SYBASE®

リファレンス・マニュアル：ビルディング・ブロック

Adaptive Server® Enterprise

15.5

ドキュメント ID：DC36422-01-1550-01
改訂：2009 年 11 月

マニュアルの注文

マニュアルの注文を承ります。ご希望の方は、サイベース株式会社営業部または代理店までご連絡ください。マニュアルの変更は、弊社の定期的なソフトウェア・リリース時にのみ提供されます。

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

目次

はじめに

『ASE リファレンス・マニュアル』は、Sybase® Adaptive Server® Enterprise と Transact-SQL® 言語について説明する 4 巻構成のマニュアルです。

- 『ビルディング・ブロック』では、データ型、組み込み関数、グローバル変数、式と識別子、予約語、SQLSTATE エラーなど、Transact-SQL の各要素について説明します。Transact-SQL を使用する前に、ビルディング・ブロックの動作とビルディング・ブロックが Transact-SQL 文の結果にどのように影響するかを理解する必要があります。
- 『コマンド』では、文の作成に使用する Transact-SQL コマンドについてのリファレンス情報が記載されています。
- 『プロシージャ』では、システム・プロシージャ、カタログ・ストアド・プロシージャ、拡張ストアド・プロシージャ、dbcc ストアド・プロシージャについてのリファレンス情報が記載されています。すべてのプロシージャは、Transact-SQL 文を使用して作成されます。
- 『テーブル』では、使用しているサーバ、データベース、ユーザ、サーバの他の詳細についての情報を格納するシステム・テーブルに関するリファレンス情報が記載されています。dbccdb データベースと dbccalt データベースのテーブルについても説明します。

対象読者

『リファレンス・マニュアル』は、リファレンスのツールとして Transact-SQL のすべてのレベルのユーザを対象としています。

このマニュアルの内容

- 「第 1 章 システム・データ型とユーザ定義データ型」では、Adaptive Server のシステム定義データ型およびユーザ定義データ型について説明し、それらを使用してユーザ定義データ型を作成する方法について説明します。
- 「第 2 章 Transact-SQL 関数」には、Adaptive Server 関数の名前と簡単な説明のリストが表形式で記載されています。
- 「第 3 章 グローバル変数」には、Active Server のシステム定義変数の名前とリターン・ステータスの簡単な説明のリストが表形式で記載されています。
- 「第 4 章 式、識別子、およびワイルドカード文字」では、Transact-SQL 言語の使用方法について説明します。
- 「第 5 章 予約語」では、Transact-SQL と ANSI SQL のキーワードについて説明します。

- 「第 6 章 SQLSTATE コードとメッセージ」では、Adaptive Server SQLSTATE ステータス・コードと各コードに対応するメッセージについて説明します。

関連マニュアル

Adaptive Server® Enterprise には次のマニュアルが用意されています。必要に応じて参照してください。

- 使用しているプラットフォームの『リリース・ノート』－ マニュアルには記載できなかった最新の情報が記載されています。

  このリリース・ノートの最新バージョン (英語版) を入手できます。製品の CD がリリースされた後で、製品またはマニュアルに関する重要な情報が追加されているかを確認するには、Sybase® Product Manuals Web サイトを使用してください。

- 使用しているプラットフォームの『インストール・ガイド』－ すべての Adaptive Server および関連する Sybase 製品のインストール、アップグレード、設定の手順について説明しています。

- 『新機能ガイド』－ Adaptive Server の新しい機能について説明しています。また、新しい機能をサポートするためのシステム変更や、既存のアプリケーションに影響を与える可能性がある変更についても説明しています。

- 『Active Messaging ユーザーズ・ガイド』－ Active Messaging を使用して、Adaptive Server Enterprise データベースでトランザクション (データ変更) を取得し、外部アプリケーションにイベントとしてリアルタイムで渡す方法について説明しています。

- 『コンポーネント統合サービス・ユーザーズ・ガイド』－ コンポーネント統合サービスを使用して、リモートの Sybase データベースおよび Sybase 以外のデータベースに接続する方法について説明しています。

- 使用しているプラットフォームの『設定ガイド』－ 特定の設定作業の手順について説明しています。

- 『用語解説』－ Adaptive Server マニュアルで使用されている技術用語について説明しています。

- 『Historical Server ユーザーズ・ガイド』－ Historical Server を使用して、Adaptive Server のパフォーマンス情報を入手する方法について説明しています。

- 『Adaptive Server Enterprise における Java』－ Adaptive Server データベースで Java クラスをデータ型、関数、ストアド・プロシージャとしてインストールして使用する方法について説明しています。

- 『Job Scheduler ユーザーズ・ガイド』－ コマンド・ラインまたはグラフィカル・ユーザ・インタフェース (GUI) を使用して、ローカルまたはリモートの Adaptive Server でジョブのインストール、設定、作成、スケジュールを行う方法について説明しています。

- 『マイグレーション技術ガイド』－ 別のバージョンの Adaptive Server にマイグレートするための方法とツールについて説明しています。

- 『Monitor Client Library プログラマーズ・ガイド』 – Adaptive Server のパフォーマンス・データにアクセスする Monitor Client Library アプリケーションの記述方法について説明しています。
- 『Monitor Server ユーザーズ・ガイド』 – Monitor Server を使用して、Adaptive Server のパフォーマンス統計を取得する方法について説明しています。
- 『モニタリング・テーブル・ダイヤグラム』 – モニタリング・テーブルと、そのエンティティの関係をポスター形式で図解しています。フル・サイズのダイアグラムは印刷版だけで参照できます。コンパクト版は PDF 形式で参照できます。
- 『パフォーマンス&チューニング・シリーズ』 – Adaptive Server で最高のパフォーマンスを実現するためのチューニング方法について説明しています。このマニュアルは以下の 4 冊に分かれています。
  - 『基本』 – Adaptive Server のパフォーマンスに関する問題の理解と調査の基本について説明しています。
  - 『統計的分析によるパフォーマンスの向上』 – Adaptive Server で統計情報がどのように保存され、表示されるかについて説明しています。また、set statistics コマンドを使用して、サーバの統計情報を分析する方法について説明しています。
  - 『ロックと同時実行制御』 – ロック・スキームを使用してパフォーマンスを向上させる方法と、同時実行性を最小限に抑えるようにインデックスを選択する方法について説明しています。
  - 『sp_sysmon による Adaptive Server の監視』 – sp_sysmon を使用してパフォーマンスをモニタリングする方法について説明しています。
  - 『モニタリング・テーブル』 – Adaptive Server のモニタリング・テーブルに統計情報や診断情報を問い合わせる方法について説明しています。
  - 『物理データベースのチューニング』 – データの物理的配置、データに割り付けられた領域、テンポラリ・データベースの管理方法について説明しています。
  - 『クエリ処理と抽象プラン』 – オプティマイザがクエリを処理する方法と、抽象プランを使用してオプティマイザのプランの一部を変更する方法について説明しています。
- 『クイック・リファレンス・ガイド』 – コマンド、関数、システム・プロシージャ、拡張システム・プロシージャ、データ型、ユーティリティの名前と構文の包括的な一覧表を記載したポケット版 (PDF 版は通常サイズ) のマニュアルです。
- 『ASE リファレンス・マニュアル』 – 詳細な Transact-SQL® 情報を記載しています。このマニュアルは以下の 4 冊に分かれています。

    - 『ビルディング・ブロック』– データ型、関数、グローバル変数、式、識別子とワイルドカード、予約語について説明しています。
    - 『コマンド』– コマンドについて説明しています。
    - 『プロシージャ』– システム・プロシージャ、カタログ・ストアド・プロシージャ、システム拡張ストアド・プロシージャ、dbcc ストアド・プロシージャについて説明しています。
    - 『テーブル』– システム・テーブル、モニタリング・テーブル、dbcc テーブルについて説明しています。
- 『システム管理ガイド』–
    - 『第 1 巻』– 設定パラメータ、リソースの問題、文字セット、ソート順、システムの問題の診断方法に関する説明を含め、システム管理の基本の概要について説明しています。『第 1 巻』の後半は、セキュリティ管理に関する詳細な説明です。
    - 『第 2 巻』– 物理的なリソースの管理、デバイスのミラーリング、メモリとデータ・キャッシュの設定、マルチプロセッサ・サーバとユーザ・データベースの管理、データベースのマウントとマウント解除、セグメントの作成と使用、reorg コマンドの使用、データベース一貫性の検査方法についての手順とガイドラインを説明しています。『第 2 巻』の後半では、システムとユーザ・データベースをバックアップおよびリストアする方法について説明しています。
- 『システム・テーブル・ダイアグラム』– システム・テーブルと、そのエンティティとの関係をポスター形式で図解しています。フル・サイズのダイアグラムは印刷版だけで参照できます。コンパクト版は PDF 形式で参照できます。
- 『Transact-SQL ユーザーズ・ガイド』– リレーショナル・データベース言語の拡張版である Sybase の Transact-SQL について説明しています。まだ経験の浅いデータベース管理システムのユーザは、このマニュアルをガイドブックとして使用してください。pubs2 および pubs3 サンプル・データベースの詳細も説明しています。
- 『トラブルシューティング：エラー・メッセージと詳細な解決方法』– 発生する可能性のある問題について、トラブルシューティング手順を説明しています。このマニュアルで取り上げられている問題は、Sybase 製品の保守契約を結んでいるサポート・センタに最も頻繁に寄せられるものです。
- 『暗号化カラム・ユーザーズ・ガイド』– Adaptive Server を使用して暗号化カラムを設定し、使用する方法について説明しています。
- 『インメモリ・データベース・ユーザーズ・ガイド』– インメモリ・データベースの設定および使用方法について説明しています。
- 『Adaptive Server 分散トランザクション管理機能の使用』– 分散トランザクション処理環境での Adaptive Server DTM 機能の設定、使用、トラブルシューティングについて説明しています。

- 『IBM® Tivoli® Storage Manager と Backup Server の使用』－ IBM Tivoli Storage Manager を設定および使用して Adaptive Server のバックアップを作成する方法について説明しています。
- 『高可用性システムにおける Sybase フェールオーバの使用』－ Sybase のフェールオーバ機能を使用して、Adaptive Server を高可用性システムのコンパニオン・サーバとして設定する方法について説明しています。
- 『Unified Agent および Agent Management Console』－ Unified Agent について説明しています。Unified Agent は、分散 Sybase リソースを管理、モニタ、制御するためのランタイム・サービスを提供します。
- 『ASE ユーティリティ・ガイド』－ オペレーティング・システム・レベルで実行される isql および bcp などの、Adaptive Server のユーティリティ・プログラムについて説明しています。
- 『Web Services ユーザーズ・ガイド』－ Adaptive Server 用の Web サービスの設定、使用、トラブルシューティング方法について説明しています。
- 『XA インタフェース統合ガイド for CICS、Encina、TUXEDO』－ X/Open XA トランザクション・マネージャを備えた Sybase DTM XA インタフェースを使用する方法について説明しています。
- 『Adaptive Server Enterprise における XML Services』－ データベースに XML 機能を導入する、Sybase ネイティブの XML プロセッサと Sybase Java ベースの XML のサポートについて、また XML サービスで使用できるクエリとマッピング用の関数について説明しています。

その他の情報

Sybase Getting Started CD、SyBooks™ CD、Sybase Product Manuals Web サイトを利用すると、製品について詳しく知ることができます。

- Getting Started CD には、PDF 形式のリリース・ノートとインストール・ガイド、SyBooks CD に含まれていないその他のマニュアルや更新情報が収録されています。この CD は製品のソフトウェアに同梱されています。Getting Started CD に収録されているマニュアルを参照または印刷するには、Adobe Acrobat Reader が必要です (CD 内のリンクを使用して Adobe の Web サイトから無料でダウンロードできます)。
- SyBooks CD には製品マニュアルが収録されています。この CD は製品のソフトウェアに同梱されています。Eclipse ベースの SyBooks ブラウザを使用すれば、使いやすい HTML 形式のマニュアルにアクセスできます。

  一部のマニュアルは PDF 形式で提供されています。これらのマニュアルは SyBooks CD の PDF ディレクトリに収録されています。PDF ファイルを開いたり印刷したりするには、Adobe Acrobat Reader が必要です。

  SyBooks をインストールして起動するまでの手順については、Getting Started CD の『SyBooks インストール・ガイド』、または SyBooks CD の *README.txt* ファイルを参照してください。

- Sybase Product Manuals Web サイトは、SyBooks CD のオンライン版であり、標準の Web ブラウザを使用してアクセスできます。また、製品マニュアルのほか、EBFs/Updates、Technical Documents、Case Management、Solved Cases、ニュース・グループ、Sybase Developer Network へのリンクもあります。

  Product Manuals Web サイトにアクセスするには、Product Manuals (http://www.sybase.com/support/manuals/) にアクセスしてください。

Web 上の Sybase 製品の動作確認情報

Sybase Web サイトの技術的な資料は頻繁に更新されます。

❖ **製品認定の最新情報にアクセスする**

1 Web ブラウザで Technical Documents を指定します。(http://www.sybase.com/support/techdocs/)

2 [Certification Report] をクリックします。

3 [Certification Report] フィルタで製品、プラットフォーム、時間枠を指定して [Go] をクリックします。

4 [Certification Report] のタイトルをクリックして、レポートを表示します。

❖ **コンポーネント認定の最新情報にアクセスする**

1 Web ブラウザで Availability and Certification Reports を指定します。(http://certification.sybase.com/)

2 [Search By Base Product] で製品ファミリとベース製品を選択するか、[Search by Platform] でプラットフォームとベース製品を選択します。

3 [Search] をクリックして、入手状況と認定レポートを表示します。

❖ **Sybase Web サイト (サポート・ページを含む) の自分専用のビューを作成する**

MySybase プロファイルを設定します。MySybase は無料サービスです。このサービスを使用すると、Sybase Web ページの表示方法を自分専用にカスタマイズできます。

1 Web ブラウザで Technical Documents を指定します。(http://www.sybase.com/support/techdocs/)

2 [MySybase] をクリックし、MySybase プロファイルを作成します。

Sybase EBF とソフトウェア・メンテナンス

❖ EBF とソフトウェア・メンテナンスの最新情報にアクセスする

1 Web ブラウザで Sybase Support Page を指定します。(http://www.sybase.com/support)

2 [EBFs/Maintenance] を選択します。MySybase のユーザ名とパスワードを入力します。

3 製品を選択します。

4 時間枠を指定して [Go] をクリックします。EBF/Maintenance リリースの一覧が表示されます。

鍵のアイコンは、「Technical Support Contact」として登録されていないため、一部の EBF/Maintenance リリースをダウンロードする権限がないことを示しています。未登録でも、Sybase 担当者またはサポート・コンタクトから有効な情報を得ている場合は、[Edit Roles] をクリックして、「Technical Support Contact」の役割を MySybase プロファイルに追加します。

5 EBF/Maintenance レポートを表示するには [Info] アイコンをクリックします。ソフトウェアをダウンロードするには製品の説明をクリックします。

表記規則

次の項では、このマニュアルで使用されている表記について説明します。

SQL は自由な形式の言語で、1 行内のワード数や、改行の仕方に規則はありません。このマニュアルでは、読みやすくするため、例や構文を文の句ごとに改行しています。複数の部分からなり、2 行以上にわたる場合は、字下げしています。複雑なコマンドの書式には、修正された BNF (Backus Naur Form) 記法が使用されています。

表 1 に構文の規則を示します。

表 *1:* このマニュアルでのフォントと構文規則

要素	例
コマンド名、プロシージャ名、ユーティリティ名、その他のキーワードは sans serif フォントで表記する。	select sp_configure
データベース名とデータ型は sans serif フォントで表記する。	master データベース
ファイル名、変数、パス名は斜体で表記する。	システム管理ガイド *sql.ini* ファイル *column_name* *$SYBASE/ASE* ディレクトリ

変数 (ユーザが入力する値を表す語) がクエリまたは文の一部である場合は Courier フォントの斜体で表記する。	`select column_name`
	`from table_name`
	`where search_conditions`

カッコはコマンドの一部として入力する。	compute *row_aggregate* (*column_name*)

要素	例
2 つのコロンと等号は、構文が BNF 表記で記述されていることを示す。この記号は入力しない。「～と定義されている」ことを意味する。	`::=`
中カッコは、その中のオプションを 1 つ以上選択しなければならないことを意味する。コマンドには中カッコは入力しない。	`{cash, check, credit}`
角カッコは、オプションを選択しても省略してもよいことを意味する。コマンドには角カッコは入力しない。	`[cash | check | credit]`
中カッコまたは角カッコの中のカンマで区切られたオプションをいくつでも選択できることを意味する。複数のオプションを選択する場合には、オプションをカンマで区切る。	`cash, check, credit`
パイプまたは縦線は複数のオプションのうち 1 つだけを選択できることを意味する。	`cash | check | credit`
省略記号 (...) は、直前の要素を必要な回数だけ繰り返し指定できることを意味する。	`buy thing = price [cash | check | credit] [, thing = price [cash | check | credit]]...` この例では、製品 (thing) を少なくとも 1 つ購入 (buy) し、価格 (price) を指定する必要があります。支払方法を選択できる。角カッコで囲まれた項目の 1 つを選択する。追加品目を、必要な数だけ購入することもできる。各 buy に対して、購入した製品 (thing)、価格 (price)、オプションで支払方法 (cash、check、credit のいずれか) を指定します。

- 次は、オプション句のあるコマンドの構文の例です。

  sp_dropdevice [*device_name*]

  複数のオプションを持つコマンドの例を示します。

  select *column_name*
  from *table_name*
  where *search_conditions*

  構文では、キーワード (コマンド) は通常のフォントで表記し、識別子は小文字で表記します。ユーザが提供するワードは斜体で表記します。

- Transact-SQL コマンドの使用例は次のように表記します。

  ```
  select * from publishers
  ```

- 次は、コンピュータからの出力例です。

```
pub_id    pub_name                city          state
-------   ---------------------   -----------   -----
0736      New Age Books           Boston        MA
0877      Binnet & Hardley        Washington    DC
1389      Algodata Infosystems    Berkeley      CA

(3 rows affected)
```

このマニュアルでは、例に使用する文字はほとんどが小文字ですが、Transact-SQL のキーワードを入力するときは、大文字と小文字は区別されません。たとえば、SELECT、Select、select はすべて同じです。

テーブル名などのデータベース・オブジェクトの大文字と小文字を Adaptive Server が区別するかどうかは、Adaptive Server にインストールされたソート順によって決まります。シングルバイト文字セットを使用している場合は、Adaptive Server のソート順を再設定することによって、大文字と小文字の区別の取り扱い方を変更できます。詳細については、『システム管理ガイド』を参照してください。

アクセシビリティ機能

このマニュアルには、アクセシビリティを重視した HTML 版もあります。この HTML 版マニュアルは、スクリーン・リーダーで読み上げる、または画面を拡大表示するなどの方法により、その内容を理解できるよう配慮されています。

Adaptive Server HTML マニュアルは、連邦リハビリテーション法第 508 条のアクセシビリティ規定に準拠していることがテストにより確認されています。第 508 条に準拠しているマニュアルは通常、World Wide Web Consortium (W3C) の Web サイト用ガイドラインなど、米国以外のアクセシビリティ・ガイドラインにも準拠しています。

注意 アクセシビリティ・ツールを効率的に使用するには、設定が必要な場合もあります。一部のスクリーン・リーダーは、テキストの大文字と小文字を区別して発音します。たとえば、すべて大文字のテキスト (ALL UPPERCASE TEXT など) はイニシャルで発音し、大文字と小文字の混在したテキスト (Mixed Case Text など) は単語として発音します。構文規則を発音するようにツールを設定すると便利かもしれません。詳細については、ツールのマニュアルを参照してください。

Sybase のアクセシビリティに対する取り組みについては、Sybase Accessibility (http://www.sybase.com/accessibility) を参照してください。Sybase Accessibility サイトには、第 508 条と W3C 標準に関する情報へのリンクもあります。

不明な点があるときは

Sybase ソフトウェアがインストールされているサイトには、Sybase 製品の保守契約を結んでいるサポート・センタとの連絡担当の方 (コンタクト・パーソン) を決めてあります。マニュアルだけでは解決できない問題があった場合には、担当の方を通して Sybase のサポート・センタまでご連絡ください。

第 1 章

システム・データ型とユーザ定義データ型

この章では、Transact-SQL データ型について説明します。データ型は、タイプ、サイズ、およびカラム、ストアド・プロシージャ・パラメータ、ローカル変数の記憶フォーマットを指定します。

データ型の種類

Adaptive Server には、数種類のシステム・データ型とユーザ定義のデータ型 (timestamp、sysname、および longsysname) が用意されています。表 1-1 は、Adaptive Server のデータ型の種類のリストを示します。それぞれの種類については、この章の各項で説明します。

表 *1-1:* データ型の種類

種類	用途
真数値データ型	正確に表示する必要のある数値 (整数値および小数値)
概数値データ型	算術演算中に丸めることができる数値データ
通貨データ型	通貨データ
timestamp データ型	Client-Library™ アプリケーションでブラウズするテーブル
日付と時刻のデータ型	日付および時刻の情報
文字データ型	文字、数字、記号で構成される文字列
バイナリ・データ型	ピクチャなどの、16 進数に似た表記のバイナリ・データ
bit データ型	true/false (真／偽) および yes/no 型のデータ
sysname データ型および longsysname データ型	システム・テーブル
text、image、unitext データ型	サーバの論理ページ・サイズの最大カラム・サイズよりも大きいカラムを必要とする、印刷可能な文字または 16 進数に似たデータ
抽象データ型	Adaptive Server は、Java クラスを介して抽象データ型をサポートする。詳細については、『Adaptive Server Enterprise における Java』を参照。
ユーザ定義データ型	この表にリストされているデータ型のルール、デフォルト、null 型、IDENTITY プロパティ、および基本データ型を継承するオブジェクトを定義する。クライアントで別の文字セットが使用されている場合に text には文字セットの変換が行われるが、image には行われない。

範囲と記憶サイズ

表 1-2 に、システムで用意されているデータ型とその同義語をリストし、各データ型の有効値の範囲と記憶サイズを示します。この表ではわかりやすくするために、データ型はすべて小文字で表記してありますが、Adaptive Server では、システム・データ型は大文字、小文字のどちらでも入力できます。timestamp などのユーザ定義データ型では大文字と小文字が区別されます。Adaptive Server のほとんどのデータ型は予約語ではないため、他のオブジェクトの名前として使用できます。

表 *1-2: Adaptive Server* のシステム・データ型

データ型 (種類別)	同義語	範囲	記憶サイズ (バイト数)
真数値：整数			
bigint		2^{63} ～ -2^{63} - 1 (-9,223,372,036,854,775,808 ～ +9,223,372,036,854,775,807) の間の整数値。	8
int	integer	2^{31} -1 (2,147,483,647) ～ -2^{31} (-2,147,483,648)	4
smallint		2^{15} -1 (32,767) ～ -2^{15} (-32,768)	2
tinyint		0 ～ 255 (負の数は使用できない)	1
unsigned bigint		0 ～ 18,446,744,073,709,551,615 の間の整数値。	8

データ型 (種類別)	同義語	範囲	記憶サイズ (バイト数)
unsigned int		0 ～ 4,294,967,295 の間の整数値。	4
unsigned smallint		0 ～ 65535 の間の整数値。	2
真数値：小数			
numeric (p, s)		10^{38} -1 ～ -10^{38}	2 ～ 17
decimal (p, s)	dec	10^{38} -1 ～ -10^{38}	2 ～ 17
概数値			
float (precision)		マシンに依存する	default precision が 16 未満の場合は 4、 default precision が 16 以上の場合は 8
double precision		マシンに依存する	8
real		マシンに依存する	4
通貨			
smallmoney		214,748.3647 ～ -214,748.3648	4
money		922,337,203,685,477.5807 ～ -922,337,203,685,477.5808	8
日付／時刻			
smalldatetime		1900 年 1 月 1 日 ～ 2079 年 6 月 6 日	4
datetime		1753 年 1 月 1 日 ～ 9999 年 12 月 31 日	8
date		0001 年 1 月 1 日 ～ 9999 年 12 月 31 日	4
time		12:00:00AM ～ 11:59:59:999PM	4
bigdatetime		0001 年 1 月 1 日 ～ 9999 年 12 月 31 日および 12:00.000000AM ～ 11:59:59.999999 PM	8
bigtime		12:00:00.000000 AM ～ 11:59:59.999999 PM	8
文字			
char(n)	character	ページサイズ	n
varchar(n)	character varying、char varying	ページサイズ	実際のエントリの長さ
unichar	Unicode 文字	ページサイズ	*n* * *@@unicharsize* (*@@unicharsize* は 2)
univarchar	Unicode 文字 varying、char varying	ページサイズ	実際の文字数 * *@@unicharsize*
nchar(n)	national character、national char	ページサイズ	*n* * *@@ncharsize*
nvarchar(n)	nchar varying、national char varying、national character varying	ページサイズ	*@@ncharsize* * 文字数
text		2^{31} -1 (2,147,483,647) バイト以下	初期化前は 0、初期化後は 2K の倍数

データ型 (種類別)　同義語	範囲	記憶サイズ (バイト数)
unitext	1 ～ 1,073,741,823	初期化前は 0、初期化後は 2K の倍数
バイナリ		
binary(n)	ページサイズ	*n*
varbinary(n)	ページサイズ	実際のエントリの長さ
image	2^{31} -1 (2,147,483,647) バイト以下	初期化前は 0、初期化後は 2K の倍数
ビット		
bit	0 または 1	1 (1 バイトで 8 つまでの bit カラムを保持する)

カラム、変数、パラメータのデータ型

カラム、ローカル変数、パラメータについては、データ型を宣言する必要があります。システムが提供するデータ型またはデータベース内のユーザ定義データ型の中から、任意のデータ型を宣言することができます。

テーブル内のカラムのデータ型の宣言

create table 文または alter table 文で新しいカラムのデータ型を宣言するには、次の構文を使用します。

```
create table [[database.]owner.]table_name
        (column_name datatype [identity | not null | null]
            [, column_name datatype [identity | not null |
                null]]...)
alter table [[database.]owner.]table_name
        add column_name datatype [identity | null
        [, column_name datatype [identity | null]...
```

たとえば、次のようにします。

```
create table sales_daily
    (stor_id char(4)not null,
     ord_num numeric(10,0)identity,
     ord_amt money null)
```

また、select into 文で convert または cast を使用して、新しいカラムのデータ型を宣言することもできます。

```
select convert(double precision, x), cast ( int, y) into
    newtable from oldtable
```

バッチまたはプロシージャ内のローカル変数のデータ型の宣言

バッチまたはプロシージャ内でローカル変数のデータ型を宣言するには、次の構文を使用します。

```
declare @variable_name datatype
        [, @variable_name datatype ]...
```

たとえば、次のようにします。

```
declare @hope money
```

ストアド・プロシージャ内のパラメータのデータ型の宣言

ストアド・プロシージャ内のパラメータのデータ型を宣言するには、次の構文を使用します。

```
create procedure [owner.]procedure_name [;number]
        [[(]@parameter_name datatype [= default] [output]
            [,@parameter_name datatype [= default]
                [output]]...[)]]
[with recompile]
as SQL_statements
```

次に例を示します。

```
create procedure auname_sp @auname varchar(40)
as
    select au_lname, title, au_ord
    from authors, titles, titleauthor
    where @auname = au_lname
      and authors.au_id = titleauthor.au_id
    and titles.title_id = titleauthor.title_id
```

リテラルのデータ型の決定

数字リテラル

E 表記で入力された数字リテラルは float として扱われます。そのほかのリテラルは、すべて真数値として扱われます。

- 小数点のない $2^{31} - 1 \sim -2^{31}$ のリテラルは、integer として扱われます。
- 小数点のあるリテラルまたは整数値の範囲外にあるリテラルは、numeric として扱われます。

注意　下位互換性を保つために、float として扱う必要のある数字リテラルには、E 表記を使用してください。

文字リテラル

12.5.1 より前のバージョンの Adaptive Server では、クライアントの文字セットがサーバの文字セットと異なる場合、SQL クエリのテキストを処理する前にサーバの文字セットに変換できるように、通常、変換が有効になっていました。サーバの文字セットで表せないために変換できない文字がある場合は、クエリ全体が拒否されました。Adaptive Server バージョン 12.5.1 からは、この文字セットの「ボトルネック」が解消されました。

文字リテラルのデータ型は宣言できません。Adaptive Server では、文字リテラルは varchar として扱われます。ただし、サーバのデフォルト文字セットに変換できない文字を含む文字リテラルは例外です。このようなリテラルは univarchar として扱われます。これにより、“sjis”(日本語) クライアントを使用して “iso_1” 用に設定されたサーバ内の unichar データを選択するクエリなどを実行できます。次に例を示します。

```
select * from mytable where unichar_column = '五'
```

文字リテラルは char データ型を使用して表せないため (“iso_1” の場合)、unichar データ型に拡大し、これによってクエリが成功します。

混合モードの式のデータ型

データ型の異なる値の連結または混合モードの算術演算を実行する場合、Adaptive Server によって結果のデータ型、長さ、精度が決定されます。

データ型階層の決定

各システム・データ型には、「データ型の階層」があります。この階層は systypes システム・テーブルに保管されています。ユーザ定義データ型は、もとになっているシステム・データ型の階層を継承します。

次のクエリは、データベースのデータ型を階層でランク付けします。クエリ結果には、次に示す情報のほかに、データベース内のユーザ定義データ型に関する情報も含まれます。

```
select name, hierarchy
    from systypes
    order by hierarchy
```

```
name           hierarchy
---------- ---------
floatn              1
float               2
datetimn            3
datetime            4
real                5
numericn            6
numeric             7
decimaln            8
decimal             9
moneyn             10
money              11
smallmoney         12
smalldatet         13
intn               14
uintn              15
bigint             16
ubigint            17
int                18
uint               19
smallint           20
usmallint          21
tinyint            22
bit                23
univarchar         24
unichar            25
unitext            26
sysname            27
varchar            27
nvarchar           27
longsysnam         27
char               28
nchar              28
timestamp          29
varbinary          29
binary             30
text               31
image              32
date               33
time               34
daten              35
timen              36
bigdatetim         37
bigtime            38
bigdatetim         39
bigtimen           40
extended t         99
```

注意 u<int 型 > は内部で使用される表現です。符号なし型の正しい構文は、unsigned {int | integer | bigint | smallint} です。

データ型階層は、異なるデータ型の値を使用して計算の結果を判別します。結果の値には、この階層リストの最上位に最も近い、つまり階層の値が最も小さいデータ型が割り当てられます。

次の例では、sales テーブルの *qty* に、roysched テーブルの royalty を乗算しています。qty は smallint であり、階層は 20 です。royalty は int であり、階層は 18 です。したがって、結果のデータ型は int になります。

```
smallint(qty) * int(royalty) = int
```

精度と位取りの決定

numeric データ型と decimal データ型の場合、精度と位取りの各組み合わせがそれぞれ別の Adaptive Server データ型になります。次の 2 つの numeric 値または decimal 値を対象に算術演算を行うとします。

- 精度が *p1*、位取りが *s1* の値 *n1*
- 精度が *p2*、位取りが *s2* の値 *n2*

この場合、Adaptive Server によって決定される結果の精度と位取りを表 1-3 に示します。

表 *1-3:* 算術演算後の精度と位取り

オペレーション	精度	位取り
n1 + n2	max(s1, s2) + max(p1 -s1, p2 - s2) + 1	max(s1, s2)
n1 - n2	max(s1, s2) + max(p1 -s1, p2 - s2) + 1	max(s1, s2)
n1 * n2	s1 + s2 + (p1 - s1) + (p2 - s2) + 1	s1 + s2
n1 / n2	max(s1 + p2 + 1, 6) + p1 - s1 + p2	max(s1 + p2 -s2 + 1, 6)

データ型の変換

あるデータ型から別のデータ型への変換は、多くの場合、Adaptive Server によって自動的に処理されます。このような自動変換を暗黙的な変換といいます。ほかの変換は、convert 関数、hextoint 関数、inttohex 関数、hextobigint 関数、および biginttohex 関数を使用して明示的に要求する必要があります。Adaptive Server でサポートしているデータ型の変換については、「データ型変換関数」(57 ページ) を参照してください。

固定長 NULL カラムの自動変換

null 値を格納できるのは、可変長データ型のカラムだけです。固定長データ型の NULL カラムを作成すると、Adaptive Server によって、そのカラムは対応する可変長データ型に自動的に変換されます。Adaptive Server は、データ型が変更されたことをユーザに通知しません。

表 1-4 は、固定長データ型とその変換後の可変長データ型をリストしたものです。moneyn, などの一部の可変長データ型は予約データ型です。カラム、変数、またはパラメータを作成するときにはこの予約データ型を使用できません。

表 *1-4:* 固定長データ型の自動変換

変換前の固定長データ型	変換後のデータ型
char	varchar
unichar	univarchar
nchar	nvarchar
binary	varbinary
datetime	datetimn
date	daten
time	timen
float	floatn
bigint、int、smallint 、tinyint	intn
unsigned bigint、unsigned int、unsigned smallint	uintn
decimal	decimaln
numeric	numericn
money、smallmoney	moneyn

オーバフロー・エラーとトランケーション・エラーの処理

算術演算エラーが発生した場合の Adaptive Server の動作を指定するには、arithabort オプションを使用します。2 つの arithabort オプションである arithabort arith_overflow と arithabort numeric_truncation によって、異なる 2 つのタイプの算術演算エラーの処理方法が決定します。各オプションを別々に設定したり、1 つの set arithabort on 文または set arithabort off 文に両オプションを設定したりできます。

- arithabort arith_overflow には、明示または暗黙のデータ型変換中に 0 による除算エラーや精度ロスが発生したあとの動作を指定します。このタイプのエラーは重大なエラーと見なされます。デフォルト設定の arithabort arith_overflow on では、エラーが発生したトランザクション全体がロールバックされます。トランザクションを含まないバッチでエラーが発生した場合、arithabort arith_overflow on はバッチ内のエラーより前のコマンドはロールバックしませんが、Adaptive Server はバッチ内のエラーを生成した文より後の文を実行しません。

  arith_overflow を on にする設定は、Adaptive Server に設定された正規化のレベルではなく、実行時に適用されます。

  arithabort arith_overflow off を設定した場合には、Adaptive Server はエラーを発生させた文をアボートしますが、トランザクションまたはバッチ内のその他の文の処理は継続します。

- arithabort numeric_truncation は、暗黙のデータ型変換中に真数値型による位取りのロスが発生した後の動作を指定します (明示的変換によって位取りのロスが発生すると、変換結果は警告なしにトランケートされます)。デフォルト設定の arithabort numeric_truncation on は、エラーを起こした文をアボートしますが、トランザクションまたはバッチ内のその他の文の処理を継続します。arithabort numeric_truncation off を設定した場合、Adaptive Server はクエリ結果をトランケートして処理を継続します。

arithignore オプションは、オーバフロー・エラー発生後に Adaptive Server が警告メッセージを出力するかどうかを指定します。デフォルトでは、arithignore オプションは off になります。この場合、Adaptive Server は数字オーバフローを起こしたクエリの後に、警告メッセージを表示します。オーバフロー・エラーを無視するには、set arithignore on を使用してください。

標準規格と準拠レベル

表 1-5 は、Transact-SQL データ型の ANSI SQL 標準規格と準拠レベルを示しています。

表 *1-5: Transact-SQL データ型の ANSI SQL 標準規格と準拠レベル*

Transact-SQL − ANSI SQL データ型	Transact-SQL 拡張機能 − ユーザ定義データ型
• char • varchar • smallint • int • bigint • decimal • numeric • float • real • date • time • double precision	• binary • varbinary • bit • nchar • datetime • smalldatetime • bigdatetime • bigtime • tinyint • unsigned smallint • unsigned int • unsigned bigint • money • smallmoney • text • unitext • image • nvarchar • unichar • univarchar • sysname • longsysname • timestamp

真数値データ型

値を正確に表す必要がある場合に、真数値データ型を使用します。Adaptive Server には、整数 (自然数) 値と小数値のための真数値データ型があります。

整数値型

Adaptive Server には、整数を保管するための真数値データ型 bigint、int (integer)、smallint、tinyint、およびこれらの各データ型の符号なしの型があります。保管する数値の予想サイズに基づいて、整数値型を選択してください。表 1-6 に示すように、内部記憶サイズは型によって異なります。

表 *1-6:* 整数値データ型

データ型	保管するデータ	記憶サイズ (バイト数)
bigint	-2^{63} ～ 2^{63} - 1 (-9,223,372,036,854,775,808 ～ +9,223,372,036,854,775,807) の間の整数値。	8
int[eger]	-2^{31} ～ 2^{31} - 1 (-2,147,483,648 ～ 2,147,483,647) の間の整数値。	4
smallint	-2^{15} ～ 2^{15} - 1 (-32,768 ～ 32,767) の間の整数値。	2
tinyint	0 ～ 255 の間の整数値 (負の数値は使用できない)。	1
unsigned bigint	0 ～ 18,446,744,073,709,551,615 の間の整数値。	8
unsigned int	0 ～ 4,294,967,295 の間の整数値。	4
unsigned smallint	0 ～ 65,535 の間の整数値。	2

整数データの入力

整数データは、カンマを含まない数字列として入力します。小数点の右側のすべての桁がゼロであれば、整数データに小数点を含めることができます。smallint、integer、bigint データ型の数値の前に、オプションでプラス符号またはマイナス符号を付けることができます。tinyint データ型の数値の前に、オプションでプラス符号を付けることができます。

表 1-7 に、integer データ型のカラムに有効な入力値と、isql がこれらの値をどのように表示するかを示します。

表 *1-7:* 有効な整数値

入力値	表示値
2	2
+2	2
-2	-2
2.	2
2.000	2

表 1-8 に、integer カラムの無効な入力値を示します。

表 *1-8:* 無効な整数値

入力値	エラー・タイプ
2,000	カンマは使用できない。
2-	マイナス符号は数字の前になければならない。
3.45	小数点の右側の数字がゼロ以外の数字である。

decimal データ型

Adaptive Server では、上記のほかに、小数点を含む数値のための 2 つの真数値データ型 numeric と dec[imal] が用意されています。numeric と decimal の 2 つのデータ型は、IDENTITY カラムには位取り 0 の numeric データ型と integer データ型だけが使用できるという点を除いて、同一です。

精度と位取りの指定

numeric と decimal データ型は、2 つのオプション・パラメータである precision と scale を受け入れます。これらのパラメータを指定するときには、パラメータをカッコで囲み、カンマで区切ります。

datatype [(*precision* [, *scale*])]

Adaptive Server は、精度と位取りの各組み合わせを別個のデータ型として扱います。たとえば numeric(10,0) と numeric(5,0) は、2 つの別々のデータ型です。precision と scale は、decimal カラムまたは numeric カラムに保管できる値の範囲を決定します。

- 精度 (precison) は、カラムに格納できる 10 進数値の最大桁数を指定します。この桁数には、小数点の左右両側にあるすべての桁が含まれます。1 桁から 38 桁までの精度を指定するか、またはデフォルトの精度である 18 桁を使用します。
- 位取りは、小数点の右側に格納できる桁数の最大数を指定します。位取りの桁数は精度の桁数以下でなければなりません。0 桁から 38 桁までの位取りを指定するか、またはデフォルトの位取りである 0 桁を使用します。

記憶サイズ

numeric または decimal カラムの記憶サイズは、その精度によって異なります。1 桁または 2 桁のカラムの場合、記憶領域は 2 バイト必要です。記憶サイズは精度が 2 桁増加するたびに約 1 バイトずつ増加します。最大記憶サイズは 17 バイトです。

numeric または decimal カラムの正確な記憶サイズを計算するには、次の式を使います。

```
ceiling (precision / log10(256)) + 1
```

たとえば、numeric(18,4) カラムの記憶サイズは 9 バイトです。

10 進数データの入力

decimal データおよび numeric データは、数字列として入力します。このとき、オプションでプラス符号またはマイナス符号を数値の前に付けたり、オプションで小数点を含めたりできます。入力された値がカラムに指定されている精度または位取りを超えている場合、Adaptive Server はエラー・メッセージを返します。位取りが 0 の真数型は小数点なしで表示されます。

表 1-9 に、データ型が numeric(5,3) のカラムに有効な入力値と、isql によってこれらの値がどのように表示されるかを示します。

表 *1-9:* 有効な *10* 進数値

入力値	表示値
12.345	12.345
+12.345	12.345
-12.345	-12.345
12.345000	12.345
12.1	12.100
12	12.000

表 1-10 に、データ型が numeric(5,3) のカラムに無効な入力値を示します。

表 *1-10:* 無効な *10* 進値

入力値	エラー・タイプ
1,200	カンマは使用できない。
12-	マイナス符号は数字の前になければならない。
12.345678	小数点の右側にゼロ以外の数字が多すぎる。

標準規格と準拠レベル

Transact-SQL には、ANSI SQL 真数値データ型の smallint、int、bigint、numeric、decimal があります。unsigned bigint、unsigned int、unsigned smallint、tinyint 型は、Transact-SQL 拡張機能です。

概数値データ型

丸めることができる数値データには、概数値型の float、double precision、real を使用できます。概数値型は、値が広範囲にわたるデータに特に適しています。概数値型は、すべての集合関数とすべての算術演算をサポートします。

概数値データ型について

概数値データ型は、浮動小数点の値を保管するために使われるもので、実数表現では多少不正確になります。このために「概数値」と呼ばれます。このデータ型を使う場合には、その制限事項をよく理解してください。これらのエントリを追加するには、次の情報が必要です。

浮動小数点の値の出力や表示のときには、出力された値は保管された値と正確には一致しません。また、保管された値は、ユーザが入力した値とは正確には一致しません。多くの場合、保管された値は十分に正確なものであり、ソフトウェアで出力された値は最初に入力された値と同じように表示されます。しかし浮動小数点を計算に使う場合、特に概数値を使って繰り返し計算する場合には、その結果が不正確なものになることを理解しておいてください。計算結果は、予想外に不正確なものになることがあります。

結果が不正確となるのは、コンピュータでは浮動小数点はバイナリ分数として (つまり、表示された値を 2 の累乗で割った値として) 保管されますが、実際に使用される値は、10 進数値 (10 の累乗の値) であるためです。このため、ごくわずかな数の数値の集合しか、正確に保管できないことになります。0.75 (3/4) はバイナリ分数なので、正確に保管することができます (4 は 2 の累乗です)。しかし 0.2 (2/10) は正確に保管することはできません (10 は 2 の累乗ではありません)。

桁数が多すぎて、正確に保管できない数値もあります。double precision は 8 バイナリ・バイトで保管され、かなり正確に約 17 桁の数値を表示できます。real は 4 バイナリ・バイトで保管され、かなり正確に表示できるのは、約 6 桁までです。

ほぼ正確な数値を使って作業を始め、別のほぼ正確な数値を使って計算しても、まったく予想外の結果になることがあります。アプリケーションにおいてこの問題が重要な場合には、真数値データ型を使用してください。

範囲、精度、記憶サイズ

real および double precision 型は、オペレーティング・システムが提供する型を基に構築されます。float データ型には、オプションのバイナリ精度をカッコで囲んで指定できます。精度が 1 ～ 15 の float カラムは、real として格納されます。これより高い精度の場合は、double precision として格納されます。

3 つのデータ型のいずれの場合も、範囲および記憶精度はマシンによって異なります。

表 1-11 に、個々の概数値型の範囲と記憶サイズを示します。isql は、小数点以下の有効桁数を 6 桁しか表示せず、残りを丸めます。

表 *1-11:* 概数値データ型

データ型	記憶サイズ (バイト数)
float[(default precision)]	default precision < 16 の場合、4 default precision >= 16 の場合、8
double precision	8
real	4

概数値データの入力

概数値データは、仮数の後ろにオプションの指数を付けて入力します。

- 仮数は符号付き、または符号なしの数字です。小数点はあってもなくてもかまいません。カラムのバイナリ精度は、仮数部に使用できるバイナリ桁数の最大数を決定します。
- “e” または ”E” で始まる指数は、整数です。

入力値が表す値は次の式の積です。

```
仮数 * 10 指数
```

たとえば、2.4E3 は値 2.4 x 10^3、すなわち 2400 です。

NaN 値と Inf 値

“NaN” と “Inf” は、IEEE754/854 浮動小数点標準でそれぞれ “not a number” (数値以外) と “infinity” (無限大) を表すために使用される特殊な値です。ANSI SQL92 標準規格に従い、Adaptive Server バージョン 12.5 以降では、これらの値をデータベースに挿入することも、生成することもできません。Adaptive Server 12.5 より前のバージョンでは、ネイティブ・モードの bcp、JDBC、ODBC などの Open Client クライアントによってこれらの値が強制的にテーブルに格納される場合がありました。

データベースで NaN 値または Inf 値が検出された場合は、その問題の再現方法に関する詳細を含めて Sybase カスタマ・サポートまでご連絡ください。

標準規格と準拠レベル

ANSI SQL － 準拠レベル：float、double precision、real データ型は初級レベルの標準に準拠します。

通貨データ型

通貨データを保管するには、money データ型と smallmoney データ型を使用します。これらのデータ型は米国ドルなどの 10 進数値の通貨に使用できますが、Adaptive Server には、ある通貨を別の通貨に変換する機能はありません。money および smallmoney データには、モジュロを除く算術演算と、すべての集合関数を使用できます。

正確度

money および smallmoney は、いずれも通貨単位の 10,000 分の 1 まで正確ですが、表示される場合は、値が小数点以下 2 桁まで丸められます。デフォルトの出力フォーマットでは、3 桁ごとにカンマが挿入されます。

範囲と記憶サイズ

表 1-12 に、通貨データ型の範囲と記憶サイズを示します。

表 *1-12:* 通貨データ型

データ型	範囲	記憶サイズ (バイト数)
money	+922,337,203,685,477.5807 と -922,337,203,685,477.5808 の間の通貨値	8
smallmoney	+214,748.3647 と 214,748.3648 の間の通貨値	4

通貨値の入力

E 表記で入力した通貨値は float として解釈されます。このため、値が money または smallmoney 値として格納されると、入力が拒否されたり、精度が若干失われたりすることがあります。

money および smallmoney 値を入力するときには、ドル記号 ($)、円記号 (¥)、通貨ポンド記号 (£) などの通貨記号を数値の前に付けても、付けなくてもかまいません。負の値を入力する場合には、通貨記号の後にマイナス符号を入力してください。入力にはカンマは含めないでください。

標準規格と準拠レベル

ANSI SQL － money データ型および smallmoney データ型は Transact-SQL 拡張機能です。

timestamp データ型

Client-Library™ アプリケーションでブラウズするテーブル (詳細については「ブラウズ・モード」を参照) には、ユーザ定義データ型の timestamp を使用します。Adaptive Server は、ローが変更されるたびに、その timestamp カラムを更新します。各テーブルには、timestamp データ型のカラムを 1 つだけ含めることができます。

timestamp カラムの作成

timestamp という名前のカラムを作成し、このカラムにデータ型を指定しないと、自動的にこのカラムが timestamp データ型として定義されます。

```
create table testing
    (c1 int, timestamp, c2 int)
```

次の例に示すように、timestamp という名前のカラムに timestamp データ型を明示的に割り当てることもできます。

```
create table testing
    (c1 int, timestamp timestamp, c2 int)
```

カラム名は他の名前でもかまいません。

```
create table testing
    (c1 int, t_stamp timestamp,c2 int)
```

timestamp という名前のカラムを作成し、別のデータ型を割り当ててもかまいません (ただし、これは他のユーザにとってはまぎらわしく、Open Client™ の browse 関数や tsequal 関数を使用することはできません)。

```
create table testing
    (c1 int, timestamp datetime)
```

日付と時刻のデータ型

絶対日付と時刻の情報を保管するには、datetime、smalldatetime、bigdatetime、bigtime、date、および time を使用します。バイナリ型の情報を保管するには、timestamp を使用します。

Adaptive Server には、日付と時刻の値の保管に使用するさまざまなデータ型が用意されています。

- date
- time
- smalldatetime
- datetime

- bigdatetime
- bigtime

日付のデフォルトの表示フォーマットは、“Apr 15 1987 10:23PM” です。bigdatetime/bigtime 型のデフォルトの表示フォーマットは、“Apr 15 1987 10:23:00.000000PM” です。他のスタイルの日付の表示には、convert 関数を使用できます。組み込み日付関数を使用して、date 値および time 値についていくつかの算術計算を実行することもできますが、Adaptive Server はミリ秒の値を丸めるか、またはトランケートする場合があります。

- datetime カラムは、1753 年の 1 月 1 日から 9999 年 12 月 31 日までの日付を保持します。datetime 値は、プラットフォームの能力が対応可能であれば、300 分の 1 秒のレベルまで正確です。記憶サイズは 8 バイトです。基本の日付である 1900 年 1 月 1 日以降の日数に 4 バイト、1 日の時刻に 4 バイトを使用します。
- smalldatetime カラムは、1900 年 1 月 1 日から 2079 年 6 月 6 日までの日付を、分の単位まで正確に保持します。記憶サイズは 4 バイトです。1900 年 1 月 1 日以降の日数に 2 バイト、夜中の 12 時以降の分数に 2 バイトを使用します。
- bigdatetime カラムは、0001 年 1 月 1 日から 9999 年 12 月 31 日までの日付および 12:00:00.000000 AM から 11:59:59.999999 PM までの時刻を保持します。記憶サイズは 8 バイトです。bigdatetime の内部で使用される表現は、0000 年 1 月 1 日以降のマイクロ秒数を含む 64 ビットの整数です。
- bigtime カラムは、12:00:00.000000 AM から 11:59:59.999999 PM までの時刻を保持します。記憶サイズは 8 バイトです。bigtime の内部で使用される表現は、午前 0 時以降のマイクロ秒数を含む 64 ビットの整数です。
- date カラムは、0001 年 1 月 1 日から 9999 年 12 月 31 日までの日付を保持します。記憶サイズは 4 バイトです。
- time は 00:00:00:000 から 23:59:59:999 の間です。24 時間制、または正午を 12AM、真夜中を 12PM とする 12 時間制のどちらでも使用できます。時刻値には、コロンか AM/PM 指示子を含めてください。AM または PM は、大文字でも小文字でもかまいません。

日付と時刻の情報を入力する場合は、時刻または日付を常に一重または二重の引用符で囲んでください。

範囲と記憶サイズ

表 1-13 に、datetime、smalldatetime、bigdatetime、bigtime、date、time の各データ型の範囲と記憶サイズを示します。

表 *1-13:* 日付と時刻を保管する *Transact-SQL* データ型

データ型	範囲	記憶サイズ (バイト数)
datetime	1753 年 1 月 1 日～ 9999 年 12 月 31 日	8
smalldatetime	1900 年 1 月 1 日～ 2079 年 6 月 6 日	4
bigdatetime	0001 年 1 月 1 日 ～ 9999 年 12 月 31 日	8
bigtime	12:00:00.000000AM ～ 11:59:59.999999PM	8
date	0001 年 1 月 1 日 ～ 9999 年 12 月 31 日	4
time	12:00:00 AM ～ 11:59:59:999 PM	4

日付および時刻データの入力

datetime、smalldatetime、bigdatetime、bigtime の各データ型は、日付部分と、その前後の時刻部分からなります (日付または時刻のいずれか、あるいは両方を省略できます)。date データ型は日付のみを含み、time データ型は時刻のみを含みます。値は、一重または二重引用符で囲む必要があります。

日付の入力

日付は月、日、年で構成され、date、datetime、bigdatetime、bigtime、および smalldatetime に対してさまざまなフォーマットで入力できます。

- 日付全体を、セパレータで区切らずに 4 桁、6 桁、または 8 桁の数字列で入力したり、日付の各要素をセパレータのスラッシュ (/)、ハイフン (-)、またはピリオド (.) で区切って入力したりすることもできます。
  - 日付を区切りのない数字列として入力する場合、その数字列の長さに適したフォーマットを使用してください。1 桁の年、月、日の前には、先行ゼロを入力してください。間違ったフォーマットで日付を入力すると、誤って解釈されたり、エラーを引き起こしたりします。
  - 日付をセパレータで区切って入力する場合、set dateformat オプションを使って、日付要素の順序を決定してください。区切られた数字列の最初の日付部分が 4 桁の場合、Adaptive Server ではその数字列を *yyyy-mm-dd* のフォーマットとして解釈します。
- 日付フォーマットの中には、2 桁の年 (*yy*) を使用するものもあります。
  - 50 未満の数値は 20*yy* として解釈されます。たとえば、`01` は 2001、`32` は 2032、`49` は 2049 です。
  - 50 以上の数値は 19*yy* と解釈されます。たとえば、`50` は 1950、`74` は 1974、`99` は 1999 です。
- 月の指定には、数字または名前を使用できます。月名とその省略形は言語ごとに異なります。月名は大文字でも小文字でも、また両方を組み合わせても入力できます。

- datetime 値または smalldatetime 値の日付部分を省略すると、Adaptive Server はデフォルト日付の 1900 年 1 月 1 日を使用します。bigdatetime の日付部分を省略すると、デフォルト値の 0001 年 1 月 1 日が追加されます。

表 1-14 に、datetime または smalldatetime 値の日付部分を入力するのに使用できるフォーマットを示します。

表 *1-14:* 日付と時刻のデータ型の日付フォーマット

日付フォーマット	解釈	入力例	意味
セパレータのない 4 桁の文字列	*yyyy* と解釈される。日付のデフォルトは、指定の年の 1 月 1 日。	“1947”	1947 年 1 月 1 日
セパレータのない 6 桁の文字列	*yymmdd* と解釈される。 *yy* < 50 の場合、年は 20*yy* となる。 *yy* >= 50 の場合、年は 19*yy* になる。	“450128” “520128”	2045 年 28 月 1 日 1952 年 28 月 1 日
セパレータのない 8 桁の文字列	*yyyymmdd* と解釈される。	“19940415”	1994 年 4 月 15 日
スラッシュ、ハイフン、ピリオド、またはこれらの組み合わせで区切った 2 桁ずつの月、日、年で構成される文字列	日付要素の順序は dateformat と language set オプションによって決まる。us_english の場合、デフォルト順序は *mdy*。 *yy* < 50 の場合、年は 20*yy* と解釈される。*yy* >= 50 の場合、年は 19*yy* と解釈される。	“4/15/94” “4.15.94” “4-15-94” “04.15/94”	dateformat オプションが mdy に設定されている場合、これらの入力値はすべて 1994 年 4 月 15 日と解釈される。
スラッシュ、ハイフン、ピリオド、またはこれらの組み合わせで区切った 2 桁の月、2 桁の日、4 桁の年で構成される文字列	日付要素の順序は dateformat と language set オプションによって決まる。us_english の場合、デフォルト順序は *mdy*。	“04/15.1994”	dateformat オプションが mdy に設定されている場合、1994 年 4 月 15 日と解釈される。
月はオプションで後ろにカンマを付けて文字形式で入力 (月のフルネームまたは標準省略名)	4 桁の年を入力する場合、日付要素は任意の順序で入力できる。	“April 15, 1994” “1994 15 apr” “1994 April 15” “15 APR 1994”	これらの入力値はすべて 1994 年 4 月 15 日と解釈される。
	日を省略する場合、年は 4 桁全体を指定する必要がある。デフォルトの日付は月の最初の日。	“apr 1994”	1994 年 4 月 1 日
	年が 2 桁だけ (*yy*) である場合、年は日の後ろに位置するものと見なされる。*yy* <= 50 の場合、年は 20*yy* と解釈される。*yy* >= 50 の場合、年は 19*yy* と解釈される。	“mar 16 17” “apr 15 94”	2017 年 3 月 16 日 1994 年 4 月 15 日
空の文字列 “”	デフォルト日付は 1900 年 1 月 1 日。	“”	1900 年 1 月 1 日

時刻の入力

datetime 値、smalldatetime 値、または time 値の時刻部分は次のように指定してください。

```
hours[:minutes[:seconds[:milliseconds]] [AM | PM]
```

bigdatetime 値または bigtime 値の時刻部分は次のように指定してください。

```
hours[:minutes[:seconds[.microseconds]] [AM | PM]
```

- 午前 0 時には 12AM を、正午には 12PM を使用してください。
- 時刻値には、コロンか AM/PM 指示子を含めてください。AM と PM は大文字でも小文字でも、または両者を組み合わせても入力できます。
- 秒指定には、小数点のあとに小数部分を、またはコロンのあとにミリ秒数を含めることができます。たとえば、“15:30:20:1” は午後 3 時 30 分から 20 秒と 1 ミリ秒が過ぎた時刻を示し、“15:30:20.1” は午後 3 時 30 分から 20.1 秒が過ぎた時刻を示します。ミリ秒は小数点で表す必要があります。
- datetime 値または smalldatetime 値の時刻部分を省略すると、デフォルト時刻の 12:00:00:000AM が使用されます。

datetime 値、*smalldatetime* 値、*date* 値の表示フォーマット

datetime 値と smalldatetime 値の表示フォーマットは、“Mon dd yyyy hh:mmAM” (または “PM” です。たとえば、“Apr 15 1988 10:23PM” のように表示されます。秒とミリ秒を表示したり、他の日付スタイルや日付部分の順序を指定したりするには、convert 関数を使用して、日付を文字列に変換してください。Adaptive Server によってミリ秒値が丸められるか、トランケートされます。

表 1-15 に、datetime の入力例と表示される値を示します。

表 *1-15: datetime* と *date* の入力例

入力値	表示値
“1947”	Jan 1 1947 12:00AM
“450128 12:30:1PM”	Jan 28 2045 12:30PM
“12:30.1PM 450128”	Jan 28 2045 12:30PM
“14:30.22”	Jan 1 1900 14:30:00
“4am”	Jan 1 1900 04:00:00
日付の例	
“1947”	1947 年 1 月 1 日
“450128”	2045 年 28 月 1 日
“520317”	Mar 17 1952

bigdatetime と bigtime の表示フォーマット

bigdatetime および bigtime の場合、表示される値はマイクロ秒の精度で示されます。bigdatetime と bigtime には、この増えた精度に対応するデフォルトの表示フォーマットがあります。

- hh:mm:ss.zzzzzzAM または PM
- hh:mm:ss.zzzzzz
- mon dd yyyy
  hh:mm:ss.zzzzzzAM(PM)
- mon dd yyyy
  hh:mm:ss.zzzzzz
- yyyy-mm-dd
  hh:mm:ss.zzzzzz

時刻のフォーマットは次のように指定する必要があります。

hours[:minutes[:seconds[.microseconds]] [*AM* | *PM*]
hours[:minutes[:seconds[number of milliseconds]] [*AM* | *PM*]

午前 0 時には 12AM を、正午には 12PM を使用してください。bigtime 値には、コロンか AM/PM 指示子を含めてください。AM と PM は大文字でも小文字でも、または両者を組み合わせても入力できます。

秒指定には、小数点のあとに小数部分を、またはコロンのあとにミリ秒数を含めることができます。たとえば、“12:30:20:1” は午後 12 時 30 分から 20 秒と 1 ミリ秒が過ぎた時刻を示し、“12:30:20.1” は午後 12 時 30 分から 20.1 秒が過ぎた時刻を示します。

ミリ秒が含まれる bigdatetime または bigtime 時刻値を格納するには、小数点を使用したリテラル文字列を指定します。“00:00:00.1” は午前 0 時から 10 分の 1 秒が経過したことを示し、“00:00:00.000001” は午前 0 時から 100 万分の 1 秒が経過したことを示します。コロンの後の小数秒を指定する値は、ミリ秒数を示します。たとえば、“00:00:00:5” は 5 ミリ秒を意味します。

time 値の表示フォーマット

time 値の表示フォーマットは “hh:mm:ss:mmmAM” (または “PM” です。たとえば、“10:23:40:022PM” のように表示されます。

表 *1-16: time* の入力例

入力値	表示値
"12:12:00"	12:12PM
“01:23PM” または “01:23:1PM”	1:23PM
“02:24:00:001”	2:24AM

パターンに一致する値の検索

特定のパターンに一致する日付を検索するには、like キーワードを使用してください。等号演算子 (=) を使用して日付や時刻の値の特定の月、日、年を検索すると、Adaptive Server は時刻が正確に 12:00:00:000AM である値だけを返します。

たとえば、arrival_time という名前のカラムに値 “9:20” を挿入すると、Adaptive Server ではこのエントリが “Jan 1, 1900 9:20AM.” に変換されるため、次の句ではこの値を検出できません。等号演算子を使用してこの入力値を検索しても、この値は見つかりません。

```
where arrival_time = "9:20" /* does not match */
```

like 演算子を使用すれば、この入力値は見つかります。

```
where arrival_time like "%9:20%"
```

like を使用すると、日付はまず datetime または date フォーマットに変換され、次に varchar に変換されます。表示フォーマットは、現在の言語で 3 文字の月名、2 文字の日、4 文字の年、時刻の時と分、“AM” または “PM” で構成されます。

datetime 値、smalldatetime 値、bigdatetime 値、bigtime 値、date 値、および time 値の日付部分の入力には、多種類の入力フォーマットを使用できますが、like を使用して検索するときは、こうした多種類の入力フォーマットは使用できません。*style* 9 または 109 と convert 関数を併用しない限り、like と一致パターンで秒数やミリ秒数を検索することはできません。

like を使用していて、月の日付が 1 ～ 9 の場合、datetime 値の変換後の varchar 値に一致させるために、月と日の間にスペースを 2 つ挿入します。同様に、時間が 10 未満である場合、変換すると年と時間の間に 2 つのスペースが入ります。次の句 (“May” と “2” の間に 1 つのスペースがあります) では、5 月 20 日から 5 月 29 日までのすべての日付が検索されますが、5 月 2 日は検索されません。

```
like "May 2%"
```

その他の日付比較では、余分なスペースを挿入する必要はありません。datetime 値が varchar に変換されるのは like 比較の場合だけなので、like 比較を実行するときにだけ、スペースを挿入してください。

日付の操作

組み込み日付関数を使用して、date データ型と time データ型の値についていくつかの算術計算を実行できます。「日付関数」(65 ページ) を参照してください。

標準規格と準拠レベル

ANSI SQL － 準拠レベル：datetime データ型および smalldatetime データ型は Transact-SQL 拡張機能です。date データ型および time データ型は初級レベルの標準に準拠します。

文字データ型

特定の状況で使用するデータ型は、保管するデータの型によって異なります。

- 文字、数字、記号からなる文字列を保管するには、文字データ型を使用します。
- varchar(n) と char(n) は、us_english などのシングルバイト文字セットと日本語などのマルチバイト文字の両方に使用します。
- unichar(n) データ型と univarchar(n) データ型は、Unicode 文字の保管に使用します。これらは、1 文字に固定のバイト数が必要な場合、シングルバイトまたはマルチバイトの文字に利用できます。
- シングルバイト文字セットと日本語などのマルチバイト文字セットの両方には、固定長データ型の nchar(n) と可変長データ型の nvarchar(n) を使用します。nchar(n) と char(n)、および nvarchar(n) と varchar(n) の違いは、nchar(n) と nvarchar(n) では、1 文字あたりのバイト数 (デフォルト文字セットよって異なる) の *n* 倍に基づいて記憶領域が割り付けられるのに対し、char(n) と varchar(n) では、*n* バイトの記憶領域が割り付けられる点です。
- 文字データ型には、最大でページ・サイズ分のデータを保管できます。
- char データ型または varchar データ型で許可されているより長い文字列を保管するには、text データ型 (詳細については 「text、image、unitext データ型」(33 ページ) を参照) またはサブテーブル内の複数のローを使用します。

unichar、univarchar

char 文字データ型および varchar 文字データ型が使用できる場所であれば、構文を変更することなく、unichar データ型および univarchar データ型をどこでも使用できます。

Adaptive Server バージョン 12.5.1 以降では、サーバの文字セットで表現できない文字リテラルを含むクエリは自動的に unichar データ型に拡大されるので、データ操作言語 (DML) 文の構文を変更する必要はありません。文字リテラルに任意の文字を指定するには追加の構文も使用できますが、リテラルを unichar に「拡大」する決定は、サーバの文字セットで表現できるかどうかのみに基づいています。

データ定義言語 (DDL) 文では、構文を変更する必要性は最低限です。たとえば、create table コマンドでは、Unicode カラムのサイズはバイトではなく 16 ビットの Unicode 値の単位で指定されます。そのため、char(200) と unichar(200) の類似性が保持されます。カラムの長さをレポートする sp_help は、同じ単位を使用します。乗算因子 (2) は新しいグローバル変数 *@@unicharsize* に保管されます。

Unicode の詳細については、『システム管理ガイド』の「第 8 章 文字セット、ソート順、言語の設定」を参照してください。

文字単位の長さと記憶サイズ

文字変数は、カーソルの varchar カラムに移植されるときに文字列から後続スペースを取り除きます。

char データ型と varchar データ型の記憶領域のバイト数を指定するには、*n* を使用します。unichar の場合は、*n* を使用して Unicode 文字数を指定します (割り付けられる記憶領域は各文字に対して 2 バイトです)。nchar と nvarchar の場合、*n* は文字数です (割り付けられる記憶領域は、サーバの現在のデフォルトの文字セットにおける、各文字のバイト数の *n* 倍です)。

長さの指定に *n* を使用しない場合は、次の点に注意してください。

- create table、alter table で作成されたカラム、および declare で作成された変数のデフォルトの長さは 1 バイトである。
- convert 関数で作成された値のデフォルトの長さは 30 バイトである。

指定の長さより短い入力値にはブランクが埋め込まれます。指定の長さより長い入力値は、set コマンドの string_rtruncation オプションが on に設定されている場合を除き、警告が出ずにトランケートされます。null 値を保管できる固定長カラムは、内部で可変長カラムに変換されます。

可変長データ型の varchar(n) 、univarchar(n)、nvarchar(n) では、*n* に文字単位で最大長を指定します。可変長カラムのデータの後続ブランクは削除されます。したがって、記憶サイズは入力されたデータの実際の長さです。可変長変数およびパラメータのデータの場合、後続ブランクはすべて保持されますが、定義されている長さになるまでブランクが埋め込まれることはありません。文字リテラルは可変長データ型として扱われます。

固定長カラムには、一般に、可変長カラムより大きな記憶領域が必要ですが、アクセスにかかる時間は多少短くなります。表 1-17 に、各文字データ型に必要な記憶サイズを示します。

表 *1-17:* 文字データ型

データ型	保管するデータ	記憶サイズ (バイト数)
char(n)	文字	*n*
unichar(n)	Unicode 文字	*n**@@*unicharsize* (@@*unicharsize* は 2)
nchar(n)	各国文字	*n* * @@*ncharsize*
varchar(n)	可変長文字	実際に入力した文字数
univarchar(n)	可変長の Unicode 文字	実際の文字数 * @@*unicharsize*
nvarchar(n)	可変長の各国文字	実際の文字数 * @@*ncharsize*

システム関数によるカラム長の決定

カラムの長さを決定するには、char_length 文字列関数と datalength システム関数を使用します。

- char_length は、カラムの文字数を返し、可変長データ型の後続ブランクを削除します。
- datalength　は、可変長カラムに保管されたデータの後続ブランクを削除し、バイト数を返します。

char 値に NULL 値 を使用できることが宣言されている場合、Adaptive Server 内部ではこの値は varchar として保管されます。

char カラムに対して min 集合関数または max 集合関数が実行されると、結果として varchar の値が返されます。このため、返された値では後続スペースがすべて削除されています。

文字データの入力

文字列は必ず一重引用符または二重引用符で囲みます。set quoted_identifier on を使用している場合は、文字列を一重引用符で囲んでください。そうしないと、Adaptive Server は文字列を識別子として扱います。

二重引用符が含まれている文字列は一重引用符で囲んでください。一重引用符が含まれている文字列は二重引用符で囲んでください。たとえば、次のようにします。

```
'George said, "There must be a better way."'
"Isn't there a better way?"
```

もう 1 つの方法として、文字列に含めたい引用符がある場合に、その引用符を 2 つ連続して入力する方法があります。たとえば、次のようにします。

```
"George said, ""There must be a better way."""
'Isn''t there a better way?'
```

文字列を画面の次の行に続けるには、次の行に進む前にバックスラッシュ (\) を入力してください。

引用符で囲む識別子の詳細については、『Transact SQL ユーザーズ・ガイド』の「区切り識別子」を参照してください。

Unicode 文字の入力

オプションの新しい構文を使用して、任意の Unicode 文字を指定できます。文字リテラルの直前に U& または u& (間にスペースなし) がある場合、パーサはリテラル内のエスケープ・シーケンスを認識します。¥xxxx (xxxx は 4 桁の 16 進数を表す) の形式のエスケープ・シーケンスは、スカラ値が xxxx の Unicode 文字に置き換えられます。同様に、¥+yyyyyy の形式のエスケープ・シーケンスは、スカラ値が yyyyyy の Unicode 文字に置き換えられます。エスケープ・シーケンス ¥¥ は、単一の ¥ で置き換えられます。たとえば、次の文は同じ意味を持ちます。

```
select * from mytable where unichar_column = '丑'
```

```
select * from mytable where unichar_column = U&'¥4e94'
```

プレフィクスの U& または u& は、単にエスケープの認識を有効にするものです。リテラルのデータ型は、文字セットで表現できるかどうかのみに基づいて選択されます。そのため、たとえば次の 2 つのクエリは同等です。

```
select * from mytable where char_column = 'A'
```

```
select * from mytable where char_column = U&'¥0041'
```

どちらの場合も、文字リテラルのデータ型は char です。これは、“A” は ASCII 文字で、ASCII は Sybase がサポートしているサーバの文字セットすべてのサブセットであるためです。

プレフィクスの U& および u& は、二重引用符で囲まれた文字リテラルでも、また引用符で囲まれた識別子に対しても有効です。ただし、引用符で囲まれた識別子は、すべてのデータベース・オブジェクトがシステム・テーブル内の名前で識別され、これらの名前が char データ型である場合、サーバの文字セットで表現可能である必要があります。

空文字列の処理

次の例では、固定長文字カラムと可変長文字カラムの両方がある spaces という名前のテーブルが作成されます。

```
create table spaces (cnot char(5) not null,
    cnull char(5) null,
    vnot varchar(5) not null,
        vnull varchar(5) null,
    explanation varchar(25) not null)

insert spaces values ("a", "b", "c", "d", "pads char-not-null only")
insert spaces values ("1    ", "2    ", "3    ", "4    ", "truncates trailing blanks")
insert spaces values ("    e", "    f", "    g", "    h", "leading blanks, no change")
insert spaces values ("   w ", "   x ", "   y ", "    z ", "truncates trailing blanks")
insert spaces values ("", "", "", "", "empty string equals space")
```

```
select "[" + cnot + "]",
       "[" + cnull + "]",
       "[" + vnot + "]",
       "[" + vnull + "]",
    explanation from spaces

                                    explanation
 -------  -------  ------- ------- --------------------
 [a    ]  [b]      [c]      [d]     pads char-not-null only
 [1    ]  [2]      [3]      [4]     truncates trailing blanks
 [    e]  [    f]  [    g]  [    h]  leading blanks, no change
 [   w ]  [   x ]  [   y ]  [   z ]  truncates trailing blanks
 [ ]      [ ]      [ ]      [ ]     empty string equals space

(5 rows affected)
```

この例は、ブランク・スペースの扱い方を決定するのにカラムのデータ型および null 型がどのように作用するかを示しています。

- char not null カラムと nchar not null カラムだけが、カラム幅いっぱいまでブランクを埋め込まれます。char null カラムは varchar と同様に扱われます。nchar null カラムは nvarchar と同様に扱われます。
- unichar not null カラムだけが、カラム幅いっぱいまでブランクを埋め込まれます。unichar null カラムは univarchar と同様に扱われます。
- 先行ブランクは影響を受けません。
- char、unichar、nchar not null のカラム以外では、後続ブランクがトランケートされます。
- 空の文字列 (“ ”) はシングル・スペースとして扱われます。char、nchar、unichar not null のカラムでは、結果はスペースだけからなるカラム長フィールドです。

文字データの操作

文字列内の特定の文字を検索するには like キーワードを使用し、文字列の内容を操作するには組み込み関数を使用します。数字からなる文字列は、convert 関数で真数値データ型および概数値データ型に変換してから、算術演算に使用できます。

標準規格と準拠レベル

ANSI SQL － 準拠レベル：Transact-SQL には、ANSI SQL データ型の char および varchar があります。nchar、nvarchar、unichar、univarchar データ型は Transact-SQL 拡張機能です。

バイナリ・データ型

ピクチャなどのロー・バイナリ・データを 16 進数に似た表記で、最大でサーバの論理ページ・サイズの最大カラム・サイズまで保管するには、バイナリ・データ型の binary(n) と varbinary(n) を使用します。

binary と *varbinary* の有効な入力値

バイナリ・データは文字 “0x” で始まり、数字と大文字および小文字の A ～ F との任意の組み合わせで構成されます。

n を使用してカラム長をバイト数単位で指定するか、デフォルト長の 1 バイトを使用してください。各バイトは、2 桁のバイナリ数を保管します。*n* より長い値を入力すると、Adaptive Server は、警告やエラーを発生せずにエントリを指定の長さにトランケートします。

すべての入力値の長さがほぼ等しいと予測されるデータには、固定長バイナリ型の binary(n) を使用します。

長さが大きく異なることが予測されるデータには、可変長バイナリ型の varbinary(n) を使用してください。

binary カラムの入力値には、カラム長 (*n*) に一致するようにゼロが埋め込まれるため、この入力値は varbinary カラムよりも多くの記憶領域を使用する場合がありますが、アクセスは多少速くなります。

長さの指定に *n* を使用しない場合は、次の点に注意してください。

- create table、alter table で作成されたカラム、および declare で作成された変数のデフォルトの長さは 1 バイトである。
- convert 関数で作成された値のデフォルトの長さは 30 バイトである。

最大カラム・サイズよりも大きいエントリ

さらに大きいブロックのバイナリ・データ (最大 2,147,483,647 バイトまで) を外部データ・ページに保管するには、image データ型を使用してください。image データ型は変数やストアド・プロシージャ内のパラメータには使用できません。詳細については、「text、image、unitext データ型」(33 ページ) を参照してください。

後続ゼロの扱い

binary not null カラムはすべて、カラム幅いっぱいまでゼロが埋め込まれます。varbinary データおよび binary null カラムでは、後続ゼロはトランケートされます。これは、null 値を受け入れるカラムは可変長として扱われるためです。

次の例では、NULL と NOT NULL、binary データ型と varbinary データ型の 4 つの組み合わせのカラムがすべて揃っているテーブルを作成します。4 つのカラムすべてに同じデータを挿入しますが、ゼロが埋め込まれるか、トランケートされるかどうかは、カラムのデータ型によって異なります。

```
create table zeros (bnot binary(5) not null,
            bnull binary(5) null,
            vnot varbinary(5) not null,
            vnull varbinary(5) null)

insert zeros values (0x12345000, 0x12345000, 0x12345000, 0x12345000)
insert zeros values (0x123, 0x123, 0x123, 0x123)

select * from zeros


bnot             bnull        vnot         vnull
------------     ---------    ----------   ---------
0x1234500000     0x123450     0x123450     0x123450
0x0123000000     0x0123       0x0123       0x0123
```

記憶領域の各バイトによってそれぞれ 2 桁のバイナリ数字が保持されるため、Adaptive Server は文字列 “0x” のあとに偶数個の数字が続くバイナリ・エントリを予期します。“0x” のあとに奇数個の数字が続く場合、Adaptive Server は先行ゼロが省略されているものとみなし、0 を追加します。

可変長バイナリ・カラム (binary null、image、varbinary カラム) では、入力値 “0x00” と “0x0” はどちらも “0x00” として保管されます。固定長バイナリ (binary not null) カラムでは、値はフィールド幅いっぱいまでゼロを埋め込まれます。

```
insert zeros values (0x0, 0x0,0x0, 0x0)
select * from zeros where bnot = 0x00

bnot           bnull       vnot        vnull
----------     ------      -----       ------------
0x0000000000   0x00        0x00        0x00
```

入力値に “0x” がない場合、Adaptive Server はその値が ASCII 値であると想定して、値を変換します。たとえば、次のようにします。

```
create table sample (col_a binary(8))

insert sample values (i002710000000ae1bí)

select * from sample
col_a
------------------
0x3030323731303030
```

プラットフォームの依存

特定の値の正確な入力形式は、使用しているプラットフォームに応じて異なります。このため、バイナリ・データを使う計算では、マシンによって異なった結果となることがあります。

集合関数 sum と avg では、バイナリ・データ型を処理できません。

16 進数の文字列と整数との間の変換で、どのプラットフォームでも同じ結果が得られるようにするには、プラットフォーム固有の変換関数ではなく、inttohex 関数と hextoint 関数を使います。詳細については、「データ型変換関数」(57 ページ) を参照してください。

標準規格と準拠レベル

ANSI SQL － 準拠レベル：binary データ型および varbinary データ型は Transact-SQL 拡張機能です。

bit データ型

true/false (真／偽) と yes/no タイプのデータを格納するカラムには、bit データ型を使用します。syscolumns システム・テーブルの status カラムは、bit データ型のカラムのユニークなオフセット位置を示します。

bit カラムは、0 または 1 を保持します。0 または 1 以外の整数値も受け入れますが、すべて 1 と解釈されます。

記憶サイズは 1 バイトです。テーブル内の複数の bit データ型がバイト単位にまとめられます。たとえば 7 つの bit カラムは 1 バイトになり、9 つの bit カラムは 2 バイトを使用します。

bit データ型のカラムには NULL を保管できません。また、インデックスも設定できません。

標準規格と準拠レベル

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

sysname データ型および *longsysname* データ型

sysname および longsysname は、Adaptive Server インストール・メディアによって配布されるユーザ定義データ型で、システム・テーブルで使用されます。定義は次のとおりです。

- sysname － varchar(30) "not null"
- longsysname － varchar(255) "not null"

カラム、パラメータ、または変数を sysname 型および longsysname 型として宣言できます。また、sysname および longsysname の基本となる型を持つユーザ定義データ型を作成し、このユーザ定義データ型を使用してカラム、パラメータ、変数を定義することもできます。

標準規格と準拠レベル

ANSI SQL － 準拠レベル：sysname 型および longsysname 型を含め、すべてのユーザ定義データ型は Transact-SQL 拡張機能です。

text、*image*、*unitext* データ型

text カラムは、最大 2,147,483,647 (2^{31} - 1) バイトまでの印刷可能な文字を保持できる可変長カラムです。

可変長の unitext データ型は、Unicode 文字で最大 1,073,741,823 文字 (2,147,483,646 バイト) まで保持できます。

image カラムは、最大 2,147,483,647 (2^{31} - 1) バイトまでのロー・バイナリ・データを保持できる可変長カラムです。

text と image の大きな違いは、Adaptive Server のデフォルト文字セットを使用していない場合に text が変換の対象になることです。image は、文字セット変換の対象になりません。

text、unitext、または image カラムは、他のカラムと同様に、create table 文または alter table 文を使用して定義します。text、unitext、または image データ型の定義には、長さは含まれません。text、unitext、image カラムでは、null 値が許可されます。これらのカラム定義の形式は次のとおりです。

column_name {text | image | unitext} [null]

たとえば、pubs2 データベースに作者の blurbs テーブルを作成し、null 値を格納できる text カラム blurb をこのテーブルに作成する create table 文は次のとおりです。

```
create table blurbs
(au_id id not null,
copy text null)
```

次の例では、null 値が使用できる unitext カラムが作成されます。

```
create table tb (ut unitext null)
```

image カラムのある au_pix テーブルを pubs2 データベースに作成するには、次の文を使用します。

```
create table au_pix
(au_id          char(11) not null,
pic             image null,
format_type     char(11) null,
bytesize        int null,
pixwidth_hor    char(14) null,
pixwidth_vert   char(14) null)
```

Adaptive Server では、text、unitext、および image データは、テーブルの他の部分とは別のデータ・ページのリンク・リストに保管されます。各 text、unitext、または image ページには、1 論理ページ・サイズ分のデータ (2、4、8、または 16K) が保管されます。テーブルに含まれている text、unitext、および image カラムの数には関係なく、1 つのテーブルの text、unitext、および image データは、1 つのページ・チェーンにすべて保管されます。

sp_placeobject を使用して、後続の text データ・ページ、unitext、および image データ・ページの割り当てを別の論理デバイスに置くことができます。

16 進数数値の桁数が奇数の image 値には、先行ゼロが埋め込まれます (“0xaaabb” を挿入すると、“0x0aaabb” になります)。

alter table コマンドの partition オプションを使用して、text、unitext、および image カラムが含まれているテーブルを分割できます。テーブルを分割することにより、テーブル内の他のカラムのための追加ページ・チェーンが作成されますが、これで text、unitext、および image カラムの保管方法が影響を受けることはありません。

unitext は、text データ型を使用する任意の場所で同じセマンティックで使用できます。unitext カラムは、Adaptive Server で使用されているデフォルトの文字セットにかかわらず、UTF-16 エンコーディングで保管されます。

text、unitext、および image データの保管に使用されるデータ構造

text、unitext、または image データを割り付ける場合、割り付けるローに 16 バイトのテキスト・ポインタが挿入されます。このテキスト・ポインタの一部は、text、unitext、または image データの先頭にある text ページのページ番号を示します。テキスト・ポインタは先頭 text ページ (FTP) とも呼ばれます。

先頭 text ページには、次の 2 つの部分があります。

- text データと image データを含む、text ページの双方向にリンクされたリストである text データ・ページ・チェーン
- ユーザの text データへのアクセスに使用されるオプションのテキスト・ノード構造

いったん先頭 text ページが text、unitext、または image データに割り付けられると、その割り付けは解除されることがありません。既存の text、unitext、または image データ・ローが更新されたことにより、現在割り付けられているよりも text ページが少なくなった場合は、余分な text ページの割り付けが解除されます。text、unitext、または image データを更新して値が NULL に設定されると、先頭 text ページ以外のすべてのページの割り付けは解除されます。

図 1-1 は、データ・ローと text ページの関係を示します。

図 *1-1:* テキスト・ポインタとデータ・ローの関係

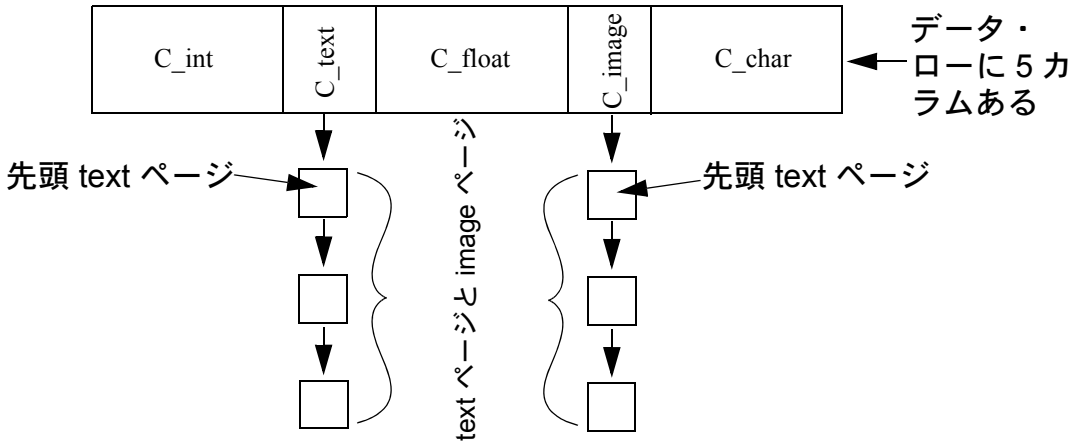

図 1-1 では、カラム c_text と c_image は、図の下部にあるページを含む text カラムおよび image カラムです。

text、*unitext*、および *image* カラムの初期化

text、unitext、image カラムは、更新するか、非 null 値を挿入するまで、初期化されません。初期化によって、text、unitext、または image の非 null データ値ごとに、少なくとも 1 つのデータ・ページが割り付けられます。text、unitext、または image データのロケーションを指すポインタもテーブルに作成されます。

たとえば次の文では、テーブル testtext を作成し、非 null 値を挿入して blurb カラムを初期化します。これで、このカラムは有効なテキスト・ポインタを持ち、最初のテキスト・ページが割り付けられます。

```
create table texttest
(title_id varchar(6), blurb text null, pub_id char(4))

insert texttest values
("BU7832", "Straight Talk About Computers is an annotated
analysis of what computers can do for you: a no-hype guide for
the critical user.", "1389")
```

次の文では、image 値のためのテーブルを作成し、image カラムを初期化します。

```
create table imagetest
(image_id varchar(6), imagecol image null, graphic_id char(4))
insert imagetest values
("94732", 0x0000008300000000000100000000013c, "1389")
```

注意 text 値は必ず引用符で囲み、image 値の前には文字 "0x" を付けてください。

Client-Library プログラムでの text、unitext、および image データの挿入と更新については、『Client-Library/C リファレンス・マニュアル 1』を参照してください。

unitext カラムの定義

unitext カラムは、他のデータ型を定義する場合と同じ方法で、create table 文または alter table 文を使用して定義できます。unitext カラムの長さは定義しません。カラムは null でもかまいません。

次の例では、null 値が使用できる unitext カラムが作成されます。

```
create table tb (ut unitext null)
```

default unicode sort order は、like 句および patindex 関数内でパターン一致に使用する unitext カラムのソート順を定義します。このソート順は、Adaptive Server のデフォルト・ソート順とは無関係に適用されます。

NULL によるスペースの節約

空の text、unitext、または image カラムの記憶領域を節約するには、カラムへの null 値入力を許可し、カラムを使用するまで null を挿入 (insert) します。null 値を挿入しても、text、unitext、または image カラムは初期化されません。したがって、テキスト・ポインタは作成されず、記憶域も割り付けられません。たとえば次の文では、上で作成した testtext テーブルの title_id と pub_id の両カラムに値が挿入されますが、blurb テキスト・カラムは初期化されません。

```
insert texttest
(title_id, pub_id) values ("BU7832", "1389")
```

sysindexes からの情報取得

text、unitext、または image カラムを持つ各テーブルに対して、これらのカラムに関する情報を提供する特別なローが sysindexes 内にあります。sysindexes の name カラムは、"tablename" という形式を使用しています。indid は常に 255 です。これらのカラムには、テキストの記憶領域に関する情報が格納されています。

表 *1-18: text および image データの記憶領域*

カラム	説明
ioampg	テキスト・ページ・チェーンのアロケーション・ページを指すポインタ
first	テキスト・データの最初のページを指すポインタ
root	最後のページを指すポインタ
segment	オブジェクトが置かれているセグメントの数

これらのカラムに関する情報について、sysindexes テーブルを問い合わせることができます。たとえば、次のクエリでは、pubs2 データベースで blurbs テーブルが使用しているデータ・ページ数がレポートされます。

```
select name, data_pages(db_id(), object_id("blurbs"), indid)
    from sysindexes
    where name = "tblurbs"
```

注意　システム・テーブルのポスターでは、sysindexes と systabstats が 1 対 1 (1-1) の関係にあることが示されています。systabstats で情報が保持されない text カラムと image カラムを除く、すべてのカラムにこの関係が該当します。

readtext と writetext の使用

text データを、writetext を使用して入力したり、readtext を使用して読み取ったりする前に、text カラムを初期化してください。詳細については、『ASE リファレンス・マニュアル：コマンド』の「readtext」および「writetext」を参照してください。

update を使用して、既存の text、unitext、および image データを NULL で置き換えると、最初のページを除いて、割り付けられていたすべてのデータ・ページが再び使用できる状態になるため、以降の writetext で使用可能になります。ローに割り付けられていたすべての記憶領域を解除するには、delete を使用して、ロー全体を削除してください。

unitext で定義されたカラムに対する readtext および writetext の使用には制限があります。詳細については、『ASE リファレンス・マニュアル：コマンド』の「readtext」および「writetext」を参照してください。

カラムが使用する領域の決定

sp_spaceused は、テキスト・データが使用する領域に関する情報を、index_size として提供します。

```
sp_spaceused blurbs
name            rowtotal  reserved  data    index_size unused
--------------- --------  --------- ------- ---------- ------
blurbs          6         32 KB     2 KB    14 KB      16 KB
```

text、image、および unitext カラムの制限

text、image、または unitext カラムには、次の制限があります。

- ストアド・プロシージャのパラメータまたはこれらのパラメータに渡す値としては使用できません。
- ローカル変数として使用できません。
- order by 句、compute 句、group by 句、および union 句内では使用できません。
- インデックス内では使用できません。
- サブクエリまたはジョイン内では使用できません。
- where 内では使用できません。ただし、キーワード like で使用する場合を除きます。
- + 連結演算子とともには使用できません。

text、unitext、image データの選択

次のグローバル変数は text、unitext、および image データに関する情報を返します。

表 *1-19: text 、unitext、image グローバル変数*

変数	説明
@@textptr	プロセスが最後に挿入または更新した、text、unitext、または image カラムのテキスト・ポインタ。このグローバル変数と textptr 関数を混同しないように注意。
@@textcolid	*@@textptr* が参照するカラムの ID。
@@textdbid	*@@textptr* が参照するカラムのオブジェクトが入っているデータベースの ID。
@@textobjid	*@@textptr* が参照するカラムが入っているオブジェクトの ID。
@@textsize	set textsize オプションの現在の値。この値は、select 文が返す text、unitext、または image データの最大長をバイト数単位で指定する。デフォルト値は 32K。*@@textsize* の最大サイズは 2^{31} - 1 (2,147,483,647)。
@@textts	*@@textptr* によって参照されるカラムのタイムスタンプ。

text および *image* データ型の変換

convert 関数を使用すれば、text 値から char、unichar、varchar、または univarchar への変換、および image 値から binary または varbinary への変換を明示的に実行できます。ただし、文字データ型またはバイナリ・データ型の最大長は制限されます。この制限は、ご使用のサーバの論理ページ・サイズの最大カラム・サイズによって決まります。長さを指定しない場合、変換された値はデフォルトの長さである 30 バイトになります。暗黙的な変換はサポートされていません。

unitext へ、または unitext からの変換

unitext を明示的に別のデータ型に変換したり、別のデータ型を unitext に変換したりするのと同様に、文字またはバイナリのデータ型を暗黙的に unitext に変換することもできます。ただし、変換結果は変換先のデータ型の最大長に制約されます。unitext 値が Unicode 文字境界上の変換先バッファに収まらない場合、データはトランケートされます。enable surrogate processing を有効にしている場合は、unitext 値はサロゲート・ペア値の途中ではトランケートされません。つまり、データ変換後に返されるバイト数が少なくなることがあります。たとえば、テーブル tb の unitext カラム ut に文字列 “U+0041U+0042U+00c2” (U+0041 は Unicode 文字 “A”) が保管されている場合、次のクエリはサーバの文字セットが UTF-8 のときに値 “AB” を返します。これは、U+00C2 が 2 バイトの UTF-8 0xc382 に変換されるからです。

```
select convert(char(3), ut) from tb
```

表 *1-20: unitext* へ、および *unitext* からの変換

暗黙的に unitext へ変換されるデータ型	暗黙的に unitext から変換されるデータ型	明示的に unitext から変換されるデータ型
char、varchar、unichar、univarchar、binary、varbinary、text、image	text、image	char、varchar、unichar、univarchar、binary、varbinary

alter table modify コマンドには、変更対象カラムとして text、image、または unitext カラムを指定できません。text カラムを unitext カラムにマイグレートするには、次の手順を実行します。

- bcp out -Jutf8 を使用して text カラム・データをコピー・アウトします。
- unitext カラムを含むテーブルを作成します。
- bcp in -Jutf8 を使用して、データを新しいテーブルに挿入します。

text データのパターン一致

text、unitext、varchar、univarchar、unichar、または char カラム内で、指定パターンが最初に出現する位置を検索するには、patindex 関数を使用します。% ワイルドカード文字は、最初の文字または最後の文字を検索する場合を除いて、パターンの前後に指定する必要があります。

特定のパターンを検索するには、like キーワードも使用できます。次の例では、blurbs テーブルの copy カラムから、パターン “Net Etiquette” が含まれている各 text データ値を選択します。

```
select copy from blurbs
where copy like "%Net Etiquette%"
```

重複ロー

text、image、および unitext データを指すポインタによって、各ローが一意に識別されます。したがって、text、image、および unitext データが含まれているテーブルには、すべての text、image、および unitext データが NULL の場合を除き、重複するローを格納できません。このような場合、ポインタが初期化されていません。

標準規格と準拠レベル

ANSI SQL － 準拠レベル：text、image、および unitext データ型は Transact-SQL 拡張機能です。

データ型と暗号化カラム

表 1-21 では、暗号化カラムでサポートされるデータ型、および Adaptive Server バージョン 15.0.2 でサポートされるデータ型の暗号化カラムのディスク上の最大長を示します。

表 *1-21:* 暗号化カラムのデータ型長

データ型	入力データ長	暗号化カラムのタイプ	暗号化データの最大長 (init_vector なし)	暗号化データの実際の長さ (init_vector なし)	暗号化データの最大長 (init_vector 使用)	暗号化データの実際の長さ (init_vector 使用)
date	4	varbinary	17	17	33	33
time	4	varbinary	17	17	33	33
smalldatetime	4	varbinary	17	17	33	33
bigdatetime	8	varbinary	17	17	33	33
bigtime	8	varbinary	17	17	33	33

データ型	入力データ長	暗号化カラムのタイプ	暗号化データの最大長 (init_vector なし)	暗号化データの実際の長さ (init_vector なし)	暗号化データの最大長 (init_vector 使用)	暗号化データの実際の長さ (init_vector 使用)
datetime	8	varbinary	17	17	33	33
smallmoney	4	varbinary	17	17	33	33
money	8	varbinary	17	17	33	33
bit	8	varbinary	17	17	33	33
bigint	8	varbinary	17	1 7	33	33
unsigned bigint	8	varbinary	17	17	33	33
unichar(10)	2 (1 unichar character)	varbinary	33	17	49	33
unichar(10)	20 (10 unichar character)	varbinary	33	33	49	49
univarchar(20)	20 (10 unichar character)	varbinary	49	33	65	49

text、image、unitext データ型は、このリリースの Adaptive Server ではサポートされていません。

ユーザ定義データ型

ユーザ定義データ型は、システム・データ型および sysname または longsysname ユーザ定義データ型から構築されます。ユーザ定義データ型を作成したあと、これを使用して、カラム、パラメータ、変数を定義できます。ユーザ定義データ型から作成されたオブジェクトは、ユーザ定義データ型の基になっているシステム・データ型のデフォルトや null 型を継承するのと同様に、ユーザ定義データ型のルール、デフォルト、null 型、IDENTITY プロパティを継承します。

ユーザ定義データ型は、そのデータ型を使用するデータベースに作成する必要があります。使用頻度の高い型は、model データベースに作成します。これらのデータ型は、新しいデータベース (テンポラリ・テーブルに使用する tempdb を含む) が作成されると、自動的に追加されます。

Adaptive Server では、sp_addtype を使用して、任意のシステム・データ型をもとにユーザ定義データ型を作成できます。timestamp や pubs2 データベースで定義されている tid データ型など、別のユーザ定義データ型をもとにユーザ定義データ型を作成することはできません。

sysname データ型および longsysname データ型には、この規則は適用されません。sysname および longsysname はユーザ定義データ型ですが、このデータ型を使用して、ユーザ定義データ型を作成できます。

ユーザ定義データ型はデータベース・オブジェクトです。名前では大文字、小文字が区別され、識別子の規則に従っている必要があります。

sp_bindrule を使用して、ユーザ定義データ型にルールをバインドできます。また、sp_bindefault を使用して、ユーザ定義データ型にデフォルトをバインドできます。

ユーザ定義データ型をもとに構築されたオブジェクトは、デフォルトでユーザ定義データ型の null 型または IDENTITY プロパティを継承します。カラムの定義時に null タイプまたは IDENTITY プロパティを上書きできます。

ユーザ定義データ型の名前を変更するには、sp_rename を使用します。

データベースからユーザ定義データ型を削除するには、sp_droptype を使用します。

注意 テーブルで既に使用されているデータ型は、削除できません。

システム・データ型またはユーザ定義データ型のプロパティの情報を表示するには、sp_help を使用します。sp_help は、テーブル内の各カラムのデータ型、長さ、精度、位取りを表示するためにも使用できます。

標準規格と準拠レベル

ANSI SQL － 準拠レベル：ユーザ定義データ型は Transact-SQL 拡張機能です。

第 2 章

Transact-SQL 関数

この章では、Transact-SQL 関数について説明します。関数は、データベースから情報を返すために使用されます。関数は、where 句、select リスト、および式を使用できる位置で使用できます。関数は、ストアド・プロシージャまたはプログラムの一部としてよく使用されます。

関数の種類

表 2-1 では、さまざまな種類の Transact-SQL 関数を示し、それぞれの関数が返す情報の種類について説明しています。

表 *2-1: Transact-SQL* 関数の種類

関数の種類	説明
集合関数	新しいカラムまたはクエリ結果の追加ローとして表示される合計値を生成する。
データ型変換関数	式のデータ型を別のデータ型に変更して、date および time の情報を示す新しい表示フォーマットを指定する。
日付関数	datetime、smalldatetime、date、time の値とそのコンポーネント、日付要素に対する計算を実行する。
算術関数	算術データの演算に共通して必要な関数。
セキュリティ関数	セキュリティ関連情報。
文字列関数	バイナリ・データ、文字列、式を処理する。
システム関数	データベースやデータベース・オブジェクトから特別な情報を取得する。
text 関数および image 関数	text、unitext、image データのオペレーションに共通して必要な値を返す。

表 2-2 に、アルファベット順の関数のリストを示します。

表 2-2: *Transact-SQL* 関数のリスト

関数	種類	戻り値
abs (72 ページ)	算術	式の絶対値。
acos (73 ページ)	算術	余弦が指定されている角度 (ラジアン)。
ascii (74 ページ)	文字列	式の先頭文字の ASCII コード。
asin (76 ページ)	算術	正弦が指定されている角度 (ラジアン)。
atan (77 ページ)	算術	正接が指定されている角度 (ラジアン)。
atn2 (78 ページ)	算術	正弦と余弦が指定されている角度 (ラジアン)。
audit_event_name (81 ページ)	セキュリティ	監査イベントの説明
avg (79 ページ)	集合	すべて (または重複を除くすべて) の値の平均値。
biginttohex (84 ページ)	データ型変換	指定した整数値に相当する、プラットフォームの影響を受けない 16 進文字列を返します。
case (87 ページ)		条件値を使用するための SQL 式を記述できます。case 式は、値の式を使用できる場所であればどこででも使用できます。
cast (90 ページ)	データ型変換	指定された値を別のデータ型に変換します。
ceiling (93 ページ)	算術	指定した値以上の最小整数値。
char (95 ページ)	文字列	整数値に相当する文字。
charindex (99 ページ)	文字列	式の開始位置を表す整数を返します。
char_length (97 ページ)	文字列	式の文字数。
col_length (102 ページ)	システム	指定したカラムの長さ。
col_name (103 ページ)	システム	テーブル ID とカラム ID が指定されているカラムの名前。
compare (104 ページ)	システム	選択した照合規則をもとに次の値を返します。 • 1 − *char_expression1* が *char_expression2* より大きいことを示します。 • 0 − *char_expression1* が *char_expression2* と等しいことを示します。 • -1 − *char_expression1* が *char_expression2* より小さいことを示します。
convert (109 ページ)	データ型変換	指定された値が、別のデータ型または異なる datetime 表示フォーマットに変換された値。
cos (114 ページ)	算術	指定した角度 (ラジアン) の余弦。
cot (115 ページ)	算術	指定した角度 (ラジアン) の余接。
count (116 ページ)	集合	null 以外の値の数 (重複を除くことも可能) を integer 型で返します。
count_big (118 ページ)	集合	null 以外の値の数 (重複を除くことも可能) を bigint 型で返します。
current_date (122 ページ)	日付	現在の日付を返します。
current_time (123 ページ)	日付	現在の時刻を返します。
curunreservedpgs (124 ページ)	システム	指定したディスク区分の空きページ数。
data_pages (126 ページ)	システム	指定したテーブルやインデックスによって使用されているページ数。
datalength (130 ページ)	システム	指定したカラムや文字列の実際の長さ (バイト数)。
dateadd (131 ページ)	日付	ある年数、四半期数、時間数、その他の日付要素を、指定した日付に加えて計算した日付。
datediff (134 ページ)	日付	2 つの日付式の差。

関数	種類	戻り値
datename (138 ページ)	日付	日付式の指定した要素の名前。
datepart (140 ページ)	日付	日付式の指定した要素の整数値。
day (144 ページ)	日付	指定した日付の datepart の日を表す整数を返します。
db_id (147 ページ)	システム	指定したデータベースの ID 番号。
db_name (149 ページ)	システム	ID 番号で指定されたデータベースの名前。
degrees (151 ページ)	算術	ラジアン数が指定されている角度の大きさ (度)。
derived_stat (152 ページ)	システム	指定したオブジェクトとインデックスに対して得られた統計を返します。
difference (157 ページ)	文字列	2 つの soundex 値の差。
exp (158 ページ)	算術	定数 e を指定した累乗にした値。
floor (159 ページ)	算術	指定した値以下の最大整数値。
get_appcontext (161 ページ)	セキュリティ	指定したコンテキストの属性値を返します。
getdate (162 ページ)	日付	システムの現在の日付と時刻。
hextobigint (170 ページ)	データ型変換	16 進文字列に相当する bigint 値。
hextoint (171 ページ)	データ型変換	指定した 16 進文字列に相当するプラットフォームに依存しない整数。
host_id (172 ページ)	システム	クライアント・コンピュータのオペレーティング・システムで使用されている、現在の Adaptive Server クライアントのプロセス ID を返します。
host_name (173 ページ)	システム	クライアント・プロセスの現在のホスト・コンピュータ名。
identity_burn_max (175 ページ)		identity_burn_max 値。
index_col (176 ページ)	システム	指定したテーブルやビューのインデックス・カラムの名前。
index_colorder (177 ページ)	システム	カラム順を返します。
inttohex (179 ページ)	データ型変換	指定した整数に相当する、プラットフォームの影響を受けない 16 進文字列。
isdate (180 ページ)	データ型変換	入力式が有効な datetime 値であるかどうかを確認します。
isnumeric (181 ページ)	データ型変換	式が有効な numeric データ型かどうかを確認します。
is_quiesced (182 ページ)	システム	データベースが quiesce database モードにあるかどうかを示します。is_quiesced は、データベースがクワイス (静止) 状態の場合に 1、クワイス状態でない場合に 0 を返します。
is_sec_service_on (184 ページ)	セキュリティ	セキュリティ・サービスが有効な場合は 1 を返し、無効な場合は 0 を返します。
isnull (185 ページ)	システム	*expression1* の値が NULL のとき、*expression2* に指定した値と置き換えます。
lct_admin (190 ページ)	システム	ラストチャンス・スレッショルドを管理します。
left (193 ページ)	文字列	文字列の左端から指定された数の文字を返します。
len (194 ページ)	文字列	後続ブランクを除き、指定した文字列式の文字数 (バイト数ではない) を返します。
license_enabled (195 ページ)	システム	機能のライセンスが有効な場合は 1 を返し、無効な場合は 0 を返します。
list_appcontext (196 ページ)	セキュリティ	現在のセッション内にある全コンテキストの属性をすべてリストします。

関数	種類	戻り値
lockscheme (197 ページ)	算術	指定のオブジェクトのロック・スキームを文字列で返します。
log (198 ページ)	算術	指定した数の自然対数。
log10 (199 ページ)	算術	指定した数の常用対数 (底が 10 の対数)。
lower (200 ページ)	文字列	指定した式に相当する小文字の文字値。
ltrim (201 ページ)	文字列	先行ブランクを削除した指定の式。
max (202 ページ)	集合	カラム内の最大値。
min (203 ページ)	集合	カラム内の最小値。
month (204 ページ)	日付	指定した日付の datepart の月を表す整数。
mut_excl_roles (205 ページ)	セキュリティ	2 つの役割の間の相互排他性。
newid (206 ページ)	システム	指定した引数に基づいて、人間が判読できるグローバルにユニークな識別子 (GUID) を 2 とおりの異なるフォーマットで生成します。
next_identity (208 ページ)	システム	次の insert に使用可能な次の identity 値を取得します。
nullif (209 ページ)		条件値を使用するための SQL 式を記述できます。nullif 式は、値式が使用できる場所であればどこででも case 式の代わりに使用できます。
object_id (215 ページ)	システム	指定したオブジェクトのオブジェクト ID。
object_name (216 ページ)	システム	指定したオブジェクト ID を持つオブジェクトの名前。
pagesize (218 ページ)	算術	指定したオブジェクトのページ・サイズをバイト数で返します。
partition_id (220 ページ)	システム	指定したデータまたはインデックス・パーティション名のパーティション ID を返します。
partition_name (221 ページ)	システム	新しいパーティションの明示的な名前 partition_name は、指定されたデータのパーティション名またはインデックス・パーティション ID を返します。
partition_object_id (222 ページ)	システム	指定されたパーティション ID とデータベース ID のオブジェクト ID を表示します。
patindex (223 ページ)	文字列、text、unitext、image	指定したパターンが最初に検出された先頭位置。
pi (226 ページ)	算術	定数値 (円周率) 3.1415926535897936。
power (227 ページ)	算術	指定した数字をある累乗した結果の値。
proc_role (228 ページ)	セキュリティ	ユーザがプロシージャを実行する正しい役割を持つ場合は 1、ユーザがこの役割を持たない場合は 0。
pssinfo (230 ページ)	システム	Adaptive Server プロセス・ステータス構造 (pss) から情報を返します。
radians (231 ページ)	算術	度数が指定されている角度の大きさ (ラジアン)。
rand (232 ページ)	算術	指定した基数値を使って作られた、0 と 1 の間の乱数値。
replicate (234 ページ)	文字列	指定した式を特定の回数だけ繰り返す文字列。
reserved_pages (237 ページ)	システム	指定したテーブルやインデックスに割り付けられたページ数。
reverse (241 ページ)	文字列	文字順を逆にした指定した文字列。
right (242 ページ)	文字列	指定した文字数を右側から取り出した結果を返す。
rm_appcontext (244 ページ)	セキュリティ	特定のアプリケーション・コンテキストまたはすべてのアプリケーション・コンテキストを削除します。
role_contain (245 ページ)	セキュリティ	*role2* に *role1* が含まれている場合は 1。

関数	種類	戻り値
role_id (246 ページ)	セキュリティ	指定した名前を持つ役割のシステム役割 ID。
role_name (247 ページ)	セキュリティ	指定したシステム役割 ID を持つ役割の名前。
round (248 ページ)	算術	指定した数を、特定の小数桁数に丸めた値。
row_count (250 ページ)	システム	指定したテーブル内のローの予測数。
rtrim (251 ページ)	文字列	後続ブランクを削除した指定の式。
set_appcontext (253 ページ)	セキュリティ	ユーザ・セッション用のアプリケーション・コンテキスト名、属性名、属性値を設定します。それらは、指定したアプリケーションの属性によって定義されます。
show_role (255 ページ)	セキュリティ	ログインの現在有効な役割。
show_sec_services (256 ページ)	セキュリティ	ユーザの現在有効なセキュリティ・サービスのリスト。
sign (257 ページ)	算術	指定した値の符号 (正の +1、0、負の -1)。
sin (258 ページ)	算術	指定した角度 (ラジアン) の正弦。
sortkey (259 ページ)	システム	照合動作をもとに、結果を並べ替えるために使用する値。これにより、デフォルトのラテン文字セット以外の文字セットについても、辞書順、大文字と小文字の区別、アクセントの区別をもとにした文字列照合動作が可能になります。
soundex (264 ページ)	文字列	式の発音を表す 4 文字コード。
space (265 ページ)	文字列	指定した数のシングルバイト・スペースからなる文字列。
square (267 ページ)	算術	指定した値の平方を、float で表したものを返します。
sqrt (268 ページ)	算術	指定した数の平方根。
str (274 ページ)	文字列	指定した数に相当する文字。
str_replace (276 ページ)	文字列	第一引数文字列に対して、第二引数で指定されている文字列と比較して一致するすべてのインスタンスを第三引数文字列と置き換えて返します。
stuff (279 ページ)	文字列	1 つの文字列から指定した数の文字を削除し、別の文字列を代入して作成した文字列。
substring (281 ページ)	文字列	指定した数の文字を別の文字列から取り出して作った文字列。
sum (282 ページ)	集合	値の合計。
suser_id	システム	syslogins システム・テーブルからのサーバ・ユーザの ID 番号。
suser_name (285 ページ)	システム	現在のサーバ・ユーザの名前、または指定したサーバ ID を持つユーザの名前を返します。
syb_quit (286 ページ)	システム	接続を終了します。
syb_sendmsg (287 ページ)	システム	UDP (ユーザ・データグラム・プロトコル) ポートにメッセージを送信します。
tan (289 ページ)	算術	指定した角度 (ラジアン) の正接。
tempdb_id (290 ページ)	システム	指定した spid に割り当てられたテンポラリ・データベースのデータベース ID。
textptr (291 ページ)	text、unitext、image	指定した text カラムの先頭ページを指すポインタ。
textvalid (292 ページ)	text および image	指定した text カラムを指すポインタが有効な場合は 1、無効な場合は 0。

関数	種類	戻り値
to_unichar (293 ページ)	文字列	整数式の値を持つ Unichar 式。
tran_dumpable_status (294 ページ)	システム	dump transaction を使用できるかどうかを示す true ／ false の値を返します。
tsequal (296 ページ)	システム	ローがブラウズ用に選択されたあとで変更された場合に、ローを更新できないようにするため timestamp 値を比較します。
uhighsurr (298 ページ)	文字列	start の位置の Unicode 値が、サロゲート・ペアの上位サロゲート (ペアの中で最初に指定されている) の場合は 1、それ以外の場合は 0。
ulowsurr (299 ページ)	文字列	start の位置の Unicode 値が、サロゲート・ペアの下位サロゲート (ペアの中で 2 番目に指定されている) の場合は 1、それ以外の場合は 0。
upper (300 ページ)	文字列	指定した文字列に相当する大文字の文字値。
uscalar (301 ページ)	文字列	式の 1 番目の Unicode 文字の Unicode スカラ値。
used_pages (302 ページ)	システム	指定したテーブルとそのクラスタード・インデックスが使うページ数。
user (304 ページ)	システム	現在のユーザの名前。
user_id (305 ページ)	システム	指定したユーザまたは現在のユーザの ID。
user_name (306 ページ)	システム	指定したユーザ、または現在のユーザのデータベース内の名前。
valid_name (307 ページ)	システム	指定した文字列が有効な識別子でない場合は 0、有効な識別子の場合は 0 以外の数。
valid_user (308 ページ)	システム	指定した ID が、Adaptive Server 上の少なくとも 1 つのデータベースで有効なユーザかエイリアスの場合は 1。
year (331 ページ)	日付	指定した日時の日付要素に含まれる年を表す整数。

以下の項では、関数の種類について詳しく説明します。この章の後半では、各関数の説明をアルファベット順で示します。

集合関数

集合関数は、クエリ結果に新しいカラムとして表示される合計値を生成します。集合関数には、次のものがあります。

- avg
- count
- count_big
- max
- min
- sum

集合関数は select リスト、または select 文の having 句か、サブクエリで使用できます。where 句では使用できません。

クエリ内の各集合には、それぞれのワーク・テーブルが必要です。したがって、集合関数を使用するクエリでは、クエリに許可されたワーク・テーブルの最大数 (46) を超える数の集合を使用できません。

集合関数は、char データ型値に適用される場合、値を暗黙的に varchar に変換して、後続ブランクをすべて削除します。同様に、unichar データ型の値は暗黙的に univarchar データ型へ変換されます。

集合関数 max、min、count は、unichar データ型を含むセマンティックを提供しています。

group by で使用される集合関数

集合関数は、多くの場合、group by 句とともに使用されます。group by によって、テーブルは複数のグループに分けられます。集合関数は、グループごとに 1 つの値を作成します。group by を使用しないと、select リスト内の集合関数は、テーブル内のすべてのローを処理するか、または where 句によって定義されたローのサブセットを処理するかに関係なく、結果的に 1 つの値を作成します。

集合関数と NULL 値

集合関数は、特定カラムにある null 以外の値の合計値を計算します。ansinull オプションを off (デフォルト値) に設定すると、集合関数で null が検出されても警告は出されません。ansinull オプションを on に設定すると、集合関数で null が検出されたときに、クエリから次の SQLSTATE 警告が返されます。

```
Warning- null value eliminated in set function
```

ベクトルおよびスカラ集合

集合関数は、テーブル内のすべてのローに適用できます。この場合、単一の値であるスカラ集合が作成されます。集合関数は、指定したカラムまたは式で同じ値を持つすべてのローにも (group by と、オプションで having 句を使用して) 適用できます。この場合は、グループごとに 1 つの値、ベクトル集合が作成されます。集合関数の結果は新しいカラムとして表示されます。

スカラ集合関数内にベクトル集合関数をネストできます。たとえば、次のようにします。

```
select type, avg(price), avg(avg(price))
  from titles
group by type
type
------------  ------------  ------------
UNDECIDED             NULL         15.23
business             13.73         15.23
mod_cook             11.49         15.23
popular_comp         21.48         15.23
psychology           13.50         15.23
trad_cook            15.96         15.23

(6 rows affected)
```

group by 句は、ベクトル集合に適用されます。この例では、avg(price) です。スカラ集合 avg(avg(price)) は、titles テーブルの種類別平均価格の平均値です。

標準的な SQL では、*select_list* に集合関数が含まれる場合、すべての *select_list* カラムに集合関数が適用されるか、またはこれらのカラムが group by リストに含まれている必要があります。Transact-SQL には、このような制限はありません。

例 1 は、標準の制限が適用される select 文を示します。例 2 では、同じ文の select リストに別の項目 (title_id) が追加されています。表示内容の違いを明確に示すため、order by も追加されています。これらの「余分な」カラムは、having 句でも参照できます。

例 1

```
select type, avg(price), avg(advance)
from titles
group by type

type
------------  ------------  ------------
UNDECIDED             NULL          NULL
business             13.73      6,281.25
mod_cook             11.49      7,500.00
popular_comp         21.48      7,500.00
psychology           13.50      4,255.00
trad_cook            15.96      6,333.33

(6 rows affected)
```

例 2

group by の後に、カラム名または他の式 (カラム見出しまたはエイリアスを除く) のどちらかを使用できます。

group by カラムの null 値は、1 つのグループにまとめられます。

```
select type, title_id, avg(price), avg(advance)
  from titles
group by type
order by type
```

```
type         title_id
-----------  --------    ---------- ---------
UNDECIDED       MC3026      NULL       NULL
business        BU1032      13.73   6,281.25
business        BU1111      13.73   6,281.25
business        BU2075      13.73   6,281.25
business        BU7832      13.73   6,281.25
mod_cook        MC2222      11.49   7,500.00
mod_cook        MC3021      11.49   7,500.00
popular_comp    PC1035      21.48   7,500.00
popular_comp    PC8888      21.48   7,500.00
popular_comp    PC9999      21.48   7,500.00
psychology      PS1372      13.50   4,255.00
psychology      PS2091      13.50   4,255.00
psychology      PS2106      13.50   4,255.00
psychology      PS3333      13.50   4,255.00
psychology      PS7777      13.50   4,255.00
trad_cook       TC3218      15.96   6,333.33
trad_cook       TC4203      15.96   6,333.33
trad_cook       TC7777      15.96   6,333.33
```

例 3

select 文の compute 句は、ロー集合を使用して合計値を作成します。ロー集合を使用すると、1 つのコマンドでディテール・ローと合計ローを検索できます。例 3 に、この機能を示します。

```
select type, title_id, price, advance
  from titles
where type = "psychology"
order by type
compute sum(price), sum(advance) by type
```

```
type        title_id    price       advance
----------- -------     ----------  ---------
psychology  PS1372          21.59   7,000.00
psychology  PS2091          10.95   2,275.00
psychology  PS2106          7.00    6,000.00
psychology  PS3333          19.99   2,000.00
psychology  PS7777          7.99    4,000.00
                            sum     sum
                            ------- ----------
                            67.52   21,275.00
```

例 3 の表示内容と、compute を使用しない例 (例 1 と 例 2) の表示内容の違いに注意してください。

集合関数は sysprocesses や syslocks などの仮想テーブルには使用できません。

カーソルの select 句に集合関数を含めた場合、このカーソルは更新できません。

ロー集合としての集合関数

ロー集合関数は、クエリ結果に追加ローとして表示される合計値を生成します。

集合関数をロー集合として使用するには、次の構文を使用します。

```
Start of select statement
compute row_aggregate(column_name)
        [, row_aggregate(column_name)]...
    [by column_name [, column_name]...]
```

ここで、

- *column_name* は、カラム名です。必ずカッコで囲んでください。sum と avg で使用できるのは、真数値、概数値、通貨カラムだけです。

  1 つの compute 句で、同じ関数をいくつかのカラムに適用できます。複数の関数を使用するには、複数の compute 句を使用してください。

- by は、ローの集約値がサブグループについて計算されることを示します。by 項目の値が変更されると、常にローの集約値が生成されます。by を使用する場合は、order by を使用してください。

  by の後に複数の項目をリストすると、グループがサブグループに分割され、グループ化される各レベルに関数が適用されます。

ロー集合を使用すると、1 つのコマンドでディテール・ローと合計ローを検索できます。これに対して、集合関数は通常、テーブル内の指定されたすべてのローに対して、または各グループに対して 1 つの値を生成し、この合計値は新しいカラムとして表示されます。

次の例は、この違いを示しています。

```
select type, sum(price), sum(advance)
from titles
where type like "%cook"
group by type

type
----------  ----------  ----------------
mod_cook         22.98         15,000.00
trad_cook        47.89         19,000.00

(2 rows affected)
```

```
select type, price, advance
from titles
where type like "%cook"
order by type
compute sum(price), sum(advance) by type

type         price          advance
----------   ----------   ----------------
mod_cook          2.99         15,000.00
mod_cook         19.99              0.00
             sum          sum
             ----------   ----------------
                  22.98         15,000.00
type           price       advance
----------   ----------   ----------------
trad_cook        11.95          4,000.00
trad_cook        14.99          8,000.00
trad_cook        20.95          7,000.00
             sum          sum
             ----------   ----------------
                  47.89         19,000.00
(7 rows affected)
type         price        advance
----------   ----------   ----------------
mod_cook          2.99         15,000.00
mod_cook         19.99              0.00

 Compute Result:
---------------------- -----------------
                  22.98         15,000.00
type           price          advance
----------   ----------   ----------------
trad_cook        11.95          4,000.00
trad_cook        14.99          8,000.00
trad_cook        20.95          7,000.00

 Compute Result:
---------------------- -----------------
                  47.89         19,000.00
(7 rows affected)
```

compute 句に指定するカラムは、select リストに含まれている必要があります。

select リストでのカラムの指定順序は、compute 句の集合の順序を上書きします。たとえば、次のようにします。

```
create table t1 (a int, b int, c int null)
insert t1 values(1,5,8)
insert t1 values(2,6,9)

(1 row affected)

compute sum(c),  max(b), min(a)
```

```
select a, b, c from t1

a           b           c
----------- ----------- -----------
          1           5           8
          2           6           9

 Compute Result:
----------- ----------- -----------
          1           6          17
```

ansinull オプションを off (デフォルト値) に設定すると、ロー集合で null が検出されても警告は出されません。ansinull オプションを on に設定すると、ロー集合で null が検出されたときに、クエリから次の SQLSTATE 警告が返されます。

```
Warning- null value eliminated in set function
```

compute 句の出力を結果テーブルに保存する方法がないため、compute 句と同じ文中で select into を使用することはできません。

統計集合関数

集合関数は、データベースに含まれるローのグループのデータを要約します。select 文の group by 句を使用してグループを作成します。

sum、avg、max、min、count_big、count などの単純な集合関数は、select リスト、having、order by 句、そして select 文の compute 句のみで許可されます。これらの関数は、データベースに含まれるローのグループのデータを要約します。

Adaptive Server Enterprise は、数値データの統計的分析を行うための統計集合関数をサポートするようになりました。これらの関数には、stddev、stddev_samp、stddev_pop、variance、var_samp、var_pop が含まれます。

stddev と variance を含むこれらの関数は、クエリの group by 句の指定に従ってローのグループの値を計算できる集合関数です。max や min などのその他の基本的な集合関数と同様に、これらの計算は入力データ内の null 値を無視します。また、分析される式のドメインに関係なく、分散と標準偏差の計算では必ず IEEE の倍精度浮動小数点数が使用されます。

分散関数または標準偏差関数への入力が空のデータ・セットである場合、これらの関数は結果として null 値を返します。分散関数または標準偏差関数への入力が単一の値である場合、これらの関数は結果として 0 を返します。

標準偏差と分散

新しい統計集合関数 (とそのエイリアス) は、次のとおりです。

- stddev_pop (同様に stdevp) － 母標準偏差。グループの各ロー (distinct が指定されている場合、重複が削除された後に残る各ロー) に対して評価される、指定された値式の母標準偏差を計算します。これは、母分散の平方根として定義されます。構文と使用方法については、stddev_pop (272 ページ) を参照してください。
- stddev_samp (同様に stdev と stddev) － 標本標準偏差。グループの各ロー (distinct が指定されている場合、重複が削除された後に残る各ロー) に対して評価される、指定された値式の母標準偏差を計算します。これは、標本分散の平方根として定義されます。構文と使用方法については、stddev_samp (273 ページ) を参照してください。
- var_pop (同様に varp) － 母分散。グループの各ロー (distinct が指定されている場合、重複が削除された後に残る各ロー) に対して評価される、指定された値式の母分散を計算します。これは、値式と値式の平均の差の 2 乗和を、グループ内のローの数で割った値として定義されます。構文と使用方法については、var_pop (310 ページ) を参照してください。
- var_samp (同様に var と variance) － 標本分散。グループの各ロー (distinct が指定されている場合は、重複が削除された後に残る各ロー) に対して評価される値式の標本分散を計算します。これは、値式と値式の平均の差の 2 乗和を、グループ内のローの数より 1 少ない数で割った値として定義されます。構文と使用方法については、var_samp (311 ページ) を参照してください。

統計集合

統計集合は、avg 集合に類似しています。

- 構文は次のとおりです。

  var_pop ([all | distinct] *expression*)
- numeric データ型の式のみが有効です。
- Null 値は計算には関与しません。
- データが計算に関与しない場合、結果は NULL のみです。
- distinct 句または all 句は、式の前に置かれます (デフォルトは all です)。
- 統計集合をベクトル集合 (group by を含む)、スカラ集合 (group by を含まない) として、または compute 句で使用できます。

ただし avg 集合とは異なり、結果は次のようになります。

- 常に float データ型 (つまり、倍精度浮動小数点)、ただし avg 集合の結果のデータ型は式のデータ型と同じです (例外あり)。

- 単一データ・ポイントでは 0.0 です。

計算式

図 2-1: 母集団に関係する統計集合関数の計算式

平均 μ (var_pop) を持つサイズ *n* の母分散を定義する計算式は、次のとおりです。母標準偏差値 (stddev_pop) は、この数値の正の平方根です。

$$\sigma^2 = \frac{\sum (x_i - \mu)^2}{n}$$

σ^2 = 分散
n = 母集団のサイズ
$\mu = x_i$ 値の平均

図 2-2: 標本に関係する統計集合関数の計算式

平均が $\bar{x}$ (var_samp) のサイズ *n* の標本からの母分散の不偏推定値を定義する計算式は、次のとおりです。標本標準偏差 (stddev_samp) は、この数値の正の平方根です。

$$s^2 = \frac{\sum (x_i - \bar{x})^2}{n - 1}$$

s^2 = 分散
n = 標本サイズ
$\bar{x} = x_i$ 値の平均

2 つの計算式の重要な違いは、*n* ではなく *n-1* による除算です。

これら 2 つの計算式は類似していますが、次の異なる目的のために使用されます。

- var_samp − 母集団の標本、つまりサブセットを全母集団の代表として評価するときに使用されます。
- var_pop − 母集団で使用できるデータがすべてある場合や、*n* が大きいために、*n* と *n-1* の差が重要でない場合に使用されます。

データ型変換関数

データ型変換関数は、式のデータ型を別のデータ型への変更と、date および time 情報の新しい表示フォーマットを指定します。データ型変換関数には、次のものがあります。

- cast
- convert
- inttohex
- hextoint
- hextobigint
- biginttohex
- str

データ型変換関数は、where 句、select リスト、および式を使用できる位置で使用できます。

Adaptive Server では、特定のデータ型の変換が自動的に実行されます。これは、「暗黙の変換」と呼ばれます。たとえば、char 式と datetime 式、または smallint 式と int 式、あるいは異なる長さの char 式を比較する場合は、Adaptive Server によって一方のデータ型が他方のデータ型に自動的に変換されます。

その他のデータ型変換については、組み込みデータ型変換関数のいずれかを使用して、明示的に指定する必要があります。たとえば、数値型を連結するには、まずこれらの式を文字式に変換する必要があります。

Adaptive Server には、暗黙的にも明示的にも、別のデータ型に変換できない特定のデータ型があります。たとえば、次の変換はできません。

- smallint データから datetime への変換
- datetime データから smallint への変換
- binary または varbinary データから smalldatetime または datetime データへの変換

サポートされていない変換を行うと、エラー・メッセージが表示されます。

bigtime データ型と bigdatetime データ型の変換

Implicit 変換は新規と既存の発生順のデータ型間で使用できます。

プライマリ・フィールドが一致しないデータ型間で暗黙的な変換が行われると、データのトランケーション、デフォルト値の挿入、またはエラー・メッセージが発生する可能性があります。たとえば、bigdatetime 値を date 値に変換すると、時刻部分がトランケートされ、日付要素のみが残ります。bigtime 値を bigdatetime 値に変換すると、新しい bigdatetime 値の日付要素に、デフォルト日付の Jan 1, 0001 が追加されます。date 値を bigdatetime 値に変換すると、bigdatetime 値の時刻部分に、デフォルト時刻の 00:00:00.000000 が追加されます。

暗黙的または明示的な変換は、精度の低下によってデータ・ロスが発生する場合に許可されます。

表 2-3 と 表 2-4 は、各データ型変換が暗黙的に実行されるか、明示的に実行されるか、サポートされていないかを示しています。

表 2-3: バイナリの明示的、暗黙的、およびサポートされていないデータ型変換 – 符号なし整数

変換前	binary	varbinary	bit	[n]char	[n]varchar	datatime	smalldatetime	bigdatetime	bigtime	tinyint	smallint	unsigned smallint	int	unsigned int
binary	–	I	I	I	I	U	U	I	I	I	I	I	I	I
varbinary	I	–	I	I	I	U	U	I	I	I	I	I	I	I
bit	I	I	–	I	I	U	U	U	U	I	I	I	I	I
[n]char	I	I	E	–	I	I	I	I	I	E	E	E	E	E
[n]varchar	I	I	E	I	–	I	I	I	I	E	E	E	E	E
datetime	I	I	U	I	I	–	I	I	I	U	U	U	U	U
smalldatetime	I	I	U	I	I	I	–	I	I	U	U	U	U	U
bigdatetime	I	I	U	I	I	I	I	-	I	U	U	U	U	U
bigtime	I	I	U	I	I	I	I	I	-	U	U	U	U	U
tinyint	I	I	I	E	E	U	U	U	U	–	I	I	I	I
smallint	I	I	I	E	E	U	U	U	U	I	–	I	I	I
unsigned smallint	I	I	I	E	E	U	U	U	U	I	I	–	I	I
int	I	I	I	E	E	U	U	U	U	I	I	I	–	I
unsigned int	I	I	I	E	E	U	U	U	U	I	I	I	I	–
bigint	I	I	I	E	E	U	U	U	U	I	I	I	I	I
unsigned bigint	I	I	I	E	E	U	U		U	I	I	I	I	I
decimal	I	I	I	E	E	U	U	U	U	I	I	I	I	I
numeric	I	I	I	E	E	U	U	U	U	I	I	I	I	I
float	I	I	I	E	E	U	U	U	U	I	I	I	I	I
real	I	I	I	E	E	U	U	U	U	I	I	I	I	I
money	I	I	I	I	I	U	U	U	U	I	I	I	I	I
smallmoney	I	I	I	I	I	U	U	U	U	I	I	I	I	I
text	U	U	U	E	E	U	U	U	U	U	U	U	U	U
unitext	E	E	E	E	E	U	U	U	U	U	U	U	U	U
image	E	E	U	U	U	U	U	U	U	U	U	U	U	U
unichar	I	I	E	I	I	I	I	I	I	E	E	E	E	E
univarchar	I	I	E	I	I	I	I	I	I	E	E	E	E	E
date	I	I	U	I	I	I	U	I	U	U	U	U	U	U
time	I	I	U	I	I	I	U	I	I	U	U	U	U	U

表 *2-4:* 暗黙的、明示的、およびサポートされていない *bigint - time* データ型変換

変換前	bigint	unsigned bigint	decimal	numeric	float	real	money	smallmoney	text	unitext	image	unichar	univarchar	data	time
binary	I	I	I	I	I	I	I	I	U	I	I	I	I	I	I
varbinary	I	I	I	I	I	I	I	I	U	I	I	I	I	I	I
bit	I	I	I	I	I	I	I	I	U	U	U	E	E	U	U
[n]char	E	E	E	E	E	E	E	E	I	I	I	I	I	I	I
[n]varchar	E	E	E	E	E	E	E	E	I	I	I	I	I	I	I
datetime	U	U	U	U	U	U	U	U	U	U	U	I	I	I	I
smalldatetime	U	U	U	U	U	U	U	U	U	U	U	I	I	I	I
bigdatetime	U	U	U	U	U	U	U	U	U	U	U	I	I	I	I
bigtime	U	U	U	U	U	U	U	U	U	U	U	I	I	U	I
tinyint	I	I	I	I	I	I	I	I	U	U	U	E	E	U	U
smallint	I	I	I	I	I	I	I	I	U	U	U	U	E	U	U
unsigned smallint	I	I	I	I	I	I	I	I	U	U	U	E	E	U	U
int	I	I	I	I	I	I	I	I	U	U	U	E	E	U	U
unsigned int	I	I	I	I	I	I	I	I	U	U	U	E	E	U	U
bigint	–	I	I	I	I	I	I	I	U	U	U	E	E	U	U
unsigned bigint	I	–	I	I	I	I	I	I	U	U	U	E	E	U	U
decimal	I	I	–	I	I	I	I	I	U	U	U	E	E	U	U
numeric	I	I	I	–	I	I	I	I	U	U	U	E	E	U	U
float	I	I	I	I	–	I	I	I	U	U	U	E	E	U	U
real	I	I	I	I	I	–	I	I	U	U	U	E	E	U	U
money	I	I	I	I	I	I	–	I	U	U	U	E	E	U	U
smallmoney	I	I	I	I	I	I	I	–	U	U	U	E	E	U	U
text	U	U	U	U	U	U	U	U	–	I	U	E	E	U	U
unitext	U	U	U	U	U	U	U	U	I	–	I	U	U	U	U
image	U	U	U	U	U	U	U	U	U	I	–	E	E	U	U
unichar	E	E	E	E	E	E	E	E	I	I	I	–	I	I	I
univarchar	E	E	E	E	E	E	E	E	I	I	I	I	–	I	I
date	U	U	U	U	U	U	U	U	U	U	U	I	I	–	I
time	U	U	U	U	U	U	U	U	U	U	U	I	I	I	–

データ型変換キー

- E − 明示的なデータ型変換が必要です。
- I − 変換は暗黙的に実行されるか、または明示的データ型変換関数によって実行できます。
- I/E − 精度または位取りのロスが発生しており、arithabortnumeric_truncation がオンになっている場合は、明示的データ型変換関数を実行する必要があります。それ以外の場合は、暗黙的変換が許可されます。
- U − 変換がサポートされていません。
- - − 同一データ型へのデータ型の変換。このような変換は可能ですが、意味はありません。

文字型データの非文字型への変換

文字データは、完全に新しい型に有効な文字からなる場合、通貨、日付／時刻、真数値、または概数値型などの非文字型に変換できます。先行ブランクは無視されます。ただし、1 つまたは複数のブランクで構成される char データ型の式を datetime データ型の式に変換した場合は、Adaptive Server によって、ブランクがデフォルトの datetime 値「Jan 1, 1900」に変換されます。

受け入れられない文字がデータに含まれている場合には構文エラーが生成されます。次に、構文エラーの原因となる文字の例をいくつか示します。

- 整数データ内のカンマまたは小数点
- 通貨データ内のカンマ
- 真数値データまたは概数値データ、あるいはビット・ストリーム・データの文字
- 日付**データ**および**時刻**データ内の月名のスペルミス

unichar/univarchar および datetime/smalldatetime 間の暗黙的な変換はサポートされています。

ある文字型から他の文字型への変換

マルチバイト文字セットをシングルバイト文字セットに変換する場合、該当するシングルバイトのない文字は疑問符に変換されます。

text カラムおよび unitext カラムは、char、nchar、varchar、unichar、univarchar、または nvarchar に明示的に変換できます。変換は、サーバの論理ページ・サイズの最大カラム・サイズによって決定される character 型の最大長に制限されます。長さを指定しない場合、変換された値はデフォルトの長さである 30 バイトになります。

数値型データの文字型への変換

真数値データと概数値データは文字型に変換できます。新しい型が短すぎて文字列が収まらない場合は、空き領域不足エラーが発生します。たとえば、次に示す変換では 1 文字の型に 5 文字の文字列を格納しようとしています。

```
select convert(char(1), 12.34)

Insufficient result space for explicit conversion
of NUMERIC value '12.34' to a CHAR field.
```

float データを文字型に変換する場合、新しい型の長さは少なくとも 25 文字にする必要があります。

注意　変換を行うときには、convert または cast よりも str 関数の使用をおすすめします。この関数には変換を詳細に制御する機能があり、エラーを回避できます。

通貨型との変換中の丸め処理

money 型と smallmoney 型は、小数点の右側に 4 桁の数字を格納しますが、表示の目的で、最も近い 100 分の 1 単位 (.01) まで数値が丸められます。通貨型に変換されたデータは、4 桁まで丸められます。

通貨型から変換されたデータでは、可能であれば同じ丸め処理が行われます。新しい型が小数点以下の桁数が 3 桁未満の真数値である場合、データは新しい型の位取りに従って丸められます。たとえば、$4.50 が整数に変換されると 5 になります。

```
select convert(int, $4.50)
 -----------
           5
```

money または smallmoney に変換されたデータの単位は、セントなどの補助貨幣単位ではなく、ドルなどの主要貨幣単位とみなされます。たとえば英語の場合、整数値 5 は、5 セントではなく 5 ドルに等しい通貨に変換されます。

日付と時刻情報の変換

日付として認識されるデータは、datetime、smalldatetime、date、または time に変換できます。誤った月の名前は、構文エラーの原因になります。また、データ型の許容範囲外の日付は、算術オーバフロー・エラーの原因になります。

datetime 値が smalldatetime 値に変換される場合、この値は最も近い分数まで丸められます。

date データを文字型に変換するときは、表 2-7 (110 ページ) に示すスタイル番号の 1 ～ 7 (101 ～ 107) または 10 ～ 12 (110 ～ 112) を使って表示フォーマットを指定します。デフォルト値は 100 (mon dd yyyy hh:miAM (または PM)) です。時刻部分を含むスタイルに date データを変換する場合は、時刻部分はデフォルト値の 0 になります。time データを文字型に変換するときは、スタイル番号の 8 または 9 (108 または 109) を使って表示フォーマットを指定します。デフォルトは 100 (mon dd yyyy hh:miAM (または PM)) です。日付部分を含むスタイルに time データを変換する場合は、デフォルト日付の Jan 1, 1900 が表示されます。

数値型間での変換

データは、ある数値型から別の数値型に変換できます。新しい型が、データを収容できるだけの精度または位取りがない真数値型である場合は、エラーが発生します。

たとえば、整数を予期する組み込み関数への引数として float 値または数値を指定すると、その float 値または数値の値はトランケートされます。ただし Adaptive Server では、小数部分を持つ数値は暗黙的に変換されず、位取りエラー・メッセージが返されます。たとえば、Adaptive Server では、小数部分を持つ数値の場合はエラー 241、他のデータ型が渡された場合はエラー 257 が返されます。

数値変換によって生じたエラーの Adaptive Server による処理方法を決定するには、arithabort と arithignore の各オプションを使用してください。

算術オーバフロー・エラーと 0 による除算エラー

0 による除算エラーは、Adaptive Server が数値をゼロで除算しようとすると発生します。算術オーバフロー・エラーは、新しい型に、結果が収まるだけの小数の桁がない場合に発生します。これは、次の変換時に発生します。

- 精度または位取りが少ない真数値型への明示的または暗黙的な変換
- 通貨型または日付／時刻型に受け入れられる範囲を外れるデータの明示的または暗黙的変換
- hextoint を使用した 4 バイト以上の記憶領域を必要とする 16 進文字列の変換

算術オーバフロー・エラーと 0 による除算エラーは、暗黙的または明示的変換のどちらで発生したかに関係なく、重大エラーとみなされます。Adaptive Server によるこれらのエラーの処理方法を決定するには、arithabort arith_overflow オプションを使用してください。デフォルト設定の arithabort arith_overflow on では、エラーが発生したトランザクション全体がロールバックされます。arithabort arith_overflow on が設定されていると、トランザクションを含まないバッチでエラーが発生した場合、エラーの発生前にバッチに含まれていたコマンドはロールバックしません。ただし Adaptive Server は、バッチでエラーを起こした文のあとでは、どの文も実行しません。arithabort arith_overflow off を設定すると、Adaptive Server は、エラーの原因になる文をアボートしますが、トランザクションまたはバッチの他の文の処理を継続します。*@@error* グローバル変数を使用すると、文の結果を検査できます。

Adaptive Server がこれらのエラー発生後にメッセージを表示するかどうかを決定するには、arithignore arith_overflow オプションを使用してください。デフォルト設定の off では、0 による除算エラーまたは精度のロスが発生すると、警告メッセージが表示されます。arithignore arith_overflow on を設定すると、これらのエラーが発生しても警告メッセージは表示されません。オプションの arith_overflow キーワードを省略しても影響はありません。

位取りのエラー

明示的な変換によって位取りのロスが発生すると、この結果は警告なしにトランケートされます。たとえば、float、numeric、または decimal 型を integer 型に明示的に変換すると、Adaptive Server では、結果を整数にして、小数点の右側にあるすべての数字をトランケートするものと想定されます。

numeric 型または decimal 型への暗黙の変換時に、位取りのロスがあると、位取りエラーが発生します。arithabort numeric_truncation オプションを使用して、エラーの重大度を調べます。デフォルト設定の arithabort numeric_truncation on は、エラーを起こした文をアボートしますが、トランザクションまたはバッチ内のその他の文の処理は継続します。arithabort numeric_truncation off を設定した場合、Adaptive Server はクエリ結果をトランケートして処理を継続します。

注意　エントリ・レベルの ANSI SQL に準拠する場合は、次のオプションを設定してください。

- arithabort arith_overflow off
- arithabort numeric_truncation on
- arithignore off

ドメイン・エラー

convert 関数は、定義されている範囲以外の引数が指定されると、ドメイン・エラーとなります。このエラーは、ほとんど発生しません。

バイナリ型と整数型の変換

binary 型と varbinary 型は、“0x” プレフィクスとその後に続く数字と文字の文字列からなる 16 進数に似たデータを格納します。

これらの文字列はプラットフォームによって解釈が異なります。たとえば、文字列「0x0000100」は、バイト 0 を最上位とみなす (リトル・エンディアン) マシンでは 65536 を表し、バイト 0 を最下位とみなす (ビッグ・エンディアン) マシンでは 256 を表します。

バイナリ型から整数値型への変換は、convert 関数を使用して明示的に実行するか、または暗黙的に実行できます。データが小さすぎて新しい型を収められない場合は、その “0x” プレフィクスと埋め込みゼロが削除されます。データが長すぎる場合は、トランケートされます。

convert と暗黙的データ型変換でのバイナリ・データの評価は、いずれもプラットフォームによって異なります。このため、結果はプラットフォームによって異なることがあります。プラットフォームの影響を受けずに 16 進文字列を整数に変換するには hextoint 関数を使用し、プラットフォームの影響を受けずに整数を 16 進値に変換するには inttohex 関数を使用してください。hextobigint 関数は、プラットフォームに依存しない 16 進文字列から 64 ビット整数への変換に使用します。また、biginttohex 関数は、プラットフォームに依存しない 64 ビット整数から 16 進数の値への変換に使用します。

バイナリ型と数値型または 10 進数型の間のデータ変換

binary および varbinary データ文字列では、“0x” に続く最初の 2 つの数字は binary 型を表します。“00” は正の数値を表し、“01” は負の数値を表します。binary または varbinary 型のデータを numeric または decimal に変換する場合は、必ず “0x” 桁の後ろに “00” または “01” の値を指定してください。このように指定されていないと変換は失敗します。

次の例では、binary 型データを numeric 型データに変換する方法を示します。

```
select convert(numeric
(38, 18),0x00000000000000006b14bd1e6eea0000000000000000000000000000000000)

----------
123.456000
```

次に、前の例と同じデータを numeric 型から binary 型に変換する例を示します。

```
select convert(binary,convert(numeric(38, 18), 123.456))

------------------------------------------------------------------
0x00000000000000006b14bd1e6eea0000000000000000000000000000000000
```

image カラムからバイナリ型への変換

convert 関数を使用すると、image カラムを binary または varbinary に変換できます。binary データ型の最大長は制限されています。この最大長は、サーバの論理ページ・サイズの最大カラム・サイズによって決定します。長さを指定しない場合、変換された値はデフォルトの長さである 30 文字になります。

他の型から *bit* への変換

真数値型と概数値型は、bit 型に暗黙的に変換できます。文字型の場合は、明示的な convert 関数が必要です。

変換される式は、数字、小数点、通貨記号、プラス符号またはマイナス符号だけで構成します。他の文字があると、構文エラーが発生します。

0 に対応する bit は 0 です。他のすべての数字に対応する bit は 1 です。

NULL 値の変換

NULL から NOT NULL への変換と NOT NULL から NULL への変換を実行するには、convert 関数を使用します。

日付関数

日付関数は、データ型 datetime、bigtime、bigdatetime、smalldatetime、date、または time の値を処理します。

日付関数は、select リストまたはクエリの where 句で使用できます。

datetime データ型は、1753 年 1 月 1 日より後の日付だけに使用します。datetime 値は一重引用符か二重引用符で囲んでください。0001 年 1 月 1 日から 9999 年 1 月 1 日までの日付には、date を使用します。date 値は一重引用符か二重引用符で囲んでください。それよりも前の日付には、char、nchar、varchar、または nvarchar を使います。Adaptive Server では、さまざまな種類の日付フォーマットを識別できます。詳細については、「データ型変換関数」(57 ページ) と「日付と時刻のデータ型」(18 ページ) を参照してください。

Adaptive Server は、必要に応じて (たとえば、文字値を datetime 値と比較するとき) 文字値と datetime 値とを自動的に変換します。

date データ型は、0001 年 1 月 1 日から 9999 年 1 月 1 日までの日付を対象にできます。

日付要素

日付要素は Adaptive Server によって認識される略語であり、受け入れられる値は次のとおりです。

日付要素	略語	値
year	yy	1753 ～ 9999 (datetime) 1753 ～ 2079 (smalldatetime) 0001 ～ 9999 (bigdatetime)
quarter	qq	1 ～ 4
month	mm	1 ～ 12
week	wk	1 ～ 54
day	dd	1 ～ 31
dayofyear	dy	1 ～ 366
weekday	dw	1 ～ 7 (日曜 ～ 土曜)
hour	hh	0 ～ 23
minute	mi	0 ～ 59
second	ss	0 ～ 59
millisecond	ms	0 ～ 999
microseconds	us	0 ～ 999999

2 桁の年 (*yy*) を入力する場合には、以下の点に注意してください。

- 50 未満の数値は 20*yy* として解釈されます。たとえば、`01` は 2001、`32` は 2032、`49` は 2049 です。
- 50 以上の数値は 19*yy* と解釈されます。たとえば、`50` は 1950、`74` は 1974、`99` は 1999 です。

datetime、smalldatetime、および time 値では、ミリ秒の前にコロンまたはピリオドを付けることができます。コロンに続ける場合、数字は 1 秒の 1000 分の 1 を意味します。ピリオドに続ける場合、1 桁は 1 秒の 10 分の 1、2 桁は 100 分の 1、3 桁は 1000 分の 1 を意味します。たとえば、「12:30:20:1」は 12 時 30 分から 20 秒と 1000 分の 1 秒が経過したことを示し、「12:30:20.1」は 12 時 30 分から 20 秒と 10 分の 1 秒が経過したことを示します。datetime データまたは time データの挿入時に、Adaptive Server によってミリ秒値は丸められるか、トランケートされます。これらのデータ型の細分性は 1000 分の 1 秒ではなく 300 分の 1 秒だからです。時刻情報には、time データ型を使用できます。

bigdatetime および bigtime 時刻値の場合、マイクロ秒には前に小数点を付けて端数を表す必要があります。

算術関数

算術関数は、算術データの演算に共通して必要な値を返します。算術関数名はキーワードではありません。

また、各関数には、指定のデータ型に暗黙的に変換できる引数を指定することもできます。たとえば、概数値型を受け入れる関数は、整数型も受け入れます。Adaptive Server により、引数は指定のデータ型に自動的に変換されます。

算術関数には、次のものがあります。

- abs
- acos
- asin
- atan
- atn2
- ceiling
- cos
- cot
- degrees
- exp
- floor
- lockscheme
- log
- log10
- pagesize
- pi
- power
- radians
- rand
- round
- sign
- sin
- sqrt
- tan

これらの関数のドメイン・エラーまたは範囲エラーを処理するために、エラー・トラップが用意されています。ユーザは、ドメイン・エラーの処理方法を指定するために、arithabort と arithignore の各オプションを設定 (set) できます。

- arithabort arith_overflow は、0 による除算エラーや精度ロスが発生した後の動作を指定します。デフォルト設定の arithabort arith_overflow on では、トランザクション全体をロールバックするか、またはエラーが発生しているバッチをアボートします。arithabort arith_overflow off を設定した場合には、Adaptive Server はエラーを発生させた文をアボートしますが、トランザクションまたはバッチ内のその他の文の処理は継続します。
- arithabort numeric_truncation には、暗黙的データ変換中に真数値型による位取りのロスが発生した後の動作を指定します (明示的変換によって位取りのロスが発生すると、変換結果は警告なしにトランケートされます)。デフォルト設定の arithabort numeric_truncation on は、エラーを起こした文をアボートしますが、トランザクションまたはバッチ内のその他の文の処理は継続します。arithabort numeric_truncation off を設定した場合、Adaptive Server はクエリ結果をトランケートして処理を継続します。
- デフォルトでは、arithignore arith_overflow オプションは off になるため、Adaptive Server では、数値オーバフローの原因になるすべてのクエリのあとで警告メッセージが表示されます。オーバフロー・エラーを無視するには、arithignore オプションを on に設定してください。

セキュリティ関数

セキュリティ関数は、セキュリティ関連情報を返します。

セキュリティ関数には、次のものがあります。

- is_sec_service_on
- show_sec_services
- get_appcontext
- list_appcontext
- set_appcontext
- rm_appcontext
- show_role
- proc_role
- role_contain
- role_id
- role_name

文字列関数

文字列関数は、バイナリ・データ、文字列、式を操作します。文字列関数には、次のものがあります。

- ascii
- char
- charindex
- char_length
- difference
- lower
- ltrim
- patindex
- replicate
- reverse
- right
- rtrim
- soundex
- space
- str
- str_replace
- stuff
- substring
- to_unichar
- uhighsurr
- ulowsurr
- upper
- uscalar

文字列関数はネストできます。また、この関数は、select リスト、where 句、または式を使用できる位置で使用します。文字列関数内で定数を使用する場合は、定数を一重または二重の引用符で囲んでください。文字列関数名は、キーワードではありません。

各文字列関数は、指定の型に暗黙的に変換可能な引数も受け入れます。たとえば、概数値の式を受け入れる関数は、整数式も受け入れます。Adaptive Server により、引数は指定のデータ型に自動的に変換されます。

文字列関数で 2 つの文字式が受け入れられるが、1 つの式が unichar の場合、もう 1 つの式が「拡大」され、内部で unichar に変換されます。これは、混合モードの式に関する既存のルールに従っています。ただし、unichar データが 2 倍のスペースを占有することがあるため、この変換によってトランケーションが発生することがあります。

文字列関数の制限

文字列関数の結果は 16K に制限されます。この制限はサーバのページ・サイズに依存しません。ページ・サイズが 2K の場合でも、Transact-SQL の文字列関数、文字列変数、リテラルは 16K まで使用できます。

set string_rtruncation が on の場合、insert または update で文字列がトランケートされると、ユーザはエラーを受け取ります。しかし、表示される文字列がトランケートされても、Adaptive Server はエラーをレポートしません。たとえば、次のようにします。

```
select replicate("a", 16383) + replicate("B", 4000)
```

この例は、全長が 20383 になっても、結果文字列は 16K に制限されていることを示します。

システム関数

システム関数は、データベースの特別な情報を返します。システム関数には、次のものがあります。

- col_length
- col_name
- curunreservedpgs
- data_pages
- datalength
- db_id
- db_name
- host_id
- host_name
- index_col
- is_quiesced
- isnull
- object_id
- object_name
- reserved_pages
- row_count
- show_role
- suser_id
- suser_name
- tempdb_id
- tran_dumpable_status
- tsequal
- used_pages
- user
- user_id
- user_name
- valid_name
- valid_user

システム関数は、select リスト、where 句内、式を使用できる位置で使用できます。

システム関数への引数がオプションの場合は、現在のデータベース、ホスト・コンピュータ、サーバ・ユーザ、またはデータベース・ユーザを前提としています。

text、unitext、image カラム

text カラム、unitext カラム、image カラムには、次のような制限事項があります。

- ストアド・プロシージャのパラメータとして使用できません。
- ストアド・プロシージャに渡される値として使用できません。
- ローカル変数として使用できません。
- order by 句、compute 句、group by 句内では使用できません。
- インデックス内では使用できません。
- キーワード like とともに使用する場合を除き、where 句内では使用できません。
- ジョインでは使用できません。

トリガでは、挿入するテキスト値と削除するテキスト値の両方で新しい値が参照され、古い値は参照できません。

text 関数および image 関数

text 関数と image 関数は、text、image、unitext データを操作します。text 関数と image 関数には、次のものがあります。

- textptr
- textvalid

text および image 組み込み関数名は、キーワードではありません。select 文によって取り出される text、image、unitext データの量を制限するには、set textsize オプションを使用してください。

text 関数 patindex は、text カラム、image カラム、unitext カラムに使用可能であり、text 関数および image 関数とみなすこともできます。

text カラム、image カラム、unitext カラムのデータの長さを表示するには、datalength 関数を使用できます。

ユーザ定義 SQL 関数

スカラ関数に次を含めることができます。

- 関数に対してローカルであるデータ変数およびカーソルを定義する declare 文
- 関数にローカルなオブジェクトに割り当てられた値 (たとえば、select コマンドまたは set コマンドを含むテーブルに対してローカルなスカラおよび変数に割り当てられた値)
- 関数で宣言、オープン、クローズ、割り付け解除されたローカル・カーソルを参照するカーソル操作
- フロー制御文
- set オプション (関数のスコープでのみ有効)

Adaptive Server は、データをクライアントに返すスカラ関数に fetch 文を許可しません。次を含めることはできません。

- データをクライアントに返す select または fetch 文
- insert、update、または delete 文
- dbcc、dump、load コマンドなどのユーティリティ・コマンド
- print 文
- rand、rand2、getdate、または newid を参照する文

ローカル変数にのみ値を割り当てる select または fetch 文を含めることができます。

abs

説明

式の絶対値を返します。

構文

abs(*numeric_expression*)

パラメータ

numeric_expression

カラム、変数、またはデータ型が真数値、概数値、通貨、またはこれらの型の 1 つに暗黙的に変換できる式です。

例

-1 の絶対値である 1 を返します。

```
select abs(-1)

-----------
          1
```

使用法

算術関数 abs は、指定された式の絶対値を返します。結果は、数値式と同じ型で、精度や位取りも同じです。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが abs 関数を実行できます。

参照

算術関数の概要については、「算術関数」(67 ページ) を参照してください。

関数 ceiling, floor, round, sign

acos

説明　余弦が指定されている角度 (ラジアン) を返します。

構文　acos(*cosine*)

パラメータ　*cosine*

カラム名、変数、または (float、real、double precision、またはこれらの型の 1 つに暗黙的に変換できる任意のデータ型の) 定数として表される角度の余弦です。

例　余弦が 0.52 である角度 1.023945 ラジアンを返します。

```
select acos(0.52)

--------------------
            1.023945
```

使用法　算術関数 acos は、指定した値を余弦とする角度 (ラジアン) を返します。

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　すべてのユーザが acos 関数を実行できます。

参照　算術関数の概要については、「算術関数」(67 ページ) を参照してください。

関数　cos, degrees, radians

ascii

説明

式の先頭文字の ASCII コードを返します。

構文

ascii(*char_expr* | *uchar_expr*)

パラメータ

char_expr

文字型のカラム名、変数、または (char、varchar、nchar、または nvarchar 型の) 定数式です。

uchar_expr

文字型のカラム名、変数、または (unichar または univarchar 型の) 定数式です。

例

ASCII コードが 70 未満の場合に、作者の姓とその先頭文字の ASCII コードを返します。

```
select au_lname, ascii(au_lname) from authors
where ascii(au_lname) < 70

 au_lname
 ------------------------------ -----------
 Bennet                                  66
 Blotchet-Halls                          66
 Carson                                  67
 DeFrance                                68
 Dull                                    68
```

使用法

- 文字列関数 ascii は式の先頭文字の ASCII コードを返します。
- 文字列関数で 2 つの文字式が受け入れられるが、1 つの式が unichar の場合、もう 1 つの式が「拡大」され、内部で unichar に変換されます。これは、混合モードの式に関する既存のルールに従っています。ただし、unichar データが 2 倍のスペースを占有することがあるため、この変換によってトランケーションが発生することがあります。
- *char_expr* または *uchar_expr* が NULL の場合は、NULL を返します。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが ascii 関数を実行できます。

参照

文字列関数の概要については、「文字列関数」(68 ページ) を参照してください。

関数 char, to_unichar

asehostname

説明　Adaptive Server が動作する物理ホストまたは仮想ホストを返します。

構文　asehostname

パラメータ

なし。

例　Adaptive Server のホスト名を返します。

```
select asehostname()

-----------------------------------
linuxkernel.sybase.com
```

規格　SQL/92 および SQL/99 対応

パーミッション　asehostname コマンドを実行できるのは、sa_role が与えられているユーザだけです。

asin

説明

正弦が指定されている角度 (ラジアン) を返します。

構文

asin(*sine*)

パラメータ

sine

カラム名、変数、または (float、real、double precision、またはこれらの型の 1 つに暗黙的に変換できる任意のデータ型の) 定数として表される角度の正弦です。

例

```
select asin(0.52)

--------------------
                0.546851
```

使用法

算術関数 asin は、指定した値を正弦とする角度 (ラジアン) を返します。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが asin 関数を実行できます。

参照

算術関数の概要については、「算術関数」(67 ページ) を参照してください。

関数 degrees, radians, sin

atan

説明　正接が指定されている角度 (ラジアン) を返します。

構文　atan(*tangent*)

パラメータ　*tangent*

カラム名、変数、または (float、real、double precision、またはこれらの型の 1 つに暗黙的に変換できる任意のデータ型の) 定数として表される角度の正接です。

例

```
select atan(0.50)

--------------------
                0.463648
```

使用法　算術関数 atan は、指定した値を正接とする角度 (ラジアン) を返します。

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　すべてのユーザが atan 関数を実行できます。

参照　算術関数の概要については、「算術関数」(67 ページ) を参照してください。

関数　atn2, degrees, radians, tan

atn2

説明

正弦と余弦が指定されている角度 (ラジアン) を返します。

構文

atn2(*sine*, *cosine*)

パラメータ

sine

カラム名、変数、または (float、real、double precision、またはこれらの型の 1 つに暗黙的に変換できる任意のデータ型の) 定数として表される角度の正弦です。

cosine

カラム名、変数、または (float、real、double precision、またはこれらの型の 1 つに暗黙的に変換できる任意のデータ型の) 定数として表される角度の余弦です。

例

```
select atn2(.50, .48)

--------------------
            0.805803
```

使用法

算術関数 atn2 は、正弦と余弦が指定されている角度 (ラジアン) を返します。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが atn2 関数を実行できます。

参照

算術関数の概要については、「算術関数」(67 ページ) を参照してください。

関数 atan, degrees, radians, tan

avg

説明

すべて (または重複を除くすべて) の値の平均値を返します。

構文

avg([all | distinct] *expression*)

パラメータ

all

すべての値に avg を適用します。デフォルト設定は all です。

distinct

avg を適用する前に重複する値を削除します。distinct はオプションです。

expression

カラム名、定数、関数、これらを算術演算子やビット処理演算子でつないだ任意の組み合わせ、またはサブクエリです。通常、集合関数では、expression はカラム名です。詳細については、「式」(341 ページ) を参照してください。

例

例 1 すべてのビジネス雑誌について前払い金の平均と総売上額を計算します。これらの各集合関数は、検索したすべてのローに対して 1 つの合計値を返します。

```
select avg(advance), sum(total_sales)
  from titles
where type = "business"

------------------------ -----------
                6,281.25       30788
```

例 2 group by 句を指定すると、集合関数はテーブル全体ではなくグループごとに 1 つの値を計算します。この文では、本の種類ごとに合計値を計算します。

```
select type, avg(advance), sum(total_sales)
  from titles
group by type

type
 ------------ ------------------------ -----------
 UNDECIDED                       NULL        NULL
 business                    6,281.25       30788
 mod_cook                    7,500.00       24278
 popular_comp                7,500.00       12875
 psychology                  4,255.00        9939
 trad_cook                   6,333.33       19566
```

例 3 titles テーブルは出版社別のグループに分けられ、前払い総額が 25,000 ドルを超え、本の平均価格が 15 ドルを超える出版社のグループだけが合計されます。

```
select pub_id, sum(advance), avg(price)
  from titles
group by pub_id
having sum(advance) > $25000 and avg(price) > $15

 pub_id
 ------ -------------------- --------------------
 0877              41,000.00                15.41
 1389              30,000.00                18.98
```

使用法

- avg 集合関数は、カラム内の値の平均を計算します。avg を使用できるのは、数値 (整数、浮動小数、通貨) のデータ型だけです。平均値の計算では null 値は無視されます。
- int, smallint, tinyint データ (符号付きまたは符号なし) を平均すると、Adaptive Server では結果が int 値として返されます。bigint データ (符号付きまたは符号なし) を平均すると、結果は bigint 値として返されます。DB-Library プログラムでのオーバフロー・エラーを避けるために、結果を格納するのに適した変数を宣言してください。
- バイナリ・データ型では、avg を使用できません。
- 平均値は数値データ型としてのみ定義されているため、avg に Unicode 式を指定して実行すると、エラーが発生します。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが avg 関数を実行できます。

参照

集合関数の概要については、「集合関数」(48 ページ) を参照してください。

関数 max, min

audit_event_name

説明　監査イベントの説明を返します。

構文　audit_event_name(*event_id*)

パラメータ　*event_id*

監査イベントの番号です。

例　**例 1** テーブル作成イベントの監査証跡を問い合わせます。

```
select * from audit_data where audit_event_name(event) = "Create Table"
```

例 2 現在の監査イベント値を取得します。監査の値と説明のリストについては、下の「使用法」を参照してください。

```
create table #tmp(event_id int, description varchar(255))
go
declare @a int
select @a=1
while (@a<120)
begin
      insert #tmp values (@a, audit_event_name(@a))
      select @a=@a + 1
end
select * from #tmp
go

----------------------------------------
event_id    description
---------   -------------------
      1     Ad hoc Audit Record
      2     Alter Database
      ...
      104   Create Index
      105   Drop Index
```

使用法　次のリストに、監査イベントの ID と名前を示します。

1 特定の監査レコード
2 データベースの変更
3 テーブルの変更
4 BCP 入力
5 NULL
6 デフォルトのバインド
7 メッセージのバインド
8 ルールのバインド
9 データベースの作成
10 テーブルの作成
11 プロシージャの作成
12 トリガの作成
13 ルールの作成
14 デフォルトの作成
15 メッセージの作成
16 ビューの作成
17 データベースへのアクセス
18 テーブルの削除
19 ビューの削除
20 ディスクの初期化
21 ディスクの修復
22 ディスクの再初期化
23 ディスクのミラーリング
24 ディスクのミラーリング解除
25 ディスクの再ミラーリング
26 データベースの削除
27 テーブルの削除
28 プロシージャの削除
29 トリガの削除
30 ルールの削除
31 デフォルトの削除
32 メッセージの削除
33 ビューの削除
34 データベースのダンプ
35 トランザクションのダンプ
36 致命的なエラー
37 致命的でないエラー
38 ストアド・プロシージャの実行
39 トリガの実行
40 grant コマンド
41 テーブルの挿入
42 ビューの挿入
43 データベースのロード
44 トランザクションのロード
45 ログイン
46 ログアウト
47 revoke コマンド
48 RPC 入力
49 RPC 出力
50 サーバのブート
51 サーバの停止
52 NULL
53 NULL
54 NULL
55 役割のオン／オフ
56 NULL
57 NULL
58 NULL
59 NULL
60 NULL
61 監査テーブルへのアクセス
62 テーブルの選択
63 ビューの選択
64 テーブルのトランケート
65 NULL
66 NULL
67 デフォルトのバインド解除
68 ルールのバインド解除
69 メッセージのバインド解除
70 テーブルの更新
71 ビューの更新
72 NULL
73 監査の有効化
74 監査の無効化
75 NULL
76 SSO パスワードの変更
79 NULL
80 役割チェックの実行
81 DBCC コマンド
82 設定
83 データベースのオンライン化
84 setuser コマンド
85 ユーザ定義関数コマンド
86 組み込み関数
87 ディスクの解放
88 set SSA コマンド
90 connect コマンド
91 参照
92 コマンド・テキスト
93 JCS install コマンド
94 JCS remove コマンド
95 管理者アカウントのロック解除
96 quiesce database コマンド
97 SQLJ 関数の作成
98 SQLJ 関数の削除
99 SSL の管理
100 ディスクのサイズ変更
101 データベースのマウント
102 データベースのマウント解除
103 login コマンド
104 インデックスの作成
105 インデックスの削除
106 NULL
107 NULL
108 NULL
109 NULL
110 UDWS の展開
111 UDWS の展開解除
115 パスワードの管理

注意 audit_event_name が NULL を返す場合、Adaptive Server はイベントをログ記録しません。

規格 ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション すべてのユーザが audit_event_name を実行できます。

参照 **コマンド** select、sp_audit

authmech

説明

指定したログイン済みサーバのプロセス ID が使用している認証メカニズムを判断します。

構文

authmech ([*spid*])

例

例 1 サーバ・プロセス ID 42 の認証メカニズムとして、KERBEROS、LDAP、またはその他のメカニズムを返します。

```
select authmech(42)
```

例 2 現在のログインのサーバ・プロセス ID の認証メカニズムを返します。

```
select authmech()
```

または

```
select authmech(0)
```

例 3 各ログイン・セッションに使用される認証メカニズムを出力します。

```
select suid, authmech(spid)
    from sysprocesses where suid!=0
```

使用法

- この関数は 1 つのオプション引数から型 varchar の出力を返します。
- サーバ・プロセス ID の値が 0 の場合は、現在のクライアント・セッションのサーバ・プロセス ID で使用されている認証方式を返します。
- 引数を指定しない場合は、サーバ・プロセス ID の値が 0 の場合と同じ出力になります。
- 可能な戻り値は、`ldap`、`ase`、`pam`、NULL などです。

パーミッション

すべてのユーザが authmech を使用して現在の個人セッションにクエリできます。別のユーザのセッションの詳細をクエリするには、sso_role 権限が必要です。

biginttohex

説明

指定した整数値に相当する、プラットフォームの影響を受けない 8 バイトの 16 進文字列を返します。

構文

biginttohex (*integer_expression*)

パラメータ

integer_expression

16 進文字列に変換される整数値です。

例

この例は、-9223372036854775808 という bigint 値を 16 進文字列に変換します。

```
1> select biginttohex(-9223372036854775808)
2> go
----------------
8000000000000000
```

使用法

- データ型変換関数 biginttohex は、指定された整数値に相当する、プラットフォームの影響を受けない 16 進文字列を、プレフィクス「0x」を付けないで返します。
- biginttohex 関数は、整数から 16 進文字列へのプラットフォーム独立型の変換に使用します。biginttohex は、bigint に評価される式をすべて受け入れます。実行するプラットフォームに関係なく、指定の式と同じ 16 進数の値を常に返します。

参照

関数 convert, hextobigint, hextoint, inttohex

bintostr

説明　一連の 16 進数を同等の英数字または varbinary データの文字列に変換します。

構文　select bintostr(*sequence of hexadecimal digits*)

パラメータ

sequence of hexadecimal digits

有効な 16 進数の数列で、[0 - 9]、[a - f]、および [A - F] で構成され、プレフィクス「0x」が付きます。

例

例 1 16 進数列「0x723ad82fe」を同じ値の英数字文字列に変換します。

```
1> select bintostr(0x723ad82fe)
2> go

-----------------------------------
0723ad82fe
```

この例では、16 進の数列とそれに相当する英数字文字列のメモリ内表現は次のようになります。

16 進数 (5 バイト)

0	7	2	3	a	d	8	2	f	e

英数字文字列 (9 バイト)

0	7	2	3	a	d	8	2	f	e

この関数は 16 進数を右から左へと処理します。この例では、入力の桁数は奇数です。そのため、英数字文字列に「0」のプレフィクスが付いて出力に反映されます。

例 2 *@bin_data* というローカル変数の 16 進数を「723ad82fe」の値に相当する英数字文字列に変換します。

```
declare @bin_data varchar(30)
select @bin_data = 0x723ad82fe
select bintostr(@bin_data)
go

----------
0723ad82fe
```

使用法

- 入力の無効な文字は、出力として null になります。
- 入力は有効な varbinary データである必要があります。
- NULL の入力結果は NULL の出力になります。

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　bintostr は、すべてのユーザが実行できます。

参照　**関数**　strtobin

cache_usage

説明

キャッシュの使用率を、テーブルが属するキャッシュのすべてのオブジェクトの割合として返します。

構文

cache_usage(*table_name*)

パラメータ

table_name

テーブルの名前です。完全修飾名を指定できます (つまり、データベースと所有者名が含まれている名前を指定できます)。

例

例 1 titles テーブルが使用しているキャッシュの割合を返します。

```
select cache_usage("titles")
----------------------------
                    98.876953
```

例 2 authors テーブルが使用しているキャッシュの割合を master データベースから返します。

```
select cache_usage ("pubs2..authors")
-------------------------------------
                                98.876953
```

使用法

- cache_usage は、キャッシュの使用状況をキャッシュのすべてのプールのパーセンテージとして提供します。
- cache_usage は、現在のオブジェクトが使用しているキャッシュの容量についての情報は提供しません。また、別のキャッシュにバウンドされている場合は、インデックスのキャッシュ使用状況についての情報も提供しません。
- クラスタ環境では、cache_usage は、オブジェクトが現在のノードでバウンドされているキャッシュのキャッシュ使用状況を提供します。

パーミッション

cache_usage は、すべてのユーザが実行できます。

case

説明

条件付き SQL 文をサポートします。value 式を使用できる場所であればどこでも使用できます。

構文

```
case
        when search_condition then expression
        [when search_condition then expression]...
        [else expression]
end
```

case と値の構文

```
case expression
        when expression then expression
        [when expression then expression]...
        [else expression]
end
```

パラメータ

case

case 式を開始します。

when

探索条件または比較する式の前に置きます。

search_condition

選択される結果に対する条件を設定するために使用します。case 式の検索条件は、where 句の中の検索条件と似ています。探索条件の詳細については、『Transact-SQL ユーザーズ・ガイド』を参照してください。

then

case の結果値を指定する式の前に置きます。

expression

カラム名、定数、関数、サブクエリ、またはこれらを算術演算子やビット処理演算子でつないだ任意の組み合わせです。式の詳細については、「式」(341 ページ) を参照してください。

例

例 1 authors テーブルからすべての作者を選択し、特定の作者に関しては居住都市を指定します。

```
select au_lname, postalcode,
    case
            when postalcode = "94705"
                then "Berkeley Author"
            when postalcode = "94609"
                then "Oakland Author"
            when postalcode = "94612"
                then "Oakland Author"
            when postalcode = "97330"
                then "Corvallis Author"
        end
from authors
```

例 2 discounts テーブルの lowqty カラムまたは highqty カラムのどちらかにある最初の NULL 以外の値を返します。

```
select stor_id, discount,
        coalesce (lowqty, highqty)
from discounts
```

また、次のフォーマットを使用して同じ結果を得ることができます。これは、coalesce が case 式の省略形だからです。

```
select stor_id, discount,
        case
            when lowqty is not NULL then lowqty
            else highqty
        end
from discounts
```

例 3 *titles* テーブルから *titles* と type を選択します。本のタイプが UNDECIDED の場合、nullif は NULL 値を返します。

```
select title,
        nullif(type, "UNDECIDED")
  from titles
```

また、次のフォーマットを使用して同じ結果を得ることができます。これは、nullif が case 式の省略形だからです。

```
select title,
        case
            when type = "UNDECIDED" then NULL
            else type
        end
  from titles
```

例 4 ではエラー・メッセージが表示されます。少なくとも 1 つの式が、null キーワード以外である必要があるからです。

```
select price, coalesce (NULL, NULL, NULL)
from titles

All result expressions in a CASE expression must not be NULL.
```

例 5 エラー・メッセージが表示されます。coalesce の後ろには、少なくとも 2 つの式を配置する必要があるからです。

```
select stor_id, discount, coalesce (highqty) from discounts

A single coalesce element is illegal in a COALESCE expression.
```

使用法

- case 式は、if 文の代わりに when...then 構造を使用して探索条件を表すことによって、標準 SQL 式を簡略化したものです。
- case 式は、SQL で式が使用できる場所であればどこででも使用できます。
- クエリがさまざまなデータ型を作成する場合は、「混合モードの式のデータ型」(6 ページ) で説明されているとおり、データ型の階層によって case 式の結果のデータ型が決定されます。Adaptive Server で、暗黙的に変換されないデータ型 (char や int など) を 2 つ指定した場合、クエリは正常に動作しません。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

case パーミッションは、すべてのユーザに対してデフォルトで設定されています。これを使用するためのパーミッションは必要ありません。

参照

コマンド　coalesce, nullif, if...else, select, where 句

cast

説明

指定された値を別のデータ型に変換して返します。cast は、変換元の式の null 入力可能性を変更したり、デフォルトのフォーマットを date データ型および time データ型に使用したりできます。

構文

cast (*expression* as *datatype* [(*length* | *precision*[, *scale*])])

パラメータ

expression

あるデータ型や日付フォーマットから別のデータ型や日付フォーマットに変換される値です。カラム、定数、関数、定数の任意の組み合わせ、および算術演算子やビット処理演算子またはサブクエリで連結された関数が含まれます。

データベースで Java が実行可能な場合、*expression* には、Java-SQL クラスに変換される値を指定できます。

変換先データ型として unichar が使用されており、長さが指定されていない場合には、デフォルトの長さである 30 Unicode 値が使用されます。

length

char、nchar、unichar、univarchar、varchar、nvarchar、binary、varbinary データ型ともに使用されるオプションのパラメータです。この引数を指定しないと、Adaptive Server は文字型のデータを 30 文字にトランケートし、バイナリ型のデータを 30 バイトにトランケートします。文字型データおよびバイナリ型データの最大長は 64K です。

precision

データ型が numeric または decimal のときの有効桁数です。float データ型のときは仮数内の有効バイナリ桁数です。numeric データ型および decimal のデータ型の場合、この引数を指定しないと、Adaptive Server はデフォルトの精度 18 を使います。

scale

データ型が numeric や decimal の場合の、小数点の右側の桁数です。この引数を指定しないと、Adaptive Server はデフォルトの位取り 0 を使います。

例

例 1 日付を読みやすい datetime フォーマットに変換します。

```
select cast("01/03/63" as datetime)
go

 --------------------------
        Jan  3 1963 12:00AM

(1 row affected)
```

例 2 title データベース内の total_sales カラムを 12 文字のカラムに変換します。

```
select title, cast(total_sales as char(12))
```

使用法

- データ型変換の詳細については、「データ型変換関数」(57 ページ) を参照してください。
- cast 関数の引数が、この関数に定義されている範囲の外にあるときは、ドメイン・エラーになります。このエラーが起こることはほとんどありません。
- ターゲット・カラムが null 入力可能かどうかを指定するには、null または not null を使用してください。select into で null または not null を使用して新しいテーブルを作成し、ソース・テーブル内の既存カラムのデータ型と null 入力可能性を変更することができます。
- cast 関数を使用して、image カラムを binary または varbinary に変換できます。変換は、サーバの論理ページ・サイズの最大カラム・サイズによって決定される binary データ型の最大長に制限されます。長さを指定しない場合、変換された値はデフォルトの長さである 30 文字になります。
- unichar 式を変換先データ型として使用できます。また、この式を別のデータ型へ変換できます。unichar 式と、サーバで使用できる任意のデータ型との間で、明示的または暗黙的にデータ型を変換できます。
- unichar が変換先の型として指定されているときに長さを指定しないと、デフォルトの長さである 30 Unicode 値が使用されます。変換先の型の長さが、指定された式を格納するのに十分でない場合は、エラー・メッセージが表示されます。

暗黙的変換

プライマリ・フィールドが一致しない場合、データ型の暗黙的な変換が行われると、データのトランケーション、デフォルト値の挿入、またはエラー・メッセージの発生のいずれかが生じる場合があります。たとえば、datetime 値を date 値に変換すると、時刻部分がトランケートされ、日付部分のみが残ります。time 値を datetime 値に変換すると、新しい datetime 値の日付部分に、デフォルト日付の Jan 1, 1900 が追加されます。date 値を datetime 値に変換すると、datetime 値の時刻部分に、デフォルト時刻の 00:00:00:000 が追加されます。

```
DATE -> VARCHAR, CHAR, BINARY, VARBINARY, DATETIME, SMALLDATETIME
TIME -> VARCHAR, CHAR, BINARY, VARBINARY, DATETIME, SMALLDATETIME
VARCHAR, CHAR, BINARY, VARBINARY, DATETIME, SMALLDATETIME -> DATE
VARCHAR, CHAR, BINARY, VARBINARY, DATETIME, SMALLDATETIME  -> TIME
```

明示的変換

date を datetime に明示的に変換しようとしたとき、その値が datetime の範囲外 (たとえば、笛 an 1, 1000・の場合は、変換が許可されず、それを知らせるエラー・メッセージが表示されます。

```
DATE -> UNICHAR, UNIVARCHAR
TIME -> UNICHAR, UNIVARCHAR
UNICHAR, UNIVARCHAR -> DATE
UNICHAR, UNIVARCHAR -> TIME
```

Java クラスに関連する変換

- データベースで Java が実行可能な場合は、cast を使用して、次の方法でデータ型を変更できます。
  - Java オブジェクト・タイプを SQL データ型に変換する。
  - SQL データ型を Java タイプに変換する。
  - 式 (ソース・クラス) のコンパイル時のデータ型がターゲット・クラスのサブクラスまたはスーパークラスである場合、Adaptive Server にインストールされた任意の Java-SQL クラスを、Adaptive Server にインストールされた任意の他の Java-SQL クラスに変換する。

  変換の結果は、現在のデータベースと関連付けられます。

規格

ANSI SQL － 準拠レベル：ANSI 準拠

パーミッション

すべてのユーザが cast 関数を実行できます。

ceiling

説明　指定された値以上の最小の整数を返します。

構文　ceiling(*value*)

パラメータ　*value*

カラム、変数、またはデータ型が真数値、概数値、通貨、またはこれらの型の 1 つに暗黙的に変換できる式です。

例

例 1 124 の値を返します。

```
select ceiling(123.45)

124
```

例 2 -123 の値を返します。

```
select ceiling(-123.45)

-123
```

例 3 24.000000 の値を返します。

```
select ceiling(1.2345E2)

24.000000
```

例 4 -123.000000 の値を返します。

```
select ceiling(-1.2345E2)

-123.000000
```

例 5 124.00 の値を返します。

```
select ceiling($123.45)

124.00
```

例 6 title_id が値「PS3333」の salesdetail テーブルから「discount」の値を返します。

```
select discount, ceiling(discount) from salesdetail where
title_id = "PS3333"

discount
 -------------------- --------------------
            45.000000            45.000000
            46.700000            47.000000
            46.700000            47.000000
            50.000000            50.000000
```

使用法

- 算術関数 ceiling は、指定した値以上の最小整数値を返します。戻り値のデータ型は、指定した値のデータ型と同じです。

  numeric 値や decimal 値の場合は、結果は指定した値と同じ精度で、位取りは 0 です。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが ceiling 関数を実行できます。

参照

算術関数の概要については、「算術関数」(67 ページ) を参照してください。

コマンド set

関数 abs, floor, round, sign

char

説明　整数に相当する文字を返します。

構文　char(*integer_expr*)

パラメータ　*integer_expr*

整数値型 (tinyint、smallint、または int) のカラム名、変数、0 ～ 255 の定数式のいずれかです。

例

例 1

```
select char(42)

-
*
```

例 2

```
select xxx = char(65)

xxx
---
A
```

使用法

- char 文字列関数は、シングルバイト整数値を文字値に変換します (char は通常 ascii の逆変換です)。
- char は char データ型を返します。結果の値がマルチバイト文字の最初のバイトの場合、その文字は未定義であることがあります。
- *char_expr* が NULL の場合は、NULL を返します。

char を使った出力フォーマットの変更

- 連結値や char 値を使用してタブや復帰改行文字を追加し、出力フォーマットを変更できます。char(10) は復帰改行文字に、char(9) はタブに変換されます。たとえば、次のようにします。

```
/* just a space */
select title_id +  " " + title from titles where title_id = "T67061"
/* a return */
select title_id + char(10) + title from titles where title_id = "T67061"
/* a tab */
select title_id + char(9) + title from titles where title_id = "T67061"
------------------------------------------------------------------------
T67061 Programming with Curses
------------------------------------------------------------------------
T67061

Programming with Curses
------------------------------------------------------------------------
T67061      Programming with Curses
```

規格	ANSI SQL － 準拠レベル：Transact-SQL 拡張機能
パーミッション	すべてのユーザが char 関数を実行できます。
参照	文字列関数の概要については、「文字列関数」(68 ページ) を参照してください。 **関数** ascii, str

char_length

説明　式の文字数を返します。

構文　char_length(*char_expr* | *uchar_expr*)

パラメータ

char_expr

文字型のカラム名、変数、または (char、varchar、nchar、または nvarchar 型の) 定数式です。

uchar_expr

文字型のカラム名、変数、または (unichar または univarchar 型の) 定数式です。

例

例 1

```
select char_length(notes) from titles
where title_id = "PC9999"

 -----------
          39
```

例 2

```
declare @var1 varchar(20), @var2 varchar(20), @char char(20)
select @var1 = "abcd", @var2 = "abcd    ", @char = "abcd"
select char_length(@var1), char_length(@var2), char_length(@char)

 ----------- ----------- -----------
           4           8          20
```

使用法

- char_length 文字列関数は、文字式やテキスト値の中の文字数を表す整数を返します。
- 可変長のカラムおよび変数の場合、char_length は (カラムまたは変数の定義された長さではなく) 文字数を返します。明示的な後続ブランクが可変長変数に含まれている場合、それらは取り除かれません。リテラルや固定長カラムまたは変数の場合、char_length は式から後続ブランクを削除しません (例 2 参照)。
- unitext、unichar、univarchar カラムの場合、char_length は Unicode 値 (16 ビット) の数を返します。このとき、1 つのサロゲート・ペアは 2 つの Unicode 値としてカウントされます。たとえば、unitext のカラム ut に U+0041U+0042U+d800dc00 というロー値が保管されている場合、次の値が返されます。

```
select char_length(ut) from unitable
------------
4
```

- マルチバイト文字セットの場合、式の文字数は、通常、バイト数より少なくなります。バイト数を指定するには、datalength 関数を使用します。
- Unicode 式の場合、式の (バイト数ではなく) Unicode 値が返されます。サロゲート・ペアは 2 つの Unicode 値としてカウントされます。
- *char_expr* または *uchar_expr* が NULL の場合は、char_length は NULL を返します。
- 文字列関数の一般的な説明については、「文字列関数」(68 ページ) を参照してください。

規格 ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション すべてのユーザが char_length 関数を実行できます。

参照 **関数** datalength

charindex

説明　式の開始位置を表す整数を返します。

構文　charindex(*expression1*, *expression2*)

パラメータ

expression

バイナリ型または文字型のカラム名、変数、または定数式です。char、varchar、nchar、nvarchar、unichar、univarchar、binary、または varbinary が使用できます。

例　titles テーブルの notes カラムにある文字式「wonderful」の開始位置を返します。

```
select charindex("wonderful", notes)
  from titles
where title_id = "TC3218"

-----------
         46
```

使用法

- 文字列関数 charindex は、*expression1* が最初に出現したときに *expression2* を検索し、その開始位置を示す整数を返します。*expression1* が見つからない場合、charindex は 0 を返します。
- *expression1* にワイルドカード文字が含まれている場合、charindex はこれらのワイルドカード文字をリテラルとして扱います。
- *expression2* が NULL の場合は、0 を返します。
- varchar 式と unichar 式がそれぞれパラメータとして指定されている場合には、varchar 式は暗黙的に unichar へ変換されます (変換時にトランケーションが発生することがあります)。

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　すべてのユーザが charindex 関数を実行できます。

参照　文字列関数の概要については、「文字列関数」(68 ページ) を参照してください。

関数　patindex

coalesce

説明

条件付き SQL 式をサポートします。値式が使用できる場所であればどこででも使用できます。case 式の代わりに使用できます。

構文

coalesce(*expression*, *expression* [, *expression*]...)

パラメータ

coalesce

リストされた式を評価し、最初の null 以外の値を返します。すべての式が null 値の場合、coalesce は NULL 値を返します。

expression

カラム名、定数、関数、サブクエリ、またはこれらを算術演算子やビット処理演算子でつないだ任意の組み合わせです。式の詳細については、「式」(341 ページ) を参照してください。

例

例 1 discounts テーブルの lowqty カラムまたは highqty カラムのどちらかにある最初の null 以外の値を返します。

```
select stor_id, discount,
        coalesce (lowqty, highqty)
from discounts
```

例 2 例 1 の別の表記です。

```
select stor_id, discount,
        case
            when lowqty is not NULL then lowqty
            else highqty
        end
from discounts
```

使用法

- coalesce 式は、when...then 構造を使用する代わりに、簡単な探索条件を表すことによって、標準 SQL 式を簡略化したものです。
- coalesce 式は、SQL 内で式が使用できる場所であればどこででも使用できます。
- coalesce 式の 1 つ以上の結果が、null 以外の値を返す必要があります。次の例では、以下のエラー・メッセージが表示されます。

```
select price, coalesce (NULL, NULL, NULL)
from titles

All result expressions in a CASE expression must not be NULL.
```

- クエリがさまざまなデータ型を作成する場合は、「混合モードの式のデータ型」(6 ページ) で説明されているとおり、データ型の階層によって case 式の結果のデータ型が決定されます。Adaptive Server で暗黙的に変換されないデータ型 (char や int など) を 2 つ指定した場合、クエリは正常に動作しません。
- coalesce は、case 式を省略した形です。例 2 は、coalesce 文の別の記述方法を示します。
- coalesce の後ろには、少なくとも 2 つの式が必要です。次の例では、以下のエラー・メッセージが表示されます。

```
select stor_id, discount, coalesce (highqty)
from discounts

A single coalesce element is illegal in a COALESCE expression.
```

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　coalesce は、すべてのユーザが実行できます。

参照　**コマンド**　case, nullif, select, if...else, where 句

col_length

説明

カラムの定義済みの長さを返します。

構文

col_length(*object_name*, *column_name*)

パラメータ

object_name

データベース・オブジェクト (テーブル、ビュー、プロシージャ、トリガ、デフォルト、またはルールなど) の名前です。完全修飾名を指定できます (つまり、データベースと所有者名が含まれている名前を指定できます)。名前は引用符で囲んでください。

column_name

カラムの名前です。

例

titles テーブル内の title カラムの長さを求めます。「x」は結果のカラム見出しです。

```
select x = col_length("titles", "title")

 x
----
   80
```

使用法

- col_length システム関数は、カラムの定義済みの長さを返します。
- システム関数の概要については、「システム関数」(69 ページ) を参照してください。
- 各ローに格納されているデータの実際の長さを求めるには、datalength を使用します。
- text、unitext、image カラムの場合は、col_length は 16 を返します。これは、実際のテキスト・ページを示す binary(16) ポインタの長さです。
- unichar カラムの場合、定義されている長さは、(表されるバイトの数ではなく) カラムの定義時に宣言されている Unicode 値の数です。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが col_length 関数を実行できます。

参照

関数 datalength

col_name

説明

指定されたテーブルとカラム ID に対応するカラム名を返します。カラム名は、最長で 255 バイトです。

構文

col_name(*object_id*, *column_id* [, *database_id*])

パラメータ

object_id

テーブル、ビュー、またはその他のデータベース・オブジェクトのオブジェクト ID を表す数値式です。この ID は sysobjects の id カラムに格納されています。

column_id

カラムのカラム ID である数値式です。この式は syscolumns の colid カラムに格納されています。

database_id

データベースの ID である数値式です。この ID は sysdatabases の db_id カラムに格納されています。

例

```
select col_name(208003772, 2)

------------------------------
title
```

使用法

- システム関数 col_name は、カラムの名前を返します。
- システム関数の概要については、「システム関数」(69 ページ) を参照してください。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが col_name 関数を実行できます。

参照

関数　db_id, object_id

compare

説明

代替照合規則をもとに、2 つの文字列を直接比較できるようにします。

構文

compare ({*char_expression1*|*uchar_expression1*},
{*char_expression2*|*uchar_expression2*}),
[{*collation_name* | *collation_ID*}]

パラメータ

char_expression1 または *uchar_expression1*
char_expression2 または *uchar_expression 2* と比較する文字式です。

char_expression2 または *uchar_expression2*
char_expression1 または *uchar_expression1* と比較する文字式です。

char_expression1 および *char_expression2* は、次のいずれかです。

- 文字型 (char、varchar、nchar、または nvarchar)
- 文字変数
- 文字定数の式 (一重引用符または二重引用符で囲む)

uchar_expression1 および *uchar_expression2* は、次のいずれかです。

- 文字型 (unichar または univarchar)
- 文字変数
- 文字定数の式 (一重引用符または二重引用符で囲む)

collation_name
引用符で囲まれた文字列、または使用する照合動作を指定する文字変数です。表 2-6 (107 ページ) に、有効値を示します。

collation_ID
使用する照合動作を指定する整数定数または変数です。表 2-6 (107 ページ) に、有効値を示します。

例

例 1 aaa と bbb を比較します。

```
1> select compare ("aaa","bbb")
2> go

 -----------
          -1
(1 row affected)
```

または、次のようなフォーマットでも、aaa と bbb を比較できます。

```
1> select compare (("aaa"),("bbb"))
2> go

 -----------
          -1
(1 row affected)
```

例 2 aaa と bbb を比較し、バイナリ・ソート順を指定します。

```
1> select compare ("aaa","bbb","binary")
2> go

 -----------
          -1
(1 row affected)
```

または、次のようなフォーマットで aaa と bbb を比較し、照合名の代わりに照合 ID を使用することもできます。

```
1> select compare (("aaa"),("bbb"),(50))
2> go

 -----------
          -1
(1 row affected)
```

使用法

- compare 関数は、選択した照合規則をもとに次の値を返します。
  - 1 － *char_expression1* または *uchar_expression1* が *char_expression2* または *uchar_expression2* より大きいことを示します。
  - 0 － *char_expression1* または *uchar_expression1* が *char_expression2* または *uchar_expression2* と等しいことを示します。
  - -1 － *char_expression1* または *uchar_expression1* が *char_expression2* または *uchar expression2* より小さいことを示します。
- compare は、それぞれの入力文字について 6 バイトの照合情報を生成します。したがって、compare の実行結果は、varbinary データ型の長さの制限値を超える可能性があります。この制限値を超える場合、結果は制限値に収まるようにトランケートされます。Adaptive Server からは警告メッセージが表示されますが、compare 関数を含むクエリまたはトランザクションは引き続き実行されます。この制限はサーバの論理ページ・サイズに依存するので、DOL テーブルおよび APL テーブルでは、結果の文字列が次のサイズより小さくなるまで、各入力文字の結果バイトはトランケーションによって削除されます。

表 2-5: ローとカラムの最大長 – *APL* および *DOL*

ロック・スキーム	ページ・サイズ	ローの最大長	カラムの最大長
APL テーブル	2K (2048 バイト)	1962	1960 バイト
	4K (4096 バイト)	4010	4008 バイト
	8K (8192 バイト)	8106	8104 バイト
	16K (16384 バイト)	16298	16296 バイト
DOL テーブル	2K (2048 バイト)	1964	1958 バイト
	4K (4096 バイト)	4012	4006 バイト
	8K (8192 バイト)	8108	8102 バイト
	16K (16384 バイト)	16300	16294 バイト – テーブルに可変長カラムがない場合
	16K (16384 バイト)	16300 (varlen = 8191 の最大開始オフセットによって変化)	8191-6-2 = 8183 バイト – 1 つ以上の可変長カラムがテーブルにある場合 *

* このサイズには、ローのオーバヘッドの 6 バイトとローの長さのフィールドの 2 バイトが含まれる。

- *char_expression1* と *uchar_expression1*、および *char_expression2* と *uchar_expression2* は、サーバのデフォルト文字セットでエンコードされている文字である必要があります。
- *char_expression1* と *char_expression2*、または *uchar_expression 1* と *uchar_expression2* では、いずれかまたは両方に空文字列を指定できます。
  - *char_expression2* または *uchar_expression2* が空文字列の場合は、1 が返されます。
  - 両方とも空文字列の場合は、等しいとみなされ、0 が返されます。
  - *char_expression1* または *uchar_expression 1* が空文字列の場合は、-1 が返されます。

  compare 関数では、空の文字列とスペースだけの文字列は等しいとみなされません。compare は、sortkey 関数を使用して、比較のための照合キーを生成します。したがって、まったく空の文字列、1 つのスペースからなる文字列、または 2 つのスペースからなる文字列は、等しいとはみなされません。
- *char_expression1* または *uchar_expression1*、あるいは *char_expression2* または *uchar_expression2* が NULL の場合、結果は NULL になります。
- varchar 式と unichar 式がそれぞれパラメータとして指定されている場合には、varchar 式は暗黙的に unichar へ変換されます (変換時にトランケーションが発生することがあります)。
- *collation_name* または *collation_ID* に値を指定しない場合、compare はバイナリ照合とみなします。

- 表 2-6 は、*collation_name* と *collation_ID* の有効値のリストを示します。

表 *2-6:* 照合名と *ID*

説明	照合名	照合 ID
デフォルト Unicode マルチ言語	default	20
タイ語辞書順	thaidict	21
ISO14651 規格	iso14651	22
UTF-16 順 − UTF-8 バイナリ順と一致	utf8bin	24
CP 850 代替言語 − アクセント記号なし	altnoacc	39
CP 850 代替言語、小文字優先	altdict	45
CP 850 西欧言語 − 大文字／小文字の優先設定なし	altnocsp	46
CP 850 スカンジナビア語 − 辞書順	scandict	47
CP 850 スカンジナビア語辞書順 − 大文字と小文字を区別しない、優先度を付けた順位	scannocp	48
GB ピンイン	gbpinyin	該当なし
バイナリ・ソート	binary	50
Latin-1 英語、フランス語、ドイツ語辞書	dict	51
Latin-1 英語、フランス語、ドイツ語、大文字／小文字の区別なし	nocase	52
Latin-1 英語、フランス語、ドイツ語、大文字／小文字の優先設定なし	nocasep	53
Latin-1 英語、フランス語、ドイツ語、アクセント記号なし	noaccent	54
Latin-1 スペイン語辞書	espdict	55
Latin-1 スペイン語、大文字／小文字の区別なし	espnocs	56
Latin-1 スペイン語、アクセント記号なし	espnoac	57
ISO 8859-5 ロシア語辞書	rusdict	58
ISO 8859-5 ロシア語、大文字小文字の区別なし	rusnocs	59
ISO 8859-5 キリル語辞書	cyrdict	63
ISO 8859-5 キリル語、大文字小文字の区別なし	cyrnocs	64
ISO 8859-7 ギリシア語辞書	elldict	65
ISO 8859-2 ハンガリー語、辞書	hundict	69
ISO 8859-2 ハンガリー語、アクセント記号の区別なし	hunnoac	70
ISO 8859-2 ハンガリー語、大文字小文字の区別なし	hunnocs	71
ISO 8859-9 トルコ語辞書	turdict	72
ISO 8859-9 トルコ語、アクセント記号の区別なし	turknoac	73
ISO 8859-9 トルコ語、大文字小文字の区別なし	turknocs	74
CP932 バイナリ順	cp932bin	129
中国語、読み方順	dynix	130
GB2312 バイナリ順	gb2312bn	137
共通キリル語辞書	cyrdict	140
トルコ語辞書	turdict	155
EUCKSC バイナリ順	euckscbn	161

説明	照合名	照合 ID
中国語、読み方順	gbpinyin	163
ロシア語辞書順	rusdict	165
SJIS バイナリ順	sjisbin	179
EUCJIS バイナリ順	eucjisbn	192
BIG5 バイナリ順	big5bin	194
Shift-JIS バイナリ順	sjisbin	259

規格 ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション すべてのユーザが compare 関数を実行できます。

参照 **関数** sortkey

convert

説明

指定した値を別のデータ型または異なる datetime 表示フォーマットに変換して返します。

構文

convert (*datatype* [(*length*) | (*precision*[, *scale*])]
[null | not null], *expression* [, *style*])

パラメータ

datatype

システムが提供するデータ型 (char(10)、unichar (10)、unichar (10)、または int など) です。式はこのデータ型に変換されます。ユーザ定義データ型は使えません。

データベースで Java が実行可能な場合、*datatype* には、現在のデータベースの Java-SQL クラスも指定できます。

length

char、nchar、unichar、univarchar、varchar、nvarchar、binary、varbinary データ型とともに使用されるオプションのパラメータです。この引数を指定しないと、Adaptive Server は文字型のデータを 30 文字にトランケートし、バイナリ型のデータを 30 バイトにトランケートします。文字型データおよびバイナリ型データの最大長は 64K です。

precision

データ型が numeric または decimal のときの有効桁数です。float データ型のときは仮数内の有効バイナリ桁数です。numeric データ型および decimal のデータ型の場合、この引数を指定しないと、Adaptive Server はデフォルトの精度 18 を使います。

scale

データ型が numeric や decimal の場合の、小数点の右側の桁数です。この引数を指定しないと、Adaptive Server はデフォルトの位取り 0 を使います。

null | not null

結果式の null 入力可能性を指定します。null または not null のどちらも指定しない場合、変換された結果の null 入力可能性は、式と同じです。

expression

あるデータ型や日付フォーマットから別のデータ型や日付フォーマットに変換される値です。

データベースで Java が実行可能な場合、*expression* には、Java-SQL クラスに変換される値を指定できます。

変換先データ型として unichar が使用されており、長さが指定されていない場合には、デフォルトの長さである 30 Unicode 値が使用されます。

style

変換後のデータに使用する表示形式です。money データや smallmoney データを文字型に変換するときは、*style* を 1 にして 3 桁ごとにカンマを表示します。

datetime データや smalldatetime データを文字型に変換するときは、表 2-7 に示すスタイル番号を使って表示フォーマットを指定します。左端のカラムの値は 2 桁の年 (*yy*) を表します。4 桁の年 (*yyyy*) については、この左端のカラムの値に 100 を加えるか、または左から 2 番目のカラムにある値を使用します。

date データを文字型に変換するときは、表 2-7 に示すスタイル番号の 1 ～ 7 (101 ～ 107) または 10 ～ 12 (110 ～ 112) を使って表示フォーマットを指定します。デフォルト値は 100 (mon dd yyyy hh:miAM (または PM)) です。時刻部分を含むスタイルに date データを変換する場合は、時刻部分はデフォルト値の 0 になります。

time データを文字型に変換するときは、スタイル番号の 8 または 9 (108 または 109) を使って表示フォーマットを指定します。デフォルトは 100 (mon dd yyyy hh:miAM (または PM)) です。日付部分を含むスタイルに time データを変換する場合は、デフォルト日付の Jan 1, 1900 が表示されます。

表 2-7: *style* パラメータを使用した日付フォーマットの変換

世紀なし (yy)	世紀あり (yyyy)	規格	出力
-	0 または 100	デフォルト	*mon dd yyyy hh:mm* AM (または PM)
1	101	USA	*mm/dd/yy*
2	2	SQL 規格	*yy.mm.dd*
3	103	イギリス／フランス	*dd/mm/yy*
4	104	ドイツ	*dd.mm.yy*
5	105		*dd-mm-yy*
6	106		*dd mon yy*
7	107		*mon dd, yy*
8	108		*HH:mm:ss*
-	9 または 109	デフォルト + ミリ秒	*mon dd yyyy hh:mm:ss* AM (または PM)
10	110	USA	*mm-dd-yy*
11	111	日本	*yy/mm/dd*
12	112	ISO	*yymmdd*
13	113		*dd/yy/mm*
14	114		*mm/yy/dd*
14	114		*hh:mi:ss:mmm*AM (または PM)
15	115		*dd/yy/mm*
-	16 または 116		*mon dd yyyy HH:mm:ss*
17	117		*hh:mmAM*

キー “mon” はスペルアウトした月名、“mm” は数字の月または分です。“HH” は 24 時間形式の時刻、“hh” は 12 時間形式の時刻です。最後のロー 23 には、フォーマット内で日付と時刻を区切るリテラル “T” が含まれます。

世紀なし (yy)	世紀あり (yyyy)　規格	出力
18	118	*HH:mm*
19	119	*hh:mm:ss:zzzAM*
20	120	*hh:mm:ss:zzz*
21	121	*yy/mm/dd HH:mm:ss*
22	122	*yy/mm/dd HH:mm* AM (または PM)
23	123	*yyyy-mm-ddTHH:mm:ss*

キー　“mon” はスペルアウトした月名、“mm” は数字の月または分です。“HH” は 24 時間形式の時刻、“hh” は 12 時間形式の時刻です。最後のロー 23 には、フォーマット内で日付と時刻を区切るリテラル “T” が含まれます。

デフォルト値 (*style* が 0 か 100) のときと、*style* が 9 か 109 のときは、常に 4 桁表示 (*yyyy*) が返されます。smalldatetime から char や varchar に変換すると、秒やミリ秒を含むスタイルでは秒やミリ秒の位置に 0 が表示されます。

例

例 1

```
select title, convert(char(12), total_sales)
  from titles
```

例 2

```
select title, total_sales
from titles
where convert(char(20), total_sales) like "1%"
```

例 3 現在の日付をスタイル 3 の dd/mm/yy に変換します。

```
select convert(char(12), getdate(), 3)
```

例 4 pubdate の値が null になる可能性がある場合は、char ではなく varchar を使用してください。そうしないと、エラーが発生することがあります。

```
select convert(varchar(12), pubdate, 3) from titles
```

例 5 文字列「0x00000100」に相当する整数を返します。結果はプラットフォームにより異なることがあります。

```
select convert(integer, 0x00000100)
```

例 6 プラットフォーム固有のビット・パターンを Sybase のバイナリ型として返します。

```
select convert (binary, 10)
```

例 7 1.11 ドルに相当するビット文字列 1 を返します。

```
select convert(bit, $1.11)
```

例 8 データ型 char(100) の total_sales を持つ #tempsales を作成し、null 値は使用できないようにします。titles.total_sales が null を使用できるように定義されている場合でも、作成される #tempsales には、null 値が許可されない #tempsales.total_sales があります。

```
select title, convert (char(100) not null, total_sales) into
#tempsales
from titles
```

使用法

- データ型変換関数 convert は各種データ型を変換し、表示のための日付／時刻データや通貨データのフォーマットを変更します。
- データ型変換の詳細については、「データ型変換関数」(57 ページ) を参照してください。
- convert － 指定した値を別のデータ型または異なる日付表示フォーマットに変換して返します。unitext データを他の文字やバイナリのデータ型に変換する場合、変換結果は変換先のデータ型の最大長に制約されます長さを指定しない場合、変換された値はデフォルト・サイズである 30 バイトになります。enable surrogate processing を使用している場合、サロゲート・ペアは全体が返されます。たとえば、U+0041U+0042U+20acU+0043 (“AB €” を表す) というデータが保管されている unitext カラムを UTF-8 varchar(3) カラムに変換した場合、次の値が返されます。

  ```
  select convert(varchar(3), ut) from untable
  ---
  AB
  ```

- convert 関数の引数が、この関数に定義されている範囲の外にある場合は、ドメイン・エラーが生成されます。このエラーが起こることはほとんどありません。
- ターゲット・カラムが null 入力可能かどうかを指定するには、null または not null を使用してください。これは特に、select into とともに使用すると、新しいテーブルを作成し、ソース・テーブル内の既存カラムのデータ型と null 入力可能性を変更することができます (前述の例 8 を参照)。

  次の場合に、結果は未定義の値です。

  - 変換される式の結果が not null になります。
  - 式の値が null になる。

  予測可能な結果を表す NULL 以外の既知の値を生成するには、次の select 文を使用します。

```
select convert(int not null, isnull(col2, 5)) from table1
```

- convert 関数を使用して、image カラムを binary または varbinary に変換できます。binary データ型の最大長は制限されています。この最大長は、サーバの論理ページ・サイズの最大カラム・サイズによって決定します。長さを指定しない場合、変換された値はデフォルトの長さである 30 文字になります。
- unichar 式を変換先データ型として使用できます。また、この式を別のデータ型へ変換できます。unichar 式と、サーバで使用できる任意のデータ型との間で、明示的または暗黙的にデータ型を変換できます。

- unichar が変換先の型として指定されているときに長さを指定しないと、デフォルトの長さである 30 Unicode 値が使用されます。変換先の型の長さが、指定された式を格納するのに十分でない場合は、エラー・メッセージが表示されます。

暗黙的変換

プライマリ・フィールドが一致しない場合、データ型の暗黙的な変換が行われると、データのトランケーション、デフォルト値の挿入、またはエラー・メッセージの発生のいずれかが生じる場合があります。たとえば、datetime 値を date 値に変換すると、時刻部分がトランケートされ、日付部分のみが残ります。time 値を datetime 値に変換すると、新しい datetime 値の日付部分に、デフォルト日付の Jan 1, 1900 が追加されます。date 値を datetime 値に変換すると、datetime 値の時刻部分に、デフォルト時刻の 00:00:00:000 が追加されます。

```
DATE -> VARCHAR, CHAR, BINARY, VARBINARY, DATETIME, SMALLDATETIME
TIME -> VARCHAR, CHAR, BINARY, VARBINARY, DATETIME, SMALLDATETIME
VARCHAR, CHAR, BINARY, VARBINARY, DATETIME, SMALLDATETIME -> DATE
VARCHAR, CHAR, BINARY, VARBINARY, DATETIME, SMALLDATETIME  -> TIME
```

明示的変換

date を datetime に明示的に変換しようとしたときに、その値が datetime の範囲外 (たとえば「Jan 1, 1000」) の場合は、変換が許可されず、そのことを知らせるエラー・メッセージが表示されます。

```
DATE -> UNICHAR, UNIVARCHAR
TIME -> UNICHAR, UNIVARCHAR
UNICHAR, UNIVARCHAR -> DATE
UNICHAR, UNIVARCHAR -> TIME
```

Java クラスに関連する変換

- データベースで Java が実行可能な場合は、convert を使用して、次の方法でデータ型を変更できます。
  - Java オブジェクト・タイプを SQL データ型に変換する。
  - SQL データ型を Java タイプに変換する。
  - 式 (ソース・クラス) のコンパイル時のデータ型がターゲット・クラスのサブクラスまたはスーパークラスである場合、Adaptive Server にインストールされた任意の Java-SQL クラスを、Adaptive Server にインストールされた任意の他の Java-SQL クラスに変換する。

  変換の結果は、現在のデータベースと関連付けられます。

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　すべてのユーザが convert 関数を実行できます。

参照

マニュアル　使用できるデータ型マッピングのリストと、Java クラスに関連するデータ型変換の詳細については、『Adaptive Server Enterprise における Java』を参照してください。

データ型　ユーザ定義データ型

関数　hextoint, inttohex

cos

説明 指定した角度の余弦を返します。

構文 cos(*angle*)

パラメータ *angle*

概数値型 (float、real、または double precision) のカラム名、変数、または定数式のいずれかです。

例

```
select cos(44)

 0.999843
```

使用法

- 算術関数 cos は、指定した角度 (ラジアン) の余弦を返します。
- 算術関数の概要については、「算術関数」(67 ページ) を参照してください。

規格 ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション すべてのユーザが cos 関数を実行できます。

参照 **関数** acos, degrees, radians, sin

cot

説明　指定した角度の余接を返します。

構文　cot(*angle*)

パラメータ　*angle*

概数値型 (float、real、または double precision) のカラム名、変数、または定数式のいずれかです。

例

```
select cot(90)

--------------------
             -0.501203
```

使用法

- 算術関数 cot は、指定した角度 (ラジアン) の余接を返します。
- 算術関数の概要については、「算術関数」(67 ページ) を参照してください。

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　すべてのユーザが cot 関数を実行できます。

参照　**関数**　degrees, radians, sin

count

説明

null でない値の数 (重複する値を除く) または選択したローの数を整数で返します。

構文

count([all | distinct] *expression*)

パラメータ

all

すべての値に count を適用します。デフォルト設定は all です。

distinct

count を適用する前に重複する値を削除します。distinct はオプションです。

expression

カラム名、定数、関数、これらを算術演算子やビット処理演算子でつないだ任意の組み合わせ、またはサブクエリです。通常、集合関数では、expression はカラム名です。詳細については、「式」(341 ページ) を参照してください。

例

例 1 作者の居住都市の数を求めます。

```
select count(distinct city)
from authors
```

例 2 titles テーブルにある種類をリストにしますが、該当する本が 1 冊だけの種類や該当する本がない種類は除きます。

```
select type
from titles
group by type
having count (*) > 1
```

使用法

- 集合関数 count は、カラムにある null 以外の値の数をカウントします。集合関数の概要については、「集合関数」(48 ページ) を参照してください。
- distinct を指定すると、count はカラムにある null 以外のユニークな値の数をカウントします。count は、unichar などのすべてのデータ型で使用できますが、text と image では使用できません。カウントでは null 値は無視されます。
- 空のテーブルや null 値だけが入っているカラムやグループでは、count(*column_name*) は値 0 を返します。
- count(*) はローの数をカウントします。count(*) には引数がないので、distinct は指定できません。null 値の有無に関係なく、すべてのローがカウントされます。
- 複数のテーブルをジョインするときは、「select リスト」に count(*) を含めて、ジョイン結果のロー数をカウントします。1 つのテーブルから基準と一致するローの数をカウントする場合は、count(*column_name*) を使用します。

- count を使って、サブクエリ内にあるものが存在するかどうかを調べることができます。たとえば、次のようにします。

```
select * from tab where 0 <
    (select count(*) from tab2 where ...)
```

  ただし、count は一致するすべての値をカウントするため、exists や in の方が結果が早く返されます。たとえば、次のようにします。

```
select * from tab where exists
    (select * from tab2 where ...)
```

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　すべてのユーザが count 関数を実行できます。

参照　**コマンド**　compute 句、group 句と having 句、select、where 句

count_big

説明

null でない値の数 (重複を除く) または選択されたローの数を bigint 型の値として返します。

構文

count_big([all | distinct] *expression*)

パラメータ

all

すべての値に count_big を適用します。デフォルトは all です。

distinct

count_big を適用する前に重複する値を削除します。distinct はオプションです。

expression

カラム名、定数、関数、これらを算術演算子やビット処理演算子でつないだ任意の組み合わせ、またはサブクエリです。通常、集合関数では、expression はカラム名です。

例

systypes 内の *name* の数をカウントします。

```
1> select count_big(name) from systypes
2> go
--------------------
42
```

使用法

- 集合関数 count_big は、カラムにある null 以外の値の数をカウントします。
- distinct を指定すると、count_big は、カラムにある null 以外のユニークな値の数をカウントします。カウントでは null 値は無視されます。
- 空のテーブルや null 値だけが入っているカラムやグループでは、count_big(*column_name*) は値 0 を返します。
- count_big(*) はローの数をカウントします。count_big(*) には引数がなく、distinct を指定することもできません。null 値の有無に関係なく、すべてのローがカウントされます。
- 複数のテーブルをジョインするときは、select リストに count_big(*) を含めて、ジョイン結果のロー数をカウントします。1 つのテーブルから基準と一致したローの数をカウントする場合は、count_big(*column_name*) を使用します。
- count_big を使用して、サブクエリ内にあるものが存在するかどうかを調べることができます。たとえば、次のようにします。

  ```
  select * from tab where 0 <
      (select count_big(*) from tab2 where ...)
  ```

  ただし、count_big は一致するすべての値をカウントするため、exists や in の方が結果が早く返されます。たとえば、次のようにします。

  ```
  select * from tab where exists
  (select * from tab2 where ...)
  ```

規格	ANSI SQL － 準拠レベル：Transact-SQL 拡張機能
パーミッション	すべてのユーザが count_big を実行できます。
参照	**コマンド**　compute 句、group by 句と having 句、select、where 句

current_bigdatetime

説明 現在の時刻を表すマイクロ秒の精度の bigtime 値を返します。現在の時刻部分の精度はシステム・クロックの精度によって制限されます。

構文 current_bigdatetime()

パラメータ なし。

例

例 1 現在の bigdatetime を求めます。

```
select current_bigdatetime()
------------------------------
Nov 25 1995 10:32:00.010101AM
```

例 2 現在の bigdatetime を求めます。

```
select datepart(us, current_bigdatetime())
------------------------------
010101
```

使用法 サーバ上にある現在の日付を取得します。

規格 ANSI SQL － 準拠レベル：初級レベル

パーミッション すべてのユーザが current_date 関数を実行できます。

参照

データ型 日付と時刻のデータ型

コマンド select, where 句

関数 dateadd、datediff, datepart、datename、current_bigtime

current_bigtime

説明　現在の時刻を表すマイクロ秒の精度の bigtime 値を返します。現在の時刻部分の精度はシステム・クロックの精度によって制限されます。

構文　current_bigtime()

パラメータ　なし。

例

例 1 現在の bigtime を求めます。

```
select current_bigtime()
------------------------------
10:32:00.010101AM
```

例 2 現在の bigtime を求めます。

```
select datepart(us, current_bigtime())
------------------------------
01010
```

使用法　サーバ上にある現在の日付を取得します。

規格　ANSI SQL － 準拠レベル：初級レベル

パーミッション　すべてのユーザが current_date 関数を実行できます。

参照

データ型　日付と時刻のデータ型

コマンド　select, where 句

関数　dateadd、datediff, datepart、datename、current_bigdatetime

current_date

説明 現在の日付を返します。

構文 current_date()

パラメータ なし。

例

例 1 datename を使用して現在の日付を確認します。

```
1> select datename(month, current_date())
2> go

------------------------------
August
```

例 2 datepart を使用して現在の日付を確認します。

```
1> select datepart(month, current_date())
2> go

-----------
8

(1 row affected)
```

使用法 サーバ上にある現在の日付を取得します。

規格 ANSI SQL － 準拠レベル：初級レベル

パーミッション すべてのユーザが current_date 関数を実行できます。

参照

データ型 日付と時刻のデータ型

コマンド select, where 句

関数 dateadd, datename, datepart, getdate

current_time

説明　現在の時刻を返します。

構文　current_time()

パラメータ　なし。

例

例 1 現在の時刻を検索します。

```
1> select current_time()
2> go

--------------------------
Aug 29 2003

(1 row affected)
```

例 2 datename と一緒に使用します。

```
1> select datename(minute, current_time())
2> go

------------------------------
45

(1 row affected)
```

使用法　サーバ上にある現在の時刻を取得します。

規格　ANSI SQL － 準拠レベル：初級レベル

パーミッション　すべてのユーザが current_time 関数を実行できます。

参照

データ型　日付と時刻のデータ型

コマンド　select, where 句

関数　dateadd, datename, datepart, getdate

curunreservedpgs

説明 指定したディスク区分の空きページ数を返します。

構文 curunreservedpgs (dbid, lstart, unreservedpgs)

パラメータ

dbid

データベースの ID です。この ID は sysdatabases の db_id カラムに格納されています。

lstart

戻り値としてカウントされるページが入っているディスク区分内のページです。

unreservedpgs

要求したデータベースの dbtable が現在ないときに返すデフォルト値です。

例

例 1 デバイス・フラグメントごとに、データベース名、デバイス名、未使用ページ数を返します。

データベースが開いている場合、curunreservedpgs は値をメモリから取得します。データベースが使用されていない場合、値は curunreservedpgs に指定した 3 番目のパラメータから取得されます。この例では、値が sysusages テーブルの unreservedpgs カラムから取得されます。

```
select
(dbid), d.name,
    curunreservedpgs(dbid, lstart, unreservedpgs)
    from sysusages u, sysdevices d
where u.vdevno=d.vdevno
and d.status &2 = 2
```

```
                              name
 ------------------------------ -------------------------- -----------
master                          master                            1634
tempdb                          master                             423
model                           master                             423
pubs2                           master                              72
sybsystemdb                     master                             399
sybsystemprocs                  master                            6577
sybsyntax                       master                             359

(7 rows affected)
```

例 2 dbid のセグメント上の空きページ数を始点 sysusages.lstart から表示します。

```
select curunreservedpgs (dbid, sysusages.lstart, 0)
```

使用法

- curunreservedpgs システム関数は、ディスク区分内の空きページ数を返します。システム関数の概要については、「システム関数」(69 ページ) を参照してください。
- データベースが開いている場合、curunreservedpgs から返される値はメモリから取得されます。データベースが使用されていない場合、値は curunreservedpgs に指定した 3 番目のパラメータから取得されます。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが curunreservedpgs 関数を実行できます。

参照

関数　db_id, lct_admin

data_pages

説明

指定されたテーブル、インデックス、または特定のパーティションで使用されるページ数を返します。結果には、内部構造に使用するページは含まれません。

この関数は、バージョン 15.0 より前の Adaptive Server の data_pgs および ptn_data_pgs を置き換えます。

構文

data_pages(*dbid*, *object_id* [, *indid* [, *ptnid*]])

パラメータ

dbid

データ・ページがあるデータベースのデータベース ID です。

object_id

テーブル、ビュー、またはその他のデータベース・オブジェクトのオブジェクト ID です。この ID は sysobjects の id カラムに格納されています。

indid

ターゲット・インデックスのインデックス ID です。

ptnid

ターゲット・パーティションのパーティション ID です。

例

例 1 指定されたデータベースにある、オブジェクト ID が 31000114 のオブジェクトで使用されるページ数を返します (インデックスも含む)。

```
select data_pages(5, 31000114)
```

例 2 クラスタード・インデックスがあるかどうかに関係なく、データ・レイヤに含まれるオブジェクトで使用されるページ数を返します。

```
select data_pages(5, 31000114, 0)
```

例 3 クラスタード・インデックスに対応するインデックス・レイヤのオブジェクトで使用されるページ数を返します。データ・レイヤで使用されるページは含まれません。

```
select data_pages(5, 31000114, 1)
```

例 4 指定されたパーティション (この例では 2323242432) のデータ・レイヤにあるオブジェクトで使用されるページ数を返します。

```
select data_pages(5, 31000114, 0, 2323242432)
```

使用法

APL (全ページロック) テーブルにクラスタード・インデックスがある場合は、*indid* に次の値を指定します。

- 0 － データ・ページをレポートします。
- 1 － インデックス・ページをレポートします。

すべてのエラーで常に値 0 が返されます。たとえば、*object_id* が現在のデータベースに存在しない、ターゲットの *indid* または *ptnid* が見つからない、などの場合です。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが data_pages 関数を実行できます。

参照

関数　object_id, row_count

システム・プロシージャ　sp_spaceused

datachange

説明

update statistics が最後に実行された後にデータ分布で行われた変更の量を測定します。具体的には、特定のオブジェクト、パーティション、カラムで発生した**挿入**、**更新**、**削除**の数が測定されます。これは、update statistics の実行がクエリ・プランに役立つかどうかを判断する目安になります。

構文

datachange(*object_name*, *partition_name*, *column_name*)

パラメータ

object_name

現在のデータベース内のオブジェクト名です。

partition_name

データ・パーティションの名前です。null を指定できます。

column_name

datachange への要求の目的であるカラム名です。null を指定できます。

例

例 1 author_ptn パーティション内の au_id カラムで行われた変更をパーセンテージで示します。

```
select datachange("authors", "author_ptn", "au_id")
```

例 2 au_ptn パーティション内の authors テーブルで行われた変更をパーセンテージで示します。*column_name* パラメータに null 値を指定すると、ヒストグラム統計があるすべてのカラムがチェックされ、その中から最大のデータ変更値が取得されます。

```
select datachange("authors", "au_ptn", null)
```

使用法

- datachange 関数には 3 つのパラメータをすべて指定する必要があります。
- datachange は insert、delete、および update を測定しますが、個別のカウントは行いません。datachange は update を delete および insert としてカウントするため、1 つの update は datachange カウンタを 2 つ進めます。
- datachange 組み込み関数は、データ変更のカウントをロー数のパーセントとして返しますが、このときに分母となるロー数は、最初に存在したロー数ではなく現在残っているロー数です。たとえば、テーブルにローが 5 つあり、1 つのローを削除した場合、現在のロー数カウントが 4 で datachange カウンタが 1 なので、datachange は 25 % という値をレポートします。
- datachange は、テーブル (パーティションを指定した場合はパーティション) 内の合計ロー数のパーセンテージとして表されます。パーセンテージの値は 100 パーセントを超えることがあります。これは、オブジェクトの変更回数が、テーブル内のロー数を大幅に超える場合があるからです。特に、1 つのテーブルで削除と更新が非常に多く行われるとこの状況になります。
- datachange が示す値は、メモリ内の値です。この値は、ディスク内の値と異なる可能性があります。ディスク内の値は、sp_flushstats を実行したときに (オブジェクト記述子がフラッシュされたときに) ハウスキーピングによって更新されます。

- 分割されたテーブルのグローバル・インデックスに対してヒストグラムが作成されたときには、datachange 値はリセットされません。

datachange が 0 にリセットまたは初期化されるのは、以下の場合です。

- 新しいカラムが追加され、その datachange 値が初期化されるとき。
- 新しいパーティションが追加され、その datachange 値が初期化されるとき。
- データ・パーティションに固有のヒストグラムが作成、削除、または更新されたとき。この場合、これらの変更が加えられたカラムとパーティションに対応するヒストグラムの datachange 値がリセットされます。
- テーブルまたはパーティションのデータがトランケートされ、それらの datachange 値がリセットされたとき。
- 他のコマンドの実行結果としてテーブルが直接または間接に再分割され、テーブルのすべてのパーティションとカラムの datachange 値がリセットされたとき。
- テーブルの分割が解除され、テーブルのすべてのカラムの datachange 値がリセットされたとき。

datachange には、次のような制限があります。

- datachange 統計は、システム tempdb、ユーザ定義 tempdb、システム・テーブル、またはプロキシ・テーブルのテーブルに保持されません。
- datachange の更新は、非トランザクション指向です。トランザクションをロールバックしても datachange 値はロールバックされません。値が不正確になる可能性があります。
- カラムレベル・カウンタにメモリを割り当てられなかった場合、カラムレベルの datachange 値ではなくパーティションレベルの値が追跡されます。
- カラムレベルの datachange 値が維持されない場合、カラムの datachange 値がリセットされるたびにパーティションレベルの datachange 値がリセットされます。

パーミッション　datachange は、すべてのユーザが実行できます。

datalength

説明

指定したカラムや文字列の実際の長さをバイト単位で返します。

構文

datalength(*expression*)

パラメータ

expression

カラム名、変数、定数式、またはこれらを任意に組み合わせたもので 1 つの値を計算するものです。*expression* には、任意のデータ型を使用できます。通常はカラム名を指定します。*expression* が文字定数のときは常に引用符で囲みます。

例

publishers テーブルの pub_name カラムの長さを計算します。

```
select Length = datalength(pub_name)
from publishers

length
-----------
         13
         16
         20
```

使用法

- システム関数 datalength は、*expression* のバイト単位の長さを返します。
- Unicode データ型で定義されたカラムの場合、datalength は、各ローに保管されているデータの実際のバイト数を返します。たとえば、unitext のカラム ut に U+0041U+0042U+d800dc00 というロー値が保管されている場合、次の値が返されます。

  ```
  select datalength(ut) from unitable
  -------------
  8
  ```

- datalength 関数は、各ローに格納されているデータの実際の長さを計算します。datalength は、varchar、univarchar、varbinary、text、image データ型の場合に便利です。これは、これらのデータ型が可変長を保管し、後続ブランクを保管しないからです。char 値または unichar 値で null を入力可能であることを宣言すると、Adaptive Server 内部では、これらの値が varchar または univarchar として格納されます。他のすべてのデータ型の場合、datalength 関数は定義済みの長さをレポートします。
- NULL データに対して datalength 関数を実行すると、常に NULL が返されます。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが datalength 関数を実行できます。

参照

関数 char_length, col_length

dateadd

説明

指定した日付または時刻に間隔を追加します。引数は 3 つ (日付要素、数字、発生順の式) 使用できます。結果のデータ型は最後の引数のデータ型と同じで、値は元の値に日付要素の数字を足したものです。最後の引数が bigtime で、日付要素が年、月、または日の場合、結果は元の bigtime 引数です。

構文

dateadd(*date_part*, *integer*, {*date* | *time* | *bigtime* | *datetime*, | *bigdatetime*})

パラメータ

date_part

日付要素または略語です。Adaptive Server が認識する日付要素と略語のリストについては、「日付要素」(66 ページ) を参照してください。

numeric

整数式です。

date expression

datetime、smalldatetime、bigdatetime、bigtime、date、time 型の式、または datetime フォーマットの文字列です。

例

例 1 bigtime に 100 万マイクロ秒を加えます。

```
declare @a bigtime
select @a = "14:20:00.010101"
select dateadd(us, 1000000, @a)
------------------------------
2:20:01.010101PM
```

例 2 bigdatetime に 25 時間を加えると日が増分します。

```
declare @a bigdatetime
select @a = "apr 12, 0001 14:20:00 "
select dateadd(hh, 25, @a)
------------------------------
Apr 13 0001   2:20PM
```

例 3 titles テーブル内のすべての本の出版日が 21 日ずれた場合、その新しい出版日を表示します。

```
select newpubdate = dateadd(day, 21, pubdate)
from titles
```

例 4 日付に 1 日加えます。

```
declare @a date
select @a = "apr 12, 9999"
select dateadd(dd, 1, @a)
-------------------------
Apr 13 9999
```

例 5 時刻から 5 分を引きます。

```
select dateadd(mi, -5, convert(time, "14:20:00"))
-------------------------
2:15PM
```

例 6 時刻に 1 日加えると時刻は変わりません。

```
declare @a time
select @a = "14:20:00"
select dateadd(dd, 1, @a)
--------------------------
2:20PM
```

例 7 各 date_part には datetime 値のように制限がありますが、制限値よりも大きい値を加算して、次の有効なフィールドに値を反映できます。

```
--Add 24 hours to a datetime
select dateadd(hh, 24, "4/1/1979")
--------------------------
Apr  2 1979 12:00AM

--Add 24 hours to a date
select dateadd(hh, 24, "4/1/1979")
--------------------------
Apr  2 1979
```

使用法

- 日付関数 dateadd は、指定した日付に特定の間隔を追加します。日付関数の詳細については、「日付関数」(65 ページ) を参照してください。
- dateadd 関数には 3 つの引数 (日付要素、数字、日付) があります。結果は、指定された日付に日付要素数を加算した datetime 値です。

  日付引数が smalldatetime 値の場合、結果も smalldatetime 値です。dateadd 関数を使用して smalldatetime に秒またはミリ秒を追加できますが、このような追加が意味を持つのは、dateadd 関数が返す結果の日付が 1 分以上変化するときだけです。
- 日付順の値ではなく文字列が引数として指定された場合、サーバは示された精度にかかわらず、その文字列を datetime 値として解釈します。デフォルトの動作は、設定パラメータ builtin date strings または設定オプション builtin_date_strings を設定すると変更できます。これらのオプションを設定すると、サーバは発生順の組み込み関数に与えられた文字列を bigdatetime として解釈します。詳細については、『システム管理ガイド』を参照してください。
- この組み込み関数にマイクロ秒の日付要素を与えると、文字列値は常に bigdatetime として解釈されます。
- datetime データ型は、1753 年 1 月 1 日より後の日付だけに使用します。datetime 値は一重引用符か二重引用符で囲んでください。0001 年 1 月 1 日から 9999 年 1 月 1 日までの日付には、date データ型を使用します。date は一重引用符か二重引用符で囲んでください。それよりも前の日付には、char、nchar、varchar、または nvarchar を使用します。Adaptive Server では、さまざまな種類の日付フォーマットを識別できます。詳細については、「ユーザ定義データ型」(41 ページ) と「データ型変換関数」(57 ページ) を参照してください。

Adaptive Server は、必要に応じて (たとえば、文字値を datetime 値と比較するとき) 文字値と datetime 値とを自動的に変換します。

- dateadd 関数で日付要素の weekday や dw を使うのは論理的でなく、疑似結果が返されます。これらの代わりに day または dd を使用してください。

表 *2-8: date_part* として認識される略語

日付要素	省略形	値
Year	yy	1753 ～ 9999 (datetime) 1900 ～ 2079 (smalldatetime) 0001 ～ 9999 (date)
Quarter	qq	1 ～ 4
Month	mm	1 ～ 12
Week	wk	1054
Day	dd	1 ～ 7
dayofyear	dy	1 ～ 366
Weekday	dw	1 ～ 7
Hour	hh	0 ～ 23
Minute	mi	0 ～ 59
Second	ss	0 ～ 59
millisecond	ms	0 ～ 999
microsecond	us	0 ～ 999999

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　すべてのユーザが dateadd 関数を実行できます。

参照

データ型　日付と時刻のデータ型

コマンド　select, where 句

関数　datediff, datename, datepart, getdate

datediff

説明

指定した 2 つの日付間または時刻間で日付要素の数を計算します。引数は 3 つあります。最初の引数は日付要素です。2 番目と 3 番目は発生順の値です。dates、times、datetimes、および bigdatetimes の場合、結果は日付要素の date2 - date1 に等しい符号付き整数値です。

•2 番目または 3 番目の引数が日付で、日付要素が時間、分、秒、ミリ秒、またはマイクロ秒の場合、日付は午前 0 時として処理されます。

•2 番目または 3 番目の引数が時刻で、日付要素が年、月、または日の場合は、ゼロが返されます。

•datediff 関数の結果が日付要素の整数倍でないときは、結果が丸められずトランケートされます。

• 小さい時間単位では、オーバフロー値が生じることがあります。これらの制限を超えた場合は、関数からオーバフロー・エラーが返されます。

構文

datediff(*datepart*, {*date, date* | *time*, *time* | *bigtime*, *bigtime* | *datetime*, *datetime* | *bigdatetime*, *bigdatetime*}])

パラメータ

datepart

日付要素または略語です。Adaptive Server が認識する日付要素と略語のリストについては、「日付要素」(66 ページ) を参照してください。

date expression1

datetime、smalldatetime、bigdatetime、bigtime、date、time 型の式、または datetime フォーマットの文字列です。

date expression2

datetime、smalldatetime、bigdatetime、bigtime、date、time 型の式、または datetime フォーマットの文字列です。

例

例 1 2 つの bigdatetime 間のマイクロ秒数を返します。

```
declare @a bigdatetime
declare @b bigdatetime
select @a = "apr 1, 1999  00:00:00.000000"
select @b = "apr 2, 1999  00:00:00.000000"
select datediff(us, @a, @b)
--------------------------
86400000000
```

例 2 オーバフロー・エラーが返されます。

```
select datediff(ms, convert(bigdatetime, "4/1/1753"),
convert(bigdatetime, "4/1/9999"))
Msg 535, Level 16, State 0:
Line 2:
Difference of two datetime fields caused overflow at runtime.
Command has been aborted
```

例 3 pubdate から (getdate 関数で求めた) 現在の日付までの経過日数を求めます。

```
select newdate = datediff(day, pubdate, getdate())
    from titles
```

例 4 2 つの時刻間の時間数を求めます。

```
declare @a time
declare @b time
select @a = "20:43:22"
select @b = "10:43:22"
select datediff(hh, @a, @b)
-----------
   -10
```

例 5 2 つの日付間の時間数を求めます。

```
declare @a date
declare @b date
select @a = "apr 1, 1999"
select @b = "apr 2, 1999"
select datediff(hh, @a, @b)
-----------
     24
```

例 6 2 つの時刻間の日数を求めます。

```
declare @a time
declare @b time
select @a = "20:43:22"
select @b = "10:43:22"
select datediff(dd, @a, @b)
-----------
     0
```

例 7 オーバフロー・エラーが返されます。

```
select datediff(ms, convert(date, "4/1/1753"), convert(date, "4/1/9999"))
Msg 535, Level 16, State 0:
Line 2:
Difference of two datetime fields caused overflow at runtime.
Command has been aborted
```

使用法

- datediff 日付関数は、指定した 2 つの日付間の日付要素の数を計算します。日付関数の詳細については、「日付関数」(65 ページ) を参照してください。
- datediff 関数には 3 つの引数があります。最初の引数は日付要素です。2 番目と 3 番目の引数は日付です。結果は日付要素の *date2 - date1* に等しい符号付きの整数です。
- datediff 関数はデータ型 int の結果を生成しますが、結果が 2,147,483,647 を超えるとエラーになります。秒値の場合、68 年 19 日 3:14:07 時間になります。
- datediff 関数の結果が日付要素の整数倍でないときは、結果は丸められず、常にトランケートされます。たとえば、日付要素として hour を使用すると、「4:00AM」と「5:50AM」の差は 1 になります。

  日付要素として day を使用すると、datediff 関数は指定した 2 つの時刻の間の午前 0 時の数をカウントします。たとえば、1992 年 1 月 1 日 23:00 時と 1992 年 1 月 2 日 01:00 時の間の差は 1 ですが、1992 年の 1 月 1 日 00:00 時と 1992 年 1 月 1 日の 23:59 の間の差は 0 です。
- 日付要素 month は、2 つの日付間の各月の第 1 日目の数をカウントします。たとえば、1 月 25 日と 2 月 2 日の差は 1 ですが、1 月 1 日と 1 月 31 日の差は 0 です。
- 日付要素 week を datediff 関数で使用すると、2 つの日付の間 (2 番目の日付は含みますが、1 番目の日付は含みません) の日曜日の数をカウントします。たとえば、1 月 4 日の日曜日と 1 月 11 日の日曜日の間における週の数は 1 です。
- smalldatetime 値を使用すると、この値は内部で計算するために datetime 値に変換されます。smalldatetime 値の秒やミリ秒は、差を計算するために自動的に 0 に設定されます。
- 2 番目または 3 番目の引数が日付であり、datepart が時間、分、秒、またはミリ秒の場合は、その日付は午前 0 時として処理されます。
- 2 番目または 3 番目の引数が時刻であり、datepart が年、月、または日の場合は、ゼロが返されます。
- datediff 関数の結果が日付要素の整数倍でないときは、結果は丸められずトランケートされます。
- 日付順の値ではなく文字列が引数として指定された場合、サーバは示された精度にかかわらず、その文字列を datetime 値として解釈します。デフォルトの動作は、設定パラメータ builtin date strings または設定オプション builtin_date_strings を設定すると変更できます。これらのオプションを設定すると、サーバは発生順の組み込み関数に与えられた文字列を bigdatetime として解釈します。詳細については、『システム管理ガイド』を参照してください。

- この組み込み関数にマイクロ秒の日付要素を与えると、文字列値は常に bigdatetime として解釈されます。
- より小さい time 単位では、オーバフロー値が生じることがあります。次の制限を超えた場合は、関数からオーバフロー・エラーが返されます。
  - マイクロ秒：約 3 日
  - ミリ秒：約 24 日
  - 秒：約 68 年
  - 分：約 4083 年
  - その他：オーバフロー制限なし

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　すべてのユーザが datediff 関数を実行できます。

参照

データ型　日付と時刻のデータ型

コマンド　select, where 句

関数　dateadd, datename, datepart, getdate

datename

説明

指定した date または time (2 番目の引数) の指定した datepart (最初の引数) を文字列として返します。date、time、bigdatetime、bigtime、datetime、smalldatetime のいずれかの値を 2 番目の引数として指定します。

構文

datename(*datepart* {*date* | *time* | *bigtime* | *datetime* | *bigdatetime*})

パラメータ

datepart

日付要素または略語です。Adaptive Server が認識する日付要素と略語のリストについては、「日付要素」(66 ページ) を参照してください。

date_expression

datetime、smalldatetime、bigdatetime、bigtime、time 型の式、または datetime フォーマットの文字列です。

例

例 1 bigdatetime の月名を求めます。

```
declare @a bigdatetime
select @a = "apr 12, 0001 00:00:000.010101"
select datename(mm, @a)
------------------------------
April
```

例 2 現在の日付が 2000 年 11 月 20 日だとします。

```
select datename(month, getdate())

November
```

例 3 日付の月名を求めます。

```
declare @a date
select @a = "apr 12, 0001"
select datename(mm, @a)
------------------------------
    April
```

例 4 時刻の秒を求めます。

```
declare @a time
select @a = "20:43:22"
select datename(ss, @a)
------------------------------
    22
```

使用法

- 日付関数 datename は、datetime 値または smalldatetime 値の指定した要素の名前 (月「June」など) を文字列として返します。結果が数値 (「23」日など) の場合でも、この値を文字列として返します。
- 日付関数の詳細については、「日付関数」(65 ページ) を参照してください。
- datename 関数で日付要素の weekday または dw を使用すると、曜日 (Sunday、Monday など) が返されます。
- smalldatetime は分単位までしか正確でないため、datename 関数で smalldatetime 値を使用するときは、秒とミリ秒は常に 0 です。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが datename 関数を実行できます。

参照

データ型　日付と時刻のデータ型

コマンド　select, where 句

関数　dateadd, datename, datepart, getdate

datepart

説明

指定した date (2 番目の引数) の指定した datepart (最初の引数) を整数として返します。date、time、datetime、bigdatetime、bigtime、smalldatetime のいずれかの値を 2 番目の引数として指定します。datepart が時間、分、秒、ミリ秒、またはマイクロ秒の場合は、結果は 0 になります。

構文

datepart(*date_part* {*date* | *time* | *datetime* | *bigtime* | *bigdatetime*})

パラメータ

date_part

日付要素です。表 2-9 に、日付要素、datepart に指定できる略語、有効値を示します。

表 *2-9:* 日付要素とその有効値

日付要素	略語	値
year	yy	1753 ～ 9999 (smalldatetime の場合は 2079)。date の場合は 0001 ～ 9999
quarter	qq	1 ～ 4
month	mm	1 ～ 12
week	wk	1 ～ 54
day	dd	1 ～ 31
dayofyear	dy	1 ～ 366
weekday	dw	1 ～ 7 (日曜 ～ 土曜)
hour	hh	0 ～ 23
minute	mi	0 ～ 59
second	ss	0 ～ 59
millisecond	ms	0 ～ 999
microsecond	us	0 ～ 999999
calweekofyear	cwk	1 ～ 53
calyearofweek	cyr	1753 ～ 9999 (smalldatetime の場合は 2079)。date の場合は 0001 ～ 9999
caldayofweek	cdw	1 ～ 7

2 桁の年 (*yy*) を入力する場合には、以下の点に注意してください。

- 50 未満の数値は 20*yy* として解釈されます。たとえば、`01` は 2001、`32` は 2032、`49` は 2049 です。
- 50 以上の数値は 19*yy* として解釈されます。たとえば、`50` は 1950、`74` は 1974、`99` は 1999 です。

  datetime、smalldatetime、および time 型のミリ秒は、前にコロンかピリオドが付く場合があります。コロンに続ける場合、数字は 1 秒の 1000 分の 1 を意味します。ピリオドに続ける場合、1 桁は 1 秒の 10 分の 1、2 桁は 100 分の 1、3 桁は 1000 分の 1 を意味します。たとえば、「12:30:20:1」は 12 時 30 分から 20 秒と 1000 分の 1 秒が経過したことを示し、「12:30:20.1」は 12 時 30 分から 20 秒と 10 分の 1 秒が経過したことを示します。

  ミリ秒には小数点を付けて端数を表す必要があります。

date_expression

datetime、smalldatetime、bigdatetime、bigtime、date、time 型の式、または datetime フォーマットの文字列です。

例

例 1 bigdatetime のマイクロ秒を求めます。

```
declare @a bigdatetime
select @a = "apr 12, 0001 12:00:00.000001"
select datepart(us, @a)
-----------
000001
```

例 2 現在の日付が 1995 年 11 月 25 日であるとします。

```
select datepart(month, getdate())

-----------
          11
```

例 3 伝統料理の本の出版年を返します。

```
select datepart(year, pubdate) from titles
    where type = "trad_cook"

 -----------
        1990
        1985
        1987
```

例 4

```
select datepart(cwk,'1993/01/01')

-----------
          53
```

例 5

```
select datepart(cyr,'1993/01/01')

-----------
        1992
```

例 6

```
select datepart(cdw,'1993/01/01')

-----------
           5
```

例 7 時刻の時間を求めます。

```
declare @a time
select @a = "20:43:22"
select datepart(hh, @a)
-----------
    20
```

例 8 datename または datepart を使用して、date から時間、分、秒の部分を求めると、結果はデフォルト時間のゼロになります。datename または datepar を使用して、time から月、日、または年を求めると、結果はデフォルト日付の 1990 年 1 月 1 日になります。

```
--Find the hours in a date
declare @a date
select @a = "apr 12, 0001"
select datepart(hh, @a)
-----------
    0

--Find the month of a time
declare @a time
select @a = "20:43:22"
select datename(mm, @a)
------------------------------
January
```

datetime 関数にパラメータとして null 値を指定した場合は、NULL が返されます。

使用法

- 日付関数 datepart は、datetime 値の指定した日付要素の整数値を返します。日付関数の詳細については、「日付関数」(65 ページ) を参照してください。
- datepart は、ISO 標準 8601 に従った数字を返します。これは週の最初の日と年の最初の週を定義します。datepart 関数に calweekofyear、calyearofweek、または caldayorweek のどの値が含まれるかによって、同じ時間単位について返される日付が異なることがあります。たとえば、Adaptive Server が英語 (U.S. English) をデフォルト言語として使用するように設定されているとします。次の関数は 1988 を返します。

  ```
  datepart(cyr, "1/1/1989")
  ```

  一方、次の関数は 1989 を返します。

  ```
  datepart(yy, "1/1/1989)
  ```

  この違いは、ISO 標準が年の最初の週を、木曜日を含みかつ月曜から始まる最初の週として定義するために発生します。

  デフォルトの言語として英語を使用しているサーバでは、週の最初の曜日は日曜日で、年の最初の週は 1 月 4 日を含む週となります。

- datepart 関数で日付要素の weekday または dw を使用すると、対応する番号が返されます。曜日の名前に対応する番号は datefirst の設定により異なります。一部の言語のデフォルト (us_english など) では、日曜日は 1、月曜日は 2 になり、他の言語では月曜日が 1、火曜日が 2 になります。デフォルト設定をセッションごとに変更するには、set datefirst を使用します。詳細については、set コマンドの datefirst オプションを参照してください。

- calweekofyear (略語は cwk) は、その年内の週の順序数を返します。calyearofweek (略語は cyr) は、その週が始まる年を返します。caldayofweek (略語は cdw) は、その週におけるその日の順序数を返します。dateadd、datediff、datename では、calweekofyear、calyearofweek、caldayofweek を日付要素としては使用できません。
- datetime および time が正確なのは 300 分の 1 秒単位までなので、datepart 関数でこれらのデータ型の値を使用すると、ミリ秒は最も近い 300 分の 1 秒に丸められます。
- smalldatetime が正確なのは分単位までなので、datepart 関数で smalldatetime 値を使うときは、秒とミリ秒は常に 0 です。
- 曜日日付要素の値は言語設定の影響を受けます。

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　すべてのユーザが datepart 関数を実行できます。

参照

データ型　日付と時刻のデータ型

コマンド　select, where 句

関数　dateadd, datediff, datename, getdate

day

説明

指定した日付の datepart の日を表す整数を返します。

構文

day(*date_expression*)

パラメータ

date_expression

datetime、smalldatetime、date 型の式、または datetime フォーマットの文字列です。

例

整数 02 を返します。

```
day("11/02/03")
----------
02
```

使用法

day(*date_expression*) は datepart(dd,*date_expression*) と同じです。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが day 関数を実行できます。

参照

データ型 datetime、smalldatetime、date、time

関数 datepart, month, year

db_attr

説明　指定したデータベースの durability、dml_logging、template の設定を返します。

構文　db_attr('*database_name*' | *database_ID* | NULL, 'attribute')

パラメータ

database_name
データベースの名前です。

database_ID
データベースの ID です。

NULL
db_attr が現在のデータベースについてレポートします。

attribute
次のいずれかです。

- help － db_attr の使用方法を表示します。
- durability － 特定のデータベースの持続性を返します。full、at_shutdown、または no_recovery。
- dml_logging － 指定したデータベースの DML ロギングの値を返します。full または minimal。
- template － 指定したデータベースに使用されたテンプレート・データベースの名前を返します。テンプレートとして使用されたデータベースがない場合は、NULL を返します。

例　**例 1** db_attr の構文を返します。

```
select db_attr(0, "help")
Usage: db_attr('dbname' | dbid | NULL, 'attribute')
List of options in attributes table:
         0 : help
         1 : durability
         2 : dml_logging
         3 : template
-----------------------------------------------------
NULL
```

例 2 名前、durability 設定、dml_logging 設定、および使用されたテンプレートを sysdatabses から選択します。

```
select  name = convert(char(20), name),
durability = convert(char(15), db_attr(name,     "durability")),
     dml_logging = convert(char(15), db_attr(dbid,     "dml_logging")),
     template = convert(char(15), db_attr(dbid, "template"))
from sysdatabases
name               durability         dml_logging                 template
-----------------   ----------------   ------------------------    --------
master              full               full                         NULL
model               full               full                         NULL
tempdb              no_recovery        full                         NULL
sybsystemdb         full               full                         NULL
sybsystemprocs      full               full                         NULL
repro               full               full                         NULL
imdb                no_recovery        full                         db1
db                  full               full                         NULL
at_shutdown_db      at_shutdown        full                         NULL
db1                 full               full                         NULL
dml                 at_shutdown        minimal                      NULL
```

例 3 存在しない DoesNotExist データベースに対して db_attr を実行します。

```
select db_attr("DoesNotExist", "durability")
------------------------------------------------------
NULL
```

例 4 ID が 12345 の存在しないデータベースに対して db_attr を実行します。

```
select db_attr(12345, "durability")
------------------------------------------------------
NULL
```

例 5 存在しない属性に対して db_attr を実行します。

```
select db_attr(1, "Cmd Does Not Exist")
------------------------------------------------------
NULL
```

使用法

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが db_attr 関数を実行できます。

参照

関数

db_id

説明　指定したデータベースの ID 番号を返します。

構文　db_id(*database_name*)

パラメータ　*database_name*

データベースの名前です。*database_name* には、文字式を指定する必要があります。expression が文字定数のときは、引用符で囲んでください。

例　sybsystemprocs の ID 番号を返します。

```
select db_id("sybsystemprocs")

------
4
```

使用法

- システム関数 db_id は、データベースの ID 番号を返します。
- *database_name* を指定しないと、db_id 関数は現在のデータベースの ID 番号を返します。
- システム関数の概要については、「システム関数」(69 ページ) を参照してください。

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　すべてのユーザが db_id 関数を実行できます。

参照　**関数**　db_name, object_id

db_instanceid

説明

クラスタ環境のみ － 指定したローカル・テンポラリ・データベースの所有インスタンスの ID を返します。指定したデータベースがグローバル・テンポラリ・データベースまたは非テンポラリ・データベースである場合は NULL を返します。

構文

db_instanceid(*database_id*)
db_instanceid(*database_name*)

パラメータ

database_id
データベースの ID です。

database_name
データベースの名前です。

例

データベース ID 5 の所有インスタンスを返します。

```
select db_instanceid(5)
```

使用法

- ローカル・テンポラリ・データベースへのアクセスは所有インスタンスからのみ可能です。db_instanceid は、指定したデータベースがローカル・テンポラリ・データベースでローカル・テンポラリ・データベースの所有インスタンスかどうかを判別します。確認後、所有インスタンスに接続して、そのローカル・テンポラリ・データベースにアクセスできます。
- db_instanceid はパラメータを指定する必要があります。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが sdc_intempdbconfig を実行できます。

db_name

説明　ID 番号が指定されているデータベースの名前を返します。

構文　db_name([*database_id*])

パラメータ　*database_id*
(sysdatabases.dbid に格納されている) データベース ID の数値式です。

例　**例 1** 現在のデータベースの名前を返します。

```
select db_name()
```

例 2 データベース ID が 4 のデータベースの名前を返します。

```
select db_name(4)


------------------------------
sybsystemprocs
```

使用法

- システム関数 db_name はデータベース名を返します。
- *database_id* を指定しないときは、db_name は現在のデータベースの名前を返します。
- システム関数の概要については、「システム関数」(69 ページ) を参照してください。

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　すべてのユーザが db_name 関数を実行できます。

参照　**関数**　col_name, db_id, object_name

db_recovery_status

説明

クラスタ環境のみ − 指定したデータベースのリカバリ・ステータスを返します。*database_ID* または *database_name* の値を含めない場合は、現在のデータベースのリカバリ・ステータスを返します。

構文

db_recovery_status([*database_ID* | *database_name*])

パラメータ

database_ID

リカバリ・ステータスを要求するデータベースの ID です。

database_name

リカバリ・ステータスを要求するデータベースの名前です。

例

例 1 現在のデータベースのリカバリ・ステータスを返します。

```
select db_recovery_status()
```

例 2 test という名前のデータベースのリカバリ・ステータスを返します。

```
select db_recovery_status("test")
```

例 3 データベース ID が 8 のデータベースのリカバリ・ステータスを返します。

```
select db_recovery_status(8)
```

使用法

戻り値 0 は、データベースがノードフェールオーバ・リカバリ状態にないことを示します。戻り値 1 は、データベースがノードフェールオーバ・リカバリ状態にあることを示します。

規格

ANSI SQL − 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが db_recovery_status を実行できます。

degrees

説明　ラジアン数が指定されている角度の大きさを度数単位で返します。

構文　degrees(*numeric*)

パラメータ

numeric
度数に変換されるラジアン数です。

例

```
select degrees(45)

-----------
       2578
```

使用法

- degrees 算術関数はラジアンを度数に変換します。結果は、数値式と同じ型になります。

  numeric 型または decimal 型の式の場合、結果の内部精度は 77 で、位取りは式と同じです。

- 算術関数の概要については、「算術関数」(67 ページ) を参照してください。

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　すべてのユーザが degrees 関数を実行できます。

参照　**関数**　radians

derived_stat

説明

指定したオブジェクトとインデックスに対して得られた統計を返します。

構文

```
derived_stat("object_name" | object_id,
        index_name | index_id,
        ["partition_name" | partition_id,]
        “statistic”
```

パラメータ

object_name
統計を取るオブジェクトの名前です。完全修飾オブジェクト名を指定しない場合は、derived_stat は現在のデータベースを検索します。

object_id
統計を取るオブジェクトのオブジェクト ID で、*object_name* の代わりに指定できます。*object_id* は、現在のデータベース内に存在する必要があります。

index_name
指定したオブジェクトに属している、統計を取るインデックスの名前です。

index_id
指定したオブジェクトに属している、統計を取るインデックスのインデックス ID です。*index_name* の代わりに指定できます。

partition_name
特定のパーティションに属している、統計を取るパーティションの名前です。*partition_name* は省略可能です。*partition_name* または *partition_id* を使用すると、Adaptive Server はオブジェクト全体ではなくターゲット・パーティションの統計を返します。

partition_id
指定したオブジェクトに属している、統計を取るパーティションのパーティション ID です。*partition_name* の代わりに指定できます。*partition_id* は省略可能です。

“*statistic*”
返される統計です。取得可能な統計値は、次のとおりです。

- data page cluster ratio または dpcr − オブジェクトとインデックスの組み合わせのデータ・ページ・クラスタ率
- index page cluster ratio または ipcr − オブジェクトとインデックスの組み合わせのインデックス・ページ・クラスタ率
- data row cluster ratio または drcr − オブジェクトとインデックスの組み合わせのデータ・ロー・クラスタ率
- large io efficiency または lgio − オブジェクトとインデックスの組み合わせの大容量 I/O 効率
- space utilization または sput − オブジェクトとインデックスの組み合わせの領域使用率

例

例 1 titles テーブルの titleidind インデックスの領域使用率を選択します。

```
select derived_stat("titles", "titleidind", "space utilization")
```

例 2　titles テーブルのインデックス ID 2 のデータ・ページ・クラスタ率を選択します。"dpcr" または "data page cluster ratio" のどちらでも使用できます。

```
select derived_stat("titles", 2, "dpcr")
```

例 3 パーティション ID もパーティション名も指定しないので、オブジェクト全体の統計がレポートされます。

```
1> select derived_stat(object_id("t1"), 2, "drcr")
2> go

----------------------------
                    0.576923
```

例 4 パーティション tl_928003396 の統計をレポートします。

```
1> select derived_stat(object_id("t1"), 0, "t1_928003306", "drcr")
2> go

 ----------------------------
                     1.000000

(1 row affected)
```

例 5 syspartitions からのデータを使用して、指定したテーブルのすべてのインデックスの抽出統計を選択します。

```
select convert(varchar(30), name) as name, indid,
    convert(decimal(5, 3), derived_stat(id, indid, 'sput')) as 'sput',
    convert(decimal(5, 3), derived_stat(id, indid, 'dpcr')) as 'dpcr',
    convert(decimal(5, 3), derived_stat(id, indid, 'drcr')) as 'drcr',
    convert(decimal(5, 3), derived_stat(id, indid, 'lgio')) as 'lgio'
from syspartitions where id = object_id('titles')
go

 name                           indid   sput     dpcr     drcr     lgio
 ------------------------------ ------ -------- -------- -------- --------
 titleidind_2133579608               1    0.895    1.000    1.000    1.000
 titleind_2133579608                 2    0.000    1.000    0.688    1.000

(2 rows affected)
```

例 6 分割されたテーブルのすべてのインデックスとパーティションの抽出統計を選択します。ここで、mymsgs_rr4 はラウンドロビン分割テーブルで、グローバル・インデックスとローカル・インデックスで作成されます。

```
1> select * into mymsgs_rr4 partition by roundrobin 4 lock datarows
2> from master..sysmessages
2> go

(7597 rows affected)

1> create clustered index mymsgs_rr4_clustind on mymsgs_rr4(error, severity)
2> go
1> create index mymsgs_rr4_ncind1 on mymsgs_rr4(severity)
2> go
1> create index mymsgs_rr4_ncind2 on mymsgs_rr4(langid, dlevel) local index
2> go

2> update statistics mymsgs_rr4
1>

2> select convert(varchar(10), object_name(id)) as name,
3>      (select convert(varchar(20), i.name) from sysindexes i
4>       where i.id = p.id and i.indid = p.indid),
5> convert(varchar(30), name) as ptnname, indid,
6> convert(decimal(5, 3), derived_stat(id, indid, partitionid, 'sput')) as 'sput',
7> convert(decimal(5, 3), derived_stat(id, indid, partitionid, 'dpcr')) as 'dpcr',
8> convert(decimal(5, 3), derived_stat(id, indid, partitionid, 'drcr')) as 'drcr',
9> convert(decimal(5, 3), derived_stat(id, indid, partitionid, 'lgio')) as 'lgio'
10> from syspartitions p
11> where id = object_id('mymsgs_rr4')
 name                         ptnname                     indid sput dpcr drcr lgio
------------------------------ -------------------------- ----- ---- ---- ---- ----
mymsgs_rr4 mymsgs_rr4          mymsgs_rr4_786098810          0 0.90 1.000 1.00 1.000
mymsgs_rr4 mymsgs_rr4          mymsgs_rr4_802098867          0 0.90 1.000 1.00 1.000
mymsgs_rr4 mymsgs_rr4          mymsgs_rr4_818098924          0 0.89 1.000 1.00 1.000
mymsgs_rr4 mymsgs_rr4          mymsgs_rr4_834098981          0 0.90 1.000 1.00 1.000
mymsgs_rr4 mymsgs_rr4_clustind mymsgs_rr4_clustind_850099038 2 0.83 0.995 1.00 1.000
mymsgs_rr4 mymsgs_rr4_ncind1   mymsgs_rr4_ncind1_882099152   3 0.99 0.445 0.88 1.000
mymsgs_rr4 mymsgs_rr4_ncind2   mymsgs_rr4_ncind2_898099209   4 0.15 1.000 1.00 1.000
mymsgs_rr4 mymsgs_rr4_ncind2   mymsgs_rr4_ncind2_914099266   4 0.88 1.000 1.00 1.000
mymsgs_rr4 mymsgs_rr4_ncind2   mymsgs_rr4_ncind2_930099323   4 0.877 1.000 1.000
1.000
mymsgs_rr4 mymsgs_rr4_ncind2   mymsgs_rr4_ncind2_946099380   4 0.945 0.993 1.000
1.000
```

例 7 現在のデータベースのページ・ロック・テーブルすべての抽出統計を選択します。

```
2> select convert(varchar(10), object_name(id)) as name
3>     (select convert(varchar(20), i.name) from sysindexes i
4>      where i.id = p.id and i.indid = p.indid),
5> convert(varchar(30), name) as ptnname, indid,
6> convert(decimal(5, 3), derived_stat(id, indid, partitionid, 'sput')) as 'sput',
7> convert(decimal(5, 3), derived_stat(id, indid, partitionid, 'dpcr')) as 'dpcr',
8> convert(decimal(5, 3), derived_stat(id, indid, partitionid, 'drcr')) as 'drcr',
9> convert(decimal(5, 3), derived_stat(id, indid, partitionid, 'lgio')) as 'lgio'
10> from syspartitions p
11> where lockscheme(id) = "allpages"
12>   and (select o.type from sysobjects o where o.id = p.id) = 'U'

 name                      ptnname                       indid sput dpcr drcr lgio
----------- ----------------- ---------------------------- ----- ---- ---- ---- ----
 stores     stores            stores_18096074               0 0.276 1.000 1.000 1.000
 discounts  discounts         discounts_50096188            0 0.075 1.000 1.000 1.000
 au_pix     au_pix            au_pix_82096302               0 0.000 1.000 1.000 1.000
 au_pix     tau_pix           tau_pix_82096302            255 NULL NULL NULL NULL
 blurbs     blurbs            blurbs_114096416              0 0.055 1.000 1.000 1.000
 blurbs     tblurbs           tblurbs_114096416           255 NULL NULL NULL NULL
 t1apl      t1apl             t1apl_1497053338              0 0.095 1.000 1.000 1.000
 t1apl      t1apl             t1apl_1513053395              0 0.082 1.000 1.000 1.000
 t1apl      t1apl             t1apl_1529053452              0 0.095 1.000 1.000 1.000
 t1apl      t1apl_ncind       t1apl_ncind_1545053509        2 0.149 0.000 1.000 1.000
 t1apl      t1apl_ncind_local t1apl_ncind_local_1561053566 3 0.066 0.000 1.000 1.000
 t1apl      t1apl_ncind_local t1apl_ncind_local_1577053623 3 0.057 0.000 1.000 1.000
 t1apl      t1apl_ncind_local t1apl_ncind_local_1593053680 3 0.066 0.000 1.000 1.000
 authors    auidind           auidind_1941578924            1 0.966 0.000 1.000 1.000
 authors    aunmind           aunmind_1941578924            2 0.303 0.000 1.000 1.000
 publishers pubind            pubind_1973579038             1 0.059 0.000 1.000 1.000
 roysched   roysched          roysched_2005579152           0 0.324 1.000 1.000 1.000
 roysched   titleidind        titleidind_2005579152         2 0.777 1.000 0.941 1.000
 sales      salesind          salesind_2037579266           1 0.444 0.000 1.000 1.000
 salesdetai salesdetail       salesdetail_2069579380        0 0.614 1.000 1.000 1.000
 salesdetai titleidind        titleidind_2069579380         2 0.518 1.000 0.752 1.000
 salesdetai salesdetailind    salesdetailind_2069579380     3 0.794 1.000 0.726 1.000
 titleautho taind             taind_2101579494              1 0.397 0.000 1.000 1.000
 titleautho auidind           auidind_2101579494            2 0.285 0.000 1.000 1.000
 titleautho titleidind        titleidind_2101579494         3 0.223 0.000 1.000 1.000
 titles     titleidind        titleidind_2133579608         1 0.895 1.000 1.000 1.000
 titles     titleind          titleind_2133579608           2 0.402 1.000 0.688 1.000

(27 rows affected)
```

使用法

- derived_stat は倍精度値を返します。
- derived_stat が返す値は、optdiag ユーティリティで表示される値と一致します。
- 指定したオブジェクトまたはインデックスがない場合は、derived_stat は NULL を返します。
- 無効な統計タイプを指定すると、エラー・メッセージが発行されます。
- オプションの *partition_name* または *partition_id* を指定すると、derived_stat はターゲット・パーティションの要求された統計をレポートします。それ以外の場合は、オブジェクト全体の統計をレポートします。
- 引数は次のように使用します。
  - 4 つの引数 － derived_stat は、3 番目の引数をパーティションとして使用し、4 番目の引数に指定された抽出統計を返します。
  - 3 つの引数 － derived_stat は、パーティションの指定が省略されたと解釈し、3 番目の引数に指定された抽出統計を返します。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

テーブルの所有者だけが derived_stat 関数を実行できます。

参照

マニュアル 『パフォーマンス&チューニング・ガイド』の次の項目を参照してください。

- 「1 つのテーブルに対するアクセス・メソッドとクエリ・コスト計算」
- 「統計テーブルおよび optdiag を使った統計の表示」

ユーティリティ optdiag

difference

説明

2 つの soundex 値の差を返します。

構文

difference(*expr1*,*expr2*)

パラメータ

expr1
文字型のカラム名、変数、または (char、varchar、nchar、nvarchar、または unichar 型の) 定数式です。

expr2
もう 1 つの文字型のカラム名、変数、または (char、varchar、nchar、nvarchar、または unichar 型の) 定数式です。

例

例 1

```
select difference("smithers", "smothers")

---------
4
```

例 2

```
select difference("smothers", "brothers")

---------
2
```

使用法

- 文字列関数 difference は 2 つの soundex 値の差を表す整数を返します。
- difference 関数は 2 つの文字列を比較し、この 2 つの文字列の類似点を評価し、0 ～ 4 の値を返します。一致度が最も高い場合の値は 4 です。

  文字列値として、一連の有効な半角または全角のローマ字からなる文字列を指定してください。
- *expr1* または *expr2* が NULL の場合は、NULL を返します。
- varchar 式と unichar 式をそれぞれパラメータとして指定した場合、varchar 式は暗黙的に unichar へ変換されます (変換時にトランケーションが発生することがあります)。
- 文字列関数の一般的な説明については、「文字列関数」(68 ページ) を参照してください。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが difference 関数を実行できます。

参照

関数　soundex

exp

説明

定数を指定した累乗にした値を返します。

構文

exp(*approx_numeric*)

パラメータ

approx_numeric

概数値型 (float、real、または double precision) のカラム名、変数、または定数式のいずれかです。

例

```
select exp(3)

--------------------
            20.085537
```

使用法

- 算術関数 exp は、指定した値の指数値を返します。
- 算術関数の概要については、「算術関数」(67 ページ) を参照してください。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが exp 関数を実行できます。

参照

関数 log, log10, power

floor

説明

指定された値以下の最大の整数を返します。

構文

floor(*numeric*)

パラメータ

numeric

真数値 (numeric、dec、decimal、tinyint、smallint、int、または bigint 型)、概数値 (float、real、または double precision 型)、money 型のカラム、変数、定数式、またはこれらの組み合わせのいずれかです。

例

例 1

```
select floor(123)

-----------
        123
```

例 2

```
select floor(123.45)

-------
    123
```

例 3

```
select floor(1.2345E2)

--------------------
         123.000000
```

例 4

```
select floor(-123.45)

-------
   -124
```

例 5

```
select floor(-1.2345E2)

--------------------
        -124.000000
```

例 6

```
select floor($123.45)

------------------------
                123.00
```

使用法

- 算術関数 floor は、指定した値以下の最大整数を返します。結果は数値式と同じ型になります。

  数値式や 10 進数式の場合、結果は式に等しく、位取りは 0 になります。

- 算術関数の概要については、「算術関数」(67 ページ) を参照してください。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが floor 関数を実行できます。

参照

関数 abs, ceiling, round, sign

get_appcontext

説明

指定されたコンテキストの属性値を返します。get_appcontext は Application Context Facility (ACF) から提供される組み込み関数です。

構文

get_appcontext (*"context_name"*, *"attribute_name"*

パラメータ

context_name

アプリケーション・コンテキスト名を示すローです。データ型 char(30) として保存されています。

attribute_name

アプリケーション・コンテキスト属性名を示すローです。データ型 char(30) として保存されています。

例

例 1 ATTR1 の値として VALUE1 が返されています。

```
select get_appcontext("CONTEXT1", "ATTR1")

--------------
VALUE1
```

ATTR1 は CONTEXT2 にはありません。

```
select get_appcontext("CONTEXT2", "ATTR1")
```

例 2 適切なパーミッションを持たないユーザがアプリケーション・コンテキストを取得しようとしたときの結果を示します。

```
select get_appcontext("CONTEXT1", "ATTR2", "VALUE1")

Select permission denied on built-in get_appcontext, database dbid
-----------
-1
```

使用法

- この関数は成功すると 0 を返し、失敗すると -1 を返します。
- 指定された属性がアプリケーション・コンテキスト内にない場合は、get_appcontext は NULL を返します。
- get_appcontext は、属性を char データ型として格納します。属性値を他のデータ型と比較するアクセス・ルールを作成している場合は、アクセス・ルールで char データを適切なデータ型に変換します。
- この関数では、すべての引数が必須です。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

パーミッションは、ACF に格納されているユーザ・プロファイルとアプリケーション・プロファイルに依存します。

参照

ACF の詳細については、『システム管理ガイド』の「第 17 章 ユーザ・パーミッションの管理」の「ロー・レベル・アクセス制御の使用」を参照してください。

関数　get_appcontext, list_appcontext, rm_appcontext, set_appcontext

getdate

説明 システムの現在の日付と時刻を返します。

構文 getdate()

パラメータ なし。

例

例 1 現在の日付と時刻が 1995 年 11 月 25 日、午前 10 時 32 分だとします。

```
select getdate()

Nov 25 1995 10:32AM
```

例 2 現在の日付が 11 月だとします。

```
select datepart(month, getdate())

11
```

例 3 現在の日付が 11 月だとします。

```
select datename(month, getdate())

November
```

使用法

- 日付関数 getdate は、システムの現在の日付と時刻を返します。
- 日付関数の詳細については、「日付関数」(65 ページ) を参照してください。

規格 ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション すべてのユーザが getdate 関数を実行できます。

参照

データ型 日付と時刻のデータ型

関数 dateadd, datediff, datename, datepart

getutcdate

説明　日時を万国標準時 (UTC) で返します。getutcdate は、ローが挿入または選択されるたびに計算されます。

構文　getutcdate()

例

```
insert t1 (c1, c2, c3) select c1, getutcdate(),
getdate() from t2)
```

参照　**関数**　biginttohex, convert

has_role

説明

指定した役割がユーザに付与されているかどうかを示す情報を返します。

構文

has_role ("*role_name*", *option*)

パラメータ

role_name

システム定義またはユーザ定義の役割の名前です。

option

返される情報の範囲を制限できます。現在サポートされているオプションは 1 だけで、監査を非表示にします。

例

例 1 ユーザがシステム管理者かどうかをチェックするプロシージャを作成します。

```
create procedure sa_check as
if (has_role("sa_role", 0) > 0)
begin
    print "You are a System Administrator."
    return(1)
end
```

例 2 ユーザにシステム・セキュリティ担当者の役割が付与されているかどうかをチェックします。

```
select has_role("sso_role", 1)
```

例 3 ユーザにオペレータの役割が付与されているかどうかをチェックします。

```
select has_role("oper_role", 1)
```

使用法

- has_role は、proc_role と同じように機能します。Adaptive Server バージョン 15.0 から、Sybase は proc_role に代えて has_role をサポートしています。また、この関数の使用が推奨されています。ただし、既存の proc_role をすべて has_role に変換する必要はありません。.
- システム関数 has_role は、指定した役割が呼び出し元ユーザに付与されてアクティブになっているかどうかをチェックします。
- ユーザの役割が次の場合に has_role は 0 を返します。
  - 指定した役割がユーザに付与されていない場合。
  - 指定した役割を含む役割がユーザに付与されていない場合。
  - 指定した役割がユーザに付与されているが、アクティブ化されていない。

- 指定した役割が呼び出し元ユーザに対して付与されており、アクティブ化されている場合、has_role は 1 を返します。
- 呼び出し元ユーザのアクティブな役割に、指定の役割が含まれている場合、has_role は 2 を返します。
- システム関数の概要については、「システム関数」(69 ページ) を参照してください。

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　すべてのユーザが has_role 関数を実行できます。

参照

コマンド　alter role, create role, drop role, grant, set, revoke

関数　mut_excl_roles, role_contain, role_id, role_name, show_role

hash

説明

固定長のハッシュ値式を生成します。

構文

hash(*expression* , [*algorithm*])

パラメータ

expression

ハッシュする値です。カラム名、変数、定数式、またはこれらを任意に組み合わせたものを指定すると、1 つの値が生成されます。image、text、unitext、またはロー状態でない Java データ型を指定することはできません。expression は、通常はカラム名です。expression が文字定数のときは常に引用符で囲みます。

algorithm

ハッシュ値の生成に使用されるアルゴリズムです。md5 または sha1、2 (md5 バイナリを意味する) または 3 (sha1 バイナリを意味する) のいずれかの値を取ることができる文字リテラル (変数名またはカラム名ではない) です。省略すると、md5 が使用されます。

アルゴリズム	結果
hash(*expression*, 'md5')	varchar 32 バイトの文字列。 md5 (メッセージ・ダイジェスト・アルゴリズム 5) は、128 ビット・ハッシュ値を使用した暗号化ハッシュ関数です。
hash(*expression*)	varchar 32 バイトの文字列。
hash(*expression*, 'sha1')	varchar 40 バイトの文字列。 sha1 (セキュア・ハッシュ・アルゴリズム) は、160 ビット・ハッシュ値を使用した暗号化ハッシュ関数です。
hash(*expression*, 2)	varbinary 16 バイト値 (md5 アルゴリズムの使用)
hash(*expression*, 3)	varbinary 20 バイト値 (sha1 アルゴリズムの使用)

例

次の例では、seal の実装方法を示します。「atable」というテーブルと id、sensitive_field、tamper seal カラムの存在。

```
update atable set tamper_seal=hash(convert(varchar(30),
id) + sensitive_field+@salt, 'sha1')
```

使用法

文字リテラルとして指定されている場合、*algorithm* に大文字と小文字の区別はありません。"md5"、"Md5"、"MD5" はすべて同じです。ただし、*expression* が文字データ型として指定されている場合、値に大文字と小文字の区別があります。"Time"、"TIME"、"time" はそれぞれ異なるハッシュ値を生成します。

algorithm が文字リテラルの場合は、varchar 型の文字列が生成されます。"md5" の場合、ハッシュ計算の 128 ビット結果の 16 進数表現を含んでいる 32 バイト文字列です。"sha1" の場合、ハッシュ計算の 160 ビット結果の 16 進数表現を含んでいる 40 バイト文字列です。

algorithm が整数リテラルの場合、varbinary 値が生成されます。2 の場合、ハッシュ計算の 128 ビット結果を含んでいる 16 バイト値が生成されます。3 の場合、ハッシュ計算の 160 ビット結果を含んでいる 20 バイト値が生成されます。

注意　後続の null 値は、varbinary カラムに挿入されるときに Adaptive Server によってトリムされます。

expression を構成する個々のバイトは、メモリ内での表示順でハッシュ・アルゴリズムに渡されます。多くのデータ型について、バイト順序は大きな意味を持ちます。たとえば、4 バイト INT 値 1 のバイナリ表現は MSB-first (ビッグエンディアン) プラットフォームの場合は 0x00、0x00、0x00、0x01 となり、LSB-first (リトルエンディアン) プラットフォームの場合は 0x01、0x00、0x00、0x00 となります。バイト・ストリームはプラットフォーム間で異なるため、ハッシュ値も異なります。プラットフォームに依存しないハッシュ値を生成するには、hashbytes 関数を使用します。

注意　ハッシュ・アルゴリズム MD5 および SHA1 は、完全にセキュアではなくなったという認識が広がっています。したがって、セキュリティが重要なコンテキストで MD5 または SHA1 を使用する場合は、そのリスクに留意してください。

規格　SQL92 および SQL99 対応

パーミッション　すべてのユーザが hash 関数を実行できます。

参照　プラットフォームに依存しないハッシュ値については、「hashbytes」も参照してください。

hashbytes

説明

固定長のハッシュ値式を生成します。

構文

hashbytes(*algorithm*, *expression*[, *expression*.....] [, using *options*])

パラメータ

expression[, *expression*...]

ハッシュする値です。この値には、カラム名、変数、定数式、またはこれらを組み合わせて指定することで、1 つの値を生成できます。image、text、unitext、またはロー状態でない Java データ型を指定することはできません。

algorithm

ハッシュ値の生成に使用されるアルゴリズムです。“md5”、“sha”、“sha1”、“ptn” のいずれかの値を使用できる文字リテラル (変数名やカラム名ではない) です。

アルゴリズム	説明
Md5	メッセージ・ダイジェスト・アルゴリズム 5 は、128 ビットのハッシュ値を使用する暗号化ハッシュ・アルゴリズムです。hashbytes('md5', *expression*[,...]) は varbinary 16 バイト値になります。
Sha-Sha1	セキュア・ハッシュ・アルゴリズムは、160 ビットのハッシュ値を使用する暗号化ハッシュ・アルゴリズムです。hashbytes('sha1', *expression*[,...]) は varbinary 20 バイト値になります。
Ptn	32 ビット・ハッシュ値を使用するパーティション・ハッシュ・アルゴリズムです。‘ptn’ アルゴリズムの場合、*using* 句は無視されます。hashbytes(*'ptn'*, *expression*[,...]) は unsigned int 4 バイト値になります。
using	プラットフォームに関係なく、バイト順序を決定します。オプションの using 句は、次の option 文字列の前に配置できます。 • lsb − すべてのバイト順序依存データは、ハッシュ処理前にリトルエンディアンのバイト順序に正規化されます。 • msb − すべてのバイト順序依存データは、ハッシュ処理前にビッグエンディアンのバイト順序に正規化されます。 • unicode − 文字データは、ハッシュ処理前に unicode (UTF-16) に正規化されます。 **注意** UTF - 16 文字列は、単精度整数の配列と似た文字列です。この文字列はバイト順序に依存するので、プラットフォーム独立性を確保するために、lsb または msb を UNICODE と一緒に使用することをおすすめします。 • unicode_lsb − unicode と lsb の組み合わせ • unicode_msb − unicode と msb の組み合わせ

例

例 1 テーブルの各ローを改ざんから守ります。次の例では、“xtable” および col1、col2、col3、tamper_seal というユーザ・テーブルが存在すると想定します。

```
update xtable set tamper_seal=hashbytes('sha1', col1,
col2, col4, @salt)
--
declare @nparts unsigned int
select @nparts= 5
select hashbytes('ptn', col1, col2, col3) % nparts from xtable
```

例 2 col1、col2、col3 がローを使用して 5 つのパーティションに分割する方法を示します。

```
alter table xtable partition by hash(col1, col2, col3) 5
```

使用法

algorithm パラメータに大文字と小文字の区別はありません。“md5”、“Md5”、“MD5” はすべて同じです。ただし、*expression* が文字データ型として指定されている場合は、値に大文字と小文字の区別があります。“Time”、“TIME”、“time” はそれぞれ異なるハッシュ値を生成します。

注意 後続の null 値は、varbinary カラムに挿入されるときに Adaptive Server によってトリムされます。

---

using 句を使用しない場合、*expression* を構成するバイトは、メモリ内での表示順でハッシュ・アルゴリズムに渡されます。多くのデータ型について、バイト順序は大きな意味を持ちます。たとえば、4 バイト INT 値 1 のバイナリ表現は MSB-first (ビッグエンディアン) プラットフォームの場合は 0x00、0x00、0x00、0x01 となり、LSB-first (リトルエンディアン) プラットフォームの場合は 0x01、0x00、0x00、0x00 となります。バイト・ストリームはプラットフォームごとに異なるので、ハッシュ値も異なります。

using 句を使用した場合、*expression* を構成するバイトは、プラットフォームに関係なくハッシュ・アルゴリズムに渡されます。また、using 句を使用すると、Unicode に文字データを変換できるので、ハッシュ値はサーバの文字設定に依存しなくなります。

注意 ハッシュ・アルゴリズム MD5 および SHA1 は、完全にセキュアではなくなったという認識が広がっています。MD5 または SHA1 をセキュリティが重要なコンテキストで使用する場合は、そのリスクに留意してください。

規格

SQL92 および SQL99 対応

パーミッション

すべてのユーザが hashbyte 関数を実行できます。

参照

プラットフォームに依存するハッシュ値については、「hash」も参照してください。

hextobigint

説明

16 進文字列に相当する bigint 値を返します

構文

hextobigint(*hexadecimal_string*)

パラメータ

hexadecimal_string

big integer に変換される 16 進数値です。文字型カラム、変数名、または有効な 16 進文字列 (“0x” プレフィクスの有無に関係なく引用符で囲む) のいずれかである必要があります。

例

次の例は、0x7fffffffffffffff という 16 進文字列を bigint 値に変換します。

```
1> select hextobigint("0x7fffffffffffffff")
2> go
--------------------
9223372036854775807
```

使用法

- データ型変換関数 hextobigint は、16 進文字列に相当する、プラットフォームの影響を受けない整数を返します。
- hextobigint 関数は、プラットフォームの影響を受けずに 16 進データを整数に変換するために使用します。hextobigint は、有効な 16 進文字列 (“0x” プレフィクスの有無に関係なく引用符で囲む)、文字型のカラム名、または変数名を受け取ります。

  hextobigint は、16 進文字列に相当する bigint を返します。この関数は、実行するプラットフォームに関係なく、16 進文字列に相当する同じ bigint を常に返します。

参照

関数 biginttohex, convert, inttohex, hextoint

hextoint

説明

指定した 16 進文字列に相当する、プラットフォームの影響を受けない整数値を返します。

構文

hextoint(*hexadecimal_string*)

パラメータ

hexadecimal_string

整数に変換される 16 進数値です。文字型カラム、変数名、または有効な 16 進文字列 (“0x” プレフィクスの有無に関係なく引用符で囲む) のいずれかである必要があります。

例

16 進文字列 “0x00000100” に相当する整数値を返します。この関数が実行されるプラットフォームに関係なく、結果は常に 256 です。

```
select hextoint("0x00000100")
```

使用法

- データ型変換関数 hextoint は、16 進文字列に相当する、プラットフォームの影響を受けない整数を返します。
- hextoint 関数は、プラットフォームの影響を受けずに 16 進数データを整数に変換するために使用します。hextoint は、有効な 16 進文字列 (“0x” プレフィクスの有無に関係なく引用符で囲む)、文字型カラム名、または変数名を受け取ります。

  hextoint 関数は 16 進文字列に相当する整数値を返します。この関数は、実行するプラットフォームに関係なく、16 進文字列に相当する同一整数値を常に返します。
- データ型変換の詳細については、「データ型変換関数」(57 ページ) を参照してください。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが hextoint 関数を実行できます。

参照

関数　biginttohex, convert, inttohex

host_id

説明

クライアント・コンピュータのオペレーティング・システムで使用されている、現在の Adaptive Server クライアントのプロセス ID を返します。

構文

host_id()

パラメータ

なし。

例

この例では、クライアント・コンピュータの名前が “ephemeris” で、コンピュータ “ephemeris” で Adaptive Server クライアント・プロセスに使用されているプロセス ID が 2309 です。

```
select host_name(), host_id()
---------------------------- ----------------------
 ephemeris                   2309
```

次は、コンピュータ “ephemeris” 上で、UNIX の ps コマンドを使用して収集したプロセス情報です。この例のクライアントは “isql” であり、プロセス ID が 2309 であることを示しています。

```
2309 pts/2    S  0:00 /work/as125/OCS-12_5/bin/isql
```

使用法

- システム関数 host_id は、(サーバ・プロセスではなく) クライアント・プロセスのプロセス ID を返します。
- システム関数の概要については、「文字列関数」(68 ページ) を参照してください。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが host_id 関数を実行できます。

参照

関数 host_name

host_name

説明　クライアント・プロセスの現在のホスト・コンピュータ名を返します。

構文　host_name()

パラメータ　なし。

例

```
select host_name()

------------------------------
violet
```

使用法

- システム関数 host_name は、(サーバ・プロセスではなく) クライアント・プロセスの現在のホスト・コンピュータ名を返します。
- システム関数の概要については、「システム関数」(69 ページ) を参照してください。

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　すべてのユーザが host_name 関数を実行できます。

参照　**関数**　host_id

instance_id

説明

クラスタ環境のみ － 指定したインスタンスの ID、または *name* の値を指定しない場合は、発行元のインスタンスの ID を返します。

構文

instance_id([*name*])

パラメータ

name

ID を調べるキャッシュのインスタンス名です。

例

ローカル・インスタンスの ID を返します。

```
select instance_id()
```

使用法

“myserver1” というインスタンスの ID を返します。

```
select instance_id(myserver1)
```

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが instance_id を実行できます。

identity_burn_max

説明　指定したテーブルの破棄される identity 値の最大値を追跡します。この関数は値を返すだけで、更新は実行しません。

構文　identity_burn_max(*table_name*)

パラメータ　*table_name*
選択したテーブルの名前です。

例

```
select identity_burn_max("t1")

t1
------------
51
```

使用法　identity_burn_max は、指定したテーブルの破棄される identity 値の最大値を追跡します。

パーミッション　テーブル所有者、システム管理者、データベース管理者のみがこのコマンドを発行できます。

index_col

説明

指定されたテーブルまたはビューにあるインデックス付きのカラム名を返します。カラム名は、最長で 255 バイトです。

構文

index_col(*object_name*, *index_id*, *key_#*[, *user_id*])

パラメータ

object_name
テーブルまたはビューの名前です。完全修飾名を指定できます (つまり、データベースと所有者名が含まれている名前を指定できます)。名前は引用符で囲んでください。

index_id
object_name のインデックス番号です。この番号は sysindexes.indid の値と同じです。

key_#
インデックス内のキーです。この値は、クラスタード・インデックスでは 1 と sysindexes.keycnt の間、ノンクラスタード・インデックスでは 1 と sysindexes.keycnt+1 の間の値です。

user_id
object_name の所有者です。*user_id* を指定しないと、デフォルトの呼び出し元のユーザ ID になります。

例

テーブル t4 のクラスタード・インデックス内のキーの名前を検索します。

```
declare @keycnt integer
select @keycnt = keycnt from sysindexes
     where id = object_id("t4")
     and indid = 1
while @keycnt > 0
begin
     select index_col("t4", 1, @keycnt)
     select @keycnt = @keycnt - 1
end
```

使用法

- システム関数 index_col は、インデックス・カラムの名前を返します。
- *object_name* がテーブル名またはビュー名でない場合、index_col 関数は NULL を返します。
- システム関数の概要については、「文字列関数」(68 ページ) を参照してください。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが index_col 関数を実行できます。

参照

関数 object_id

システム・プロシージャ sp_helpindex

index_colorder

説明　カラム順を返します。

構文　index_colorder(*object_name*, *index_id*, *key_#*[, *user_id*])

パラメータ

object_name
テーブルまたはビューの名前です。完全修飾名を指定できます (つまり、データベースと所有者名が含まれている名前を指定できます)。名前は引用符で囲んでください。

index_id
object_name のインデックス番号です。この番号は sysindexes.indid の値と同じです。

key_#
インデックス内のキーです。有効な値は 1 と、インデックス内のキーの数です。キーの数は、sysindexes.keycnt に格納されます。

user_id
object_name の所有者です。*user_id* を指定しないと、デフォルトの呼び出し元のユーザ ID になります。

例　sales テーブルの salesind インデックスは降順であるため、“DESC” を返します。

```
select name, index_colorder("sales", indid, 2)
from sysindexes
where id = object_id ("sales")
and indid > 0

name
------------------------- -------------------------
salesind                  DESC
```

使用法

- システム関数 index_colorder は、昇順のカラムの場合に “ASC”、降順のカラムの場合に “DESC” を返します。
- *object_name* がテーブル名ではないか、*key_#* が有効なキー番号ではない場合、index_colorder は NULL を返します。
- システム関数の概要については、「文字列関数」(68 ページ) を参照してください。

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　すべてのユーザが index_colorder 関数を実行できます。

index_name

説明

インデックス ID、データベース ID、およびインデックスが定義されているオブジェクトを指定すると、インデックスの名前を返します。

構文

index_name(*dbid*, *objid*, *indid*)

パラメータ

dbid

インデックスを定義するデータベースの ID です。

objid

インデックスを定義するテーブル (指定したデータベース内) の ID です。

indid

名前を知りたいインデックスの ID です。

例

例 1

この関数の通常の使用法を示します。

```
select index_name(db_id("testdb"),
    object_id("testdb..tab_apl"),1)
------------
```

例 2 データベース ID が NULL で現在のデータベース ID を使用する場合の出力を示します。

```
select index_name(NULL,object_id("testdb..tab_apl"),1)
--------------------------
```

例 3 インデックス ID が 0 で、データベース ID とオブジェクト ID が有効な場合は、テーブル名が表示されます。

```
select index_name(db_id("testdb"),
    object_id("testdb..tab_apl"),0)
-----------------------------
```

使用法

- *dbid* パラメータで NULL の値を渡した場合、index_name は現在のデータベース ID を使用します。
- *dbid* パラメータで NULL の値を渡した場合、index_name は NULL を返します。
- インデックス ID が 0 で、オブジェクト ID とデータベース ID に有効な入力値を渡した場合、index_name はオブジェクト名を返します。

パーミッション

この関数は、すべてのユーザが実行できます。

参照

db_id, object_id

inttohex

説明

指定した整数値に相当する、プラットフォームの影響を受けない 16 進文字列を返します。

構文

inttohex(*integer_expression*)

パラメータ

integer_expression
16 進文字列に変換される整数値です。

例

```
select inttohex (10)

--------
0000000A
```

使用法

- データ型変換関数 inttohex は、指定された整数値に相当する、プラットフォームの影響を受けない 16 進文字列を、プレフィクス “0x” を付けないで返します。
- inttohex 関数は、プラットフォームの影響を受けずに整数を 16 進文字列に変換するために使用します。inttohex は、整数に評価される任意の式を受け入れます。実行するプラットフォームに関係なく、指定の式と同じ 16 進数の値を常に返します。
- データ型変換の詳細については、「データ型変換関数」(57 ページ) を参照してください。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが inttohex 関数を実行できます。

参照

関数　convert, hextobigint, hextoint

isdate

説明

入力式が、有効な datetime 値であるかどうかを確認します。

構文

isdate(*character_expression*)

パラメータ

character_expression

文字型の変数、定数式、またはカラム名です。

例

例 1 文字列 12/21/2005 が有効な datetime 値かどうかを確認するには、次のように入力します。

```
select isdate('12/21/2005')
```

例 2 sales テーブル内の stor_id と date が有効な datetime 値かどうかを確認するには、次のように入力します。

```
select isdate(stor_id), isdate(date) from sales
---- ----
0    1
```

store_id は有効な datetime 値ではありませんが、date は有効です。

使用法

式が有効な datetime 値である場合は 1 を返し、有効でない場合は 0 を返します。NULL の入力の場合は 0 を返します。

isnumeric

説明

式が有効な numeric データ型かどうかを確認します。

構文

isnumeric (*character_expression*)

パラメータ

character_expression
文字型の変数、定数式、またはカラム名です。

例

例 1 authors テーブルの postalcode カラムの値に有効な numeric データ型が含まれているかどうか確認するには、次のように入力します。

```
select isnumeric(postalcode) from authors
```

例 2 値 $100.12345 が有効な numeric データ型かどうか確認するには、次のように入力します。

```
select isnumeric("$100.12345")
```

使用法

- 入力式が有効な整数型、浮動小数点型、通貨型、10 進数型のいずれかである場合は 1 を返し、有効でない場合または入力が NULL 値の場合は 0 を返します。戻り値が 1 の場合は、式をこれらの numeric 型のいずれかに変換できます。
- 入力には通貨記号を含めることができます。

is_quiesced

説明

データベースが quiesce database モードにあるかどうかを示します。is_quiesced は、データベースがクワイス (静止) 状態の場合に 1、クワイス状態でない場合に 0 を返します。

構文

is_quiesced(*dbid*)

パラメータ

dbid

データベースのデータベース ID です。

例

例 1 test データベースを使用します。データベース ID は 4 で、クワイス状態ではありません。

```
1> select is_quiesced(4)
2> go

-----------
          0

(1 row affected)
```

例 2 quiesce database を実行してアクティビティをサスペンドしたあとで、test データベースを使用します。

```
1> quiesce database tst hold test
2> go
1> select is_quiesced(4)
2> go

-----------
          1

(1 row affected)
```

例 3 quiesce database を使用してアクティビティを再開したあとで、test データベースを使用します。

```
1> quiesce database tst release
2> go
1> select is_quiesced(4)
2> go

 -----------
           0

(1 row affected)
```

例 4 無効なデータベース ID を使用する is_quiesced を指定して select 文を実行します。

```
1>select is_quiesced(-5)
2> go

 ------------
       NULL

(1 row affected)
```

使用法

- is_quiesced にデフォルト値はありません。データベースを指定しないで is_quiesced 関数を実行すると、エラーになります。
- is_quiesced は、存在しないデータベース ID が指定された場合に NULL を返します。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

is_quiesced は、すべてのユーザが実行できます。

参照

コマンド　quiesce database

is_sec_service_on

説明

セキュリティ・サービスが有効な場合は 1 を返し、無効な場合は 0 を返します。

構文

is_sec_service_on(*security_service_nm*)

パラメータ

security_service_nm

セキュリティ・サービスの名前です。

例

```
select is_sec_service_on("unifiedlogin")
```

使用法

- セッション中に、特定のセキュリティ・サービスが有効かどうかを調べるには、is_sec_service_on を使用します。
- セキュリティ・サービスの有効な名前を調べるには、次のクエリを実行します。

  ```
  select * from syssecmechs
  ```

  次のような結果が出力されます。

  ```
  sec_mech_name available_service
  ------------ ------------------
  dce          unifiedlogin
  dce          mutualauth
  dce          delegation
  dce          integrity
  dce          confidentiality
  dce          detectreplay
  dce          detectseq
  ```

  available_service カラムには、Adaptive Server がサポートするセキュリティ・サービスが表示されます。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが is_sec_service_on 関数を実行できます。

参照

関数 show_sec_services

isnull

説明　*expression1* の値が NULL のとき、*expression2* に指定した値と置き換えます。

構文　isnull(*expression1*, *expression2*)

パラメータ

expression

カラム名、変数、定数式、またはこれらを任意に組み合わせたもので 1 つの値を計算するものです。unichar など、任意のデータ型を使用できます。通常は *expression* にはカラム名を指定します。*expression* が文字定数のときは常に引用符で囲みます。

例　price の null 値を 0 に置き換えて、titles テーブルのすべてのローを返します。

```
select isnull(price,0)
  from titles
```

使用法

- システム関数 isnull は、*expression1* の値が NULL の場合、*expression2* に指定した値と置き換えます。システム関数の概要については、「文字列関数」(68 ページ) を参照してください。
- 式のデータ型が暗黙的に変換されるか、または convert 関数を使用する必要があります。

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　すべてのユーザが isnull 関数を実行できます。

参照　**関数**　convert

isnumeric

説明

式が有効な numeric データ型かどうかを確認します。

構文

isnumeric (*character_expression*)

パラメータ

character_expression

文字型の変数、定数式、またはカラム名です。

例

例 1 authors テーブルの postalcode カラムの値に有効な numeric データ型が含まれているかどうか確認するには、次のように入力します。

```
select isnumeric(postalcode) from authors
```

例 2 値 $100.12345 が有効な numeric データ型かどうか確認するには、次のように入力します。

```
select isnumeric("$100.12345")
```

使用法

- 入力式が有効な整数型、浮動小数点型、通貨型、10 進数型のいずれかである場合は 1 を返し、有効でない場合または入力が NULL 値の場合は 0 を返します。戻り値が 1 の場合は、式をこれらの numeric 型のいずれかに変換できます。
- 入力には通貨記号を含めることができます。

instance_name

説明　クラスタ環境のみ － 指定した ID の Adaptive Server の名前、または *id* の値を指定しない場合は、発行元の Adaptive Server の名前を返します。

構文　instance_name([*id*])

パラメータ　*id*

調べる Adaptive Server の ID です。

例　データベース ID が 12 のインスタンスの名前を返します。

```
select instance_name(12)
```

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　すべてのユーザが instance_name 関数を実行できます。

lc_id

説明

クラスタ環境のみ － 指定した名前の論理クラスタの ID、または名前を指定しない場合は現在の論理クラスタの ID を返します。

構文

lc_id(*logical_cluster_name*)

パラメータ

logical_cluster_name
論理クラスタの名前です。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが lc_id を実行できます。

lc_name

説明　クラスタ環境のみ − 指定した ID の論理クラスタの名前、または ID を指定しない場合は現在の論理クラスタを返します。

構文　lc_name([*logical_cluster_ID*])

パラメータ　*logical_cluster_ID*
論理クラスタの ID です。

規格　ANSI SQL − 準拠レベル：Transact-SQL 拡張機能

パーミッション　すべてのユーザが lc_name 関数を実行できます。

lct_admin

説明

ラストチャンス・スレッショルド (LCT) を管理する関数です。LCT の現在の値をかえしたり、LCT に達したトランザクションをアボートしたりします。

構文

```
lct_admin({{"lastchance" | "logfull" | "reserved_for_rollbacks"},
        database_id
        |"reserve", {log_pages | 0 }
        | "abort", process-id [, database-id]})
```

パラメータ

lastchance
指定されたデータベースに LCT を作成します。

logfull
指定されたデータベースが LCT を超えた場合には 1 を返し、超えていない場合には 0 を返します。

reserved_for_rollbacks
データベースがロールバック用に現在確保しているページ数を判断します。

database_id
データベースを指定します。

reserve
LCT の現在の値、または指定サイズのトランザクション・ログのダンプに必要なログ・ページ数を取得します。

log_pages
LCT を決定するために使用されるページ数です。

0
LCT の現在の値を返します。独立したログとデータ・セグメントを持つデータベース内の LCT のサイズは、動的に変化しません。この値は、トランザクション・ログのサイズに基づいて固定しています。混合ログとデータ・セグメントを持つデータベースでは、LCT は動的に変化します。

abort
トランザクション・ログがラストチャンス・スレッショルドに達したデータベース内のトランザクションをアボートします。ログ中断モードのトランザクションだけをアボートできます。

logsegment_freepages
ログ・セグメント用に使用可能な空き領域を示します。これは、空き領域の合計値であり、ディスクごとの値ではありません。

process-id
ログ中断モードのプロセスの ID (*spid*) です。ラストチャンス・スレッショルド (LCT) に達したトランザクション・ログ内にオープン・トランザクションがある場合、プロセスはログ中断モードになります。

database-id
トランザクション・ログが LCT に達したデータベースの ID です。*process-id* が 0 の場合は、指定データベース内のオープン・トランザクションすべてが終了します。

例

例 1 dbid が 1 のデータベースのログ・セグメントのラストチャンス・スレッショルドが作成されます。新しいスレッショルドが作成されたページの数を返します。前回のラストチャンス・スレッショルドが検出されると、置き換えられます。

```
select lct_admin("lastchance", 1)
```

例 2 dbid が 6 のデータベースのラストチャンス・スレッショルドを超えている場合は 1 を返し、そうでない場合は 0 を返します。

```
select lct_admin("logfull", 6)
```

例 3 64 ページのトランザクション・ログをダンプするために必要となるログ・ページ数を計算して返します。

```
select lct_admin("reserve", 64)

----------
          16
```

例 4 コマンドの発行元であるデータベース内のトランザクション・ログの、現在のラストチャンス・スレッショルドを返します。

```
select lct_admin("reserve", 0)
```

例 5 プロセス 83 に属するトランザクションをアボートします。このプロセスは、ログ中断モードである必要があります。LCT に達したトランザクション・ログ内のトランザクションだけが終了します。

```
select lct_admin("abort", 83)
```

例 6 dbid が 5 のデータベース内にあるすべてのオープン・トランザクションをアボートします。これにより、ログ・セグメントのラストチャンス・スレッショルドで中断されている可能性のあるプロセスがすべて起動します。

```
select lct_admin("abort", 0, 5)
```

例 7 dbid が 5 である、pubs2 データベース内でロールバック用に確保されているページ数を判断します。

```
select lct_admin("reserved_for_rollbacks", 5, 0)
```

例 8 dbid が 4 のデータベースで使用可能な空き領域を示します。

```
select lct_admin("logsegment_freepages", 4)
```

使用法

- システム関数 lct_admin は、ログ・セグメントのラストチャンス・スレッショルドを管理します。システム関数の概要については、「システム関数」(69 ページ) を参照してください。
- lct_admin(“lastchance”, *dbid*) が 0 を返す場合、ログはこのデータベース内の別のセグメントにはないため、ラストチャンス・スレッショルドは存在しません。
- 別のログ・セグメントを持つデータベースを作成すると、サーバでは常に、sp_thresholdaction をデフォルトで呼び出すラストラヤンス・スレッショルドが作成されます。サーバ上に sp_thresholdaction プロシージャがない場合にも、このようにラストチャンス・スレッショルドが作成されます。

  ログがラストチャンス・スレッショルドを超えると、Adaptive Server は活動を中断して sp_thresholdaction を呼び出そうとします。このプロシージャが存在していないことが判明すると、エラーを生成し、ログがトランケート可能になるまでプロセスを中断します。
- LCT に達したトランザクション・ログ内で最も古いオープン・トランザクションを終了するには、そのトランザクションを開始したプロセスの ID を入力します。
- LCT に達したトランザクション・ログ内のすべてのオープン・トランザクションを終了するには、*process-id* として 0 を入力し、*database-id* パラメータにデータベース ID を指定します。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

システム管理者だけが lct_admin abort を実行できます。すべてのユーザはその他の lct_admin オプションを実行できます。

参照

マニュアル 『システム管理ガイド』を参照してください。

コマンド dump transaction

関数 curunreservedpgs

システム・プロシージャ sp_thresholdaction

left

説明　文字列の左端から指定された数の文字を返します。

構文　left(*character_expression*, *integer_expression*)

パラメータ

character_expression
左側にある文字が選択される文字列です。

integer_expression
返される文字数を指定する正の整数値です。*integer_expression* が負の場合は、エラーが返されます。

例

例 1 各本のタイトルの左端の 5 文字を返します。

```
use pubs
select left(title, 5) from titles
order by title_id

-----
The B
Cooki
You C
.....
Sushi

(18 row(s) affected)
```

例 2 文字列 "abcdef" の左端の 2 文字を返します。

```
select left("abcdef", 2)
---------
ab
(1 row(s) affected)
```

使用法

- *character_expression* には、varchar または nvarchar への暗黙的な変換が可能な任意のデータ型 (text と image を除く) を使用できます。*character_expression* は、定数、変数、またはカラム名にできます。character_expression は、convert を使用して明示的に変換できます。
- left は substring(*character_expression*, *1*, *integer_expression*) と同じです。この関数の詳細については、substring (281 ページ) を参照してください。

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　すべてのユーザが left 関数を実行できます。

参照

データ型　varchar、nvarchar

関数　len, str_replace, substring

len

説明

後続ブランクを除き、指定した文字列式の文字数 (バイト数ではない) を返します。

構文

len(*string_expression*)

パラメータ

string_expression
評価する文字列式です。

例

title_id エントリ "PC9999" の文字数を返します。

```
select len(notes) from titles
where title_id = "PC9999"
-----------
39
```

使用法

この関数は char_length(*string_expression*) と同じです。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが len 関数を実行できます。

参照

データ型 char、nchar、varchar、または nvarchar

関数 char_length, left, str_replace

license_enabled

説明

機能のライセンスが有効になっている場合は 1 を返し、有効になっていない場合は 0 を返します。また、無効なライセンス名が指定された場合は NULL を返します。

構文

```
license_enabled("ase_server" | "ase_ha" | "ase_dtm" | "ase_java" |
        "ase_asm")
```

パラメータ

ase_server
Adaptive Server のライセンスを指定します。

ase_ha
Adaptive Server の高可用性機能のライセンスを指定します。

ase_dtm
Adaptive Server 分散トランザクション管理機能のライセンスを指定します。

ase_java
Adaptive Server 機能の Java のライセンスを指定します。

ase_asm
Adaptive Server の高度セキュリティ・メカニズムのライセンスを指定します。

例

Adaptive Server 分散トランザクション管理機能が有効であることを示します。

```
select license_enabled("ase_dtm")

-------------
            1
```

使用法

- Adaptive Server 機能のライセンス・キーをインストールする方法については、『インストール・ガイド』を参照してください。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが license_enabled 関数を実行できます。

参照

マニュアル　使用しているプラットフォームの『インストール・ガイド』

システム・プロシージャ　sp_configure

list_appcontext

説明

現在のセッション内にある全コンテキストの属性をすべてリストします。list_appcontext は Application Context Facility (ACF) から提供される組み込み関数です。

構文

list_appcontext(["*context_name*"])

パラメータ

context_name

セッション内にあるすべてのアプリケーション・コンテキストの属性を指定するときに使用するオプションの引数です。

例

例 1 適切なパーミッションを持つユーザがアプリケーション・コンテキストをリストしようとしたときの結果を示します。

```
select list_appcontext ([context_name])

Context Name: (CONTEXT1)
Attribute Name: (ATTR1) Value: (VALUE2)
Context Name: (CONTEXT2)
Attribute Name: (ATTR1) Value: (VALUE1)
```

例 2 適切なパーミッションを持たないユーザがアプリケーション・コンテキストをリストしようとしたときの結果を示します。

```
select list_appcontext()

Select permission denied on built-in list_appcontext, database
DBID
--------
-1
```

使用法

- この関数は成功すると 0 を返します。
- 組み込み関数が複数の結果セットを返すことはないので、クライアント・アプリケーションは list_appcontext の戻り値をメッセージとして受け取ります。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

パーミッションは、ACF に格納されているユーザ・プロファイルとアプリケーション・プロファイルに依存します。

参照

ACF の詳細については、『システム管理ガイド』の「第 17 章 ユーザ・パーミッションの管理」の「ロー・レベル・アクセス制御の使用」を参照してください。

関数 get_appcontext, list_appcontext, rm_appcontext, set_appcontext

lockscheme

説明

指定のオブジェクトのロック・スキームを文字列で返します。

構文

lockscheme(*object_name*)
lockscheme(*object_id* [, *db_id*])

パラメータ

object_name
ロック・スキームを返すオブジェクトの名前です。*object_name* には完全修飾名も指定できます。

db_id
object_id によって指定されたデータベースの ID です。

object_id
ロック・スキームを返すオブジェクトの ID です。

例

例 1 現在のデータベースの titles テーブルのロック・スキームを返します。

```
select lockscheme("titles")
```

例 2 データベース ID が 4 のデータベース (pubs2 データベース) の *object_id* 224000798 のオブジェクト (この例では titles テーブル) のロック・スキームを返します。

```
select lockscheme(224000798, 4)
```

例 3 titles テーブルのロック・スキームを返します (この例では、*object_name* には完全修飾名を指定しています)。

```
select lockscheme(tempdb.ownerjoe.titles)
```

使用法

- lockscheme は varchar(11) を返します。このコマンドでは、NULL を使用できます。
- 次の場合は、lockscheme のデフォルトは現在のデータベースになります。
  - 完全に修飾された *object_name* を指定していない。
  - *db_id* を指定していない。
  - *db_id* に null を指定した。
- 指定されたオブジェクトがテーブルでない場合は、lockscheme は文字列 “not a table.” を返します。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが lockscheme 関数を実行できます。

log

説明

指定した数値の自然対数を返します。

構文

log(*approx_numeric*)

パラメータ

approx_numeric

概数値型 (float、real、または double precision) のカラム名、変数、または定数式のいずれかです。

例

```
select log(20)

--------------------
              2.995732
```

使用法

- 算術関数 log は、指定した値の自然対数を返します。
- 算術関数の概要については、「算術関数」(67 ページ) を参照してください。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが log 関数を実行できます。

参照

関数 log10, power

log10

説明　指定した数の常用対数 (底が 10 の対数) を返します。

構文　log10(*approx_numeric*)

パラメータ　*approx_numeric*

概数値型 (float、real、または double precision) のカラム名、変数、または定数式のいずれかです。

例

```
select log10(20)

--------------------
          1.301030
```

使用法

- 算術関数 log10 は、指定した値の常用対数 (底が 10 の対数) を返します。
- 算術関数の概要については、「算術関数」(67 ページ) を参照してください。

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　すべてのユーザが log10 関数を実行できます。

参照　**関数**　log, power

lower

説明

指定した式に相当する小文字の文字値を返します。

構文

lower(*char_expr* | *uchar_expr*)

パラメータ

char_expr
文字型のカラム名、変数、または (char、varchar、nchar、または nvarchar 型の) 定数式です。

uchar_expr
文字型のカラム名、変数、または (unichar または univarchar 型の) 定数式です。

例

```
select lower(city) from publishers

--------------------
boston
washington
berkeley
```

使用法

- 文字列関数 lower は大文字を小文字に変換して、文字値を返します。
- lower 関数は upper 関数の逆変換です。
- *char_expr* または *uchar_expr* が NULL の場合は、NULL を返します。
- 文字列関数の一般的な説明については、「文字列関数」(68 ページ) を参照してください。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが lower 関数を実行できます。

参照

関数 upper

ltrim

説明　指定した式から先行ブランクを削除して返します。

構文　ltrim(*char_expr* | *uchar_expr*)

パラメータ

char_expr
文字型のカラム名、変数、または (char、varchar、nchar、または nvarchar 型の) 定数式です。

uchar_expr
文字型のカラム名、変数、または (unichar または univarchar 型の) 定数式です。

例

```
select ltrim("    123")

-------
123
```

使用法

- 文字列関数ltrim は文字式から先行ブランクを削除します。現在の文字セットのスペース文字に相当する値だけを削除します。
- *char_expr* または *uchar_expr* が NULL の場合は、NULL を返します。
- Unicode 式の場合は、指定した式に相当する小文字の Unicode 文字列を返します。指定した式の文字のうち、対応する小文字がない文字は変換されません。

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　すべてのユーザが ltrim 関数を実行できます。

参照　文字列関数の概要については、「文字列関数」(68 ページ) を参照してください。

関数　rtrim

max

説明

式内の最大値を返します。

構文

max(*expression*)

パラメータ

expression
カラム名、定数、関数、これらを算術演算子やビット処理演算子でつないだ任意の組み合わせ、またはサブクエリです。

例

例 1 salesdetail テーブルの discount カラム内の最大値を返します。

```
select max(discount) from salesdetail

------------------
              62.200000
```

例 2 salesdetail テーブルの discount カラム内の最大値を返します。

```
select discount from salesdetail
compute max(discount)
```

使用法

- 集合関数 max はカラムまたは式の最大値を返します。集合関数の概要については、「集合関数」(48 ページ) を参照してください。
- max 関数は、真数値、概数値、文字、datetime のカラムに対して使用できます。bit カラムに対しては使用できません。文字型のカラムの場合、max は照合順の最大値を返します。max は、null 値を無視します。max 関数は、char データ型を varchar に、unichar データ型を univarchar に暗黙的に変換し、後続ブランクをすべて削除します。
- unichar データは、デフォルトの Unicode ソート順序に従って照合されます。
- 集合カラムにインデックスがあるときには、次に示す状況の場合を除き、Adaptive Server はインデックスの終わりに直接移動して max の最終ローを検索します。
  - *expression* がカラムでない場合。
  - カラムがインデックスの先頭カラムでない場合。
  - クエリに別の集合関数がある場合。
  - group by 句または where 句がある場合。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが max 関数を実行できます。

参照

コマンド compute 句、group 句と having 句、select、where 句

関数 avg, min

min

説明　カラム内の最小値を返します。

構文　min(*expression*)

パラメータ

expression

カラム名、定数、関数、これらを算術演算子やビット処理演算子でつないだ任意の組み合わせ、またはサブクエリです。通常、集合関数では、expression はカラム名です。詳細については、「式」(341 ページ) を参照してください。

例

```
select min(price) from titles
    where type = "psychology"

----------------------
                  7.00
```

使用法

- 集合関数 min はカラム内の最小値を返します。
- 集合関数の概要については、「集合関数」(48 ページ) を参照してください。
- min 関数は、数値、文字、time カラム、datetime カラムに対して使用できます。bit カラムに対しては使用できません。文字型のカラムの場合、min は、ソート順での最小値を返します。min 関数は、char データ型を varchar に、unichar データ型を univarchar に暗黙的に変換し、後続ブランクをすべて削除します。min は、null 値を無視します。distinct は意味がないため、min では使用できません。
- unichar データは、デフォルトの Unicode ソート順序に従って照合されます。
- 集合カラムにインデックスがあるときは、次に示す状況の場合を除き、Adaptive Server は min 関数の条件を満たす最初のローに直接移動します。
  - *expression* がカラムでない場合。
  - カラムがインデックスの先頭カラムでない場合。
  - クエリに別の集合関数がある場合。
  - group by 句がある場合。

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　すべてのユーザが min 関数を実行できます。

参照

コマンド　compute 句、group 句と having 句、select、where 句

関数　avg, max

month

説明

指定した日付の datepart の月を表す整数を返します。

構文

month(*date_expression*)

パラメータ

date_expression

datetime、smalldatetime、date 型の式、または datetime フォーマットの文字列です。

例

整数 11 を返します。

```
day("11/02/03")
----------
11
```

使用法

month(*date_expression*) は datepart(*mm*, *date_expression*) と同じです。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが month 関数を実行できます。

参照

データ型 datetime、smalldatetime、date

関数 datepart, day, year

mut_excl_roles

説明　2 つの役割の間の相互排他性についての情報を返します。

構文　mut_excl_roles (*role1, role2* [membership | activation])

パラメータ

role1
相互排他性の関係にある、ユーザ定義の役割です。

role2
相互排他性の関係にある、もう 1 つのユーザ定義の役割です。

level
指定した役割が排他関係になるレベル (membership または activation) です。

例　admin と supervisor の役割がお互いに排他関係であることを示しています。

```
alter role admin add exclusive membership supervisor
select
mut_excl_roles("admin", "supervisor", "membership")

----------
          1
```

使用法

- システム関数 mut_excl_roles は、2 つの役割の間の相互排他性についての情報を返します。システム・セキュリティ担当者が、role1 を role2 または role2 に直接含まれている役割と相互に排他的であると定義すると、mut_excl_roles は 1 を返します。これらの役割が相互に排他的ではないときは、mut_excl_roles は 0 を返します。

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　すべてのユーザが mut_excl_roles 関数を実行できます。

参照　システム関数の概要については、「システム関数」(69 ページ) を参照してください。

コマンド　alter role, create role, drop role, grant, set, revoke

関数　proc_role, role_contain, role_id, role_name

システム・プロシージャ　sp_activeroles, sp_displayroles, sp_role

newid

説明

指定した引数に基づいて、人間が判読できるグローバルにユニークな識別子 (GUID) を 2 とおりの異なるフォーマットで生成します。人間が判読できる GUID 値のフォーマットの長さは、32 バイト (ダッシュなし) または 36 バイト (ダッシュ付き) のいずれかです。

構文

newid([*optionflag*])

パラメータ

option flag

- 0 または値の指定なし − 生成される GUID は、人間が判読できる値 (varchar) ですが、ダッシュは含まれません。この引数はデフォルトであり、値を varbinary に変換するのに便利です。
- -1 − 生成される GUID は、人間が判読できる値 (varchar) であり、ダッシュが含まれます。
- -0x0 − GUID を varbinary として返します。
- newid のその他の値はすべて NULL を返します。

例

例 1 32 バイト長の varchar カラムを持つテーブルを作成し、insert 文で引数なしの newid を使います。

```
create table t (UUID varchar(32))
go
insert into t values (newid())
insert into t values (newid())
go
select * from t

UUID
--------------------------------
f81d4fae7dec11d0a76500a0c91e6bf6
7cd5b7769df75cefe040800208254639
```

例 2 ダッシュ付きの GUID を生成します。

```
select newid(1)
-------------------------------------
b59462af-a55b-469d-a79f-1d6c3c1e19e3
```

例 3 ダッシュなしの GUID フォーマットを varbinary(16) カラムに変換するデフォルトを作成します。

```
create table t (UUID_VC varchar(32), UUID varbinary(16))
go
create default default_guid
as
strtobin(newid())
go
sp_bindefault default_guid, "t.UUID"
go
insert t (UUID_VC) values (newid())
go
```

例 4 クエリで返されるローのそれぞれに varbinary 型の新しい GUID を作成して返します。

```
select newid(0x0) from sysobjects
```

例 5 newid に varbinary データ型を使用します。

```
sp_addtype binguid, "varbinary(16)"
create default binguid_dflt
as
newid(0x0)
sp_bindefault "binguid_dflt","binguid"
create table T1 (empname char(60), empid int, emp_guid binguid)
insert T1 (empname, empid) values ("John Doe", 1)
insert T1 (empname, empid( values ("Jane Doe", 2)
```

使用法

- newid は、newid に渡された引数に基づいて、2 とおりの GUID 値を生成します。デフォルトの引数では、ダッシュなしの GUID が生成されます。デフォルトで newid は、フィルタされたローごとに新しい値を返します。
- newid は、他の関数と同様に、デフォルト、ルール、トリガで使用できます。
- ダッシュなしの GUID フォーマットでは varchar カラムの長さが 32 バイト以上あることを、ダッシュ付きの GUID フォーマットでは 36 バイト以上あることを確認してください。カラム長がこれらの最低限必要な長さで宣言されていない場合は、カラム長がトランケートされます。カラム長がトランケートされると、重複値が生じる可能性があります。
- 引数ゼロはデフォルトと同じです。
- strtobin 関数でダッシュなしの GUID フォーマットを使用すると、GUID 値を 16 バイトのバイナリ・データに変換できます。一方、strtobin 関数でダッシュ付きの GUID 形式を使用すると、結果は NULL 値になります。
- GUID はグローバルにユニークなため、ドメイン間でトランスポートしても、値が重複することはありません。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能。

パーミッション

すべてのユーザが newid 関数を実行できます。

next_identity

説明

次の insert に使用可能な次の identity 値を返します。

構文

next_identity(*table_name*)

パラメータ

table_name
使用するテーブルを指定します。

例

テーブル t1 の identity 値は 10 まで利用されています。次に使用可能な値は 11 です。

```
select next_identity ("t1")
t1
--------
11
```

使用法

- next_identity は、このタスクによって挿入される次の値を返します。複数のユーザが同じテーブルに値を挿入しようとしている場合、次の値として報告された値でも、他のユーザがその間に値を挿入してしまうと、実際に挿入される値と異なる場合があります。
- next_identity は、任意の精度の identity カラムをサポートする varchar 文字を返します。テーブルがプロキシ・テーブル、ユーザ・テーブル以外のテーブル、または identity プロパティを持たないテーブルの場合は、NULL が返されます。

パーミッション

テーブル所有者、システム管理者、データベース管理者のみがこのコマンドを発行できます。

nullif

説明

条件付き SQL 式をサポートします。値式が使用可能な任意の場所で使用できます。これは、case 式の代替コマンドです。

構文

nullif(*expression*, *expression*)

パラメータ

nullif

2 つの式の値を比較します。最初の式が 2 番目の式と同じであれば、nullif は NULL を返します。最初の式と 2 番目の式が異なれば、nullif は最初の式を返します。

expression

カラム名、定数、関数、サブクエリ、またはこれらを算術演算子やビット処理演算子でつないだ任意の組み合わせです。式の詳細については、「式」(341 ページ) を参照してください。

例

例 1 titles テーブルから title と type を選択します。本のタイプが UNDECIDED の場合、nullif は NULL 値を返します。

```
select title,
    nullif(type, "UNDECIDED")
  from titles
```

例 2 これは、上記の例 1 の別の記述方法です。

```
select title,
    case
        when type = "UNDECIDED" then NULL
        else type
    end
  from titles
```

使用法

- nullif 式は case 式の代替として使用されます。
- nullif 式は、when...then 構成を使用する代わりに探索条件を単純な比較として表すことによって、標準 SQL 式を簡略化します。
- nullif 式は、SQL 内で式が使用可能な任意の場所で使用できます。
- case 式の 1 つ以上の結果が null 以外の値を返す必要があります。たとえば、次の場合はエラー・メッセージが表示されます。

```
select price, coalesce (NULL, NULL, NULL)
from titles
All result expressions in a CASE expression must not be NULL.
```

- クエリがさまざまなデータ型を作成する場合は、「混合モードの式のデータ型」(6 ページ) で説明されているとおり、データ型の階層によって case 式の結果のデータ型が決定されます。Adaptive Server で暗黙的に変換されないデータ型 (char や int など) を 2 つ指定した場合、クエリは正常に動作しません。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが nullif 関数を実行できます。

参照

コマンド case, coalesce, select, if...else, where 句

object_attr

説明

セッション、テーブル設定、データベースワイド設定に応じて、テーブルの現在のロギング・モードをレポートします。

構文

object_attr(*table_name*, *string*)

パラメータ

table_name

テーブルの名前です。

string

クエリ対象のテーブルのプロパティ名です。サポートされている文字列値は以下のとおりです。

- dml_logging − 要求した有効なオブジェクトの DML ロギング・レベルを、明示的に設定されたテーブルまたはデータベースの DML ロギング・レベルに基づいて返します。
- dml_logging for session − 現在のセッションの DML ロギング・レベルを、object_attr を実行しているユーザ、テーブルのスキーマ、複数文のトランザクションに関するルールなどを考慮して返します。この引数の戻り値は、ユーザによって異なる場合があり、同じユーザでも文やトランザクションごとに異なる可能性があります。
- help − サポートされている文字列引数のリストを出力します。

例

例 1 どのプロパティをクエリできるかを確認するには、次のコマンドを実行します。

```
select object_attr('sysobjects', 'help')
Usage: object_attr('tabname', 'attribute')

List of options in attributes table:
     0 : help
     1 : dml_logging
     2 : dml_logging for session
```

dml_logging は、オブジェクトの統計的に定義された dml_logging レベルをレポートし、dml_logging for session は、データベース固有の設定とセッションの設定に応じてオブジェクトに選択された実行時ロギング・レベルをレポートします。

例 2 持続性を full に設定したテーブルのデフォルトのロギング・モード：

```
select object_attr("pubs2..authors",
         "dml_logging")

Returns: FULL
```

例 3 セッションですべてのテーブルのロギングを無効にしている場合、このユーザが所有するテーブルに対して返されるロギング・モードは minimal です。

```
select object_attr("pubs2..authors",
         "dml_logging")

Returns: FULL

SET DML_LOGGING MINIMAL
go

select object_attr("pubs2..authors",
         "dml_logging for session")

Returns: MINIMAL
```

例 4 テーブルが最低限のロギングを明示的に選択するように変更されている場合は、セッションおよびデータベース全体のロギングが FULL でも、は object_attr minimal の値を返します。

```
create database testdb WITH DML_LOGGING = FULL
go

create table non_logged_table (...)
WITH DML_LOGGING=MINIMAL
go

select object_attr("non_logged_table",
         "dml_logging")
Returns: MINIMAL
```

例 5 テーブルのロギングを full から minimal に変更します。full ロギングを使用して明示的にテーブルを作成した場合は、テーブルの所有者または sa_role のユーザであれば、セッション中にロギングを minimal に再設定できます。

1 最低限のロギングを使用して testdb データベースを作成します。

```
create database testdb
with dml_logging = minimal
```

2 dml_logging を full に設定してテーブルを作成します。

```
create table logged_table(...)
with dml_logging = full
```

3 セッションのロギングを minimal に再設定します。

```
set dml_logging minimal
```

4 テーブルのロギングは最小限です。

```
select object_attr("logged_table",
          "dml_logging for session")
--------------------
minimal
```

例 6 ロギング・モードを指定せずにテーブルを作成した場合に、セッションのロギング・モードを変更すると、テーブルのロギング・モードも変更されます。

- テーブル normal_table を作成します。

```
create table normal_table
```

- セッションのロギングをチェックします。

```
select object_attr("normal_table", "dml_logging")
-------------------------
FULL
```

- セッションのロギングを minimal に設定します。

```
set dml_logging minimal
```

- テーブルのロギングを minimal に設定します。

```
select object_attr("normal_table",
          "dml_logging for session")
-----------------------------
minmimal
```

例 7 object_attr から返されるロギング・モードは、それを実行するテーブルによって異なります。この例では、ユーザ joe がスクリプトを実行しますが、Adaptive Server が返すロギング・モードが変わります。テーブル joe.own_table と mary.other_table は、full ロギング・モードを使用します。

```
select object_attr("own_table","dml_logging")
--------------
          FULL
```

joe が mary.other_table に object_attr を実行すると、このテーブルも full に設定されます。

```
select object_attr("mary.other_table", "dml_logging")
--------------
          FULL
```

joe が dml_logging を minimal に変更した場合は、joe が所有しているテーブルのロギング・モードのみが影響を受けます。

```
set dml_logging minimal
select object_attr("own_table", "dml_logging for session")
--------------
       MINIMAL
```

他のユーザが所有しているテーブルは、それぞれのデフォルトのロギング・モードで処理を続行します。

```
Select object_attr("mary.other_table", "dml_logging for
session")
--------------
          FULL
```

例 8 新しい show_exec_info をロギングする実行時の選択を識別し、それを SQL バッチで使用します。

1 set showplan を有効にします。

```
set showplan on
```

2 set コマンドを有効にします。

```
set show_exec_info on
```

3 dml_logging を minimal に設定し、object_attr でロギングをチェックします。

```
set dml_logging minimal
select object_attr("logged_table", "dml_logging for session")
```

4 テーブルからローを削除します。

```
delete logged_table
```

Adaptive Server は実行時にテーブルのロギング・モードを show_exec_info パラメータでレポートします。

使用法

- 戻り型は varchar で、クエリ対象のプロパティに応じてプロパティの値 (たとえば、on または off) を適切に返します。
- 拡張から showplan 出力にレポートされるロギング・モードは、DML の実行前、同じバッチにテーブルのロギング・モードを変更する set 文があると、実行時に影響を受ける可能性があります。
- 不明のプロパティの戻り値は値 NULL です (文字列 “NULL” ではありません)。
- 特殊な文字列パラメータ、help はセッションの出力に、現在サポートされている object_attr のプロパティすべてを出力します。そのため、object_attr がどのプロパティをサポートしているかを即座に確認できます。

パーミッション

参照

object_id

説明　指定したオブジェクトのオブジェクト ID を返します。

構文　object_id(*object_name*)

パラメータ

object_name

データベース・オブジェクト (テーブル、ビュー、プロシージャ、トリガ、デフォルト、またはルールなど) の名前です。完全修飾名を指定できます (つまり、データベースと所有者名が含まれている名前を指定できます)。*object_name* は引用符で囲んでください。

例

例 1

```
select object_id("titles")

-----------
  208003772
```

例 2

```
select object_id("master..sysobjects")

-----------
          1
```

使用法　システム関数 object_id はオブジェクトの ID を返します。オブジェクト ID は sysobjects の id カラムに格納されています。

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能。

パーミッション　すべてのユーザが object_id 関数を実行できます。

参照　システム関数の概要については、「システム関数」(69 ページ) を参照してください。

関数　col_name, db_id, object_name

システム・プロシージャ　sp_help

object_name

説明

指定されたオブジェクト ID を持つオブジェクトの名前を返します。名前は、最長で 255 バイトです。

構文

object_name(*object_id*[, *database_id*])

パラメータ

object_id

(テーブル、ビュー、プロシージャ、トリガ、デフォルト、ルールなどの) データベース・オブジェクトのオブジェクト ID です。オブジェクト ID は sysobjects の id カラムに格納されています。

database_id

オブジェクトが現在のデータベースにない場合に使用するデータベースの ID です。データベース ID は sysdatabases の db_id カラムに格納されています。

例

例 1

```
select object_name(208003772)

------------------------------
titles
```

例 2

```
select object_name(1, 1)

------------------------------
sysobjects
```

使用法

システム関数 object_name は、オブジェクトの名前を返します。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが object_name 関数を実行できます。

参照

システム関数の概要については、「システム関数」(69 ページ) を参照してください。

関数 col_name, db_id, object_id

システム・プロシージャ sp_help

object_owner_id

説明　オブジェクトの所有者 ID を返します。

構文　object_owner_id(*object_id*[, *database_id*])

パラメータ

object_id
　調べるオブジェクトの ID です。

database_id
　オブジェクトが存在するデータベースの ID です。

例　ID が 1 のデータベース (マスタ・データベース) で ID が 1 のオブジェクトの所有者の ID を返します。

```
select object_owner_id(1,1)
```

パーミッション　すべてのユーザが object_owner_id を実行できます。

pagesize

説明

指定したオブジェクトのページ・サイズをバイト数で返します。

構文

pagesize(*object_name*[,])
pagesize(*object_id*[,*db_id*[, *index_id*]])

パラメータ

object_name
この関数によってページ・サイズが返されるオブジェクトの名前です。

index_name
返されるページ・サイズのインデックスの名前です。

object_id
この関数によってページ・サイズが返されるオブジェクトの ID です。

db_id
オブジェクトのデータベース ID です。

index_id
返されるオブジェクトのインデックス ID です。

例

例 1 現在のデータベースの title_id インデックスのページ・サイズを返します。

```
select pagesize("title", "title_id")
```

例 2 *db_id* が 2 のデータベースで、*object_id* が 1234 のオブジェクトのデータ・レイヤのページ・サイズを返します (前の例は、デフォルトとして現在のデータベースになります)。

```
select pagesize(1234,2, null)
select pagesize(1234.2)
select pagesize(1234)
```

例 3 次の例は、すべて現在のデータベースのデフォルトになります。

```
select pagesize(1234, null, 2)
select pagesize(1234)
```

例 4 pubs2 データベース (db_id 4) の titles テーブル (object_id 224000798) のページ・サイズを返します。

```
select pagesize(224000798, 4)
```

例 5 現在のデータベースにある、ノンクラスタード・インデックスのページ・テーブル mytable のページ・サイズを返します。

```
pagesize(object_id('mytable'), NULL, 2)
```

例 6 現在のデータベースから、オブジェクト titles_clustindex のページ・サイズを返します。

```
select pagesize("titles", "titles_clustindex")
```

使用法

- インデックス名または *index_id* を指定しない場合 (たとえば、`select pagesize("t1")`)、またはパラメータに“null”を使用した場合 (たとえば、`select pagesize("t1", null)`) は、pagesize の対象はデフォルトでデータ・レイヤになります。
- 指定されたオブジェクトがページ用の物理データ記憶領域を必要としないオブジェクトの場合は (たとえば、ビューの名前を指定した場合)、pagesize は 0 を返します。
- 指定したオブジェクトが存在していない場合、pagesize は NULL を返します。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが pagesize 関数を実行できます。

partition_id

説明

指定したデータまたはインデックス・パーティション名のパーティション ID を返します。

構文

partition_id(*table_name*, *partition_name*[,*index_name*])

パラメータ

table_name
テーブルの名前です。

partition_name
テーブル・パーティションまたはインデックス・パーティションのパーティション名です。

index_name
対象となるインデックスの名前です。

例

例 1 パーティション名 testtable_ptn1 とインデックス ID 0 (ベース・テーブル) に対応するパーティション ID を返します。testtable は、現在のデータベースに存在している必要があります。

```
select partition_id("testtable", "testtable_ptn1")
```

例 2 インデックス名 clust_index1 のパーティション名 testtable_clust_ptn1 に対応するパーティション ID を返します。testtable は、現在のデータベースに存在している必要があります。

```
select partition_id("testtable", "testtable_clust_ptn1", "clust_index1")
```

例 3 これは前の例と同じですが、ターゲット・テーブルが現在のデータベースにある必要はありません。

```
select partition_id("mydb.dbo.testtable", "testtable_clust_ptn1",
    "clust_index1")
```

使用法

table_name、*partition_name*、*index_name* は引用符で囲む必要があります。

参照

関数 data_pages, object_id, partition_name, reserved_pages, row_count, used_pages

partition_name

説明

新しいパーティションの明示的な名前 partition_name は、指定されたデータのパーティション名またはインデックス・パーティション ID を返します。

構文

partition_name(*indid*, *ptnid* [, *dbid*])

パラメータ

indid
ターゲット・パーティションのインデックス ID です。

ptnid
ターゲット・パーティションの ID です。

dbid
ターゲット・パーティションのデータベース ID です。このパラメータを指定しない場合、ターゲット・パーティションは現在のデータベースにあると解釈されます。

例

例 1 ベース・テーブル (インデックス ID 0) に属する特定のパーティション ID を持つパーティション名を返します。ここではデータベース ID を指定しないため、検索は現在のデータベースを対象として行われます。

```
select partition_name(0, 1111111111)
```

例 2 testdb データベース内で、クラスタード・インデックス (インデックス ID に 1 を指定) に属する特定のパーティション ID を持つパーティション名を返します。

```
select partition_name(1, 1212121212, db_id("testdb")
```

使用法

- ターゲット・パーティションが見つからない場合、NULL が返されます。

参照

関数　data_pages, object_id, partition_id, reserved_pages, row_count

partition_object_id

説明

指定されたパーティション ID とデータベース ID のオブジェクト ID を返します。

構文

partition_object_id(*partition_id* [, *database_id*])

パラメータ

partition_id

取得するオブジェクト ID が設定されているパーティションの ID です。

database_id

パーティションのデータベース ID です。

例

例 1 パーティション ID が 2 であるパーティションのオブジェクト ID を表示するには、次のように入力します。

```
select partition_object_id(2)
```

例 2 パーティション ID が 14 でデータベース ID が 7 であるパーティションのオブジェクト ID を表示するには、次のように入力します。

```
select partition_object_id(14,7)
```

例 3 次の場合は、関数に NULL 値が渡されているため、このデータベース ID に対して NULL 値を返します。

```
select partition_object_id( 1424005073, NULL)

 -----------
NULL
(1 row affected)
```

使用法

- データベース ID を含まない場合、partition_object_id は現在のデータベース ID を使用します。
- *partition_id* に NULL 値を使用すると、partition_object_id は NULL を返します。
- データベース ID に NULL 値を含めると、partition_object_id は NULL 値を返します。
- 無効または存在しない *partition_id* または *database_id* を指定すると、partition_object_id は NULL を返します。

patindex

説明

指定したパターンが最初に検出された先頭位置を返します。

構文

patindex("%*pattern*%", *char_expr*|*uchar_expr*[, using
{bytes | characters | chars}])

パラメータ

pattern

データ型が char または varchar の文字式です。この式には、Adaptive Server で使用可能なパターン一致ワイルドカード文字を含めることができます。*pattern* の前後には、% ワイルドカード文字を挿入してください (最初の文字または最後の文字を検索する場合は除く)。ワイルドカード文字については、「ワイルドカード文字によるパターン一致」(358 ページ) を参照してください。

char_expr

文字型のカラム名、変数、または (char、varchar、nchar、または nvarchar 型の) 定数式です。

uchar_expr

文字型のカラム名、変数、または (unichar または univarchar 型の) 定数式です。

using

先頭位置のフォーマットを指定します。

bytes

オフセットをバイト数で返します。

chars または characters

オフセットを文字数で返します (デフォルト)。

例

例 1 作者 ID と copy カラムの “circus” という語の先頭文字の位置を返します。

```
select au_id, patindex("%circus%", copy)
from blurbs

au_id
 ----------- -----------
  486-29-1786           0
  648-92-1872           0
  998-72-3567          38
  899-46-2035          31
  672-71-3249           0
  409-56-7008           0
```

例 2

```
select au_id, patindex("%circus%", copy,
     using chars)
from blurbs
```

例 3 sysobjects 内で、“sys” で始まり、4 番目の文字が “a”、“b”、“c”、または “d” であるすべてのローを返します。

```
select name
from sysobjects
where patindex("sys[a-d]%", name) > 0

name
------------------------------
sysalternates
sysattributes
syscharsets
syscolumns
syscomments
sysconfigures
sysconstraints
syscurconfigs
sysdatabases
sysdepends
sysdevices
```

使用法

- 文字列関数 patindex は、指定した文字式内で *pattern* が最初に検出された先頭位置を表す整数を返します。*pattern* がないときは 0 を返します。
- patindex 関数では、text データや image データを含むすべての文字データを使用できます。
- unichar、univarchar、unitext の場合、patindex は、Unicode 文字内のオフセットを返します。パターン文字列は、比較の前に UTF-16 に暗黙的に変換されます。この変換は default unicode sort order の設定に基づいて実行されます。たとえば、unitext カラムに U+0041U+0042U+d800U+dc00U+0043 というロー値が保管されている場合、次の値が返されます。

  ```
  select patindex("%C%", ut) from unitable
  -----------
  4
  ```

- デフォルトでは、patindex 関数はオフセットを文字数で返します。オフセットをバイト数 (マルチバイト文字列) で返すには、using bytes 句を指定します。
- *pattern* 引数の前後にはパーセント記号を指定してください。カラム内で、*pattern* に指定した値で始まる文字列を検索するには、この引数の前に % を指定しないでください。カラム内で、*pattern* に指定した値で終わる文字列を検索するには、この引数の後に % を指定しないでください。

- *char_expr* または *uchar_expr* が NULL の場合は、patindex は 0 を返します。
- varchar 式と unichar 式をそれぞれパラメータとして指定した場合、varchar 式は暗黙的に unichar へ変換されます (変換時にトランケーションが発生することがあります)。
- 文字列関数の一般的な説明については、「文字列関数」(68 ページ) を参照してください。

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　すべてのユーザが patindex 関数を実行できます。

参照　**関数**　charindex, substring

pi

説明 定数値 (円周率) 3.1415926535897936 を返します。

構文 pi()

パラメータ なし。

例

```
select pi()

-------------------
          3.141593
```

使用法 算術関数 pi は、定数値 3.1415926535897931 を返します。

規格 ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション すべてのユーザが pi 関数を実行できます。

参照 算術関数の概要については、「算術関数」(67 ページ) を参照してください。

関数 degrees, radians

power

説明　指定した数字を指定の累乗で計算した結果の値を返します。

構文　power(*value*, *power*)

パラメータ

value
数値です。

power
真数値、概数値、または通貨値です。

例

```
select power(2, 3)

-----------
          8
```

使用法

- 算術関数 power は、*value* の値を *power* 乗した結果の値を返します。結果は *value* と同じ型になります。

  numeric 型または decimal 型の式の場合、この関数は精度 38 と位取り 18 を返します。

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　すべてのユーザが power 関数を実行できます。

参照　算術関数の概要については、「算術関数」(67 ページ) を参照してください。

関数　exp, log, log10

proc_role

説明

指定した役割がユーザに付与されているかどうかを示す情報を返します。

注意 Sybase は、has_role に代えて proc_role をサポートしています。また、この関数の使用を推奨しています。ただし、既存の proc_role をすべて has_role に変換する必要はありません。

構文

proc_role("*role_name*")

パラメータ

role_name

システム定義またはユーザ定義の役割の名前です。

例

例 1 ユーザがシステム管理者かどうかをチェックするプロシージャを作成します。

```
create procedure sa_check as
if (proc_role("sa_role") > 0)
begin
    print "You are a System Administrator."
    return(1)
end
```

例 2 ユーザにシステム・セキュリティ担当者の役割が付与されているかどうかをチェックします。

```
select proc_role("sso_role")
```

例 3 ユーザにオペレータの役割が付与されているかどうかをチェックします。

```
select proc_role("oper_role")
```

使用法

- 名前が“sp_”で始まるプロシージャを指定して proc_role を使用すると、エラーが返されます。
- システム関数 proc_role は、指定した役割が呼び出し元ユーザに対して付与されているかどうかをチェックします。
- ユーザの役割が次の場合に proc_role は 0 を返します。
  - 指定した役割がユーザに付与されていない。
  - 指定した役割を含む役割がユーザに付与されていない。
  - 指定した役割がユーザに付与されているが、アクティブ化されていない場合。
- 指定した役割が呼び出し元ユーザに対して付与されており、アクティブ化されている場合、proc_role は 1 を返します。
- 呼び出し元ユーザのアクティブな役割に、指定の役割が含まれている場合、proc_role は 2 を返します。

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　すべてのユーザが proc_role 関数を実行できます。

参照　システム関数の概要については、「システム関数」(69 ページ) を参照してください。

コマンド　alter role, create role, drop role, grant, set, revoke

関数　mut_excl_roles, role_contain, role_id, role_name, show_role

pssinfo

説明

Adaptive Server プロセス・ステータス構造 (pss) から情報を返します。

構文

pssinfo(*spid* | 0, '*pss_field*')

パラメータ

SPID

0 を指定すると現在のプロセスが使用されます。

pss_field

プロセス・ステータス構造体フィールドです。有効な値は以下のとおりです。

- dn − LDAP 認証を使用する場合の識別名。
- extusername − PAM や LDAP のような外部認証を使用する場合、extusername は、使用される外部 PAM ユーザ名または LDAP ユーザ名を返します。
- ipaddr − クライアントの IP アドレス。
- ipport − クエリ対象のユーザ・タスクに関連するクライアント接続に使用されるクライアント IP ポート番号。
- isolation_level − 現在のセッションの分離レベル。
- tempdb_pages − 使用される tempdb ページ数。

例

spid 14 番のポート番号を返します。

```
select pssinfo(14,'ipport')
----------------
           52039
```

使用法

- pssinfo 関数には、外部ユーザ名と識別名を表示するオプションもあります。
- ipport 出力と ipaddr 出力を組み合わせると、Adaptive Server とクライアント間のネットワーク・トラフィックを一意に識別できます。

radians

説明　指定した度数の角度の大きさをラジアンで返します。

構文　radians(*numeric*)

パラメータ　*numeric*

真数値 (numeric、dec、decimal、tinyint、smallint、または int 型)、概数値 (float、real、または double precision 型)、money 型のカラム、変数、定数式、またはこれらの組み合わせのいずれかです。

例

```
select radians(2578)

-----------
         44
```

使用法

- 算術関数 radians は、度数をラジアン数に変換します。結果は *numeric* と同じ型になります。

  numeric データ型または numeric データ型を表すため、この関数は精度 38、位取り 18 を返します。

  通貨データ型を使用すると、float 型への内部変換で精度ロスが発生することがあります。

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　すべてのユーザが radians 関数を実行できます。

参照　算術関数の概要については、「算術関数」(67 ページ) を参照してください。

関数　degrees

rand

説明

指定した基数値を使って生成される 0 と 1 の間の乱数値を返します。

構文

rand([*integer*])

パラメータ

integer

整数値型 (tinyint、smallint、または int 型) のカラム名、変数、定数式、またはこれらの組み合わせのいずれかです。

例

例 1

```
select rand()

--------------------
           0.395740
```

例 2

```
declare @seed int
select @seed=100
select rand(@seed)

--------------------
             0.000783
```

使用法

- 算術関数 rand は、オプションの整数値を基数値として使用し、0 と 1 の間の浮動小数の乱数値を返します。
- rand 関数は、32 ビット整数の疑似乱数値ジェネレータの出力を使用します。整数値を 32 ビットの最大整数値で除算し、0.0 ～ 1.0 の間の 2 倍の値を求めます。rand 関数の基数値はサーバの起動時に無作為に決まるので、ユーザが最初に定数の基数値でこの関数を初期化しないかぎり、同じ乱数順になることはほとんどありません。rand 関数はグローバル・リソースです。rand 関数を呼び出している複数のユーザは、一連の疑似乱数値にそって処理されます。一連の乱数を繰り返したいときは、ユーザはその関数の基数値を最初に同じ値にし、繰り返したい間は他のユーザが rand 関数を呼び出さないことを確かめてください。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが rand 関数を実行できます。

参照

算術関数の概要については、「算術関数」(67 ページ) を参照してください。

データ型 概数値データ型

関数 rand2

rand2

説明

指定したシード値を使用して生成され、select リストで使用されたときに返された各ローに対して計算された 0 と 1 の間のランダム値を返します。

構文

rand2([*integer*])

パラメータ

integer

整数値型 (tinyint、smallint、または int 型) のカラム名、変数、定数式、またはこれらの組み合わせのいずれかです。

例

t に n 個のローある場合、次の select 文によって 1 つだけではなく n 個の異なる乱数値が返されます。

```
select rand2() from t
------------
```

使用法

- 算術関数 rand2 は、オプションの整数値を基数値として使用し、0 と 1 の間の浮動小数の乱数値を返します。rand とは異なり、select リストで使用される場合は、返されたローごとに計算されます。
- rand2 の動作は、現在 select リストを除いて未定義です。
- 32 ビット整数の擬似乱数ジェネレータの詳細については、rand の「使用法」を参照してください。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが rand 関数を実行できます。

参照

算術関数の概要については、「算術関数」(67 ページ) を参照してください。

データ型　概数値データ型

関数　rand

replicate

説明

指定した式を指定した回数だけ繰り返した結果の文字列を返します。

構文

replicate(*char_expr* | *uchar_expr*, *integer_expr*)

パラメータ

char_expr
文字型のカラム名、変数、または (char、varchar、nchar、または nvarchar 型の) 定数式です。

uchar_expr
文字型のカラム名、変数、または (unichar または univarchar 型の) 定数式です。

integer_expr
整数値型 (tinyint、smallint、または int) のカラム名、変数、定数式のいずれかです。

例

```
select replicate("abcd", 3)

------------
abcdabcdabcd
```

使用法

- 文字列関数 replicate は、指定した回数か、または 16 KB の記憶領域に保存可能な最大回数のうち、少ない回数で繰り返された式を含む文字列を返します。返される文字列のデータ型は *char_expr* または *uchar_expr* と同じデータ型です。
- *char_expr* または *uchar_expr* が NULL の場合は、NULL を返します。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが replicate 関数を実行できます。

参照

文字列関数の概要については、「文字列関数」(68 ページ) を参照してください。

関数 stuff

reserve_identity

説明

reserve_identity では、あるプロセスが identity 値のブロックを予約し、そのプロセスがこの値を利用できるようにします。

プロセスが reserve_identity を呼び出して値のブロックを予約した後、このプロセスが必要とする後続の identity 値は、この予約済みプールから導出されます。これらの予約済み数字が使い終わるか、別のテーブルにデータを挿入すると、既存の identity オプションが適用されます。reserve_identity には identity 値の複数のブロックを保持できます。そのため、別のテーブルに挿入されたデータが 1 つのプロセスによってインタリーブされると、テーブルの予約済みブロック内の次の値が使用されます。

具体的なテーブルの identity 値の特定のサイズ・ブロックを予約します。これらの identity 値は、呼び出し元プロセスによって排他的に使用されます。予約済みの開始番号が返されます。このプロセスによって、指定されたテーブルに対してその後 insert が行われると、これらの値が使用されます。プロセスが終了すると、未使用の値は排除されます。

構文

reserve_identity (*table_name*, *number_of_values*)

パラメータ

table_name

予約するテーブルの名前です。完全修飾名を指定できます。つまり、*database_name*、*owner_name*、*object_name* を含めた名前を引用符で囲んで指定できます。

number_of_values

このプロセスに予約されている identity 連続値の数です。この数値には、いずれの予約値も identity カラムのデータ型の最大値を超えないような正の数値を指定してください。

例

reserve_identity の代表的な使用シナリオについて説明します。ここでは、table1 に col1 (データ型は int) と col2 (データ型 int の identity カラム) が含まれていると想定します。これは spid 3 のプロセスです。

```
select reserve_identity( table1, 5 )
---------------
10
```

spid 3 および 4 の値を挿入します。

```
Insert table1 values(56) -> spid 3
Insert table1 values(48) -> spid 3
Insert table1 values(96) -> spid 3
Insert table1 values(02) -> spid 4
Insert table1 values(84) -> spid 3
```

テーブル table1 から選択します。

```
select * from table1

Col1          col2
--------       -----
```

```
3               1-> spid 3 reserved 1-5
3               2-> spid 3
3               3-> spid 3
4               6<= spid 4 gets next unreserved value
3               4<= spid 3 continues with reservation
```

spid 3 が identity 値 1 ～ 5 を予約したこと、また spid 4 が次の未予約の値を受け取っていること、さらに spid 3 が次の identity 値を予約していることが結果セットに表示されます。

使用法

- 戻り値 *start_value* は、予約済み identity 値のブロックの開始値です。呼び出し元プロセスは、この値を使用して指定されたテーブルに次の挿入を行います。
- reserve_identity を使用すると、次の操作ができます。
  - insert 文を発行せずに identity 値を予約する。
  - insert 文を発行する前に予約された値を把握する。
  - 必要に応じて、identity 値のさまざまなサイズ・ブロックを「取得」する。
  - 必要な値だけを予約して「余分なギャップ」を制御する (つまり、事前設定されたサーバ取得サイズによってギャップが制限されることがありません)。
- 値は、insert 構文を変更しないで自動的に使用されます。
- 次の場合は、NULL 値が返されます。
  - 負の値またはゼロがブロック・サイズとして指定された。
  - テーブルが存在しない。
  - テーブルに identity カラムが含まれていない。
- このプロセスがこれらの identity 値をすでに予約しているテーブルに対して reserve_identity を発行すると、この関数は正常に実行され、最新の値グループが使用されます。
- reserve_identity を使用してプロキシ・テーブル上で identity 値を予約することはできません。ローカル・サーバが reserve_identity を呼び出すリモート・プロシージャを呼び出すと、このローカル・サーバは、リモート・テーブルについて reserve_identity 関数を使用できます。これらの reserved 値はリモート・サーバに保存されますが、その後にリモート・テーブルにデータを挿入するときに、ローカル・サーバのセッションでこの reserved 値が使用されます。
- identity_gap が予約済みブロック・サイズより小さい場合、値の指定ブロック・サイズ (identity_gap サイズではない) を予約することで予約を正常に実行できます。これらの値がプロセスによって使用されない場合、identity_gap 設定に関係なく、指定されたブロック・サイズを上限としたギャップが生成される可能性があります。

パーミッション

identity 値を予約するには、insert パーミッションが必要です。

reserved_pages

説明

データベース、オブジェクト、またはインデックス用に予約されているページ数をレポートします。レポートには、内部構造に使用するページが含まれます。

この関数は、バージョン 15.0 より前の Adaptive Server で使用された古い reserved_pgs 関数を置き換えるものです。

構文

reserved_pages(*dbid*, *object_id*[, *indid*[, *ptnid*]])

パラメータ

dbid
ターゲット・オブジェクトが存在するデータベースのデータベース ID です。

object_id
テーブルのオブジェクト ID です。

indid
ターゲット・インデックスのインデックス ID です。

ptnid
ターゲット・パーティションのパーティション ID です。

例

例 1 指定されたデータベースにある、オブジェクト ID が 31000114 のオブジェクトに予約されているページ数を返します (インデックスも含む)。

```
select reserved_pages(5, 31000114)
```

例 2 クラスタード・インデックスがあるかどうかに関係なく、データ・レイヤに含まれるオブジェクトで予約されているページ数を返します。

```
select reserved_pages(5, 31000114, 0)
```

例 3 クラスタード・インデックスに対応するインデックス・レイヤのオブジェクトで予約されているページ数を返します。データ・レイヤで使用されるページは含まれません。

```
select reserved_pages(5, 31000114, 1)
```

例 4 指定されたパーティション (この例では 2323242432) のデータ・レイヤにあるオブジェクトで予約されているページ数を返します。

```
select reserved_pages(5, 31000114, 0, 2323242432)
```

例 5 以下の 3 とおりの方法のいずれかを使用して、reserved_pages でデータベースのスペースを計算します。

- case 式を使用して、調べるインデックスに適した値を選択し、このデータベースの sysindexes のログ以外のインデックスを選択します。このクエリでは、以下のとおりになります。
  - データに “index 0” の値があり、文 `when sysindexes.indid = 0` または `sysindexes.indid = 1` を含めるときに使用できます。
  - 1 より大きい indid 値はインデックスです。このクエリはデータ領域をインデックス・カウントに合計しないため、indid が 0 のページ・カウントは含んでいません。

- 各オブジェクトには 0 か 1 のインデックス・エントリがあり、両方が含まれていることはありません。
- このクエリはテーブル 1 つにつきインデックス 0 を 1 度だけカウントします。

```
select
'data rsvd' = sum( case
          when indid > 1 then 0
          else reserved_pages(db_id(), id, 0)
          end ),
'index rsvd' = sum( case
          when indid = 0 then 0
          else reserved_pages(db_id(), id, indid)
          end )
from sysindexes
where id != 8
data rsvd   index rsvd
----------   -----------
       812           1044
```

- sysindexes を複数回クエリして、すべてのクエリが完了した跡で結果を表示します。

```
declare @data int,
@dbsize int,
@dataused int,
@indices int,
@indused int
select @data = sum( reserved_pages(db_id(), id, 0) ),
     @dataused = sum( used_pages(db_id(), id, 0) )
from sysindexes
where id != 8
and indid <= 1
select @indices = sum( reserved_pages(db_id(), id, indid) ),
     @indused = sum( used_pages(db_id(), id, indid) )
from sysindexes
where id != 8 and indid > 0
select @dbsize   as 'db size',
@data as 'data rsvd'
db size    data rsvd
----------- -----------
       NULL         820
```

- データ領域情報には sysobjects、インデックス情報には sysindexes をクエリします。sysobjects からテーブルのオブジェクト [S]ystem または [U]ser を選択します。

```
declare   @data int,
          @dbsize int,
          @dataused int,
          @indices int,
          @indused int
```

```
select @data = sum( reserved_pages(db_id(), id, 0) ),
@dataused = sum( used_pages(db_id(), id, 0) )
from sysobjects
where id != 8
and type in ('S', 'U')
select @indices = sum( reserved_pages(db_id(), id, indid) ),
       @indused = sum( used_pages(db_id(), id, indid) )
from sysindexes
where id != 8
and indid > 0
select   @dbsize as 'db size',
         @data as 'data rsvd',
         @dataused as 'data used',
         @indices as 'index rsvd',
         @indused as 'index used'
db size    data rsvd    data used    index rsvd    index used
---------  -----------  -----------  -----------   ----------
     NULL          812          499          1044          381
```

使用法

- クラスタード・インデックスが、すべてのページがロックされたテーブルにある場合、インデックス ID 0 を渡すと予約済みデータ・ページがレポートされ、インデックス ID 1 を渡すと予約済みインデックス・ページがレポートされます。すべてのエラーで常に値 0 が返されます。
- reserved_pages は指定したアイテムをカウントします。有効なデータベース、オブジェクト、インデックス (データはすべてのテーブルで「インデックス 0」) を指定すると、このデータベース、オブジェクト、またはインデックス用に予約された領域が返されます。ただし、データベース、オブジェクト、またはインデックスを複数回カウントすることもできます。複数のインデックスがあるテーブルのすべてのインデックスについてデータ領域をカウントさせる場合は、各インデックスにつきデータ領域を 1 回カウントさせます。これらの結果を合計すると、オブジェクトの合計データページ数ではなく、インデックス数に合計データ領域を掛けた数が得られます。
- Adaptive Server バージョン 15.0 では、reserved_pages 関数が reserved_pgs 関数に取って代わります。reserved_pages と reserved_pgs には違いがあります。
  - Adaptive Server バージョン 12.5 以前では、データとインデックスの OAM ページは sysindexes に保存されました。Adaptive Server バージョン 15.0 以降では、この情報は sysparitions にパーティションごとに保存されます。この情報の保存方法の違いにより、reserved_pages と reserved_pgs は別のパラメータを必要とし、結果セットも異なります。

- reserved_pgs にはページ ID が必要でした。一致する sysindexes ローがない値を指定する場合、入力する ID は 0 でした (たとえば、ノンクラスタード・インデックス・ローのデータ OAM ページ)。0 は有効な OAM ページではないため、0 のページ ID を入力した場合、reserved_pgs は 0 を返しました。入力値が無効なため、reserved_pgs は何もカウントできませんでした。

  一方、reserved_pages にはインデックス ID が必要で、0 は有効なインデックス ID です (たとえば、すべてのテーブルでデータが “index 0”。reserved_pages は、indid 0 または 1 を除いてインデックス・ローのデータ領域を再カウントする必要がないことをコンテキストから判別できないため、インデックス ID として 0 を渡すたびにデータ領域を数えます。reserved_pages はこのデータ領域を各ローにつき 1 度カウントするため、真の値の何倍もの合計を出します。

  これらの違いを以下に説明します。

  - OAM ページ入力のページ ID の値として 0 を指定した場合、reserved_pgs は合計に影響せず、0 の値を返すだけです。
  - インデックス ID を 0 の値にして reserved_pages を指定した場合は、データ領域をカウントします。データをカウントしたい場合にのみ reserved_pages を発行してください。そうしないと、合計に影響します。

規格 ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション すべてのユーザが reserved_pgs 関数を実行できます。

参照

コマンド update statistics

関数 data_pages, reserved_pages, row_count, used_pages

reverse

説明　指定した文字列の文字順を逆にして返します。

構文　reverse(*expression* | *uchar_expr*)

パラメータ

expression
文字型またはバイナリ型のカラム名、変数、または (char、varchar、nchar、nvarchar、binary、または varbinary 型の) 定数式です。

uchar_expr
文字型のカラム名、変数、または (unichar または univarchar 型の) 定数式です。

例

例 1

```
select reverse("abcd")

----
dcba
```

例 2

```
select reverse(0x12345000)

---------
0x00503412
```

使用法

- 文字列関数 reverse は、*expression* の文字順を逆にした文字列を返します。
- *expression* が NULL の場合、reverse は NULL を返します。
- サロゲート・ペアは分割不可能なデータとして処理されるため、文字順が逆にされることはありません。

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　すべてのユーザが reverse 関数を実行できます。

参照　文字列関数の概要については、「文字列関数」(68 ページ) を参照してください。

関数　lower, upper

right

説明

式の右端から、指定した文字数の部分を返します。

構文

right(*expression*, *integer_expr*)

パラメータ

expression

文字型またはバイナリ型のカラム名、変数、または (char、varchar、nchar、unichar、nvarchar、univarchar、binary、または varbinary 型の) 定数式です。

integer_expr

整数値型 (tinyint、smallint、または int) のカラム名、変数、定数式のいずれかです。

例

例 1

```
select right("abcde", 3)

---
 cde
```

例 2

```
select right("abcde", 2)

--
 de
```

例 3

```
select right("abcde", 6)

-----
 abcde
```

例 4

```
select right(0x12345000, 3)

-------
 0x345000
```

例 5

```
select right(0x12345000, 2)

------
 0x5000
```

例 6

```
select right(0x12345000, 6)

--------
 0x12345000
```

使用法

- 文字列関数 right は、文字式やバイナリ式の右端から、指定した文字数の部分を返します。
- 指定した右端部分が、サロゲート・ペアの 2 番目のサロゲート (下位サロゲート) の場合、次の完全文字から始まる文字列値が返されます。したがって、指定の文字数より 1 文字少ない文字数の文字列が返されます。
- 戻り値のデータ型は、文字式やバイナリ式と同じデータ型になります。
- *expression* が NULL のとき、right は NULL を返します。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが right 関数を実行できます。

参照

文字列関数の概要については、「文字列関数」(68 ページ) を参照してください。

関数　rtrim, substring

rm_appcontext

説明

特定のアプリケーション・コンテキストまたはすべてのアプリケーション・コンテキストを削除します。rm_appcontext は Application Context Facility (ACF) が提供する関数です。

構文

rm_appcontext(“*context_name*”, “*attribute_name*”

パラメータ

context_name

アプリケーション・コンテキスト名を指定するローです。データ型 char(30) として格納されます。

attribute_name

アプリケーション・コンテキスト属性名を指定するローです。データ型 char(30) として格納されます。

例

例 1 一部またはすべての属性を指定して、アプリケーション・コンテキストを削除します。

```
select rm_appcontext("CONTEXT1", "*")
----------
0

select rm_appcontext("*", "*")
---------
0

select rm_appcontext("NON_EXISTING_CTX","ATTR")
---------
-1
```

例 2 適切なパーミッションを持たないユーザがアプリケーション・コンテキストを削除しようとしたときの結果を示します。

```
select rm_appcontext("CONTEXT1","ATTR2")
----------
-1
```

使用法

- この関数は、成功すると常に 0 を返します。
- この関数では、すべての引数が必須です。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

パーミッションは、ACF に格納されているユーザ・プロファイルとアプリケーション・プロファイルに依存します。

参照

ACF の詳細については、『システム管理ガイド』の「第 17 章 ユーザ・パーミッションの管理」の「ロー・レベル・アクセス制御の使用」を参照してください。

関数 get_appcontext, list_appcontext, set_appcontext

role_contain

説明

role2 に *role1* が含まれている場合は、1 を返します。

構文

role_contain("*role1*", "*role2*")

パラメータ

role1

システム定義またはユーザ定義の役割の名前です。

role2

もう 1 つのシステム定義またはユーザ定義の役割の名前です。

例

例 1

```
select role_contain("intern_role", "doctor_role")

-----------
1
```

例 2

```
select role_contain("specialist_role", "intern_role")
-----------
0
```

使用法

システム関数 role_contain は、*role1* が *role2* に含まれている場合には 1 を返します。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが role_contain 関数を実行できます。

参照

システム関数の詳細については、「システム関数」(69 ページ) を参照してください。

マニュアル　含まれている役割と役割階層の詳細については、『システム管理ガイド』を参照してください。

関数　mut_excl_roles, proc_role, role_id, role_name

コマンド　alter role

システム・プロシージャ　sp_activeroles, sp_displayroles, sp_role

role_id

説明

指定した名前を持つ役割のシステム役割 ID を返します。

構文

role_id("*role_name*")

パラメータ

role_name

システム定義またはユーザ定義の役割の名前です。役割名と役割 ID は、syssrvroles システム・テーブルに格納されています。

例

例 1 sa_role のシステム役割 ID を返します。

```
select role_id("sa_role")
------
0
```

例 2 “intern_role” のシステム役割 ID を返します。

```
select role_id("intern_role")
------
6
```

使用法

- システム関数 role_id は、システム役割 ID (srid) を返します。システム役割 ID は、syssrvroles システム・テーブルの srid カラムに格納されています。
- *role_name* がシステムの有効な役割でない場合、Adaptive Server は NULL を返します。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが role_id を実行できます。

参照

システム関数の詳細については、「システム関数」(69 ページ) を参照してください。

マニュアル 役割の詳細については、『システム管理ガイド』を参照してください。

関数 mut_excl_roles,proc_role,role_contain, role_name

role_name

説明　指定したシステム役割 ID の名前を返します。

構文　role_name(*role_id*)

パラメータ　*role_id*

役割のシステム役割 ID (srid) です。役割名は syssrvroles に格納されています。

例

```
select role_name(01)

----------------------------
sso_role
```

使用法　システム関数 role_name は役割名を返します。

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　すべてのユーザが role_name を実行できます。

参照　システム関数の詳細については、「システム関数」(69 ページ) を参照してください。

関数　mut_excl_roles, proc_role, role_contain, role_id

round

説明

指定した数値を、指定した小数桁数に丸めた値を返します。

構文

round(*number*, *decimal_places*)

パラメータ

number

真数値 (numeric、dec、decimal、tinyint、smallint、int、または bigint 型)、概数値 (float、real、または double precision 型)、money 型のカラム、変数、定数式、またはこれらの組み合わせのいずれかです。

decimal_places

丸めたあとの小数桁数値です。

例

例 1

```
select round(123.4545, 2)
----------
  123.4500
```

例 2

```
select round(123.45, -2)
----------
  100.00
```

例 3

```
select round(1.2345E2, 2)
-----------------
       123.450000
```

例 4

```
select round(1.2345E2, -2)
-----------------
      100.000000
```

使用法

- 算術関数 round は、*number* を有効小数桁数 *decimal_places* まで丸めます。
- *decimal_places* が正の数値の場合は有効小数桁数を示し、*decimal_places* が負の数値の場合は有効整数桁数を示します。
- 結果の型は *number* と同じ型になります。numeric 型または decimal 型の式では、結果の内部精度は第 1 引数に 1 を加算した精度になり、位取りは *number* と同じになります。
- round 関数は常に値を 1 つだけ返します。*decimal_places* が負の数値であり、*number* に指定された有効桁数を超えている場合、Adaptive Server は 0 を返します (小数点以下の 0 の数が numeric の位取りと同じ場合は、この値は 0.00 の形式で表されます)。たとえば、次の式は、0.00 の値を返します。

  ```
  select round(55.55, -3)
  ```

規格	ANSI SQL － 準拠レベル：Transact-SQL 拡張機能
パーミッション	すべてのユーザが round 関数を実行できます。
参照	算術関数の概要については、「算術関数」(67 ページ) を参照してください。 **関数**　abs, ceiling, floor, sign, str

row_count

説明

指定したテーブル内のローの予測数を返します。

構文

row_count(*dbid*, *object_id* [,*ptnid*])

パラメータ

dbid
ターゲット・オブジェクトが存在するデータベースの ID です。

object_id
テーブルのオブジェクト ID です。

ptnid
対象のパーティション ID です。

例

例 1 指定したテーブル内のローの予測数を返します。

```
select row_count(5, 31000114)
```

例 2 オブジェクト ID 31000114 のオブジェクトにある指定のパーティション (パーティション ID 2323242432) に存在するローの予測数を返します。

```
select row_count(5, 31000114, 2323242432)
```

使用法

すべてのエラーで常に値 0 が返されます。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが row_count を実行できます。

参照

関数 reserved_pages, used_pages

rtrim

説明　指定した式から後続ブランクを削除した結果を返します。

構文　rtrim(*char_expr* | *uchar_expr*)

パラメータ

char_expr
文字型のカラム名、変数、または (char、varchar、nchar、または nvarchar 型の) 定数式です。

uchar_expr
文字型のカラム名、変数、または (unichar または univarchar 型の) 定数式です。

例

```
select rtrim("abcd    ")

--------
abcd
```

使用法

- 文字列関数 rtrim は、後続ブランクを削除します。
- Unicode では、ブランクは Unicode 値 U+0020 として定義されています。
- *char_expr* または *uchar_expr* が NULL の場合は、NULL を返します。
- 現在の文字セットのスペース文字に相当する値だけを削除します。

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　すべてのユーザが rtrim 関数を実行できます。

参照　文字列関数の概要については、「文字列関数」(68 ページ) を参照してください。

関数　ltrim

sdc_intempdbconfig

説明

クラスタ環境のみ － システムが現在、テンポラリ・データベース設定モードにある場合は 1 を返し、それ以外の場合は 0 を返します。

構文

sdc_intempdbconfig()

例

```
select sdc_intempdbconfig()
```

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが sdc_intempdbconfig を実行できます。

set_appcontext

説明

ユーザ・セッション用のアプリケーション・コンテキスト名、属性名、属性値を設定します。それらは、指定したアプリケーションの属性によって定義されます。set_appcontext は Application Context Facility (ACF) から提供される組み込み関数です。

構文

set_appcontext(*context_name*, *attribute_name*・*attribute_value*・

パラメータ

context_name
アプリケーション・コンテキスト名を指定するローです。データ型 char(30) として格納されます。

attribute_name
アプリケーション・コンテキストの属性名を指定するローです。データ型 char(30) として格納されます。

attribute_value
アプリケーションの属性値を指定するローです。データ型 char(30) として格納されます。

例

例 1 CONTEXT1 という名前のアプリケーション・コンテキストを作成し、それに属性 ATTR1 とその値 VALUE1 を設定します。

```
select set_appcontext("CONTEXT1", "ATTR1", "VALUE1")

---------------
0
```

既存のアプリケーション・コンテキストを上書きしようとすると、次のようになります。

```
select set_appcontext("CONTEXT1", "ATTR1", "VALUE1")

-------------
-1
```

例 2 値のデータ型変換を含む set_appcontext を示します。

```
declare@numericvarchar varchar(25)
select @numericvar = "20"
select set_appcontext ("CONTEXT1", "ATTR2",
convert(char(20), @numericvar))

------------
0
```

例 3 適切なパーミッションを持たないユーザがアプリケーション・コンテキストを設定しようとしたときの結果を示します。

```
select set_appcontext("CONTEXT1", "ATTR2", "VALUE1")
--------------
-1
```

使用法

- set_appcontext は、成功すると 0 を返し、失敗すると -1 を返します。
- 現在のセッションにすでに存在する値を設定すると、set_appcontext は -1 を返します。
- この関数では、既存のアプリケーション・コンテキストの値は上書きできません。コンテキストに新しい値を割り当てるには、コンテキストを削除してから、新しい値を使用してコンテキストを再作成してください。
- set_appcontext は、属性を char データ型として格納します。属性値を他のデータ型と比較する必要があるアクセス・ルールを作成する場合は、アクセス・ルールで char データを適切なデータ型に変換します。
- この関数では、すべての引数が必須です。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

パーミッションは、ACF に格納されているユーザ・プロファイルとアプリケーション・プロファイルに依存します。

参照

マニュアル ACF の詳細については、『システム管理ガイド』の「第 17 章 ユーザ・パーミッションの管理」の「ロー・レベル・アクセス制御の使用」を参照してください。

関数 get_appcontext, list_appcontext, rm_appcontext

show_role

説明　ログインの現在有効なシステム定義役割を表示します。

構文　show_role()

パラメータ　なし。

例

例 1

```
select show_role()

sa_role sso_role oper_role replication_role
```

例 2

```
if charindex("sa_role", show_role()) >0
begin
    print "You have sa_role"
end
```

使用法

- システム関数 show_role は、ログインの現在有効なシステム定義の役割 (sa_role、sso_role、oper_role、または replication_role) があればそれを返します。ログインに役割がない場合、show_role は NULL を返します。
- データベース所有者が setuser を実行したあとで show_role 関数を呼び出すと、show_role 関数は setuser で指定したユーザではなく、データベース所有者の有効な役割を表示します。

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　すべてのユーザがシステム関数 show_role を実行できます。

参照　システム関数の概要については、「システム関数」(69 ページ) を参照してください。

コマンド　alter role, create role, drop role, grant, set, revoke

関数　proc_role, role_contain

システム・プロシージャ　sp_activeroles, sp_displayroles, sp_role

show_sec_services

説明 セッションに有効なセキュリティ・サービスをリストします。

構文 show_sec_services()

パラメータ なし。

例 ユーザの現在のセッションがデータの暗号化とリプレイの検出のチェックを実行していることを示します。

```
select show_sec_services()

encryption, replay_detection
```

使用法

- セッション中の有効なセキュリティ・サービスをリストにするには、show_sec_services 関数を使用します。
- 有効なセキュリティ・サービスがない場合、show_sec_services 関数は NULL を返します。

規格 ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション すべてのユーザが show_sec_services 関数を実行できます。

参照 **関数** is_sec_service_on

sign

説明　指定した値の符号 (正の 1、0、または負の -1) を返します。

構文　sign(*numeric*)

パラメータ　*numeric*

真数値 (numeric、dec、decimal、tinyint、smallint、int、または bigint 型)、概数値 (float、real、または double precision 型)、money 型のカラム、変数、定数式、またはこれらの組み合わせのいずれかです。

例

例 1

```
select sign(-123)
-----------
         -1
```

例 2

```
select sign(0)
-----------
          0
```

例 3

```
select sign(123)
-----------
          1
```

使用法

- 算術関数 sign は、正 (1)、ゼロ (0)、または負 (-1) を返します。
- 結果は、引数の数値式と同じ型、精度、位取りになります。

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　すべてのユーザが sign 関数を実行できます。

参照　算術関数の概要については、「算術関数」(67 ページ) を参照してください。

関数　abs, ceiling, floor, round

sin

説明

指定した角度 (ラジアン) の正弦を返します。

構文

sin(*approx_numeric*)

パラメータ

approx_numeric

概数値型 (float、real、または double precision) のカラム名、変数、または定数式のいずれかです。

例

```
select sin(45)
--------------------
            0.850904
```

使用法

算術関数 sin は、指定した角度 (ラジアン) の正弦を返します。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが sin 関数を実行できます。

参照

算術関数の概要については、「算術関数」(67 ページ) を参照してください。

関数 cos, degrees, radians

sortkey

説明

照合動作をもとに、結果を並べ替えるために使用する値を生成します。これにより、デフォルト・セットのラテン文字以外の文字セットについても、辞書順、大文字と小文字の区別、アクセントの区別をもとにした文字列照合動作が可能になります。

構文

sortkey(*char_expression* | *uchar_expression*)[, {*collation_name* | *collation_ID*}])

パラメータ

char_expression

文字型のカラム名、変数、または (char、varchar、nchar、または nvarchar 型の) 定数式です。

uchar_expression

文字型のカラム名、変数、または (unichar または univarchar 型の) 定数式です。

collation_name

引用符で囲まれた文字列、または使用する照合動作を指定する文字変数です。表 2-11 (262 ページ) に、有効値を示します。

collation_ID

使用する照合動作を指定する整数定数または変数です。表 2-11 (262 ページ) に、有効値を示します。

例

例 1 西欧言語辞書順のソートを示します。

```
select * from cust_table where cust_name like "TI%" order by
    (sortkey(cust_name, "dict")
```

例 2 中国語 (簡体字) の読み方順のソートを示します。

```
select *from cust_table where cust name like "TI%" order by
    (sortkey(cust-name, "gbpinyin")
```

例 3 インライン・オプションを使用した西欧言語辞書順のソートを示します。

```
select *from cust_table where cust_name like "TI%" order by cust_french_sort
```

例 4 既存のキーを使用した中国語 (簡体字) の読み方順のソートを示します。

```
select * from cust_table where cust_name like "TI%" order by
    cust_chinese_sort.
```

使用法

- システム関数 sortkey は、照合動作をもとに、結果を並べ替えるために使用する値を生成します。これにより、デフォルト・セットのラテン文字以外の文字セットについても、辞書順、大文字と小文字の区別、アクセントの区別をもとにした文字列照合動作が可能になります。戻り値は varbinary データ型値で、sortkey 関数から返される入力文字列に対応するコード化照合情報が含まれています。

  たとえば、sortkey によって返された値を、ソース文字列を含むカラムに格納できます。文字列データを希望の順序で検索する場合は、select 文に、sortkey の実行結果を含むカラムに関する order by 句を含めます。

  sortkey により、特定の照合基準のセットに対して返される値が、varbinary データ型について実行されるバイナリ比較に使用できるようになります。

- sortkey は、それぞれの入力文字について 6 バイトの照合情報を生成します。したがって、sortkey の実行結果は、varbinary データ型の長さの制限値である 255 バイトを超える可能性があります。この制限値を超える場合、結果は制限値に収まるようにトランケートされます。この制限はサーバの論理ページ・サイズに依存するので、DOL テーブルおよび APL テーブルでは、結果の文字列が次のサイズより小さくなるまで、各入力文字の結果バイトはトランケーションによって削除されます。

表 *2-10:* ローとカラムの最大長 – *APL* テーブルおよび *DOL* テーブル

ロック・スキーム	ページ・サイズ	ローの最大長	カラムの最大長
APL テーブル	2K (2048 バイト)	1962	1960 バイト
	4K (4096 バイト)	4010	4008 バイト
	8K (8192 バイト)	8106	8104 バイト
	16K (16384 バイト)	16298	16296 バイト
DOL テーブル	2K (2048 バイト)	1964	1958 バイト
	4K (4096 バイト)	4012	4006 バイト
	8K (8192 バイト)	8108	8102 バイト
	16K (16384 バイト)	16300	16294 バイト テーブルに可変長カラムがない場合
	16K (16384 バイト)	16300 (varlen = 8191 の最大開始オフセットによって変化)	8191-6-2 = 8183 バイト 1 つ以上の可変長カラムがテーブルにある場合 *

* このサイズには、ローのオーバヘッドの 6 バイトとローの長さのフィールドの 2 バイトが含まれる。

このような状態が発生した場合は、Adaptive Server によって警告メッセージが表示されますが、sortkey 関数を含むクエリまたはトランザクションは引き続き実行されます。

- *char_expression* または *uchar_expression* は、サーバのデフォルトの文字セットでコード化されている文字の組み合わせでなければなりません。
- *char_expression* または *uchar_expression* は、空文字列でもかまいません。空の文字列の場合、sortkey は長さがゼロの varbinary 値を返し、空の文字列にブランクを格納します。

  空文字列は、データベース・カラムの NULL 文字列とは異なる照合値を持ちます。
- *char_expression* または *uchar_expression* が NULL の場合、sortkey は null を返します。
- unicode 式にソート順序が指定されていない場合は、binary ソート順序が使用されます。
- *collation_name* または *collation_ID* に値を指定しない場合、sortkey はバイナリ照合とみなします。
- sortkey 関数によって生成されるバイナリ値は、Adaptive Server のメジャー・バージョン間で異なる可能性があります (たとえば、バージョン 12.0 と 12.5、バージョン 12.9.2 と 12.0 など)。現在のバージョンの Adaptive Server にアップグレードする場合は、キーを再生成して隠しカラムを再配置してから、バイナリ比較を実行する必要があります。

注意 バージョン 12.5 から 12.5.0.1 へのアップグレードでは、この手順は不要で、キーを再生成しなくてもエラー・メッセージや警告メッセージは表示されません。隠しカラムを持つクエリは正常に実行されますが、その比較結果はアップグレード前のサーバとは異なる可能性があります。

照合テーブル

マルチ言語ソートを実行するときには、次の 2 種類の照合テーブルを使用できます。

1　sortkey 関数で作成された「組み込み」照合テーブル。照合名または照合 ID を使用して、組み込みテーブルを指定できます。

2　Unilib ライブラリ・ソート機能を使用する外部照合テーブル。外部テーブルを指定するときには、照合名を使用します。これらのファイルは、*$SYBASE/collate/unicode* に格納されています。

どちらのテーブルを使用しても適切に機能します。ただし、「組み込み」テーブルは Adaptive Server データベースに関連付けられていますが、外部テーブルは関連付けられていません。Adaptive Server データベースを使用する場合は、組み込みテーブルを使用するとパフォーマンスを最大限に引き出すことができます。どちらのテーブルでも、英語、西欧言語、アジア言語を組み合わせて使用できます。

sortkey は、次の 2 とおりの方法で使用できます。

1 インライン － order by 句に sortkey を指定する方法。変更内容を最小限に抑えて既存のアプリケーションを更新する場合に役立ちます。ただし、この方法では実行中にソート・キーが生成されるため、レコード数が 1000 を超える大きなデータセットでは、最適なパフォーマンスを実現できません。

2 既存のキー － 新規顧客名などの新しいレコードで、テーブルに多言語ソート機能を追加する必要がある場合に sortkey を呼び出す方法。データベース内に隠しカラム (binary または varbinary 型) を設定する必要があります。この場合、使用するソート順 (フランス語、中国語の順など) ごとに 1 つのカラムを、同一テーブル内に設定することをおすすめします。クエリの出力をソートする必要がある場合は、order by 句に隠しカラムを 1 つ指定します。この方法では、パフォーマンスを最大限に引き出すことができます。これは、キーがすでに生成、ソートされており、バイナリ値にもとづいて比較を迅速に実行できるためです。

使用可能な照合ルールのリストを表示できます。このリストを出力するには、sp_helpsort を実行するか、または syscharsets の name、id、description を問い合わせて選択します (type は 2003 ～ 2999 の間です)。

- 表 2-11 は、*collation_name* と *collation_ID* の有効値のリストを示します。

表 *2-11:* 照合名と *ID*

説明	照合名	照合 ID
デフォルト Unicode マルチ言語	default	20
タイ語辞書順	thaidict	21
ISO14651 規格	iso14651	22
UTF-16 順 － UTF-8 バイナリ順と一致	utf8bin	24
CP 850 代替言語 － アクセント記号なし	altnoacc	39
CP 850 代替言語、小文字優先	altdict	45
CP 850 西欧言語 － 大文字／小文字の優先設定なし	altnocsp	46
CP 850 スカンジナビア語 － 辞書順	scandict	47
CP 850 スカンジナビア語辞書順 － 大文字と小文字を区別しない、優先度を付けた順位	scannocp	48
GB ピンイン	gbpinyin	なし
バイナリ・ソート	binary	50
Latin-1 英語、フランス語、ドイツ語辞書	dict	51
Latin-1 英語、フランス語、ドイツ語、大文字／小文字の区別なし	nocase	52
Latin-1 英語、フランス語、ドイツ語、大文字／小文字の優先設定なし	nocasep	53
Latin-1 英語、フランス語、ドイツ語、アクセント記号なし	noaccent	54
Latin-1 スペイン語辞書	espdict	55
Latin-1 スペイン語、大文字／小文字の区別なし	espnocs	56

説明	照合名	照合 ID
Latin-1 スペイン語、アクセント記号なし	espnoac	57
ISO 8859-5 ロシア語辞書	rusdict	58
ISO 8859-5 ロシア語、大文字小文字の区別なし	rusnocs	59
ISO 8859-5 キリル語辞書	cyrdict	63
ISO 8859-5 キリル語、大文字小文字の区別なし	cyrnocs	64
ISO 8859-7 ギリシア語辞書	elldict	65
ISO 8859-2 ハンガリー語、辞書	hundict	69
ISO 8859-2 ハンガリー語、アクセント記号の区別なし	hunnoac	70
ISO 8859-2 ハンガリー語、大文字小文字の区別なし	hunnocs	71
ISO 8859-9 トルコ語辞書	turdict	72
ISO 8859-9 トルコ語、アクセント記号の区別なし	turknoac	73
ISO 8859-9 トルコ語、大文字小文字の区別なし	turknocs	74
CP932 バイナリ順	cp932bin	129
中国語、読み方順	dynix	130
GB2312 バイナリ順	gb2312bn	137
共通キリル語辞書	cyrdict	140
トルコ語辞書	turdict	155
EUCKSC バイナリ順	euckscbn	161
中国語、読み方順	gbpinyin	163
ロシア語辞書順	rusdict	165
SJIS バイナリ順	sjisbin	179
EUCJIS バイナリ順	eucjisbn	192
BIG5 バイナリ順	big5bin	194
Shift-JIS バイナリ順	sjisbin	259

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　すべてのユーザが sortkey 関数を実行できます。

参照　**関数**　compare

soundex

説明

式の発音を表す 4 文字コードを返します。

構文

soundex(*char_expr* | *uchar_expr*)

パラメータ

char_expr

文字型のカラム名、変数、または (char、varchar、nchar、または nvarchar 型の) 定数式です。

uchar_expr

文字型のカラム名、変数、または (unichar または univarchar 型の) 定数式です。

例

```
select soundex ("smith"), soundex ("smythe")
----- -----
S530  S530
```

使用法

- 文字列関数 soundex は、一連の有効なシングルバイトまたはダブルバイトのローマ字からなる文字列の 4 文字 soundex コードを返します。
- soundex 関数は、アルファベット文字列を 4 桁コードに変換します。このコードは、発音が似ている単語や名前を検索するために使用されます。文字列の先頭文字以外のすべての母音は無視されます。
- *char_expr* または *uchar_expr* が NULL の場合は、は NULL を返します。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが soundex 関数を実行できます。

参照

文字列関数の概要については、「文字列関数」(68 ページ) を参照してください。

関数 difference

space

説明　指定した数のシングルバイト・スペースからなる文字列を返します。

構文　space(*integer_expr*)

パラメータ　*integer_expr*

整数値型 (tinyint、smallint、または int) のカラム名、変数、定数式のいずれかです。

例

```
select "aaa", space(4), "bbb"

--- ---- ---
aaa      bbb
```

使用法　文字列関数 space は、指定した数のシングルバイト・スペースからなる文字列を返します。

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　すべてのユーザが space 関数を実行できます。

参照　文字列関数の概要については、「文字列関数」(68 ページ) を参照してください。

関数　isnull, rtrim

spid_instance_id

説明

クラスタ環境のみ − 指定したプロセス ID (spid) を実行しているインスタンスの ID を返します。

構文

spid_instance_id(*spid_value*)

パラメータ

spid_value
インスタンス ID を要求する spid 番号

例

プロセス ID 番号 27 をインスタンスの ID を返します。

```
select spid_instance_id(27)
```

使用法

- spid の値を含めない場合、spid_instance_id は NULL を返します。
- 無効な値や存在しないプロセス ID 値を入力すると、spid_instance_id は NULL を返します。

規格

ANSI SQL − 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが spid_instance_id を実行できます。

square

説明　指定した値の平方を、float で表したものを返します。

構文　square(*numeric_expression*)

パラメータ　*numeric_expression*
float 型の数値式です。

例　**例 1** 整数カラムの平方を返します。

```
select square(total_sales)from titles
------------
16769025.00000
15023376.00000
350513284.00000
...
16769025.00000
(18 row(s) affected)
```

例 2 通貨カラムの平方を返します。

```
select square(price) from titles
-----------
399.600100
142.802500
8.940100
NULL
...
224.700100
(18 row(s) affected)
```

使用法　この関数は power(*numeric_expression*,2) と同じですが、int 型ではなく float 型を返します。

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　すべてのユーザが square 関数を実行できます。

参照　**関数**　power

データ型　exact_numeric、approximate_numeric、money、float

sqrt

説明

指定した数値の平方根を返します。

構文

sqrt(*approx_numeric*)

パラメータ

approx_numeric

概数値型 (float、real、または double precision) のカラム名、変数、または正の数になる定数式のいずれかです。

例

```
select sqrt(4)

 2.000000
```

使用法

- 算術関数 sqrt は、指定した値の平方根を返します。
- 負の数値の平方根を求めようとすると、Adaptive Server は次のようなエラー・メッセージを返します。

```
Domain error occurred.
```

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが sqrt 関数を実行できます。

参照

算術関数の概要については、「算術関数」(67 ページ) を参照してください。

関数 power

stddev

説明

1 つの数値式で構成される標本標準偏差を double 型として計算します。

注意 stddev と stdev は、stddev_samp のエイリアスです。詳細については、stddev_samp (273 ページ) を参照してください。

stdev

説明

1 つの数値式で構成される標本標準偏差を double 型として計算します。

注意 stddev と stdev は、stdev のエイリアスです。詳細については、stddev_samp (273 ページ) を参照してください。

stdevp

説明

1 つの数値式で構成される母標準偏差を double 型として計算します。

注意　stdevp は、stddev_pop のエイリアスです。詳細については、stddev_pop (272 ページ) を参照してください。

stddev_pop

説明

1 つの数値式で構成される母標準偏差を double 型として計算します。stdevp は、stddev_pop のエイリアスで、同じ構文を使用します。

構文

stddev_pop ([all | distinct] *expression*)

パラメータ

all
すべての値に stddev_pop を適用します。デフォルト設定は all です。

distinct
stddev_pop を適用する前に重複する値を削除します。

expression
その母集団ベースの標準偏差がローのセットに対して計算される式です (通常はカラム名)。

例

次の文は、pubs2 データベースの書籍の種類ごとに前払い金の平均偏差と標準偏差を示します。

```
select type, avg(advance) as "avg", stddev_pop(advance)
  as "stddev" from titles group by type order by type
```

使用法

グループの各ロー (distinct が指定されている場合、重複が削除された後に残る各ロー) に対して評価される、指定された値式の母標準偏差を計算します。これは、母分散の平方根として定義されます。

図 *2-3:* 母集団に関係する統計集合関数の計算式

平均 μ (var_pop) を持つサイズ *n* の母分散を定義する計算式は、次のとおりです。母標準偏差値 (stddev_pop) は、この数値の正の平方根です。

$$\sigma^2 = \frac{\sum (x_i - \mu)^2}{n}$$

σ^2 = 分散
n = 母集団のサイズ
$\mu = x_i$ 値の平均

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが stddev_pop を実行できます。

参照

集合関数の概要については、「集合関数」(48 ページ) を参照してください。

関数 stddev_samp, var_pop, var_samp

stddev_samp

説明

1 つの数値式で構成される標本標準偏差を double 型として計算します。stdev と stddev は、stddev_samp のエイリアスで、同じ構文を使用します。

構文

stddev_samp ([all | distinct] *expression*)

パラメータ

all
　すべての値に stddev_samp を適用します。デフォルト設定は all です。

distinct
　stddev_samp を適用する前に重複する値を削除します。

expression
　任意の numeric データ型 (float、real、または double precision) の式です。

例

次の文は、pubs2 データベースの書籍の種類ごとに前払い金の平均偏差と標準偏差を示します。

```
select type, avg(advance) as "avg",
  stddev_samp(advance) as "stddev" from titles
  where total_sales > 2000 group by type order by type
```

使用法

グループの各ロー (distinct が指定されている場合、重複が削除された後に残る各ロー) に対して評価される、指定された値式の標本標準偏差を計算します。これは、標本分散の平方根として定義されます。

図 *2-4:* 標本に関係する統計集合関数の計算式

平均が $\bar{x}$ (var_samp) のサイズ n の標本からの母分散の不偏推定値を定義する計算式は、次のとおりです。標本標準偏差 (stddev_samp) は、この数値の正の平方根です。

$$s^2 = \frac{\sum (x_i - \bar{x})^2}{n - 1}$$

s^2 = 分散
n = 標本サイズ
$\bar{x} = x_i$ 値の平均

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが stddev_samp を実行できます。

参照

集合関数の概要については、「集合関数」(48 ページ) を参照してください。

関数　stddev_pop, var_pop, var_samp

str

説明

指定した数値に相当する文字を返します。

構文

str(*approx_numeric*[, *length* [, *decimal*]])

パラメータ

approx_numeric

概数値型 (float、real、または double precision) のカラム名、変数、または定数式のいずれかです。

length

返す文字の数 (小数点、小数桁、整数桁、ブランクを含む) を指定します。デフォルトは 10 です。

decimal

返す文字の数 (小数点、小数桁、整数桁、ブランクを含む) を指定します。デフォルトは 10 です。

例

例 1

```
select str(1234.7, 4)

----
1235
```

例 2

```
select str(-12345, 6)

------
-12345
```

例 3

```
select str(123.45, 5, 2)

-----
123.5
```

使用法

- 文字列関数 str は、浮動小数点数値を文字で表したものを返します。文字列関数の一般的な説明については、「文字列関数」(68 ページ) を参照してください。
- *length* と *decimal* はオプションです。使用する場合は、負ではない値を指定してください。str 関数は、結果が指定した長さになるよう数値の小数部を丸めます。長さは小数点や数字の符号 (負数の場合) が十分入る長さにしてください。結果の小数部分は、指定された長さに収まるように丸められます。ただし、数字の整数部分が指定された長さを超える場合、str は指定された長さ分のアスタリスクのローを返します。たとえば、次のようにします。

  ```
  select str(123.456, 2, 4)
  --
  **
  ```

short 型の *approx_numeric* は、指定された長さの範囲で右揃えにされ、long 型の *approx_numeric* は指定された 10 進の桁数にトランケートされます。

- *approx_numeric* が NULL の場合、NULL を返します。

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　すべてのユーザが str 関数を実行できます。

参照　**関数**　abs, ceiling, floor, round, sign

str_replace

説明

最初の文字列式 (*string_expression1*) 内に出現する 2 番目の文字列式 (*string_expression2*) のすべてのインスタンスを、3 番目の式 (*string_expression3*) で置き換えます。

構文

str_replace("*string_expression1*", "*string_expression2*", "*string_expression3*")

パラメータ

string_expression1

検索される側のソース文字列または文字列式です。char、varchar、unichar、univarchar、varbinary、または binary データ型として表されます。

string_expression2

最初の式 (*string_expression1*) 内で検索するパターン文字列または文字列式です。*string_expression2* は char、varchar、unichar、univarchar、varbinary、または binary データ型として表されます。

string_expression3

2 番目の式を置き換える文字列式です。char、varchar、unichar、univarchar、binary、または varbinary データ型として表されます。

例

例 1 文字列 *cdefghi* 内の文字列 *def* を *yyy* に置き換えます。

```
str_replace("cdefghi","def","yyy")
-------------
cyyyghi
(1 row(s) affected)
```

例 2 すべてのスペースを "toyota" に置き換えます。

```
select str_replace("chevy, ford, mercedes", "","toyota")
----------
chevy,toyotaford,toyotamercedes
(1 row(s) affected)
```

注意 Adaptive Server では、空の文字列を NULL 値と区別するために、空の文字列定数は自動的に 1 つのスペース文字列に変換されます。

例 3 “abcghijklm” を返します。

```
select str_replace("abcdefghijklm", "def", NULL)

----------
abcghijklm
(1 row affected)
```

使用法

- *string_expression* (1、2、または 3) が char または varchar の場合は、varchar データを返します。
- *string_expression* (1、2、または 3) が unichar または univarchar の場合は、univarchar データを返します。
- *string_expression* (1、2、または 3) が binary または varbinary の場合は、varbinary データを返します。
- すべての引数を同じデータ型にしてください。
- 3 つの引数のいずれかが NULL の場合は、null を返します。

  str_replace では、3 つ目のパラメータに NULL を使用できるようになりました。この場合、*string_expression2* が NULL に置き換えられます。この方法で str_replace を使用して文字列を削除する操作を行うことができます。

  たとえば、次の例は “abcghijklm” を返します。

  ```
  str_replace("abcdefghijklm", "def", NULL)
  ```

- 結果の長さは、式がコンパイルされるときに、引数値が既知かどうかによって異なります。すべての引数が既知の定数値を持つ変数の場合は、結果の長さは、Adaptive Server によって次のように計算されます。

  ```
  result_length = ((s/p)*(r-p)+s)
  where
  s = length of source string
  p = length of pattern string
  r = length of replacement string
  if (r-p) <= 0, result length = s
  ```

- ソース文字列 (*string_expression1*) がカラムで、*string_expression2* と *string_expression3* がコンパイル時に既知の定数値である場合は、結果の長さは上記の計算式を使用して計算されます。
- 式のコンパイル時に引数値がわからないため Adaptive Server が結果の長さを計算できない場合は、トレース・フラグ 244 が on でないかぎり、結果の長さは 255 になります。トレース・フラグ 244 が on の場合は、結果の長さは 16384 になります。
- result_len が 16384 を超えることはありません。

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　すべてのユーザが str_replace 関数を実行できます。

参照

データ型　char、varchar、binary、varbinary、unichar、univarchar

関数　length

strtobin

説明

一連の英数字をそれに相当する 16 進数に変換します。

構文

select strtobin(”*string of valid alphanumeric characters*”)

パラメータ

string of valid alphanumeric characters

[1 - 9]、[a - f]、および [A - F] からなる有効な英数字の文字列です。

例

例 1 英数字「723ad82fe」を 16 進の数列に変換します。

```
select strtobin("723ad82fe")
go

-----------------------------------
0x0723ad82fe
```

この例では、英数字文字列とそれに相当する 16 進数のメモリ内表現は次のようになります。

英数字文字列 (9 バイト)

0	7	2	3	a	d	8	2	f	e

16 進数 (5 バイト)

0	7	2	3	a	d	8	2	f	e

この関数は文字を右から左へと処理します。この例では、入力の文字数は奇数です。そのため、16 進の数列に「0」のプレフィクスが付いて出力に反映されます。

例 2 *@str_data* というローカル変数の英数字文字列を「723ad82fe」の値に相当する 16 進の数列に変換します。

```
declare @str_data varchar(30)
select @str_data = "723ad82fe"
select strtobin(@str_data)
go

----------
0x0723ad82fe
```

使用法

- 入力の無効な文字は、出力として NULL になります。
- 16 進数の入力シーケンスには “0x” のプレフィクスが必要です。
- NULL の入力結果は NULL の出力になります。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

strtobin は、すべてのユーザが実行できます。

参照

関数 bintostr

stuff

説明

1 つの文字列から指定した数の文字を削除して、代わりに別の文字列を入れて作成した文字列を返します。

構文

stuff(*char_expr1* | *uchar_expr1*, *start*, *length*, *char_expr2* | *uchar_expr2*)

パラメータ

char_expr1
文字型のカラム名、変数、または (char、varchar、nchar、または nvarchar 型の) 定数式です。

uchar_expr1
文字型のカラム名、変数、または (unichar または univarchar 型の) 定数式です。

start
削除する文字の先頭位置を指定します。

length
削除する文字の数を指定します。

char_expr1
別の文字型のカラム名、変数、または (char、varchar、nchar、または nvarchar 型の) 定数式です。

uchar_expr2
別の文字型のカラム名、変数、または (unichar または univarchar 型の) 定数式です。

例

例 1

```
select stuff("abc", 2, 3, "xyz")

----
axyz
```

例 2

```
select stuff("abcdef", 2, 3, null)

go
---
aef
```

例 3

```
select stuff("abcdef", 2, 3, "")

----
a ef
```

使用法

- 文字列関数 stuff は、*char_expr1* または *uchar_expr1* の位置 *start* から *length* の数の文字を削除し、次に *char_expr2* または *uchar_expr2* を、*char_expr1* または *uchar_expr2* の *start* の位置に挿入します。文字列関数の一般的な説明については、「文字列関数」(68 ページ) を参照してください。
- 開始位置または長さが負の場合は、NULL 文字列が返されます。先頭位置がゼロまたは *expr1* より長い場合、NULL 文字列を返します。長さが *expr1* より長いときは、*expr1* は最後の文字まで削除されます (例 1 を参照)。
- 指定した先頭位置がサロゲート・ペアの中央にある場合、先頭位置は指定の位置より 1 文字少ない位置に調整されます。指定した長さの終わりの位置がサロゲート・ペアの中央にある場合、長さは指定の長さより 1 文字少なくなります。
- stuff 関数を使用して文字を削除するには、*expr2* には空の引用符ではなく NULL を使用します。“ ” を使用して null 文字を指定すると、スペースに置き換わります (例 2 と 3 を参照)。
- *char_expr1* または *uchar_expr1* が NULL のとき、stuff は NULL を返します。*char_expr1* または *or uchar_expr1* が文字列値で *char_expr2* または *uchar_expr2* が NULL の場合、stuff は、削除された文字の代わりに何も挿入しません。
- varchar 式と unichar 式をそれぞれパラメータとして指定した場合、varchar 式は暗黙的に unichar へ変換されます (変換時にトランケーションが発生することがあります)。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが stuff 関数を実行できます。

参照

関数 replicate, substring

substring

説明

指定した数の文字を文字列から取り出して作った文字列を返します。

構文

substring(*expression*, *start, length*)

パラメータ

expression

バイナリ型または文字型のカラム名、変数、または定数式です。char、nchar、unichar、varchar、univarchar、または nvarchar のデータ、binary または varbinary を使用できます。

start

部分文字列の先頭文字の位置を指定します。

length

部分文字列の文字数を指定します。

例

例 1 “Bennet A” のように、各作者の姓と名前の頭文字を表示します。

```
select au_lname, substring(au_fname, 1, 1)
from authors
```

例 2 作者の姓を大文字に変換し、最初の 3 文字を表示します。

```
select substring(upper(au_lname), 1, 3)
from authors
```

例 3 pub_id と title_id を連結して作成された文字列の最初の 6 文字を表示します。

```
select substring((pub_id + title_id), 1, 6)
  from titles
```

例 4 各桁が 2 つのバイナリ値を表すバイナリ・フィールドから、下 4 桁を取り出します。

```
select substring(xactid,5,2)
from syslogs
```

使用法

- 文字列関数 substring は、文字列またはバイナリ文字列の一部分を返します。文字列関数の一般的な説明については、「文字列関数」(68 ページ) を参照してください。
- substring の 2 番目の引数が NULL の場合、結果は NULL です。substring の最初および 3 番目の引数が NULL の場合、結果はブランクです。
- *uchar_expr1* の式で指定の先頭位置がサロゲート・ペアの中央にある場合、*start* は指定の位置より 1 文字少ない位置に調整されます。*uchar_expr1* の式で長さの終わりの位置がサロゲート・ペアの中央にある場合、*length* は指定の長さより 1 文字少なくなります。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが substring 関数を実行できます。

参照

関数　charindex, patindex, stuff

sum

説明

値の合計を返します。

構文

sum([all | distinct] *expression*)

パラメータ

all

すべての値に sum を適用します。デフォルト設定は all です。

distinct

sum を適用する前に、重複する値を削除します。distinct はオプションです。

expression

カラム名、定数、関数、これらを算術演算子やビット処理演算子でつないだ任意の組み合わせ、またはサブクエリです。通常、集合関数では、expression はカラム名です。詳細については、「式」(341 ページ) を参照してください。

例

例 1 すべてのビジネス雑誌について前払い金の平均と総売上額を計算します。これらの各集合関数は、検索したすべてのローに対して 1 つの合計値を返します。

```
select avg(advance), sum(total_sales)
from titles
where type = "business"
```

例 2 group by 句を指定すると、集合関数はテーブル全体ではなくグループごとに 1 つの値を計算します。この文では、本の種類ごとに合計値を計算します。

```
select type, avg(advance), sum(total_sales)
from titles
group by type
```

例 3 titles テーブルは出版社別のグループに分けられ、前払い総額が 25,000 ドルを超え、本の平均価格が 15 ドルを超える出版社のグループだけが合計されます。

```
select pub_id, sum(advance), avg(price)
from titles
group by pub_id
having sum(advance) > $25000 and avg(price) > $15
```

使用法

- 集合関数 sum は、カラム内のすべての値の合計を求めます。sum 関数を使用できるのは、数値 (整数、浮動小数点、または通貨) データ型だけです。null 値は合計では無視されます。
- 集合関数の概要については、「集合関数」(48 ページ) を参照してください。
- 整数データを合計すると、カラムのデータ型が smallint または tinyint であっても、Adaptive Server では結果が int 値として扱われます。bigint のデータを合計すると、結果は bigint 値として扱われます。DB-Library プログラムでのオーバフロー・エラーを避けるために、平均や合計の結果が格納されるすべての変数を適切に宣言してください。
- バイナリ・データ型では sum 関数は使用できません。
- この関数では数値型だけを定義します。Unicode 式に対してこの関数を実行すると、エラーが発生します。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが sum 関数を実行できます。

参照

コマンド　compute 句、group 句と having 句、select、where 句

関数　count, max, min

suser_id

説明

syslogins テーブルからサーバ・ユーザの ID 番号を返します。

構文

suser_id([*server_user_name*])

パラメータ

server_user_name
Adaptive Server のログイン名です。

例

例 1

```
select suser_id()

------
     1
```

例 2

```
select suser_id("margaret")

------
     5
```

使用法

- システム関数 suser_id は、syslogins からサーバ・ユーザの ID 番号を返します。システム関数の概要については、「システム関数」(69 ページ) を参照してください。
- 特定のデータベース内のユーザ ID を sysusers テーブルから検索するには、システム関数 user_id を使用します。
- *server_user_name* を指定しない場合、suser_id は現在のユーザのサーバ ID を返します。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが suser_id 関数を実行できます。

参照

関数 suser_name, user_id

suser_name

説明

現在のサーバ・ユーザの名前、または指定したサーバ ID を持つユーザの名前を返します。

構文

suser_name([*server_user_id*])

パラメータ

server_user_id

Adaptive Server のユーザ ID です。

例

例 1

```
select suser_name()

------------------------------
sa
```

例 2

```
select suser_name(4)

------------------------------
margaret
```

使用法

システム関数 suser_name は、サーバ・ユーザの名前を返します。サーバ・ユーザ ID は syslogins に格納されています。*server_user_id* を指定しない場合、suser_name は現在のユーザの名前を返します。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが suser_name 関数を実行できます。

参照

システム関数の概要については、「システム関数」(69 ページ) を参照してください。

関数　suser_id, user_name

syb_quit

説明 接続を終了します。

構文 syb_quit()

例 関数が実行されている接続を終了し、エラー・メッセージを返します。

```
select syb_quit()

-------------
CT-LIBRARY error:
    ct_results(): network packet layer:
internal net library error: Net-Library operation terminated
due to disconnect
```

使用法 isql プリプロセッサ・コマンド exit でエラーが発生した場合は、syb_quit を使用するとスクリプトを終了できます。

パーミッション すべてのユーザが syb_quit 関数を実行できます。

syb_sendmsg

説明

UNIX のみ　UDP (ユーザ・データグラム・プロトコル) ポートにメッセージを送信します。

構文

syb_sendmsg *ip_address*, *port_number*, *message*

パラメータ

ip_address
UDP アプリケーションが稼働しているマシンの IP アドレスです。

port_number
UDP ポートのポート番号です。

メッセージ
送信されるメッセージです。最大 255 文字数までです。

例

例 1 “Hello” というメッセージを IP アドレス 120.10.20.5 のポート 3456 に送信します。

```
select syb_sendmsg("120.10.20.5", 3456, "Hello")
```

例 2 IP アドレスとポート番号をユーザ・テーブルから読み取り、送信するメッセージの変数を使用します。

```
declare @msg varchar(255)
    select @msg = "Message to send"
    select syb_sendmsg (ip_address, portnum, @msg)
    from sendports
    where username = user_name()
```

使用法

- UDP のメッセージ機能を使用できるようにするには、システム・セキュリティ担当者がallow sendmsg 設定パラメータを 1 に設定する必要があります。
- syb_sendmsg では、セキュリティ・チェックは実行されません。ネットワーク経由で重要な情報を送信する目的には syb_sendmsg を使用しないよう強くおすすめします。この機能を有効にした場合、ユーザは、このメッセージ機能を使用したために発生するあらゆるセキュリティ上の問題を承認したことになります。
- UDP ポートを作成するサンプル C プログラムについては、sp_sendmsg を参照してください。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが syb_sendmsg 関数を実行できます。

参照

システム・プロシージャ　sp_sendmsg

sys_tempdbid

説明

クラスタ環境のみ − 指定したインスタンスの有効なローカル・システム・テンポラリ・データベースの ID を返します。*instance_id* を指定しない場合は、現在のインスタンスの有効なローカル・システム・テンポラリ・データベースの ID を返します。

構文

sys_tempdbid(*instance_id*)

パラメータ

instance_id

インスタンスの ID です。

例

インスタンス ID が 3 のインスタンスの有効なローカル・システム・テンポラリ・データベースの ID を返します。

```
select sys_tempdbid(3)
```

使用法

インスタンス ID を指定しない場合、sys_tempdbid は現在のインスタンスの有効なローカル・システム・テンポラリ・データベースの ID を返します。

規格

ANSI SQL − 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが sys_tempdbid を実行できます。

tan

説明　指定した角度 (ラジアン) の正接を返します。

構文　tan(*angle*)

パラメータ　*angle*

カラム名、変数、または (float、real、double precision、またはこれらの型の 1 つに暗黙的に変換できる任意のデータ型の) 式として表される角度の大きさ (ラジアン) です。

例

```
select tan(60)

--------------------
            0.320040
```

使用法　算術関数 tan は、指定した角度 (ラジアン単位) の正接を返します。

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　すべてのユーザが tan 関数を実行できます。

参照　算術関数の概要については、「算術関数」(67 ページ) を参照してください。

関数　atan, atn2, degrees, radians

tempdb_id

説明

指定のセッションが割り当てられているテンポラリ・データベースをレポートします。tempdb_id 関数の入力はサーバ・プロセス ID であり、出力はそのプロセスが割り当てられているテンポラリ・データベースです。サーバ・プロセスを指定しない場合、tempdb_id は、現在のプロセスに割り当てられているテンポラリ・データベースの dbid をレポートします。

構文

tempdb_id()

例

指定したテンポラリ・データベースに割り当てられているすべてのサーバ・プロセスを検索するには、次を実行します。

```
select spid from master..sysprocesses
    where tempdb_id(spid) = db_id("tempdatabase")
```

使用法

select tempdb_id は、select @@*tempdbid* と同じ結果になります。

参照

コマンド select

textptr

説明

text、image、または unitext カラムの先頭ページを指すポインタを返します。

構文

textptr(*column_name*)

パラメータ

column_name
text カラムの名前です。

例

例 1 textptr 関数を使用して、作者に関する blurbs テーブル内の au_id 486-29-1786 に対応する、text カラムの copy を検索します。テキスト・ポインタはローカル変数 *@val* に配置され、readtext コマンドのパラメータとして指定されます。このコマンドは 2 番目のバイト (オフセット 1) から 5 バイト分を返します。

```
declare @val binary(16)
    select @val = textptr(copy) from blurbs
    where au_id = "486-29-1786"
    readtext blurbs.copy @val 1 5
```

例 2 blurbs テーブルから title_id カラムと、copy カラムの 16 バイト・テキスト・ポインタを返します。

```
select au_id, textptr(copy) from blurbs
```

使用法

- テキストおよびイメージ関数 textptr は、テキスト・ポインタ値 (16 バイトの varbinary 値) を返します。
- text、unitext、または image カラムが、null 以外の insert 文や任意の update 文により初期化されていない場合、textptr 関数は NULL ポインタを返します。テキスト・ポインタが存在するかどうかは、textvalid 関数を使ってチェックします。有効なテキスト・ポインタがない場合は、writetext または readtext を使用できません。

注意 varbinary 値がテーブルに格納される場合、この値の後続の f はトランケートされます。テキスト・ポインタ値をテーブルに格納する場合は、カラムのデータ型として binary を使用します。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが textptr 関数を実行できます。

参照

テキストおよびイメージ関数の概要については、「text 関数および image 関数」(70 ページ) を参照してください。

データ型　text、image、unitext データ型

関数　textvalid

コマンド　insert, update, readtext, writetext

textvalid

説明

指定した text カラムまたは unitext カラムを指すポインタが有効な場合は 1 を返し、無効な場合は 0 を返します。

構文

textvalid("*table_name.column_name*", *textpointer*)

パラメータ

table_name.column_name
テーブルの名前とその text カラムです。

textpointer
テキスト・ポインタ値です。

例

texttest テーブルの blurb カラムの各値に、有効なテーブル・ポインタがあるかどうかをレポートします。

```
select textvalid ("texttest.blurb", textptr(blurb)) from
texttest
```

使用法

- テキストおよびイメージ関数 textvalid は、特定のテキスト・ポインタが有効かどうかをチェックします。ポインタが有効な場合は 1 を返し、無効な場合は 0 を返します。
- text カラムや image カラムの識別子には、テーブル名を含めてください。
- テキストおよびイメージ関数の概要については、「text 関数および image 関数」(70 ページ) を参照してください。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが textvalid 関数を実行できます。

参照

データ型 text、image、unitext データ型

関数 textptr

to_unichar

説明　整数式の値が含まれた unichar 式を返します。

構文　to_unichar(*integer_expr*)

パラメータ

integer_expr
整数値型 (tinyint、smallint、または int) のカラム名、変数、定数式のいずれかです。

使用法

- 文字列関数 to_unichar は、Unicode 整数値を Unicode 文字値に変換します。
- unichar 式がサロゲート・ペアのいずれか 1 つのサロゲートだけを参照している場合、エラー・メッセージが表示されて処理がアボートされます。
- *integer_expr* が NULL の場合、to_unichar は NULL を返します。

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　すべてのユーザが to_unichar 関数を実行できます。

参照　文字列関数の概要については、「文字列関数」(68 ページ) を参照してください。

データ型　text、image、unitext データ型

関数　char

tran_dumpable_status

説明

dump transaction を使用できるかどうかを示す true/false の値を返します。

構文

tran_dumpable_status("*database_name*")

パラメータ

database_name
ターゲット・データベースの名前です。

例

pubs2 データベースをダンプできるかどうかをチェックします。

```
1> select tran_dumpable_status("pubs2")
2> go

 -----------
         106


(1 row affected)
```

この例では、pubs2 をダンプできません。リターン・コード 106 は、一致した条件 (2、8、32、64) すべての合計です。リターン・コードについては、「使用法」の項を参照してください。

使用法

tran_dumpable_status を使用して、データベースでダンプ・トランザクションが許可されるかどうかをコマンドを実行せずに確認できます。tran_dumpable_status は、ダンプ・トランザクションの実行時に Adaptive Server で行われるすべてのチェックを実行します。

tran_dumpable_status が 0 を返す場合は、データベースに dump transaction コマンドを実行できます。その他の値を返す場合は、実行できません。0 以外の値の意味は次のとおりです。

- 1 − 指定した名前のデータベースは存在しない。
- 2 − ログが別のデバイスに存在しない。
- 4 − ログの最初のページがデータ専用ディスク・フラグメントの範囲にある。
- 8 − データベースに trunc log on chkpt オプションが設定されている。
- 16 − ログを取らない書き込みがデータベースで実行された。
- 32 − トランケート・オンリーの dump tran が、ダンプ・デバイスへの連続したダンプに割り込みをかけた。
- 64 − データベースが新規に作成されたか、アップグレードされた。トランザクション・ログは、dump database が実行されるまでダンプされません。

- 128 － データベースの持続性はトランザクションのダンプを許可していない。
- 256 － データベースは読み込み専用である。dump transaction がトランザクションを開始しましたが、読み込み専用データベースでは許可されていません。
- 512 － データベースはスタンバイ・アクセスのためオンラインである。dump transaction がトランザクションを開始しましたが、ローカル・シーケンスを妨害するため、スタンバイ・アクセスのデータベースでは許可されていません。
- 1024 － データベースは、dump transaction をサポートしていないアーカイブ・データベースである。

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　この関数は、すべてのユーザが実行できます。

参照　**コマンド**　dump transaction

tsequal

説明

ローがブラウズ用に選択されたあとで変更された場合に、ローを更新できないようにするため timestamp 値を比較します。

構文

tsequal(*browsed_row_timestamp*, *stored_row_timestamp*)

パラメータ

browsed_row_timestamp
ブラウズされたローの timestamp カラムです。

stored_row_timestamp
保管されたローの timestamp カラムです。

例

publishers テーブルの現在のバージョンから timestamp カラムを検索し、保管された timestamp カラムの値と比較します。2 つの timestamp カラムの値が等しい場合、tsequal はローを更新します。等しくない場合、tsequal は次のエラー・メッセージを返します。

```
update publishers
set city = "Springfield"
where pub_id = "0736"
and tsequal(timestamp,0x0001000000002ea8)
```

使用法

- システム関数 tsequal は、timestamp カラムの値を比較して、ブラウズ用に選択されたあとで変更されたローの更新ができないようにします。システム関数の概要については、「システム関数」(69 ページ) を参照してください。
- tsequal 関数を使用すると、DB-Library の dbequal 関数を呼び出さずにブラウズ・モードを使用できます。ブラウズ・モードでは、データの表示中に更新を実行できます。ブラウズ・モードは Open Client やホスト・プログラミング言語を使うフロントエンド・アプリケーションで使います。ローにタイムスタンプが記録されているテーブルをブラウズできます。
- フロントエンド・アプリケーションでテーブルをブラウズするには、Adaptive Server に送信する select 文の最後に for browse キーワードを追加します。たとえば、次のようにします。

```
Start of select statement in an Open Client application
...
   for browse

Completion of the Open Client application routine
```

- tsequal 関数は、select 文の where 句では使用しないでください。where 句にこの関数を使用できるのは、insert 文と、where 句によりユニークな 1 ローのみが対象となる update 文です。

  timestamp カラムを検索句として使用する場合は、通常の varbinary カラムのように、timestamp1 = timestamp2 として比較してください。

ブラウズ用の新規テーブルにタイムスタンプを設定する

- ブラウズ用に新しいテーブルを作成するときは、timestamp というカラムをテーブル定義に指定してください。このカラムには自動的に timestamp データ型が割り当てられるので、データ型を指定する必要はありません。たとえば、次のようにします。

```
create table newtable(col1 int, timestamp, col3 char(7))
```

  ローを挿入または更新するたびに、Adaptive Server は timestamp カラムにユニークな varbinary 値を自動的に割り当て、タイムスタンプを設定します。

既存のテーブルにタイムスタンプを設定する

- ブラウズ用に既存のテーブルを準備するには、alter table を使用して timestamp カラムを追加します。たとえば、NULL の値を持つ timestamp カラムを既存のローに追加するには、次のクエリを使用します。

```
alter table oldtable add timestamp
```

  タイムスタンプを生成するには、新しいカラム値を指定しないで、既存の各ローを更新します。

```
update oldtable
set col1 = col1
```

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　すべてのユーザが tsequal 関数を実行できます。

参照　**データ型**　timestamp データ型

uhighsurr

説明

start の位置の Unicode 値がサロゲート・ペアの上位サロゲート (ペアの中で最初に指定されている) の場合は、1 を返します。それ以外の場合は 0 を返します。

構文

uhighsurr(*uchar_expr*, start)

パラメータ

uchar_expr

文字型のカラム名、変数、または (unichar または univarchar 型の) 定数式です。

start

検査する文字位置を指定します。

使用法

- 文字列関数 uhighsurr を使用すれば、サロゲート処理のための明示的なコードを作成できます。特に、uhighsurr が true である Unicode 文字で部分文字列が始まる場合、少なくとも 2 つの Unicode 値を持つ部分文字列を抽出します (*substr* はサロゲート・ペアのいずれか 1 つのサロゲートを抽出しません)。
- *uchar_expr* が NULL の場合、uhighsurr は NULL を返します。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが uhighsurr 関数を実行できます。

参照

文字列関数の概要については、「文字列関数」(68 ページ) を参照してください。

関数 ulowsurr

ulowsurr

説明

start の位置の Unicode 値が、サロゲート・ペアの下位サロゲート (ペアの中で 2 番目に指定されている) の場合は、1 を返します。それ以外の場合は 0 を返します。

構文

ulowsurr(*uchar_expr*, start)

パラメータ

uchar_expr

文字型のカラム名、変数、または (unichar または univarchar 型の) 定数式です。

start

検査する文字位置を指定します。

使用法

- 文字列関数 ulowsurr を使用すれば、substr、stuff、right によって実行される調整処理の明示的なコードを作成できます。特に、ulowsurr が true である Unicode 値で部分文字列が終わる場合、1 少ない (または 1 多い) 部分文字列が抽出されます。substr では、対になっていないサロゲート・ペアが含まれている文字列は抽出されません。
- *uchar_expr* が NULL の場合、ulowsurr は NULL を返します。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが ulowsurr 関数を実行できます。

参照

文字列関数の概要については、「文字列関数」(68 ページ) を参照してください。

関数　uhighsurr

upper

説明

指定した文字列に相当する大文字を返します。

構文

upper(*char_expr*)

パラメータ

char_expr

文字型のカラム名、変数、または (char、unichar、varchar、nchar、nvarchar、または univarchar 型の) 定数式です。

例

```
select upper("abcd")

----
ABCD
```

使用法

- 文字列関数 upper は、小文字を大文字に変換して文字値を返します。
- *char_expr* または *uchar_expr* が NULL の場合、upper は NULL を返します。
- 対応する大文字がない文字は、変換されません。
- 作成される unichar 式にサロゲート・ペアのいずれか 1 つのサロゲートだけが含まれる場合は、エラー・メッセージが表示され、処理がアボートされます。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが upper 関数を実行できます。

参照

文字列関数の概要については、「文字列関数」(68 ページ) を参照してください。

関数 lower

uscalar

説明　式の最初の Unicode 文字に対する Unicode スカラ値を返します。

構文　uscalar(*uchar_expr*)

パラメータ

uchar_expr
文字型のカラム名、変数、または (unichar または univarchar 型の) 定数式です。

使用法

- 文字列関数 uscalar は、式の最初の Unicode 文字の Unicode 値を返します。
- *uchar_expr* が NULL の場合は、NULL を返します。
- 対になっていない上位または下位のサロゲートが含まれている *uchar_expr* に対して uscalar が呼び出されると、エラーが発生して処理がアボートされます。

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　すべてのユーザが uscalar 関数を実行できます。

参照　文字列関数の概要については、「文字列関数」(68 ページ) を参照してください。

関数　ascii

used_pages

説明

テーブル、インデックス、または特定のパーティションで使用されているページ数をレポートします。data_pages とは異なり、used_pages は、内部構造に使用されるページを数に含めません。この関数は、バージョン 15.0 より前の Adaptive Server で使用された古い used_pgs 関数を置き換えるものです。

構文

used_pages(*dbid*, *object_id*[, *indid*[, *ptnid*]])

パラメータ

dbid

ターゲット・オブジェクトが存在するデータベースの ID です。

object_id

使用ページを表示するテーブルのオブジェクト ID です。インデックスが使用するページを表示するには、そのインデックスが設定されているテーブルのオブジェクト ID を指定してください。

indid

対象のインデックス ID です。

ptnid

対象のパーティション ID です。

例

例 1 指定されたデータベースにある、オブジェクト ID が 31000114 のオブジェクトで使用されるページ数を返します (インデックスも含む)。

```
select used_pages(5, 31000114)
```

例 2 クラスタード・インデックスがあるかどうかに関係なく、データ・レイヤに含まれるオブジェクトで使用されるページ数を返します。

```
select used_pages(5, 31000114, 0)
```

例 3 インデックス ID が 2 のインデックス・レイヤにあるオブジェクトが使用しているページの数を返します。これには、データ・レイヤが使用するページは含まれていません (式の「使用法」の最初の項を参照してください)。

```
select used_pages(5, 31000114, 2)
```

例 4 指定されたパーティション (この例では 2323242432) のデータ・レイヤにあるオブジェクトで使用されるページ数を返します。

```
select used_pages(5, 31000114, 0, 2323242432)
```

使用法

- クラスタ・インデックス付きの全ページ・ロック・テーブルでは、最後のパラメータの値によって、使用されたどのページが返されるかが決まります。
  - used_pages(dbid, objid, 0) − インデックス ID として明示的に 0 を渡し、データ・レイヤで使用されているページのみを返します。
  - used_pages(dbid, objid, 1) − インデックス・レイヤで使用されているページと、データ・レイヤで使用されているページを返します。

  クラスタード・インデックスを持つ全ページロック・テーブルの場合、インデックス・レイヤで使用されているページを取得するには、used_pages(dbid, objid, 1) から used_pages(dbid, objid, 0) を減算します。
- クラスタード・インデックスを持つ全ページ・ロック・テーブルでは、`indid = 0` の値に対して used_pages はデータ・レイヤの使用ページ数のみを渡されます。`indid=1` を渡した場合は、以前のバージョンと同じく、データ・レイヤとクラスタード・インデックス・レイヤの使用ページ数が返されます。
- used_pages は、以前の used_pgs(objid, doampg, ioampg) 関数に似ています。
- すべてのエラーで常に戻り値 0 が返されます。

規格

ANSI SQL − 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが used_pgs 関数を実行できます。

参照

関数　data_pages, object_id

user

説明

現在のユーザの名前を返します。

構文

user

パラメータ

なし。

例

```
select user

------
dbo
```

使用法

- システム関数 user は、ユーザの名前を返します。
- sa_role が有効な場合は、どのデータベースを使っていても、自動的にそのデータベースの所有者になります。データベース内では、データベース所有者のユーザ名は常に “dbo” です。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが user 関数を実行できます。

参照

システム関数の概要については、「システム関数」(69 ページ) を参照してください。

関数 user_name

user_id

説明

指定したユーザの ID、またはデータベース内の現在のユーザの ID を返します。

構文

user_id([*user_name*])

パラメータ

user_name
ユーザの名前です。

例

例 1

```
select user_id()

------
     1
```

例 2

```
select user_id("margaret")

------
     4
```

使用法

- システム関数 user_id は、ユーザの ID 番号を返します。システム関数の概要については、「システム関数」(69 ページ) を参照してください。
- user_id 関数は、現在のデータベース内の sysusers から番号を返します。*user_name* を指定しないと、user_id 関数は現在のユーザの ID を返します。Adaptive Server のすべてのデータベースで同じ番号であるサーバ・ユーザ ID を検索するには、suser_id 関数を使います。
- データベース内では “guest” ユーザ ID は常に 2 です。
- データベース内では、データベース所有者の user_id は常に 1 です。sa_role が有効な場合は、どのデータベースを使っていても、自動的にそのデータベースの所有者になります。実際のユーザ ID へ戻るには、set sa_role off を実行してから user_id を実行します。データベースの有効なユーザでない場合には、set sa_role off を実行すると、Adaptive Server からエラーが返されます。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

自分以外の user_name でこの関数を使用できるのは、システム管理者とシステム・セキュリティ担当者だけです。

参照

コマンド　setuser

関数　suser_id, user_name

user_name

説明

指定したユーザまたは現在のユーザのデータベース内の名前を返します。

構文

user_name([*user_id*])

パラメータ

user_id

ユーザの ID です。

例

例 1

```
select user_name()

------------------------------
dbo
```

例 2

```
select user_name(4)

------------------------------
margaret
```

使用法

- システム関数 user_name は、現在のデータベース内のユーザ ID をもとに、ユーザの名前を返します。
- *user_id* を指定しない場合、user_name 関数は現在のユーザの名前を返します。
- sa_role が有効な場合は、どのデータベースを使っていても、自動的にそのデータベースの所有者になります。データベース内のデータベース所有者の user_name は常に “dbo” です。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

自分以外の user_id でこの関数を使用できるのは、システム管理者とシステム・セキュリティ担当者だけです。

参照

システム関数の概要については、「システム関数」(69 ページ) を参照してください。

関数 suser_name, user_id

valid_name

説明

指定された文字列が有効な識別子でない場合は、0 を返します。有効な識別子の場合は、0 以外の値を返します。文字列の最大長は 255 バイトです。

構文

valid_name(*character_expression*[, *maximum_length*])

パラメータ

character_expression

文字型のカラム名、変数、または (char、varchar、nchar、または nvarchar 型の) 定数式です。定数式は、引用符で囲んでください。

maximum_length

0 より大きく 255 以下の整数です。デフォルト値は 30 です。識別子の長さが 2 番目の引数より大きい場合に、valid_name は 0 を返し、識別子の長さが有効な場合に 0 より大きな値を返します。

例

識別子が有効であることを検証するプロシージャを作成します。

```
create procedure chkname
@name varchar(30)
as
     if valid_name(@name) = 0
     print "name not valid"
```

使用法

- システム関数 valid_name は、*character_expression* が有効な識別子でない (不正な文字列、30 バイトを超える文字列や予約語) 場合は 0 を返し、有効な識別子の場合は 0 以外の数を返します。
- Adaptive Server の識別子の最大長は、使用されている文字がシングルバイト文字でもダブルバイト文字でも 16384 バイトです。識別子の先頭文字は、現在の文字セットに定義されている文字か、アンダースコア (_) 文字を使用してください。この規則の例外として、シャープ記号 (#) で始まるテンポラリ・テーブル名と at 記号 (@) で始まるローカル変数名があります。識別子がシャープ記号 (#) と at 記号 (@) で始まる場合に valid_name 関数は 0 を返します。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが valid_name 関数を実行できます。

参照

システム関数の概要については、「システム関数」(69 ページ) を参照してください。

システム・プロシージャ　sp_checkreswords

valid_user

説明

指定した ID が、Adaptive Server 上の少なくとも 1 つのデータベースで有効なユーザまたはエイリアスの場合は 1 を返し、そうでない場合は 0 を返します。

構文

valid_user(*server_user_id*)

パラメータ

server_user_id

サーバ・ユーザ ID です。サーバ・ユーザ ID は syslogins の suid カラムに格納されています。

例

```
select valid_user(4)

---------------
              1
```

使用法

システム関数 valid_user は、指定した ID が、この Adaptive Server 上の少なくとも 1 つのデータベース内で有効なユーザかエイリアスの場合に 1 を返します。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

自分以外の server_user_id でこの関数を使用できるのは、システム管理者とシステム・セキュリティ担当者だけです。

参照

システム関数の概要については、「システム関数」(69 ページ) を参照してください。

システム・プロシージャ sp_addlogin, sp_adduser

var

説明

1 つの数値式で構成される標本の統計分散を double 型として計算し、数値セットの分散を返します。

注意　var と variance は、var_samp のエイリアスです。詳細については、var_samp (311 ページ) を参照してください。

var_pop

説明

1 つの数値式で構成される母統計分散を double 型として計算します。varp は、var_pop のエイリアスで、同じ構文を使用します。

構文

var_pop ([all | distinct] *expression*)

パラメータ

all
すべての値に var_pop を適用します。デフォルト設定は all です。

distinct
var_pop を適用する前に重複する値を削除します。

expression
その母集団ベースの分散がローのセットに対して計算される式です (通常はカラム名)。

例

pubs2 データベースにおける書籍の各種類の前払い金の平均と分散を示します。

```
select type, avg(advance) as "avg", var_pop(advance)
  as "variance" from titles group by type order by type
```

使用法

グループの各ロー (distinct が指定された場合は、重複が削除されたあとで残る各ロー) に対して評価される指定の値式の母分散を計算します。これは、値式と値式の平均の差の 2 乗和を、グループまたはパーティション内のローの数で割った値として定義されます。

図 *2-5:* 母集団に関係する統計集合関数の計算式

平均 μ (var_pop) を持つサイズ *n* の母分散を定義する計算式は、次のとおりです。母標準偏差値 (stddev_pop) は、この数値の正の平方根です。

$$\sigma^2 = \frac{\sum (x_i - \mu)^2}{n}$$

σ^2 = 分散
n = 母集団のサイズ
$\mu = x_i$ 値の平均

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが var_pop を実行できます。

参照

集合関数の概要については、『ASE リファレンス・マニュアル：ビルディング・ブロック』の「集合関数」を参照してください。

関数 stddev_pop, stddev_samp, var_samp

var_samp

説明

1 つの数値式で構成される標本統計分散を double 型として計算し、数値セットの分散を返します。var と variance は、var_samp のエイリアスで、同じ構文を使用します。

構文

var_samp ([all | distinct] *expression*)

パラメータ

all

すべての値に var_samp を適用します。デフォルト設定は all です。

distinct

var_samp を適用する前に重複する値を削除します。

expression

任意の numeric データ型 (float、real、または double) の式です。

例

pubs2 データベースにおける書籍の各種類の前払い金の平均と分散を示します。

```
select type, avg(advance) as "avg", var_samp(advance)
  as "variance" from titles where
  total_sales > 2000 group by type order by type
```

使用法

var_samp は、結果を倍精度浮動小数点数のデータ型で返します。空のセットが指定された場合の結果は NULL になります。

図 *2-6:* 標本に関係する統計集合関数の計算式

平均が $\bar{x}$ (var_samp) のサイズ n の標本からの母分散の不偏推定値を定義する計算式は、次のとおりです。標本標準偏差 (stddev_samp) は、この数値の正の平方根です。

$$s^2 = \frac{\sum (x_i - \bar{x})^2}{n - 1}$$

s^2 = 分散
n = 標本サイズ
$\bar{x} = x_i$ 値の平均

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが var_samp を実行できます。

参照

集合関数の概要については、『ASE リファレンス・マニュアル：ビルディング・ブロック』の「集合関数」を参照してください。

関数　stddev_pop, stddev_samp, var_pop

variance

説明

1 つの数値式で構成される標本の統計分散を double 型として計算し、数値セットの分散を返します。

注意 var と variance は、var_samp のエイリアスです。詳細については、var_samp (311 ページ) を参照してください。

varp

説明

1 つの数値式で構成される母統計分散を double 型として計算します。

注意　varp は、var_pop のエイリアスです。詳細については、var_pop (310 ページ) を参照してください。

workload_metric

説明

クラスタ環境のみ － 指定したインスタンスの現在の負荷測定基準値のクエリを実行するか、指定したインスタンスの測定基準値を更新します。

構文

workload_metric(*instance_id* | *instance_name* [, *new_value*])

パラメータ

instance_id
インスタンスの ID です。

instance_name
インスタンスの名前です。

new_value
新しい測定基準を表す浮動小数値です。

例

例 1 現在のインスタンスのユーザ測定基準を見ます。

```
select workload_metric()
```

例 2 インスタンス “ase2” のユーザ測定基準を見ます。

```
select workload_metric("ase2")
```

例 3 “ase2” から 27.54 までのユーザ測定基準の値を設定します。

```
select workload_metric("ase3", 27.54)
```

使用法

- NULL 値は現在のインスタンスを示します。
- *new_value* に値を指定した場合は、その値が現在のユーザ測定基準になります。*new_value* に値を指定しなかった場合は、現在の負荷測定基準が返されます。
- *new_value* の値は 0 以上でなければなりません。
- *new_value* に値を指定すると、workload_metric は処理に成功した場合にその値を返します。それ以外の場合、workload_metric は -1 を返します。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

workload_metric を実行するには、sa_role または ha_role が必要です。

xa_bqual

説明

ASCII XA トランザクション ID の bqual コンポーネントのバイナリ・バージョンを返します。

構文

xa_bqual(*xid*, 0)

パラメータ

xid

systransactions の xactname カラム、または sp_transactions で取得される Adaptive Server トランザクションの ID です。

0

今後のために予約済みです。

例

例 1 Adaptive Server トランザクション ID “0000000A_IphIT596iC7bF2#AUfkzaM_8DY6OE0” の分岐修飾子をバイナリ変換した “0x227f06ca80” を返します。Adaptive Server トランザクション ID は、最初は sp_transactions を使用して次のように取得されます。

```
1>sp_transactions

xactkey                          type     coordinator starttime           state
connection dbid  spid  loid  failover    srvname  namelen  xactname
------------------------------ -------- ----------- ------------------- ---------- -
--------- ----- ----- ----- ----------- -------  -------  --------------------------
-------------
0x5316000006000000017e4885b0700 External XA          Dec 9 2005  5:15PM In
Command  Attached     7   20    877  Resident Tx  NULL         39
0000000A_IphIT596iC7bF2#AUfkzaM_8DY6OE0

1> select xa_bqual("0000000A_IphIT596iC7bF2#AUfkzaM_8DY6OE0", 0)
2> go

...

------------------------------------------------------------------------


0x227f06ca80
```

例 2 xa_bqual は、xa_gtrid と一緒に使用される場合が多くあります。この例は、systransactions のすべてのローからのグローバル・トランザクション ID と分岐修飾子を返します。coordinator カラムの値は “3” です。

```
1> select gtrid=xa_gtrid(xactname,0),
    bqual=xa_bqual(xactname,0)
    from systransactions where coordinator = 3
2> go

        gtrid

        bqual

------------------------------------------------------------------------

------------------------------------------------------------------------

         0xb1946cdc52464a61cba42fe4e0f5232b

         0x227f06ca80
```

使用法

Adaptive Server で外部トランザクションがブロックされており、ブロックされているトランザクションを sp_lock および sp_transactions を使用して識別する場合、XA トランザクション・マネージャを使用してグローバル・トランザクションを終了させることができます。ただし、sp_transactions を実行する場合、返される *xactname* の値は ASCII 文字列フォーマットです。一方、XA Server は、復号化されていないバイナリ値を使用します。xa_bqual を使用すると、XA トランザクション・マネージャが解読できる形式のトランザクション名の bqual 部分を特定できます。

xa_bqual は、以下を返します。

- この文字列の変換バージョンで、2 番目の “_” (アンダースコア) より後、3 番目の “_” または end-of-string 値のうち先に出現した方の前にあります。
- トランザクション ID が復号化できないか予想外の形式の場合は NULL。

注意 xa_bqual は、xid に検証チェックを実行せず、変換後の文字列を返すだけです。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザは、xa_bqual を使用できます。

参照

関数 xa_gtrid

ストアド・プロシージャ sp_lock、sp_transactions

xa_gtrid

説明

ASCII XA トランザクション ID の gtrid コンポーネントのバイナリ・バージョンを返します。

構文

xa_gtrid(*xactname*, *int*)

パラメータ

xid

systransactions の xactname カラム、または sp_transactions で取得される Adaptive Server トランザクションの ID です。

0

今後のために予約済みです。

例

例 1 一般的な状況では、Adaptive Server トランザクション ID “0000000A_IphIT596iC7bF2#AUfkzaM_8DY6OE0” に対して “0x227f06ca80” (分岐修飾子のバイナリ変換)、“0xb1946cdc52464a61cba42fe4e0f5232b” (グローバル・トランザクション ID) が返されます。

```
1> select xa_gtrid("0000000A_IphIT596iC7bF2#AUfkzaM_8DY6OE0", 0)
2> go

       ...

------------------------------------------------------------------------------
          0xb1946cdc52464a61cba42fe4e0f5232b

(1 row affected)
```

例 2 xa_bqual は、xa_gtrid と一緒に使用される場合が多くあります。この例は、systransactions のすべてのローからのグローバル・トランザクション ID と分岐修飾子を返します。coordinator カラムの値は “3” です。

```
1> select gtrid=xa_gtrid(xactname,0),
    bqual=xa_bqual(xactname,0)
    from systransactions where coordinator = 3
2> go

        gtrid

        bqual

------------------------------------------------------------------------------

------------------------------------------------------------------------------

         0xb1946cdc52464a61cba42fe4e0f5232b

         0x227f06ca80
```

使用法

Adaptive Server で外部トランザクションがブロックされており、ブロックされているトランザクションを sp_lock および sp_transactions を使用して識別する場合、XA トランザクション・マネージャを使用してグローバル・トランザクションを終了させることができます。ただし、sp_transactions を実行する場合、返される *xactname* の値は ASCII 文字列フォーマットです。一方、XA Server は、復号化されていないバイナリ値を使用します。xa_gtrid を使用すると、XA トランザクション・マネージャが解読できる形式のトランザクション名の gtrid 部分を特定できます。

xa_gtrid は、以下を返します。

- 最初の "_" (アンダースコア) より後にあり、2 番目の "_" または end-of-string 値のうち先に出現した方の前にある、この文字列の変換バージョン。
- トランザクション ID が復号化できないか予想外の形式の場合は NULL。

注意 xid は、xa_gtrid に対して検証チェックを実行せず、単に変換された文字列を返します。

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが xa_gtrid を使用できます。

参照

関数 xa_bqual

ストアド・プロシージャ sp_lock、sp_transactions

xmltable

説明

XML ドキュメントからデータを抽出し、SQL テーブルとして返します。

構文

xmltable_expression ::= xmltable
(*row_pattern* passing *xml_argument*
columns *column_definitions*
options_parameter)
row_pattern ::= *character_string_literal*
xml_argument ::= xml_expression | column_reference | variable_reference
column_definitions ::=
column_definition [{ , *column_definition* }]
column_definition ::=
ordinality_column | *regular_column*

ordinality_column ::= *column_name* datatype for ordinality
regular_column ::=
column_name datatype [default *literal*] [null | not null]
[path *column_pattern*]
column_pattern ::= *character_string_literal*
options_parameter ::=[,] option *option_string*
options_string ::= *basic_string_expression*

抽出テーブルの構文　SQL の from 句から SQL テーブルを返します。

from_clause ::= from *table_reference* [, *table_reference*]...
table_reference ::= *table_view_name* | *ANSI_join* |*derived_table*
table_view_name::= See the select command in Reference Manual
Volume 2, "Commands."
ANSI_join::= See the select command in Reference Manual
Volume 2, "Commands."
derived_table ::=
(subquery) as *table_name* [(*column_name* [, *column_name*]...)|
xmltable_expression as *table_name*

パラメータ

xml_argument
XML ドキュメントを参照する式、カラム参照、または変数です。

for
予約された XML キーワードです。

ordinality
予約されていない XML キーワードです。

passing
予約されていない XML キーワードです。

row_pattern
結果が指定したドキュメントの要素のシーケンスである XPath クエリ式です。xmltable 呼び出しは、シーケンスの要素ごとに 1 つのローのテーブルを返します。

columns
予約されていない XML キーワードです。

column_name
ユーザ指定のカラムの名前です。

column_pattern
row_pattern によって返されたシーケンスの要素に適用して結果テーブルのカラムのデータを抽出する XPath クエリ式です。*column_pattern* を省略すると、*column_pattern* は *column_name* をデフォルトで使用します。

ordinality_column
入力 XML ドキュメントの要素の順序付けを示す integer、smallint、tinyint、decimal、numeric のデータ型のカラムです。

regular_column
ordinality カラム以外のすべてのカラムです。

derived_table
SQL クエリの from 句で指定したカッコ内のサブクエリです。

path
予約された XML キーワードです。

option
XML Services で定義された *option_string* で、予約された XML キーワードです。

例

例 1 ドキュメントをリテラル文字列に指定する単純な xmltable の呼び出しを示します。

```
select * from xmltable ('/doc/item'
  passing '<doc><item><id>1</id><name>Box</name></item>'
    +'<item><id>2</id><name>Jar</name></item></doc>'
  columns id int path 'id', name varchar(20) path 'name') as items_table
id              name
-----           -----
1               Box
2               Jar

(2 rows affected)
```

例 2 Transact-SQL 変数にドキュメントを格納し、その変数を xmltable 呼び出しで参照します。

```
declare @doc varchar(16384)
set @doc = '<doc><item><id>1</id><name>Box</name></item>'
    +'<item><id>2</id><name>Jar</name></item></doc>'

select * from xmltable('/doc/item' passing @doc
  columns id int path 'id', name varchar(20) path 'name') as items_table

id            name
----          -----
1             Box
2             Jar

(2 rows affected)
```

例 3 ドキュメントをテーブルに格納し、サブクエリで参照します。

```
select 100 as doc_id,'<doc><item><id>1</id><name>Box</name></item><item><id>2</id>
  <name>Jar</name></item></doc>' as doc
into #sample_docs
select * from xmltable('/doc/item'
  passing(select doc from #sample_docs where doc_id=100)
  columns id int path 'id', name varchar(20) path 'name') as items_table

id            name
-------       -----
1             Box
2             Jar

(2 rows affected)
```

例 4 ロー・パターンで空のシーケンスが返されると、結果は次のように空のテーブルになります。

```
select * from xmltable ('/doc/item_entry'
  passing '<doc><item><id>1</id><name>Box</name></item>'
    +'<item><id>2</id><name>Jar</name></item></doc>'
  columns id int path 'id',
    name varchar(20) path 'name')as items_table

id               name
-------         ------

(0 rows affected)
```

例 5 columns キーワードに続く引数は、カラム定義のリストで構成されています。各カラム定義は、create table のように、カラム名とデータ型、およびカラム・パターンと呼ばれるパスを指定します。

カラムのデータが XML 属性に含まれている場合は、“@” を使用してカラム・パターンを指定し、属性を参照します。たとえば、次のようにします。

```
select * from xmltable ('/doc/item'
  passing '<doc><item id="1"><name>Box</name></item>'
    +'<item id="2"><name>Jar</name></item></doc>'
  columns id int path '@id', name varchar(20)) as items_table

id              name
--------------------
1               Box
2               Jar

(2 rows affected)
```

例 6 *column-pattern* は一般に、name のように、指定されている *column_name* と同じです。この場合、column_pattern を省略すると、デフォルトで *column_name* が使用されます。

```
select * from xmltable ('/doc/item'
  passing '<doc><item><id>1</id><name>Box</name></item>'
    +'<item><id>2</id><name>Jar</name></item></doc>'
  columns id int, name varchar(20)) as items_table

id              name
--------------------
1               Box
2               Jar

(2 rows affected)
```

例 7 カラム・パターンでカラム名をデフォルトに使用する場合は、値が XML 属性であるカラムに引用符付き識別子を使用します。次のように結果でカラム名を参照する場合は、その識別子を引用符で囲む必要があります。

```
set quoted_identifier on
select "@id", name from xmltable ('/doc/item'
    passing '<doc><item id="1"><name>Box</name></item>'
        +'<item id="2"><name>Jar</name></item></doc>'
    columns "@id" int, name varchar(20)) as items_table

@id              name
--------------------
1               Box
2               Jar

(2 rows affected)
```

例 8 引用符付き識別子を使用することで、さらに複雑な XPath 式を使用してカラム名をデフォルトのカラム・パターンに指定できます。次に例を示します。

```
set quoted_identifier on
select "@id", "name/short", "name/full" from xmltable ('/doc/item'
  passing '<doc><item id="1"><name><short>Box</short>
      <full>Box, packing, moisture resistant, plain</full>
      </name></item>'
  +'<item id="2"><name><short>Jar</short>
      <full>Jar, lidded, heavy duty</full>
      </name></item></doc>'
  columns "@id" int, "name/short" varchar(20), "name/full" varchar(50))
  as items_table

@id              name/short    name/full
--------------------
1               Box           Box, packing, moisture resistant, plain
2               Jar           Jar, lidded, heavy duty

(2 rows affected)
```

例 9 関数 text はカラム・パターンでは暗黙的です。次の例は、id カラムにも name カラムにもカラム・パターンに text を指定しません。

```
select * from xmltable ('/doc/item'
passing '<doc><item><id>1</id><name>Box</name></item>'
+'<item><id>2</id><name>Jar</name></item></doc>'
columns id int path 'id', name varchar(20) path 'name') as items_table
id    name
---   --------
1     Box
2     Jar
(2 rows affected)
```

例 10 暗黙の SQL converst 文をカラム・パターンから抽出されたデータに適用すると、データ型変換のカラム値が導出されます。

```
select * from xmltable ('/emps/emp'
  passing '<emps>
<emp><id>1</id><salary>123.45</salary><hired>1/2/2003</hired></emp>'
+'<emp><id>2</id><salary>234.56 </salary><hired>2/3/2004</hired></emp>'
    +'</emps>'
  columns id int path 'id', salary dec(5,2), hired date)
as items_table

id          salary        hired
-------     ---------     --------
1           123.45        Jan 2, 2003
2           234.56        Feb 3, 2004
(2 rows affected)
```

例 11 xmltable で *ordinality_column* を使用して、入力 XML ドキュメントの要素の順序付けを記録できます。

```
declare @doc varchar(16384)
set @doc = '<doc><item><id>25</id><name>Box</name></item>'
  +'<item><id>15</id><name>Jar</name></item></doc>'
select * from xmltable('/doc/item' passing @doc
  columns item_order int for ordinality,
    id int path 'id',
    name varchar(20) path 'name')as items_table
order by item_order

item_order          id            name
----------          ---           ----
1                   25            Box
2                   15            Jar
(2 rows affected)
```

for ordinality 句と item_order カラムを指定しない場合、id 25 のローが id 15 に先行することを示す id も name カラムもありません。for ordinality 句は、出力 SQL ローを入力 XML ドキュメントの要素の順序付けと同じ順序にします。

ordinality カラムのデータ型には任意の固定数値データ型を使用できます。int、tinyint、bigint、numeric、または decimal。numeric と decimal の位取りは 0 でなければなりません。ordinality 列が real または float であってはなりません。

例 12 <name> 要素を 2 番目の <item> から省きます。name カラムでは、名前をデフォルトで NULL にすることができます。

```
select * from xmltable ('/doc/item'
  passing '<doc><item><id>1</id><name>Box</name></item>'
  +'<item><id>2</id></item></doc>'
columns id int path 'id', name varchar(20) path 'name')
as items_table
id            name
------------------
1             Box
2             NULL

(2 rows affected)
```

例 13 <name> 要素を 2 番目の <item> から省き、name カラムに not null を指定します。

```
select * from xmltable ('/doc/item'
  passing '<doc><item><id>1</id><name>Box</name></item>'
  +'<item><id>2</id></item></doc>'
columns id int path 'id', name varchar(20) not null path 'name')
as items_table

id              name
-----           --------
1              Box
Msg 14847, Level 16, State 1:
Line 1:
XMLTABLE column 0, does not allow null values.
```

例 14 デフォルトの句を name カラムに追加し、<name> 要素を 2 番目の <item> から省きます。

```
select * from xmltable ('/doc/item'
  passing '<doc><item><id>1</id><name>Box</name></item>'
  +'<item><id>2</id></item></doc>'
columns id int path 'id', name varchar(20) default '***' path 'name')
as items_table
id   name
---  ------
1    Box
2    '***'
(2 rows affected)
```

例 15 抽出テーブル式で xmltable 呼び出しに使用できる SQL コマンドを示します。次の例は、xmltable を単純な select 文で使用します。

```
select * from xmltable ('/doc/item'
  passing '<doc><item><id>1</id><name>Box</name></item>'
  +'<item><id>2</id><name>Jar</name></item></doc>'
columns id int path 'id',
  name varchar(20) path 'name')as items_table

id            name
--            ----
1             Box
2             Jar
(2 rows affected)
```

例 16 ビュー定義で xmltable を使用します。ドキュメントをテーブルに保管し、xmltable を使用してテーブルからデータを抽出することによって、保管されたドキュメントを create view 文で参照します。

```
select 100 as doc_id,
'<doc><item><id>1</id><name>Box</name></item>'
  +'<item><id>2</id><name>Jar</name></item></doc>' as doc
into sample_docs
create view items_table as
  select * from xmltable('/doc/item'
    passing (select doc from sample_docs where doc_id=100)
    columns id int path 'id',
    name varchar(20) path 'name')as xml_extract
select * from items_table

id            name
------------------
1             Box
2             Jar
(2 rows affected)
```

例 17 カーソルで xmltable を使用します。

```
declare C cursor for
select * from xmltable('/doc/item'
  passing (select doc from sample_docs where id=100)
  columns id int path 'id',
  name varchar(20) path 'name')as items_table
  go
declare @idvar int
declare @namevar varchar(20)
open C
while @@sqlstatus=0
begin
fetch C into @idvar, @namevar
print 'ID "%1!"NAME"%2!"', @idvar, @namevar
end
-------------------------
```

```
ID "1" NAME "Box"
ID "2" NAME "Jar"

(2 rows affected)
```

他のテーブルから update、insert、delete を実行するなど、生成した各ローで複数の動作が必要になるアプリケーションでは、カーソル・ループを使用して xmltable の結果を処理できます。または xmltable 結果をテンポラリ・テーブルに保存して、カーソル・ループでそのテーブルを処理することもできます。

例 18 次の例は、xmltable を select into で使用します。

```
select * into #extracted_table
from xmltable('/doc/item'
  passing (select doc from sample_docs where doc_id=100
  columns id int path 'id',
  name varchar(20) path 'name')as items_table

select * from #extracted_table

id            name
----          ------------
1             Box
2             Jar
```

例 19 xmltable を insert コマンドで使用します。

```
create table #extracted_data (idcol int, namecol varchar(20))
insert into #extracted_data
select * from xmltable('/doc/item'
  passing (select doc from sample_docs where doc_id=100)
  columns id int path 'id', name varchar(20) path 'name') as items_table
select * from #extracted_data

idcol          namecol
-----          ---------
1              Box
2              Jar
(2 rows affected)
```

例 20 xmltable をサブクエリで使用します。xmltable は SQL テーブルを返すため、サブクエリで集合か選択のいずれかを実行して、単一のローとカラムをサブクエリ結果に返す必要があります。

```
declare @idvar int
set @idvar = 2
select @idvar,
(select name from xmltable ('/doc/item'
  passing (select doc from sample_docs where doc_id=100)
  columns id int path 'id',name varchar(20) path 'name') as item_table
where items_table.id=@idvar)
----------------------------
2                     Jar
(1 rows affected)
```

例 21　from 句の複数テーブルのジョイン、または外部ジョインを使用して、xmltable の結果を他のテーブルにジョインします。

```
create table prices (id int, price decimal (5,2))
insert into prices values (1,123.45)
insert into prices values (2,234.56)
select prices.id,extracted_table.name, prices.price
from prices,(select * from xmltable('/doc/item'
  passing (select doc from sample_docs where doc_id=100)
  columns id int path 'id', name varchar(20) path 'name') as a) as
extracted_table
where prices.id=extracted_table.id
id name  price
-- ----  -----
1 Box  123.45
2 Jar  234.56
(2 rows affected)
```

例 22　xmltable を使用して、xmltable として from 句の前のテーブルに存在するカラムをラテラル参照します。

```
create table deptab (col1 int, col2 image)
insert deptab values (1, '<dept>
                          <dept-id>1</dept-id>
                          <dept-name>Finance</dept-name>
                          <employees>
                            <emp><name>John</name><id>e11</id></emp>
                            <emp><name>Bela</name><id>e12</id></emp>
                            <emp><name>James</name><id>e13</id></emp>
                          </employees>
                          </dept>')

insert deptab values (2, '<dept>
                          <dept-id>2</dept-id>
                          <dept-name>Engineering</dept-name>
                          <employees>
                            <emp><name>Tom</name><id>e21</id></emp>
                            <emp><name>Jeff</name><id>e22</id></emp>
                            <emp><name>Mary</name><id>e23</id></emp>
                          </employees>
                          </dept>')

select id, empname from deptab, xmltable ('/dept/employees/emp' passing
deptab.co12 columns empname varchar (8) path 'name', id varchar (8)
path 'id') as sample_tab

id                    empname
---------             -----------
e11                   John
e12                   Bela
e13                   James
e21                   Tom
```

```
e22                  Jeff
e23                  Mary
(6 rows affected)
```

使用法

- xmltable は、組み込みのテーブル値関数です。
- この抽出テーブルの構文では、参照しない場合もテーブル名を指定する必要があります。したがって、xmltable の各式でもテーブル名を指定する必要があります。
- passing に続く引数は入力 XML ドキュメントです。
- xmltable 式の結果タイプは SQL テーブルで、そのカラム名とデータ型は *column_definitions* によって指定されます。
- ドキュメントを処理するには、xmltable を XML ドキュメントのテーブルの各ローに適用できます。
- これらのキーワードは xmltable に関連付けられています。
  - 予約されている － for、option、xmltable、path
  - 予約されていない － columns、ordinality、passing
- xmltable 呼び出しの引数の式は、xmltable 呼び出しを含む from 句の前のテーブルのカラム名を参照できます。xmltable 呼び出しに先行するテーブルだけを参照できます。このように、同じ from 句の前のテーブルのカラムを参照することをラテラル参照と言います。たとえば、次のようにします。

```
select * from T1, xmltable(...passing T1.C1...)
       as XT2, xmltable(...passing XT2.C2...)as XT3
```

  最初の xmltable 呼び出しでの T1.C1 の参照は、テーブル T1 のカラム C1 のラテラル参照です。2 番目の xmltable 呼び出しの XT2.C2 の参照は、最初の xmltable 呼び出しによって生成されたテーブルのカラム C2 のラテラル参照です。

- xmltable を update または delete 文の from 句で使用することはできません。たとえば、次の文は失敗します。

```
update T set T.C=...
from xmltable(...) as T
where...
```

- *regular_columns* のデータ型には任意の SQL データ型を使用できます。
- フォーマットが SQL convert 関数に適していない XML データを処理するには、データを文字列カラムに抽出します (varchar、text、image、java.lang.String)。
- カラム用に抽出された XML データは、カラムのデータ型に変換可能である必要があります。変換できない場合、例外が発生します。
- カラム・パターンによって空の結果が返された場合の対応は、`default` 句と `{null | not null}` 句によって異なります。

- *regular_column* で default に続く literal は、カラムのデータ型に割り当て可能である必要があります。
- *ordinality_column* は 1 つしか使用できません。この変数に指定するデータ型は、integer、smallint、tinyint、decimal、numeric でなければなりません。decimal と numeric には位取り 0 を使用する必要があります。
- *ordinality_column* がある場合、null 入力はできません。

注意　このデフォルトは create table のデフォルト値とは異なります。

- 他のカラムの null 入力可能プロパティは、`{null | not null}` 句によって指定されます。デフォルトは null です。
- set quoted_identifier の現在の設定は、xmltable 式の句に適用されます。たとえば、次のようにします。
  - set quoted_identifier が on の場合、カラム名に引用符付き識別子を使用できます。その場合、*row_pattern*、*column_pattern*、デフォルトのリテラルにある各文字列リテラルを一重引用符で囲む必要があります。
  - set quoted_identifierset quoted_identifier が off の場合、カラム名に引用符付き識別子を使用することはできません。*row_pattern*、*column_pattern*、デフォルトのリテラルにある各文字列リテラルは、一重引用符か二重引用符のいずれかで囲むことができます。
- option_string の一般的なフォーマットについては、XML Services, Adaptive Server 15.0 の「option_strings：一般的なフォーマット」を参照してください。

xmltable のロー・パターンとカラム・パターン

- xmltable のロー・パターンとカラム・パターンでは、単純なパスだけを使用できます。XPath の単純なパスは、'/' と要素名および属性名を使用した前方検索だけで構成されます。
- *row_pattern* が *xml_argument* によって指定されたドキュメントのルート・レベルから開始されないと、例外が発生します。ロー・パターンは XML ドキュメントのルートから開始する必要があります。
- ロー・パターン式に XPath 関数を含めることはできません。
- カラム・パターンは相対パスである必要があります。
- *row_pattern* で XML 関数を指定すると、例外が発生します。ロー・パターンで XML 関数を指定することはできません。
- *column_definition* でパスを指定しないと、デフォルトの *column_pattern* がカラム定義の *column_name* になります。このデフォルトは、サーバのパスワードの大文字と小文字の区別に従います。たとえば、次の文があるとします。

```
select * from xmltable(...columns name
   varchar(30),...)
```

サーバが大文字と小文字を区別しない場合、これは次の文と等しくなります。

```
select * from xmltable(...columns name varchar(30) path
'name',...)
```

サーバが大文字と小文字を区別する場合、最初の文は次の文と等しくなります。

```
select * from xmltable
   (...columns name varchar(30)path 'NAME',...)
```

結果テーブルのローの生成

- xmltable 式の結果値は T-SQL テーブル RT として次のように定義されます。
- RT は、*row_pattern* を *xml_argument* に適用した結果として得られる XML シーケンスの各要素にローがあります。
- RT のローには各 *column_definition* のカラムがあり、その *column_name* とデータ型は *column_definition* によって指定されます。
- *column_definition* が *ordinality_column* である場合、N 番目のローの値は整数 N になります。
- *column_definition* が *regular_column* である場合、N 番目のローの値は次のようになります。
  - この XPath 式を *xml_argument* に適用した結果が XVAL になります。

    ```
    (row_pattern[N])/column_pattern/text()
    ```

  - XVAL が空で、*column_definition* にデフォルトの句が含まれている場合、カラムの値はそのデフォルト値になります。

    XVAL が空で、*column_definition* を null 以外に指定すると、例外が発生します。

    それ以外の場合、カラムの値は null 値になります。
  - XVAL が空ではなく、カラムのデータ型が char、varchar、text、unitext、unichar、univarchar、java.lang.String である場合は、XVAL を非エンティティ化してください。
  - カラムの値は次の式の結果になります。

    ```
    convert(datatype,XVAL)
    ```

year

説明　指定した日付の datepart の年を表す整数を返します。

構文　year(*date_expression*)

パラメータ　*date_expression*

datetime、smalldatetime、date、または time 型の式、または datetime フォーマットの文字列です。

例　整数 03 を返します。

```
year("11/02/03")
----------
03
(1 row(s) affected)
```

使用法　year(*date_expression*) は datepart(*yy*, *date_expression*) と同じです。

規格　ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション　すべてのユーザが year 関数を実行できます。

参照　**データ型**　datetime、smalldatetime、date

関数　datepart, day, month

第 3 章

グローバル変数

Adaptive Server のグローバル変数

グローバル変数は、システムの実行中に Adaptive Server によって継続的に更新されるシステム定義変数です。一部のグローバル変数はセッション固有の変数で、それ以外はサーバ・インスタンス固有の変数です。たとえば、*@@error* には、特定のユーザ接続に対してシステムによって生成された最新のエラー番号が保管されます。

アプリケーション・コンテキストの変数を指定するには、get_appcontext および set_appcontext を参照してください。

グローバル変数の値を表示するには、次のように入力します。

```
select variable_name
```

たとえば、次のようにします。

```
select @@char_convert
```

表 3-1 に、Adaptive Server で使用できるグローバル変数をリストします。

表 *3-1: Adaptive Server* のグローバル変数

グローバル変数	定義
@@active_instances	クラスタ内のアクティブなインスタンスの数を返す。
@@authmech	ユーザの認証に使用するメカニズムを指定する読み込み専用変数。
@@bootcount	Adaptive Server が起動された回数を返す。
@@boottime	Adaptive Server が最後に起動された日付と時刻を返す。
@@bulkarraysize	バルク・コピー・インタフェースを使用して転送される前にローカル・サーバ・メモリにバッファリングされるローの数を返す。コンポーネント統合サービスで select into を使用してリモート・サーバにローを転送する場合にのみ使用する。詳細については、『コンポーネント統合サービス・ユーザーズ・ガイド』を参照。
@@bulkbatchsize	バルク・インタフェースを使用して select into *proxy_table* によってリモート・サーバに転送されるローの数を返す。コンポーネント統合サービスで select into を使ってリモート・サーバにローを転送する場合にのみ使用する。詳細については、『コンポーネント統合サービス・ユーザーズ・ガイド』を参照。
@@char_convert	文字セット変換が有効でない場合は 0 を返す。文字セット変換が有効な場合は 1 を返す。
@@cis_rpc_handling	cis rpc handling がオフの場合は 0 を返す。cis rpc handling がオンの場合は 1 を返す。詳細については、『コンポーネント統合サービス・ユーザーズ・ガイド』を参照。

グローバル変数	定義
@@cis_version	コンポーネント統合サービスの日付とバージョンを返す。
@@client_csexpansion	サーバの文字セットをクライアントの文字セットに変換するときに使用する拡張係数を返す。たとえば、値が 2 の場合、サーバの文字セットの文字は、クライアントの文字セットに変換後、最大 2 倍までのバイト数を使用できる。
@@client_csid	クライアントの文字セットが初期化されていない場合は -1 を返す。クライアントの文字セットが初期化されている場合は、接続の syscharsets からクライアントの文字セット ID を返す。
@@client_csname	クライアントの文字セットが初期化されていない場合は、NULL を返す。クライアントの文字セットが初期化されている場合は、接続の文字セットの名前を返す。
@@clusterboottime	クラスタの起動を最初に開始したインスタンスが停止している場合でも、クラスタが最初に起動した日時を返す。
@@clustercoordid	現在のクラスタ・コーディネータのインスタンス ID を返す。
@@clustermode	文字列 “shared-disk cluster” を返す。
@@clustername	クラスタの名前を返す。
@@cmpstate	高可用性環境における Adaptive Server の現在のモードを返す。高可用性環境以外では使用されない。
@@connections	ユーザ・ログインが試行された回数を返す。
@@cpu_busy	Adaptive Server が最後に起動してから、CPU が Adaptive Server の作業に費やした時間 (チック単位) を返す。

@@cursor_rows	スクロール可能カーソルのために設計されたグローバル変数。カーソル結果セットに含まれるローの総数を示す。次の値が返される。
	• -1 − カーソルが次のような場合：
	  • 動的 − 動的カーソルにはすべての変更が反映されるため、カーソルを構成するローの数は常に変動する。構成するローがすべて取得されたかどうかは確認できない。
	  • semi_sensitive であり、スクロール可能だが、スクロールのワーク・テーブルにまだ値が完全には格納されていない − カーソルを構成するロー数は、この値を取得した時点では不明となる。
	• 0 − カーソルが開いていない、最後に開いたカーソルを構成するローがない、または最後に開いたカーソルが閉じられたか割り付け解除された。
	• *n* − 最後に開いた、またはフェッチしたカーソル結果セットに値が完全に格納された。返される値は、カーソル結果セット内のローの総数である。

@@curloid	現在のセッションのロック所有者 ID を返す。
@@datefirst	set datefirst *n* を使用して設定する。ここで *n* は 1 ～ 7 の値を示す。*@@datefirst* で表される各週の指定された最初の曜日を示す tinyint の現在の値を返す。 Adaptive Server のデフォルト値は日曜日 (us_language のデフォルトに基づく) で、`set datefirst 7` を指定して設定する。設定と値の詳細については、set コマンドの datefirst オプションを参照。
@@dbts	現在のデータベースのタイムスタンプを返す。
@@error	システムによって生成された最新のエラー番号を返す。
@@errorlog	Adaptive Server エラー・ログが置かれているディレクトリを、*$SYBASE* ディレクトリ (NT では *%SYBASE%*) の相対パスとしてフル・パスで返す。

グローバル変数	定義
@@failedoverconn	プライマリ・コンパニオンへの接続がフェールオーバして、セカンダリ・コンパニオン・サーバが実行中の場合は、0 より大きい値を返す。この変数は高可用性環境でのみ使用し、セッション固有のものである。
@@fetch_status	戻り値： • 0 － フェッチが成功した。 • -1 － フェッチが成功しなかった。 • -2 － 今後のために予約済み。
@@guestuserid	guest ユーザの ID を返す。
@@hacmpservername	高可用性設定のコンパニオン・サーバの名前を返す。
@@haconnection	接続でフェールオーバ・プロパティが有効になっている場合は、0 より大きい値を返す。これはセッション固有のプロパティである。
@@heapmemsize	ヒープ・メモリ・プールのサイズ (バイト単位) を返す。ヒープ・メモリの詳細については、『システム管理ガイド』を参照。
@@identity	生成された最新の IDENTITY カラム値を返す。
@@idle	Adaptive Server が最後に起動してからアイドル状態になっていた時間 (チック単位) を返す。
@@instanceid	実行されたインスタンスの ID を返す。
@@instancename	実行されたインスタンスの名前を返す。
@@invaliduserid	無効なユーザ ID に対して -1 の値を返す。
@@io_busy	Adaptive Server が I/O 動作に費やした時間 (チック単位) を返す。
@@isolation	現在の Transact-SQL プログラムのセッション固有の独立性レベルの値 (0、1、または 3) を返す。
@@jsinstanceid	Job Scheduler が実行されている、または有効化後に実行されるインスタンスの ID。
@@kernel_addr	カーネル領域を含む最初の共有メモリ領域の開始アドレスを返す。結果は、0x*address pointer value* の形式となる。
@@kernel_size	最初の共有メモリ領域の一部であるカーネル領域のサイズを返す。
@@langid	syslanguages.langid に指定されている、現在使用中の言語のサーバワイドの言語 ID を返す。
@@language	syslanguages.name に指定されている、使用中の言語名を返す。
@@lastlogindate	各ユーザ・ログイン・セッションで使用可能。*@@lastlogindate* には datetime データ型が入り、その値は、現在のセッションが確立される前のそのログイン・アカウントの lastlogindate カラムとなる。この変数は各ログイン・セッションに特有のもので、そのアカウントへの前回のログイン日時を知るために各セッションで使用できる。以前に使用されたことのないアカウントの場合、または “sp_passwordpolicy 'set', enable last login updates” が 0 に設定されている場合、*@@lastlogindate* の値は NULL となる。
@@lock_timeout	set lock wait n を使用して設定する。現在の *lock_timeout* の設定 (ミリ秒単位) を返す。*@@lock_timeout* は n の値を返す。デフォルト値はタイムアウトなし。セッションの開始時に set lock wait n が実行されない場合、*@@lock_timeout* は -1 を返す。
@@maxcharlen	Adaptive Server のデフォルト文字セット中の 1 文字の最大長 (バイト単位) を返す。
@@max_connections	現在のコンピュータ環境の Adaptive Server に対して、同時に確立できる接続の最大数を返す。number of user connections 設定パラメータを使用して、*@@max_connections* の値以下の任意の接続数に対応するように、Adaptive Server を設定できる。
@@maxgroupid	最大のグループ・ユーザ ID を返す。最大値は 1048576。

グローバル変数	定義
@@maxpagesize	サーバの論理ページ・サイズを返す。
@@max_precision	サーバによって設定された decimal および numeric データ型で使用される精度レベルを返す。この値は定数 38。
@@maxspid	spid の最大有効値を返す。
@@maxsuid	最大のサーバ・ユーザ ID を返す。デフォルト値は 2147483647。
@@maxuserid	最大のユーザ ID を返す。最大値は 2147483647。
@@mempool_addr	グローバル・メモリ・プールのテーブルのアドレスを返す。結果は、0x*address pointer value* の形式となる。この変数は内部で使用される。
@@min_poolsize	名前付きキャッシュ・プールの最小サイズ (キロバイト) を返す。256 である DEFAULT_POOL_SIZE および max database page size の現在の値に基づいて計算される。
@@mingroupid	最小のグループ・ユーザ ID を返す。最小値は 16384。
@@minspid	spid の最小値である 1 を返す。
@@minsuid	最小のサーバ・ユーザ ID を返す。最小値は -32768。
@@minuserid	最小のユーザ ID を返す。最小値は -32768。
@@monitors_active	sp_sysmon によって表示されるメッセージ数を減らす。
@@ncharsize	現在のサーバのデフォルト文字セット中の 1 文字の最大長 (バイト単位) を返す。
@@nestlevel	現在のネスト・レベルを返す。
@@nodeid	現在のインストール環境の 48 ビットのノード識別子を返す。Adaptive Server は、マスタ・デバイスを最初に使用するときにノード ID を生成し、Adaptive Server インストール環境をユニークに識別する。
@@optgoal	クエリの最適化に使用される現在の optimization goal 設定を返す。
@@options	セッションの set オプションの 16 進表現を返す。
@@opttimeoutlimit	クエリの最適化に使用される現在の optimization timeout limit 設定を返す。
@@pack_received	Adaptive Server が読み込んだ入力パケット数を返す。
@@pack_sent	Adaptive Server が書き込みを行った出力パケット数を返す。
@@packet_errors	パケットの読み込みおよび書き込み中に Adaptive Server が検出したエラーの数を返す。
@@pagesize	サーバの仮想ページ・サイズを返す。
@@parallel_degree	現在の最大並列度の設定を返す。
@@probesuid	プローブ・ユーザ ID に対して 2 の値を返す。
@@procid	現在実行中のプロシージャのストアド・プロシージャ ID を返す。
@@quorum_physname	クォーラム・デバイスの物理パスを返す。

グローバル変数	定義
@@recovery_state	次のいずれかの値を返し、これらに基づいて Adaptive Server がリカバリ中かどうかを示す。 • NOT_IN_RECOVERY − Adaptive Server は起動時のリカバリ中またはフェールオーバ・リカバリ中ではない。リカバリは完了しており、オンラインにできるすべてのデータベースがオンラインになっている。 • RECOVERY_TUNING − Adaptive Server はリカバリ中 (起動時またはフェールオーバ) であり、最適なリカバリ・タスク数をチューニング中である。 • BOOTIME_RECOVERY − Adaptive Server は起動時のリカバリ中で、最適なタスク数のチューニングを完了した。すべてのデータベースがリカバリされたわけでなはい。 • FAILOVER_RECOVER − Adaptive Server は HA フェールオーバ時にリカバリ中であり、最適なリカバリ・タスク数のチューニングを完了した。すべてのデータベースがオンラインになっているわけではない。
@@repartition_degree	現在の動的な repartitioning degree 設定を返す。
@@resource_granularity	クエリの最適化に使用される最大リソース使用量ヒントの設定を返す。
@@rowcount	最後のクエリによる影響を受けたローの数を返す。*@@rowcount* の値は、指定されたカーソルが前方スクロールのみか、スクロール可能であるかで異なる。 カーソルがデフォルトの非スクロール可能カーソルであれば、*@@rowcount* の値は、結果セットにフェッチされたロー数に等しくなるまで、前方のみに 1 つずつ増える。これらのローは、基本となるテーブルからクライアントへフェッチされる。*@@ rowcount* の最大値は、結果セットに含まれるロー数に等しい。 デフォルト・カーソルでは、ローを返さないかローに影響を与えないコマンドが実行されると、*@@rowcount* が 0 に設定される。たとえば、if コマンドや set コマンド、または update 文や delete 文は、ローに影響を与えない。 スクロール可能カーソルの場合、*@@rowcount* に最大値はない。値は fetch のたびに方向に関係なく増加し、最大値はない。スクロール可能カーソルの *@@rowcount* 値には、基本となるテーブルからではなく、結果セットからクライアントにフェッチされたローの数が反映される。
@@scan_parallel_degree	ノンクラスタード・インデックス・スキャンの現在の最大並列度の設定を返す。
@@servername	Adaptive Server の名前を返す。
@@setrowcount	set rowcount の現在の値を返す。
@@shmem_flags	共有メモリ領域のプロパティを返す。この変数は内部で使用される。合計 13 の異なるプロパティ値があり、それぞれ整数の 13 ビットに対応している。有効な値は、低いビットから順に、MR_SHARED、MR_SPECIAL、MR_PRIVATE、MR_READABLE、MR_WRITABLE、MR_EXECUTABLE、MR_HWCOHERENCY、MR_SWCOHERENC、MR_EXACT、MR_BEST、MR_NAIL、MR_PSUEDO、MR_ZERO である。
@@spid	現在のプロセスのサーバ・プロセス ID を返す。
@@sqlstatus	fetch 文の実行結果であるステータス情報 (例外の警告) を返す。
@@ssl_ciphersuite	現在の接続で SSL が使用されていない場合に NULL を返し、それ以外の場合に現在の接続での SSL ハンドシェイクに選択された暗号スイートの名前を返す。
@@stringsize	toString() メソッドから返される文字データのサイズを返す。デフォルトは 50。最大値は 2GB。0 の値を指定すると、デフォルト値となる。詳細については、『コンポーネント統合サービス・ユーザーズ・ガイド』を参照。
@@system_busy	Adaptive Server でシステム・タスクが実行されていた時間 (チック単位)。[1]

グローバル変数	定義
@@sys_tempdbid	実行中のインスタンスの有効なローカル・システム・テンポラリ・データベースのデータベース ID を返す。
@@system_view	セッション固有のシステム・ビュー設定として “instance” または “cluster” を返す
@@tempdbid	セッションに割り当てられたテンポラリ・データベースの有効なテンポラリ・データベース ID (dbid) を返す。
@@textcolid	*@@textptr* が参照するカラムのカラム ID を返す。
@@textdataptnid	@@textptr から参照されるカラムが含まれるテキスト・パーティションのパーティション ID を返す。
@@textdbid	*@@textptr* が参照するカラムを持つオブジェクトを格納しているデータベース ID を返す。
@@textobjid	*@@textptr* が参照するカラムを持つオブジェクトのオブジェクト ID を返す。
@@textptnid	@@textptr から参照されるカラムが含まれているデータ・パーティションのパーティション ID を返す。
@@textptr	プロセスが最後に挿入または更新した、text、unitext、または image カラムのテキスト・ポインタを返す (textptr 関数とは異なる)。
@@textptr_parameters	textptr_parameters 設定パラメータの現在のステータスがオフの場合は、0 を返す。textptr_parameters の現在のステータスがオンの場合は、1 を返す。詳細については、『コンポーネント統合サービス・ユーザーズ・ガイド』を参照。
@@textsize	select が返す text、unitext、または image データのバイト数に対する制限を返す。isql のデフォルトの制限値は 32K バイト。デフォルト値は、クライアント・ソフトウェアによって異なる。デフォルトごとに set textsize を使用して変更できる。
@@textts,	*@@textptr* が参照するカラムのテキスト・タイムスタンプを返す。
@@thresh_hysteresis	スレッショルドをアクティブ化するために必要な空き領域の減少量を返す。ヒステリシス値と呼ばれるこの値は、2 KB のデータベース・ページ単位で測定される。これは、データベース・セグメント上にスレッショルドをどのくらい接近させて配置できるかを判別する。
@@timeticks	チックごとのマイクロ秒数を返す。1 チックの長さはマシンによって異なる。
@@total_errors	読み込みおよび書き込み中に Adaptive Server が検出したエラーの数を返す。
@@total_read	Adaptive Server によるディスク読み込み回数を返す。
@@total_write	Adaptive Server によるディスク書き込み回数を返す。
@@tranchained	Transact-SQL プログラムの現在のトランザクション・モードが非連鎖モードの場合は、0 を返す。Transact-SQL プログラムの現在のトランザクション・モードが連鎖モードである場合は 1 を返す。
@@trancount	現在のユーザ・セッションにおけるトランザクションのネスト・レベルを返す。
@@transactional_rpc	リモート・サーバへの RPC がトランザクション指向の場合は 0 を返す。リモート・サーバへの RPC が非トランザクション指向の場合は 1 を返す。詳細については、『リファレンス・マニュアル』の enable xact coordination および set option transactional_rpc を参照。また、『コンポーネント統合サービス・ユーザーズ・ガイド』も参照。

グローバル変数	定義
@@transtate	現在のユーザ・セッションで文を実行した後のトランザクションの現在の状態を返す。
@@unicharsize	unichar の文字のサイズである 2 を返す。
@@user_busy	Adaptive Server でユーザ・タスクが実行されていた時間 (チック単位)。[1]
@@version	Adaptive Server の現在のリリースの日付、バージョン文字列などを返す。
@@version_number	Adaptive Server の現在のリリースのバージョン全体を整数として返す。
@@version_as_integer	「Adaptive Server の現在のリリースの最新のアップグレード・バージョン番号を整数で返す。たとえば、Adaptive Server バージョン 12.5、12.5.0.3、または 12.5.1 を実行している場合、*@@version_as_integer* は 12500 を返す。

[1]*@@user_busy* と *@@system_busy* の値の合計は、*@@cpu_busy* の値と一致します。

クラスタード環境でのグローバル変数の使用

@@servername を使用する場合、Cluster Edition ではインスタンス名ではなくクラスタの名前が返されます。インスタンスの名前を返すには、*@@instancename* を使用します。

ノンクラスタード環境の Adaptive Server では、レコードが挿入されるたびに *@@identity* の値が変わります。挿入された最新のレコードに、IDENTITY プロパティが指定されたカラムが入っている場合、*@@identity* はこのカラムの値に設定されます。そうでない場合は、“0”(無効な値) に設定されます。この変数がセッション固有である場合は、このセッションで最後に行われた挿入に基づいて値が設定されます。

クラスタード環境では、テーブルに対して複数のノードで挿入が実行されるので、セッション固有の動作は *@@identity* に保持されません。クラスタード環境では、*@@identity* の値は、クラスタに最後に挿入されたレコードではなく、現在のセッションでノードに最後に挿入されたレコードによって決まります。

第 4 章

式、識別子、およびワイルドカード文字

この章では、Transact-SQL の式、有効な識別子、ワイルドカード文字について説明します。

この章では、次の項目について説明します。

式

式は、1 つまたは複数の定数、リテラル、関数、カラム識別子、変数またはその組み合わせを演算子で区切ったもので、式は 1 つの値を返します。式には、「算術式」、「関係式」、「論理式 (またはブール式)」、および「文字列式」など、いくつかの種類があります。Transact-SQL 句には、式中にサブクエリを使用できるものがあります。式では、case 式を使用できます。

表 4-1 は、Adaptive Server の構文で使用できる各種の式のリストを示します。

表 *4-1:* 構文で使用されている式の種類

使用法	定義
expression	定数、リテラル、関数、カラム識別子、変数、パラメータが含まれる。
logical expression	TRUE (真)、FALSE (偽)、UNKNOWN (未知または認識できない) のいずれかを返す。
constant expression	常に同じ値を返す。たとえば “5+3”、“ABCDE” など。
float_expr	任意の浮動小数の式、または暗黙的に浮動小数の値に変換される式。
integer_expr	任意の整数式、または暗黙的に整数の値に変換される式。
numeric_expr	1 つの値を返す数値式。
char_expr	1 つの文字値を返す式。
binary_expression	1 つの binary 値または varbinary 値を返す式。

式のサイズ

バイナリ・データまたは文字データを返す式の長さは、最大 16384 バイトです。しかし、以前のバージョンの Adaptive Server では、式の長さは最大で 255 バイトまででした。以前のリリースの Adaptive Server からアップグレードしたときに、ストアド・プロシージャまたはスクリプトで 255 バイトまでの結果文字列が格納されている場合は、残りの部分はトランケートされます。式の追加された長さに対応するには、これらのストアド・プロシージャとスクリプトの書き換えが必要になる場合があります。

算術式と文字式

算術式および文字式の一般的なパターンを次に示します。

{*constant* | *column_name* | *function* | (*subquery*)
| (*case_expression*)}
[{*arithmetic_operator* | *bitwise_operator* |
string_operator | *comparison_operator* }
{*constant* | *column_name* | *function* | (*subquery*)
| *case_expression*}]...

関係式と論理式

論理式または関係式は、TRUE (真)、FALSE (偽)、UNKNOWN (未知または認識できない) のいずれかを返します。一般的なパターンを示します。

expression *comparison_operator* [any | all] *expression*

expression [not] in *expression*

[not]exists *expression*

expression [not] between expression and *expression*

expression [not] like "*match_string*"
[escape "*escape_character* "]

not expression like "*match_string*"
[escape "*escape_character* "]

expression is [not] null

not *logical_expression*

logical_expression {and | or} *logical_expression*

演算子の優先度

演算子の優先度レベルを次に示します。1 が最も高く、6 が最も低いレベルになります。

1　単項 (単独の引数) - + ~

2　* / %

3　2 項 (2 つの引数) + - & | ^

4　not

5　and

6　or

式のすべての演算子が同じ優先度であれば、左から右に実行されます。実行の順序は、カッコを使用して変更できます。最も内側にネストされた式が最初に処理されます。

算術演算子

Adaptive Server は、次の算術演算子を使用します。

表 *4-2:* 算術演算子

演算子	意味
+	加算
-	減算
*	乗算
/	除算
%	モジュロ (Transact-SQL 拡張機能)

加算、減算、除算、乗算は、真数、概数、通貨型のカラムで使用できます。

モジュロ演算子は smallmoney、money、numeric、float、または real カラムには使用できません。モジュロは、2 つの整数を使用する除算の整数の剰余を検出します。たとえば、21 % 11 = 10 は、21 割る 11 は、商が 1 で余りが 10 になることを表します。

float と int など、混合データ型に対して算術演算を行うと、Adaptive Server は特定の規則に従って結果の型を決定します。詳細については、「第 1 章 システム・データ型とユーザ定義データ型」を参照してください。

ビット処理演算子

ビット処理演算子は、整数値型のデータで使用する Transact-SQL 拡張機能です。これらの演算子は、それぞれの整数のオペランドをバイナリ表現に変換し、次にカラムごとにオペランドを評価します。値 1 は true に対応します。値 0 は false に対応します。

表 4-3 に、0 と 1 のオペランドの結果を示します。どちらかのオペランドが NULL の場合は、ビット処理演算子は NULL を返します。

表 *4-3:* ビット処理演算の真理値表

& (and)	1	0
1	1	0
0	0	0

| (or)	1	0
1	1	1
0	1	0

^ (排他 or)	1	0
1	0	1
0	1	0

~ (not)	
1	FALSE
0	0

表 4-4 の例は、次の 2 つの tinyint 引数を使用します。A = 170 (バイナリ形式では 10101010) および B = 75 (バイナリ形式では 01001011) を使用しています。

表 *4-4:* ビット処理演算の例

オペレーション	バイナリ形式	結果	説明
(A & B)	10101010 01001011 ---------- 00001010	10	A と B の両方が 1 のときは、結果カラムには 1 が表示される。それ以外の場合は、結果カラムには 0 が表示される。
(A | B)	10101010 01001011 ---------- 11101011	235	A と B のいずれか片方または両方が 1 のときは、結果カラムには 1 が表示される。それ以外の場合は、結果カラムには 0 が表示される。

オペレーション	バイナリ形式	結果	説明
(A ^ B)	10101010 01001011 ---------- 11100001	225	A か B の片方だけが 1 のときは、結果カラムには 1 が表示される。
(~A)	10101010 ---------- 01010101	85	すべての 1 は 0 に変更され、すべての 0 は 1 に変更される。

文字列連結演算子

複数の文字式やバイナリ式を連結するには、文字列演算子 + や || (二重パイプ) を使用します。次の例では、著者の名前をカラム見出し Name の下に姓、名前の順で表示します。姓のあとにはカンマを置きます。たとえば、“Bennett,Abraham” のようになります。

```
select Name = (au_lname + ", " + au_fname)
    from authors
```

次の例の結果は、"abcdef", "abcdef" になります。

```
select "abc" + "def", "abc" || "def"
```

次の演算では文字列 “abc def” を返します。char、varchar、unichar、nchar、nvarchar、text の連結文と、varchar、univarchar の挿入文と代入文では、空の文字列は 1 つのスペースとして解釈されます。

```
select "abc" + "" + "def"
```

文字式でもバイナリ式でもない式を連結するには、必ず convert を使用します。

```
select "The date is " +
    convert(varchar(12), getdate())
```

NULL と連結された文字列は、その文字列の値として評価されます。これは、NULL で連結された文字列は、NULL に評価すると定められている SQL 標準では例外です。

比較演算子

Adaptive Server が使用する比較演算子を表 4-5 に示します。

表 *4-5:* 比較演算子

演算子	意味
=	等しい
>	より大きい
<	より小さい
>=	以上
<=	以下
<>	等しくない
!=	Transact-SQL 拡張機能　等しくない
!>	Transact-SQL 拡張機能　より大きくない
!<	Transact-SQL 拡張機能　より小さくない

文字データの比較では、< はサーバのソート順の初めの方に近いことを意味し、> はソート順の終わりの方に近いことを意味します。大文字と小文字を区別しないソート順では、ある文字の大文字と小文字は同じ文字として扱われます。使用している Adaptive Server のソート順を表示するには、sp_helpsort を使用します。比較の場合、後続ブランクは無視されます。たとえば、“Dirk” と “Dirk” は同じです。

データの比較では、< はより前を意味し、> はより後を意味します。

比較演算子とともに使用するすべての文字および datetime データは、一重引用符か二重引用符で囲みます。

```
= "Bennet"
> "May 22 1947"
```

標準以外の演算子

次に示す演算子は Transact-SQL 拡張機能です。

- モジュロ演算子：%
- 否定の比較演算子：!>、!<、!=
- ビット処理演算子：~、^、|、&
- ジョイン演算子：*=、=*

any、*all*、および *in* の使用

any は <、>、または = のいずれかと、サブクエリとともに使用します。これは、サブクエリ内で検索された値のいずれかが、外側の文の where 句または having 句の値と一致すると、結果を返します。詳細については、『Transact-SQL ユーザーズ・ガイド』を参照してください。

all は < か > のいずれかと、サブクエリとともに使用します。これは、サブクエリ内で検索されたすべての値が、外側の文の where 句または having 句の値より小さい (<) か、または大きい (>) と、結果を返します。詳細については、『Transact-SQL ユーザーズ・ガイド』を参照してください。

in は、2 番目の式が返した値のいずれかが、1 番目の式の値と一致する場合に、結果を返します。2 番目の式は、サブクエリ、またはカッコで囲んだ値のリストです。in は = any と等しくなります。詳細については、『リファレンス・マニュアル：ビルディング・ブロック』の where 句 の を参照してください。

否定と存在確認

not はキーワードの意味または論理式の意味を否定します。

特定の結果が存在するかどうかをテストするには、exists の後にサブクエリを指定します。

範囲

between は範囲開始のキーワードです。and は範囲終了のキーワードです。次の範囲は包括的範囲です。

```
where column1 between x and y
```

次の範囲は包括的範囲ではありません。

```
where column1 > x and column1 < y
```

式での null の使用

is null と is not null は、null 値が使用可能であると定義されたカラムのクエリで使用します。

ビット処理演算子と算術演算子のある式は、いずれかのオペランドが null である場合は、NULL と評価されます。次の式は *column1* が NULL であれば NULL と評価されます。

```
1 + column1
```

TRUE を返す比較

一般に、null の値を比較した結果は、UNKNOWN になります。これは、NULL が特定の値や他の NULL に等しいか (または等しくないか) どうかを判断できないためです。ただし、次のような場合には、*expression* が、NULL と評価されるカラム、変数、リテラル、またはその組み合わせであると、TRUE が返されます。

- *expression* is null
- *expression* = null
- *expression* = *@x* (この *@x* は NULL を含む変数またはパラメータです。この例外によって、null のデフォルト・パラメータを使用したストアド・プロシージャの作成が簡単になります。)
- *expression* != *n* (この *n* は NULL を含まないリテラルです。この場合 *expression* は NULL に評価されます。)

これらの式の否定は、式が NULL と評価されないときに TRUE を返します。

- *expression* is not null
- *expression* != null
- *expression* != *@x*

注意 これらの式の右端の値は、リテラルの null であるか、NULL を含む変数またはパラメータです。比較の一番右側が式 (*@nullvar* + 1 など) の場合、式全体が NULL に評価されます。

この規則に従って、null のカラムの値は、他の null のカラムの値に結合されません。1 つの where 句のなかで、null のカラムの値を他の null のカラムの値と比較すると、どの比較演算子を使った場合にも、null の値には UNKNOWN が返され、結果にはローは含まれません。たとえば次のクエリは、両方のテーブルにおいて column1 に NULL が含まれている場合、ローを返しません (ただし他のローを返すことはあります)。

```
select column1
from table1, table2
where table1.column1 = table2.column1
```

FALSE と UNKNOWN の違い

FALSE も UNKNOWN も値を返しませんが、FALSE と UNKNOWN には論理的に重要な違いがあります。これは、false の反対 (“not false”) が true であるためです。たとえば、“1 = 2” は false と評価され、その反対の “1 != 2” は true と評価されます。しかし、“not unknown” は、unknown のままです。比較に null 値が含まれている場合は、式を否定して反対のローの集合や反対の真理値を取得することはできません。

文字列としての “NULL” の使用

create table 文内で NULL が指定され、ユーザが明示的に NULL を (引用符を付けずに) 挿入したカラム、またはデータが挿入されていないカラムだけが、null 値を含みます。文字カラムには、文字列 “NULL” (引用符付き) をデータとして入力しないでください。入力すると、混乱を招きます。代わりに “N/A” や “none” などの値を使用してください。値 NULL を明示的に入力したい場合は、一重または二重の引用符を使用しないでください。

NULL と空の文字列の比較

空の文字列 (“ ” や ‘ ’) は、常にシングル・スペースとして変数やカラム・データに格納されます。次の連結文は、“abc def” と等価ですが、“abcdef” とは等価ではありません。

```
"abc" + "" + "def"
```

空文字列は NULL として評価されることはありません。

式の接続

and は 2 つの式を接続し、両方が true の場合に結果を返します。or は 2 つ以上の条件を接続し、いずれかの条件が true であれば、結果を返します。

1 つの文中で複数の論理演算子が使用されるときは、or の前に and が評価されます。実行の順序はカッコを使用して変更できます。

表 4-6 に、null 値を使用するものも含めて、論理演算の結果を示します。

表 *4-6:* 論理式の真理値表

and	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	UNKNOWN
FALSE	FALSE	FALSE	FALSE
NULL	UNKNOWN	FALSE	UNKNOWN

or	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	UNKNOWN
NULL	TRUE	UNKNOWN	UNKNOWN

not	
TRUE	FALSE
FALSE	TRUE
NULL	UNKNOWN

UNKNOWN という結果は、1 つ以上の式が NULL と評価されて、演算の結果が TRUE であるか FALSE であるか判別できないことを示します。詳細については、「式での null の使用」(347 ページ) を参照してください。

式でのカッコの使用

式で要素をグループにまとめるにはカッコを使用します。構文内で「式」が変数として指定されている場合は、単純式が想定されます。1 つの論理式だけが受け入れ可能な場合は、「論理式」が指定されます。

文字式の比較

文字定数式は、varchar として扱われます。varchar 以外の変数やカラム・データと比較する場合、比較にはデータ型の優先度の規則が適用されます (つまり、優先度の低いデータ型が優先度の高いデータ型に変換されます)。暗黙的なデータ型の変換がサポートされていない場合は、convert 関数を使用する必要があります。

char 式と varchar 式との比較は、データ型の優先度の規則に従います。「低い」データ型は「高い」データ型に変換されます。すべての varchar 式は、比較のために char に変換されます (つまり、後続ブランクが追加されます)。unichar 式と char (varchar、nchar、nvarchar) 式を比較する場合、char 式が unichar に変換されます。

空文字列の使用

varchar データや univarchar データの代入文や insert では、空の文字列 ("") や ('') は 1 つのブランクとして解釈されます。varchar、char、nchar、nvarchar データの連結では、空の文字列は 1 つのスペースとして解釈されます。たとえば、次の例では、“abc def” として格納されます。

```
"abc" + "" + "def"
```

空文字列は NULL として評価されることはありません。

文字式に含む引用符

char または varchar エントリ内にリテラル引用符を指定するには、2 つの方法があります。1 つ目の方法は、引用符を 2 つ使用する方法です。たとえば一重引用符で文字の入力を始めて、入力の一部に一重引用符を使いたい場合には、一重引用符を 2 つ使います。

```
'I don''t understand.'
```

二重引用符の場合には、次のようにします。

```
"He said, ""It's not really confusing."""
```

2 つ目の方法は、片方の引用符を別の引用符で囲む方法です。つまり、二重引用符を含むエントリを一重引用符で (または一重引用符を含むエントリを二重引用符で) 囲みます。例を示します。

```
'George said, "There must be a better way."'
"Isn't there a better way?"
'George asked, "Isn"t there a better way?"'
```

継続文字の使用

画面上で次の行に文字列を続けるには、改行する前にバックスラッシュ (¥) を入力します。

識別子

識別子は、データベース、テーブル、ビュー、カラム、インデックス、トリガ、プロシージャ、デフォルト、ルール、カーソルなどのデータベース・オブジェクトの名前です。

オブジェクト名または識別子の長さに対する制限は、通常の識別子の場合に 255 バイト、区切り識別子の場合に 253 バイトです。この制限は、テーブル名、カラム名、インデックス名などのほとんどのユーザ定義識別子に適用されます。制限が拡張されたため、一部のシステム・テーブル (カタログ) および組み込み関数も拡張されました。

変数では “@” が 1 文字としてカウントされるため、変数名は最大で 254 文字です。

次のリストに、この制限の影響を受ける識別子、システム・テーブル、組み込み関数を示します。

次の識別子の最大長は 255 バイトになりました。

- テーブル名
- カラム名
- インデックス名
- ビュー名
- ユーザ定義データ型
- トリガ名
- デフォルト名
- ルール名
- 制約名

- プロシージャ名
- 変数名
- JAR 名
- LWP または動的文の名前
- 関数名
- 時間範囲の名前
- アプリケーション・コンテキスト名

Adaptive Server のほとんどのユーザ定義識別子では、使われている文字がシングルバイト文字でもダブルバイト文字でも、最大長は 255 バイトです。それ以外の場合は、最大 30 バイトです。255 バイトの識別子と 30 バイトの識別子のリストについては、『Transact-SQL ユーザーズ・ガイド』を参照してください。

識別子の先頭文字は、現在の文字セットに定義されている文字か、アンダースコア (_) 文字を使用してください。

注意 この規則の例外として、シャープ記号 (#) で始まるテンポラリ・テーブル名と、at 記号 (@) で始まるローカル変数名があります。

後続の文字として、英字、数字、記号 (#、@、_)、ドル記号 ($)、円記号 (•)、または通貨ポンド記号 (£) などの通貨記号を使用できます。識別子には、!、%、^、&、*、. などの特別な記号や、埋め込みスペースを使うことはできません。

Transact-SQL コマンドなどの予約語も、識別子には使用できません。予約語の完全なリストについては、「第 5 章 予約語」を参照してください。

ダッシュ記号 (-) は識別子として使用できません。

短い識別子

次の識別子の最大長は 30 バイトです。

- カーソル名
- サーバ名
- ホスト名
- ログイン名
- パスワード
- ホスト・プロセス ID
- アプリケーション名
- 初期言語名

- 文字セット名
- ユーザ名
- グループ名
- データベース名
- 論理デバイス名
- セグメント名
- セッション名
- 実行クラス名
- エンジン名
- クワイス・タグ名
- キャッシュ名

で始まるテーブル (テンポラリ・テーブル)

名前がシャープ記号 (#) で始まるテーブルは、テンポラリ・テーブルです。別の種類のオブジェクトを作成するときに、シャープ記号で始まる名前を指定することはできません。

Adaptive Server は、セッションごとにユニークな名前を付けるために、テンポラリ・テーブル名に対して特別な操作を行います。テンポラリ・テーブルを 238 バイトより短い名前で作成すると、sysobjects は tempdb 内でテーブル名をユニークにするために名前に 17 バイトを追加します。テーブル名が 238 バイトより長い場合、sysobjects は最初の 238 バイトだけを使用し、それに 17 バイトを付加してテーブル名をユニークにします。

15.0 より前のバージョンの Adaptive Server では、sysobjects のテンポラリ・テーブル名は 30 バイトでした。13 バイトより短いテーブル名を使用していた場合は、長さが 13 バイトになるまで名前にアンダースコア (_) が付加され、さらに別の 17 バイトの文字が追加されて 30 バイトになりました。

大文字と小文字の区別と識別子

識別子やデータの大文字と小文字の区別は、ご使用の Adaptive Server にインストールされているソート順に応じて決まります。1 バイト文字セットの大文字と小文字の区別の指定を変更するには、Adaptive Server のソート順の設定を変更します。詳細については、『システム管理ガイド』を参照してください。ユーテイリティ・プログラム・オプションでは、大文字と小文字の区別が重要になります。

Adaptive Server が、大文字と小文字を区別しないソート順でインストールされている場合は、MyTable や mytable のような名前のテーブルがすでに存在していると、MYTABLE という名前のテーブルを作成することはできません。同様に、次のコマンドは MYTABLE、MyTable、mytable、あるいは大文字と小文字を任意に組み合わせた同じ名前のテーブルから、ローを返します。

```
select * from MYTABLE
```

オブジェクト名の一意性

データベースでは、オブジェクト名を一意 (ユニーク) にする必要はありません。しかし、テーブルの内部のカラム名とインデックス名は、一意にする必要があります。また、他のオブジェクト名も、データベース内のそれぞれの所有者に対して一意にする必要があります。Adaptive Server では、データベース名を一意にしてください。

区切り識別子の使用

「区切り識別子」は、二重引用符で囲まれたオブジェクト名です。区切り識別子を使用することによって、オブジェクト名に関する一定の制限を回避できます。テーブル名、ビュー名、カラム名は、引用符で区切ることができます。その他のオブジェクトには、区切り識別子は使えません。

区切り識別子には予約語を使用でき、最初の文字に英字以外のものを使用できます。また、規則では認められていない文字も使用できます。253 バイトを超えることはできません。

警告！ すべてのフロント・エンド・アプリケーションで、区切り識別子が認識されるとは限りません。区切り識別子をシステム・プロシージャへのパラメータとして使用しないでください。

区切り識別子を作成または参照する前に、以下の文を実行してください。

```
set quoted_identifier on
```

文で区切り識別子を使用するときには、必ずこの識別子を二重引用符で囲んでください。たとえば、次のようにします。

```
create table "1one"(col1 char(3))
create table "include spaces" (col1 int)
create table "grant"("add" int)
insert "grant"("add") values (3)
```

quoted_identifier オプションが on に設定されている間は、二重引用符ではなく一重引用符で文字やデータ文字列を囲んでください。これらの文字列を二重引用符で区切ると、Adaptive Server はこれを識別子として扱います。たとえば *1table* の *col1* に文字列を挿入するには、次のように指定します。

```
insert "1one"(col1) values ('abc')
```

次のようには指定しないでください。

```
insert "1one"(col1) values ("abc")
```

カラムに一重引用符を挿入するには、連続する 2 つの一重引用符を使用してください。たとえば、*col1* に “a’b” を挿入するには、次のようにします。

```
insert "1one"(col1) values('a''b')
```

引用符を含む構文

文の構文で識別子を引用符付き文字列に入れるように要求されている場合、quoted_identifier オプションが on に設定されていれば、識別子の前後に二重引用符を使用する必要はありません。たとえば、次のようにします。

```
set quoted_identifier on
create table '1one' (c1 int)
```

ただし、object_id() では文字列が必須のため、テーブル名を引用符で囲んで情報を選択します。

```
select object_id('1one')
-----------------------
  896003192
```

引用符を 2 つ使用すると、引用符付き識別子に埋め込み二重引用符を含めることができます。

```
create table "embedded""quote" (c1 int)
```

ただし、文の構文でオブジェクト名を文字列として表現することが要求されている場合には、引用符を 2 つ使用する必要はありません。

```
select object_id('embedded"quote')
```

修飾されたオブジェクト名によるテーブルまたはカラムの識別

テーブルやカラムを一意に識別するには、これらを修飾する別の名前を追加します。たとえばデータベース名、所有者名、(カラムについては) テーブル名やビュー名で修飾できます。それぞれの修飾子はピリオドで区切ります。たとえば、次のようにします。

```
database.owner.table_name.column_name
database.owner.view_name.column_name
```

命名規則は次のようになります。

```
[[database.]owner.]table_name
[[database.]owner.]view_name
```

オブジェクト名の中での区切り識別子の使用

set quoted_identifier on を使用すると、修飾されたオブジェクト名の個々の部分を二重引用符で囲むことができます。引用符が必要なそれぞれの修飾子について、引用符を一組ずつ使います。たとえば、次のようにします。

```
database.owner."table_name"."column_name"
```

次のような使い方はできません。

```
database.owner."table_name.column_name"
```

所有者名の省略

オブジェクトを識別するための十分な情報がシステムに提供されている場合には、名前の中間部分を省略し、省略した場所をドットで示すことができます。

```
database..table_name

database..view_name
```

現在のデータベース内のユーザ自身が所有するオブジェクトの参照

現在のデータベース内にある、ユーザ自身が所有するオブジェクトを参照するときに、データベース名や所有者名を使う必要はありません。*owner* のデフォルト値は現在のユーザで、*database* のデフォルト値は現在のデータベースです。

データベース名と所有者名で修飾されていないオブジェクトを参照する場合は、Adaptive Server は現在のデータベースで、ユーザが所有しているオブジェクトの中から、そのオブジェクトを見つけようとします。

データベース所有者が所有しているオブジェクトの参照

所有者名を省略し、その名前のオブジェクトを所有していない場合は、Adaptive Server はデータベース所有者が所有している同名のオブジェクトを検索します。データベース所有者が所有しているオブジェクトを参照する場合に、そのオブジェクトを修飾する必要があるのは、同じ名前のオブジェクトをユーザが所有している場合だけです。しかし、他のユーザが所有しているオブジェクトについては、ユーザ自身が同じ名前のオブジェクトを所有しているかどうかとは関係なく、他のユーザの名前でオブジェクトを修飾する必要があります。

修飾された識別子の一貫した使用

同じ文でカラム名とテーブル名を修飾する場合には、それぞれに同じ修飾式を使用するようにしてください。これらは文字列として評価され、一致する必要があります。そうでない場合はエラーが返されます。例 2 は正しくない使い方です。この例では、カラム名の構文スタイルが、テーブル名で使用している構文スタイルと一致していないためです。

例 1

```
select demo.mary.publishers.city
from demo.mary.publishers

city
-----------------------
Boston
Washington
Berkeley
```

例 2

```
select demo.mary.publishers.city
from demo..publishers
```

カラム・プレフィクス "demo.mary.publishers" が、クエリで使用されるテーブル名またはエイリアス名と一致していません。

識別子の有効性の確認

文字セットを変更したあと、またはテーブルやビューを作成する前に、システム関数 valid_name を使用して、オブジェクト名が Adaptive Server で使用できるかどうかを確認してください。構文は次のとおりです。

```
select valid_name("Object_name")
```

object_name が有効な識別子でない場合 (たとえば無効な文字が入っていたり、30 バイトの長さを超えていたりする場合) は、Adaptive Server は 0 を返します。*object_name* が有効な識別子である場合は、Adaptive Server はゼロ以外の数字を返します。

データベース・オブジェクト名の変更

sp_rename でユーザ・オブジェクト (ユーザ定義データ型など) の名前を変更します。

警告！ テーブルやカラムの名前を変更した場合は、名前が変更されたオブジェクトを使用するプロシージャ、トリガ、ビューもすべて定義し直してください。

マルチバイト文字セットの使用

マルチバイト文字セットでは、さまざまな文字を識別子に使うことができます。たとえば日本語がインストールされたサーバでは、識別子の最初の文字に、全角と半角のカタカナ、ひらがな、漢字、ローマ字、ギリシャ文字、キリル文字、ASCII などの文字を使用できます。

半角カタカナは日本語システムの識別子では有効ですが、異機種間システムでの使用はおすすめできません。半角カタカナは EUC-JIS 文字セットと Shift-JIS 文字セットの間では変換できません。

これは 8 ビット欧州文字のいくつかにもあてはまります。たとえば、Macintosh の文字セット (コードポイント 0xCE) 中にある OE の合字です。この文字は、ISO 8859-1 (iso_1) 文字セットには含まれていません。Macintosh から ISO 8859-1 文字セットに変換するデータに O と E の合字があると、変換エラーが発生します。

変換できない文字がオブジェクト識別子に含まれていると、クライアントはそのオブジェクトに直接アクセスできなくなります。

ワイルドカード文字によるパターン一致

match_string では、ワイルドカード文字は、1 文字以上の文字を表すか、または文字の範囲を表します。*match_string* は、式の中で検索するパターンからなる文字列です。次のように、定数、変数、カラム名、連結した式などの任意の組み合わせを使うことができます。

```
like @variable + "%".
```

一致する文字列が定数の場合は、必ず一重引用符か二重引用符で囲んでください。

特定のパターンと一致する文字や日付文字列を検索するには、キーワード like とワイルドカード文字を使います。ただし、秒とミリ秒を検索する場合は、like を使うことはできません。詳細については、「datetime データでのワイルドカード文字の使用」(364 ページ) を参照してください。

一致する文字列に似た (like) 文字または日付／時刻の情報、あるいは似ていない (not like) 文字または日付／時刻の情報を検索するには、where と having 句で次のようなワイルドカード文字を使用します。

{where | having} [not]
expression [not] like *match_string*
[escape "*escape_character* "]

この *expression* には、文字値を持つ関数、定数、カラム名の任意の組み合わせを指定できます。

ワイルドカード文字を like とともに使用しない場合、ワイルドカード文字に特別な意味はありません。たとえば次のクエリでは、“415%” の 4 文字で始まる任意の電話番号が検索されます。

```
select phone
from authors
where phone = "415%"
```

not like の使用

not like は、特定のパターンと一致しない文字列を見つけるために使用します。次に示す 2 つのクエリは、同じことを意味します。いずれも authors テーブルから、地域コード 415 以外で始まるすべての電話番号を探します。

```
select phone
from authors
where phone not like "415%"

select phone
from authors
where not phone like "415%"
```

たとえば次のクエリは、データベース内の名前が “sys” で始まるシステム・テーブルを検索します。

```
select name
from sysobjects
where name like "sys%"
```

システム・テーブルでないすべてのオブジェクトを検索するには、次のように指定します。

```
 not like "sys%"
```

オブジェクトの合計数が 32 で、like でパターンが一致する名前が 13 個見つかったとすると、not like ではパターンに一致しないオブジェクトが 19 個見つかります。

not like を否定のワイルドカード文字 [^] とともに使うと、異なった結果になることがあります (「脱字記号 (^) ワイルドカード文字」(362 ページ) を参照してください)。not like のパターンで得られた結果は、like と ^ によって得られた結果と異なることがあります。これは、not like は like のパターン全体と一致しない項目を検索しますが、like と否定のワイルドカード文字の組み合わせでは 1 文字ずつ順に評価されるためです。

like "[^s][^y][^s]%" のようなパターンでは、同じ結果が得られない可能性があります。この場合、19 個ではなく、14 個だけが検出されることがあります。ここでは、“s” で始まる名前、“y” を 2 番目の文字として持つ名前、“s” を 3 番目の文字として持つ名前がすべて、システム・テーブル名と同様に検索結果から外されます。これは、否定のワイルドカードでは、一致文字列が 1 文字ずつ順に評価されるためです。評価のある時点で失敗した一致は、削除されます。

大文字と小文字の区別とアクセントの区別の無視

Adaptive Server で使用しているソート順が大文字と小文字を区別しないものである場合は、*expression* と *match_string* を比較する際に、大文字と小文字の区別は無視されます。たとえば、大文字と小文字を区別しない Adaptive Server では、次の句を指定すると、“Smith”、“smith”、“SMITH” などが返されます。

```
where col_name like "Sm%"
```

ご使用の Adaptive Server がアクセントについても無視する場合は、大文字でも小文字でも、アクセントのある文字はどれもアクセントのない文字と同様に扱われます。システム・プロシージャ sp_helpsort では、同じ文字として取り扱われた文字を “=” で結んで表示します。

ワイルドカード文字の使用

一致文字列を各種ワイルドカードと一緒に使用できます。以降の項では、このワイルドカード文字について詳しく説明します。表 4-7 に、ワイルドカード文字を示します。

表 *4-7:* ワイルドカード文字の使用

記号	意味
%	0 個以上の文字の文字列
_	任意の 1 文字
[ ]	指定の範囲 ([a-f]) またはセット ([abcdef]) 内の任意の 1 文字
[^]	指定した範囲 [^a-f] またはセット [^abcdef] に含まれない任意の 1 文字

ワイルドカード文字と一致文字列は、一重または二重引用符で囲みます (like “[dD]eFr_nce”。

パーセント記号 (%) ワイルドカード文字

0 文字以上の文字列を表すには、% ワイルドカード文字を使用します。たとえば、authors テーブルで局番が 415 で始まるすべての電話番号を検索するには、次のように指定します。

```
select phone
from authors
where phone like "415%"
```

“en” という文字が含まれている文字列 (Bennet、Green、McBadden) を検索するには、次のように指定します。

```
select au_lname
from authors
where au_lname like "%en%"
```

like 句では、“%” に続く複数の後続ブランクは切り捨てられ 1 つの後続ブランクになります。たとえば、“%” の後に 2 つのスペースが続く場合、この “%” は、“X ”(スペース 1 つ)、“X ”(スペース 2 つ)、“X   ”(スペース 3 つ)、または任意の後続スペースと一致します。

アンダースコア (_) ワイルドカード文字

1 文字を表すためには、アンダースコア (_) ワイルドカード文字を使用します。たとえば “heryl” で終わる 6 文字の名前 (Cheryl など) をすべて検索する場合は、次のように指定します。

```
select au_fname
from authors
where au_fname like "_heryl"
```

角カッコ ([ ]) ワイルドカード文字

[a-f] のような文字の範囲、または [a2Br] のような文字セットを指定するには、これを角カッコで囲みます。範囲を指定した場合は、ソート順で *rangespec1* と *rangespec2* の間に含まれるすべての値が返されます (両端の文字を含みます)。たとえば、“[0-z]” は、7 ビット ASCII での 0-9、A-Z、a-z (およびその他の句読点記号) と一致します。

“inger” で終わり、M から Z までの 1 文字で始まる名前を検索するには、次のように指定します。

```
select au_lname
from authors
where au_lname like "[M-Z]inger"
```

“DeFrance” と “deFrance” の両方を検索するには、次のように指定します。

```
select au_lname
from authors
where au_lname like "[dD]eFrance"
```

オブジェクトを作成するときに、create table [*table_name*] や create dstabase [*dbname*] のように角カッコ識別子を使用するときは、少なくとも 1 つの有効な文字を含める必要があります。

角カッコ識別子内のすべての後続スペースは、オブジェクト名から削除されます。たとえば、次の create table コマンドの実行結果はすべて同じになります。

- create table [tab1<space><space>]
- create table [tab1]
- create table [tab1<space><space><space>]
- create table tab1

この規則は、角カッコ識別子を使用して作成できるすべてのオブジェクトに適用されます。

脱字記号 (^) ワイルドカード文字

脱字記号は、否定のワイルドカード文字です。特定のパターンに一致しない文字列を検索するときに、この記号を使います。たとえば、“[^a-f]” と指定すると、a ～ f の範囲にない文字列が検索されます。また、“[^a2bR]” と指定すると、“a”、“2”、“b”、“R” でない文字列が検索されます。

“M” で始まり、2 番目の文字が “c” でない名前を見つけるには、次のように指定します。

```
select au_lname
from authors
where au_lname like "M[^c]%"
```

範囲を指定した場合は、ソート順で *rangespec1* と *rangespec2* の間に含まれるすべての値が返されます (両端の文字を含みます)。For example,
“[0-z]” は 7 ビット ASCII での 0 ～ 9、A ～ Z、a ～ z、およびその他の句読点記号と一致します。

マルチバイト・ワイルドカード文字の使用

ご使用の Adaptive Server でマルチバイト文字セットが設定されていて、ワイルドカード文字 _、%、-、[、]、^ に対応する 2 バイトの文字が定義されている場合には、対応する文字を一致文字列で使用できます。アンダースコアは、一致文字列のシングルバイト文字またはダブルバイト文字を表します。

リテラル文字としてのワイルドカード文字の使用

文字列の中の %、_、[、]、^ そのものを検索したい場合には、エスケープ文字を使用します。エスケープ文字とともにワイルドカード文字を使うと、Adaptive Server ではワイルドカード文字が他の文字を示すものとは解釈せず、この文字をリテラル文字として解釈します。

Adaptive Server には次の 2 種類のエスケープ文字があります。

- 角カッコ (Transact-SQL 拡張機能)
- escape 句の直後の 1 文字 (SQL 標準への準拠)

エスケープ文字としての角カッコ ([ ]) の使用

パーセント記号、アンダースコア、左カッコのエスケープ文字として、角カッコを使います。右カッコには、エスケープ文字は不要です。右カッコをそのまま使うことができます。ハイフンをリテラル文字として使う場合は、1 組の角カッコの中の最初の文字として使います。

表 4-8 は、like で角カッコをエスケープ文字として使用する例を示します。

表 *4-8:* 角カッコを使用したワイルドカード文字の検索

like 述部	意味
like "5%"	後に 0 文字以上の文字列が続く 5
like "5[%]"	5%
like "_n"	an 、in 、on など
like "[_]n"	_n
like "[a-cdf]"	a、b、c、d、または f
like "[-acdf]"	-、a、c、d、または f
like "[[]"	[
like "]"	]
like “[[]ab]”	[]ab

escape 句の使用

エスケープ文字を指定するには、escape 句を使用します。サーバのデフォルトの文字セットにある任意の 1 文字を、エスケープ文字として指定できます。複数の文字をエスケープ文字として使おうとすると、Adaptive Server は例外を生成します。

次のような理由から、既存のワイルドカード文字をエスケープ文字として指定しないでください。

- アンダースコア (_) やパーセント記号 (%) をエスケープ文字として指定すると、その like 述部では特別な意味を失ってしまい、エスケープ文字としてしか使えなくなります。

- 左または右のカッコ ([と]) をエスケープ文字として指定すると、like 述部ではカッコの Transact-SQL 機能としての意味が無効になります。
- ハイフン (-) または脱字記号 (^) をエスケープ文字として指定すると、特別な意味を失い、エスケープ文字としてしか使用できなくなります。

アンダースコア、パーセント記号、開きカッコなどとは異なり、エスケープ文字は角カッコの中でも、その特別な意味を失いません。

エスケープ記号は、指定された like 述部の中だけで有効で、同じ文の他の like 述部では無効です。エスケープ記号のあとで有効な文字は、ワイルドカード文字 (_、%、[、]、[^]) とエスケープ文字そのものだけです。エスケープ文字はその直後の 1 文字だけに有効で、そのあとに続く文字には影響しません。

パターンに、たまたまエスケープ文字として指定した文字がリテラル文字として 2 文字ある場合は、その文字列ではエスケープ文字を 4 つ続けて使う必要があります。パターンがエスケープ文字によって 1 文字または 2 文字に分割されない場合は、Adaptive Server はエラー・メッセージを返します。表 4-9 は、like で escape 句を使用した例を示します。

表 *4-9: escape* 句の使用

like 述部	意味
like "5@%" escape "@"	5%
like "*_n" escape "*"	_n
like "%80@%%" escape "@"	80% を含む文字列
like "*_sql**%" escape "*"	_sql* を含む文字列
like "%#####_#%%" escape "#"	##_% を含んでいる文字列

datetime データでのワイルドカード文字の使用

datetime の値で like を使用すると、Adaptive Server は日付を標準の datetime フォーマットに変換し、次に varchar に変換します。標準の格納フォーマットには秒やミリ秒は含まれていないため、like とパターンを使用して秒やミリ秒を検索することはできません。

datetime のエントリにはさまざまな日付要素が含まれている可能性があるので、datetime 値を検索するときは like を使用することをおすすめします。たとえば、arrival_time カラムに、“9:20” という値と現在の日付を入力する場合は、この値は次の句では検索できません。

```
where arrival_time = '9:20'
```

これは、Adaptive Server が入力された内容を “Jan 1 1900 9:20AM” に変換するためです。しかし、次の句ではこの値が検出されます。しかし、次の句ではこの値が検出されます。

```
where arrival_time like '%9:20%'
```

第 5 章

予約語

予約語として知られるキーワードは、特別な意味を持つワードです。この章では、Transact-SQL と ANSI SQL のキーワードについて説明します。

この章では、次の項目について説明します。

Transact-SQL 予約語

表 5-1 に、Adaptive Server によって予約されているキーワード (SQL コマンドの構文の一部) を示します。これらの予約語は、データベース、テーブル、ルール、デフォルトなどのデータベース・オブジェクトの名前として使用することはできません。ただし、これらの予約語をローカル変数名、ストアド・プロシージャのパラメータ名として使用することはできます。

予約語が使用されている既存のオブジェクト名を検索するには、『リファレンス・マニュアル：プロシージャ』の sp_checkreswords を使用します。

表 *5-1: Transact-SQL* 予約語のリスト

	予約語
A	add、all、alter、and、any、arith_overflow、as、asc、at、authorization、avg
B	begin、between、break、browse、bulk、by
C	cascade、case、char_convert、check、checkpoint、close、clustered、coalesce、commit、compute、confirm、connect、constraint、continue、controlrow、convert、count、count_big、create、current、cursor
D	database、dbcc、deallocate、declare、decrypt、default、delete、desc、deterministic、disk distinct、drop、dummy、dump
E	else、encrypt、end、endtran、errlvl、errordata、errorexit、escape、except、exclusive、exec、execute、exists、exit、exp_row_size、external
F	fetch、fillfactor、for、foreign、from
G	goto、grant、group
H	having、holdlock
I	identity、identity_gap、identity_start、if、in、index、inout、insensitive、insert、install、intersect、into、is、isolation
J	jar、join
K	key、kill

	予約語
L	level、like、lineno、load、lock
M	materialized、max、max_rows_per_page、min、mirror、mirrorexit、modify
N	national、new、noholdlock、nonclustered、nonscrollable、non_sensitive、not、null、nullif、numeric_truncation **注意** “new” は Transact-SQL の予約語ではありませんが、今後予約語となる可能性があるため、使用を避ける (データベース・オブジェクトの名前などに使用しない) ことをおすすめします。“New” は、spt_values テーブルに含まれ、また sp_checkreswords では予約語として表示されるため、特殊な例です (他の予約語の詳細については、「ANSI SQL の予約語となる可能性のあるワード」(368 ページ) を参照してください)。
O	of、off、offsets、on、once、online、only、open、option、or、order、out、output、over
P	partition、perm、permanent、plan、prepare、primary、print、privileges、proc、procedure、processexit、proxy_table、public
Q	quiesce
R	raiserror、read、readpast、readtext、reconfigure、references、remove、reorg、replace、replication、reservepagegap、return、returns、revoke、role、rollback、rowcount、rows、rule
S	save、schema、scroll、scrollable、select、semi_sensitive、set、setuser、shared、shutdown、some、statistics、stringsize、stripe、sum、syb_identity、syb_restree、syb_terminate
T	table、temp、temporary、textsize、to、tracefile、tran、transaction、trigger、truncate、tsequal
U	union、unique、unpartition、update、use、user、user_option、using
V	values、varying、view
W	waitfor、when、where、while、with、work、writetext
X	xmlextract、xmlparse、xmltest、xmlvalidate

ANSI SQL 予約語

Adaptive Server は、初級レベル ANSI SQL の機能を備えています。完全実装の ANSI SQL は、コマンド構文として次の表に示すワードを含んでいます。識別子のアップグレードが複雑になることがあるため、ユーザの便を図ってこのキーワード・リストを作成しました。しかし、このリストを提供しても、Sybase がこれら ANSI SQL の全機能を今後のリリースで提供することを保証するものではありません。また、今後のリリースでは、このリストにはないキーワードが含まれる可能性もあります。

表 5-2 は、ANSI SQL のキーワードを示します (Transact-SQL の予約語ではありません)。

表 *5-2: ANSI SQL* 予約語のリスト

	予約語
A	absolute、action、allocate、are、assertion
B	bit、bit_length、both
C	cascaded、case、cast、catalog、char、char_length、character、character_length、coalesce、collate、collation、column、connection、constraints、corresponding、cross、current_date、current_time、current_timestamp、current_user
D	date、day、dec、decimal、deferrable、deferred、describe、descriptor、diagnostics、disconnect、domain
E	end-exec、exception、extract
F	false、first、float、found、full
G	get、global、go
H	hour
I	immediate、indicator、initially、inner、input、insensitive、int、integer、interval
J	join
L	language、last、leading、left、local、lower
M	match、minute、module、month
N	names、natural、nchar、next、no、nullif、numeric
O	octet_length、outer、output、overlaps
P	pad、partial、position、preserve、prior
R	real、relative、restrict、right
S	scroll、second、section、semi_sensitive、session_user、size、smallint、space、sql、sqlcode、sqlerror、sqlstate、substring、system_user
T	then、time、timestamp、timezone_hour、timezone_minute、trailing、translate、translation、trim、true
U	unknown、upper、usage
V	value、varchar
W	when、whenever、write、year
Z	zone

ANSI SQL の予約語となる可能性のあるワード

ISO/IEC 9075:1989 標準を使用する場合は、次のリストに示すワードの使用は避けてください。これらのワードは、ANSI SQL の予約語になる可能性があります。

表 *5-3: ANSI SQL* の予約語となる可能性のあるワードのリスト

	予約語
A	after、alias、async
B	before、boolean、breadth
C	call、completion、cycle
D	data、depth、dictionary
E	each、elseif、equals
G	general
I	ignore
L	leave、less、limit、loop
M	modify
N	new、none
O	object、oid、old、operation、operators、others
P	parameters、pendant、preorder、private、protected
R	recursive、ref、referencing、resignal、return、returns、routine、row
S	savepoint、search、sensitive、sequence、signal、similar、sqlexception、structure
T	test、there、type
U	under
V	variable、virtual、visible
W	wait、without

第 6 章

SQLSTATE コードとメッセージ

この章では、Adaptive Server の SQLSTATE ステータス・コードと各コードに関連付けられているメッセージについて説明します。

この章では、次の項目について説明します。

初級レベル ANSI SQL 準拠の場合、SQLSTATE コードが必要です。これらは、次の 2 種類の条件についての診断情報を提供します。

- 「警告」– ユーザに通知は必要だが、SQL 文の正常な実行の妨げとなるほど重大ではない状態。
- 「例外」– データベースに対する SQL 文の実行が無効となる状態。

個々の SQLSTATE コードは、2 文字クラスとそれに続く 3 文字サブクラスで構成されます。クラスはエラー・タイプについての一般的な情報を指定します。サブクラスは、より詳細な情報を指定します。

SQLSTATE コードは、前述の条件が検出されたときに表示されるメッセージと一緒に、sysmessages システム・テーブルに保管されます。Adaptive Server のエラー条件はすべての SQLSTATE コードと一致するわけではありません (ANSI SQL に準拠しているものだけです)。また、複数の Adaptive Server のエラー条件が、1 つの SQLSTATE の値と重複する場合もあります。

警告

現在、Adaptive Server では次の SQLSTATE 警告条件が 1 つだけ検出されます。表 6-1 にこの警告条件を示します。

表 *6-1: SQLSTATE* 警告

メッセージ	値	説明
警告：null 値は set 関数では使用できません。	01003	null 値をとる式で集合関数 (avg、max、min、sum、または count) を使用した場合。
Adaptive Server から返された文字またはバイナリ・データがトランケートされました。クライアント・アプリケーションは、結果カラムまたは出力パラメータとして、255 バイトを超えるデータをサポートしていません。	01004	文字、unichar、またはバイナリ・データが 255 バイトにトランケートされる場合。データには次のものがある。 • クライアントが WIDE TABLES プロパティをサポートしない select 文の結果。 • WIDE TABLES プロパティをサポートしないリモート Adaptive Server または Open Server 上の RPC へのパラメータ。

例外

Adaptive Server は、次に示すタイプの例外を検出します。

- 基本的な違反
- データ例外
- 整合性の制約違反
- 無効なカーソル状態
- 構文エラーおよびアクセス・ルール違反
- トランザクション・ロールバック
- with check option 違反

表 6-2 から表 6-8 では、例外条件について説明します。例外の各クラスは該当する表に示します。各表では、条件はメッセージ・テキストのアルファベット順にソートされています。

基本的な違反

相互関係による違反は、1 つのローだけを返すクエリが Embedded SQL™ アプリケーションに複数のローを返す場合に発生します。

表 *6-2:* 基本的な違反

メッセージ	値	説明
サブクエリが 2 個以上の値を返しました。サブクエリが、=、!=、<、<=、>、>= に続く場合、またはサブクエリが式で使用される場合は、無効です。	21000	次のような条件のときに発生する。 • スカラ・サブクエリ、またはロー・サブクエリが複数のローを返す場合。 • Embedded SQL の select into parameter_list クエリが複数のローを返す場合。

データ例外

次のような値が入力された場合、データ例外が発生します。

- データ型に対してエントリが長すぎる
- 不正なエスケープ・シーケンスを含んでいる
- その他のフォーマット・エラーを含んでいる

表 *6-3:* データ例外

メッセージ	値	説明
算術オーバフローが起こりました。	22003	次のような条件のときに発生する。 • 算術演算子または sum 関数を使い、真数値型が精度または位取りを失う場合。 • トランケート、丸め、または sum 関数によって、概数値型が精度または位取りを失う場合。
データの例外 − 文字列データが正しくトランケートされました。	22001	挿入または更新するデータに対して、char、unichar、univarchar、または varchar カラムが短すぎる場合、またはブランク以外の文字がトランケートされた場合。
ゼロによる除算が行われました。	22012	数値式が評価され除数の値が 0 の場合。
無効なエスケープ文字が検出されました。有効な文字を形成できるだけのバイト数がありません。	22019	指定されたパターンに一致する文字列の検索中に、複数文字からなるエスケープ・シーケンスが検出された場合。
パターン文字列が無効です。エスケープ文字に続く文字は、パーセント記号、アンダースコア、左大カッコ、右大カッコ、またはエスケープ文字でなければなりません。	22025	次のように検索文字列が特定のパターンと一致した場合。 • エスケープ文字の直後にパーセント記号、アンダースコア、またはエスケープ文字自身がない場合。 • エスケープ文字が 1 または 2 文字以外の部分文字列にパターンを区切る場合。

整合性の制約違反

insert、update、delete 文が、**主キー制約**、**外部キー制約**、**検査制約**、**一意性制約**、ユニーク・インデックスに違反していると、整合性の制約違反が発生します。

表 *6-4:* 整合性の制約違反

メッセージ	値	説明
オブジェクト *object_name*（ユニーク・インデックス *index_name* 付き）に、重複キーを持つローを挿入しようとしました。	23000	重複するローが一意性制約、またはユニーク・インデックスのあるテーブルに挿入された場合。
チェック制約違反が発生しました。データベース名 = *database_name*、テーブル名 = *table_name*、制約名 = *constraint_name*。	23000	update または delete がカラムの検査制約に違反した場合。
参照整合性の制約で従属する外部キーの制約違反があります。 データベース名 = *database_name*、 テーブル名 = *table_name*、制約名 = *constraint_name*。	23000	プライマリ・キー・テーブルの update または delete が外部キー制約に違反した場合。
外部キーの制約違反が発生しました。データベース名 = *database_name*、テーブル名 = *table_name*、制約名 = *constraint_name*。	23000	プライマリ・キー・テーブルに一致する値がない状態で外部キー・テーブルの insert または update を実行した場合。

無効なカーソル状態

無効なカーソル条件が発生する原因は次のとおりです。

- fetch が現在オープンしていないカーソルを使用する。
- update where current of または delete where current of が、修正または削除されたカーソル・ローに影響する。
- update where current of または delete where current of が、フェッチされていないカーソル・ローに影響する。

表 *6-5:* 無効なカーソル状態

メッセージ	値	説明
オープンしていないカーソル *cursor_name* を使用しようとしました。詳細は、システム・ストアド・プロシージャ sp_cursorinfo で確認してください。	24000	オープンしていないカーソルを使ってフェッチしようとしたか、commit 文または（明示的および暗黙的な）rollback 文によってクローズされたカーソルを使ってフェッチしようとした場合。カーソルを再度オープンして、fetch を繰り返して実行する必要がある。

メッセージ	値	説明
更新または削除によって現在のカーソルの位置が削除されたために、カーソル *cursor_name* が暗黙にクローズされました。カーソル・スキャンの位置は回復できませんでした。これは、複数のテーブルを参照するカーソルで起こります。	24000	複数テーブルのカーソルのジョイン・カラムに削除または変更が加えられた場合。別の fetch を発行して、カーソルを再度配置する必要がある。
DELETE/UPDATE WHERE CURRENT OF、または通常の探索型の DELETE/UPDATE のために、カーソル *cursor_name* の現在のスキャンの位置が削除されています。UPDATE または DELETE WHERE CURRENT OF を実行する前に、新しく FETCH を実行しなければなりません。	24000	削除または変更が加えられた現在のカーソル位置で update/delete where current of を発行した場合。update/delete where current of を再実行する前に別の fetch を発行する必要がある。
カーソル *cursor_name* がローに位置付けされていないため、UPDATE/DELETE WHERE CURRENT OF が実行できませんでした。	24000	次のような状態のカーソル上で update/delete where current of を発行した場合。 • まだローをフェッチしていない場合。 • 結果セットの最後に達した後にさらにローをフェッチした場合。

構文エラーおよびアクセス・ルール違反

SQL 文内に終了していないコメント、Adaptive Server がサポートしていない暗黙的なデータ型の変換、または他の不正な構文がある場合に、構文エラーが生成されます。

存在しないオブジェクトまたは適正なパーミッションを得ていないオブジェクトにアクセスしようとすると、アクセス・ルール違反になります。

表 *6-6:* 構文エラーおよびアクセス・ルール違反

メッセージ	値	説明
オブジェクト *object_name*、データベース *database_name*、所有者 *owner_name* について *command* パーミッションが拒否されました。	42000	適正なパーミッションを持たないユーザがオブジェクトにアクセスしようとした場合。
データ型 '*datatype*' から '*datatype*' への暗黙の変換はできません。CONVERT 関数を使用して、このクエリを実行してください。	42000	あるデータ型を、Adaptive Server が暗黙に変換することができない別のデータ型にユーザが変換しようとした場合。
object_name の近くに、構文エラーがあります。	42000	指定したオブジェクトの近くで不正な SQL 構文が検出された場合。
insert エラー。カラム名または与えられた値の数がテーブルの定義と一致しません。	42000	挿入時に無効なカラム名を使用した場合、または値の数を間違えて挿入した場合。
コメント終了マーク '*/' がありません。	42000	/* 開始デリミタで始まるコメントに、*/ 終了デリミタがない場合。

メッセージ	値	説明
object_name が見つかりません。owner.objectname を指定するか、sp_help を使用してオブジェクトの存在を確認してください (sp_help ではその他のオブジェクトも表示されます)。	42000	自分が所有していないオブジェクトを参照した場合。別のユーザが所有しているオブジェクトを参照する場合は、オブジェクト名をその所有者名で修飾すること。
object_name に与えられるサイズ (*size*) は、最大値を超えています。許される最大のサイズは *size* です。	42000	次のような条件のときに発生する。 • テーブル定義のすべてのカラムの合計サイズがロー・サイズの許容値を超える場合。 • 単一のカラムまたはパラメータのサイズがそのデータ型の許容値を超える場合。

トランザクション・ロールバック

独立性レベルが 3 に設定されても、Adaptive Server が同時トランザクションの直列化を保証できない場合にトランザクションのロールバックが発生します。通常、この例外は、ディスク・クラッシュやオフライン・ディスクなどのシステム障害が原因で発生します。

表 *6-7:* トランザクション・ロールバック

メッセージ	値	説明
サーバ・コマンド (プロセス id #*process_id*) がデッドロックを検出しました。コマンドを再実行してください。	40001	Adaptive Server が、複数の同時トランザクションの直列化を保証できないことを検出した場合。

with check option 違反

ビューによって挿入または更新するデータをビューで表示することができない場合に、この例外クラスが発生します。

表 *6-8: with check option* 違反

メッセージ	値	説明
対象のビューは、WITH CHECK OPTION によって作成されたか、WITH CHECK OPTION によって作成された他のビュー上に定義されているために、挿入または更新が失敗しました。コマンドからの 1 個以上の結果ローが、WITH CHECK OPTION 制約での条件を満たしません。	44000	ビュー、またはそのビューに依存しているビューが with check option 句で作成された場合。

索引

記号

数字

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W

Y

あ

い

う

え

お

か

き

く

け

こ

さ

し

す

せ

そ

た

ち

つ

て

と

な

に

ね

は

ひ

ふ

へ

や

ゆ

よ

ら

り

る

れ

ろ

わ

ん