERL                17     18B 0
D  SQLERM                19     88A
D  SQLERP                89     96A
```

```
D  SQLERRD               97    120B 0 DIM(6)
D  SQLERR                97    120A
D   SQLER1               97    100B 0
D   SQLER2              101    104B 0
D   SQLER3              105    108B 0
D   SQLER4              109    112B 0
D   SQLER5              113    116B 0
D   SQLER6              117    120B 0
D  SQLWRN                121    131A
D   SQLWN0               121    121A
D   SQLWN1               122    122A
D   SQLWN2               123    123A
D   SQLWN3               124    124A
D   SQLWN4               125    125A
D   SQLWN5               126    126A
D   SQLWN6               127    127A
D   SQLWN7               128    128A
D   SQLWN8               129    129A
D   SQLWN9               130    130A
D   SQLWNA               131    131A
D  SQLSTT                132    136A
D*  End of SQLCA
```

付録 C. SQL 記述子域 (SQLDA)

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あり、PREPARE、OPEN、CALL、FETCH、および EXECUTE ステートメントで、必要に応じて使用することができます。 SQLDA は、DESCRIBE ステートメントで使用し、ホスト変数のアドレスによって変更し、その後で FETCH ステートメントで再度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ますが、事前定義宣言が用意されているのは、C (および C++)、COBOL、ILE RPG/400、PL/I、および REXX の場合だけです。 REXX の場合、SQLDA は他の言語の場合と多少異なります。REXX での SQLDA の使用法については、DB2 UDB for iSeries ホスト言語での SQL プログラミングを参照してください。

SQLDA の情報の意味は、その用途によって異なります。 PREPARE および DESCRIBE ステートメントで使用すると、SQLDA によって準備済みステートメントに関する情報がアプリケーション・プログラムに提供されます。
OPEN、CALL、EXECUTE、および FETCH で使用すると、SQLDA によって、ホスト変数に関する情報がデータベース・マネージャーに提供されます。

フィールドの説明

SQLDA は、ヘッダー構造の中の 4 つの変数と、それに続く一連の 5 つの変数からなる任意の数のオカレンス (これらは一括して SQLVAR という名前が付けられている) から構成されます。 OPEN、CALL、FETCH、および EXECUTE では、SQLVAR の各オカレンスで、それぞれホスト変数を 1 つずつ記述します。
PREPARE および DESCRIBE では、SQLVAR の各オカレンスで、結果表の列を記述します。

SQL INCLUDE ステートメントを使用することにより、次のようなフィールド名が組み込まれます。

76. この欄では、小文字の名前は C の名前を示し、大文字の名前は COBOL、PL/I、または RPG の名前を示しています。

SQLDA

表 68. SQLDA ヘッダーのフィールドの説明

C の名前[76] PL/I の名前 COBOL の名前	フィールド・データ・タイプ	DESCRIBE および PREPARE で使用する場合 (SQLN 以外は、データベース・マネージャーによってセットされる)	FETCH、OPEN、CALL、または EXECUTE で使用する場合 (ユーザーがステートメントを実行する前にセットする)
sqldaid SQLDAID	CHAR(8)	記憶ダンプのための「目印」として、'SQLDA ' が入ります。 SQLDAID の 7 番目のバイトは、各列に複数の SQLVAR 項目が必要かどうかを判断するために使用できます。詳細は、 880 ページの『必要な SQLVAR オカレンスの数の決定』を参照してください。	7 番目のバイトの '2' は、各列に 2 つの SQLVAR 項目が割り振られたことを示します。 7 番目のバイトの '3' は、各列に 3 つの SQLVAR 項目が割り振られたことを示します。 7 番目のバイトの '4' は、各列に 4 つの SQLVAR 項目が割り振られたことを示します。
sqldabc SQLDABC	INTEGER	SQLDA の長さ。	SQLDA として割り振られた記憶域のサイズ (バイト数)。 SQLN に指定されているオカレンスが入るだけの記憶域を割り振る必要があります。 SQLDABC には、16+SQLN*(80) 以上の値をセットする必要があります (80 は SQLVAR のオカレンスの長さ)。 LOB または特殊タイプが指定された場合には、各パラメーター・マーカーごとに 2 つの SQLVAR 項目が必要です。
sqln SQLN	SMALLINT	データベース・マネージャーでは変更しません。 PREPARE または DESCRIBE ステートメントを実行する前に、ユーザー側でゼロ以上の値をセットしなければなりません。この値は、結果内の列の数と同じか、それより大きい値にセットするか、複数の SQLVAR 項目セットが必要な場合には、結果内の列の数の倍数にセットする必要があります。 SQLVAR 配列のオカレンス数を指示します。	SQLDA に用意する SQLVAR 配列の合計オカレンス数。 SQLN には、ゼロ以上の値をセットしなければなりません。 LOB または特殊タイプが指定された場合には、各パラメーター・マーカーごとに 2 つの SQLVAR 項目が必要であり、SQLN はパラメーター・マーカー数の 2 倍をセットしなければなりません。
sqld SQLD	SMALLINT	SQLVAR 配列の各オカレンスによって記述する列の数。(記述するステートメントが選択ステートメント以外の場合は、ゼロ)。	このステートメントが実行されたときに SQLDA で使用するように SQLVAR によって記述するホスト変数の数。 SQLD には、ゼロ以上で SQLN 以下の値をセットしなければなりません。

SQLVAR のオカレンスのフィールドの説明

SQLDA によって記述される各列またはホスト変数ごとに、2 つのタイプの SQLVAR 項目があります。

基本 SQLVAR 項目

基本 SQLVAR 項目は、常に存在する項目です。この項目のフィールドには、その列またはホスト変数に関する基本情報 (データ・タイプ・コード、長さ属性 (LOB の場合を除く)、列名 (またはラベル)、CCSID、ホスト変数アドレス、標識変数アドレスなど) が含まれます。

拡張 SQLVAR 項目

拡張 SQLVAR 項目は、結果に LOB または特殊タイプの列が含まれている場合に (各列ごとに) 必要となります。 特殊タイプの場合、拡張 SQLVAR には特殊タイプ名が入ります。 LOB の場合、拡張 SQLVAR には、ホスト変数の長さ属性と、実際の長さを含むバッファーを指すポインターが入ります。 ロケーターまたはファイル参照変数を使用して LOB を表す場合、拡張 SQLVAR は不要です。

拡張 SQLVAR 項目は、次の場合にも各列ごとに必要となります。

- USING BOTH が指定されている場合。これは、列名およびラベルが戻されることを示します。
- USING ALL が指定されている場合。これは、列名、ラベル、およびシステム列名が戻されることを示します。

LOB および特殊タイプ情報を戻す拡張 SQLVAR 内のフィールドはオーバーラップせず、LOB およびラベル情報を戻すフィールドもオーバーラップしません。ラベル、LOB、および特殊タイプの組み合わせによっては、情報を戻すのに列ごとに複数の拡張 SQLVAR 項目が必要になる場合があります。 880 ページの『必要な SQLVAR オカレンスの数の決定』を参照してください。

表 69、 880 ページの表 70、および 880 ページの表 71 は、基本および拡張 SQLVAR 項目をマップする方法を示しています。 基本と拡張の両方の SQLVAR 項目を含む SQLDA の場合、基本 SQLVAR 項目は最初のブロック内にあり、その後に拡張 SQLVAR 項目のブロックが続き、さらに必要であれば、その後に第 2、第 3 の拡張 SQLVAR 項目のブロックが続きます。 各ブロックでの SQLVAR 項目のオカレンス数は、多数の拡張 SQLVAR 項目が未使用である可能性があっても、SQLD 内の値と同じになります。

表 69. USING NAMES、USING SYSTEM NAMES、USING LABELS または USING ANY の SQLVAR 配列の内容

LOB	DISTINCT タイプ	SQLDAID の 7 番目のバイト	SQLN 最小値	最初のセット (基本)	2 番目のセット (拡張)	3 番目のセット (拡張)	4 番目のセット (拡張)
なし	なし	ブランク	n	列名、システム列名、またはラベル	使用不可	使用不可	使用不可
あり	なし	2	2n	列名、システム列名、またはラベル	LOB	使用不可	使用不可
なし	あり	2	2n	列名、システム列名、またはラベル	特殊タイプ	使用不可	使用不可

表 69. USING NAMES、USING SYSTEM NAMES、USING LABELS または USING ANY の SQLVAR 配列の内容 (続き)

LOB	DISTINCT タイプ	SQLDAID の 7 番目のバイト	SQLN 最小値	最初のセット (基本)	2 番目のセット (拡張)	3 番目のセット (拡張)	4 番目のセット (拡張)
あり	あり	2	2n	列名、システム列名、またはラベル	LOB および特殊タイプ	使用不可	使用不可

表 70. USING BOTH の SQLVAR 配列の内容

LOB	DISTINCT タイプ	SQLDAID の 7 番目のバイト	SQLN 最小値	最初のセット (基本)	2 番目のセット (拡張)	3 番目のセット (拡張)	4 番目のセット (拡張)
なし	なし	2	2n	列名	ラベル	使用不可	使用不可
あり	なし	2	2n	列名	LOB およびラベル	使用不可	使用不可
なし	あり	3	3n	列名	特殊タイプ	ラベル	使用不可
あり	あり	3	3n	列名	LOB および特殊タイプ	ラベル	使用不可

表 71. USING ALL の SQLVAR 配列の内容

LOB	DISTINCT タイプ	SQLDAID の 7 番目のバイト	SQLN 最小値	最初のセット (基本)	2 番目のセット (拡張)	3 番目のセット (拡張)	4 番目のセット (拡張)
なし	なし	3	3n	システム列名	ラベル	列名	使用不可
あり	なし	3	3n	システム列名	LOB およびラベル	列名	使用不可
なし	あり	4	4n	システム列名	特殊タイプ	ラベル	列名
あり	あり	4	4n	システム列名	LOB および特殊タイプ	ラベル	列名

必要な SQLVAR オカレンスの数の決定

必要な SQLVAR オカレンス数は、SQLDA に提供されたステートメントと、記述されている列またはパラメーターのデータ・タイプによって決まります。 詳細については、上記の表を参照してください。

SQLDAID の 7 番目のバイトは常に、必要な SQLVAR のセット数にセットされます。

SQLD が十分な数の SQLVAR オカレンスにセットされない場合、

- SQLD は、すべてのセットに必要な SQLVAR オカレンスの合計数にセットされます。
- 少なくとも基本 SQLVAR 項目用に十分な数の SQLVAR が指定されている場合、SQLCA の SQLCODE フィールドに警告 (SQLSTATE 01594) が戻されます。この場合、基本 SQLVAR 項目は戻されますが、拡張 SQLVAR は戻されません。

- 基本 SQLVAR 項目用にさえも十分な数の SQLVAR が指定されていない場合は、SQLCA の SQLCODE フィールドに警告 (SQLSTATE 01005) が戻されます。 SQLVAR 項目は戻されません。

表 72. SQLVAR のフィールドの説明

C の名前[77] **COBOL の名前** **PL/I の名前** **RPG の名前**	**フィールド・データ・タイプ**	**DESCRIBE および PREPARE で使用する場合 (データベース・マネージャーがセットする)**	**FETCH、OPEN、CALL、および EXECUTE で使用する場合 (ユーザーがステートメントを実行する前にセットする)**
sqltype SQLTYPE	SMALLINT	列のデータ・タイプと、その列にヌルを入れられるかどうかを指示します。データ・タイプを示すコードについては、 884 ページの表 74 を参照してください。 特殊タイプの場合、特殊タイプの基本となっているデータ・タイプがこのフィールドに置かれます。基本 SQLVAR には、これが特殊タイプの記述の一部であることを示すものは含まれません。	ホスト変数のデータ・タイプと、その変数に標識変数が指定されているかどうかを指示します。データ・タイプを示すコードについては、 884 ページの表 74 を参照してください。
sqllen SQLLEN	SMALLINT	列の長さ属性です。 日付/時刻の列の場合、その値のストリング表現の長さ。 884 ページの表 74 を参照してください。 LOB の場合、その LOB の長さ属性に関係なく、この値は 0 になります。 拡張 SQLVAR 項目内のフィールド SQLLONGLEN には、LOB の長さ属性が入ります。	ホスト変数の長さ属性です。 884 ページの表 74 を参照してください。 LOB の場合、その LOB の長さ属性に関係なく、この値は 0 になります。 拡張 SQLVAR 項目内のフィールド SQLLONGLEN には、LOB の長さ属性が入ります。
sqlres SQLRES	CHAR(12)	予約済み。 SQLDATA の境界合わせ用。	予約済み。 SQLDATA の境界合わせ用。
sqldata SQLDATA	ポインター	ストリング列の CCSID (886 ページの表 75 を参照してください)。	ホスト変数のアドレスが入ります。 LOB ホスト変数の場合、拡張 SQLVAR 内の SQLDATALEN フィールドがヌルであれば、4 バイトの LOB 長を指し、その直後に LOB データが続きます。 拡張 SQLVAR 内の SQLDATALEN フィールドがヌルでない場合は、LOB データを指し、SQLDATALEN フィールドは 4 バイトの LOB 長を指します。

表 72. SQLVAR のフィールドの説明 (続き)

C の名前[77] COBOL の名前 PL/I の名前 RPG の名前	フィールド・データ・タイプ	DESCRIBE および PREPARE で使用する場合 (データベース・マネージャーがセットする)	FETCH、OPEN、CALL、および EXECUTE で使用する場合 (ユーザーがステートメントを実行する前にセットする)
sqlind SQLIND	ポインター	予約済み	標識変数のアドレスが入ります。標識変数がない (SQLTYPE の値が偶数である) 場合は、使用されません。
sqlname SQLNAME	VARCHAR (30)	修飾のない列名。 列に名前がない場合、ストリングは式から構成されて、戻されます。 この名前には大文字小文字の区別があります。全体を区切り文字で囲んではなりません。	ホスト変数の CCSID (886 ページの表 75 を参照) が入ります。

表 73. 拡張 SQLVAR のフィールドの説明

C の名前[78] COBOL の名前 PL/I の名前 RPG の名前	フィールド・データ・タイプ	DESCRIBE および PREPARE で使用する場合 (データベース・マネージャーがセットする)	FETCH、OPEN、CALL、および EXECUTE で使用する場合 (ユーザーがステートメントを実行する前にセットする)
len.sqllonglen SQLLONGL SQLLONGLEN	INTEGER	LOB 列の長さ属性。	LOB ホスト変数の長さ属性。データベース・マネージャーは、これらのデータ・タイプの場合、基本 SQLVAR 内の SQLLEN フィールドは無視します。長さ属性は、BLOB または CLOB のバイト数と、DBCLOB の文字数を示します。
*	CHAR(12)	予約済み。 SQLDATALEN の境界合わせ用。	予約済み。 SQLDATALEN の境界合わせ用。
*	ポインター	予約済み。	予約済み。

77. この欄の小文字の名前は C の名前を示し、大文字の名前は PL/I、COBOL、および RPG の名前を示しています。

表 73. 拡張 SQLVAR のフィールドの説明 (続き)

C の名前[78] COBOL の名前 PL/I の名前 RPG の名前	フィールド・データ・タイプ	DESCRIBE および PREPARE で使用する場合 (データベース・マネージャーがセットする)	FETCH、OPEN、CALL、および EXECUTE で使用する場合 (ユーザーがステートメントを実行する前にセットする)
sqldatalen SQLDATAL SQLDATALEN	ポインター	使用されません。	LOB ホスト変数にのみ使用されます。 このフィールドの値がヌルの場合、LOB の実際の長さは、一致する基本 SQLVAR 内の SQLDATA フィールドが指す最初の 4 バイトに保管され、LOB データは 4 バイトの長さの直後に続きます。 実際の長さは、BLOB または CLOB のバイト数と、DBCLOB の 2 バイトの文字数を示します。 このフィールドの値がヌルではない場合は、このフィールドは、LOB の実際の長さ (バイト単位) を含む 4 バイトの長さのバッファーを指します (DBCLOB の場合でも)。そして、一致する基本 SQLVAR の SQLDATA フィールドは、LOB データを指します。 このフィールドが使用されるかどうかに関係なく、フィールド SQLLONGLEN はセットしなければなりません。
sqldatatype_name SQLTNAME SQLDATATYPE-NAME	VARCHAR (30)	拡張 SQLVAR の SQLTNAME フィールドは、次のいずれかにセットされます。 • 特殊タイプの列の場合、データベース・マネージャーはこれを完全修飾特殊タイプ名にセットします。この修飾名が 30 バイトより長い場合は、切り捨てられます。 • ラベルの場合、データベース・マネージャーは、これをラベルの最初の 20 バイトにセットします。 • 列名の場合、データベース・マネージャーはこれを列名にセットします。	使用されません。

SQLTYPE と SQLLEN

以下の表は、SQLDA の SQLTYPE および SQLLEN フィールドに入る値を示したものです。 PREPARE および DESCRIBE で使用した場合に、SQLTYPE の値が偶数ならば、その列にはヌルが許されないことを示します。また、SQLTYPE の値が奇数ならば、その列にヌルが許されることを示します。

注: PREPARE および DESCRIBE ステートメントにおいて、1 つのオペランドがヌル可能であるか、または式の結果が -2 のマッピング・エラーヌル値になる場合、その式に奇数値が戻されます。

FETCH、OPEN、CALL、および EXECUTE で使用した場合、SQLTYPE の値が偶数ならば、標識変数が指定されていないことを意味し、奇数ならば、SQLIND に標識変数のアドレスが入っていることを意味します。

表 74. PREPARE、DESCRIBE、FETCH、OPEN、CALL、または EXECUTE の場合の SQLTYPE および SQLLEN の値

	PREPARE および DESCRIBE の場合		FETCH、OPEN、CALL、および EXECUTE の場合	
SQLTYPE	COLUMN DATA TYPE	SQLLEN	HOST VARIABLE DATA TYPE	SQLLEN
384/385	日付	10	日付の固定長文字ストリング表現	ホスト変数の長さ属性
388/389	時刻	8	時刻の固定長文字ストリング表現	ホスト変数の長さ属性
392/393	タイム・スタンプ	26	タイム・スタンプの固定長文字ストリング表現	ホスト変数の長さ属性
396/397	データ・リンク	列の長さ属性	データ・リンク	ホスト変数の長さ属性
400/401	該当しない	該当しない	NUL で終了するグラフィック・ストリング	ホスト変数の長さ属性
404/405	BLOB	0 [80]	BLOB	使用されません。[80]
408/409	CLOB	0 [80]	CLOB	使用されません。[80]
412/413	DBCLOB	0 [80]	DBCLOB	使用されません。[80]
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	長可変長文字ストリング	列の長さ属性	長可変長文字ストリング	ホスト変数の長さ属性
460/461	該当しない	該当しない	NUL で終了する文字ストリング	ホスト変数の長さ属性
464/465	可変長グラフィック・ストリング	列の長さ属性	可変長グラフィック・ストリング	ホスト変数の長さ属性
468/469	固定長グラフィック・ストリング	列の長さ属性	固定長グラフィック・ストリング	ホスト変数の長さ属性

78. この欄の小文字の名前は C の名前を示し、最初の大文字の名前は PL/I および RPG の名前を示しています。 2 番目の大文字の名前は COBOL の名前を示しています。

表 74. PREPARE、DESCRIBE、FETCH、OPEN、CALL、または EXECUTE の場合の SQLTYPE および SQLLEN の値 (続き)

SQLTYPE	PREPARE および DESCRIBE の場合		FETCH、OPEN、CALL、および EXECUTE の場合	
	COLUMN DATA TYPE	SQLLEN	HOST VARIABLE DATA TYPE	SQLLEN
472/473	長い可変長グラフィック・ストリング	列の長さ属性	長いグラフィック・ストリング	ホスト変数の長さ属性
476/477	該当しない	該当しない	PASCAL の L-ストリング	ホスト変数の長さ属性
480/481	浮動小数点数	単精度では 4、倍精度では 8	浮動小数点数	単精度では 4、倍精度では 8
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
488/489	ゾーン 10 進数	バイト 1 は精度、バイト 2 は位取り	ゾーン 10 進数	バイト 1 は精度、バイト 2 は位取り
492/493	大整数	8 [79]	大整数	8
496/497	大整数	4 [79]	大整数	4
500/501	短整数	2 [79]	短整数	2
504/505	該当しない	該当しない	DISPLAY SIGN LEADING SEPARATE	バイト 1 は精度、バイト 2 は位取り
904/905	ROWID	40	ROWID	40
960/961	該当しない	該当しない	BLOB ロケーター	4
964/965	該当しない	該当しない	CLOB ロケーター	4
968/969	該当しない	該当しない	DBCLOB ロケーター	4
916/917	該当しない	該当しない	BLOB ファイル参照変数	267
920/921	該当しない	該当しない	CLOB ファイル参照変数	267
924/925	該当しない	該当しない	DBCLOB ファイル参照変数	267

SQLDATA または SQLNAME

OPEN、FETCH、CALL、および EXECUTE ステートメントでは、SQLVAR エレメントの SQLNAME フィールドを使用して、ストリング・ホスト変数の CCSID を指定することができます。 SQLNAME フィールドによって CCSID を指定する場合は、SQLNAME の長さを 8 にセットしなければなりません。さらに、SQLNAME の最初の 4 バイトは次の表に従ってセットする必要があります。 CCSID の指定がない場合、ジョブの CCSID が使用されます。

79. SQLDA では、2 進数は長さ 2、4、または 8 で表される場合と、バイト 1 の精度とバイト 2 の位取りによって表される場合があります。最初のバイトが x'00' より大きい場合は、精度および位取りが入っていることを示します。

80. 拡張 SQLVAR 内のフィールド SQLLONGLEN には、列の長さ属性が入ります。

DESCRIBE、DESCRIBE TABLE、および PREPARE ステートメントにおいて、結果表の列がストリング列の場合、SQLVAR のエレメントの SQLDATA フィールドにその列の CCSID が入ります。その CCSID は、表 75 に示すようにバイト 3 とバイト 4 に入ります。

表 75. SQLDATA または SQLNAME に示される CCSID の値

データ・タイプ	サブタイプ	バイト 1 と 2	バイト 3 と 4
文字	SBCS データ	X'0000'	ccsid
文字	MIXED (混合) データ	X'0000'	ccsid
文字	ビット・データ	X'0000'	65535
図形	該当しない	X'0000'	ccsid
上記以外のデータ・タイプ	該当しない	該当しない	該当しない

認識されずサポートされない SQLTYPES

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の双方で使用可能なデータ・タイプ・サポートのレベルに依存しています。 これは、新しいデータ・タイプをプロダクトに追加する際に特に重要です。

データの送信側または受信側が、新しいデータ・タイプをサポートしている場合とサポートしていない場合があり、認識している場合と認識さえしていない場合があります。 状況に応じて、新しいデータ・タイプが戻される場合、データの送信側および受信側の両者が合意した互換データ・タイプが戻される場合、あるいはエラーが発生する場合があります。

以下の表は、送信側および受信側の両者が互換データ・タイプを使用することを合意した場合に、発生する可能性があるマッピングを示しています。 このマッピングが発生するのは、送信側または受信側の少なくとも一方が提供されたデータ・タイプをサポートしていない場合です。サポートされないデータ・タイプは、アプリケーションまたはデータベース・マネージャーのいずれかによって提供される可能性があります。

表 76. サポートされないデータ・タイプの互換データ・タイプ

データ・タイプ	互換データ・タイプ
BIGINT	DECIMAL(19,0)
ROWID	VARCHAR(40) ビット・データ用

INCLUDE SQLDA の宣言

C および C++ の場合

C および C++ の場合、INCLUDE SQLDA 宣言は以下と同等です。

```
#ifndef SQLDASIZE
struct sqlda
{
       unsigned char  sqldaid[8];
       long           sqldabc;
       short          sqln;
       short          sqld;
       struct sqlvar
       {
              short          sqltype;
              short          sqllen;
              unsigned char *sqldata;
              short         *sqlind;
              struct sqlname
              {
                     short          length;
                     unsigned char  data[30];
              } sqlname;
       } sqlvar[1];
};

struct sqlvar2
       { struct
                {  long           sqllonglen;
                   char           reserve1[28];
                }  len;
                char *sqldatalen;
                struct sqldistinct_type
                       {  short          length;
                          unsigned char  data[30];
                       } sqldatatype_name;
       };

#define  SQLDASIZE(n) (sizeof(struct sqlda)+(n-1) * sizeof(struct sqlvar))
#endif
```

図 11. C および C++ の場合の INCLUDE SQLDA 宣言 (1/3)

```
/*********************************************************************/
/* Macros for using the sqlvar2 fields.                              */
/*********************************************************************/

/*********************************************************************/
/*   '2' in the 7th byte of sqldaid indicates a doubled number of    */
/*       sqlvar entries.                                             */
/*   '3' in the 7th byte of sqldaid indicates a tripled number of    */
/*       sqlvar entries.                                             */
/*********************************************************************/
#define SQLDOUBLED '2'
#define SQLSINGLED ' '

/*********************************************************************/
/* GETSQLDOUBLED(daptr) returns 1 if the SQLDA pointed to by         */
/* daptr has been doubled, or 0 if it has not been doubled.          */
/*********************************************************************/
#define GETSQLDOUBLED(daptr) (((daptr)->sqldaid[6]==  \
 (char) SQLDOUBLED)  ?                                \
    (1)           :      \
    (0)             )

/*********************************************************************/
/* SETSQLDOUBLED(daptr, SQLDOUBLED) sets the 7th byte of sqldaid     */
/* to '2'.                                                           */
/* SETSQLDOUBLED(daptr, SQLSINGLED) sets the 7th byte of sqldaid     */
/* to be a ' '.                                                      */
/*********************************************************************/
#define SETSQLDOUBLED(daptr, newvalue)               \
  (((daptr)->sqldaid[6] =(newvalue)))

/*********************************************************************/
/* GETSQLDALONGLEN(daptr,n) returns the data length of the nth       */
/* entry in the sqlda pointed to by daptr. Use this only if the      */
/* sqlda was doubled or tripled and the nth SQLVAR entry has a       */
/* LOB datatype.                                                     */
/*********************************************************************/
#define GETSQLDALONGLEN(daptr,n) ((long) (((struct sqlvar2 *) \
&((daptr)->sqlvar[(n) +((daptr)->sqld)])) ->len.sqllonglen))

/*********************************************************************/
/* SETSQLDALONGLEN(daptr,n,len) sets the sqllonglen field of the     */
/* sqlda pointed to by daptr to len for the nth entry. Use this only */
/* if the sqlda was doubled or tripled and the nth SQLVAR entry has  */
/* a LOB datatype.                                                   */
/*********************************************************************/
#define SETSQLDALONGLEN(daptr,n,length) {            \
 struct sqlvar2 *var2ptr;                            \
 var2ptr = (struct sqlvar2 *) &((daptr)->sqlvar[(n)+  \
   ((daptr)->sqld)]);                                \
 var2ptr->len.sqllonglen = (long) (length);              \
}

/*********************************************************************/
/* SETSQLDALENPTR(daptr,n,ptr) sets a pointer to the data length for */
/* the nth entry in the sqlda pointed to by daptr.                   */
/* Use this only if the sqlda has been doubled or tripled.           */
/*********************************************************************/
#define SETSQLDALENPTR(daptr,n,ptr) {                \
 struct sqlvar2 *var2ptr;                            \
 var2ptr = (struct sqlvar2 *) &((daptr)->sqlvar[(n)+  \
   ((daptr)->sqld)]);                                \
 var2ptr->sqldatalen = (char *) ptr;                 \
 }
```

図 11. C および C++ の場合の INCLUDE SQLDA 宣言 (2/3)

```
/*********************************************************************/
/* GETSQLDALENPTR(daptr,n) returns a pointer to the data length for  */
/* the nth entry in the sqlda pointed to by daptr. Unlike the inline */
/* value (union sql8bytelen len), which is 8 bytes, the sqldatalen   */
/* pointer field returns a pointer to a long (4 byte) integer.       */
/* If the SQLDATALEN pointer is zero, a NULL pointer is be returned. */
/*                                                                   */
/* NOTE: Use this only if the sqlda has been doubled or tripled.     */
/*********************************************************************/
#define GETSQLDALENPTR(daptr,n) (                       \
  (((struct sqlvar2 *) &(daptr)->sqlvar[(n) +           \
    (daptr)->sqld])->sqldatalen == NULL) ?              \
   ((long *) NULL ) : ((long *) ((struct sqlvar2 *)   \
  &(daptr)->sqlvar[(n) + (daptr) ->sqld])->sqldatalen))
```

図 11. C および C++ の場合の INCLUDE SQLDA 宣言 (3/3)

COBOL の場合

COBOL の場合、INCLUDE SQLDA の宣言は、以下のステートメントと同等です。

```
1 SQLDA.
  05 SQLDAID     PIC X(8).
  05 SQLDABC     PIC S9(9) BINARY.
  05 SQLN        PIC S9(4) BINARY.
  05 SQLD        PIC S9(4) BINARY.
  05 SQLVAR OCCURS 0 TO 409 TIMES DEPENDING ON SQLD.
     10 SQLTYPE   PIC S9(4) BINARY.
     10 SQLLEN    PIC S9(4) BINARY.
     10 FILLER  REDEFINES SQLLEN.
        15 SQLPRECISION PIC X.
        15 SQLSCALE     PIC X.
     10 SQLRES    PIC X(12).
     10 SQLDATA   POINTER.
     10 SQLIND    POINTER.
     10 SQLNAME.
        49 SQLNAMEL PIC S9(4) BINARY.
        49 SQLNAMEC PIC X(30).
```

図 12. INCLUDE SQLDA の宣言 (COBOL の場合)

ILE COBOL の場合

ILE COBOL の場合、INCLUDE SQLDA の宣言は、以下のステートメントと同等です。

```
1 SQLDA.
  05 SQLDAID     PIC X(8).
  05 SQLDABC     PIC S9(9) BINARY.
  05 SQLN        PIC S9(4) BINARY.
  05 SQLD        PIC S9(4) BINARY.
  05 SQLVAR OCCURS 0 TO 409 TIMES DEPENDING ON SQLD.
     10 SQLVAR1.
        15 SQLTYPE   PIC S9(4) BINARY.
        15 SQLLEN    PIC S9(4) BINARY.
        15 FILLER  REDEFINES SQLLEN.
           20 SQLPRECISION PIC X.
           20 SQLSCALE     PIC X.
        15 SQLRES    PIC X(12).
        15 SQLDATA   POINTER.
        15 SQLIND    POINTER.
        15 SQLNAME.
           49 SQLNAMEL PIC S9(4) BINARY.
           49 SQLNAMEC PIC X(30).
     10 SQLVAR2 REDEFINES SQLVAR1.
        15 SQLVAR2-RESERVED-1 PIC S9(9) BINARY.
        15 SQLLONGLEN         REDEFINES SQLVAR2-RESERVED-1
                              PIC S9(9) BINARY.
        15 SQLVAR2-RESERVED-2 PIC X(28).
        15 SQLDATALEN         POINTER.
        15 SQLDATATYPE-NAME.
           49 SQLDATATYPE-NAMEL PIC S9(4) BINARY.
           49 SQLDATATYPE-NAMEC PIC X(30).
```

図 13. INCLUDE SQLDA の宣言 (ILE COBOL の場合)

PL/I の場合

PL/I の場合、INCLUDE SQLDA の宣言は、以下のステートメントと同等です。

```
DCL 1 SQLDA BASED(SQLDAPTR),
      2 SQLDAID     CHAR(8),
      2 SQLDABC     BIN FIXED(31),
      2 SQLN        BIN FIXED,
      2 SQLD        BIN FIXED,
      2 SQLVAR      (99),
        3 SQLTYPE   BIN FIXED,
        3 SQLLEN    BIN FIXED,
        3 SQLRES    CHAR(12),
        3 SQLDATA   PTR,
        3 SQLIND    PTR,
        3 SQLNAME   CHAR(30) VAR,

    1 SQLDA2 BASED(SQLDAPTR),
      2 SQLDAID2    CHAR(8),
      2 SQLDABC2    FIXED(31) BINARY,
      2 SQLN2       FIXED(15) BINARY,
      2 SQLD2       FIXED(15) BINARY,
      2 SQLVAR2     (99),
        3 SQLBIGLEN,
          4 SQLLONGL FIXED(31) BINARY,
          4 SQLRSVDL FIXED(31) BINARY,
        3 SQLDATAL  POINTER,
        3 SQLTNAME  CHAR(30) VAR;

    DECLARE SQLSIZE    FIXED(15) BINARY;
    DECLARE SQLDAPTR   PTR;
    DECLARE SQLDOUBLED CHAR(1)   INITIAL('2') STATIC;
    DECLARE SQLSINGLED CHAR(1)   INITIAL(' ') STATIC;
```

図 14. INCLUDE SQLDA の宣言 (PL/I の場合)

ILE RPG/400 の場合

ILE RPG/400 の場合、INCLUDE SQLDA の宣言は、以下と同等です。

```
D*      SQL Descriptor area
D SQLDA           DS
D  SQLDAID                1      8A
D  SQLDABC                9     12B 0
D  SQLN                  13     14B 0
D  SQLD                  15     16B 0
D  SQL_VAR                      80A   DIM(SQL_NUM)
D                        17     18B 0
D                        19     20B 0
D                        21     32A
D                        33     48*
D                        49     64*
D                        65     66B 0
D                        67     96A
D*
D SQLVAR          DS
D  SQLTYPE                1      2B 0
D  SQLLEN                 3      4B 0
D  SQLRES                 5     16A
D  SQLDATA               17     32*
D  SQLIND                33     48*
D  SQLNAMELEN            49     50B 0
D  SQLNAME               51     80A
D*
D SQLVAR2         DS
D  SQLLONGL               1      4B 0
D  SQLRSVDL               5     32A
D  SQLDATAL              33     48*
D  SQLTNAMELN            49     50B 0
D  SQLTNAME              51     80A
D*  End of SQLDA
```

図 15. INCLUDE SQLDA の宣言 (ILE RPG/400 の場合)

ユーザーは、SQL_NUM の定義を行わなければなりません。 SQL_NUM は、SQL_VAR に必要な次元を持つ数値定数として定義する必要があります。

RPG は配列内の構造をサポートしていないので、SQLDA は 3 つのデータ構造として生成されます。2 番目および 3 番目のデータ構造を使用すると、フィールド記述の入っている SQLDA の部分をセットアップして参照できます。

SQLDA のフィールド記述をセットするには、プログラムは SQLVAR (または SQLVAR2) のサブフィールドにフィールド記述をセットアップしてから、SQLVAR (または SQLVAR2) の MOVEA を SQL_VAR,n に対して実行します。ここで、n は SQLDA の中のフィールドの数を表しています。 この動作は、すべてのフィールド記述がセットされるまで繰り返されます。

SQLDA フィールド記述を参照するときに、ユーザーは SQL_VAR,n の MOVEA を SQLVAR (または SQLVAR2) に対して実行します。ここで、n は処理されるフィールド記述の数を表しています。

付録 D. 予約語

次の表は、現時点での DB2 UDB for iSeries の予約語のリストを示しています。新たな語が、必要に応じて追加されることがあります。 将来予約語として追加される可能性のある語のリストについては、*IBM SQL Reference Version 1* (SC26-3255) の IBM SQL および ANSI の予約語の項を参照してください。

予約語

表 77. SQL 予約語

ADD	CURRENT_TIMEZONE	GENERAL	MINUTE
ALIAS	CURRENT_USER	GENERATED	MINUTES
ALL	CURSOR	GET	MINVALUE
ALLOCATE	CYCLE	GLOBAL	MODE
ALLOW	DATABASE	GO	MODIFIES
ALTER	DAY	GOTO	MONTH
AND	DAYS	GRANT	MONTHS
ANY	DBINFO	GRAPHIC	NEW
AS	DB2GENERAL	GROUP	NEW_TABLE
AUTHORIZATION	DB2GENRL	HANDLER	NO
BEGIN	DB2SQL	HAVING	NOCACHE
BETWEEN	DECLARE	HOLD	NOCYCLE
BINARY	DEFAULT	HOUR	NODENAME
BY	DEFAULTS	HOURS	NODENUMBER
CACHE	DEFINITION	IDENTITY	NOMAXVALUE
CALL	DELETE	IF	NOMINVALUE
CALLED	DESCRIPTOR	IMMEDIATE	NOORDER
CARDINALITY	DETERMINISTIC	IN	NOT
CASE	DISALLOW	INCLUDING	NULL
CAST	DISCONNECT	INCREMENT	OF
CCSID	DISTINCT	INDEX	OLD
CHAR	DO	INDICATOR	OLD_TABLE
CHARACTER	DOUBLE	INNER	ON
CHECK	DROP	INOUT	OPEN
CLOSE	DYNAMIC	INSENSITIVE	OPTIMIZE
COLLECTION	EACH	INSERT	OPTION
COLUMN	ELSE	INTEGRITY	OR
COMMENT	ELSEIF	INTO	ORDER
COMMIT	END	IS	OUT
CONCAT	END-EXEC (COBOL のみ)	ISOLATION	OUTER
CONDITION	ESCAPE	ITERATE	OVERRIDING
CONNECT	EXCEPTION	JAVA	PACKAGE
CONNECTION	EXCLUDING	JOIN	PARAMETER
CONSTRAINT	EXECUTE	KEY	PARTITION
CONTAINS	EXISTS	LABEL	PATH
CONTINUE	EXIT	LANGUAGE	POSITION
COUNT	EXTERNAL	LEAVE	PREPARE
COUNT_BIG	FENCED	LEFT	PRIMARY
CREATE	FETCH	LIKE	PRIVILEGES
CROSS	FILE	LINKTYPE	PROCEDURE
CURRENT	FINAL	LOCK	PROGRAM
CURRENT_DATE	FOR	LONG	READ
CURRENT_PATH	FOREIGN	LOOP	READS
CURRENT_SERVER	FREE	MAXVALUE	RECOVERY
CURRENT_TIME	FROM	MICROSECOND	REFERENCES
CURRENT_TIMESTAMP	FUNCTION	MICROSECONDS	REFERENCING

表 78. SQL 予約語 (続き)

RELEASE	RRN	SQLID	UNTIL
RENAME	RUN	START	UPDATE
REPEAT	SAVEPOINT	STATIC	USAGE
RESET	SCHEMA	SUBSTRING	USER
RESIGNAL	SCRATCHPAD	SYNONYM	USING
RESTART	SECOND	TABLE	VALUES
RESULT	SECONDS	THEN	VARIABLE
RETURN	SELECT	TO	VARIANT
RETURNS	SET	TRANSACTION	VIEW
REVOKE	SIGNAL	TRIGGER	WHEN
RIGHT	SIMPLE	TRIM	WHERE
ROLLBACK	SOME	TYPE	WHILE
ROUTINE	SOURCE	UNDO	WITH
ROW	SPECIFIC	UNION	WRITE
ROWS	SQL	UNIQUE	YEAR
			YEARS

付録 E. CCSID の値

次の表は、IBM リレーショナル・データベース・プロダクトによって提供される CCSID と変換を示しています。

- DB2 UDB (OS/390 および z/OS 版) および DB2 (VM および VSE 版) の場合は、これらの表は、当初そのカタログ表で提供される CCSID と CCSID 変換の対だけを表しています。管理権限を持つユーザーは、いつでも、SBCS CCSID や SBCS 変換表を追加することができます。また、SBCS や DBCS 変換を行うユーザー出口ルーチンを用意することも可能です。
- DB2 UDB for iSeries および DB2 UDB UWO の場合、これらの表は、使用できる CCSID と変換表だけを表しています。これ以外の CCSID や変換表を追加する方法はありません。

次のリストは、以下の表の IBM リレーショナル・データベース・プロダクトの欄で使用されている記号を定義しています。

X 該当の CCSID へ、および該当の CCSID からの変換を行う変換表が存在することを示しています。

C 該当の CCSID から他の CCSID へ変換する変換表が存在することを示しています。これは、該当の CCSID が外部コード化体系であるため、その CCSID をローカル・データのタグ付けには使用できないことも意味しています (例えば、850 などのような PC データの CCSID は、DB2 UDB for iSeries のローカル・データのタグ付けには使用できません)。

T 該当の CCSID でデータのタグ付けはできますが、該当のプロダクトでは変換表は提供されないことを示しています。

この付録の説明で参照しているのは、以下に挙げるプロダクトのバージョンです。

- DB2 UDB サーバー (OS/390 版) バージョン 7
- DB2 (VM および VSE 版) バージョン 7.1
- DB2 UDB サーバー for AS/400 バージョン 5 リリース 2
- DB2 UDB サーバー (OS/2 版) バージョン 7
- DB2 UDB サーバー (AIX/6000 版) バージョン 7
- DB2 UDB サーバー (HP 版) バージョン 7
- DB2 UDB サーバー (SUN 版) バージョン 7
- DB2 UDB サーバー (NT および Windows 95 版) バージョン 7
- DB2 UDB サーバー (SCO Open Server 版) バージョン 7

表 79. 汎用文字セット (UCS-2、UTF-16 および UTF-8)

CCSID	説明	DB2 UDB UWO (OS/390)	DB2 (VM および VSE 版)	DB2 UDB UWO (AS/400)	DB2 UDB UWO (OS/2)	DB2 UDB UWO (AIX/6000)	DB2 UDB UWO (HP)	DB2 UDB UWO (SUN)	DB2 UDB UWO (NT)	DB2 UDB UWO (SCO)
1200	UTF-16	X		C	C	C	C	C	C	

表 79. 汎用文字セット (UCS-2、UTF-16 および UTF-8) (続き)

CCSID	説明	DB2 UDB UWO (OS/390)	DB2 (VM および VSE 版)	DB2 UDB UWO (AS/400)	DB2 UDB UWO (OS/2)	DB2 UDB UWO (AIX/6000)	DB2 UDB UWO (HP)	DB2 UDB UWO (SUN)	DB2 UDB UWO (NT)	DB2 UDB UWO (SCO)
1208	UTF-8 レベル 3	X		C	X	X	X	X	X	
13488	UCS-2 のレベル 1	C		X	X	X	X	X	X	

表 80. EBCDIC グループ 1 (ラテン-1) の国用の CCSID

CCSID	説明	DB2 UDB UWO (OS/390)	DB2 (VM および VSE 版)	DB2 UDB UWO (AS/400)	DB2 UDB UWO (OS/2)	DB2 UDB UWO (AIX/6000)	DB2 UDB UWO (HP)	DB2 UDB UWO (SUN)	DB2 UDB UWO (NT)	DB2 UDB UWO (SCO)
37	米国、カナダ (S/370)、オランダ、ポルトガル、ブラジル、オーストラリア、ニュージーランド	X	X	X	C	C	C	C	C	C
256	ワード・プロセッシング、オランダ	T	T	X						
273	オーストリア、ドイツ	X	X	X	C	C	C	C	C	C
277	デンマーク、ノルウェー	X	X	X	C	C	C	C	C	C
278	フィンランド、スウェーデン	X	X	X	C	C	C	C	C	C
280	イタリア	X	X	X	C	C	C	C	C	C
284	スペイン、ラテン・アメリカ (スペイン語圏)	X	X	X	C	C	C	C	C	C
285	英国	X	X	X	C	C	C	C	C	C
297	フランス	X	X	X	C	C	C	C	C	C
500	ベルギー、カナダ (AS/400)、スイス、国際ラテン-1	X	X	X	C	C	C	C	C	C
871	アイスランド	X	X	X	C	C	C	C	C	C
924	ラテン 0	T	T	X						

表 80. EBCDIC グループ 1 (ラテン-1) の国用の CCSID (続き)

CCSID	説明	DB2 UDB UWO (OS/390)	DB2 (VM および VSE 版)	DB2 UDB UWO (AS/400)	DB2 UDB UWO (OS/2)	DB2 UDB UWO (AIX/6000)	DB2 UDB UWO (HP)	DB2 UDB UWO (SUN)	DB2 UDB UWO (NT)	DB2 UDB UWO (SCO)
1140	米国、カナダ (S/370)、オランダ、ポルトガル、ブラジル、オーストラリア、ニュージーランド	T	T	X						
1141	オーストリア、ドイツ	T	T	X						
1142	デンマーク、ノルウェー	T	T	X						
1143	フィンランド、スウェーデン	T	T	X						
1144	イタリア	T	T	X						
1145	スペイン、ラテン・アメリカ (スペイン語圏)	T	T	X						
1146	英国	T	T	X						
1147	フランス	T	T	X						
1148	ベルギー、カナダ (AS/400)、スイス、国際ラテン-1	T	T	X						
1149	アイスランド	T	T	X						

表 81. PC-データおよび ISO グループ 1 (ラテン-1) の国用の CCSID

CCSID	説明	DB2 UDB UWO (OS/390)	DB2 (VM および VSE 版)	DB2 UDB UWO (AS/400)	DB2 UDB UWO (OS/2)	DB2 UDB UWO (AIX/6000)	DB2 UDB UWO (HP)	DB2 UDB UWO (SUN)	DB2 UDB UWO (NT)	DB2 UDB UWO (SCO)
437	USA	X	C	C	X	C	C	C	C	C
819	ラテン-1 の各国 (ISO 8859-1)	X	C	C	C	X	X	X	C	X
850	ラテン・アルファベット番号 1; ラテン-1 の各国	X	C	C	X	X	C	C	C	C
858	ラテン・アルファベット番号 1; ラテン-1 の各国 (ユーロ対応)	T	T	C						
860	ポルトガル (850 のサブセット)	C	C	C	X	C	C	C	C	C

表 81. PC-データおよび ISO グループ 1 (ラテン-1) の国用の CCSID (続き)

CCSID	説明	DB2 UDB UWO (OS/390)	DB2 (VM および VSE 版)	DB2 UDB UWO (AS/400)	DB2 UDB UWO (OS/2)	DB2 UDB UWO (AIX/6000)	DB2 UDB UWO (HP)	DB2 UDB UWO (SUN)	DB2 UDB UWO (NT)	DB2 UDB UWO (SCO)
861	アイスランド	C		C						
863	カナダ (850 のサブセット)	C	C	C	X	C	C	C	C	C
865	デンマーク、ノルウェー、フィンランド、スウェーデン	C	C	C						
923	ラテン 0			C						
1009	IRV 7 ビット			C						
1010	フランス 7 ビット			C						
1011	ドイツ 7 ビット			C						
1012	イタリア 7 ビット			C						
1013	英国 7 ビット			C						
1014	スペイン 7 ビット			C						
1015	ポルトガル 7 ビット			C						
1016	ノルウェー 7 ビット			C						
1017	デンマーク 7 ビット			C						
1018	フィンランドおよびスウェーデン 7 ビット			C						
1019	ベルギーおよびオランダ 7 ビット			C						
1051	HP エミュレーション	X		C	C	C	X	C	C	
1252	Windows** ラテン-1	X	C	C	C	C	C	C	X	C
1275	Macintosh** ラテン-1	X		C	C	C	C	C	C	C

表 82. EBCDIC グループ 1a (非ラテン-1 SBCS) の国用の CCSID

CCSID	説明	DB2 UDB UWO (OS/390)	DB2 (VM および VSE 版)	DB2 UDB UWO (AS/400)	DB2 UDB UWO (OS/2)	DB2 UDB UWO (AIX/6000)	DB2 UDB UWO (HP)	DB2 UDB UWO (SUN)	DB2 UDB UWO (NT)	DB2 UDB UWO (SCO)
420	アラビア語(タイプ 4)ビジュアル LTR	X	X	X	C	C	C	C	C	C
423	ギリシャ語	X	X	X	C	C	C	C	C	C
424	ヘブライ語(タイプ 4)	X	X	X	C	C	C	C	C	C
870	ラテン-2 マルチリンガル	X	X	X	C	C	C	C	C	C
875	ギリシャ語	X	X	X	C	C	C	C	C	C
880	キリル文字 マルチリンガル	T	T	X						
905	トルコ・ラテン-3 マルチリンガル	T	T	X						
918	ウルドゥー語	T	T	X						
1025	キリル文字 マルチリンガル	X	X	X	C	C	C	C	C	C
1026	トルコ・ラテン-5	X	T	X	C	C	C	C	C	C
1097	ペルシア語	T	T	X						
1112	バルト語 マルチリンガル	X	X	X	C	C	C	C	C	C
1122	エストニア語	T	X	X	C	C	C	C	C	C
1123	ウクライナ語	T	X	X	C	C	C	C	C	C
1137	デーバナーガリー文字	T	T	X						
1153	ラテン-2 (ユーロ対応)	T	T	X						
1154	キリル文字 (ユーロ対応)	T	T	X						
1155	トルコ・ラテン-5 (ユーロ対応)	T	T	X						
1156	バルト語 (ユーロ対応)	T	T	X						
1157	エストニア語 (ユーロ対応)	T	T	X						
1158	ウクライナ語 (ユーロ対応)	T	T	X						
4971	ギリシャ語 (ユーロ対応)	T	T	X						

表 82. EBCDIC グループ 1a (非ラテン-1 SBCS) の国用の CCSID (続き)

CCSID	説明	DB2 UDB UWO (OS/390)	DB2 (VM および VSE 版)	DB2 UDB UWO (AS/400)	DB2 UDB UWO (OS/2)	DB2 UDB UWO (AIX/6000)	DB2 UDB UWO (HP)	DB2 UDB UWO (SUN)	DB2 UDB UWO (NT)	DB2 UDB UWO (SCO)
8612	アラビア語(タイプ 5)			X						
8616	ヘブライ語(タイプ 6)			X						
12708	アラビア語(タイプ 7)			X						
62211	ヘブライ語(タイプ 5)			X	C	C	C	C	C	C
62224	アラビア語(タイプ 6)			X	C	C	C	C	C	C
62229	ヘブライ語(タイプ 8)				C	C	C	C	C	C
62233	アラビア語(タイプ 8)				C	C	C	C	C	C
62234	アラビア語(タイプ 9)				C	C	C	C	C	C
62235	ヘブライ語(タイプ 10)			X	C	C	C	C	C	C
62240	ヘブライ語(タイプ 11)				C	C	C	C	C	C
62245	ヘブライ語(タイプ 10)			X						

ストリング・タイプ:

4 ビジュアル / 左から右 / 形状あり

5 暗黙 / 左から右 / 形状なし

6 暗黙 / 右から左 / 形状なし

7 ビジュアル / コンテキスト / 形状なし

8 ビジュアル / 右から左 / 形状あり

9 ビジュアル / 右から左 / 形状あり

10 暗黙 / コンテキスト - 左

11 暗黙 / コンテキスト - 右

表 83. PC-データおよび ISO グループ 1a (非ラテン-1 SBCS) の国用の CCSID

CCSID	説明	DB2 UDB UWO (OS/390)	DB2 (VM および VSE 版)	DB2 UDB UWO (AS/400)	DB2 UDB UWO (OS/2)	DB2 UDB UWO (AIX/6000)	DB2 UDB UWO (HP)	DB2 UDB UWO (SUN)	DB2 UDB UWO (NT)	DB2 UDB UWO (SCO)
720	アラビア語 (MS-DOS)			C						

表 83. PC-データおよび ISO グループ 1a (非ラテン-1 SBCS) の国用の CCSID (続き)

CCSID	説明	DB2 UDB UWO (OS/390)	DB2 (VM および VSE 版)	DB2 UDB UWO (AS/400)	DB2 UDB UWO (OS/2)	DB2 UDB UWO (AIX/6000)	DB2 UDB UWO (HP)	DB2 UDB UWO (SUN)	DB2 UDB UWO (NT)	DB2 UDB UWO (SCO)
737	ギリシャ語 (MS-DOS)			C	C	C	C	C	C	C
775	バルト語 (MS-DOS)			C						
813	ギリシャ/ラテン (ISO 8859-7)	C	C	C	X	X	X	C	C	X
851	ギリシャ語			C						
852	ラテン-2 マルチリンガル	C	C	C	X	C	C	C	C	C
855	キリル文字 マルチリンガル		C	C	X	C	C	C	C	C
856	アラビア語(タイプ 5)			C						
857	トルコ・ラテン-5	C		C	X	C	C	C	C	C
862	ヘブライ語(タイプ 10)	C	C	C	X	C	C	C	C	C
864	アラビア語(タイプ 5)	C	C	C	X	C	C	C	C	C
866	キリル文字		C	C	X	C	C	C	C	C
868	ウルドゥー語			C						
869	ギリシャ語	C	C	C	X	C	C	C	C	C
878	ロシア語インターネット			C						
912	ラテン-2 (ISO 8859-2)	C	C	C	C	X	X	C	C	X
914	ラテン 4 (ISO 8859-4)			C						
915	キリル文字 マルチリンガル (ISO 8859-5)		C	C	X	X	X	C	C	X
916	ヘブライ語/ラテン (ISO 8859-8) (タイプ 5)	C	C	C	C	X	C	C	C	C
920	トルコ・ラテン-5 (ISO 8859-9)	C		C	C	X	X	C	C	X
921	バルト語 8 ビット	C		C	X	X	C	C	X	C
922	エストニア語 8 ビット			C	X	X	C	C	X	C

表 83. PC-データおよび ISO グループ 1a (非ラテン-1 SBCS) の国用の CCSID (続き)

CCSID	説明	DB2 UDB UWO (OS/390)	DB2 (VM および VSE 版)	DB2 UDB UWO (AS/400)	DB2 UDB UWO (OS/2)	DB2 UDB UWO (AIX/6000)	DB2 UDB UWO (HP)	DB2 UDB UWO (SUN)	DB2 UDB UWO (NT)	DB2 UDB UWO (SCO)
1008	アラビア語 8 ビット ISO			C						
1046	アラビア語(タイプ 5)	C		C	C	X	C	C	C	C
1089	アラビア語 (ISO 8859-6) (タイプ 5)	C		C	C	X	X	C	C	C
1098	ペルシア語			C						
1124	ウクライナ語 8 ビット ISO			C						
1125	ウクライナ語			C	X	C	C	C	C	C
1131	ベラルーシ語			C	X	C	C	C	C	C
1250	Windows ラテン-2	C	C	C	C	C	C	C	X	C
1251	Windows キリル文字	C	C	C	C	C	C	C	X	C
1253	Windows ギリシャ語	C	C	C	C	C	C	C	X	C
1254	Windows トルコ語	C		C	C	C	C	C	X	C
1255	Windows ヘブライ語(タイプ 5)	C	C	C	C	C	C	C	X	C
1256	Windows アラビア語(タイプ 5)	C	C	C	C	C	C	C	X	C
1257	Windows バルト語	C		C						
1280	Macintosh** ギリシャ語	C		C	C	C	C	C	C	C
1281	Macintosh** トルコ語	C		C	C	C	C	C	C	C
1282	Macintosh** ラテン-2	C		C	C	C	C	C	C	C
1283	Macintosh** キリル文字	C		C	C	C	C	C	C	C
4948	ラテン-2 マルチリンガル			C						
4951	キリル文字 マルチリンガル			C						
4952	ヘブライ語			C						
4953	トルコ・ラテン-5			C						

表 83. PC-データおよび ISO グループ 1a (非ラテン-1 SBCS) の国用の CCSID (続き)

CCSID	説明	DB2 UDB UWO (OS/390)	DB2 (VM および VSE 版)	DB2 UDB UWO (AS/400)	DB2 UDB UWO (OS/2)	DB2 UDB UWO (AIX/6000)	DB2 UDB UWO (HP)	DB2 UDB UWO (SUN)	DB2 UDB UWO (NT)	DB2 UDB UWO (SCO)
9056	アラビア語 (記憶域交換)			C						
4960	アラビア語			C						
4965	ギリシャ語			C						
62208	ヘブライ語(タイプ 4)				X	X	X	X	X	X
62209	ヘブライ語(タイプ 4)			C	X	C	C	C	C	C
62210	ヘブライ語/ラテン (ISO 8859-8)(タイプ 4)			C	C	X	X	C	C	C
62213	ヘブライ語(タイプ 5)			C	X	C	C	C	C	C
62215	Windows ヘブライ語(タイプ 4)			C	C	C	C	C	X	C
62218	アラビア語(タイプ 4)			C	X	C	C	C	C	C
62220	ヘブライ語(タイプ 6)				X	X	X	X	X	X
62221	ヘブライ語(タイプ 6)			C	X	C	C	C	C	C
62222	ヘブライ語/ラテン (ISO 8859-8)(タイプ 6)			C	C	X	X	C	C	C
62223	Windows ヘブライ語(タイプ 6)			C	C	C	C	C	X	C
62225	アラビア語(タイプ 6)				X	C	C	C	C	C
62226	アラビア語(タイプ 6)				C	X	C	C	C	C
62227	アラビア語 (ISO 8859-6)(タイプ 6)				C	X	X	C	C	C
62228	Windows アラビア語(タイプ 6)			C	C	C	C	C	X	C
62230	ヘブライ語(タイプ 8)				X	X	X	X	X	X
62231	ヘブライ語(タイプ 8)				X	C	C	C	C	C

表 83. PC-データおよび ISO グループ 1a (非ラテン-1 SBCS) の国用の CCSID (続き)

CCSID	説明	DB2 UDB UWO (OS/390)	DB2 (VM および VSE 版)	DB2 UDB UWO (AS/400)	DB2 UDB UWO (OS/2)	DB2 UDB UWO (AIX/6000)	DB2 UDB UWO (HP)	DB2 UDB UWO (SUN)	DB2 UDB UWO (NT)	DB2 UDB UWO (SCO)
62232	ヘブライ語/ラテン (ISO 8859-8)(タイプ 8)				C	X	X	C	C	C
62236	ヘブライ語(タイプ 10)				X	X	X	X	X	X
62238	ヘブライ語/ラテン (ISO 8859-8)(タイプ 10)			C	C	X	X	C	C	C
62239	Windows ヘブライ語(タイプ 10)			C	C	C	C	C	X	C
62241	ヘブライ語(タイプ 11)				X	X	X	X	X	X
62242	ヘブライ語(タイプ 11)				X	C	C	C	C	C
62243	ヘブライ語/ラテン (ISO 8859-8)(タイプ 11)				C	X	X	C	C	C
62244	Windows ヘブライ語(タイプ 11)				C	C	C	C	X	C

ストリング・タイプ:

4 ビジュアル / 左から右 / 形状あり

5 暗黙 / 左から右 / 形状なし

6 暗黙 / 右から左 / 形状なし

7 ビジュアル / コンテキスト / 形状なし

8 ビジュアル / 右から左 / 形状あり

9 ビジュアル / 右から左 / 形状あり

10 暗黙 / コンテキスト - 左

11 暗黙 / コンテキスト - 右

表 84. EBCDIC グループ 2 (DBCS) の国用の SBCS CCSID

CCSID	説明	DB2 UDB UWO (OS/390)	DB2 (VM および VSE 版)	DB2 UDB UWO (AS/400)	DB2 UDB UWO (OS/2)	DB2 UDB UWO (AIX/6000)	DB2 UDB UWO (HP)	DB2 UDB UWO (SUN)	DB2 UDB UWO (NT)	DB2 UDB UWO (SCO)
290	日本カタカナ (拡張)	X	X	X	C	C	C	C	C	C
833	韓国 (拡張)	X	X	X	C	C	C	C	C	C

表 84. EBCDIC グループ 2 (DBCS) の国用の SBCS CCSID (続き)

CCSID	説明	DB2 UDB UWO (OS/390)	DB2 (VM および VSE 版)	DB2 UDB UWO (AS/400)	DB2 UDB UWO (OS/2)	DB2 UDB UWO (AIX/6000)	DB2 UDB UWO (HP)	DB2 UDB UWO (SUN)	DB2 UDB UWO (NT)	DB2 UDB UWO (SCO)
836	中国語 (簡体字) (拡張)	X	X	X	C	C	C	C	C	C
838	タイ (拡張)	X	X	X	C	C	C	C	C	C
1027	日本ローマ字 (拡張)	X	X	X	C	C	C	C	C	C
1130	ベトナム	T	X	X						
1132	ラオ語	T	X	X						
1160	タイ語 (ユーロ対応)	T	T	X						
1164	ベトナム (ユーロ対応)	T	T	X						
5123	日本 (ユーロ対応)	T	T	X						
9030	タイ (拡張)	T	T	X						
13121	韓国、Windows	T	T	X						
13124	中国語 (繁体字)	T	T	X						
28709	中国語 (繁体字) (拡張)	X	X	X	C	C	C	C	C	C

表 85. PC-データ・グループ 2 (DBCS) の国用の SBCS CCSID

CCSID	説明	DB2 UDB UWO (OS/390)	DB2 (VM および VSE 版)	DB2 UDB UWO (AS/400)	DB2 UDB UWO (OS/2)	DB2 UDB UWO (AIX/6000)	DB2 UDB UWO (HP)	DB2 UDB UWO (SUN)	DB2 UDB UWO (NT)	DB2 UDB UWO (SCO)
367	韓国語および中国語 (簡体字) EUC	X	C	C		X			C	
874	タイ (拡張)	C	C	C	X	X			X	
891	韓国 (非拡張)		C	C						
895	日本 EUC - JISX201 ローマ字セット	C	C							
896	日本 EUC - JISX201 カタカナ・セット		C							
897	日本 (非拡張)	X	C	C						
903	中国語 (簡体字) (非拡張)		C	C						
904	中国語 (繁体字) (非拡張)	C	C	C						

表 85. PC-データ・グループ 2 (DBCS) の国用の SBCS CCSID (続き)

CCSID	説明	DB2 UDB UWO (OS/390)	DB2 (VM および VSE 版)	DB2 UDB UWO (AS/400)	DB2 UDB UWO (OS/2)	DB2 UDB UWO (AIX/6000)	DB2 UDB UWO (HP)	DB2 UDB UWO (SUN)	DB2 UDB UWO (NT)	DB2 UDB UWO (SCO)
1040	韓国 (拡張)		C	C						
1041	日本 (拡張)	X	C	C						
1042	中国語 (簡体字) (拡張)		C	C						
1043	中国語 (繁体字) (拡張)	C	C	C						
1088	韓国 (KS コード 5601-89)	X	C	C						
1114	中国語 (繁体字) (Big-5)	C	C	C						
1115	中国語 (簡体字) GB コード	C	C	C						
1126	韓国、Windows		C	C						
1129	ベトナム			C						
1133	ラオ語 ISO			C						
1258	ベトナム			C						
4970	タイ (拡張)			C	X	X			X	
5210	中国語 (繁体字)			C						
9066	タイ (拡張)			C						

表 86. EBCDIC グループ 2 (DBCS) の国用の DBCS CCSID

CCSID	説明	DB2 UDB UWO (OS/390)	DB2 (VM および VSE 版)	DB2 UDB UWO (AS/400)	DB2 UDB UWO (OS/2)	DB2 UDB UWO (AIX/6000)	DB2 UDB UWO (HP)	DB2 UDB UWO (SUN)	DB2 UDB UWO (NT)	DB2 UDB UWO (SCO)
300	日本 - 4370 のユーザー定義文字 (UDC) を含む	X	X	X	C	C	C	C	C	C
834	韓国 - 1880 UDC を含む	X	X	X	C	C	C	C	C	C
835	中国語 (繁体字) - 6204 UDC を含む	X	X	X	C	C	C	C	C	C
837	中国語 (簡体字) - 1880 UDC を含む	X	X	X	C	C	C	C	C	C
4396	日本 - 1880 UDC を含む	X	X	X	C	C	C	C	C	C
4930	韓国、Windows			X	C	C	C	C	C	C
4933	中国語 (簡体字)			X	C	C	C	C	C	C

表 87. PC-データ・グループ 2 (DBCS) の国用の DBCS CCSID

CCSID	説明	DB2 UDB UWO (OS/390)	DB2 (VM および VSE 版)	DB2 UDB UWO (AS/400)	DB2 UDB UWO (OS/2)	DB2 UDB UWO (AIX/6000)	DB2 UDB UWO (HP)	DB2 UDB UWO (SUN)	DB2 UDB UWO (NT)	DB2 UDB UWO (SCO)
301	日本 - 1880 UDC を含む	X	C	C	X	X	C	C	C	C
926	韓国 - 1880 UDC を含む		C	C						
927	中国語 (繁体字) - 6204 UDC を含む	C	C	C	X	C	C	C	C	C
928	中国語 (簡体字) - 1880 UDC を含む		C	C						
941	日本、Windows	X	C	C	C	C	C	C	X	C
947	中国語 (繁体字) (Big-5)	C	C	C	X	X	C	C	X	C
951	韓国 (KS コード 5601-89) - 1880 UDC を含む	X	C	C	X	C	C	C	X	C
952	日本 (EUC) X208-1990 セット		C							
953	日本 (EUC) X212-1990 セット		C							
971	韓国 (EUC) - 188 UDC を含む	X	C	C	C	X	X	X	C	C
1351	日本 HP-UX (J15)	X		C	C	C	X	C	C	C
1362	韓国、Windows		C	C	C	C	C	C	X	C
1380	中国語 (簡体字) (GB コード) - 1880 UDC を含む	C	C	C	X	C	C	C	X	X
1382	中国語 (簡体字) (EUC) - 1360 UDC を含む	C	C	C	C	X	X	X	C	X
1385	中国語 (繁体字)			C	C	C	C	C	X	C

CCSID の値

表 88. EBCDIC グループ 2 (DBCS) の国用の混合 CCSID

CCSID	説明	DB2 UDB UWO (OS/390)	DB2 (VM および VSE 版)	DB2 UDB UWO (AS/400)	DB2 UDB UWO (OS/2)	DB2 UDB UWO (AIX/6000)	DB2 UDB UWO (HP)	DB2 UDB UWO (SUN)	DB2 UDB UWO (NT)	DB2 UDB UWO (SCO)
930	日本カタカナ/漢字 (拡張) - 4370 UDC を含む	X	X	X	C	C	C	C	C	C
933	韓国 (拡張) - 1880 UDC を含む	X	X	X	C	C	C	C	C	C
935	中国語 (簡体字) (拡張) - 1880 UDC を含む	X	X	X	C	C	C	C	C	C
937	中国語 (繁体字) (拡張) - 4370 UDC を含む	X	X	X	C	C	C	C	C	C
939	日本ローマ字/漢字 (拡張) - 4370 UDC を含む	X	X	X	C	C	C	C	C	C
1364	韓国 (拡張)			X	C	C	C	C	C	C
1388	中国語 (簡体字)			X	C	C	C	C	C	C
5026	日本カタカナ/漢字 (拡張) - 1880 UDC を含む	X	X	X	C	C	C	C	C	C
5035	日本ローマ字/漢字 (拡張) - 1880 UDC を含む	X	X	X	C	C	C	C	C	C

表 89. PC-データ・グループ 2 (DBCS) の国用の混合 CCSID

CCSID	説明	DB2 UDB UWO (OS/390)	DB2 (VM および VSE 版)	DB2 UDB UWO (AS/400)	DB2 UDB UWO (OS/2)	DB2 UDB UWO (AIX/6000)	DB2 UDB UWO (HP)	DB2 UDB UWO (SUN)	DB2 UDB UWO (NT)	DB2 UDB UWO (SCO)
932	日本 (非拡張) - 1880 UDC を含む	X	C	C	X	X	C	C	C	C
934	韓国 (非拡張) - 1880 UDC を含む		C	C						
936	中国語 (簡体字) (非拡張) - 1880 UDC を含む		C	C						
938	中国語 (繁体字) (非拡張) - 6204 UDC を含む	C	C	C	X	C	C	C	C	C

表 89. PC-データ・グループ 2 (DBCS) の国用の混合 CCSID (続き)

CCSID	説明	DB2 UDB UWO (OS/390)	DB2 (VM および VSE 版)	DB2 UDB UWO (AS/400)	DB2 UDB UWO (OS/2)	DB2 UDB UWO (AIX/6000)	DB2 UDB UWO (HP)	DB2 UDB UWO (SUN)	DB2 UDB UWO (NT)	DB2 UDB UWO (SCO)
942	日本 (拡張) - 1880 UDC を含む	X	C	C	X	C	C	C	C	C
943	日本 NT	X	C	C	X	C	C	C	X	C
944	韓国 (拡張) - 1880 UDC を含む	C	C	C						
946	中国語 (簡体字) (拡張) - 1880 UDC を含む		C	C						
948	中国語 (繁体字) (拡張) - 6204 UDC を含む	C	C	C	X	C	C	C	C	C
949	韓国 (KS コード 5601-89) - 1880 UDC を含む	X	C	C	X	C	C	C	C	C
950	中国語 (繁体字) (Big-5)	C	C	C	X	X	X	X	X	C
954	日本 (EUC)	C	C	C	C	X	X	X	C	X
956	日本 2022 TCP			C						
957	日本 2022 TCP			C						
958	日本 2022 TCP			C						
959	日本 2022 TCP			C						
964	中国語 (繁体字) (EUC)	C	C	C	C	X	X	X	C	C
965	中国語 (繁体字) 2022 TCP			C						
970	韓国 EUC	X	C	C	C	X	X	X	C	C
1363	韓国、Windows		C	C	C	C	C	C	X	C
1381	中国語 (簡体字) GB コード	C	C	C	X	C	C	C	X	C
1383	中国語 (簡体字) EUC	C	C	C	C	X	X	X	C	X
1386	中国語 (簡体字)			C	X	X	C	C	X	C
1392	中国語 (簡体字) GB18030			C						
5039	日本 HP-UX (J15)	C			C	C	X	C	C	C
5050	日本 (EUC)			C						
5052	日本 2022 TCP			C						

表 89. PC-データ・グループ 2 (DBCS) の国用の混合 CCSID (続き)

CCSID	説明	DB2 UDB UWO (OS/390)	DB2 (VM および VSE 版)	DB2 UDB UWO (AS/400)	DB2 UDB UWO (OS/2)	DB2 UDB UWO (AIX/6000)	DB2 UDB UWO (HP)	DB2 UDB UWO (SUN)	DB2 UDB UWO (NT)	DB2 UDB UWO (SCO)
5053	日本 2022 TCP			C						
5054	日本 2022 TCP			C						
5055	日本 2022 TCP			C						
17354	韓国 2022 TCP			C						
25546	韓国 2022 TCP			C						
33722	日本 EUC			C						

付録 F. SQL ステートメントの特性

この付録では、SQL ステートメントの特性に関する情報を、それらのステートメントが使用されるさまざまな場所と関連付けて示します。

- SQL ステートメントで許されるアクションの表では、SQL ステートメントが実行可能かどうか、対話式または動的に準備できるかどうか、および、そのステートメントを処理するのがリクエスター、サーバー、プリコンパイラーのいずれであるかを示します。 『SQL ステートメントで許されるアクション』を参照してください。
- データ・アクセス指示表には、ルーチン内で SQL ステートメントを使用する場合に指定しなければならない SQL データ・アクセスのレベルを示してあります。 915 ページの『ルーチン内での SQL ステートメントのデータ・アクセス指示』 を参照してください。
- 分散リレーショナル・データベースの使用に関する考慮事項には、アプリケーション・サーバーがアプリケーション・リクエスターと異なる場合の SQL ステートメントの使用に関する情報を示します。 917 ページの『分散リレーショナル・データベースの使用に関する考慮事項』を参照してください。

SQL ステートメントで許されるアクション

表 90 は、特定の DB2 ステートメントが実行可能かどうか、対話式または動的に準備できるかどうか、および、そのステートメントを処理するのがリクエスター、サーバー、プリコンパイラーのいずれであるかを示しています。文字 **Y** は、*yes* (可能/該当) を意味します。

表 90. SQL ステートメントで許されるアクション

			処理の主体		
SQL ステートメント	**実行可能**	**対話式または動的な準備**	**要求元システム**	**サーバー**	**プリコンパイラー**
ALTER	**Y**	**Y**		**Y**	
BEGIN DECLARE SECTION					**Y**
CALL	**Y**	**Y**		**Y**	
CLOSE	**Y**			**Y**	
COMMENT	**Y**	**Y**		**Y**	
COMMIT	**Y**	**Y**		**Y**	
CONNECT (タイプ 1 およびタイプ 2)	**Y**		**Y**		
CREATE ...	**Y**	**Y**		**Y**	
DECLARE CURSOR					**Y**
DECLARE GLOBAL TEMPORARY TABLE	**Y**	**Y**		**Y**	
DECLARE PROCEDURE					**Y**
DECLARE STATEMENT					**Y**

表 90. SQL ステートメントで許されるアクション (続き)

SQL ステートメント	実行可能	対話式または動的な準備	処理の主体		
			要求元システム	サーバー	プリコンパイラー
DECLARE VARIABLE					Y
DELETE	Y	Y		Y	
DESCRIBE	Y			Y	
DESCRIBE TABLE	Y			Y	
DISCONNECT	Y		Y		
DROP ...	Y	Y		Y	
END DECLARE SECTION					Y
EXECUTE	Y			Y	
EXECUTE IMMEDIATE	Y			Y	
FETCH	Y			Y	
FREE LOCATOR	Y	Y		Y	
GET DIAGNOSTICS[2]	Y			Y	
GRANT ...	Y	Y		Y	
HOLD LOCATOR	Y	Y		Y	
INCLUDE					Y
INSERT	Y	Y		Y	
LABEL	Y	Y		Y	
LOCK TABLE	Y	Y		Y	
OPEN	Y			Y	
PREPARE	Y			Y	
RELEASE CONNECTION	Y		Y		
RELEASE SAVEPOINT	Y	Y		Y	
RENAME	Y	Y		Y	
REVOKE ...	Y	Y		Y	
ROLLBACK	Y	Y		Y	
SAVEPOINT	Y	Y		Y	
SELECT INTO	Y			Y	
SET CONNECTION	Y		Y		
SET OPTION					Y
SET PATH	Y	Y		Y	
SET RESULT SETS[3]	Y			Y	
SET SCHEMA	Y	Y		Y	
SET TRANSACTION	Y	Y		Y	
SET 変数	Y		Y		
SIGNAL SQLSTATE[2]	Y			Y	
UPDATE	Y	Y		Y	
VALUES[1]	Y			Y	
VALUES INTO	Y	Y		Y	

表 90. SQL ステートメントで許されるアクション (続き)

SQL ステートメント	実行可能	対話式または動的な準備	処理の主体: 要求元システム	処理の主体: サーバー	処理の主体: プリコンパイラー
WHENEVER					Y

注 :

1. このステートメントは、トリガーのトリガー・アクション内でのみ使用できます。
2. このステートメントは、SQL 関数、SQL プロシージャー、または SQL トリガー内でのみ使用できます。
3. このステートメントは、プロシージャー内でのみ使用できます。

ルーチン内での SQL ステートメントのデータ・アクセス指示

次の表は、SQL ステートメント (最初の欄に示されているもの) が、SQL データ・アクセス指示で指定する関数またはプロシージャー内で実行できるかどうかを示しています。 NO SQL と定義された関数またはプロシージャーの中で、実行可能 SQL ステートメントが検出された場合は、SQLSTATE 38001 が戻されます。その他の実行コンテキストの場合は、そのコンテキストでサポートされない SQL ステートメントがあれば、すべて SQLSTATE 38003 が戻されます。 CONTAINS SQL コンテキスト内での使用を許可されていないその他の SQL ステートメントの場合は、SQLSTATE 38004 が戻され、READS SQL DATA コンテキストの場合は SQLSTATE 38002 が戻されます。 SQL 関数または SQL プロシージャーの作成中は、SQL データ・アクセス指示に一致しないステートメントがあると、SQLSTATE 42895 が戻されます。

表 91. 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 TABLE	N	N	N	Y
BEGIN DECLARE SECTION	Y[1]	Y	Y	Y
CALL	N	Y	Y	Y
CLOSE	N	N	Y	Y
COMMENT	N	N	N	Y
COMMIT	N	N	N	N
CONNECT (タイプ 1 およびタイプ 2)[3]	N	N	N	N
CREATE ...	N	N	N	Y
DECLARE CURSOR	Y[1]	Y	Y	Y
DECLARE GLOBAL TEMPORARY TABLE	N	N	N	Y
DECLARE PROCEDURE	Y[1]	Y	Y	Y
DECLARE STATEMENT	Y[1]	Y	Y	Y
DECLARE VARIABLE	Y[1]	Y	Y	Y
DELETE	N	N	N	Y
DESCRIBE	N	N	Y	Y

表 91. 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 TABLE	N	N	Y	Y
DISCONNECT[3]	N	N	N	N
DROP ...	N	N	N	Y
END DECLARE SECTION	Y[1]	Y	Y	Y
EXECUTE	N	Y[2]	Y[2]	Y
EXECUTE IMMEDIATE	N	Y[2]	Y[2]	Y
FETCH	N	N	Y	Y
FREE LOCATOR	N	Y	Y	Y
GRANT ...	N	N	N	Y
HOLD LOCATOR	N	Y	Y	Y
INCLUDE	Y[1]	Y	Y	Y
INSERT	N	N	N	Y
LABEL	N	N	N	Y
LOCK TABLE	N	Y	Y	Y
OPEN	N	N	Y	Y
PREPARE	N	Y	Y	Y
RELEASE CONNECTION[3]	N	N	N	N
RELEASE SAVEPOINT	N	N	N	Y
RENAME	N	N	N	Y
REVOKE ...	N	N	N	Y
ROLLBACK	N	Y	Y	Y
ROLLBACK TO SAVEPOINT	N	N	N	Y
SAVEPOINT	N	N	N	Y
SELECT INTO	N	N	Y	Y
SET CONNECTION[3]	N	N	N	N
SET OPTION	Y[1]	Y	Y	Y
SET PATH	N	N	Y	Y
SET RESULT SETS	N	Y	Y	Y
SET SCHEMA	N	N	Y	Y
SET TRANSACTION	N	Y	Y	Y
SET 変数	N	Y	Y	Y
UPDATE	N	N	N	Y
VALUES	N	Y	Y	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に示しますが、非実行ステートメントを制限するものではありません。

2. 実行されるステートメントによって決まります。 EXECUTE ステートメントとして指定するステートメントは、そのとき有効な特定の SQL アクセス・レベルのコンテキストの中で許されるステートメントである必要があります。例えば、有効な SQL アクセス・レベルが READS SQL DATA の場合、ステートメントは、INSERT、UPDATE、または DELETE 以外でなければなりません。
3. 接続管理ステートメントは、ストアード・プロシージャー実行コンテキストでは指定できません。

分散リレーショナル・データベースの使用に関する考慮事項

以下に示す表には、アプリケーション・リクエスターとは異なるプロダクトのサーバーを使用するアプリケーションを開発するときに役立つ情報を収めてあります。

IBM のリレーショナル・データベースの製品はすべて、IBM SQL の拡張機能をサポートしています。このような拡張機能には、製品に特有の機能や複数の製品に共通する機能があります。

これらの大部分では、ステートメントおよび文節の一部をサポートしていないデータベース・マネージャーのアプリケーション・リクエスターを介して、アプリケーションが実行されている場合でも、現行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および文節であれば、そのアプリケーションで使用することができます。この一般的規則に対する制約事項は、 918 ページの表 92、 920 ページの表 93、 922 ページの表 94、および 924 ページの表 95 に示してあります。

表の中の 'R' は、この SQL 機能が指定された環境でサポートされていないことを示しています。同じ行のすべての欄に 'R' があるのは、その機能を使用できるのが、現行サーバーとリクエスターが同じプロダクトである場合だけに限られることを意味しています。

次の表の DB2 UDB UWO は、DB2 UDB (OS/390 および z/OS 版)、DB2 (VM および VSE 版)、または DB2 UDB for iSeries 以外のすべての DB2 UDB プロダクトを示します。

この付録の説明で参照しているのは、以下に挙げるプロダクトのバージョンです。

- DB2 UDB (OS/390 および z/OS 版) バージョン 7
- DB2 (VM および VSE 版) バージョン 7.1
- DB2 UDB for iSeries バージョン 5 リリース 2
- DB2 UDB UWO サーバー (OS/2 版) バージョン 7
- DB2 UDB UWO サーバー (AIX/6000 版) バージョン 7
- DB2 UDB UWO サーバー (HP 版) バージョン 7
- DB2 UDB UWO サーバー (NT 版) バージョン 7
- DB2 UDB UWO サーバー (SUN 版) バージョン 7
- DB2 UDB UWO サーバー (SCO Open Server 版) バージョン 7

表 92. DB2 UDB (OS/390 および z/OS 版) アプリケーション・リクエスター

SQL のステートメントまたは関数	DB2 UDB (OS/390 および z/OS 版) サーバー	DB2 (VM および VSE 版) サーバー	DB2 UDB for iSeries サーバー	DB2 UDB UWO サーバー
結果セットを含む CALL				R
COMMIT HOLD	R	R	R	R
COMMIT RELEASE	R	R	R	R
CONNECT (タイプ 2)		注 1		注 1
DECLARE CURSOR WITH HOLD		R		
DECLARE STATEMENT				
DECLARE TABLE				
DECLARE VARIABLE				
存在の据え置き (注 3)				R
DESCRIBE TABLE		R		R
DESCRIBE、USING 文節付き				R
DISCONNECT	R	R	R	R
拡張動的ステートメント	R	R	R	R
ホスト構造				
ホスト変数 - コロンは任意指定		R	R	R
ラージ・オブジェクト (LOB) データ・タイプ		R		
DATALINK データ・タイプ	R	R	R	R
特殊データ・タイプ		R		R
ROWID データ・タイプ		R		R
非 IBM SQL ホスト宣言		注 2	注 2	注 2
PREPARE、INTO 文節付き				
PREPARE、USING 文節付き				R
PUT	R	R	R	R
RELEASE				
ROLLBACK HOLD	R	R	R	R
ROLLBACK RELEASE	R	R	R	R
SET CONNECTION				
SET CURRENT PACKAGESET				
SET ホスト変数		R	R	R
SET TRANSACTION	R	R	R	R
スクロール可能カーソル・ステートメント	R	R	R	R
UPDATE カーソル - FOR UPDATE OF 文節が指定されていない				
WHENEVER、STOP 付き	R	R	R	R

表 92. DB2 UDB (OS/390 および z/OS 版) アプリケーション・リクエスター (続き)

SQL のステートメントまたは関数	DB2 UDB (OS/390 および z/OS 版) サーバー	DB2 (VM および VSE 版) サーバー	DB2 UDB for iSeries サーバー	DB2 UDB UWO サーバー

注:

1 他のすべての接続がすでに読み取り専用である場合を除き、サーバーは、読み取り専用の操作だけが許されます。サーバーが DB2 (VM および VSE 版) または DB2 UDB UWO の場合、これが、TCP/IP が使用される場合の唯一の制限です。

2 このステートメントは、アプリケーション・リクエスターがこのステートメントを理解できる場合にサポートされます。

3 オブジェクトは、データ操作ステートメントで参照されるバインド時には存在している必要はありません。

表 93. DB2 (VM および VSE 版) アプリケーション・リクエスター

SQL のステートメントまたは関数	DB2 UDB (OS/390 および z/OS 版) サーバー	DB2 (VM および VSE 版) サーバー	DB2 UDB for iSeries サーバー	DB2 UDB UWO サーバー
結果セットを含む CALL				R
COMMIT HOLD	R	R	R	R
COMMIT RELEASE				
CONNECT (タイプ 2)	R	R	R	R
DECLARE CURSOR WITH HOLD		R		
DECLARE STATEMENT	R	R	R	R
DECLARE TABLE	R	R	R	R
DECLARE VARIABLE	R	R	R	R
存在の据え置き (注 2)			R	
DESCRIBE TABLE	R	R	R	R
DESCRIBE、USING 文節付き			R	
DISCONNECT	R	R	R	R
拡張動的ステートメント	R 注 3		R 注 3	R 注 3
ホスト変数 - コロンは任意指定	R	R	R	R
ホスト構造				
ラージ・オブジェクト (LOB) データ・タイプ	R	R	R	R
DATALINK データ・タイプ	R	R	R	R
特殊データ・タイプ	R	R	R	R
ROWID データ・タイプ	R	R	R	R
非 IBM SQL ホスト宣言	注 1		注 1	注 1
PREPARE、INTO 文節付き	R	R	R	R
PREPARE、USING 文節付き	R	R	R	R
PUT				
RELEASE	R	R	R	R
ROLLBACK HOLD	R	R	R	R
ROLLBACK RELEASE				
SET CONNECTION	R	R	R	R
SET CURRENT PACKAGESET	R	R	R	R
SET ホスト変数	R	R	R	R
SET TRANSACTION	R	R	R	R
スクロール可能カーソル・ステートメント	R	R	R	R
UPDATE カーソル - FOR UPDATE OF 文節が指定されていない				
WHENEVER、STOP 付き				

表 93. DB2 (VM および VSE 版) アプリケーション・リクエスター (続き)

SQL のステートメントまたは関数	DB2 UDB (OS/390 および z/OS 版) サーバー	DB2 (VM および VSE 版) サーバー	DB2 UDB for iSeries サーバー	DB2 UDB UWO サーバー

注:

1 このステートメントは、アプリケーション・リクエスターがこのステートメントを理解できる場合にサポートされます。

2 オブジェクトは、データ操作ステートメントで参照されるバインド時には存在している必要はありません。

3 変更できないパッケージに関する拡張動的ステートメントは、大部分が使用できます。詳細については、DB2 (VM および VSE 版) のプロダクトの資料を参照してください。

表 94. DB2 UDB for iSeries アプリケーション・リクエスター

SQL のステートメントまたは関数	DB2 UDB (OS/390 および z/OS 版) サーバー	DB2 (VM および VSE 版) サーバー	DB2 UDB for iSeries サーバー	DB2 UDB UWO サーバー
結果セットを含む CALL	R	R		R
COMMIT HOLD	R	R		R
COMMIT RELEASE	R	R	R	R
CONNECT (タイプ 2)		注 1		注 1
DECLARE CURSOR WITH HOLD		R		
DECLARE PROCEDURE				
DECLARE STATEMENT				
DECLARE TABLE				
DECLARE VARIABLE				
存在の据え置き (注 3)				R
DESCRIBE TABLE		R		R
DESCRIBE、USING 文節付き				R
DISCONNECT				
拡張動的ステートメント	R	R	R	R
ホスト変数 - コロンは任意指定	R	R	R	R
ホスト構造				
ラージ・オブジェクト (LOB) データ・タイプ		R		R
DATALINK データ・タイプ	R	R		R
特殊データ・タイプ		R		R
ROWID データ・タイプ		R		R
非 IBM SQL ホスト宣言	注 2	注 2		注 2
PREPARE、INTO 文節付き				
PREPARE、USING 文節付き				R
PUT	R	R	R	R
RELEASE				
ROLLBACK HOLD	R	R		R
ROLLBACK RELEASE	R	R	R	R
SET CONNECTION				
SET CURRENT PACKAGESET	R	R	R	R
SET ホスト変数	R	R	R	R
SET TRANSACTION	R	R		R
スクロール可能カーソル・ステートメント	R	R		R
UPDATE カーソル - FOR UPDATE OF 文節が指定されていない	R	R		
WHENEVER、STOP 付き	R	R	R	R

表 94. DB2 UDB for iSeries アプリケーション・リクエスター (続き)

SQL のステートメントまたは関数	DB2 UDB (OS/390 および z/OS 版) サーバー	DB2 (VM および VSE 版) サーバー	DB2 UDB for iSeries サーバー	DB2 UDB UWO サーバー

注:

1 他のすべての接続がすでに読み取り専用である場合を除き、サーバーは、読み取り専用の操作だけが許されます。サーバーが DB2 (VM および VSE 版) または DB2 UDB UWO の場合、これが、TCP/IP が使用される場合の唯一の制限です。

2 このステートメントは、アプリケーション・リクエスターがこのステートメントを理解できる場合にサポートされます。

3 オブジェクトは、データ操作ステートメントで参照されるバインド時には存在している必要はありません。

表 95. DB2 UDB UWO アプリケーション・リクエスター

SQL のステートメントまたは関数	DB2 UDB (OS/390 および z/OS 版) サーバー	DB2 (VM および VSE 版) サーバー	DB2 UDB for iSeries サーバー	DB2 UDB UWO サーバー
結果セットを含む CALL				
COMMIT HOLD	R	R	R	R
COMMIT RELEASE	R	R	R	R
CONNECT (タイプ 2)		注 1		注 1
DECLARE CURSOR WITH HOLD		R		
DECLARE STATEMENT	R	R	R	R
DECLARE TABLE	R	R	R	R
DECLARE VARIABLE	R	R	R	R
存在の据え置き (注 3)				R
DESCRIBE TABLE	R	R	R	R
DESCRIBE、USING 文節付き	R	R	R	R
DISCONNECT				
拡張動的ステートメント	R	R	R	R
ホスト変数 - コロンは任意指定	R	R	R	R
ホスト構造	注 4	注 4	注 4	注 4
ラージ・オブジェクト (LOB) データ・タイプ		R		
DATALINK データ・タイプ	R	R	R	R
特殊データ・タイプ		R		
ROWID データ・タイプ	R	R	R	R
非 IBM SQL ホスト宣言	注 2	注 2	注 2	
PREPARE、INTO 文節付き				
PREPARE、USING 文節付き	R	R	R	R
PUT	R	R	R	R
RELEASE				
ROLLBACK HOLD	R	R	R	R
ROLLBACK RELEASE	R	R	R	R
SET CONNECTION				
SET CURRENT PACKAGESET				
SET ホスト変数	R	R	R	R
SET TRANSACTION	R	R	R	R
スクロール可能カーソル・ステートメント	R	R	R	R
UPDATE カーソル - FOR UPDATE OF 文節が指定されていない	R	R		
WHENEVER、STOP 付き	R	R	R	R

表 95. DB2 UDB UWO アプリケーション・リクエスター (続き)

SQL のステートメントまたは関数	DB2 UDB (OS/390 および z/OS 版) サーバー	DB2 (VM および VSE 版) サーバー	DB2 UDB for iSeries サーバー	DB2 UDB UWO サーバー

注:

1 他のすべての接続がすでに読み取り専用である場合を除き、サーバーは、読み取り専用の操作だけが許されます。サーバーが DB2 (VM および VSE 版)の場合、これが、TCP/IP が使用される場合の唯一の制限です。

2 このステートメントは、アプリケーション・リクエスターがこのステートメントを理解できる場合にサポートされます。

3 オブジェクトは、データ操作ステートメントで参照されるバインド時には存在している必要はありません。

4 DB2 UDB UWO ホスト構造は、COBOL でのみサポートされます。

CONNECT (タイプ 1) と CONNECT (タイプ 2) の相違点

CONNECT ステートメントには 2 つのタイプがあります。それらは、構文は同じですが、意味が異なります。

- CONNECT (タイプ 1) は、リモート作業単位に対して使用されます。 29 ページの『リモート作業単位』を参照してください。
- CONNECT (タイプ 2) は、分散作業単位に対して使用されます。 431 ページの『CONNECT (タイプ 2)』を参照してください。

次の表は、CONNECT (タイプ 1) と CONNECT (タイプ 2) の規則の相違点を要約しています。

表 96. CONNECT (タイプ 1) と CONNECT (タイプ 2) の相違点

タイプ 1 の規則	タイプ 2 の規則
CONNECT ステートメントは、活動化グループが接続可能状態である場合にのみ実行が可能です。同じ作業単位内では、CONNECT ステートメントを複数実行することはできません。	接続可能状態に関する規則はありません。同じ作業単位内で複数の CONNECT ステートメントを実行することができます。
該当のサーバー名がローカル・ディレクトリーにリストされていないことによって、CONNECT ステートメントが失敗した場合、その活動化グループの接続状態は変わりません。 該当の活動化グループが接続可能状態でないことによって、CONNECT ステートメントが失敗した場合、その活動化グループの SQL 接続状態は変わりません。 上記以外の理由で CONNECT ステートメントが失敗した場合、その活動化グループは未接続状態になります。	CONNECT ステートメントが失敗すると、現行 SQL 接続は変わらず、それ以後の SQL ステートメントはいずれも現行サーバーによって実行されます。
CONNECT は、その活動化グループの既存の接続をすべて終了させます。 したがって、CONNECT はまた、その活動化グループのオープン・カーソルをいずれもクローズします。	CONNECT は、接続の終了やカーソルのクローズを行いません。
現行サーバーに対する CONNECT は、該当のアプリケーション・グループが接続可能状態であれば、正常に行われます。	該当の活動化グループの既存の SQL 接続に対する CONNECT は、エラーになります。したがって、現行サーバーに対する CONNECT はエラーになります。

適用される CONNECT の規則の判別

プログラムによって行われる CONNECT のタイプの指定には、プログラム準備オプションが使用されます。 プログラム準備オプションは、CRTSQLxxx コマンドの RDBCNNMTH パラメーターを使用して指定します。

リモート作業単位だけをサポートするサーバーへの接続

リモートの作業単位だけをサポートするサーバーに対する CONNECT (タイプ 2) の接続は、読み取り専用の接続になる場合があります。

リモートの作業単位だけをサポートするサーバーに対して、CONNECT (タイプ 2) が行われた場合[81]

- その接続の時点で更新を許す休止状態の接続が存在する場合、その接続では読み取りだけの操作が可能です。この場合、その接続では更新は許されません。
- これ以外の場合、その接続で更新が可能です。

分散作業単位をサポートするサーバーに対して、CONNECT (タイプ 2) が行われた場合

- リモートの作業単位だけをサポートするサーバーに対して更新を許す休止状態の接続がある場合、その接続では読み取りだけの操作が可能です。 この場合、その休止状態の接続が終了するとただちにその接続での更新が可能になります。
- これ以外の場合、その接続で更新が可能です。

81. 固有 TCP/IP の初期 DRDA サポートを使用する DB2 UDB for iSeriesは、リモート作業単位のみをサポートするサーバーの例です。

付録 G. DB2 UDB for iSeries のカタログ・ビュー

この付録では、DB2 UDB for iSeries のカタログに入っているビューについて説明します。データベース・マネージャーは、それぞれのリレーショナル・データベース中のデータに関する情報が入っている一組の表を維持管理しています。これらの表をまとめて*カタログ* と呼びます。 *カタログ表* には、DB2 UDB for iSeries によってサポートされている、表、ユーザー定義関数、特殊タイプ、パラメーター、プロシージャー、パッケージ、ビュー、索引、別名、制約、トリガー、および言語が含まれています。カタログには、このシステムからアクセス可能なすべてのリレーショナル・データベースに関する情報も含まれています。

カタログ・ビューには次の 3 つのクラスがあります。

- iSeries のカタログ表およびカタログ・ビュー

  iSeries のカタログ表およびカタログ・ビューは、ANS および ISO のカタログ・ビューをモデルにしていますが、ANS および ISO のカタログ・ビューとまったく同じというわけではありません。 iSeries のカタログ表およびビューは、DB2 UDB for iSeries の旧リリースと互換性があります。

  これらの表およびビューは、スキーマ QSYS および QSYS2 の中に入っています。

  カタログ表およびビューには、リレーショナル・データベース全体にわたるすべての表、パラメーター、プロシージャー、関数、特殊タイプ、パッケージ、ビュー、索引、トリガー、および制約が含まれています。 SQL スキーマの作成時に、そのスキーマにある表、パッケージ、ビュー、索引、および制約に関する情報のみが含まれているビューの追加セット (SYSPARMS、 SYSPROCS、 SYSFUNCS、 SYSROUTINES、 SYSROUTINEDEP、および SYSTYPES を除く) が作成され、スキーマに組み込まれます。

- ODBC および JDBC のカタログ・ビュー

  ODBC および JDBC のカタログ・ビューは、ODBC および JDBC のメタデータ API 要求を満たすように設計されています (例えば、SQLColumns)。これらのビューは、DB2 UDB (OS/390 および z/OS 版) および DB2 UDB UWO バージョン 8 のビューと互換性があります。また、これらのビューは、ODBC または JDBC がそのメタデータ API を拡張または変更すると、それに応じて変更されます。

  これらのビューはスキーマ SYSIBM の中に入っています。

- ANS および ISO のカタログ・ビュー

  ANS および ISO のカタログ・ビューは、ANS および ISO の SQL 標準 (情報スキーマ (Information Schema) カタログ・ビュー) に準拠するよう設計されています。これらのビューは、DB2 UDB UWO バージョン 8 のビューと互換性があります。また、これらのビューは、ANS および ISO 標準が拡張または変更されると、それに応じて変更されます。

  これらのビューはスキーマ QSYS2 および SYSIBM の中に入っています。

  ビューには、将来標準が拡張された場合に備えて予約されている列がいくつか含まれています。

注: これらのビューのいくつかでは、ビュー定義の一部として、特殊なカタログ関数を使用します。これらの関数は、SYSIBM の中に入っていますが、アプリケーションで直接使用してはなりません。これらの関数は特定の独立補助記憶域プール (IASP) 用に作成されているもので、将来のリリースで変更されることもあります。

使用上の注意

カタログ内の名前: 一般に、カタログ表の列に格納されるすべての名前は、区切り文字なしで、大文字小文字の区別があります。例えば、次の表が作成されたとします。

```
CREATE TABLE "colname"/"long_table_name"
                        ("long_column_name" CHAR(10),
                         INTCOL INTEGER)
```

SQL 名とシステム名の間のマッピングに関する情報を戻すには、次の選択ステートメントを使用できます。

```
SELECT TABLE_NAME, SYSTEM_TABLE_NAME, COLUMN_NAME, SYSTEM_COLUMN_NAME
    FROM QSYS2/SYSCOLUMNS
    WHERE TABLE_NAME = 'long_table_name' AND
          TABLE_SCHEMA = 'colname'
```

次の行が戻されます。

TABLE_NAME	SYSTEM_TABLE_NAME	COLUMN_NAME	SYSTEM_COLUMN_NAME
long_table_name	″long0001″	long_column_name	LONG_00001
long_table_name	″long0001″	INTCOL	INTCOL

カタログ内のシステム名 : 通常は、短いシステム列名より、長い SQL 列名を使用してください。 iSeries カタログ表およびビュー用の短いシステム列名は、旧リリースおよび他の DB2 UDB プロダクトとの互換性を確保するために明示指定できるように、保持されているものです。 ODBC と JDBC のカタログ・ビュー、および ANS と ISO のカタログ・ビュー用の短いシステム列名は、明示的には維持されず、リリース間で変わる可能性があります。

カタログ内のヌル値: 列の情報が適用されない場合は、ヌル値が戻されます。上記で作成された表を使用すると、次の選択ステートメントで NUMERIC_SCALE および CHARACTER_MAXIMUM_LENGTH を照会し、データが列のデータ・タイプに適用されなかった場合は、ヌル値が戻されます。

```
SELECT COLUMN_NAME, NUMERIC_SCALE, CHARACTER_MAXIMUM_LENGTH
  FROM QSYS2/SYSCOLUMNS
  WHERE TABLE_NAME = 'long_table_name'    AND
        TABLE_SCHEMA = 'colname'
```

次の行が戻されます。

COLUMN_NAME	NUMERIC_SCALE	CHARACTER_MAXIMUM_LENGTH
long_column_name	?	10
INTCOL	0	?

数値の位取りは文字の列では無効であるため、″long_column_name″ の列の NUMERIC_SCALE にはヌル値が戻されます。文字長は数値の列では無効であるため、INTCOL の列の CHARACTER_MAXIMUM_LENGTH にはヌル値が戻されます。

インストールとバックアップに関する考慮事項: ある種のカタログ表、およびカタログ表とビューに関して作成されたビューは、定期的に保管してください。

- カタログ表 QSYS.QADBXRDBD には、リレーショナル・データベース情報が入っています。この表は定期的に保管する必要があります。
- ILE の外部関数またはプロシージャー、または SQL の関数またはプロシージャーを復元すると、これらのカタログ表に情報が自動的に挿入されます。ただし、非 ILE 外部関数およびプロシージャーの場合には、情報の自動挿入は行われません。非 ILE 外部関数またはプロシージャーの定義をバックアップするには、カタログ表 SYSROUTINES および SYSPARMS を確実に保管するか、またはこれらの関数およびプロシージャーを作成するために使用した SQL ソース・ステートメントのバックアップを必ずとっておくようにしてください。
- スキーマ QSYS2 または SYSIBM 内のカタログ・ビューは、すべてシステム・オブジェクトです。つまり、これらのカタログ・ビューに関して作成されたユーザー・ビューは、オペレーティング・システムのインストール時にすべて削除されます。従属オブジェクトも、すべて削除されます。この削除要件を避けるために、インストールの前にビューを保管しておき、後で復元することができます。
- QSYS ライブラリー内のカタログ表もシステム・オブジェクトです。しかし、QSYS ライブラリー内のカタログ表は、インストール時には削除されません。したがって、これらの表に関して作成されたビューはすべて、インストール・プロセスの終了後も保存されています。

カタログ・ビューへの特権の認可: カタログの表およびビューは、他のデータベース表およびデータベース・ビューと類似しています。権限を持っているユーザーであれば、他の表からデータを検索するときと同じように、SQL ステートメントを使用してカタログ・ビューのデータを見ることができます。 カタログの表およびビューでは、出荷時に、PUBLIC に SELECT 特権が認可されています。この特権を取り消して、個々のユーザーに SELECT 特権を認可することができます。

QSYS カタログ表: ほとんどのカタログ・ビューは、QSYS ライブラリー (データベース相互参照ファイルとも言う) の中の次の表に基づいています。これらの表は、出荷時に SELECT 特権は PUBLIC には認可されていません。また、これらの表を直接使用してはなりません。

QADBCCST	QADBKFLD	QADBXSFLD
QADBFDEP	QADBPKG	QADBXTRIGB
QADBFCST	QADBXRDBD	QADBXTRIGC
QADBIFLD	QADBXREF	QADBXTRIGD

iSeries のカタログ表およびカタログ・ビュー

iSeries カタログには、QSYS2 スキーマの中の以下のビューおよび表が含まれます。

DB2 UDB for iSeries の名前	対応する ANSI/ISO の名前	説明
934 ページの『SYSCATALOGS』	CATALOGS	リレーショナル・データベースについての情報
936 ページの『SYSCHKCST』	CHECK_CONSTRAINTS	検査制約についての情報
937 ページの『SYSCOLUMNS』	COLUMNS	列属性についての情報
947 ページの『SYSCST』	TABLE_CONSTRAINTS	すべての制約についての情報
948 ページの『SYSCSTCOL』	CONSTRAINT_COLUMN_USAGE	制約で参照される列についての情報
949 ページの『SYSCSTDEP』	CONSTRAINT_TABLE_USAGE	表に関する制約従属関係についての情報
950 ページの『SYSFUNCS』	ROUTINES	ユーザー定義関数についての情報
957 ページの『SYSINDEXES』		索引についての情報
958 ページの『SYSJARCONTENTS』		Java ルーチンの jar についての情報
959 ページの『SYSJAROBJECTS』		Java ルーチンの jar についての情報
960 ページの『SYSKEYCST』	KEY_COLUMN_USAGE	固有、基本、および外部キーについての情報
961 ページの『SYSKEYS』		索引キーについての情報
962 ページの『SYSPACKAGE』		パッケージについての情報
964 ページの『SYSPARMS』	PARAMETERS	ルーチン・パラメーターについての情報
969 ページの『SYSPROCS』	ROUTINES	プロシージャーについての情報
974 ページの『SYSREFCST』	REFERENTIAL_CONSTRAINTS	参照制約についての情報
976 ページの『SYSROUTINES』	ROUTINES	関数およびプロシージャーについての情報
975 ページの『SYSROUTINEDEP』	ROUTINE_TABLE_USAGE	関数およびプロシージャーの従属関係についての情報
986 ページの『SYSTABLES』	TABLES	表およびビューについての情報
988 ページの『SYSTRIGCOL』	TRIGGER_COLUMN_USAGE	トリガーで使用される列についての情報
989 ページの『SYSTRIGDEP』	TRIGGER_TABLE_USAGE	トリガーで使用されるオブジェクトについての情報
990 ページの『SYSTRIGGERS』	TRIGGERS	トリガーについての情報
994 ページの『SYSTRIGUPD』	TRIGGERED_UPDATE_COLUMNS	トリガーの WHEN 文節内の列についての情報
995 ページの『SYSTYPES』	USER_DEFINED_TYPES	組み込みのデータ・タイプおよび特殊タイプについての情報
1001 ページの『SYSVIEWDEP』	VIEW_TABLE_USAGE	表のビューの従属関係についての情報
1003 ページの『SYSVIEWS』	VIEWS	ビューの定義についての情報

SYSCATALOGS

SYSCATALOGS ビューには、ユーザーが接続できる各リレーショナル・データベースごとに、行が 1 つずつ入ります。次の表は、SYSCATALOGS ビューの列について説明しています。

表 97. SYSCATALOGS ビュー

列名	システム列名	データ・タイプ	説明
CATALOG_NAME	LOCATION	VARCHAR(18)	リレーショナル・データベース名
CATALOG_STATUS	RDBASPSTAT	CHAR(10)	リレーショナル・データベースの状況。 **ACTIVE** リレーショナル・データベースは、アクティブな独立補助記憶域プール (IASP) に関連付けられていますが、まだ使用可能ではありません。 **AVAILABLE** リレーショナル・データベースは使用可能です。 **VARYOFF** リレーショナル・データベースは、オフに変更された独立補助記憶域プール (IASP) に関連付けられています。 **VARYON** リレーショナル・データベースは、オンに変更された独立補助記憶域プール (IASP) に関連付けられていますが、まだ使用可能ではありません。 **UNKNOWN** リレーショナル・データベースの状況は不明です。リモート・リレーショナル・データベースの状況は、常に不明です。
CATALOG_TYPE	RDBTYPE	CHAR(7)	リレーショナル・データベースのタイプ。 **LOCAL** リレーショナル・データベースは、このシステムにとってローカルのデータベースです。 **REMOTE** リレーショナル・データベースは、リモート・システム上にあります。

表 97. *SYSCATALOGS* ビュー *(続き)*

列名	システム列名	データ・タイプ	説明
CATALOG_ASPGRP	RDBASPGRP	VARCHAR(10) ヌル可能	独立補助記憶域プール (IASP) の名前。 リレーショナル・データベースの状況が UNKNOWN の場合は、ヌル値が入ります。
CATALOG_ASPNUM	RDBASPNUM	VARCHAR(10) ヌル可能	独立補助記憶域プール (IASP) の番号。 リレーショナル・データベースの状況が UNKNOWN の場合は、ヌル値が入ります。
CATALOG_TEXT	RDBTEXT	CHAR(50)	リレーショナル・データベースのテキスト記述。

SYSCHKCST

SYSCHKCST ビューには、SQL のスキーマにある各検査制約ごとに、行が 1 つずつ入ります。 次の表は、SYSCHKCST ビューの列について説明しています。

表 98. SYSCHKCST ビュー

列名	システム列名	データ・タイプ	説明
CONSTRAINT_SCHEMA	DBNAME	VARCHAR(128)	該当の制約が入っているスキーマの名前。
CONSTRAINT_NAME	RELNAME	VARCHAR(128)	制約の名前
CHECK_CLAUSE	CHECK	VARCHAR(2000) ヌル可能	検査制約文節のテキスト。 切り捨てなければ検査文節を表示できない場合は、ヌル値が入ります。

SYSCOLUMNS

SYSCOLUMNS ビューには、SQL スキーマの中の各表または各ビューの各列 (SQL カタログの列を含む) ごとに行が 1 つずつ入ります。次の表は、SYSCOLUMNS ビューの列について説明しています。

表 99. SYSCOLUMNS ビュー

列名	システム列名	データ・タイプ	説明
COLUMN_NAME	NAME	VARCHAR(128)	列の名前。 SQL の列名が存在する場合は、その SQL の列名になります。存在しない場合は、システムの列名になります。
TABLE_NAME	TBNAME	VARCHAR(128)	該当の列を含む表またはビューの名前。 SQL の表名またはビュー名が存在する場合は、その SQL の表名またはビュー名です。存在しない場合は、システムの表名またはビュー名になります。
TABLE_OWNER	TBCREATOR	VARCHAR(128)	表またはビューの所有者。
ORDINAL_POSITION	COLNO	INTEGER	表またはビューにおける該当の列の数値位置 (左から右への順序)。

SYSCOLUMNS

表 99. SYSCOLUMNS ビュー (続き)

列名	システム列名	データ・タイプ	説明	
DATA_TYPE	COLTYPE	VARCHAR(8)	列のタイプ:	
			BIGINT	大整数
			INTEGER	長整数
			SMALLINT	短整数
			DECIMAL	パック 10 進数
			NUMERIC	ゾーン 10 進数
			FLOAT	浮動小数点数; FLOAT、REAL、または DOUBLE PRECISION
			CHAR	固定長文字ストリング
			VARCHAR	可変長文字ストリング
			CLOB	文字ラージ・オブジェクト・ストリング
			GRAPHIC	固定長グラフィック・ストリング
			VARG	可変長グラフィック・ストリング
			DBCLOB	2 バイト文字ラージ・オブジェクト・ストリング
			BLOB	バイナリー・ラージ・オブジェクト・ストリング
			DATE	日付
			TIME	時刻
			TIMESTMP	タイム・スタンプ
			DATALINK	データ・リンク
			ROWID	行 ID
			DISTINCT	特殊タイプ

表 99. SYSCOLUMNS ビュー (続き)

列名	システム列名	データ・タイプ	説明
LENGTH	LENGTH	INTEGER	列の長さ属性。ただし、10 進数、数値、または非ゼロ精度 2 進数の列の場合は、その精度: **8 バイト** BIGINT **4 バイト** INTEGER **2 バイト** SMALLINT **数値の精度** DECIMAL **数値の精度** NUMERIC **8 バイト** FLOAT、FLOAT(n) (ここで n = 25 〜 53)、または DOUBLE PRECISION **4 バイト** FLOAT(n) (ここで n = 1 〜 24)、または REAL **ストリングの長さ** CHAR **ストリングの最大長** VARCHAR または CLOB **グラフィック・ストリングの長さ** GRAPHIC **グラフィック・ストリングの最大長** VARGRAPHIC または DBCLOB **2 進ストリングの最大長** BLOB **4 バイト** DATE **3 バイト** TIME **10 バイト** TIMESTAMP **データ・リンクの最大長** **URL およびコメント** DATALINK **40 バイト** ROWID **ソース・タイプと同じ値** DISTINCT
NUMERIC_SCALE	SCALE	INTEGER ヌル可能	数値データの位取り。 列が 10 進数、数値、または 2 進数でない場合は、ヌル値が入ります。

SYSCOLUMNS

表 99. SYSCOLUMNS ビュー (続き)

列名	システム列名	データ・タイプ	説明
IS_NULLABLE	NULLS	CHAR(1)	列にヌル値を入れることが可能かどうか: **N** 不可 **Y** 可
IS_UPDATABLE	UPDATES	CHAR(1)	列が更新可能かどうか: **N** 不可 **Y** 可
LONG_COMMENT	REMARKS	VARCHAR(2000) ヌル可能	COMMENT ステートメントで指定された文字ストリング。 詳細コメントがない場合は、ヌル値が入ります。
HAS_DEFAULT	DEFAULT	CHAR(1)	列がデフォルト値をもつ (DEFAULT 文節またはヌル使用可能) かどうか: **N** 不可 **Y** 可 **A** 列は、ROWID データ・タイプおよび GENERATED ALWAYS 属性をもちます。 **D** 列は、ROWID データ・タイプおよび GENERATED BY DEFAULT 属性をもちます。 **I** 列は、AS IDENTITY 属性および GENERATED ALWAYS 属性により定義されます。 **J** 列は、AS IDENTITY 属性および GENERATED 属性により定義されます。
COLUMN_HEADING	LABEL	VARCHAR(60) ヌル可能	LABEL ステートメント (列見出し) で指定された文字ストリング。 列見出しがない場合は、ヌル値が入ります。

表 99. SYSCOLUMNS ビュー (続き)

列名	システム列名	データ・タイプ	説明
STORAGE	STORAGE	INTEGER	列の記憶域所要量: **8 バイト** BIGINT **4 バイト** INTEGER **2 バイト** SMALLINT **(精度/2) + 1** DECIMAL **数値の精度** NUMERIC **8 バイト** FLOAT、FLOAT(n) (ここで n = 25 〜 53)、または DOUBLE PRECISION **4 バイト** FLOAT(n) (ここで n = 1 〜 24)、または REAL **ストリングの長さ** CHAR **ストリングの最大長 + 2** VARCHAR **ストリングの最大長 + 29** CLOB **ストリングの長さ * 2** GRAPHIC **ストリングの最大長 * 2 + 2** VARGRAPHIC **ストリングの最大長 * 2 + 29** DBCLOB **4 バイト** DATE **3 バイト** TIME **10 バイト** TIMESTAMP **データ・リンクの最大長 + 24** **URL** およびコメント DATALINK **42 バイト** ROWID **ソース・タイプと同じ値** DISTINCT **注:** この列は、すべてのデータ・タイプの記憶域所要量を示します。

SYSCOLUMNS

表 99. SYSCOLUMNS ビュー (続き)

列名	システム列名	データ・タイプ	説明
NUMERIC_PRECISION	PRECISION	INTEGER ヌル可能	数値の列すべての精度。 **注:** この列では、すべての数値データ・タイプ (単精度および倍精度の浮動小数点数を含む) の精度を指定します。NUMERIC_PRECISION_RADIX 列は、この列の値が 2 進数であるか、または 10 進数であるかを示します。 列が数値の列でない場合は、ヌル値が入ります。
CCSID	CCSID	INTEGER ヌル可能	CHAR、VARCHAR、CLOB、DATE、TIME、TIMESTAMP、GRAPHIC、VARGRAPHIC、DBCLOB、および DATALINK 列の CCSID の値。 列が BLOB または ROWID の場合は、65535 が入ります。 列が数値データ・タイプの場合は、ヌル値が入ります。
TABLE_SCHEMA	DBNAME	VARCHAR(128)	該当の表またはビューが入っている SQL スキーマの名前。

表 99. SYSCOLUMNS ビュー (続き)

列名	システム列名	データ・タイプ	説明
COLUMN_DEFAULT	DFTVALUE	VARCHAR(2000) ヌル可能	列のデフォルト値が存在する場合は、そのデフォルト値。列のデフォルト値が、切り捨てなければ表示できない場合は、その列の値はストリング 'TRUNCATED' になります。デフォルト値は文字形式で保管されます。以下の特殊値も存在します。 **CURRENT_DATE** デフォルト値は、現在の日付です。 **CURRENT_TIME** デフォルト値は、現在の時刻です。 **CURRENT_TIMESTAMP** デフォルト値は、現在のタイム・スタンプです。 **NULL** デフォルト値は、ヌル値です。 **USER** デフォルト値は、現在のジョブ・ユーザーです。 列がデフォルト値をもたない場合は、ヌル値が入ります。例えば、列に IDENTITY 属性が指定されている場合、または列が行 ID である場合です。
CHARACTER_MAXIMUM_LENGTH	CHARLEN	INTEGER ヌル可能	データ・タイプが 2 進数、文字およびグラフィック・ストリングの場合は、ストリングの最大長。 列がストリングでない場合は、ヌル値が入ります。
CHARACTER_OCTET_LENGTH	CHARBYTE	INTEGER ヌル可能	データ・タイプが 2 進数、文字およびグラフィック・ストリングの場合は、バイト数。 列がストリングでない場合は、ヌル値が入ります。

SYSCOLUMNS

表 99. SYSCOLUMNS ビュー (続き)

列名	システム列名	データ・タイプ	説明
NUMERIC_PRECISION_RADIX	RADIX	INTEGER ヌル可能	NUMERIC_PRECISION の列で指定される精度が、2 進数 と 10 進数のどちらの数値で指定されるかを指示します。 **2** 2 進数: 浮動小数点数の精度は 2 進数で指定されます。 **10** 10 進数: 他の数値タイプはすべて 10 進数で指定されます。 列が数値の列でない場合は、ヌル値が入ります。
DATETIME_PRECISION	DATPRC	INTEGER ヌル可能	日付、時刻、またはタイム・スタンプの小数部分。 **0** データ・タイプが DATE および TIME の場合 **6** データ・タイプが TIMESTAMP の場合 (マイクロ秒数) 列が日付、時刻、またはタイム・スタンプの列でない場合は、ヌル値が入ります。
COLUMN_TEXT	LABELTEXT	VARCHAR(50) ヌル可能	LABEL ステートメント (列テキスト) で指定された文字ストリング。 列テキストがない場合は、ヌル値が入ります。
SYSTEM_COLUMN_NAME	SYS_CNAME	CHAR(10)	列のシステム名
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10)	表またはビューのシステム名
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10)	スキーマのシステム名
USER_DEFINED_TYPE_SCHEMA	TYPESCHEMA	VARCHAR(128) ヌル可能	これが 特殊タイプ の場合、スキーマの名前。 列が 特殊タイプ の列でない場合は、ヌル値が入ります。
USER_DEFINED_TYPE_NAME	TYPENAME	VARCHAR(128) ヌル可能	特殊タイプ の名前。 列が 特殊タイプ の列でない場合は、ヌル値が入ります。
IS_IDENTITY	IDENTITY	VARCHAR(3)	この列は、列が識別列かどうかを指定します。 **NO** 列は識別列ではありません。 **YES** 列は識別列です。

表 99. SYSCOLUMNS ビュー (続き)

列名	システム列名	データ・タイプ	説明
IDENTITY_GENERATION	GENERATED	VARCHAR(10) ヌル可能	この列は、列が GENERATED ALWAYS か GENERATED BY DEFAULT かを識別します。 **ALWAYS** 列の値は常に生成されます。 **BY DEFAULT** 列の値はデフォルトにより生成されます。 列が ROWID 列または IDENTITY 列でない場合は、ヌル値が入ります。
IDENTITY_START	START	DECIMAL(31,0) ヌル可能	識別列の開始値。 列が IDENTITY 列でない場合は、ヌル値が入ります。
IDENTITY_INCREMENT	INCREMENT	DECIMAL(31,0) ヌル可能	識別列の増分値。 列が IDENTITY 列でない場合は、ヌル値が入ります。
IDENTITY_MINIMUM	MINVALUE	DECIMAL(31,0) ヌル可能	識別列の最小値。 列が IDENTITY 列でない場合は、ヌル値が入ります。
IDENTITY_MAXIMUM	MAXVALUE	DECIMAL(31,0) ヌル可能	識別列の最大値。 列が IDENTITY 列でない場合は、ヌル値が入ります。
IDENTITY_CYCLE	CYCLE	VARCHAR(3) ヌル可能	この列は、識別列の値が最小値または最大値に達した後も、値の生成を続けるかどうかを識別します。 **NO** 値の生成は継続されません。 **YES** 値の生成は継続されます。 列が IDENTITY 列でない場合は、ヌル値が入ります。
IDENTITY_CACHE	CACHE	INTEGER ヌル可能	アクセスを高速化するために事前割り振りが可能な識別値の数を指定します。ゼロは、値が事前割り振りされないことを示します。 列が IDENTITY 列でない場合は、ヌル値が入ります。

表 99. SYSCOLUMNS ビュー (続き)

列名	システム列名	データ・タイプ	説明
IDENTITY_ORDER	ORDER	VARCHAR(3) ヌル可能	識別値を要求された順序で生成しなければならないかどうかを指定します。 **NO** 値は、要求された順序で生成する必要はありません。 **YES** 値は、要求された順序で生成する必要があります。 列が IDENTITY 列でない場合は、ヌル値が入ります。

SYSCST

SYSCST ビューには、SQL のスキーマにある各制約ごとに、行が 1 つずつ入ります。次の表は、SYSCST ビューの列について説明しています。

表 100. SYSCST ビュー

列名	システム列名	データ・タイプ	説明
CONSTRAINT_SCHEMA	CDBNAME	VARCHAR(128)	該当の制約が入っているスキーマの名前。
CONSTRAINT_NAME	RELNAME	VARCHAR(128)	制約の名前。
CONSTRAINT_TYPE	TYPE	VARCHAR(11)	制約のタイプ CHECK UNIQUE PRIMARY KEY FOREIGN KEY
TABLE_SCHEMA	TDBNAME	VARCHAR(128)	該当の表が入っているスキーマの名前。
TABLE_NAME	TBNAME	VARCHAR(128)	該当の制約が作成される表の名前。SQL の表名が存在する場合は、その SQL の表名になります。存在しない場合は、システムの表名になります。
IS_DEFERRABLE	ISDEFER	VARCHAR(3)	制約の検査が据え置きできるかどうかを示します。 常に 'NO' になります。
INITIALLY_DEFERRED	INITDEFER	VARCHAR(3)	制約が初期据え置きとして定義されたかどうかを示します。 常に 'NO' になります。
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10)	表のシステム名。
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10)	該当の表が入っているスキーマのシステム名。
CONSTRAINT_KEYS	COLCOUNT	SMALLINT ヌル可能	これが UNIQUE、PRIMARY KEY、または FOREIGN KEY 制約の場合は、キー列の数を指定します。 制約が CHECK 制約の場合は、ヌル値が入ります。
IASP_NUMBER	IASPNUMBER	SMALLINT	独立補助記憶域プール (IASP) 番号を指定します。

SYSCSTCOL

ビュー SYSCSTCOL は、制約が定義される対象となる列を記録します。固有キー制約または基本キー制約の列ごとに、および参照制約の参照列に対して、行が 1 つずつあります。 次の表は、SYSCSTCOL ビューの列について説明しています。

表 101. SYSCSTCOL ビュー

列名	システム列名	データ・タイプ	説明
TABLE_SCHEMA	TDBNAME	VARCHAR(128)	該当の制約が従属している表が入っている SQL スキーマの名前。
TABLE_NAME	TBNAME	VARCHAR(128)	該当の制約が従属している表の名前。SQL の表名が存在する場合は、その SQL の表名です。存在しない場合は、システムの表名になります。
COLUMN_NAME	COLUMN	VARCHAR(128)	該当の制約が作成された列。 SQL の列名が存在する場合は、その SQL の列名です。存在しない場合は、システムの列名になります。
CONSTRAINT_SCHEMA	CDBNAME	VARCHAR(128)	該当の制約のスキーマの名前。
CONSTRAINT_NAME	RELNAME	VARCHAR(128)	制約の名前。
SYSTEM_COLUMN_NAME	SYS_CNAME	CHAR(10)	列のシステム名。
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10)	表のシステム名。
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10)	該当の表が入っているスキーマのシステム名。

SYSCSTDEP

ビュー SYSCSTDEP は、制約が定義される対象となる表を記録します。次の表は、SYSCSTDEP ビューの列について説明しています。

表 102. SYSCSTDEP ビュー

列名	システム列名	データ・タイプ	説明
TABLE_SCHEMA	TDBNAME	VARCHAR(128)	該当の制約が従属している表が入っている SQL スキーマの名前。
TABLE_NAME	TBNAME	VARCHAR(128)	該当の制約が従属している表の名前。SQL の表名が存在する場合は、その SQL の表名です。存在しない場合は、システムの表名になります。
CONSTRAINT_SCHEMA	CDBNAME	VARCHAR(128)	該当の制約のスキーマの名前。
CONSTRAINT_NAME	RELNAME	VARCHAR(128)	制約の名前。
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10)	表のシステム名。
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10)	該当の表が入っているスキーマのシステム名。
IASP_NUMBER	IASPNUMBER	SMALLINT	独立補助記憶域プール (IASP) 番号を指定します。

SYSFUNCS

SYSFUNCS ビューには、CREATE FUNCTION ステートメントによって作成された各関数ごとに行が 1 つずつ入ります。次の表は、SYSFUNCS ビューの列について説明しています。

表 103. SYSFUNCS ビュー

列名	システム列名	データ・タイプ	説明
SPECIFIC_SCHEMA	SPECSCHEMA	VARCHAR(128)	ルーチン (関数) インスタンスのスキーマ名。
SPECIFIC_NAME	SPECNAME	VARCHAR(128)	ルーチン・インスタンスの特定名。
ROUTINE_SCHEMA	FUNCSCHEMA	VARCHAR(128)	該当のルーチンが入っている SQL スキーマの名前。
ROUTINE_NAME	FUNCNAME	VARCHAR(128)	ルーチンの名前。
ROUTINE_CREATED	FUNCCREATE	TIMESTAMP	ルーチンが作成されたときのタイム・スタンプを識別します。
ROUTINE_DEFINER	DEFINER	VARCHAR(128)	該当のルーチンを定義したユーザーの名前。
ROUTINE_BODY	BODY	VARCHAR(8)	ルーチン本体のタイプ: **EXTERNAL** これは外部ルーチンです。 **SQL** これは SQL ルーチンです。
EXTERNAL_NAME	EXTNAME	VARCHAR(279) ヌル可能	これが外部ルーチンである場合は、この列は外部プログラム名を識別します。 • ILE サービス・プログラムの場合、外部プログラム名はスキーマ名/サービス・プログラム名 *(入り口名)* です。 • Java プログラムの場合、外部プログラム名は任意選択の jar-id の後に完全修飾クラス名!メソッド名 または 完全修飾クラス名.メソッド名 が続きます。 • その他のすべての言語では、外部プログラム名は、スキーマ名/プログラム名 です。 これが外部ルーチンでない場合は、ヌル値が入ります。

表 103. SYSFUNCS ビュー (続き)

列名	システム列名	データ・タイプ	説明
EXTERNAL_LANGUAGE	LANGUAGE	VARCHAR(8) ヌル可能	これが外部ルーチンである場合は、この列は外部プログラム名を識別します。 **C** 外部プログラムは C で作成されます。 **C++** 外部プログラムは C++ で作成されます。 **CL** 外部プログラムは CL で作成されます。 **COBOL** 外部プログラムは COBOL で作成されます。 **COBOLLE** 外部プログラムは ILE COBOL で作成されます。 **JAVA** 外部プログラムは JAVA で作成されます。 **PLI** 外部プログラムは PL/I で作成されます。 **RPG** 外部プログラムは RPG で作成されます。 **RPGLE** 外部プログラムは ILE RPG で作成されます。 これが外部ルーチンでない場合は、ヌル値が入ります。

SYSFUNCS

表 103. SYSFUNCS ビュー (続き)

列名	システム列名	データ・タイプ	説明
PARAMETER_STYLE	PARM_STYLE	VARCHAR(7) ヌル可能	これが外部ルーチンである場合は、この列はパラメーターのスタイル (呼び出し規則) を識別します。 **DB2SQL** これは DB2SQL 呼び出し規則です。 **DB2GNRL** これは DB2GENERAL 呼び出し規則です。 **GENERAL** これは GENERAL 呼び出し規則です。 **JAVA** これは JAVA 呼び出し規則です。 **NULLS** これは GENERAL WITH NULLS 呼び出し規則です。 **SQL** これは SQL 標準呼び出し規則です。 これが外部ルーチンでない場合は、ヌル値が入ります。
IS_DETERMINISTIC	DETERMINE	VARCHAR(3)	この列はルーチンが deterministic であるかどうかを識別します。 つまり、同じ引き数のルーチンに対する呼び出しが、常に同じ結果を戻すかどうかを識別します。 **NO** ルーチンは deterministic ではありません。 **YES** ルーチンは deterministic です。

表 103. SYSFUNCS ビュー (続き)

列名	システム列名	データ・タイプ	説明
SQL_DATA_ACCESS	DATAACCESS	VARCHAR(8)	この列は、ルーチンに SQL が含まれているか、およびルーチンがデータの読み取りまたは変更を行うかを識別します。 **NONE** ルーチンは SQL ステートメントを含みません。 **CONTAINS** ルーチンは SQL ステートメントを含みます。 **READS** ルーチンは、おそらく表またはビューからデータを読み取ります。 **MODIFIES** ルーチンは、おそらく表またはビュー内のデータを変更するか、SQL DDL ステートメントを発行します。
SQL_PATH	SQL_PATH	VARCHAR(3483) ヌル可能	これが SQL ルーチンの場合、この列はパスを識別します。 これが外部ルーチンの場合は、ヌル値が入ります。
PARM_SIGNATURE	SIGNATURE	VARCHAR(510)	この列はルーチン・シグニチャーを識別します。
NUMBER_OF_RESULTS	NUMRESULTS	SMALLINT	結果の数を識別します。
IN_PARMS	IN_PARMS	SMALLINT	入力パラメーターの数を識別します。0 は入力パラメーターがないことを示します。
LONG_COMMENT	REMARKS	VARCHAR(2000) ヌル可能	COMMENT ステートメントで指定された文字ストリング。 詳細コメントがない場合は、ヌル値が入ります。
ROUTINE_DEFINITION	ROUTINEDEF	VARCHAR(24000) ヌル可能	これが SQL ルーチンの場合、この列は SQL ルーチン本体を含みます。 これが SQL ルーチンでない場合、または切り捨てなければルーチン本体をこの列に収容できない場合は、ヌル値が入ります。

SYSFUNCS

表 103. SYSFUNCS ビュー (続き)

列名	システム列名	データ・タイプ	説明
FUNCTION_ORIGIN	ORIGIN	CHAR(1)	関数のタイプを識別します。これがプロシージャーの場合、この列にはブランクが入ります。 **B** これは組み込み関数 (DB2 UDB for iSeriesによって定義された) です。 **E** これはユーザー定義関数です。 **U** これは、他の関数に基づいているユーザー定義関数です。 **S** これはシステム生成関数です。
FUNCTION_TYPE	TYPE	CHAR(1)	関数の形式を識別します。これがプロシージャーの場合、この列にはブランクが入ります。 **S** これはスカラー関数です。 **C** これは列関数です。 **T** これは表関数です。
EXTERNAL_ACTION	EXTACTION	CHAR(1) ヌル可能	関数の呼び出しに外部的な作用があるかどうかを識別します。 **E** この関数には、外部的な副次作用があります。 **N** この関数には、外部的な副次作用はありません。
IS_NULL_CALL	NULL_CALL	VARCHAR(3) ヌル可能	入力パラメーターがヌル値である場合に、関数を呼び出す必要があるかどうかを識別します。 **NO** この関数は、入力パラメーターがヌル値の場合に呼び出す必要はありません。これがスカラー関数の場合は、いずれかのオペランドがヌルであれば、この関数の結果は暗黙的にヌルになります。これが表関数の場合は、いずれかのオペランドがヌル値であれば、この関数の結果は空の表になります。 **YES** この関数は、入力オペランドがヌルでも呼び出す必要があります。

表 103. SYSFUNCS ビュー (続き)

列名	システム列名	データ・タイプ	説明
SCRATCH_PAD	SCRATCHPAD	INTEGER ヌル可能	静的メモリー域 (スクラッチパッド) のアドレスが関数に渡されるかどうかを識別します。 **0** 関数にはスクラッチパッドはありません。 **整数** 関数に渡されるスクラッチパッドのサイズを示します。
FINAL_CALL	FINAL_CALL	VARCHAR(3) ヌル可能	関数がその作業域 (スクラッチパッド) の終結処理を行えるようにするために、関数への最終呼び出しを行う必要があるかどうかを示します。 **NO** 最終呼び出しは行いません。 **YES** ステートメントが完了したときに関数への最終呼び出しを行います。
PARALLELIZABLE	PARALLEL	VARCHAR(3) ヌル可能	関数が並行して実行できるかどうかを識別します。 **NO** 関数は同期させなければなりません。 **YES** 関数は並行して実行できます。
DBINFO	DBINFO	VARCHAR(3) ヌル可能	データベースに関する情報を関数に渡すかどうかを識別します。 **NO** 関数にデータベース情報を渡しません。 **YES** 関数にデータベースに関する情報を渡します。
SOURCE_ SPECIFIC_SCHEMA	SRCSCHEMA	VARCHAR(128) ヌル可能	これがソース化関数であり、ソースがユーザー定義の場合、この列にはソース・スキーマが入ります。これがソース化関数であり、ソースが組み込みである場合、この列には 'QSYS2' が入ります。 これがソース化関数でない場合は、ヌル値が入ります。
SOURCE_SPECIFIC_NAME	SRCNAME	VARCHAR(128) ヌル可能	これがソース化関数であり、ソースがユーザー定義である場合、この列にはソース関数名の特定名が入ります。 これがソース化関数でない場合は、ヌル値が入ります。

SYSFUNCS

表 103. SYSFUNCS ビュー (続き)

列名	システム列名	データ・タイプ	説明
IS_USER_DEFINED_CAST	CAST_FUNC	VARCHAR(3) ヌル可能	この関数は、特殊タイプ の作成時に作成されたキャスト関数であるかどうかを識別します。 **NO** この関数はキャスト関数ではありません。 **YES** この関数はキャスト関数です。
CARDINALITY	CARD	BIGINT ヌル可能	表関数の基数を指定します。 この関数が表関数でない場合、または基数が指定されていなかった場合は、ヌル値が入ります。
FENCED	FENCED	VARCHAR(3) ヌル可能	関数を隔離するかどうかを指定します。 **NO** 関数を隔離しません。 **YES** 関数を隔離します。
IASP_NUMBER	IASPNUMBER	SMALLINT	独立補助記憶域プール (IASP) 番号を指定します。

SYSINDEXES

SYSINDEXES ビューには、SQL CREATE INDEX ステートメントを使用して作成された SQL スキーマにある各索引 (SQL カタログに関する索引を含む) ごとに、行が 1 つずつ入ります。 次の表は、SYSINDEXES ビューの列について説明しています。

表 104. SYSINDEXES ビュー

列名	システム列名	データ・タイプ	説明
INDEX_NAME	NAME	VARCHAR(128)	索引の名前。 SQL の索引名が存在する場合は、その SQL の索引名になります。存在しない場合は、システムの索引名になります。
INDEX_OWNER	CREATOR	VARCHAR(128)	索引の所有者
TABLE_NAME	TBNAME	VARCHAR(128)	該当の索引が定義される表の名前。 SQL の表名が存在する場合は、その SQL の表名になります。存在しない場合は、システムの表名になります。
TABLE_OWNER	TBCREATOR	VARCHAR(128)	表の所有者
TABLE_SCHEMA	TBDBNAME	VARCHAR(128)	該当の索引が定義される表が入っている SQL スキーマの名前
IS_UNIQUE	UNIQUERULE	CHAR(1)	索引が固有索引の場合 : **D** NO (重複が許されます) **V** YES (重複 NULL 値が許されます) **U** あり **E** コード化ベクトル索引
COLUMN_COUNT	COLCOUNT	INTEGER	キーの中の列の数
INDEX_SCHEMA	DBNAME	VARCHAR(128)	該当の索引が入っている SQL スキーマの名前
SYSTEM_INDEX_NAME	SYS_IXNAME	CHAR(10)	システムの索引名
SYSTEM_INDEX_SCHEMA	SYS_IDNAME	CHAR(10)	システムの索引スキーマ名
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10)	システムの表名
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10)	システムの表スキーマ名
LONG_COMMENT	REMARKS	VARCHAR(2000) ヌル可能	COMMENT ステートメントで指定された文字ストリング。 詳細コメントがない場合は、ヌル値が入ります。
IASP_NUMBER	IASPNUMBER	SMALLINT	独立補助記憶域プール (IASP) 番号を指定します。

SYSJARCONTENTS

SYSJARCONTENTS 表には、SQL スキーマの jarid によって定義された各クラスごとに、行が 1 つずつ入ります。 次の表は、SYSJARCONTENTS ビューの列について説明しています。

表 105. SYSJARCONTENTS ビュー

列名	システム列名	データ・タイプ	説明
JARSCHEMA	JARSCHEMA	VARCHAR(128)	jar_id が入っているスキーマの名前。
JAR_ID	JAR_ID	VARCHAR(128)	jar_id の名前。
CLASS	CLASS	VARCHAR(128)	クラスの名前。
CLASS_SOURCE	CLASSSRC	DBCLOB(10485760) ヌル可能	予約済み。ヌル値が入ります。
IASP_NUMBER	IASPNUMBER	SMALLINT	独立補助記憶域プール (IASP) 番号を指定します。

SYSJAROBJECTS

SYSJAROBJECTS 表には、SQL スキーマの各 jarid ごとに、行が 1 つずつ入ります。次の表は、SYSJAROBJECTS ビューの列について説明しています。

表 106. SYSJAROBJECTS ビュー

列名	システム列名	データ・タイプ	説明
JARSCHEMA	JARSCHEMA	VARCHAR(128)	jar_id が入っているスキーマの名前。
JAR_ID	JAR_ID	VARCHAR(128)	jar_id の名前。
DEFINER	DEFINER	VARCHAR(128)	jarid の所有者の名前。
JAR_DATA	JAR_DATA	BLOB(104857600) ヌル可能	jar のバイト・コード。
IASP_NUMBER	IASPNUMBER	SMALLINT	独立補助記憶域プール (IASP) 番号を指定します。
JAR_CREATED	CREATEDTS	TIMESTAMP	Jar 作成のタイム・スタンプ。
LAST_ALTERED	ALTEREDTS	TIMESTAMP ヌル可能	予約済み。ヌル値が入ります。
DEBUG_MODE	DEBUG_MODE	CHAR(1)	関数がデバッグ可能かどうかを識別します。 **0** 関数はデバッグ不可能です。 **2** 関数はデバッグ可能です。
DEBUG_DATA	DEBUG_DATA	CLOB(1048576) ヌル可能	予約済み。ヌル値が入ります。

SYSKEYCST

SYSKEYCST ビューには、SQL スキーマにある各 UNIQUE KEY、PRIMARY KEY、または FOREIGN KEY ごとに、1 つまたは複数の行が入ります。固有キー制約または基本キー制約の列ごとに、および参照制約の参照列に対して、行が 1 つずつあります。次の表は、SYSKEYCST ビューの列について説明しています。

表 107. SYSKEYCST ビュー

列名	システム列名	データ・タイプ	説明
CONSTRAINT_SCHEMA	CDBNAME	VARCHAR(128)	該当の制約が入っているスキーマの名前。
CONSTRAINT_NAME	RELNAME	VARCHAR(128)	制約の名前。
TABLE_SCHEMA	TDBNAME	VARCHAR(128)	該当の表が入っているスキーマの名前。
TABLE_NAME	TBNAME	VARCHAR(128)	表の名前。
COLUMN_NAME	COLNAME	VARCHAR(128)	列の名前。
ORDINAL_POSITION	COLSEQ	INTEGER	キー内における列の位置。
COLUMN_POSITION	COLNO	INTEGER	行内における列の位置。
TABLE_OWNER	CREATOR	VARCHAR(128)	表の所有者。
SYSTEM_COLUMN_NAME	SYS_CNAME	CHAR(10)	列のシステム名。
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10)	表のシステム名。
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10)	該当のスキーマ表が入っているスキーマのシステム名。

SYSKEYS

SYSKEYS ビューには、SQL スキーマにある索引 (SQL カタログの索引のキーを含む) の各列ごとに、行が 1 つずつ入ります。次の表は、SYSKEYS ビューの列について説明しています。

表 108. SYSKEYS ビュー

列名	システム列名	データ・タイプ	説明
INDEX_NAME	IXNAME	VARCHAR(128)	索引の名前。 SQL の索引名が存在する場合は、その SQL の索引名になります。存在しない場合は、システムの索引名になります。
INDEX_OWNER	IXCREATOR	VARCHAR(128)	索引の所有者
COLUMN_NAME	COLNAME	VARCHAR(128)	該当のキーの列の名前。 SQL の列名が存在する場合は、その SQL の列名になります。存在しない場合は、システムの列名になります。
COLUMN_POSITION	COLNO	INTEGER	行内における列の数値位置
ORDINAL_POSITION	COLSEQ	INTEGER	キー内における列の数値位置
ORDERING	ORDERING	CHAR(1)	キー内における列の順序: **A** 昇順 **D** 降順
INDEX_SCHEMA	IXDBNAME	VARCHAR(128)	該当の索引が入っているスキーマの名前
SYSTEM_COLUMN_NAME	SYS_CNAME	CHAR(10)	列のシステム名
SYSTEM_INDEX_NAME	SYS_IXNAME	CHAR(10)	索引のシステム名
SYSTEM_INDEX_SCHEMA	SYS_IDNAME	CHAR(10)	該当の索引が入っているスキーマのシステム名

SYSPACKAGE

SYSPACKAGE ビューには、SQL のスキーマにある各 SQL パッケージごとに、行が 1 つずつ入ります。次の表は、SYSPACKAGE ビューの列について説明しています。

表 109. SYSPACKAGE ビュー

列名	システム列名	データ・タイプ	説明
PACKAGE_CATALOG	LOCATION	VARCHAR(128)	SQL パッケージのリレーショナル・データベース名 (RDBNAME)
PACKAGE_SCHEMA	COLLID	VARCHAR(128)	スキーマの名前
PACKAGE_NAME	NAME	VARCHAR(128)	SQL パッケージの名前
PACKAGE_OWNER	OWNER	VARCHAR(128)	SQL パッケージの所有者
PACKAGE_CREATOR	CREATOR	VARCHAR(128)	SQL パッケージの作成者
CREATION_TIMESTAMP	TIMESTAMP	CHAR(26)	SQL パッケージ作成時のタイム・スタンプ
DEFAULT_SCHEMA	QUALIFIER	VARCHAR(128)	修飾されていない表、ビュー、および索引の暗黙の名前
PROGRAM_NAME	PROGNAME	VARCHAR(128)	パッケージ作成の元になったプログラムの名前
PROGRAM_SCHEMA	LIBRARY	VARCHAR(128)	該当のプログラムが入っているスキーマの名前
PROGRAM_CATALOG	RDB	VARCHAR(128)	該当のプログラムが常駐するリレーショナル・データベースの名前
ISOLATION	ISOLATION	CHAR(2)	分離オプションの指定: RR 反復可能読み取り RS 読み取り固定 (*ALL) CS カーソル固定 (*CS) UR 非コミット読み取り (*CHG) NO なし (*NONE)
QUOTE	QUOTE	CHAR(1)	エスケープ文字の指定 (Y/N): Y = 引用符 N = アポストロフィ
COMMA	COMMA	CHAR(1)	コンマ・オプションの指定 (Y/N): Y = コンマ N = ピリオド
PACKAGE_TEXT	LABEL	VARCHAR(50)	ユーザーが LABEL ステートメントで指定する文字ストリング。
LONG_COMMENT	REMARKS	VARCHAR(2000)	COMMENT ステートメントで指定された文字ストリング。 詳細コメントがない場合は、ヌル値が入ります。
CONSISTENCY_TOKEN	CONTOKEN	CHAR(8) ビット・データ用	パッケージの整合性トークン
SYSTEM_PACKAGE_NAME	SYS_NAME	CHAR(10)	パッケージのシステム名
SYSTEM_PACKAGE_SCHEMA	SYS_DNAME	CHAR(10)	該当のパッケージが入っているスキーマのシステム名

表 109. SYSPACKAGE ビュー (続き)

列名	システム列名	データ・タイプ	説明
SYSTEM_DEFAULT_SCHEMA	SYS_DDNAME	CHAR(10)	修飾されていない表、ビュー、索引、およびパッケージの暗黙の修飾子のシステム名。
SYSTEM_PROGRAM_NAME	SYS_PNAME	CHAR(10)	プログラムのシステム名。
SYSTEM_PROGRAM_SCHEMA	SYS_PDNAME	CHAR(10)	該当のプログラムが入っているスキーマのシステム名。
IASP_NUMBER	IASPNUMBER	SMALLINT	独立補助記憶域プール (IASP) 番号を指定します。

SYSPARMS

SYSPARMS 表には、CREATE PROCEDURE ステートメントによって作成されたプロシージャー、または CREATE FUNCTION ステートメントによって作成された関数の、各パラメーターごとに行が 1 つずつ入ります。 次の表は、SYSPARMS 表の列について説明しています。

表 110. SYSPARMS 表

列名	システム列名	データ・タイプ	説明
SPECIFIC_SCHEMA	SPECSCHEMA	VARCHAR(128)	ルーチン (関数) インスタンスのスキーマ名。
SPECIFIC_NAME	SPECNAME	VARCHAR(128)	ルーチン・インスタンスの特定名。
ORDINAL_POSITION	PARMNO	INTEGER	パラメーター・リストにおける該当のパラメーターの数値位置 (左から右への順序)。
PARAMETER_MODE	PARMMODE	VARCHAR(5)	パラメーターのタイプ: **IN** これは入力パラメーターです。 **OUT** これは出力パラメーターです。 **INOUT** これは入出力パラメーターです。
PARAMETER_NAME	PARMNAME	VARCHAR(128) ヌル可能	パラメーターの名前。 パラメーターに名前がない場合は、ヌル値が入ります。

表 110. SYSPARMS 表 (続き)

列名	システム列名	データ・タイプ	説明
DATA_TYPE	DATA_TYPE	VARCHAR(128)	列のタイプ: **BIGINT** 大整数 **INTEGER** 長整数 **SMALLINT** 短整数 **DECIMAL** パック 10 進数 **NUMERIC** ゾーン 10 進数 **DOUBLE PRECISION** 浮動小数点数; DOUBLE PRECISION **REAL** 浮動小数点数; REAL **CHARACTER** 固定長文字ストリング **CHARACTER VARYING** 可変長文字ストリング **CHARACTER LARGE OBJECT** 文字ラージ・オブジェクト・ストリング **GRAPHIC** 固定長グラフィック・ストリング **GRAPHIC VARYING** 可変長グラフィック・ストリング **DOUBLE-BYTE CHARACTER LARGE OBJECT** 2 バイト文字ラージ・オブジェクト・ストリング **BINARY LARGE OBJECT** バイナリー・ラージ・オブジェクト・ストリング **DATE** 日付 **TIME** 時刻 **TIMESTAMP** タイム・スタンプ **DATALINK** データ・リンク **ROWID** 行 ID **DISTINCT** 特殊タイプ

表 110. SYSPARMS 表 (続き)

列名	システム列名	データ・タイプ	説明
NUMERIC_SCALE	SCALE	INTEGER ヌル可能	数値データの位取り パラメーターが 10 進数、数値、または 2 進数でない場合は、ヌル値が入ります。
NUMERIC_PRECISION	PRECISION	INTEGER ヌル可能	数値パラメーターすべての精度。 **注:** この列では、すべての数値データ・タイプ (単精度および倍精度の浮動小数点数を含む) の精度を指定します。NUMERIC_PRECISION_RADIX 列は、この列の値が 2 進数であるか、または 10 進数であるかを示します。 パラメーターが数値パラメーターでない場合は、ヌル値が入ります。
CCSID	CCSID	INTEGER ヌル可能	CHAR、VARCHAR、CLOB、DATE、TIME、TIMESTAMP、GRAPHIC、VARGRAPHIC、DBCLOB および DATALINK パラメーターの CCSID の値。 パラメーターが数値パラメーターでない場合は、ヌル値が入ります。
CHARACTER_MAXIMUM_LENGTH	CHARLEN	INTEGER ヌル可能	データ・タイプが 2 進数、文字およびグラフィック・ストリングの場合は、ストリングの最大長。 パラメーターがストリングでない場合は、ヌル値が入ります。
CHARACTER_OCTET_LENGTH	CHARBYTE	INTEGER ヌル可能	データ・タイプが 2 進数、文字およびグラフィック・ストリングの場合は、バイト数。 パラメーターがストリングでない場合は、ヌル値が入ります。
NUMERIC_PRECISION_RADIX	RADIX	INTEGER ヌル可能	NUMERIC_PRECISION の列で指定される精度が、2 進数 と 10 進数のどちらの数値で指定されるかを指示します。 **2** 2 進数: 浮動小数点数の精度は 2 進数で指定されます。 **10** 10 進数: 他の数値タイプはすべて 10 進数で指定されます。 パラメーターが数値パラメーターでない場合は、ヌル値が入ります。

表 110. SYSPARMS 表 (続き)

列名	システム列名	データ・タイプ	説明
DATETIME_PRECISION	DATPRC	INTEGER ヌル可能	日付、時刻、またはタイム・スタンプの小数部分。 **0** データ・タイプが DATE および TIME の場合 **6** データ・タイプが TIMESTAMP の場合 (マイクロ秒数) パラメーター日付、時刻、またはタイム・スタンプのパラメーターでない場合は、ヌル値が入ります。
IS_NULLABLE	NULLS	VARCHAR(3)	パラメーターがヌル可能かどうかを示します。 **NO** パラメーターにヌルは許されません。 **YES** パラメーターにヌルが許されます。
LONG_COMMENT	REMARKS	VARCHAR(2000) ヌル可能	COMMENT ステートメントで指定された文字ストリング。 詳細コメントがない場合は、ヌル値が入ります。
ROW_TYPE	ROWTYPE	CHAR(1)	行のタイプを識別します。これがプロシージャーに対するパラメーターの場合、この列にはヌル値が入ります。 **P** パラメーター。 **R** キャスト前の結果。 **C** キャスト後の結果。
DATA_TYPE_SCHEMA	TYPESCHEMA	VARCHAR(128) ヌル可能	これが 特殊タイプ の場合、データ・タイプのスキーマ。 パラメーターが 特殊タイプ パラメーターでない場合は、ヌル値が入ります。
DATA_TYPE_NAME	TYPENAME	VARCHAR(128) ヌル可能	これが 特殊タイプ の場合、データ・タイプの名前。 パラメーターが 特殊タイプ パラメーターでない場合は、ヌル値が入ります。

表 110. SYSPARMS 表 (続き)

列名	システム列名	データ・タイプ	説明
AS_LOCATOR	ASLOCATOR	VARCHAR(3)	パラメーターがロケーターとして指定されたかどうかを識別します。 **NO** パラメーターはロケーターとして指定されませんでした。 **YES** パラメーターはロケーターとして指定されました。
IASP_NUMBER	IASPNUMBER	SMALLINT	独立補助記憶域プール (IASP) 番号を指定します。

SYSPROCS

SYSPROCS ビューには、CREATE PROCEDURE ステートメントで作成された各プロシージャーごとに、行が 1 つずつ入ります。 次の表は、SYSPROCS ビューの列について説明しています。

表 111. SYSPROCS ビュー

列名	システム列名	データ・タイプ	説明
SPECIFIC_SCHEMA	SPECSCHEMA	VARCHAR(128)	ルーチン (プロシージャー) インスタンスのスキーマ名。
SPECIFIC_NAME	SPECNAME	VARCHAR(128)	ルーチン・インスタンスの特定名。
ROUTINE_SCHEMA	PROCSCHEMA	VARCHAR(128)	該当のルーチンが入っている SQL スキーマの名前。
ROUTINE_NAME	PROCNAME	VARCHAR(128)	ルーチンの名前。
ROUTINE_CREATED	RTNCREATE	TIMESTAMP	ルーチンが作成されたときのタイム・スタンプを識別します。
ROUTINE_DEFINER	DEFINER	VARCHAR(128)	該当のルーチンを定義したユーザーの名前。
ROUTINE_BODY	BODY	VARCHAR(8)	ルーチン本体のタイプ: **EXTERNAL** これは外部ルーチンです。 **SQL** これは SQL ルーチンです。
EXTERNAL_NAME	EXTNAME	VARCHAR(279) ヌル可能	これが外部ルーチンである場合は、この列は外部プログラム名を識別します。 • REXX の場合は、外部プログラム名は、*スキーマ名/ソース・ファイル名(メンバー名)* です。 • Java プログラムの場合、外部プログラム名は任意選択の jar-id の後に *完全修飾クラス名!メソッド名* または *完全修飾クラス名.メソッド名* が続きます。 • その他のすべての言語では、外部プログラム名は、*スキーマ名/プログラム名* です。 これが外部ルーチンでない場合は、ヌル値が入ります。

表 111. SYSPROCS ビュー (続き)

列名	システム列名	データ・タイプ	説明
EXTERNAL_LANGUAGE	LANGUAGE	VARCHAR(8) ヌル可能	これが外部ルーチンである場合は、この列は外部プログラム名を識別します。 **C** 外部プログラムは C で作成されます。 **C++** 外部プログラムは C++ で作成されます。 **CL** 外部プログラムは CL で作成されます。 **COBOL** 外部プログラムは COBOL で作成されます。 **COBOLLE** 外部プログラムは ILE COBOL で作成されます。 **FORTRAN** 外部プログラムは FORTRAN で作成されます。 **JAVA** 外部プログラムは JAVA で作成されます。 **PLI** 外部プログラムは PL/I で作成されます。 **REXX** 外部プログラムは REXX プロシージャーです。 **RPG** 外部プログラムは RPG で作成されます。 **RPGLE** 外部プログラムは ILE RPG で作成されます。 これが外部ルーチンでない場合は、ヌル値が入ります。

表 111. SYSPROCS ビュー (続き)

列名	システム列名	データ・タイプ	説明
PARAMETER_STYLE	PARM_STYLE	VARCHAR(7) ヌル可能	これが外部ルーチンである場合は、この列はパラメーターのスタイル (呼び出し規則) を識別します。 **DB2GNRL** これは DB2GENERAL 呼び出し規則です。 **DB2SQL** これは DB2SQL 呼び出し規則です。 **GENERAL** これは GENERAL 呼び出し規則です。 **JAVA** これは JAVA 呼び出し規則です。 **NULLS** これは GENERAL WITH NULLS 呼び出し規則です。 **SQL** これは SQL 標準呼び出し規則です。 これが外部ルーチンでない場合は、ヌル値が入ります。
IS_DETERMINISTIC	DETERMINE	VARCHAR(3)	この列はルーチンが deterministic であるかどうかを識別します。 つまり、同じ引き数のルーチンに対する呼び出しが、常に同じ結果を戻すかどうかを識別します。 **NO** ルーチンは deterministic ではありません。 **YES** ルーチンは deterministic です。

SYSPROCS

表 111. SYSPROCS ビュー (続き)

列名	システム列名	データ・タイプ	説明
SQL_DATA_ACCESS	DATAACCESS	VARCHAR(8)	この列は、ルーチンに SQL が含まれているか、およびルーチンがデータの読み取りまたは変更を行うかを識別します。 **NONE** ルーチンは SQL ステートメントを含みません。 **CONTAINS** ルーチンは SQL ステートメントを含みます。 **READS** ルーチンは、おそらく表またはビューからデータを読み取ります。 **MODIFIES** ルーチンは、おそらく表またはビュー内のデータを変更するか、SQL DDL ステートメントを発行します。
SQL_PATH	SQL_PATH	VARCHAR(3483) ヌル可能	これが SQL ルーチンの場合、この列はパスを識別します。 これが SQL ルーチンでない場合は、ヌル値が入ります。
PARM_SIGNATURE	SIGNATURE	VARCHAR(510)	この列はルーチン・シグニチャーを識別します。
RESULT_SETS	RESULTS	SMALLINT	戻される結果セットの最大数を識別します。 0 は結果セットがないことを示します。
IN_PARMS	IN_PARMS	SMALLINT	入力パラメーターの数を識別します。 0 は入力パラメーターがないことを示します。
OUT_PARMS	OUT_PARMS	SMALLINT	出力パラメーターの数を識別します。 0 は出力パラメーターがないことを示します。
INOUT_PARMS	INOUT_PARM	SMALLINT	入出力パラメーターの数を識別します。 0 は入出力パラメーターがないことを示します。
LONG_COMMENT	REMARKS	VARCHAR(2000) ヌル可能	COMMENT ステートメントで指定された文字ストリング。 詳細コメントがない場合は、ヌル値が入ります。

表 111. SYSPROCS ビュー (続き)

列名	システム列名	データ・タイプ	説明
ROUTINE_DEFINITION	ROUTINEDEF	VARCHAR(24000) ヌル可能	これが SQL ルーチンの場合、この列は SQL ルーチン本体を含みます。 これが SQL ルーチンでない場合、または切り捨てなければルーチン本体をこの列に収容できない場合は、ヌル値が入ります。
DBINFO	DBINFO	VARCHAR(3) ヌル可能	データベースに関する情報をプロシージャーに渡すかどうかを識別します。 **NO** データベースに関する情報をプロシージャーに渡しません。 **YES** データベースに関する情報をプロシージャーに渡します。
COMMIT_ON_RETURN	CMTONRET	VARCHAR(3) ヌル可能	この列は、プロシージャーから正常に戻った時点でそのプロシージャーをコミットするかどうかを識別します。 **NO** プロシージャーから正常に戻ったときに、コミットは行われません。 **YES** プロシージャーから正常に戻ったときに、コミットが行われます。
IASP_NUMBER	IASPNUMBER	SMALLINT	独立補助記憶域プール (IASP) 番号を指定します。
NEW_SAVEPOINT_LEVEL	NEWSAVEPTL	VARCHAR(3) ヌル可能	この列は、ルーチンが新しい保管ポイント・レベルを開始するかどうかを識別します。 **NO** 新しい保管ポイント・レベルを開始しません。 **YES** 新しい保管ポイント・レベルを開始します。

SYSREFCST

SYSREFCST ビューには、SQL のスキーマにある各外部キーごとに、行が 1 つずつ入ります。次の表は、SYSREFCST ビューの列について説明しています。

表 112. SYSREFCST ビュー

列名	システム列名	データ・タイプ	説明
CONSTRAINT_SCHEMA	CDBNAME	VARCHAR(128)	該当の制約が入っているスキーマの名前。
CONSTRAINT_NAME	RELNAME	VARCHAR(128)	制約の名前。
UNIQUE_CONSTRAINT_SCHEMA	UNQDBNAME	VARCHAR(128)	参照制約によって参照された固有制約が入っている SQL のスキーマの名前。
UNIQUE_CONSTRAINT_NAME	UNQNAME	VARCHAR(128)	参照制約によって参照された固有制約の名前。
MATCH_OPTION	MATCH	VARCHAR(7)	突き合わせオプション。 常に NONE になります。
UPDATE_RULE	UPDATE	VARCHAR(11)	UPDATE の規則。 • NO ACTION • RESTRICT
DELETE_RULE	DELETE	VARCHAR(11)	DELETE の規則。 • NO ACTION • CASCADE • SET NULL • SET DEFAULT • RESTRICT
COLUMN_COUNT	COLCOUNT	INTEGER	外部キーの中の列の数。

SYSROUTINEDEP

SYSROUTINEDEP ビューは、ルーチンの従属関係を記録します。次の表は、SYSROUTINEDEP ビューの列について説明しています。

表 113. SYSROUTINEDEP ビュー

列名	システム列名	データ・タイプ	説明
SPECIFIC_SCHEMA	SPECSCHEMA	VARCHAR(128)	ルーチン (関数) インスタンスのスキーマ名。
SPECIFIC_NAME	SPECNAME	VARCHAR(128)	ルーチン・インスタンスの特定名。
OBJECT_SCHEMA	BSCHEMA	VARCHAR(128)	該当のオブジェクトが入っている SQL スキーマの名前。
OBJECT_NAME	BNAME	VARCHAR(128)	該当のルーチンが従属しているオブジェクトの名前。
OBJECT_TYPE	BTYPE	CHAR(10)	ルーチンで参照されたオブジェクトのオブジェクト・タイプを示します。 **ALIAS** オブジェクトは別名です。 **FUNCTION** オブジェクトは関数です。 **INDEX** オブジェクトは索引です。 **PROCEDURE** オブジェクトはプロシージャーです。 **SCHEMA** オブジェクトはスキーマです。 **TABLE** オブジェクトは表です。 **TYPE** オブジェクトは 特殊タイプです。 **VIEW** オブジェクトはビューです。
PARM_SIGNATURE	SIGNATURE	VARCHAR(10000) ヌル可能	この列はルーチン・シグニチャーを識別します。 オブジェクトがルーチンでない場合は、ヌル値が入ります。
IASP_NUMBER	IASPNUMBER	SMALLINT	オブジェクトの独立補助記憶域プール (IASP) 番号を指定します。
NUMBER_OF_PARMS	NUMPARMS	SMALLINT ヌル可能	パラメーターの数を識別します。 オブジェクトがルーチンでない場合は、ヌル値が入ります。

SYSROUTINES

SYSROUTINES 表には、CREATE PROCEDURE ステートメントによって作成された各プロシージャーごとに、および CREATE FUNCTION ステートメントによって作成された各関数ごとに、行が 1 つずつ入ります。次の表は、SYSROUTINES 表の列について説明しています。

表 114. SYSROUTINES 表

列名	システム列名	データ・タイプ	説明
SPECIFIC_SCHEMA	SPECSCHEMA	VARCHAR(128)	ルーチン (関数) インスタンスのスキーマ名。
SPECIFIC_NAME	SPECNAME	VARCHAR(128)	ルーチン・インスタンスの特定名。
ROUTINE_SCHEMA	RTNSCHEMA	VARCHAR(128)	該当のルーチンが入っている SQL スキーマの名前。
ROUTINE_NAME	RTNNAME	VARCHAR(128)	ルーチンの名前。
ROUTINE_TYPE	RTNTYPE	VARCHAR(9)	ルーチンのタイプ。 **PROCEDURE** これはプロシージャーです。 **FUNCTION** これは関数です。
ROUTINE_CREATED	RTNCREATE	TIMESTAMP	ルーチンが作成されたときのタイム・スタンプを識別します。
ROUTINE_DEFINER	DEFINER	VARCHAR(128)	該当のルーチンを定義したユーザーの名前。
ROUTINE_BODY	BODY	VARCHAR(8)	ルーチン本体のタイプ: **EXTERNAL** これは外部ルーチンです。 **SQL** これは SQL ルーチンです。

表 114. SYSROUTINES 表 (続き)

列名	システム列名	データ・タイプ	説明
EXTERNAL_NAME	EXTNAME	VARCHAR(279) ヌル可能	これが外部ルーチンである場合は、この列は外部プログラム名を識別します。 • REXX の場合は、外部プログラム名は、*スキーマ名/ソース・ファイル名 (メンバー名)* です。 • ILE サービス・プログラムの場合、外部プログラム名はスキーマ名/*サービス・プログラム名 (入り口名)* です。 • Java プログラムの場合、外部プログラム名は任意選択の jar-id の後に *完全修飾クラス名!メソッド名* または *完全修飾クラス名.メソッド名* が続きます。 • その他のすべての言語では、外部プログラム名は、*スキーマ名/プログラム名* です。 これが外部ルーチンでない場合は、ヌル値が入ります。

表 114. SYSROUTINES 表 (続き)

列名	システム列名	データ・タイプ	説明
EXTERNAL_LANGUAGE	LANGUAGE	VARCHAR(8) ヌル可能	これが外部ルーチンである場合は、この列は外部プログラム名を識別します。 **C** 外部プログラムは C で作成されます。 **C++** 外部プログラムは C++ で作成されます。 **CL** 外部プログラムは CL で作成されます。 **COBOL** 外部プログラムは COBOL で作成されます。 **COBOLLE** 外部プログラムは ILE COBOL で作成されます。 **FORTRAN** 外部プログラムは FORTRAN で作成されます。 **JAVA** 外部プログラムは JAVA で作成されます。 **PLI** 外部プログラムは PL/I で作成されます。 **REXX** 外部プログラムは REXX プロシージャーです。 **RPG** 外部プログラムは RPG で作成されます。 **RPGLE** 外部プログラムは ILE RPG で作成されます。 これが外部ルーチンでない場合は、ヌル値が入ります。

表 114. SYSROUTINES 表 (続き)

列名	システム列名	データ・タイプ	説明
PARAMETER_STYLE	PARM_STYLE	VARCHAR(7) ヌル可能	これが外部ルーチンである場合は、この列はパラメーターのスタイル (呼び出し規則) を識別します。 **DB2GNRL** これは DB2GENERAL 呼び出し規則です。 **DB2SQL** これは DB2SQL 呼び出し規則です。 **GENERAL** これは GENERAL 呼び出し規則です。 **JAVA** これは JAVA 呼び出し規則です。 **NULLS** これは GENERAL WITH NULLS 呼び出し規則です。 **SQL** これは SQL 標準呼び出し規則です。 これが外部ルーチンでない場合は、ヌル値が入ります。
IS_DETERMINISTIC	DETERMINE	VARCHAR(3)	この列はルーチンが deterministic であるかどうかを識別します。 つまり、同じ引き数のルーチンに対する呼び出しが、常に同じ結果を戻すかどうかを識別します。 **NO** ルーチンは deterministic ではありません。 **YES** ルーチンは deterministic です。

SYSROUTINES

表 114. SYSROUTINES 表 (続き)

列名	システム列名	データ・タイプ	説明
SQL_DATA_ACCESS	DATAACCESS	VARCHAR(8)	この列は、ルーチンに SQL が含まれているか、およびルーチンがデータの読み取りまたは変更を行うかを識別します。 **NONE** ルーチンは SQL ステートメントを含みません。 **CONTAINS** ルーチンは SQL ステートメントを含みます。 **READS** ルーチンは、おそらく表またはビューからデータを読み取ります。 **MODIFIES** ルーチンは、おそらく表またはビュー内のデータを変更するか、SQL DDL ステートメントを発行します。
SQL_PATH	SQL_PATH	VARCHAR(3483) ヌル可能	これが SQL ルーチンの場合、この列はパスを識別します。 これが SQL ルーチンでない場合は、ヌル値が入ります。
PARM_SIGNATURE	SIGNATURE	VARCHAR(510)	この列はルーチン・シグニチャーを識別します。
NUMBER_OF_RESULTS	NUMRESULTS	SMALLINT	結果の数を識別します。
MAX_DYNAMIC_RESULT_SETS	RESULTS	SMALLINT	戻される結果セットの最大数を識別します。 0 は結果セットがないことを示します。
IN_PARMS	IN_PARMS	SMALLINT	入力パラメーターの数を識別します。 0 は入力パラメーターがないことを示します。
OUT_PARMS	OUT_PARMS	SMALLINT	出力パラメーターの数を識別します。 0 は出力パラメーターがないことを示します。
INOUT_PARMS	INOUT_PARM	SMALLINT	入出力パラメーターの数を識別します。 0 は入出力パラメーターがないことを示します。
PARSE_TREE	PARSE_TREE	VARCHAR(666) FOR BIT DATA	これがルーチンである場合、この列は CREATE FUNCTION ステートメントまたは CREATE PROCEDURE ステートメントの解析ツリーを識別します。これは内部的にしか使用されません。

表 114. SYSROUTINES 表 (続き)

列名	システム列名	データ・タイプ	説明
PARM_ARRAY	PARM_ARRAY	VARCHAR(10008) ビット・データ用	これが外部ルーチンである場合、この列は CREATE FUNCTION ステートメントまたは CREATE PROCEDURE ステートメントから構築されたパラメーター配列を識別します。これは内部的にしか使用されません。
LONG_COMMENT	REMARKS	VARCHAR(2000) ヌル可能	COMMENT ステートメントで指定された文字ストリング。 詳細コメントがない場合は、ヌル値が入ります。
ROUTINE_DEFINITION	ROUTINEDEF	DBCLOB(1048576) ヌル可能	これが SQL ルーチンの場合、この列は SQL ルーチン本体を含みます。 これが SQL ルーチンでない場合、または切り捨てなければルーチン本体をこの列に収容できない場合は、ヌル値が入ります。
FUNCTION_ORIGIN	ORIGIN	CHAR(1)	関数のタイプを識別します。これがプロシージャーの場合、この列にはブランクが入ります。 **B** これは組み込み関数 (DB2 UDB for iSeriesによって定義された) です。 **E** これはユーザー定義関数です。 **U** これは、他の関数をソースとしているユーザー定義関数です。 **S** これはシステム生成関数です。
FUNCTION_TYPE	TYPE	CHAR(1)	関数の形式を識別します。これがプロシージャーの場合、この列にはブランクが入ります。 **S** これはスカラー関数です。 **C** これは列関数です。 **T** これは表関数です。

表 114. SYSROUTINES 表 (続き)

列名	システム列名	データ・タイプ	説明
EXTERNAL_ACTION	EXTACTION	CHAR(1) ヌル可能	関数の呼び出しに外部的な作用があるかどうかを識別します。 **E** この関数には、外部的な副次作用があります。 **N** この関数には、外部的な副次作用はありません。 ルーチンがプロシージャーの場合は、ヌル値が入ります。
IS_NULL_CALL	NULL_CALL	VARCHAR(3) ヌル可能	入力パラメーターがヌル値である場合に、関数を呼び出す必要があるかどうかを識別します。 **NO** この関数は、入力パラメーターがヌル値の場合に呼び出す必要はありません。これがスカラー関数の場合は、いずれかのオペランドがヌルであれば、この関数の結果は暗黙的にヌルになります。これが表関数の場合は、いずれかのオペランドがヌル値であれば、この関数の結果は空の表になります。 **YES** この関数は、入力オペランドがヌルでも呼び出す必要があります。 ルーチンがプロシージャーの場合は、ヌル値が入ります。
SCRATCH_PAD	SCRATCHPAD	INTEGER ヌル可能	静的メモリー域 (スクラッチパッド) のアドレスが関数に渡されるかどうかを識別します。 **0** 関数にはスクラッチパッドはありません。 **整数** 関数に渡されるスクラッチパッドのサイズを示します。 ルーチンがプロシージャーの場合は、ヌル値が入ります。

表 114. SYSROUTINES 表 (続き)

列名	システム列名	データ・タイプ	説明
FINAL_CALL	FINAL_CALL	VARCHAR(3) ヌル可能	関数がその作業域 (スクラッチパッド) の終結処理を行えるようにするために、関数への最終呼び出しを行う必要があるかどうかを示します。 **NO** 最終呼び出しは行いません。 **YES** ステートメントが完了したときに関数への最終呼び出しを行います。 ルーチンがプロシージャーの場合は、ヌル値が入ります。
PARALLELIZABLE	PARALLEL	VARCHAR(3) ヌル可能	関数が並行して実行できるかどうかを識別します。 **NO** 関数は同期させなければなりません。 **YES** 関数は並行して実行できます。 ルーチンがプロシージャーの場合は、ヌル値が入ります。
DBINFO	DBINFO	VARCHAR(3) ヌル可能	データベースに関する情報をルーチンに渡すかどうかを識別します。 **NO** データベース情報をルーチンに渡しません。 **YES** データベースに関する情報をルーチンに渡します。 ルーチンがプロシージャーの場合は、ヌル値が入ります。
SOURCE_SPECIFIC_SCHEMA	SRCSCHEMA	VARCHAR(128) ヌル可能	これがソース化関数であり、ソースがユーザー定義の場合、この列にはソース・スキーマが入ります。これがソース化関数であり、ソースが組み込みである場合、この列には 'QSYS2' が入ります。 ルーチンがソース化関数でない場合は、ヌル値が入ります。
SOURCE_SPECIFIC_NAME	SRCNAME	VARCHAR(128) ヌル可能	これがソース化関数であり、ソースがユーザー定義である場合、この列にはソース関数名の特定名が入ります。 ルーチンがソース化関数でない場合は、ヌル値が入ります。

SYSROUTINES

表 114. SYSROUTINES 表 (続き)

列名	システム列名	データ・タイプ	説明
IS_USER_ DEFINED_CAST	CAST_FUNC	VARCHAR(3) ヌル可能	この関数が、特殊タイプの作成時に作成されたキャスト関数であるかどうかを判別します。 **NO** この関数はキャスト関数ではありません。 **YES** この関数はキャスト関数です。 ルーチンがプロシージャーの場合は、ヌル値が入ります。
CARDINALITY	CARD	BIGINT ヌル可能	表関数の基数を指定します。 この関数が表関数でない場合、または基数が指定されていなかった場合は、ヌル値が入ります。
FENCED	FENCED	VARCHAR(3) ヌル可能	関数を隔離するかどうかを指定します。 **NO** 関数を隔離しません。 **YES** 関数を隔離します。 ルーチンがプロシージャーの場合は、ヌル値が入ります。
COMMIT_ON_RETURN	CMTONRET	VARCHAR(3) ヌル可能	この列は、プロシージャーから正常に戻った時点でそのプロシージャーをコミットするかどうかを識別します。 **NO** プロシージャーから正常に戻ったときに、コミットは行われません。 **YES** プロシージャーから正常に戻ったときに、コミットが行われます。 ルーチンが関数の場合は、ヌル値が入ります。
IASP_NUMBER	IASPNUMBER	SMALLINT	独立補助記憶域プール (IASP) 番号を指定します。
NEW_SAVEPOINT_LEVEL	NEWSAVEPTL	VARCHAR(3) ヌル可能	この列は、ルーチンが新しい保管ポイント・レベルを開始するかどうかを識別します。 **NO** 新しい保管ポイント・レベルを開始しません。 **YES** 新しい保管ポイント・レベルを開始します。 ルーチンが関数の場合は、ヌル値が入ります。

表 114. SYSROUTINES 表 (続き)

列名	システム列名	データ・タイプ	説明
LAST_ALTERED	ALTEREDTS	TIMESTAMP ヌル可能	ルーチンの最後に変更されたタイム・スタンプ。 ヌル値が入ります。
DEBUG_MODE	DEBUG_MODE	CHAR(1)	ルーチンがデバッグ可能かどうかを識別します。 **0** ルーチンはデバッグ不可能です。 **2** ルーチンはデバッグ可能です。
DEBUG_DATA	DEBUG_DATA	CLOB(1048576) ヌル可能	予約済み。ヌル値が入ります。

SYSTABLES

SYSTABLES ビューには、SQL スキーマの中にある各表、ビューまたは別名 (SQL カタログの表およびビューを含む) ごとに行が 1 つずつ入ります。次の表は、SYSTABLES ビューの列について説明しています。

表 115. SYSTABLES ビュー

列名	システム列名	データ・タイプ	説明
TABLE_NAME	NAME	VARCHAR(128)	その表、ビュー、または別名の名前。SQL の表名、ビュー名、または別名が存在する場合は、その SQL の表名、ビュー名または別名です。存在しない場合は、システムの表名、ビュー名、または別名です。
TABLE_OWNER	CREATOR	VARCHAR(128)	表、ビュー、または別名の所有者。
TABLE_TYPE	TYPE	CHAR(1)	表、ビュー、または別名を記述する行の場合 : **A** 別名 **L** 論理ファイル **P** 物理ファイル **T** 表 **V** ビュー
COLUMN_COUNT	COLCOUNT	INTEGER	表またはビューの列の数。別名の場合はゼロです。
ROW_LENGTH	RECLENGTH[82]	INTEGER	表にあるレコードの最大長。別名の場合はゼロです。
TABLE_TEXT	LABEL	VARCHAR(50)	LABEL ステートメントで指定された文字ストリング。
LONG_COMMENT	REMARKS	VARCHAR(2000) ヌル可能	COMMENT ステートメントで指定された文字ストリング。 詳細コメントがない場合は、ヌル値が入ります。
TABLE_SCHEMA	DBNAME	VARCHAR(128)	該当の表、ビュー、または別名を含む SQL スキーマの名前。
LAST_ALTERED_TIMESTAMP	ALTEREDTS	TIMESTAMP	表の最後に変更されたタイム・スタンプ
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10)	システムの表名。
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10)	システムのスキーマ名
FILE_TYPE	FILETYPE	CHAR(1)	ファイル・タイプ **D** データ・ファイルまたは別名 **S** ソース・ファイル

表 115. SYSTABLES ビュー (続き)

列名	システム列名	データ・タイプ	説明
BASE_TABLE_SCHEMA	TBDBNAME	VARCHAR(128) ヌル可能	別名の場合、これは、その別名の基になっている表またはビューを含む SQL スキーマの名前です。 表が別名でない場合は、ヌル値が入ります。
BASE_TABLE_NAME	TBNAME	VARCHAR(128) ヌル可能	別名の場合、これは、その別名の基になっている表またはビューの名前です。 表が別名でない場合は、ヌル値が入ります。
BASE_TABLE_MEMBER	TBMEMBER	VARCHAR(10) ヌル可能	別名の場合、これは、その別名の基になっているファイル・メンバーの名前です。これが別名であって、メンバー名が指定されていなかった場合は、*FIRST が入ります。 表が別名でない場合は、ヌル値が入ります。
SYSTEM_TABLE	SYSTABLE	CHAR(1)	システム表 **N** 表は、システム表ではありません。 **Y** 表は、システム表です。
SELECT_OMIT	SELECTOMIT	CHAR(1)	選択/除外論理ファイル **N** 表は、選択/除外論理ファイルではありません。 **Y** 表は、選択/除外論理ファイルです。
IS_INSERTABLE_INTO	INSERTABLE	VARCHAR(3)	表で INSERT を使用できるかどうかを識別します。 **NO** この表では INSERT は使用できません。 **YES** この表では INSERT を使用できます。
IASP_NUMBER	IASPNUMBER	SMALLINT	独立補助記憶域プール (IASP) 番号を指定します。

82. 長さはデータベース・バッファーで渡されたバイトの数であり、内部記憶長さではありません。

SYSTRIGCOL

SYSTRIGCOL ビューには、WHEN 文節またはトリガーの起動された SQL ステートメントによって暗黙的または明示的に参照された各列ごとに行が 1 つずつ入ります。 次の表は、SYSTRIGCOL ビューの列について説明しています。

表 116. SYSTRIGCOL ビュー

列名	システム列名	データ・タイプ	説明
TRIGGER_SCHEMA	TRIGSCHEMA	VARCHAR(128)	トリガーが入っているスキーマの名前。
TRIGGER_NAME	TRIGNAME	VARCHAR(128)	トリガーの名前。
TABLE_SCHEMA	TABSCHEMA	VARCHAR(128)	トリガーで参照された列を含む表またはビューが入っているスキーマの名前。
TABLE_NAME	TABNAME	VARCHAR(128)	トリガーで参照された列を含む表またはビューの名前。
COLUMN_NAME	TABCOLUMN	VARCHAR(128)	トリガーで参照された列の名前。
OBJECT_TYPE	BTYPE	VARCHAR(10)	トリガーで参照された列が入っているオブジェクトのオブジェクト・タイプを示します。 **FUNCTION** オブジェクトは関数です。 **TABLE** オブジェクトは表です。 **VIEW** オブジェクトはビューです。

SYSTRIGDEP

SYSTRIGDEP ビューには、WHEN 文節またはトリガーの起動された SQL ステートメントによって参照された各列ごとに行が 1 つずつ入ります。次の表は、SYSTRIGDEP ビューの列について説明しています。

表 117. SYSTRIGDEP ビュー

列名	システム列名	データ・タイプ	説明
TRIGGER_SCHEMA	TRIGSCHEMA	VARCHAR(128)	トリガーが入っているスキーマの名前。
TRIGGER_NAME	TRIGNAME	VARCHAR(128)	トリガーの名前。
OBJECT_SCHEMA	BSCHEMA	VARCHAR(128)	トリガーで参照されたオブジェクトが入っているスキーマの名前。
OBJECT_NAME	BNAME	VARCHAR(128)	トリガーで参照されたオブジェクトの名前。
OBJECT_TYPE	BTYPE	CHAR(10)	トリガーで参照されたオブジェクトのオブジェクト・タイプを示します。 **ALIAS** オブジェクトは別名です。 **FUNCTION** オブジェクトは関数です。 **INDEX** オブジェクトは索引です。 **PACKAGE** オブジェクトはパッケージです。 **PROCEDURE** オブジェクトはプロシージャーです。 **SCHEMA** オブジェクトはスキーマです。 **TABLE** オブジェクトは表です。 **TYPE** オブジェクトは 特殊タイプです。 **VIEW** オブジェクトはビューです。
PARM_SIGNATURE	SIGNATURE	VARCHAR(10000) ヌル可能	この列はルーチン・シグニチャーを識別します。 オブジェクトがルーチンでない場合は、ヌル値が入ります。

SYSTRIGGERS

SYSTRIGGERS ビューには、SQL スキーマにある各トリガーごとに、行が 1 つずつ入ります。次の表は、SYSTRIGGERS ビューの列について説明しています。

表 118. SYSTRIGGERS ビュー

列名	システム列名	データ・タイプ	説明
TRIGGER_SCHEMA	TRIGSCHEMA	VARCHAR(128)	トリガーが入っているスキーマの名前。
TRIGGER_NAME	TRIGNAME	VARCHAR(128)	トリガーの名前。
EVENT_MANIPULATION	TRIGEVENT	VARCHAR(6)	トリガーを起動するイベントを示します。 **DELETE** DELETE 時にトリガーが起動されます。 **INSERT** INSERT 時にトリガーが起動されます。 **UPDATE** DELETE 時にトリガーが起動されます。 **READ** 行の読み取り時にトリガーが起動されます。 これは、ADDPFTRG コマンドによって作成されたトリガーに対してのみ有効です。
EVENT_OBJECT_SCHEMA	TABSCHEMA	VARCHAR(128)	トリガーの対象表が入っているスキーマの名前。
EVENT_OBJECT_TABLE	TABNAME	VARCHAR(128)	トリガーの対象表の名前。
ACTION_ORDER	ORDERSEQNO	INTEGER	表のトリガー・リスト内のこのトリガーの順位。これはトリガーが起動される順序を示します。
ACTION_CONDITION	CONDITION	DBCLOB(1048576) ヌル可能	トリガーの WHEN 文節のテキスト。 WHEN 文節がない場合は、ヌル値が入ります。
ACTION_STATEMENT	TEXT	DBCLOB(1048576) ヌル可能	トリガー・アクション内の SQL ステートメントのテキスト。 これが ADDPFTRG コマンドによって作成されたトリガーの場合は、ヌル値が入ります。

表 118. SYSTRIGGERS ビュー (続き)

列名	システム列名	データ・タイプ	説明
ACTION_ORIENTATION	GRANULAR	VARCHAR(9)	これが行トリガーであるか、ステートメント・トリガーであるかを示します。 **ROW** トリガーは各行ごとに起動されます。 **STATEMENT** トリガーは各ステートメントごとに起動されます。
ACTION_TIMING	TRIGTIME	VARCHAR(6)	これが前トリガーであるか、後トリガーであるかを示します。 **BEFORE** トリガーはトリガー・イベントの前に起動されます。 **AFTER** トリガーはトリガー・イベントの後に起動されます。
TRIGGER_MODE	TRIGMODE	VARCHAR(6)	トリガーの起動モードを示します。 **DB2SQL** トリガー・モードは DB2SQL です。 **DB2ROW** トリガー・モードは DB2ROW です。
ACTION_REFERENCE_OLD_ROW	OLD_ROW	VARCHAR(128) ヌル可能	OLD ROW 相関名。 OLD ROW 相関名が指定されていない場合は、ヌル値が入ります。
ACTION_REFERENCE_NEW_ROW	NEW_ROW	VARCHAR(128) ヌル可能	NEW ROW 相関名。 NEW ROW 相関名が指定されていない場合は、ヌル値が入ります。
ACTION_REFERENCE_OLD_TABLE	OLD_TABLE	VARCHAR(128) ヌル可能	OLD TABLE 相関名。 OLD TABLE 相関名が指定されていない場合は、ヌル値が入ります。
ACTION_REFERENCE_NEW_TABLE	NEW_TABLE	VARCHAR(128) ヌル可能	NEW TABLE 相関名。 NEW TABLE 相関名が指定されていない場合は、ヌル値が入ります。
SQL_PATH	SQL_PATH	VARCHAR(3483) ヌル可能	トリガーを作成するときに使用された SQL パス。 このトリガーが ADDPFTRG コマンドによって作成された場合は、ヌル値が入ります。
CREATED	CREATE_DTS	TIMESTAMP	トリガーが作成されたときのタイム・スタンプ。

SYSTRIGGERS

表 118. SYSTRIGGERS ビュー (続き)

列名	システム列名	データ・タイプ	説明
TRIGGER_PROGRAM_NAME	TRIGPGM	VARCHAR(128)	トリガー・プログラムの名前。
TRIGGER_PROGRAM_LIBRARY	TRIGPGMLIB	VARCHAR(128)	トリガー・プログラムが入っているスキーマのシステム名。
OPERATIVE	OPERATIVE	VARCHAR(1)	トリガーが作動可能かどうか (メンバーをもつファイルに関連付けられているかどうか) を示します。 **Y** トリガーが作動可能です。 **N** トリガーは作動不能です。
ENABLED	ENABLED	VARCHAR(1)	トリガーが使用可能かどうかを示します (CL コマンド CHGPFTRG を参照)。 **Y** トリガーは使用可能です。 **N** トリガーは使用不可です。
THREADSAFE	THDSAFE	VARCHAR(8)	トリガーがスレッド・セーフかどうかを示します。 **YES** トリガーがスレッド・セーフです。 **NO** トリガーはスレッド・セーフではありません。 **UNKNOWN** トリガーのスレッド・セーフティは不明です。
MULTITHREADED_JOB_ACTION	MLTTHDACN	VARCHAR(8)	マルチスレッド・ジョブでトリガー・プログラムが呼び出されたときに取るアクションを示します。 **SYSVAL** QMLTTHDACN システム値を使用して、取るアクションを判別します。 **MSG** マルチスレッド・ジョブでトリガー・プログラムを実行しますが、診断メッセージを送信します。 **NORUN** マルチスレッド・ジョブでトリガー・プログラムを実行しません。 **RUN** マルチスレッド・ジョブでトリガー・プログラムを実行します。

表 118. SYSTRIGGERS ビュー (続き)

列名	システム列名	データ・タイプ	説明
ALLOW_REPEATED_CHANGE	ALWREPCHG	VARCHAR(8)	更新イベントがトリガーを起動する条件を示します。 **YES** トリガーは同じ行に対する反復変更を許します。 **NO** トリガーは同じ行に対する反復変更を許します。
TRIGGER_UPDATE_CONDITION	TRGUPDCND	CHAR(8) ヌル可能	UPDATE トリガーは、更新イベント時には常に起動するのか、列値が実際に変更されているときにだけ起動するのかを示します。 **ALWAYS** トリガーは更新イベント時に常に起動されます。 **CHANGE** トリガーは、更新イベント時に列値が実際に変更されているときだけ起動します。 トリガーが UPDATE トリガーでない場合は、ヌル値が入ります。
LONG_COMMENT	REMARKS	VARGRAPHIC(2000) ヌル可能	COMMENT ステートメントで指定された文字ストリング。 詳細コメントがない場合は、ヌル値が入ります。

SYSTRIGUPD

SYSTRIGUPD ビューには、UPDATE 列リスト (存在する場合) に識別された各列ごとに行が 1 つずつ入ります。次の表は、SYSTRIGUPD ビューの列について説明しています。

表 119. SYSTRIGUPD ビュー

列名	システム列名	データ・タイプ	説明
TRIGGER_SCHEMA	TRIGSCHEMA	VARCHAR(128)	トリガーが入っているスキーマの名前。
TRIGGER_NAME	TRIGNAME	VARCHAR(128)	トリガーの名前。
EVENT_OBJECT_SCHEMA	TABSCHEMA	VARCHAR(128)	トリガーの対象表が入っているスキーマの名前。
EVENT_OBJECT_TABLE	TABNAME	VARCHAR(128)	トリガーの対象表の名前。
TRIGGERED_UPDATE_COLUMNS	TABCOLUMN	VARCHAR(128)	トリガーの UPDATE 列リストに指定された列の名前。

SYSTYPES

SYSTYPES 表には、CREATE DISTINCT TYPE ステートメントによって作成された各組み込みデータ・タイプおよび各特殊タイプごとに、行が 1 つずつ入ります。次の表は、SYSTYPES 表の列について説明しています。

表 120. SYSTYPES 表

列名	システム列名	データ・タイプ	説明
USER_DEFINED_TYPE_SCHEMA	TYPESCHEMA	VARCHAR(128)	データ・タイプのスキーマ名。
USER_DEFINED_TYPE_NAME	TYPENAME	VARCHAR(128)	データ・タイプの名前。
USER_DEFINED_TYPE_DEFINER	DEFINER	VARCHAR(128)	該当のデータ・タイプを作成したユーザーの名前。
SOURCE_SCHEMA	SRCSCHEMA	VARCHAR(128) ヌル可能	このデータ・タイプのソース・データ・タイプのスキーマ。 これが組み込みデータ・タイプである場合は、ヌル値が入ります。
SOURCE_TYPE	SRCTYPE	VARCHAR(128) ヌル可能	このデータ・タイプのソース・データ・タイプの名前。 これが組み込みデータ・タイプである場合は、ヌル値が入ります。
SYSTEM_TYPE_SCHEMA	SYSTSCHEMA	CHAR(10)	データ・タイプのシステム・スキーマ名。
SYSTEM_TYPE_NAME	SYSTNAME	CHAR(10)	データ・タイプのシステム名。
METATYPE	METATYPE	CHAR(1)	データ・タイプのタイプを識別します。 **S** システム事前定義データ・タイプ。 **T** ユーザー定義特殊タイプ。

SYSTYPES

表 120. SYSTYPES 表 (続き)

列名	システム列名	データ・タイプ	説明
LENGTH	LENGTH	INTEGER	データ・タイプの長さ属性。ただし、10 進数、数値、または非ゼロ精度 2 進数の列の場合は、その精度。 **8 バイト** BIGINT **4 バイト** INTEGER **2 バイト** SMALLINT **数値の精度** DECIMAL **数値の精度** NUMERIC **8 バイト** FLOAT、FLOAT(n) (ここで n = 25 〜 53)、または DOUBLE PRECISION **4 バイト** FLOAT(n) (ここで n = 1 〜 24)、または REAL **ストリングの長さ** CHARACTER **ストリングの最大長** VARCHAR または CLOB **グラフィック・ストリングの長さ** GRAPHIC **グラフィック・ストリングの最大長** VARGRAPHIC または DBCLOB **2 進ストリングの最大長** BLOB **4 バイト** DATE **3 バイト** TIME **10 バイト** TIMESTAMP **データ・リンクの最大長** **URL およびコメント** DATALINK **40 バイト** ROWID **ソース・タイプと同じ値** DISTINCT
NUMERIC_SCALE	SCALE	INTEGER ヌル可能	数値データの位取り データ・タイプが 10 進数、数値、または 2 進数でない場合は、ヌル値が入ります。

表 120. SYSTYPES 表 (続き)

列名	システム列名	データ・タイプ	説明
CCSID	CCSID	INTEGER ヌル可能	CHAR、VARCHAR、CLOB、DATE、TIME、TIMESTAMP、GRAPHIC、VARGRAPHIC、DBCLOB、および DATALINK データ・タイプの CCSID の値。 データ・タイプが数値の場合は、ヌル値が入ります。

SYSTYPES

表 120. SYSTYPES 表 (続き)

列名	システム列名	データ・タイプ	説明
STORAGE	STORAGE	INTEGER	列の記憶域所要量: **8 バイト** BIGINT **4 バイト** INTEGER **2 バイト** SMALLINT **(精度/2) + 1** DECIMAL **数値の精度** NUMERIC **8 バイト** FLOAT、FLOAT(n) (ここで n = 25 〜 53)、または DOUBLE PRECISION **4 バイト** FLOAT(n) (ここで n = 1 〜 24)、または REAL **ストリングの長さ** CHAR **ストリングの最大長 + 2** VARCHAR **ストリングの最大長 + 29** CLOB **ストリングの長さ * 2** GRAPHIC **ストリングの最大長 * 2 + 2** VARGRAPHIC **ストリングの最大長 * 2 + 29** DBCLOB **4 バイト** DATE **3 バイト** TIME **10 バイト** TIMESTAMP **データ・リンク URL およびコメントの最大長 + 24** DATALINK **42 バイト** ROWID **ソース・タイプと同じ値** DISTINCT **注:** この列には、すべてのデータ・タイプの記憶域所要量があります。

表 120. SYSTYPES 表 (続き)

列名	システム列名	データ・タイプ	説明
NUMERIC_PRECISION	PRECISION	INTEGER ヌル可能	すべての数値データ・タイプの精度。 **注:** この列では、すべての数値データ・タイプ (単精度および倍精度の浮動小数点数を含む) の精度を指定します。NUMERIC_PRECISION_RADIX 列は、この列の値が 2 進数であるか、または 10 進数であるかを示します。 データ・タイプが数値でない場合は、ヌル値が入ります。
CHARACTER_MAXIMUM_LENGTH	CHARLEN	INTEGER ヌル可能	データ・タイプが 2 進数、文字およびグラフィック・ストリングの場合は、ストリングの最大長。 データ・タイプがストリングでない場合は、ヌル値が入ります。
CHARACTER_OCTET_LENGTH	CHARBYTE	INTEGER ヌル可能	データ・タイプが 2 進数、文字およびグラフィック・ストリングの場合は、バイト数。 データ・タイプがストリングでない場合は、ヌル値が入ります。
ALLOCATE	ALLOCATE	INTEGER ヌル可能	データ・タイプが 2 進数、可変長文字、および可変長グラフィック・ストリングの場合は、ストリングの割り振り済みの長さ。 データ・タイプが数値または固定長の場合は、ヌル値が入ります。
NUMERIC_PRECISION_RADIX	RADIX	INTEGER ヌル可能	NUMERIC_PRECISION の列で指定される精度が、2 進数 と 10 進数のどちらの数値で指定されるかを指示します。 **2** 2 進数: 浮動小数点数の精度は 2 進数で指定されます。 **10** 10 進数: 他の数値タイプはすべて 10 進数で指定されます。 データ・タイプが数値でない場合は、ヌル値が入ります。

表 120. SYSTYPES 表 (続き)

列名	システム列名	データ・タイプ	説明
DATETIME_PRECISION	DATPRC	INTEGER ヌル可能	日付、時刻、またはタイム・スタンプの小数部分。 **0** データ・タイプが DATE および TIME の場合 **6** データ・タイプが TIMESTAMP の場合 (マイクロ秒数) データ・タイプが日付、時刻、またはタイム・スタンプでない場合は、ヌル値が入ります。
CREATE_TIME	CRTTIME	TIMESTAMP	データ・タイプが作成されたときのタイム・スタンプを識別します。
LONG_COMMENT	REMARKS	VARCHAR(2000) ヌル可能	COMMENT ステートメントで指定された文字ストリング。 詳細コメントがない場合は、ヌル値が入ります。
IASP_NUMBER	IASPNUMBER	SMALLINT	データ・タイプの独立補助記憶域プール (IASP) 番号を指定します。
LAST_ ALTERED	ALTEREDTS	TIMESTAMP ヌル可能	予約済み。ヌル値が入ります。

SYSVIEWDEP

SYSVIEWDEP ビューは、表に対するビューの従属関係 (SQL カタログのビューを含む) を記録します。次の表は、SYSVIEWDEP ビューの列について説明しています。

表 121. SYSVIEWDEP ビュー

列名	システム列名	データ・タイプ	説明
VIEW_NAME	DNAME	VARCHAR(128)	ビューの名前。 SQL ビュー名が存在する場合は、その SQL ビュー名です。存在しない場合は、システム・ビュー名です。
VIEW_OWNER	DCREATOR	VARCHAR(128)	ビューの所有者
OBJECT_NAME	ONAME	VARCHAR(128)	該当のビューが従属しているオブジェクトの名前。
OBJECT_SCHEMA	OSCHEMA	VARCHAR(128)	該当のビューが従属しているオブジェクトが入っている SQL スキーマの名前。
OBJECT_TYPE	OTYPE	CHAR(10)	該当のビューの基となったオブジェクトのタイプ: **FUNCTION** 機能 **TABLE** 表 **TYPE** 特殊タイプ **VIEW** ビュー
VIEW_SCHEMA	DDBNAME	VARCHAR(128)	ビューのスキーマ名
SYSTEM_VIEW_NAME	SYS_VNAME	CHAR(10)	システム・ビュー名
SYSTEM_VIEW_SCHEMA	SYS_VDNAME	CHAR(10)	システム・ビュー・スキーマ
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10) ヌル可能	システム表名。 オブジェクトが関数または特殊タイプの場合は、ヌル値が入ります。
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10) ヌル可能	システム表スキーマ。 オブジェクトが関数または特殊タイプの場合は、ヌル値が入ります。
TABLE_NAME	BNAME	VARCHAR(128) ヌル可能	該当のビューが従属している表またはビューの名前。 SQL ビュー名が存在する場合は、その SQL ビュー名です。存在しない場合は、システム・ビュー名です。 オブジェクトが関数または特殊タイプの場合は、ヌル値が入ります。

SYSVIEWDEP

表 121. SYSVIEWDEP ビュー (続き)

列名	システム列名	データ・タイプ	説明
TABLE_OWNER	BCREATOR	VARCHAR(128) ヌル可能	該当のビューが従属している表またはビューの所有者。 オブジェクトが関数または特殊タイプの場合は、ヌル値が入ります。
TABLE_SCHEMA	BDBNAME	VARCHAR(128) ヌル可能	該当のビューが従属している表、またはビューが入っている SQL のスキーマの名前。 オブジェクトが関数または特殊タイプの場合は、ヌル値が入ります。
TABLE_TYPE	BTYPE	CHAR(1) ヌル可能	該当のビューの基となったオブジェクトのタイプ: **T** 表 **P** 物理ファイル **V** ビュー **L** 論理ファイル オブジェクトが関数または特殊タイプの場合は、ヌル値が入ります。
IASP_NUMBER	IASPNUMBER	SMALLINT	独立補助記憶域プール (IASP) 番号を指定します。
PARM_SIGNATURE	SIGNATURE	VARCHAR(10000) ヌル可能	この列はルーチン・シグニチャーを識別します。 オブジェクトがルーチンでない場合は、ヌル値が入ります。

SYSVIEWS

SYSVIEWS ビューには、SQL のスキーマにある各ビュー (SQL カタログのビューを含む) ごとに、行が 1 つずつ入ります。 次の表は、SYSVIEWS ビューの列について説明しています。

表 122. SYSVIEWS ビュー

列名	システム列名	データ・タイプ	説明
TABLE_NAME	NAME	VARCHAR(128)	ビューの名前。 SQL ビュー名が存在する場合は、その SQL ビュー名です。存在しない場合は、システム・ビュー名です。
VIEW_OWNER	CREATOR	VARCHAR(128)	ビューの所有者
SEQNO	SEQNO	INTEGER	該当の行の順序番号。常に 1 になります。
CHECK_OPTION	CHECK	CHAR(1)	該当のビューに対して使用された検査オプション **N** 検査オプションは指定されませんでした **Y** ローカル・オプションが指定されました **C** カスケード・オプションが指定されました
VIEW_DEFINITION	TEXT	VARCHAR(10000) ヌル可能	CREATE VIEW ステートメントの QUERY 式の部分。 切り捨てなければビュー定義を列に収容できない場合は、ヌル値が入ります。
IS_UPDATABLE	UPDATES	CHAR(1)	ビューが更新可能かどうかを指定します。 **Y** 更新可能なビューです **N** 読み取り専用のビューです
TABLE_SCHEMA	DBNAME	VARCHAR(128)	該当のビューが入っている SQL のスキーマの名前。
SYSTEM_VIEW_NAME	SYS_VNAME	CHAR(10)	システム・ビュー名
SYSTEM_VIEW_SCHEMA	SYS_VDNAME	CHAR(10)	システム・ビュー・スキーマ名
IS_INSERTABLE_INTO	INSERTABLE	VARCHAR(3)	ビューで INSERT を使用できるかどうかを識別します。 **NO** このビューでは INSERT は使用できません。 **YES** このビューでは INSERT を使用できます。
IASP_NUMBER	IASPNUMBER	SMALLINT	独立補助記憶域プール (IASP) 番号を指定します。

ODBC および JDBC のカタログ・ビュー

カタログには、SYSIBM ライブラリー内にある以下のビューおよび表が含まれます。

ビュー名	説明
1005 ページの『SQLCOLPRIVILEGES』	列に対して認可された特権についての情報
1006 ページの『SQLCOLUMNS』	列属性についての情報
1011 ページの『SQLFOREIGNKEYS』	外部キーについての情報
1012 ページの『SQLPRIMARYKEYS』	基本キーについての情報
1013 ページの『SQLPROCEDURECOLS』	プロシージャー・パラメーターについての情報
1018 ページの『SQLPROCEDURES』	プロシージャーについての情報
1019 ページの『SQLSCHEMAS』	スキーマについての情報
1020 ページの『SQLSPECIALCOLUMNS』	行を一意的に識別するために使用できる表の列についての情報
1022 ページの『SQLSTATISTICS』	表についての統計情報
1023 ページの『SQLTABLEPRIVILEGES』	表に認可された特権についての情報
1024 ページの『SQLTABLES』	表についての情報
1025 ページの『SQLTYPEINFO』	表のタイプについての情報
1031 ページの『SQLUDTS』	組み込みのデータ・タイプおよび特殊タイプについての情報

SQLCOLPRIVILEGES

SQLCOLPRIVILEGES ビューには、列に対して認可された各特権ごとに、行が 1 つずつ入ります。このカタログ・ビューを使用して、特定ユーザーが特定の列に対する権限を持っているかどうかを判別することはできないので、注意してください。なぜなら、列を使用するための特権は、グループ・ユーザー・プロファイルまたは特殊権限 (*ALLOBJ など) を通じて獲得できるからです。次の表は、ビューの列について説明しています。

表 123. SQLCOLPRIVILEGES ビュー

列名	データ・タイプ	説明
TABLE_CAT	VARCHAR(128)	リレーショナル・データベース名
TABLE_SCHEM	VARCHAR(128)	該当の表が入っている SQL のスキーマの名前。
TABLE_NAME	VARCHAR(128)	表名
COLUMN_NAME	VARCHAR(128)	列名
GRANTOR	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
GRANTEE	VARCHAR(128)	特権を認可する対象のユーザー・プロファイル。
PRIVILEGE	VARCHAR(10)	認可される特権 : **UPDATE** 列を更新する特権。 **REFERENCES** 参照制約モードで列を参照する特権。
IS_GRANTABLE	VARCHAR(3)	特権を他のユーザーに認可できるかどうかを示します。 **NO** 特権は認可できません。 **YES** 特権を認可できます。
DBNAME	VARCHAR(8) ヌル可能	予約済み。列にはヌル値が入ります。

SQLCOLUMNS

SQLCOLUMNS ビューには、表、ビュー、または別名の中の各列ごとに、行が 1 つずつ入ります。次の表は、ビューの列について説明しています。

表 124. SQLCOLUMNS ビュー

列名	データ・タイプ	説明	
TABLE_CAT	VARCHAR(128)	リレーショナル・データベース名	
TABLE_SCHEM	VARCHAR(128)	該当の表が入っている SQL のスキーマの名前。	
TABLE_NAME	VARCHAR(128)	表名	
COLUMN_NAME	VARCHAR(128)	列名	
DATA_TYPE	SMALLINT	列のデータ・タイプ。	
		-5	BIGINT
		4	INTEGER
		5	SMALLINT
		3	DECIMAL
		2	NUMERIC
		8	DOUBLE PRECISION
		7	REAL
		1	CHARACTER
		-2	CHARACTER FOR BIT DATA
		12	VARCHAR
		-3	VARCHAR FOR BIT DATA
		40	CLOB
		-95	GRAPHIC
		-96	VARGRAPHIC
		-350	DBCLOB
		30	BLOB
		9	DATE
		10	TIME
		11	TIMESTAMP
		70	DATALINK
		-100	ROWID
		17	DISTINCT

表 124. SQLCOLUMNS ビュー (続き)

列名	データ・タイプ	説明
TYPE_NAME	VARCHAR(128)	列のデータ・タイプの名前。 **BIGINT** BIGINT **INTeger** INTEGER **SMALLINT** SMALLINT **DECIMAL** DECIMAL **NUMERIC** NUMERIC **FLOAT** DOUBLE PRECISION **REAL** REAL **CHARacter** CHARACTER **CHARacter FOR BIT DATA** CHARACTER FOR BIT DATA **VARCHAR** VARCHAR **VARCHAR FOR BIT DATA** VARCHAR FOR BIT DATA **CLOB** CLOB **GRAPHIC** GRAPHIC **VARGRAPHIC** VARGRAPHIC **DBCLOB** DBCLOB **BLOB** BLOB **DATE** DATE **TIME** TIME **TIMESTAMP** TIMESTAMP **DATALINK** DATALINK **ROWID** ROWID **修飾タイプ名** DISTINCT
COLUMN_SIZE	INTEGER	列の長さです。
BUFFER_LENGTH	INTEGER	バッファー内の列の長さを示します。
DECIMAL_DIGITS	SMALLINT ヌル可能	数値列の桁数を示します。 オブジェクトが数値でない場合は、ヌル値が入ります。
NUM_PREC_RADIX	SMALLINT ヌル可能	数値列の基数を示します。 オブジェクトが数値でない場合は、ヌル値が入ります。
NULLABLE	SMALLINT	列にヌル値を入れることができるかどうかを示します。 **0** この列ではヌルは許されません。 **1** この列ではヌルが許されます。

表 124. SQLCOLUMNS ビュー (続き)

列名	データ・タイプ	説明
REMARKS	VARCHAR(2000) ヌル可能	COMMENT ステートメントで指定された文字ストリング。 詳細コメントがない場合は、ヌル値が入ります。
COLUMN_DEF	VARCHAR(2000) ヌル可能	列のデフォルト値。 デフォルト値がない場合は、ヌル値が入ります。
SQL_DATA_TYPE	SMALLINT	列の SQL データ・タイプを示します。
SQL_DATETIME_SUB	SMALLINT ヌル可能	データ・タイプの日時サブタイプ: **1** DATE **2** TIME **3** TIMESTAMP 列が日時データ・タイプでない場合は、ヌル値が入ります。
CHAR_OCTET_LENGTH	INTEGER ヌル可能	列の長さを文字数で示します。 列がストリングでない場合は、ヌル値が入ります。
ORDINAL_POSITION	INTEGER	表内の列の順序位置を示します。
IS_NULLABLE	VARCHAR(3)	列にヌル値を入れることができるかどうかを示します。 **NO** 列はヌル可能ではありません。 **YES** 列はヌル可能です。

表 124. SQLCOLUMNS ビュー (続き)

列名	データ・タイプ	説明
JDBC_DATA_TYPE	SMALLINT	列の JDBC データ・タイプを示します。
		-5 BIGINT
		4 INTEGER
		5 SMALLINT
		3 DECIMAL
		2 NUMERIC
		8 DOUBLE PRECISION
		7 REAL
		1 CHARACTER
		-2 CHARACTER FOR BIT DATA
		12 VARCHAR
		-3 VARCHAR FOR BIT DATA
		2005 CLOB
		1 GRAPHIC
		12 VARGRAPHIC
		1111 DBCLOB
		2004 BLOB
		91 DATE
		92 TIME
		93 TIMESTAMP
		70 DATALINK
		1111 ROWID
		2001 DISTINCT
SCOPE_CATALOG	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
SCOPE_SCHEMA	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
SCOPE_TABLE	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
SOURCE_DATA_TYPE	VARCHAR(128) ヌル可能	列のデータ・タイプが特殊データ・タイプである場合は、ソース・データ・タイプ。 データ・タイプが特殊タイプでない場合は、ヌル値が入ります。
DBNAME	VARCHAR(8) ヌル可能	予約済み。ヌル値が入ります。
COLUMN_TEXT	VARCHAR(50) ヌル可能	列のテキスト。 列テキストがない場合は、ヌル値が入ります。

表 124. SQLCOLUMNS ビュー (続き)

列名	データ・タイプ	説明
PSEUDO_COLUMN	SMALLINT	これが ROWID (行 ID) であるか、識別列であるかを示します。 **1** 列は、ROWID または識別列ではありません。 **2** 列は、ROWID または識別列です。

SQLFOREIGNKEYS

SQLFOREIGNKEYS ビューには、表の各参照制約キーごとに、行が 1 つずつ入ります。次の表は、ビューの列について説明しています。

表 125. SQLFOREIGNKEYS ビュー

列名	データ・タイプ	説明
PKTABLE_CAT	VARCHAR(128)	リレーショナル・データベース名
PKTABLE_SCHEM	VARCHAR(128)	親表が入っている SQL スキーマの名前。
PKTABLE_NAME	VARCHAR(128)	親表の名前。
PKCOLUMN_NAME	VARCHAR(128)	親キーの列名。
FKTABLE_CAT	VARCHAR(128)	リレーショナル・データベース名
FKTABLE_SCHEM	VARCHAR(128)	参照制約の従属表が入っている SQL スキーマの名前。
FKTABLE_NAME	VARCHAR(128)	参照制約の従属表名。
FKCOLUMN_NAME	VARCHAR(128)	従属キー名。
KEY_SEQ	SMALLINT	キー内における列の位置。
UPDATE_RULE	SMALLINT	UPDATE の規則。 **1** RESTRICT **3** NO ACTION
DELETE_RULE	SMALLINT	削除規則: **0** CASCADE **1** RESTRICT **2** SET NULL **3** NO ACTION **4** SET DEFAULT
FK_NAME	VARCHAR(128)	参照制約の名前。
PK_NAME	VARCHAR(128)	固有制約の名前。
DEFERRABILITY	SMALLINT	制約の検査が据え置きできるかどうかを示します。 常に 7 になります。

SQLPRIMARYKEYS

SQLPRIMARYKEYS ビューには、表の各基本制約キーごとに、行が 1 つずつ入ります。次の表は、ビューの列について説明しています。

表 126. SQLPRIMARYKEYS ビュー

列名	データ・タイプ	説明
TABLE_CAT	VARCHAR(128)	リレーショナル・データベース名
TABLE_SCHEM	VARCHAR(128)	基本キーをもつ表が入っているスキーマの名前。
TABLE_NAME	VARCHAR(128)	基本キーをもつ表の名前。
COLUMN_NAME	VARCHAR(128)	基本キー列の名前。
KEY_SEQ	SMALLINT	キー内における列の位置。
PK_NAME	VARCHAR(128)	基本キー制約の名前。

SQLPROCEDURECOLS

SQLPROCEDURECOLS ビューには、プロシージャーの各パラメーターごとに、行が 1 つずつ入ります。次の表は、ビューの列について説明しています。

表 127. SQLPROCEDURECOLS ビュー

列名	データ・タイプ	説明
PROCEDURE_CAT	VARCHAR(128)	リレーショナル・データベース名
PROCEDURE_SCHEM	VARCHAR(128)	プロシージャー・インスタンスのスキーマ名。
PROCEDURE_NAME	VARCHAR(128)	プロシージャー・インスタンスの名前。
COLUMN_NAME	VARCHAR(128) ヌル可能	プロシージャー・パラメーターの名前。 パラメーターに名前がない場合は、ヌル値が入ります。
COLUMN_TYPE	SMALLINT	パラメーターのタイプ: **1** IN **2** INOUT **4** OUT
DATA_TYPE	SMALLINT	パラメーターのデータ・タイプ。 **-5** BIGINT **4** INTEGER **5** SMALLINT **3** DECIMAL **2** NUMERIC **8** DOUBLE PRECISION **7** REAL **1** CHARACTER **-2** CHARACTER FOR BIT DATA **12** VARCHAR **-3** VARCHAR FOR BIT DATA **40** CLOB **-95** GRAPHIC **-96** VARGRAPHIC **-350** DBCLOB **30** BLOB **9** DATE **10** TIME **11** TIMESTAMP **70** DATALINK **-100** ROWID **17** DISTINCT

表 127. SQLPROCEDURECOLS ビュー (続き)

列名	データ・タイプ	説明
TYPE_NAME	VARCHAR(260)	パラメーターのデータ・タイプの名前。 **BIGINT** BIGINT **INTeger** INTEGER **SMALLINT** SMALLINT **DECIMAL** DECIMAL **NUMERIC** NUMERIC **FLOAT** DOUBLE PRECISION **REAL** REAL **CHARacter** CHARACTER **CHARacter FOR BIT DATA** CHARACTER FOR BIT DATA **VARCHAR** VARCHAR **VARCHAR FOR BIT DATA** VARCHAR FOR BIT DATA **CLOB** CLOB **GRAPHIC** GRAPHIC **VARGRAPHIC** VARGRAPHIC **DBCLOB** DBCLOB **BLOB** BLOB **DATE** DATE **TIME** TIME **TIMESTAMP** TIMESTAMP **DATALINK** DATALINK **ROWID** ROWID **修飾タイプ名** DISTINCT
COLUMN_SIZE	INTEGER	パラメーターの長さ。
BUFFER_LENGTH	INTEGER	バッファー内のパラメーターの長さを示します。
DECIMAL_DIGITS	SMALLINT ヌル可能	数値データまたは日時データの位取り。 パラメーターが 10 進数、数値、2 進数、時刻、またはタイム・スタンプでない場合は、ヌル値が入ります。

表 127. SQLPROCEDURECOLS ビュー (続き)

列名	データ・タイプ	説明
NUM_PREC_RADIX	SMALLINT ヌル可能	NUMERIC_PRECISION の列で指定される精度が、2 進数 と 10 進数のどちらの数値で指定されるかを指示します。 **2** 2 進数: 浮動小数点数の精度は 2 進数で指定されます。 **10** 10 進数: 他の数値タイプはすべて 10 進数で指定されます。 パラメーターが数値パラメーターでない場合は、ヌル値が入ります。
NULLABLE	SMALLINT	パラメーターがヌル可能かどうかを示します。 **0** パラメーターにヌルは許されません。 **1** パラメーターにヌルが許されます。
REMARKS	VARCHAR(2000) ヌル可能	COMMENT ステートメントで指定された文字ストリング。 詳細コメントがない場合は、ヌル値が入ります。
COLUMN_DEF	VARCHAR(1) ヌル可能	列のデフォルト値。 デフォルト値がない場合は、ヌル値が入ります。

表 127. SQLPROCEDURECOLS ビュー (続き)

列名	データ・タイプ	説明	
SQL_DATA_TYPE	SMALLINT	パラメーターの SQL データ・タイプ。	
		-5	BIGINT
		4	INTEGER
		5	SMALLINT
		3	DECIMAL
		2	NUMERIC
		8	DOUBLE PRECISION
		7	REAL
		1	CHARACTER
		-2	CHARACTER FOR BIT DATA
		12	VARCHAR
		-3	VARCHAR FOR BIT DATA
		-99	CLOB
		-95	GRAPHIC
		-96	VARGRAPHIC
		-350	DBCLOB
		-98	BLOB
		9	DATE
		10	TIME
		11	TIMESTAMP
		70	DATALINK
		-100	ROWID
		17	DISTINCT
SQL_DATETIME_SUB	SMALLINT ヌル可能	パラメーターの日時サブタイプ。	
		1	DATE
		2	TIME
		3	TIMESTAMP
		データ・タイプが日時データ・タイプでない場合は、ヌル値が入ります。	
CHAR_OCTET_LENGTH	INTEGER ヌル可能	パラメーターの長さを文字数で示します。 列がストリングでない場合は、ヌル値が入ります。	
ORDINAL_POSITION	INTEGER	パラメーター・リストにおける該当のパラメーターの数値位置 (左から右への順序)。	
IS_NULLABLE	VARCHAR(3)	パラメーターがヌル可能かどうかを示します。	
		NO	パラメーターにヌルは許されません。
		YES	パラメーターにヌルが許されます。

表 127. SQLPROCEDURECOLS ビュー (続き)

列名	データ・タイプ	説明	
JDBC_DATA_TYPE	SMALLINT	パラメーターの JDBC データ・タイプ。	
		-5	BIGINT
		4	INTEGER
		5	SMALLINT
		3	DECIMAL
		2	NUMERIC
		8	DOUBLE PRECISION
		7	REAL
		1	CHARACTER
		-2	CHARACTER FOR BIT DATA
		12	VARCHAR
		-3	VARCHAR FOR BIT DATA
		2005	CLOB
		1	GRAPHIC
		12	VARGRAPHIC
		1111	DBCLOB
		2004	BLOB
		91	DATE
		92	TIME
		93	TIMESTAMP
		70	DATALINK
		1111	ROWID
		2001	DISTINCT

SQLPROCEDURES

SQLPROCEDURES ビューには、各プロシージャーごとに行が 1 つずつ入ります。次の表は、ビューの列について説明しています。

表 128. SQLPROCEDURES ビュー

列名	データ・タイプ	説明
PROCEDURE_CAT	VARCHAR(128)	リレーショナル・データベース名
PROCEDURE_SCHEM	VARCHAR(128)	プロシージャー・インスタンスのスキーマ名
PROCEDURE_NAME	VARCHAR(128)	プロシージャーの名前。
NUM_INPUT_PARAMS	SMALLINT	入力パラメーターの数を識別します。 0 は入力パラメーターがないことを示します。
NUM_OUTPUT_PARAMS	SMALLINT	出力パラメーターの数を識別します。 0 は出力パラメーターがないことを示します。
NUM_RESULT_SETS	SMALLINT	戻される結果セットの最大数を識別します。 0 は結果セットがないことを示します。
REMARKS	VARCHAR(2000) ヌル可能	COMMENT ステートメントで指定された文字ストリング。 詳細コメントがない場合は、ヌル値が入ります。
PROCEDURE_TYPE	SMALLINT	予約済み。 0 が入ります。
NUM_INOUT_PARAMS	SMALLINT	入出力パラメーターの数を識別します。 0 は入出力パラメーターがないことを示します。

SQLSCHEMAS

SQLSCHEMAS ビューには、各スキーマごとの行が 1 つずつ入ります。次の表は、ビューの列について説明しています。

表 129. SQLSCHEMAS ビュー

列名	データ・タイプ	説明
TABLE_CAT	VARCHAR(128)	リレーショナル・データベース名
TABLE_SCHEM	VARCHAR(128)	スキーマの名前。
TABLE_NAME	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
TABLE_TYPE	VARCHAR(128)	予約済み。ヌル値が入ります。 ヌル可能
REMARKS	VARCHAR(2000)	予約済み。ヌル値が入ります。 ヌル可能
DBNAME	VARCHAR(8)	予約済み。ヌル値が入ります。 ヌル可能
SCHEMA_TEXT	VARCHAR(50)	スキーマを記述する文字ストリング。 テキストがない場合は、空ストリングが入ります。

SQLSPECIALCOLUMNS

SQLSPECIALCOLUMNS ビューには、表の 1 行を識別できる基本キー、固有制約、または固有索引の各列ごとに、行が 1 つずつ入ります。次の表は、ビューの列について説明しています。

表 130. SQLSPECIALCOLUMNS ビュー

列名	データ・タイプ	説明	
SCOPE	SMALLINT	予約済み。 0 が入ります。	
COLUMN_NAME	VARCHAR(128)	列名	
DATA_TYPE	SMALLINT	列のデータ・タイプ。	
		-5	BIGINT
		4	INTEGER
		5	SMALLINT
		3	DECIMAL
		2	NUMERIC
		8	DOUBLE PRECISION
		7	REAL
		1	CHARACTER
		-2	CHARACTER FOR BIT DATA
		12	VARCHAR
		-3	VARCHAR FOR BIT DATA
		40	CLOB
		-95	GRAPHIC
		-96	VARGRAPHIC
		-350	DBCLOB
		30	BLOB
		9	DATE
		10	TIME
		11	TIMESTAMP
		70	DATALINK
		-100	ROWID
		17	DISTINCT
TYPE_NAME	VARCHAR(260)	列のデータ・タイプの名前。	
COLUMN_SIZE	INTEGER	列の長さです。	
BUFFER_LENGTH	INTEGER	バッファー内の列の長さを示します。	
DECIMAL_DIGITS	SMALLINT ヌル可能	数値列の桁数を示します。 列が数値の列でない場合は、ヌル値が入ります。	

表 130. SQLSPECIALCOLUMNS ビュー (続き)

列名	データ・タイプ	説明
PSEUDO_COLUMN	SMALLINT	これが ROWID (行 ID) であるか、識別列であるかを示します。
		1 列は、ROWID または識別列ではありません。
		2 列は、ROWID または識別列です。
TABLE_CAT	VARCHAR(128)	リレーショナル・データベース名
TABLE_SCHEM	VARCHAR(128)	該当の表が入っている SQL のスキーマの名前。
TABLE_NAME	VARCHAR(128)	表の名前。
NULLABLE	SMALLINT	列にヌル値を入れることができるかどうかを示します。
		0 列はヌル可能ではありません。
		1 列はヌル可能です。
JDBC_DATA_TYPE	SMALLINT	列の JDBC データ・タイプを示します。
		-5 BIGINT
		4 INTEGER
		5 SMALLINT
		3 DECIMAL
		2 NUMERIC
		8 DOUBLE PRECISION
		7 REAL
		1 CHARACTER
		-2 CHARACTER FOR BIT DATA
		12 VARCHAR
		-3 VARCHAR FOR BIT DATA
		2005 CLOB
		1 GRAPHIC
		12 VARGRAPHIC
		1111 DBCLOB
		2004 BLOB
		91 DATE
		92 TIME
		93 TIMESTAMP
		70 DATALINK
		1111 ROWID
		2001 DISTINCT

SQLSTATISTICS

SQLSTATISTICS ビューには、表についての統計情報が入ります。次の表は、ビューの列について説明しています。

表 131. SQLSTATISTICS ビュー

列名	データ・タイプ	説明
TABLE_CAT	VARCHAR(128)	リレーショナル・データベース名
TABLE_SCHEM	VARCHAR(128)	該当の表が入っている SQL スキーマの名前。
TABLE_NAME	VARCHAR(128)	表の名前。
NON_UNIQUE	SMALLINT ヌル可能	同一索引の重複キーを表で禁止するかどうかを示します。 TYPE が 0 の場合は、ヌル値が入ります。
INDEX_QUALIFIER	VARCHAR(128) ヌル可能	索引のスキーマ名 TYPE が 0 の場合は、ヌル値が入ります。
INDEX_NAME	VARCHAR(128) ヌル可能	索引の名前。 TYPE が 0 の場合は、ヌル値が入ります。
TYPE	SMALLINT	戻される情報のタイプを示します。 **0** 表の行数。 **3** 表の索引。
ORDINAL_POSITION	SMALLINT ヌル可能	索引内のキーの順序位置を示します。 TYPE が 0 の場合は、ヌル値が入ります。
COLUMN_NAME	VARCHAR(128) ヌル可能	索引内のキーに対応する列の名前。 TYPE が 0 の場合は、ヌル値が入ります。
ASC_OR_DESC	CHAR(1) ヌル可能	キー内における列の順序: **A** 昇順 **D** 降順 TYPE が 0 の場合は、ヌル値が入ります。
CARDINALITY	INTEGER ヌル可能	予約済み。ヌル値が入ります。
PAGES	INTEGER ヌル可能	予約済み。ヌル値が入ります。
FILTER_CONDITION	VARCHAR(128) ヌル可能	索引が選択/除外索引かどうかを示します。 **空ストリング** これは選択/除外索引です。 TYPE が 0 の場合、またはこれが選択/除外索引でない場合は、ヌル値が入ります。

SQLTABLEPRIVILEGES

SQLTABLEPRIVILEGES ビューには、表に対して認可された各特権ごとに、行が 1 つずつ入ります。次の表は、ビューの列について説明しています。

表 132. SQLTABLEPRIVILEGES ビュー

列名	データ・タイプ	説明
TABLE_CAT	VARCHAR(128)	リレーショナル・データベース名
TABLE_SCHEM	VARCHAR(128)	該当の表が入っている SQL スキーマの名前。
TABLE_NAME	VARCHAR(128)	表の名前。
GRANTOR	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
GRANTEE	VARCHAR(128)	特権を認可する対象のユーザー・プロファイル。
PRIVILEGE	VARCHAR(10)	認可される特権 : **ALTER** 表を変更する特権。 **DELETE** 表から行を削除する特権。 **INDEX** 表の索引を作成する特権。 **INSERT** 表に行を挿入する特権。 **REFERENCES** 参照制約の中で表を参照する特権。 **SELECT** 表から行を選択する特権。 **UPDATE** 表を更新する特権。
IS_GRANTABLE	VARCHAR(3)	特権を他のユーザーに認可できるかどうかを示します。 **NO** 特権は認可できません。 **YES** 特権を認可できます。
DBNAME	VARCHAR(8) ヌル可能	予約済み。ヌル値が入ります。

SQLTABLES

SQLTABLES ビューには、各表、ビュー、および別名ごとに、行が 1 つずつ入ります。次の表は、ビューの列について説明しています。

表 133. SQLTABLES ビュー

列名	データ・タイプ	説明
TABLE_CAT	VARCHAR(128)	リレーショナル・データベース名
TABLE_SCHEM	VARCHAR(128)	該当の表が入っているスキーマの名前。
TABLE_NAME	VARCHAR(128)	表の名前。
TABLE_TYPE	VARCHAR(10)	表のタイプを識別します。 **ALIAS** 表は別名です。 **TABLE** 表は、SQL 表または物理ファイルです。 **VIEW** 表は、SQL ビューまたは論理ファイルです。
REMARKS	VARCHAR(128) ヌル可能	COMMENT ステートメントで指定された文字ストリング。 詳細コメントがない場合は、ヌル値が入ります。
TYPE_CAT	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
TYPE_SCHEM	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
TYPE_NAME	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
SELF_REF_COL_NAME	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
REF_GENERATION	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
DBNAME	VARCHAR(8) ヌル可能	予約済み。ヌル値が入ります。
TABLE_TEXT	VARCHAR(50)	LABEL ステートメントで指定された文字ストリング。

SQLTYPEINFO

SQLTYPEINFO ビューには、各組み込みデータ・タイプごとに、行が 1 つずつ入ります。次の表は、ビューの列について説明しています。

表 134. SQLTYPEINFO ビュー

列名	データ・タイプ	説明	
TYPE_NAME	VARCHAR(128)	組み込みデータ・タイプの名前:	
		BIGINT	BIGINT
		INTeger	INTEGER
		SMALLINT	SMALLINT
		DECIMAL	DECIMAL
		NUMERIC	NUMERIC
		FLOAT	DOUBLE PRECISION
		REAL	REAL
		CHARacter	CHARACTER
		CHARacter FOR BIT DATA	CHARACTER FOR BIT DATA
		VARCHAR	VARCHAR
		VARCHAR FOR BIT DATA	VARCHAR FOR BIT DATA
		CLOB	CLOB
		GRAPHIC	GRAPHIC
		VARGRAPHIC	VARGRAPHIC
		DBCLOB	DBCLOB
		BLOB	BLOB
		DATE	DATE
		TIME	TIME
		TIMESTAMP	TIMESTAMP
		DATALINK	DATALINK
		ROWID	ROWID

SQLTYPEINFO

表 134. SQLTYPEINFO ビュー (続き)

列名	データ・タイプ	説明	
DATA_TYPE	SMALLINT	列のデータ・タイプ。	
		-5	BIGINT
		4	INTEGER
		5	SMALLINT
		3	DECIMAL
		2	NUMERIC
		8	DOUBLE PRECISION
		7	REAL
		1	CHARACTER
		-2	CHARACTER FOR BIT DATA
		12	VARCHAR
		-3	VARCHAR FOR BIT DATA
		40	CLOB
		-95	GRAPHIC
		-96	VARGRAPHIC
		-350	DBCLOB
		30	BLOB
		9	DATE
		10	TIME
		11	TIMESTAMP
		70	DATALINK
		-100	ROWID
COLUMN_SIZE	INTEGER	データ・タイプの最大長。	
LITERAL_PREFIX	VARCHAR(128) ヌル可能	ストリング・リテラルのプレフィックスを示します。 データ・タイプがストリングでない場合は、ヌル値が入ります。	
LITERAL_SUFFIX	VARCHAR(128) ヌル可能	ストリング・リテラルのサフィックスを示します。 データ・タイプがストリングでない場合は、ヌル値が入ります。	

表 134. SQLTYPEINFO ビュー (続き)

列名	データ・タイプ	説明
CREATE_PARAMS	VARCHAR(128) ヌル可能	データ・タイプでサポートされるパラメーターを示します。 **LENGTH** パラメーターは、長さです。すべてのストリング・データ・タイプおよび DATALINK に対して戻されます。 **PRECISION,SCALE** パラメーターには、精度および位取りが含まれます。すべての DECIMAL および NUMERIC データ・タイプに対して戻されます。 その他のすべてのデータ・タイプの場合は、ヌル値が入ります。
NULLABLE	SMALLINT	データ・タイプがヌル可能かどうかを示します。 **0** このデータ・タイプではヌルは許されません。 **1** このデータ・タイプではヌルが許されます。
CASE_SENSITIVE	SMALLINT	データ・タイプで、大文字小文字が区別されるかどうかを示します。 **0** このデータ・タイプでは大文字小文字は区別されません。 **1** このデータ・タイプでは、大文字小文字が区別されます。
SEARCHABLE	SMALLINT	データ・タイプを述部で使用できるかどうかを示します。 **0** このデータ・タイプは述部では使用できません。 **2** このデータ・タイプは LIKE 述部以外のすべての述部で使用できます。 **3** このデータ・タイプは、LIKE 述部を含め、すべての述部で使用できます。
UNSIGNED_ATTRIBUTE	SMALLINT ヌル可能	数値データ・タイプが符号付きか符号なしかを示します。 **0** データ・タイプは符号付きです。 **1** データ・タイプは符号なしです。 データ・タイプが数値でない場合は、ヌル値が入ります。
FIXED_PREC_SCALE	SMALLINT	データ・タイプに固定した精度および位取りがあるかどうかを示します。 **0** データ・タイプには、固定した精度および位取りはありません。 **1** データ・タイプには、固定した精度および位取りがあります。

SQLTYPEINFO

表 134. SQLTYPEINFO ビュー (続き)

列名	データ・タイプ	説明
AUTO_UNIQUE_VALUE	SMALLINT ヌル可能	数値データ・タイプが自動増分かどうかを示します。 **0** データ・タイプは自動増分ではありません。 **1** データ・タイプは自動増分です。 データ・タイプが数値でない場合は、ヌル値が入ります。
LOCAL_TYPE_NAME	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
MINIMUM_SCALE	SMALLINT ヌル可能	数値データ・タイプの最小位取りを示します。 データ・タイプが数値でない場合は、ヌル値が入ります。
MAXIMUM_SCALE	SMALLINT ヌル可能	数値データ・タイプの最大位取りを示します。 データ・タイプが数値でない場合は、ヌル値が入ります。
SQL_DATA_TYPE	SMALLINT	データ・タイプの SQL データ・タイプ値を示します。 **-5** BIGINT **4** INTEGER **5** SMALLINT **3** DECIMAL **2** NUMERIC **8** DOUBLE PRECISION **7** REAL **1** CHARACTER **-2** CHARACTER FOR BIT DATA **12** VARCHAR **-3** VARCHAR FOR BIT DATA **-99** CLOB **-95** GRAPHIC **-96** VARGRAPHIC **-350** DBCLOB **-98** BLOB **9** DATE **10** TIME **11** TIMESTAMP **70** DATALINK **-100** ROWID

表 134. SQLTYPEINFO ビュー (続き)

列名	データ・タイプ	説明
SQL_DATETIME_SUB	SMALLINT ヌル可能	データ・タイプの日時サブタイプ: **1** DATE **2** TIME **3** TIMESTAMP データ・タイプが日時データ・タイプでない場合は、ヌル値が入ります。
NUM_PREC_RADIX	INTEGER ヌル可能	NUMERIC_PRECISION の列で指定される精度が、2 進数 と 10 進数のどちらの数値で指定されるかを指示します。 **2** 2 進数: 浮動小数点数の精度は 2 進数で指定されます。 **10** 10 進数: 他の数値タイプはすべて 10 進数で指定されます。 パラメーターが数値パラメーターでない場合は、ヌル値が入ります。
INTERVAL_PRECISION	SMALLINT ヌル可能	予約済み。ヌル値が入ります。

SQLTYPEINFO

表 134. SQLTYPEINFO ビュー (続き)

列名	データ・タイプ	説明	
JDBC_DATA_TYPE	SMALLINT	データ・タイプの JDBC データ・タイプ値:	
		-5	BIGINT
		4	INTEGER
		5	SMALLINT
		3	DECIMAL
		2	NUMERIC
		8	DOUBLE PRECISION
		7	REAL
		1	CHARACTER
		-2	CHARACTER FOR BIT DATA
		12	VARCHAR
		-3	VARCHAR FOR BIT DATA
		2005	CLOB
		1	GRAPHIC
		12	VARGRAPHIC
		1111	DBCLOB
		2004	BLOB
		91	DATE
		92	TIME
		93	TIMESTAMP
		70	DATALINK
		1111	ROWID

SQLUDTS

SQLUDTS ビューには、各特殊タイプごとに、行が 1 つずつ入ります。次の表は、ビューの列について説明しています。

表 135. SQLUDTS ビュー

列名	データ・タイプ	説明
TYPE_CAT	VARCHAR(128)	リレーショナル・データベース名
TYPE_SCHEM	VARCHAR(128)	ユーザー定義タイプが入っているスキーマの名前。
TYPE_NAME	VARCHAR(128)	ユーザー定義タイプの名前。
CLASS_NAME	VARCHAR(20)	ユーザー定義タイプの Java クラス名。 **java.math.BigInteger** BIGINT **java.lang.Integer** INTEGER **java.lang.Short** SMALLINT **java.math.BigDecimal** DECIMAL **java.sql.BigDecimal** NUMERIC **java.lang.Double** DOUBLE PRECISION **java.lang.Float** REAL **java.lang.String** CHARACTER **byte[]** CHARACTER FOR BIT DATA **java.lang.String** VARCHAR **byte[]** VARCHAR FOR BIT DATA **java.sql.Clob** CLOB **java.lang.String** GRAPHIC **java.lang.String** VARGRAPHIC **java.sql.Clob** DBCLOB **java.sql.Blob** BLOB **java.sql.Date** DATE **java.sql.Time** TIME **java.sql.Timestamp** TIMESTAMP **java.net.URL** DATALINK **byte[]** ROWID
DATA_TYPE	SMALLINT	予約済み。 2001 が入ります。

表 135. SQLUDTS ビュー (続き)

列名	データ・タイプ	説明	
BASE_TYPE	SMALLINT	ユーザー定義のデータ・タイプのソース・データ・タイプ:	
		-5	BIGINT
		4	INTEGER
		5	SMALLINT
		3	DECIMAL
		2	NUMERIC
		8	DOUBLE PRECISION
		7	REAL
		1	CHARACTER
		-2	CHARACTER FOR BIT DATA
		12	VARCHAR
		-3	VARCHAR FOR BIT DATA
		2005	CLOB
		1	GRAPHIC
		12	VARGRAPHIC
		1111	DBCLOB
		2004	BLOB
		91	DATE
		92	TIME
		93	TIMESTAMP
		70	DATALINK
		1111	ROWID
REMARKS	VARCHAR(2000) ヌル可能	COMMENT ステートメントで指定された文字ストリング。 コメントがない場合は、ヌル値が入ります。	

ANS および ISO のカタログ・ビュー

一部の ANS および ISO のカタログ・ビューには、2 つのバージョンがあります。本書に記載してあるバージョンは、正規セットの ANS および ISO のビューです。2 番目のセットのビューは、18 文字以下に名前が制限されているもので、本書にはビュー名のみを示してあります。

ANS および ISO カタログには、QSYS2 ライブラリー内にある以下の表が含まれます。

ビュー名	短いビュー名	説明
1056 ページの『SQL_FEATURES』		データベース・マネージャーでサポートされている機能についての情報
1057 ページの『SQL_LANGUAGES』	SQL_LANGUAGES_S	サポートされている言語についての情報
1059 ページの『SQL_SIZING』		データベース・マネージャーでサポートされている限度についての情報

ANS および ISO カタログには、SYSIBM ライブラリー内にある以下のビューおよび表が含まれます。

ビュー名	短いビュー名	説明
1034 ページの『CHARACTER_SETS』	CHARACTER_SETS_S	サポートされている CCSID についての情報
1035 ページの『CHECK_CONSTRAINTS』		検査制約についての情報
1036 ページの『COLUMNS』	COLUMNS_S	列についての情報
1040 ページの『INFORMATION_SCHEMA_CATALOG_NAME』	CATALOG_NAME	リレーショナル・データベースについての情報
1041 ページの『PARAMETERS』	PARAMETERS_S	プロシージャー・パラメーターについての情報
1045 ページの『REFERENTIAL_CONSTRAINTS』	REF_CONSTRAINTS	参照制約についての情報
1046 ページの『ROUTINES』	ROUTINES_S	ルーチンについての情報
1055 ページの『SCHEMATA』	SCHEMATA_S	スキーマについての統計情報
1060 ページの『TABLE_CONSTRAINTS』		制約についての情報
1061 ページの『TABLES』	TABLES_S	表についての情報
1062 ページの『USER_DEFINED_TYPES』	UDT_S	特殊タイプについての情報
1066 ページの『VIEWS』		ビューについての情報

CHARACTER_SETS

CHARACTER_SETS ビューには、サポートされている各 CCSID ごとに、行が 1 つずつ入ります。次の表は、ビューの列について説明しています。

表 136. CHARACTER_SETS ビュー

列名	データ・タイプ	説明
CHARACTER_SET_CATALOG	VARCHAR(128)	リレーショナル・データベース名
CHARACTER_SET_SCHEMA	VARCHAR(128)	文字セットのスキーマ名。 'SYSIBM' が入ります。
CHARACTER_SET_NAME	VARCHAR(128)	文字セット名。
FORM_OF_USE	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
NUMBER_OF_CHARACTERS	INTEGER ヌル可能	予約済み。ヌル値が入ります。
DEFAULT_COLLATE_CATALOG	VARCHAR(128)	予約済み。リレーショナル・データベース名が入ります。
DEFAULT_COLLATE_SCHEMA	VARCHAR(128)	予約済み。 SYSIBM が入ります。
DEFAULT_COLLATE_NAME	VARCHAR(128)	予約済み。 IBMDEFAULT が入ります。

CHECK_CONSTRAINTS

CHECK_CONSTRAINTS ビューには、各検査制約ごとに、行が 1 つずつ入ります。次の表は、ビューの列について説明しています。

表 137. CHECK_CONSTRAINTS ビュー

列名	データ・タイプ	説明
CONSTRAINT_CATALOG	VARCHAR(128)	リレーショナル・データベース名
CONSTRAINT_SCHEMA	VARCHAR(128)	該当の制約が入っているスキーマの名前
CONSTRAINT_NAME	VARCHAR(128)	制約の名前
CHECK_CLAUSE	VARCHAR(2000) ヌル可能	検査制約文節のテキスト 切り捨てなければ検査文節を列に収容できない場合は、ヌル値が入ります。

COLUMNS

COLUMNS ビューには、各列ごとに行が 1 つずつ入ります。次の表は、ビューの列について説明しています。

表 138. COLUMNS ビュー

列名	データ・タイプ	説明
TABLE_CATALOG	VARCHAR(128)	リレーショナル・データベース名
TABLE_SCHEMA	VARCHAR(128)	該当の表またはビューが入っている SQL スキーマの名前。
TABLE_NAME	VARCHAR(128)	該当の列が入っている表またはビューの名前。
COLUMN_NAME	VARCHAR(128)	列の名前
ORDINAL_POSITION	INTEGER	表またはビューにおける該当の列の数値位置 (左から右への順序)
COLUMN_DEFAULT	VARCHAR(2000) ヌル可能	列のデフォルト値が存在する場合は、そのデフォルト値。列のデフォルト値が、切り捨てなければ表示できない場合は、その列の値はストリング 'TRUNCATED' になります。デフォルト値は文字形式で保管されます。以下の特殊値も存在します。 **CURRENT_DATE** デフォルト値は、現在の日付です。 **CURRENT_TIME** デフォルト値は、現在の時刻です。 **CURRENT_TIMESTAMP** デフォルト値は、現在のタイム・スタンプです。 **NULL** デフォルト値は、ヌル値です。 **USER** デフォルト値は、現在のジョブ・ユーザーです。 列がデフォルト値をもたない場合は、ヌル値が入ります。例えば、列に IDENTITY 属性が指定されている場合、または列が行 ID である場合です。
IS_NULLABLE	VARCHAR(3)	列にヌル値を入れることができるかどうかを示します。 **NO** 列にはヌル値を入れることはできません。 **YES** 列にはヌル値を入れることができます。

表 138. COLUMNS ビュー (続き)

列名	データ・タイプ	説明
DATA_TYPE	VARCHAR(128)	列のタイプ: **BIGINT** 大整数 **INTEGER** 長整数 **SMALLINT** 短整数 **DECIMAL** パック 10 進数 **NUMERIC** ゾーン 10 進数 **DOUBLE PRECISION** 倍精度浮動小数点数 **REAL** 単精度浮動小数点数 **CHARACTER** 固定長文字ストリング **CHARACTER VARYING** 可変長文字ストリング **CHARACTER LARGE OBJECT** 文字ラージ・オブジェクト・ストリング **GRAPHIC** 固定長グラフィック・ストリング **GRAPHIC VARYING** 可変長グラフィック・ストリング **DOUBLE-BYTE CHARACTER LARGE OBJECT** 2 バイト文字ラージ・オブジェクト・ストリング **BINARY LARGE OBJECT** バイナリー・ラージ・オブジェクト・ストリング **DATE** 日付 **TIME** 時刻 **TIMESTAMP** タイム・スタンプ **DATALINK** データ・リンク **ROWID** 行 ID **USER-DEFINED** 特殊タイプ
CHARACTER_MAXIMUM_LENGTH	INTEGER ヌル可能	データ・タイプが 2 進数、文字およびグラフィック・ストリングの場合は、ストリングの最大長。 列がストリングでない場合は、ヌル値が入ります。
CHARACTER_OCTET_LENGTH	INTEGER ヌル可能	データ・タイプが 2 進数、文字およびグラフィック・ストリングの場合は、バイト数。 列がストリングでない場合は、ヌル値が入ります。

表 138. COLUMNS ビュー (続き)

列名	データ・タイプ	説明
NUMERIC_PRECISION	INTEGER ヌル可能	数値の列すべての精度。 **注:** この列では、すべての数値データ・タイプ (単精度および倍精度の浮動小数点数を含む) の精度を指定します。 NUMERIC_PRECISION_RADIX 列は、この列の値が 2 進数であるか、または 10 進数であるかを示します。 列が数値の列でない場合は、ヌル値が入ります。
NUMERIC_PRECISION_RADIX	INTEGER ヌル可能	NUMERIC_PRECISION の列で指定される精度が、2 進数 と 10 進数のどちらの数値で指定されるかを指示します。 **2** 2 進数: 浮動小数点数の精度は 2 進数で指定されます。 **10** 10 進数: 他の数値タイプはすべて 10 進数で指定されます。 列が数値の列でない場合は、ヌル値が入ります。
NUMERIC_SCALE	INTEGER ヌル可能	数値データの位取り 列が 10 進数、数値、または 2 進数でない場合は、ヌル値が入ります。
DATETIME_PRECISION	INTEGER ヌル可能	日付、時刻、またはタイム・スタンプの小数部分。 **0** データ・タイプが DATE および TIME の場合 **6** データ・タイプが TIMESTAMP の場合 (マイクロ秒数) 列が日付、時刻、またはタイム・スタンプの列でない場合は、ヌル値が入ります。
INTERVAL_TYPE	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
INTERVAL_PRECISION	INTEGER ヌル可能	予約済み。ヌル値が入ります。
CHARACTER_SET_CATALOG	VARCHAR(128) ヌル可能	リレーショナル・データベース名 列がストリングでない場合は、ヌル値が入ります。
CHARACTER_SET_SCHEMA	VARCHAR(128) ヌル可能	文字セットのスキーマ名。 SYSIBM が入ります。 列がストリングでない場合は、ヌル値が入ります。
CHARACTER_SET_NAME	VARCHAR(128) ヌル可能	文字セット名。 列がストリングでない場合は、ヌル値が入ります。
COLLATION_CATALOG	VARCHAR(128) ヌル可能	リレーショナル・データベース名 列がストリングでない場合は、ヌル値が入ります。

表 138. COLUMNS ビュー (続き)

列名	データ・タイプ	説明
COLLATION_SCHEMA	VARCHAR(128) ヌル可能	照合のスキーマ。 SYSIBM が入ります。 列がストリングでない場合は、ヌル値が入ります。
COLLATION_NAME	VARCHAR(128) ヌル可能	照合名。 IBMBINARY が入ります。 列がストリングでない場合は、ヌル値が入ります。
DOMAIN_CATALOG	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
DOMAIN_SCHEMA	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
DOMAIN_NAME	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
UDT_CATALOG	VARCHAR(128) ヌル可能	これが特殊タイプの場合、リレーショナル・データベース。 これが特殊タイプでない場合は、ヌル値が入ります。
UDT_SCHEMA	VARCHAR(128) ヌル可能	これが特殊タイプの場合、スキーマの名前。 これが特殊タイプでない場合は、ヌル値が入ります。
UDT_NAME	VARCHAR(128) ヌル可能	特殊タイプ の名前。 これが特殊タイプでない場合は、ヌル値が入ります。
SCOPE_CATALOG	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
SCOPE_SCHEMA	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
SCOPE_NAME	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
MAXIMUM_CARDINALITY	INTEGER ヌル可能	予約済み。ヌル値が入ります。
DTD_IDENTIFIER	VARCHAR(128) ヌル可能	列の固有の内部 ID。
IS_SELF_REFERENCING	VARCHAR(3)	予約済み。 'NO' が入ります。

INFORMATION_SCHEMA_CATALOG_NAME

INFORMATION_SCHEMA_CATALOG_NAME ビューには、リレーショナル・データベースに対応する行が 1 つ入ります。次の表は、ビューの列について説明しています。

表 139. INFORMATION_SCHEMA_CATALOG_NAME ビュー

列名	データ・タイプ	説明
CATALOG_NAME	VARCHAR(128)	リレーショナル・データベース名

PARAMETERS

PARAMETERS ビューには、リレーショナル・データベース内のルーチンの各パラメーターごとに、行が 1 つずつ入ります。次の表は、ビューの列について説明しています。

表 140. PARAMETERS ビュー

列名	データ・タイプ	説明
SPECIFIC_CATALOG	VARCHAR(128)	リレーショナル・データベース名
SPECIFIC_SCHEMA	VARCHAR(128)	ルーチン (関数) インスタンスのスキーマ名
SPECIFIC_NAME	VARCHAR(128)	ルーチン・インスタンスの特定名。
ORDINAL_POSITION	INTEGER	パラメーター・リストにおける該当のパラメーターの数値位置 (左から右への順序)。
PARAMETER_MODE	VARCHAR(5)	パラメーターのタイプ: **IN** これは入力パラメーターです。 **OUT** これは出力パラメーターです。 **INOUT** これは入出力パラメーターです。
IS_RESULT	VARCHAR(3)	予約済み。 'NO' が入ります。
AS_LOCATOR	VARCHAR(3)	パラメーターがロケーターとして指定されたかどうかを識別します。 **NO** パラメーターはロケーターとして指定されませんでした。 **YES** パラメーターはロケーターとして指定されました。
PARAMETER_NAME	VARCHAR(128) ヌル可能	パラメーターの名前 パラメーターに名前がない場合は、ヌル値が入ります。
FROM_SQL_SPECIFIC_CATALOG	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
FROM_SQL_SPECIFIC_SCHEMA	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
FROM_SQL_SPECIFIC_NAME	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
TO_SQL_SPECIFIC_CATALOG	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
TO_SQL_SPECIFIC_SCHEMA	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
TO_SQL_SPECIFIC_NAME	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。

PARAMETERS

表 140. PARAMETERS ビュー (続き)

列名	データ・タイプ	説明
DATA_TYPE	VARCHAR(128) ヌル可能	パラメーターのタイプ: **BIGINT** 大整数 **INTEGER** 長整数 **SMALLINT** 短整数 **DECIMAL** パック 10 進数 **NUMERIC** ゾーン 10 進数 **DOUBLE PRECISION** 浮動小数点数; DOUBLE PRECISION **REAL** 浮動小数点数; REAL **CHARACTER** 固定長文字ストリング **CHARACTER VARYING** 可変長文字ストリング **CHARACTER LARGE OBJECT** 文字ラージ・オブジェクト・ストリング **GRAPHIC** 固定長グラフィック・ストリング **GRAPHIC VARYING** 可変長グラフィック・ストリング **DOUBLE-BYTE CHARACTER LARGE OBJECT** 2 バイト文字ラージ・オブジェクト・ストリング **BINARY LARGE OBJECT** バイナリー・ラージ・オブジェクト・ストリング **DATE** 日付 **TIME** 時刻 **TIMESTAMP** タイム・スタンプ **DATALINK** データ・リンク **ROWID** 行 ID **USER-DEFINED** 特殊タイプ
CHARACTER_MAXIMUM_LENGTH	INTEGER ヌル可能	データ・タイプが 2 進数、文字およびグラフィック・ストリングの場合は、ストリングの最大長。 パラメーターがストリングでない場合は、ヌル値が入ります。

表 140. PARAMETERS ビュー (続き)

列名	データ・タイプ	説明
CHARACTER_OCTET_LENGTH	INTEGER ヌル可能	データ・タイプが 2 進数、文字およびグラフィック・ストリングの場合は、バイト数。 パラメーターがストリングでない場合は、ヌル値が入ります。
CHARACTER_SET_CATALOG	VARCHAR(128) ヌル可能	リレーショナル・データベース名 列がストリングでない場合は、ヌル値が入ります。
CHARACTER_SET_SCHEMA	VARCHAR(128) ヌル可能	文字セットのスキーマ名。 'SYSIBM' が入ります。 列がストリングでない場合は、ヌル値が入ります。
CHARACTER_SET_NAME	VARCHAR(128) ヌル可能	文字セット名。 列がストリングでない場合は、ヌル値が入ります。
COLLATION_CATALOG	VARCHAR(128) ヌル可能	リレーショナル・データベース名 列がストリングでない場合は、ヌル値が入ります。
COLLATION_SCHEMA	VARCHAR(128) ヌル可能	照合のスキーマ。 SYSIBM が戻されます。 列がストリングでない場合は、ヌル値が入ります。
COLLATION_NAME	VARCHAR(128) ヌル可能	照合名。 IBMBINARY が戻されます。 列がストリングでない場合は、ヌル値が入ります。
NUMERIC_PRECISION	INTEGER ヌル可能	数値パラメーターすべての精度。 **注:** この列では、すべての数値データ・タイプ (単精度および倍精度の浮動小数点数を含む) の精度を指定します。 NUMERIC_PRECISION_RADIX 列は、この列の値が 2 進数であるか、または 10 進数であるかを示します。 パラメーターが数値パラメーターでない場合は、ヌル値が入ります。
NUMERIC_PRECISION_RADIX	INTEGER ヌル可能	NUMERIC_PRECISION の列で指定される精度が、2 進数 と 10 進数のどちらの数値で指定されるかを指示します。 **2** 2 進数: 浮動小数点数の精度は 2 進数で指定されます。 **10** 10 進数: 他の数値タイプはすべて 10 進数で指定されます。 パラメーターが数値パラメーターでない場合は、ヌル値が入ります。
NUMERIC_SCALE	INTEGER ヌル可能	数値データの位取り 10 進数、数値、または 2 進数のパラメーターでない場合は、ヌル値が入ります。

PARAMETERS

表 140. PARAMETERS ビュー (続き)

列名	データ・タイプ	説明
DATETIME_PRECISION	INTEGER ヌル可能	日付、時刻、またはタイム・スタンプの小数部分。 **0** データ・タイプが DATE および TIME の場合 **6** データ・タイプが TIMESTAMP の場合 (マイクロ秒数) パラメーターが日付、時刻、またはタイム・スタンプでない場合は、ヌル値が入ります。
INTERVAL_TYPE	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
INTERVAL_PRECISION	INTEGER ヌル可能	予約済み。ヌル値が入ります。
UDT_CATALOG	VARCHAR(128) ヌル可能	これが特殊タイプの場合は、リレーショナル・データベース名。 これが特殊タイプでない場合は、ヌル値が入ります。
UDT_SCHEMA	VARCHAR(128) ヌル可能	これが特殊タイプの場合は、スキーマの名前。 これが特殊タイプでない場合は、ヌル値が入ります。
UDT_NAME	VARCHAR(128) ヌル可能	特殊タイプ の名前。 これが特殊タイプでない場合は、ヌル値が入ります。
SCOPE_CATALOG	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
SCOPE_SCHEMA	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
SCOPE_NAME	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
MAXIMUM_CARDINALITY	INTEGER ヌル可能	予約済み。ヌル値が入ります。
DTD_IDENTIFIER	VARCHAR(128) ヌル可能	パラメーターの固有の内部 ID。

REFERENTIAL_CONSTRAINTS

REFERENTIAL_CONSTRAINTS ビューには、各参照制約ごとに、行が 1 つずつ入ります。次の表は、ビューの列について説明しています。

表 141. REFERENTIAL_CONSTRAINTS ビュー

列名	データ・タイプ	説明
CONSTRAINT_CATALOG	VARCHAR(128)	リレーショナル・データベース名
CONSTRAINT_SCHEMA	VARCHAR(128)	該当の制約が入っているスキーマの名前。
CONSTRAINT_NAME	VARCHAR(128)	制約の名前。
UNIQUE_CONSTRAINT_CATALOG	VARCHAR(128)	参照制約によって参照された固有制約が入っているリレーショナル・データベースの名前。
UNIQUE_CONSTRAINT_SCHEMA	VARCHAR(128)	参照制約によって参照された固有制約が入っているSQL のスキーマの名前。
UNIQUE_CONSTRAINT_NAME	VARCHAR(128)	参照制約によって参照された固有制約の名前。
MATCH_OPTION	VARCHAR(7)	予約済み。 'NONE' が入ります。
UPDATE_RULE	VARCHAR(11)	UPDATE の規則。 • NO ACTION • RESTRICT
DELETE_RULE	VARCHAR(11)	DELETE の規則。 • NO ACTION • CASCADE • SET NULL • SET DEFAULT • RESTRICT

ROUTINES

ROUTINES ビューには、各ルーチンごとに行が 1 つずつ入ります。次の表は、ビューの列について説明しています。

表 142. ROUTINES ビュー

列名	データ・タイプ	説明
SPECIFIC_CATALOG	VARCHAR(128)	リレーショナル・データベース名
SPECIFIC_SCHEMA	VARCHAR(128)	ルーチン (関数) インスタンスのスキーマ名。
SPECIFIC_NAME	VARCHAR(128)	ルーチンの特定名。
ROUTINE_CATALOG	VARCHAR(128)	リレーショナル・データベース名
ROUTINE_SCHEMA	VARCHAR(128)	該当のルーチンが入っている SQL スキーマの名前。
ROUTINE_NAME	VARCHAR(128)	ルーチンの名前。
ROUTINE_TYPE	VARCHAR(15)	ルーチンのタイプ。 **PROCEDURE** これはプロシージャーです。 **FUNCTION** これは関数です。 **INSTANCE METHOD** これは特殊タイプ用に作成された組み込みデータ・タイプ関数です。
MODULE_CATALOG	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
MODULE_SCHEMA	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
MODULE_NAME	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
UDT_CATALOG	VARCHAR(128) ヌル可能	リレーショナル・データベース名 これが INSTANCE METHOD でない場合は、ヌル値が入ります。
UDT_SCHEMA	VARCHAR(128) ヌル可能	この関数に関連した特殊タイプが入っている SQL スキーマの名前。 これが INSTANCE METHOD でない場合は、ヌル値が入ります。
UDT_NAME	VARCHAR(128) ヌル可能	この関数に関連した特殊タイプの名前。 これが INSTANCE METHOD でない場合は、ヌル値が入ります。

表 142. ROUTINES ビュー (続き)

列名	データ・タイプ	説明
DATA_TYPE	VARCHAR(128) ヌル可能	関数の結果のタイプ。 **BIGINT** 大整数 **INTEGER** 長整数 **SMALLINT** 短整数 **DECIMAL** パック 10 進数 **NUMERIC** ゾーン 10 進数 **DOUBLE PRECISION** 浮動小数点数; DOUBLE PRECISION **REAL** 浮動小数点数; REAL **CHARACTER** 固定長文字ストリング **CHARACTER VARYING** 可変長文字ストリング **CHARACTER LARGE OBJECT** 文字ラージ・オブジェクト・ストリング **GRAPHIC** 固定長グラフィック・ストリング **GRAPHIC VARYING** 可変長グラフィック・ストリング **DOUBLE-BYTE CHARACTER LARGE OBJECT** 2 バイト文字ラージ・オブジェクト・ストリング **BINARY LARGE OBJECT** バイナリー・ラージ・オブジェクト・ストリング **DATE** 日付 **TIME** 時刻 **TIMESTAMP** タイム・スタンプ **DATALINK** データ・リンク **ROWID** 行 ID **USER-DEFINED** 特殊タイプ これがスカラー関数でない場合は、ヌル値が入ります。
CHARACTER_MAXIMUM_LENGTH	INTEGER ヌル可能	データ・タイプが 2 進数、文字、およびグラフィック・ストリングの場合は、関数の結果ストリングの最大長。 これがスカラー関数でない場合、またはパラメーターがストリングでない場合は、ヌル値が入ります。

表 142. ROUTINES ビュー (続き)

列名	データ・タイプ	説明
CHARACTER_OCTET_LENGTH	INTEGER ヌル可能	データ・タイプが 2 進数、文字、およびグラフィック・ストリングの場合は、関数の結果ストリングのバイト数。 これがスカラー関数でない場合、またはパラメーターがストリングでない場合は、ヌル値が入ります。
CHARACTER_SET_CATALOG	VARCHAR(128) ヌル可能	関数の結果のリレーショナル・データベース名。 これがスカラー関数でない場合、または結果がストリングでない場合は、ヌル値が入ります。
CHARACTER_SET_SCHEMA	VARCHAR(128) ヌル可能	関数の結果の文字セットのスキーマ名。 'SYSIBM' が入ります。 これがスカラー関数でない場合、または結果がストリングでない場合は、ヌル値が入ります。
CHARACTER_SET_NAME	VARCHAR(128) ヌル可能	関数の結果の文字セット名。 これがスカラー関数でない場合、または結果がストリングでない場合は、ヌル値が入ります。
COLLATION_CATALOG	VARCHAR(128) ヌル可能	関数の結果のリレーショナル・データベース名。 これがスカラー関数でない場合、または結果がストリングでない場合は、ヌル値が入ります。
COLLATION_SCHEMA	VARCHAR(128) ヌル可能	関数の結果の照合のスキーマ。 SYSIBM が戻されます。 これがスカラー関数でない場合、または結果がストリングでない場合は、ヌル値が入ります。
COLLATION_NAME	VARCHAR(128) ヌル可能	関数の結果の照合名。 IBMBINARY が戻されます。 これがスカラー関数でない場合、または結果がストリングでない場合は、ヌル値が入ります。
NUMERIC_PRECISION	INTEGER ヌル可能	関数の結果の精度。 **注:** この列では、すべての数値データ・タイプ (単精度および倍精度の浮動小数点数を含む) の精度を指定します。 NUMERIC_PRECISION_RADIX 列は、この列の値が 2 進数であるか、または 10 進数であるかを示します。 これがスカラー関数でない場合、または結果が数値でない場合は、ヌル値が入ります。

表 142. ROUTINES ビュー (続き)

列名	データ・タイプ	説明
NUMERIC_PRECISION_RADIX	INTEGER ヌル可能	NUMERIC_PRECISION の列で指定される精度が、2 進数 と 10 進数のどちらの数値で指定されるかを指示します。 **2** 2 進数: 浮動小数点数の精度は 2 進数で指定されます。 **10** 10 進数: 他の数値タイプはすべて 10 進数で指定されます。 これがスカラー関数でない場合、または結果が数値でない場合は、ヌル値が入ります。
NUMERIC_SCALE	INTEGER ヌル可能	関数の結果である数値の位取り。 これがスカラー関数でない場合、または結果が数値でない場合は、ヌル値が入ります。
DATETIME_PRECISION	INTEGER ヌル可能	関数の結果である日付、時刻、またはタイム・スタンプの小数部分。 **0** データ・タイプが DATE および TIME の場合 **6** データ・タイプが TIMESTAMP の場合 (マイクロ秒数) これがスカラー関数でない場合、または結果が日付、時刻、またはタイム・スタンプでない場合は、ヌル値が入ります。
INTERVAL_TYPE	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
INTERVAL_PRECISION	INTEGER ヌル可能	予約済み。ヌル値が入ります。
TYPE_UDT_CATALOG	VARCHAR(128) ヌル可能	関数の結果が特殊タイプである場合は、リレーショナル・データベース名。 これがスカラー関数でない場合、または結果が特殊タイプでない場合は、ヌル値が入ります。
TYPE_UDT_SCHEMA	VARCHAR(128) ヌル可能	関数の結果が特殊タイプである場合は、スキーマの名前。 これがスカラー関数でない場合、または結果が特殊タイプでない場合は、ヌル値が入ります。
TYPE_UDT_NAME	VARCHAR(128) ヌル可能	関数の結果が特殊タイプである場合は、特殊タイプの名前。 これがスカラー関数でない場合、または結果が特殊タイプでない場合は、ヌル値が入ります。
SCOPE_CATALOG	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
SCOPE_SCHEMA	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。

表 142. ROUTINES ビュー (続き)

列名	データ・タイプ	説明
SCOPE_NAME	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
MAXIMUM_CARDINALITY	INTEGER ヌル可能	予約済み。ヌル値が入ります。
DTD_IDENTIFIER	VARCHAR(128) ヌル可能	関数の結果の固有の内部 ID。
ROUTINE_BODY	VARCHAR(8)	ルーチン本体のタイプ: **EXTERNAL** これは外部ルーチンです。 **SQL** これは SQL ルーチンです。
ROUTINE_DEFINITION	DBCLOB ヌル可能	これが SQL ルーチンの場合、この列は SQL ルーチン本体を含みます。 これが SQL ルーチンでない場合、または切り捨てなければルーチン本体をこの列に収容できない場合は、ヌル値が入ります。
EXTERNAL_NAME	VARCHAR(279) ヌル可能	これが外部ルーチンである場合は、この列は外部プログラム名を識別します。 • REXX の場合は、外部プログラム名は、*スキーマ名/ソース・ファイル名(メンバー名)* です。 • ILE サービス・プログラムの場合、外部プログラム名は*スキーマ名/サービス・プログラム名 (入り口名)* です。 • Java プログラムの場合、外部プログラム名は任意選択の jar-id の後に*完全修飾クラス名!メソッド名* または *完全修飾クラス名.メソッド名* が続きます。 • その他のすべての言語では、外部プログラム名は、*スキーマ名/プログラム名* です。 これが外部ルーチンでない場合は、ヌル値が入ります。

表 142. ROUTINES ビュー (続き)

列名	データ・タイプ	説明	
EXTERNAL_LANGUAGE	VARCHAR(8) ヌル可能	これが外部ルーチンである場合は、この列は外部プログラム名を識別します。	
		C	外部プログラムは C で作成されます。
		C++	外部プログラムは C++ で作成されます。
		CL	外部プログラムは CL で作成されます。
		COBOL	外部プログラムは COBOL で作成されます。
		COBOLLE	外部プログラムは ILE COBOL で作成されます。
		FORTRAN	外部プログラムは FORTRAN で作成されます。
		JAVA	外部プログラムは JAVA で作成されます。
		PLI	外部プログラムは PL/I で作成されます。
		REXX	外部プログラムは REXX プロシージャーです。
		RPG	外部プログラムは RPG で作成されます。
		RPGLE	外部プログラムは ILE RPG で作成されます。
		これが外部ルーチンでない場合は、ヌル値が入ります。	

表 142. ROUTINES ビュー (続き)

列名	データ・タイプ	説明
PARAMETER_STYLE	VARCHAR(18) ヌル可能	これが外部ルーチンである場合は、この列はパラメーターのスタイル (呼び出し規則) を識別します。 **DB2GENERAL** これは DB2GENERAL 呼び出し規則です。 **DB2SQL** これは DB2SQL 呼び出し規則です。 **GENERAL** これは GENERAL 呼び出し規則です。 **JAVA** これは JAVA 呼び出し規則です。 **GENERAL WITH NULLS** これは GENERAL WITH NULLS 呼び出し規則です。 **SQL** これは SQL 標準呼び出し規則です。 これが外部ルーチンでない場合は、ヌル値が入ります。
IS_DETERMINISTIC	VARCHAR(3)	この列はルーチンが deterministic であるかどうかを識別します。 つまり、同じ引き数のルーチンに対する呼び出しが、常に同じ結果を戻すかどうかを識別します。 **NO** ルーチンは deterministic ではありません。 **YES** ルーチンは deterministic です。
SQL_DATA_ACCESS	VARCHAR(17)	この列は、ルーチンに SQL が含まれているか、およびルーチンがデータの読み取りまたは変更を行うかを識別します。 **NO SQL** ルーチンは SQL ステートメントを含みません。 **CONTAINS SQL** ルーチンは SQL ステートメントを含みます。 **READS SQL DATA** ルーチンは、おそらく表またはビューからデータを読み取ります。 **MODIFIES SQL DATA** ルーチンは、おそらく表またはビュー内のデータを変更するか、SQL DDL ステートメントを発行します。

表 142. ROUTINES ビュー (続き)

列名	データ・タイプ	説明
IS_NULL_CALL	VARCHAR(3) ヌル可能	入力パラメーターがヌル値である場合に、関数を呼び出す必要があるかどうかを識別します。 **NO** この関数は、入力パラメーターがヌル値の場合に呼び出す必要はありません。これがスカラー関数の場合は、いずれかのオペランドがヌルであれば、この関数の結果は暗黙的にヌルになります。これが表関数の場合は、いずれかのオペランドがヌル値であれば、この関数の結果は空の表になります。 **YES** この関数は、入力オペランドがヌルでも呼び出す必要があります。 これが関数でない場合は、ヌル値が入ります。
SQL_PATH	VARCHAR(3483) ヌル可能	これが SQL ルーチンの場合、この列はパスを識別します。 これが SQL ルーチンでない場合は、ヌル値が入ります。
SCHEMA_LEVEL_ROUTINE	VARCHAR(3)	予約済み。 'YES' が入ります。
MAX_DYNAMIC_RESULT_SETS	SMALLINT	戻される結果セットの最大数を識別します。 0 は結果セットがないことを示します。
IS_USER_DEFINED_CAST	VARCHAR(3) ヌル可能	この関数が、特殊タイプの作成時に作成されたキャスト関数であるかどうかを判別します。 **NO** この関数はキャスト関数ではありません。 **YES** この関数はキャスト関数です。 ルーチンが関数でない場合は、ヌル値が入ります。
IS_IMPLICITLY_INVOCABLE	VARCHAR(3) ヌル可能	この関数が、特殊タイプの作成時に作成されたキャスト関数であって、暗黙的に呼び出せるかどうかを判別します。 **NO** この関数はキャスト関数ではありません。 **YES** この関数はキャスト関数であって、暗黙的に呼び出すことができます。 ルーチンが関数でない場合は、ヌル値が入ります。
SECURITY_TYPE	VARCHAR(22) ヌル可能	予約済み。これが外部ルーチンである場合は、'IMPLEMENTATION DEFINED' が入ります。 ルーチンが外部ルーチンでない場合は、ヌル値が入ります。
TO_SQL_SPECIFIC_CATALOG	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
TO_SQL_SPECIFIC_SCHEMA	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。

表 142. ROUTINES ビュー (続き)

列名	データ・タイプ	説明
TO_SQL_SPECIFIC_NAME	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
AS_LOCATOR	VARCHAR(3) ヌル可能	結果がロケーターとして指定されたかどうかを識別します。 **NO** 結果はロケーターとして指定されませんでした。 **YES** 結果はロケーターとして指定されました。 これがスカラー関数でない場合は、ヌル値が入ります。
CREATED	TIMESTAMP	ルーチンが作成されたときのタイム・スタンプを識別します。
LAST_ALTERED	TIMESTAMP	予約済み。 'CREATED' が入ります。

SCHEMATA

SCHEMATA ビューには、各スキーマごとに行が 1 つずつ入ります。次の表は、ビューの列について説明しています。

表 143. SCHEMATA ビュー

列名	データ・タイプ	説明
CATALOG_NAME	VARCHAR(128)	リレーショナル・データベース名
SCHEMA_NAME	VARCHAR(128)	スキーマの名前
SCHEMA_OWNER	VARCHAR(128)	スキーマの所有者
DEFAULT_CHARACTER_SET_CATALOG	VARCHAR(128)	リレーショナル・データベース名
DEFAULT_CHARACTER_SET_SCHEMA	VARCHAR(128)	デフォルト文字セットのスキーマ名。 'SYSIBM' が入ります。
DEFAULT_CHARACTER_SET_NAME	VARCHAR(128)	デフォルト文字セット名。
SQL_PATH	VARCHAR(3483) ヌル可能	予約済み。ヌル値が入ります。

SQL_FEATURES

SQL_FEATURES ビューには、データベース・マネージャーでサポートされている各フィーチャーごとに、行が 1 つずつ入ります。次の表は、ビューの列について説明しています。

表 144. SQL_FEATURES ビュー

列名	データ・タイプ	説明
FEATURE_ID	VARCHAR(7)	ANS および ISO のフィーチャー ID
FEATURE_NAME	VARCHAR(128)	ANS および ISO のフィーチャーの名前。
SUB_FEATURE_ID	VARCHAR(7)	ANS および ISO のサブフィーチャー ID
SUB_FEATURE_NAME	VARCHAR(256)	ANS および ISO のサブフィーチャーの名前。
IS_SUPPORTED	VARCHAR(3)	該当のフィーチャーがサポートされているかどうかを示します。 **YES** このフィーチャーはサポートされています。 **NO** このフィーチャーはサポートされていません。
IS_VERIFIED_BY	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
COMMENTS	VARCHAR(2000) ヌル可能	予約済み。ヌル値が入ります。

SQL_LANGUAGES

SQL_LANGUAGES (システム名 SYSLANGS) 表には、適合性が要求される SQL 言語バインディングおよびプログラム言語ごとに行が 1 つずつ入ります。次の表は、SQL_LANGUAGES ビューの列について説明しています。

表 145. SQL_LANGUAGES ビュー

列名	データ・タイプ	説明
SQL_LANGUAGE_SOURCE	VARCHAR(254)	標準の名前
SQL_LANGUAGE_YEAR	VARCHAR(254)	標準が承認された年
SQL_LANGUAGE_CONFORMANCE	VARCHAR(254) ヌル可能	適合性のレベル。 **2** 1987 年および 1989 年の標準の場合、レベル 2 の適合性が要求されることを示します。 **ENTRY** 1992 年の標準の場合、エントリー・レベルの適合性が要求されることを示します。 **CORE** 1999 年の標準の場合、コア・レベルの適合性が要求されることを示します。 適合性がまだ要求されていない場合は、ヌル値が入ります。
SQL_LANGUAGE_INTEGRITY	VARCHAR(254) ヌル可能	整合性機能のサポート。 **YES** 整合性に対して適合性が要求されます。 **NO** 整合性に対して適合性は要求されません。 標準に単独の整合性フィーチャーがない場合は、ヌル値が入ります。
SQL_LANGUAGE_IMPLEMENTATION	VARCHAR(254) ヌル可能	予約済み。ヌル値が入ります。
SQL_LANGUAGE_BINDING_STYLE	VARCHAR(254)	SQL 言語のバインディングのスタイル **EMBEDDED** 以下の言語に関する組み込み SQL のサポート SQL_LANGUAGE_PROGRAMMING_LANG **DIRECT** DIRECT SQL がサポートされます (例えば、対話式 SQL)。 **CLI** 以下の言語に関する CLI のサポート SQL_LANGUAGE_PROGRAMMING_LANG

表 145. SQL_LANGUAGES ビュー (続き)

列名	データ・タイプ	説明
SQL_LANGUAGE_PROGRAMMING_LANG	VARCHAR(254) ヌル可能	EMBEDDED または CLI によってサポートされている言語 **C** C 言語がサポートされます。 **COBOL** COBOL 言語がサポートされます。 **PLI** PL/I 言語がサポートされます。 SQL_LANGUAGE_BINDING_STYLE が DIRECT でない場合は、ヌル値が入ります。

SQL_SIZING

SQL_SIZING ビューには、データベース・マネージャーでサポートされている各限度ごとに、行が 1 つずつ入ります。次の表は、ビューの列について説明しています。

表 146. SQL_SIZING ビュー

列名	データ・タイプ	説明
SIZING_ID	INTEGER	ANS および ISO のサイジング ID。
SIZING_NAME	VARCHAR(128)	ANS および ISO のサイジングの名前。
SUPPORTED_VALUE	INTEGER ヌル可能	サイジング限度を示します。 サイジング限度が適用されない場合は、ヌル値が入ります。
COMMENTS	VARCHAR(2000) ヌル可能	予約済み。ヌル値が入ります。

TABLE_CONSTRAINTS

TABLE_CONSTRAINTS ビューには、各制約ごとに行が 1 つずつ入ります。次の表は、ビューの列について説明しています。

表 147. TABLE_CONSTRAINTS ビュー

列名	データ・タイプ	説明
CONSTRAINT_CATALOG	VARCHAR(128)	リレーショナル・データベース名
CONSTRAINT_SCHEMA	VARCHAR(128)	該当の制約が入っているスキーマの名前。
CONSTRAINT_NAME	VARCHAR(128)	制約の名前。
TABLE_CATALOG	VARCHAR(128)	リレーショナル・データベース名
TABLE_SCHEMA	VARCHAR(128)	該当の表が入っているスキーマの名前。
TABLE_NAME	VARCHAR(128)	該当の制約が作成される表の名前。
CONSTRAINT_TYPE	VARCHAR(11)	制約のタイプ CHECK UNIQUE PRIMARY KEY FOREIGN KEY
IS_DEFERRABLE	VARCHAR(3)	制約の検査が据え置きできるかどうかを示します。'NO' が入ります。
INITIALLY_DEFERRED	VARCHAR(3)	制約が初期据え置きとして定義されたかどうかを示します。 'NO' が入ります。

TABLES

TABLES ビューには、各表、ビュー、および別名ごとに、行が 1 つずつ入ります。次の表は、ビューの列について説明しています。

表 148. TABLES ビュー

列名	データ・タイプ	説明
TABLE_CATALOG	VARCHAR(128)	リレーショナル・データベース名
TABLE_SCHEMA	VARCHAR(128)	該当の表、ビュー、または別名を含む SQL スキーマの名前。
TABLE_NAME	VARCHAR(128)	その表、ビュー、または別名の名前。
TABLE_TYPE	VARCHAR(10)	表のタイプを識別します。 **ALIAS** 表は別名です。 **BASE_TABLE** 表は、SQL 表または物理ファイルです。 **VIEW** 表は、SQL ビューまたは論理ファイルです。
SELF_REFERENCING_COLUMN_NAME	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
REFERENCE_GENERATION	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
USER_DEFINED_TYPE_CATALOG	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
USER_DEFINED_TYPE_SCHEMA	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
USER_DEFINED_TYPE_NAME	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
IS_INSERTABLE_INTO	VARCHAR(3)	表で INSERT を使用できるかどうかを識別します。 **NO** この表では INSERT は使用できません。 **YES** この表では INSERT を使用できます。

USER_DEFINED_TYPES

USER_DEFINED_TYPES ビューには、各特殊タイプごとに行が 1 つずつ入ります。[83] 次の表は、ビューの列について説明しています。

表 149. USER_DEFINED_TYPES ビュー

列名	データ・タイプ	説明
USER_DEFINED_TYPE_CATALOG	VARCHAR(128)	リレーショナル・データベース名
USER_DEFINED_TYPE_SCHEMA	VARCHAR(128)	特殊タイプのスキーマ名。
USER_DEFINED_TYPE_NAME	VARCHAR(128)	特殊タイプを作成したユーザーの名前。
USER_DEFINED_TYPE_CATEGORY	VARCHAR(128)	ユーザー定義タイプのタイプを示します。'DISTINCT' が入ります。
IS_INSTANTIABLE	VARCHAR(3)	予約済み。 'YES' が入ります。
IS_FINAL	VARCHAR(3)	予約済み。 'YES' が入ります。
ORDERING_FORM	VARCHAR(4)	この特殊タイプが被比較数の場合に、許される述部の種類を示します。 **FULL** すべての述部が許されます。 **NONE** 述部は許されません。
ORDERING_CATEGORY	VARCHAR(8)	予約済み。 'MAP' が入ります。
ORDERING_ROUTINE_CATALOG	VARCHAR(128) ヌル可能	リレーショナル・データベース名。 ORDERING_FORM が 'NONE' である場合は、ヌル値が入ります。
ORDERING_ROUTINE_SCHEMA	VARCHAR(128) ヌル可能	予約済み。 'SYSIBM' が入ります。 ORDERING_FORM が 'NONE' である場合は、ヌル値が入ります。
ORDERING_ROUTINE_NAME	VARCHAR(128) ヌル可能	予約済み。データ・タイプ名が入ります。 ORDERING_FORM が 'NONE' である場合は、ヌル値が入ります。
REFERENCE_TYPE	VARCHAR(16) ヌル可能	予約済み。ヌル値が入ります。

83. このビューには、組み込みデータ・タイプについての情報は含まれていません。

表 149. USER_DEFINED_TYPES ビュー (続き)

列名	データ・タイプ	説明
DATA_TYPE	VARCHAR(128) ヌル可能	特殊タイプのソース・データ・タイプ: **BIGINT** 大整数 **INTEGER** 長整数 **SMALLINT** 短整数 **DECIMAL** パック 10 進数 **NUMERIC** ゾーン 10 進数 **DOUBLE PRECISION** 浮動小数点数; DOUBLE PRECISION **REAL** 浮動小数点数; REAL **CHARACTER** 固定長文字ストリング **CHARACTER VARYING** 可変長文字ストリング **CHARACTER LARGE OBJECT** 文字ラージ・オブジェクト・ストリング **GRAPHIC** 固定長グラフィック・ストリング **GRAPHIC VARYING** 可変長グラフィック・ストリング **DOUBLE-BYTE CHARACTER LARGE OBJECT** 2 バイト文字ラージ・オブジェクト・ストリング **BINARY LARGE OBJECT** バイナリー・ラージ・オブジェクト・ストリング **DATE** 日付 **TIME** 時刻 **TIMESTAMP** タイム・スタンプ **DATALINK** データ・リンク **ROWID** 行 ID **USER-DEFINED** 特殊タイプ
CHARACTER_MAXIMUM_LENGTH	INTEGER ヌル可能	データ・タイプが 2 進数、文字、およびグラフィック・ストリングの場合は、特殊タイプの最大長。 特殊タイプがストリングでない場合は、ヌル値が入ります。

表 149. USER_DEFINED_TYPES ビュー (続き)

列名	データ・タイプ	説明
CHARACTER_OCTET_LENGTH	INTEGER ヌル可能	データ・タイプが 2 進数、文字、およびグラフィック・ストリングの場合は、特殊タイプのバイト数。 特殊タイプがストリングでない場合は、ヌル値が入ります。
CHARACTER_SET_CATALOG	VARCHAR(128) ヌル可能	特殊タイプのリレーショナル・データベース名。 特殊タイプがストリングでない場合は、ヌル値が入ります。
CHARACTER_SET_SCHEMA	VARCHAR(128) ヌル可能	特殊タイプの文字セットのスキーマ名。 'SYSIBM' が入ります。 特殊タイプがストリングでない場合は、ヌル値が入ります。
CHARACTER_SET_NAME	VARCHAR(128) ヌル可能	特殊タイプの文字セット名。 特殊タイプがストリングでない場合は、ヌル値が入ります。
COLLATION_CATALOG	VARCHAR(128) ヌル可能	特殊タイプのリレーショナル・データベース名。 特殊タイプがストリングでない場合は、ヌル値が入ります。
COLLATION_SCHEMA	VARCHAR(128) ヌル可能	特殊タイプの照合のスキーマ。 SYSIBM が戻されます。 特殊タイプがストリングでない場合は、ヌル値が入ります。
COLLATION_NAME	VARCHAR(128) ヌル可能	特殊タイプの照合名。 IBMBINARY が戻されます。 特殊タイプがストリングでない場合は、ヌル値が入ります。
NUMERIC_PRECISION	INTEGER ヌル可能	特殊タイプの精度。 **注:** この列では、すべての数値データ・タイプ (単精度および倍精度の浮動小数点数を含む) の精度を指定します。 NUMERIC_PRECISION_RADIX 列は、この列の値が 2 進数であるか、または 10 進数であるかを示します。 特殊タイプが数値でない場合は、ヌル値が入ります。

表 149. USER_DEFINED_TYPES ビュー (続き)

列名	データ・タイプ	説明
NUMERIC_PRECISION_RADIX	INTEGER ヌル可能	NUMERIC_PRECISION の列で指定される精度が、2 進数 と 10 進数のどちらの数値で指定されるかを指示します。 **2** 2 進数: 浮動小数点数の精度は 2 進数で指定されます。 **10** 10 進数: 他の数値タイプはすべて 10 進数で指定されます。 特殊タイプが数値でない場合は、ヌル値が入ります。
NUMERIC_SCALE	INTEGER ヌル可能	数値特殊タイプの位取り。 特殊タイプが 10 進数、数値、または 2 進数でない場合は、ヌル値が入ります。
DATETIME_PRECISION	INTEGER ヌル可能	日付、時刻、またはタイム・スタンプを表す特殊タイプの小数部分。 **0** データ・タイプが DATE および TIME の場合 **6** データ・タイプが TIMESTAMP の場合 (マイクロ秒数) 特殊タイプが日付、時刻、またはタイム・スタンプでない場合は、ヌル値が入ります。
INTERVAL_TYPE	VARCHAR(128) ヌル可能	予約済み。ヌル値が入ります。
INTERVAL_PRECISION	INTEGER ヌル可能	予約済み。ヌル値が入ります。
SOURCE_DTD_IDENTIFIER	VARCHAR(128) ヌル可能	ソース・データ・タイプの固有の内部 ID。 この特殊タイプが他の特殊タイプをソースとしていない場合は、ヌル値が入ります。
REF_DTD_IDENTIFIER	VARCHAR(256) ヌル可能	予約済み。ヌル値が入ります。

VIEWS

VIEWS ビューには、各ビューごとに行が 1 つずつ入ります。次の表は、ビューの列について説明しています。

表 150. VIEWS ビュー

列名	データ・タイプ	説明
TABLE_CATALOG	VARCHAR(128)	リレーショナル・データベース名。
TABLE_SCHEMA	VARCHAR(128)	該当のビューが入っている SQL のスキーマの名前。
TABLE_NAME	VARCHAR(128)	ビューの名前。
VIEW_DEFINITION	VARCHAR(10000) ヌル可能	CREATE VIEW ステートメントの QUERY 式の部分。 切り捨てなければビュー定義を列に収容できない場合は、ヌル値が入ります。
CHECK_OPTION	VARCHAR(8)	該当のビューに対して使用された検査オプション **NONE** 検査オプションは指定されませんでした **LOCAL** ローカル・オプションが指定されました **CASCADED** カスケード・オプションが指定されました
IS_UPDATABLE	VARCHAR(3)	ビューが更新可能かどうかを指定します。 **YES** 更新可能なビューです。 **NO** 読み取り専用のビューです。

参考文献

ここにリストした資料には、本書で説明した事項や参照したトピックに関する追加情報が収録されています。いずれの資料も、それぞれの正式表題と資料番号を付けて示してあります。本書で参照している資料の場合、略称を使用しています。

- バックアップおよび回復の手引き 

  バックアップおよび回復の計画、保管および復元手順のために使用できる各種のメディア、およびディスク回復手順に関する情報が収録されています。バックアップからシステムを再インストールする方法も紹介されています。

- ILE COBOL プログラマーの手引き 

  この資料には、iSeries 400 システムで COBOL/400 プログラムの設計、作成、テスト、および保守を行う上で必要な情報が収録されています。

- ILE RPG プログラマーの手引き 

  この資料には、iSeries 400 システムで ILE RPG/400 プログラムの設計、作成、テスト、および保守を行う上で必要な情報が収録されています。

- REXX/400 Programmer's Guide 

  この資料には、iSeries 400 システムで REXX/400 プログラムの設計、作成、テスト、および保守を行う上で必要な情報が収録されています。

- CL プログラミング 

  本書は、iSeries 400 のプログラミングに関係する事柄を広範囲にわたって論じています。主なものを挙げると、オブジェクトとライブラリーの全般的な説明、CL プログラミング、プログラム相互間の流れと通信の制御、CL プログラムにおけるオブジェクトの扱い方、CL プログラムの作成方法などがあります。その他に、事前定義のメッセージと即時メッセージ、ユーザーが定義するコマンドとメニューの取り扱い、定義、および作成の方法、デバッグ・モード、停止点、トレースなどを含むアプリケーションのテスト、および表示機能についても説明しています。

- ファイル管理

  アプリケーション・プログラムにおけるファイルの使い方を説明しています。

- データベース・プログラミング

  この資料では、システム上でデータベース・ファイルを作成、記述、および更新する方法の説明を含め、iSeries データベース編成の詳しい説明をしています。

- 分散データベース・プログラミング

  この資料では、分散リレーショナル・データベース・アーキテクチャー (DRDA) を使用して、iSeries システムを分散リレーショナル・データベースで準備および管理する方法について説明しています。この資料では、類似のシステム環境における複数の iSeries システム上で、分散リレーショナル・データベースを計画、設定、プログラミング、管理、および操作する方法について説明しています。

- iSeries セキュリティーの手引き 

  本書には、システム・セキュリティーの概念、セキュリティーのための計画方法、およびシステムにおけるセキュリティーのセットアップ方法に関する情報が記載されています。また、正当な権限を持たないユーザーの使用からシステムやデータを保護する方法、故意または過失による損傷や破壊からデータを保護する方法、セキュリティーを最新に保つ方法、システム上にセキュリティーを設定する方法についても説明しています。

- SQL プログラミング 概念

  この資料では、DB2 UDB for iSeries ステートメントの設計、作成、実行、およびテストの方法について概説しています。また、対話式構造化照会言語 (SQL) についても説明しています。

- ホスト言語での SQL プログラミング

この資料には、COBOL、ILE COBOL/400、ILE RPG/400、ILE C/400、および PL/I プログラムで SQL ステートメントを使用する方法の例が示してあります。

- データベース・パフォーマンスおよび Query 最適化

  この資料では、使用可能なツールおよび手法を使用して、照会のパフォーマンスを最適化するための情報が記載されています。

- IDDU Use 

  この資料は、iSeries の対話式データ定義ユーティリティー (IDDU) を使用して、データ・ディクショナリー、ファイル、およびレコードをシステムに対して記述する方法を説明しています。

- SQL 呼び出しレベル・インターフェース (ODBC)

  この資料では、X/Open SQL 呼び出しレベル・インターフェースを使用し、DB2 UDB for iSeries で用意されているサービス・プログラムに対するプロシージャー呼び出しを直接介して、SQL 関数にアクセスする方法を説明しています。

- iSeries Information Center の Client Access Express カテゴリー

  この資料では、クライアント・アクセス ODBC を使用して、クライアント上で ODBC アプリケーションをセットアップし実行する方法を説明しています。この資料には、パフォーマンス、例、および クライアント・アクセス ODBC によって実行する特定のアプリケーションの構成に関する章が含まれています。

- IBM Toolbox for Java

  この資料では、IBM Toolbox for Java を使用して、クライアントで JDBC アプリケーションをセットアップし、実行する方法を説明しています。この資料には、パフォーマンス、例、および IBM Toolbox for Java によって実行する特定のアプリケーションの構成に関する章が含まれています。

- IBM Developer Kit for Java

  この資料には、iSeries システムで JAVA プログラムの設計、作成、テスト、および保守を行う上で必要な情報が収録されています。この資料には、IBM Developer Kit for Java JDBC ドライバーに関する情報も含まれています。

- DB2 マルチ・システム

  この資料は、分散リレーショナル・データベース・ファイル、ノード・グループ、および区分化についての、基本的な概念を説明しています。これには、複数のシステムにわたって区分化されるデータベース・ファイルを作成し、使用するために必要な情報が収録されています。システムの構成方法、ファイルの作成方法、およびアプリケーションにおけるファイルの使用方法について説明してあります。

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

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

[特殊文字]

Printed in Japan

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