SYBASE®
An SAP® Company

リファレンス：ビルディング・ブロック、テーブル、およびプロシージャ

Sybase IQ 15.4

ドキュメント ID： DC01135-01-1540-01
改訂： 2011 年 11 月

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

目次

対象読者

このマニュアルは、SQL 文、言語要素、データ型、関数、システム・プロシージャ、システム・テーブルに関するリファレンス資料を必要とする Sybase® IQ ユーザを対象としています。

他のマニュアルでは、特定のタスクを実行する方法を詳しく説明します。このマニュアルを使用して、SQL 構文、パラメータ、オプションの情報を確認してください。コマンドライン・ユーティリティの起動パラメータについては、『ユーティリティ・ガイド』を参照してください。

ファイル・ロケーションとインストール設定

この項では、Sybase IQ で使用するインストール環境とオペレーティング・システムの設定について説明します。

これらの設定は、オペレーティング・システムに応じて、環境変数、初期化ファイル・エントリ、またはレジストリ・エントリとして保管できます。

インストール・ディレクトリ構造

Sybase IQ をインストールすると、いくつかのディレクトリが作成されます。 この項ではディレクトリ構造について説明します。

作成されるディレクトリは、インストール時に選択したオプションと Sybase ディレクトリ内の既存のディレクトリ (UNIX では *$SYBASE* によって定義されたディレクトリ、Windows では *%SYBASE%* によって定義されたディレクトリ) によって決まります。

デフォルトでは、Sybase IQ ソフトウェアは Sybase ディレクトリの固有のサブディレクトリにインストールされます。 このサブディレクトリは、インストール・ディレクトリと呼ばれます。 Sybase IQ とともに提供される他のツールも同様に、Sybase ディレクトリ以下の固有のサブディレクトリにインストールされます。 この項では、Sybase IQ のサブディレクトリ構造についてのみ説明します。

Sybase IQ のデフォルトのディレクトリは `IQ-15_3` です。`IQ-15_3` の場所は、Sybase IQ をどこにインストールするかで変わります。 `IQ-15_3` ディレクトリは、UNIX の環境変数 *$IQDIR15* または Windows の環境変数 *%IQDIR15%* によっても参照されます。

Sybase IQ ディレクトリ内には、多くのディレクトリとファイルがあります。

- **デモ・ディレクトリ (`%ALLUSERSPROFILE%¥SybaseIQ¥demo`)** – iqdemo データベースの構築に必要なツールが格納されています。 iqdemo データベース・ファイルとして、`iqdemo.db`、`iqdemo.iq`、 `iqdemo.iqmsg`、`iqdemo.iqtmp` があります。デモ・データベース自体は Sybase IQ に同梱されていません。
- **サブディレクトリ `demo/adata`** – 15.x の iqdemo データベースを作成するための 15.x データがあります。 サブディレクトリ `/demo/demodata` には、IQ 12.7 asiqdemo データベースと同じスキーマ・レイアウトとデータを持つ iqdemo データベースを作成するための Sybase IQ 12.7 データがあります。 15.x の iqdemo データベースを作成するには、Windows では `/demo/mkiqdemo.bat`

を使用し、UNIX では demo/mkiqdemo.sh を使用します。 iqdemo データベースは、サポート・センタに問題を伝えるために使用できます。

- **スクリプト・ディレクトリ (IQ-15_3/scripts)** – 例として使用するスクリプトやストアド・プロシージャのようなカタログ・オブジェクトの作成時に使用されるスクリプトがあります。これらのスクリプトは編集しないでください。これらのスクリプトを編集、削除、または移動すると、サーバが正しく動作しなくなります。
- **サンプル・ディレクトリ** – samples ディレクトリには、SQL サンプルとユーザ定義関数 (UDF) サンプルがあります。%ALLUSERSPROFILE%/SybaseIQ/samples/sqlanywhere には、SQL サンプルのディレクトリがあります。sqlanywhere/c directory には、SQL Anywhere® で ESQL (埋め込み SQL) と C を使用する方法を示した C++ のサンプルがあります。 SQL Anywhere と Sybase IQ は同じコードを共有しているので、これらのサンプルを Sybase IQ 用に変更できます。%ALLUSERSPROFILES%/SybaseIQ/samples/udf ディレクトリには、サンプルの C++ スカラと集合 UDF があります。
- **実行プログラム・ディレクトリ** – 実行プログラム、ライブラリ、ヘルプ・ファイルなどがあります。UNIX では、実行可能ファイル用サブディレクトリとして、IQ-15_3 のサブディレクトリ bin64、lib64、logfiles、res、tix があります。Windows では、実行可能ファイル用サブディレクトリとして、IQ-15_3 のサブディレクトリ h、install、java、bin32 があります。

Sybase IQ によるファイルの配置

Sybase IQ は、起動時と実行時にいくつかのファイルを検出してアクセスする必要があります。 システム上には、同じ名前を持つディレクトリまたはファイルが複数存在する可能性があります。

適切なファイルを使用するには、Sybase IQ によるファイルの検出方法を理解しておくことが重要です。 次に、アクセスする必要があるファイル・タイプの一部を示します。

- ライブラリ ― 製品ライブラリまたはシステム・ライブラリが含まれています。 ファイル名拡張子は、UNIX では .so.nnn または .so、Windows では .dll または .lib です。 これらのファイルは、Sybase IQ の実行に必要です。不適切な DLL が見つかった場合、バージョン不一致エラーが発生する可能性があります。 たとえば、ライブラリ・ファイルは、UNIX では $IQDIR15/lib64 または $SYBASE/$SYBASE_OCS/lib64 にあり、Windows では %IQDIR15%¥bin32 または %SYBASE¥%SYBASE_OCS¥dll にあります。 **start_iq** には、通常のライブラリ・ディレクトリよりも前に usrlib が含まれ

ているため、空のディレクトリ `$IQDIR15/usrlib` では、デフォルトのライブラリをカスタム・ライブラリおよびパッチで置き換えることができます。Sybase IQ は、Adaptive Server Enterprise と SQL Anywhere の両方のライブラリを使用します。 これらの製品のいずれかがすでにシステムにインストールされている場合は、混乱を避けるため、インストールされているディレクトリを確認しておいてください。

- インタフェース・ファイル – Sybase IQ の実行に必要です。 インタフェース・ファイルの例としては、UNIX の `.odbc.ini` と `utility_db.ini` や、Windows の `util_db.ini` があります。これらのファイルの詳細については、『システム管理ガイド：第 1 巻』と『インストールおよび設定ガイド』を参照してください。
- 設定ファイル — 接続パラメータを指定するために使用します。たとえば、設定ファイルは、Windows では `default.cfg`、または `iqdemo.cfg` です。
- データベース・ファイル — データとメタデータを格納します。たとえば、`iqdemo.db`、`iqdemo.iq`、`iqdemo.iqmsg`、`iqdemo.iqtmp` などです。
- ログ・ファイル — サーバ上の現在のセッションおよび接続されているデータベースに関する情報を格納します。たとえば、サーバ・ログには `%ALLUSERSPROFILE%¥SybaseIQ¥logfiles¥yourservername.0006.srvlog` などの名前が付けられることもあります。データベースに接続すると、データベース・ログ (`%ALLUSERSPROFILE%¥SybaseIQ¥demo¥iqdemo.log` など) が作成されます。これらのファイルの詳細については、『インストールおよび設定ガイド』を参照してください。
- プロダクト・スクリプト — データベースを作成、移植、アップグレードする方法が示されたサンプル・ファイルです。
- ユーザ・ファイル — LOAD コマンドで使用されるフラット・ファイルおよび Interactive SQL などのツールで使用される SQL スクリプトがあります。
- テンポラリ・ファイル — クエリのソートの実行などの操作に関するテンポラリ情報を格納するために、Sybase IQ が作成します。

いくつかのファイル名は、SQL 文で指定され、実行時に検出される必要があります。ファイル名を使用する SQL 文の例は次のとおりです。

- **INSTALL** 文 — Java クラスを保持するファイルの名前。
- **LOAD TABLE** 文 — データのロード元となるファイルの名前。
- **CREATE DATABASE** 文 — この文およびファイルを作成できる同様の文には、ファイル名が必要です。

Sybase IQ は、簡単なアルゴリズムを使用してファイルを検出する場合があります。 より広範囲の検索が実行される場合もあります。

簡単なファイル検索

多くの SQL 文 (**LOAD TABLE** や **CREATE DATABASE** など) では、ファイル名はデータベース・サーバの現在の作業ディレクトリ (サーバが起動されたディレクトリ) に対して相対的なものとして解釈されます。

また、データベース・サーバを起動し、データベース・ファイル名 (DBF パラメータ) を指定すると、サーバが起動されたディレクトリに対する相対パスとして解釈されます。

広範囲なファイル検索

データベース・サーバおよび管理ユーティリティを含む Sybase IQ プログラムは、DLL や共有ライブラリなど必要なファイルを広範囲に検索します。 この場合、Sybase IQ プログラムは次の順序でファイルを検索します。

1. *実行ディレクトリ* ― 実行プログラムのあるディレクトリです。また、実行プログラム・ディレクトリに対して次の相対パスを持つディレクトリです。
   - 実行プログラムの親ディレクトリ
   - `scripts` という親ディレクトリの子
2. *現在の作業ディレクトリ* ― 開始したプログラムには、現在の作業ディレクトリ (そのプログラムが起動したディレクトリ) があります。必要なファイルは、このディレクトリ内で検索されます。
3. *LOCATION レジストリ・エントリ* ― Windows インストール環境では、Sybase IQ は LOCATION レジストリ・エントリを追加します。指定されたディレクトリに続けて、次のディレクトリが検索されます。
   - `scripts` という名前の子ディレクトリ
   - オペレーティング・システム名 (`bin32`、`bin` など)の付いた子ディレクトリ
4. *システム固有ディレクトリ* ― 一般的なオペレーティング・システム・ファイルが格納されているディレクトリが含まれます。たとえば、Windows では、Windows ディレクトリと Windows¥system ディレクトリがこれに含まれます。
5. *CLASSPATH ディレクトリ* ― Java ファイルの場合は、CLASSPATH 環境変数に指定されているディレクトリでファイルが検索されます。
6. *PATH ディレクトリ* ― システム・パスおよびユーザのパスに含まれるディレクトリでファイルが検索されます。
7. *LIBRARY PATH ディレクトリ* ― `LIBPATH` 環境変数に指定されているディレクトリで共有ライブラリが検索されます。

環境変数

Sybase IQ では、一連の環境変数を使用してさまざまなタイプの情報を格納します。必ずしもすべての変数を指定する必要はありません。

Windows での環境変数の設定

Windows プラットフォームでは、すべての環境変数がインストール・プログラムによって自動的に設定されるため、変更は不要です。ただし、オプションの変数を設定したり、デフォルト値から変更する必要がある場合は、次の手順に従ってください。

1. デスクトップで、[マイ コンピュータ] を右クリックし、サブメニューから [プロパティ] を選択します。
2. [詳細設定] タブをクリックします。
3. [環境変数] ボタンをクリックします。

   [環境変数] ダイアログが開きます。

   a) 環境変数がまだない場合は、[新規] をクリックして変数名とその値を入力し、[OK] をクリックします。
   b) 変数がある場合、[システム環境変数] または [ユーザ環境変数] リストから変数を選択し、[編集] をクリックし、[値] フィールドを必要に応じて変更します 次に、[OK] をクリックして設定を保存します。

   注意： ユーザ環境変数とシステム環境変数については、Microsoft Windows のマニュアルを参照してください。

UNIX 環境ソース・ファイルの実行

UNIX では、環境ソース・ファイルによって必須の環境変数を設定します。

すべての必須環境変数を設定するには、次のコマンドを実行します。

1. Bourne/Korn シェルでは次のように入力します。

   ```
   . $SYBASE/IQ-15_3/IQ-15_3.sh
   ```

2. C シェルでは、次のように入力します。

   ```
   source $SYBASE/IQ-15_3/IQ-15_3.csh;
   rehash
   ```

UNIX での環境変数の設定

UNIX プラットフォームでは、環境ソース・ファイルを実行して必須環境変数を設定します。ただし、オプションの変数を設定したり、デフォルト値から変更する必要がある場合は、次の手順に従ってください。

1. 環境変数の設定をチェックするには、次のコマンドを使用します。
   ```
   echo $variable-name
   ```
   たとえば、$SYBASE 変数の設定を確認するには、次のコマンドを実行します。
   ```
   % echo $SYBASE
   ```
   ```
   /server1/users/test/sybase
   ```
2. 起動ファイル (`.cshrc`、`.shrc`、`.login`)の 1 つに、変数を設定する行を追加します。
   一部のシェル (sh、bash、ksh など) では、次のような行になります。
   ```
   VARIABLE=value;export VARIABLE
   ```
   その他のシェル (csh、tsch など) では、次のような行になります。
   ```
   setenv VARIABLE "
                    value"
   ```

IQCHARSET 環境変数

IQCHARSET は、デフォルトの文字セットを設定します。

Charset は文字セットの名前です。たとえば、**IQCHARSET=cp1252** と指定すると、デフォルトの文字セットが cp1252 に設定されます。

次の値セットのうち、最初の値がデフォルトの文字セットを決定します。

- IQCHARSET 環境変数
- OS をクエリ

文字セット情報が指定されていない場合は、UNIX では `iso_1`、その他の場合は `cp850` を使用します。

設定値

IQCHARSET=*charset*

IQDIR15 環境変数

IQDIR15 は、Sybase IQ ディレクトリの場所を示すとともに、そのディレクトリ内の他のディレクトリおよびファイルの場所です。

- `$IQDIR15/bin[64]/util_db.ini` には、ユーティリティ・データベース `utility_db` のログイン ID とパスワードが含まれています。インストール・プログラムでは、ログイン ID とパスワードのデフォルトの値 (それぞれ、"DBA" と "sql") を変更できます。
- `$IQDIR15/logfiles` は、サーバ・ログとバックアップ／リストア・ログ (バックアップ履歴ファイル) のデフォルト・ロケーションです。IQLOGDIR15 環境変数を設定することで、このデフォルト設定を変更できます。
- `$IQDIR15/demo` は、データベース・ファイル `iqdemo` の場所です。

設定値

```
IQDIR15 = ${SYBASE}/IQ-15_3
```

オペレーティング・システム
必須。環境ソース・ファイルまたはインストール・プログラムによって設定されます。このデフォルト設定は、Windows では変更可能です。

IQPORT 環境変数

IQPORT は、Sybase IQ プラグインとエージェントとの間の通信に使用される、Sybase IQ Agent のポート番号のデフォルト値を上書きします。

注意： エージェントが起動すると、ポート番号を変更できません。

1099 は、特定のポートでエージェント・プロセスを検索するときに使用される、プラグインのデフォルト値です。プラグインは、このポートでエージェントを検出できない場合、正しいポート番号を指定するよう求めるメッセージを表示します。

設定値

```
IQPORT = 5556
```

オペレーティング・システム
オプション。環境ソース・ファイル内にユーザが IQPORT を指定していない場合、ポート番号はデフォルトで 1099 になります。このデフォルト値は、プラグインの起動前であれば変更できます。この変数は、Sybase Central の開始時に **-DIQPORT** 引数を **scjview** に指定することで設定できます。次に例を示します。

```
scjview -DIQPORT=3345
```

IQLANG 環境変数

IQLANG はデフォルト言語を設定します。

Language_code は、2 文字で表された言語名です。たとえば、**IQLANG=DE** の場合、デフォルトの言語がドイツ語に設定されます。

次の値セットのうち、最初の値がデフォルトの言語を決定します。

- IQLANG 環境変数
- インストーラによって設定されたレジストリ (Windows のみ)
- OS をクエリ

言語情報が設定されていない場合のデフォルト値は英語です。

設定値

IQLANG=*language_code*

オペレーティング・システム
オプション。ただし英語以外の環境では推奨。

言語ラベル値のリスト

有効な言語ラベルの値と対応する ISO 639 言語コードは次のとおりです。
IQ_LANG 環境変数に 2 文字の ISO_639 言語コードを設定してください。

言語	ISO_639 言語コード	言語ラベル	代替ラベル
アラビア語	AR	arabic	該当なし
チェコ語	CS	czech	該当なし
デンマーク語	DA	danish	該当なし
オランダ語	NL	dutch	該当なし
英語	EN	us_english	english
フィンランド語	FI	finnish	該当なし
フランス語	FR	french	該当なし
ドイツ語	DE	german	該当なし
ギリシャ語	EL	greek	該当なし
ヘブライ語	HE	hebrew	該当なし
ハンガリー語	HU	hungarian	該当なし

言語	ISO_639 言語コード	言語ラベル	代替ラベル
イタリア語	IT	italian	該当なし
日本語	JA	japanese	該当なし
韓国語	KO	korean	該当なし
リトアニア語	LT	lithuanian	該当なし
ノルウェー語	NO	norwegian	norweg
ポーランド語	PL	polish	該当なし
ポルトガル語	PT	portuguese	portugue
ロシア語	RU	ロシア語	該当なし
中国語 (簡体字)	ZH	chinese	simpchin
スペイン語	ES	spanish	該当なし
スウェーデン語	SV	swedish	該当なし
タイ語	TH	thai	該当なし
中国語 (繁体字)	TW	tchinese	tradchin
トルコ語	TR	トルコ語	該当なし
ウクライナ語	UK	ukrainian	該当なし

IQLOGDIR15 環境変数

IQLOGDIR15 環境変数は、さまざまなログ・ファイルの場所を定義します。
IQLOGDIR15 は、インストール・プログラムでは設定されません。

- サーバ・ログは、$IQLOGDIR15 で指定されたディレクトリ内の `servername.nnnn.srvlog` ファイル (nnnn は、サーバが起動された回数) 内にあります。

IQLOGDIR15 が有効な書き込み可能ディレクトリとして設定されていない場合、**start_iq** を含む大部分のユーティリティは、すべてのサーバ・ログに対してデフォルトの場所 `$IQDIR15/logfiles` を使用します。

設定値

```
IQLOGDIR15 = path
```

オペレーティング・システム
オプション。

IQTIMEOUT 環境変数

IQTIMEOUT はデフォルトの待機時間 (5 分) を変更します。

引数 *nnn* によって、Sybase IQ Agent が待機する時間 (分単位) を指定します。 例を示します。

- 待機時間を 45 分にするには、次のように指定します (Korn シェルまたは Bourne シェルの場合)。
  ```
  IQTIMEOUT=45 export IQTIMEOUT
  ```
- 待機時間を 1 時間にするには、次のように指定します (C シェルの場合)。
  ```
  setenv IQTIMEOUT 60
  ```

Sybase IQ Agent は、処理の完了を無期限で待機します。非常に大きなカタログ・ストアを持つマルチプレックスのクエリ・サーバを作成したり同期したりするときには、待機時間を設定することをおすすめします。 カタログ・ストアが大きい場合、同期の dbbackup 部分に必要な時間が増大します。待機時間を増やすことによって、より長時間の同期に対応できます。

注意： IQTIMEOUT を設定してから、Sybase IQ Agent を起動してください。

設定値

IQTIMEOUT = *nnn*

オペレーティング・システム
オプション。ただしマルチプレックス環境では推奨。詳細については、『Sybase IQ Multiplex の使用』を参照してください。

IQTMP15 環境変数

IQTMP15 環境変数は、インストール・プログラムでは設定されません。 IQTMP15 は、テンポラリ・ファイルが格納されるディレクトリを示すために Sybase IQ で使用されます。

IQTMP15 環境変数は、NFS (Network File System) を使用しているローカル・ディレクトリを示す必要があります。これによって、IQTMP15 ディレクトリは、クライアント接続が終了したときに不必要となったディレクトリやファイルを消去できます。クライアント接続が行われるたびに、ディレクトリやファイルがテンポラリ・ディレクトリ内に作成されます。これらは、接続中にだけ必要です。サーバに接続するすべてのユーザが、このディレクトリに対する書き込みパーミッションを持っている必要があります。

注意： IQTMP15 変数によって定義されるロケーションにあるテンポラリ・ファイルは、クライアントとサーバによって使用されます。この変数は、テンポラリ IQ

ストアのデフォルト・ロケーションを制御しません。テンポラリ IQ ストアのデフォルト・ロケーションは、**CREATE DATABASE** 文によって制御します。『リファレンス：文とオプション』を参照してください。

警告！ IQTMP15 を $SYBASE や $IQDIR15 に設定しないでください。

IQTMP15 変数が明示的に設定されていない場合、Sybase IQ Agent は、IQTMP15 変数を UNIX ディレクトリ `/tmp` のサブディレクトリに設定します。

マシン上で複数のデータベース・サーバが実行されている場合、それぞれのサーバと関連するローカル・クライアントは、競合を避けるために別々のテンポラリ・ディレクトリを必要とします。 接続に使用するポート番号またはエンジン番号を指定しない場合、Sybase IQ は、ネットワーク接続の代わりに共有メモリ接続を使用します。

共有メモリを使用したときに競合を避けるには、次の手順に従います。

- 各サーバに専用のテンポラリ・ディレクトリを作成します。両方の環境に明示的に IQTMP15 環境変数を設定して、各ローカル・クライアントがサーバと同じテンポラリ・ディレクトリを使用するようにします。
- 各サーバの `.odbc.ini` ファイル (UNIX の場合) にデータ・ソース名を作成し、詳細な接続情報を指定します。『インストールおよび設定ガイド』を参照してください。
- デフォルトでなく、明示的にパラメータを指定する接続文字列を使用します。
- 次のコマンドを実行して、接続を確認します。

```
SELECT "database name is" = db_name(), "servername_is" =
@@servername
```

設定値

```
IQTMP15 = temp_directory
```

オペレーティング・システム
UNIX の場合、オプションです。Windows プラットフォームでは使用されません。

JAVA_HOME 環境変数

bin/java を含むディレクトリを指す JRE ホームを定義します。

Java VM のロケーションが $SYBASE_JRE6_32、$SYBASE_JRE6_64、または $SYBASE_JRE5_64 環境変数で設定されていない場合に使用されます。

`JAVA_HOME` は、通常、VM のインストール時に作成されます。

UNIX では、`SYBASE.csh` (C シェル) または `SYBASE.sh` (Bourne または Korn シェル) 環境ソース・ファイルを実行することによって、IQ エンジンの正しい JRE を

検索し、開始します。JAVA_HOME で指定されている Java VM のロケーションは、SYBASE.csh または SYBASE.sh によって返されるロケーションよりも優先されます。JAVA_HOME またはスクリプト(SYBASE.csh または SYBASE.sh) のいずれでも Java VM を見つけられなかった場合、IQ は Java VM をロードしません。

設定

```
JAVA_HOME = Sybase/shared/JRE<version>
```

オペレーティング・システム
必須。

LIBPATH 環境変数と LD_LIBRARY_PATH 環境変数

LIBPATH または LD_LIBRARY_PATH は、Sybase IQ 共有ライブラリが配置されているディレクトリを指定します。

UNIX では、環境ソース・ファイルを実行することによってライブラリ・パス変数を設定します。

設定

```
For AIX:
LIBPATH = installation_path/lib
```

```
For all other UNIX/LINUX platforms:
LD_LIBRARY_PATH = installation_path/lib
```

オペレーティング・システム
必須。変数名はプラットフォームに依存します。UNIX のみ。

PATH 環境変数

PATH は Sybase IQ 実行プログラムの配置ディレクトリを含んでいる、オペレーティング・システムの必須変数です。

Windows では、インストール・プログラムが PATH を変更します。UNIX では、環境ソース・ファイルを実行して必要なディレクトリを含めます。

Windows では、PATH は LIBRARY_PATH 変数の代わりになります。したがって、実行プログラムと DLL の場所は PATH 変数を使って参照できます。

設定値

```
PATH = installation_path
```

オペレーティング・システム
必須。

SQLCONNECT 環境変数

SQLCONNECT は、データベース・サーバに接続するときに、Interactive SQL、**dbinfo**、**dbstop** などのデータベース管理ユーティリティが使用する接続パラメータを指定します。

SQLCONNECT はオプションの環境変数で、インストール・プログラムでは設定を行いません。

この文字列は、**parameter** = *value* 形式のパラメータ設定をセミコロンで区切ったリストで指定します。

シャープ記号 "#" は、等号 (=) の代わりに使用します。SQLCONNECT 環境変数内に接続パラメータ文字列を設定する場合は、シャープ記号を使用してください。環境変数設定の中で "=" を使用すると構文エラーになります。"=" 記号は、Windows でのみ使用できます。

注意： 接続パラメータをコマンド・ラインではなく SQLCONNECT 変数で指定することで、UNIX システムのセキュリティが向上します。これにより、ユーザは **ps -ef** で他のユーザのパスワードを表示できなくなります。これは、Interactive SQL を実行するとき、またはその他のユーティリティをクワイエット・モードで実行するときに特に有用です。接続パラメータをコマンド・ラインではなく SQLCONNECT 変数で指定することでセキュリティは向上しますが、それで完全になるわけではないことに注意してください。パスワードはプレーン・テキストであるため、悪意を持って操作すれば、環境コンテキストからパスワードを抽出することが可能です。詳細については、『システム管理ガイド：第 1 巻』の「接続パラメータと通信パラメータ」>「接続パラメータ」を参照してください。

設定

```
SQLCONNECT = parameter#value ; ...
```

オペレーティング・システム
オプション。

SYBASE 環境変数

SYBASE 変数は、Open Client や Open Server などの Sybase アプリケーションの場所を指定します。

Sybase IQ を UNIX システムにインストールする前に SYBASE 変数を設定する必要があります。 この環境変数は、UNIX システムで Sybase Central を使用する場合に必要です。

設定値

```
SYBASE = path
```

オペレーティング・システム
必須。

$SYBASE_JRE6_32、$SYBASE_JRE6_64、$SYBASE_JRE5_64 環境変数

この変数は、Sybase IQ の Sybase Central プラグインで使用する Java Runtime Environment の場所を指定します。

Windows および UNIX の場合、この環境変数は $SYBASE_JRE6_32 または $SYBASE_JRE6_64 です。AIX/LinuxIBM の場合、この環境変数は $SYBASE_JRE5_64 です。

UNIX では、`SYBASE.csh` (C シェル) または `SYBASE.sh` (Bourne または Korn シェル) 環境ソース・ファイルを実行することによって、正しい JRE を検索し、特定します。`JAVA_HOME` が優先されます。Windows では、インストール・プログラムが Open Client Software Developer's Kit をインストールするときにこの変数を設定します。

設定値

```
SYBASE_JRE= "${SYBASE}/shared/jre-6_0"
```

SYBASE_OCS 環境変数

SYBASE_OCS は、Open Client 製品のホーム・ディレクトリを指定します。

この変数は Window でのみ使用されます。Windows では、インストール・プログラムが Open Client/Server Software Developers Kit をインストールするときに SYBASE_OCS を設定します。

設定値
```
SYBASE_OCS = "OCS-15_3"
```

オペレーティング・システム
必須。

レジストリ・エントリ

Windows オペレーティング・システムでは、Sybase IQ は複数のレジストリ設定を使用します。

これらの設定はソフトウェアが行うため、通常、ユーザがレジストリにアクセスする必要はありません。ここでは、操作環境を変更するユーザのために、これらの設定について説明します。

警告！ Sybase は、レジストリを変更しないことを推奨しています。不正な変更を行うと、システムが損傷する場合があるためです。

カレント・ユーザ設定とローカル・マシン設定

Windows などのオペレーティング・システムには、2 つのレベルのシステム設定 (ユーザ設定とローカル・マシン設定) があります。

カレント・ユーザ設定は、特定のユーザがログオンしたときにだけ使用できる、ユーザ固有の設定です。ローカル・マシン設定は、ログオンしているユーザに関わりなく使用できる、マシン全体に関連する設定です。ローカル・マシン設定を行うには、マシンの管理者パーミッションを入手する必要があります。

Sybase IQ は、カレント・ユーザ設定とローカル・マシン設定の両方を許可します。 Windows では、これらの設定はそれぞれ HKEY_CURRENT_USER レジストリと HKEY_LOCAL_MACHINE レジストリに格納されます。

Sybase IQ では、設定の対象をカレント・ユーザのみにするか、ローカル・マシン・レベルにするかを選択できます。

設定がカレント・ユーザ設定とローカル・マシン設定の両方で行われている場合、カレント・ユーザ設定が優先されます。

Windows で Sybase IQ プログラムをサービスとして実行する場合は、*ローカル・マシン・レベル*で設定されていることを確認してください。

サービスは、マシンをログオフしても、マシンを完全に停止しないかぎり、特別なアカウントのもとで実行し続けます。サービスは、個々のアカウントから独立して実行可能であり、ローカル・マシン設定にアクセスする必要があります。

一般的に、Sybase はローカル・マシン設定を使用することを推奨しています。

レジストリ構造

Windows では、レジストリ・エディタを使用してレジストリに直接アクセスできます。

注意： 読み込み専用モードは、間違ってレジストリ・データを変更するのを防ぐことができます。読み込み専用モードにするには、レジストリ・エディタを起動して [編集] - [アクセス許可] を選択し、読み取りパーミッションをオンにします。

Sybase IQ レジストリ・エントリは、HKEY_LOCAL_MACHINE キーの次のロケーションにあります。

```
SOFTWARE
  Sybase
    IQ 15.4
```

レジストリ・エディタの起動

レジストリ・エディタを起動して、Windows レジストリにアクセスします。

1. [スタート] > [ファイル名を指定して実行]を選びます。
2. [名前] ボックスに、次のように入力します。

   ```
   regedt32
   ```

インストール時のレジストリ設定

インストール・プログラムは、Sybase レジストリ内でレジストリ設定を自動的に行います。

- Location ― Sybase IQ レジストリでは、このエントリにインストール・ディレクトリのロケーションが保持されます。 例を示します。

  ```
  Location:REG_SZ:C:¥Program Files¥Sybase
  ¥IQ-15_4
  ```

Sybase IQ レジストリには、インストールされているアプリケーションの他のエントリが含まれています。 Sybase Central レジストリには、Sybase Central のバージョンとインストール済みのプラグインの情報が含まれます。

SQL 言語の要素

この項では、Sybase IQ SQL の言語要素と規則について詳細に説明します。

キーワード

各 SQL 文には 1 つまたは複数のキーワードが含まれています。

SQL 文のキーワードでは大文字と小文字を区別しませんが、この Sybase IQ マニュアルではキーワードを大文字で表記します。 たとえば、次の文では SELECT と FROM がキーワードになります。

```
SELECT *
FROM Employees
```

次の文は、上の文と同じです。

```
Select *
From Employees
select * from Employees
sELECT * FRoM Employees
```

予約語

SQL キーワードには予約語として定義されているものがあります。

SQL 文で予約語を識別子として使用するには、二重引用符で囲む必要があります。SQL 文で使用するキーワードのうち、すべてではありませんがその多くが予約語です。たとえば、次の構文を使用して、SELECT テーブルの内容を取り出します。

```
SELECT *
FROM "SELECT"
```

Embedded SQL を使用している場合、データベース・ライブラリ関数 **sql_needs_quotes** を使用すると、文字列に二重引用符が必要かどうかを判別できます。文字列が予約語であるか、通常は識別子に使用できない文字が文字列に含まれている場合は、文字列に二重引用符を付けます。

次の表は、Sybase IQ の SQL 予約語を示します。SQL はキーワードの大文字と小文字を区別しないため、この表の各語は大文字、小文字、またはその組み合わせで使用できます。 次の語のいずれかと、大文字／小文字の区別のみが違う文字列はすべて予約語となります。

表 1 : SQL 予約語

add	all	alter	and
any	as	asc	attach
backup	begin	between	bigint
binary	bit	bottom	break
by	call	capability	cascade
case	cast	char	char_convert
character	check	checkpoint	close
comment	commit	compressed	conflict
connect	constraint	contains	continue
convert	create	cross	cube
current	current_timestamp	current_user	cursor
date	datetimeoffset	dbspace	deallocate
dec	decimal	declare	default
delete	deleting	desc	detach
distinct	do	double	drop
dynamic	else	elseif	encrypted
end	endif	escape	except
exception	exec	execute	existing
exists	externlogin	fetch	first
float	for	force	foreign
forward	from	full	goto
grant	group	having	holdlock
identified	if	in	index
inner	inout	insensitive	insert
inserting	install	instead	int
integer	integrated	intersect	into
is	isolation	join	kerberos
key	lateral	left	like
limit	lock	login	long

match	membership	merge	message
mode	modify	natural	nchar
new	no	noholdlock	not
notify	null	numeric	nvarchar
of	off	on	open
openstring	openxml	option	options
or	order	others	out
outer	over	passthrough	precision
prepare	primary	print	privileges
proc	procedure	publication	raiserror
readtext	real	reference	references
refresh	release	remote	remove
rename	reorganize	resource	restore
restrict	return	revoke	right
rollback	rollup	save	savepoint
scroll	select	sensitive	session
set	setuser	share	smallint
some	spatial	sqlcode	sqlstate
start	stop	subtrans	subtransaction
synchronize	table	temporary	then
time	timestamp	tinyint	to
top	tran	treat	trigger
truncate	tsequal	unbounded	union
unique	uniqueidentifier	unknown	unsigned
update	updating	user	using
validate	values	varbinary	varbit
varchar	variable	varying	view
wait	waitfor	when	where
while	window	with	within
work	writetext	xml	

参照：
- 識別子 (22 ページ)
- quoted_identifier オプション (34 ページ)
- 検索条件内のサブクエリ (38 ページ)
- 式内のカラム名 (26 ページ)

識別子

識別子は、ユーザ ID、テーブル、カラムなどのデータベースのオブジェクト名を表します。

識別子の最大長は、128 バイトです。識別子は、次のいずれかの条件に当てはまる場合、二重引用符または角カッコで囲む必要があります。

- 識別子にスペースが含まれている。
- 識別子の最初の文字がアルファベット文字 (以下で説明) ではない。
- 識別子に予約語が含まれている。
- 識別子にアルファベット文字と数字以外の文字が含まれている。
  *アルファベット文字*に含まれるのは、アルファベット、アンダースコア文字 (_)、アットマーク (@)、シャープ記号 (#)、ドル記号 ($) です。データベースの照合順によって、どの文字をアルファベットまたは数字として扱うかが決まります。

アポストロフィ (一重引用符) を識別子の中で使用するには、一重引用符を 2 つ続けます。

`QUOTED_IDENTIFIER` データベース・オプションが OFF に設定されている場合は、SQL 文字列を二重引用符で区切る必要があります。この SQL 文字列は識別子としては使用できません。ただし、角カッコは `QUOTED_IDENTIFIER` の設定にかかわらず、どのような場合でも識別子の区切りとして使用できます。

`QUOTED_IDENTIFIER` オプションのデフォルトの設定は、Open Client および jConnect 接続では OFF、その他の場合は ON です。

制限事項
識別子には、次の制限事項があります。

- テーブル名に二重引用符を含めることはできない。
- ユーザ名に二重引用符またはセミコロン文字を含めることはできない (一重引用符は使用できる)。
- データベース名に二重引用符、一重引用符、セミコロン文字を含めることはできない。

- ユーザ名およびデータベース名の先頭または末尾にスペースを使用できない。
- **CREATE DATABASE...CASE IGNORE** または **CASE RESPECT** の指定にかかわらず、データベース名は常に大文字と小文字によって区別される。

サーバ名およびデータベース名のその他の制限事項については、『ユーティリティ・ガイド』で -n start_iq サーバ・オプションの説明を参照してください。

例
次の例は、いずれも有効な識別子です。

```
Surname
"Surname"
[Surname]
SomeBigName
"Client Number"
```

参照：
- 予約語 (19 ページ)
- quoted_identifier オプション (34 ページ)
- 検索条件内のサブクエリ (38 ページ)
- 式内のカラム名 (26 ページ)

文字列

文字列は、リテラル文字列またはデータ型 CHAR／VARCHAR を持つ式です。

リテラル文字列とは、アポストロフィ ('一重引用符') で囲まれ、任意のシーケンスで並べられた文字のことです。文字データ型の SQL 変数には、文字列を入れることができます。次に、リテラル文字列の簡単な例を示します。

組み込み関数やユーザ定義関数、またはそれ以外の多くの使用可能な式が CHAR データ型を持つことができます。

文字列中の特殊文字
文字列中の特殊文字を表すには、次のように、エスケープ・シーケンスを使用します。

- 文字列内でアポストロフィを表すには、アポストロフィを 2 つ続けて記述します。次に例を示します。
  ```
  'John''s database'
  ```
- 文字列内で改行を表すには、円記号に続けて n (¥n) を記述します。次に例を示します。
  ```
  'First line:¥nSecond line:'
  ```

- 円記号を表すには、円記号を続けて 2 つ (¥¥) 記述します。次に例を示します。
  ```
  'c:¥¥temp'
  ```
- 16 進のエスケープ・シーケンスは、印刷できるかどうかに関係なくあらゆる文字に使用できます。16 進のエスケープ・シーケンスは、円記号とその後に x と 2 桁の 16 進数がある文字列です (たとえば、¥x6d は、文字 m を表します)。次に例を示します。
  ```
  '¥x00¥x01¥x02¥x03'
  ```

互換性
Adaptive Server® Enterprise との互換性を保つには、`QUOTED_IDENTIFIER` データベース・オプションを OFF に設定します。この設定では、文字列の先頭と末尾のマークに二重引用符を使用することもできます。このオプションは、デフォルトでは ON になっています。

参照：
- 比較条件 (36 ページ)
- 式 (24 ページ)
- NULL 値 (71 ページ)
- 検索条件 (35 ページ)
- 3 値的論理 (51 ページ)
- SQL 演算子 (27 ページ)
- 検索条件内のサブクエリ (38 ページ)

式

式は、定数、カラム名、SQL 演算子、サブクエリなどの数種類の要素で構成されます。

構文
```
expression:
case-expression
| constant
| [ correlation-name. ] column-name [ java-ref ]
| - expression
| expression operator  expression
| ( expression )
| function-name ( expression, … )
| if-expression
| [ java-package-name. ] java-class-name java-ref
| ( subquery )
| variable-name [ java-ref ]
```

パラメータ

```
case-expression:
{ CASE  search-condition
... WHEN  expression
                THEN  expression [ , … ]
... [ ELSE  expression ]
END
| CASE
... WHEN  search-condition
                THEN  expression [ , … ]
... [ ELSE  expression ]
END }
```

```
constant:
{ integer | number | 'string' | special-constant | host-variable }
```

```
special-constant:
{ CURRENT { DATE | TIME | TIMESTAMP | USER }
| LAST USER
| NULL
| SQLCODE
| SQLSTATE }
```

```
if-expression:
IF  condition
... THEN  expression
... [ ELSE  expression ]
ENDIF
```

```
java-ref:
{ . field-name [ java-ref ]
| >> field-name [ java-ref ]
| . method-name ( [ expression ] [ , … ] ) [ java-ref ]
| >> method-name ( [ expression ] [ , … ] ) [ java-ref ] }
```

```
operator:
{ + | - | * | / | || | % }
```

使用法
すべての場所

権限
データベースに接続しておく必要があります。

関連する動作
なし。

互換性

- Adaptive Server Enterprise では、IF 条件はサポートされません。

- Adaptive Server Enterprise では、Java 式は現在サポートされません。
- その他の相違点については、以降の項で説明する式の各クラスの説明を参照してください。

参照：
- 比較条件 (36 ページ)
- NULL 値 (71 ページ)
- 検索条件 (35 ページ)
- 文字列 (23 ページ)
- 3 値的論理 (51 ページ)
- SQL 演算子 (27 ページ)
- 検索条件内のサブクエリ (38 ページ)
- 特別値 (58 ページ)
- CASE 文のサポート (734 ページ)

式内の定数

定数とは、数値または文字列です。

文字列定数は、アポストロフィで囲まれています。文字列内でアポストロフィを表すには、アポストロフィを 2 つ続けて記述します。

式内のカラム名

カラム名は識別子の 1 つであり、前に相関名が付くことがあります。通常、相関名はテーブル名です。

カラム名に英字、数字、アンダースコア以外の文字が使用されている場合は、二重引用符 ("") で囲んでください。以下は、有効なカラム名の例です。

```
Employees.Surname
City
"StartDate"
```

参照：
- 検索条件内のサブクエリ (38 ページ)
- 予約語 (19 ページ)
- 識別子 (22 ページ)

式のサブクエリ

サブクエリとは、カッコで囲まれた **SELECT** 文です。**SELECT** 文には、リスト項目を 1 つだけ指定できます。式として使用すると、通常、スカラ・サブクエリはゼロまたは 1 つの値しか返せません。

最上位レベルの **SELECT** の SELECT リスト、または **UPDATE** 文の **SET** 句の中では、カラム名を使用できる場所ではどこでもスカラ・サブクエリを使用できます。ただし、条件式 (**CASE**、**IF**、**NULLIF**、**ARGN**) 内でサブクエリを使用することはできません。

たとえば、次の文は、各部署の従業員数を、部署名でグループ化して返します。

```
SELECT DepartmentName, COUNT(*), 'out of',
(SELECT COUNT(*) FROM Employees)
FROM Departments AS D, Employees AS E
WHERE D.DepartmentID = E.DepartmentID
GROUP BY DepartmentName;
```

SQL 演算子

この項では、Sybase IQ で使用可能な算術演算子、文字列演算子、ビット処理演算子について説明します。

一般的な演算の優先度が適用されます。カッコ内の式が最初に評価され、続いて乗算と除算、最後に加算と減算が評価されます。その後、文字列の連結が行われます。

参照：

- 比較条件 (36 ページ)
- 式 (24 ページ)
- NULL 値 (71 ページ)
- 検索条件 (35 ページ)
- 文字列 (23 ページ)
- 3 値的論理 (51 ページ)

算術演算子

Sybase IQ では、以下の算術演算子を使用できます。

表 2 : 算術演算子

演算子	説明
expression + expression	加算。いずれかの式が NULL 値の場合、結果は NULL 値になります。

演算子	説明
expression - expression	減算。いずれかの式が NULL 値の場合、結果は NULL 値になります。
- expression	反転。式が NULL 値の場合、結果は NULL 値になります。
expression * expression	乗算。いずれかの式が NULL 値の場合、結果は NULL 値になります。
expression / expression	除算。いずれかの式が NULL 値か、または 2 番目の式が 0 の場合、結果は NULL 値になります。
expression % expression	モジュロによる、2 つの整数での除算の余り (整数) の算出。たとえば、21 を 11 で割った場合の余りは 10 なので、21 % 11 = 10 になります。

文字列演算子

Sybase IQ では、以下の文字列演算子を使用できます。

表 3 : 文字列演算子

演算子	説明
expression || expression	文字列連結 (2 本の縦線)。いずれかの文字列が NULL 値の場合、連結では空文字列として扱われます。
expression + expression	代替の文字列連結。+ 連結演算子を使用する場合は、暗黙的データ変換を行わないで、必ずオペランドを文字データ型に明示設定してください。

文字列連結演算子の結果データ型は、`LONG VARCHAR` です。**SELECT INTO** 文で文字列連結演算子を使用する場合は、非構造化データ分析オプションのライセンスを所有しているか、**CAST** を使用して **LEFT** を正しいデータ型とサイズに設定する必要があります。

標準と互換性

- SQL — ISO/ANSI SQL 準拠。|| 演算子は、ISO/ANSI SQL の文字列連結演算子です。
- Sybase — + 演算子は、Adaptive Server Enterprise でサポートされています。

参照：

- REVERSE 関数 [文字列] (299 ページ)

ビット処理演算子

Sybase IQ と Adaptive Server Enterprise では、次のビット処理演算子をすべての位取りされていない整数データ型で使用できます。

演算子	説明
&	AND
|	OR
^	EXCLUSIVE OR
~	NOT

AND 演算子 (&)

AND 演算子は 2 つのビットを比較します。両方とも 1 の場合、結果は 1 です。

Bit 1	Bit 2	Bit 1 & Bit 2
0	0	0
0	1	0
1	0	0
1	1	1

ビット処理 OR (|)

OR 演算子は 2 つのビットを比較します。どちらかが 1 の場合、結果は 1 です。

Bit 1	Bit 2	Bit 1 | Bit 2
0	0	0
0	1	1
1	0	1
1	1	1

EXCLUSIVE OR (^)

2 つのオペランドの両方ではなく一方だけが 1 である場合、EXCLUSIVE OR 演算子の結果は 1 です。

Bit 1	Bit 2	Bit 1 ^Bit 2
0	0	0
0	1	1
1	0	1

Bit 1	Bit 2	Bit 1 ^Bit 2
1	1	0

NOT (~)
NOT 演算子は、そのオペランドの逆の値を返す単行演算子です。

Bit	~ Bit
1	0
0	1

ジョイン演算子
Sybase IQ では、**FROM** 句でテーブル式を使用する ISO/ANSI SQL のジョイン構文の他にも、Transact-SQL™ 外部ジョイン演算子 *= と =* をサポートしています。

互換性

- モジュロ ― 新規データベースについては、デフォルト値は OFF です。
- 文字列連結 ― Sybase IQ で + 連結演算子を使用する場合は、暗黙的データ変換を行わずに、必ずオペランドを文字データ型に明示設定してください。 たとえば、次のクエリは整数値 579 を返します。

  ```
  SELECT 123 + 456
  ```

  これに対し、次のクエリは文字列 123456 を返します。

  ```
  SELECT '123' + '456'
  ```

  CAST または **CONVERT** 関数を使用すると、データ型を明示的に変換できます。

  注意： BINARY または VARBINARY データ型で使用すると、+ 演算子は加算ではなく連結演算子として機能します。

  Adaptive Server Enterprise では、|| 連結演算子をサポートしていません。

演算子の優先度
次の推奨事項に従って、実行順序を明確に指定してください。

1 つの式に複数の演算子を使用している場合は、Adaptive Server Enterprise と Sybase IQ で同一の演算子の優先度に依存しないで、カッコを使用して演算の順序を明示指定することをおすすめします。

IF 式

IF 式は IF-THEN-ELSE SQL 式を提供します。

IF 式の構文は、次のとおりです。

```
IF condition

                THEN expression1
[ ELSE expression2 ]
ENDIF
```

この式は、次の値を返します。

- *condition* が TRUE の場合、IF 式は *expression1* を返します。
- *condition* が FALSE の場合、IF 式は *expression2* を返します。
- *condition* が FALSE で *expression2* がない場合、IF 式は NULL を返します。
- 条件が NULL の場合、IF 式は NULL を返します。

注意： IF 文は IF 式とは異なります。

IF 式の構文と IF 文の構文を混同しないでください。

CASE 式

CASE 式は条件付きの SQL 式を提供します。

CASE 式は、式を使用できる場所ならどこでも使用できます。**CASE** 式の構文は、次のとおりです。

```
CASE expression

               WHEN expression
               THEN expression [, …]
[ ELSE expression ] END
```

CASE 文で、値式としてサブクエリを使用することはできません。

CASE 文に続く式が **WHEN** 文に続く式と等しい場合、**THEN** 文に続く式が返されます。それ以外の場合、**ELSE** 文があれば、それに続く式が返されます。

たとえば、下記のコードでは CASE 式が **SELECT** 文の 2 番目の句として使用されています。

```
SELECT ID,
  (CASE name
  WHEN 'Tee Shirt' THEN 'Shirt'
  WHEN 'Sweatshirt' THEN 'Shirt'
  WHEN 'Baseball Cap' THEN 'Hat'
  ELSE 'Unknown'
  END) as Type
FROM "GROUPO".Products
```

次の構文も使用できます。

```
CASE
```

```
                WHEN search-condition
                THEN expression [, …]
[ ELSE  expression ] END
```

WHEN 文に続く検索条件が満たされた場合は、**THEN** 文に続く式が返されます。それ以外の場合、**ELSE** 文があれば、それに続く式が返されます。

たとえば、次の文では、CASE 式を **SELECT** 文の 3 番目の句として使用し、検索条件と文字列を関連付けています。

```
SELECT ID, name,
  (CASE
  WHEN name='Tee Shirt' THEN 'Sale'
  WHEN quantity >= 50  THEN 'Big Sale'
  ELSE 'Regular price'
  END) as Type
FROM "GROUPO".Products
```

参照：

- NULLIF 関数 [その他] (260 ページ)
- 省略形 CASE 式の NULLIF 関数 (32 ページ)

省略形 CASE 式の NULLIF 関数

NULLIF 関数は、**CASE** 文を省略形で記述する方法の 1 つです。

NULLIF の構文は、次のとおりです。

```
NULLIF ( expression-1, expression-2 )
```

NULLIF は 2 つの式の値を比較します。1 番目の式と 2 番目の式が等しい場合、**NULLIF** は NULL を返します。1 番目の式と 2 番目の式が異なる場合、**NULLIF** は 1 番目の式を返します。

参照：

- CASE 式 (31 ページ)
- NULLIF 関数 [その他] (260 ページ)

式と定数の互換性

この項では、Adaptive Server Enterprise と Sybase IQ の式と定数の互換性について説明します。

式の互換性

次の表は、Adaptive Server Enterprise と Sybase IQ の式の互換性を示します。

この表は、簡単な説明を目的としてまとめたものです。**両方** と記されていても、状況や目的に左右されることなく、同じ方法で式を実行できるわけではありませ

ん。詳細については、Adaptive Server Enterprise と Sybase IQ の各マニュアルに記載されている個々の式の説明を参照してください。

この表では、**expr** は式、**op** は演算子を表しています。

表 4 : Adaptive Server Enterprise と Sybase IQ の式の互換性

式	サポート
constant	両方
column name	両方
variable name	両方
function (expr)	両方
- expr	両方
expr op expr	両方
(expr)	両方
(subquery)	両方
if-expression	Sybase IQ のみ

定数の互換性

次の表は、Adaptive Server Enterprise と Sybase IQ の定数の互換性を示します。

この表は、簡単な説明を目的としてまとめたものです。**両方** と記されていても、状況や目的に左右されることなく、同じ方法で式を実行できるわけではありません。詳細については、Adaptive Server Enterprise と Sybase IQ の各マニュアルに記載されている個々の式の説明を参照してください。

表 5 : Adaptive Server Enterprise と Sybase IQ の定数の互換性

定数	サポート
integer	両方
number	両方
'string'	両方
special-constant	両方
host-variable	Sybase IQ

区切り文字列のデフォルト解釈

デフォルトでは、Adaptive Server Enterprise と Sybase IQ の区切り文字列の意味は異なります。 区切り文字列とは、アポストロフィ (一重引用符) や引用符 (二重引用符) で囲まれた文字列です。

Sybase IQ では、アポストロフィで囲まれた文字列を定数式、二重引用符で囲まれた文字列を区切り識別子 (データベース・オブジェクト用の名前) とする SQL92 の表記規則を採用しています。Adaptive Server Enterprise では、二重引用符で囲まれた文字列を定数とし、デフォルトでは区切り識別子を識別子としてではなく文字列として扱う表記規則を採用しています。

quoted_identifier オプション

Adaptive Server Enterprise と Sybase IQ には、区切り文字列の解釈を変更できる **quoted_identifier** オプションがあります。**quoted_identifier** オプションのデフォルト設定は、Adaptive Server Enterprise では OFF、Sybase IQ では ON です。

quoted_identifier オプションが OFF の場合、SQL の予約語は識別子として使用できません。

Transact-SQL の **SET** 文は、Adaptive Server Enterprise の大部分の接続オプションに対応していませんが、**SET** は **quoted_identifier** オプションには対応しています。

Sybase IQ または Adaptive Server Enterprise では、次の文を使って **quoted_identifier** オプションの設定を ON に変更します。

```
SET quoted_identifier ON
```

quoted_identifier オプションを ON に設定すると、Adaptive Server Enterprise では、テーブル、ビュー、カラム名を二重引用符で区切ることができます。 他のオブジェクト名は、Adaptive Server Enterprise では区切ることができません。

Sybase IQ または Adaptive Server Enterprise では、次の文を使って **quoted_identifier** オプションの設定を OFF に変更します。

```
SET quoted_identifier OFF
```

Adaptive Server Enterprise と Sybase IQ の各 DBMS で **quoted_identifier** オプションが同じ値に設定されている場合は、SQL92 の表記規則またはデフォルトの Transact-SQL 表記規則のどちらも使用できます。

例

quoted_identifier オプションを ON (Sybase IQ のデフォルト設定) にして動作するように選択した場合、SQL キーワード **user** を指定した次の文はいずれの DBMS でも有効です。

```
CREATE TABLE "user" (
    col1 char(5)
) ;
INSERT "user" ( col1 )
VALUES ( 'abcde' ) ;
```

quoted_identifier オプションを OFF (Adaptive Server Enterprise のデフォルト設定) にして動作するように選択した場合、次の文はいずれの DBMS でも有効です。

```
SELECT *
FROM Employees
WHERE Surname = "Chin"
```

参照：

- 予約語 (19 ページ)
- 識別子 (22 ページ)

検索条件

条件を使ってテーブルからローのサブセットを選択したり、**IF** 文などの制御文中で条件を使ってフロー制御を行います。

SQL の条件式は、条件が真 (TRUE) または偽 (FALSE) であるブール論理には従いません。SQL では、すべての条件が TRUE、FALSE、または UNKNOWN のいずれかに評価されます。これを、3 値的論理といいます。比較対象となる値のいずれかが NULL の場合、比較結果は UNKNOWN になります。

比較結果が TRUE の場合のみ、ローは検索条件を満たしたことになります。比較結果が UNKNOWN のローは、検索条件を満たしません。

サブクエリは、多数の検索条件で使用される式の重要なクラスを構成します。

以降の項で、異なるタイプの検索条件について説明します。

WHERE 句、**HAVING** 句、**CHECK** 句、**JOIN** 句、または **IF** 式に対して検索条件を指定します。

構文

```
{ expression compare expression
| expression compare { ANY | SOME| ALL } ( subquery )
| expression IS [ NOT ] DISTINCT FROM
| expression IS [ NOT ] NULL expression
| expression [ NOT ] BETWEEN expression AND expression
| expression [ NOT ] LIKE expression [ ESCAPE expression ]
| expression [ NOT ] IN ( { expression | subquery |
... value-expr1 , value-expr2 [, value-expr3 ] … } )
| column-name [ NOT ] CONTAINS ( … word1 [ , word2, ] [ , word3 ] … )
| CONTAINS ( column-name [ ,...], contains-query string)
| EXISTS ( subquery )
```

```
| NOT condition
| condition AND condition
| condition OR condition
| ( condition )
| ( condition , estimate )
| condition IS [ NOT ] { TRUE | FALSE | UNKNOWN }
```

パラメータ

```
compare:
{ = | > | < | >= | <= | <> | != | !< | !> }
```

使用法
すべての場所

権限
データベースに接続しておく必要があります。

例
たとえば、次のクエリは最年長の従業員の名前と誕生年を取得します。

```
SELECT Surname, BirthDate
FROM Employees
WHERE BirthDate <= ALL (SELECT BirthDate FROM Employees);
```

限定比較述語に比較値を提供するサブクエリは、複数のローを取得できますが、カラムは 1 つしか持つことができません。

関連する動作
なし

参照：

- 比較条件 (36 ページ)
- 式 (24 ページ)
- NULL 値 (71 ページ)
- 文字列 (23 ページ)
- 3 値的論理 (51 ページ)
- SQL 演算子 (27 ページ)
- 検索条件内のサブクエリ (38 ページ)

比較条件

検索条件で比較条件を指定する場合は、比較演算子を使用します。

比較条件の構文は、次のとおりです。

```
expression  compare  expression
```

上記の *compare* は比較演算子です。 次の表は、Sybase IQ で使用可能な比較演算子の一覧です。

演算子	説明
=	等価
>	より大きい
<	より小さい
>=	より大きいまたは等価
<=	以下
!=	等価ではない
<>	等価ではない
!>	以下
!<	以上

例
たとえば、次のクエリは最年長の従業員の名前と誕生年を取得します。

```
SELECT Surname, BirthDate
FROM Employees
WHERE BirthDate <= ALL (SELECT MIN(BirthDate)FROM Employees);
```

限定比較述部に比較値を提供するサブクエリは、前掲の例のとおり複数のローを取得できますが、カラムは 1 つしか持つことができません。

注意： すべての文字列比較は

- データベースが case respect (デフォルト値) として作成されている場合、*大文字と小文字を区別します。*
- データベースが case ignore として作成されている場合、*大文字と小文字を区別しません。*

互換性

- 後続ブランク ― Adaptive Server Enterprise で比較を行う場合、文字データの後続ブランクは無視されます。 文字列を比較するときの Sybase IQ の動作は、[Ignore Trailing Blanks in String Comparison] データベース作成オプションによって制御されます。
- 大文字と小文字の区別 ― デフォルトでは、Sybase IQ データベースは Adaptive Server Enterprise データベースと同様、大文字小文字を区別するように作成されます。 比較処理は、該当のデータベースの文字の区別に合わせて実行されま

す。 Sybase IQ データベースで大文字と小文字を区別するかどうかは、データベースの作成時に制御できます。

参照：
- 式 (24 ページ)
- NULL 値 (71 ページ)
- 検索条件 (35 ページ)
- 文字列 (23 ページ)
- 3 値的論理 (51 ページ)
- SQL 演算子 (27 ページ)
- 検索条件内のサブクエリ (38 ページ)

検索条件内のサブクエリ

サブクエリとは、カッコで囲まれた **SELECT** 文です。このような **SELECT** 文には、リスト項目を 1 つだけ指定してください。

サブクエリが複数のローを返す場合、比較条件 (>、<、!= など) によってカラムをサブクエリと比較できます。(カラムを 1 つ持つ) サブクエリがローを 1 つ返す場合、そのローの値が式と比較されます。サブクエリがローを返さない場合、その値は NULL です。

1 つのカラムと任意の数のローを返すサブクエリは、**IN**、**ANY**、**ALL**、**EXISTS** の各検索条件で使用できます。以降の項で、これらの条件について説明します。

Sybase IQ では、非相関サブクエリの述部でのみ **UNION** を使用できます。スカラ値サブクエリまたは相関サブクエリの述部では使用できません。

サブクエリは、**CONTAINS** または **LIKE** 述部内では使用できません。

Sybase IQ では、単独の **OR** 句で複数のサブクエリを使用することはできません。たとえば、次のクエリには、**OR** でジョインされた 2 つのサブクエリがあります。

```
CREATE VARIABLE @ln int;SELECT @ln = 1;select count(*) FROM
lineitemWHERE l_shipdate IN (select l_shipdate FROM lineitem WHERE
l_orderkey IN (2,4,6))OR l_shipdate IN (select l_shipdate FROM
lineitem WHERE l_orderkey IN (1,3,5))OR l_linenumber = @ln;
```

AND および **BETWEEN** でジョインされた同様のサブクエリも使用できます。

参照：
- 式内のカラム名 (26 ページ)
- 予約語 (19 ページ)
- 識別子 (22 ページ)
- 比較条件 (36 ページ)

- 式 (24 ページ)
- NULL 値 (71 ページ)
- 検索条件 (35 ページ)
- 文字列 (23 ページ)
- 3 値的論理 (51 ページ)

サブクエリ述部の分離

SQL89 規格では、サブクエリ述部をいくつかの形式で指定できます。

各サブクエリは、**WHERE** 句や **HAVING** 句内で他の述部とともに使用したり、AND 演算子や OR 演算子によって結合したりできます。Sybase IQ では、相関 (外部クエリ内のテーブルへの参照を含んでおり、単独で評価できない) サブクエリまたは非相関 (リモート・テーブルへの参照を含まない) サブクエリをサポートしています。

サブクエリ述部の形式は次のとおりです。

- 非限定比較述部
  ```
  <scalar-expression> <comparison-operator> <subquery>
  ```
  比較演算子は =、<>、>、>=、< または <= です。
  非限定比較サブクエリは値を 1 つだけ返します。サブクエリが複数の値を返すと、エラー・メッセージが表示されます。このタイプのクエリは、スカラ・サブクエリ述部とも呼ばれます。
- **IN** 述部
  ```
  <scalar-expression> [NOT] IN <subquery>
  ```
  IN サブクエリ述部は、値のリストまたは 1 つの値を返します。このタイプのクエリは、限定サブクエリ述部とも呼ばれます。
- 存在述部
  ```
  [NOT] EXISTS <subquery>
  ```
  EXISTS 述部はサブクエリの存在を示します。**EXISTS** <subquery> という式は、サブクエリ結果が空でない場合にのみ true と評価されます。**EXISTS** 述部は、外部クエリ・ブロック内のカラムや式と結果を比較しません。通常は相関サブクエリとともに使用されます。
- 限定比較述部
  ```
  <scalar-expression> <comparison-operator> [ANY | ALL] <subquery>
  ```
  限定比較述部は、サブクエリから返された 1 つの値または値の集合を比較します。

実行できるクエリのタイプは次のとおりです。

- **WHERE** 句または **HAVING** 句内で垂直に実行できない、非相関スカラ・サブクエリまたは IN サブクエリの分離
- **WHERE** 句または **HAVING** 句内の相関／非相関 **EXISTS** サブクエリの分離
- **WHERE** 句または **HAVING** 句内の任意の相関／非相関スカラ・サブクエリ、**IN** または **EXISTS** サブクエリ、あるいは限定比較サブクエリの分離
- AND／OR (連結／分離) および単純な述部またはサブクエリ述部と組み合わせた任意の非相関／相関サブクエリ述部
- ビュー／抽出テーブルの上にあるサブクエリ述部の連結／分離
- **UPDATE** 文、**DELETE** 文、**SELECT INTO** 文のサブクエリ述部の分離

SUBQUERY_CACHING_PREFERENCE オプションを使用すると、経験豊富な DBA は使用するサブクエリ・キャッシュ方法を選択できます。『リファレンス：文とオプション』を参照してください。

例

非相関 **EXISTS** サブクエリと **IN** サブクエリの分離

```
SELECT COUNT(*)
FROM supplier
WHERE s_suppkey IN (SELECT MAX(l_suppkey)
            FROM lineitem
            GROUP BY l_linenumber)
OR EXISTS (SELECT p_brand
      FROM part
      WHERE p_brand = ‘Brand#43’);
```

非相関 **EXISTS** サブクエリの分離

```
SELECT COUNT(*)
FROM supplier
WHERE EXISTS (SELECT l_suppkey
        FROM lineitem
        WHERE l_suppkey = 12345)
OR EXISTS   (SELECT p_brand
        FROM part
        WHERE p_brand = ‘Brand#43’);
```

非相関スカラまたは **IN** サブクエリ述部の分離

```
SELECT COUNT(*)
FROM supplier
WHERE s_acctbal*10 > (SELECT MAX(o_totalprice)
             FROM orders
             WHERE o_custkey = 12345)
OR substring(s_name, 1, 6) IN (SELECT c_name
                  FROM Customers
                  WHERE c_nationkey = 10);
```

相関／非相関限定比較サブクエリの分離

```
SELECT COUNT(*)
FROM lineitem
```

```
WHERE l_suppkey > ANY (SELECT MAX(s_suppkey)
               FROM supplier
               WHERE s_acctbal >100
               GROUP BY s_nationkey)
OR l_partkey >= ANY (SELECT MAX(p_partkey)
             FROM part
             GROUP BY p_mfgr);
```

相関サブクエリ述部の分離

```
SELECT COUNT(*)
FROM supplier S
WHERE EXISTS (SELECT l_suppkey
        FROM lineitem
        WHERE l_suppkey = S.s_suppkey)

OR EXISTS (SELECT p_brand FROM part
      WHERE p_brand = 'Brand#43'
       AND p_partkey > S.s_suppkey);
```

サブクエリの分離がサポートされる前は、2 つの部分にクエリを記述してから、**UNION** を使用して最終結果をマージする必要がありました。

次のマージされたクエリは、相関サブクエリの述部の分離例で示したクエリと同じ結果を得ることができます。マージされたクエリは supplier テーブルを 2 回スキャンしてから、各 **UNION** からの結果をマージして最終結果を返すため、最適なパフォーマンスは得られません。

```
SELECT COUNT(*)
FROM (SELECT s_suppkey FROM supplier S
   WHERE EXISTS (SELECT l_suppkey
            FROM lineitem
            WHERE l_suppkey = S.s_suppkey)
UNION

SELECT s_suppkey
FROM supplier S
WHERE EXISTS (SELECT p_brand
        FROM part
        WHERE p_brand = 'Brand#43'
         AND p_partkey > S.s_suppkey)) as UD;
```

ALL または ANY 条件

ALL または ANY 条件は検索条件のサブクエリで使用します。

ALL 条件の構文を次に示します。

```
expression  compare
              ALL ( subquery )
```

compare は比較演算子です。

ANY 条件の構文を次に示します。

```
expression  compare
              ANY ( subquery )
```

compare は比較演算子です。

たとえば、等号演算子のある **ANY** 条件は、*expression* がサブクエリ結果のいずれかの値に等しい場合は TRUE、式が NULL ではなく、サブクエリのいずれのカラムにも当てはまらない場合は FALSE です。

```
expression = ANY ( subquery )
```

expression が NULL 値の場合、サブクエリ結果にローがあれば **ANY** 条件は UNKNOWN です。サブクエリ結果にローがなければ、条件は必ず FALSE になります。

ANY の代わりにキーワード **SOME** を使用できます。

制限事項
限定比較述部の左右どちらかに複数の式があると、エラー・メッセージが返されます。次に例を示します。

```
Subquery allowed only one select list item
```

このような種類のクエリは、**IN** サブクエリ、または **MIN** セット関数や **MAX** セット関数を使用するスカラ・サブクエリに常に置き換えることができます。

互換性
ANY と **ALL** のサブクエリは、Adaptive Server Enterprise と Sybase IQ の間で互換性があります。**SOME** を **ANY** の同意語として扱うのは、Sybase IQ だけです。

BETWEEN 条件

サブクエリで **BETWEEN** 条件を使用して、一定範囲内の値を取得します。

BETWEEN 条件の構文は、次のとおりです。

```
expr [ NOT ] BETWEEN  start-expr
               AND  end-expr
```

BETWEEN 条件は TRUE、FALSE、または UNKNOWN として評価できます。**NOT** キーワードがない場合は、*expr* が *start-expr* と *end-expr* との間にあれば、条件は TRUE と評価されます。**NOT** キーワードを使用すると条件の意味が逆になりますが、UNKNOWN は変わりません。

BETWEEN 条件は、次の 2 つの不等式の組み合わせに相当します。

```
expr >= start-expr
               AND  expr <= end-expr
```

BETWEEN 述部は、"A between B and C" の形式で使用します。"B" または "C" のいずれか、あるいは "B" と "C" の両方をサブクエリにできます。"A" は値式またはカラムでなければなりません。

互換性
BETWEEN 条件は、Sybase IQ と Adaptive Server Enterprise の間で互換性があります。

LIKE 条件

サブクエリで **LIKE** 条件を指定して、WHERE 句でワイルドカードを使用し、パターン一致を行います。

LIKE 条件の構文を次に示します。

```
expression [ NOT ] LIKE  pattern [ ESCAPE  escape-expr ]
```

LIKE 条件は TRUE、FALSE、または UNKNOWN として評価できます。**LIKE** は、文字列データのみで使用できます。

サブクエリは、**LIKE** 述部内では使用できません。

HG インデックスまたは **LF** インデックスが使用可能な場合、ワイルドカード以外の文字で始まる **LIKE** 述部の実行を高速化できます。

WD インデックスが使用可能な場合、特定の **LIKE** 述部の実行が高速化します。

NOT キーワードがない場合は、*expression* が *pattern* と一致すれば、条件は TRUE と評価されます。*expression* または *pattern* が NULL 値の場合、この条件は UNKNOWN です。**NOT** キーワードを使用すると条件の意味が逆になりますが、UNKNOWN は変わりません。

パターンには、任意の数のワイルド・カード文字を指定できます。ワイルドカード文字は次のとおりです。

ワイルドカード	一致条件
_ (アンダースコア)	任意の 1 文字
% (パーセント記号)	0 個以上の文字からなる任意の文字列
[]	指定した範囲または文字セット内の任意の 1 文字
[^]	指定した範囲または文字セット以外の任意の 1 文字

ワイルド・カード以外の指定文字は正確に一致しなければなりません。

たとえば、次のような検索条件があるとします。

```
name LIKE 'a%b_'
```

文字 a で始まり、末尾から 2 つ目の文字が b のローの場合、TRUE になります。

escape-expr が指定されている場合は、エスケープ文字は 1 文字として評価されます。*pattern* 内のパーセント記号、アンダースコア、左側の角カッコ、または別のエスケープ文字の前にエスケープ文字を置くことによって、特殊文字に特別な意味を持たせないようにできます。このようにすると、パーセント記号はパーセント記号として、アンダースコアはアンダースコアとして検出されます。

サポートされているパターン
126 文字以下のパターンは、すべてサポートされています。

127 ～ 254 文字のパターンの中には、特定の場合のみサポートされるものがあります。例として、以下のサブクエリを参照してください。

255 文字以上のパターンはまったくサポートされません。

127 ～ 254 文字のパターン：例 1
パターン内に定数文字が隣接して存在する場合、長さが 127 ～ 254 文字のパターンがサポートされる場合があります。文字列パターン内の定数文字の 1 文字は、その文字が 1 バイト文字の場合でも 2 バイトを必要とします。**LIKE** 述部内の文字列パターンは、256 バイト (または 255/2 文字) 未満でなければなりません。これを超えると、次のようなエラーが表示されます。

```
There was an error reading the results of the SQL statement. The
displayed results may be incorrect or incomplete. Cannot compile Like
pattern: either bad pattern or pattern too long.
```

Sybase IQ は、隣接する定数文字をまとめて 1 文字とみなします。 たとえば、次のように、**LIKE** 述部に長さが 130 文字の文字列を指定した場合を考えてみます。

```
select col2 from tablen where col2 like
'12345678901234567890123456789012345678901234567890123456
78901234567890123456789012345678901234567890123456%%%%' ;
```

Sybase IQ は、文字列の最後にある隣接する 4 つの定数文字 `%%%%` をまとめて 1 つの `%` とします。これにより、文字列の長さが 130 文字から 127 文字に短縮されます。 これは最大長 256 バイト (または 255/2 文字) より短いので、エラーは生成されません。

つまり、LIKE 述部の文字列内で定数が隣接する場合、それらをまとめた文字列全体の長さが 256 バイト (または 255/2 文字) 未満であれば、長さが 127 ～ 254 文字のパターンはサポートされます。

127 ～ 254 文字のパターン：例 2
この例では、130 文字の **LIKE** 述部の最後にある隣接する 4 つの定数文字 `%%%%` を、定数文字 `7890` に置き換えます。

```
select col2 from tablen where col2 like
'12345678901234567890123456789012345678901234567890123456
78901234567890123456789012345678901234567890123456789012345678901234567890' ;
```

この場合、文字がまとめられることはありません。文字列の長さは 130 文字のままなので、Sybase IQ はエラーを生成します。

127 ～ 254 文字のパターン：例 3

この例では、130 文字の **LIKE** 述部の最後にある隣接する 4 つの定数文字 %%%% を、アンダースコア ____ (特殊文字) に置き換えます。

```
select col2 from tablen where col2 like
'12345678901234567890123456789012345678901234567890123456
78901234567890123456789012345678901234567890123456____' ;
```

Sybase IQ は、隣接する特殊文字をまとめることはありません。文字列の長さは 130 文字のままなので、Sybase IQ はエラーを生成します。

127 ～ 254 文字のパターン：例 4

この例では、130 文字の **LIKE** 述部の最後にある隣接する 4 つの定数文字 %%%% を、範囲を表す [1-3] に置き換えます。

```
select col2 from tablen where col2 like
'12345678901234567890123456789012345678901234567890123456
78901234567890123456789012345678901234567890123456[1-3]'
;
```

LIKE 述部の長さのバイト数は次のように計算されます。126 (定数文字) * 2 + 1 (角カッコ内の １) + 1 (角カッコ内の ３) + 2 (セットと範囲の式)

このように計算すると 256 バイトとなるので、Sybase IQ はエラーを生成します。

文字セットからの検索

検索の対象となる文字セットは、角カッコ内に並べて指定します。たとえば、次の条件は文字列 *smith* と *smyth* を検出します。

```
LIKE 'sm[iy]th'
```

範囲内の文字の検索

検索対象の文字範囲は、角カッコ内に範囲を書いて指定します。範囲の始めと終わりの間にハイフンを書きます。たとえば、次の条件は、文字列 *bough* と *rough* を検出しますが、*tough* は検出しません。

```
LIKE '[a-r]ough'
```

文字の範囲 [a-z] は「a 以上 z 以下」と解釈され、データベースの照合では大なり演算と小なり演算が実行されます。照合での文字の順序について詳細は、『システム管理ガイド：第 1 巻』の「各国語と文字セット」を参照してください。

範囲には、まず小さい方の値、次に大きい方の値を指定してください。たとえば、式[z-a]がある **LIKE** 条件は、ローを返しません。これは、[z-a]の範囲に一致する文字がないためです。

データベースが大文字と小文字を区別するように作成されていないかぎり、文字の範囲では大文字と小文字を区別しません。たとえば、次の条件は文字列 *Bough*、*rough*、*TOUGH* を検出します。

```
LIKE '[a-z]ough'
```

データベースが大文字と小文字を区別するように作成されている場合は、検索条件でも大文字と小文字を区別します。

文字範囲と文字セットによる検索
文字範囲と文字セットを組み合わせて角カッコ内に指定できます。たとえば、次の条件は文字列 *bough*、*rough*、*tough* を検出します。

```
LIKE '[a-rt]ough'
```

角カッコ *[a-mpqs-z]* は、「*a* から *m* の範囲内の 1 文字か、*p* または *q* か、あるいは *s* から *z* の範囲内の 1 文字」と解釈されます。

範囲外の 1 文字の検索
脱字記号 (^) は、探索から除外する文字範囲を指定します。たとえば、次の条件は文字列 *tough* を検出しますが、文字列 *rough* や *bough* は検出しません。

```
LIKE '[^a-r]ough'
```

脱字記号は、カッコ内の内容をすべて否定します。たとえば、角カッコ *[^a-mpqs-z]* は、「*a* から *m* の範囲外で、*p* でも *q* でもなく、*s* から *z* の範囲外の 1 文字」と解釈されます。

特殊な文字範囲と文字セット
角カッコ内の任意の 1 文字は、その文字を指しています。たとえば、*[a]* は文字 *a* にのみ一致します。*[^]* は脱字記号にのみ、*[%]* はパーセント記号にのみ (この場合、パーセント記号はワイルドカードとして機能しません)、*[_]* はアンダースコア文字にのみ一致します。また、*[[]* は文字 *[* にのみ一致します。

その他の特殊なケースには、次のものがあります。

- 式 *[a-]* は、文字 *a* または - に一致します。
- 式 *[]* は、一致することがないのでローを返しません。
- 式 *[* または *[abp-q* は、正しくないので構文エラーになります。
- 角カッコ内にワイルドカード文字を使用できません。式 *[a%b]* は、*a*、*%*、または *b* のいずれかを検出します。

- 脱字記号を使用しても、カッコ内の先頭になければ範囲を否定できません。式 *[a^b]* は *a*、^、または *b* のいずれかを検出します。

互換性
ESCAPE 句をサポートしているのは Sybase IQ のみです。

注意： **LIKE** 述部でのラージ・オブジェクト・データおよび変数のサポートについては、『Sybase IQ の非構造化データ分析の概要』の「非構造化データのクエリ」を参照してください。

ラージ・オブジェクト・データ型の `LONG BINARY` と `LONG VARCHAR` を使用するには、専用のライセンスを取得しておく必要があります。非構造化データ分析オプションの詳細については、『Sybase IQ の非構造化データ分析の概要』を参照してください。

参照：
- PATINDEX 関数 [文字列] (265 ページ)
- LOCATE 関数 [文字列] (239 ページ)

IN 条件

サブクエリで IN 条件を使用すると、OR 条件の使用回数を減らすことができます。

IN 条件の構文を次に示します。

```
{ expression [ NOT ] IN ( subquery )
| expression [ NOT ] IN ( expression )
| expression [ NOT ] IN ( value-expr1 , value-expr2
[ , value-expr3 ] … ) }
```

NOT キーワードがない場合、*expression* がリストされた値のいずれかに一致すれば **IN** 条件は TRUE、*expression* が NULL 値の場合は UNKNOWN、それ以外の場合は FALSE です。**NOT** キーワードを使用すると条件の意味が逆になりますが、UNKNOWN は変わりません。

IN 条件リスト内の値の最大数は 250,000 です。

互換性
IN 条件は、Adaptive Server Enterprise と Sybase IQ の間で互換性があります。

CONTAINS 条件

サブクエリで CONTAINS 条件を使用して、テキスト一致条件を定義します。

WD インデックスのあるカラムに対する **CONTAINS** 条件の構文は、次のとおりです。

```
{ column-name [ NOT ] CONTAINS ( ( word1 [ , word2 ] [ , word3 ] … )
```

column-name は、ベーステーブルの CHAR、VARCHAR、または LONG VARCHAR (CLOB) カラムであり、**WD** インデックスが付いている必要があります。*word1*、*word2*、*word3* 式は、255 バイト以内の文字列定数で、それぞれ 1 単語を含む必要があります。この単語の長さは、カラムのワード・インデックスに許可されている最大長を超えてはなりません。

NOT キーワードがない場合、*column-name* が各単語を含んでいれば **CONTAINS** 条件は TRUE、*column-name* が NULL 値の場合は UNKNOWN、それ以外の場合は FALSE です。**NOT** キーワードを使用するとこれらの値が逆になりますが、UNKNOWN は変わりません。

たとえば、次のような検索条件があるとします。

```
varchar_col CONTAINS ('cat', 'mat')
```

varchar_col の値が `The cat is on the mat` の場合は、TRUE になります。*varchar_col* の値が `The cat chased the mouse` の場合、この条件は FALSE になります。

Sybase IQ で **LIKE** と **CONTAINS** の両方を含む文を実行する場合は、**CONTAINS** 条件が優先されます。

ユーザ定義関数を含むビューでは、**CONTAINS** の基準が無視されるため、**CONTAINS** 述部を使用しないでください。代わりに、**LIKE** 述部とワイルドカードを使用するか、ビューの外部でクエリを発行します。

CONTAINS 条件で **TEXT** インデックスを使用する方法については、『Sybase IQ の非構造化データ分析の概要』を参照してください。

EXISTS 条件

EXISTS 条件は、サブクエリ結果にローが 1 つでもあれば満たされます。

EXISTS 条件の構文は、次のとおりです。

```
EXISTS( subquery )
```

EXISTS 条件は、サブクエリ結果にローが少なくとも 1 つあれば TRUE で、ローがなければ FALSE です。**EXISTS** 条件は、UNKNOWN にはなりません。

互換性

EXISTS 条件は、Adaptive Server Enterprise と Sybase IQ の間で互換性があります。

IS DISTINCT FROM 検索条件と IS NOT DISTINCT FROM 検索条件

IS DISTINCT FROM 検索条件と IS NOT DISTINCT FROM 検索条件は検索引数であり、TRUE または FALSE として評価されます。

IS [NOT] DISTINCT FROM 条件の構文を次に示します。

```
expression1  IS [ NOT ] DISTINCT FROM expression2
```

備考

IS NOT DISTINCT FROM 検索条件は、*expression1* と *expression2* が等しい場合、または両方の式が NULL の場合に TRUE として評価されます。これは、次の 2 つの検索条件を組み合わせた場合と同じ結果になります。

```
expression1 = expression2 OR ( expression1 IS NULL AND expression2 IS
NULL )
```

IS DISTINCT FROM 構文では意味が逆になります。つまり、**IS DISTINCT FROM** は、expression1 と expression2 が等しくなく、2 つの式のうち少なくとも 1 つが NULL ではない場合に TRUE として評価されます。これは、次の条件と同義です。

```
NOT( expression1 = expression2 OR ( expression1 IS NULL AND
expression2 IS NULL ))
```

標準と互換性

- **SQL/2008 –** 述部の **IS [NOT] DISTINCT FROM** は SQL/2008 標準で定義されています。**IS DISTINCT FROM** 述部は、SQL/2008 標準の機能 T151 (「DISTINCT predicate」) に定義されています。**IS NOT DISTINCT FROM** 述部は、SQL/2008 標準の機能 T152 (「DISTINCT predicate with negation」) に定義されています。

IS NULL 条件

サブクエリで IS NULL 条件を使用すると、欠落している未知のデータがある場合は NULL 値が示されます。

IS NULL 条件の構文を次に示します。

```
expression
                IS [ NOT ] NULL
```

NOT キーワードがない場合、式が NULL 値なら **IS NULL** 条件は TRUE、それ以外の場合は FALSE です。**NOT** キーワードを使用すると条件の意味が逆になります。

互換性

IS NULL 条件は、 Adaptive Server Enterprise と Sybase IQ の間で互換性があります。

論理演算子を使用した条件

AND、**OR**、**NOT** を使用して、サブクエリの検索条件を結合します。

AND を使用して、次のように条件を結合します。

```
condition1 AND condition2
```

両方の条件が TRUE の場合、結合した条件は TRUE になります。一方の条件が FALSE の場合、結合した条件は FALSE になります。それ以外の場合は、結合した条件は UNKNOWN になります。

OR を使用して、次のように条件を結合します。

```
condition1 OR condition2
```

両方の条件が TRUE の場合、結合した条件は TRUE になります。一方の条件が FALSE の場合、結合した条件は FALSE になります。それ以外の場合は、結合した条件は UNKNOWN になります。*condition1* と *condition2* のどちらの条件が先に評価されるかという決まった順序はありません。

互換性
AND 演算子と **OR** 演算子は、Sybase IQ と Adaptive Server Enterprise の間で互換性があります。

NOT 条件

NOT 条件は TRUE、FALSE、または UNKNOWN として評価できます。

NOT 条件の構文を次に示します。

```
NOT  condition1
```

condition1 が FALSE の場合、**NOT** 条件は TRUE です。*condition1* が TRUE の場合は FALSE、*condition1* が UNKNOWN の場合は UNKNOWN になります。

真理値条件

条件の真理値は TRUE または FALSE です。

真理値条件の構文は、次のとおりです。

```
IS [ NOT ] truth-value
```

NOT キーワードがない場合、*condition* が指定された *truth-value* (TRUE、FALSE、UNKNOWN のいずれか) と評価されれば、検索条件は TRUE になります。それ以

外の場合、値は FALSE です。**NOT** キーワードを使用すると条件の意味が逆になりますが、UNKNOWN は変わりません。

互換性
真理値条件がサポートされているのは、Sybase IQ のみです。

3 値的論理

SQL の論理演算子 AND、OR、NOT、IS は 3 値的論理で機能します。

以下の表は、3 値的論理を示しています。

AND 演算子

AND	TRUE	FALSE	UNKNOWN
TRUE	TRUE	FALSE	UNKNOWN
FALSE	FALSE	FALSE	FALSE
UNKNOWN	UNKNOWN	FALSE	UNKNOWN

OR 演算子

OR	TRUE	FALSE	UNKNOWN
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	UNKNOWN
UNKNOWN	TRUE	UNKNOWN	UNKNOWN

NOT 演算子

TRUE	FALSE	UNKNOWN
FALSE	TRUE	UNKNOWN

IS 演算子

IS	TRUE	FALSE	UNKNOWN
TRUE	TRUE	FALSE	FALSE
FALSE	FALSE	TRUE	FALSE
UNKNOWN	FALSE	FALSE	TRUE

参照：
- 比較条件 (36 ページ)
- 式 (24 ページ)
- NULL 値 (71 ページ)
- 検索条件 (35 ページ)
- 文字列 (23 ページ)
- SQL 演算子 (27 ページ)
- 検索条件内のサブクエリ (38 ページ)

ユーザ指定の条件ヒント

条件の選択性とは、テーブル内の条件を満たすローの割合のことです。

Sybase IQ クエリ・オプティマイザは、使用可能なインデックスからの情報を使用して、クエリを実行するための適切な方式を選択します。クエリ内の各条件について、オプティマイザはインデックスを使用して条件を実行できるかどうかを決定します。条件を実行できる場合、オプティマイザはインデックスを選択し、そのテーブル上の他の条件に対する順序を決定します。 これらの決定で最も重要な要因になるのは、条件の選択性、つまり条件を満たすテーブル・ローの割合です。

オプティマイザは通常、ユーザの介入なしに、一般的に最適な決定を行います。ただし、状況によっては、オプティマイザが条件の実行前にその選択性を正確に決定できない場合があります。これらの状況は通常、条件が適切なインデックスを使用できないカラムを対象としている場合、または算術演算または関数式が含まれるために条件が複雑すぎてオプティマイザが正確に予測できない場合に発生します。

頻繁に実行されるクエリが存在する場合、最適な実行方式を選択するために役立つ追加情報をオプティマイザに提供することによりクエリのパフォーマンスが向上するかどうかを実際に試してみたい場合があります。

ユーザ指定の条件の選択性

条件ヒントの最も簡単な形式は、オプティマイザが計算する値の代わりに使用される選択性の値を指定することです。

選択性ヒントは、クエリ・テキスト内で条件をカッコで囲むことにより指定します。次に、カッコ内の条件の後に、カンマと、選択性として使用する数値を追加します。

この選択性の値は、条件を満たすテーブル・ローのパーセンテージとして表されます。したがって、選択性の有効な数値は、100.0 ～ 0.0 です。

注意： クエリ・プランでは、選択性はパーセンテージではなく端数として表されます。したがって、ユーザ指定の選択性が 35.5 である場合、クエリ・プランでの選択性は 0.355000 となります。

例

- 次のクエリは、ship_date 値の 1.5% が 1994/06/30 より前であるという予測値を提供しています。
```
SELECT  ShipDate
FROM  SalesOrderItems
WHERE ( ShipDate < '2001/06/30', 1.5 )
ORDER BY ShipDate DESC
```
- 次のクエリでは、該当するロー全体の 0.5% が条件を満たすという予測値を提供しています。
```
SELECT *
FROM Customers c, SalesOrders o
WHERE (o.SalesRepresentative > 1000.0, 0.5)
  AND c.ID = o.customerID
```

端数のパーセンテージにより、ユーザはより正確な見積もりを指定できます。これは特に、大きなテーブルで重要になります。

互換性

SQL Anywhere Studio® では、ユーザ指定の選択性の見積もりがサポートされます。

Adaptive Server Enterprise では、ユーザ指定の選択性の見積もりはサポートされていません。

参照：

- 選択性ヒント (54 ページ)

ユーザ指定の条件ヒント文字列

条件ヒント文字列を使用して、オプティマイザに追加ヒント情報を指定できます。

これらの条件単位のヒント文字列により、ユーザは条件の実行設定を追加で指定できます。オプティマイザは、可能であればこの設定に従います。これらの設定には、条件に使用するインデックス、条件の選択性、条件の実行時の実行フェーズ、および 1 実行フェーズ内で実行される条件セット間の順序に影響する条件の有用性が含まれます。

ユーザ指定の選択性の見積もりなどの条件ヒント文字列は、クエリのテキスト内で条件をカッコで囲むことにより指定します。次に、このカッコ内の条件の後に、カンマを追加し、適切なヒントを含む文字列を引用符で囲んで指定します。この引用符で囲まれた文字列内では、各ヒントはヒント・タイプ識別子として表され、それにコロンとヒント・タイプ値が続きます。同じヒント文字列内に複数のヒントがある場合は、ヒントをカンマで区切ります。また、複数のヒントの順序は任意です。ヒント文字列内では、2 つの要素の間に空白文字を挿入できます。

選択性ヒント
ヒント文字列内に含めることのできる最初のヒント・タイプは、選択性ヒントです。選択性ヒントは、"S" または "s" のいずれかのヒント・タイプ識別子により識別されます。

ユーザ指定の選択性の見積もりの場合と同様に、この選択性の値は常に、条件を満たすテーブル・ローのパーセンテージとして表されます。

例
次の例は、選択性の項で示した 2 番目のユーザ指定条件の例とまったく同じです。

```
SELECT *
FROM Customers c, SalesOrders o
WHERE (o.SalesRepresentative > 1000.0, 's: 0.5')
  AND c.ID = o.CustomerID
```

参照：
- ユーザ指定の条件の選択性 (52 ページ)

インデックス設定ヒント
サポートされている 2 番目のヒント・タイプは、インデックス設定ヒントです。これは、"I" または "i" のいずれかのヒント・タイプ識別子により識別されます。

インデックス設定ヒントの値は -10 ～ 10 の整数です。正の整数値は、特定のインデックス・タイプが優先されることを意味し、負の値は特定のインデックス・タイプが回避されることを意味します。

インデックス設定ヒントの影響は、設定がクエリ内のすべての条件ではなく、関連する条件にのみ適用される点を除いて、**INDEX_PREFERENCE** オプションと同じです。指定のインデックス・タイプが該当カラムに存在し、関連条件の評価時に該当インデックス・タイプを使用できる場合、インデックス設定は条件の実行にのみ影響を及ぼすことができます。すべてのインデックス・タイプがすべての条件で使用できるわけではありません。『リファレンス：文とオプション』を参照してください。

例
次の例は、3% の選択性を指定し、可能であれば HG インデックスを使用して条件が評価されることを示します。

```
SELECT *
FROM Customers c, SalesOrders o
WHERE (o.SalesRepresentative > 1000.0, 'S:3.00, I:+2')
  AND c.ID = o.CustomerID
```

次の例は、37.5% の選択性を指定し、可能であれば HG インデックスを使用して条件が評価されないことを示します。

```
SELECT *
FROM Customers c, SalesOrders o
WHERE (o.SalesRepresentative > 1000.0, 'i:-2, s:37.500')
  AND c.ID = o.CustomerID
```

INDEX_PREFERENCE オプション
クエリ処理に使用するインデックスの選択を制御します。

指定できる値
-10 ～ 10

デフォルト値
0

スコープ
このオプションを設定するために、DBA パーミッションは必要ありません。個々の接続または PUBLIC グループに一時的に設定することもできます。。すぐに有効になります。

説明
Sybase IQ オプティマイザは、通常最適なインデックスを使用して、ローカルな **WHERE** 句の述部など、1 つの IQ インデックスの範囲内で処理できる操作を実行します。INDEX_PREFERENCE は、テスト目的にオプティマイザの選択を無効にするために使用します。通常の使用の場合はこのオプションの値を変更しないでください。

表 6 : INDEX_PREFERENCE の有効な値

値	アクション
0	オプティマイザの選択に従う。
1	**LF** インデックスを優先する。
2	**HG** インデックスを優先する。
3	**HNG** インデックスを優先する。
4	**CMP** インデックスを優先する。
5	デフォルトのインデックスを優先する。
6	**WD** インデックスを優先する。
8	**DATE** インデックスを優先する。
9	**TIME** インデックスを優先する。

値	アクション
10	**DTTM** インデックスを優先する。
-1	**LF** インデックスを回避する。
-2	**HG** インデックスを回避する。
-3	**HNG** インデックスを回避する。
-4	**CMP** インデックスを回避する。
-5	デフォルトのインデックスを回避する。
-6	**WD** インデックスを回避する。
-8	**DATE** インデックスを回避する。
-9	**TIME** インデックスを回避する。
-10	**DTTM** インデックスを回避する。

実行フェーズ・ヒント

サポートされる第 3 のヒント・タイプは、実行フェーズ・ヒントです。これは、"E" または "e" のいずれかのヒント・タイプ識別子により識別されます。

Sybase IQ クエリ・エンジン内には、条件を評価できる不変、遅延、バインド、水平という別個の実行フェーズがあります。

オプティマイザはデフォルトで、条件の評価のために必要な情報がすべて使用可能である最初の実行フェーズで、各条件を評価することを選択します。したがって、各条件は評価されるデフォルトの実行フェーズを持ちます。

必要な情報が使用可能になるまで条件を評価できないので、実行フェーズ・ヒントはデフォルトのフェーズより後のフェーズまで条件の実行を遅延する目的でのみ使用できます。実行フェーズ・ヒントを使用して、デフォルトのフェーズより前のフェーズで強制的に条件を評価することはできません。

次に、4 つの条件実行フェーズについて、実行される順に説明します。

不変 ― 単一の列 (または同じテーブルの 2 つのカラム) を参照し、インデックスを使用して評価できる条件は一般的に、単純な不変条件と呼ばれます。単純な不変条件は通常、最適化処理内で早期に評価されます。つまり、これらの不変条件をすべて満たすローの数を使用して、オプティマイザが、使用する最適なジョイン順序とジョイン・アルゴリズムを決定できることを意味します。これは最初の実行フェーズであるため、ユーザは条件をこのフェーズで強制的に実行することはできません。ただし、このフェーズからその後のフェーズに条件を実行させることはできます。

遅延 — 一部の条件は、他のクエリ部分が実行されるまで評価できません。この遅延条件は、関連するクエリ・ノードが最初にフェッチされたときに評価されます。これらの条件は、非相関のサブクエリ条件と、オプティマイザによって作成される IN または PROBABLY_IN のプッシュダウン・ジョイン条件の、2 つのカテゴリに分類されます。

バインド — 一部の条件は、複数回評価する必要があります。これらの条件は一般的に、相関サブクエリ内の外部参照を含む条件と、オプティマイザによって作成されるプッシュダウン等号ジョイン条件の、2 つのカテゴリに分類されます。たとえば、外部参照条件は、クエリの実行中に外部参照値が変更されるたびに再評価されます。

水平 — 1 つのテーブルの複数のカラムを含む条件など、一部の条件はインデックスを使用するのではなく、一度に 1 つずつローを評価する必要があります。

実行フェーズ・ヒントには、条件を評価する実行フェーズを識別する値を指定します。各値は、大文字と小文字が区別されない単一の文字です。

- D ― 遅延
- B ― バインド
- H ― 水平

例
次の例の条件ヒント文字列は、条件を「遅延」実行フェーズに移動し、可能であれば LF インデックスを使用して条件を評価することを示します。

```
SELECT *
FROM Customers c, SalesOrders o
WHERE (o.SalesRepresentative > 10000.0, 'E:D, I:1')
  AND c.id = o.CustomerID
```

有用性ヒント
サポートされる最後のヒント・タイプは、有用性ヒントです。これは、"U" または "u" のいずれかのヒント・タイプ識別子により識別されます。

有用性ヒントの値は、0.0 ～ 10.0 の任意の数値です。オプティマイザでは、有用性の値は条件ごとに計算されます。この有用性の値は、同じ実行フェーズ内の条件セットの評価順序を決定するために使用されます。有用性の値が大きいほど、評価順序が前になります。ユーザは、有用性ヒントを指定することにより、評価順序内の特定位置に条件を配置できます。ただし、それによって、条件が評価される実行フェーズを変更することはできません。

例
次の例の条件ヒント文字列は、条件を「遅延」実行フェーズに移動し、「遅延」フェーズ内で有用性を 3.25 に設定することを示します。

```
SELECT *
FROM Customers c, SalesOrders o
WHERE (o.SalesRepresentative > 10000.0, 'U: 3.25,  E: D')
AND c.id = o.CustomerID
```

互換性
SQL Anywhere Studio では、ユーザ指定の条件ヒント文字列はサポートされていません。

Adaptive Server Enterprise では、ユーザ指定の条件ヒント文字列はサポートされていません。

ジョイン等号条件に関するユーザ指定のヒント
ジョイン・アルゴリズムの優先順位を指定できます。この順位によってクエリ内のすべてのジョインが影響を受けるとは限りません。

単純な等号ジョイン述部に述部ヒントのタグを付けることができます。このヒントにより、まさにその 1 つのジョインのためにジョインの優先順位を指定できます。ローカルなジョインの優先順位が設定されたジョイン条件が、同じジョインに複数あり、しかもそれらのヒントの値が異なる場合、そのジョインに対するローカルな優先順位がすべて無視されます。ローカルなジョインの優先順位は、オプティマイザが選択したジョインの順序に影響を与えません。

次の例はハッシュ・ジョインを要求します。

```
AND (T.X = 10 * R.x, 'J:4')
```

ユーザ指定の条件ヒントの使用ガイドライン
条件ヒントは一般的に、頻繁に実行するクエリでのみ使用します。

条件ヒントの試行は経験豊富なユーザのみが行ってください。一般的に、使用可能なインデックスからの条件に関する正確な情報を推定できない場合を除いて、オプティマイザがオプションの決定を行います。

オプティマイザは頻繁に元の条件を書き直したり簡略化したりします。また、元の条件から新しい条件を推定します。条件ヒントは、オプティマイザにより推定された条件に至るまで、または簡略化された条件に至るまで、新規に実行されることはありません。

特別値

特別値は、式の中で使用したり、テーブル作成時にカラムのデフォルトとして使用したりできます。

参照：
- 式 (24 ページ)

CURRENT DATABASE 特別値

CURRENT DATABASE は現在のデータベースの名前を返します。

データ型
STRING

CURRENT DATE 特別値

CURRENT DATE は、現在の年、月、日を返します。

データ型
DATE

参照：
- TIMESTAMP 特別値 (62 ページ)
- CURRENT TIMESTAMP 特別値 (60 ページ)
- CURRENT TIME 特別値 (59 ページ)
- 日付と時刻のデータ型 (91 ページ)
- 日付と時刻の取得 (94 ページ)

CURRENT PUBLISHER 特別値

CURRENT PUBLISHER は、SQL Remote レプリケーション用データベースのパブリッシャ・ユーザ ID を含む文字列を返します。

データ型
STRING

CURRENT PUBLISHER は、文字データ型のカラムでデフォルト値として使用できます。

CURRENT TIME 特別値

CURRENT TIME は、現在の時、分、秒 (小数位あり) で構成される時刻を返します。

データ型
TIME

説明
秒の小数位は 6 桁まで格納されますが、現在の時刻の精度はシステム・クロックの精度によって制限されます。

参照：
- TIMESTAMP 特別値 (62 ページ)
- CURRENT TIMESTAMP 特別値 (60 ページ)
- CURRENT DATE 特別値 (59 ページ)
- 日付と時刻のデータ型 (91 ページ)
- 日付と時刻の取得 (94 ページ)

CURRENT TIMESTAMP 特別値

CURRENT DATE と **CURRENT TIME** を結合して形成された **TIMESTAMP** 値です。年、月、日、時、分、秒、秒の小数位で構成されます。

CURRENT TIME と同様に、秒の小数位の精度はシステム・クロックによって制限されます。

CURRENT TIMESTAMP のデフォルト値は 3 桁です。

データ型
`TIMESTAMP`

参照：
- TIMESTAMP 特別値 (62 ページ)
- CURRENT TIME 特別値 (59 ページ)
- CURRENT DATE 特別値 (59 ページ)
- 日付と時刻のデータ型 (91 ページ)
- 日付と時刻の取得 (94 ページ)
- CURRENT USER 特別値 (60 ページ)
- LAST USER 特別値 (61 ページ)
- USER 特別値 (62 ページ)

CURRENT USER 特別値

CURRENT USER は、現在の接続のユーザ ID を含む文字列を返します。

UPDATE では、**CURRENT USER** のデフォルト値を持つカラムは変更されません。

データ型
`STRING`

CURRENT USER は文字データ型のカラムでデフォルト値として使用できます。

参照：
- CURRENT TIMESTAMP 特別値 (60 ページ)

- LAST USER 特別値 (61 ページ)
- USER 特別値 (62 ページ)

LAST USER 特別値

LAST USER は、ローを最後に更新したユーザの名前を返します。

INSERT と **LOAD** の場合、この定数は **CURRENT USER** と同じ効果があります。**UPDATE** では、**LAST USER** のデフォルト値を持つカラムが明示的に変更されていない場合、現在のユーザ名に変更されます。

LAST USER のデフォルト値を **DEFAULT TIMESTAMP** と組み合わせて使用すると、ローを最後に変更したユーザと日時の両方を (別々カラムに) 記録できます。

データ型
`STRING`

LAST USER は、文字データ型のカラムでデフォルト値として使用できます。

参照：
- CURRENT USER 特別値 (60 ページ)
- CURRENT TIMESTAMP 特別値 (60 ページ)
- USER 特別値 (62 ページ)

SQLCODE 特別値

SQLCODE は現在の **SQLCODE** 値を返します。

SQLCODE 値は各文の後に設定されます。**SQLCODE** をチェックして、文の実行が成功したかどうかを確認できます。

データ型
`STRING`

SQLSTATE 特別値

SQLSTATE は現在の **SQLSTATE** 値を返します。

SQLSTATE 値は各文の後に設定されます。**SQLSTATE** をチェックして、文の実行が成功したかどうかを確認できます。

データ型
`STRING`

TIMESTAMP 特別値

TIMESTAMP は、テーブルの各ローが最後に修正された日時を示します。

DEFAULT TIMESTAMP 型としてカラムを宣言すると、挿入処理およびロード処理に対してデフォルト値が提供されます。この値は、ローが更新されたときに常に、最新の日時に更新されます。

INSERT と **LOAD** の場合、**DEFAULT TIMESTAMP** は **CURRENT TIMESTAMP** と同じ効果があります。**UPDATE** の場合、**TIMESTAMP** のデフォルト値を持つカラムは、明示的に変更されないかぎり、その値が現在の日時に変更されます。

注意： Sybase IQ は、**UTC TIMESTAMP** や **CURRENT UTC TIMESTAMP** の **DEFAULT** 値をサポートしていません。また、データベース・オプション `DEFAULT_TIMESTAMP_INCREMENT` もサポートしていません。**UTC TIMESTAMP** 型または **CURRENT UTC TIMESTAMP** 型のカラムで **DEFAULT** 値の挿入または更新が試みられるたびに、Sybase IQ はエラーを生成します。

データ型
`TIMESTAMP`

参照：
- CURRENT TIMESTAMP 特別値 (60 ページ)
- CURRENT TIME 特別値 (59 ページ)
- CURRENT DATE 特別値 (59 ページ)
- 日付と時刻のデータ型 (91 ページ)
- 日付と時刻の取得 (94 ページ)

USER 特別値

USER は、現在の接続のユーザ ID を含む文字列を返します。

UPDATE では、**USER** のデフォルト値を含むカラムは変更されません。

データ型
`STRING`

USER は、文字データ型のカラムでデフォルト値として使用できます。

参照：
- CURRENT USER 特別値 (60 ページ)
- CURRENT TIMESTAMP 特別値 (60 ページ)
- LAST USER 特別値 (61 ページ)

変数

Sybase IQ は、ローカル変数、接続レベル変数、グローバル変数をサポートしています。

グローバル変数の名前はすべて、2 つのアット・マーク (@) で始まります。たとえば、グローバル変数 *@@version* の値は、データベース・サーバの現在のバージョン番号です。ユーザはグローバル変数を定義できません。

ローカル変数

ローカル変数はユーザが宣言します。この変数を SQL 文のプロシージャまたはバッチ内で使用して、情報を保持できます。

ローカル変数は **DECLARE** 文で宣言し、複合文 (**BEGIN** キーワードと **END** キーワードで囲まれた部分) の中でのみ使用できます。変数の初期設定値は NULL です。変数の値は、**SET** 文によって設定するか、**INTO** 句のある **SELECT** 文で割り当てることができます。

DECLARE 文の構文は、次のとおりです。

```
DECLARE variable-name data-type
```

ローカル変数は、プロシージャが複合文中から呼び出されるかぎり、プロシージャに引数として引き渡すことができます。

例

- 次のバッチは、ローカル変数の使用例を示します。

  ```
  BEGIN
      DECLARE local_var INT ;
      SET local_var = 10 ;
      MESSAGE 'local_var = ', local_var ;
  END
  ```

  ISQL からこのバッチを実行すると、サーバのウィンドウに次のメッセージが表示されます。

  ```
  local_var = 10
  ```

- 変数 `local_var` は、変数が宣言された複合文の外側には存在しません。次のバッチは無効で、「カラムが見つかりません」というエラーになります。

  ```
  -- This batch is invalid.
  BEGIN
      DECLARE local_var INT ;
      SET local_var = 10 ;
      MESSAGE 'local_var = ', local_var ;
  ```

```
END;
MESSAGE 'local_var = ', local_var ;
```

- 次の例は、**INTO** 句のある **SELECT** 文を使用してローカル変数を設定する方法を示します。

```
BEGIN
    DECLARE local_var INT ;
    SELECT 10 INTO local_var ;
    MESSAGE 'local_var = ', local_var ;
END
```

ISQL からこのバッチを実行すると、サーバのウィンドウに次のメッセージが表示されます。

```
local_var = 10
```

互換性

名前 — Adaptive Server Enterprise と Sybase IQ は、いずれもローカル変数をサポートしています。 Adaptive Server Enterprise では、すべての変数名の先頭にアット・マーク (@) が付いています。 Sybase IQ では、@ プレフィックスはオプションです。 互換性のある SQL 文を書く場合は、変数名の先頭に必ずアット・マーク (@) を付けます。

スコープ — Sybase IQ と Adaptive Server Enterprise では、ローカル変数のスコープが異なります。Sybase IQ では、バッチ内でのローカル変数の宣言に **DECLARE** 文を使用できます。 ただし、**DECLARE** が複合文で実行される場合、変数のスコープは複合文内に制限されます。

宣言 − Sybase IQ では、各 **DECLARE** 文で宣言できる変数は 1 つだけです。 Adaptive Server Enterprise では、1 つの文で複数の変数を宣言できます。

接続レベル変数

接続レベル変数はユーザが宣言します。この変数を SQL 文のプロシージャまたはバッチ内で使用して、情報を保持できます。

接続レベル変数は、**CREATE VARIABLE** 文で宣言します。**CREATE VARIABLE** 文は、複合文内を除き、任意の位置で使用できます。接続レベル変数は、プロシージャにパラメータとして引き渡すことができます。

CREATE VARIABLE の構文は、次のとおりです。

```
CREATE VARIABLE variable-name data-type
```

変数が作成されると、値は NULL に初期設定されます。接続レベル変数の値は、ローカル変数と同じように、**SET** 文または **INTO** 句のある **SELECT** 文を使用して設定できます。

接続レベル変数は、接続が終了するまで、または **DROP VARIABLE** 文を使用して変数が明示的に削除されるまで存在します。次の文は、変数 *con_var* を削除します。

```
DROP VARIABLE con_var
```

例

- 次の SQL 文のバッチは、接続レベル変数の使用例を示します。

  ```
  CREATE VARIABLE con_var INT;
  SET con_var = 10;
  MESSAGE 'con_var = ', con_var;
  ```

  ISQL からこのバッチを実行すると、サーバのウィンドウに次のメッセージが表示されます。

  ```
  con_var = 10
  ```

互換性
Adaptive Server Enterprise は接続レベル変数をサポートしません。

グローバル変数

グローバル変数は、システム定義の値が設定されるシステム定義の変数です。

グローバル変数の値は Sybase IQ によって設定されます。たとえば、グローバル変数 *@@version* の値は、データベース・サーバの現在のバージョン番号です。

グローバル変数は、名前の先頭に付けられた 2 つの at 記号 (@) によって、ローカル変数および接続レベル変数と区別されます。たとえば、*@@error* はグローバル変数です。ユーザは、グローバル変数を定義したり、その値を直接更新したりできません。

一部のグローバル変数 (*@@spid* など) は、接続に固有の情報と値を保持します。その他の変数 (*@@connections* など) は、すべての接続に共通の値を保持します。

グローバル変数と特殊定数
特殊定数 (**CURRENT DATE**、**CURRENT TIME**、**USER**、**SQLSTATE** など) は、グローバル変数に類似しています。

次の文は、グローバル変数 version の値を取得します。

```
SELECT @@version
```

プロシージャでは、グローバル変数を変数リストに選択できます。次のプロシージャは、*ver* パラメータにサーバのバージョン番号を返します。

```
CREATE PROCEDURE VersionProc ( OUT ver
                 VARCHAR ( 100) )
BEGIN
    SELECT @@version
```

```
    INTO ver;
END
```

Embedded SQL では、グローバル変数をホスト変数リストに選択できます。

グローバル変数のリスト
次の表は、Sybase IQ で使用可能なグローバル変数の一覧です。

表 7 : Sybase IQ グローバル変数

変数名	意味
@@error	通常、直前に実行された文のエラー・ステータス (成功または失敗) のチェックに使用する。直前のトランザクションが成功していれば値は 0 で、それ以外の場合、システムが生成した最新のエラー番号が設定される。エラーが発生した場合、if @@error != 0 return のような文によって終了する。すべての SQL 文は @@error をリセットするため、実行の成否を判断する文の直後にステータス・チェックを行うこと。
@@fetch_status	最後の FETCH 文によって得られたステータス情報を保持する。 @@fetch_status には、次のいずれかの値が含まれる。 • 0：フェッチ文は正常終了した。 • fetch 文がエラーになった。 • -2：結果セットにこれ以上データがない。 この機能は、返す値が異なる点を除いて @@sqlstatus と同様。これは Microsoft SQL Server との互換性のため。
@@identity	insert、load、update 文により Identity/Autoincrement カラムに挿入された最後の値。ローがテーブルに挿入されると、*@@identity* は必ずリセットされる。文によって複数のローが挿入された場合、*@@identity* は、最後に挿入されたローの Identity/Autoincrement 値を反映する。関連するテーブルに Identity/Autoincrement カラムがない場合、*@@identity* は 0 に設定される。insert、load、update 文が正常に実行されなかったり、失敗した文を含むトランザクションがロールバックされても、*@@identity* の値には影響しない。*@@identity* は、Identity/Autoincrement カラムに最後の値を挿入した文がコミットに失敗した場合でも、その値を保持する。
@@isolation	現在の独立性レベル。@@isolation には、アクティブ・レベルの値が入る。
@@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。

変数名	意味
@@rowcount	最後の文の影響を受けるローの数。*@@rowcount* の値は、文の直後にチェックする必要があります。挿入、更新、削除を実行すると、影響を受けたローの数が *@@rowcount* に設定されます。カーソルを使用する場合、*@@rowcount* は、最後のフェッチ要求までにカーソルの結果セットからクライアントに返されたローの累積数を表します。*@@rowcount* が、**IF** 文のようにローに影響を及ぼさない文によって 0 にリセットされることはありません。
@@servername	現在のデータベース・サーバの名前。
@@sqlstatus	最後の **FETCH** 文によって得られたステータス情報を保持する。
@@version	現在の Sybase IQ のバージョン番号。

Sybase IQ がサポートする Adaptive Server Enterprise グローバル変数

次の表は、Sybase IQ でサポートされているすべての Adaptive Server Enterprise グローバル変数を示します。Sybase IQ でサポートされない Adaptive Server Enterprise グローバル変数は、このリストに含まれません。

この表に記載されているグローバル変数はすべて値を返します。値には、NULL、1、-1、0 などの固定値のほか、意味のないものも含まれます。

表 8 : Sybase IQ がサポートする Adaptive Server Enterprise グローバル変数

グローバル変数	戻り値
@@char_convert	0 を返す。
@@client_csname	Adaptive Server Enterprise では、クライアントの文字セット名。 クライアントの文字セットが一度も初期化されていない場合は NULL が設定され、それ以外の場合は、直前に使用された文字セットの名前が入る。 Sybase IQ では NULL が返される。
@@client_csid	Adaptive Server Enterprise では、クライアント文字セットの ID を表す。クライアントの文字セットが一度も初期化されていない場合は -1 に設定され、それ以外の場合は、syscharsets の最近使用された文字セットの ID が入る。 Sybase IQ では -1 が返される。
@@connections	最後にサーバが起動されてからのログイン数。
@@cpu_busy	Adaptive Server Enterprise では、Adaptive Server Enterprise が最後に起動されてから、CPU が Adaptive Server Enterprise での処理に費やした時間 (チック単位)。Sybase IQ では 0 が返される。

グローバル変数	戻り値
@@error	通常、直前に実行された文のエラー・ステータス (成功または失敗) のチェックに使用する。直前のトランザクションが成功していれば値は 0 で、それ以外の場合、システムが生成した最新のエラー番号が設定される。次のような文の場合、 `if @@error != 0 return` エラーが発生すると終了する。**PRINT** 文や **IF** テストなどを含め、すべての SQL 文は *@@error* をリセットしてしまうので、成否の確認が必要な文については、その直後にステータス・チェックを行う必要がある。
@@identity	Adaptive Server Enterprise では、**INSERT** 文、**LOAD** 文、または **SELECT INTO** 文によって `IDENTITY` カラムに最後に挿入された値。ローがテーブルに挿入されると、*@@identity* は毎回リセットされる。 文によって複数のローが挿入された場合、*@@identity* は、最後に挿入されたローの `IDENTITY` 値を反映する。影響を受けるテーブルに `IDENTITY` カラムがない場合、*@@identity* は 0 に設定される。**INSERT** 文または **SELECT INTO** 文の実行に失敗したり、この失敗した文を含むトランザクションをロールバックしたりした場合でも、*@@identity* には影響しない。*@@identity* は、`IDENTITY` カラムに最後に挿入された値を保持し、その値を挿入した式がコミットに失敗してもかまわない。
@@idle	Adaptive Server Enterprise では、サーバが最後に起動してからの Adaptive Server Enterprise のアイドル時間 (チック単位)。 Sybase IQ では 0 が返される。
@@io_busy	Adaptive Server Enterprise では、最後にサーバが起動されてから、Adaptive Server Enterprise が入出力処理に費やした時間数 (チック単位)。 Sybase IQ では 0 が返される。
@@isolation	接続の現在の独立性レベル。Adaptive Server Enterprise では、*@@isolation* にはアクティブ・レベルの値が入る。
@@langid	Adaptive Server Enterprise では、現在使用中の言語のローカル言語 ID を定義する。 Sybase IQ では 0 が返される。
@@language	Adaptive Server Enterprise では、現在使用中の言語の名前を定義する。 Sybase IQ では "English" が返される。
@@maxcharlen	Adaptive Server Enterprise では、Adaptive Server Enterprise のデフォルト文字セット中の 1 文字の最大長 (バイト)。 Sybase IQ では 1 が返される。

グローバル変数	戻り値
@@max_ connections	ネットワーク・サーバの場合は、アクティブ・クライアントの最大数 (各クライアントが複数の接続をサポートできるため、データベース接続数ではない)。 Adaptive Server Enterprise の場合は、サーバとの接続の最大数。
@@ncharsize	Adaptive Server Enterprise では、各国語 1 文字の平均の長さ (バイト単位)。 Sybase IQ では 1 が返される。
@@nestlevel	Adaptive Server Enterprise では、現在の実行のネスト・レベル (初期値 0)。 ストアド・プロシージャまたはトリガが別のストアド・プロシージャやトリガを呼び出すたびに、ネスト・レベルは増加する。 Sybase IQ では -1 が返される。
@@pack_received	Adaptive Server Enterprise では、サーバが最後に起動してから Adaptive Server Enterprise によって読み込まれた受信パケットの数。 Sybase IQ では 0 が返される。
@@pack_sent	Adaptive Server Enterprise では、サーバが最後に起動してから Adaptive Server Enterprise によって書き込まれた送信パケットの数。 Sybase IQ では 0 が返される。
@@packet_errors	Adaptive Server Enterprise では、Adaptive Server Enterprise のパケットの送受信中に発生したエラーの数。 Sybase IQ では 0 が返される。
@@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。
@@servername	ローカルの Adaptive Server Enterprise サーバまたは Sybase IQ サーバの名前。
@@spid	Adaptive Server Enterprise では、現在のプロセスのサーバ・プロセス ID 番号。 Sybase IQ では、現在の接続のコネクション・ハンドル。 これは、**sa_conn_info** プロシージャによって表示される値と同じ。
@@sqlstatus	最後の **FETCH** 文によって得られたステータス情報を保持する。 *@@sqlstatus* は次の値を含む。 • 0 ― FETCH 文が正常に終了した。 • 1 ― FETCH 文がエラーになった。 • 2 ― 結果セットには値が残されていない。
@@thresh_hysteresis	Adaptive Server Enterprise では、スレッショルドのアクティブ化に必要な空き領域の変更。 Sybase IQ では 0 が返される。
@@timeticks	Adaptive Server Enterprise では、チックあたりのマイクロ秒数。 1 チックの長さはマシンによって異なる。Sybase IQ では 0 が返される

グローバル変数	戻り値
@@total_errors	Adaptive Server Enterprise では、Adaptive Server Enterprise の読み取りと書き込み中に発生したエラーの数。 Sybase IQ では 0 が返される。
@@total_read	Adaptive Server Enterprise では、サーバが最後に起動してから Adaptive Server Enterprise によって実行されたディスクの読み取りの数。 Sybase IQ では 0 が返される。
@@total_write	Adaptive Server Enterprise では、サーバが最後に起動してから Adaptive Server Enterprise によって実行されたディスクの書き込みの数。 Sybase IQ では 0 が返される。
@@tranchained	Transact-SQL プログラムの現在のトランザクション・モード。 *@@tranchained* は、非連鎖モードの場合は 0、連鎖モードの場合は 1 を返す。
@@trancount	トランザクションのネスト・レベル。バッチ内の **BEGIN TRANSACTION** ごとに、トランザクション・カウントが増分する。
@@transtate	Adaptive Server Enterprise では、文を実行した後のトランザクションの現在の状態。 Sybase IQ では -1 が返される。
@@version	Adaptive Server Enterprise または Sybase IQ の現在のバージョン情報。

コメント

コメントは、SQL 文または文ブロックに説明テキストを付加するために使用します。データベース・サーバは、コメントを実行しません。

Sybase IQ では、以下のコメント・インジケータを使用できます。

コメント・インジケータ	説明
-- (二重ハイフン)	データベース・サーバは、その行のハイフンに続く文字を無視します。これは、SQL92 のコメント・インジケータです。
// (二重スラッシュ)	二重スラッシュは、二重ハイフンと同じ意味です。
/* … */ (スラッシュ ― アスタリスク)	2 つのコメント・マーカの間にある文字は、すべて無視されます。2 つのコメント・マーカは、同じ行にあっても別の行にあってもかまいません。このスタイルで示されたコメントはネストできます。このスタイルは、C スタイル・コメントとも呼ばれます。

コメント・インジケータ	説明
% (パーセント記号)	パーセント記号は、二重ハイフンと同じ意味になります。% をコメント・インジケータとして使用しないことをおすすめします。

注意： -- (二重ハイフン) と /* (スラッシュ - アスタリスク) のコメント・スタイルは、Adaptive Server Enterprise と互換性があります。

例
次に、二重ハイフンを使用したコメントの記述例を示します。

```
CREATE FUNCTION fullname (firstname CHAR(30),
            lastname CHAR(30))
RETURNS CHAR(61)
-- fullname concatenates the firstname and lastname
-- arguments with a single space between.
BEGIN
    DECLARE name CHAR(61);
    SET name = firstname || ' ' || lastname;
    RETURN ( name );
END
```

次に、C スタイルを使ったコメントの記述例を示します。

```
/*
    Lists the names and employee IDs of employees
    who work in the sales department.
*/
CREATE VIEW SalesEmployee AS
SELECT emp_id, emp_lname, emp_fname
FROM "GROUPO".Employees
WHERE DepartmentID = 200
```

NULL 値

NULL 値を使用して、未知、欠落、適用不可の値を指定します。

NULL 値は、あらゆるデータ型の有効な値とは異なる特別な値です。ただし、NULL 値はすべてのデータ型で使用できます。NULL 値が使用されるこれら 2 つのケースは、それぞれが個別で性質も異なることに注意してください。

状況	説明
欠落	フィールドには値があるが、未知の値である
適用不可	フィールドは、この特定のローに適用できない

SQL では、NOT NULL 制限を使用してカラムを作成できます。このカラムには NULL 値を挿入できません。

NULL 値によって、SQL に 3 値的論理の概念が導入されました。任意の比較演算子を使って、NULL 値を含む任意の値と NULL 値を比較すると UNKNOWN になります。TRUE が返る唯一の検索条件は、IS NULL 述語です。SQL では、**WHERE** 句の検索条件が TRUE と評価された場合のみ、ローが選択されます。UNKNOWN または FALSE と評価されたローは、選択されません。

IS [NOT]*truth-value* 句は、NULL 値があるローを選択するために使用します (*truth-value* は TRUE、FALSE、UNKNOWN のいずれかです)。

次の例では、カラム `Salary` が NULL 値を持っています。

条件	真理値	選択
Salary = NULL	UNKNOWN	なし
Salary <> NULL	UNKNOWN	なし
NOT (Salary = NULL)	UNKNOWN	なし
NOT (Salary <> NULL)	UNKNOWN	なし
Salary = 1000	UNKNOWN	なし
Salary IS NULL	TRUE	あり
Salary IS NOT NULL	FALSE	なし
Salary = 1000 IS UNKNOWN	TRUE	あり

2 つの異なるテーブルのカラムを比較する場合、同じ規則が適用されます。そのため、2 つのテーブルを結合すると、比較したカラムに NULL 値があるローは選択されません。

数値式で使用する場合も、NULL 値は特別な性質を持っています。NULL 値が含まれる数値式の結果は、すべて NULL 値になります。NULL 値を数値に加算しても結果は NULL 値であり、数値にはなりません。NULL 値を 0 として扱う場合は、**ISNULL(expression, 0)** 関数を使用します。

SQL クエリの作成で生じるエラーの多くは、NULL の性質によるものです。注意して、このような問題を避けるようにしてください。検索条件を組み合わせる場合は 3 値的論理の影響に注意してください。

構文

```
NULL
```

使用法
すべての場所

パーミッション
データベースに接続しておく必要があります。

関連する動作
なし

例
次の **INSERT** 文は、`Borrowed_book` テーブルの `date_returned` カラムに NULL を挿入します。

```
INSERT
INTO Borrowed_book
( date_borrowed, date_returned, book )
VALUES ( CURRENT DATE, NULL, '1234' )
```

参照：
- 比較条件 (36 ページ)
- 式 (24 ページ)
- 検索条件 (35 ページ)
- 文字列 (23 ページ)
- 3 値的論理 (51 ページ)
- SQL 演算子 (27 ページ)
- 検索条件内のサブクエリ (38 ページ)

SQL データ型

SQL データ型によって、格納するデータのタイプ (文字列、数値、日付など) を定義します。

文字データ型

文字データ型は、文字、数字、記号などの文字列を格納するために使用します。

構文

文字データ型には次の構文を使用します。

```
CHAR [ ( max-length ) ]

CHARACTER [ ( max-length ) ]

CHARACTER
                    VARYING [ ( max-length ) ]

VARCHAR [ ( max-length ) ]

UNIQUEIDENTIFIERSTR
```

使用法

次の表は、文字データ型の説明です。

表 9 : 文字データ型

文字データ型	説明
CHAR	最大長 *max-length* バイトの文字データです。*max-length* が指定されなかった場合のデフォルトは 1 で、指定可能な最大サイズは 32KB - 1 です。255 バイトより大きい CHAR データに対する制限については「注意」をご覧ください。 データベース内の文字データ表現と、長い文字列の記憶領域に関する次の注意を参照してください。 CHAR 値では、BLANK PADDING オプションが指定されているかどうかに関わらず、*max-length* までブランクが埋め込まれます。マルチバイト文字の文字列が CHAR 型として格納される場合でも、最大長は文字数ではなくバイト数になります。
CHARACTER	CHAR と同じです。

文字データ型	説明
CHARACTER VARYING	VARCHAR と同じです。
LONG VARCHAR	任意の長さの文字データ。最大サイズは、データベース・ファイルの最大サイズ (現在は 2 ギガバイト) により制限されます。
TEXT	これは、ユーザ定義データ型です。NULL を許可する LONG VARCHAR として実装されます。
VARCHAR	文字列の記憶領域にブランクが追加されないことを除き、CHAR と同じです。VARCHAR 文字列に指定可能な最大長は (32KB - 1) です。255 バイトより大きい VARCHAR データに対する制限については「注意」をご覧ください。
UNIQUEIDENTI-FIERSTR	CHAR(36) として実装されるドメインです。このデータ型は、Microsoft SQL Server の uniqueidentifier 列をマッピングするとき、リモート・データへのアクセスに使用されます。

注意： 別途ライセンスが必要ですが、Sybase IQ オプションがサポートするキャラクタ・ラージ・オブジェクト (CLOB: Character Large Object) データを使用すると、IQ ページのサイズが 128KB であれば 0 ～ 512TB (テラバイト)、また IQ ページが 512KB であれば 0 ～ 2PB (ペタバイト) のサイズを扱うことができます。 最大長は、4GB にデータベース・ページ・サイズを掛けた値になります。 詳細については、『Sybase IQ の非構造化データ分析の概要』を参照してください。

参照：

- バイナリ・データ型 (84 ページ)
- NEWID 関数 [その他] (253 ページ)
- STRTOUUID 関数 [文字列] (328 ページ)
- UUIDTOSTR 関数 [文字列] (351 ページ)
- バイナリ・データ型 (710 ページ)
- 文字データ型 (709 ページ)

記憶領域サイズ

文字データの記憶領域サイズ、所定のカラム定義サイズと入力データ・サイズ。

表 10 : 文字データの記憶領域サイズ

データ型	カラム定義	入力データ	記憶領域
CHARACTER、CHAR	幅 (32K – 1) バイト	(32K – 1) バイト	(32K – 1) バイト

データ型	カラム定義	入力データ	記憶領域
VARCHAR、CHARACTER VARYING	幅 (32K – 1) バイト	(32K – 1) バイト	(32K – 1) バイト

文字セットとコード・ページ

アプリケーションから渡されるバイナリ表現とまったく同じバイナリ表現を使用して、データベースの中に文字データを配置します。

通常、文字データは、システムで使用されている文字セットのバイナリ表現を使用してデータベースの中に格納されます。文字セットに関する説明は、オペレーティング・システムに添付のマニュアルに記載されています。

Windows では、コード・ページの最初の 128 文字は共通です。コード・ページの上位の半分にある特殊文字 (アクセントが付いた国際文字) を使用する場合は、データベースについて注意する必要があります。具体的には、別のコード・ページを使用している異なる種類のマシンにデータベースをコピーする場合に注意が必要です。このとき、特殊文字が元のコード・ページ表現を使用するデータベースから取得されますが、表示には新しいコード・ページが使用されるため、これらの文字はウィンドウ上に異なる文字で表示されます。

2 つのクライアントが、異なるコード・ページを実行する同じマルチユーザ・サーバを使用している場合にも、このような問題が発生します。一方のクライアントから挿入または更新されたデータが、もう一方のクライアントでは正しく表示されない場合があります。

この問題は、データベースが複数のプラットフォームで使用される場合にも発生します。PowerBuilder などの多くの Windows アプリケーションは、標準 ANSI 文字セットでデータベースにデータを挿入します。Windows 以外のアプリケーションがこのデータを使用しようとしても、正しく表示しなかったり、拡張文字を正しく更新しなかったりします。

この問題は非常に複雑です。アプリケーションの上位の半分にあるコード・ページの拡張文字を使用する場合は、データベースを使用するすべてのクライアントとすべてのマシンで、同じまたは互換性のあるコード・ページを使用するようにしてください。

インデックス

DATE、TIME、DTTM を除くすべてのインデックス・タイプを、データ長が 255 バイト以下の CHAR データと VARCHAR データで使用できます。

VARCHAR データと後続ブランク

データ型が VARCHAR のカラムの場合、挿入されるデータ内の後続ブランクは、データが引用符で囲まれているかどうかに応じて異なる方法で処理されます。

INSERT、**UPDATE**、**LOAD TABLE** によって挿入されるデータは、次のいずれかです。

- 引用符で囲まれている
- 引用符で囲まれていない
- バイナリ

データ型が VARCHAR のカラムの場合、挿入されるデータ内の後続ブランクは次のように処理されます。

- 引用符で囲まれたデータの場合、後続ブランクは削除されません。
- 引用符で囲まれていないデータの場合：
  - 後続ブランクは、挿入時と更新時に常に削除されます。
  - **LOAD** 文の場合は、STRIP RTRIM/OFF**LOAD** オプションを使用して、後続ブランクを削除するかどうかを指定できます。STRIP RTRIM/OFF オプションは、可変長の非バイナリ・データにのみ適用されます。たとえば、次のようなスキーマを想定できます。

    ```
    CREATE TABLE t( c1 VARCHAR(3) ); LOAD TABLE t( c1 ',' ) ........
    STRIP RTRIM    // trailing blanks trimmed  LOAD TABLE t( c1
    ',' ) ........ STRIP OFF      // trailing blanks not trimmed
    LOAD TABLE t( c1 ASCII(3) ) ... STRIP RTRIM    // trailing
    blanks not trimmed LOAD TABLE t( c1 ASCII(3) ) ... STRIP
    OFF      // trailing blanks trimmed  LOAD TABLE t( c1
    BINARY ) ..... STRIP RTRIM    // trailing blanks trimmed LOAD
    TABLE t( c1 BINARY ) ..... STRIP OFF      // trailing blanks
    trimmed
    ```

  - バイナリ・データの場合、後続ブランクは常に削除されます。

アプリケーションを作成する際に、VARCHAR カラムにおける後続ブランクの存在に依存しないでください。アプリケーションが後続ブランクに依存している場合は、VARCHAR カラムではなく CHAR カラムを使用してください。

255 バイトを超える CHAR データと VARCHAR データに関する制限

サイズが 255 バイトを超える CHAR カラムと VARCHAR カラムの場合は、デフォルト・インデックス、およびインデックス・タイプ **WD**、**TEXT**、**CMP** のみがサポートされます。

これらのカラムに対して、**LF**、**HG**、**HNG**、**DATE**、**TIME**、**DTTM** インデックスは作成できません。

文字データの互換性

文字データの互換性は、Sybase IQ、Adaptive Server Enterprise、SQL Anywhere でそれぞれ異なります。

- CHARACTER (*n*) を CHAR の代わりに使用することは、Adaptive Server Enterprise ではサポートされていません。
- Sybase IQ は、Adaptive Server Enterprise によって提供される NCHAR、NVARCHAR、UNICHAR、UNIVARCHAR の各データ型をサポートしません。Sybase IQ は、CHAR データ型と VARCHAR データ型への Unicode の格納をサポートします。
- Sybase IQ は、SQL Anywhere よりも長い LONG VARCHAR データ型をサポートします。 詳細については、『Sybase IQ の非構造化データ分析の概要』を参照してください。
- Sybase IQ と Adaptive Server Enterprise の互換性を保つため、文字データ型には必ず長さを指定してください。

長い文字列

254 文字までの値は、短い文字列として保存され、先頭に長さのバイトが付加されます。255 バイトより長い値はすべて、長い文字列と見なされます。255 番目より後ろの文字は、長い文字列値を含むローとは個別に格納されます。

SQL Anywhere は、CHAR、VARCHAR、LONG VARCHAR の各カラムをすべて同じ型として扱います。

255 番目より後ろの文字をすべて無視するいくつかの関数が存在します (「SQL 関数」を参照)。該当するのは、**soundex**、**similar**、およびすべての日付関数です。また、長い文字列を数字に変換する算術演算はすべて、最初の 255 文字にのみ適用されます。こうした制限が問題になるのはきわめてまれです。

他のすべての関数および演算子は、長い文字列の全長を処理します。

数値データ型

数値データ型は、数値データを格納するために使用します。

構文

数値データ型には次の構文を使用します。

```
[ UNSIGNED ] BIGINT
```

```
[ UNSIGNED ] { INT | INTEGER }
```

```
SMALLINT
```

```
TINYINT
```

```
DECIMAL [ ( precision [ , scale ] ) ]
```

```
NUMERIC [ ( precision [ , scale ] ) ]
```

```
DOUBLE
```

```
FLOAT [ ( precision ) ]
```

```
REAL
```

数値データ型の使用法

数値データ型を使用する場合は、次の点に注意してください。

- INTEGER、NUMERIC、DECIMAL のデータ型が真数値データ型とも呼ばれるのに対し、FLOAT、DOUBLE、REAL は概算値データ型と呼ばれます。算術演算後に、指定した最少有効桁数までの正確性が保証されるのは、真数値データだけです。
- TINYINT カラムは、CHAR または UNSIGNED CHAR と定義された Embedded SQL 変数にフェッチしないでください。これは、カラムの値を文字列に変換し、最初のバイトをプログラムの変数に割り当てようとしてしまうためです。
- 小数セパレータ (小数点) はピリオドだけです。カンマは小数セパレータとしてサポートされていません。

表 11 : 数値データ型

数値データ型	説明
BIGINT	記憶領域を 8 バイト必要とする符号付き 64 ビット整数です。 整数を `UNSIGNED` として指定できます。デフォルトでは、このデータ型は符号付きです。整数の範囲は、-9223372036854775808 から 9223372036854775807 の間 (符号付き) または 0 から 18446744073709551615 の間 (符号なし) です。
INT または INTEGER	-2147483648 から 2147483647 の間の値を取る符号付き 32 ビット整数。4 バイトの記憶領域が必要です。 `INTEGER` データ型は真数値データ型です。精度は算術演算の後で保存されます。 整数を UNSIGNED として指定できます。デフォルトでは、このデータ型は符号付きです。符号なし整数値の範囲は、0 から 4294967295 の間です。
SMALLINT	-32768 から 32767 の間の値を取る符号付き 16 ビット整数。2 バイトの記憶領域が必要です。 SMALLINT データ型は真数値データ型です。精度は算術演算の後で保存されます。
TINYINT	0 から 255 の間の値を取る符号なし 8 ビット整数。1 バイトの記憶領域が必要です。 `TINYINT` データ型は真数値データ型です。精度は算術演算の後で保存されます。
DECIMAL	総桁数の *precision* と小数点以下の桁数の *scale* を持つ符号付き 10 進数です。精度は 1 から 126 の値を取ることができ、位取りは 0 から精度の値までを取ることができます。デフォルトでは、位取りが 38 で精度が 126 です。結果はカラムの実際のデータ型に基づき、精度を保って計算されますが、`MAX_CLIENT_NUMERIC_SCALE` オプションを使用すると、アプリケーションに返される結果に対して位取りの最大値を設定することができます。
NUMERIC	`DECIMAL` と同じです。
DOUBLE	8 バイトで格納される符号付きの倍精度浮動小数点数です。ゼロでない絶対値の範囲は、2.2250738585072014e-308 ～ 1.79769313486231570e+308 です。`DOUBLE` として保持される値の有効桁数は厳密には 15 桁です。15 桁を超えると丸め誤差が出る可能性があります。 `DOUBLE` データ型は概数値データ型です。算術演算後に丸め誤差が出ます。

数値データ型	説明
FLOAT	*precision* を指定しない場合、FLOAT データ型は REAL データ型と同じになります。*precision* を指定した場合、FLOAT データ型は、精度の値に応じて、REAL データ型または DOUBLE データ型と同じになります。REAL または DOUBLE のどちらのデータ型と同じになるかは、プラットフォームによって異なります。その境界となるのは、プラットフォームの単精度浮動小数点数の少数部で使用されているビット数です。 FLOAT データ型を使用してカラムを作成した場合、すべてのプラットフォーム上でカラムが少なくとも指定の最小精度の値を持つことが保証されます。対照的に、REAL および DOUBLE ではプラットフォーム非依存の最小精度は保証されません。 FLOAT データ型は概数値データ型です。算術演算後に丸め誤差が出ます。
REAL	4 バイトで格納される符号付きの単精度浮動小数点数。ゼロでない絶対値の範囲は、1.175494351e-38 ～ 3.402823466e+38 です。REAL として保持される値の有効桁数は厳密には 6 桁です。6 桁を超えると丸め誤差が出る可能性があります。 REAL データ型は概数値データ型です。算術演算後に丸め誤差が出ます。

次の表は、10 進数に必要な記憶領域を示します。

表 12 : 10 進数に必要な記憶領域

精度	記憶領域
1 から 4	2 バイト
5 から 9	4 バイト
10 から 18	8 バイト
19 から 126	以下を参照

精度の値が 18 より大きい 10 進数の値に必要とされる記憶領域は、次の式を使用して計算できます。

```
4 + 2 * (int(((prec - scale) + 3) / 4) +
int((scale + 3) / 4) + 1)
```

int は、引数の整数部分です。カラムが使用する記憶領域は、そのカラムの精度と位取りによって決まります。カラム内の各セルは、その精度と位取りの最大値を記憶するのに十分な領域を持っています。次に例を示します。

```
NUMERIC(18,4) takes 8 bytes per cell
NUMERIC(19,4) takes 16 bytes per cell
```

DECIMAL データ型は真数値データ型です。精度は、算術演算の後、最小の有効桁数まで保存されます。その絶対値の最大値は、[*precision* - *scale*] で定義された桁数だけ 9 が並び、次に小数点、その後に *scale* で定義された桁数だけ 9 が続きます。ゼロを除く最小の絶対値は、小数点の後に、[*scale* - 1] の数だけゼロが続き、最後に 1 が 1 つ置かれます。次に例を示します。

```
NUMERIC (3,2) Max positive = 9.99 Min non-zero = 0.01 Max negative =
-9.99
```

NULL から NUMERIC への明示的な変換に対して、精度も位取りも指定しない場合は、デフォルトで NUMERIC(1,0) が設定されます。次に例を示します。

```
SELECT CAST( NULL AS NUMERIC ) A,       CAST( NULL AS NUMERIC(15,2) )
B
```

上の式は次のように記述されます。

```
A NUMERIC(1,0) B NUMERIC(15,2)
```

注意： SQL Anywhere の数値関数でサポートされる最大値は 255 です。数値関数の精度が SQL Anywhere の最大サポート値を超えている場合、次のエラーが表示されます。 `"The result datatype for function '_funcname' exceeds the maximum supported numeric precision of 255. Please set the proper value for precision in numeric function, 'location'"`

数値データの互換性

数値データの互換性は、Sybase IQ、Adaptive Server Enterprise、SQL Anywhere でそれぞれ異なります。

- Embedded SQL の TINYINT カラムは、2 バイトまたは 4 バイト整数にフェッチしてください。また、TINYINT 値をデータベースに送信する場合、C 変数は整数でなければなりません。
- Adaptive Server Enterprise のバージョン 12.5.x は、符号なし整数をサポートしません。 Sybase IQ の符号なし整数は、Adaptive Server Enterprise の符号付き整数または数値データにマップしてください。マップすると、データが自動的に変換されます。
  - Map IQ UNSIGNED SMALLINT データは ASE INT にマップします。
  - 負の値がある場合は、UNSIGNED BIGINT から ASE NUMERIC (*precision*、*scale*) にマップします。
    UNSIGNED BIGINT カラムで、データベース間のジョインを行う際にパフォーマンスが低下するのを避けるため、Sybase IQ の側で (符号付き) BIGINT にキャストすることをおすすめします。
- NUMERIC と DECIMAL データ型は、精度と位取りのデフォルト設定が製品によって異なるため、デフォルトで使用するのは避けてください。以下にその違いを示します。

Database	デフォルトの精度	デフォルトの位取り
Sybase IQ	126	38
Adaptive Server Enterprise	18	0
SQL Anywhere	30	6

- FLOAT (*p*) データ型は、REAL または DOUBLE (*p* の値によって決まる) と同義です。 Adaptive Server Enterprise では、*p* が 15 以下の場合に REAL が使用され、*p* が 15 より大きい場合に DOUBLE が使用されます。Sybase IQ の場合、カットオフ値はプラットフォームによって異なりますが、すべてのプラットフォームでカットオフ値は 22 より大きい値になります。
- Sybase IQ では、MONEY と SMALLMONEY の 2 つのユーザ定義データ型を使用できます。これらはそれぞれ、NUMERIC(19,4) および NUMERIC(10,4) として実装されています。 これらは主に Adaptive Server Enterprise との互換性を保つために用意されています。

インデックス

この項では、インデックス・タイプと数値データ型の関係について説明します。

- **CMP** と **HNG** インデックス・タイプは、FLOAT、DOUBLE、REAL データ型をサポートしていません。また、**HG** インデックス・タイプは使用しないことをおすすめします。
- **WD**、**DATE**、**TIME**、**DTTM** インデックス・タイプは、数値データ型をサポートしていません。

バイナリ・データ型

バイナリ・データ型は、ピクチャなどのロー・バイナリ・データを格納するために使用します。16 進に似た表記で、最大 (32K - 1) バイトまで格納できます。

構文

```
BINARY [ ( length ) ]
```

```
VARBINARY [ ( max-length ) ]
```

```
UNIQUEIDENTIFIER
```

参照：

- NEWID 関数 [その他] (253 ページ)
- STRTOUUID 関数 [文字列] (328 ページ)
- UUIDTOSTR 関数 [文字列] (351 ページ)

- 文字データ型 (75 ページ)
- バイナリ・データ型 (710 ページ)

バイナリ・データ型の使用法

バイナリ・データは文字 "0x" または "0X" から始まり、数字と A ～ F (大文字および小文字) の任意の組み合わせで構成されます。

カラム長はバイト数単位で指定するか、デフォルトの長さの 1 バイトを使用することができます。 各バイトには、2 桁の 16 進数が格納されます。デフォルト長は 1 バイトですが、BINARY カラムと VARBINARY カラムの長さに対して、常に偶数の文字数を指定することをおすすめします。 指定したカラム長より長い値が入力されると、Sybase IQ は入力値を指定された長さまでトランケートします。その際、警告やエラー・メッセージは表示されません。

表 13 : バイナリ・データ型

バイナリ・データ型	説明
BINARY	長さが *length* バイトのバイナリ・データです。*length* を省略した場合は、デフォルトで 1 バイトに設定されます。 指定可能な最大サイズは 32767 バイトです。 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 を使用します。 BINARY カラムのエントリには、*length* カラム長まで 0 が埋め込まれるので、VARBINARY カラムのエントリよりも多くの記憶領域が必要になる場合があります。
VARBINARY	最大長 *max-length* バイトのバイナリ データです。*max-length* を指定しない場合は、デフォルトで 1 バイトが設定されます。指定可能な最大サイズは (32K - 1) バイトです。長さが大きく変化することが予測されるデータには、可変長バイナリ型の VARBINARY を使用してください。
UNIQUEIDENTIFIER	UNIQUEIDENTIFIER データ型は、UUID (GUID とも呼ばれます) 値の記憶領域として使用されます。

後続ゼロの扱い

BINARY 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。すべての VARBINARY カラムで、後続ゼロがトランケートされます。

次の例では、BINARY データ型と VARBINARY データ型に、それぞれ NULL と NOT NULL を定義した 4 種類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,
```

```
        vbnot VARBINARY(5) NOT NULL,
        vbnull VARBINARY(5) NULL);
INSERT zeros VALUES (0x12345000, 0x12345000,
        0x12345000, 0x12345000);
INSERT zeros VALUES (0x123, 0x123, 0x123, 0x123);
INSERT zeros VALUES (0x0, 0x0, 0x0, 0x0);
INSERT zeros VALUES ('002710000000ae1b',
'002710000000ae1b', '002710000000ae1b',
'002710000000ae1b');
SELECT * FROM zeros;
```

bnot	bnull	vbnot	vbnull
0x1234500000	0x1234500000	0x12345000	0x12345000
0x0123000000	0x0123000000	0x0123	0x0123
0x0000000000	0x0000000000	0x00	0x00
0x3030323731	0x3030323731	0x3030323731	0x3030323731

記憶領域の各バイトによってそれぞれ 2 桁の 16 進数が保持されるため、Sybase IQ は文字列 "0x" の後に偶数個の数字が続くバイナリ・エントリを受け入れます。"0x" の後に奇数個の数字が続く場合、Sybase IQ 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。

"0x00" と "0x0" の入力値は、可変長バイナリ・カラム (VARBINARY) に "0x00" として格納されます。固定長バイナリ・カラム (BINARY) では、値はフィールド幅いっぱいまでゼロが埋め込まれます。

```
INSERT zeros VALUES (0x0, 0x0, 0x0, 0x0); SELECT * FROM zeros
```

bnot	bnull	vbnot	vbnull
0x0000000000	0x0000000000	0x00	0x00

入力値に "0x" がない場合、Sybase IQ はその値が ASCII 値であると想定して、値を変換します。 次に例を示します。

```
CREATE TABLE sample (col_bin BINARY(8));
INSERT sample VALUES ('002710000000ae1b');
SELECT * FROM sample;
```

col_bin
0x3030323731303030

注意： 上記の例では、`string_rtruncation` オプションを "off" に設定しておいてください。

BINARY 値を選択する場合は、ゼロを埋め込んで値を指定するか、**CAST** 関数を使用する必要があります。次に例を示します。

```
SELECT * FROM zeros WHERE bnot = 0x0123000000;
```

または

```
SELECT * FROM zeros WHERE bnot = CAST(0x0123 as binary(5));
```

フラット・ファイルの ASCII データ

フラット・ファイルからバイナリ型のカラム (BINARY または VARBINARY) にロードされた ASCII データは、必ずニブルで格納されます。

たとえば、フラット・ファイルからバイナリ・カラムに 0x1234 または 1234 が読み込まれた場合、Sybase IQ は 16 進数の 1234 として値を格納します。Sybase IQ は先頭の "0x" を無視します。 入力データに 0 ～ 9、a ～ f、A ～ F 以外の文字が含まれる場合、そのデータは拒否されます。

記憶領域サイズ

バイナリ・データの記憶領域サイズをよく理解してください。

表 14 : バイナリ・データの記憶領域サイズ

データ型	カラム定義	入力データ	記憶領域
VARBINARY	幅 (32K – 1) バイト	(32K – 1) バイトのバイナリ	(32K – 1) バイト
VARBINARY	幅 (32K – 1) バイト	(64K – 2) バイトの ASCII	(32K – 1) バイト
BINARY	幅 (32K – 1) バイト	(32K – 1) バイトのバイナリ	(32K – 1) バイト
BINARY	幅 (32K – 1) バイト	(64K – 2) バイトの ASCII	(32K – 1) バイト

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。

16 進数の文字列と整数との変換において、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 **CONVERT** 関数ではなく、**INTTOHEX** 関数と **HEXTOINT** 関数を使用します。

参照：

- データ型変換関数 (115 ページ)
- データ型変換 (100 ページ)

文字列演算子

文字列を連結する演算子である || と + は、どちらもバイナリ・データ型をサポートします。

|| 演算子を使用する場合、バイナリ・オペランドから文字データ型への明示的な変換は必要ありません。ただし、明示的なデータ変換と暗示的なデータ変換とでは、結果に違いが生じます。

BINARY データと VARBINARY データに対する制限

BINARY データや VARBINARY データを含むカラムには、次のような制限があります。

- 集合関数 **SUM**、**AVG**、**STDDEV**、**VARIANCE** では、バイナリ・データ型を使用できません。集合関数 **MIN**、**MAX**、**COUNT** は、バイナリ・データ型 BINARY と VARBINARY をサポートしています。
- **HNG**、**WD**、**DATE**、**TIME**、**DTTM** インデックスは、BINARY データや VARBINARY データをサポートしていません。
- サイズが 255 バイトより大きい BINARY データと VARBINARY データの場合は、デフォルト・インデックス (**CMP** インデックス) と **TEXT** インデックス・タイプだけがサポートされます。
- ビット操作は、8 バイト以下の BINARY データと VARBINARY データでサポートされます。

バイナリ・データの互換性

バイナリ・データ型での後続ゼロの扱いは、Sybase IQ、SQL Anywhere、Adaptive Server Enterprise でそれぞれ異なります。

表 15 : 後続ゼロの扱い

データ型	Sybase IQ	SQL Anywhere	Adaptive Server Enterprise
BINARY NOT NULL	埋め込みあり	埋め込みなし	埋め込みあり
BINARY NULL	埋め込みあり	埋め込みなし	埋め込みなし
VARBINARY NOT NULL	トランケートする、埋め込みなし	トランケートする、埋め込みなし	トランケートする、埋め込みなし
VARBINARY NULL	トランケートする、埋め込みなし	トランケートする、埋め込みなし	トランケートする、埋め込みなし

Adaptive Server Enterprise、SQL Anywhere、Sybase IQ はいずれも、**STRING_RTRUNCATION** データベース・オプションをサポートします。このオプ

ションを使用すると、**INSERT** 文字列または **UPDATE** 文字列がトランケートされたとき表示されるエラー・メッセージに影響が出ます。 Transact-SQL と互換性のある文字列比較を行う場合は、双方のデータベースで **STRING_RTRUNCATION** オプションを同じ値にセットしてください。

また、データをテーブルにロードする際に **STRING_RTRUNCATION** オプションを ON にすると、データが大きすぎてフィールドにロードできない場合に警告を表示させることができます。デフォルト値は ON です。

バイナリ型でのビット操作は Adaptive Server Enterprise ではサポートされていません。SQL Anywhere はバイナリ型データの先頭の 4 バイトに対してのみ、ビット操作をサポートします。Sybase IQ はバイナリ型データの先頭の 8 バイトに対して、ビット操作をサポートします。

UNIQUEIDENTIFIER

`UNIQUEIDENTIFIER` データ型は、UUID (GUID とも呼ばれます) 値の記憶領域として使用されます。

多くの場合、`UNIQUEIDENTIFIER` データ型は、プライマリ・キーやその他のユニーク・カラムで、ローを一意に識別する UUID (Universally Unique Identifier) 値を保持するために使用されます。**NEWID** 関数は UUID 値を、1 台のコンピュータで生成した UUID 値が別のコンピュータで生成した UUID 値と一致しないように生成します。したがって、**NEWID** を使用して生成された `UNIQUEIDENTIFIER` 値は、同期環境でキーとして使用できます。

たとえば、次の文は、テーブル `mytab` を更新し、カラム `uid_col` の現在値が NULL の場合は、その値に **NEWID** 関数で生成された一意の識別子を設定します。

```
UPDATE mytab
    SET uid_col = NEWID()
      WHERE uid_col IS NULL
```

次の文を実行します。

```
SELECT NEWID()
```

一意の識別子が `BINARY(16)` として返されます。たとえば、値は 0xd3749fe09cf446e399913bc6434f1f08 になります。この文字列を **UUIDTOSTR**() 関数を使用して読みやすい形式に変換できます。

UUID は、GUID (Globally Unique Identifier) とも呼ばれます。

STRTOUUID 関数と **UUIDTOSTR** 関数を使用して、`UNIQUEIDENTIFIER` と文字列表記の間で値を変換できます。

`UNIQUEIDENTIFIER` 値は `BINARY(16)` として格納され、返されます。

UNIQUEIDENTIFIER 値は大きいので、データベース間に一意の識別子が必要ない場合は、UNIQUEIDENTIFIER の代わりに、UNSIGNED BIGINT または UNSIGNED INT identity カラムを使用する方がより効率的です。

UNIQUEIDENTIFIER の標準と互換性

UNIQUEIDENTIFIER の値に関する標準と互換性は、次のとおりです。

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — SQL Anywhere によるサポート有り。Adaptive Server Enterprise によるサポートなし。
- 下位互換性 — Sybase IQ バージョン 12.7 より前に作成されたデータベースでは、**STRTOUUID**、**UUIDTOSTR**、**NEWID** の関数が CIS 補正機能を通じてサポートされていました。 バージョン 15.3 以降では、**STRTOUUID**、**UUIDTOSTR**、**NEWID** 関数は、Sybase IQ のネイティブな関数です。

バイナリ・ラージ・オブジェクト・データ

別途ライセンスが必要ですが、Sybase IQ オプションがサポートするバイナリ・ラージ・オブジェクト (BLOB: Binary Large Object) データを使用すると、IQ ページのサイズが 128KB であれば 0 ～ 512TB (テラバイト)、また IQ ページが 512KB であれば 0 ～ 2PB (ペタバイト) のサイズを扱うことができます。

最大長は、4GB にデータベース・ページ・サイズを掛けた値になります。 詳細については、『Sybase IQ の非構造化データ分析の概要』を参照してください。

Bit データ型

BIT データ型は、ブール値の格納に使用します。

データ型	値	サポート
BIT	0 または 1	Sybase IQ および Adaptive Server Enterprise

使用法

BIT は 0 または 1 のみ格納します。

BIT カラムにゼロ以外の値を挿入すると、1 が格納されます。BIT カラムに値ゼロを挿入すると、0 が格納されます。

BIT データに対しては、デフォルトのインデックス・タイプだけがサポートされています。

Bit データの互換性

Adaptive Server Enterprise の `BIT` データ型では、0 または 1 の値のみを扱うことができます。

日付と時刻のデータ型

日付と時刻のデータ型は、日付と時刻の格納に使用されます。

構文

日付と時刻のデータ型には次の構文を使用します。

DATE

DATETIME

SMALLDATETIME

TIME

TIMESTAMP

参照：

- TIMESTAMP 特別値 (62 ページ)
- CURRENT TIMESTAMP 特別値 (60 ページ)
- CURRENT TIME 特別値 (59 ページ)
- CURRENT DATE 特別値 (59 ページ)
- 日付と時刻の取得 (94 ページ)
- BIGTIME と BIGDATETIME のサポート (712 ページ)

日付と時刻のデータ型の使用法

日付と時刻のデータ型は、次の使用上の考慮事項をよく理解してから使用してください。

表 16 :

日付と時刻のデータ型	説明
DATE	年、月、日などの暦日です。年の範囲は 0001 ～ 9999 です。日がゼロの値を持つことはできません。したがって、最小値は 0001-01-01 になります。DATE 値には 4 バイトの記憶領域が必要です。
DATETIME	TIMESTAMP として実装されたドメイン。DATETIME は主に、Adaptive Server Enterprise との互換性を保つために用意されています。
SMALLDATETIME	TIMESTAMP として実装されたドメイン。SMALLDATETIME は主に、Adaptive Server Enterprise との互換性を保つために用意されています。
時刻	時、分、秒、秒以下で構成される時間です。秒の小数点以下は 6 桁まで格納されます。TIME 値には 8 バイトの記憶領域が必要です。(ODBC 規格では、TIME データ型の精度を秒単位までに制限しています。そのため、**WHERE** 句の比較に、秒単位よりも高い精度に基づく TIME データ型を使用しないでください)。
TIMESTAMP	年、月、日、時、分、秒、秒以下で構成される時刻です。秒の小数点以下は 6 桁まで格納されます。日にはゼロでない値を格納してください。TIMESTAMP 値には 8 バイトの記憶領域が必要です。

TIMESTAMP データ型の有効な値の範囲は、0001-01-01 00:00:00.000000 から 9999-12-31 23:59:59.999999 です。1600-02-28 23:59:59 から 7911-01-01 00:00:00 の範囲外の TIMESTAMP データは表示が不完全になりますが、データベースには正確な日時の値が格納されます。最初にデータを文字列に変換することにより、正確な値を表示することができます。それを行うには、**CAST()** 関数を以下の例のように使用します。この例では、最初に DATETIME カラムと TIMESTAMP カラムを持つテーブルを作成し、日付が 7911-01-01 より大きい値を挿入します。

```
create table mydates (id int, descript char(20),
  datetime_null datetime, timestamp_null timestamp);
```

```
insert into mydates values (1, 'example', '7911-12-30
  23:59:59','7911-12-30 06:03:44');
commit;
```

CAST を使用せずに select を実行すると、時と分が 00:00: にセットされます。

```
select * from mydates;

1, 'example', '7911-12-30 00:00:59.000', '7911-12-30
00:00:44.000'
```

キャストを使用して select を実行すると、正しいタイムスタンプが表示されます。

```
select id, descript, cast(datetime_null as char(21)),
  cast(timestamp_null as char(21)) from mydates;

1, 'example', '7911-12-30 23:59:59.0', '7911-12-30 06:03:44.0'
```

参照：

- 文字列から日付／時刻への変換の互換性 (101 ページ)

サポートされているインデックス・タイプ

日付と時刻のデータでは、以下のインデックス・タイプがサポートされています。

- 日付と時刻のデータ型はすべて、**CMP**、**HG**、**HNG**、**LF** インデックス・タイプをサポートしています。**WD** インデックス・タイプはサポートされていません。
- DATE データは、**DATE** インデックスをサポートしています。
- TIME データは、**TIME** インデックスをサポートしています。
- DATETIME と TIMESTAMP データは、**DTTM** インデックスをサポートしています。

日付と時刻の送信

次のいずれかの方法で、日付と時刻をデータベースに送信します。

- インタフェースを使うときは、文字列として
- ODBC を使用し、TIMESTAMP 構造体として
- Embedded SQL を使用し、SQLDATETIME 構造体として

データベースに時刻を文字列として (TIME データ型の場合)、または文字列の一部として (TIMESTAMP または DATE データ型の場合) 送信する場合は、時間、分、秒をコロンで区切り、*hh*:*mm*:*ss*.*sss* のフォーマットにする必要はありますが、文字列中のどこに置いてもかまいません。オプションとして、*hh*:*mm*:*ss*.*sss* のように秒と小数点以下の秒をピリオドで区切ることができます。以下は、有効かつ明確に時刻を指定するための文字列です。

```
21:35 -- 24 hour clock if no am or pm specified
10:00pm -- pm specified, so interpreted as 12 hour clock
10:00 -- 10:00am in the absence of pm
10:23:32.234 -- seconds and fractions of a
                    second included
```

データベースに日付を文字列として送信する場合、日付の変換は自動的に行われます。次の 2 つの方法のいずれかで、文字列を送信します。

- データベースで確実に解釈される *yyyy/mm/dd* または *yyyy-mm-dd* フォーマットの文字列として
- DATE_ORDER データベース・オプションに従って解釈される文字列として

日付フォーマットの文字列に、マルチバイト文字を格納することはできません。日付、時刻、日時のフォーマットの文字列に格納できるのは、シングルバイト文字だけです。これはデータベースの照合順序が、932JPN のようにマルチバイトの照合順序である場合も同じです。

日付と時刻の取得

次のいずれかの方法で、日付と時刻をデータベースから取得します。

- インタフェースを使うときは、文字列として
- ODBC を使用し、TIMESTAMP 構造体として
- Embedded SQL を使用し、SQLDATETIME 構造体として

使用法

日付や時刻を文字列として取得する場合は、データベース・オプション DATE_FORMAT、TIME_FORMAT、TIMESTAMP_FORMAT によって指定されているフォーマットを使用します。

日付については次の演算子が許可されています。

表 17 : 演算子

演算子	説明
timestamp + integer	指定された値の日数を日付またはタイムスタンプに加えます。
timestamp - integer	指定された値の日数を日付またはタイムスタンプから引きます。
date - date	2 つの日付 (タイムスタンプ) 間の日数を計算します。
date + time	与えられた日付と時刻を結合するタイムスタンプを作成する。

参照：

- TIMESTAMP 特別値 (62 ページ)
- CURRENT TIMESTAMP 特別値 (60 ページ)
- CURRENT TIME 特別値 (59 ページ)
- CURRENT DATE 特別値 (59 ページ)
- 日付と時刻のデータ型 (91 ページ)

日付と時刻の比較

日付を文字列として比較する場合は、**DATEFORMAT** 関数または **CAST** 関数によって日付を文字列に変換してから比較します。

使用法

```
DATEFORMAT(invoice_date,'yyyy/mm/dd') = '1992/05/23'
```

DATEFORMAT の文字列式には、許可されている日付フォーマットをすべて使用できます。

日付フォーマットの文字列に、マルチバイト文字を格納することはできません。日付、時刻、日時のフォーマットの文字列に格納できるのは、シングルバイト文字だけです。これはデータベースの照合順序が、932JPN のようにマルチバイトの照合順序である場合も同じです。

次の *'?* がマルチバイト文字を表す場合、このクエリはエラーになります。

```
SELECT DATEFORMAT ( StartDate, ‘yy?’) FROM Employees;
```

代わりに、連結演算子を使用して、マルチバイト文字を日付フォーマットの文字列の外に移動します。

```
SELECT DATEFORMAT (StartDate, ‘yy’) + ‘?’ FROM Employees;
```

あいまいさのない日付と時刻

あいまいさのない日付フォーマットを使用することにより、ユーザの DATE_ORDER 設定に従った日付が間違って解釈されるのを防ぐことができます。

使用法

yyyy/mm/dd または *yyyy-mm-dd* フォーマットの日付は、DATE_ORDER の設定に関係なく、常に日付として認識されます。区切り文字として他の文字 (疑問符、スペース文字、カンマなど) を使用することもできます。各ユーザの DATE_ORDER 設定が異なる可能性がある場合は、必ずこのフォーマットを使用してください。たとえば、ストアド・プロシージャでは、あいまいさのない日付フォーマットを使用することにより、ユーザの DATE_ORDER 設定に従った日付が間違って解釈されるのを防ぐことができます。

また、*hh:mm:ss:sss* のフォーマットの文字列は、あいまいさのない時刻として解釈されます。

日付と時刻を組み合わせる場合は、あいまいさのない日付と時刻を組み合わせることにより、結果的にあいまいさのない日付／時刻として評価されます。次の形式も、あいまいさのない日付／時刻値です。

```
YYYY-MM-DD HH.MM.SS.SSSSSS
```

ピリオドは、日付と組み合わせた時刻に対してのみ使用できます。

そのほかの場合にも、非常に柔軟な日付フォーマットを使用できます。Sybase IQ では、さまざまな文字列をフォーマットとして解釈することが可能です。 この解釈は、DATE_ORDER データベース・オプションに左右されます。DATE_ORDER データベース・オプションには、‘MDY’、‘YMD’、‘DMY’ の値を指定できます。たとえば、DATE_ORDER オプションを 'DMY' に設定するには、次のように入力します。

```
SET OPTION DATE_ORDER = 'DMY' ;
```

デフォルトの DATE_ORDER 設定は、'YMD' です。ODBC ドライバは、接続されるときに必ず DATE_ORDER オプションを 'YMD' に設定します。値を変更するには、**SET OPTION** 文を使用します。

データベース・オプション DATE_ORDER は、文字列 10/11/12 が、データベースで 1912 年 10 月 11 日、1910 年 11 月 12 日、または 1912 年 11 月 10 日のいずれとして解釈されるかを決定します。日付文字列の年、月、日を記号 (/、-、スペースなど) で区切ると、DATE_ORDER オプションで指定されている順序で表示されます。

年は、2 桁か 4 桁で入力できます。NEAREST_CENTURY オプション [TSQL] の値によって、2 桁の年の解釈が変わります。NEAREST_CENTURY よりも小さい値には 2000 が追加され、それ以外の値には 1900 が追加されます。このオプションのデフォルト値は 50 です。したがって、デフォルトでは、50 は 1950、49 は 2049 と解釈されます。

月は月の名前または数字です。時間と分はコロンで区切りますが、文字列のどこにでも置けます。

Sybase では、年には常に 4 桁のフォーマットを使用して指定することを推奨しています。

DATE_ORDER の適切な設定を使用した次の文字列は、すべて有効な日付です。

```
99-05-23 21:35
99/5/23
1999/05/23
May 23 1999
23-May-1999
Tuesday May 23, 1999 10:00pm
```

文字列に部分的な日付指定だけがある場合、デフォルト値を使って日付が満たされます。次のデフォルトを使います。

- 年 ― 1900
- 月 ― デフォルトなし
- 日 ― 1 (月のフィールドで役に立ちます。たとえば、1999 年 5 月は、日付 1999-05-01 00:00 となります)
- 時、分、秒、秒以下 ― 0

ドメイン

ドメインは、必要に応じて精度と小数点以下の桁数を含めた組み込みデータ型のエイリアスです。

ドメインはユーザ定義データ型とも呼ばれ、データベース全体を通して、カラムを同じ NULL または NOT NULL 条件を持つ同じデータ型に自動的に定義できます。 これにより、データベース全体の一貫性が高まります。 ドメイン名の大文字と小文字は区別されません。大文字と小文字の表記だけが異なる既存のドメインと同じ名前のドメインを作成しようとすると、Sybase IQ はエラーを返します。

単純なドメイン

ドメインは CREATE DOMAIN 文を使用して作成します。

次の文は street_address という名前の、35 文字の文字列のデータ型を作成します。

```
CREATE DOMAIN street_address CHAR( 35 )
```

CREATE DOMAIN の代わりに、**CREATE DATATYPE** を使用することもできますが、**CREATE DOMAIN** は ISO/ANSI SQL 標準で使用されている構文なので、こちらを使用することをおすすめします。

データ型を作成するには RESOURCE 権限が必要です。データ型を一度作成すると、**CREATE DOMAIN** 文を実行したユーザ ID がそのデータ型の所有者となります。このデータ型はすべてのユーザが使用できます。他のデータベース・オブジェクトとは異なり、所有者名をデータ型のプレフィックスとして使用しません。

カラムを定義する場合は、street_address データ型を他のデータ型とまったく同じように使用します。たとえば、2 つのカラムがある次のテーブルでは、2 番目のカラムが street_address カラムです。

```
CREATE TABLE twocol (id INT,
street street_address)
```

所有者または DBA は、**COMMIT** を発行してから **DROP DOMAIN** 文を使用して、ドメインを削除できます。

```
DROP DOMAIN street_address
```

データベース内のどのテーブルもデータ型を使用していない場合にのみ、この文を実行できます。

ユーザ定義データ型に関する制約とデフォルト
NULL 値の許可、DEFAULT 値の設定など、カラムに関連する属性の多くは、ユーザ定義データ型に組み込むことができます。データ型で自動的に定義されるカラムはすべて、NULL 設定、CHECK 条件、および DEFAULT 値を継承します。これにより、データベース全体にわたって同様の意味を持つ一貫したカラムを作成できます。

たとえば、デモ・データベース内の多くのプライマリ・キー・カラムは ID 番号を持つ整数カラムです。次の文は、こうしたカラムで役立つデータ型を作成します。

```
CREATE DOMAIN id INT
NOT NULL
DEFAULT AUTOINCREMENT
CHECK( @col > 0 )
```

データ型 ID を使用して作成されたカラムはすべて、NULL を持つことができず、デフォルトで自動インクリメント値になります。また、正の数を持つ必要があります。*@col* 変数では、*col* の代わりに任意の ID を使用できます。

カラムに対して明示的に属性を指定することにより、必要に応じてデータ型の属性を上書きできます。あるデータ型 ID で作成したカラムで明示的に NULL 値を許可した場合、その ID データ型の設定にかかわらず、NULL が許可されます。

CREATE DOMAIN 文

データベースにユーザ定義データ型を作成します。

構文

```
CREATE { DOMAIN | DATATYPE } domain-name
            data-type
... [ NOT ] NULL ]
... [ DEFAULT
            default-value ]
```

パラメータ

- **domain-name：** – 識別子
- **data-type：** – 精度と位取りを指定した組み込みデータ型
- **default-value：** – *special-value* | *string* | *global variable* | [-] *number* | (*constant-expression*) | *built-in-function*(*constant-expression*) | **AUTOINCREMENT** | **CURRENT DATABASE** | **CURRENT REMOTE USER** | **NULL** | **TIMESTAMP** | **LAST USER**
- **special-value：** – **CURRENT** { **DATE** | **TIME** | **TIMESTAMP** | **USER** | **PUBLISHER** } | **USER**

例

- **例 1 –** 次の文は、35 文字の文字列を保持し、NULL が使用できる **address** という名前のデータ型を作成します。

```
CREATE DOMAIN address CHAR( 35 ) NULL
```

使用法

ユーザ定義データ型は、必要に応じて精度と位取りを含めた組み込みデータ型のエイリアスです。データベース内の使いやすさを改善し、一貫性を高めます。

CREATE DOMAIN は ANSI/ISO SQL3 用語なので、**CREATE DATATYPE** ではなく、**CREATE DOMAIN** を使用することをおすすめします。

データ型を作成するユーザは、自動的にそのデータ型の所有者となります。**CREATE DATATYPE** 文の中では、所有者を指定できません。ユーザ定義型の名前はユニークにする必要があります。また、すべてのユーザはプレフィクスとして所有者を使わなくてもデータ型にアクセスできます。

ユーザ定義データ型は、データベースの中のオブジェクトです。名前を付けるときは識別子の規則に従う必要があります。ユーザ定義データ型名の大文字と小文字は常に区別されません。組み込みデータ型名の場合と同じです。

デフォルトでユーザ定義データ型が NULL を使用するのは、**allow_nulls_by_default** オプションが OFF に設定されていないときです。この場合、新しいユーザ定義データ型は、デフォルトで NULL を使いません。ユーザ定義データ型に作成したカラムの NULL 入力可能性は、そのカラムを参照するときの **allow_nulls_by_default** オプションの設定ではなく、ユーザ定義データ型の定義での設定に応じて異なります。カラム定義の中で NULL または NOT NULL を明示的に設定すると、ユーザ定義データ型設定は上書きされます。

CREATE DOMAIN 文を使用して、ユーザ定義データ型の DEFAULT 値を指定できます。DEFAULT 値の指定は、そのデータ型で定義されたすべてのカラムに継承されます。そのカラムに対して明示的に指定したすべての DEFAULT 値は、データ型に対して指定したものよりも優先されます。カラムの DEFAULT 値の使用の詳細については、『システム管理ガイド：第 1 巻』の「データ整合性」>「カラムのデフォルトを使用したデータ整合性の向上」を参照してください。

CREATE DOMAIN 文では、CHECK 条件と呼ばれる規則を、ユーザ定義データ型の定義に組み込むことができます。

Sybase IQ では、ベース・テーブル、グローバル・テンポラリ・テーブル、ローカル・テンポラリ・テーブル、ユーザ定義のデータ型に CHECK 制約を適用します。

データベースからデータ型を削除するには **DROP** 文を使います。ユーザ定義データ型を削除するには、データ型の所有者になるか DBA 権限を持っている必要があります。

『リファレンス： ビルディング・ブロック、テーブル、およびプロシージャ』の「SQL データ型」を参照してください。

関連する動作：

- オートコミット。

標準

- SQL—ISO/ANSI SQL 準拠。
- Sybase — Adaptive Server Enterprise によるサポートなし。Transact-SQL は、**sp_addtype** システム・プロシージャ、**CREATE DEFAULT** 文、**CREATE RULE** 文を使用して、同様の機能を提供しています。

パーミッション

RESOURCE 権限が必要です。

ドメインの互換性

ドメインの互換性は、Sybase IQ、Adaptive Server Enterprise、SQL Anywhere でそれぞれ異なります。

- 名前付き制約と名前付きデフォルト ― Sybase IQ では、ユーザ定義データ型はベース・データ型で作成され、オプションとして NULL または NOT NULL 条件も使用できます。 名前付き制約と名前付きデフォルトはサポートしません。
- データ型の作成 — Sybase IQ では、`sp_addtype` システム・プロシージャまたは **CREATE DOMAIN** 文を使用してドメインを追加できます。 Adaptive Server Enterprise では `sp_addtype` を使用します。`sp_addtype` の所有者、および SQL Anywhere から継承した他のストアド・プロシージャの所有者は dbo です。SQL Anywhere ストアド・プロシージャを使用して作成されたオブジェクトの作成者も dbo です。このため、DBA 権限のないユーザは、`sp_addtype` を使用して作成されたドメインを変更または削除できません。**CREATE DOMAIN** で作成されたドメインを変更または削除するには、DBA 権限が必要です。

データ型変換

型変換は自動的に発生することもあれば、**CAST** または **CONVERT** 関数を使用した明示的な型変換が必要になることもあります。

使用法

文字列が数値式で使用されているか、または数値引数の想定される関数への引数として使用されている場合、文字列は使用前に数値に変換されます。

数値が文字列式で使用されているか、または文字列関数の引数として使用されている場合、数値は使用前に文字列に変換されます。

すべての日付定数は文字列として指定されます。文字列は、自動的に日付に変換されてから使用されます。

自動的なデータ型変換が適切でない場合があります。

```
'12/31/90' + 5 -- Tries to convert the string to a number
'a' > 0        -- Tries to convert 'a' to a number
```

型変換を強制的に実行するには、**CAST** または **CONVERT** 関数を使用します。

以下の関数は、型変換を強制的に行うために使用することができます。

- **DATE(expression)** ― 式を日付に変換し、時間、分、秒を削除します。変換エラーがあればレポートします。
- **DATETIME(expression)** ― 式をタイムスタンプに変換します。変換エラーがあればレポートします。
- **STRING(expression)** ― `CAST(value AS CHAR)` に類似していますが、`CAST(NULL AS CHAR)` が NULL 値であるのに対して、`string(NULL)` は空の文字列 ('') です。

参照：
- データ型変換関数 (115 ページ)
- 記憶領域サイズ (87 ページ)

文字列から日付／時刻への変換の互換性

文字列を日付データ型や時刻データ型に変更する場合、Sybase IQ と Adaptive Server Enterprise の動作にはいくつかの相違があります。

時刻値だけ (日付なし) の文字列を日付／時刻データ型に変換する場合、Sybase IQ と Adaptive Server Enterprise はデフォルトの日付「1900 年 1 月 1 日」を使用しますが、SQL Anywhere は現在の日付を使用します。

使用法

時刻のミリ秒の部分が 3 桁より小さい場合、前にピリオドが付くかコロンが付くかによって、Adaptive Server Enterprise による値の解釈方法は違います。 前にコロンが付く場合、値は 1000 分の 1 秒を意味します。 前にピリオドが付く場合は、1 桁であれば 10 分の 1、2 桁であれば 100 分の 1、3 桁であれば 1000 分の 1 秒を意味します。Sybase IQ と SQL Anywhere は、区切り文字に関係なく、値を同じ方法で解釈します。

- Adaptive Server Enterprise は、以下のように値を変換します。

```
12:34:56.7 to 12:34:56.700
12.34.56.78 to 12:34:56.780
12:34:56.789 to 12:34:56.789
12:34:56:7 to 12:34:56.007
12.34.56:78 to 12:34:56.078
12:34:56:789 to 12:34:56.789
```

- Sybase IQ は、どちらの場合も、Adaptive Server Enterprise でピリオドが前についた値を変換するのと同じ方法で、ミリ秒の値を変換します。

```
12:34:56.7 to 12:34:56.700
12.34.56.78 to 12:34:56.780
12:34:56.789 to 12:34:56.789
12:34:56:7 to 12:34:56.700
12.34.56:78 to 12:34:56.780
12:34:56:789 to 12:34:56.789
```

参照：

- 日付と時刻のデータ型の使用法 (92 ページ)

エクスポートされた日付の互換性

月の 9 日目まで、および 10 未満の時間の値に対して、Adaptive Server Enterprise は 1 桁目のブランクをサポートし、Sybase IQ はゼロまたはブランクをサポートします。

対象のデータを Adaptive Server Enterprise から Sybase IQ にロードする方法の詳細については、『システム管理ガイド：第 1 巻』の「データのインポートとエクスポート」を参照してください。

BIT から BINARY へのデータ型の変換

Sybase IQ は、BIT から BINARY 、および BIT から VARBINARY への暗黙的および明示的な変換をサポートしています。これらの変換は、Adaptive Server Enterprise でサポートされている変換と互換性があります。

Sybase IQ は、比較演算子と算術演算子および **INSERT** 文と **UPDATE** 文について、BIT から BINARY へ、および BIT から VARBINARY へと暗黙的にデータ型を変換します。

BIT から BINARY への変換では、ビット値 'b' がバイナリ文字列の最初のバイトにコピーされ、残りのバイトに 0 が入力されます。たとえば、ビット 値 1 は、2^n のニブルを持つ BINARY(n) 文字列 0x0100...00 に変換されます。ビット 値 0 は BINARY 文字列 0x00...00 に変換されます。

BIT から VARBINARY への変換では、ビット 値 'b' が BINARY 文字列の最初のバイトにコピーされ、残るバイトは使用されません。つまり、1 バイトのみが使用さ

れます。たとえば、ビット 値 1 は、2 個のニブルを持つ VARBINARY(n) 文字列 0x01 に変換されます。

BIT データ型から BINARY データ型、および BIT データ型から VARBINARY データ型への変換は、暗黙的な変換も明示的な変換も同じ結果になります。次の表に、BIT から BINARY および VARBINARY への変換の例を示します。

ビット値 '1' を下記に変換	結果
BINARY(3)	0x010000
VARBINARY(3)	0x01
BINARY(8)	0x0100000000000000
VARBINARY(8)	0x01

以下の例は、BIT データ型から BINARY データ型、および BIT データ型から VARBINARY データ型への暗黙的および明示的な変換を示しています。

次のようなテーブルとデータがあるとします。

```
CREATE TABLE tbin(c1 BINARY(9))
CREATE TABLE tvarbin(c2 VARBINARY(9))
CREATE TABLE tbar(c2 BIT)

INSERT tbar VALUES(1)
INSERT tbar VALUES(0)
```

BIT から BINARY への暗黙的な変換：

```
INSERT tbin SELECT c2 FROM tbar

c1
---
0x010000000000000000   (18 nibbles)
0x000000000000000000   (18 nibbles)
```

BIT から VARBINARY への暗黙的な変換：

```
INSERT tvarbin SELECT c2 FROM tbar

c2
---
0x01
0x00
```

BIT から BINARY への明示的な変換：

```
INSERT tbin SELECT CONVERT (BINARY(9), c2) FROM tbar

c1
---
0x010000000000000000   (18 nibbles)
0x000000000000000000   (18 nibbles)
```

BIT から VARBINARY への明示的な変換：

```
INSERT tvarbin SELECT CONVERT(VARBINARY(9), c2) FROM tbar

c2
---
0x01 0x00
```

BIT と CHAR/VARCHAR 間のデータ型変換

Sybase IQ は、比較演算子と算術演算子および **INSERT** 文と **UPDATE** 文について、BIT から CHAR へ、および BIT から VARCHAR への暗黙的なデータ型の変換をサポートしています。

以下の例は、BIT データ型と CHAR データ型、および BIT データ型と VARCHAR データ型との間の暗黙的および明示的な変換を示しています。

次のようなテーブルとデータがあるとします。

```
CREATE TABLE tchar(c1 CHAR(9))
CREATE TABLE tvarchar(c2 VARCHAR(9))
CREATE TABLE tbar(c2 BIT)
CREATE TABLE tbit(c2 BIT)

INSERT tbar VALUES(1)
INSERT tbar VALUES(0)
```

BIT から VARCHAR または VARCHAR から BIT への暗黙的な変換、および BIT から VARCHAR への暗黙的な変換：

```
INSERT tvarchar SELECT c2 FROM tbar
SELECT c2, char_length(c2) FROM tvarchar

c2,char_length(tvarchar.c2)
---------------------------
‘1’,1
‘0’,1
```

VARCHAR から BIT への暗黙的な変換：

```
INSERT tbit SELECT c2 FROM tvarchar
SELECT c2 FROM tbit

c2
--
0
1
```

BIT から CHAR または CHAR から BIT への明示的な変換、および BIT から CHAR への明示的な変換：

```
INSERT tchar SELECT CONVERT (CHAR(9), c2) FROM tbar
SELECT c1, char_length(c1) FROM tchar

c1,char_length(tchar.c1)
```

```
------------------------
‘1’,9
‘0’,9
```

CHAR から BIT への明示的な変換：

```
INSERT tbit SELECT CONVERT (BIT, c1) FROM tchar
SELECT c2 FROM tbit

c2
--
0
1
```

BIT から VARCHAR または VARCHAR から BIT への明示的な変換、および BIT から VARCHAR への明示的な変換：

```
INSERT tvarchar SELECT CONVERT(VARCHAR(9), c2)
  FROM tbar
SELECT c2, char_length(c2) FROM tvarchar

c2,char_length(tvarchar.c2)
---------------------------
‘1’,1
‘0’,1
```

VARCHAR から BIT への明示的な変換：

```
INSERT tbit SELECT CONVERT (BIT, c2) FROM tvarchar
SELECT c2 FROM tbit

c2
--
0
1
```

SQL 関数

関数は、データベースから情報を返します。関数は、式が使用できる場所であればどこでも使用できます。

Sybase IQ で関数を使用する場合は、特に記述がないかぎり、NULL 値をパラメータとして受け取る関数は、NULL 値を返します。

FROM 句を省略した場合、またはクエリ内のすべてのテーブルが SYSTEM DB 領域にある場合、クエリは Sybase IQ ではなく SQL Anywhere によって処理されます。このため、特に構文およびセマンティックの制限やオプションの設定方法の違いによって、動作が変わる可能性があります。 処理に適用されるルールについては SQL Anywhere のマニュアルを参照してください。

FROM 句を必要としないクエリがある場合は、"FROM iq_dummy" 句を追加することによって、強制的に Sybase IQ で処理させることができます。この `iq_dummy` は、ユーザが自身のデータベースに作成する 1 ロー 1 カラムのテーブルです。

参照：

- その他の関数 (369 ページ)

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。

使用法
単純な集合関数 (**SUM()**、**MIN()**、**MAX()**、**AVG()**、**COUNT()** など) は、**SELECT** 文の select リストの中および **HAVING** 句や **ORDER BY** 句の中だけで使用できます。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。**SELECT** 文の **GROUP BY** 句を使用してグループを作成します。

「*ウィンドウ関数*」と呼ばれる集約関数の新しいクラスでは、「ダウ工業株 30 種平均の四半期の移動平均」や「各部署のすべての従業員とその累積給与をリストする」などのクエリに対する回答を算出する移動平均および累積方法を提供します。

- 単純な集合関数 (**AVG()**、**COUNT()**、**MAX()**、 **MIN()**、**SUM()** など) 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。

- 1 つの引数を取る新しい統計集合関数には、**STDDEV()**、**STDDEV_SAMP()**、**STDDEV_POP()**、**VARIANCE()**、**VAR_SAMP()**、**VAR_POP()** などがあります。

単純な集合関数と新しい集合関数はどちらも、SQL クエリの指定に **<window clause>** (*ウィンドウ*) を組み込むウィンドウ関数として使用できます。これにより、処理時に結果セットに対して概念的に移動ウィンドウを作成できます。

ウィンドウ集合関数のもう 1 つのクラスは、時系列データの分析をサポートします。単純な集合関数および統計集合関数と同様に、これらのウィンドウ集合関数は、SQL クエリの指定 (または *window-spec*) と併用できます。時系列ウィンドウ集合関数は、相関、直線回帰、ランク付け、加重平均の結果を計算します。

- 時系列分析用として、次のような ISO/ANSI SQL:2008 OLAP 関数があります。**CORR**()、**COVAR_POP()**、**COVAR_SAMP()**、**CUME_DIST()**、**FIRST_VALUE()**、**LAST_VALUE**()、**REGR_AVGX()**、**REGR_AVGY()**、 **REGR_COUNT()**、**REGR_INTERCEPT()**、**REGR_R2()**、**REGR_SLOPE()**、**REGR_SXX()**、**REGR_SXY**()、**REGR_SYY()**。
- データベース業界で使用される ISO/ANSI SQL:2008 以外の OLAP 集合関数の拡張機能には、**FIRST_VALUE()**、**MEDIAN()**、**LAST_VALUE()** があります。
- 加重移動平均を計算する OLAP の加重集合関数には、**EXP_WEIGHTED_AVG()** および **WEIGHTED_AVG()** があります。

金融時系列の予測と分析専用に設計された時系列関数は、名前が "**TS_**" で始まります。

注意： 時系列機能は、RAP – The Trading Edition Enterprise でのみ使用できます。『時系列ガイド』を参照してください。

OLAP の使用方法については、『システム管理ガイド：第 2 巻』を参照してください。

集合関数による `LONG BINARY` データ型と `LONG VARCHAR` データ型のサポートについては、『Sybase IQ の非構造化データ分析の概要』を参照してください。

表 18 : 集合関数

集合関数	パラメータ
AVG	([DISTINCT] { *column-name* | *numeric-expr* })
CORR	(dependent-expression, independent-expression)
COUNT	(*)
COUNT	([DISTINCT] { *column-name* | *numeric-expr* })
COVAR_POP	(dependent-expression, independent-expression)

集合関数	パラメータ
COVAR_SAMP	(dependent-expression, independent-expression)
CUME_DIST	( )
EXP_WEIGHTED_AVG	(*expression*, *period-expression*)
FIRST_VALUE	(expression)
LAST_VALUE	(expression)
LIST	([DISTINCT] *string-expression* [, *'delimiter-string'*] [ORDER BY *order-by-expression* [ASC | DESC], ...)
MAX	([DISTINCT] { *column-name* | *numeric-expr* })
MEDIAN	(expression)
MIN	([DISTINCT] { *column-name* | *numeric-expr* })
REGR_AVGX	(dependent-expression, independent-expression)
REGR_AVGY	(dependent-expression, independent-expression)
REGR_COUNT	(dependent-expression, independent-expression)
REGR_INTERCEPT	(dependent-expression, independent-expression)
REGR_R2	(dependent-expression, independent-expression)
REGR_SLOPE	(dependent-expression, independent-expression)
REGR_SXX	(dependent-expression, independent-expression)
REGR_SXY	(dependent-expression, independent-expression)
REGR_SYY	(dependent-expression, independent-expression)
STDDEV	([ALL] *expression*)
SUM	([DISTINCT] { *column-name* | *numeric-expr* })
VARIANCE	([ALL] *expression*)
WEIGHTED_AVG	(*expression*, *period-expression*

集合関数 **AVG**、**SUM**、**STDDEV**、**VARIANCE** は、バイナリ・データ型 (BINARY と VARBINARY) をサポートしていません。

参照：

- 分析関数 (110 ページ)

分析関数

分析関数として、単純な集合関数、ウィンドウ関数、数値関数があります。

- 単純な集合関数 ― **AVG**、**COUNT**、**MAX**、 **MIN**、**SUM**、**STDDEV**、**VARIANCE**。

  注意： Grouping() 関数以外の単純な集合関数は OLAP ウィンドウ関数と併用できます。
- ウィンドウ関数
  - ウィンドウ集合関数 ― **AVG**、**COUNT**、**MAX**、 **MIN**、**SUM**。
  - ランク付け関数 ― **RANK**、**DENSE_RANK**、**PERCENT_RANK**、**ROW_NUMBER**、**NTILE**。
  - 統計関数 ― **STDDEV**、**STDDEV_SAMP**、**STDDEV_POP**、**VARIANCE**、**VAR_SAMP**、**VAR_POP**。
  - 分散統計関数 ― **PERCENTILE_CONT**、**PERCENTILE_DISC**。
  - Interrow 関数 ― **LAG**、**LEAD**。
- 数値関数 ― **WIDTH_BUCKET**、**CEIL**、**LN**、**EXP**、**POWER**、**SQRT**、**FLOOR**。

  注意： ランク付け統計関数と逆分散統計関数は、Adaptive Server Enterprise ではサポートされません。

次の表は、統計関数とそのパラメータを示します。一部の集合関数とは異なり、ウィンドウ関数で **DISTINCT** は指定できません。

表 19 : 分析関数

関数	パラメータ
AVG	({ *column-name* | *numeric-expr* })
COUNT	(*)
COUNT	({ *column-name* | *expression* })
DENSE_RANK	()
GROUPING *	({ GROUPING *group-by-expression*})
MAX	({ *column-name* | *expression* })
MIN	({ *column-name* | *expression* })
NTILE	(*integer*)
PERCENT_RANK	()

関数	パラメータ
PERCENTILE_CONT	(*numeric-expr*)
PERCENTILE_DISC	(*numeric-expr*)
RANK	()
ROW_NUMBER	()
STDDEV	([ALL] *expression*)
STDDEV_POP	([ALL] *expression*)
STDDEV_SAMP	([ALL] *expression*)
SUM	({ *column-name* | *expression* })
VAR_POP	([ALL] *expression*)
VAR_SAMP	([ALL] *expression*)
VARIANCE	([ALL] *expression*)

* OLAP SQL 標準では、Grouping() は **GROUP BYCUBE** または **GROUP BY ROLLUP** オペレーションでのみ使用できます。

参照：
- 集合関数 (107 ページ)

ウィンドウ集合関数の使用法

ISO/ANSI SQL 拡張で導入された OLAP 用の主な機能として、「*ウィンドウ*」という名前の構成体があります。このウィンドウ拡張により、ユーザはクエリの結果セット (クエリの論理パーティション) をパーティションと呼ばれるローのグループに分割し、現在のローについて集計するローのサブセットを決定することができます。

1 つのウィンドウで 3 つのウィンドウ関数クラス (ランク付け関数、ロー・ナンバリング関数、ウィンドウ集合関数) を使用できます。

ウィンドウ拡張は、ウィンドウ名または指定に対するウィンドウ関数の種類を指定し、1 つのクエリ式のスコープ内のパーティション化された結果セットに適用されます。ウィンドウ・パーティションは、特殊な **OVER** 句の 1 つ以上のカラムで定義されている、クエリから返されるローのサブセットです。

```
OVER (PARTITION BY col1, col2...)
```

ウィンドウ操作では、パーティション内の各ローのランク付け、パーティション内のローの値の分布、および類似の操作などの情報を設定できます。また、デー

タの移動平均や合計を計算し、データおよびそのデータの操作に対する影響を評価する機能を拡張することもできます。

ウィンドウ・パーティションは、特殊な **OVER()** 句の 1 つ以上のカラムで定義されている、クエリから返されるローのサブセットです。

```
OVER (PARTITION BY col1, col2...)
```

分析関数による LONG BINARY データ型および LONG VARCHAR データ型のサポートについては、『Sybase IQ の非構造化データ分析の概要』の「関数のサポート」を参照してください。

参照：

- CORR 関数 [集合] (159 ページ)
- COUNT 関数 [集合] (164 ページ)
- EXP_WEIGHTED_AVG 関数 [集合] (200 ページ)
- FIRST_VALUE 関数 [集合] (202 ページ)
- GROUPING 関数 [集合] (208 ページ)
- LAST_VALUE 関数 [集合] (229 ページ)
- MAX 関数 [集合] (244 ページ)
- MEDIAN 関数 [集合] (245 ページ)
- MIN 関数 [集合] (246 ページ)
- REGR_AVGX 関数 [集合] (282 ページ)
- REGR_COUNT 関数 [集合] (284 ページ)
- REGR_INTERCEPT 関数 [集合] (285 ページ)
- REGR_R2 関数 [集合] (287 ページ)
- REGR_SLOPE 関数 [集合] (288 ページ)
- REGR_SXX 関数 [集合] (289 ページ)
- REGR_SXY 関数 [集合] (291 ページ)
- REGR_SYY 関数 [集合] (292 ページ)
- STDDEV 関数 [集合] (320 ページ)
- SUM 関数 [集合] (332 ページ)
- STDDEV_POP 関数 [集合] (321 ページ)
- STDDEV_SAMP 関数 [集合] (323 ページ)
- VAR_POP 関数 [集合] (352 ページ)
- VAR_SAMP 関数 [集合] (354 ページ)
- VARIANCE 関数 [集合] (355 ページ)
- WEIGHTED_AVG 関数 [集合] (358 ページ)

ランク付け関数の使用法

OLAP ランク付け関数を使用すると、アプリケーション開発者は、「今年度出荷した製品の中で総売り上げが上位 10 位の製品名」や「15 社以上から受注した営業部員の上位 5%」などの情報を取得するクエリを、単一の SQL 文で作成することができます。

このようなランク付け関数には、**RANK()**、**DENSE_RANK()**、**PERCENT_RANK()**、**ROW_NUMBER()**、**NTILE()** があります。

ランク付け統計関数は、グループ内の項目をランク付けし、分布を計算して結果セットを複数のグループに分類します。ランク付け統計関数 (**RANK()**、**DENSE_RANK()**、**PERCENT_RANK()**、**ROW_NUMBER()**、**NTILE()**) には、すべて **OVER** (**ORDER BY**) 句が必要です。次に例を示します。

```
RANK() OVER ( [PARTITION BY] ORDER BY <expression>
[ ASC | DESC ] )
```

ORDER BY 句は、ランク付けを実行するパラメータと、各グループでローをソートする順序を指定します。この **ORDER BY** 句は OVER 句内でのみ使用するもので、**SELECT** 文の **ORDER BY** とは異なります。ランク付けクエリ **ROW** 内の集合関数で **DISTINCT** を指定することはできません。

注意： **ROW_NUMBER()** 関数の **OVER** (**ORDER_BY**) 句に **ROWS** や **RANGE** 句を含めることはできません。

OVER 句は、関数がクエリの結果セットに対して処理を行うことを示します。結果セットは、**FROM**、**WHERE**、**GROUP BY**、**HAVING** の各句がすべて評価された後で返されるローです。**OVER** 句は、ランク付け統計関数の計算の対象となるローのデータ・セットを定義します。

expression にはソートを指定します。カラムの参照、集合関数、またはこれらの項目を起動する式など、有効な式を何でも指定できます。

ASC と DESC のパラメータでは、昇順または降順の順序付けシーケンスを指定します。昇順がデフォルトです。

ランク付け統計関数が使用できるのは、**SELECT** 文や **INSERT** 文の select リストの中、または **SELECT** 文の **ORDER BY** 句の中のみです。ランク付け関数は、ビューまたは union に含めることができます。 ランク付け関数は、サブクエリや **HAVING** 句の他、**UPDATE** 文や **DELETE** 文の select リストでは使用できません。Sybase IQ 15.3 では 1 つのクエリで複数のランク付け統計関数を使用できます。

統計集合分析関数の使用法

統計集合分析関数は、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。集合関数は、**SELECT** 文の select リストおよび **HAVING** 句と **ORDER BY** 句でのみ使用できます。これらの関数としては、**STDDEV**、**STDDEV_POP**、**STDDEV_SAMP**、**VARIANCE**、**VAR_POP**、**VAR_SAMP** があります。

OLAP 関数を、処理時に結果セットに対して概念的に移動ウィンドウを作成する SQL クエリの指定に **OVER()** 句があるウィンドウ関数として使用できます。

分散統計関数の使用法

逆分散統計関数 (**PERCENTILE_CONT** と **PERCENTILE_DISC**) は、パーセンタイル値を関数の引数として受け取り、**WITHIN GROUP** 句で指定されたデータ・グループまたはデータ・セット全体に対して処理を実行します。

これらの関数は、グループごとに 1 つの値を返します。**PERCENTILE_DISC** では、結果のデータ型は **WITHIN GROUP** 句で指定されている **ORDER BY** 項目のデータ型と同じになります。**PERCENTILE_CONT** では、結果のデータ型は、numeric (**WITHIN GROUP** 句の **ORDER BY** 項目が numeric の場合) または double (**ORDER BY** 項目が整数または浮動小数点の場合) となります。

逆分散統計関数では、**WITHIN GROUP** (**ORDER BY**) 句を指定する必要があります。次に例を示します。

```
PERCENTILE_CONT ( expression1 ) WITHIN GROUP ( ORDER BY expression2
[ASC | DESC ] )
```

expression1 の値には、numeric データ型の定数を、0 以上 1 以下の範囲で指定します。引数が NULL であれば、"wrong argument for percentile" エラーが返されます。引数の値が 0 よりも小さいか、1 よりも大きい場合は、"data value out of range" エラーが返されます。

必須の **ORDER BY** 句には、パーセンタイル関数の実行の対象となる式と、各グループ内でのローのソート順を指定します。この **ORDER BY** 句は **WITHIN GROUP** 句内でのみ使用するもので、**SELECT** 文の **ORDER BY** とは異なります。

WITHIN GROUP 句は、クエリ結果を順序付けられたデータ・セットに分類します。関数はこのデータ・セットに基づいて結果を計算します。

expression2 には、カラム参照を含む 1 つの式でソートを指定します。このソート式に、複数の式やランク付け統計関数、set 関数、またはサブクエリを指定することはできません。

ASC と DESC のパラメータでは、昇順または降順の順序付けシーケンスを指定します。昇順がデフォルトです。

逆分散統計関数は、サブクエリ、**HAVING** 句、ビュー、union で使用することが可能です。逆分散統計関数は、分析を行わない単純な集合関数が使用されるところであれば、どこでも使用できます。逆分散統計関数は、データ・セット内の NULL 値を無視します。

Interrow 関数の使用法

Interrow 関数 **LAG** と **LEAD** は、一連のデータ内で前の値または後ろの値にアクセスすることを可能にします。

これらの関数は、セルフジョインなしで、テーブルまたはパーティションの複数のローに同時にアクセスできます。**LAG** 関数は、テーブルまたはパーティション内の **CURRENT ROW** から特定の物理的オフセット分だけ前にあるローにアクセスします。**LEAD** 関数は、テーブルまたはパーティション内の **CURRENT ROW** から特定の物理的オフセット分だけ後ろにあるローにアクセスします。**LAG** 関数と **LEAD** 関数を使用して、「現在のローよりインターバル 2 つ前の株価」や、「現在のローよりインターバル 1 つ後ろの株価」といったクエリを作成できます。『システム管理ガイド：第 2 巻』の「OLAP の使用」>「分析関数」>「ウィンドウ」>「Interrow 関数」を参照してください。

Interrow 関数は **OVER** (**ORDER_BY**) 句を必要とします。

データ型変換関数

データ型変換関数は、引数をあるデータ型から別のデータ型に変換します。

次の表は、データ型変換関数とそのパラメータを示します。

表 20 : データ型変換関数

データ型変換関数	パラメータ
BIGINTTOHEX	(*integer-expression*)
CAST	(*expression* **AS** *data type*)
CONVERT	(*data type*, *expression* [, *format-style*])
HEXTOBIGINT	(*hexadecimal-string*)
HEXTOINT	(*hexadecimal-string*)
INTTOHEX	(*integer-expr*)
ISDATE	(*string*)

データ型変換関数	パラメータ
ISNUMERIC	(*string*)

説明
データベース・サーバは、多数のデータ型変換を自動的に行っています。たとえば、数値式が必要なところに文字列が与えられた場合、文字列は自動的に数値に変換されます。

参照：
- データ型変換 (100 ページ)
- 記憶領域サイズ (87 ページ)

日付と時刻の関数

日付関数と時刻関数は、日付および時刻データ型の変換、抽出、操作を行い、日付および時刻の情報を返します。

以下の表は、日付関数と時刻関数、およびそのパラメータを示しています。

構文 1

表 21 : 日付および時刻関数

日付および時刻関数	パラメータ
DATE	(*expression*)
DATECEILING	(*date-part, datetime-expr, [multiple-expr]*)
DATEFLOOR	(*date-part, datetime-expr, [multiple-expr]*)
DATEFORMAT	(*datetime-expr, string-expr*)
DATENAME	(*date-part, date-expr*)
DATEROUND	(*date-part, datetime-expr, [multiple-expr]*)
DATETIME	(*expression*)
DAY	(*date-expr*)
DAYNAME	(*date-expr*)
DAYS	(*date-expr*)
DAYS	(*date-expr, date-expr*)

日付および時刻関数	パラメータ
DAYS	(*date-expr*, *integer-expr*)
DOW	(*date-expr*)
HOUR	(*datetime-expr*)
HOURS	(*datetime-expr*)
HOURS	(*datetime-expr*, *datetime-expr*)
HOURS	(*datetime-expr*, *integer-expr*)
ISDATE	(*string*)
MINUTE	(*datetime-expr*)
MINUTES	(*datetime-expr*)
MINUTES	(*datetime-expr*, *datetime-expr*)
MINUTES	(*datetime-expr*, *integer-expr*)
MONTH	(*date-expr*)
MONTHNAME	(*date-expr*)
MONTHS	(*date-expr*)
MONTHS	(*date-expr*, *date-expr*)
MONTHS	(*date-expr*, *integer-expr*)
NOW	(*)
QUARTER	(*date-expr*)
SECOND	(*datetime-expr*)
SECONDS	(*datetime-expr*)
SECONDS	(*datetime-expr*, *datetime-expr*)
SECONDS	(*datetime-expr*, *integer-expr*)
TODAY	(*)
WEEKS	(*date-expr*)
WEEKS	(*date-expr*, *date-expr*)
WEEKS	(*date-expr*, *integer-expr*)
YEAR	(*date-expr*)

日付および時刻関数	パラメータ
YEARS	(*date-expr*)
YEARS	(*date-expr*, *date-expr*)
YEARS	(*date-expr*, *integer-expr*)
YMD	(*year-num*, *month-num*, *day-num*)

構文 2

表 22 : Transact-SQL と互換性のある日付関数と時刻関数

Transact-SQL と互換性のある日付関数と時刻関数	パラメータ
DATEADD	(*date-part,numeric-expression,date-expr*)
DATEDIFF	(*date-part*, *date-expr1*, *date-expr2*)
DATENAME	(*date-part*, *date-expr*)
DATEPART	(*date-part*, *date-expr*)
GETDATE	()

説明
Sybase IQ には、日付と時刻の関数の 2 つのクラスがあり、どちらを使用してもかまいませんが、これらのクラスはそれぞれスタイルが異なります。 片方のセットは、Transact-SQL 互換です。

構文 1 の表に示されている日付と時刻の関数では、時間単位で操作できます。ほとんどの時間単位 (`MONTH` など) には、時間を操作するために 4 つの関数がありますが、2 つの名前 (`MONTH` と `MONTHS` など) だけが使用されます。

構文 2 の表に示されている関数は、Transact-SQL の日付と時刻の関数です。これらは、日付および時刻情報へのアクセスおよび操作の代替方法です。

日付関数の引数は、日付に変換してから使用する必要があります。たとえば、以下は誤りです。

```
days ( '1995-11-17', 2 )
```

正しくは以下のとおりです。

```
days ( date( '1995-11-17' ), 2 )
```

Sybase IQ で使用する定数やデータ型は、SQL Anywhere とは異なりますが、ユーザ・インタフェースは共通です。 **SELECT** 文を `FROM` 句なしで発行すると、文は

SQL Anywhere に渡されます。次の文は SQL Anywhere によって排他的に処理されます。

```
SELECT WEEKS('1998/11/01');
```

Sybase IQ が処理する次の文では、**WEEKS** 関数に対して上記とは異なる開始ポイントが指定されており、異なる結果を返します。

```
SELECT WEEKS('1998/11/01') FROM iq_dummy;
```

別の例を考えてみます。**MONTHS** は、「任意の開始日」から経過した月数を返します。 Sybase IQ の「任意の開始日」である仮想日付 0000-01-01 は、最も効率的に日付計算を行うために決められた日付で、さまざまなデータ部分で一貫しています。SQL Anywhere に単一の開始日はありません。 次の 2 つの文は、最初の文が SQL Anywhere によって、次の文が Sybase IQ によって処理され、どちらも 12 を返します。

```
SELECT MONTHS('0001/01/01');
```

```
SELECT MONTHS('0001/01/01') FROM iq_dummy;
```

しかし、以下の文も考えてみてください。

```
SELECT DAYS('0001/01/01');
```

```
SELECT DAYS('0001/01/01') FROM iq_dummy;
```

SQL Anywhere によって処理される最初の文は値 307 を生成しますが、Sybase IQ によって処理される 2 番目の文は 166 を生成します。

結果の一貫性を保つために、必要かどうかにかかわらず、必ず **FROM** 句にテーブル名を含めてください。

注意： カラムとローを 1 つだけ含むダミー テーブルを作成します。日付または時刻の関数を使用する **SELECT** 文の **FROM** 句でこのテーブルを参照すると、必ず Sybase IQ によって処理され、一貫性のある結果を得ることができます。

日付要素

日付関数の多くは、日付要素で構成される日付を使用します。

次の表に、*date-part* に指定可能な値を示します。

表 23 : 日付要素の値

日付部分	省略形	値
Year	yy	0001 – 9999
Quarter	qq	1 – 4
Month	mm	1 – 12

日付部分	省略形	値
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	mcs または us	0 – 999999
Calyearofweek	cyr	integer 型。指定した週が始まる年。週の一部が年初にあたる場合、新年の最初の曜日によっては、その週は前年の最終週の一部となる。新年が木曜日～土曜日に始まる場合、その最初の週は前年の最後の日曜日から開始される。新年が日曜日～水曜日に始まる場合、どの曜日も前年の一部とならない。
Calweekofyear	cwk	年の何週目に指定日が含まれるかを表す 1 ～ 54 の整数。
Caldayofweek	cdw	曜日番号 (日曜日 = 1、土曜日 = 7)。

注意： デフォルトでは、1 週間の始まりは日曜日です。月曜日を週の始まりに指定するには、次のオプションを使用します。

```
set option ‘Date_First_Day_Of_Week’ = ‘1’
```

週の始まりの曜日の指定方法については、『リファレンス：文とオプション』の「データベース・オプション」>「アルファベット順のオプション・リスト」>「DATE_FIRST_DAY_OF_WEEK オプション」を参照してください。

互換性

Adaptive Server Enterprise との互換性が必要な場合は、Transact-SQL 日付／時刻関数を使用します。

参照：

- DATEADD 関数 [日付と時刻] (168 ページ)
- DATECEILING 関数 [日付と時刻] (170 ページ)
- DATEDIFF 関数 [日付と時刻] (173 ページ)
- DATEFLOOR 関数 [日付と時刻] (175 ページ)

- DATEPART 関数 [日付と時刻] (180 ページ)
- DATENAME 関数 [日付と時刻] (179 ページ)
- DATEROUND 関数 [日付と時刻] (182 ページ)

HTTP 関数

HTTP 関数は、Web Service 内での HTTP 要求の処理を容易にします。

次の表は、すべての HTTP 関数とそのパラメータを示します。

表 24 : HTTP 関数

HTTP 関数	パラメータ
HTML_DECODE	(*string*)
HTML_ENCODE	(*string*)
HTTP_DECODE	(*string*)
HTTP_ENCODE	(*string*)
HTTP_HEADER	(*header-field-name*)
HTTP_VARIABLE	(*var-name* [[, *instance*], *header-field*])
NEXT_HTTP_HEADER	(*header-name*)
NEXT_HTTP_VARIABLE	(*var-name*)

数値関数

数値関数は、数値データ型に対して算術演算を行うか、数値情報を返します。

関数
Sybase IQ で使用する定数やデータ型は、SQL Anywhere とは異なりますが、ユーザ・インタフェースは共通です。 **SELECT** 文を **FROM** 句なしで発行すると、文は SQL Anywhere に渡されます。 結果の一貫性を保つために、必要かどうかにかかわらず、**FROM** 句にテーブル名を含めてください。

注意： このような場合は、ダミー・テーブルの使用について検討してください。

次の表は、数値関数とそのパラメータを示します。

表 25 : 数値関数

数値関数	パラメータ
ABS	(*numeric-expr*)
ACOS	(*numeric-expr*)
ASIN	(*numeric-expr*)
ATAN	(*numeric-expr*)
ATAN2	(*numeric-expr1*, *numeric-expr2*)
CEIL	(*numeric-expr*)
CEILING	(*numeric-expr*)
COS	(*numeric-expr*)
COT	(*numeric-expr*)
DEGREES	(*numeric-expr*)
EXP	(*numeric-expr*)
FLOOR	(*numeric-expr*)
LN	(*numeric-expr*)
LOG	(*numeric-expr*)
LOG10	(*numeric-expr*)
MOD	(*dividend*, *divisor*)
PI	(*)
POWER	(*numeric-expr1*, *numeric-expr2*)
RADIANS	(*numeric-expr*)
RAND	([*integer-expr*])
REMAINDER	(*numeric-expr*, *numeric-expr*)
ROUND	(*numeric-expr*, *integer-expr*)
SIGN	(*numeric-expr*)
SIN	(*numeric-expr*)
SQRT	(*numeric-expr*)
SQUARE	(*numeric-expr*)

数値関数	パラメータ
TAN	(*numeric-expr*)
“TRUNCATE”	(*numeric-expr*, *integer-expr*)
TRUNCNUM	(*numeric-expression*, *integer-expression*)
WIDTH_BUCKET	(*expression*, *min_value*, *max_value*, *num_buckets*)

文字列関数

文字列関数は、文字列の変換、抽出、操作を行い、また文字列に関する情報を返します。

マルチバイト文字セットを操作するときは、使用している関数が文字情報を返すのか、バイト情報を返すのかに注意してください。

大部分の文字列関数は、*string-expr* パラメータで指定されたバイナリ・データ (16 進文字) を処理できますが、**LCASE**、**UCASE**、**LOWER**、**LTRIM** など一部の関数では、文字列式に文字列以外を指定できません。

LONG VARCHAR 型の結果を返す関数 (**SPACE**、**REPEAT** など) に定数の **LENGTH** 引数を指定しない場合、デフォルトの長さは最大許容値となります。

これらの関数を 1 つ以上含む Sybase IQ クエリは、次のエラーのいずれかを返す場合があります。

```
ASA Error -1009080: Key doesn't fit on a single database page:
65560(4, 1)
```

```
ASA Error -1009119: Record size too large for database page size
```

次に例を示します。

```
SELECT COUNT(*) FROM test1 a WHERE (a.col1 + SPACE(4-LENGTH(a.col1))
+ a.col2 + space(2- LENGTH(a.col2))) IN (SELECT (b.col3) FROM test1
b);
```

このようなエラーを防ぐには、次の例のように適切な最大長を指定して関数の結果をキャストします。

```
SELECT COUNT(*) FROM test1 a WHERE (a.col1 + CAST(SPACE(4-
LENGTH(a.col1)) AS VARCHAR(4)) + a.col2 + CAST(SPACE(2-LENGTH
(a.col2)) AS VARCHAR(4))) IN (SELECT (b.col3) FROM test1 b);
```

64K の IQ ページ・サイズやマルチバイト照合ではエラーが発生しやすくなります。

次の表は、文字列関数とそのパラメータを示します。

表 26 : 文字列関数

文字列関数	パラメータ
ASCII	(*string-expr*)
BIT_LENGTH	(*column-name*)
BYTE_LENGTH	(*string-expr*)
CHAR	(*integer-expr*)
CHAR_LENGTH	(*string-expr*)
CHARINDEX	(*string-expr1*, *string-expr2*)
DIFFERENCE	(*string-expr1*, *string-expr2*)
GRAPHICAL_PLAN	(*string-expr*)
HTML_PLAN	(*string-expr*)
INSERTSTR	(*numeric-expr*, *string-expr1*, *string-expr2*)
LCASE	(*string-expr*)
LEFT	(*string-expr*, *numeric-expr*)
LEN	(*string-expr*)
LENGTH	(*string-expr*)
LOCATE	(*string-expr1*, *string-expr2* [, *numeric-expr*])
LOWER	(*string-expr*)
LTRIM	(*string-expr*)
OCTET_LENGTH	(*column-name*)
PATINDEX	('%*pattern*%', *string_expr*)
REPEAT	(*string-expr*, *numeric-expr*)
REPLACE	(*original-string*, *search-string*, *replace-string*)
REVERSE	(*expression* | *uchar_expr*)
REPLICATE	(*string-expr*, *integer-expr*)
RIGHT	(*string-expr*, *numeric-expr*)
RTRIM	(*string-expr*)
SIMILAR	(*string-expr1*, *string-expr2*)
SORTKEY	(*string-expression* [, { *collation-id* | *collation-name* [(*collation-tailoring-string*)] }])

文字列関数	パラメータ
SOUNDEX	(*string-expr*)
SPACE	(*integer-expr*)
STR	(*numeric_expr* [, *length* [, *decimal*]])
STR_REPLACE	(*string_expr1*, *string_expr2*, *string_expr3*)
STRING	(*string1* [, *string2*, …, *string99*])
STUFF	(*string-expr1*, *start*, *length*, *string-expr2*)
SUBSTRING	(*string-expr*, *integer-expr* [, *integer-expr*])
TRIM	(*string-expr*)
UCASE	(*string-expr*)
UPPER	(*string-expr*)

文字列関数による LONG BINARY データ型と LONG VARCHAR データ型のサポートについては、『Sybase IQ の非構造化データ分析の概要』の「関数のサポート」を参照してください。

参照：
- 物理的制限 (371 ページ)

システム関数

システム関数はシステム情報を返します。

次の表は、システム関数とそのパラメータを示します。

表 27：システム関数

システム関数	パラメータ
COL_LENGTH	(*table-name*, *column-name*)
COL_NAME	(*table-id*, *column-id* [, *database-id*])
CONNECTION_PROPERTY	({ *property-id* | *property-name* } [, *connection-id*])
DATALENGTH	(*expression*)
DB_ID	([*database-name*])
DB_NAME	([*database-id*])

システム関数	パラメータ
DB_PROPERTY	({ *property-id* | *property-name* } [,{ *database-id* | *database-name* }])
EVENT_CONDITION	(*condition-name*)
EVENT_CONDITION_NAME	(*integer*)
EVENT_PARAMETER	(*context-name*)
GROUP_MEMBER	(*group-name-string-expression* [, *user-name-string-expression*])
INDEX_COL	(*table-name*, *index-id*, *key_#* [,*user-id*])
NEXT_CONNECTION	(*[connection-id] [, database-id]*)
NEXT_DATABASE	({ **NULL** | *database-id* })
OBJECT_ID	(*object-name*)
OBJECT_NAME	(*object-id* [, *database-id*])
PROPERTY	({ *property-id* | *property-name* })
PROPERTY_DESCRIPTION	(*property-id* | *property-name* })
PROPERTY_NAME	(*property-id*)
PROPERTY_NUMBER	(*property-name*)
SUSER_ID	([*user-name*])
SUSER_NAME	([*user-id*])
USER_ID	([*user-name*])
USER_NAME	([*user-id*])

説明
サーバ上で現在実行中のデータベースは、データベース名とデータベース ID 番号で識別されます。`db_id` 関数と `db_name` 関数を使用して、これらの情報を取得できます。

システム関数のセットは、データベース・サーバ上で現在実行中のデータベースのプロパティや接続のプロパティに関する情報を提供します。これらのシステム関数は、データベース名、ID、または接続名をオプションの引数として使用し、プロパティが要求されているデータベースまたは接続を識別します。

パフォーマンス
システム関数は、他の Sybase IQ 関数とは異なる方法で処理されます。Sybase IQ テーブルへのクエリにシステム関数を含めると、パフォーマンスが低下します。

互換性

次の表は、Adaptive Server Enterprise のシステム関数と、Sybase IQ におけるそれらのステータスを示します。

表 28 : Sybase IQ における ASE システム関数のステータス

関数	ステータス
col_length	実装済み
col_name	実装済み
db_id	実装済み
db_name	実装済み
index_col	実装済み
object_id	実装済み
object_name	実装済み
proc_role	常にゼロを返す
show_role	常に NULL を返す
tsequal	実装されていない
user_id	実装済み
user_name	実装済み
suser_id	実装済み
suser_name	実装済み
datalength	実装済み
curunreservedpgs	実装されていない
data_pgs	実装されていない
host_id	実装されていない
host_name	実装されていない
lct_admin	実装されていない
reserved_pgs	実装されていない
rowcnt	実装されていない

関数	ステータス
used_pgs	実装されていない
valid_name	実装されていない
valid_user	実装されていない

注意

- システム関数の中には、Sybase IQ にシステム・ストアド・プロシージャとして実装されているものがあります。
- `db_id`、`db_name`、`datalength`、`suser_id`、`suser_name` は、組み込み関数として実装されています。

接続プロパティ

特定の接続プロパティの値、またはすべての接続プロパティの値を取得します。

接続プロパティは、個々の接続に対して適用されます。すべての接続プロパティの説明については、『SQL Anywhere サーバー - データベース管理』の「データベースの設定」>「接続、データベース、データベースサーバーのプロパティ」>「接続プロパティ」を参照してください。 接続プロパティ **QueryBypassedCosted**、**QueryBypassedOptimized**、**QueryDescribedOptimizer**、**StatementPostAnnotatesSimple** は SQL Anywhere オブジェクトにのみ適用されます。IQ テーブルには適用されません。

注意： 参照先は SQL Anywhere のマニュアルです。

例

特定の接続プロパティの値を取得するには、`connection_property` システム関数を使用します。次の文は、現在の接続によってファイルから読み取られたページ数を返します。

```
select connection_property ( 'DiskRead' )
```

すべての接続プロパティの値を取得するには、`sa_conn_properties` システム・プロシージャを使用します。

```
call sa_conn_properties
```

接続ごと、プロパティごとに、個別のローが表示されます。

参照：

- PROPERTY 関数 [システム] (274 ページ)
- PROPERTY_NAME 関数 [システム] (276 ページ)
- PROPERTY_NUMBER 関数 [システム] (277 ページ)

- CONNECTION_PROPERTY 関数 [システム] (153 ページ)

サーバで使用可能なプロパティ

特定のサーバ・プロパティの値、またはすべてのサーバ・プロパティの値を取得します。

サーバ・プロパティは、サーバ全体に対して適用されます。すべてのサーバ・プロパティの説明については、『SQL Anywhere サーバー - データベース管理』の「データベースの設定」>「接続、データベース、データベースサーバーのプロパティ」>「データベースサーバープロパティ」を参照してください。

Server Edition プロパティは、Sybase IQ のエディションではなく、SQL Anywhere のエディションを返します。 Sybase IQ のライセンス情報を表示するには、**sp_iqlmconfig** システム・プロシージャを使用します。

Server Edition プロパティは、Sybase IQ のエディションではなく、SQL Anywhere のエディションを返します。 Sybase IQ のライセンス情報を表示するには、**sp_iqlmconfig** システム・プロシージャを使用します。

例
特定のサーバ・プロパティの値を取得するには、`property` システム関数を使用します。次の文は、メイン・ヒープの保持に使用されるキャッシュ・ページの数を返します。

```
select property ( 'MainHeapPages') from iq_dummy
```

すべてのサーバ・プロパティの値を取得するには、`sa_eng_properties` システム・プロシージャを使用します。

```
call sa_eng_properties
```

参照：
- sp_iqlmconfig プロシージャ (475 ページ)
- PROPERTY 関数 [システム] (274 ページ)
- PROPERTY_NAME 関数 [システム] (276 ページ)
- PROPERTY_NUMBER 関数 [システム] (277 ページ)
- CONNECTION_PROPERTY 関数 [システム] (153 ページ)

各データベースで使用可能なプロパティ

特定のデータベース・プロパティの値、またはすべてのデータベース・プロパティの値を取得できます。データベース・プロパティは、データベース全体に対して適用されます。

すべてのデータベース・プロパティの説明については、『SQL Anywhere サーバー - データベース管理』の「データベースの設定」>「接続、データベース、データ

ベースサーバーのプロパティ」>「データベースサーバープロパティ」を参照してください。 サーバ・プロパティ **QueryBypassedCosted**、**QueryBypassedOptimized**、**QueryDescribedOptimizer**、**StatementPostAnnotatesSimple** は、SQL Anywhere テーブルに対するクエリについてのみ更新されます。

例
特定のデータベース・プロパティの値を取得するには、`db_property` システム関数を使用します。 次の文は、現在のデータベースのページ・サイズをで返します。

```
select db_property ( 'PageSize')
```

すべてのデータベース・プロパティの値を取得するには、`sa_db_properties` システム・プロシージャを使用します。

```
call sa_db_properties
```

参照：
- PROPERTY 関数 [システム] (274 ページ)
- PROPERTY_NAME 関数 [システム] (276 ページ)
- PROPERTY_NUMBER 関数 [システム] (277 ページ)
- CONNECTION_PROPERTY 関数 [システム] (153 ページ)

SQL および Java のユーザ定義関数

Sybase IQ には、ユーザ定義関数を作成する 2 つのメカニズムがあります。 SQL 言語または Java 言語を使用して、関数を作成できます。

注意： ユーザ定義関数は、SQL Anywhere で処理されます。Sybase IQ のパフォーマンス機能は使用されません。そのため、ユーザ定義関数を含むクエリは、そうでないクエリと比較して、少なくとも 10 倍の処理時間が必要です。

ごくまれに、SQL Anywhere と Sybase IQ のセマンティクスの違いによって、ユーザ定義関数から発行されたクエリの結果に違いが生じることがあります。 たとえば、Sybase IQ では CHAR と VARCHAR を区別し、異なるデータ型として扱いますが、SQL Anywhere は CHAR データを VARCHAR と同じように扱います。

SQL のユーザ定義関数

CREATE FUNCTION 文を使用して、独自の関数を SQL に実装できます。関数のデータ型は、**CREATE FUNCTION** 文内の **RETURN** 文で定義します。

SQL ユーザ定義関数を作成したら、データ型が同じである組み込み関数が使用されるところであれば、どこでも使用できます。

注意： ユーザ定義関数を含むビューでは、CONTAINS の基準が無視されるため、CONTAINS 述部を使用しないでください。LIKE 述部を使用するか、ビューの外部でクエリを発行してください。

SQL 関数の作成方法については、『システム管理ガイド：第 2 巻』の「プロシージャとバッチの使用」を参照してください。

Java のユーザ定義関数

SQL 関数も便利ですが、Java クラスを使用すれば、必要に応じてデータベース・サーバからクライアント・アプリケーションへ関数を移動できるなど Java 固有の利点を活用して、より強力かつ柔軟にユーザ定義関数を実装できます。

インストールされた Java クラスの「*クラス・メソッド*」は、データ型が同じである組み込み関数が使用されるところであればどこでも、ユーザ定義関数として使用できます。

インスタンス・メソッドは、クラスの特定のインスタンスに関連付けられます。そのため、通常のユーザ定義関数と異なる動作をします。

Java クラスの作成とクラス・メソッドの詳細については、『SQL Anywhere サーバー - プログラミング』の「データベースにおける Java」を参照してください。

時系列関数および予測関数

時系列関数および予測関数は、金融時系列分析を目的として専用に設計されています。時系列関数および予測関数の説明は、『時系列ガイド』に記載されています。

注意： 時系列および予測の機能は、RAP – The Trading Edition Enterprise でのみ使用できます。

その他の関数

その他の関数は、他の関数の戻り値を含めた、算術式、文字列式、または日付／時刻式を操作します。

次の表は、その他の関数とパラメータを示します。

表 29 : その他の関数

その他の関数	パラメータ
ARGN	(*integer-expr*, *expression* [, …)
COALESCE	(*expression*, *expression* [, *expression* …])
IFNULL	(*expression1*, *expression2* [, *expression3*])
ISNULL	(*expression*, *expression* [, *expression* …])
NULLIF	(*expression1*, *expression2*)
NUMBER	(*)
ROWID	(*table-name*)

互換性

Adaptive Server Enterprise は、COALESCE 関数、ISNULL 関数、NULLIF 関数のみをサポートします。

アルファベット順の関数リスト

この項では、各 SQL 関数について個別に説明します。

関数の種類 (数値、文字列など) は、関数名の隣に角カッコで囲んで示してあります。

例として示す実行結果の一部に、丸めまたはトランケートされているものがあります。

テーブルのオブジェクト ID やカラム ID など、例として示すデータベースのオブジェクト ID の値は、実際の値と異なる場合があります。

ABS 関数 [数値]

数値式の絶対値を返します。

構文

```
ABS ( numeric-expression )
```

パラメータ

パラメータ	説明
numeric-expression	絶対値を返す数値。

戻り値
数値式の絶対値。

数値式データ型	戻り値
INT	INT
FLOAT	FLOAT
DOUBLE	DOUBLE
NUMERIC	NUMERIC

例
次の文は、値 66 を返します。

```
SELECT ABS( -66 ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise 互換。

ACOS 関数 [数値]

数値式のアーク・コサインをラジアンで返します。

構文

```
ACOS ( numeric-expression )
```

パラメータ

表 30 : パラメータ

パラメータ	説明
numeric-expression	角度の余弦。

戻り値
DOUBLE

例
次の文は、値 1.023945 を返します。

```
SELECT ACOS( 0.52 ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise 互換。

参照：
- COS 関数 [数値] (160 ページ)
- ATAN2 関数 [数値] (138 ページ)
- ATAN 関数 [数値] (137 ページ)
- ASIN 関数 [数値] (136 ページ)
- COT 関数 [数値] (161 ページ)
- SIN 関数 [数値] (309 ページ)
- TAN 関数 [数値] (335 ページ)

ARGN 関数 [その他]

引数のリストから選択した引数を返します。

構文

```
ARGN ( integer-expression, expression [ , …] )
```

パラメータ

表 31 : パラメータ

パラメータ	説明
integer-expression	式のリスト内での引数の位置。
expression	関数に渡される、任意のデータ型の式。ここに記述する式のデータ型は、すべて同じであることが必要です。

戻り値
integer-expression の値を *n* とした場合、引数リストの n 番目の引数 (1 から開始) を返します。

例
次の文は、値 6 を返します。

```
SELECT ARGN( 6, 1,2,3,4,5,6 ) FROM iq_dummy
```

使用法
integer-expression の値を *n* とした場合、引数リストの *n* 番目の引数 (1 から開始) を返します。式のデータ型に制限はありませんが、すべての式でデータ型が同じであることが必要です。整数式には、1 からリストにある式の数までの値を指定します。それ以外を指定すると、Null が返ります。式は、カンマで区切って入力します。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise 互換。

ASCII 関数 [文字列]

文字列式の先頭バイトの ASCII 値を整数で返します。

構文

```
ASCII ( string-expression )
```

パラメータ

表 32 : パラメータ

パラメータ	説明
string-expression	文字列。

戻り値
SMALLINT

例
照合順序にデフォルトの ISO_BINENG が設定されている場合、次の文を実行すると値 90 が返ります。

```
SELECT ASCII( 'Z' ) FROM iq_dummy
```

使用法
文字列が空の場合、**ASCII** ではゼロが返されます。リテラル文字列は引用符で囲む必要があります。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise 互換。

ASIN 関数 [数値]

数値のアーク・サインをラジアンで返します。

構文

```
ASIN ( numeric-expression )
```

パラメータ

表 33 : パラメータ

パラメータ	説明
numeric-expression	角度のサイン。

戻り値
DOUBLE

例
次の式を実行すると、値 0.546850 が返ります。

```
SELECT ASIN( 0.52 ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise 互換。

参照：

- COS 関数 [数値] (160 ページ)
- ATAN2 関数 [数値] (138 ページ)
- ATAN 関数 [数値] (137 ページ)
- ACOS 関数 [数値] (133 ページ)
- COT 関数 [数値] (161 ページ)
- SIN 関数 [数値] (309 ページ)
- TAN 関数 [数値] (335 ページ)

ATAN 関数 [数値]

数値のアーク・タンジェントをラジアンで返します。

構文

```
ATAN ( numeric-expression )
```

パラメータ

表 34 : パラメータ

パラメータ	説明
numeric-expression	角度のタンジェント。

戻り値
DOUBLE

例
次の文は、値 0.479519 を返します。

```
SELECT ATAN( 0.52 ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise 互換。

参照：
- COS 関数 [数値] (160 ページ)
- ATAN2 関数 [数値] (138 ページ)
- ASIN 関数 [数値] (136 ページ)
- ACOS 関数 [数値] (133 ページ)
- COT 関数 [数値] (161 ページ)
- SIN 関数 [数値] (309 ページ)
- TAN 関数 [数値] (335 ページ)

ATAN2 関数 [数値]

2 つの数値の比率のアーク・タンジェントをラジアンで返します。

構文
```
ATAN2 ( numeric-expression1, numeric-expression2 )
```

パラメータ

パラメータ	説明
numeric-expression1	アーク・タンジェントを計算する比率における分子。
numeric-expression2	アーク・タンジェントを計算する比率における分母。

戻り値
DOUBLE

例
次の文は、値 0.00866644968879073143 を返します。
```
SELECT ATAN2( 0.52, 060 ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — ATAN2 は、Adaptive Server Enterprise ではサポートされていません。

参照：
- COS 関数 [数値] (160 ページ)
- ATAN 関数 [数値] (137 ページ)
- ASIN 関数 [数値] (136 ページ)
- ACOS 関数 [数値] (133 ページ)
- COT 関数 [数値] (161 ページ)

- SIN 関数 [数値] (309 ページ)
- TAN 関数 [数値] (335 ページ)

AVG 関数 [集合]

ロー・セットに対する数値式の平均を計算します。または、ユニークな値のセットの平均を計算します。

構文

```
AVG ( numeric-expression | DISTINCT column-name )
```

パラメータ

表 35 : パラメータ

パラメータ	説明
numeric-expression	ロー・セットに対して平均を計算する値。
DISTINCT column-name	column-name の一意な値の平均を計算します。この方法で使用することはほとんどありませんが、万全を期すために含まれています。

戻り値

ローがまったくないグループに対しては、NULL 値が返されます。

引数が DOUBLE の場合は DOUBLE を、それ以外の場合は NUMERIC を返します。

例

次の文は、値 49988.6 を返します。

```
SELECT AVG ( salary ) FROM Employees
```

使用法

この平均には、*numeric -expression* が NULL 値であるローは含まれません。ローがまったくないグループに対しては、NULL 値が返されます。

標準と互換性

- SQL — ISO/ANSI SQL 準拠。
- Sybase — Adaptive Server Enterprise 互換。

参照：

- COUNT 関数 [集合] (164 ページ)
- SUM 関数 [集合] (332 ページ)

BFILE 関数 [データ抽出]

個別の LONG BINARY セルと LONG VARCHAR セルをサーバ上の個別のオペレーティング・システム・ファイルに抽出します。

使用法

IQ データ抽出機能には、個別の LONG BINARY セルと LONG VARCHAR セルをサーバ上の個別のオペレーティング・システム・ファイルに抽出できる **BFILE** 関数が含まれています。**BFILE** は、データ抽出機能と一緒に使用できるほか、単独でも使用できます。

非構造化データ分析機能の使用ライセンスを取得している場合は、この関数でラージ・オブジェクト・データを使用できます。

詳細については、『Sybase IQ の非構造化データ分析の概要』の「関数のサポート」を参照してください。

BIGINTTOHEX 関数 [データ型変換]

10 進の整数を、データ型 VARCHAR(16) の 16 進数に変換して返します。

構文

```
BIGINTTOHEX ( integer-expression )
```

パラメータ

パラメータ	説明
integer-expression	16 進数に変換される整数。

例

次の例は、値 0000000000000009 を返します。

```
SELECT BIGINTTOHEX(9) FROM iq_dummy
```

次の例は、値 FFFFFFFFFFFFFFF7 を返します。

```
SELECT BIGINTTOHEX (-9) FROM iq_dummy
```

使用法

BIGINTTOHEX は、BIGINT データ型として評価される整数式を受け取り、等価の 16 進文字列を返します。返された値は 16 桁になるまで左側に 0 が埋め込まれます。位取りされていない整数データ型のすべての型が、整数式として処理されます。

必要に応じて、変換は自動的に実行されます。小数値がゼロの場合にのみ、定数はトランケートされます。 カラムが正の位取り値で宣言されている場合、カラムはトランケートできません。変換に失敗すると、CONVERSION_ERROR オプションが OFF にされていないかぎり、Sybase IQ はエラーを返します。 OFF に設定されている場合は、NULL を返します。

標準と互換性

- SQL — ISO/ANSI SQL 文法の Transact-SQL 拡張。
- Sybase — Adaptive Server Enterprise 互換。

参照：
- HEXTOBIGINT 関数 [データ型変換] (210 ページ)
- HEXTOINT 関数 [データ型変換] (211 ページ)
- INTTOHEX 関数 [データ型変換] (222 ページ)

BIT_LENGTH 関数 [文字列]

カラム・パラメータのビット長を、符号なし 64 ビット値で返します。

構文

```
BIT_LENGTH( column-name )
```

パラメータ

表 36 : パラメータ

パラメータ	説明
column-name	カラムの名前。

戻り値
INT

使用法
引数が NULL の場合は、NULL 値を返します。

BIT_LENGTH 関数は、すべての Sybase IQ データ型をサポートしています。

非構造化データ分析機能の使用ライセンスを取得している場合は、この関数でラージ・オブジェクト・データを使用できます。

詳細については、『Sybase IQ の非構造化データ分析の概要』の「関数のサポート」を参照してください。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — SQL Anywhere または Adaptive Server Enterprise によるサポートなし。

参照：

- BYTE_LENGTH 関数 [文字列] (142 ページ)
- CHAR_LENGTH 関数 [文字列] (148 ページ)
- COL_LENGTH 関数 [システム] (151 ページ)
- DATALENGTH 関数 [システム] (166 ページ)
- LEN 関数 [文字列] (235 ページ)
- LENGTH 関数 [文字列] (236 ページ)
- OBJECT_NAME 関数 [システム] (263 ページ)
- OCTET_LENGTH 関数 [文字列] (264 ページ)
- STR_REPLACE 関数 [文字列] (325 ページ)

BYTE_LENGTH 関数 [文字列]

文字列のバイト数を返します。

構文

```
BYTE_LENGTH ( string-expression )
```

パラメータ

パラメータ	説明
string-expression	長さが計算される文字列。

戻り値

INT

例

値として 12 を返す。

```
SELECT BYTE_LENGTH( 'Test Message' ) FROM iq_dummy
```

使用法

末尾にあるスペース文字を含めた長さが返されます。

NULL 文字を指定すると、NULL 値が返ります。

マルチバイト文字セットの文字列の場合、**BYTE_LENGTH** の値は **CHAR_LENGTH** で返される文字数と異なります。

非構造化データ分析機能の使用ライセンスを取得している場合は、この関数でラージ・オブジェクト・データを使用できます。

詳細については、『Sybase IQ の非構造化データ分析の概要』の「関数のサポート」を参照してください。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise によるサポートなし。

参照：

- BIT_LENGTH 関数 [文字列] (141 ページ)
- CHAR_LENGTH 関数 [文字列] (148 ページ)
- COL_LENGTH 関数 [システム] (151 ページ)
- DATALENGTH 関数 [システム] (166 ページ)
- LEN 関数 [文字列] (235 ページ)
- LENGTH 関数 [文字列] (236 ページ)
- OBJECT_NAME 関数 [システム] (263 ページ)
- OCTET_LENGTH 関数 [文字列] (264 ページ)
- STR_REPLACE 関数 [文字列] (325 ページ)

BYTE_LENGTH64 関数

BYTE_LENGTH64 は、LONG BINARY カラム・パラメータのバイト長を表す符号なし 64 ビット値を返します。

使用法

BYTE_LENGTH64 は、LONG VARCHAR データ型、および任意のデータ・サイズの LONG BINARY 変数と LONG VARCHAR 変数もサポートしています。

非構造化データ分析機能の使用ライセンスを取得している場合は、この関数でラージ・オブジェクト・データを使用できます。

詳細については、『Sybase IQ の非構造化データ分析の概要』の「関数のサポート」を参照してください。

BYTE_SUBSTR64 関数と BYTE_SUBSTR 関数

BYTE_SUBSTR64 関数と **BYTE_SUBSTR** 関数は、LONG BINARY カラム・パラメータの long binary 型バイト部分文字列を返します。

使用法

BYTE_SUBSTR64 関数と **BYTE_SUBSTR** 関数は、LONG VARCHAR データ型、および任意のデータ・サイズの LONG BINARY 変数と LONG VARCHAR 変数もサポートしています。

CHAR_LENGTH64 は、任意のデータ・サイズの LONG VARCHAR 変数もサポートしています。

非構造化データ分析機能の使用ライセンスを取得している場合は、この関数でラージ・オブジェクト・データを使用できます。

詳細については、『Sybase IQ の非構造化データ分析の概要』の「関数のサポート」を参照してください。

CAST 関数 [データ型変換]

式を指定されたデータ型に変換した値が返されます。

構文

```
CAST ( expression
                AS  data type )
```

パラメータ

パラメータ	説明
expression	変換される式。
データ型	ターゲットのデータ型。

戻り値

指定されたデータ型。

例

次のように関数を指定すると、文字列が日付として使用されます。

```
CAST( '2000-10-31' AS DATE )
```

次では、式 **1 + 2** の値が計算され、結果が 1 文字の文字列に、データ・サーバが割り当てる長さでキャストされます。

```
CAST( 1 + 2 AS CHAR )
```

次のように **CAST** 関数を使用して文字列を短縮できます。

```
SELECT CAST( lname AS CHAR(5) ) FROM Customers
```

使用法
文字列型への変換に長さを指定しない場合は、Sybase IQ によって適切な長さが選択されます。 DECIMAL 変換で精度と小数位桁数のどちらも指定されない場合は、データベース・サーバが適切な値を設定します。

NULL から NUMERIC への明示的な変換に対して、精度も位取りも指定しない場合は、デフォルトで NUMERIC(1,0) が設定されます。次に例を示します。

```
SELECT CAST( NULL AS NUMERIC ) A,
       CAST( NULL AS NUMERIC(15,2) ) B
```

上の式は次のように記述されます。

```
A NUMERIC(1,0)
B NUMERIC(15,2)
```

標準と互換性

- SQL — ISO/ANSI SQL 準拠。
- Sybase — Adaptive Server Enterprise 互換。

参照：

- CONVERT 関数 [データ型変換] (155 ページ)
- HOURS 関数 [日付と時刻] (214 ページ)
- MINUTES 関数 [日付と時刻] (248 ページ)
- MONTHS 関数 [日付と時刻] (251 ページ)
- REPLACE 関数 [文字列] (295 ページ)
- SECOND 関数 [日付と時刻] (306 ページ)
- WEEKS 関数 [日付と時刻] (357 ページ)
- YEAR 関数 [日付と時刻] (362 ページ)
- YEARS 関数 [日付と時刻] (363 ページ)
- DAYS 関数 [日付と時刻] (186 ページ)

CEIL 関数 [数値]

指定された式以上の最小の整数を返します。

CEIL は **CEILING** と同意語です。

構文

```
CEIL ( numeric-expression )
```

パラメータ

パラメータ	説明
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。他のデータ型を指定すると、**CEIL** ではエラーが返ります。戻り値のデータ型は、指定した値のデータ型と同じです。

使用法
指定された式について、**CEIL** 関数は 1 つの引数を取ります。たとえば、**CEIL (-123.45)** は -123 を返し、**CEIL (123.45)** は 124 を返します。

『システム管理ガイド：第 1 巻』の「各国語と文字セット」を参照してください。

標準と互換性

- SQL — ISO/ANSI SQL 準拠。
- Sybase － Adaptive Server Enterprise 互換。

参照：
- FLOOR 関数 [数値] (204 ページ)
- CEILING 関数 [数値] (146 ページ)

CEILING 関数 [数値]

数値の上限値 (その値以上の最も小さい整数) を返します。

CEIL は **CEILING** と同意語です。

構文
```
CEILING ( numeric-expression )
```

パラメータ

パラメータ	説明
numeric-expression	切り上げ値を計算する対象の数値。

戻り値
DOUBLE

例
次の文は、値 60.00000 を返します。

```
SELECT CEILING( 59.84567 ) FROM iq_dummy
```

次の文は、値 123 を返します。

```
SELECT CEILING( 123 ) FROM iq_dummy
```

次の文は、値 124.00 を返します。

```
SELECT CEILING( 123.45 ) FROM iq_dummy
```

次の文は、値 -123.00 を返します。

```
SELECT CEILING( -123.45 ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise 互換。

参照：

- FLOOR 関数 [数値] (204 ページ)
- CEIL 関数 [数値] (145 ページ)

CHAR 関数 [文字列]

数値を ASCII 値とした文字を返します。

構文

```
CHAR ( integer-expression )
```

パラメータ

パラメータ	説明
integer-expression	ASCII 文字に変換する数値。数値には、0 から 255 まで (0 と 255 を含む) を指定してください。

戻り値
VARCHAR

例
次の文を実行すると、値 "Y" が返ります。

```
SELECT CHAR( 89 ) FROM iq_dummy
```

次の文を実行すると、値 "S" が返ります。

```
SELECT CHAR( 83 ) FROM iq_dummy
```

使用法
与えられた数値を 256 で割った余りの値に対応する、現在のデータベースの文字セットに含まれる文字が返されます。

整数式の値が 255 より大きいか、0 より小さい場合、**CHAR** は NULL を返します。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise 互換。

CHAR_LENGTH 関数 [文字列]

文字列の文字数を返します。

構文

```
CHAR_LENGTH ( string-expression )
```

パラメータ

パラメータ	説明
string-expression	長さが計算される文字列。

戻り値
INT

使用法
末尾にあるスペース文字を含めた長さが返されます。

NULL 文字を指定すると、NULL 値が返ります。

マルチバイト文字セットの場合、**CHAR_LENGTH** の結果は **BYTE_LENGTH** の結果より小さくなる場合もあります。

CHAR_LENGTH64 は、任意のデータ・サイズの LONG VARCHAR 変数もサポートしています。

非構造化データ分析機能の使用ライセンスを取得している場合は、この関数でラージ・オブジェクト・データを使用できます。 詳細については、『Sybase IQ の非構造化データ分析の概要』の「関数のサポート」を参照してください。

例
次の文は、値 8 を返します。

```
SELECT CHAR_LENGTH( 'Chemical' ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 準拠。
- Sybase － Adaptive Server Enterprise 互換。

参照：

- BIT_LENGTH 関数 [文字列] (141 ページ)
- BYTE_LENGTH 関数 [文字列] (142 ページ)
- COL_LENGTH 関数 [システム] (151 ページ)
- DATALENGTH 関数 [システム] (166 ページ)
- LEN 関数 [文字列] (235 ページ)
- LENGTH 関数 [文字列] (236 ページ)
- OBJECT_NAME 関数 [システム] (263 ページ)
- OCTET_LENGTH 関数 [文字列] (264 ページ)
- STR_REPLACE 関数 [文字列] (325 ページ)

CHAR_LENGTH64 関数

CHAR_LENGTH64 関数は、LONG VARCHAR カラム・パラメータの文字長 (後続ブランクを含む) を表す符号なし 64 ビット値を返します。

使用法

CHAR_LENGTH64 は、任意のデータ・サイズの LONG VARCHAR 変数もサポートしています。

非構造化データ分析機能の使用ライセンスを取得している場合は、この関数でラージ・オブジェクト・データを使用できます。 詳細については、『Sybase IQ の非構造化データ分析の概要』の「関数のサポート」を参照してください。

CHARINDEX 関数 [文字列]

指定した文字列が別の文字列で最初に出現する位置を返します。

構文

CHARINDEX (*string-expression1, string-expression2*)

パラメータ

パラメータ	説明
string-expression1	検索する文字列。255 バイトまでの文字列を指定してください。

パラメータ	説明
string-expression2	検索される文字列。検索される文字列内で、先頭の文字の位置が 1 になります。

戻り値
INT

例
次に例を示します。

```
SELECT Surname, GivenName
FROM Employees
WHERE CHARINDEX('K', Surname ) = 1
```

この例では、次の値が返されます。

Surname	GivenName
Klobucher	James
Kuo	Felicia
Kelly	Moira

使用法
CHARINDEX 関数で返されるか指定される位置またはオフセットはすべて、常に文字オフセットであり、マルチバイト・データの場合はバイト・オフセットとは異なることがあります。

指定した文字列が検索対象の文字列に 2 つ以上含まれる場合、**CHARINDEX** は最初の文字列の位置を返します。

指定した文字列が検索対象の文字列に含まれない場合、**CHARINDEX** はゼロ (0) を返します。

長さが 0 の文字列を検索すると、1 が返されます。

引数のどれか 1 つでも NULL の場合、結果は NULL になります。

CHARINDEX は、CHAR カラムと VARCHAR カラムの場合、32 ビット符号付き整数で位置を返します。

非構造化データ分析機能の使用ライセンスを取得している場合は、この関数でラージ・オブジェクト・データを使用できます。 詳細については、『Sybase IQ の非構造化データ分析の概要』の「関数のサポート」を参照してください。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise 互換。

参照：
- SUBSTRING 関数 [文字列] (330 ページ)

COALESCE 関数 [その他]

リストから、最初の NULL でない式を返します。

構文

```
COALESCE ( expression, expression [ , … ] )
```

パラメータ

表 37 : パラメータ

パラメータ	説明
expression	任意の式。

戻り値
ANY

例
次の文は、値 34 を返します。

```
SELECT COALESCE( NULL, 34, 13, 0 ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 準拠。
- Sybase — Adaptive Server Enterprise 互換。

参照：
- ISNULL 関数 [その他] (225 ページ)

COL_LENGTH 関数 [システム]

カラムの定義済みの長さを返します。

構文

```
COL_LENGTH ( table-name, column-name )
```

パラメータ	説明
table-name	テーブル名。
column-name	カラム名。

例
次の文は、カラム長 35 を返します。

```
SELECT COL_LENGTH ( 'CUSTOMERS', 'ADDRESS' ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Sybase IQ 用に実装された Adaptive Server Enterprise 関数。

参照：
- BIT_LENGTH 関数 [文字列] (141 ページ)
- BYTE_LENGTH 関数 [文字列] (142 ページ)
- CHAR_LENGTH 関数 [文字列] (148 ページ)
- DATALENGTH 関数 [システム] (166 ページ)
- LEN 関数 [文字列] (235 ページ)
- LENGTH 関数 [文字列] (236 ページ)
- OBJECT_NAME 関数 [システム] (263 ページ)
- OCTET_LENGTH 関数 [文字列] (264 ページ)
- STR_REPLACE 関数 [文字列] (325 ページ)

COL_NAME 関数 [システム]

カラム名を返します。

構文

```
COL_NAME ( table-id, column-id [ , database-id ] )
```

パラメータ

表 38 : パラメータ

パラメータ	説明
table-id	テーブルのオブジェクト ID。
column-id	カラムの ID。

パラメータ	説明
database-id	データベース ID。

例
次の文は、カラム名 `lname` を返します。`Customers` テーブルのオブジェクト ID は 100209 です。この値は **OBJECT_ID** 関数で取得できます。カラム ID は、`syscolumn` システム・テーブルの `column_id` カラムに格納されています。`iqdemo` データベースのデータベース ID は 0 です。この値は DB_ID 関数で取得できます。

```
SELECT COL_NAME( 100209, 3, 0 ) FROM iq_dummy
```

次の文は、カラム名 **city** を返します。

```
SELECT COL_NAME ( 100209, 5 )FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Sybase IQ 用に実装された Adaptive Server Enterprise 関数。

参照：
- DB_ID 関数 [システム] (188 ページ)
- DB_NAME 関数 [システム] (189 ページ)
- DB_PROPERTY 関数 [システム] (190 ページ)
- NEXT_DATABASE 関数 [システム] (256 ページ)
- OBJECT_ID 関数 [システム] (262 ページ)
- OBJECT_NAME 関数 [システム] (263 ページ)

CONNECTION_PROPERTY 関数 [システム]

指定された接続プロパティの値を文字列で返します。

構文

```
CONNECTION_PROPERTY ( { integer-expression1 | string-expression }
                                                    … [ , integer-
expression2 ] )
```

注意： CIS 機能補正のパフォーマンスに関する考慮事項が適用されます。

パラメータ

パラメータ	説明
integer-expression1	ほとんどの場合、文字列式を最初の引数に指定したほうが便利です。integer-expression1 を指定する場合、これは接続プロパティ ID になります。これは、PROPERTY_NUMBER 関数を使用して確認できます。
string-expression	接続プロパティ名。プロパティ ID またはプロパティ名を指定する必要があります。
integer-expression2	現在のデータベース接続の接続 ID。この引数を省略すると、現在の接続を使用します。

戻り値
VARCHAR

例
次の文は、保持している準備文の数 (4 など) を返します。

```
SELECT connection_property( 'PrepStmt' )FROM iq_dummy
```

使用法
第 2 の引数を省略すると、現在の接続を使用します。

標準と互換性

- ISO/ANSI SQL 文法のベンダ拡張。
- Adaptive Server Enterprise 互換。

参照：
- PROPERTY 関数 [システム] (274 ページ)
- PROPERTY_NAME 関数 [システム] (276 ページ)
- PROPERTY_NUMBER 関数 [システム] (277 ページ)
- サーバで使用可能なプロパティ (129 ページ)
- 各データベースで使用可能なプロパティ (129 ページ)
- 接続プロパティ (128 ページ)
- sp_iqshowpsexe プロシージャ (509 ページ)
- sp_iqcontext プロシージャ (412 ページ)

CONVERT 関数 [データ型変換]

式を指定されたデータ型に変換して返します。

構文

```
CONVERT ( data-type, expression [ , format-style ] )
```

パラメータ

パラメータ	説明
data-type	変換後の式のデータ型。
expression	変換される式。
format-style	文字列から日付または時刻データ型に変換 (あるいはその逆) する場合に、使用する日付フォーマット文字列を表すスタイル・コード番号を指定します。

format-style 引数が指定されない場合は、データベース・オプション設定が使用されます。

表 39 : CONVERT のスタイル・コードの出力形式

世紀なし (yy)	世紀あり (yyyy)	出力
-	0 または 100	mmm dd yyyy hh:nnAM (または PM)
1	101	mm/dd/yy[yy]
2	102	[yy]yy.mm.dd
3	103	dd/mm/yy[yy]
4	104	dd.mm.yy[yy]
5	105	dd-mm-yy[yy]
6	106	dd mmm yy[yy]
7	107	mmm dd, yy[yy]
8	108	hh:nn:ss
-	9 または 109	mmm dd yyyy hh:nn:ss:sssAM (または PM)
10	110	mm-dd-yy[yy]
11	111	[yy]yy/mm/dd

世紀なし (yy)	世紀あり (yyyy)	出力
12	112	[yy]yymmdd
-	13 または 113	dd mmm yyyy hh:nn:ss:sss (24 時間形式、欧州デフォルト + ミリ秒、年 4 桁)
14	114	hh:nn:ss (24 時間形式)
-	20 または 120	yyyy-mm-dd hh:nn:ss (24 時間表示、ODBC 標準、年 4 桁)
-	21 または 121	yyyy-mm-dd hh:nn:ss.sss (24 時間表示、ODBC 標準 + ミリ秒、年 4 桁)
36	136	hh:nn:ss.ssssssAM (または PM)
37	137	hh:nn:ss.ssssss
38	138	mmm dd yy[yy] hh:nn:ss.ssssssAM (または PM)
39	139	mmm dd yy[yy] hh:nn:ss.ssssss
40	140	[yy]yy-mm-dd hh:nn:ss.ssssss
-	365	yyyyjjj (文字列または整数で指定。jjj は年内のユリウス日の番号 (1 ～ 366)

次の表は、**CONVERT** フォーマット・スタイルでの日付要素の省略形と値を示します。

省略形	日付要素	値
hh	時間	0 – 23
nn	分	0 – 59
ss	秒	0 – 59
sss	ミリ秒	0 – 999
ssssss	マイクロ秒	0 – 999999
mmm	月	Jan から Dec
dd	日	1 – 31
yyyy	年	0001 – 9999
mm	月	1 – 12

戻り値
指定されたデータ型。

例
format-style は次の文のように使用します。

```
SELECT CONVERT( CHAR( 20 ), order_date, 104 )
FROM sales_order
```

order_date
16.03.1993
20.03.1993
23.03.1993
25.03.1993
...

```
SELECT CONVERT( CHAR( 20 ), order_date, 7 )
FROM sales_order
```

order_date
mar 16, 93
mar 20, 93
mar 23, 93
mar 25, 93
...

```
SELECT order_datetime, CONVERT(CHAR(30), order_datetime, 40)
order_datetime40,
CONVERT(CHAR(30), order_datetime, 140) order_datetime140
FROM sales_order;
```

order_datetime	order_datetime40	order_datetime140
03/05/2009 01:03.05.123456	09-03-05 01:03:05.123456	2009-03-05 01:03:05.123456
03/05/2009 13:05.07.654321	09-03-05 13:05:07.654321	2009-03-05 13:05:07.654321

`SELECT CONVERT(CHAR(50), DATETIME('2009-11-03 11:10:42.033189'), 136) FROM iq_dummy` returns `11:10:42.033189AM`

`SELECT CONVERT(CHAR(50), NOW(), 137) FROM iq_dummy` returns `14:54:48.794122`

次の文は format-style 365 の使用例です。この文では、`DATE` 型と `DATETIME` 型のデータが、文字列型または整数型のデータ (およびその逆) に変換されます。

```
CREATE TABLE tab
   (date_col DATE, int_col INT, char7_col CHAR(7));
INSERT INTO tab (date_col, int_col, char7_col)
   VALUES (‘Dec 17, 2004’, 2004352, ‘2004352’);
```

`SELECT CONVERT(VARCHAR(8), tab.date_col, 365) FROM tab;` returns `‘2004352’`

`SELECT CONVERT(INT, tab.date_col, 365) from tab;` returns `2004352`

`SELECT CONVERT(DATE, tab.int_col, 365) FROM TAB;` returns `2004-12-17`

`SELECT CONVERT(DATE, tab.char7_col, 365) FROM tab;` returns `2004-12-17`

次の文は、整数への変換を示します。この文では値 5 が返されます。

```
SELECT CONVERT( integer, 5.2 ) FROM iq_dummy
```

使用法

CONVERT 関数の結果データ型は、`LONG VARCHAR` です。**SELECT INTO** 文で **CONVERT** を使用する場合は、非構造化データ分析オプションのライセンスを所有しているか、**CAST** を使用して **CONVERT** を正しいデータ型とサイズに設定する必要があります。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise および SQL Anywhere 互換。ただし、format-style 365 は Sybase IQ でのみ使用可能。

参照：

- CAST 関数 [データ型変換] (144 ページ)
- HOURS 関数 [日付と時刻] (214 ページ)
- MINUTES 関数 [日付と時刻] (248 ページ)
- MONTHS 関数 [日付と時刻] (251 ページ)
- REPLACE 関数 [文字列] (295 ページ)
- SECOND 関数 [日付と時刻] (306 ページ)
- WEEKS 関数 [日付と時刻] (357 ページ)

- YEAR 関数 [日付と時刻] (362 ページ)
- YEARS 関数 [日付と時刻] (363 ページ)

CORR 関数 [集合]

一連の数値ペアの相関係数を返します。

構文 1

```
CORR (dependent-expression, independent-expression)
```

構文 2

```
CORR (dependent-expression, independent-expression)
OVER (window-spec)
```

パラメータ

表 40 : パラメータ

パラメータ	説明
dependent-expression	independent-expression の影響を受ける変数。
independent-expression	結果に影響を与える変数。

戻り値
DOUBLE

例
次の例は、相関を実行して年齢が収入に関連付けられているかどうかを調べます。この関数は、値 0.440227 を返します。

```
SELECT CORR( Salary, ( YEAR( NOW( ) ) - YEAR( BirthDate ) ) ) FROM
Employees;
```

使用法
CORR 関数は、引数を DOUBLE に変換して、倍精度浮動小数点計算を実行し、DOUBLE を結果として返します。空のセットが指定された場合、NULL が返されます。

dependent-expression および independent-expression は両方とも数値です。関数は、dependent-expression または independent-expression が NULL であるペアをすべて排除した後に、(dependent-expression, independent-expression) のセットに適用されます。次の計算が行われます。

COVAR_POP (y, x) / (STDDEV_POP (x) * STDDEV_POP (y))

x は dependent-expression を表し、y は independent-expression を表します。

注意： ROLLUP と CUBE は、構文 1 の **GROUP BY** 句ではサポートされていません。

構文 2 は、**SELECT** 文でウィンドウ関数として使用する場合の用法を示しています。その場合は、関数構文 (インライン) または **SELECT** 文の WINDOW 句で、*window-spec* の要素を指定できます。

標準と互換性

- SQL — ISO/ANSI SQL 準拠。主要な SQL に含まれない、SQL/foundation の機能です。
- Sybase — SQL Anywhere 互換。

参照：
- ウィンドウ集合関数の使用法 (111 ページ)

COS 関数 [数値]

数値の余弦をラジアンで返します。

構文

```
COS ( numeric-expression )
```

パラメータ

表 41 : パラメータ

パラメータ	説明
numeric-expression	角度 (ラジアン)。

戻り値
この関数は、引数を DOUBLE に変換し、計算を倍精度浮動小数点で行い、結果を DOUBLE で返します。パラメータが NULL 値の場合、結果は NULL 値になります。

例
次の文は、値 0.86781 を返します。

```
SELECT COS( 0.52 ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise 互換。

参照：
- ATAN2 関数 [数値] (138 ページ)
- ATAN 関数 [数値] (137 ページ)
- ASIN 関数 [数値] (136 ページ)
- ACOS 関数 [数値] (133 ページ)
- COT 関数 [数値] (161 ページ)
- SIN 関数 [数値] (309 ページ)
- TAN 関数 [数値] (335 ページ)

COT 関数 [数値]

数値の余接をラジアンで返します。

構文

```
COT ( numeric-expression )
```

パラメータ

表 42 : パラメータ

パラメータ	説明
numeric-expression	角度 (ラジアン)。

戻り値

この関数は、引数を DOUBLE に変換し、計算を倍精度浮動小数点で行い、結果を DOUBLE で返します。パラメータが NULL 値の場合、結果は NULL 値になります。

例

次の文は、値 1.74653 を返します。

```
SELECT COT( 0.52 ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise 互換。

参照：
- COS 関数 [数値] (160 ページ)
- ATAN2 関数 [数値] (138 ページ)
- ATAN 関数 [数値] (137 ページ)

- ASIN 関数 [数値] (136 ページ)
- ACOS 関数 [数値] (133 ページ)
- SIN 関数 [数値] (309 ページ)
- TAN 関数 [数値] (335 ページ)

COVAR_POP 関数 [集合]

一連の数値ペアの母共分散を返します。

構文 1

```
COVAR_POP ( dependent-expression, independent-expression )
```

構文 2

```
COVAR_POP ( dependent-expression, independent-expression )
OVER ( window-spec )
```

パラメータ

表 43 : パラメータ

パラメータ	説明
dependent-expression	独立した変数の影響を受ける変数。
independent-expression	結果に影響を与える変数。

例

次の例は、従業員の年齢と給与の関連性の度合いを測定します。この関数は、値 73785.840059 を返します。

```
SELECT COVAR_POP( Salary, ( YEAR( NOW( ) ) - YEAR( BirthDate ) ) )
FROM Employees;
```

使用法

この関数は、引数を DOUBLE に変換し、倍精度浮動小数点の計算を行って、DOUBLE を結果として返します。空のセットが指定された場合、**COVAR_POP** は NULL を返します。

dependent-expression および independent-expression は両方とも数値です。関数は、dependent-expression または independent-expression が NULL であるペアをすべて排除した後に、(dependent-expression, independent-expression) のセットに適用されます。次の計算が行われます。

(SUM(x*y) - SUM(x) * SUM(y) / n) / n

x は dependent-expression を表し、y は independent-expression を表します。

詳細については、『SQL Anywhere サーバー - SQL の使用法』の「データのクエリと変更」>「OLAP のサポート」>「SQL Anywhere の Window 関数」>「ロー番号付け関数」>「集合関数に対応する数式」を参照してください。

注意： ROLLUP と CUBE は、構文 1 の **GROUP BY** 句ではサポートされていません。DISTINCT はサポートされていません。

構文 2 は、**SELECT** 文でウィンドウ関数として使用する場合の用法を示しています。その場合は、関数構文 (インライン) または **SELECT** 文の **WINDOW** 句で、*window-spec* の要素を指定できます。

標準と互換性

- SQL — ISO/ANSI SQL 準拠。主要な SQL に含まれない、SQL/foundation の機能です。
- Sybase — SQL Anywhere 互換。

COVAR_SAMP 関数 [集合]

一連の数値ペアの標本共分散を返します。

構文 1

```
COVAR_SAMP ( dependent-expression, independent-expression )
```

構文 2

```
COVAR_SAMP ( dependent-expression, independent-expression )
OVER ( window-spec )
```

パラメータ

表 44 : パラメータ

パラメータ	説明
dependent-expression	独立した変数の影響を受ける変数。
independent-expression	結果に影響を与える変数。

例

次の例は、従業員の年齢と給与の関連性の度合いを測定します。この関数は、値 74782.946005 を返します。

```
SELECT COVAR_SAMP( Salary, ( 2008 - YEAR( BirthDate ) ) ) FROM
Employees;
```

使用法
この関数は、引数を DOUBLE に変換し、倍精度浮動小数点の計算を行って、DOUBLE を結果として返します。空のセットが指定された場合、**COVAR_SAMP** は NULL を返します。

dependent-expression および independent-expression は両方とも数値です。関数は、dependent-expression または independent-expression が NULL であるペアをすべて排除した後に、(dependent-expression, independent-expression) のセットに適用されます。

```
(SUM(x*y) - SUM(x) * SUM(y) / n) / (n-1)
```

x は dependent-expression を表し、y は independent-expression を表します。

詳細については、『SQL Anywhere サーバー - SQL の使用法』の「データのクエリと変更」>「OLAP のサポート」>「SQL Anywhere の Window 関数」>「ロー番号付け関数」>「集合関数に対応する数式」を参照してください。

注意： ROLLUP と CUBE は、構文 1 の **GROUP BY** 句ではサポートされていません。DISTINCT はサポートされていません。

構文 2 は、**SELECT** 文でウィンドウ関数として使用する場合の用法を示しています。その場合は、関数構文 (インライン) または **SELECT** 文の **WINDOW** 句で、*window-spec* の要素を指定できます。

標準と互換性

- SQL — ISO/ANSI SQL 準拠。主要な SQL に含まれない、SQL/foundation の機能です。
- Sybase — SQL Anywhere 互換。

COUNT 関数 [集合]

グループ内で、指定されたパラメータに合致するローの数をカウントします。

構文

```
COUNT ( * |  expression |
                DISTINCT  column-name )
```

パラメータ

パラメータ	説明
*	各グループのローの数を返します。

パラメータ	説明
expression	各グループ内で、expression が NULL 値にならないローの数を返します。
DISTINCT column-name	column-name に含まれる、共通する値を除いた値の数を返します。値が NULL 値であるローはカウントされません。

注意： クエリ結果が表示されるときに、カラム・ヘッダに * は表示されません。次のように表示されます。

```
Count()
```

戻り値
INT

例
一意な各都市と、その都市を値に持つローの数を返します。

```
SELECT city , Count(*)
FROM Employees
GROUP BY city
```

標準と互換性

- SQL — ISO/ANSI SQL 準拠。
- Sybase — Adaptive Server Enterprise 互換。

参照：
- AVG 関数 [集合] (139 ページ)
- SUM 関数 [集合] (332 ページ)
- ウィンドウ集合関数の使用法 (111 ページ)

CUME_DIST 関数 [統計]

CUME_DIST 関数はランク付け統計関数です。1 つの値について、ローのグループ内での相対位置を取得します。0 ～ 1 の 10 進値を返します。

構文

```
CUME_DIST () OVER (window-spec)
```

戻り値
0 ～ 1 の DOUBLE 値

例
次の例は、カリフォルニア (CA) に住む従業員の給与の累積分布を示す結果セットを返します。

```
SELECT DepartmentID, Surname, Salary,CUME_DIST() OVER (PARTITION BY
DepartmentID ORDER BY Salary DESC) "Rank" FROM Employees WHERE State
IN ('CA');
```

次の結果セットが返されます。

表 45 : CUME_DIST の結果セット

DepartmentID	Surname	Salary	Rank
200	Savarino	72,300.000	0.333333
200	Clark	45,000.000	0.666667
200	Overbey	39,300.000	1.000000

使用法
Sybase IQ は、次の式を使用してサイズ N のセット S に含まれる x の値の累積分布を計算します。CUME_DIST(x) = (S に含まれ、指定された順番で x 以前にある値の数) ÷ N

現在、**CUME_DIST** 関数では、複合ソート・キーは許可されていません。複合ソート・キーは、その他のランク付け関数で使用できます。

関数構文 (インライン) または **SELECT** 文の **WINDOW** 句で、*window-spec* の要素を指定できます。*window-spec* には **ORDER BY** 句を含める必要がありますが、**ROWS** や **RANGE** 句を含めることはできません。

注意： DISTINCT はサポートされていません。

標準と互換性

- SQL — ISO/ANSI SQL 準拠。SQL 機能 T612 です。
- Sybase — SQL Anywhere 互換。

DATALENGTH 関数 [システム]

式の長さをバイト数で返します。

構文

```
DATALENGTH ( expression )
```

パラメータ

パラメータ	説明
expression	expression には通常、カラム名を指定します。式が文字列定数の場合は、引用符で囲んでください。

戻り値
UNSIGNED INT

使用法

表 46 : DATALENGTH の戻り値

データ型	DATALENGTH
SMALLINT	2
INTEGER	4
DOUBLE	8
CHAR	データの長さ
BINARY	データの長さ

例
`CompanyName` カラムの最も長い文字列の長さである 32 を返します。

```
SELECT MAX( DATALENGTH( CompanyName ) )
FROM Customers
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Sybase IQ 用に実装された Adaptive Server Enterprise 関数。

参照：

- BIT_LENGTH 関数 [文字列] (141 ページ)
- BYTE_LENGTH 関数 [文字列] (142 ページ)
- CHAR_LENGTH 関数 [文字列] (148 ページ)
- COL_LENGTH 関数 [システム] (151 ページ)
- LEN 関数 [文字列] (235 ページ)
- LENGTH 関数 [文字列] (236 ページ)
- OBJECT_NAME 関数 [システム] (263 ページ)

- OCTET_LENGTH 関数 [文字列] (264 ページ)
- STR_REPLACE 関数 [文字列] (325 ページ)

DATE 関数 [日付と時刻]

式を日付に変換し、時間、分、または秒を削除します。

構文

```
DATE ( expression )
```

パラメータ

表 47 : パラメータ

パラメータ	説明
expression	日付フォーマットに変換される値。通常、文字列を指定します。

戻り値
DATE

例
次の文は、値 1988-11-26 を日付として返します。

```
SELECT DATE( '1988-11-26 21:20:53' ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise ではサポートされていません。

DATEADD 関数 [日付と時刻]

指定された日付要素と数値を、日付に加算して作成された日付を返します。

構文

```
DATEADD ( date-part, numeric-expression, date-expression )
```

パラメータ

パラメータ	説明
日付要素	日付に追加する日付要素。

パラメータ	説明
numeric-expression	日付に加算される日付の単位の数。numeric-expression には、任意の数値型を指定できます。この値は整数にトランケートされます。numeric-expression の最大マイクロ秒は 2147483647 です。これは、35:47.483647 (35 分 47 秒 483647 マイクロ秒) です。
date-expression	変更される日付。

戻り値
TIMESTAMP

例
次の文は、値 1995-11-02 00:00:00.000 を返します。

```
SELECT DATEADD( MONTH, 102, '1987/05/02' ) FROM iq_dummy
```

次の文は、値 2009-11-10 14:57:52.722016 を返します。

```
SELECT DATEADD(MICROSECOND, 15, '2009-11-10
14:57:52.722001') FROM iq_dummy
```

次の文は、値 1985-05-02 00:00:00.123456 を返します。

```
SELECT DATEADD(MICROSECOND, 123456, '1985/05/02')
FROM iq_dummy
```

次の文は、値 1985-05-01 23:59:59.876544 を返します。

```
SELECT DATEADD(MICROSECOND, -123456, '1985/05/02')
FROM iq_dummy
```

次の文は、値 2009-11-03 11:10:42.033192 を返します。

```
SELECT DATEADD(MCS, 2, '2009-11-03 11:10:42.033190')
FROM iq_dummy
```

使用法
DATEADD は Transact-SQL 互換のデータ操作関数です。

標準と互換性

- SQL — ISO/ANSI SQL 文法の Transact-SQL 拡張。
- Sybase — Adaptive Server Enterprise 互換。

参照：

- DATECEILING 関数 [日付と時刻] (170 ページ)
- DATEDIFF 関数 [日付と時刻] (173 ページ)
- DATEFLOOR 関数 [日付と時刻] (175 ページ)
- DATEPART 関数 [日付と時刻] (180 ページ)

- DATENAME 関数 [日付と時刻] (179 ページ)
- DATEROUND 関数 [日付と時刻] (182 ページ)
- 日付要素 (119 ページ)

DATECEILING 関数 [日付と時刻]

渡された値より大きく、指定された粒度の最も近い値で、新しい date、time、または datetime 値を計算します。

構文

DATECEILING (*date-part, datetime-expression [,multiple -expression]*)

パラメータ

パラメータ	説明
date-part	日付に追加する日付要素。
datetime-expression	評価する値を含む date 式、time 式、または datetime 式。
multiple-expression	(オプション)。date-part パラメータによって指定された単位の何倍を使用するかを指定する 0 以外の正の整数値式。たとえば、multiple-expression を使用して、データを 200 マイクロ秒間隔または 10 分間隔で正規化するよう指定できます。 multiple-expression が 0 に評価された場合、負の数値に評価された場合、明示的な NULL 定数である場合、指定された date-part に対して有効な値でない場合、Sybase IQ はエラーを生成します。 multiple-expression が NULL に評価された場合、関数の結果は NULL になります。

例

次の文は、値 August 13, 2009 10:40.00.000 AM を返します。

```
SELECT DATECEILING( MI, 'August 13, 2009, 10:32.00.132AM', 10) FROM
iq_dummy
```

次の文は、値 August 13, 2009 10:32.35.456800 AM を返します。

```
SELECT DATECEILING( US, 'August 13, 2009, 10:32.35.456789AM', 200 )
FROM iq_dummy
```

次の文は、値 August 13, 2009 10:32.35.600000 AM を返します。

```
SELECT DATECEILING( US, 'August 13, 2009, 10:32.35.456789AM',
200000 ) FROM iq_dummy
```

次の文は、値 August 13, 2009 10:32.35.456789 AM を返します。

```
SELECT DATECEILING( US, 'August 13, 2009, 10:32.35.456789AM') FROM
iq_dummy
```

使用法
この関数は、渡された値を、指定された粒度で最大値付近まで大きくして、新しい date、time、または datetime 値を計算します。オプションの *multiple-expression* パラメータを含めた場合、date および time が、指定された粒度で、指定された倍数の最大近似値まで大きくなります。

計算された date および time のデータ型は、*multiple-expression* パラメータのデータ型に一致します。

次の日付要素は、**DATECEILING** と互換性がありません。

- DayofYear
- WeekDay
- CalYearofWeek
- CalWeekofYear
- CalDayofWeek

multiple-expression を microsecond、millisecond、second、minute、または hour 日付要素に指定した場合、Sybase IQ では、粒度が次に大きい単位の最初から倍数が適用されるものとします。

- microsecond の倍数は、現在の second から開始します。
- millisecond の倍数は、現在の second から開始します。
- second の倍数は現在の minute から開始します。
- minute の倍数は現在の hour から開始します。
- hour の倍数は現在の day から開始します。

たとえば、2 分という倍数を指定した場合、Sybase IQ は現在の hour から始まる 2 分の間隔を適用します。

microsecond、millisecond、second、minute、および hour の日付要素の場合は、指定した日付要素の範囲を均等に分割する *multiple-expression* 値を指定します。

- hour の場合、有効な *multiple-expression* の値は、1、2、3、4、6、8、12、24 です。
- second および minute の場合、有効な *multiple-expression* の値は、1、2、3、4、5、6、10、12、15、20、30、60 です。
- millisecond の場合、有効な *multiple-expression* の値は、1、2、4、5、8、10、20、25、40、50、100、125、200、250、500、1000 です。
- microsecond の場合、有効な *multiple-expression* の値は、次のとおりです。

1	40	400	4000	40000
2	50	500	5000	50000
4	64	625	6250	62500
5	80	800	8000	100000
8	100	1000	10000	125000
10	125	1250	12500	200000
16	160	1600	15625	250000
20	200	2000	20000	500000
25	250	2500	25000	1000000
32	320	3125	31250	

multiple-expression を day、week、month、quarter、または year の日付要素に指定した場合、Sybase IQ は、最小の date 値 (0001-01-01)、最小の time 値 (00:00:00.000000)、または最小の datetime 値 (0001-01-01.00:00:00.000000) から間隔が開始するものとします。たとえば、10 日という倍数を指定した場合、Sybase IQ では、0001-01-01 から始まる 10 日間隔が計算されます。

day、week、month、quarter、または year の日付要素の場合、時間粒度の次の大きな単位に均等に分割される倍数を指定する必要はありません。

Sybase IQ が week 日付要素の倍数に丸める場合、date 値は常に Sunday (日曜日) です。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise または SQL Anywhere によるサポートなし。

参照：

- DATEADD 関数 [日付と時刻] (168 ページ)
- DATEDIFF 関数 [日付と時刻] (173 ページ)
- DATEFLOOR 関数 [日付と時刻] (175 ページ)
- DATEPART 関数 [日付と時刻] (180 ページ)
- DATENAME 関数 [日付と時刻] (179 ページ)
- DATEROUND 関数 [日付と時刻] (182 ページ)
- 日付要素 (119 ページ)

DATEDIFF 関数 [日付と時刻]

2 つの日付のインターバルを返します。

構文

```
DATEDIFF ( date-part, date-expression1, date-expression2 )
```

パラメータ

表 48 : パラメータ

パラメータ	説明
date-part	インターバルを計算する日付要素を指定します。
date-expression1	インターバルの起点となる日付。この値を date-expression2 から引いて、2 つの引数の間の日付要素の数を返します。
date-expression2	インターバルの終点となる日付。*date-expression1* をこの値から引いて、2 つの引数の間の日付要素の数を返します。

戻り値
INT

例
次の文は、1 を返します。

```
SELECT DATEDIFF( HOUR, '4:00AM', '5:50AM' )
FROM iq_dummy
```

次の文は、102 を返します。

```
SELECT DATEDIFF( MONTH, '1987/05/02', '1995/11/15' )
FROM iq_dummy
```

次の文は、0 を返します。

```
SELECT DATEDIFF( DAY, '00:00', '23:59' ) FROM iq_dummy
```

次の文は、4 を返します。

```
SELECT DATEDIFF( DAY, '1999/07/19 00:00', '1999/07/23
23:59' ) FROM iq_dummy
```

次の文は、0 を返します。

```
SELECT DATEDIFF( MONTH, '1999/07/19', '1999/07/23' )
FROM iq_dummy
```

次の文は、1 を返します。

```
SELECT DATEDIFF( MONTH, '1999/07/19', '1999/08/23' )
FROM iq_dummy
```

次の文は、4 を返します。

```
SELECT DATEDIFF(MCS, '2009-11-03 11:10:42.033185',
'2009-11-03 11:10:42.033189') FROM iq_dummy
```

次の文は、15 を返します。

```
SELECT DATEDIFF(MICROSECOND, '2009-11-10
14:57:52.722001', '2009-11-10 14:57:52.722016')
FROM iq_dummy
```

次の文は、1,500,000 を返します。

```
SELECT DATEDIFF(MCS, '2000/07/07 07:07:06.277777',
'2000/07/07 07:07:07.777777') FROM iq_dummy
```

使用法
この関数は、指定した 2 つの日付の間における日付要素の数を計算します。結果は日付要素の **(date2 - date1)** に等しい符号付きの整数です。

DATEDIFF 関数の結果が日付要素の整数倍でない場合、結果は丸められずトランケートされます。

day を日付要素として使用すると、**DATEDIFF** 関数は指定された 2 つの時刻の間にある深夜 (午前 0 時) の回数を返します。このとき、2 番目の日付は計算に含まれますが、最初の日付は含まれません。たとえば、次の文は値 5 を返します。最初の日付である 2003/08/03 の深夜は結果に含まれていません。2 番目の日付の深夜は、指定された時刻は深夜よりも前ですが、含まれています。

```
SELECT DATEDIFF( DAY, '2003/08/03 14:00', '2003/08/08 14:00' ) FROM
iq_dummy
```

month を日付要素として使用すると、**DATEDIFF** 関数は 2 つの日付の間にある月の初日の数を返します。このとき、2 番目の日付は計算に含まれますが、最初の日付は含まれません。たとえば、次の文はどちらも 9 を返します。

```
SELECT DATEDIFF( MONTH, '2003/02/01', '2003/11/15' ) FROM iq_dummy;
SELECT DATEDIFF( MONTH, '2003/02/01', '2003/11/01' ) FROM iq_dummy;
```

最初の日付、2003/02/01 は月の初日ですが、どちらのクエリの結果にも含まれていません。2 つ目のクエリの 2 番目の日付、2003/11/01 も月の初日であり、これは結果に含まれています。

week を日付要素として使用すると、**DATEDIFF** 関数は 2 つの日付の間にある日曜日の数を返します。このとき、2 番目の日付は計算に含まれますが、最初の日付は含まれません。たとえば、2003/08 月の日曜日は 03、10、17、24、31 日ですが、次のクエリを実行すると値 4 が返ります。

```
SELECT DATEDIFF( week, '2003/08/03', '2003/08/31' ) FROM iq_dummy;
```

第 1 日曜日 (2003/08/03) は結果に含まれていません。

標準と互換性

- SQL — ISO/ANSI SQL 文法の Transact-SQL 拡張。
- Sybase — Adaptive Server Enterprise 互換。

参照：

- DATEADD 関数 [日付と時刻] (168 ページ)
- DATECEILING 関数 [日付と時刻] (170 ページ)
- DATEFLOOR 関数 [日付と時刻] (175 ページ)
- DATEPART 関数 [日付と時刻] (180 ページ)
- DATENAME 関数 [日付と時刻] (179 ページ)
- DATEROUND 関数 [日付と時刻] (182 ページ)
- 日付要素 (119 ページ)

DATEFLOOR 関数 [日付と時刻]

渡された値を、指定された粒度で、指定された倍数の最小値付近まで小さくして、新しい date、time、または datetime 値を計算します。

構文

DATEFLOOR (*date-part, datetime-expression [,multiple-expression]*)

パラメータ

パラメータ	説明
日付要素	日付に追加する日付要素。
datetime-expression	評価する値を含む date 式、time 式、または datetime 式。
multiple-expression	(オプション)。date-part によって指定された単位の何倍を使用するかを指定する 0 以外の正の整数値式。たとえば、multiple-expression を使用して、データを 200 マイクロ秒間隔または 10 分間隔で正規化するよう指定できます。 multiple-expression が 0 に評価された場合、負の数値に評価された場合、明示的な NULL 定数である場合、指定された date-part に対して有効な値でない場合、Sybase IQ はエラーを生成します。 multiple-expression が NULL に評価された場合、関数の結果は NULL になります。

例

- 次の文は、値 August 13, 2009 10:35:00.000 AM を返します。
```
SELECT DATEFLOOR( MINUTE, 'August 13, 2009 10:35:22.123AM') FROM
iq_dummy
```
- 次の文は、値 August 13, 2009 10:32:35.456600 AM を返します。
```
SELECT DATEFLOOR( US, 'August 13, 2009, 10:32:35.456789AM', 200 )
FROM iq_dummy
```
- 次の文は、値 August 13, 2009 10:32:35.400000 AM を返します。
```
SELECT DATEFLOOR( US, 'August 13, 2009, 10:32:35.456789AM',
200000 ) FROM iq_dummy
```
- 次の文は、値 August 13, 2009 10:32:35.456789 AM を返します。
```
SELECT DATEFLOOR( US, 'August 13, 2009, 10:32:35.456789AM') FROM
iq_dummy
```

使用法
この関数は、渡された値を、指定された粒度で最小値付近まで小さくして、新しい date、time、または datetime 値を計算します。オプションの *multiple-expression* パラメータを含めた場合、date および time が、指定された粒度で、指定された倍数の最小近似値まで小さくなります。

計算された date および time のデータ型は、*multiple-expression* パラメータのデータ型に一致します。

次の日付要素は、**DATEFLOOR** と互換性がありません。

- DayofYear
- WeekDay
- CalYearofWeek
- CalWeekofYear
- CalDayofWeek

multiple-expression を microsecond、millisecond、second、minute、または hour 日付要素に指定した場合、Sybase IQ では、粒度が次に大きい単位の最初から倍数が適用されるものとします。

- microsecond の倍数は、現在の second から開始します。
- millisecond の倍数は、現在の second から開始します。
- second の倍数は現在の minute から開始します。
- minute の倍数は現在の hour から開始します。
- hour の倍数は現在の day から開始します。

たとえば、2 分という倍数を指定した場合、Sybase IQ は現在の hour から始まる 2 分の間隔を適用します。

microsecond、millisecond、second、minute、および hour の日付要素の場合は、指定した日付要素の範囲を均等に分割する *multiple-expression* 値を指定します。

- hour の場合、有効な *multiple-expression* の値は、1、2、3、4、6、8、12、24 です。
- second および minute の場合、有効な *multiple-expression* の値は、1、2、3、4、5、6、10、12、15、20、30、60 です。
- millisecond の場合、有効な *multiple-expression* の値は、1、2、4、5、8、10、20、25、40、50、100、125、200、250、500、1000 です。
- microsecond の場合、有効な *multiple-expression* の値は、次のとおりです。

1	40	400	4000	40000
2	50	500	5000	50000
4	64	625	6250	62500
5	80	800	8000	100000
8	100	1000	10000	125000
10	125	1250	12500	200000
16	160	1600	15625	250000
20	200	2000	20000	500000
25	250	2500	25000	1000000
32	320	3125	31250	

multiple-expression を day、week、month、quarter、または year の日付要素に指定した場合、Sybase IQ は、最小の date 値 (0001-01-01)、最小の time 値 (00:00:00.000000)、または最小の datetime 値 (0001-01-01.00:00:00.000000) から間隔が開始するものとします。たとえば、10 日という倍数を指定した場合、Sybase IQ では、0001-01-01 から始まる 10 日間隔が計算されます。

day、week、month、quarter、または year の日付要素の場合、時間粒度の次の大きな単位に均等に分割される倍数を指定する必要はありません。

Sybase IQ が week 日付要素の倍数に丸める場合、date 値は常に Sunday (日曜日) です。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise または SQL Anywhere によるサポートなし。

参照：
- DATEADD 関数 [日付と時刻] (168 ページ)
- DATECEILING 関数 [日付と時刻] (170 ページ)
- DATEDIFF 関数 [日付と時刻] (173 ページ)
- DATEPART 関数 [日付と時刻] (180 ページ)
- DATENAME 関数 [日付と時刻] (179 ページ)
- DATEROUND 関数 [日付と時刻] (182 ページ)
- 日付要素 (119 ページ)

DATEFORMAT 関数 [日付と時刻]

日付式を指定されたフォーマットで表す文字列を返します。

構文

```
DATEFORMAT ( datetime-expression, string-expression )
```

パラメータ

表 49 : パラメータ

パラメータ	説明
datetime-expression	変換される日時。日付、時刻、タイムスタンプ、または文字列を指定する必要があります。
string-expression	変換後の日付のフォーマット。

戻り値
VARCHAR

例
次の文は、"Jan 01, 1989" のような文字列の値を返します。

```
SELECT DATEFORMAT( StartDate, 'Mmm dd, yyyy' ) from Employees;
```

次の文は、文字列 "Feb 19, 1987" を返します。

```
SELECT DATEFORMAT( CAST ( '1987/02/19' AS DATE ), 'Mmm Dd, yyyy' )
FROM iq_dummy
```

使用法
変換する *datetime-expression* は、データ型が日付、時刻、またはタイムスタンプである必要がありますが、CHAR や VARCHAR の文字列を使用することもできます。日付を文字列で指定すると、Sybase IQ によって文字列が自動的に日付、時刻、タイムスタンプのデータ型に変換されます。したがって、上記の例のように、キャストを明示的に行う必要はありません。

string-expression には、許可されている日付フォーマットをすべて使用できます。日付フォーマットの文字列に、マルチバイト文字を格納することはできません。日付、時刻、日時のフォーマットの文字列に格納できるのは、シングルバイト文字だけです。これはデータベースの照合順序が、932JPN のようにマルチバイトの照合順序である場合も同じです。

次の *'?* がマルチバイト文字を表す場合、このクエリはエラーになります。

```
SELECT DATEFORMAT ( StartDate, 'yy?') FROM Employees;
```

代わりに、連結演算子を使用して、マルチバイト文字を日付フォーマットの文字列の外に移動します。

```
SELECT DATEFORMAT (StartDate, 'yy') + '?' FROM Employees;
```

データベースから取得した日付のフォーマット設定方法については、『リファレンス：文とオプション』の「データベース・オプション」>「アルファベット順のオプション・リスト」>「DATE_FORMAT オプション」を参照してください。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise または SQL Anywhere によるサポートなし。

DATENAME 関数 [日付と時刻]

指定した日時の特定要素の名前を (月ならば "June" のように) 文字列で返します。

構文

```
DATENAME ( date-part, date-expression )
```

パラメータ

表 50 : パラメータ

パラメータ	説明
date-part	名前を返す日付要素。
date-expression	日付要素の名前を返す日付。必要な date-part を含めて指定します。

戻り値
VARCHAR

例
次の文を実行すると、値 May が返ります。

```
SELECT DATENAME( MONTH , '1987/05/02' )
FROM iq_dummy
```

次の文は、値 722,001 を返します。

```
SELECT DATENAME(MICROSECOND, '2009-11-10
14:57:52.722001') FROM iq_dummy
```

次の文は、値 777,777 を返します。

```
SELECT DATENAME(MICROSECOND, '2000/07/07
07:07:07.777777') FROM iq_dummy
```

次の文は、値 33,189 を返します。

```
SELECT DATENAME(MCS, '2009-11-03 11:10:42.033189') FROM iq_dummy
```

使用法

DATENAME は、結果が 23 日などの数値であっても、文字列を返します。

標準と互換性

- SQL — ISO/ANSI SQL 文法の Transact-SQL 拡張。
- Sybase — Adaptive Server Enterprise 互換。

参照：

- DATEADD 関数 [日付と時刻] (168 ページ)
- DATECEILING 関数 [日付と時刻] (170 ページ)
- DATEDIFF 関数 [日付と時刻] (173 ページ)
- DATEFLOOR 関数 [日付と時刻] (175 ページ)
- DATEPART 関数 [日付と時刻] (180 ページ)
- DATEROUND 関数 [日付と時刻] (182 ページ)
- 日付要素 (119 ページ)

DATEPART 関数 [日付と時刻]

日時の値の指定された要素に対する整数値を返します。

構文

DATEPART (*date-part*, *date-expression*)

パラメータ

表 51 : パラメータ

パラメータ	説明
date-part	返される日付要素。
date-expression	要素を返す日付。date-part のフィールドを含める必要があります。

戻り値
INT

例
次の文は、値 5 を返します。

```
SELECT DATEPART( MONTH, '1987/05/02' )
FROM iq_dummy
```

次の文は、値 722,001 を返します。

```
SELECT DATEPART(MICROSECOND, '2009-11-10
14:57:52.722001') FROM iq_dummy
```

次の文は、値 777,777 を返します。

```
SELECT DATEPART(MICROSECOND, '2000/07/07
07:07:07.777777') FROM iq_dummy
```

次の文は、値 33,189 を返します。

```
SELECT DATEPART(MCS, '2009-11-03 11:10:42.033189')
FROM iq_dummy
```

使用法
DATE、**TIME**、**DTTM** インデックスは、一部の日付要素 (Calyearofweek、Calweekofyear、Caldayofweek、Dayofyear、Millisecond、Microsecond) をサポートしていません。

標準と互換性

- SQL — ISO/ANSI SQL 文法の Transact-SQL 拡張。
- Sybase — Adaptive Server Enterprise 互換。

参照：
- DATEADD 関数 [日付と時刻] (168 ページ)
- DATECEILING 関数 [日付と時刻] (170 ページ)
- DATEDIFF 関数 [日付と時刻] (173 ページ)
- DATEFLOOR 関数 [日付と時刻] (175 ページ)
- DATENAME 関数 [日付と時刻] (179 ページ)
- DATEROUND 関数 [日付と時刻] (182 ページ)
- 日付要素 (119 ページ)

DATEROUND 関数 [日付と時刻]

渡された値を、指定された粒度で、指定された値に最も近い倍数まで丸めて、新しい date、time、または datetime 値を計算します。

構文

```
DATEROUND ( date-part, datetime-expression  [,multiple-expression] )
```

パラメータ

パラメータ	説明
日付要素	返される日付要素。
datetime-expression	評価する値を含む date 式、time 式、または datetime 式。
multiple-expression	(オプション)。date-part によって指定された単位の何倍を使用するかを指定する 0 以外の正の整数値式。たとえば、multiple-expression を使用して、データを 200 マイクロ秒間隔または 10 分間隔で正規化するよう指定できます。 *multiple-expression* が 0 に評価された場合、負の数値に評価された場合、明示的な NULL 定数である場合、指定された *date-part* に対して有効な値でない場合は、Sybase IQ がエラーを生成します。 *multiple-expression* が NULL に評価された場合、関数の結果は NULL になります。

例

次の文は、値 August 13, 2009, 10:30.000 AM を返します。

```
SELECT DATEROUND( MI, 'August 13, 2009, 10:33.123AM', 10) FROM
iq_dummy
```

次の文は、値 August 13, 2009 10:32:35.456600 AM を返します。

```
SELECT DATEROUND( US, 'August 13, 2009, 10:32:35.456500AM', 200 )
FROM iq_dummy
```

次の文は、値 August 13, 2009 10:32:35.456789 AM を返します。

```
SELECT DATEROUND( US, 'August 13, 2009, 10:32:35.456789AM') FROM
iq_dummy
```

次の文は、値 August 13, 2009 10:32:35.456400 AM を返します。

```
SELECT DATEROUND( US, 'August 13, 2009, 10:32:35.456499AM', 200 )
FROM iq_dummy
```

使用法
この関数は、渡された値を、指定された粒度で近似値まで丸めて、新しい date、time、または datetime 値を計算します。オプションの *multiple-expression* パラメータを含めた場合、date および time が、指定された粒度で、指定された倍数の近似値まで丸められます。

計算された date および time のデータ型は、*multiple-expression* パラメータのデータ型に一致します。

次の日付要素は、**DATEROUND** と互換性がありません。

- DayofYear
- WeekDay
- CalYearofWeek
- CalWeekofYear
- CalDayofWeek

multiple-expression を microsecond、millisecond、second、minute、または hour 日付要素に指定した場合、Sybase IQ では、粒度が次に大きい単位の最初から倍数が適用されるものとします。

- microsecond の倍数は、現在の second から開始します。
- millisecond の倍数は、現在の second から開始します。
- second の倍数は現在の minute から開始します。
- minute の倍数は現在の hour から開始します。
- hour の倍数は現在の day から開始します。

たとえば、2 分という倍数を指定した場合、Sybase IQ は現在の hour から始まる 2 分の間隔を適用します。

microsecond、millisecond、second、minute、および hour の日付要素の場合は、指定した日付要素の範囲を均等に分割する *multiple-expression* 値を指定します。

- hour の場合、有効な *multiple-expression* の値は、1、2、3、4、6、8、12、24 です。
- second および minute の場合、有効な *multiple-expression* の値は、1、2、3、4、5、6、10、12、15、20、30、60 です。
- millisecond の場合、有効な *multiple-expression* の値は、1、2、4、5、8、10、20、25、40、50、100、125、200、250、500、1000 です。
- microsecond の場合、有効な *multiple-expression* の値は、次のとおりです。

1	40	400	4000	40000

2	50	500	5000	50000
4	64	625	6250	62500
5	80	800	8000	100000
8	100	1000	10000	125000
10	125	1250	12500	200000
16	160	1600	15625	250000
20	200	2000	20000	500000
25	250	2500	25000	1000000
32	320	3125	31250	

multiple-expression を day、week、month、quarter、または year の日付要素に指定した場合、Sybase IQ は、最小の date 値 (0001-01-01)、最小の time 値 (00:00:00.000000)、または最小の datetime 値 (0001-01-01.00:00:00.000000) から間隔が開始するものとします。たとえば、10 日という倍数を指定した場合、Sybase IQ では、0001-01-01 から始まる 10 日間隔が計算されます。

day、week、month、quarter、または year の日付要素の場合、時間粒度の次の大きな単位に均等に分割される倍数を指定する必要はありません。

Sybase IQ が week 日付要素の倍数に丸める場合、date 値は常に Sunday (日曜日) です。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise または SQL Anywhere によるサポートなし。

参照：

- DATEADD 関数 [日付と時刻] (168 ページ)
- DATECEILING 関数 [日付と時刻] (170 ページ)
- DATEDIFF 関数 [日付と時刻] (173 ページ)
- DATEFLOOR 関数 [日付と時刻] (175 ページ)
- DATEPART 関数 [日付と時刻] (180 ページ)
- DATENAME 関数 [日付と時刻] (179 ページ)
- 日付要素 (119 ページ)

DATETIME 関数 [日付と時刻]

式をタイムスタンプに変換します。

構文

```
DATETIME ( expression )
```

パラメータ

表 52 : パラメータ

パラメータ	説明
expression	変換される式。通常、文字列を指定します。変換エラーがあればレポートします。

戻り値
TIMESTAMP

例
次の文を実行すると、値 1998-09-09 12:12:12.000 のタイムスタンプが返ります。

```
SELECT DATETIME( '1998-09-09 12:12:12.000' ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise または SQL Anywhere によるサポートなし。

DAY 関数 [日付と時刻]

指定された日付の日に対応する 1 から 31 までの整数を返します。

構文

```
DAY ( date-expression )
```

パラメータ

表 53 : パラメータ

パラメータ	説明
date-expression	日付。

戻り値
SMALLINT

例
次の文は、値 12 を返します。

```
SELECT DAY( '2001-09-12' ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise 互換。

DAYNAME 関数 [日付と時刻]

指定の日付について、曜日の名前を返します。

構文

```
DAYNAME ( date-expression )
```

パラメータ

表 54 : パラメータ

パラメータ	説明
date-expression	日付。

戻り値
VARCHAR

例
次の文を実行すると、値 Saturday が返ります。

```
SELECT DAYNAME ( '1987/05/02' ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise ではサポートされていません。

DAYS 関数 [日付と時刻]

任意の開始日からの日数を返すか、指定された 2 つの日付の間の日数を返すか、または *integer-expression* で指定された日数を、指定の日付に追加します。

DAYS 関数では、時、分、秒は無視されます。

構文

```
DAYS ( datetime-expression )
|   ( datetime-expression, datetime-expression )
|   ( datetime-expression, integer-expression )
```

パラメータ

表 55 :

パラメータ	説明
datetime-expression	日時。
integer-expression	datetime-expression に追加する日数。integer-expression が負の場合、指定された日数が日時から減算されます。整数式を指定する場合、datetime-expression は日付として明示的にキャストする必要があります。

戻り値

datetime-expression を 2 つ指定した場合は INT。

2 番目の引数が整数の場合は TIMESTAMP。

例

次の文を実行すると、整数値 729948 が返ります。

```
SELECT DAYS( '1998-07-13 06:07:12' ) FROM iq_dummy
```

次の文を実行すると、整数値 -366、すなわち 2 つの日付の差が返ります。

```
SELECT DAYS( '1998-07-13 06:07:12',
'1997-07-12 10:07:12' ) FROM iq_dummy
```

次の文を実行すると、値 1999-07-14 が返ります。

```
SELECT DAYS( CAST('1998-07-13' AS DATE ), 366 )
FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise ではサポートされていません。

参照：

- CAST 関数 [データ型変換] (144 ページ)

DB_ID 関数 [システム]

データベース ID 番号を返します。

構文

```
DB_ID ( [ database-name ] )
```

注意： CIS 機能補正のパフォーマンスに関する考慮事項が適用されます。

パラメータ

表 56 : パラメータ

パラメータ	説明
database-name	データベース名を含む文字列式。database-name が文字列定数の場合は、引用符で囲んでください。database-name を指定しない場合、現在のデータベースの ID 番号が返されます。

戻り値
INT

例
実行中のデータベースが `iqdemo` だけの場合、次の文を実行すると、値 0 が返されます。

```
SELECT DB_ID( 'iqdemo' ) FROM iq_dummy
```

次の文を、唯一実行中のデータベースに対して実行すると、値 0 が返されます。

```
SELECT DB_ID() FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Sybase IQ 用に実装された Adaptive Server Enterprise 関数。

参照：

- COL_NAME 関数 [システム] (152 ページ)
- DB_NAME 関数 [システム] (189 ページ)
- DB_PROPERTY 関数 [システム] (190 ページ)
- NEXT_DATABASE 関数 [システム] (256 ページ)
- OBJECT_ID 関数 [システム] (262 ページ)
- OBJECT_NAME 関数 [システム] (263 ページ)

DB_NAME 関数 [システム]

データベース名を返します。

構文

```
DB_NAME ( [ database-id ] )
```

注意： CIS 機能補正のパフォーマンスに関する考慮事項が適用されます。

パラメータ

表 57 : パラメータ

パラメータ	説明
database-id	データベースの ID。database-id には数値式を指定してください。

戻り値
VARCHAR

例
次の文をデモ・データベースに対して実行すると、データベース名 **iqdemo** が返されます。

```
SELECT DB_NAME( 0 ) FROM iq_dummy
```

使用法
database-id を指定しない場合、現在のデータベースの名前が返されます。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Sybase IQ 用に実装された Adaptive Server Enterprise 関数。

参照：

- COL_NAME 関数 [システム] (152 ページ)
- DB_ID 関数 [システム] (188 ページ)
- DB_PROPERTY 関数 [システム] (190 ページ)
- NEXT_DATABASE 関数 [システム] (256 ページ)
- OBJECT_ID 関数 [システム] (262 ページ)
- OBJECT_NAME 関数 [システム] (263 ページ)

DB_PROPERTY 関数 [システム]

指定されたプロパティの値を返します。

構文

```
DB_PROPERTY ( { property-id | property-name }
[ , { database-id | database-name } ] )
```

注意： CIS 機能補正のパフォーマンスに関する考慮事項が適用されます。

パラメータ

表 58 : パラメータ

パラメータ	説明
property-id	データベースのプロパティ ID。
property-name	データベースのプロパティの名前。
database-id	データベースの ID 番号。DB_ID で取得できます。通常は、データベース名が使用されます。
database-name	データベースの名前。DB_NAME で取得できます。

戻り値
VARCHAR

例
次の文は、現在のデータベースのページ・サイズをバイト単位で返します。

```
SELECT DB_PROPERTY( 'PAGESIZE' ) FROM iq_dummy
```

使用法
文字列を返します。2 番目の引数を省略すると、現在のデータベースが使用されます。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise ではサポートされていません。

参照：

- COL_NAME 関数 [システム] (152 ページ)
- DB_ID 関数 [システム] (188 ページ)
- DB_NAME 関数 [システム] (189 ページ)

- NEXT_DATABASE 関数 [システム] (256 ページ)
- OBJECT_ID 関数 [システム] (262 ページ)
- OBJECT_NAME 関数 [システム] (263 ページ)

DEGREES 関数 [数値]

数値をラジアンから度に変換します。

構文

```
DEGREES ( numeric-expression )
```

パラメータ

表 59 : パラメータ

パラメータ	説明
numeric-expression	角度 (ラジアン)。

戻り値

numeric-expression で指定された角度を返します。

DOUBLE

例

次の文は、値 29.793805 を返します。

```
SELECT DEGREES( 0.52 ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise 互換。

DENSE_RANK 関数 [統計]

グループ内の項目をランク付けします。

構文

```
DENSE_RANK ( ) OVER ( ORDER BY expression [ ASC | DESC ] )
```

パラメータ

表 60 : パラメータ

パラメータ	説明
expression	ソートを指定します。カラムの参照、集合関数、またはこれらの項目を起動する式など、有効な式を何でも指定できます。

戻り値
INTEGER

例
次の文は、**DENSE_RANK** 関数の使い方を示しています。

```
SELECT s_suppkey, DENSE_RANK()
OVER ( ORDER BY ( SUM(s_acctBal) DESC )
AS rank_dense FROM supplier GROUP BY s_suppkey;

s_suppkey        sum_acctBal       rank_dense
supplier#011     200,000            1
supplier#002     200,000            1
supplier#013     123,000            2
supplier#004     110,000            3
supplier#035     110,000            3
supplier#006     50,000             4
supplier#021     10,000             5
```

使用法
DENSE_RANK はランク付け統計関数です。ロー R のランクは、R 以前のローの数のうち、**OVER** 句で指定されたグループ間で同等なロー、または結果セット全体で同等なローを引いた数になります。**DENSE_RANK** と **RANK** の相違点は、順位が同じである場合に、**DENSE_RANK** ではランク順に隔たりが置かれないことです。**RANK** では隔たりが置かれます。

DENSE_RANK には **OVER (ORDER BY)** 句が必須です。**ORDER BY** 句は、ランク付けを実行するパラメータと、各グループでローをソートする順序を指定します。この **ORDER BY** 句は **OVER** 句内でのみ使用するもので、**SELECT** 文の **ORDER BY** とは異なります。ランク付けクエリ内の集合関数に **DISTINCT** を指定することはできません。

OVER 句は、関数がクエリの結果セットに対して処理を行うことを示します。結果セットは、**FROM**、**WHERE**、**GROUP BY**、**HAVING** の各句がすべて評価された後で返されるローです。**OVER** 句は、ランク付け統計関数の計算の対象となるローのデータ・セットを定義します。

ASC と DESC のパラメータでは、昇順または降順の順序付けシーケンスを指定します。昇順がデフォルトです。

DENSE_RANK は、**SELECT** または **INSERT** 文の select リスト、および **SELECT** 文の **ORDER BY** 句でのみ使用できます。**DENSE_RANK** は、ビューまたは union に含めることができます。**DENSE_RANK** 関数は、サブクエリ、HAVING 句、および **UPDATE** または **DELETE** 文の select リストでは使用できません。1 つのクエリで使用可能なランク付け統計関数は、1 つだけです。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise または SQL Anywhere によるサポートなし。

参照：

- RANK 関数 [統計] (280 ページ)

DIFFERENCE 関数 [文字列]

2 つの文字列を比較し、それらの間の類似性を評価して 0 ～ 4 の値を返します。

最高点は 4 です。

構文

```
DIFFERENCE ( string-expression1, string-expression2 )
```

パラメータ

表 61 : パラメータ

パラメータ	説明
string-expression1	比較を行う最初の文字列。
string-expression2	比較を行う 2 つ目の文字列。

戻り値
SMALLINT

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise 互換。

参照：

- SOUNDEX 関数 [文字列] (315 ページ)

DIFFERENCE 関数の例

以下の **DIFFERENCE** 関数の使用例を参考にしてください。

次の文は、値 4 を返します。

```
SELECT DIFFERENCE( 'Smith', 'Smith' ) FROM iq_dummy
```

次の文は、値 4 を返します。

```
SELECT DIFFERENCE( 'Smith', 'Smyth' ) FROM iq_dummy
```

次の文は、値 3 を返します。

```
SELECT DIFFERENCE( 'Smith', 'Sweeney' ) FROM iq_dummy
```

次の文は、値 2 を返します。

```
SELECT DIFFERENCE( 'Smith', 'Jones' ) FROM iq_dummy
```

次の文は、値 1 を返します。

```
SELECT DIFFERENCE( 'Smith', 'Rubin' ) FROM iq_dummy
```

次の文は、値 0 を返します。

```
SELECT DIFFERENCE( 'Smith', 'Wilkins' ) FROM iq_dummy
```

DOW 関数 [日付と時刻]

Sunday = 1、Monday = 2 のような、指定された日付の曜日を表す 1 ～ 7 までの数字を返します。

構文

```
DOW ( date-expression )
```

パラメータ

表 62 : パラメータ

パラメータ	説明
date-expression	日付。

戻り値

SMALLINT

例

次の文は、値 5 を返します。

```
SELECT DOW( '1998-07-09' ) FROM iq_dummy
```

使用法
月曜日 (または別の曜日) を週の始まりの曜日として設定する方法については、『リファレンス：文とオプション』の「アルファベット順のオプション・リスト」>「DATE_FIRST_DAY_OF_WEEK オプション」を参照してください。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise ではサポートされていません。

ERRORMSG 関数 [その他]

現在のエラー、または指定された SQLSTATE または SQLCODE 値のエラー・メッセージを提供します。

構文

```
ERRORMSG
              ( [ sqlstate | sqlcode ] )
```

```
sqlstate: string
```

```
sqlcode: integer
```

パラメータ

表 63 : パラメータ

パラメータ	定義
sqlstate	エラー・メッセージが返される SQLSTATE 値。
sqlcode	エラー・メッセージが返される SQLCODE 値。

戻り値
エラー・メッセージを含む文字列。

VARCHAR

例
次の文を実行すると、SQLCODE -813 のエラー・メッセージを返します。

```
select errormsg( -813 )
```

戻り値
エラー・メッセージを含む文字列。引数を指定しない場合、現在の状態のエラー・メッセージが返されます。置き換え (テーブル名やカラム名など) が行われます。

引数を指定すると、指定した SQLSTATE または SQLCODE のエラー・メッセージが置き換えなしで返されます。テーブル名およびカラム名はプレースホルダ ('???') として返されます。

ERRORMSG 関数は、SQL Anywhere および Sybase IQ のエラー・メッセージを返します。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise によるサポートなし。

EVENT_CONDITION 関数 [システム]

イベント・ハンドラがトリガされる条件を指定します。

イベントおよび関連するハンドラを定義するには、**CREATE EVENT** 文を使用します。

『リファレンス：文とオプション』の「SQL 文」>「CREATE EVENT 文」を参照してください。

構文

```
EVENT_CONDITION ( condition-name )
```

注意： CIS 機能補正のパフォーマンスに関する考慮事項が適用されます。

パラメータ

表 64 : パラメータ

パラメータ	定義
condition-name	イベントをトリガする条件。指定可能な値はデータベースにあらかじめセットされており、大文字小文字を区別して指定します。各条件は、特定のイベント・タイプにのみ有効です。

表 65 : 各イベントに対して有効な条件

条件名	単位	イベント	コメント
DBFreePercent	該当なし	DBDiskSpace	DBDiskSpace は、IQ ストアではなく、システム・データベース・ファイル (.db ファイル) の空き領域を表示します。
DBFreeSpace	メガバイト	DBDiskSpace	
DBSize	メガバイト	GrowDB	
ErrorNumber	該当なし	RAISERROR	
IdleTime	秒	ServerIdle	
Interval	秒	ALL	ハンドラが前回実行されてからの時間。
LogFreePercent	該当なし	LogDiskSpace	
LogFreeSpace	メガバイト	LogDiskSpace	
LogSize	メガバイト	GrowLog	
RemainingValues	整数値	GlobalAutoincre-ment	残った値の数。
TempFreePercent	該当なし	TempDiskSpace	TempDiskSpace は、IQ テンポラリ・ストアではなく、システム・テンポラリ・ファイル (TEMP または IQTMP15 の環境変数でポイントされる) の空き領域を表示します。
TempFreeSpace	メガバイト	TempDiskSpace	
TempSize	メガバイト	GrowTemp	

戻り値
INT

例
次のイベント定義には、**EVENT_CONDITION** 関数が使用されています。

```
create event LogNotifier
type LogDiskSpace
where event_condition( 'LogFreePercent' ) < 50
handler
begin
```

```
    message 'LogNotifier message'
end
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise ではサポートされていません。

参照：

- EVENT_PARAMETER 関数 [システム] (199 ページ)

EVENT_CONDITION_NAME 関数 [システム]

この関数を使用すると、**EVENT_CONDITION** に対して指定可能なパラメータのリストを取得できます。

イベントおよび関連するハンドラを定義するには、**CREATE EVENT** 文を使用します。

『リファレンス：文とオプション』の「SQL 文」>「CREATE EVENT 文」を参照してください。

構文

EVENT_CONDITION_NAME (*integer*)

注意： CIS 機能補正のパフォーマンスに関する考慮事項が適用されます。

パラメータ

パラメータ	説明
integer	0 以上の整数である必要があります。

戻り値
VARCHAR

使用法
EVENT_CONDITION_NAME を使用すると、関数が NULL を返すまでループ処理を指定した回数だけ実行し、**EVENT_CONDITION** のすべての引数のリストを取得できます。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise ではサポートされていません。

EVENT_PARAMETER 関数 [システム]

イベント・ハンドラのコンテキスト情報を取得できます。

イベントおよび関連するハンドラを定義するには、**CREATE EVENT** 文を使用します。

『リファレンス：文とオプション』の「SQL 文」>「CREATE EVENT 文」を参照してください。

構文

```
EVENT_PARAMETER ( context-name )
```

```
context-name:
  'ConnectionID'
| 'User'
| 'EventName'
| 'Executions'
| 'IQDBMainSpaceName'
| 'NumActive'
| 'TableName'
| condition-name
```

注意： CIS 機能補正のパフォーマンスに関する考慮事項が適用されます。

パラメータ

表 66 : パラメータ

パラメータ	説明
context-name	あらかじめセットされた文字列のいずれか。文字列は大文字と小文字を区別します。また、以下の情報を保持します。
ConnectionId	接続 ID。次の文で返されます。 `connection_property( 'id' )`
User	イベントをトリガさせたユーザのユーザ ID。
EventName	トリガされたイベントの名前。
Executions	イベント・ハンドラが実行された回数。
NumActive	イベント・ハンドラのアクティブなインスタンスの数。これは、イベント・ハンドラの数を制限して、一度に 1 つのインスタンスのみが実行されるようにする場合に便利です。
TableName	RemainingValues で使用するテーブル名。

さらに、**EVENT_PARAMETER** 関数からは、**EVENT_CONDITION** 関数に対して有効なすべての *condition-name* 引数にアクセスできます。

戻り値
VARCHAR

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise ではサポートされていません。

参照：

- EVENT_CONDITION 関数 [システム] (196 ページ)

EXP 関数 [数値]

指数関数 (e の数値乗) を返します。

構文

```
EXP ( numeric-expression )
```

パラメータ

表 67 : パラメータ

パラメータ	説明
numeric-expression	指数。

戻り値
DOUBLE

例
次の文は、値 3269017.3724721107 を返します。

```
SELECT EXP( 15 ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise 互換。

EXP_WEIGHTED_AVG 関数 [集合]

指数加重移動平均を計算します。

加重は、平均を構成する各数量の相対的な重要性を決定します。

構文

```
EXP_WEIGHTED_AVG (expression, period-expression)
```

```
OVER (window-spec)
```

パラメータ

パラメータ	説明
expression	計算する加重値の数値式。
period-expression	平均を計算する間隔を指定する数値式。

使用法

WEIGHTED_AVG 関数と同様に、**EXP_WEIGHTED_AVG** の加重は時間経過とともに減少します。しかし、WEIGHTED_AVG の加重は等差階級的に減少しますが、EXP_WEIGHTED_AVG の加重は指数関数的に減少します。指数関数的な加重は、最新の値に加重を適用し、加重を適用しながらも、古い値の加重を減少します。

Sybase IQ は、次の式を使用して指数加重移動平均を計算します。

S*C+(1-S)*PEMA

上の計算では、Sybase IQ は、現在の終値 (C) を補整定数 (S) の積に対して、1 から補整定数を引いた値に前日の指数加重移動平均値 (PEMA) を乗算した結果を追加することによって補整係数を適用します。

Sybase IQ は、**OVER** 句によって指定された期間全体にわたる指数加重移動平均を計算します。*period-expression* は、指数加重平均の移動範囲を指定します。

関数構文 (インライン) または **SELECT** 文の **WINDOW** 句で、*window-spec* の要素を指定できます。*window-spec* は **ORDER BY** 文を含む必要がありますが、フレーム指定を含むことはできません。

注意： ROLLUP と CUBE は、**GROUP BY** 句ではサポートされていません。DISTINCT はサポートされていません。

例

次の例は、フロリダの従業員の給与の指数加重移動平均、および平均のほとんどの加重に関係する現在雇用されている従業員の給与を返します。加重では 3 つのローが使用されます。

```
SELECT DepartmentID, Surname, Salary,EXP_WEIGHTED_AVG(Salary, 3)
OVER (ORDER BY YEAR(StartDate) DESC) as "W_AVG"FROM EmployeesWHERE
State IN ('FL') ORDER BY StartDate DESC
```

次の結果セットが返されます。

表 68 : EXP_WEIGHTED_AVG 結果セット

DepartmentID	Surname	Salary	W_AVG
400	Evans	68,940.000	34,470.000000
300	Litton	58,930.000	46,700.000000
200	Sterling	64,900.000	55,800.000000
200	Kelly	87,500.000	71,650.000000
400	Charlton	28,300.000	49,975.000000
100	Lull	87,900.000	68,937.500000
100	Gowda	59,840.000	60,621.875000
400	Francis	53,870.000	61,403.750000

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。

参照：

- WEIGHTED_AVG 関数 [集合] (358 ページ)
- ウィンドウ集合関数の使用法 (111 ページ)

FIRST_VALUE 関数 [集合]

一連の値の最初の値を返します。

構文

```
FIRST_VALUE (expression [IGNORE NULLS | RESPECT NULLS])
OVER (window-spec)
```

パラメータ

パラメータ	説明
expression	順序付けられたセットの最初の値を決定する式。

戻り値
引数のデータ型。

使用法
FIRST_VALUE は、(通常は順序付けられた) 値のセット内の最初の値を返します。**IGNORE NULLS** が指定されていない場合、セット内の最初の値が NULL のときに

は、NULL が返されます。IGNORE NULLS が指定されている場合、**FIRST_VALUE** は、セット内の NULL ではない最初の値を返します。すべての値が NULL の場合は NULL が返されます。

戻り値のデータ型は、入力値のデータ型と同じです。

expression に **FIRST_VALUE** やその他の分析関数を使用することはできません。したがって、分析関数をネストすることはできませんが、その他の組み込み関数式を expression に使用できます。

window-spec が **ORDER BY** 式を含まない場合、または **ORDER BY** 式の精度が不十分で一意の順序を確保できない場合、結果は不定になります。window-spec がない場合も、結果が不定になります。

関数構文 (インライン) または **SELECT** 文の **WINDOW** 句で、*window-spec* の要素を指定できます。

注意： DISTINCT はサポートされていません。

例
次の例は、各従業員の給与と同じ部署で最も最近雇用された従業員の給与との関係をパーセンテージとして返します。

```
SELECT DepartmentID, EmployeeID,
100 * Salary / ( FIRST_VALUE( Salary ) OVER (
PARTITION BY DepartmentID  ORDER BY Year(StartDate) DESC ) )
AS percentage
FROM Employees order by DepartmentID DESC;
```

次の結果セットが返されます。

表 69 : FIRST_VALUE 結果セット

DepartmentID	EmployeeID	パーセンテージ
500	1,658	100.00000000000000000000
500	1,570	138.842709713689113761394
500	1,615	110.428462434244870095972
500	1,013	109.585190539292454724330
500	750	137.734409508894510701521
500	921	167.449704854836766654619
500	868	113.239368750752921334778
500	703	222.867927558928643135365

DepartmentID	EmployeeID	パーセンテージ
500	191	119.664297474199895594908
400	1,684	100.000000000000000000000
400	1,740	76.128652163477274215016
400	1,751	76.353400685155687446813
400	1,607	133.758100765890593292456
400	1,507	77.996465120338650199655
400	1,576	150.428767810774836893669

この例では、従業員 1658 は部署 500 の最初のローで、その部門で最も最近雇用されたこと、および 100% のパーセンテージを受け取っていることを示しています。部署 500 の残りの従業員のパーセンテージは、従業員 1658 のパーセンテージに相対的に計算されます。たとえば、従業員 1570 は、従業員 1658 の給与の 139% を受け取っています。

標準と互換性

- SQL — ISO/ANSI SQL 準拠。SQL/OLAP 機能 T612 です。
- Sybase — SQL Anywhere 互換。

参照：

- ウィンドウ集合関数の使用法 (111 ページ)

FLOOR 関数 [数値]

数値の下限値 (その値以下の最も大きい整数) を返します。

構文

```
FLOOR ( numeric-expression )
```

パラメータ

表 70 : パラメータ

パラメータ	説明
numeric-expression	数値。通常は float 型です。

戻り値

DOUBLE

例
次の文は、値 123.00 を返します。

```
SELECT FLOOR ( 123 ) FROM iq_dummy
```

次の文は、値 123 を返します。

```
SELECT FLOOR ( 123.45 ) FROM iq_dummy
```

次の式を実行すると、値 -124.00 が返ります。

```
SELECT FLOOR ( -123.45 ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise 互換。

参照：
- CEILING 関数 [数値] (146 ページ)
- CEIL 関数 [数値] (145 ページ)

GETDATE 関数 [日付と時刻]

現在の日付と時刻を返します。

構文

```
GETDATE ( )
```

戻り値
TIMESTAMP

例
次の文は、システムの日付および時刻を返します。

```
SELECT GETDATE( ) FROM iq_dummy
```

使用法
GETDATE は Transact-SQL 互換のデータ操作関数です。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise 互換。

GRAPHICAL_PLAN 関数 [文字列]

グラフィカルなクエリ・プランを Interactive SQL に XML フォーマットの文字列で返します。

構文

```
GRAPHICAL_PLAN ( string-expression
[, statistics-level
[, cursor-type
[, update-status ]]])
```

注意： CIS 機能補正のパフォーマンスに関する考慮事項が適用されます。

パラメータ

パラメータ	説明
string-expression	プランを生成する SQL 文。通常、string-expression は **SELECT** 文ですが、**UPDATE**、**DELETE**、**INSERT SELECT**、**SELECT INTO** 文を指定することもできます。
statistics-level	整数。Statistics-level は次のいずれかです。 • 0 ― オプティマイザの推定のみ (デフォルト) • 2 ― ノード統計値を含む詳細な統計情報 • 3 ― 詳細な統計情報
cursor-type	カーソル・タイプ。文字列として表現されます。有効な値は次のとおりです。asensitive、insensitive、sensitive、または keyset-driven。cursor-type が指定されない場合、デフォルトで asensitive が使用されます。
update-status	次のいずれかの値を受け入れる文字列パラメータ。これらの値は、指定されたカーソルをオプティマイザがどのように処理するかを示します。 READ-ONLY ― このカーソルは読み込み専用。 READ-WRITE (デフォルト) ― このカーソルは読み込みや書き込みが可能。

戻り値
LONG VARCHAR

注意： 結果データ型は LONG VARCHAR です。**SELECT INTO** 文で **GRAPHICAL_PLAN** を使用する場合は、非構造化データ分析オプションのライセンスを所有しているか、**CAST** を使用して **GRAPHICAL_PLAN** を正しいデータ型とサイズに設定する必要があります。

使用法
GRAPHICAL_PLAN 関数に引数を指定しない場合、クエリ・プランはキャッシュから返されます。キャッシュにクエリ・プランがない場合は、次のメッセージが表示されます。

```
plan not available
```

GRAPHICAL_PLAN 関数の動作は、データベース・オプション QUERY_PLAN_TEXT_ACCESS と QUERY_PLAN_TEXT_CACHING で制御されます。QUERY_PLAN_TEXT_ACCESS が OFF (デフォルト) の場合、次のメッセージが表示されます。

```
Plan not available. The database option QUERY_PLAN_TEXT_ACCESS is OFF
```

ユーザがプランにアクセスする必要がある場合、DBA はそのユーザに対してオプション QUERY_PLAN_TEXT_ACCESS を ON に設定する必要があります。

『リファレンス：文とオプション』の「データベース・オプション」>「アルファベット順のオプション・リスト」>「QUERY_PLAN_TEXT_ACCESS オプション」と「QUERY_PLAN_TEXT_CACHING オプション」を参照してください。

QUERY_PLAN_TEXT_ACCESS が ON であり、しかもサーバに保持されているキャッシュに文字列式のクエリ・プランがある場合は、キャッシュからクエリ・プランが返されます。

クエリ・プランがキャッシュになく、ユーザがクライアント上でプランを表示する許可を得ている場合、オプティマイザの見積もりがあるクエリ・プランが生成され (NOEXEC オプションが ON のクエリ・プラン)、Interactive SQL クライアント・プラン・ウィンドウに表示されます。

『リファレンス：文とオプション』の「データベース・オプション」>「アルファベット順のオプション・リスト」>「NOEXEC オプション」を参照してください。

まだ実行されたことがないクエリ・プランを要求しても、そのクエリ・プランはキャッシュにないので、 代わりにオプティマイザの見積もりがあるクエリ・プランが返されます。ただし、これには QUERY_PLAN_AFTER_RUN 統計はありません。

『リファレンス：文とオプション』の「データベース・オプション」>「アルファベット順のオプション・リスト」>「QUERY_PLAN_AFTER_RUN オプション」を参照してください。

ストアド・プロシージャのクエリ・プランには、**GRAPHICAL_PLAN** 関数を使用してアクセスできません。

Sybase IQ クエリのために開くカーソルのクエリ・プランを表示できます。カーソルは、**DECLARE CURSOR** コマンドと **OPEN CURSOR** コマンドによって宣言されて

開かれます。 直近に開いたカーソルのクエリ・プランを取得するには、次の文を使用します。

```
SELECT GRAPHICAL_PLAN ( );
```

QUERY_PLAN_AFTER_RUN オプションが OFF の場合、プランは **OPEN CURSOR** または **CLOSE CURSOR** の実行後に表示されます。ただし、QUERY_PLAN_AFTER_RUN が ON の場合は、**CLOSE CURSOR** を実行してから、プランを要求する必要があります。

Interactive SQL の [プラン・ビューワ] ウィンドウで SQL 文に対するクエリ・オプティマイザの実行プランを表示する方法については、『SQL Anywhere サーバー - データベース管理』の「データベースの管理」>「SQL Anywhere グラフィカル管理ツール」>「Interactive SQL の使用」>「プランビューアを使用したグラフィカルなプランの表示」を参照してください。

例

次の例は **SELECT** 文を文字列パラメータとして渡し、クエリを実行するためのプランを返します。プランはファイル gplan.xml に保存されます。

注意： フォーマットされたプラン出力を得る際に、**OUTPUT** 文の **HEXADECIMAL** 句を **ASIS** に設定すると、各文字の値に制御文字が含まれている場合でも、文字の値はエスケープされずに書き出されます。**ASIS** は、テキストにタブや復帰改行などのフォーマット文字列が含まれる場合に使用します。

```
SELECT GRAPHICAL_PLAN ('SELECT * FROM Employees');OUTPUT to 'C:
¥gplan.xml' HEXADECIMAL ASIS quote '';
```

次の例は、クエリ・プランがキャッシュにある場合、そこから返します。

```
SELECT GRAPHICAL_PLAN ( );
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise によるサポートなし。

参照：

- HTML_PLAN 関数 [文字列] (216 ページ)

GROUPING 関数 [集合]

ROLLUP または **CUBE** 演算の結果セット内のカラムが NULL である場合、その理由が小計ローの一部であるためか、または基本データによるためかを識別します。

構文

GROUPING (*group-by-expression*)

パラメータ

パラメータ	説明
group-by-expression	GROUP BY 句に ROLLUP または CUBE キーワードを使用したクエリの結果セットにおいて、グループ化カラムとして表示される式。この関数を使用すると、ROLLUP 演算または CUBE 演算によって結果セットに追加された小計ローを識別することができます。

現在、Sybase IQ は、**PERCENTILE_CONT** 関数および **PERCENTILE_DISC** 関数での **GROUP BY CUBE** 演算をサポートしていません。

戻り値

値	説明
1	group-by-expression が、小計ローの一部なので NULL になっていることを示します。カラムは、小計ローの前にあるカラムではありません。
0	group-by-expression が小計ローの前にあるカラムであることを示します。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise によるサポートなし。

参照：

- ウィンドウ集合関数の使用法 (111 ページ)

GROUP_MEMBER 関数 [システム]

ユーザが指定されたグループに属しているかどうかを示します。

構文

```
GROUP_MEMBER ( group-name-string-expression[ , user-name-string-expression ] )
```

パラメータ

パラメータ	説明
group-name-string-expression	対象となるグループを示します。

パラメータ	説明
user-name-string-expression	対象となるユーザを示します。指定しなければ、現在のユーザ名とみなされます。

戻り値

表 71 : 戻り値

値	説明
0	グループが存在しない場合、ユーザが存在しない場合、またはユーザが指定されたグループに属していない場合は 0 を返します。
1	ユーザが指定されたグループのメンバである場合は 0 以外の整数を返します。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise ではサポートされていません。

HEXTOBIGINT 関数 [データ型変換]

16 進文字列に相当する BIGINT を返します。

構文

HEXTOBIGINT (*hexadecimal-string*)

パラメータ

パラメータ	説明
hexadecimal-string	big integer (BIGINT) に変換される 16 進数値です。次のように入力します。先頭に小文字または大文字の 0x を付けるか、または省略することもできます。 `0x`*hex-string* `0X`*hex-string* *hex-string*

例

次の文を実行すると、値 4294967287 が返ります。

```
SELECT HEXTOBIGINT ( '0xfffffff7' ) FROM iq_dummy
SELECT HEXTOBIGINT ( '0Xfffffff7' ) FROM iq_dummy
```

```
SELECT HEXTOBIGINT ( 'fffffff7' ) FROM iq_dummy
```

使用法

HEXTOBIGINT 関数は 16 進数の整数を受け入れ、等価の BIGINT を返します。16 進数の整数は CHAR および VARCHAR 値の式としてだけでなく、BINARY および VARBINARY の式としても指定できます。

HEXTOBIGINT 関数は有効な 16 進文字列 (0x または 0X プレフィクスの有無に関係なく引用符で囲む) を受け入れます。

16 桁未満の入力は左に 0 が埋め込まれるものとします。

入力時に変換に失敗すると、CONVERSION_ERROR オプションが OFF に設定されていないかぎり、Sybase IQ はエラーを返します。 CONVERSION_ERROR が OFF に設定されている場合は、無効な 16 進入力に対して NULL が返されます。

『リファレンス：文とオプション』の「データベース・オプション」>「アルファベット順のオプション・リスト」>「CONVERSION_ERROR オプション [TSQL]」を参照してください。

値に 0x が追加されている場合を除き、BINARY または VARBINARY 値では 8 バイトを超えるとエラーが返され、CHAR または VARCHAR 値では 16 文字を超えるとエラーが返されます。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise 互換。

参照：

- BIGINTTOHEX 関数 [データ型変換] (140 ページ)
- HEXTOINT 関数 [データ型変換] (211 ページ)
- INTTOHEX 関数 [データ型変換] (222 ページ)

HEXTOINT 関数 [データ型変換]

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

パラメータ	説明
hexadecimal-string	整数に変換される文字列。次のように入力します。先頭に小文字または大文字の x を付けるか、または省略することもできます。 `0x`*`hex-string`* `0X`*`hex-string`* *`hex-string`*

戻り値

HEXTOINT 関数は、プラットフォームに依存しない SQL INTEGER 相当の 16 進文字列を返します。右から 8 桁目が数値 8 ～ 9 か、大文字または小文字の A ～ F のいずれかであり、その前の桁がすべて大文字または小文字の F の場合は、16 進値が負の整数値になります。次の HEXTOINT は無効な使用例です。これは、引数が符号付き 32 ビット整数で表現できない正の整数値を示しているためです。

```
SELECT HEXTOINT( '0x0080000001' );
```

INT

例

次の文を実行すると、値 420 が返ります。

```
SELECT HEXTOINT ( '0x1A4' ) FROM iq_dummy
```

```
SELECT HEXTOINT ( '0X1A4' ) FROM iq_dummy
```

```
SELECT HEXTOINT ( '1A4' ) FROM iq_dummy
```

使用法

CONVERSION_ERROR オプションが OFF の場合を除き、無効な 16 進入力に対して Sybase IQ からエラーが返されます。 CONVERSION_ERROR が OFF に設定されている場合は、無効な 16 進入力に対して NULL が返されます。

『リファレンス：文とオプション』の「データベース・オプション」>「アルファベット順のオプション・リスト」>「CONVERSION_ERROR オプション [TSQL]」を参照してください。

データベース・オプション ASE_FUNCTION_BEHAVIOR は、**INTTOHEX** と **HEXTOINT** を含む Sybase IQ 関数の出力が Adaptive Server Enterprise 関数の出力と一致するように指定します。

『リファレンス：文とオプション』の「データベース・オプション」>「アルファベット順のオプション・リスト」>「ASE_FUNCTION_BEHAVIOR オプション」を参照してください。

ASE_FUNCTION_BEHAVIOR オプションが ON の場合は次の処理が行われます。

- Sybase IQ**HEXTOINT** では、入力は 8 文字の 16 進文字列であるとみなされます。長さが 8 文字未満の場合、文字列の左側には 0 が埋め込まれます。
- Sybase IQ**HEXTOINT** は、先頭に 0x が付いた最大 16 文字の文字列 (合計で 18 文字) を受け入れます。入力値が大きいと、整数値が 32 ビットの符号付き整数出力サイズをオーバフローする可能性があるため、注意が必要です。
- Sybase IQ**HEXTOINT** 関数の出力のデータ型は、32 ビットの符号付き整数とみなされます。
- Sybase IQ**HEXTOINT** 関数は、32 ビットの 16 進整数を符号付き表現として受け入れます。
- 8 文字を超える 16 進文字列の場合、Sybase IQ**HEXTOINT** は関連の文字のみを考慮します。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise 互換。

参照：

- BIGINTTOHEX 関数 [データ型変換] (140 ページ)
- HEXTOBIGINT 関数 [データ型変換] (210 ページ)
- INTTOHEX 関数 [データ型変換] (222 ページ)

HOUR 関数 [日付と時刻]

指定された日時から、時間 (hour) に対応する 0 から 23 までの数字を返します。

構文

```
HOUR ( datetime-expression )
```

表 72 : パラメータ

パラメータ	定義
datetime-expression	日時。

戻り値
SMALLINT

例
次の文は、値 21 を返します。

```
SELECT HOUR( '1998-07-09 21:12:13' ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise ではサポートされていません。

HOURS 関数 [日付と時刻]

任意の開始日時からの時間数を返すか、指定された 2 つの時刻の間の総時間数を返すか、または integer-expression で指定された時間数を時刻に追加します。

構文

```
HOURS ( datetime-expression
| datetime-expression, datetime-expression
| datetime-expression, integer-expression )
```

パラメータ

表 73 : パラメータ

パラメータ	説明
datetime-expression	日時。
integer-expression	*datetime-expression.* に加算する時間数。*integer-expression* が負の場合、日時から適切な時間数が引かれます。整数式を指定する場合、*datetime-expression* は日時データ型として明示的にキャストする必要があります。

戻り値
INT

例
次の文は、値 17518758 を返します。

```
SELECT HOURS( '1998-07-13 06:07:12' ) FROM iq_dummy
```

次の文を実行すると、2 つの時刻の間の差である値 4 が返されます。

```
SELECT HOURS( '1999-07-13 06:07:12',
    '1999-07-13 10:07:12' ) FROM iq_dummy
```

次の文を実行すると、日時の値 13.05.99 02:05:07.000 が返ります。

```
SELECT HOURS( CAST( '1999-05-12 21:05:07'
AS DATETIME ), 5 ) FROM iq_dummy
```

使用法
2 つ目の構文は、最初の引数の日時から 2 番目の引数の日時までの総時間数を返します。負の値が返ることもあります。

標準と互換性

- SQL － ISO/ANSI SQL 文法のベンダ拡張。
- Sybase － Adaptive Server Enterprise ではサポートされていません。

参照：
- CAST 関数 [データ型変換] (144 ページ)
- CONVERT 関数 [データ型変換] (155 ページ)
- MINUTES 関数 [日付と時刻] (248 ページ)
- MONTHS 関数 [日付と時刻] (251 ページ)
- REPLACE 関数 [文字列] (295 ページ)
- SECOND 関数 [日付と時刻] (306 ページ)
- WEEKS 関数 [日付と時刻] (357 ページ)
- YEAR 関数 [日付と時刻] (362 ページ)
- YEARS 関数 [日付と時刻] (363 ページ)

HTML_DECODE 関数 [HTTP]

HTML リテラル文字列に含まれる特殊文字のエンティティをデコードします。

構文

```
HTML_DECODE ( string )
```

注意： CIS 機能補正のパフォーマンスに関する考慮事項が適用されます。

HTML_DECODE 関数は SQL Anywhere 関数の 1 つです。 詳細については、『SQL Anywhere サーバー - SQL リファレンス』の「SQL 関数」>「関数」>「HTML_DECODE 関数 [HTTP]」を参照してください。

戻り値
LONG VARCHAR

注意： 結果データ型は LONG VARCHAR です。**SELECT INTO** 文で **HTML_DECODE** を使用する場合は、非構造化データ分析オプションのライセンスを所有しているか、**CAST** を使用して **HTML_DECODE** を正しいデータ型とサイズに設定する必要があります。

HTML_ENCODE 関数 [HTTP]

HTML ドキュメントに挿入する文字列内の特殊文字をエンコードします。

構文
HTML_ENCODE (*string*)

注意： CIS 機能補正のパフォーマンスに関する考慮事項が適用されます。

HTML_ENCODE 関数は SQL Anywhere 関数の 1 つです。 詳細については、『SQL Anywhere サーバー - SQL リファレンス』の「SQL 関数」>「関数」>「HTML_ENCODE 関数 [HTTP]」を参照してください。

戻り値
LONG VARCHAR

注意： 結果データ型は LONG VARCHAR です。**SELECT INTO** 文で **HTML_ENCODE** を使用する場合は、非構造化データ分析オプションのライセンスを所有しているか、**CAST** を使用して **HTML_ENCODE** を正しいデータ型とサイズに設定する必要があります。

HTML_PLAN 関数 [文字列]

クエリ・プランを HTML フォーマット文字列で返します。

構文
HTML_PLAN (*string-expression*)

注意： CIS 機能補正のパフォーマンスに関する考慮事項が適用されます。

パラメータ

パラメータ	説明
string-expression	プランを生成する SQL 文。通常は **SELECT** 文ですが、**UPDATE** 文と **DELETE** 文も可能です。

HTML_PLAN 関数に引数を指定しない場合、クエリ・プランはキャッシュから返されます。キャッシュにクエリ・プランがない場合は、次のメッセージが表示されます。

```
No plan available
```

HTML_PLAN 関数の動作は、データベース・オプション QUERY_PLAN_TEXT_ACCESS と QUERY_PLAN_TEXT_CACHING で制御されます。QUERY_PLAN_TEXT_ACCESS が OFF (デフォルト) の場合、次のメッセージが表示されます。

```
Plan not available. The database option QUERY_PLAN_TEXT_ACCESS is OFF
```

QUERY_PLAN_TEXT_ACCESS が ON であり、しかもサーバに保持されているキャッシュに文字列式のクエリ・プランがある場合は、キャッシュからクエリ・プランが返されます。

『リファレンス：文とオプション』の「データベース・オプション」>「アルファベット順のオプション・リスト」>「QUERY_PLAN_TEXT_ACCESS オプション」と「QUERY_PLAN_TEXT_CACHING オプション」を参照してください。

HTML_PLAN 関数を使用して、**SELECT**、**UPDATE**、**DELETE**、**INSERT SELECT**、**SELECT INTO** を使用した Interactive SQL にクエリ・プランを返すことができます。

Sybase IQ クエリのために開くカーソルのクエリ・プランを表示できます。直近に開いたカーソルのクエリ・プランを取得するには、次の文を使用します。

```
SELECT HTML_PLAN ( );
```

QUERY_PLAN_AFTER_RUN オプションが OFF の場合、プランは **OPEN CURSOR** または **CLOSE CURSOR** の実行後に表示されます。ただし、QUERY_PLAN_AFTER_RUN が ON の場合は、**CLOSE CURSOR** を実行してから、プランを要求する必要があります。

『リファレンス：文とオプション』の「データベース・オプション」>「アルファベット順のオプション・リスト」>「QUERY_PLAN_AFTER_RUN オプション」を参照してください。

Interactive SQL の [プラン・ビューワ] ウィンドウで SQL 文に対するクエリ・オプティマイザの実行プランを表示する方法については、『SQL Anywhere サーバー -

データベース管理』の「データベースの管理」>「SQL Anywhere グラフィカル管理ツール」>「Interactive SQL の使用」>「プランビューアーを使用したグラフィカルなプランの表示」を参照してください。

SQL Anywhere クエリまたは OMNI/CIS 分解クエリの **HTML_PLAN** を要求すると、次のメッセージが返されます。

```
No plan. HTML_PLAN function is not supported for this type of
statement or database.
```

例

次の例は **SELECT** 文を文字列パラメータとして渡し、クエリを実行するための HTML プランを返します。プランはファイル `hplan.html` に保存されます。

```
SELECT HTML_PLAN ('SELECT * FROM Employees'); OUTPUT to 'C:
¥hplan.html' HEXADECIMAL ASIS QUOTE '';
```

OUTPUT TO 句の **HEXADECIMAL ASIS** は、テキストにタブや復帰改行などのフォーマット文字列が含まれる場合に使用します。**ASIS** に設定すると、値はそのまま書き込まれます。値が制御文字を含む場合も、エスケープはされません。

次の例は、HTML クエリ・プランがキャッシュにある場合、そこから返します。

```
SELECT HTML_PLAN ( );
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise によるサポートなし。

参照：

- GRAPHICAL_PLAN 関数 [文字列] (206 ページ)

HTTP_DECODE 関数 [HTTP]

文字列内の特殊文字をデコードして、HTTP で使えるようにします。

構文

```
HTTP_DECODE ( string )
```

注意： CIS 機能補正のパフォーマンスに関する考慮事項が適用されます。

`HTTP_DECODE` 関数は SQL Anywhere 関数の 1 つです。 詳細については、『SQL Anywhere サーバー - SQL リファレンス』の「SQL 関数」>「関数」>「HTTP_DECODE 関数 [HTTP]」を参照してください。

HTTP_ENCODE 関数 [HTTP]

文字列内の特殊文字をエンコードして、HTTP で使えるようにします。

構文

```
HTTP_ENCODE ( string )
```

注意： CIS 機能補正のパフォーマンスに関する考慮事項が適用されます。

HTTP_ENCODE 関数は SQL Anywhere 関数の 1 つです。 詳細については、『SQL Anywhere サーバー - SQL リファレンス』の「SQL 関数」>「関数」>「HTTP_ENCODE 関数 [HTTP]」を参照してください。

HTTP_HEADER 関数 [HTTP]

HTTP ヘッダの値を取得します。

構文

```
HTTP_HEADER ( field-name )
```

注意： CIS 機能補正のパフォーマンスに関する考慮事項が適用されます。

HTTP_HEADER 関数は SQL Anywhere 関数の 1 つです。 詳細については、『SQL Anywhere サーバー - SQL リファレンス』の「SQL 関数」>「関数」>「HTTP_HEADER 関数 [HTTP]」を参照してください。

戻り値

LONG VARCHAR

注意： 結果データ型は LONG VARCHAR です。**SELECT INTO** 文で **HTTP_HEADER** を使用する場合は、非構造化データ分析オプションのライセンスを所有しているか、**CAST** を使用して **HTTP_HEADER** を正しいデータ型とサイズに設定する必要があります。

HTTP_VARIABLE 関数 [HTTP]

HTTP の変数の値を取得します。

構文

```
HTTP_VARIABLE ( var-name [ [ , instance ] , header-field )
```

注意： CIS 機能補正のパフォーマンスに関する考慮事項が適用されます。

HTTP_VARIABLE 関数は SQL Anywhere 関数の 1 つです。 詳細については、『SQL Anywhere サーバー - SQL リファレンス』の「SQL 関数」>「関数」>「HTTP_VARIABLE 関数 [HTTP]」を参照してください。

戻り値
LONG VARCHAR

注意： 結果データ型は LONG VARCHAR です。**SELECT INTO** 文で **HTTP_VARIABLE** を使用する場合は、非構造化データ分析オプションのライセンスを所有しているか、**CAST** を使用して **HTTP_VARIABLE** を正しいデータ型とサイズに設定する必要があります。

IFNULL 関数 [その他]

最初の NULL 以外の式、または NULL を返します。

最初の式が NULL 値の場合は、第 2 の式の値を返します。最初の式が NULL でない場合、第 3 の式の値を返します。第 3 の式が指定されておらず、最初の式が NULL でない場合は、NULL 値を返します。

構文

```
IFNULL ( expression1, expression2 [ , expression3 ] )
```

パラメータ

表 74 : パラメータ

パラメータ	説明
expression1	評価される式。この値によって、*expression2* と *expression3* のどちらを返すかが決まります。
expression2	*expression1* が NULL の場合の戻り値。
expression3	*expression1* が NULL でない場合の戻り値。

戻り値
返されるデータ型は、*expression-2* と *expression-3* のデータ型によって異なります。

例
次の文は、値 -66 を返します。

```
SELECT IFNULL( NULL, -66 ) FROM iq_dummy
```

次の文は、第 1 の式が NULL でなく、第 3 の式が指定されていないため、NULL を返します。

```
SELECT IFNULL( -66, -66 ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法の Transact-SQL 拡張。
- Sybase — Adaptive Server Enterprise ではサポートされていません。

INDEX_COL 関数 [システム]

インデックス・カラムの名前を返します。

構文

```
INDEX_COL ( table-name, index-id, key_# [ , user-id ] )
```

パラメータ

パラメータ	定義
table-name	テーブル名。
index-id	*table-name* のインデックスのインデックス ID。
key_#	*index-id* で指定されたインデックスのキー。このパラメータには、インデックス内のカラム番号を指定します。単一カラムのインデックスの場合、*key_#* は 0 になります。複数カラムのインデックスでは、1 つ目のカラムの *key_#* は 0、2 つ目のカラムは 1、というように指定します。
user-id	*table-name* の所有者のユーザ ID。*user-id* が省略された場合、この値は、デフォルトで、関数を呼び出したユーザの ID になります。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Sybase IQ 用に実装された Adaptive Server Enterprise 関数。

参照：

- OBJECT_ID 関数 [システム] (262 ページ)

INSERTSTR 関数 [文字列]

文字列を、別の文字列の指定された位置に挿入します。

構文

```
INSERTSTR ( numeric-expression, string-expression1, string-
expression2 )
```

パラメータ

表 75 : パラメータ

パラメータ	定義
numeric-expression	この位置の後に、*string-expression2* が挿入されます。先頭に文字列を挿入するには、ゼロを使用します。
string-expression1	*string-expression2* が挿入される文字列。
string-expression2	挿入する文字列。

戻り値
LONG VARCHAR

注意： 結果データ型は `LONG VARCHAR` です。**SELECT INTO** 文で **INSERTSTR** を使用する場合は、非構造化データ分析オプションのライセンスを所有しているか、**CAST** を使用して **INSERTSTR** を正しいデータ型とサイズに設定する必要があります。

例
次の文は、値 "backoffice" を返します。

```
SELECT INSERTSTR( 0, 'office ', 'back' ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise によるサポートなし。STUFF 関数が同等の機能を持っており、Adaptive Server Enterprise および Sybase IQ でサポートされています。

INTTOHEX 関数 [データ型変換]

10 進の整数を 16 進数に変換して返します。

構文

```
INTTOHEX ( integer-expression )
```

パラメータ

表 76 : パラメータ

パラメータ	説明
integer-expression	16 進数に変換される整数。

戻り値
VARCHAR

例
次の文は、値 3B9ACA00 を返します。

```
SELECT INTTOHEX( 1000000000 ) FROM iq_dummy
```

次の文は、値 00000002540BE400 を返します。

```
SELECT INTTOHEX ( 10000000000) FROM iq_dummy
```

使用法
INTTOHEX への入力データの変換に失敗すると、CONVERSION_ERROR オプションが OFF に設定されていない場合、Sybase IQ はエラーを返します。 OFF に設定されている場合は、NULL を返します。

『リファレンス：文とオプション』の「データベース・オプション」>「アルファベット順のオプション・リスト」>「CONVERSION_ERROR オプション [TSQL]」を参照してください。

データベース・オプション ASE_FUNCTION_BEHAVIOR は、**INTTOHEX** と **HEXTOINT** を含む Sybase IQ 関数の出力が Adaptive Server Enterprise 関数の出力と一致するように指定します。ASE_FUNCTION_BEHAVIOR のデフォルト値は OFF です。

『リファレンス：文とオプション』の「データベース・オプション」>「アルファベット順のオプション・リスト」>「ASE_FUNCTION_BEHAVIOR オプション」を参照してください。

ASE_FUNCTION_BEHAVIOR オプションが無効な場合 (値が OFF の場合) は、次のようになります。

- **INTTOHEX** の出力は、SQL Anywhere 互換です。
- **INTTOHEX** の出力は、入力に応じて 8 桁または 16 桁となり、左側に 0 が埋め込まれます。戻り値のデータ型は VARCHAR です。
- **INTTOHEX** の出力には、0x および 0X プレフィクスはありません。

- **INTTOHEX** には、64 ビットまでの整数を入力できます。

ASE_FUNCTION_BEHAVIOR オプションが有効な場合 (値が ON の場合) は、次のようになります。

- **INTTOHEX** の出力は Adaptive Server Enterprise 互換です。
- **INTTOHEX** の出力は常に 8 桁で、左側に 0 が埋め込まれます。戻り値のデータ型は VARCHAR です。
- **INTTOHEX** の出力には、0x および 0X プレフィクスはありません。
- Sybase IQ**INTTOHEX** では、入力を 32 ビットの符号付き整数とみなします。大きい値を指定するとオーバフローが発生し、変換エラーが起こる場合があります。 たとえば、

  ```
  SELECT INTTOHEX( 1000000000 ) FROM iq_dummy
  ```

  という文は 3B9ACA00 という値を返しますが、

  ```
  SELECT INTTOHEX( 10000000000 ) FROM iq_dummy
  ```

  では変換エラーが発生します。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise 互換。

参照：

- BIGINTTOHEX 関数 [データ型変換] (140 ページ)
- HEXTOBIGINT 関数 [データ型変換] (210 ページ)
- HEXTOINT 関数 [データ型変換] (211 ページ)

ISDATE 関数 [日付および時刻]

引数の文字列が、日付に変換可能かどうかを調べます。

変換可能であれば 1 が返され、可能でなければ 0 が返されます。引数が null に設定されている場合は、0 が返されます。

構文

```
ISDATE ( string )
```

パラメータ

表 77 : パラメータ

パラメータ	説明
string	文字列表現が日付として有効かどうかを調べて判断される文字列。

戻り値
INT

例
次の例では、`birth_date` カラムに格納されている値が有効な日付かどうかを調べ、無効な日付であれば NULL を返し、有効な日付であれば `date` フォーマットで返しています。

```
select birth_date from MyData;
------------------------------
1990/32/89
0101/32/89
1990/12/09
```

```
select
  case when isdate(birth_date)=0 then NULL
  else cast(birth_date as date)
  end
  from MyData;
-------------------------------------
(NULL)
(NULL)
1990-12-09
```

標準と互換性

- SQL － ISO/ANSI SQL 文法のベンダ拡張。
- Sybase：
  - SQL Anywhere は ISO 8601 データ交換形式を使用。
  - Adaptive Server Enterprise によるサポートなし。

ISNULL 関数 [その他]

パラメータ・リスト内で最初の NULL でない式の値を返します。

少なくとも 2 つの式を関数に渡す必要があります。

構文

```
ISNULL ( expression, expression [ …, expression ] )
```

パラメータ

表 78 : パラメータ

パラメータ	説明
expression	NULL かどうかが調べられる式。

戻り値
この関数の戻り値は、指定した式によって異なります。具体的には、データベース・サーバが関数を評価するとき、まず、式の比較が可能なデータ型を検索します。該当するデータ型が見つかると、データベース・サーバは式を比較し、比較に使用したデータ型で結果を返します。データベース・サーバは、一般に比較が可能なデータ型を見つけることができないと、エラーを返します。

例
次の文は、値 -66 を返します。

```
SELECT ISNULL( NULL ,-66, 55, 45, NULL, 16 ) FROM iq_dummy
```

使用法
ISNULL 関数は **COALESCE** 関数と同じです。

標準と互換性

- SQL — ISO/ANSI SQL 文法の Transact-SQL 拡張。
- Sybase — Adaptive Server Enterprise ではサポートされていません。

参照：
- COALESCE 関数 [その他] (151 ページ)

ISNUMERIC 関数 [その他]

引数の文字列が、数値に変換可能かどうかを調べます。

変換可能であれば 1 が返され、可能でなければ 0 が返されます。引数が null に設定されている場合は、0 が返されます。

構文

```
ISNUMERIC ( string )
```

注意： CIS 機能補正のパフォーマンスに関する考慮事項が適用されます。

パラメータ

表 79 : パラメータ

パラメータ	説明
string	文字列表現が数値として有効かどうかを調べて判断される文字列。

戻り値
INT

使用法
パフォーマンスを最適化するため、述部で **ISNUMERIC** を使用しないでください。使用した場合、製品の SQL Anywhere の部分で処理され、Sybase IQ のパフォーマンス機能を利用できません。

例
次の例では、`height_in_cms` カラムに格納されている値が有効な数値データかどうかを調べ、無効な数値データであれば NULL を返し、有効な数値データであれば `int` フォーマットで返しています。

```
data height_in_cms
------------------------
asde
asde
180
156
```

```
select case
   when isnumeric(height_in_cms)=0
   then NULL
   else cast(height_in_cms as int)
   end
from MyData
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise によるサポートなし。

LAG 関数 [統計]

テーブルまたはパーティション内の前のローの属性値を返す Interrow 関数。

構文

```
LAG (value_expr) [, offset [, default]]) OVER ([PARTITION BY  window
partition] ORDER BY  window ordering)
```

パラメータ

パラメータ	説明
value_expr	テーブルから返すオフセット・データを定義するテーブル・カラムまたは式。
offset	現在のローより上のロー数。負でない真数値リテラル、または真数値データの SQL 変数で表現します。許容範囲は 0 ～ 231 です。
default	*offset* 値がテーブルまたはパーティションのカーディナリティの範囲を超える場合に返される値。
ウィンドウ・パーティション	(オプション) 結果ロー・セットの分割方法を示す、カンマ区切りの 1 つ以上の値式。
ウィンドウ順序	ローのソートのための式を定義します。ウィンドウ・パーティションを指定した場合はウィンドウ・パーティション内でのソート、指定しなかった場合は結果セット内でのソートです。

使用法

LAG 関数は、**OVER** (**ORDER_BY**) ウィンドウ指定を必要とします。**OVER** (**ORDER_BY**) 句内のウィンドウ・パーティション句はオプションです。**OVER** (**ORDER_BY**) 句に、ウィンドウ・フレーム **ROWS**/**RANGE** 指定を含めることはできません。

value_expr に分析式を定義することはできません。したがって、分析関数をネストすることはできませんが、その他の組み込み関数式を *value_expr* に使用できます。

offset には、負でない数値データ型を入力する必要があります。**0** を入力すると、現在のローが返されます。負の数を入力すると、エラーが生成されます。

default のデフォルト値は **NULL** です。*default* のデータ型は、*value_expr* 値のデータ型に暗黙的に変換できる必要があります。変換できない場合、Sybase IQ は変換エラーを生成します。

例

次の例は、Employees テーブルから給与データを返し、部署 ID でデータをパーティションに分割し、入社日でデータを並べ替えます。**LAG** 関数は、前のロー (物理的なオフセットは 1 ロー) から給与を返し、それを **LAG (Salary)** カラムに表示します。

```
SELECT DepartmentID dID, StartDate, Salary, LAG(Salary, 1)
OVER(PARTITION BY dID ORDER BY StartDate) FROM Employees ORDER BY
1,2;
```

次の結果セットが返されます。

```
dID         StartDate    Salary       Lag(Salary)
=========  ===========  ==========  =============
100         1984-08-28   45,700.000  NULL
100         1985-01-01   62,000.000  45,700.000
100         1985-06-17   57,490.000  62,000.000
100         1986-06-07   72,995.000  57,490.000
100         1986-07-01   48,023.690  72,995.000
...
200         1985-02-03   38,500.000  NULL
200         1985-12-06   54,800.000  38,500.000
200         1987-02-19   39,300.000  54,800.000
200         1987-07-10   49,500.000  39,300.000
200         1988-10-04   54,600.000  49,500.000
200         1988-11-12   39,800.000  54,600.000
...
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。

参照：

- LEAD 関数 [統計] (232 ページ)

LAST_VALUE 関数 [集合]

一連の値の最後の値を返します。

構文

```
LAST_VALUE (expression [IGNORE NULLS | RESPECT NULLS])
OVER (window-spec)
```

パラメータ

パラメータ	定義
expression	順序付けられたセットの最後の値を決定する式。

戻り値

引数のデータ型。

使用法

LAST_VALUE は、(通常順序付けられた) 値のセット内の最後の値を返します。セットの最後の値が NULL の場合、IGNORE NULLS が指定されていなければ NULL が返されます。IGNORE NULLS を指定した場合、**LAST_VALUE** は、セット内の NULL ではない最後の値を返します。すべての値が NULL の場合は NULL が返されます。

戻り値のデータ型は、入力値のデータ型と同じです。

expression に **LAST_VALUE** やその他の分析関数を使用することはできません。したがって、分析関数をネストすることはできませんが、その他の組み込み関数式を expression に使用できます。

window-spec が **ORDER BY** 式を含まない場合、または **ORDER BY** 式の精度が不十分で一意の順序を確保できない場合、結果は不定になります。window-spec がない場合も、結果が不定になります。

関数構文 (インライン) または **SELECT** 文の **WINDOW** 句で、*window-spec* の要素を指定できます。

注意： DISTINCT はサポートされていません。

例
次の例は、各従業員の給与、およびその部署で最も高い給与の従業員の名前を返します。

```
SELECT GivenName + ' ' + Surname AS employee_name,
    Salary, DepartmentID,
    LAST_VALUE( employee_name ) OVER Salary_Window AS
highest_paid
FROM Employees
WINDOW Salary_Window AS ( PARTITION BY DepartmentID ORDER BY Salary
    RANGE BETWEEN UNBOUNDED PRECEDING
    AND UNBOUNDED FOLLOWING )
ORDER BY DepartmentID DESC;
```

次の結果セットが返されます。

表 80 : LAST_VALUE 結果セット

employee_name	Salary	DepartmentID	highest_paid
Michael Lynch	24,903.000	500	Jose Martinez
Joseph Barker	27,290.000	500	Jose Martinez
Sheila Romero	27,500.000	500	Jose Martinez
Felicia Kuo	28,200.000	500	Jose Martinez
Jeannette Bertrand	29,800.000	500	Jose Martinez
Jane Braun	34,300.000	500	Jose Martinez
Anthony Rebeiro	34,576.000	500	Jose Martinez
Charles Crowley	41,700.000	500	Jose Martinez
Jose Martinez	55,500.800	500	Jose Martinez
Doug Charlton	28,300.000	400	Scott Evans

employee_name	Salary	DepartmentID	highest_paid
Elizabeth Lambert	29,384.000	400	Scott Evans
Joyce Butterfield	34,011.000	400	Scott Evans
Robert Nielsen	34,889.000	400	Scott Evans
Alex Ahmed	34,992.000	400	Scott Evans
Ruth Wetherby	35,745.000	400	Scott Evans
...	...	...	...

標準と互換性

- SQL — ISO/ANSI SQL 準拠。SQL/OLAP 機能 T612 です。
- Sybase — SQL Anywhere 互換。

参照：
- ウィンドウ集合関数の使用法 (111 ページ)

LCASE 関数 [文字列]

文字列内のすべての文字を小文字に変換します。

構文

```
LCASE ( string-expression )
```

パラメータ

表 81 : パラメータ

パラメータ	説明
string-expression	小文字に変換される文字列。

戻り値

CHAR

NCHAR

LONG VARCHAR

VARCHAR

NVARCHAR

注意： 結果データ型は LONG VARCHAR です。**SELECT INTO** 文で **LCASE** を使用する場合は、非構造化データ分析オプションのライセンスを所有しているか、**CAST** を使用して **LCASE** を正しいデータ型とサイズに設定する必要があります。

例
次の文を実行すると、値 "lower case" が返ります。

```
SELECT LCASE( 'LOWER CasE' ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — **LCASE** は Adaptive Server Enterprise でサポートされません。**LOWER** を使用すると同じ結果が得られます。

参照：
- LEFT 関数 [文字列] (234 ページ)
- LOWER 関数 [文字列] (242 ページ)
- REPLACE 関数 [文字列] (295 ページ)
- REVERSE 関数 [文字列] (299 ページ)
- RIGHT 関数 [文字列] (300 ページ)
- UCASE 関数 [文字列] (348 ページ)
- UPPER 関数 [文字列] (349 ページ)

LEAD 関数 [統計]

テーブルまたはパーティション内の後ろのローの属性値を返す Interrow 関数。

構文

```
LEAD (value_expr) [, offset [, default]]) OVER ([PARTITION BY window partition] ORDER BY window ordering)
```

パラメータ

パラメータ	説明
value_expr	テーブルから返すオフセット・データを定義するテーブル・カラムまたは式。
offset	現在のローより下のロー数。負でない真数値リテラル、または真数値データの SQL 変数で表現します。許容範囲は 0 ～ 231 です。

パラメータ	説明
default	*offset* 値がテーブルまたはパーティションの範囲を超える場合に返される値。
ウィンドウ・パーティション	(オプション) 結果ロー・セットの分割方法を示す、カンマ区切りの 1 つ以上の値式。
ウィンドウ順序	ローのソートのための式を定義します。ウィンドウ・パーティションを指定した場合はウィンドウ・パーティション内でのソート、指定しなかった場合は結果セット内でのソートです。

使用法

LEAD 関数は、**OVER** (**ORDER_BY**) ウィンドウ指定を必要とします。**OVER** (**ORDER_BY**) 句内のウィンドウ・パーティション句はオプションです。**OVER** (**ORDER_BY**) 句に、ウィンドウ・フレーム **ROWS**/**RANGE** 指定を含めることはできません。

value_expr に分析式を定義することはできません。したがって、分析関数をネストすることはできませんが、その他の組み込み関数式を *value_expr* に使用できます。

offset には、負でない数値データ型を入力する必要があります。**0** を入力すると、現在のローが返されます。負の数を入力すると、エラーが生成されます。

default のデフォルト値は **NULL** です。*default* のデータ型は、*value_expr* 値のデータ型に暗黙的に変換できる必要があります。変換できない場合、Sybase IQ は変換エラーを生成します。

例

次の例は、Employees テーブルから給与データを返し、部署 ID でデータをパーティションに分割し、入社日でデータを並べ替えます。**LEAD** 関数は、次のロー (物理的なオフセットは 1 ロー) から給与を返し、それを **LEAD (Salary)** カラムに表示します。

```
SELECT DepartmentID dID, StartDate, Salary, LEAD(Salary, 1)
OVER(PARTITION BY dID ORDER BY StartDate) FROM  Employees ORDER BY
1,2;
```

次の結果セットが返されます。

```
dID         StartDate    Salary       Lead(Salary)
=========  ===========  ==========  =============
100         1984-08-28    45,700.000    62,000.000
100         1985-01-01    62,000.000    57,490.000
100         1985-06-17    57,490.000    72,995.000
100         1986-06-07    72,995.000    48,023.690
...
100         1990-08-19    54,900.000    NULL
```

```
200         1985-02-03    38,500.000    39,300.000
200         1987-02-19    39,300.000    49,500.000
200         1987-07-10    49,500.000    54,600.000
200         1988-11-28    46,200.000    34,892.000
200         1989-06-01    34,892.000    87,500.000
...
200         1993-08-12    47,653.000    NULL
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。

参照：

- LAG 関数 [統計] (227 ページ)

LEFT 関数 [文字列]

文字列の先頭から、指定された数だけ文字を返します。

構文

```
LEFT ( string-expression, numeric-expression )
```

パラメータ

表 82 : パラメータ

パラメータ	説明
string-expression	文字列。
numeric-expression	返される文字数。

戻り値

LONG VARCHAR

LONG NVARCHAR

注意： 結果データ型は LONG VARCHAR です。**SELECT INTO** 文で **LEFT** を使用する場合は、非構造化データ分析オプションのライセンスを所有しているか、**CAST** を使用して **LEFT** を正しいデータ型とサイズに設定する必要があります。

例

次の文は、値 "choco" を返します。

```
SELECT LEFT( 'chocolate', 5 ) FROM iq_dummy
```

使用法

文字列にマルチバイト文字が含まれ、照合が適切に使用されている場合は、返されるバイト数が、指定された文字数よりも多くなることがあります。

注意： **LEFT** 関数の結果データ型は、LONG VARCHAR です。**SELECT INTO** 文で **LEFT** を使用する場合は、非構造化データ分析オプションのライセンスを所有しているか、**CAST** を使用して **LEFT** を正しいデータ型とサイズに設定する必要があります。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise ではサポートされていません。

参照：

- LCASE 関数 [文字列] (231 ページ)
- LOWER 関数 [文字列] (242 ページ)
- REPLACE 関数 [文字列] (295 ページ)
- REVERSE 関数 [文字列] (299 ページ)
- RIGHT 関数 [文字列] (300 ページ)
- UCASE 関数 [文字列] (348 ページ)
- UPPER 関数 [文字列] (349 ページ)

LEN 関数 [文字列]

BINARY データ型または STRING データ型の入力として 1 つの引数を取り、指定された文字列表現の後続ブランクを除いた、データベースの照合順序で定義されている文字数を返します。

マルチバイト文字セットの場合、結果が文字列のバイト長と異なる場合があります。

BINARY および VARBINARY も使用できます。その場合は、LEN() によって入力のバイト数が返されます。

LEN は、**LENGTH** 関数のエイリアスです。

構文

```
LEN ( string_expr )
```

パラメータ

表 83 : パラメータ

パラメータ	説明
string_expr	評価する文字列式。

例
次の例は、値 3152 を返します。

```
select len(Photo) from Productswhere ID = 500
```

使用法
この関数は、**CHAR_LENGTH** (*string_expression*) と同じです。

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。

標準と互換性

- SQL — ISO/ANSI SQL 文法の Transact-SQL 拡張。

参照：
- BIT_LENGTH 関数 [文字列] (141 ページ)
- BYTE_LENGTH 関数 [文字列] (142 ページ)
- CHAR_LENGTH 関数 [文字列] (148 ページ)
- COL_LENGTH 関数 [システム] (151 ページ)
- DATALENGTH 関数 [システム] (166 ページ)
- LENGTH 関数 [文字列] (236 ページ)
- OBJECT_NAME 関数 [システム] (263 ページ)
- OCTET_LENGTH 関数 [文字列] (264 ページ)
- STR_REPLACE 関数 [文字列] (325 ページ)

LENGTH 関数 [文字列]

指定された文字列に含まれる文字数を返します。

構文

```
LENGTH ( string-expression )
```

パラメータ

表 84 : パラメータ

パラメータ	説明
string-expression	文字列。

戻り値
INT

例
次の文は、値 9 を返します。

```
SELECT LENGTH( 'chocolate' ) FROM iq_dummy
```

使用法
文字列にマルチバイト文字があり、適切な照合が使用されている場合、**LENGTH** はバイト数ではなく、文字数を返します。文字列のデータ型が BINARY である場合、**LENGTH** 関数は **BYTE_LENGTH** と同じように機能します。

LENGTH 関数は **CHAR_LENGTH** 関数と同じです。

標準と互換性

- SQL － ISO/ANSI SQL 文法のベンダ拡張。
- Sybase － Adaptive Server Enterprise ではサポートされていません。代わりに、**CHAR_LENGTH** 関数を使用してください。

参照：
- BIT_LENGTH 関数 [文字列] (141 ページ)
- BYTE_LENGTH 関数 [文字列] (142 ページ)
- CHAR_LENGTH 関数 [文字列] (148 ページ)
- COL_LENGTH 関数 [システム] (151 ページ)
- DATALENGTH 関数 [システム] (166 ページ)
- LEN 関数 [文字列] (235 ページ)
- OBJECT_NAME 関数 [システム] (263 ページ)
- OCTET_LENGTH 関数 [文字列] (264 ページ)
- STR_REPLACE 関数 [文字列] (325 ページ)

LIST 関数 [集合]

カンマで区切られた値のリストを返します。

構文

```
LIST(
[DISTINCT] string-expresssion
[, 'delimiter-string']
[ORDER BY order-by-expression [ ASC | DESC ], ... ] )
```

注意： CIS 機能補正のパフォーマンスに関する考慮事項が適用されます。

LIST 関数 [集合] は SQL Anywhere 関数の 1 つです。詳細については、『SQL Anywhere サーバー - SQL リファレンス』の「SQL 関数」>「関数」>「LIST 関数 [集合]」を参照してください。

戻り値
LONG VARCHAR

注意： 結果データ型は LONG VARCHAR です。**SELECT INTO** 文で **LIST** を使用する場合は、非構造化データ分析オプションのライセンスを所有しているか、**CAST** を使用して **LIST** を正しいデータ型とサイズに設定する必要があります。

LN 関数 [数値]

指定された式の自然対数を返します。

構文

```
LN ( numeric-expression )
```

パラメータ

パラメータ	説明
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。他のデータ型を指定すると、**LN** 関数ではエラーが返ります。戻り値は、DOUBLE データ型です。

使用法
LN は 1 つの引数を取ります。たとえば **LN** (*20*) は、2.995732 を返します。

LN 関数は **LOG** 関数のエイリアスです。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise ではサポートされていません。代わりに、LOG 関数を使用してください。

参照：
- LOG 関数 [数値] (240 ページ)
- LOG10 関数 [数値] (241 ページ)

LOCATE 関数 [文字列]

ある文字列の、別の文字列内の位置を返します。

構文

```
LOCATE ( string-expression1, string-expression2
[ , numeric-expression ] )
```

パラメータ

パラメータ	説明
string-expression1	検索される文字列。
string-expression2	検索する文字列。255 バイトまでの文字列を指定してください。
numeric-expression	文字列内で検索を開始する文字位置。最初の文字の位置は 1 です。開始オフセットが負の場合、**LOCATE** は、最初ではなく最後にマッチする文字列のオフセットを返します。負のオフセットは、文字列の末尾から何文字を検索から除外するかを示します。除外されるバイト数は、(-1 * offset) - 1 で計算されます。

numeric-expression は、CHAR カラム、VARCHAR カラム、BINARY カラムの 32 ビット符号付き整数です。

戻り値
INT

例
次の文は、値 8 を返します。

```
SELECT LOCATE( 'office party this week – rsvp as soon as possible',
'party', 2 ) FROM iq_dummy
```

2 つ目の例は、検索の開始オフセットである *numeric-expression* に負の数が指定されています。

```
CREATE TABLE t1(name VARCHAR(20), dirname VARCHAR(60));
  INSERT INTO t1     VALUES('m1000','c:¥test¥functions¥locate.sql');
  INSERT INTO t1     VALUES('m1001','d:¥test¥functions¥trim.sql');
COMMIT;

SELECT LOCATE(dirname, '¥', -1), dirname FROM t1;
```

結果は次のようになります。

```
18   c:¥test¥functions¥locate.sql
18   d:¥test¥functions¥trim.sql
```

使用法

numeric-expression を指定した場合は、検索対象の文字列内のそのオフセット位置から検索を開始します。

numeric-expression を指定しなかった場合、**LOCATE** は、指定した文字列の最初のインスタンスの位置だけを返します。

最初の文字列には、サイズの大きい (255 バイトより大きい) 文字列を指定できますが、2 番目の文字列は 255 バイト以下である必要があります。2 番目の引数にサイズの大きい文字列が指定されると、関数からは NULL 値が返されます。

引数のどれか 1 つでも NULL の場合、結果は NULL になります。

長さが 0 の文字列を検索すると、1 が返されます。

指定した文字列が文字列に含まれない場合、**LOCATE** はゼロ (0) を返します。

LOCATE 関数で返されるか指定される位置またはオフセットはすべて、常に文字オフセットであり、マルチバイト・データの場合はバイト・オフセットとは異なることがあります。

非構造化データ分析機能の使用ライセンスを取得している場合は、この関数でラージ・オブジェクト・データを使用できます。

詳細については、『Sybase IQ の非構造化データ分析の概要』の「関数のサポート」を参照してください。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise によるサポートなし。

参照：

- PATINDEX 関数 [文字列] (265 ページ)
- LIKE 条件 (43 ページ)

LOG 関数 [数値]

数値の自然対数を返します。

LN は、**LOG** のエイリアスです。

構文

```
LOG ( numeric-expression )
```

パラメータ

表 85 : パラメータ

パラメータ	説明
numeric-expression	数値。

戻り値
この関数は、引数を DOUBLE に変換し、計算を倍精度浮動小数点で行い、結果を DOUBLE で返します。パラメータが NULL 値の場合、結果は NULL 値になります。

例
次の文は、値 3.912023 を返します。

```
SELECT LOG( 50 ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise 互換。

参照：
- LN 関数 [数値] (238 ページ)
- LOG10 関数 [数値] (241 ページ)

LOG10 関数 [数値]

数値の、底を 10 とする対数を返します。

構文

```
LOG10 ( numeric-expression )
```

パラメータ

表 86 : パラメータ

パラメータ	説明
numeric-expression	数値。

戻り値
この関数は、引数を DOUBLE に変換し、倍精度浮動小数点で計算を行います。パラメータが NULL 値の場合、結果は NULL 値になります。

例
次の式を実行すると、値 1.698970 が返ります。

```
SELECT LOG10( 50 ) FROM iq_dummy
```

標準と互換性

- SQL ― ISO/ANSI SQL 文法のベンダ拡張。
- Sybase ― Adaptive Server Enterprise 互換。

参照：
- LN 関数 [数値] (238 ページ)
- LOG 関数 [数値] (240 ページ)

LOWER 関数 [文字列]

文字列内のすべての文字を小文字に変換します。

構文

```
LOWER ( string-expression )
```

パラメータ

表 87 : パラメータ

パラメータ	説明
string-expression	変換される文字列。

戻り値
CHAR

NCHAR

LONG VARCHAR

VARCHAR

NVARCHAR

注意： 結果データ型は LONG VARCHAR です。**SELECT INTO** 文で **LOWER** を使用する場合は、非構造化データ分析オプションのライセンスを所有しているか、**CAST** を使用して **LOWER** を正しいデータ型とサイズに設定する必要があります。

例
次の文を実行すると、値 "lower case" が返ります。

```
SELECT LOWER( 'LOWER CasE' ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 準拠。
- Sybase － Adaptive Server Enterprise 互換。

参照：
- LCASE 関数 [文字列] (231 ページ)
- LEFT 関数 [文字列] (234 ページ)
- REPLACE 関数 [文字列] (295 ページ)
- REVERSE 関数 [文字列] (299 ページ)
- RIGHT 関数 [文字列] (300 ページ)
- UCASE 関数 [文字列] (348 ページ)
- UPPER 関数 [文字列] (349 ページ)

LTRIM 関数 [文字列]

文字列から先行ブランクを削除します。

構文

LTRIM (*string-expression*)

パラメータ

パラメータ	説明
string-expression	削除される文字列。

戻り値
VARCHAR

NVARCHAR

LONG VARCHAR

LONG NVARCHAR

注意： 結果データ型は LONG VARCHAR です。**SELECT INTO** 文で **LTRIM** を使用する場合は、非構造化データ分析オプションのライセンスを所有しているか、**CAST** を使用して **LTRIM** を正しいデータ型とサイズに設定する必要があります。

例
次の文は、先行ブランクがすべて削除された値 "Test Message" を返します。

```
SELECT LTRIM( '     Test Message' ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise 互換。

参照：

- RTRIM 関数 [文字列] (305 ページ)

MAX 関数 [集合]

ローの各グループ内での *expression* の最大値を返します。

構文

```
MAX ( expression
| DISTINCT  column-name )
```

パラメータ

パラメータ	説明
expression	最大値が計算される式。通常はカラムを指定します。
DISTINCT column-name	MAX (expression) を使用した場合と戻り値は同じです。万全を期すために含まれています。

戻り値

引数と同じデータ型。

例

次の文は、`Employees` テーブル内の salary の最大値 138948.000 を返します。

```
SELECT MAX ( Salary )
FROM Employees
```

使用法

expression が NULL になるローは無視されます。ローがまったくないグループに対しては、NULL 値を返します。

標準と互換性

- SQL — ISO/ANSI SQL 準拠。
- Sybase — Adaptive Server Enterprise 互換。

参照：

- ウィンドウ集合関数の使用法 (111 ページ)

- MIN 関数 [集合] (246 ページ)

MEDIAN 関数 [集合]

式の中央値を返します。

構文 1

```
MEDIAN([ALL | DISTINCT] expression)
```

構文 2

```
MEDIAN([ALL | DISTINCT] expression)
OVER (window-spec)
```

パラメータ

パラメータ	説明
expression	計算する中央値の数値式。

使用法

中央値は、標本分布、人口分布、または確率分布の上半分と下半分を分割する数値です。

戻り値のデータ型は、入力値のデータ型と同じです。NULL は、中央値の計算で無視されます。オプションのキーワード **DISTINCT** を使用して、集合関数を適用する前に重複値を排除できます。すべてのローにオペレーションを実行する **ALL** がデフォルトです。

注意： ROLLUP と CUBE は、構文 1 の **GROUP BY** 句ではサポートされていません。

構文 2 は、**SELECT** 文でウィンドウ関数として使用する場合の用法を示しています。その場合は、関数構文 (インライン) または **SELECT** 文の **WINDOW** 句で、*window-spec* の要素を指定できます。

注意： *window-spec* には、**ROW**、**RANGE**、**ORDER BY** 指定を含めることはできません。*window-spec* では **PARTITION** 句のみ指定できます。**WINDOW** 句を使用する場合、DISTINCT はサポートされません。

例

次のクエリは、フロリダの各部署の給与の中央値を返します。

```
SELECT DepartmentID, Surname, Salary,
MEDIAN(Salary) OVER (PARTITION BY DepartmentID) "Median"
FROM Employees
WHERE State IN ('FL')
```

結果は次のようになります。

表 88 : MEDIAN 結果セット

DepartmentID	Surname	Salary	Median
100	Lull	87,900.000	73,870.000
100	Gowda	59,840.000	73,870.000
200	Sterling	64,900.000	76,200.000
200	Kelly	87,500.000	76,200.000
300	Litton	58,930.000	58,930.000
400	Francis	53,870.000	53,870.000
400	Charlton	28,300.000	53,870.000
400	Evans	68,940.000	53,870.000

標準と互換性

- SQL － ISO/ANSI SQL 文法のベンダ拡張。

参照：
- ウィンドウ集合関数の使用法 (111 ページ)

MIN 関数 [集合]

ローの各グループ内での expression の最小値を返します。

構文

```
MIN ( expression
| DISTINCT  column-name )
```

パラメータ

パラメータ	説明
expression	最小値が計算される式。通常はカラムを指定します。
DISTINCT column-name	MIN (expression) を使用した場合と戻り値は同じです。万全を期すために含まれています。

戻り値
引数と同じデータ型。

例
次の文は、Employees テーブル内の salary の最小値である値 24903.000 を返します。

```
SELECT MIN ( Salary )
FROM Employees
```

使用法
expression が NULL になるローは無視されます。ローがまったくないグループに対しては、NULL 値を返します。

標準と互換性

- SQL — ISO/ANSI SQL 準拠。
- Sybase — Adaptive Server Enterprise 互換。

参照：
- ウィンドウ集合関数の使用法 (111 ページ)
- MAX 関数 [集合] (244 ページ)

MINUTE 関数 [日付と時刻]

指定された日時の分に対応する 0 から 59 までの数値を返します。

構文

```
MINUTE ( datetime-expression )
```

パラメータ

パラメータ	説明
datetime-expression	日時の値。

戻り値
SMALLINT

例
次の文は、値 22 を返します。

```
SELECT MINUTE( '1998-07-13 12:22:34' ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise 互換。

MINUTES 関数 [日付と時刻]

任意の日時からの分の数を返すか、指定された 2 つの時刻の間の分の数 (何分以上あるか) を返すか、または integer-expression で指定された分の数を時刻に追加します。

構文

```
MINUTES ( datetime-expression
| datetime-expression, datetime-expression
| datetime-expression, integer-expression )
```

パラメータ

パラメータ	説明
datetime-expression	日時。
integer-expression	*datetime-expression* に追加する分数。*integer-expression* が負の場合、日時から適切な分数が引かれます。整数式を指定する場合は、*datetime-expression* を `datetime` データ型として明示的にキャストする必要があります。

戻り値
INT

TIMESTAMP

例
値として 1051125487 を返す。

```
SELECT MINUTES( '1998-07-13 06:07:12' ) FROM iq_dummy
```

2 つの時刻の差である値 240 が返されます。

```
SELECT MINUTES( '1999-07-13 06:07:12',
    '1999-07-13 10:07:12' ) FROM iq_dummy
```

日時の値 1999-05-12 21:10:07.000 が返されます。

```
SELECT MINUTES( CAST( '1999-05-12 21:05:07'
AS DATETIME ), 5) FROM iq_dummy
```

使用法
2 つ目の構文は、最初の引数の日時から 2 番目の引数の日時までが何分以上あるかを返します。負の値が返ることもあります。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise ではサポートされていません。

参照：

- CAST 関数 [データ型変換] (144 ページ)
- CONVERT 関数 [データ型変換] (155 ページ)
- HOURS 関数 [日付と時刻] (214 ページ)
- MONTHS 関数 [日付と時刻] (251 ページ)
- REPLACE 関数 [文字列] (295 ページ)
- SECOND 関数 [日付と時刻] (306 ページ)
- WEEKS 関数 [日付と時刻] (357 ページ)
- YEAR 関数 [日付と時刻] (362 ページ)
- YEARS 関数 [日付と時刻] (363 ページ)

MOD 関数 [数値]

ある整数を別の整数で割った余りを返します。

構文

```
MOD ( dividend, divisor )
```

パラメータ

パラメータ	説明
dividend	割られる数。つまり分数の分子。
divisor	割る数。つまり分数の分母。

戻り値

SMALLINT

INT

NUMERIC

例

次の文は、値 2 を返します。

```
SELECT MOD( 5, 3 ) FROM iq_dummy
```

使用法
負の *dividend* が含まれる除算の場合、結果は負または 0 になります。*divisor* の符号は計算結果に影響を与えません。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise によるサポートなし。Adaptive Server Enterprise では、% 演算子をモジュロ演算子として使用します。

参照：
- REMAINDER 関数 [数値] (293 ページ)

MONTH 関数 [日付と時刻]

指定された日付の月に対応する 1 から 12 までの数字を返します。

構文
```
MONTH ( date-expression )
```

パラメータ

パラメータ	説明
date-expression	日時の値。

戻り値
SMALLINT

例
次の文は、値 7 を返します。
```
SELECT MONTH( '1998-07-13' ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise ではサポートされていません。

MONTHNAME 関数 [日付と時刻]

指定された日付式から月の名前を返します。

構文
```
MONTHNAME ( date-expression )
```

パラメータ

パラメータ	説明
date-expression	日時の値。

戻り値
VARCHAR

例
DATE_ORDER オプションにデフォルト値の *ymd* が設定されている場合、次の文は、**September** を返します。

```
SELECT MONTHNAME( '1998-09-05' ) FROM iq_dummy
```

『リファレンス：文とオプション』の「データベース・オプション」>「アルファベット順のオプション・リスト」>「DATE_ORDER オプション」を参照してください。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise ではサポートされていません。

MONTHS 関数 [日付と時刻]

任意の開始日時からの月数を返すか、指定された 2 つの日時の間の月数を返すか、または integer-expression で指定された月数を日時に追加します。

構文

```
MONTHS ( date-expression
| date-expression, datetime-expression
| date-expression, integer-expression )
```

パラメータ

パラメータ	説明
date-expression	日時。
integer-expression	*date-expression* に追加する月数。*integer-expression* が負の場合、日時から適切な月数が引かれます。整数式を指定する場合は、*date-expression* を datetime データ型として明示的にキャストする必要があります。

戻り値
INT

TIMESTAMP

例
次の文は、値 23982 を返します。

```
SELECT MONTHS( '1998-07-13 06:07:12' ) FROM iq_dummy
```

次の文を実行すると、2 つの日付の間の差である値 2 が返されます。

```
SELECT MONTHS( '1999-07-13 06:07:12',
    '1999-09-13 10:07:12' ) FROM iq_dummy
```

次の文を実行すると、日時の値 1999-10-12 21:05:07.000 が返ります。

```
SELECT MONTHS( CAST( '1999-05-12 21:05:07'
AS DATETIME ), 5) FROM iq_dummy
```

使用法
最初の構文は、任意の開始日から経過した月数を返します。この値は、2 つの日付／時刻式が同年同月かどうかを判別する場合に役立ちます。

```
MONTHS( invoice_sent ) = MONTHS( payment_received )
```

MONTH 関数の比較は、間違って、請求書が送られてから 12 か月後の支払も含む可能性があります。

2 番目の構文は、最初の引数の日付から 2 番目の引数の日付までの月数を返します。負の値が返ることもあります。MONTHS 関数の値は、2 つの日付の間にある月初めの日 (各月の 1 日) の数から算出されます。時間、分、秒は無視されます。

3 番目の構文は、指定された日付に *integer-expression* の月数を追加します。新しい日付が月末 (**MONTHS** ('1992-01-31', 1) など) を過ぎている場合、結果を演算結果の月の末日に設定します。*integer-expression* が負の場合、日時から適切な月数が引かれます。時間、分、秒は無視されます。

標準と互換性

- SQL － ISO/ANSI SQL 文法のベンダ拡張。
- Sybase － Adaptive Server Enterprise ではサポートされていません。

参照：
- CAST 関数 [データ型変換] (144 ページ)
- CONVERT 関数 [データ型変換] (155 ページ)
- HOURS 関数 [日付と時刻] (214 ページ)
- MINUTES 関数 [日付と時刻] (248 ページ)
- REPLACE 関数 [文字列] (295 ページ)
- SECOND 関数 [日付と時刻] (306 ページ)

- WEEKS 関数 [日付と時刻] (357 ページ)
- YEAR 関数 [日付と時刻] (362 ページ)
- YEARS 関数 [日付と時刻] (363 ページ)

NEWID 関数 [その他]

UUID (Universally Unique Identifier) 値を生成します。

返される UUID 値はバイナリです。UUID は GUID (Globally Unique Identifier) と同じです。

構文

```
NEWID ( )
```

パラメータ

NEWID() に指定するパラメータはありません。

戻り値

UNIQUEIDENTIFIER

例

次の文では、テーブル `t1` を作成して更新し、カラム `uid_col` の現在の値が NULL の場合、カラムの値を **NEWID** 関数で生成された一意な識別子に設定します。

```
CREATE TABLE t1 (uid_col int);
UPDATE t1
    SET uid_col = NEWID()
      WHERE uid_col IS NULL
```

次の文を実行します。

```
SELECT NEWID()
```

一意の識別子が `BINARY(16)` として返されます。たとえば、値は 0xd3749fe09cf446e399913bc6434f1f08 になります。この文字列を **UUIDTOSTR**() 関数を使用して読みやすい形式に変換できます。

使用法

NEWID() 関数は一意な識別子の値を生成します。

UUID を使うと、データベース内のオブジェクトを一意に識別できます。値は、1 台のコンピュータで生成された値が別のコンピュータで生成された値と一致しないように生成されます。このため、複写や同期の環境でキーとして使用することが可能です。

NEWID 関数は、次の位置でのみサポートされます。

- 上位レベルのクエリ・ブロックの **SELECT** リスト
- **UPDATE** 文の **SET** 句
- **INSERT...VALUES** の **VALUES** 句

NEWID 関数によって生成された値は、Sybase IQ テーブルのカラムのデフォルト値として使用できます。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise によるサポートなし。

参照：
- バイナリ・データ型 (84 ページ)
- STRTOUUID 関数 [文字列] (328 ページ)
- UUIDTOSTR 関数 [文字列] (351 ページ)
- 文字データ型 (75 ページ)
- バイナリ・データ型 (710 ページ)

NEXT_CONNECTION 関数 [システム]

次の接続番号を返します。パラメータが NULL の場合は、最初の接続番号を返します。

構文

```
NEXT_CONNECTION ( {connection-id }, {database-id } )
```

注意： CIS 機能補正のパフォーマンスに関する考慮事項が適用されます。

戻り値
INT

パラメータ

パラメータ	説明
connection-id	整数。通常は、前回の **NEXT_CONNECTION** の呼び出しで返された整数です。*connection-id* が NULL の場合、**NEXT_CONNECTION** は最新の接続 ID を返します
database-id	現在のサーバ上のデータベースの 1 つを表す整数です。*database-id* を指定しない場合は、現在のデータベースが使用されます。NULL を指定すると、**NEXT_CONNECTION** はデータベースに関係なく次の接続を返します。

使用法

NEXT_CONNECTION を使用して、データベースへの接続を列挙します。NULL を指定して最初の接続を取得し、前回の戻り値を指定して後続の各接続を取得します。それ以上接続がなくなると、NULL が返されます。

NEXT_CONNECTION を使用して、データベースへの接続を列挙できます。接続 ID は、通常、単純増加する昇順で作成されます。この関数は、反対の降順で次の接続 ID を返します。

最新の接続の接続 ID 値を取得するには、*connection-id* に NULL を入力します。その次の接続を取得するには、前の戻り値を入力します。その順序で次の接続がなくなると、NULL が返されます。

NEXT_CONNECTION は、特定の時点より前に作成されたすべての接続を切断する場合に便利です。ただし、**NEXT_CONNECTION** は接続 ID を降順で返すため、関数の開始後に作成された接続は返されません。すべての接続を確実に切断するには、新しい接続が作成されないようにしてから **NEXT_CONNECTION** を実行します。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。

例

次の文は、現在のデータベースの最初の接続の ID を返します。ID は 10 などの整数値です。

```
SELECT NEXT_CONNECTION( NULL );
```

次の文は、5 などの値を返します。

```
SELECT NEXT_CONNECTION( 10 );
```

次の呼び出しは、現在のデータベースについて、指定された *connection-id* から降順で次の接続 ID を返します。

```
SELECT NEXT_CONNECTION( connection-id );
```

次の呼び出しは、データベースに関係なく、指定された *connection-id* から降順で次の接続 ID を返します。

```
SELECT NEXT_CONNECTION( connection-id, NULL );
```

次の呼び出しは、指定されたデータベースについて、指定された *connection-id* から降順で次の接続 ID を返します。

```
SELECT NEXT_CONNECTION( connection-id, database-id );
```

次の呼び出しは、データベースに関係なく、最初の (最も古い) 接続を返します。

```
SELECT NEXT_CONNECTION( NULL, NULL );
```

次の呼び出しは、指定されたデータベースについて、最初の (最も古い) 接続を返します。

```
SELECT NEXT_CONNECTION( NULL, database-id );
```

NEXT_DATABASE 関数 [システム]

次のデータベース ID 番号を返します。パラメータが NULL の場合は、最初のデータベースを返します。

構文

```
NEXT_DATABASE ( { NULL | database-id } )
```

注意： CIS 機能補正のパフォーマンスに関する考慮事項が適用されます。

パラメータ

パラメータ	説明
database-id	データベースの ID 番号を指定する整数。

戻り値
INT

例
次の文は、最初のデータベースの値である 0 を返します。

```
SELECT NEXT_DATABASE( NULL ) FROM iq_dummy
```

次の文は NULL を返します。これは、サーバにそれ以上データベースが存在しないことを示します。

```
SELECT NEXT_DATABASE( 0 ) FROM iq_dummy
```

使用法
NEXT_DATABASE を使用すると、データベース・サーバ上で動作しているデータベースを列挙できます。NULL を指定して最初のデータベースを取得し、前回の戻り値を指定して後続の各データベースを取得します。それ以上データベースがなくなると、NULL が返されます。

標準と互換性

- SQL — ISO/ANSI SQL 文法の Transact-SQL 拡張。
- Sybase — Adaptive Server Enterprise ではサポートされていません。

参照：

- COL_NAME 関数 [システム] (152 ページ)
- DB_ID 関数 [システム] (188 ページ)
- DB_NAME 関数 [システム] (189 ページ)
- DB_PROPERTY 関数 [システム] (190 ページ)
- OBJECT_ID 関数 [システム] (262 ページ)
- OBJECT_NAME 関数 [システム] (263 ページ)

NEXT_HTTP_HEADER 関数 [HTTP]

次の HTTP ヘッダ名を取得します。

構文

```
NEXT_HTTP_HEADER ( header-name )
```

注意： CIS 機能補正のパフォーマンスに関する考慮事項が適用されます。

NEXT_HTTP_HEADER 関数は SQL Anywhere 関数の 1 つです。 詳細については、『SQL Anywhere サーバー - SQL リファレンス』の「SQL 関数」>「関数」>「NEXT_HTTP_HEADER 関数 [HTTP]」を参照してください。

注意： 参照先は SQL Anywhere のマニュアルです。

戻り値

LONG VARCHAR

注意： 結果データ型は LONG VARCHAR です。**SELECT INTO** 文で **NEXT_HTTP_HEADER** を使用する場合は、非構造化データ分析オプションのライセンスを所有しているか、**CAST** を使用して **NEXT_HTTP_HEADER** を正しいデータ型とサイズに設定する必要があります。

NEXT_HTTP_VARIABLE 関数 [HTTP]

次の HTTP の変数名を取得します。

構文

```
NEXT_HTTP_VARIABLE ( var-name )
```

注意： CIS 機能補正のパフォーマンスに関する考慮事項が適用されます。

NEXT_HTTP_VARIABLE 関数は SQL Anywhere 関数の 1 つです。 詳細については、『SQL Anywhere サーバー - SQL リファレンス』の「SQL 関数」>「関数」>「NEXT_HTTP_VARIABLE 関数 [HTTP]」を参照してください。

戻り値
LONG VARCHAR

注意： 結果データ型は LONG VARCHAR です。**SELECT INTO** 文で **NEXT_HTTP_VARIABLE** を使用する場合は、非構造化データ分析オプションのライセンスを所有しているか、**CAST** を使用して **NEXT_HTTP_VARIABLE** を正しいデータ型とサイズに設定する必要があります。

NOW 関数 [日付と時刻]

現在の日付と時刻を返します。これは **CURRENT TIMESTAMP** に対応する古い構文です。

構文

```
NOW ( * )
```

戻り値
TIMESTAMP

例
次の文は、現在の日付および時刻を返します。

```
SELECT NOW(*) FROM iq_dummy
```

標準と互換性

- SQL － ISO/ANSI SQL 文法のベンダ拡張。
- Sybase － Adaptive Server Enterprise ではサポートされていません。

NTILE 関数 [統計]

クエリ結果を指定された数のバケットに分散させ、バケット内の各ローにバケット番号を割り当てます。

構文

```
NTILE ( expression1 )
OVER ( ORDER BY expression2 [ ASC | DESC ] )
```

パラメータ

パラメータ	説明
expression1	バケットの数を指定する、1 ～ 32767 の定数。
expression2	ソートを指定します。カラムの参照、集合関数、またはこれらの項目を起動する式など、有効な式を何でも指定できます。

例
次に、**NTILE** 関数を使用してカー・ディーラの販売状況を調査する例を示します。販売した車の数に基づいて、ディーラが 4 つのグループに分類されています。ntile = 1 になっているのは、車の販売台数で上位 25% までのディーラです。

```
SELECT dealer_name, sales,
NTILE(4) OVER ( ORDER BY sales DESC )
FROM carSales;


dealer_name         sales       ntile
Boston              1000        1
Worcester           950         1
Providence          950         1
SF                  940         1
Lowell              900         2
Seattle             900         2
Natick              870         2
New Haven           850         2
Portland            800         3
Houston             780         3
Hartford            780         3
Dublin              750         3
Austin              650         4
Dallas              640         4
Dover               600         4
```

販売台数で上位 10% のカー・ディーラを調べるには、この例の **SELECT** 文に **NTILE(10)** と指定します。同様に、販売台数で 50% の販売店を調べるには、**NTILE(2)** を指定します。

使用法
NTILE はクエリ結果を指定された数のバケットに分散させ、バケット内の各ローにバケット番号を割り当てるランク付け統計関数です。結果セットを 100 分位(パーセンタイル)、10 分位(ディサイル)、4 分位(クォータイル)、またはその他の数のグループに分類できます。

NTILE には **OVER (ORDER BY)** 句が必須です。**ORDER BY** 句は、ランク付けを実行するパラメータと、各グループでローをソートする順序を指定します。この **ORDER BY** 句は **OVER** 句内でのみ使用するもので、**SELECT** 文の **ORDER BY** とは異なります。ランク付けクエリ内の集合関数で **DISTINCT** を指定することはできません。

OVER 句は、関数がクエリの結果セットに対して処理を行うことを示します。結果セットは、**FROM**、**WHERE**、**GROUP BY**、**HAVING** の各句がすべて評価された後で返されるローです。**OVER** 句は、ランク付け統計関数の計算の対象となるローのデータ・セットを定義します。

ASC と DESC のパラメータでは、昇順または降順の順序付けシーケンスを指定します。昇順がデフォルトです。

NTILE は、**SELECT** または **INSERT** 文の select リスト、および **SELECT** 文の **ORDER BY** 句でのみ使用できます。**NTILE** は、ビューまたは union に含めることができます。**NTILE** 関数は、サブクエリ、**HAVING** 句、**UPDATE** 文や **DELETE** 文の select リストでは使用できません。1 つのクエリで使用可能な **NTILE** 関数は、1 つだけです。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise または SQL Anywhere によるサポートなし。

参照：

- PERCENTILE_CONT 関数 [統計] (268 ページ)
- PERCENTILE_DISC 関数 [統計] (271 ページ)
- YEAR 関数 [日付と時刻] (362 ページ)

NULLIF 関数 [その他]

式を比較します。**CASE** 式の省略形として使用できます。

構文

```
NULLIF ( expression1, expression2 )
```

パラメータ

パラメータ	説明
expression1	比較される式。
expression2	比較される式。

戻り値
最初の引数のデータ型。

例
次の文は、a を返します。

```
SELECT NULLIF( 'a', 'b' ) FROM iq_dummy
```

次の文は、NULL を返します。

```
SELECT NULLIF( 'a', 'a' ) FROM iq_dummy
```

使用法
NULLIF は 2 つの式の値を比較します。

1 番目の式と 2 番目の式が等しい場合、**NULLIF** は NULL を返します。

1 番目の式と 2 番目の式が異なる場合、または 2 番目の式が NULL の場合、**NULLIF** は 1 番目の式を返します。

NULLIF 関数を使用すると、**CASE** 式を手短に書くことができます。**NULLIF** は次の式と同等です。

```
CASE WHEN  expression1 = expression2 THEN NULL
ELSE  expression1  END
```

標準と互換性

- SQL — ISO/ANSI SQL 文法の Transact-SQL 拡張。
- Sybase — Adaptive Server Enterprise ではサポートされていません。

参照：

- CASE 式 (31 ページ)
- 省略形 CASE 式の NULLIF 関数 (32 ページ)

NUMBER 関数 [その他]

1 で始まる数を、クエリの結果内にある連続したローのすべてに生成します。

構文

```
NUMBER ( * )
```

戻り値
INT

使用法
NUMBER 関数は、**UPDATE** 文の select リストまたは **SET** 句でのみ使用してください。たとえば、次の文は `seq_id` カラムの各ローに対して直前のローより 1 大きい値で更新します。番号は、**ORDER BY** 句で指定された順番に適用されます。

```
update empl
set seq_id = number(*)
order by empl_id
```

UPDATE 文で、**NUMBER**(*) 関数が **SET** 句に使用され、**FROM** 句に 1 対多のジョインが指定されている場合、**NUMBER**(*) は増加する一意な数値を生成しますが、ローが削除されるため、連続的には増加しません。

NUMBER によるプライマリ・キーの生成は、**SELECT** 文から **INSERT** を使用して行うことも可能です。しかし、連続したプライマリ・キーの生成には、**IDENTITY/AUTOINCREMENT** を使用することをおすすめします。

注意： **NUMBER** を **DELETE** 文、**WHERE** 句、**HAVING** 句、**ORDER BY** 句、サブクエリ、集合に関連するクエリ、制約、**GROUP BY** 句、**DISTINCT** 句、**UNION ALL** を含むクエリ、または派生テーブルで使用すると、構文エラーが発生します。

例
次の文は、番号の付いたリストを返します。

number(*)
1
2
3
4
5

```
SELECT NUMBER( * )
FROM Departments
WHERE DepartmentID > 10
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise ではサポートされていません。

OBJECT_ID 関数 [システム]

オブジェクト ID を返す。

構文

OBJECT_ID (*object-name*)

パラメータ

表 89 : パラメータ

パラメータ	説明
object-name	オブジェクト名。

例
次の文は、*Customers* テーブルのオブジェクト ID である 100209 を返します。

```
SELECT OBJECT_ID ('CUSTOMERS') FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Sybase IQ 用に実装された Adaptive Server Enterprise 関数。

参照：

- COL_NAME 関数 [システム] (152 ページ)
- DB_ID 関数 [システム] (188 ページ)
- DB_NAME 関数 [システム] (189 ページ)
- DB_PROPERTY 関数 [システム] (190 ページ)
- NEXT_DATABASE 関数 [システム] (256 ページ)
- OBJECT_NAME 関数 [システム] (263 ページ)
- INDEX_COL 関数 [システム] (221 ページ)

OBJECT_NAME 関数 [システム]

オブジェクト名を返します。

構文

```
OBJECT_NAME ( object-id [ , database-id ] )
```

パラメータ

表 90 : パラメータ

パラメータ	説明
object id	オブジェクト ID。
database-id	データベース ID。

例

次の文は、名前 "customer" を返します。

```
SELECT OBJECT_NAME ( 100209 ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Sybase IQ 用に実装された Adaptive Server Enterprise 関数。

参照：

- BIT_LENGTH 関数 [文字列] (141 ページ)

- BYTE_LENGTH 関数 [文字列] (142 ページ)
- CHAR_LENGTH 関数 [文字列] (148 ページ)
- COL_LENGTH 関数 [システム] (151 ページ)
- DATALENGTH 関数 [システム] (166 ページ)
- LEN 関数 [文字列] (235 ページ)
- LENGTH 関数 [文字列] (236 ページ)
- OCTET_LENGTH 関数 [文字列] (264 ページ)
- STR_REPLACE 関数 [文字列] (325 ページ)
- COL_NAME 関数 [システム] (152 ページ)
- DB_ID 関数 [システム] (188 ページ)
- DB_NAME 関数 [システム] (189 ページ)
- DB_PROPERTY 関数 [システム] (190 ページ)
- NEXT_DATABASE 関数 [システム] (256 ページ)
- OBJECT_ID 関数 [システム] (262 ページ)

OCTET_LENGTH 関数 [文字列]

カラムのバイト長を保持する、符号なしの 64 ビット値を返します。

構文

OCTET_LENGTH(*column-name*)

パラメータ

パラメータ	説明
column-name	カラムの名前。

使用法

引数が NULL の場合は、NULL 値を返します。

OCTET_LENGTH 関数は、すべての Sybase IQ データ型をサポートしています。

非構造化データ分析機能の使用ライセンスを取得している場合は、この関数でラージ・オブジェクト・データを使用できます。

詳細については、『Sybase IQ の非構造化データ分析の概要』の「関数のサポート」を参照してください。

標準と互換性

- Sybase — SQL Anywhere または Adaptive Server Enterprise によるサポートなし。

参照：

- BIT_LENGTH 関数 [文字列] (141 ページ)
- BYTE_LENGTH 関数 [文字列] (142 ページ)
- CHAR_LENGTH 関数 [文字列] (148 ページ)
- COL_LENGTH 関数 [システム] (151 ページ)
- DATALENGTH 関数 [システム] (166 ページ)
- LEN 関数 [文字列] (235 ページ)
- LENGTH 関数 [文字列] (236 ページ)
- OBJECT_NAME 関数 [システム] (263 ページ)
- STR_REPLACE 関数 [文字列] (325 ページ)

PATINDEX 関数 [文字列]

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
                %', string-expression )
```

パラメータ

パラメータ	説明
pattern	検索するパターンです。パターンは、ワイルドカードを使用して 126 バイトまでの文字列を指定してください。最初の % ワイルドカードを省略すると、**PATINDEX** は、パターンが文字列の最初に出現する場合に 1 を、そうでない場合は 0 を返します。*pattern* の先頭の文字が % の場合は、% を 2 つ続けると 1 つとして処理されます。 パターンにワイルドカード (パーセント % またはアンダースコア _) を使用しない場合は、255 バイトの長さまで指定できます。
string-expression	パターンが検索される文字列。

戻り値
INT

例
次の文は、値 2 を返します。

```
SELECT PATINDEX( '%hoco%', 'chocolate' ) FROM iq_dummy
```

次の文は、値 11 を返します。

```
SELECT PATINDEX ('%4_5_', '0a1A 2a3A 4a5A') FROM iq_dummy
```

使用法

PATINDEX は、パターンが最初に検出された先頭位置を返します。文字列パターンが検索対象の文字列に 2 つ以上含まれる場合、**PATINDEX** は最初の文字列の位置だけを返します。

パターンに使用するワイルドカードは、**LIKE** での比較の場合と同じです。次の表は、パターンのワイルドカードを示します。

表 91 : PATINDEX でのパターンのワイルドカード

ワイルドカード	一致条件
_ (アンダースコア)	任意の 1 文字
% (パーセント記号)	0 個以上の文字からなる任意の文字列
[]	指定した範囲または文字セット内の任意の 1 文字
[^]	指定した範囲または文字セット以外の任意の 1 文字

パターンを検出できなかった場合、**PATINDEX** はゼロ (0) を返します。

長さが 126 バイトを超えるパターンを検索すると、NULL が返されます。

長さが 0 の文字列を検索すると、1 が返されます。

引数のどれか 1 つでも NULL の場合、結果はゼロ (0) となります。

PATINDEX 関数で返されるか指定される位置またはオフセットはすべて、常に文字オフセットであり、マルチバイト・データの場合はバイト・オフセットとは異なることがあります。

PATINDEX は、CHAR カラムと VARCHAR カラムの場合、32 ビット符号なし整数で位置を返します。

非構造化データ分析機能の使用ライセンスを取得している場合は、この関数でラージ・オブジェクト・データを使用できます。

詳細については、『Sybase IQ の非構造化データ分析の概要』の「関数のサポート」を参照してください。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise 互換。

参照：
- LIKE 条件 (43 ページ)
- LOCATE 関数 [文字列] (239 ページ)

PERCENT_RANK 関数 [統計]

ORDER BY 句の定義に従い、クエリから返される 1 つのローの、クエリから返されるその他のローに対する (小数による) 位置を返します。

0 ～ 1 の 10 進値を返します。

構文

```
PERCENT_RANK ( ) OVER ( ORDER BY  expression [ ASC | DESC ] )
```

パラメータ

パラメータ	説明
expression	ソートを指定します。カラムの参照、集合関数、またはこれらの項目を起動する式など、有効な式を何でも指定できます。

戻り値

PERCENT_RANK 関数は、0 ～ 1 の DOUBLE 値を返します。

例

次の文は、**PERCENT_RANK** 関数の使い方を示しています。

```
SELECT s_suppkey, SUM(s_acctBal) AS sum_acctBal,
PERCENT_RANK() OVER ( ORDER BY SUM(s_acctBal) DESC )
AS percent_rank_all FROM supplier GROUP BY s_suppkey;

s_suppkey        sum_acctBal       percent_rank_all
supplier#011     200000            0
supplier#002     200000            0
supplier#013     123000            0.3333
supplier#004     110000            0.5
supplier#035     110000            0.5
supplier#006     50000             0.8333
supplier#021     10000             1
```

使用法

PERCENT_RANK はランク付け統計関数です。ロー R のパーセント・ランクは、**OVER** 句に指定された各グループ内でのローのランクから 1 を引いた値を、**OVER** 句に指定された各グループのローの合計数から 1 を引いた数で割って算出します。**PERCENT_RANK** は 0 ～ 1 の値を返します。最初のローのパーセント・ランクはゼロになります。

ローの **PERCENT_RANK** は、以下のように計算されます。

```
(Rx - 1) / (NtotalRow - 1)
```

*Rx*はグループのローのランク位置で、*NtotalRow*は **OVER** 句に指定されたグループでのローの合計数です。

PERCENT_RANK には **OVER (ORDER BY)** 句が必須です。**ORDER BY** 句は、ランク付けを実行するパラメータと、各グループでローをソートする順序を指定します。この **ORDER BY** 句は **OVER** 句内でのみ使用するもので、**SELECT** 文の **ORDER BY** とは異なります。ランク付けクエリ内の集合関数に **DISTINCT** を指定することはできません。

OVER 句は、関数がクエリの結果セットに対して処理を行うことを示します。結果セットは、**FROM**、**WHERE**、**GROUP BY**、**HAVING** の各句がすべて評価された後で返されるローです。**OVER** 句は、ランク付け統計関数の計算の対象となるローのデータ・セットを定義します。

ASC と DESC のパラメータでは、昇順または降順の順序付けシーケンスを指定します。昇順がデフォルトです。

PERCENT_RANK は、**SELECT** または **INSERT** 文の select リスト、および **SELECT** 文の **ORDER BY** 句でのみ使用できます。**PERCENT_RANK** は、ビューまたは union に含めることができます。**PERCENT_RANK** 関数は、サブクエリ、**HAVING** 句、**UPDATE** 文や **DELETE** 文の select リストでは使用できません。1 つのクエリで使用可能なランク付け統計関数は、1 つだけです。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise または SQL Anywhere によるサポートなし。

PERCENTILE_CONT 関数 [統計]

指定されたパーセンタイルから、対応する値を返します。連続分布データ・モデルを前提としています。

注意： パーセンタイルを計算するだけであれば、**NTILE** 関数に値 100 を指定して使用してください。

構文

```
PERCENTILE_CONT ( expression1 )
WITHIN GROUP ( ORDER BY expression2 [ ASC | DESC ] )
```

パラメータ

パラメータ	説明
expression1	数値データ型の定数を、0 以上 1 以下で指定します。引数が NULL であれば、"wrong argument for percentile" エラーが返されます。引数の値が 0 よりも小さいか、1 よりも大きい場合は、"data value out of range" エラーが返されます。
expression2	ソートを指定します。カラム参照を含む単一の式で指定してください。このソート式に、複数の式やランク付け統計関数、set 関数、またはサブクエリを指定することはできません。

例

次の例は **PERCENTILE_CONT** 関数を使用し、各地域の自動車販売の 10 番目のパーセンタイル値を調べています。

この例では、次のデータ・セットが使用されています。

```
sales       region        dealer_name
900         Northeast     Boston
800         Northeast     Worcester
800         Northeast     Providence
700         Northeast     Lowell
540         Northeast     Natick
500         Northeast     New Haven
450         Northeast     Hartford
800         Northwest     SF
600         Northwest     Seattle
500         Northwest     Portland
400         Northwest     Dublin
500         South         Houston
400         South         Austin
300         South         Dallas
200         South         Dover
```

次の **SELECT** 文には、**PERCENTILE_CONT** 関数が含まれています。

```
SELECT region, PERCENTILE_CONT(0.1)
WITHIN GROUP ( ORDER BY sales DESC )
FROM carSales GROUP BY region;
```

この **SELECT** 文の結果には、各地域の自動車販売の 10 番目のパーセンタイル値が一覧表示されます。

```
region           percentile_cont
Northeast        840
Northwest        740
South            470
```

使用法
逆分散統計関数は、K-理論パーセンタイル値を返します。これは、データ・セットのスレッショルド許容値を決定する際に使用します。**PERCENTILE_CONT** 関数は、パーセンタイル値を引数として受け取り、**WITHIN GROUP** 句で指定されたデータ・グループか、データ・セット全体に対して処理を実行します。グループごとに 1 つの値を返し、クエリの **GROUP BY** に指定したカラムが存在しなければ、結果は単一のローになります。結果のデータ型は、**WITHIN GROUP** 句に指定した **ORDER BY** の項目のデータ型と同じになります。**PERCENTILE_CONT** の **ORDER BY** 式のデータ型は、数値型である必要があります。

PERCENTILE_CONT には **WITHIN GROUP (ORDER BY)** 句を指定する必要があります。

必須の **ORDER BY** 句には、パーセンタイル関数の実行の対象となる式と、各グループ内でのローのソート順を指定します。**PERCENTILE_CONT** 関数の場合、この式のデータ型は数値型であることが必要です。この **ORDER BY** 句は **WITHIN GROUP** 句内でのみ使用するもので、**SELECT** 文の **ORDER BY** とは異なります。

WITHIN GROUP 句は、クエリ結果を順序付けられたデータ・セットに分類します。関数はこのデータ・セットに基づいて結果を計算します。**WITHIN GROUP** 句には、単一のソート項目を含めてください。**WITHIN GROUP** 句に指定されたソート項目が 1 つより多い (または少ない) 場合は、エラーが報告されます。

ASC と DESC のパラメータでは、昇順または降順の順序付けシーケンスを指定します。昇順がデフォルトです。

PERCENTILE_CONT 関数は、サブクエリ、**HAVING** 句、ビュー、union で使用できます。**PERCENTILE_CONT** は、統計を行わない単純な集合関数が使用されるところであれば、どこでも使用できます。**PERCENTILE_CONT** 関数では、データ・セット内の NULL 値は無視されます。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise または SQL Anywhere によるサポートなし。

参照：
- NTILE 関数 [統計] (258 ページ)
- PERCENTILE_DISC 関数 [統計] (271 ページ)

PERCENTILE_DISC 関数 [統計]

指定されたパーセンタイルから、対応する値を返します。離散分布データ・モデルを前提としています。

注意： パーセンタイルを計算するだけであれば、**NTILE** 関数に値 100 を指定して使用してください。

構文

```
PERCENTILE_DISC ( expression1 )
WITHIN GROUP ( ORDER BY expression2 [ ASC | DESC ] )
```

パラメータ

パラメータ	説明
expression1	数値データ型の定数を、0 以上 1 以下で指定します。引数が NULL であれば、"wrong argument for percentile" エラーが返されます。引数の値が 0 よりも小さいか、1 よりも大きい場合は、"data value out of range" エラーが返されます。
expression2	ソートを指定します。カラム参照を含む単一の式で指定してください。このソート式に、複数の式やランク付け統計関数、set 関数、またはサブクエリを指定することはできません。

例

次の例は **PERCENTILE_DISC** 関数を使用し、各地域の自動車販売の 10 番目のパーセンタイル値を調べています。

この例では、次のデータ・セットが使用されています。

```
sales       region        dealer_name
900          Northeast     Boston
800         Northeast     Worcester
800          Northeast     Providence
700          Northeast     Lowell
540          Northeast     Natick
500          Northeast     New Haven
450          Northeast     Hartford
800          Northwest     SF
600          Northwest     Seattle
500          Northwest     Portland
400          Northwest     Dublin
500          South         Houston
400          South         Austin
300          South         Dallas
200          South         Dover
```

次の **SELECT** 文には、**PERCENTILE_DISC** 関数が含まれています。

```
SELECT region, PERCENTILE_DISC(0.1)
WITHIN GROUP ( ORDER BY sales DESC )
FROM carSales GROUP BY region;
```

この **SELECT** 文の結果には、各地域の自動車販売の 10 番目のパーセンタイル値が一覧表示されます。

```
region           percentile_cont
Northeast        900
Northwest        800
South            500
```

使用法

逆分散統計関数は、K-理論パーセンタイル値を返します。これは、データ・セットのスレッショルド許容値を決定する際に使用します。 **PERCENTILE_DISC** 関数は、パーセンタイル値を引数として受け取り、**WITHIN GROUP** 句で指定されたデータ・グループか、データ・セット全体に対して処理を実行します。 この関数は、グループごとに 1 つの値を返します。 クエリの **GROUP BY** に指定したカラムが存在しなければ、結果は単一のローになります。 結果のデータ型は、**WITHIN GROUP** 句で指定された **ORDER BY** 項目のデータ型と同じです。**PERCENTILE_DISC** は、Sybase IQ でソート可能なすべてのデータ型をサポートします。

PERCENTILE_DISC には **WITHIN GROUP (ORDER BY)** 句を指定する必要があります。

必須の **ORDER BY** 句には、パーセンタイル関数の実行の対象となる式と、各グループ内でのローのソート順を指定します。この **ORDER BY** 句は **WITHIN GROUP** 句内でのみ使用するもので、**SELECT** 文の **ORDER BY** とは異なります。

WITHIN GROUP 句は、クエリ結果を順序付けられたデータ・セットに分類します。関数はこのデータ・セットに基づいて結果を計算します。**WITHIN GROUP** 句には、単一のソート項目を含めてください。**WITHIN GROUP** 句に指定されたソート項目が 1 つより多い (または少ない) 場合は、エラーが報告されます。

ASC と DESC のパラメータでは、昇順または降順の順序付けシーケンスを指定します。昇順がデフォルトです。

PERCENTILE_DISC 関数は、サブクエリ、**HAVING** 句、ビュー、union で使用できます。**PERCENTILE_DISC** は、分析を行わない単純な集合関数が使用されるところであれば、どこでも使用できます。**PERCENTILE_DISC** 関数では、データ・セット内の NULL 値は無視されます。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise または SQL Anywhere によるサポートなし。

参照：

- NTILE 関数 [統計] (258 ページ)
- PERCENTILE_CONT 関数 [統計] (268 ページ)

PI 関数 [数値]

円周率の値を数値で返します。

構文

```
PI ( * )
```

戻り値
DOUBLE

例
次の文は、値 3.141592653.... を返します。

```
SELECT PI( * ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — **PI()** 関数は Adaptive Server Enterprise でサポートされますが、**PI(*)** はサポートされません。

POWER 関数 [数値]

ある数値を、別の数値で累乗します。

構文

```
POWER ( numeric-expression1, numeric-expression2 )
```

パラメータ

パラメータ	説明
numeric-expression1	底。
numeric-expression2	指数。

戻り値
DOUBLE

例
次の文は、値 64 を返します。

```
SELECT Power( 2, 6 ) FROM iq_dummy
```

使用法
numeric-expression1 を *numeric-expresson2* で累乗します。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise 互換。

PROPERTY 関数 [システム]

指定されたサーバ・レベル・プロパティの値を文字列で返します。

構文

```
PROPERTY ( { property-id  |  property-name } )
```

注意： CIS 機能補正のパフォーマンスに関する考慮事項が適用されます。

パラメータ

表 92 :

パラメータ	説明
property-id	サーバ・レベル・プロパティのプロパティ番号である整数。この番号は、**PROPERTY_NUMBER** 関数で調べることができます。プロパティのセットを繰り返し処理する場合は、*property-id* がよく使用されます。
property-name	プロパティ名を指定する文字列。

戻り値
VARCHAR

例
次の文は、現在のデータベース・サーバの名前を返します。

```
SELECT PROPERTY( 'Name' ) FROM iq_dummy
```

使用法
各プロパティには名前と番号がありますが、番号はバージョンごとに変わる場合があります。よって、プロパティを指定する際、この番号を信頼できる識別子として使用しないでください。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。

- Sybase — Adaptive Server Enterprise ではサポートされていません。

参照：
- PROPERTY_NAME 関数 [システム] (276 ページ)
- PROPERTY_NUMBER 関数 [システム] (277 ページ)
- CONNECTION_PROPERTY 関数 [システム] (153 ページ)
- サーバで使用可能なプロパティ (129 ページ)
- 各データベースで使用可能なプロパティ (129 ページ)
- 接続プロパティ (128 ページ)

PROPERTY_DESCRIPTION 関数 [システム]

プロパティの説明を返します。

構文

```
PROPERTY_DESCRIPTION ( { property-id | property-name } )
```

注意： CIS 機能補正のパフォーマンスに関する考慮事項が適用されます。

パラメータ

パラメータ	説明
property-id	プロパティの番号である整数。この番号は、**PROPERTY_NUMBER** 関数で調べることができます。プロパティのセットを繰り返し処理する場合は、*property-id* がよく使用されます。
property-name	プロパティ名を指定する文字列。

戻り値
VARCHAR

例
次の文は、「インデックス挿入数」という説明を返します。

```
SELECT PROPERTY_DESCRIPTION( 'IndAdd' ) FROM iq_dummy
```

使用法
各プロパティには名前と番号がありますが、番号はリリースごとに変わる場合があります。よって、プロパティを指定する際、この番号を信頼できる識別子として使用しないでください。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise ではサポートされていません。

PROPERTY_NAME 関数 [システム]

指定されたプロパティ番号のプロパティ名を返します。

構文

```
PROPERTY_NAME ( property-id )
```

注意： CIS 機能補正のパフォーマンスに関する考慮事項が適用されます。

パラメータ

パラメータ	説明
property-id	プロパティのプロパティ番号。

戻り値

VARCHAR

例

次の文は、プロパティ番号 126 のプロパティを返します。この番号で参照される実際のプロパティは、バージョンの種類によって異なります。

```
SELECT PROPERTY_NAME( 126 ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise ではサポートされていません。

参照：

- PROPERTY 関数 [システム] (274 ページ)
- PROPERTY_NUMBER 関数 [システム] (277 ページ)
- CONNECTION_PROPERTY 関数 [システム] (153 ページ)
- サーバで使用可能なプロパティ (129 ページ)
- 各データベースで使用可能なプロパティ (129 ページ)
- 接続プロパティ (128 ページ)

PROPERTY_NUMBER 関数 [システム]

指定されたプロパティ名のプロパティ番号を返します。

構文

```
PROPERTY_NUMBER ( property-name )
```

注意： CIS 機能補正のパフォーマンスに関する考慮事項が適用されます。

パラメータ

パラメータ	説明
property-name	プロパティ名。

戻り値

INT

例

次の文は、整数の値を返します。実際の値は、バージョンの種類によって異なります。

```
SELECT PROPERTY_NUMBER( 'PAGESIZE' ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise ではサポートされていません。

参照：

- PROPERTY 関数 [システム] (274 ページ)
- PROPERTY_NAME 関数 [システム] (276 ページ)
- CONNECTION_PROPERTY 関数 [システム] (153 ページ)
- サーバで使用可能なプロパティ (129 ページ)
- 各データベースで使用可能なプロパティ (129 ページ)
- 接続プロパティ (128 ページ)

QUARTER 関数 [日付と時刻]

指定された日付式が、その年のどの四半期にあるかを示す数値を返します。

構文

```
QUARTER( date-expression )
```

パラメータ

パラメータ	説明
date-expression	日付。

戻り値
INT

例
DATE_ORDER オプションにデフォルト値の *ymd* が設定されている場合、次の文は、値 2 を返します。

```
SELECT QUARTER ( '1987/05/02' ) FROM iq_dummy
```

『リファレンス：文とオプション』の「データベース・オプション」>「アルファベット順のオプション・リスト」>「DATE_ORDER オプション」を参照してください。

使用法
次の表は、1 年の各四半期に含まれる日付を示します。

表 93 : 1 年の四半期の値

四半期	期間
1	1 月 1 日 ～ 3 月 31 日
2	4 月 1 日 ～ 6 月 30 日
3	7 月 1 日 ～ 9 月 30 日
4	10 月 1 日 ～ 12 月 31 日

標準と互換性

- SQL ― ISO/ANSI SQL 文法のベンダ拡張。
- Sybase ― Adaptive Server Enterprise ではサポートされていません。

RADIANS 関数 [数値]

数値を度からラジアンに変換します。

構文

```
RADIANS ( numeric-expression )
```

パラメータ

パラメータ	説明
numeric-expression	数値 (度)。この角度がラジアンに変換されます。

戻り値
DOUBLE

例
次の文は、約 0.5236 の値を返します。

```
SELECT RADIANS( 30 ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise ではサポートされていません。

RAND 関数 [数値]

0 <= x <1 範囲の乱数 x を DOUBLE 型の精度で返します。オプションでシードを使用できます。

構文

```
RAND ( [ integer-expression ] )
```

パラメータ

パラメータ	説明
integer-expression	乱数の生成に使用されるオプションのシード。この引数を指定すると、再生可能な乱数シーケンスを作成できます。

戻り値
DOUBLE

例
次の文は、テーブルの 5% のサンプリングを返します。

```
SELECT AVG(table1.number_of_cars), AVG(table1.number_of_tvs)FROM
table1 WHERE RAND(ROWID(table1)) < .05 and table1.income < 50000;
```

次の文は、約 941392926249216914 の値を返します。

```
SELECT RAND( 4 ) FROM iq_dummy
```

使用法
RAND が **FROM** 句および IQ ストアのテーブルのみを含むクエリの引数を使用して呼び出されると、任意の繰り返し可能な値が返されます。

引数が呼び出されない場合、**RAND** は非決定性関数になります。**RAND** を連続して呼び出すと、異なる値を返す場合があります。クエリ・オプティマイザは、**RAND** 関数の結果をキャッシュしません。

注意： **RAND** から返される値は、**FROM** 句を使用しているかどうか、および参照されているテーブルが SYSTEM または IQ ストアのどちらで作成されているかによって変わります。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise 互換。

RANK 関数 [統計]

グループ内の項目をランク付けします。

構文
RANK ( ) **OVER** ([**PARTITION BY**] **ORDER BY** *expression* [**ASC** | **DESC**])

パラメータ

パラメータ	説明
expression	ソートを指定します。カラムの参照、集合関数、またはこれらの項目を起動する式など、有効な式を何でも指定できます。

戻り値
INTEGER

例
RANK 関数は、次の文のように使用します。

```
SELECT Surname, Sex, Salary, RANK() OVER (PARTITION BY Sex
ORDER BY Salary DESC) AS RANK FROM Employees
WHERE State IN ('CA', 'AZ') AND DepartmentID IN (200, 300)
ORDER BY Sex, Salary DESC;
```

このクエリの結果セットを次に示します。

```
Surname           Sex      Salary     RANK
-------           ---      ------     ----
Savarino           F        72300.000  1
Jordan             F        51432.000  2
```

```
Clark               F        45000.000  3
Coleman             M        42300.000  1
Overbey             M        39300.000  2
```

使用法

RANK はランク付け統計関数です。ロー R のランクは、R 以前にあり R と同等でないローの数で決まります。**OVER** 句で指定されたグループどうしで、2 つ以上のローが同等な場合、または結果セット全体で同等な場合は、ランク付けの順番に 1 つ以上の隔たりが生じます。**RANK** と **DENSE_RANK** の相違点は、順位が同じである場合に、**DENSE_RANK** ではランク順に隔たりが置かれないことです。**RANK** では隔たりが置かれます。

RANK には **OVER (ORDER BY)** 句が必須です。**ORDER BY** 句は、ランク付けを実行するパラメータと、各グループでローをソートする順序を指定します。この **ORDER BY** 句は **OVER** 句内でのみ使用するもので、**SELECT** 文の **ORDER BY** とは異なります。ランク付けクエリ内の集合関数に DISTINCT を指定することはできません。

OVER (ORDER BY) 句内の **PARTITION BY** ウィンドウ・パーティション句はオプションです。

ASC と DESC のパラメータでは、昇順または降順の順序付けシーケンスを指定します。昇順がデフォルトです。

OVER 句は、関数がクエリの結果セットに対して処理を行うことを示します。結果セットは、**FROM**、**WHERE**、**GROUP BY**、**HAVING** の各句がすべて評価された後で返されるローです。**OVER** 句は、ランク付け統計関数の計算の対象となるローのデータ・セットを定義します。

RANK は、**SELECT** または **INSERT** 文の select リスト、および **SELECT** 文の **ORDER BY** 句でのみ使用できます。**RANK** は、ビューまたは union に含めることができます。**RANK** 関数は、サブクエリ、**HAVING** 句、**UPDATE** 文や **DELETE** 文の select リストでは使用できません。1 つのクエリで使用可能なランク付け統計関数は、1 つだけです。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise または SQL Anywhere によるサポートなし。

参照：

- DENSE_RANK 関数 [統計] (191 ページ)

REGR_AVGX 関数 [集合]

回帰線の独立変数の平均を計算します。

構文 1

```
REGR_AVGX ( dependent-expression, independent-expression )
```

構文 2

```
REGR_AVGX ( dependent-expression, independent-expression )
OVER ( window-spec )
```

パラメータ

パラメータ	説明
dependent-expression	独立した変数の影響を受ける変数。
independent-expression	結果に影響を与える変数。

戻り値
DOUBLE

使用法
この関数は、引数を DOUBLE に変換し、倍精度浮動小数点の計算を行って、DOUBLE を結果として返します。空のセットが指定された場合、**REGR_AVGX** は NULL を返します。

関数は、dependent-expression または independent-expression が NULL のペアをすべて排除した後、(dependent-expression と independent-expression) ペアのセットに適用されます。関数は、データを 1 回参照して同時に計算されます。NULL 値を排除した後、次の計算が行われます (x は independent-expression を示します)。

```
AVG (x)
```

詳細については、『SQL Anywhere サーバー - SQL の使用法』の「データのクエリと変更」>「OLAP のサポート」>「SQL Anywhere の Window 関数」>「ロー番号付け関数」>「集合関数に対応する数式」を参照してください。

注意： ROLLUP と CUBE は、構文 1 の **GROUP BY** 句ではサポートされていません。DISTINCT はサポートされていません。

構文 2 は、**SELECT** 文でウィンドウ関数として使用する場合の用法を示しています。その場合は、関数構文 (インライン) または **SELECT** 文の **WINDOW** 句で、*window-spec* の要素を指定できます。

例
次の例は、従属変数 (従業員の年齢) の平均を計算します。

```
SELECT REGR_AVGX( Salary, ( YEAR( NOW() ) - YEAR( BirthDate ) ) )FROM
Employees;
```

標準と互換性

- SQL — ISO/ANSI SQL 準拠。SQL/OLAP 機能 T612 です。
- Sybase — SQL Anywhere 互換。

参照：
- ウィンドウ集合関数の使用法 (111 ページ)

REGR_AVGY 関数 [集合]

回帰線の従変数の平均を計算します。

構文 1

```
REGR_AVGY(dependent-expression, independent-expression)
```

構文 2

```
REGR_AVGY(dependent-expression, independent-expression)
OVER (window-spec)
```

パラメータ

パラメータ	説明
dependent-expression	独立した変数の影響を受ける変数。
independent-expression	結果に影響を与える変数。

戻り値
DOUBLE

使用法
この関数は、引数を DOUBLE に変換し、倍精度浮動小数点の計算を行って、DOUBLE を結果として返します。空のセットが指定された場合、**REGR_AVGY** は NULL を返します。

関数は、dependent-expression または independent-expression が NULL のペアをすべて排除した後、(dependent-expression と independent-expression) ペアのセットに適用されます。関数は、データを 1 回参照して同時に計算されます。NULL 値を排除した後、次の計算が行われます (y は dependent-expression を示します)。

```
AVG(y)
```

詳細については、『SQL Anywhere サーバー - SQL の使用法』の「データのクエリと変更」>「OLAP のサポート」>「SQL Anywhere の Window 関数」>「ロー番号付け関数」>「集合関数に対応する数式」を参照してください。

注意： ROLLUP と CUBE は、構文 1 の **GROUP BY** 句ではサポートされていません。DISTINCT はサポートされていません。

構文 2 は、**SELECT** 文でウィンドウ関数として使用する場合の用法を示しています。その場合は、関数構文 (インライン) または **SELECT** 文の **WINDOW** 句で、*window-spec* の要素を指定できます。

例
次の例は、独立変数 (従業員の給与) の平均を計算します。この関数は、値 49988.6232 を返します。

```
SELECT REGR_AVGY( Salary, ( YEAR( NOW( )) - YEAR( BirthDate ) ) )FROM
Employees;
```

標準と互換性

- SQL — ISO/ANSI SQL 準拠。SQL/OLAP 機能 T612 です。
- Sybase — SQL Anywhere 互換。

REGR_COUNT 関数 [集合]

回帰線適合のために使用された非 NULL 値のペアの数を示す整数を返します。

構文 1

```
REGR_COUNT(dependent-expression, independent-expression)
```

構文 2

```
REGR_COUNT(dependent-expression, independent-expression)
OVER (window-spec)
```

パラメータ

パラメータ	説明
dependent-expression	独立した変数の影響を受ける変数。
independent-expression	結果に影響を与える変数。

戻り値
INTEGER

使用法
この関数は、結果として UNSIGNED BIGINT を返します。

注意： ROLLUP と CUBE は、構文 1 の **GROUP BY** 句ではサポートされていません。DISTINCT はサポートされていません。

構文 2 は、**SELECT** 文でウィンドウ関数として使用する場合の用法を示しています。その場合は、関数構文 (インライン) または **SELECT** 文の **WINDOW** 句で、*window-spec* の要素を指定できます。

例
次の例は、回帰直線の調整に使用される NULL 以外のペアの数を示す値を返します。この関数は、値 75 を返します。

```
SELECT REGR_COUNT( Salary, ( YEAR( NOW() ) -
YEAR( BirthDate ) ) )FROM Employees;
```

標準と互換性

- SQL — ISO/ANSI SQL 準拠。SQL/OLAP 機能 T612 です。
- Sybase － SQL Anywhere 互換。

参照：
- ウィンドウ集合関数の使用法 (111 ページ)

REGR_INTERCEPT 関数 [集合]

従変数と独立変数が最も適合する線形回帰線の y 切片を計算します。

構文 1

```
REGR_INTERCEPT(dependent-expression, independent-expression)
```

構文 2

```
REGR_INTERCEPT(dependent-expression, independent-expression)
OVER (window-spec)
```

パラメータ

パラメータ	説明
dependent-expression	独立した変数の影響を受ける変数。
independent-expression	結果に影響を与える変数。

戻り値
DOUBLE

使用法
この関数は、引数を DOUBLE に変換し、倍精度浮動小数点の計算を行って、DOUBLE を結果として返します。空のセットが指定された場合、**REGR_INTERCEPT** は NULL を返します。

関数は、dependent-expression または independent-expression が NULL のペアをすべて排除した後、(dependent-expression と *independent-expression*) ペアのセットに適用されます。関数は、データを 1 回参照して同時に計算されます。NULL 値を排除した後、次の計算が行われます (y は dependent-expression を表し、x は *independent-expression* を表します)。

```
AVG(y) - REGR_SLOPE(y, x) * AVG(x)
```

詳細については、『SQL Anywhere サーバー - SQL の使用法』の「データのクエリと変更」>「OLAP のサポート」>「SQL Anywhere の Window 関数」>「ロー番号付け関数」>「集合関数に対応する数式」を参照してください。

注意： ROLLUP と CUBE は、構文 1 の **GROUP BY** 句ではサポートされていません。DISTINCT はサポートされていません。

構文 2 は、**SELECT** 文でウィンドウ関数として使用する場合の用法を示しています。その場合は、関数構文 (インライン) または **SELECT** 文の **WINDOW** 句で、*window-spec* の要素を指定できます。

例
次の例は、値 1874.5805688517603 を返します。

```
SELECT REGR_INTERCEPT( Salary, ( YEAR( NOW() ) -
YEAR( BirthDate ) ) )FROM Employees;
```

標準と互換性

- SQL — ISO/ANSI SQL 準拠。SQL/OLAP 機能 T612 です。
- Sybase — SQL Anywhere 互換。

参照：
- ウィンドウ集合関数の使用法 (111 ページ)

REGR_R2 関数 [集合]

回帰線の決定係数 (R-squared または適合度統計とも呼ばれる) を計算します。

構文 1

```
REGR_R2(dependent-expression, independent-expression)
```

構文 2

```
REGR_R2(dependent-expression, independent-expression)
OVER (window-spec)
```

パラメータ

パラメータ	説明
dependent-expression	独立した変数の影響を受ける変数。
independent-expression	結果に影響を与える変数。

戻り値
DOUBLE

使用法
この関数は、引数を DOUBLE に変換し、倍精度浮動小数点の計算を行って、DOUBLE を結果として返します。空のセットが指定された場合、**REGR_R2** は NULL を返します。

REGR_R2 関数は、dependent-expression または *independent-expression* が NULL のペアをすべて排除した後、(dependent-expression と *independent-expression*) ペアのセットに適用されます。その後、Sybase IQ は次のアルゴリズムを適用します。

- VAR_POP(x) = 0 の場合、**REGR_R2** は VAR_POP(x) を計算して NULL を返します。VAR_POP(y) = 0 の場合、VAR_POP(y) を計算して値 1 を返します。
- VAR_POP(x) と VAR_POP(y) がいずれも 0 でない場合、次の値が返されます。

y は *dependent-expression* を表し、x は *independent-expression* を表します。

詳細については、『SQL Anywhere サーバー - SQL の使用法』の「データのクエリと変更」>「OLAP のサポート」>「SQL Anywhere の Window 関数」>「ロー番号付け関数」>「集合関数に対応する数式」を参照してください。

注意： ROLLUP と CUBE は、構文 1 の **GROUP BY** 句ではサポートされていません。**DISTINCT** はサポートされていません。

構文 2 は、**SELECT** 文でウィンドウ関数として使用する場合の用法を示しています。その場合は、関数構文 (インライン) または **SELECT** 文の **WINDOW** 句で、*window-spec* の要素を指定できます。

例
次の例は、値 0.19379959710325653 を返します。

```
SELECT REGR_R2( Salary, ( YEAR( NOW() ) - YEAR( BirthDate ) ) )FROM
Employees;
```

標準と互換性

- SQL — ISO/ANSI SQL 準拠。SQL/OLAP 機能 T612 です。
- Sybase — SQL Anywhere 互換。

参照：

- ウィンドウ集合関数の使用法 (111 ページ)

REGR_SLOPE 関数 [集合]

非 NULL 値のペアに適合された線形回帰線の傾斜を計算します。

構文 1

```
REGR_SLOPE(dependent-expression, independent-expression)
```

構文 2

```
REGR_SLOPE(dependent-expression, independent-expression)
OVER (window-spec)
```

パラメータ

パラメータ	説明
dependent-expression	独立した変数の影響を受ける変数。
independent-expression	結果に影響を与える変数。

戻り値
DOUBLE

使用法
この関数は、引数を DOUBLE に変換し、倍精度浮動小数点の計算を行って、DOUBLE を結果として返します。空のセットが指定された場合、**REGR_SLOPE** は NULL を返します。

REGR_SLOPE 関数は、dependent-expression または independent-expression が NULL のペアをすべて排除した後、(dependent-expression と independent-expression) ペアの

セットに適用されます。関数は、データを 1 回参照して同時に計算されます。NULL 値を排除した後、次の計算が行われます (y は dependent-expression を表し、x は independent-expression を表します)。

```
COVAR_POP(x, y) / VAR_POP(y)
```

詳細については、『SQL Anywhere サーバー - SQL の使用法』の「データのクエリと変更」>「OLAP のサポート」>「SQL Anywhere の Window 関数」>「ロー番号付け関数」>「集合関数に対応する数式」を参照してください。

注意： ROLLUP と CUBE は、構文 1 の **GROUP BY** 句ではサポートされていません。DISTINCT はサポートされていません。

構文 2 は、**SELECT** 文でウィンドウ関数として使用する場合の用法を示しています。その場合は、関数構文 (インライン) または **SELECT** 文の **WINDOW** 句で、*window-spec* の要素を指定できます。

例
次の例は、値 935.3429749445614 を返します。

```
SELECT REGR_SLOPE( Salary, ( YEAR( NOW() ) -
YEAR( BirthDate ) ) )FROM Employees;
```

標準と互換性

- SQL — ISO/ANSI SQL 準拠。SQL/OLAP 機能 T612 です。
- Sybase — SQL Anywhere 互換。

参照：
- ウィンドウ集合関数の使用法 (111 ページ)

REGR_SXX 関数 [集合]

非 NULL 値のペアに適合された線形回帰線の傾斜を計算します。

構文 1

```
REGR_SXX(dependent-expression, independent-expression)
```

構文 2

```
REGR_SXX(dependent-expression, independent-expression)
OVER (window-spec)
```

パラメータ

パラメータ	説明
dependent-expression	独立した変数の影響を受ける変数。
independent-expression	結果に影響を与える変数。

戻り値
DOUBLE

使用法
この関数は、引数を DOUBLE に変換し、倍精度浮動小数点の計算を行って、DOUBLE を結果として返します。空のセットが指定された場合、**REGR_SXX** は NULL を返します。

関数は、dependent-expression または *independent-expression* が NULL のペアをすべて排除した後、(dependent-expression と *independent-expression*) ペアのセットに適用されます。関数は、データを 1 回参照して同時に計算されます。NULL 値を排除した後、次の計算が行われます (y は dependent-expression を表し、x は *independent-expression* を表します)。

```
REGR_COUNT(y, x) * VAR_POP(x)
```

詳細については、『SQL Anywhere サーバー - SQL の使用法』の「データのクエリと変更」>「OLAP のサポート」>「SQL Anywhere の Window 関数」>「ロー番号付け関数」>「集合関数に対応する数式」を参照してください。

注意： ROLLUP と CUBE は、構文 1 の **GROUP BY** 句ではサポートされていません。DISTINCT はサポートされていません。

構文 2 は、**SELECT** 文でウィンドウ関数として使用する場合の用法を示しています。その場合は、関数構文 (インライン) または **SELECT** 文の **WINDOW** 句で、*window-spec* の要素を指定できます。

例
次の例は、値 5916.4800000000105 を返します。

```
SELECT REGR_SXX( Salary, ( YEAR( NOW() ) - YEAR( BirthDate ) ) )FROM
Employees;
```

標準と互換性

- SQL — ISO/ANSI SQL 準拠。SQL/OLAP 機能 T612 です。
- Sybase — SQL Anywhere 互換。

参照：
- ウィンドウ集合関数の使用法 (111 ページ)

REGR_SXY 関数 [集合]

従変数および独立変数の積和を返します。REGR_SXY 関数は、回帰モデルの統計的な有効性を評価するときに使用できます。

構文 1

```
REGR_SXY(dependent-expression, independent-expression)
```

構文 2

```
REGR_SXY(dependent-expression, independent-expression)
OVER (window-spec)
```

パラメータ

パラメータ	説明
dependent-expression	独立した変数の影響を受ける変数。
independent-expression	結果に影響を与える変数。

戻り値
DOUBLE

使用法
この関数は、引数を DOUBLE に変換し、倍精度浮動小数点の計算を行って、DOUBLE を結果として返します。空のセットが指定された場合、NULL が返されます。

関数は、dependent-expression または *independent-expression* が NULL のペアをすべて排除した後、(dependent-expression と *independent-expression*) ペアのセットに適用されます。関数は、データを 1 回参照して同時に計算されます。NULL 値を排除した後、次の計算が行われます (y は dependent-expression を表し、x は *independent-expression* を表します)。

```
REGR_COUNT(x, y) * COVAR_POP(x,
y)
```

詳細については、『SQL Anywhere サーバー - SQL の使用法』の「データのクエリと変更」>「OLAP のサポート」>「SQL Anywhere の Window 関数」>「ロー番号付け関数」>「集合関数に対応する数式」を参照してください。

注意： ROLLUP と CUBE は、構文 1 の **GROUP BY** 句ではサポートされていません。DISTINCT はサポートされていません。

構文 2 は、**SELECT** 文でウィンドウ関数として使用する場合の用法を示しています。その場合は、関数構文 (インライン) または **SELECT** 文の **WINDOW** 句で、*window-spec* の要素を指定できます。

例
次の例は、値 5533938.004400015 を返します。

```
SELECT REGR_SXY( Salary, ( YEAR( NOW() ) - YEAR( BirthDate ) ) )FROM
Employees;
```

標準と互換性

- SQL — ISO/ANSI SQL 準拠。SQL/OLAP 機能 T612 です。
- Sybase — SQL Anywhere 互換。

参照：
- ウィンドウ集合関数の使用法 (111 ページ)

REGR_SYY 関数 [集合]

回帰モデルの統計的な有効性を評価できる値を返します。

構文 1

```
REGR_SYY(dependent-expression, independent-expression)
```

構文 2

```
REGR_SYY(dependent-expression, independent-expression)
OVER (window-spec)
```

パラメータ

パラメータ	説明
dependent-expression	独立した変数の影響を受ける変数。
independent-expression	結果に影響を与える変数。

戻り値
DOUBLE

使用法
この関数は、引数を DOUBLE に変換し、倍精度浮動小数点の計算を行って、DOUBLE を結果として返します。空のセットが指定された場合、**REGR_SYY** は NULL を返します。

関数は、*dependent-expression* または independent-expression が NULL のペアをすべて排除した後、(*dependent-expression* と independent-expression) ペアのセットに適用されます。関数は、データを 1 回参照して同時に計算されます。NULL 値を排除した後、次の計算が行われます (y は *dependent-expression* を表し、x は independent-expression を表します)。

```
REGR_COUNT(x, y) * VAR_POP(y)
```

詳細については、『SQL Anywhere サーバー - SQL の使用法』の「データのクエリと変更」>「OLAP のサポート」>「SQL Anywhere の Window 関数」>「ロー番号付け関数」>「集合関数に対応する数式」を参照してください。

注意： ROLLUP と CUBE は、構文 1 の **GROUP BY** 句ではサポートされていません。DISTINCT はサポートされていません。

構文 2 は、**SELECT** 文でウィンドウ関数として使用する場合の用法を示しています。その場合は、関数構文 (インライン) または **SELECT** 文の **WINDOW** 句で、*window-spec* の要素を指定できます。

例
次の例は、値 26、708、672,843.3002 を返します。

```
SELECT REGR_SYY( Salary, ( YEAR( NOW() ) - YEAR( BirthDate ) ) )FROM
Employees;
```

標準と互換性

- SQL — ISO/ANSI SQL 準拠。SQL/OLAP 機能 T612 です。
- Sybase — SQL Anywhere 互換。

参照：
- ウィンドウ集合関数の使用法 (111 ページ)

REMAINDER 関数 [数値]

ある整数を別の整数で割った余りを返します。

構文

```
REMAINDER ( dividend, divisor )
```

パラメータ

パラメータ	説明
dividend	割られる数。つまり分数の分子。
divisor	割る数。つまり分数の分母。

戻り値
INTEGER

NUMERIC

例
次の文は、値 2 を返します。

```
SELECT REMAINDER( 5, 3 ) FROM iq_dummy
```

使用法
REMAINDER 関数は **MOD** 関数と同じです。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise ではサポートされていません。% (モジュロ) 演算子と除法演算子を使用して、余りを算出してください。

参照：
- MOD 関数 [数値] (249 ページ)

REPEAT 関数 [文字列]

文字列を指定された回数だけ連結します。

構文

```
REPEAT ( string-expression, integer-expression )
```

パラメータ

パラメータ	説明
string-expression	繰り返される文字列。
integer-expression	文字列を繰り返す回数。*integer-expression* が正数でない場合は、空の文字列を返します。

戻り値
LONG VARCHAR

LONG NVARCHAR

注意： 結果データ型は LONG VARCHAR です。**SELECT INTO** 文で **REPEAT** を使用する場合は、非構造化データ分析オプションのライセンスを所有しているか、**CAST** を使用して REPEAT を正しいデータ型とサイズに設定する必要があります。

例
次の文は、値 "repeatrepeatrepeat" を返します。

```
SELECT REPEAT( 'repeat', 3 ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise ではサポートされていませんが、**REPLICATE** で同じ機能を利用できます。

参照：
- REPLACE 関数 [文字列] (295 ページ)
- REPLICATE 関数 [文字列] (298 ページ)

REPLACE 関数 [文字列]

検出されたすべての部分文字列を、別の部分文字列に置換します。

構文

```
REPLACE ( original-string, search-string, replace-string )
```

パラメータ
いずれかの引数が NULL であれば、関数から NULL が返されます。

パラメータ	説明
original-string	検索される文字列。この文字列の長さに制限はありません。
search-string	検索して *replace-string* に置き換えられる文字列。255 バイトまでの文字列を指定してください。*search-string* が空の文字列の場合は、元の文字列がそのまま返されます。

パラメータ	説明
replace-string	置換文字列。*search-string* を置き換えます。この文字列の長さに制限はありません。*replace-string* が空の文字列の場合は、検索されたすべての *search-string* が削除されます。

戻り値
LONG VARCHAR

LONG NVARCHAR

注意： 結果データ型は LONG VARCHAR です。**SELECT INTO** 文で **REPLACE** を使用する場合は、非構造化データ分析オプションのライセンスを所有しているか、**CAST** を使用して **REPLACE** を正しいデータ型とサイズに設定する必要があります。

例
次の文は、値 "xx.def.xx.ghi" を返します。

```
SELECT REPLACE( 'abc.def.abc.ghi', 'abc', 'xx' ) FROM iq_dummy
```

次の文は、**ALTER PROCEDURE** 文を含む結果セットを生成します。この結果セットを実行すると、名前が変更されたテーブルを参照するストアド・プロシージャが修復されます (使用するためには、テーブル名を一意にする必要があります)。

```
SELECT REPLACE(
  replace(proc_defn,'OldTableName','NewTableName'),
  'create procedure',
  'alter procedure')
FROM SYS.SYSPROCEDURE
WHERE proc_defn LIKE '%OldTableName%'
```

次の例では、**LIST** 関数の区切り文字に、コンマでないものを使用します。

```
SELECT REPLACE( list( table_id ), ',', '--')
FROM  SYS.ISYSTAB
WHERE table_id <= 5
```

使用法
REPLACE 関数の結果データ型は、LONG VARCHAR です。**SELECT INTO** 文で **REPLACE** を使用する場合は、非構造化データ分析オプションのライセンスを所有しているか、**CAST** を使用して **REPLACE** を正しいデータ型とサイズに設定する必要があります。

この問題には、次の 2 つの対処方法があります。

- ローカル・テンポラリ・テーブルを宣言し、**INSERT** を実行します。

  ```
  DECLARE local temporary table #mytable
    (name_column char(10)) on commit preserve rows;
  ```

```
INSERT INTO #mytable SELECT REPLACE(name,'0','1')   FROM
dummy_table01;
```

- **CAST** を使用します。

```
SELECT CAST(replace(name, '0', '1') AS Char(10)) into #mytable
from dummy_table01;
```

replace-string が *search-string* よりも長く、置換後のカラムの長さをコントロールする必要がある場合は、**CAST** 関数を使用してください。次に例を示します。

```
CREATE TABLE aa(a CHAR(5));
INSERT INTO aa VALUES('CCCCC');
COMMIT;
SELECT a, CAST(REPLACE(a,'C','ZZ') AS CHAR(5)) FROM aa;
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise 互換。

参照：

- CAST 関数 [データ型変換] (144 ページ)
- CONVERT 関数 [データ型変換] (155 ページ)
- HOURS 関数 [日付と時刻] (214 ページ)
- MINUTES 関数 [日付と時刻] (248 ページ)
- MONTHS 関数 [日付と時刻] (251 ページ)
- SECOND 関数 [日付と時刻] (306 ページ)
- WEEKS 関数 [日付と時刻] (357 ページ)
- YEAR 関数 [日付と時刻] (362 ページ)
- YEARS 関数 [日付と時刻] (363 ページ)
- LCASE 関数 [文字列] (231 ページ)
- LEFT 関数 [文字列] (234 ページ)
- LOWER 関数 [文字列] (242 ページ)
- REVERSE 関数 [文字列] (299 ページ)
- RIGHT 関数 [文字列] (300 ページ)
- UCASE 関数 [文字列] (348 ページ)
- UPPER 関数 [文字列] (349 ページ)
- REPEAT 関数 [文字列] (294 ページ)
- REPLICATE 関数 [文字列] (298 ページ)

REPLICATE 関数 [文字列]

文字列を指定された回数だけ連結します。

構文

```
REPLICATE ( string-expression, integer-expression )
```

パラメータ

パラメータ	説明
string-expression	繰り返される文字列。
integer-expression	文字列を繰り返す回数。

戻り値

LONG VARCHAR

LONG NVARCHAR

注意： 結果データ型は LONG VARCHAR です。**SELECT INTO** 文で **REPLICATE** を使用する場合は、非構造化データ分析オプションのライセンスを所有しているか、**CAST** を使用して **REPLICATE** を正しいデータ型とサイズに設定する必要があります。

例

次の文は、値 "repeatrepeatrepeat" を返します。

```
SELECT REPLICATE( 'repeat', 3 ) FROM iq_dummy
```

使用法

REPLICATE は **REPEAT** 関数と同じです。

注意： **REPLICATE** 関数の結果データ型は、LONG VARCHAR です。**SELECT INTO** 文で **REPLICATE** を使用する場合は、非構造化データ分析オプションのライセンスを所有しているか、**CAST** を使用して **REPLICATE** を正しいデータ型とサイズに設定する必要があります。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise 互換。

参照：

- REPEAT 関数 [文字列] (294 ページ)
- REPLACE 関数 [文字列] (295 ページ)

REVERSE 関数 [文字列]

BINARY データ型または STRING データ型の入力として 1 つの引数を取り、指定され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

パラメータ	説明
expression	文字またはバイナリ型のカラム名、変数、または CHAR、VARCHAR、NCHAR、NVARCHAR、BINARY、VARBINARY 型の定数式。

戻り値
LONG VARCHAR

LONG NVARCHAR

注意： 結果データ型は LONG VARCHAR です。**SELECT INTO** 文で **REVERSE** を使用する場合は、非構造化データ分析オプションのライセンスを所有しているか、**CAST** を使用して **REVERSE** を正しいデータ型とサイズに設定する必要があります。

例 1

```
select reverse("abcd")
----
dcba
```

例 2

```
select reverse(0x12345000)
----------
0x00503412
```

使用法

- **REVERSE** は文字列関数で、式の文字順を逆にした文字列を返します。
- 式が 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。

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。

標準と互換性

- SQL — ISO/ANSI SQL 文法の Transact-SQL 拡張。

参照：
- 文字列演算子 (28 ページ)
- LCASE 関数 [文字列] (231 ページ)
- LEFT 関数 [文字列] (234 ページ)
- LOWER 関数 [文字列] (242 ページ)
- REPLACE 関数 [文字列] (295 ページ)
- RIGHT 関数 [文字列] (300 ページ)
- UCASE 関数 [文字列] (348 ページ)
- UPPER 関数 [文字列] (349 ページ)

RIGHT 関数 [文字列]

文字列の右端の文字を返します。

構文

```
RIGHT ( string-expression, numeric-expression )
```

パラメータ

パラメータ	説明
string-expression	左トランケートされる文字列。
numeric-expression	返す (文字列の末尾の) 文字数。

戻り値

LONG VARCHAR

LONG NVARCHAR

注意： 結果データ型は LONG VARCHAR です。**SELECT INTO** 文で **RIGHT** を使用する場合は、非構造化データ分析オプションのライセンスを所有しているか、**CAST** を使用して **RIGHT** を正しいデータ型とサイズに設定する必要があります。

例

次の文は、値 "olate" を返します。

```
SELECT RIGHT( 'chocolate', 5 ) FROM iq_dummy
```

使用法
文字列にマルチバイト文字が含まれ、照合が適切に使用されている場合は、返されるバイト数が、指定された文字数よりも多くなることがあります。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise 互換。

参照：
- LCASE 関数 [文字列] (231 ページ)
- LEFT 関数 [文字列] (234 ページ)
- LOWER 関数 [文字列] (242 ページ)
- REPLACE 関数 [文字列] (295 ページ)
- REVERSE 関数 [文字列] (299 ページ)
- UCASE 関数 [文字列] (348 ページ)
- UPPER 関数 [文字列] (349 ページ)

ROUND 関数 [数値]

numeric-expression を *integer-expression* で指定した小数点以下の桁数に丸めます。

構文
```
ROUND ( numeric-expression, integer-expression )
```

パラメータ

パラメータ	説明
numeric-expression	関数に渡され、丸められる数値。
integer-expression	正の整数は、丸めを行う小数点以下の有効桁数を指定します。負の式は、丸めを行う小数点の左側の有効桁数を指定します。

戻り値
NUMERIC

例
次の文は、値 123.200 を返します。
```
SELECT ROUND( 123.234, 1 ) FROM iq_dummy
```

このほかの、**ROUND** 関数の実行結果を次の表に示します。

値	ROUND (値)
123.4567	round (a.n,4)
123.4570	round (a.n,3)
123.4600	round (a.n,2)
123.5000	round (a.n,1)
123.0000	round (a.n, 0)
120.0000	round (a.n, -1)
100.0000	round (a.n, -2)
0.0000	round (a.n, -3)

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise 互換。

参照：

- TRUNCNUM 関数 [数値] (337 ページ)

ROW_NUMBER 関数 [統計]

ウィンドウ・パーティション内の各ローにユニークなロー番号を返すランク付け関数です。ウィンドウ・パーティションごとに新しい番号を割り当てます。

ウィンドウ・パーティションが存在しない場合は、結果セット内のローに 1 からテーブルのカーディナリティまでの番号が割り当てられます。

構文

```
ROW_NUMBER( ) OVER ( [PARTITION BY window partition] ORDER BY window ordering)
```

パラメータ

パラメータ	説明
ウィンドウ・パーティション	(オプション) 結果ロー・セットの分割方法を示す、カンマ区切りの 1 つ以上の値式。
ウィンドウ順序	ローのソートのための式を定義します。ウィンドウ・パーティションを指定した場合はウィンドウ・パーティション内でのソート、指定しなかった場合は結果セット内でのソートです。

使用法

ROW_NUMBER 関数は、**OVER** (**ORDER_BY**) ウィンドウ指定を必要とします。**OVER** (**ORDER_BY**) 句内のウィンドウ・パーティション句はオプションです。**OVER** (**ORDER_BY**) 句に、ウィンドウ・フレーム **ROWS**／**RANGE** 指定を含めることはできません。

例

次の例は、Employees テーブルから給与データを返し、部署 ID で結果セットをパーティションに分割し、入社日でデータを並べ替えます。**ROW_NUMBER** 関数は、各ローにロー番号を割り当てます。ウィンドウ・パーティションごとに新しいロー番号を割り当てます。

```
SELECT DepartmentID dID, StartDate, Salary,
ROW_NUMBER()OVER(PARTITION BY dID ORDER BY StartDate) FROM  Employees
ORDER BY 1,2;
```

次の結果セットが返されます。

```
dID         StartDate    Salary       Row_number()
=========  ===========  ==========  =============
100          1986-10-14  42,998.000     1
100          1987-07-23  39,875.500     2
100          1988-03-23  37,400.000     3
100          1989-04-20  42,500.000     4
100          1990-01-15  42,100.000     5
200          1985-02-03  38,500.000     1
200          1987-02-19  39,300.000     2
200          1988-11-22  39,800.000     3
200          1989-06-01  34,892.000     4
200          1990-05-13  33,890.000     5
200          1990-07-11  37,803.000     6
```

標準と互換性

- SQL — ISO/ANSI SQL 準拠。SQL/OLAP 機能 T611 です。

ROWID 関数 [その他]

テーブルの各ローに対して、内部ロー ID を返します。

構文

ROWID (*table-name*) …**FROM** *table-name*

パラメータ

パラメータ	説明
table-name	テーブル名。テーブル名は二重引用符を付け、カッコで括って指定します。二重引用符は省略できますが、引用符を使うことはできません。

戻り値
UNSIGNED BIGINT

例
次の文は、1 ～ 10 のロー ID の値を返します。

```
SELECT ROWID( “PRODUCTS” ) FROM PRODUCTS
```

rowid(Products)
1
2
3
.
.
.
10

次の文は、product ID の値が 400 よりも小さいすべてのローの product ID とロー ID の値を返します。

```
SELECT PRODUCTS.ID, ROWID ( PRODUCTS )
FROM PRODUCTS
WHERE PRODUCTS.ID < 400
```

ID	rowid(Products)
300	1
301	2
302	3

次の文は、ロー ID の値が 50 よりも大きいすべてのローを削除します。

```
DELETE FROM PRODUCTS
WHERE ROWID ( PRODUCTS ) > 50
```

使用法
ROWID 関数は、テーブルの特定のローを操作する場合に他の句と組み合わせて使用できます。

FROM*table-name* 句を指定する必要があります。

ROWID 関数には、テーブルのジョイン・インデックスを使用できないという制限事項があります。このため、ジョイン・インデックスを使用できる場合と比較すると、通常よりパフォーマンスが劣ります。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise ではサポートされていません。

RTRIM 関数 [文字列]

後続ブランクを取り除いた文字列を返します。

構文
```
RTRIM ( string-expression )
```

パラメータ

パラメータ	説明
string-expression	削除される文字列。

戻り値
VARCHAR

NVARCHAR

LONG VARCHAR

LONG NVARCHAR

注意： 結果データ型は LONG VARCHAR です。**SELECT INTO** 文で **RTRIM** を使用する場合は、非構造化データ分析オプションのライセンスを所有しているか、**CAST** を使用して **RTRIM** を正しいデータ型とサイズに設定する必要があります。

例
次の文は、後続ブランクがすべて削除された文字列 "Test Message" を返します。
```
SELECT RTRIM( 'Test Message     ' ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise 互換。

参照：
- LTRIM 関数 [文字列] (243 ページ)

SECOND 関数 [日付と時刻]

指定された日時の値の秒に対応する 0 から 59 までの数字を返します。

構文

```
SECOND ( datetime-expression )
```

パラメータ

パラメータ	説明
datetime-expression	日時の値。

戻り値
SMALLINT

例
次の文は、値 5 を返します。

```
SELECT SECOND( '1998-07-13 08:21:05' ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise 互換。

参照：
- CAST 関数 [データ型変換] (144 ページ)
- CONVERT 関数 [データ型変換] (155 ページ)
- HOURS 関数 [日付と時刻] (214 ページ)
- MINUTES 関数 [日付と時刻] (248 ページ)
- MONTHS 関数 [日付と時刻] (251 ページ)
- REPLACE 関数 [文字列] (295 ページ)
- WEEKS 関数 [日付と時刻] (357 ページ)
- YEAR 関数 [日付と時刻] (362 ページ)

- YEARS 関数 [日付と時刻] (363 ページ)

SECONDS 関数 [日付と時刻]

任意の開始日時から経過した秒数を返すか、2 つの時刻の間の秒数を返すか、または整数で指定された秒を時刻に追加します。

構文

```
SECONDS ( datetime-expression
| datetime-expression, datetime-expression
| datetime-expression, integer-expression )
```

パラメータ

パラメータ	説明
datetime-expression	日時。
integer-expression	datetime-expression に追加する秒数。integer-expression が負の場合、指定された秒数が日時の値から減算されます。整数式を指定する場合、datetime-expression は日時データ型として明示的にキャストする必要があります。

戻り値
INTEGER

TIMESTAMP

例
次の文は、値 3600 を返します。

```
SELECT ( SECONDS( '1998-07-13 06:07:12' ) -
SECONDS( '1998-07-13 05:07:12' )) FROM iq_dummy
```

次の文を実行すると、2 つの時刻の間の差である値 14400 が返されます。

```
SELECT SECONDS( '1999-07-13 06:07:12',
    '1999-07-13 10:07:12' ) FROM iq_dummy
```

次の文を実行すると、日時の値 1999-05-12 21:05:12.000 が返ります。

```
SELECT SECONDS( CAST( '1999-05-12 21:05:07'
AS TIMESTAMP ), 5) FROM iq_dummy
```

使用法
2 つ目の構文は、最初の引数の日時から 2 番目の引数の日時までが何秒以上かを返します。負の値が返ることもあります。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise ではサポートされていません。

SIGN 関数 [数値]

数値の符号を返します。

構文

```
SIGN ( numeric-expression )
```

パラメータ

パラメータ	説明
numeric-expression	符号を返す数値。

戻り値
SMALLINT

例
次の文は、値 -1 を返します。

```
SELECT SIGN( -550 ) FROM iq_dummy
```

使用法
負の数を指定すると、**SIGN** 関数は -1 を返します。

0 を指定すると、**SIGN** 関数は 0 を返します。

正の数を指定すると、**SIGN** 関数は 1 を返します。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise 互換。

SIMILAR 関数 [文字列]

2 つの文字列の類似性を表す 0 から 100 の間の整数を返します。

構文

```
SIMILAR ( string-expression1, string-expression2 )
```

パラメータ

パラメータ	説明
string-expression1	比較を行う最初の文字列。
string-expression2	比較を行う 2 つ目の文字列。

戻り値
SMALLINT

例
次の文は、値 75 を返します。

```
SELECT SIMILAR( 'toast', 'coast' ) FROM iq_dummy
```

これは、2 つの値の類似性が 75% であることを示します。

使用法
この関数は、2 つの文字列の類似性を表す 0 から 100 の間の整数を返します。結果は、2 つの文字列の間で文字が一致する割合と解釈できます。値 100 は、2 つの文字列が同じであることを意味します。

この関数は、名前 (顧客名など) のリストを訂正するときに使用します。顧客の中には、少しだけ違う名前で何回もリストに追加される人がいるかもしれません。テーブルを連結し、類似度が 90% 以上 100% 未満であるものすべてについて、レポートを作成してみてください。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise ではサポートされていません。

SIN 関数 [数値]

数値の正弦をラジアンで返します。

構文

```
SIN ( numeric-expression )
```

パラメータ

パラメータ	説明
numeric-expression	角度 (ラジアン)。

戻り値
DOUBLE

例
次の文は、値 0.496880 を返します。

```
SELECT SIN( 0.52 ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise 互換。

参照：
- COS 関数 [数値] (160 ページ)
- ATAN2 関数 [数値] (138 ページ)
- ATAN 関数 [数値] (137 ページ)
- ASIN 関数 [数値] (136 ページ)
- ACOS 関数 [数値] (133 ページ)
- COT 関数 [数値] (161 ページ)
- TAN 関数 [数値] (335 ページ)

SORTKEY 関数 [文字列]

代替の照合規則に基づいて、文字列をソートするために使用する値を生成します。

構文

```
SORTKEY ( string-expression
[, { collation-id
| collation-name [(collation-tailoring-string)] } ]
)
```

パラメータ

パラメータ	説明
string-expression	文字列式は、データベースの文字セットでエンコードされた文字を含む必要があり、STRING データ型である必要があります。 string-expression が NULL の場合、SORTKEY 関数は NULL 値を返します。空の文字列は、データベース・カラムの NULL 文字列とは異なるソート順の値を持ちます。 SORTKEY 関数が処理できる入力文字列の長さに制限はありません。SORTKEY の結果は、常に 1024 バイトに制限された VARBINARY データ型です。実際の結果が 1024 バイトを超えた場合、結果には最初の 1024 バイトだけが含まれます。
collation-name	使用するソート順の名前を指定する文字列または文字変数。エイリアス char_collation、または db_collation を指定して、データベースによって使用される CHAR 照合で使用されるソート・キーを生成することもできます。 同様に、エイリアス NCHAR_COLLATION を指定して、データベースによって使用される NCHAR 照合で使用されるソート・キーを生成できます。 ただし、Sybase IQ は、Sybase IQ 固有のオブジェクトに対して NCHAR_COLLATION をサポートしません。 NCHAR_COLLATION は、Sybase IQ サーバ上の SQL Anywhere オブジェクトに対してサポートされます。
collation-id	使用するソート順の ID 番号を指定する変数、整数定数、または文字列。 このパラメータは、対応する照合 ID によって参照される Adaptive Server Enterprise 照合にのみ適用されます。

パラメータ	説明
collation-tailoring-string	(オプション) ソートおよび文字の比較に追加の制御を行う照合調整オプション (*collation-tailoring-string*) を指定します。これらのオプションは、照合名の後にカッコで囲まれた「キーワード=値」ペアの形を取ります。次に例を示します。 `'UCA(locale=es;case=LowerFirst;accent=respect)'` これらのオプションを指定する構文は、**CREATE DATABASE** 文の **COLLATION** 句と同じです。 『リファレンス：文とオプション』の「SQL 文」>「CREATE DATABASE 文」を参照してください。 **注意：** Unicode Collation Algorithm (UCA) 照合を指定した場合、すべての照合調整オプションは SQL Anywhere データベースでサポートされます。 その他のすべての照合では、大文字と小文字を区別する調整だけがサポートされます。

戻り値
BINARY

例
次の文は、Employees テーブルをクエリし、すべての従業員の FirstName および Surname を Surname カラムのソート・キー値でソートして返します。ソートには、dict 照合 (Latin-1、英語、フランス語、ドイツ語辞書) が使用されます。

```
SELECT Surname, GivenName FROM Employees ORDER BY SORTKEY( Surname,
'dict' );
```

使用法
SORTKEY 関数が生成する値を使用し、事前定義済みのソート順の動作に基づいて、結果を順序付けることができます。データベース照合で使用できない文字ソート順の動作を使用できます。戻り値は、入力文字列に対して **SORTKEY** 関数が保持するソート順の情報がコーディングされたバイナリ値です。

たとえば、元の文字列を指定して、**SORTKEY** 関数から返される値をカラムに格納できます。次の **SELECT** 文は、テーブル T1 からタイ語辞書に従って c1 のソート順でデータを取得します。

```
SELECT rid, c1 from T1 ORDER BY SORTKEY(c1)
```

あるいは、元の文字列を指定して、**SORTKEY** から返される値をカラムに格納することもできます。必要な順序で文字データを取得するために、**SELECT** 文は、

SORTKEY 関数を実行した結果を含むカラムに 1 つの **ORDER BY** 句だけを含む必要があります。

```
UPDATE T1 SET shadowc1=SORTKEY(c1) FROM T1;
SELECT rid, c1 FROM T1 ORDER BY shadowc1
```

SORTKEY 関数により、指定されたソート順の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ことが保証されます。

クエリにソート・キーを生成すると負荷がかかる場合があります。頻繁に要求されるソート・キーの代替として、ソート・キー値を格納する計算カラムを作成して、クエリの **ORDER BY** 句でそのカラムを参照することを検討してください。

照合名または照合 ID を指定しない場合、デフォルトはデフォルト Unicode マルチ言語です。

有効な照合は次のとおりです。

- Sybase IQ でサポートされる照合をラベル順に表示するには、`iqinit -l` を実行します。
- 次の表に Adaptive Server Enterprise 照合を示します。

説明	照合名	照合 ID
デフォルト Unicode マルチ言語	default	0
CP 850 代替：アクセント記号なし	altnoacc	39
CP 850 代替：小文字優先	altdict	45
CP 850 西ヨーロッパ：大文字小文字の優先指定なし	altnocsp	46
CP 850 スカンジナビア語辞書	scandict	47
CP 850 スカンジナビア：大文字小文字の優先指定なし	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

説明	照合名	照合 ID
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 8859-5 トルコ語辞書	turdict	72
ISO 8859-5 トルコ語、アクセント記号なし	turnoac	73
ISO 8859-5 トルコ語、大文字小文字の区別なし	turnocs	74
CP 874 (TIS 620) タイ語辞書	thaidict	1
ISO 14651 標準の順序	14651	22
Shift-JIS バイナリ順	sjisbin	179
Unicode UTF-8 バイナリ・ソート	utf8bin	24
EUC JIS バイナリ順	eucjisbn	192
GB2312 バイナリ順	gb2312bn	137
CP932 MS バイナリ順	cp932bin	129
Big5 バイナリ順	big5bin	194
EUC KSC バイナリ順	euckscbn	161

照合調整の面から、ソート・キーを作成するとき、一般的に完全な大文字と小文字の区別は意図されたものです。したがって、UCA 以外の照合を指定する場合は、適用されるデフォルトの調整は case=Respect と同等です。たとえば、次の 2 つの文は同じです。

```
SELECT SORTKEY( 'abc', '1252LATIN1' ); SELECT SORTKEY( 'abc',
'1252LATIN1(case=Respect)' );
```

UCA 以外の照合を指定するとき、デフォルトでは、照合調整はアクセント記号付きで大文字と小文字が区別されます。しかし、UCA 以外の照合では、照合調整を使用して上書きできるのは大文字と小文字の区別だけです。次に例を示します。

```
SELECT SORTKEY( 'abc', '1252LATIN1(case=LowerFirst)' );
```

調整オプションを指定せずにデータベースを作成した場合、**SORTKEY** 関数にデータベース照合名を指定しても、次の 2 つの句では異なるソート順が生成されることがあります。

```
ORDER BY string-expression
```

```
ORDER BY SORTKEY( string-expression, database-collation-name )
```

データベース作成と **SORTKEY** 関数に使用されるデフォルトの調整設定が異なるので、異なるソート順が生成されることがあります。データベース照合と同じ動作を **SORTKEY** から得るには、データベース照合の設定に一致する調整構文を *collation-tailoring-string* に指定するか、照合名に `db_collation` を指定します。次に例を示します。

```
SORTKEY( expression, 'db_collation' )
```

注意： 15.0 より前のバージョンの Sybase IQ を使用して作成されたソート・キー値には、15.0 以降のバージョンを使用して作成された値と同じ値は含まれていません。 これは、15.0 以前のデータベースにソート・キーが格納されている場合、特にソート・キー値の比較がアプリケーションで必要な場合に問題になることがあります。 Sybase IQ の 15.0 より前のバージョンを使用して生成されたデータベース内のソート・キー値を再生成してください。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。

SOUNDEX 関数 [文字列]

文字列の音を表す数値を返します。

構文

```
SOUNDEX ( string-expression )
```

パラメータ

パラメータ	説明
string-expression	文字列。

戻り値
SMALLINT

例
次の文は、それぞれの名前の音を表す 2 つの数値を返します。引数に対する **SOUNDEX** の値は、どちらも 3827 です。

```
SELECT SOUNDEX( 'Smith' ), SOUNDEX( 'Smythe' ) FROM iq_dummy
```

SOUNDEX ('Smith') と **SOUNDEX** ('Smythe') は同じ値を返します。

使用法
文字列の **SOUNDEX** 関数値は、最初の英字と、それに続く H、Y、W 以外の 3 つの子音を基に生成されます。重なる英字は 1 字としてカウントします。次に例を示します。

```
SOUNDEX( 'apples' ) FROM iq_dummy
```

これは、A、P、L、S の英字に基づいて処理されます。

SOUNDEX 関数では、マルチバイト文字は無視されます。

完全ではありませんが、同じように発音し、同じ英字で始まる言葉に対して、**SOUNDEX** は通常同じ数値を返します。

標準と互換性

- SQL ― ISO/ANSI SQL 文法のベンダ拡張。
- Sybase ― Adaptive Server Enterprise 互換。ただし、Adaptive Server Enterprise は結果を CHAR(4) で返し、Sybase IQ は整数で返します。

参照：
- DIFFERENCE 関数 [文字列] (193 ページ)

SPACE 関数 [文字列]

指定された数のスペースを返します。

構文

```
SPACE ( integer-expression )
```

パラメータ

パラメータ	説明
integer-expression	返されるスペースの数。

戻り値
LONG VARCHAR

注意： 結果データ型は LONG VARCHAR です。**SELECT INTO** 文で **SPACE** を使用する場合は、非構造化データ分析オプションのライセンスを所有しているか、**CAST** を使用して **SPACE** を正しいデータ型とサイズに設定する必要があります。

例
次の文は、10 個のスペースを含む文字列を返します。

```
SELECT SPACE( 10 ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise 互換。

SQLFLAGGER 関数 [その他]

特定の SQL 文が、指定した標準に準拠しているかどうかを返します。

構文

```
SQLFLAGGER ( sql-standard-string, sql-statement-string )
```

パラメータ

パラメータ	説明
sql-standard-string	準拠をテストする規格レベル。指定できる値は、SQL_FLAGGER_ERROR_LEVEL データベース・オプションと同じです。 • SQL:2003/Core コア SQL/2003 構文に対する準拠性をテストします。 • SQL:2003/Package 上級 SQL/2003 構文に対する準拠性をテストします。 • SQL:1999/Core コア SQL/1999 構文に対する準拠性をテストします。 • SQL:1999/Package 上級 SQL/1999 構文に対する準拠性をテストします。 • SQL:1992/Entry 初級レベル SQL/1992 構文に対する準拠性をテストします。 • SQL:1992/Intermediate 中級レベル SQL/1992 構文に対する準拠性をテストします。 • SQL:1992/Full 上級 SQL/1992 構文に対する準拠性をテストします。

パラメータ	説明
sql-statement-string	準拠性をチェックする SQL 文。

戻り値
LONG VARCHAR

注意： 結果データ型は LONG VARCHAR です。**SELECT INTO** 文で **SQLFLAGGER** を使用する場合は、非構造化データ分析オプションのライセンスを所有しているか、**CAST** を使用して **SQLFLAGGER** を正しいデータ型とサイズに設定する必要があります。

例
次の文は、使用できない拡張機能が検出されたときに返されるメッセージの例を示します。

```
SELECT SQLFLAGGER( 'SQL:2003/Package', 'SELECT top 1 dummy_col FROM sys.dummy ORDER BY dummy_col' );
```

この文は、メッセージ '0AW03 Disallowed language extension detected in syntax near 'top' on line 1' を返します。

次の文は、使用できない拡張性を含まないので、'00000' を返します。

```
SELECT SQLFLAGGER( 'SQL:2003/Package', 'SELECT dummy_col FROM sys.dummy' );
```

使用法
また、SQL プリプロセッサ・ユーティリティ iqsqlpp を使用して、特定の SQL92 セットの一部ではない Embedded SQL にフラグを設定することもできます。『ユーティリティ・ガイド』の SQL プリプロセッサ・ユーティリティ iqsqlpp に関する説明を参照してください。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise 互換。

SQRT 関数 [数値]

数値の平方根を返します。

構文

```
SQRT ( numeric-expression )
```

パラメータ

パラメータ	説明
numeric-expression	平方根が計算される数値。

戻り値
DOUBLE

例
次の文は、値 3 を返します。

```
SELECT SQRT( 9 ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise 互換。

SQUARE 関数 [数値]

指定した式の平方を、float で表したものを返します。

構文

```
SQUARE ( numeric-expression )
```

パラメータ

パラメータ	説明
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。他のデータ型を指定すると、SQUARE 関数ではエラーが返ります。戻り値は、DOUBLE データ型です。

使用法
SQUARE 関数は 1 つの引数を取ります。たとえば **SQUARE** (*12.01*) は、144.240100 を返します。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise 互換。

STDDEV 関数 [集合]

数値セットの標準偏差を返します。

構文

```
STDDEV ( [ ALL ] expression )
```

パラメータ

パラメータ	説明
expression	任意の数値式 (FLOAT、REAL、または DOUBLE 精度)。

戻り値

DOUBLE

例

次のようなデータがあるとします。

```
SELECT Salary FROM Employees WHERE DepartmentID = 300
```

Salary
51432.000
57090.000
42300.000
43700.00
36500.000
138948.000
31200.000
58930.00
75400.00

次の文は、値 32617.8446712838471 を返します。

```
SELECT STDDEV ( Salary ) FROM Employees
WHERE DepartmentID = 300
```

次のようなデータがあるとします。

```
SELECT UnitPrice FROM Products WHERE Name = 'Tee Shirt'
```

名前	UnitPrice
Tee Shirt	9.00
Tee Shirt	14.00
Tee Shirt	14.00

次の文は、値 2.88675134594813049 を返します。

```
SELECT STDDEV ( UnitPrice ) FROM Products
WHERE Name = 'Tee Shirt'
```

使用法

STDDEV では、次の計算式が使用されます。

$$stddev = \sqrt{variance}$$

STDDEV は、結果を DOUBLE 精度浮動小数点数のデータ型で返します。空のセットに適用すると、結果は NULL になり、1 要素の入力セットに NULL が返されます。

STDDEV は、キーワード DISTINCT をサポートしません。**STDDEV** で DISTINCT を使用すると、構文エラーが返されます。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise ではサポートされていません。

参照：

- ウィンドウ集合関数の使用法 (111 ページ)
- STDDEV_SAMP 関数 [集合] (323 ページ)
- VARIANCE 関数 [集合] (355 ページ)

STDDEV_POP 関数 [集合]

数値式からなる母標準偏差を DOUBLE として計算します。

構文

```
STDDEV_POP ( [ ALL ] expression )
```

パラメータ

パラメータ	説明
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 (通常はカラム名)。

戻り値
DOUBLE

例
次の文は、異なる期間における注文ごとの項目数で平均と平方偏差をリストします。

```
SELECT year( ship_date )AS Year, quarter( ship_date )
  AS Quarter, AVG( quantity ) AS Average,
  STDDEV_POP ( quantity ) AS Variance
FROM SalesOrderItems GROUP BY Year, Quarter
  ORDER BY Year, Quarter;
```

Year	Quarter	Average	Variance
2000	1	25.775148	14.2794
2000	2	27.050847	15.0270
...	...	...	...

使用法
グループまたはパーティションの各ロー (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 *value expression* 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。

$$\sqrt{\frac{\sum (x_i - \bar{x})^2}{n}}$$

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise ではサポートされていません。

参照：
- ウィンドウ集合関数の使用法 (111 ページ)

STDDEV_SAMP 関数 [集合]

1 つの数値式で構成される標本標準偏差を DOUBLE として計算します。

構文

```
STDDEV_SAMP ( [ ALL ] expression )
```

パラメータ

パラメータ	説明
expression	その標本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 (通常はカラム名)。

戻り値
DOUBLE

例
次の文は、異なる期間における注文ごとの項目数で平均と平方偏差をリストします。

```
SELECT year( ship_date ) AS Year, quarter( ship_date )
  AS Quarter, AVG( quantity ) AS Average,
  STDDEV_SAMP( quantity ) AS Variance
FROM SalesOrderItems GROUP BY Year, Quarter
  ORDER BY Year, Quarter;
```

Year	Quarter	Average	Variance
2000	1	25.775148	14.3218
2000	2	27.050847	15.0696
...	...	...	...

使用法

注意： STDDEV_SAMP は、**STDDEV** のエイリアスです。

グループまたはパーティションの各ロー (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 *value expression* 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。

1 要素の入力セットの場合、NULL によって NULL が返されます。

標準偏差は次の式に従って計算されます。これは正規分布とみなされます。

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

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise ではサポートされていません。

参照：

- ウィンドウ集合関数の使用法 (111 ページ)
- STDDEV 関数 [集合] (320 ページ)
- VARIANCE 関数 [集合] (355 ページ)

STR 関数 [文字列]

数値に対応する文字列を返します。

構文

```
STR ( numeric-expression [ , length [ , decimal ] ] )
```

パラメータ

パラメータ	説明
numeric-expression	任意の概数 (FLOAT、REAL、または DOUBLE 精度) 式。
length	返される文字 (小数点、小数点の両側にあるすべての桁、必要な符号、ブランクを含む) の数。デフォルトは 10 で、最大長は 255 です。
decimal	返される文字の小数点以下の桁数。デフォルトは 0 です。

戻り値
VARCHAR

例
次の文は、6 つのスペースの後に 1234 が続く、合計で 10 の文字からなる文字列を返します。

```
SELECT STR( 1234.56 ) FROM iq_dummy
```

次の文は、1234.5 を返します。

```
SELECT STR( 1234.56, 6, 1 ) FROM iq_dummy
```

使用法
数値の整数部分が指定された長さを超える場合は、NULL が返されます。たとえば、次の文は NULL を返します。

```
SELECT STR( 1234.56, 3 ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise 互換。

STR_REPLACE 関数 [文字列]

BINARY 型または STRING 型の入力として 3 つの引数を取り、最初の文字列式 (*string_expr1*) 内に出現する 2 番目の文字列式 (*string_expr2*) のすべてのインスタンスを、3 番目の式 (*string_expr3*) で置き換えます。

構文

```
REPLACE ( string_expr1, string_expr2, string_expr3 )
```

パラメータ

表 94 : パラメータ

パラメータ	説明
string_expr1	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_expr2	最初の式 (*string_expr1*) 内で検索するパターン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_expr3	置換文字列式です。CHAR、VARCHAR、UNICHAR、UNIVARCHAR、VARBINARY、または BINARY データ型として表されます。

例 1
文字列 *cdefghi* 内の文字列 *def* を *yyy* に置き換えます。

```
select 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)
```

例 3
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_expr2* が NULL に置き換えられます。この方法で STR_REPLACE を使用して、文字列を削除するオペレーションを行うことができます。“abcghijklm” を返します。

```
select str_replace("abcdefghijklm", "def", NULL)
----------
abcghijklm
(1 row affected)
```

使用法
STR_REPLACE は、**REPLACE** 関数のエイリアスです。

- 入力として任意のデータ型を取り、STRING または BINARY を返します。たとえば、引数として渡された空の文字列 ("") は、さらに評価される前に 1 つのスペース (" ") に置き換えられます。これは、`BINARY` 型でも `STRING` 型でも同じです。
- すべての引数に、`BINARY` データ型と `STRING` データ型の組み合わせを指定できます。
- 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。すべての引数が定数に割り当てられたカラムまたはホスト数である場合、結果の長さは、Sybase IQ 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
  ```

- 式のコンパイル時に引数値がわからないため Sybase IQ が結果の長さを計算できない場合は、結果の長さは 255 になります。
- **RESULT_LEN** が 32767 を超えることはありません。

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。

標準と互換性

- SQL — ISO/ANSI SQL 文法の Transact-SQL 拡張。

参照：

- BIT_LENGTH 関数 [文字列] (141 ページ)
- BYTE_LENGTH 関数 [文字列] (142 ページ)
- CHAR_LENGTH 関数 [文字列] (148 ページ)
- COL_LENGTH 関数 [システム] (151 ページ)
- DATALENGTH 関数 [システム] (166 ページ)
- LEN 関数 [文字列] (235 ページ)
- LENGTH 関数 [文字列] (236 ページ)
- OBJECT_NAME 関数 [システム] (263 ページ)
- OCTET_LENGTH 関数 [文字列] (264 ページ)

STRING 関数 [文字列]

1 つ以上の文字列を連結して、1 つの大きな文字列にします。

構文

```
STRING ( string-expression [ , … ] )
```

パラメータ

パラメータ	説明
string-expression	文字列。引数が 1 つしか指定されなければ、単一の式に変換されます。引数が複数指定されると、それらが 1 つの文字列に連結されます。NULL は空の文字列 ('') として扱われます。

戻り値

LONG VARCHAR

LONG NVARCHAR

LONG BINARY

注意： 結果データ型は LONG VARCHAR です。**SELECT INTO** 文で **STRING** を使用する場合は、非構造化データ分析オプションのライセンスを所有しているか、**CAST** を使用して **STRING** を正しいデータ型とサイズに設定する必要があります。

例

次の文を実行すると、値 testing123 が返ります。

```
SELECT STRING( 'testing', NULL, 123 )
FROM iq_dummy
```

使用法
数値または日付のパラメータは、文字列に変換されてから連結されます。**STRING** 関数を使用して、1 つの式を文字列に変換することもできます。それには、変換する式を唯一のパラメータとして指定します。

すべてのパラメータが NULL の場合、**STRING** は NULL を返します。

標準と互換性

- SQL — ISO/ANSI SQL 文法の Transact-SQL 拡張。
- Sybase — Adaptive Server Enterprise ではサポートされていません。

STRTOUUID 関数 [文字列]

文字列の値を、一意な識別子 (UUID または GUID) の値に変換します。

構文
STRTOUUID (*string-expression*)

パラメータ

パラメータ	説明
string-expression	*xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx* 形式の文字列。

戻り値
UNIQUEIDENTIFIER

例

```
CREATE TABLE T (
 pk uniqueidentifier primary key,
 c1 int);
INSERT INTO T (pk, c1)
 VALUES (STRTOUUID
 ('12345678-1234-5678-9012-123456789012'), 1);
```

使用法
xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx の形式の文字列 (*x* は 16 進の桁) を、一意な識別子の値に変換します。文字列が UUID として有効でなければ、NULL が返されます。

STRTOUUID を使用して、UUID 値を Adaptive Server EnterpriseSybase IQ データベースに挿入できます。

標準と互換性

- SQL — ISO/ANSI SQL 文法の Transact-SQL 拡張。
- Sybase — Adaptive Server Enterprise によるサポートなし。

参照：
- バイナリ・データ型 (84 ページ)
- NEWID 関数 [その他] (253 ページ)
- UUIDTOSTR 関数 [文字列] (351 ページ)
- 文字データ型 (75 ページ)
- バイナリ・データ型 (710 ページ)

STUFF 関数 [文字列]

ある文字列からいくつかの文字を削除し、そこへ別の文字列を置きます。

構文

```
STUFF ( string-expression1, start, length, string-expression2 )
```

パラメータ

パラメータ	説明
string-expression1	**STUFF** 関数によって変更される文字列。
start	文字の削除を開始する位置です。文字列の先頭文字の位置が 1 になります。
length	削除される文字数です。
string-expression2	挿入する文字列。**STUFF** 関数を使用して文字列の一部を削除するには、NULL の置換文字列を使用します。

戻り値
LONG NVARCHAR

例
次の文を実行すると、値 "chocolate pie" が返ります。

```
SELECT STUFF( 'chocolate cake', 11, 4, 'pie' )
FROM iq_dummy
```

使用法
STUFF を使って文字列の一部を削除するには、置換後の文字列に NULL を指定します。**STUFF** を使って文字列を挿入するには、length をゼロに指定します。

STUFF 関数は、次のような場合は NULL を返します。

- 最初の 3 つのパラメータのいずれかが NULL 値である。
- start パラメータまたは length パラメータのいずれかが負の値である。
- start パラメータが string-expression1 の長さよりも大きい。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise 互換。

SUBSTRING 関数 [文字列]

文字列の一部を返します。

構文
```
{ SUBSTRING | SUBSTR } ( string-expression, start [ , length ] )
```

パラメータ

パラメータ	説明
string-expression	部分文字列が返される文字列。
start	返される部分文字列の開始位置。負の開始位置を指定する場合は、文字列の最初からの文字数ではなく、文字列の最後からの文字数を指定します。文字列の先頭文字の位置が 1 になります。
length	返される部分文字列の長さ。正の *length* を指定すると、開始位置から右側へ *length* 文字の位置で、部分文字列が終了します。一方、負の *length* を指定すると、開始位置から左側へ *length* 文字の位置で終了します。

戻り値
LONG VARCHAR

LONG NVARCHAR

LONG BINARY

注意： 結果データ型は LONG VARCHAR です。**SELECT INTO** 文で **STRING** を使用する場合は、非構造化データ分析オプションのライセンスを所有しているか、**CAST** を使用して **STRING** を正しいデータ型とサイズに設定する必要があります。

例
次の文は、"back" を返します。

```
SELECT SUBSTRING ( 'back yard', 1 , 4 )
FROM iq_dummy
```

次の文は、yard を返します。

```
SELECT SUBSTR ( 'back yard', -1 , -4 )
FROM iq_dummy
```

次の文は、0x2233 を返します。

```
SELECT SUBSTR ( 0x112233445566, 2, 2 )
FROM iq_dummy
```

使用法
length を指定すると、指定した長さの部分文字列に制限されます。長さの指定を省略すると、*start* の位置を開始点として、文字列の残りがすべて返されます。

start と *length* の両方に負の値を指定できます。正と負を適切に組み合わせて使うと、文字列の最初または最後から部分文字列を取得できます。

非構造化データ分析機能の使用ライセンスを取得している場合は、この関数でラージ・オブジェクト・データを使用できます。

詳細については、『Sybase IQ の非構造化データ分析の概要』の「関数のサポート」を参照してください。

標準と互換性

- SQL － ISO/ANSI SQL 文法のベンダ拡張。
- Sybase － **SUBSTR** は、Adaptive Server Enterprise ではサポートされていません。代わりに **SUBSTRING** を使用してください。

参照：
- CHARINDEX 関数 [文字列] (149 ページ)

SUBSTRING64 関数 [文字列]

SUBSTRING64 関数は、ラージ・オブジェクト・カラムまたは変数パラメータの可変長文字列を返します。

使用法

SUBSTRING64 は、LONG VARCHAR カラムと LONG BINARY カラム、および任意のデータ・サイズの LONG VARCHAR 変数と LONG BINARY 変数の検索をサポートします。現在、SQL 変数で保持できる最大の長さは 2GB - 1 です。

非構造化データ分析機能の使用ライセンスを取得している場合は、この関数でラージ・オブジェクト・データを使用できます。

詳細については、『Sybase IQ の非構造化データ分析の概要』の「関数のサポート」を参照してください。

SUM 関数 [集合]

ローの各グループに対して、指定された式の合計を返します。

構文

```
SUM ( expression | DISTINCT  column-name )
```

パラメータ

パラメータ	説明
expression	合計する対象。通常はカラムを指定します。
DISTINCT column-name	ローの各グループに対する *column-name* のうち、一意な値の合計を計算します。この方法で使用することはほとんどありませんが、万全を期すために含まれています。

戻り値

INTEGER

DOUBLE

NUMERIC

例

次の文は、値 3749146.740 を返します。

```
SELECT SUM( salary )
FROM Employees
```

使用法
指定された式が NULL になるローは含まれません。

ローがまったくないグループに対しては、NULL 値を返します。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise 互換。

参照：
- AVG 関数 [集合] (139 ページ)
- COUNT 関数 [集合] (164 ページ)
- ウィンドウ集合関数の使用法 (111 ページ)

SUSER_ID 関数 [システム]

ユーザ ID 番号を整数で返します。

構文
```
SUSER_ID ( [ user-name ] )
```

パラメータ

パラメータ	説明
user-name	ユーザ名。

戻り値
INT

例
次の文は、ユーザ ID 番号 1 を返します。
```
SELECT SUSER_ID ('DBA') FROM iq_dummy
```
次の文は、ユーザ ID 番号 0 を返します。
```
SELECT SUSER_ID ('SYS') FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Sybase IQ 用に実装された Adaptive Server Enterprise 関数。

参照：
- SUSER_NAME 関数 [システム] (334 ページ)
- USER_ID 関数 [システム] (350 ページ)
- USER_NAME 関数 [システム] (350 ページ)

SUSER_NAME 関数 [システム]

ユーザ名を返します。

構文
```
SUSER_NAME ( [ user-id ] )
```

パラメータ

パラメータ	説明
user-id	ユーザ ID 番号。

戻り値
LONG VARCHAR

注意： 結果データ型は LONG VARCHAR です。**SELECT INTO** 文で **SUSER_NAME** を使用する場合は、非構造化データ分析オプションのライセンスを所有しているか、**CAST** を使用して **SUSER_NAME** を正しいデータ型とサイズに設定する必要があります。

例
次の文は、値 DBA を返します。
```
SELECT SUSER_NAME ( 1 ) FROM iq_dummy
```
次の文は、値 SYS を返します。
```
SELECT SUSER_NAME ( 0 ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Sybase IQ 用に実装された Sybase IQAdaptive Server Enterprise 関数。Adaptive Server Enterprise の場合、**SUSER_NAME** はサーバ・ユーザ名を返す。

参照：
- SUSER_ID 関数 [システム] (333 ページ)
- USER_ID 関数 [システム] (350 ページ)
- USER_NAME 関数 [システム] (350 ページ)

TAN 関数 [数値]

数値の正接を返します。

構文

```
TAN ( numeric-expression )
```

パラメータ

パラメータ	説明
numeric-expression	角度 (ラジアン)。

戻り値
DOUBLE

例
値として 0.572561 を返す。

```
SELECT TAN( 0.52 ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise 互換。

参照：

- COS 関数 [数値] (160 ページ)
- ATAN2 関数 [数値] (138 ページ)
- ATAN 関数 [数値] (137 ページ)
- ASIN 関数 [数値] (136 ページ)
- ACOS 関数 [数値] (133 ページ)
- COT 関数 [数値] (161 ページ)
- SIN 関数 [数値] (309 ページ)

TODAY 関数 [日付と時刻]

現在の日付を返します。これは **CURRENT DATE** に対応する古い構文です。

構文

```
TODAY ( * )
```

戻り値
DATE

例
次の文は、システム・クロックによる現在の日付を返します。

```
SELECT TODAY( * ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise ではサポートされていません。

TRIM 関数 [文字列]

文字列から先行ブランクと後続ブランクを削除します。

構文

```
TRIM ( string-expression )
```

パラメータ

パラメータ	説明
string-expression	削除される文字列。

戻り値
VARCHAR

NVARCHAR

LONG VARCHAR

LONG NVARCHAR

注意： 結果データ型は LONG VARCHAR です。**SELECT INTO** 文で **TRIM** を使用する場合は、非構造化データ分析オプションのライセンスを所有しているか、**CAST** を使用して **TRIM** を正しいデータ型とサイズに設定する必要があります。

例
次の文は、先行ブランクや後続ブランクがすべて削除された値 "chocolate" を返します。

```
SELECT TRIM( '   chocolate   ' ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise ではサポートされていません。代わりに **LTRIM** と **RTRIM** を使用してください。

TRUNCNUM 関数 [数値]

数値を小数点以下の指定された桁数で切り捨てます。

構文

```
TRUNCNUM ( numeric-expression, integer-expression )
```

パラメータ

パラメータ	説明
numeric-expression	切り捨てられる数値。
integer-expression	正の整数は、丸めを行う小数点以下の有効桁数を指定します。負の式は、丸めを行う小数点の左側の有効桁数を指定します。

戻り値
NUMERIC

例
次の文は、値 600 を返します。

```
SELECT TRUNCNUM( 655, -2 ) FROM iq_dummy
```

次の文は、 値 655.340 を返します。

```
SELECT TRUNCNUM( 655.348, 2 ) FROM iq_dummy
```

使用法
この関数は **TRUNCATE** と同じですが、キーワードの矛盾が起こりません。

ROUND、**FLOOR**、**CEILING** を組み合わせて使用することで類似した機能を提供できます。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise ではサポートされていません。

参照：

- ROUND 関数 [数値] (301 ページ)

TS_ARMA_AR 関数 [時系列]

自己回帰移動平均 (ARMA) モデルのパラメータの最小二乗近似を計算し、要求された自己回帰近似を返します。

構文

```
TS_ARMA_AR (timeseries_expression, ar_count, ar_elem, method)
OVER (window-spec)
```

注意： この関数は、RAP - The Trading Edition Enterprise でのみ使用できます。この関数の詳細については、『時系列ガイド』を参照してください。

TS_ARMA_CONST 関数 [時系列]

自己回帰移動平均 (ARMA) モデルのパラメータの最小二乗近似を計算し、推定定数を返します。

構文

```
TS_ARMA_CONST (timeseries_expression, method)
OVER (window-spec)
```

注意： この関数は、RAP – The Trading Edition Enterprise でのみ使用できます。この関数の詳細については、『時系列ガイド』を参照してください。

TS_ARMA_MA 関数 [時系列]

自己回帰移動平均 (ARMA) モデルのパラメータの最小二乗近似を計算し、要求された自己回帰移動平均近似を返します。

構文

```
TS_ARMA_MA (timeseries_expression, ma_count, ma_elem, method)
OVER (window-spec)
```

注意： この関数は、RAP – The Trading Edition Enterprise でのみ使用できます。この関数の詳細については、『時系列ガイド』を参照してください。

TS_AUTOCORRELATION 関数 [時系列]

定常時系列のサンプル自己相関関数を計算します。

構文

```
TS_AUTOCORRELATION (timeseries_expression, lagmax, lag_elem)
OVER (window-spec)
```

注意： この関数は、RAP – The Trading Edition Enterprise でのみ使用できます。この関数の詳細については、『時系列ガイド』を参照してください。

TS_AUTO_ARIMA 関数 [時系列]

乗法的季節自己回帰和分移動平均 (ARIMA) モデルのパラメータを決定し、系列の終端を超えて影響が持続する異常値の影響を組み込んで予測を生成します。

構文

```
TS_AUTO_ARIMA( time_value , timeseries_expression      [, max_lag [,
critical        [, epsilon [, criterion     [, confidence [, model [,
n_predictions ]]]]]] )
OVER ( window-spec )
```

注意： この関数は、RAP – The Trading Edition Enterprise でのみ使用できます。この関数の詳細については、『時系列ガイド』を参照してください。

TS_AUTO_ARIMA_OUTLIER 関数 [時系列]

TS_AUTO_ARIMA 集合関数と同様、**TS_AUTO_ARIMA_OUTLIER** は入力時系列を受け入れ、自動的に乗法的季節自己回帰和分移動平均 (ARIMA) モデルのパラメータを決定します。

ただし、**TS_AUTO_ARIMA** は ARIMA モデルを使って一連の入力から値を予測しますが、TS_AUTO_ARIMA_OUTLIER は、ARIMA モデルを使って統計的異常値である要素を入力時系列内で特定し、それぞれの異常値タイプを返します。

構文

```
TS_AUTO_ARIMA_OUTLIER( time_value , timeseries_expression      [,
max_lag [, critical        [, epsilon [, criterion     [, confidence [,
model [, delta ]]]]]] )
OVER ( window-spec )
```

注意： この関数は、RAP – The Trading Edition Enterprise でのみ使用できます。この関数の詳細については、『時系列ガイド』を参照してください。

TS_AUTO_ARIMA_RESULT_AIC 関数 [時系列]

TS_AUTO_ARIMA によって生成された赤池情報量基準 (AIC) 出力パラメータを取得します。

構文

```
TS_AUTO_ARIMA_ RESULT_AIC( auto_arima_result )
```

注意： この関数は、RAP – The Trading Edition Enterprise でのみ使用できます。この関数の詳細については、『時系列ガイド』を参照してください。

TS_AUTO_ARIMA_RESULT_AICC 関数 [時系列]

TS_AUTO_ARIMA によって生成された補正 AIC (AICC) 出力パラメータを取得します。

構文

```
TS_AUTO_ARIMA_ RESULT_AICC( auto_arima_result )
```

注意： この関数は、RAP – The Trading Edition Enterprise でのみ使用できます。この関数の詳細については、『時系列ガイド』を参照してください。

TS_AUTO_ARIMA_RESULT_BIC 関数 [時系列]

TS_AUTO_ARIMA によって生成されたベイズ情報量基準 (BIC) 出力パラメータを取得します。

構文

```
TS_AUTO_ARIMA_ RESULT_BIC( auto_arima_result )
```

注意： この関数は、RAP – The Trading Edition Enterprise でのみ使用できます。この関数の詳細については、『時系列ガイド』を参照してください。

TS_AUTO_ARIMA_RESULT_FORECAST_VALUE 関数 [時系列]

TS_AUTO_ARIMA によって生成された、要求された異常値のない系列の予測値を取得します。

構文

```
TS_AUTO_ARIMA_RESULT_FORECAST_VALUE( auto_arima_result ,
model_element_number )
```

注意： この関数は、RAP – The Trading Edition Enterprise でのみ使用できます。この関数の詳細については、『時系列ガイド』を参照してください。

TS_AUTO_ARIMA_RESULT_FORECAST_ERROR 関数 [時系列]

TS_AUTO_ARIMA によって生成された、元の入力系列の予測標準誤差値を取得します。

構文

```
TS_AUTO_ARIMA_RESULT_FORECAST_ERROR( auto_arima_result ,
forecast_element_number )
```

注意： この関数は、RAP – The Trading Edition Enterprise でのみ使用できます。この関数の詳細については、『時系列ガイド』を参照してください。

TS_AUTO_ARIMA_RESULT_MODEL_D 関数 [時系列]

ARIMA モデル記述の計算において、TS_AUTO_ARIMA によって生成された *d* 値を取得します。

構文

```
TS_AUTO_ARIMA_ RESULT_MODEL_D( auto_arima_result )
```

注意： この関数は、RAP – The Trading Edition Enterprise でのみ使用できます。この関数の詳細については、『時系列ガイド』を参照してください。

TS_AUTO_ARIMA_RESULT_MODEL_P 関数 [時系列]

ARIMA モデル記述の計算において、TS_AUTO_ARIMA によって生成された *p* 値を取得します。

構文

```
TS_AUTO_ARIMA_ RESULT_MODEL_P( auto_arima_result )
```

注意： この関数は、RAP – The Trading Edition Enterprise でのみ使用できます。この関数の詳細については、『時系列ガイド』を参照してください。

TS_AUTO_ARIMA_RESULT_MODEL_Q [時系列]

ARIMA モデル記述の計算において、TS_AUTO_ARIMA によって生成された *q* 値を取得します。

構文

```
TS_AUTO_ARIMA_ RESULT_MODEL_Q( auto_arima_result )
```

注意： この関数は、RAP – The Trading Edition Enterprise でのみ使用できます。この関数の詳細については、『時系列ガイド』を参照してください。

TS_AUTO_ARIMA_RESULT_MODEL_S 関数 [時系列]

ARIMA モデル記述の計算において、TS_AUTO_ARIMA によって生成された *s* 値を取得します。

構文

```
TS_AUTO_ARIMA_ RESULT_MODEL_S( auto_arima_result )
```

注意： この関数は、RAP – The Trading Edition Enterprise でのみ使用できます。この関数の詳細については、『時系列ガイド』を参照してください。

TS_AUTO_ARIMA_RESULT_RESIDUAL_SIGMA [時系列]

異常値のないデータ・ポイントの残差標準誤差を取得します。

構文

```
TS_AUTO_ARIMA_ RESULT_RESIDUAL_SIGMA ( auto_arima_result )
```

注意： この関数は、RAP – The Trading Edition Enterprise でのみ使用できます。この関数の詳細については、『時系列ガイド』を参照してください。

TS_AUTO_UNI_AR 関数 [時系列]

単変量自己回帰時系列モデルの自動選択と適合を実行します。

構文

```
TS_AUTO_UNI_AR (timeseries_expression, ar_count, ar_elem, method)
OVER (window-spec)
```

注意： この関数は、RAP – The Trading Edition Enterprise でのみ使用できます。この関数の詳細については、『時系列ガイド』を参照してください。

TS_BOX_COX_XFORM 関数 [時系列]

前方または逆 Box-Cox ベキ変換を実行します。

構文

```
TS_BOX_COX_XFORM (timeseries_expression, power [, shift [,
inverse] ]) OVER (window-spec)
```

注意： この関数は、RAP – The Trading Edition Enterprise でのみ使用できます。この関数の詳細については、『時系列ガイド』を参照してください。

TS_DIFFERENCE 関数 [時系列]

季節時系列と非季節時系列の差異を計算します。

構文

```
TS_DIFFERENCE (timeseries_expression, period1 [, period2 [, ...period 10] ]) OVER (window-spec)
```

注意： この関数は、RAP - The Trading Edition Enterprise でのみ使用できます。この関数の詳細については、『時系列ガイド』を参照してください。

TS_DOUBLE_ARRAY [時系列]

TS_GARCH 関数のサポート関数。3 個から 10 個の定数の倍精度浮動小数点値を含む論理配列を作成し、単一の varbinary 値を返します。

構文

```
TS_DOUBLE_ARRAY(xguess1, xguess2, xguess3, [ ... [ , xguess10] ...] ])
```

注意： この関数は、RAP – The Trading Edition Enterprise でのみ使用できます。この関数の詳細については、『時系列ガイド』を参照してください。

TS_ESTIMATE_MISSING 関数 [時系列]

時系列内の欠落値を推定し、元の時系列に組み込み、新しい時系列として返します。

構文

```
TS_ESTIMATE_MISSING (timeseries_expression, method)
OVER (window-spec)
```

注意： この関数は、RAP - The Trading Edition Enterprise でのみ使用できます。この関数の詳細については、『時系列ガイド』を参照してください。

TS_GARCH 関数 [時系列]

GARCH(p, q) モデルのパラメータの予測値を計算します。

構文

```
TS_GARCH ( <time series expression> , <garch_count> , <arch_count>,
<xguess_binary_encoding> ,[, <max_sigma> ])
OVER (window-spec)
```

注意： この関数は、RAP – The Trading Edition Enterprise でのみ使用できます。この関数の詳細については、『時系列ガイド』を参照してください。

TS_GARCH_RESULT_A 関数 [時系列]

TS_GARCH 関数のサポート関数。**TS_GARCH** 集合関数によって生成された、対数尤度出力パラメータ *A* を取得します。

構文

```
TS_GARCH_RESULT_A ( ts_garch_result )
```

注意： この関数は、RAP – The Trading Edition Enterprise でのみ使用できます。この関数の詳細については、『時系列ガイド』を参照してください。

TS_GARCH_RESULT_AIC 関数 [時系列]

TS_GARCH 関数のサポート関数。**TS_GARCH** 集合関数によって生成された、赤池情報量基準出力パラメータ *AIC* を取得します。

構文

```
TS_GARCH_RESULT_AIC ( ts_garch_result )
```

注意： この関数は、RAP – The Trading Edition Enterprise でのみ使用できます。この関数の詳細については、『時系列ガイド』を参照してください。

TS_GARCH_RESULT_USER [時系列]

TS_GARCH 関数のサポート関数。GARCH(p,q) モデルを記述する論理配列内の各要素にアクセスします。

構文

```
TS_GARCH_RESULT_USER ( ts_garch_result , model_element_number )
```

注意： この関数は、RAP – The Trading Edition Enterprise でのみ使用できます。この関数の詳細については、『時系列ガイド』を参照してください。

TS_INT_ARRAY [時系列]

TS_AUTO_ARIMA 関数と TS_AUTO_ARIMA_OUTLIER 関数のためのサポート関数。varbinary 値としてコード化された定数整数値を含む論理配列を作成します。

構文

```
TS_INT_ARRAY( int1 , int2 , int3 , int4 ,[...[,  int10 ]...])
```

注意： この関数は、RAP – The Trading Edition Enterprise でのみ使用できます。この関数の詳細については、『時系列ガイド』を参照してください。

TS_LACK_OF_FIT 関数 [時系列]

適切な相関関数を前提として単変量時系列または伝達関数の適合性不足テスト (LOF) を実行します。

構文

```
TS_LACK_OF_FIT (timeseries_expression, p_value, q_value, lagmax,
[tolerance])

OVER (window-spec)
```

注意： この関数は、RAP – The Trading Edition Enterprise でのみ使用できます。この関数の詳細については、『時系列ガイド』を参照してください。

TS_LACK_OF_FIT_P 関数 [時系列]

単変量時系列に対して適合性不足テストを実行します。この関数は、q を返すのではなく、q の p 値を返す点を除き、TS_LACK_OF_FIT 関数と同じです。

構文

```
TS_LACK_OF_FIT_P (timeseries_expression, p_value, q_value, lagmax,
[tolerance])

OVER (window-spec)
```

注意： この関数は、RAP – The Trading Edition Enterprise でのみ使用できます。この関数の詳細については、『時系列ガイド』を参照してください。

TS_MAX_ARMA_AR 関数 [時系列]

単変量 ARMA (自己回帰移動平均) 時系列モデルにある引数の正確な最尤推定を計算し、要求された自己回帰推定を返します。

構文

```
TS_MAX_ARMA_AR (timeseries_expression, ar_count, ar_elem)
OVER (window-spec)
```

注意： この関数は、RAP – The Trading Edition Enterprise でのみ使用できます。この関数の詳細については、『時系列ガイド』を参照してください。

TS_MAX_ARMA_CONST 関数 [時系列]

単変量 ARMA (自己回帰移動平均) 時系列モデルにある引数の正確な最尤推定を計算し、定数推定を返します。

構文

```
TS_MAX_ARMA_CONST (timeseries_expression)
OVER (window-spec)
```

注意： この関数は、RAP – The Trading Edition Enterprise でのみ使用できます。この関数の詳細については、『時系列ガイド』を参照してください。

TS_MAX_ARMA_LIKELIHOOD 関数 [時系列]

単変量 ARMA (自己回帰移動平均) 時系列モデルにある引数の正確な最尤推定を計算し、適合モデルの尤度値 (ln) を返します。

構文

```
TS_MAX_ARMA_LIKELIHOOD (timeseries_expression)
OVER (window-spec)
```

注意： この関数は、RAP – The Trading Edition Enterprise でのみ使用できます。この関数の詳細については、『時系列ガイド』を参照してください。

TS_MAX_ARMA_MA 関数 [時系列]

単変量 ARMA (自己回帰移動平均) 時系列モデルにある引数の正確な最尤推定を計算し、要求された移動平均推定を返します。

構文

```
TS_MAX_ARMA_MA (timeseries_expression, ma_count, ma_elem)
```

```
OVER (window-spec)
```

注意： この関数は、RAP – The Trading Edition Enterprise でのみ使用できます。この関数の詳細については、『時系列ガイド』を参照してください。

TS_OUTLIER_IDENTIFICATION 関数 [時系列]

異常値の検出および判定を行うと同時に、異常値のない基礎系列が一般的な季節または非季節 ARMA モデルに従う時系列におけるモデル引数を推定します。

構文

```
TS_OUTLIER_IDENTIFICATION (timeseries_expression, p_value, q_value,
s_value, d_value, [, delta_value[, critical_value]])

OVER (window-spec)
```

注意： この関数は、RAP – The Trading Edition Enterprise でのみ使用できます。この関数の詳細については、『時系列ガイド』を参照してください。

TS_PARTIAL_AUTOCORRELATION 関数 [時系列]

定常時系列のサンプル偏自己相関関数を計算します。

構文

```
TS_PARTIAL_AUTOCORRELATION (timeseries_expression, lagmax, lag_elem)

OVER (window-spec)
```

注意： この関数は、RAP – The Trading Edition Enterprise でのみ使用できます。この関数の詳細については、『時系列ガイド』を参照してください。

TS_VWAP 関数 [時系列]

VWAP は Volume-Weighted Average Price (出来高加重平均価格) の略語です。
TS_VWAP は、特定の対象期間の取引値の総出来高に対する比率を計算します。

VWAP は、証券の定義済みの取引期間における平均価格の評価基準です。
TS_VWAP は、単純な集合関数としても OLAP スタイルの集合関数としても使用できます。

他の時系列関数とは異なり、TS_VWAP は IMSL ライブラリを呼び出しません。

構文 1

```
TS_VWAP (price_expression, volume_expression)
```

構文 2

```
TS_VWAP (price_expression, volume_expression)
```

```
OVER (window-spec)
```

注意： この関数は、RAP – The Trading Edition Enterprise でのみ使用できます。この関数の詳細については、『時系列ガイド』を参照してください。

UCASE 関数 [文字列]

文字列内のすべての文字を大文字に変換します。

構文

```
UCASE ( string-expression )
```

パラメータ

パラメータ	説明
string-expression	大文字に変換される文字列。

戻り値

VARCHAR

NVARCHAR

LONG VARCHAR

LONG NVARCHAR

注意： 結果データ型は LONG VARCHAR です。**SELECT INTO** 文で **UCASE** を使用する場合は、非構造化データ分析オプションのライセンスを所有しているか、**CAST** を使用して **UCASE** を正しいデータ型とサイズに設定する必要があります。

例

次の文を実行すると、値 "CHOCOLATE" が返ります。

```
SELECT UCASE( 'ChocoLate' ) FROM iq_dummy
```

標準と互換性

- SQL － ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — **UCASE** は Adaptive Server Enterprise でサポートされないが、**UPPER** が同等の機能を提供。

参照：

- LCASE 関数 [文字列] (231 ページ)
- LEFT 関数 [文字列] (234 ページ)
- LOWER 関数 [文字列] (242 ページ)
- REPLACE 関数 [文字列] (295 ページ)

- REVERSE 関数 [文字列] (299 ページ)
- RIGHT 関数 [文字列] (300 ページ)
- UPPER 関数 [文字列] (349 ページ)

UPPER 関数 [文字列]

文字列内のすべての文字を大文字に変換します。

構文

```
UPPER ( string-expression )
```

パラメータ

パラメータ	説明
string-expression	大文字に変換される文字列。

戻り値

VARCHAR

NVARCHAR

LONG VARCHAR

LONG NVARCHAR

注意： 結果データ型は LONG VARCHAR です。**SELECT INTO** 文で **UPPER** を使用する場合は、非構造化データ分析オプションのライセンスを所有しているか、**CAST** を使用して **UPPER** を正しいデータ型とサイズに設定する必要があります。

例

次の文を実行すると、値 "CHOCOLATE" が返ります。

```
SELECT UPPER( 'ChocoLate' ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise 互換。

参照：

- LCASE 関数 [文字列] (231 ページ)
- LEFT 関数 [文字列] (234 ページ)
- LOWER 関数 [文字列] (242 ページ)
- REPLACE 関数 [文字列] (295 ページ)
- REVERSE 関数 [文字列] (299 ページ)

- RIGHT 関数 [文字列] (300 ページ)
- UCASE 関数 [文字列] (348 ページ)

USER_ID 関数 [システム]

ユーザ ID 番号を整数で返します。

構文

```
USER_ID ( [ user-name ] )
```

パラメータ

パラメータ	説明
user-name	ユーザ名。

戻り値
INT

例
次の文は、ユーザ ID 番号 1 を返します。

```
SELECT USER_ID ('DBA') FROM iq_dummy
```

次の文は、ユーザ ID 番号 0 を返します。

```
SELECT USER_ID ('SYS') FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Sybase IQ 用に実装された Adaptive Server Enterprise 関数。

参照：

- SUSER_ID 関数 [システム] (333 ページ)
- SUSER_NAME 関数 [システム] (334 ページ)
- USER_NAME 関数 [システム] (350 ページ)

USER_NAME 関数 [システム]

ユーザ名を返します。

構文

```
USER_NAME ( [ user-id ] )
```

パラメータ

パラメータ	説明
user-id	ユーザ ID 番号。

戻り値
LONG VARCHAR

注意： 結果データ型は LONG VARCHAR です。**SELECT INTO** 文で **USER_NAME** を使用する場合は、非構造化データ分析オプションのライセンスを所有しているか、**CAST** を使用して **USER_NAME** を正しいデータ型とサイズに設定する必要があります。

例
次の文は、値 "DBA" を返します。

```
SELECT USER_NAME ( 1 ) FROM iq_dummy
```

次の文は、値 "SYS" を返します。

```
SELECT USER_NAME ( 0 ) FROM iq_dummy
```

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Sybase IQ 用に実装された Adaptive Server Enterprise 関数。 Adaptive Server Enterprise の場合、USER_NAME はサーバのユーザ名ではなくユーザ名を返します。

参照：
- SUSER_ID 関数 [システム] (333 ページ)
- SUSER_NAME 関数 [システム] (334 ページ)
- USER_ID 関数 [システム] (350 ページ)

UUIDTOSTR 関数 [文字列]

一意な識別子 (UUID、GUID とも呼ばれる) の値を、文字列値に変換します。

構文

```
UUIDTOSTR ( uuid-expression )
```

パラメータ

表 95 : パラメータ

パラメータ	説明
uuid-expression	一意な識別子の値。

戻り値
VARCHAR

例
一意な識別子の値を読みやすい形式に変換するには、次のようなクエリを実行します。

```
CREATE TABLE T3 (
pk uniqueidentifier primary key,c1 int);
INSERT INTO T3 (pk, c1)
values (0x12345678123456789012123456789012, 1)
SELECT UUIDTOSTR(pk) FROM T3
```

使用法
一意な識別子を、*xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx* の形式の文字列値に変換します (x は 16 進の桁)。バイナリ値が一意な識別子として有効でない場合は、NULL が返されます。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise ではサポートされていません。

参照：
- バイナリ・データ型 (84 ページ)
- NEWID 関数 [その他] (253 ページ)
- STRTOUUID 関数 [文字列] (328 ページ)
- 文字データ型 (75 ページ)
- バイナリ・データ型 (710 ページ)

VAR_POP 関数 [集合]

1 つの数値式で構成される母集団の統計分散を DOUBLE 型として計算します。

構文

```
VAR_POP ( [ ALL ] expression )
```

パラメータ

パラメータ	説明
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 (通常はカラム名)。

戻り値
DOUBLE

例
次の文は、異なる期間における注文ごとの項目数で平均と平方偏差をリストします。

```
SELECT year( ShipDate ) AS Year, quarter( ShipDate )
  AS Quarter, AVG( Quantity ) AS Average,
  VAR_POP( Quantity ) AS Variance
FROM SalesOrderItems GROUP BY Year, Quarter
  ORDER BY Year, Quarter
```

Year	Quarter	Average	Variance
2000	1	25.775148	203.9021
2000	2	27.050847	225.8109
...	...	...	...

使用法
グループまたはパーティションの各ロー (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 *value expression* の母分散を計算します。これは、*value expression* から、グループまたはパーティション内の (残りの) ローの数で割られた *value expression* の平均を差し引いた値の 2 乗和として定義されます。

母集団ベースの分散は、次の式に従って計算されます。

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

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise ではサポートされていません。

参照：

- ウィンドウ集合関数の使用法 (111 ページ)

VAR_SAMP 関数 [集合]

1 つの数値式で構成される標本の統計分散を DOUBLE 型として計算します。

構文

```
VAR_SAMP ( [ ALL ] expression )
```

パラメータ

パラメータ	説明
expression	その標本ベースの分散がローのセットに対して計算される式 (通常はカラム名)。

戻り値
DOUBLE

例
次の文は、異なる期間における注文ごとの項目数で平均と平方偏差をリストします。

```
SELECT year( ShipDate ) AS Year, quarter( ShipDate )
  AS Quarter, AVG( Quantity ) AS Average,
  VAR_SAMP( Quantity ) AS Variance
FROM SalesOrderItems GROUP BY Year, Quarter
  ORDER BY Year, Quarter
```

Year	Quarter	Average	Variance
2000	1	25.775148	205.1158
2000	2	27.050847	227.0939
...	...	...	...

使用法

注意： VAR_SAMP は、**VARIANCE** のエイリアスです。

グループまたはパーティションの各ロー (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 *value expression* の標本分散を計算します。これは、*value expression* から、グループまたはパーティション内の残りのローより 1 少ない数で割った *value expression* の平均を差し引いた値の 2 乗和として定義されます。

Sybase IQ 12.7 以降では、1 要素の入力セットの場合、NULL によって NULL が返されます。 12.7 よりも前のバージョンの場合、NULL はゼロを返します。

分散は次の式に従って計算されます。これは、正規分布とみなされます。

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

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise によるサポートなし。

参照：

- ウィンドウ集合関数の使用法 (111 ページ)

VARIANCE 関数 [集合]

数値セットの分散を返します。

構文

```
VARIANCE ( [ ALL ] expression )
```

パラメータ

パラメータ	説明
expression	任意の数値式 (FLOAT、REAL、または DOUBLE)。 その標本ベースの分散がローのセットに対して計算される式 (通常はカラム名)。

戻り値

DOUBLE

例

次のようなデータがあるとします。

```
SELECT Salary FROM Employees WHERE DepartmentID = 300
```

salary
51432.000
57090.000
42300.000
43700.00

salary
36500.000
138948.000
31200.000
58930.00
75400.00

次の文は、値 1063923790.99999994 を返します。

```
SELECT VARIANCE ( Salary ) FROM Employees
WHERE DepartmentID = 300
```

次のようなデータがあるとします。

```
SELECT UnitPrice FROM Products WHERE name = 'Tee Shirt'
```

UnitPrice
9.00
14.00
14.00

次の文は、値 8.33333333333334327 を返します。

```
SELECT VARIANCE ( UnitPrice ) FROM Products
WHERE name = 'Tee Shirt'
```

使用法

VARIANCE では、次の計算式が使用されます。

$$var = \frac{n\sum x^2 - (\sum x)^2}{n(n-1)}$$

VARIANCE は、結果を `double-precision floating-point` のデータ型で返します。空のセットに適用すると、結果は NULL になり、1 要素の入力セットに NULL が返されます。

VARIANCE は、キーワード DISTINCT をサポートしません。**VARIANCE** で DISTINCT を使用すると、構文エラーが返されます。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise ではサポートされていません。

参照：

- ウィンドウ集合関数の使用法 (111 ページ)
- STDDEV 関数 [集合] (320 ページ)
- STDDEV_SAMP 関数 [集合] (323 ページ)

WEEKS 関数 [日付と時刻]

任意の開始日時から経過した週の数を返すか、指定された 2 つの日時の間の週の数を返すか、または integer-expression で指定された週の数を日時に追加します。

構文

```
WEEKS ( datetime-expression
| datetime-expression, datetime-expression
| datetime-expression, integer-expression )
```

パラメータ

パラメータ	説明
datetime-expression	日時。
integer-expression	*datetime-expression* に追加する週数。*integer-expression* が負の場合、日時の値から適切な週数が引かれます。時間、分、秒は無視されます。整数式を指定する場合は、*datetime-expression* を DATETIME データ型として明示的にキャストする必要があります。

戻り値

構文 1 は、INTEGER を返します。

構文 2 は、TIMESTAMP を返します。

例

次の文は、値 104278 を返します。

```
SELECT WEEKS( '1998-07-13 06:07:12' ) FROM iq_dummy
```

次の文を実行すると、2 つの日付の間の差である値 9 が返されます。

```
SELECT WEEKS( '1999-07-13 06:07:12',
    '1999-09-13 10:07:12' ) FROM iq_dummy
```

次の文を実行すると、タイムスタンプの値 1999-06-16 21:05:07.000 が返ります。

```
SELECT WEEKS( CAST( '1999-05-12 21:05:07'
AS TIMESTAMP ), 5) FROM iq_dummy
```

使用法
週は、北米で使用されているカレンダーに従い、日曜日に始まり土曜日に終わるものとして定義されています。1 つ目の構文で返される数値は、2 つの日付が同じ週かどうかを判断するのによく利用されます。

```
WEEKS ( invoice_sent ) = WEEKS ( payment_received ) FROM iq_dummy
```

2 つ目の構文では、**WEEKS** の値は 2 つの日付の間にある日曜日の数で計算されます。時間、分、秒は無視されます。この関数は、 DATE_FIRST_DAY_OF_WEEK オプションの影響を受けません。

標準と互換性

- SQL ― ISO/ANSI SQL 文法のベンダ拡張。
- Sybase ― Adaptive Server Enterprise ではサポートされていません。

参照：

- CAST 関数 [データ型変換] (144 ページ)
- CONVERT 関数 [データ型変換] (155 ページ)
- HOURS 関数 [日付と時刻] (214 ページ)
- MINUTES 関数 [日付と時刻] (248 ページ)
- MONTHS 関数 [日付と時刻] (251 ページ)
- REPLACE 関数 [文字列] (295 ページ)
- SECOND 関数 [日付と時刻] (306 ページ)
- YEAR 関数 [日付と時刻] (362 ページ)
- YEARS 関数 [日付と時刻] (363 ページ)

WEIGHTED_AVG 関数 [集合]

算術 (または線形) 加重平均を計算します。

加重平均は、平均に使用される各数量に加重が付加される平均です。加重は、平均を構成する各数量の相対的な重要性を決定します。

構文

```
WEIGHTED_AVG ( expression )

                    OVER ( window-spec )
```

window-spec： 以下の「使用法」を参照してください。

パラメータ

パラメータ	説明
expression	計算する加重値の数値式。

使用法

WEIGHTED_AVG 関数を使用して、加重移動平均を作成します。加重移動平均では、加重は時間の経過に従って等差階級的に減少します。加重は、直近のデータ・ポイントの最高加重から 0 に減少します。

図 1 : WEIGHTED_AVG の計算

$$WMA_M = \frac{np_M + (n-1)p_{M-1} + \cdots + 2p_{M-n+2} + p_{M-n+1}}{n + (n-1) + \cdots + 2 + 1}$$

加重を計算するために、2 つ異常の加重移動平均を一緒に平均するか、**EXP_WEIGHTED_AVG** 関数を使用できます。

関数構文 (インライン) または **SELECT** 文の **WINDOW** 句で、*window-spec* の要素を指定できます。

window-spec:

- 1 つの ORDER BY 指定子を含む必要があります。
- FOLLOWING および RANGE 指定子を含むことはできません。
- ROW 指定子 (指定した場合) の 2 番目の引数は CURRENT ROW である必要があります。
- NULL 値を含むことはできません。
- DISTINCT 指定子を含むことはできません。
- UNBOUNDED PRECEDING はサポートされていますが、使用した場合のパフォーマンスは低い場合があります。

例

次の例は、フロリダの部門ごとの従業員の給与の加重移動平均、および平均のほとんどの加重に関係する現在雇用されている従業員の給与を返します。

```
SELECT DepartmentID, Surname, Salary, WEIGHTED_AVG(Salary) OVER
(PARTITION BY DepartmentID
ORDER BY YEAR(StartDate) DESC) as "W_AVG"
FROM Employees
WHERE State IN ('FL') ORDER BY DepartmentID
```

次の結果セットが返されます。

表 96 : WEIGHTED_AVG の結果セット

DepartmentID	Surname	Salary	W_AVG
100	Lull	87,900.000	87,900.000000
100	Gowda	59,840.000	69,193.333333
200	Sterling	64,900.000	64,900.000000
200	Kelly	87,500.000	79,966.666667
300	Litton	58,930.000	58,930.000000
400	Evans	68,940.000	68,940.000000
400	Charlton	28,300.000	41,846.666667
400	Francis	53,870.000	47,858.333333

標準と互換性

- SQL － ISO/ANSI SQL 文法のベンダ拡張。

参照：

- EXP_WEIGHTED_AVG 関数 [集合] (200 ページ)
- ウィンドウ集合関数の使用法 (111 ページ)

WIDTH_BUCKET 関数 [数値]

与えられた式に対して、**WIDTH_BUCKET** 関数は、この式の評価後の結果に割り当てられるバケット番号を返します。

構文

```
WIDTH_BUCKET ( expression, min_value, max_value, num_buckets )
```

パラメータ

パラメータ	説明
expression	ヒストグラムが作成されている式です。この式は、数値または日時の値、または暗黙で数値または日時の値に変換できる値に評価される必要があります。*expr* が null に評価されると、式は null を返します。
min_value	*expr* に使用できる範囲の各ポイントに解決される式。数値または日時値にも評価される必要があり、null には評価できません。

パラメータ	説明
max_value	*expr* に使用できる範囲の各ポイントに解決される式。数値または日時値にも評価される必要があり、null には評価できません。
num_buckets	バケット数を示す定数に解決される式。この式は正の整数に評価される必要があります。

例

次の例では、サンプル・テーブル内のマサチューセッツ州の顧客の `credit_limit` カラムに 10 のバケット・ヒストグラムを作成し、各顧客のバケット数 ("Credit Group") を返します。最大値を超える限度額が設定されている顧客は、オーバフロー・バケット 11 に割り当てられます。

```
select EmployeeID, Surname, Salary, WIDTH_BUCKET(Salary, 29000,
60000, 4) "Wages" from Employees where State = 'FL' order by "Wages"
```

```
EMPLOYEEID   SURNAME               SALARY         Wages
----------   -------               ------         -----
888          Charlton           28300.000             0
1390         Litton             58930.000             4
207          Francis            53870.000             4
266          Gowda              59840.000             4
445          Lull               87900.000             5
1021         Sterling           64900.000             5
902          Kelly              87500.000             5
1576         Evans              68940.000             5
```

範囲が逆の場合、バケットはオープン・クローズ間隔になります。次に例を示します。**WIDTH_BUCKET** (*credit_limit*, *5000*, *0*, *5*). この例では、バケット番号 1 は (4000, 5000)、バケット番号 2 は (3000, 4000)、およびバケット番号 5 は (0, 1000) です。オーバフロー・バケットには 0 (5000,+infinity) の番号が付き、アンダフロー・バケットには 6 (-infinity, 0) の番号が付きます。

使用法

WIDTH_BUCKET 関数を使用して等幅ヒストグラムを生成できます。等幅ヒストグラムでは、データ・セットを間隔サイズ (最も高い値から最も低い値まで) の同じバケットに分割します。保持されるロー数はバケットごとに異なります。関連する関数の **NTILE** は、等高バケットを作成します。

等幅ヒストグラムは数値、日付、日時データ型でのみ生成されるため、最初の 3 つのパラメータはすべて数値式またはすべて日付式にする必要があります。他の型の式は使用できません。最初のパラメータが NULL の場合、結果は NULL です。2 番目および 3 番目のパラメータが NULL の場合、エラー・メッセージが返されます。これは、NULL 値は日付または数値次元の範囲のどの終了ポイント (またはあらゆるポイント) も示すことができないためです。最後のパラメータ (バケットの

数) は、正の整数値に評価される数値式にする必要があります。0、NULL、または負の値にすると、エラーが発生します。

バケットには 0 から (n+1) まで番号が付けられます。バケット 0 は、最小値未満の値のカウントを保持します。バケット (n+1) は、指定された最大値以上の値のカウントを保持します。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise ではサポートされていません。

YEAR 関数 [日付と時刻]

指定された日付／時刻の年に対応する 4 桁の数を返します。

構文

```
YEAR ( datetime-expression )
```

パラメータ

パラメータ	説明
datetime-expression	日時。

戻り値
SMALLINT

例
次の文は、値 1998 を返します。

```
SELECT YEAR( '1998-07-13 06:07:12' ) FROM iq_dummy
```

使用法
YEAR 関数は **YEARS** 関数の最初の構文と同じです。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise ではサポートされていません。

参照：

- CAST 関数 [データ型変換] (144 ページ)
- CONVERT 関数 [データ型変換] (155 ページ)
- HOURS 関数 [日付と時刻] (214 ページ)

- MINUTES 関数 [日付と時刻] (248 ページ)
- MONTHS 関数 [日付と時刻] (251 ページ)
- REPLACE 関数 [文字列] (295 ページ)
- SECOND 関数 [日付と時刻] (306 ページ)
- WEEKS 関数 [日付と時刻] (357 ページ)
- YEARS 関数 [日付と時刻] (363 ページ)
- NTILE 関数 [統計] (258 ページ)

YEARS 関数 [日付と時刻]

指定された日時の年に対応する 4 桁の数を返すか、指定された 2 つの日時の間の年数を返すか、または integer-expression で指定された年数を日時に追加します。

構文

```
YEARS ( datetime-expression
| datetime-expression, datetime-expression
| datetime-expression, integer-expression )
```

パラメータ

パラメータ	説明
datetime-expres-sio	日時。
integer-expres-sion	*datetime-expression* に追加する年数。*integer-expression* が負の場合は、指定された年数が日時の値から減算されます。整数式を指定する場合は、*datetime-expression* を DATETIME データ型として明示的にキャストする必要があります。

戻り値

構文 1 は、INTEGER を返します。

構文 2 は、TIMESTAMP を返します。

例

次の文は、値 1998 を返します。

```
SELECT YEARS( '1998-07-13 06:07:12' ) FROM iq_dummy
```

次の文を実行すると、2 つの日付の間の差である値 2 が返されます。

```
SELECT YEARS( '1997-07-13 06:07:12',
    '1999-09-13 10:07:12' ) FROM iq_dummy
```

次の文を実行すると、YEARS(cast('1999-05-12 21:05:07' as timestamp), 5) 値 2004-05-12 21:05:07.000 を返します。

```
SELECT YEARS( CAST( '1999-05-12 21:05:07'
AS TIMESTAMP ), 5) FROM iq_dummy
```

使用法

YEARS 関数の 1 つ目の構文は **YEAR** 関数と同じです。

2 つ目の構文は、最初の日付から次の日付までの年数を返します。これは、2 つの日付の間にある、年の初日の数で計算されます。負の値が返ることもあります。時間、分、秒は無視されます。たとえば、次の文は指定された日付の間にある、年の初日の数である 2 を返します。

```
SELECT YEARS ( '2000-02-24', '2002-02-24' ) FROM iq_dummy
```

次の文で指定されている日付の差は、暦年で完全な 2 年間に達していませんが、この文を実行しても 2 が返されます。値 2 は、2 つの日付の間の、年の初日の数 (この例では 2001 年 1 月 1 日と 2002 年 1 月 1 日) です。

```
SELECT YEARS ( '2000-02-24', '2002-02-20' ) FROM iq_dummy
```

3 つ目の構文は、指定された日付に *integer-expression* の年数を追加します。処理後の日付が月の末日を過ぎてしまった場合 (**SELECT YEARS** (**CAST** ('1992-02-29' AS **TIMESTAMP**), 1) など)、結果はその月の末日にセットされます。*integer-expression* が負の場合、指定された年数が日付から減算されます。時間、分、秒は無視されます。

標準と互換性

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — Adaptive Server Enterprise ではサポートされていません。

参照：

- CAST 関数 [データ型変換] (144 ページ)
- CONVERT 関数 [データ型変換] (155 ページ)
- HOURS 関数 [日付と時刻] (214 ページ)
- MINUTES 関数 [日付と時刻] (248 ページ)
- MONTHS 関数 [日付と時刻] (251 ページ)
- REPLACE 関数 [文字列] (295 ページ)
- SECOND 関数 [日付と時刻] (306 ページ)
- WEEKS 関数 [日付と時刻] (357 ページ)
- YEAR 関数 [日付と時刻] (362 ページ)

YMD 関数 [日付と時刻]

指定された年、月、日に対応する日付を返します。

構文

```
YMD ( integer-expression1, integer-expression2, integer-expression3 )
```

パラメータ

パラメータ	説明
integer-expression1	年。
integer-expression2	月の番号。1 ～ 12 の範囲を超えて月を指定すると、それに応じて年が調整されます。
integer-expression3	日の番号。任意の整数値を日として指定できます。指定の値に応じて、日付が調整されます。

戻り値
DATE

例
次の文を実行すると、値 1998-06-12 が返ります。

```
SELECT YMD( 1998, 06, 12 ) FROM iq_dummy
```

通常の範囲を超えた値が指定されると、日付がその値に応じて調整されます。たとえば、次の文は値 1993-03-01 を返します。

```
SELECT YMD( 1992, 15, 1 ) FROM iq_dummy
```

次の文を実行すると、値 28.02.93 が返ります。

```
SELECT YMD ( 1992, 15, 1-1 ) FROM iq_dummy
```

次の文を実行すると、値 29.02.92 が返ります。

```
SELECT YMD ( 1992, 3, 1-1 ) FROM iq_dummy
```

標準と互換性

- SQL ― ISO/ANSI SQL 文法のベンダ拡張。
- Sybase ― Adaptive Server Enterprise ではサポートされていません。

他の SQL 言語との違い

Sybase IQ は、基本的には ANSI SQL89 規格に準拠しますが、IBM の DB2 と SAA 仕様、および ANSI SQL92 規格で定義された機能が追加されています。

Sybase IQ の以下の機能は、他の多くの SQL ソフトウェアには見られません。

日付

Sybase IQ には日付、時刻、タイムスタンプのデータ型があり、年、月、日、時、分、秒、小数点以下の秒が含まれます。日付フィールドへの挿入または更新、および日付フィールド間の比較については、フリー・フォーマットの日付がサポートされています。

また、日付に関しては以下の演算が可能です。

表 97 : 日付の演算

日付の演算	説明
date + integer	指定された値の日数を日付に加えます。
date - integer	指定された値の日数を日付から引きます。
date - date	2 つの日付間の日数を計算します。
date + time	日付と時刻からタイムスタンプを作成します。

日付と時刻の処理に使用できる関数は、数多くあります。

整合性

Sybase IQ では、エンティティ整合性と参照整合性の両方がサポートされています。

これは、**CREATE TABLE** および **ALTER TABLE** に対する次の 2 つの拡張機能によって実装されています。

```
PRIMARY KEY ( column-name, ... )
[NOT NULL] FOREIGN KEY [role-name]
                [(column-name, ...)]
            REFERENCES table-name [(column-name, ...)]
                [ CHECK ON COMMIT ]
```

PRIMARY KEY 句では、関係のプライマリ・キーを宣言します。IQ は、これにより、プライマリ・キーの一意性を確保し、プライマリ・キーのカラムに NULL 値が含まれないようにします。

FOREIGN KEY 句は、このテーブルと他のテーブルの関係を定義します。この関係は、別のテーブルのプライマリ・キーの値を含むこのテーブルのカラム (1 つまたは複数) によって表されます。システムは、この定義に基づいて、これらのカラムの参照整合性を確認します。これらのカラムが変更されるか、このテーブルにローが挿入されると、これらのカラムについて、1 つまたは複数が NULL であること、または値が他のテーブルのプライマリ・キーのローの対応するカラムに一致していることが、常にチェックされます。詳細については、「**CREATE TABLE** 文」を参照してください。

ジョイン

Sybase IQ は、テーブル間の **automatic joins** をサポートします。

他のソフトウェアでもサポートされる **NATURAL** ジョイン演算子と **OUTER** ジョイン演算子に加え、Sybase IQ では外部キー関係に基づく **KEY** ジョイン演算子がサポートされています。 これにより、ジョイン実行時における **WHERE** 句の複雑さを軽減できます。

更新

Sybase IQ では、複数のテーブルを **UPDATE** によって参照できます。

複数のテーブルで定義されているビューを更新することもできます。多くの SQL ソフトウェアでは、ジョイン・テーブルに対して更新を行えません。

テーブルの変更

ALTER TABLE が拡張されました。

エンティティ整合性や参照整合性の変更に加えて、次の種類の変更が可能です。

```
ADD column data-type
MODIFY column data-type
DELETE column
RENAME new-table-name
RENAME old-column TO new-column
```

MODIFY を使用して文字カラムの最大長を変更できます。また、あるデータ型から別のデータ型に変換できます。

サブクエリが許容されない場合

SQL Anywhere とは異なり、Sybase IQ では、式が使用できる場所でいつでもサブクエリを使用できるわけではありません。

Sybase IQ では、SQL-1989 の文法で許可されているケースと、最上位レベルのクエリ・ブロックの **SELECT** リスト、または **UPDATE** 文の **SET** 句内でのみサブクエリを使用できます。Sybase IQ は SQL Anywhere 拡張子をサポートしません。

多くの SQL ソフトウェアでは、比較演算子の右側でのみサブクエリを使用できます。 たとえば、次のコマンドは Sybase IQ では有効ですが、他のほとんどの SQL ソフトウェアでは有効ではありません。

```
SELECT        SurName,
            BirthDate,
            (    SELECT DepartmentName
                FROM Departments
                WHERE DepartmentID = Employees.EmployeeID
                AND DepartmentID = 200 )
FROM Employees
```

その他の関数

Sybase IQ は、ANSI SQL 定義にはない関数をいくつかサポートしています。

参照：

- SQL 関数 (107 ページ)

カーソル

Embedded SQL を使用する場合は、FETCH 文でカーソル位置を任意に移動できます。カーソルは、現在位置から相対的に、またはカーソルの最初または最後から指定のレコード数分、前後に移動できます。

物理的制限

Sybase IQ データベースでは、オブジェクトのサイズと数に制限があります。 ほとんどの場合、コンピュータのメモリおよびディスク容量から受ける制限の方が、大きな影響を持っています。

特定のプラットフォームにのみ適用される制限については、そのプラットフォームのマニュアルを参照してください。

表 98 : Sybase IQ データベース・オブジェクトのサイズと数の制限

項目	制限事項
カタログ・ファイル・サイズ	すべてのプラットフォームで上限は 1TB。 NTFS を使用する Windows システムは、最大 1TB をサポートする。
データベース名のサイズ	250 バイト
データベース・サイズ	最大データベース・サイズは、おおよそで、特定のプラットフォーム上のファイル数とファイル・サイズを乗算した値で、最大ディスク設定に依存する。 ファイルの最大数に影響するカーネル・パラメータについては、オペレーティング・システムのマニュアルを参照してください。
DB ファイルのサイズ	オペレーティング・システム・ファイルのサイズによって異なる。
DB 領域のサイズ	ロー・デバイス： 最大サイズは 4TB。 ファイル・システム・デバイス： 最大サイズは 4TB。 オペレーティング・システム・ファイル： オペレーティング・システムでサポートされる最大サイズ。 NAS (ネットワーク接続ストレージ) デバイス上での DB 領域の作成はおすすめしません。
フィールド・サイズ	BINARY の場合は 255 バイト、VARBINARY の場合は 32,767 バイト。 CHAR、VARCHAR の場合は 32,767 バイト。 LONG BINARY、LONG VARCHAR では、128KB ページの場合は最大 512TB、512KB ページの場合は 1PB。
IQ ページ・サイズ	64 ～ 512KB の間。
キーの最大サイズ	単一カラムのインデックスでは 255 バイト。複数カラムのインデックスでは 5300 バイト。

項目	制限事項
文字列リテラルの最大長	32KB
SQL 文の最大長	SQL 文の最大長は、IQ カタログに使用できるメモリ容量とカタログ・スタックのサイズまでに制限されます。 SQL 文が長すぎる場合は、**-gss** を使用してカタログ・スタックのサイズを大きくし、**-c**、または **-ch** と **-cl** の組み合わせを使用して、カタログ・メモリ・キャッシュの容量を増やしてください。 SQL 文をエラー・メッセージに出力する場合、テキストは IQ カタログ・ページのサイズに制限されます。**-gp** の設定を増やしてサーバを起動すると、長いコマンドを出力できますが、通常はデフォルトの **-gp** 4096 を使用してください。
可変長 FILLER カラムの最大長	512 バイト
DB ファイルの最大数	開けるファイルの総数は、オペレーティング・システムがサポートできるユニーク・ファイル記述子の数によって異なる。
(同時接続中の) ユーザの最大数	64 ビット・プラットフォーム (AIX、HP、Linux、Sun Solaris) では 1000。 Windows の 64 ビット・プラットフォームでは 200。
一時抽出ファイルの最大サイズ	`TEMP_EXTRACT_SIZEn` オプションで設定。プラットフォーム別の制限は以下のとおり。 AIX、HP-UX：0 ～ 64GB Sun Solaris：0 ～ 512GB Windows：0 ～ 128GB Linux：0 ～ 512GB
カラム数／テーブル	Sybase IQ では 1 つのテーブルで 45,000 までのカラムをサポートする。 テーブル内に 10,000 を超えるカラムがあると、パフォーマンスが低下する可能性がある。 NULL を許可するカラム数のテーブルごとの制限は、約 8*(database-page-size - 30)。
イベント数／データベース	2^31 – 1 = 2,147,483,647
ファイル数／データベース	ユーザが調整可能な (`NOFILE` を使うなど) OS による上限。 一般的には、データベースごとに 2047 ファイル。
インデックス数	2^32 (～ 4,000,000)／テーブル。
ロー数／テーブル	テーブル・サイズにより制限される。上限は 2^48 -1。

項目	制限事項
データベースごとのストアド・プロシージャの数	2^32 – 1 = 4,294,967,295
単独の **FROM** 句内のテーブルまたはビューの数	ジョイン・オプティマイザが有効になっている場合、クエリに応じて、16 〜 64 の間。
クエリが参照するテーブルまたはビューの数	512
テーブル数／データベース	4,293,918,719
ジョイン・インデックスごとのテーブル数 (1 つのクエリ・ブロックでジョイン可能なテーブル数)	31
参照されるテーブル数／トランザクション	制限なし。
クエリごとの UNION ブランチ数	512. 各ブランチの **FROM** 句に複数のテーブルがある場合、クエリごとのテーブル数の上限によって、**UNION** ブランチの許容数は少なくなる。
IN リスト内の値の数	250,000
ロー・サイズ	Sybase ではページ・サイズの半分を上限として推奨。
テーブル・サイズ	データベース・サイズにより制限。

参照：

- 文字列関数 (123 ページ)

システム・プロシージャ

システム情報を取得するには、Sybase IQ データベースでシステム提供のストアド・プロシージャを使用します。

Sybase IQ には、次の種類のシステム・プロシージャが用意されています。

- ストアド・プロシージャとして実装されるシステム関数。
- システム情報を表形式で表示する、カタログ・ストアド・プロシージャ。
- 上記 2 種類のプロシージャをマルチプレックス・サーバの操作向けにしたマルチプレックス・ストアド・プロシージャ。
  詳細については、『Sybase IQ Multiplex の使用』の「マルチプレックス・リファレンス」>「システム・プロシージャ」を参照してください。
- Transact-SQL システムとカタログ・プロシージャ。

sp_iqsetcompression や **sp_iqshowcompression** など、特にラージ・オブジェクト・データに関連付けられたシステム・ストアド・プロシージャについては、『Sybase IQ の非構造化データ分析の概要』の「ストアド・プロシージャのサポート」を参照してください。

ストアド・プロシージャの構文規則

Interactive SQL と同様にストアド・プロシージャ名を直接入力する場合と、**CALL** 文で呼び出す場合とでは、ストアド・プロシージャ・コールでのカッコと引用符の使い方が異なります。

この製品は Sybase IQ SQL と Transact-SQL のどちらの構文もサポートしているため、多少の変化形は許容されています。 Transact-SQL との互換性を維持する必要がある場合は、Transact-SQL の構文に従ってください。

表 99 : ストアド・プロシージャの構文のバリエーション

構文	構文タイプ	説明
`procedure_name ('param')`	Sybase IQ	パラメータをカッコで囲むときは引用符が必要
`procedure_name 'param'`	Sybase IQ	パラメータを引用符で囲むときは、カッコはオプション

構文	構文タイプ	説明
procedure_name param	Transact-SQL	パラメータ前後の引用符を省く場合は、カッコも省く必要があります。 **注意：** 所有者を指定する場合は、常にパラメータを引用符で囲む必要があります。たとえば、所有者が *dba* の場合、 *sp_iqtablesize 'dba.emp1'* はパラメータを引用符で囲む必要がありますが、 *sp_iqtablesize emp1* では不要です。
procedure_name	Sybase IQ または Transact-SQL	Interactive SQL でパラメータなしにプロシージャを直接実行する場合は、この構文を使用します。プロシージャにパラメータはありません。
`call` *procedure_name* `(`*param='value'*`)`	Sybase IQ	パラメータ値を渡すプロシージャを呼び出すときはこの構文を使用します。

Transact-SQL ストアド・プロシージャを使うときは、Transact-SQL 構文を使用してください。

ストアド・プロシージャが報告する統計情報を理解する

多くのストアド・プロシージャは、プロシージャの実行時に、Sybase IQ の状態に関する情報をレポートします。

つまり、スナップショットが得られます。たとえば、接続で使用されている領域を列挙するレポート・カラムは、プロシージャが実行された瞬間の領域だけを示し、接続で使用される最大領域を示すものではありません。

長期間にわたって Sybase IQ の利用状況を監視するには、Sybase IQ モニタを利用します。このモニタは、開始から終了まで、統計情報を収集し、指定した間隔で報告します。

システム・ストアド・プロシージャ

システム・プロシージャは、IQ メイン・ストアでのシステム管理者タスクを実行します。

システム・ストアド・プロシージャは、ユーザ ID dbo によって所有されています。

注意： デフォルトでは、Interactive SQL Classic が表示できるカラム値の最大長は 30 文字です。これでは、**sp_iqstatus** などのストアド・プロシージャの出力を表示するには不十分です。出力がトランケートされないようにするため、Interactive SQL メニューから [コマンド] - [オプション] を選択し、[表示カラムの制限値]、[出力カラムの制限値]、またはその両方の値を増やすことで最大長を大きくしてください。

sa_char_terms システム・プロシージャ

CHAR 文字列を単語に分割し、各単語をローとして、その位置とともに返します。

詳細については、『Sybase IQ の非構造化データ分析の概要』の「ストアド・プロシージャのサポート」>「TEXT インデックス内の単語の管理」>「sa_char_terms システム・プロシージャ」を参照してください。

sa_dependent_views システム・プロシージャ

指定したテーブルまたはビューのすべての従属ビュー・リストを返します。

構文

```
sa_dependent_views( 'tbl_name ' [, ' owner_name ] ' )
```

引数

- ***tbl_name*** **–** テーブルまたはビューの名前を指定する CHARACTER パラメータ。
- ***owner_name*** **–** *tbl_name* の所有者を指定する任意の CHARACTER パラメータ。

結果セット

カラム名	データ型	説明
table_id	UNSIGNED INTEGER	テーブルまたはビューのオブジェクト ID。
dep_view_id	UNSIGNED INTEGER	従属ビューのオブジェクト ID。

備考
このプロシージャを使用して、従属ビューの ID リストを取得します。または、ビュー名などビューに関する詳細情報を返す文のプロシージャを使用できます。

テーブル名と所有者名について指定された条件を満たす既存のテーブルがない場合、エラーは発生しません。また、次の点にも留意してください。

- *tbl_name* の指定は任意で、デフォルト値は NULL です。
- *owner* と *tbl_name* がどちらも NULL の場合は、従属ビューを持つすべてのテーブルに関する情報が返されます。
- *tbl_name* は NULL だが *owner* は指定されている場合は、指定された所有者が所有するすべてのテーブルに関する情報が返されます。
- *tbl_name* は指定されているが *owner* は NULL の場合は、指定された名前を持ついずれかのテーブルに関する情報が返されます。

デフォルトでは、このプロシージャの実行に必要なパーミッションはなく、PUBLIC でカタログにアクセスできることが想定されています。DBA は、必要に応じて、ビューやカタログに対するアクセスを制御できます。

パーミッション
なし。

関連する動作
なし。

例
次の例では、sa_dependent_views システム・プロシージャを使用して、SalesOrders テーブルに依存するビューの ID リストを取得します。このプロシージャは、SalesOrders の場合に table_id、従属ビュー ViewSalesOrders の場合には dep_view_id を返します。

```
CALL sa_dependent_views( 'SalesOrders' );
```

次の例では、sa_dependent_views システム・プロシージャを SELECT 文で使用して、SalesOrders テーブルに依存するビューの名前リストを取得します。このプロシージャは、ViewSalesOrders ビューを返します。

```
SELECT t.table_name FROM SYSTAB t,
sa_dependent_views( 'SalesOrders' ) v
WHERE t.table_id = v.dep_view_id;
```

sa_external_library_unload プロシージャ

外部ライブラリをアンロードします。

詳細については、『Sybase IQ の非構造化データ分析の概要』の「ストアド・プロシージャのサポート」>「外部ライブラリの確認」>「sa_external_library_unload システム・プロシージャ」を参照してください。

sa_list_external_library プロシージャ

現在サーバにロードされている外部ライブラリをリストします。

詳細については、『Sybase IQ の非構造化データ分析の概要』の「ストアド・プロシージャのサポート」>「外部ライブラリの確認」>「sa_list_external_library プロシージャ」を参照してください。

sa_nchar_terms システム・プロシージャ

NCHAR 文字列を単語に分割し、各単語をローとして、その位置とともに返します。

詳細については、『Sybase IQ の非構造化データ分析の概要』の「ストアド・プロシージャのサポート」>「TEXT インデックス内の単語の管理」>「sa_nchar_terms システム・プロシージャ」を参照してください。

sa_text_index_vocab システム・プロシージャ

TEXT インデックスに含まれるすべての単語と、各単語が含まれるインデックス値の合計数のリストを返します。

詳細については、『Sybase IQ の非構造化データ分析の概要』の「ストアド・プロシージャのサポート」>「TEXT インデックス内の単語の管理」>「sa_text_index_vocab システム・プロシージャ」を参照してください。

sa_get_user_status システム・プロシージャ

ユーザの現在のステータスを特定できます。

構文

```
sa_get_user_status( )
```

引数

なし。

結果セット

カラム名	データ型	説明
user_id	UNSIGNED INT	ユーザを識別するユニークな番号。
user_name	CHAR(128)	ユーザの名前。
接続	INT	このユーザによる現在の接続数。
failed_logins	UNSIGNED INT	ユーザがログインしようとして失敗した回数。
last_login_time	TIMESTAMP	ユーザが最後にログインした時刻。
locked	TINYINT	ユーザ・アカウントがロックされているかどうかを示すインジケータ。
reason_locked	LONG VARCHAR	アカウントがロックされた理由。

備考
このプロシージャは、ユーザの現在のステータスを示す結果セットを返します。基本的なユーザ情報に加えて、ユーザがロックアウトされているかどうかを示すカラムや、ロックアウトの理由が格納されたカラムが含まれています。ユーザは、ポリシーによるロック、パスワードの失効、または失敗した試行回数の過多などの理由によって、ロックアウトされる可能性があります。

DBA 権限のないユーザは、DBA が所有するカバー・プロシージャを作成および実行することによって、ユーザ情報を取得できます。

パーミッション
すべてのユーザに関する情報を表示するには DBA 権限が必要です。DBA 権限のないユーザが表示できるのは、自分自身の情報です。また、DBA 権限のないユーザは、DBA が所有するカバー・プロシージャを実行することによって、他のユーザに関する情報を表示できます。

関連する動作
なし。

例
次の例は、sa_get_user_status システム・プロシージャを使用して、データベース・ユーザのステータスを返します。

```
CALL sa_get_user_status;
```

sp_expireallpasswords プロシージャ

すべてのユーザ・パスワードをただちに期限切れにします。

構文 1

```
call sp_expireallpasswords
```

構文 2

```
sp_expireallpasswords
```

パーミッション
DBA 権限または USER ADMIN 権限が必要です。

例
すべてのユーザ・パスワードをただちに期限切れにします。

```
call sp_expireallpasswords
```

参照：

- sp_iqaddlogin プロシージャ (381 ページ)
- sp_iqcopyloginpolicy プロシージャ (415 ページ)
- sp_iqmodifyadmin プロシージャ (481 ページ)
- sp_iqmodifylogin プロシージャ (482 ページ)
- sp_iqpassword プロシージャ (489 ページ)

sp_iqaddlogin プロシージャ

新しい Sybase IQ ユーザ・アカウントを指定のログイン・ポリシーに追加します。

構文 1

```
call sp_iqaddlogin ('username_in', 'pwd' [,
'password_expiry_on_next_login'] [, 'policy_name'] )
```

構文 2

```
sp_iqaddlogin 'username_in', 'pwd' [, 'password_expiry_on_next_login']
[, 'policy_name']
```

構文 3

```
sp_iqaddlogin  username_in, pwd [, password_expiry_on_next_login] [,
policy_name]
```

使用法

表 100 : パラメータ

パラメータ	説明
username_in	ユーザのログイン名です。ログイン名は識別子の規則に従う必要があります。
pwd	ユーザのパスワード。パスワードは、パスワード規則に準拠する必要があります。つまり、有効な識別子である必要があります。
password_expiry_on_next_login	(オプション) ユーザのログインが作成されたらすぐに、ユーザのパスワードを失効させるかどうかを指定します。デフォルトの設定は OFF (パスワードに有効期限はない) です。
policy_name	(オプション) 指定のログイン・ポリシーの下にユーザを作成します。指定しないと、ルート・ログイン・ポリシーの下にユーザが作成されます。

sp_iqaddlogin を使って作成し、1 日で有効期限が切れるように設定した username_in/pwd は、翌日は終日有効であり、翌々日に無効になります。つまり、ログインを今日作成し、 *n* 日で有効期限が切れるように設定した場合、日付が *(n +1)* 日目になると使用できなくなります。

パーミッション
DBA 権限が必要です。

説明
新しい Sybase IQ ユーザ・アカウントを追加し、ログイン・ポリシーをユーザに割り当てて、ユーザを ISYSUSER システム・テーブルに追加します。 ユーザがすでにそのデータベースのユーザ ID を持っているが、ISYSUSER 内に登録されていない場合 (**GRANT CONNECT** 文または Sybase Central によってユーザ ID が追加された場合など) は、**sp_iqaddlogin** によってユーザがテーブルに追加されます。

Sybase IQ では、プロシージャを呼び出すときにログイン・ポリシー名を指定しないと、ユーザがルート・ログイン・ポリシーに割り当てられます。

注意： ログイン・ポリシーに対する最大ログイン数が無制限の場合、そのログイン・ポリシーに属するユーザが持つことができる接続は無制限になります。

最初のユーザ・ログインでは、パスワードの変更が強制され、ログイン・ポリシーが新しく作成されたユーザに割り当てられます。**CREATE USER** を使用して新しいユーザを作成することをおすすめしますが、下位互換性のために **sp_iqaddlogin** も引き続きサポートされています。

『リファレンス：文とオプション』の「SQL 文」>「CREATE USER 文」を参照してください。

例
この呼び出しでは、パスワード irk324 を持つユーザ rose が expired_password というログイン・ポリシーに追加されます。この例では、expired_password ログイン・ポリシーがすでに存在しているものとします。

```
call sp_iqaddlogin('rose', 'irk324', 'ON', 'expired_password')
```

```
sp_iqaddlogin 'rose','irk324', 'ON', 'expired_password'
```

参照：
- sp_expireallpasswords プロシージャ (381 ページ)
- sp_iqcopyloginpolicy プロシージャ (415 ページ)
- sp_iqmodifyadmin プロシージャ (481 ページ)
- sp_iqmodifylogin プロシージャ (482 ページ)
- sp_iqpassword プロシージャ (489 ページ)
- sp_iqdroplogin プロシージャ (436 ページ)

sp_iqbackupdetails プロシージャ

特定のバックアップに含まれるすべての dbfiles を表示します。

構文
sp_iqbackupdetails *backup_id*

パラメータ

表 101 : パラメータ

パラメータ	説明
backup_id	バックアップ操作のトランザクション識別子を指定します。

注意： SYSIQBACKUPHISTORY テーブルから backup_id 値を取得できます。次のクエリを実行します。select * from sysiqbackuphistory

パーミッション
DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

説明
sp_iqbackupdetails は次の値を返します。

表 102 : sp_iqbackupdetails のカラム

カラム名	説明
backup_id	バックアップ・トランザクションの識別子。
backup_time	バックアップの時間。
backup_type	バックアップの種類："Full"、"Incremental since incremental"、または "Incremental since full"。
selective_type	バックアップのサブタイプ："All inclusive"、"All RW files in RW dbspaces"、"Set of RO dbspace/file"。
depends_on_id	バックアップが依存する以前のバックアップの識別子。
dbspace_id	バックアップされる DB 領域の識別子。
dbspace_name	SYSIQBACKUPHISTORYDETAIL からの DB 領域の名前。DB 領域名が、指定の dbspace_id の SYSDBSPACE の DB 領域名と一致する場合。それ以外は "null"。
dbspace_rwstatus	"ReadWrite" または "Read Only"。
dbspace_createid	DB 領域作成トランザクション識別子。
dbspace_alterid	Alter DBSPACE 読み込み／書き込みモード・トランザクション識別子。
dbspace_online	ステータス。値は "Online" または "Offline"。
dbspace_size	バックアップ時の DB 領域のサイズ (キロバイト)。
dbspace_backup_size	DB 領域でのバックアップ時のデータのサイズ (キロバイト)。
dbfile_id	バックアップされる dbfile の識別子。
dbfile_name	バックアップ操作後に名前が変更されなかった場合は、論理ファイル名。変更された場合は "null"。
dbfile_rwstatus	"ReadWrite" または "Read Only"。
dbfile_createid	dbfile 作成トランザクション識別子。
dbfile_alterid	Alter DBSPACE alter FILE 読み込み／書き込みモード・トランザクション識別子。
dbfile_size in MB	dbfile のサイズ (キロバイト)。

カラム名	説明
dbfile_backup_size	dbfile バックアップのサイズ (キロバイト)。
dbfile_path	指定された dbspace_id および dbfile_id の SYSDBFILE の物理ファイル・パス ("file_name") と一致する場合は、SYSBACKUPDETAIL からの dbfile パス。それ以外は "null"。

例

sp_iqbackupdetails の出力例を次に示します。

```
backup_id   backup_time             backup_type   selective_type   d
epends_on_id
      883   2008-09-23 13:58:49.0   Full          All
inclusive                0

dbspace_id   dbspace_name   dbspace_rwstatus   dbspace_createid
         0   system         ReadWrite                         0

dbspace_alterid   dbspace_online dbspace_size dbspace_backup_size
dbfile_id
            0              0        2884              2884        0

dbfile_name dbfile_rwstatus dbfile_createid dbfile_alterid
dbfile_size
    system       ReadWrite               0             0      2884
```

```
dbfile_backup_size dbfile_path
             2884  C:¥¥Documents and Settings¥¥All Users¥¥SybaseIQ¥
¥demo¥¥iqdemo.db
```

参照：

- SYSIQBACKUPHISTORY システム・ビュー (633 ページ)

sp_iqbackupsummary プロシージャ

実行されたバックアップ操作の概要を示します。

構文

```
sp_iqbackupsummary [ timestamp or backup_id ]
```

パラメータ

表 103 : パラメータ

パラメータ	説明
timestamp または backup_id	バックアップ操作をレポートする間隔を指定します。タイムスタンプまたはバックアップ ID を指定した場合、指定した時間に等しいか、それより大きい backup_time を持つレコードのみが返されます。タイムスタンプを指定しない場合、ISYSIQBACKUPHISTORY のすべてのバックアップ・レコードが返されます。

パーミッション
DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

説明
sp_iqbackupsummary は次の値を返します。

表 104 : sp_iqbackupsummary のカラム

カラム名	説明
backup_id	バックアップ・トランザクションの識別子
backup_time	バックアップの時間
backup_type	バックアップの種類："Full"、"Incremental since incremental"、または "Incremental since full"。
selective_type	バックアップのサブタイプ："All Inclusive"、"All RW files in RW dbspaces"、"Set of RO dbspace/file"。
virtual_type	仮想バックアップの種類："Non-virtual"、"Decoupled"、または "Encapsulated"。
depends_on_id	バックアップが依存するバックアップの識別子
作成者	バックアップの作成者
backup_size	バックアップのサイズ (キロバイト)
user_comment	ユーザ・コメント
backup_command	発行された backup 文 (コメントなし)

例

sp_iqbackupsummary の出力例を次に示します。

```
backup_id   backup_time             backup_type   selective_type   v
irtual_type
     883   2008-09-23 13:58:49.0   Full          All inclusive    Non
virtual

depends_on_id   creator   backup_size   user_comment   backup_command
           0   DBA              10864              backup database to
                                                          'c:¥¥¥¥temp
¥¥¥¥b1'
```

sp_iqcardinality_analysis プロシージャ

テーブル内のカラムのカーディナリティを分析し、追加すべきインデックスを推奨することによって、インデックスの生成を支援します。

sp_iqcardinality_analysis では、インデックスの作成のためにすぐに実行できる SQL 文を含むオプションの SQL スクリプトを生成できます。

sp_iqcardinality_analysis はインデックス・アドバイザとは別に動作します。インデックス・アドバイザは実際のクエリでのカラムの使用に基づいて推奨を提供しますが、カーディナリティを考慮するのはカラムに LF または HG インデックスがあり、重複値を除くカウントが 1 バイト FP および 2 バイト FP インデックスの再構築を許可する場合のみです。**sp_iqcardinality_analysis** は、あらゆる場合においてカーディナリティを考慮します。ただし、このインデックス推奨リストにはインデックス・アドバイザの推奨は含まれません。

sp_iqcardinality_analysis の推奨リストには、HNG インデックスおよび CMP インデックスは含まれません。推奨されるインデックスには、次のものがあります。

- LF インデックス
- HG インデックス
- 1 バイト FP
- 2 バイト FP
- 3 バイト FP
- 2 バイト FP から 1 バイト FP への変換
- プライマリ・キー制約 (HG インデックス)
- 一意性制約 (HG インデックス)
- DATE、TIME、DTTM インデックス
- WD インデックス
- WORD インデックス

注意： ワイド・テーブルに対して **sp_iqcardinality_analysis** を実行すると、テーブルに含まれるカラムのデータ型とインデックスの複雑さに応じて、数分かかる場合があります。

構文

```
sp_iqcardinality_analysis [ 'table_name' ] [, 'table_owner' ] [, 'script' ]
```

パラメータ

パラメータ	説明
table_name	テーブルの名前。
table_owner	テーブル所有者名。このパラメータが指定されない場合、プロシージャは現在のユーザが所有するテーブルを検索します。
スクリプト	インデックスの作成のためにすぐに実行できる SQL 文を含む SQL スクリプトを生成し、インデックスの推奨を表示します。オプションを指定しない場合は、コンソールに次の情報が表示されます。 • table_name • table_owner • column_name • cardinality • index_type • インデックスの推奨

使用法

パラメータをまったく指定しない場合、IQ は現在のユーザが所有するすべてのテーブルのすべてのカラムについて **create_index** SQL 文を表示します。

script を指定すると、次のように出力をリダイレクトしてスクリプト・ファイルを生成できます。

```
OUTPUT TO ‘indexfile.sql’ FORMAT ASCII QUOTE '';
```

パーミッション

DBA 権限が必要です。

例 1

```
sp_iqcardinality_analysis 'onebytefp', 'DBA'
```

表 105 : 例 1 コンソール出力

table_name	table_owner	column_name	cardinality	インデックス・タイプ	インデックスの推奨
onebytefp	DBA	c1	10	1 バイト FP	-- カラム 'c1' には 1 バイト FP インデックスがなく、カーディナリティは 256 未満です。-- 1 バイト FP インデックスを作成できます。-- 次のストアド・プロシージャを呼び出してください。 sp_iqrebuildindex 'onebytefp','column c1 255'
onebytefp	DBA	c1	10	LF	-- カラム 'c1' には LF インデックスがなく、カーディナリティは 1000 未満です。-- 次の CREATE INDEX 文を使用して LF インデックスを作成できます。CREATE LF INEDX IQ_T400_c1_LF ON DBA.onebytefp (c1)

例 2

```
sp_iqcardinality_analysis 'onebytefp', 'DBA', 'script'
```

表 106 : 例 2 コンソール出力

インデックスの推奨
-- カラム 'c1' には 1 バイト FP インデックスがなく、カーディナリティは 256 未満です。-- 1 バイト FP インデックスを作成できます。-- 次のストアド・プロシージャを呼び出してください。sp_iqrebuildindex 'onebytefp','column c1 255' -- カラム 'c1' には LF インデックスがなく、カーディナリティは 1000 未満です。-- 次の CREATE INDEX 文を使用して LF インデックスを作成できます。CREATE LF INDEX IQ_T400_c1_LF ON onebytefp (c1)

sp_iqcheckdb プロシージャ

現在のデータベースの妥当性を確認します。オプションで、DB 領域またはデータベースの割り付けの問題を解決します。

sp_iqcheckdb はデータベース内のすべての記憶領域を読み込みます。成功すると、データベースのフリー・リスト (内部割り付けマップ) が更新され、データベースの実際の記憶領域割り当てがリストに反映されます。**sp_iqcheckdb** は次に、実行した作業をリストしたレポートを生成します。

エラーが検出されると、**sp_iqcheckdb** がエラーのタイプとオブジェクト名をレポートします。エラーが検出された場合、**sp_iqcheckdb** はフリー・リストを更新しません。

sp_iqcheckdb でも、指定されたテーブル、インデックス、インデックス・タイプ、またはデータベース全体の一貫性を検査できます。

注意： **sp_iqcheckdb** は、IQ データベース一貫性チェッカ (DBCC) のユーザ・インタフェースで、**DBCC** と呼ばれる場合もあります。

構文

```
sp_iqcheckdb 'mode  target [ … ] [ resources resource-percent ]'
```

これは **sp_iqcheckdb** の一般的な構文です。データベースの一貫性をチェックするモードは 3 種類あり、アロケーション・マップをリセットするモードは 1 つです。 以下に、それぞれのモードの構文を示します。 パラメータ文字列でモードとターゲットの両方を指定しないと、Sybase IQ により次のエラー・メッセージが返されます。

```
At least one mode and target must be specified to DBCC.
```

パラメータ

mode: { **allocation** | **check** | **verify** } | **dropleaks**

target: [**indextype***index-type* […]] **database** | **database resetclocks** | { [**indextype***index-type*] […] **table** *table-name* [**partition** *partition-name*] […] | **index** *index-name* | […] **dbspace** *dbspace-name*}

allocation モード

```
sp_iqcheckdb 'allocation target [ resources resource-percent ]'
```

check モード

```
sp_iqcheckdb 'check target [ resources resource-percent ]'
```

verify モード

```
sp_iqcheckdb 'verify target [ resources resource-percent ]'
```

dropleaks モード

```
sp_iqcheckdb 'dropleaks target [ resources resource-percent ]'
```

使用法

表 107 : パラメータ

パラメータ	説明
データベース	ターゲットがデータベースの場合、すべての DB 領域がオンラインである必要があります。
index-type	次のいずれかのインデックス・タイプ： **FP**、**CMP**、**LF**、**HG**、**HNG**、**WD**、**DATE**、**TIME**、**DTTM**、**TEXT**。 指定した *index-type* 存在しない場合は、エラー・メッセージが返されます。複数のインデックス・タイプが指定されていて、ターゲットにこれらのインデックス・タイプの一部のみが含まれる場合、**sp_iqcheckdb** では存在するインデックス・タイプが処理されます。
index-name	所有者とテーブル識別子を含めることができます。 `[[owner.]table-name.]index-name` *owner* を指定しない場合は、現在のユーザとデータベース所有者 (dbo) がこの順序で代わりに使用されます。*table* を指定しない場合、*index-name* はユニークである必要があります。

パラメータ	説明
table-name	所有者の識別子を含めることができます：`[owner.]table-name` *owner* を指定しない場合は、現在のユーザとデータベース所有者 (dbo) がこの順番で代わりに使用されます。*table-name* にテンポラリ・テーブルやプリジョイン・テーブルを指定することはできません。 **注意：** テーブル名またはインデックス名にスペースが含まれる場合は、*table-name* または *index-name* パラメータを次のように二重引用符で囲みます。 `sp_iqcheckdb 'check index "dbo.sstab.i2" resources 75'` *owner* を指定しない場合は、現在のユーザとデータベース所有者 (dbo) がこの順番で代わりに使用されます。*table-name* にテンポラリ・テーブルやプリジョイン・テーブルを指定することはできません。 **注意：** テーブル名またはインデックス名にスペースが含まれる場合は、*table-name* または *index-name* パラメータを次のように二重引用符で囲みます。 `sp_iqcheckdb 'check index "dbo.sstab.i2" resources 75'`
partition-name	*partition-name* パラメータには、修飾子は含まれません。スペースが含まれる場合は、二重引用符で囲みます。 パーティション・フィルタを使用すると、**sp_iqcheckdb** は、そのパーティションに属する対応テーブルのローのサブセットを調べます。テーブルのパーティション・フィルタと、パーティション・フィルタのないテーブル・ターゲットは、テーブルに 1 つのパーティションのみがある場合、意味的に同じことになります。
dbspace-name	*dbspace-name* パラメータには、修飾子は含まれません。スペースが含まれる場合は、二重引用符で囲みます。 DB 領域ターゲットは、DB 領域に属するデータベースのページのサブセットを調べます。DB 領域はオンラインである必要があります。DB 領域とデータベース・ターゲットは、テーブルに 1 つの DB 領域のみがある場合、意味的に同じことになります。

パラメータ	説明
resource-percent	入力パラメータ *resource-percent* は 0 より大きい整数である必要があります。リソースのパーセンテージを利用して、CPU の数に応じてスレッドの数を制御すれば、データベース一貫性チェッカの CPU の使用率を制限できます。*resource-percent* = 100 (デフォルト値) の場合、1 つの CPU に 1 つのスレッドが作成されます。*resource-percent* が 100 よりも大きい場合、CPU の数以上のスレッドがあり、一部のマシン構成ではパフォーマンスが向上することがあります。スレッドの最小数は 1 です。

注意： **sp_iqcheckdb** のパラメータ文字列は、一重引用符で囲みます。また、文字列の長さが 255 バイトを超えることはできません。

割り付けの問題は dropleaks モードで修正できます。

パーミッション
DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

説明
sp_iqcheckdb は、データベース内の各ブロックの割り付けをチェックし、次の **sp_iqdbstatistics** プロシージャが実行されるまで現在のセッション内の情報を保存します。**sp_iqdbstatistics** は、最近実行された **sp_iqcheckdb** の最新の結果を表示します。

sp_iqcheckdb は、指定されたパラメータに応じて、いくつかの関数を実行できます。

allocation モード
データベース全体、特定のインデックス、特定のインデックス・タイプ、特定のパーティション、特定のテーブル、または特定の DB 領域のブロックマップ情報で割り付けをチェックします。インデックスの一貫性は検査しません。

重複ブロック (複数のオブジェクトが所有権を主張するブロック) または余分なブロック (オブジェクトが所有する割り付けられていないブロック) を検出します。

データベースまたは DB 領域ターゲットのリーク・ブロック (指定されたターゲットのオブジェクトによって要求されていない割り付けブロック) を検出します。

ターゲットが分割されたテーブルの場合、**allocation mode** では次のことが行われます。

- すべてのテーブルのパーティション割り付けビットマップのメタデータのチェック
- テーブル割り付けビットマップのメタデータのチェック
- ブロックマップ・エントリがテーブルの割り付けビットマップと一致していることの確認
- テーブルのパーティション割り付けビットマップが重複していないことの確認
- テーブルのパーティション割り付けビットマップで定義されたローが、テーブルの存在ビットマップのスーパーセットを構成していることのチェック
- テーブルのパーティション割り付けビットマップで定義されたローが、テーブルの割り付けビットマップのスーパーセットを構成していることのチェック

注意： **sp_iqcheckdb** の入力パラメータ文字列で、単一のインデックス、インデックス・タイプ、またはテーブルの名前を指定すると、すべての割り付けの問題をチェックできません。

次の場合に allocation モードで実行します。

- 重複ブロックや未所有ブロックを検出する (データベースまたは特定のテーブルやインデックスをターゲットとして使用する)。
- ページ・ヘッダ・エラーが発生した場合

DBCC オプション **resetclocks** は、allocation モードでのみ使用されます。**resetclocks** は、強制リカバリとともに使用し、マルチプレックス・セカンダリ・サーバをコーディネータに変換します。 マルチプレックス機能については、『Sybase IQ Multiplex の使用』を参照してください。**resetclocks** は、内部データベース・バージョン管理クロックが遅れている場合に、クロックの値を修正します。 Sybase IQ サポート・センタに連絡した場合を除き、**resetclocks** オプションはその他の目的には使用しません。

resetclocks オプションは、シングル・ユーザ・モードで実行する必要があり、DBCC 文 **allocation database** でのみ使用できます。**resetclocks** の構文は、次のとおりです。

```
sp_iqcheckdb 'allocation database resetclocks'
```

check モード

すべてのデータベース・ページが、データベース全体、特定のインデックス、特定のインデックス・タイプ、特定のテーブル、特定のパーティション、または特定の DB 領域について読み込めることを確認します。テーブルがパーティションに分割されている場合、check モードはテーブルのパーティション割り付けビットマップをチェックします。

クエリを実行したときに、メタデータ、NULL カウント、または重複した値を除くカウントのエラーが返された場合、check モードで実行します。

verify モード
データベース全体、特定のインデックス、特定のインデックス・タイプ、特定のテーブル、特定のパーティション、または特定の DB 領域について、非 FP インデックスの内容を、対応する FP インデックスで確認します。FP および対応する非 FP インデックスのすべてのデータ・ページが指定のターゲットに含まれている場合、verify モードで次の不整合が検出されます。

- 欠落キー ― FP インデックスに存在するが、非 FP インデックスには存在しないキー。
- 余分なキー ― 非 FP インデックスに存在するが、FP インデックスには存在しないキー。
- 欠落ロー ― FP インデックスに存在するが、非 FP インデックスには存在しないロー。
- 余分なロー ― 非 FP インデックスに存在するが、FP インデックスには存在しないロー。

指定のターゲットに FP ページのサブセットのみが含まれる場合、verify モードでは次の不整合のみが検出できます。

- 欠落キー
- 欠落ロー

ターゲットが分割されたテーブルの場合、verify モードでは、テーブルまたはテーブル・パーティション内の各ローが正確なパーティションに割り当てられていることも確認します。

クエリを実行したときに、メタデータ、NULL カウント、または重複した値を除くカウントのエラーが返された場合、verify モードで実行します。

注意： sp_iqcheckdb は、参照整合性の検査や、参照整合性違反の修復は行いません。

dropleaks モード
Sybase IQ サーバがシングルノード・モードで実行されている場合は、dropleak モードをデータベースまたは DB 領域ターゲットで使用して、データベース全体または指定の DB 領域ターゲットの割り付けマップをリセットできます。 ターゲットが DB 領域の場合の dropleaks 操作は、その DB 領域での読み取り／書き込み操作も防ぐ必要があります。 指定されたデータベースまたは DB 領域内のすべての DB 領域はオンラインにしてください。

マルチプレックスで dropleaks モードを実行する方法については、『Sybase IQ Multiplex の使用』を参照してください。

sp_iqcheckdb プロシージャは、次の例のように使用します。

例 1
次の例では、**sp_iqcheckdb** はデータベース全体の割り付けをチェックします。

```
sp_iqcheckdb 'allocation database'
```

例 2
2 番目の例では、**sp_iqcheckdb** がインデックス i1、i2、dbo.t1.i3 に対して詳細なチェックを行います。新しいモードを指定しない場合、次のコマンドで示すように、**sp_iqcheckdb** は残りのターゲットにも同じモードを適用します。

```
sp_iqcheckdb 'verify index i1 index i2 index dbo.t1.i3'
```

例 3
すべてのモードを組み合わせて、1 つのセッションで複数のチェックをデータベースに対して行うことができます。次の例では、**sp_iqcheckdb** は、CPU の半分を使って、テーブル t2 のパーティション p1 に対しては簡単なチェックを、インデックス i1 に対しては詳細なチェックを、データベース全体に対しては割り付けチェックを行います。

```
sp_iqcheckdb 'check table t2 partition p1 verify index i1 allocation
database resources 50'
```

例 4
次の例では、データベース内の **FP** タイプのインデックスをすべてチェックします。

```
sp_iqcheckdb 'check indextype FP database'
```

例 5
次の例では、テーブル t1 内の **FP** および **HG** インデックスと、テーブル t2 内の **LF** インデックスを検証します。

```
sp_iqcheckdb 'verify indextype FP indextype HG table t1 indextype LF
table t2'
```

例 6
次の例は、**sp_iqcheckdb** の出力にある 3 つの "LVC cells" メッセージの 1 つを示します。

```
sp_iqcheckdb 'check index EFG2JKL.ASIQ_IDX_T208_C504_FP'
-----------------------------------
Index Statistics:
** Inconsistent Index: abcd.EFG2JKL.ASIQ_IDX_T208_C504_FP ****** FP
Indexes Checked: 1
** Unowned LVC Cells: 212 ******
```

sp_iqcheckdb LVC セル・メッセージには、次の項目が含まれています。

- Unowned LVC cells
- 重複した LVC セル・ロー
- 割り付けられていない LVC セル・ロー

これらのメッセージは、VARCHAR カラム、VARBINARY カラム、LONG BINARY (BLOB) カラム、または LONG VARCHAR (CLOB) カラムに矛盾があることを示しています。未所有の LVC セルは、小容量の使用不可ディスク領域を表し、無視しても問題ありません。重複した LVC セルと未割り付けの LVC セルは、ダメージを受けたカラムを削除しないと解決できない重大なエラーです。

ダメージを受けたカラムを削除するには、古いカラムのコピーから新しいカラムを作成した後で、元のカラムを削除し、新しいカラムの名前を古いカラムの名前に変更します。

注意： LVC は、幅が 255 より大きい VARCHAR カラムまたは VARBINARY カラムです。LVC は LONG BINARY (BLOB) と LONG VARCHAR (CLOB) によっても使用されます。

DBCC のパフォーマンス

DBCC の実行時間は、データベース検査全体のデータベースのサイズ、指定するテーブルやインデックスの数、マシンのサイズによって異なります。データベースの一部、つまり指定したテーブル、インデックス、またはインデックス・タイプだけをチェックする場合は、データベース全体を検査する場合より時間がかかりません。

sp_iqcheckdb dropleaks モードの処理時間は、DB 領域ターゲットの数によって異なります。

このテーブルは、4 つの **sp_iqcheckdb** モードの動作と出力内容を示します。

表 108 : sp_iqcheckdb の各モードの動作と出力

モード	検出されるエラー	出力	処理速度
割り付け	割り付けのエラー	割り付けの統計情報のみ	1 時間あたり 4TB
check	割り付けのエラー 大部分のインデックス・エラー	表示可能なすべての統計情報	1 時間あたり 60GB
verify	割り付けのエラー すべてのインデックス・エラー	表示可能なすべての統計情報	1 時間あたり 15GB

モード	検出されるエラー	出力	処理速度
dropleaks	割り付けのエラー	割り付けの統計情報のみ	1 時間あたり 4TB

出力
実行モードによっては、**sp_iqcheckdb** 出力に、結果の要約、エラー、統計情報、修復の統計が含まれます。1 つのセッションで複数のモードを指定した場合、出力には最高で 3 つの結果セットが含まれます。エラーの統計情報は、エラーが検出された場合のみ、アスタリスク (*****) で表示されます。

sp_iqcheckdb の出力は、Sybase IQ メッセージ・ファイル `.iqmsg` にもコピーされます。**DBCC_LOG_PROGRESS** オプションが ON の場合、**sp_iqcheckdb** は進行メッセージをこの IQ メッセージ・ファイルに送信します。これにより、ユーザは DBCC の実行状況を把握できます。

出力例
sp_iqcheckdb 'allocation database' を実行し、リークされた領域が見つかった場合に表示される出力の例を次に示します。リークされた領域とは、データベースのフリー・リスト (内部割り付けマップ) に従って割り付けられたが、どのデータベース・オブジェクトの一部でもないと DBCC が判断したブロックのことです。この例では、DBCC は 32 のリーク・ブロックをレポートしています。

```
                    Stat                          Value              Flags
=====================================|============================|
=====
DBCC Allocation Mode Report
=====================================|============================|
=====
** DBCC Status                       |Errors Detected             |*****
=====================================|============================|
=====
```

```
Allocation Summary                   |                            |
=====================================|============================|
=====
   Blocks Total                      |8192                        |
   Blocks in Current Version         |4954                        |
   Blocks in All Versions            |4954                        |
   Blocks in Use                     |4986                        |
   % Blocks in Use                   |60                          |
** Blocks Leaked                    |32                          |*****
                                     |                            |
=====================================|============================|
=====
```

```
Allocation Statistics                |                            |
=====================================|============================|
=====
```

```
   Marked Logical Blocks              |8064                          |
   Marked Physical Blocks             |4954                          |
   Marked Pages                       |504                           |
   Blocks in Freelist                 |126553                        |
   Imaginary Blocks                   |121567                        |
   Highest PBN in Use                 |5432                          |
** 1st Unowned PBN                   |452                         |*****
   Total Free Blocks                  |3206                          |
   Usable Free Blocks                 |3125                          |
   % Free Space Fragmented            |2                             |
   Max Blocks Per Page                |16                            |
   1  Block Page Count                |97                            |
   3  Block Page Count                |153                           |
   4  Block Page Count                |14                            |
   ...
   9  Block Hole Count                |2                             |
   16 Block Hole Count                |194                           |
                                      |                              |
   Database Objects Checked           |1                             |
   B-Array Count                      |1                             |
   Blockmap Identity Count            |1                             |
====================================|============================|
=====Connection Statistics               |
                           |
```

sp_iqcheckoptions プロシージャ

接続されているユーザについては、**sp_iqcheckoptions** は、現在の値とデータベースのデフォルト値、そして、デフォルト値から変更されたサーバ起動オプションを表示します。

構文

```
sp_iqcheckoptions
```

パーミッション

なし。DBA には、すべてのグループとユーザに対して永続的に設定されたすべてのオプションと、DBA に設定された temporary オプションが表示されます。DBA 以外のユーザには、そのユーザ自身の temporary オプションが表示されます。デフォルトでないサーバ起動オプションはすべてのユーザに表示されます。

使用法

パラメータは不要です。デフォルト値から変更された各オプションに対して 1 つのローを返します。出力はオプション名、ユーザ名の順でソートされます。

説明

接続されているユーザについては、デフォルト値から変更されたデータベースおよびサーバの起動オプションに関し、**sp_iqcheckoptions** ストアド・プロシージャが現在値とデフォルト値のリストを表示します。**sp_iqcheckoptions** は、Sybase IQ

と SQL Anywhere のすべてのデータベース・オプションを考慮します。Sybase IQ は一部の SQL Anywhere オプションのデフォルト値を変更し、これらの変更後の値が新しいデフォルト値になります。 新しい Sybase IQ のデフォルト値が再度変更されないかぎり、**sp_iqcheckoptions** はこのオプションを一覧に表示しません。

sp_iqcheckoptions を実行すると、DBA にはすべてのグループとユーザに永続的に設定されているすべてのオプションと、DBA に設定されているテンポラリ・オプションが表示されます。DBA 以外のユーザには、そのユーザ自身の temporary オプションが表示されます。デフォルトでないサーバ起動オプションはすべてのユーザに表示されます。

表 109 : sp_iqcheckoptions カラム

カラム名	説明
User_name	オプションが設定されたユーザまたはグループの名前。データベース作成時は、すべてのオプションが `public` グループに設定されます。 `public` 以外のグループまたはユーザに設定されたオプションが表示されます。
Option_name	オプション名。
Current_value	オプションの現在の値。
Default_value	オプションのデフォルト値。
Option_type	TEMPORARY オプションの場合は "Temporary"、その他の場合は "Permanent"。

例
次の例では、テンポラリ・オプション `APPEND_LOAD` が ON に設定され、グループ `mygroup` のオプション `MAX_WARNINGS` が 9 に設定されています。ユーザ `joel` では、`MAX_WARNINGS` にテンポラリ値として 55 が設定されています。

最初の例では、**sp_iqcheckoptions** が DBA によって実行されています。

```
User_name Option_name          Current_value   Default_value   Optio
n_type
DBA       Ansi_update_constr   CURSORS         Off             Permanent
PUBLIC    Ansi_update_constr   Cursors         Off             Permanent
DBA       Append_Load          ON              OFF             Temporary
DBA       Checkpoint_time      20              60              Temporary
DBA       Connection_authent   Company=MyComp;                 Temporary
                                 Application=DBTools;Signa
DBA       Login_procedure      DBA.sp_iq_proce sp_login_envir  Perma
nent
PUBLIC    Login_procedure      DBA.sp_iq_proce sp_login_envir  Perma
nent
mygroup   Max_Warnings         9               281474976710655 Permanent
DBA       Thread_count         25              0               Temporary
```

2 番目の例では、**sp_iqcheckoptions** がユーザ joel によって実行されています。

```
User_name Option_name          Current_value   Default_value   Optio
n_type
joel     Ansi_update_constr   CURSORS        Off           Permanent
PUBLIC    Ansi_update_constr   Cursors        Off           Permanent
joel     Checkpoint_time      20             60            Temporary
joel     Connection_authent   Company=MyComp;               Temporary
                                Application=DBTools;Signa
joel      Login_procedure      DBA.sp_iq_proce sp_login_envir  Perma
nent
PUBLIC    Login_procedure      DBA.sp_iq_proce sp_login_envir  Perma
nent
joel     Max_Warnings        55             281474976710655 Temporary
joel      Thread_count         25             0               Temporary
```

sp_iqclient_lookup プロシージャ

特定のクライアント IP アドレス／ポートを起点とするネットワーク・アナライザと同様に、特定のデータ・ストリームを担当する Sybase IQ ユーザ・アカウントを特定するために、クライアント・アプリケーションによって使用されます。

構文

sp_iqclient_lookup ['*IPaddress*'] [, *Port*] [, *UserID*]

パラメータ

表 110 : パラメータ

パラメータ	説明
IPaddress	起点となるクライアント・アプリケーションの IP アドレスを指定します。
ポート	起点となるクライアント・アプリケーションのポート番号を指定します。
UserID	Sybase IQ ユーザの ID を指定します。

パーミッション

DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

説明

sp_iqclient_lookup プロシージャは、クライアントの IP アドレスとポート番号を受け取って、Number (接続 ID)、IPaddress、Port、UserID を含む、単一のローを返します。

```
1> sp_iqclient_lookup '158.76.235.71',3360
2> go

Number   IPaddress      Port    UserID
```

```
------   ---------      ----    ------
15       158.76.235.71  3360    rdeniro
```

オプションで、3 番目の引数を渡して、UserID のみを選択することもできます。引数が渡されなかった場合、sp_iqclient_lookup は、現在のすべてのログインを IP アドレスとポート番号とともに返します。次に例を示します。

```
sp_iqclient_lookup
```

```
Number   IPaddress        Port    UserID
------   ---------        ----    ------
11       162.66.131.36    2082    mbrando
21       162.66.100.233   1863    apacino
22       162.66.100.206   8080    jcaan
23       162.66.100.119   6901    rduvall
24       162.66.100.125   7001    dkeaton
25       162.66.100.124   6347    jcazale

(6 rows affected)
(return status = 0)
```

クライアント・アプリケーションが内部接続に TCP/IP を使用していない場合、アドレスは 127.0.0.1 と表示されます。

注意： これらの情報にアクセスできるのは、ログオンしているユーザのみです。履歴ログイン・データがこの目的でサーバ上に維持されることはありません。

関連する動作

sp_iqclient_lookup ストアド・プロシージャは、インストールごとに異なるサーバのパフォーマンスに影響を与える可能性があります。ログイン名を検索するには、サーバ上で現在アクティブになっている全接続をスキャンする必要があります。そのため、接続数の多いサーバほど、パフォーマンスに与える影響も大きくなります。また、これらの情報は動的 (場合によっては高度に動的) であるため、キャッシュできません。したがって、サーバ機能を使用する他のクライアント・アプリケーションと同様に、このストアド・プロシージャの使用を管理することと、サーバに対する影響をモニタすることは、ローカル・システムの管理者の責任となります。

例

UserID jcazale の IP アドレスを表示します。

```
sp_iqclient_lookup null, null, jcazale
```

```
Number   IPaddress        Port    UserID
------   ----------       ----    ------

11       162.66.131.36    2082    jcazale
15       164.66.131.36    1078    jcazale
```

クライアント IP 162.66.131.36 の IP アドレスを表示します。

```
sp_iqclient_lookup '162.66.131.36'
```

```
Number   IPaddress         Port    UserID
------   ----------        ----    ------
11       162.66.131.36     2082    jcazale
12       162.66.131.36     1078    jcaan
```

注意： ユーザの指定した引数が正しくない場合、結果は空になります。

sp_iqcolumn プロシージャ

データベース内のカラムに関する情報を表示します。

構文 1

```
sp_iqcolumn [ table_name ] [, table_owner ] [, table_loc]
```

構文 2

```
sp_iqcolumn [ table_name='table_name' ],[ table_owner='tableowner' ],
[table_loc='table_loc']
```

使用法

構文	説明
構文 1	*table_name* を指定しないで *table_owner* を指定する場合は、*table_name* を NULL に置き換える必要があります。たとえば、`sp_iqcolumn NULL,DBA` のように記述します。
構文 2	パラメータはどのような順番で指定しても構いません。*'table_name'* と *'table_owner'* を一重引用符で囲みます。

パーミッション

DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、`EXECUTE` パーミッションが付与される必要があります。

説明

データベース内のカラムに関する情報を表示します。*table_name* パラメータを指定すると、その名前のテーブル内のカラムだけが返されます。table_owner パラメータを指定すると、そのユーザが所有するテーブルだけが返されます。*table_name* パラメータと table_owner パラメータを両方とも指定すると、ユニークなテーブル (存在する場合) からカラムが選択されます。table_loc を指定すると、そのセグメント・タイプで定義されたテーブルだけが返されます。パラメータを指定しないと、データベース内のすべてのテーブルのすべてのカラムが返されます。**sp_iqcolumn** は、システム・テーブルのカラム情報は返しません。

表 111 : sp_iqcolumn のカラム

カラム名	説明
table_name	テーブルの名前。
table_owner	テーブルの所有者
column_name	カラム名。
domain_name	データ型。
width	精度と位取りを持つ数値データ型の精度、または位取りを持たない数値データ型の格納領域の幅、または文字データ型の幅。
scale	数値データ型の位取り。
nulls	カラムが NULL を許容する場合は 'Y'、許容しない場合は 'N'。
default	カラムが identity/autoincrement カラムである場合は 'Identity/Autoincrement'、そうでない場合は null。
cardinality	重複する値を除いたインデックスによるカウント (判明している場合)。
est_cardinality	重複した値を除いた値の予測数。`MINIMIZE_STORAGE` オプションを ON にしてカラムを作成した場合は自動的に 255 に設定されます。または、**CREATE TABLE** に指定された IQ UNIQUE 制約内のユーザが指定した値です。
location	TEMP = IQ テンポラリ・ストア、MAIN = IQ メイン・ストア、SYSTEM = カタログ・ストア。
isPartitioned	カラムが、分割されたテーブルに属しており、かつテーブル・パーティションの DB 領域と異なる DB 領域を持つ 1 つ以上のパーティションを持っている場合は 'Y'。カラムのテーブルが分割されていないか、またはカラムの各パーティションがテーブル・パーティションと同じ DB 領域に存在する場合は 'N'。
remarks	**COMMENT** 文で追加されたユーザ・コメント
check	検査制約の式

参照：

- sp_iqconstraint プロシージャ (410 ページ)
- sp_iqdatatype プロシージャ (419 ページ)
- sp_iqevent プロシージャ (443 ページ)
- sp_iqhelp プロシージャ (448 ページ)
- sp_iqindex および sp_iqindex_alt プロシージャ (456 ページ)

- sp_iqjoinindex プロシージャ (471 ページ)
- sp_iqpkeys プロシージャ (491 ページ)
- sp_iqprocparm プロシージャ (496 ページ)
- sp_iq_reset_identity プロシージャ (505 ページ)
- sp_iqtable プロシージャ (525 ページ)
- sp_iqview プロシージャ (541 ページ)

sp_iqcolumn プロシージャの例

sp_iqcolumn を使用する場合は、この例を参照してください。

次の構文は両方とも、テーブル Departments のすべてのカラムを返します。

```
sp_iqcolumn Departments
```

```
call sp_iqcolumn (table_name='Departments')
```

```
table_name   table_owner  column_name     domain_name  width  scale
  nulls  default
Departments  GROUPO       DepartmentID    integer       4        0
  N      (NULL)
Departments  GROUPO       DepartmentName  char         40        0
  N      (NULL)
Departments  GROUPO       DepartmentHead  integer       4        0
  Y      (NULL)
cardinality  est_cardinality  location  isPartitioned  remarks  check
5            5                Main      N             (NULL)   (NULL)
0            5                Main      N             (NULL)   (NULL)
5            5                Main      N             (NULL)   (NULL)
```

次の構文は、DBA が所有するすべてのテーブルのすべてのカラムを返します。

```
sp_iqcolumn table_owner='DBA'
```

sp_iqcolumnuse プロシージャ

負荷によってアクセスされるカラムの使用状況の情報を詳細にレポートします。

構文

```
sp_iqcolumnuse
```

パーミッション

DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

説明

SYSTEM で作成されたテーブルのカラムはレポートされません。

表 112 : sp_iqcolumnuse のカラム

カラム名	説明
TableName	テーブル名
ColumnName	カラム名
所有者	カラム所有者のユーザ名
UID**	カラムのユニークな識別子
LastDT	前回のアクセスの日時
NRef	クエリ参照の数

**UID はシステムが割り当てた番号であり、カラムのインスタンスをユニークに識別します (インスタンスはオブジェクト作成時に定義されます)。

カラム・インデックスの詳細については、『リファレンス： 文とオプション』の「データベース・オプション」>「アルファベット順のオプション・リスト」>「INDEX_ADVISOR オプション」を参照してください。

例

sp_iqcolumnuse プロシージャからの出力例を示します。

```
TableName       ColumnName     Owner   UID              LastDT
         NRef
orders          o_orderdate    DBA               151   20070917
22:41:22..  13
orders          o_shippriority DBA               154   20070917
22:41:22..  13
lineitem        l_orderkey     DBA               186   20070917
22:41:22..  13
lineitem        l_extendedp..  DBA               191   20070917
22:41:22..  13
lineitem        l_discount     DBA               192   20070917
22:41:22..  13
lineitem        l_shipdate     DBA               196   20070917
22:41:22..  13
#tmp1           expression     DBA     10000000001218   20070917
22:57:36..   1
#tmp1           expression     DBA     10000000001222   20070917
22:41:58..   1
...
```

注意： 前述の例での長い番号は、テンポラリ ID です。

参照：

- sp_iqindexadvice プロシージャ (460 ページ)
- sp_iqindexuse プロシージャ (469 ページ)

- sp_iqtableuse プロシージャ (530 ページ)
- sp_iqunusedcolumn プロシージャ (536 ページ)
- sp_iqunusedindex プロシージャ (537 ページ)
- sp_iqunusedtable プロシージャ (538 ページ)
- sp_iqworkmon プロシージャ (547 ページ)

sp_iqconnection プロシージャ

接続およびバージョンについての情報を表示します。この情報には、テンポラリ DB 領域を使用しているユーザ、バージョンを有効にしているユーザ、各接続が Sybase IQ 内で行っている作業、接続ステータス、データベース・バージョン・ステータスなどが含まれます。

構文

```
sp_iqconnection [ connhandle ]
```

使用法

入力パラメータ *connhandle* は、`Number` 接続プロパティに等しい、接続の ID 番号です。**connection_property** システム関数は、次のように接続 ID を返します。

```
SELECT connection_property ( 'Number' )
```

有効な *connhandle* の入力パラメータで呼び出されると、**sp_iqconnection** はその接続に対応する 1 つのローのみを返します。

パーミッション

DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、`EXECUTE` パーミッションが付与される必要があります。

説明

sp_iqconnection は、アクティブな各接続に対して 1 つのローを返します。 カラムの ConnHandle、Name、Userid、LastReqTime、ReqType、CommLink、NodeAddr、および LastIdle は、接続プロパティの Number、Name、Userid、LastReqTime、ReqType、CommLink、NodeAddr、および LastIdle にそれぞれ対応しており、システム関数 **sa_conn_info** と同じ値を返します。追加のカラムは、Sybase IQ エンジンの Sybase IQ 側から接続データを返します。 ローは、ConnCreateTime の順で並べられます。

MPXServerName カラムには、次の表に示すようにマルチプレックスのノード間通信 (INC) に関連する情報が格納されています。

表 113 : MPXServerName カラム値

実行されているサーバ	MPXServerName カラムの内容
シンプレックス・サーバ	NULL (すべての接続はローカル接続またはユーザ接続です)。
マルチプレックス・コーディネータ	• ローカル／ユーザ接続の場合は NULL。 • 各 INC 接続 (オンデマンド接続または専用ハートビート接続のいずれか) のセカンダリ・ノードのサーバ名 (接続元) の値を含む。
マルチプレックス・セカンダリ	• ローカル／ユーザ接続の場合は NULL。 • コーディネータのサーバ名 (接続元) の値を含む。

Java アプリケーションでは、RemotePWD フィールドに、TDS クライアントから Sybase IQ 固有の接続プロパティを指定します。次の例は、IQ 固有の接続パラメータの指定方法を示します。**myconnection** には IQ 接続名が入ります。

```
p.put("RemotePWD",",,CON=myconnection");
```

詳細については、『SQL Anywhere サーバー – プログラミング』を参照してください。

表 114 : sp_iqconnection カラム

カラム名	説明
ConnHandle	接続の ID 番号。
Name	サーバの名前を指定します。
Userid	接続のユーザ ID。
LastReqTime	指定された接続に対する直前の要求が開始された時刻。
ReqType	最後の要求のタイプを示す文字列。
IQCmdType	Sybase IQ 側で現在実行されているコマンド (ある場合)。 コマンドの種類には、エンジンの実装レベルで定義されたコマンドが反映されます。 これらのコマンドは、トランザクション・コマンド、IQ ストア内のデータを対象とした DDL および DML コマンド、内部 IQ カーソル・コマンド、特殊な制御コマンド (**OPEN** と **CLOSE DB**、**BACKUP**、**RESTORE** など) で構成されます。

カラム名	説明
LastIQCmdTime	この接続の Sybase IQ エンジンの IQ 側で最後の IQ コマンドが開始または完了した時刻。
IQCursors	この接続の IQ ストアでオープンしているカーソルの数。
LowestIQCursorState	IQ カーソルの状態 (あれば)。接続に複数のカーソルがある場合、すべてのカーソルの中で最小のカーソル状態、つまり完了までの時間が最も長いものが表示されます。カーソル状態は内部の Sybase IQ 実装の詳細を反映するもので、将来的に変更される可能性があります。 このバージョンのカーソル状態は、NONE、INITIALIZED、PARSED、DESCRIBED、COSTED、PREPARED、EXECUTED、FETCHING、END_OF_DATA、CLOSED、COMPLETED です。 名前からもわかるように、カーソル状態は操作の最後に変更されます。 たとえば、状態 PREPARED は、カーソルが実行中であることを示します。
IQthreads	現在、接続に割り当てられている Sybase IQ スレッドの数。 割り当て済みのスレッドには、アイドルなスレッドも含まれます。このカラムから、どの接続がリソースを最も多く使用しているかを判断できます。
TxnID	接続の現在のトランザクションのトランザクション ID。この ID は、BeginTxn、CmtTxn、および PostCmtTxn メッセージによって `.iqmsg` ファイルに表示されるトランザクション ID、また、データベースが開かれたときにログ記録される Txn ID Seq と同じです。
ConnCreateTime	接続が作成された時刻。
TempTableSpaceKB	この接続が IQ テンポラリ・テーブルに格納されているデータに使用している IQ テンポラリ・ストアの領域 (KB 単位)。
TempWorkSpaceKB	この接続が、ソート、ハッシュ、テンポラリ・ビットマップなどの作業領域として使用している IQ テンポラリ・ストアの領域 (キロバイト)。ビットマップや、Sybase IQ テンポラリ・テーブルのインデックスの一部分であるその他のオブジェクトによって使用されている領域は、TempTableSpaceKB に反映されます。
IQConnID	`.iqmsg` ファイル内のすべてのメッセージの一部として表示される 10 桁の接続 ID。これは、サーバ・セッション内でユニークな、単純増加する整数です。
satoiq_count	Sybase IQ エンジンの SQL Anywhere 側から IQ 側への超過の数を表示するのに使用される内部カウンタ。これは、接続のアクティビティを確認するのに役立つことがあります。結果セットはローのバッファに返され、satoiq_count や iqtosa_count をローごとに 1 回増分することはありません。

カラム名	説明
iqtosa_count	Sybase IQ エンジンの IQ 側から SQL Anywhere 側への超過の数を表示するのに使用される内部カウンタ。これは、接続のアクティビティを確認するのに役立つことがあります。
CommLink	接続用の通信リンク。これは、Sybase IQ がサポートするネットワーク・プロトコルであり、同一マシン接続用の「ローカル」なプロトコルです。
NodeAddr	クライアント／サーバ接続のクライアント用ノード。
LastIdle	要求間のチックの数。
MPXServerName	INC 通信の場合、varchar(128) 値には、INC 通信を開始したマルチプレックス・サーバの名前が含まれます。INC 通信でない場合、NULL となります。
LSName	接続の論理サーバ名。論理サーバのコンテキストが未知または適用不可の場合、NULL となります。

例

sp_iqconnection の出力例を次に示します。

```
ConnHandle      Name    Userid                 LastReqTime   ReqType
=== ================== ======  ==========================   =======
 1  'SQL_DBC_100525210' 'DBA'   '2011-03-28  09:29:24.466'  'OPEN'


           IQCmdType    LastIQCmdTime      IQCursors  LowestIQCursorState
====================    ==============     =========  ===================
'IQUTILITYOPENCURSOR'  2011-03-28 09:29:24.0       0             'NONE'


IQthreads   TxnID          ConnCreateTime TempTableSpaceKB TempWorkSpaceKB
=========  =======  ====================  ============== ===============
        0  3352568  2011-03-28 09:29:20.0               0           0


IQconnID satoiq_count iqtosa_count CommLink NodeAdd LastIdle MPXServerName LSName
======== ============ ============ ======== ======= ======== ============  ============
      34           43          2  'local'      ''      244  (NULL)        Finance_LS
```

sp_iqconstraint プロシージャ

CREATE TABLE または **ALTER TABLE** を使用して、指定されたテーブルまたはカラムに定義した参照整合性制約を一覧表示します。

構文

```
sp_iqconstraint ['table-name', 'column-name', 'table-owner' ]
```

パーミッション

DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

説明
テーブル名とカラム名を指定しなかった場合、現在接続中のデータベースにある、テンポラリ・テーブルも含むすべてのテーブルのすべての参照整合性制約が返されます。この情報には、ユニークまたはプライマリ・キー制約、参照制約が含まれ、**CREATE TABLE** 文と **ALTER TABLE** 文の両方またはいずれか一方で定義されたロール名が割り当てられています。

例
これは、すべてのテーブルのうち、候補キーまたは外部キーに、所有者が bob のカラム ck1 が含まれているすべてのプライマリ・キー／外部キーの組み合わせを表示した出力例です。

```
call sp_iqconstraint('','ck1','bob')
```

```
PTAB1 bob ASIQ_IDX_T27_HG  unique   ck1,ck2  selftab bob CK6FK3  Y
ASIQ_IDX_T42_HG  ck1,ck2PTAB2 bob ASIQ_IDX_T27_HG  unique   ck1,ck2
selftab bob CK6FK4  Y
ASIQ_IDX_T206_I42_HG  ck1,ck2selftab bob ASIQ_IDX_T26_HG  unique
ck1,ck2  selftab bob CK3FK1  Y
ASIQ_IDX_T206_I42_HG  ck1,ck2
```

表示されるカラムは次のとおりです。

- プライマリ実施テーブル
- テーブル所有者
- 候補キー・インデックス
- プライマリ・キーまたはユニーク・キー
- プライマリ・キー・カラム
- 外部テーブル
- 外部テーブルの所有者
- 外部キー・ロール名
- 実施ステータス (実施されている場合は "Y"、実施されていない場合は "N")
- 外部キー・インデックス
- 外部キー・カラム
- ロケーション ("TEMP"、"MAIN"、または "SYSTEM")

参照：
- sp_iqcolumn プロシージャ (403 ページ)
- sp_iqdatatype プロシージャ (419 ページ)
- sp_iqevent プロシージャ (443 ページ)
- sp_iqhelp プロシージャ (448 ページ)
- sp_iqindex および sp_iqindex_alt プロシージャ (456 ページ)
- sp_iqjoinindex プロシージャ (471 ページ)

- sp_iqpkeys プロシージャ (491 ページ)
- sp_iqprocparm プロシージャ (496 ページ)
- sp_iq_reset_identity プロシージャ (505 ページ)
- sp_iqtable プロシージャ (525 ページ)
- sp_iqview プロシージャ (541 ページ)

sp_iqcontext プロシージャ

接続ごとに、現在実行されている文に関する情報を追跡して表示します。

構文

```
sp_iqcontext [ connhandle ]
```

使用法

入力パラメータ *connhandle* は、Number 接続プロパティに等しい、接続の ID 番号です。たとえば、**SELECT CONNECTION_PROPERTY('NUMBER')** のように記述します。

有効な *connhandle* の入力パラメータで呼び出されると、**sp_iqcontext** はその接続に対応する情報のみを返します。

パーミッション

DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

ワーカ・スレッドのリーダ・ノードであるサーバに MULTIPLEX ADMIN 権限または DBA 権限がない場合、分散クエリ処理 (DQP) ワーカ・スレッドは表示されません。

説明

sp_iqcontext では、DBA が、特定の時間にシステム上で実行中の文を確認したり、その文を発行したユーザおよび接続を調べたりすることができます。 この情報を基に、このユーティリティでは次のような作業を行えます。

- 文のテキストを、**sp_iqconnection** 内の相当する行に対応させ、各接続でのリソースの利用状況やトランザクション情報を入手する。
- 文のテキストを、**-zr** サーバ・オプションが ALL または SQL に設定されている場合に作成される SQL ログ内の相当する行に対応させる。
- 接続情報を使用して、**sp_iqcontext** 内の文のテキストを、ファイル `.iqmsg` (Sybase IQ がクエリ・プランを収集できる場合はクエリ・プランが含まれる) 内の相当する行に対応させる。

- IQ スタック・トレース (`stktrc-yyyymmdd-hhnnss_#.iq`) が作成されていれば、文のテキストをこれに対応させる。
- OS のスタック・トレース (Sun Solaris の pstack など) が生成されている場合、この情報をそれと照合する。

収集された文のテキストの最大サイズは、カタログ・ストアのページ・サイズです。

表 115 : sp_iqcontext のカラム

カラム名	説明
ConnOrCursor	CONNECTION、CURSOR、または DQP。
ConnHandle	接続の ID 番号。DQP の場合は 0。
Name	サーバの名前 (リーダ名)。
Userid	接続、カーソル、または DQP ワーカのユーザ ID。
numIQCursors	カラム 1 が CONNECTION の場合、この接続で開かれているカーソルの数を表します。 カラム 1 が CURSOR の場合、この接続に関連付けられているカーソルに割り当てられた通し番号を表します。 カラム 1 が DQP の場合、CONNECTION は 0 の値を返します。
IQthreads	現在、接続に割り当てられている IQ スレッドの数。割り当て済みのスレッドでも、アイドルである可能性はあります。DQP スレッドの場合、DQP ワーカに割り当てられているスレッドの数を表します。
TxnID	現在のトランザクションのトランザクション ID。ワーカ・スレッドの場合、リーダのトランザクション ID を表します。
ConnOrCurCreate-Time	この接続、カーソル、または DQP が作成された時間。
IQConnID	`.iqmsg` ファイル内のすべてのメッセージの一部として表示される接続 ID。これは、サーバ・セッション内でユニークな、単純増加する整数です。

カラム名	説明
IQGovernPriority	ユーザのクエリが実行キューに入れられる順番を示す値。1 は優先度高、2 (デフォルト) は優先度中、3 は優先度低を示します。値 -1 は、IQGovernPriority を操作に適用しないことを示します。IQGovernPriority 値は、データベース・オプション IQGOVERN_PRIORITY を使用して設定します。 DQP 接続の場合、このカラムには、「コマンドはありません」と表示されます。
CmdLine	実行されるユーザ・コマンドの最初の 4096 文字。 DQP 接続の場合、このカラムには、「コマンドはありません」と表示されます。
属性	分散されているクエリのユニークな ID

例

次の例は、sp_iqcontext をパラメータなしで発行し、現在のすべての接続について結果を表示した場合の出力の一部です。カラム名は、表示幅の制限によってトランケートされます。

```
ConnOrCu.. ConnHandle Name UserId numIQ.. IQthr.. TxnID Conn.. IQcon..
IQGov.. Cmd.. Attributes
CONNECTION 2 sun7bar dbo 0 0 0 2010-08-04 15:15:40.0 15 No command NO
COMMAND
CONNECTION 7 sun7bar dbo 0 0 0 2010-08-04 15:16:00.0 32 No command NO
COMMAND
CONNECTION 10 sun7bar dbo 0 0 0 2010-08-04 15:16:21.0 46 No command NO
COMMAND
...
CONNECTION 229 sun7bar DBA 0 0 1250445 2010-08-05 18:28:16.0 50887 2
select server_name,
inc_state, coordinator_failover from sp_iqmpxinfo() order by server_name
...
DQP 0 dbsrv2873_node_c1DBA 0 1 10000 2010-08-05 18:28:16.0 no command no
command Query ID:
12345; Condition: c1 > 100;
DQP 0 dbsrv2873_node_c1DBA 0 1 10001 2010-08-05 18:28:16.0 no command no
command Query ID:
12346; Node #12 Join (Hash);
```

出力の 1 行目には、connection 2 (IQ 接続 ID は 15) が表示されています。この接続のサーバは sun7bar、ユーザは dbo です。sp_iqcontext の発行時に、この接続はコマンドを実行していませんでした。

connection 229 は、実行中のユーザ・コマンド (カラムに表示できる最大 4096 文字よりも少ない文字のコマンド) を示します。ユーザ・コマンド・フラグメントの前の 2 は、これが優先度中のクエリであることを示します。

接続ハンドル (この例の最初の接続では 2) は、**-zr** ログ内の結果を識別します。IQ 接続 ID (この例の最初の接続の 15) は、`.iqmsg` ファイル内の結果を識別します。UNIX システムでは、**grep** を使用して、接続ハンドルまたは接続 ID のすべてのインスタンスを探すことができます。これにより、すべてのソースからの情報を簡単に関連付けることができます。

最終行から 2 番目の行 (TxnID `10000`) は、DQP ワーカ・スレッドを示します。ワーカ接続は 2 つの不変条件を実行しています。

最終行 (TxnID `10001`) は、接続がハッシュ・ジョインを実行中であることを示します。

参照：

- CONNECTION_PROPERTY 関数 [システム] (153 ページ)
- sp_iqshowpsexe プロシージャ (509 ページ)

sp_iqcopyloginpolicy プロシージャ

既存のログイン・ポリシーをコピーして、新しいログイン・ポリシーを作成します。

構文 1

```
call sp_iqcopyloginpolicy ('existing-policy-name', 'new-policy-name' )
```

構文 2

```
sp_iqcopyloginpolicy 'existing-policy-name', 'new-policy-name'
```

構文 3

```
sp_iqcopyloginpolicy  existing-policy-name, new-policy-name
```

使用法

表 116 : パラメータ

パラメータ	説明
existing policy name	コピーするログイン・ポリシー。
new policy name	作成する新しいログイン・ポリシーの名前 (CHAR(128))。

パーミッション
DBA 権限が必要です。

例
次に示すストアド・プロシージャは、ログイン・ポリシー・オプション値を *root* という既存のログイン・ポリシーからコピーすることで、*lockeduser* という名前の新しいログイン・ポリシーを作成します。

```
call sp_iqcopyloginpolicy ('root','lockeduser')
```

参照：
- sp_expireallpasswords プロシージャ (381 ページ)
- sp_iqaddlogin プロシージャ (381 ページ)
- sp_iqmodifyadmin プロシージャ (481 ページ)
- sp_iqmodifylogin プロシージャ (482 ページ)
- sp_iqpassword プロシージャ (489 ページ)

sp_iqcursorinfo プロシージャ

サーバで現在オープンしているカーソルに関する詳細情報を表示します。

構文

```
sp_iqcursorinfo [ cursor-name ] [, conn-handle ]
```

パーミッション
DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

使用法

パラメータ	説明
cursor-name	カーソルの名前です。このパラメータのみを指定した場合、**sp_iqcursorinfo** はすべての接続内で指定の名前を持つすべてのカーソルに関する情報を返します。 コピーするログイン・ポリシー。
conn-handle	接続 ID を表す整数。このパラメータのみを指定した場合、**sp_iqcursorinfo** は指定された接続のすべてのカーソルに関する情報を返します。

sp_iqcursorinfo プロシージャは、パラメータなしで呼び出せます。パラメータを指定しない場合、**sp_iqcursorinfo** は、現在サーバ上でオープンしているすべての

カーソルに関する情報を返します。両方のパラメータを指定した場合、**sp_iqcursorinfo** は、指定の名前を持つすべてのカーソルに関する情報と、指定の接続内のすべてのカーソルに関する情報をレポートします。

最初のパラメータを指定せずに、2 番目のパラメータを指定する場合には、指定しないパラメータの位置に NULL を入力する必要があります。たとえば、`sp_iqcursorinfo NULL, 1` のように記述します。

表 117 : sp_iqcursorinfo の使用例

構文	出力
sp_iqcursorinfo	サーバで現在オープンしているすべてのカーソルに関する情報を表示します。
sp_iqcursorinfo ‘cursor1’	すべての接続内で `cursor1` という名前のすべてのカーソルに関する情報を表示します。
sp_iqcursorinfo NULL, 3	接続 3 のすべてのカーソルに関する情報を表示します。
sp_iqcursorinfo ‘cursor2’, 4	接続 4 内で `cursor2` という名前のすべてのカーソルに関する情報を表示します。

説明

sp_iqcursorinfo ストアド・プロシージャは、サーバ上で現在オープンしているカーソルについて詳細情報を表示します。**sp_iqcursorinfo** プロシージャによって、データベース管理者は、この 1 つのストアド・プロシージャを使用するだけでカーソル・ステータスを監視できます。また、更新、削除、および挿入されたローの数などの統計情報を表示できます。

1 つ以上のパラメータを指定した場合、指定したパラメータによって結果がフィルタされます。たとえば、*cursor-name* を指定した場合、指定のカーソルに関する情報のみが表示されます。*conn-handle* を指定した場合、**sp_iqcursorinfo** は指定した接続内のカーソルに関する情報のみを返します。パラメータを指定しない場合、**sp_iqcursorinfo** は、現在サーバ上でオープンしているすべてのカーソルに関する情報を表示します。

sp_iqcursorinfo プロシージャは、次のカラムに情報を返します。

表 118 : sp_iqcursorinfo のカラム

カラム名	説明
Name	カーソルの名前。

カラム名	説明
ConnHandle	接続の ID 番号。
IsUpd	Y：カーソルは更新可能です。N：カーソルは更新可能ではありません。
IsHold	Y：カーソルはホールド・カーソルです。N：カーソルはホールド・カーソルではありません。
IQConnID	`.iqmsg` ファイル内のすべてのメッセージの一部として表示される 10 桁の接続 ID。これは、サーバ・セッション内でユニークな、単純増加する整数です。
UserID	カーソルを作成、実行したユーザのユーザ ID (またはユーザ名)。
CreateTime	カーソルが作成された時間。
CurrentRow	結果セット内のカーソルの現在位置。
NumFetch	カーソルがローをフェッチする回数。同じローを複数回フェッチできます。
NumUpdate	カーソルが更新可能な場合に、カーソルがローを更新する回数。同じローを複数回更新できます。
NumDelete	カーソルが更新可能な場合に、カーソルがローを削除する回数。
NumInsert	カーソルが更新可能な場合に、カーソルがローを挿入する回数。
RWTabOwner	カーソルによって RW モードで開かれるテーブルの所有者。
RWTabName	カーソルによって RW モードで開かれるテーブルの名前。
CmdLine	ユーザが実行したコマンドの最初の 4096 文字。

『システム管理ガイド： 第 2 巻』の「プロシージャとバッチの使用」>「プロシージャでのカーソルの使用」を参照してください。トランザクション内のカーソルの情報については、『システム管理ガイド： 第 1 巻』の「トランザクションとバージョン管理」>「トランザクション内のカーソル」を参照してください。

CURSOR データベース・オプションについては、『リファレンス： 文とオプション』の「データベース・オプション」>「アルファベット順のオプション・リスト」を参照してください。

CURSOR 文については、『リファレンス： 文とオプション』の「SQL 文」を参照してください。

例
サーバで現在オープンしているすべてのカーソルに関する情報を表示します。

```
sp_iqcursorinfo

Name      ConnHandle      IsUpd      IsHold      IQConnID      UserID
------------------------------------------------------------------------
--
crsr1              1          Y           N           118         DBA
crsr2              3          N           N           118         DBA

CreateTime               CurrentRow      NumFetch      NumUpdate
-------------------------------------------------------------------
2009-06-26 15:24:36.000          19      100000000     200000000
2009-06-26 15:38:38.000       20000      200000000

NumDelete     NumInsert      RWTabOwner      RWTabName        CmdLine
------------------------------------------------------------------------
---
 20000000    3000000000            DBA          test1    call proc1()
                                                         call proc2()
```

sp_iqdatatype プロシージャ

システム・データ型およびユーザ定義データ型に関する情報を表示します。

構文

sp_iqdatatype [*type-name*], [*type-owner*], [*type-type*]

パーミッション

DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

使用法

表 119 : パラメータ

パラメータ	説明
type-name	データ型の名前。
type-owner	データ型の作成者の名前。
type-type	データ型のタイプ。指定できる値は次のとおりです。 • **SYSTEM**： システム定義のデータ型(ユーザ SYS または dbo が所有するデータ型) に関する情報のみを表示します。 • **ALL**： ユーザ・データ型およびシステム・データ型に関する情報を表示します。 • その他の値：ユーザ・データ型に関する情報を表示します。

sp_iqdatatype プロシージャは、パラメータなしで呼び出せます。パラメータを指定しない場合、ユーザ定義データ型 (dbo または SYS が所有していないデータ型) に関する情報のみがデフォルトで表示されます。

最初の 2 つのパラメータのいずれかを指定せずに、その次のパラメータを指定する場合は、指定しないパラメータの位置に NULL を指定する必要があります。たとえば、`sp_iqdatatype NULL, NULL, SYSTEM` および `sp_iqdatatype NULL, user1` とします。

表 120 : sp_iqdatatype の使用例

構文	出力
sp_iqdatatype	データベース内のすべてのユーザ定義データ型に関する情報を表示します。
sp_iqdatatype country_t	`country_t` という名前のユーザ定義データ型に関する情報を表示します。
sp_iqdatatype non_existing_type	データ型 `non_existing_type` が存在しないため、ローは返されません。
sp_iqdatatype NULL, DBA	`DBA` が所有するすべてのユーザ定義データ型に関する情報を表示します。
sp_iqdatatype country_t, DBA	`DBA` が所有するデータ型 `country_t` に関する情報を表示します。
sp_iqdatatype rowid	`rowid` は、システム定義のデータ型です。`rowid` という名前のユーザ定義データ型が存在しない場合、ローは返されません (デフォルトでは、ユーザ定義データ型のみが返されます)。
sp_iqdatatype rowid, SYS	`rowid` データ型はユーザ定義データ型ではないため、ローは返されません (デフォルトでは、ユーザ定義データ型のみが返されます)。
sp_iqdatatype NULL, NULL, SYSTEM	`dbo` または SYS が所有するすべてのシステム定義データ型に関する情報を表示します。
sp_iqdatatype rowid, NULL, SYSTEM	システム・データ型 `rowid` に関する情報を表示します。
sp_iqdatatype NULL, NULL, 'ALL'	ユーザ定義データ型およびシステム・データ型に関する情報を表示します。

説明
sp_iqdatatype ストアド・プロシージャは、データベース内のシステムおよびユーザ定義データ型に関する情報を表示します。ユーザ定義データ型は、ドメインと呼ばれる場合もあります。あらかじめ定義されているドメイン名は、**sp_iqdatatype** 出力には含まれません。

1 つ以上のパラメータを指定した場合、指定したパラメータによって、**sp_iqdatatype** の結果がフィルタされます。たとえば、*type-name* を指定した場合、指定のデータ型に関する情報のみが表示されます。*type-owner* を指定した場合、**sp_iqdatatype** は指定の所有者が所有するデータ型に関する情報のみを返します。パラメータを指定しない場合、**sp_iqdatatype** はデータベース内のすべてのユーザ定義データ型に関する情報を表示します。

sp_iqdatatype プロシージャは、次のカラムに情報を返します。

表 121 : sp_iqdatatype カラム

カラム名	説明
type_name	データ型の名前。
作成者	データ型の所有者。
NULL	Y はユーザ定義データ型が NULL を許可することを示し、N は許可しないことを示します。U は、データ型について NULL 値が指定されていないことを示します。
width	文字列カラムでは長さ、数値カラムでは精度、その他のデータ型では格納サイズ (バイト数) を表示します。
scale	数値データ型カラムでは小数点以下の桁数、その他のデータ型では 0 を表示します。
“default”	データ型のデフォルト値
"check"	データ型の CHECK 条件

例
`country_t` という名前のユーザ定義データ型に関する情報を表示します。

```
sp_iqdatatype country_t
type_name   creator   nulls   width   scale   “default”   “check”
country_t   DBA       U       15      0       (NULL)      (NULL)
```

参照：
- sp_iqcolumn プロシージャ (403 ページ)
- sp_iqconstraint プロシージャ (410 ページ)

- sp_iqevent プロシージャ (443 ページ)
- sp_iqhelp プロシージャ (448 ページ)
- sp_iqindex および sp_iqindex_alt プロシージャ (456 ページ)
- sp_iqjoinindex プロシージャ (471 ページ)
- sp_iqpkeys プロシージャ (491 ページ)
- sp_iqprocparm プロシージャ (496 ページ)
- sp_iq_reset_identity プロシージャ (505 ページ)
- sp_iqtable プロシージャ (525 ページ)
- sp_iqview プロシージャ (541 ページ)

sp_iqdbsize プロシージャ

現在のデータベースのサイズを表示します。

構文

```
sp_iqdbsize( [ main ] )
```

パーミッション

DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

説明

データベースの合計サイズを返します。また、メモリにデータベースを保持するために必要なページ数と、(ディスク上で) データベースを圧縮した場合の IQ ページ数も返します。

表 122 : sp_iqdbsize のカラム

カラム名	説明
Database	データベース・ファイルのパス名。
Physical Blocks	データベースの合計サイズ (ブロック数)。 IQ データベースは、1 つまたは複数の DB 領域で構成されています。各 DB 領域は、当初メガバイト単位で指定されたサイズに固定されています。このメガバイトの値は、IQ ページ・サイズと、その IQ ページ・サイズに対応するブロック・サイズを使ってブロックに変換されます。Physical Blocks カラムには、各 Sybase IQ DB 領域のサイズの累積値がブロック単位で反映されます。

カラム名	説明
KBytes	データベースの合計サイズ (キロバイト) です。この値は、ブロックで表したデータベースの合計サイズ (前の **sp_iqdbsize** カラムの Physical Blocks) に、ブロック・サイズを乗算したものです。このブロック・サイズは IQ ページ・サイズに依存します。
Pages	テーブルおよびジョイン・インデックスに格納されているすべてのデータとそれらのオブジェクトのメタデータをメモリ内に表示するために必要な IQ ページの合計数です。この値は常に、Compressed Pages (次の **sp_iqdbsize** カラム) の値よりも大きいか、または等しくなります。
Compressed Pages	テーブルおよびジョイン・インデックスに格納されているすべてのデータと、それらのオブジェクトのメタデータをディスクに格納するために必要な IQ ページの合計数です。 IQ ページがメモリからディスクに書き込まれるとき、Sybase IQ はページを圧縮するため、この値は、Pages (前の **sp_iqdbsize** カラム) の値よりも小さいか、または等しくなります。 **sp_iqdbsize** Compressed Pages カラムは、圧縮されたページの数を表します。
NBlocks	テーブルおよびジョイン・インデックスへのデータの格納に使用されている領域の合計サイズ (ブロック単位)。この値は常に、**sp_iqdbsize** Physical Blocks の値より小さいか、または等しくなります。
Catalog Blocks	テーブルおよびジョイン・インデックスのメタデータの格納に使用されている領域の合計サイズ (ブロック単位)。

マルチプレックス環境では、このプロシージャを使用できます。詳細については、『Sybase IQ Multiplex の使用』の「マルチプレックス・リファレンス」> 「sp_iqdbsize プロシージャ」を参照してください。

例
この例は、データベース `iqdemo` のサイズ情報を表示します。

```
sp_iqdbsize
```

```
Database

PhysicalBlocks KBytes Pages CompressedPages NBlocks CatalogBlocks
============== ====== ===== =============== ======= =============
/system1/sybase/IQ-15_3/demo/iqdemo.db
          1280    522   688             257    1119            18
```

sp_iqdbspace プロシージャ

各 IQ DB 領域についての詳細情報を表示します。

構文

```
sp_iqdbspace [ dbspace-name ]
```

パーミッション

DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

説明

sp_iqdbspace の情報を使用して、データの移動が必要かどうかを判断できます。また、移動されたデータに関しては、旧バージョンの割り付けが解除されているかどうかを確認できます。**sp_iqdbspace** は次の情報を表示します。

表 123 : sp_iqdbspace のカラム

カラム名	説明
DBSpaceName	**CREATE DBSPACE** 文で指定された DB 領域の名前。**CREATE DATABASE...CASE IGNORE** または **CASE RESPECT** の指定に関係なく、DB 領域名の大文字と小文字は常に区別されません。
DBSpaceType	DB 領域のタイプ (MAIN、SHARED_TEMP、または TEMPORARY)。
Writable	T (書き込み可能) または F (書き込み不可)。
オンライン	T (オンライン) または F (オフライン)。
使用法	DB 領域のすべてのファイルで現在使用されている DB 領域の割合。
TotalSize	DB 領域のすべてのファイルの合計サイズ。単位は、B (バイト)、K (キロバイト)、M (メガバイト)、G (ギガバイト)、T (テラバイト)、または P (ペタバイト)。
予約	DB 領域のすべてのファイルに追加できる予約領域の合計。
NumFiles	DB 領域内のファイルの数。
NumRWFiles	DB 領域内の読み込み／書き込みファイルの数。
Stripingon	T (オン) または F (オフ)。
StripeSize	ディスク・ストライピングが有効になっている場合、次の DB 領域に移動するまでに DB 領域に書き込まれたデータの量。

カラム名	説明
BlkTypes	ユーザ・データと内部システム構造が使用している領域。
OkToDrop	DB 領域を削除できる場合は 'Y'、それ以外の場合は 'N'。

BlkTypes ブロック・タイプ識別子の値は、次のとおりです。

表 124 : sp_iqdbspace のブロック・タイプ

識別子	ブロック・タイプ
A	アクティブなバージョン
B	バックアップ構造
C	チェックポイント・ログ
D	データベースの識別情報
F	フリー・リスト
G	グローバル・フリー・リスト・マネージャ
H	フリー・リストのヘッダ・ブロック
I	インデックス・アドバイスの格納
M	マルチプレックス CM*
O	旧バージョン
T	テーブルの使用
U	インデックスの使用
N	カラムの使用
X	チェックポイントでの削除

*マルチプレックス・コミット ID ブロック (実際は 128 ブロック) は、シンプレックス・データベースで使用されていない場合でも、すべての IQ データベースに存在します。

マルチプレックス機能の詳細については、『Sybase IQ Multiplex の使用』の「マルチプレックス・リファレンス」>「システム・プロシージャ」>「sp_iqdbspace プロシージャ」を参照してください。

例
次の出力には、DB 領域に関する情報が表示されています。

```
sp_iqdbspace;
```

注意： 出力内容をわかりやすくするため、次の例に iqdemo データベース内のオブジェクトを示します。iqdemo には、iq_main というサンプルのユーザ DB 領域が含まれています。ただし、この領域は、ユーザが所有するデータベースには存在しない場合があります。

DBSpaceName	DBSpaceType	Writable	オンライン	使用法	合計サイズ	予約	NumFiles	NumRWFiles	Stripin
IQ_MAIN	MAIN	T	T	55	75M	200M	1	1	T
IQ__ SYSTEM_ MAIN	MAIN	T	T	21	300M	50M	1	1	F
IQ_SYSTEM_ TEMP	TEMPORARY	T	T	1	100M	50M	1	1	F

参照：

- sp_iqindexinfo プロシージャ (463 ページ)
- sp_iqdbspaceinfo プロシージャ (426 ページ)
- sp_iqspaceinfo プロシージャ (511 ページ)

sp_iqdbspaceinfo プロシージャ

指定のテーブルまたはジョイン・インデックスで使用される各オブジェクトおよびサブオブジェクトのサイズを表示します。

構文

```
sp_iqdbspaceinfo [ dbspace-name ] [ , owner_name ] [ ,
object_name ] [ , object-type  ]
```

パーミッション

DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

使用法

パラメータ	説明
dbspace_name	指定した場合、**sp_iqdbspaceinfo** は、指定の DB 領域内のコンポーネントを持つ各テーブルを1行に表示します。指定しない場合、プロシージャはデータベース内のすべての DB 領域の情報を表示します。
owner_name	オブジェクトの所有者。指定した場合、**sp_iqdbspaceinfo** は、指定の所有者を持つテーブルおよびジョイン・インデックスについてのみ出力を表示します。指定しない場合、**sp_iqdbspaceinfo** は、データベース内のすべてのユーザのテーブルおよびジョイン・インデックスに関する情報を表示します。
object_name	テーブルまたはジョイン・インデックスの名前。指定しない場合、**sp_iqdbspaceinfo** は、データベース内のすべてのテーブルおよびジョイン・インデックスに関する情報を表示します。
object_type	有効なオブジェクト・タイプは、**table** (テーブル情報の場合) または **joinindex** (ジョイン・インデックス情報の場合) です。指定しない場合、オブジェクト・タイプのデフォルトは **table** になります。

すべてのパラメータがオプションであり、どのパラメータも別のパラメータの値とは関係なく指定できます。

sp_iqdbspaceinfo ストアド・プロシージャは、*dbspace_name*、*object_name*、*owner_name* の解釈のために、ワイルドカード文字をサポートしています。これは、**LIKE** 句がクエリ内のパターンを照合するのと同じ方法で、指定のパターンと一致するすべての DB 領域の情報を表示します。

説明

sp_iqdbspaceinfo は、各 DB 領域に存在するオブジェクトによって使用される領域の大きさを DBA に示します。DBA はこの情報を使用して、DB 領域を削除する前に移動する必要のあるオブジェクトを判断できます。サブオブジェクト・カラムには、整数の量でレポートされるサイズが表示されます。各値の後ろには、サフィックス B、K、M、G、T、または P が付き、これらはそれぞれキロバイト、メガバイト、ギガバイト、テラバイト、およびペタバイトを表します。

テーブルの場合、**sp_iqdbspaceinfo** は、すべてのサブオブジェクトのサイジング情報を表示します (サフィックス B、K、M、G、T、または P を持つ整数の量を使用します)。ジョイン・インデックスの場合は、ジョイン・インデックスと、それに関連するすべてのサブオブジェクトについてのサイジング情報を表示します。出力は、*dbspace_name*、*object_name*、*owner_name* でソートされます。

表 125 : sp_iqdbspaceinfo のカラム

カラム名	説明
dbspace_name	DB 領域の名前。
object_type	オブジェクトのタイプ (**table** または **joinindex** のみ)。
owner	オブジェクトの所有者の名前。
object_name	DB 領域にあるオブジェクトの名前 (テーブルおよびジョイン・インデックス・タイプのみ)。
object_id	オブジェクトのグローバル・オブジェクト ID。
id	オブジェクトのテーブル ID またはジョイン・インデックス ID。
columns	指定の DB 領域のカラム記憶領域のサイズ。
indexes	指定の DB 領域のインデックス記憶領域のサイズ。システムで作成されたインデックス (一意性制約または FP インデックスの HG インデックスなど) は使用できません。
metadata	指定の DB 領域のメタデータ・オブジェクトの記憶領域サイズ。
primary_key	指定の DB 領域のプライマリ・キー関連オブジェクトの記憶領域サイズ。
unique_constraint	指定の DB 領域の一意性制約関連オブジェクトの記憶領域サイズ。
foreign_key	指定の DB 領域の外部キー関連オブジェクトの記憶領域サイズ。
dbspace_online	DB 領域がオンライン (**Y**) か、オフライン (**N**) かを示します。

注意： -r スイッチ (読み込み専用) で開始したサーバに対して sp_iqdbspaceinfo を実行した場合、`Msg 13768, Level 14, State 0: SQL Anywhere Error -757: Modifications not permitted for read-only database` というエラーが表示されます。 これは予期された動作です。 `sp_iqdbspace`、`sp_iqfile`、`sp_iqdbspaceobjectinfo`、`sp_iqobjectinfo` などの他のストアド・プロシージャでは、エラーは発生しません。

マルチプレックス環境では、このプロシージャを使用できます。詳細については、『Sybase IQ Multiplex の使用』の「マルチプレックス・リファレンス」> 「sp_iqdbspaceinfo プロシージャ」を参照してください。

例
データベース内のすべての DB 領域のすべてのテーブルにあるすべてのオブジェクトおよびサブオブジェクトのサイズを表示します。

注意： 出力内容をわかりやすくするため、次の 3 つの例は iqdemo データベース内のオブジェクトを示しています。iqdemo には、iq_main というサンプルのユーザ DB 領域が含まれています。ただし、この領域は、ユーザが所有するデータベースには存在しない場合があります。

```
sp_iqdbspaceinfo
```

```
dbspace_name    object_type  owner   object_name     object_id  id
  columns
iq_main         table        DBA     emp1            3689     741   96K
iq_main         table        DBA     iq_dummy        3686     740   24K
iq_main         table        DBA     sale            3698     742   96K
iq_main          table         GROUPO  Contacts          3538      732
   288K
iq_main          table         GROUPO  Customers         3515      731
   240K
iq_main         table        GROUPO  Departments      3632     738   72K
iq_main          table         GROUPO  Employees         3641      739
   408K
iq_main          table         GROUPO  FinancialCodes    3612      736
   72K
iq_main         table        GROUPO  FinancialData    3621     737   96K
iq_main          table         GROUPO  Products
 3593     735   272K
iq_main          table         GROUPO  SalesOrderItems   3580      734
   120K
iq_main          table         GROUPO  SalesOrders       3565      733
   144K

indexes  metadata  primary_key  unique_constraint  foreign_key  dbsp
ace_online
0B       1.37M     0B           0B                 0B           Y
0B       464K      0B           0B                 0B           Y
0B       1.22M     0B           0B                 0B           Y
0B       5.45M     24K          0B                 48K          Y
48K      4.63M     24K          0B                 0B           Y
0B       1.78M     24K          0B                 48K          Y
0B       8.03M     24K          0B                 48K          Y
0B       1.53M     24K          0B                 0B           Y
0B       2.19M     24K          0B                 48K          Y
192K     4.67M     24K          0B                 0B           Y
0B       2.7M      24K          0B                 104K         Y
0B       3.35M     24K          0B                 144K         Y
```

データベース内で指定した DB 領域の指定ユーザが所有する、すべてのオブジェクトおよびサブオブジェクトのサイズを表示します。

```
sp_iqdbspaceinfo iq_main,GROUPO
```

```
dbspace_name    object_type  owner   object_name     object_id  id
  columns
iq_main          table         GROUPO  Contacts          3538      732
   288K
iq_main          table         GROUPO  Customers         3515      731
   240K
```

```
iq_main        table       GROUPO  Departments      3632    738   72K
iq_main          table        GROUPO  Employees          3641     739
   408K
iq_main          table        GROUPO  FinancialCodes     3612     736
   72K
iq_main        table       GROUPO  FinancialData     3621    737   96K
iq_main          table        GROUPO  Products           3593     735
   272K
iq_main          table        GROUPO  SalesOrderItems    3580     734
   120K
iq_main          table        GROUPO  SalesOrders        3565     733
   144K
indexes  metadata  primary_key  unique_constraint  foreign_key  dbsp
ace_online 0B      5.45M    24K         0B              48K         Y
48K      4.63M     24K          0B                 0B           Y
0B       1.78M     24K          0B                 48K          Y
0B       8.03M     24K          0B                 48K          Y
0B       1.53M     24K          0B                 0B           Y
0B       2.19M     24K          0B                 48K          Y
192K     4.67M     24K          0B                 0B           Y
0B       2.7M      24K          0B                 104K         Y
0B       3.35M     24K          0B                 144K         Y
```

データベース内で指定した DB 領域の指定のユーザが所有する、指定のオブジェクトとそのサブオブジェクトのサイズを表示します。

```
sp_iqdbspaceinfo iq_main,GROUPO,Departments
```

```
dbspace_name    object_type  owner   object_name     object_id  id
  columns
iq_main        table       GROUPO  Departments      3632    738   72K

indexes  metadata  primary_key  unique_constraint  foreign_key  dbsp
ace_online
0B       1.78M     24K          0B                 48K          Y
```

参照：

- sp_iqindexinfo プロシージャ (463 ページ)
- sp_iqdbspace プロシージャ (424 ページ)
- sp_iqspaceinfo プロシージャ (511 ページ)

sp_iqdbspaceobjectinfo プロシージャ

指定の DB 領域のテーブルおよびジョイン・インデックス・タイプのオブジェクトと、そのサブオブジェクト (カラム、インデックス、メタデータ、プライマリ・キー、一意性制約、外部キー、およびパーティション) をリストします。

構文

sp_iqdbspaceobjectinfo [*dbspace-name*] [, *owner_name*] [, *object_name*] [, *object-type*]

パーミッション
DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

使用法
すべてのパラメータがオプションであり、どのパラメータも他のパラメータの値とは関係なく指定できます。

表 126 : パラメータ

パラメータ	説明
dbspace-name	指定した場合、**sp_iqdbspaceobjectinfo** は、指定の DB 領域についてのみ出力を表示します。指定しない場合、データベース内のすべての DB 領域の情報を表示します。
owner-name	オブジェクトの所有者。指定した場合、**sp_iqdbspaceobjectinfo** は、指定の所有者を持つテーブルおよびジョイン・インデックスについてのみ出力を表示します。指定しない場合、**sp_iqdbspaceobjectinfo** は、データベース内のすべてのユーザのテーブルおよびジョイン・インデックスに関する情報を表示します。
object-name	テーブルまたはジョイン・インデックスの名前。指定しない場合、**sp_iqdbspaceobjectinfo** は、データベース内のすべてのテーブルおよびジョイン・インデックスに関する情報を表示します。
object-type	有効なオブジェクト・タイプは、**table** (テーブル情報の場合) または **joinindex** (ジョイン・インデックス情報の場合) です。指定しない場合、オブジェクト・タイプのデフォルトは **table** になります。

sp_iqdbspaceobjectinfo ストアド・プロシージャは、*dbspace_name*、*object_name*、*owner_name* の解釈のために、ワイルドカード文字をサポートしています。これは、**LIKE** 句がクエリ内のパターンを照合するのと同じ方法で、指定のパターンと一致するすべての DB 領域の情報を表示します。

説明
テーブルの場合、**sp_iqdbspaceobjectinfo** は、関連するすべてのサブオブジェクトの要約情報を表示します。ジョイン・インデックスの場合は、ジョイン・インデックスと、それに関連するすべてのサブオブジェクトについてのサイジング情報を表示します。ストアド・プロシージャの出力は、dbspace_name、owner、および object_name でソートされます。

sp_iqdbspaceobjectinfo は、入力パラメータ値に基づいて次の情報を表示します。

表 127 : sp_iqdbspaceobjectinfo のカラム

カラム名	説明
dbspace_name	DB 領域の名前。
dbspace_id	DB 領域の識別子。
object_type	テーブルまたはジョイン・インデックス。
owner	オブジェクトの所有者の名前。
object_name	DB 領域にあるオブジェクトの名前 (テーブルおよびジョイン・インデックス・タイプのみ)。
object_id	オブジェクトのグローバル・オブジェクト ID。
id	オブジェクトのテーブル ID またはジョイン・インデックス ID。
columns	指定の DB 領域にあるテーブル・カラムの数。カラム、またはいずれかのカラムパーティションが DB 領域にある場合、その DB 領域に存在しているものとして数えられます。結果は n/N フォームで表示されます (テーブルの合計 N 個のカラムのうち n 個が、指定された DB 領域に存在します)。
indexes	指定の DB 領域にあるテーブルのユーザ定義インデックスの数。n/N フォームで表示されます (テーブル上の合計 N 個のインデックスのうち n 個が、指定された DB 領域に存在します)。一意性制約の場合、これには、FP インデックスや HG インデックスなどのシステム作成のインデックスは含まれません。
metadata	サブオブジェクトのメタデータ情報もこの DB 領域にあるかどうかを示すブール・フィールド (Y/N)。
primary_key	テーブルのプライマリ・キー (存在する場合) がこの DB 領域にあるかどうかを示すブール・フィールド (1/0)。
unique_constraint	指定の DB 領域にあるテーブルの一意性制約の数。n/N フォームで表示されます (テーブル上の合計 N 個の一意性制約のうち n 個が、指定された DB 領域に存在します)。
foreign_key	指定の DB 領域にあるテーブルの外部キーの数。n/N フォームで表示されます (テーブル上の合計 N 個の外部キーのうち n 個が、指定された DB 領域に存在します)。

カラム名	説明
partitions	指定の DB 領域にあるテーブルのパーティションの数。n/N フォームで表示される (テーブルの合計 N 個のパーティションのうち n 個が、指定された DB 領域に存在します)。

例

注意： 出力内容をわかりやすくするため、次の 2 つの例は `iqdemo` データベース内のオブジェクトを示しています。iqdemo には、`iq_main` というサンプルのユーザ DB 領域が含まれています。ただし、この領域は、ユーザが所有するデータベースには存在しない場合があります。

データベース内の特定の DB 領域に関する情報を表示します。

```
sp_iqdbspaceobjectinfo iq_main
```

```
dbspace_name  dbspace_id object_type  owner   object_name  object_i
d  id  columns
iq_main       16387      table        DBA     emp1           3689
   741  4/4
iq_main       16387      table        DBA     iq_dummy       3686
   740  1/1
iq_main       16387      table        DBA     sale           3698
   742  4/4
iq_main       16387      table        GROUPO  Contacts       3538
   732  12/12
iq_main       16387      table        GROUPO  Customers      3515
   731  10/10
iq_main       16387      table        GROUPO  Departments    3632
   738  3/3
iq_main       16387      table        GROUPO  Employees      3641
   739  21/21
iq_main       16387      table        GROUPO  FinancialCodes 3612
   736  3/3
iq_main       16387      table        GROUPO  FinancialData  3621
   737  4/4
iq_main       16387      table        GROUPO  Products       3593
   735  8/8
iq_main       16387      table        GROUPO  SalesOrderItems3580
   734  5/5
iq_main       16387      table        GROUPO  SalesOrders    3565
   733  6/6

indexes  metadata  primary_key  unique_constraint  foreign_key  part
itions
0/0      Y                   0  0/0                0/0          0/0
0/0      Y                   0  0/0                0/0          0/0
0/0      Y                   0  0/0                0/0          0/0
0/0      Y                   1  0/0                1/1          0/0
1/1      Y                   1  0/0                0/0          0/0
0/0      Y                   1  0/0                1/1          0/0
```

```
0/0      Y                    1  0/0                 1/1          0/0
0/0      Y                    1  0/0                 0/0          0/0
0/0      Y                    1  0/0                 1/1          0/0
4/4      Y                    1  0/0                 0/0          0/0
0/0      Y                    1  0/0                 2/2          0/0
0/0      Y                    1  0/0                 3/3          0/0
```

データベース内で指定した DB 領域の指定ユーザが所有するオブジェクトに関する情報を表示します。

```
sp_iqdbspaceobjectinfo iq_main,GROUPO
```

```
dbspace_name  dbspace_id object_type  owner   object_name    object_
id  id  columns
iq_main       16387      table        GROUPO  Contacts       3538
     732 2/12
iq_main       16387      table        GROUPO  Customers      3515
     731 10/10
iq_main       16387      table        GROUPO  Departments    3632
     738 3/3
iq_main       16387      table        GROUPO  Employees      3641
     739 21/21
iq_main       16387      table        GROUPO  FinancialCodes 3612
     736 3/3
iq_main       16387      table        GROUPO  FinancialData  3621
     737 4/4
iq_main       16387      table        GROUPO  Products       3593
     735 8/8
iq_main       16387      table        GROUPO  SalesOrderItems3580
     734 5/5
iq_main       16387      table        GROUPO  SalesOrders    3565
     733 6/6

indexes  metadata  primary_key  unique_constraint  foreign_key  part
itions
0/0      Y                   1  0/0                1/1          0/0
1/1      Y                   1  0/0                0/0          0/0
0/0      Y                   1  0/0                1/1          0/0
0/0      Y                   1  0/0                1/1          0/0
0/0      Y                   1  0/0                0/0          0/0
0/0      Y                   1  0/0                1/1          0/0
4/4      Y                   1  0/0                0/0          0/0
0/0      Y                   1  0/0                2/2          0/0
0/0      Y                   1  0/0                3/3          0/0
```

sp_iqdbspaceobjectinfo を使用して、オブジェクトの移動に使用できるコマンドを構成します。次の例では、コマンドで dbspace_x 上のすべてのテーブルを dbspace_y に移動します。

```
SELECT 'ALTER TABLE ' || owner || '.' ||
object_name || ' MOVE TO dbspace_y;'
FROM sp_iqdbspaceobjectinfo()
WHERE object_type = 'table' AND
dbspace_name = 'dbspace_x';
```

次の **ALTER TABLE** コマンドはその結果です。

```
ALTER TABLE DBA.dt1 MOVE TO dbspace_y;
ALTER TABLE DBA.dt2 MOVE TO dbspace_y;
ALTER TABLE DBA.dt3 MOVE TO dbspace_y;
```

sp_iqdbstatistics プロシージャ

最後に実行された **sp_iqcheckdb** の結果をレポートします。

構文

```
sp_iqdbstatistics
```

パーミッション

DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

説明

最後に実行された **sp_iqcheckdb** が収集したデータベースの統計情報を表示します。

sp_iqcheckdb の使用例と **sp_iqcheckdb** 出力の解釈の詳細については、『システム管理ガイド： 第 1 巻』の「システムのリカバリとデータベースの修復」を参照してください。

例

次の例は、**sp_iqdbstatistics** からの出力を表示します。この例では、最後に実行された **sp_iqcheckdb** は、コマンド **sp_iqcheckdb 'allocation database'** です。

```
           DB Statistics                    Value               Flags
=====================================|============================|
=====
DBCC Allocation Mode Report          |                            |
=====================================|============================|
=====
** DBCC Status                      |Errors Detected           |*****
   DBCC Work units Dispatched        |163                         |
   DBCC Work units Completed         |163                         |
=====================================|============================|
=====
Allocation Summary                   |                            |
=====================================|============================|
=====
   Blocks Total                      |8192                        |
   Blocks in Current Version         |4954                        |
   Blocks in All Versions            |4954                        |
   Blocks in Use                     |4986                        |
   % Blocks in Use                   |60                          |
** Blocks Leaked                    |32                          |*****
```

```
                                    |                              |
====================================|==============================|
=====
Allocation Statistics               |                              |
====================================|==============================|
=====
   Blocks Created in Current TXN    |382                           |
   Blocks To Drop in Current TXN    |382                           |
   Marked Logical Blocks            |8064                          |
   Marked Physical Blocks           |4954                          |
   Marked Pages                     |504                           |
   Blocks in Freelist               |126553                        |
   Imaginary Blocks                 |121567                        |
   Highest PBN in Use               |5432                          |
** 1st Unowned PBN                 |452                        |*****
   Total Free Blocks                |3206                          |
   Usable Free Blocks               |3125                          |
   % Free Space Fragmented          |2                             |
   Max Blocks Per Page              |16                            |
   1  Block Page Count              |97                            |
   3  Block Page Count              |153                           |
   4  Block Page Count              |14                            |
   ...
   9  Block Hole Count              |2                             |
   16 Block Hole Count              |194                           |
                                    |                              |
   Database Objects Checked         |1                             |
   B-Array Count                    |1                             |
   Blockmap Identity Count          |1                             |
====================================|==============================|
=====
Connection Statistics               |                              |
====================================|==============================|
=====
```

sp_iqdroplogin プロシージャ

Sybase IQ のユーザ・アカウントを削除します。

構文 1

```
call sp_iqdroplogin (‘userid’)
```

構文 2

```
sp_iqdroplogin ‘userid’
```

構文 3

```
sp_iqdroplogin  userid
```

構文 4

```
sp_iqdroplogin (‘userid’)
```

パーミッション
DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

使用法

表 128 : パラメータ

パラメータ	説明
userid	削除するユーザのユーザ ID。

説明
sp_iqdroplogin は、指定されたユーザを削除します。

例
次のストアド・プロシージャ・コールは、ユーザ rose を削除します。

```
sp_iqdroplogin 'rose'
```

```
sp_iqdroplogin rose
```

```
call sp_iqdroplogin ('rose')
```

参照：
- sp_iqaddlogin プロシージャ (381 ページ)

sp_iqemptyfile プロシージャ

dbfile を空にし、dbfile 内のオブジェクトを、同じ DB 領域にある別の使用可能な読み込み／書き込み dbfile に移動します。

構文
sp_iqemptyfile (*logical-file--name*)

パーミッション
DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

説明
sp_iqemptyfile は、dbfile を空にします。**sp_iqemptyfile** プロシージャを実行するには、DB 領域を読み込み専用にする必要があります。このプロシージャは、ファイル内のオブジェクトを、同じ DB 領域にある別の使用可能な読み込み／書き込み dbfile に移動します。使用可能な他の読み込み／書き込み dbfile がない場合は、Sybase IQ によってエラー・メッセージが表示されます。

注意：マルチプレックス環境では、コーディネータで **sp_iqemptyfile** のみを実行できます。プロシージャを正常に完了するには、1 つの読み込み／書き込み DB 領域が使用可能である必要があります。

例
次の例では、dbfile **dbfile1** を空にします。

```
sp_iqemptyfile 'dbfile1'
```

sp_iqestjoin プロシージャ

指定したテーブルのジョイン・インデックスを作成するために必要な領域を見積もります。

構文

```
sp_iqestjoin ( table1_name, table1_row_#, table2_name,
table2_row_#, relation, iq_page_size )
```

パーミッション
DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

説明
ジョインするテーブルに基づいて、ジョイン・インデックスで使用される領域の見積もりを返します。このプロシージャでは、指定された IQ ページ・サイズのデフォルト・ブロック・サイズでデータベースが作成されていることを前提として計算します (それ以外の場合、正確な値が返されません)。

修飾されていないテーブル名を指定した場合、ジョインするテーブルを所有するユーザであることを確認します。テーブルの所有者でないユーザは、各テーブルに、'owner.tablename' のような修飾されたテーブル名を指定します。

表 129 : sp_iqestjoin のパラメータ

名前	データ型	説明
table1_name	char(256)	最初にジョインするテーブルの名前。
table1_row_#	int	最初のテーブル内で、ジョイン対象となるロー数。
table2_name	char(256)	2 番目にジョインするテーブルの名前。
table2_row_#	int	2 番目のテーブル内で、ジョイン対象となるロー数。

名前	データ型	説明
relation	char(9)	ジョインのタイプ。"one>>many" または "one>>one" (単語と演算子の間にスペースは入らない)。デフォルトは "one>>many" です。
iq_page_size	smallint	データベースの IQ セグメント用に定義されているページ・サイズ。2 の累乗で、範囲は 1024 ～ 524288、デフォルトは 131072 です。

例

```
call sp_iqestjoin ( 'Customers', 1500000, 'SalesOrders', 15000000,
'one>>many', 65536 )
```

Cases	Indexsize	Create time	Msg
Table1:Customers			
Rows: 1500000			
Columns:			
8			
Width:			
223			
Table2:SalesOrders			
Rows: 15000000			
Columns:			
9			
Width:			
134			
IQpagesize:			
65536			
Min Case	48001024	3h0m/CPU	
Max Case	95449088	9h6m/CPU	
Avg Case	70496256	5h53m/CPU	

sp_iqestdbspaces プロシージャ

指定した合計インデックス・サイズに必要な DB 領域の数とサイズを見積もります。

構文

```
sp_iqestdbspaces ( db_size_in_bytes, iq_page_size,
min_#_of_bytes, max_#_of_bytes )
```

パーミッション

DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

説明

データベース・サイズ、IQ ページ・サイズ、DB 領域セグメントあたりのバイト数の範囲に基づいて、DB 領域セグメントの数とサイズに関する情報を表示します。このプロシージャでは、指定された IQ ページ・サイズのデフォルト・ブロック・サイズでデータベースが作成されていることを前提として計算します (それ以外の場合、正確な値が返されません)。

表 130 : sp_iqestdbspaces のパラメータ

名前	データ型	説明
db_size_in_bytes	decimal(16)	データベースのサイズ (バイト)。
iq_page_size	smallint	データベースの IQ セグメント用に定義されているページ・サイズ。2 の累乗で、範囲は 65536 ～ 524288、デフォルトは 131072 です。
min_#_of_bytes	int	dbspace セグメントあたりの最小バイト数。デフォルトは 20,000,000 (20MB) です。
max_#_of_bytes	int	DB 領域セグメントあたりの最大バイト数。デフォルトは 2,146,304,000 (2.146GB) です。

使用法

sp_iqestdbspaces は、データのユニーク性に応じて、4 種類の推奨を表示します。

表 131 : 推奨事項

推奨	説明
min	データの差が少ない場合は、**min** で推奨されているサイズの DB 領域セグメントだけを作成することもできます。これらの推奨は、最小限の差を持つデータを最大限に圧縮した場合のものです。
avg	データの差が平均的であれば、**min** で推奨されているとおりに DB 領域セグメントを作成し、**avg** で推奨されているサイズで追加のセグメントを作成します。
max	データの差が大きい (ユニークな値が多い) 場合は、**min**、**avg**、**max** で推奨されているとおりに DB 領域セグメントを作成します。
spare	データ内のユニークな値の数がわからない場合は、**min**、**avg**、**max**、**spare** で推奨されているとおりに DB 領域セグメントを作成します。データをロードした後で、使用していないセグメントはいつでも削除できますが、作成するセグメントが少なすぎると時間がかかります。

sp_iqestdbspaces プロシージャの例

sp_iqestdbspaces を使用する場合は、この例を参照してください。

```
sp_iqestdbspaces 12000000000, 65536, 500000000, 2146304000
```

dbspace files	種類	Size	Msg
1	min	2146304000	
2	min	2146304000	
3	min	507392000	
4	avg	2146304000	
5	max	2053697536	
6	spare	1200001024	

この例では、12GB のデータベースに必要な DB 領域セグメントのサイズと数を見積もります。Sybase IQ では、データ間の差が少ない場合、最大限の圧縮をするためには最低でも 3 つのセグメント (**min** と表示) を作成することをおすすめします。データの差が標準的であれば、もう 1 つセグメント (**avg** と表示) を作成します。データの差が大きい (一意の値が多く、高度なインデックス化を必要とする) 場合は、さらに 1 つのセグメント (**max** と表示) を作成する必要があります。1200001024 バイトの予備のセグメントを作成すれば、最初のロードを必ず成功させることができます。 データベースをロードした後で、使用していない DB 領域セグメントはすべて削除できます。

sp_iqestdbspaces を他のシステム・ストアド・プロシージャと一緒に使用する

sp_iqestdbspaces に必要なパラメータ *db_size_in_bytes* を提供するには、2 つのストアド・プロシージャを実行する必要があります。

sp_iqestdbspaces の結果は、インデックスの平均サイズを基にして見積もっただけのものです。実際のサイズは、テーブルに格納されたデータ、特に、データ間の差がどれだけあるかに依存します。

Sybase では、予備の DB 領域セグメントを作成することを強くおすすめしています。使用しなかった場合は、後から削除できます。

1. 頻繁にジョインすることが予想されるテーブルの組み合わせすべてに対して、**sp_iqestjoin** を実行します。
2. テーブルの各ペアに対して、推奨されたインデックス・サイズのいずれかを選択します。
3. すべてのテーブルに対して選択したインデックス・サイズを合計します。
4. すべてのテーブルに対して **sp_iqestspace** を実行します。
5. **sp_iqestspace** が返したすべての RAW DATA インデックス・サイズを合計します。
6. 手順 3 の合計を手順 5 の合計に加え、インデックス・サイズの合計を出します。
7. 手順 6 で出したインデックス・サイズの合計を、**sp_iqestdbspaces** の *db_size_in_bytes* パラメータとして使用します。

sp_iqestspace プロシージャ

テーブル内のロー数に基づいて、インデックスの作成に必要な領域を見積もります。

構文

```
sp_iqestspace ( table_name, #_of_rows, iq_page_size )
```

パーミッション

DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

説明

基本となるデータベース・テーブルのロー数と、データベースの IQ ページ・サイズに基づいて、データベースに必要な領域の見積もりを表示します。このプロシージャでは、指定された IQ ページ・サイズのデフォルト・ブロック・サイズでデータベースが作成されていることを前提として計算します (それ以外の場合、正確な値が返されません)。次の表に、**sp_iqestspace** パラメータを示します。

表 132 : sp_iqestspace のパラメータ

名前	データ型	説明
table_name	char(256)	テーブルの名前。
#_of_rows	int	テーブルにあるロー数。
iq_page_size	smallint	データベースの IQ セグメント用に定義されているページ・サイズ。2 の累乗で、範囲は 65536 ～ 524288、デフォルトは 131072 です。

sp_iqevent プロシージャ

システム・イベントおよびユーザ定義イベントに関する情報を表示します。

構文

sp_iqevent [*event-name*], [*event-owner*], [*event-type*]

パーミッション

DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

使用法

表 133 : パラメータ

パラメータ	説明
event-name	イベント名。
event-owner	イベントの所有者。
event-type	イベントのタイプです。指定できる値は次のとおりです。 • **SYSTEM**： システム・イベント (ユーザ SYS または dbo が所有するイベント) に関する情報のみを表示します。 • **ALL**： ユーザ・イベントおよびシステム・イベントに関する情報を表示します。 • その他の値：ユーザ・イベントに関する情報を表示します。

sp_iqevent プロシージャは、パラメータなしで呼び出せます。パラメータを指定しない場合、ユーザ・イベント (dbo または SYS が所有していないイベント) に関する情報のみがデフォルトで表示されます。

最初の 2 つのパラメータのいずれかを指定せずに、その次のパラメータを指定する場合は、指定しないパラメータの位置に NULL を指定する必要があります。た

とえば、sp_iqevent NULL, NULL, SYSTEM および sp_iqevent NULL, user1 とします。

表 134 : sp_iqevent の使用例

構文	出力
sp_iqevent	データベース内のすべてのユーザ・イベントに関する情報を表示します。
sp_iqevent e1	イベント e1 に関する情報を表示します。
sp_iqevent non_existing_event	イベント non_existing_event が存在しないため、ローは返されません。
sp_iqevent NULL, DBA	DBA が所有するすべてのイベントに関する情報を表示します。
sp_iqevent e1, DBA	DBA が所有するイベント e1 に関する情報を表示します。
sp_iqevent ev_iqbegintxn	ev_iqbegintxn はシステム定義のイベントです。ev_iqbegintxn という名前のユーザ定義イベントが存在しない場合、ローは返されません (デフォルトでは、ユーザ定義イベントのみが返されます)。
sp_iqevent ev_iqbegintxn, dbo	イベント ev_iqbegintxn はユーザ・イベントではないため、ローは返されません (デフォルトでは、ユーザ・イベントのみが返されます)。
sp_iqevent NULL, NULL, SYSTEM	システム定義のイベント (dbo または SYS が所有するイベント) に関する情報を表示します。
sp_iqevent ev_iqbegintxn, NULL, SYSTEM	システム・イベント ev_iqbegintxn に関する情報を表示します。
sp_iqevent ev_iqbegintxn, dbo, ALL	dbo が所有するシステム・イベント ev_iqbegintxn に関する情報を表示します。

説明

sp_iqevent ストアド・イベントは、データベース内のイベントに関する情報を表示します。1 つ以上のパラメータを指定した場合、指定したパラメータによって結果がフィルタされます。たとえば、*event-name* を指定した場合、指定のイベントに関する情報のみが表示されます。*event-owner* を指定した場合、**sp_iqevent** は指定の所有者が所有するイベントに関する情報のみを返します。パラメータを指定しない場合、**sp_iqevent** はデータベース内のすべてのユーザ・イベントに関する情報を表示します。

sp_iqevent プロシージャは、次のカラムに情報を返します。

表 135 : sp_iqevent のカラム

カラム名	説明
event_name	イベント名。
event_owner	イベントの所有者。
event_type	システム・イベントの場合、SYSEVENTTYPE システム・テーブルに挙げられたイベント・タイプ。
enabled	イベントの発生が許可されているかどうかを示す (Y/N)
action の値	イベント・ハンドラの定義
condition	イベント・ハンドラの起動を制御するのに使用される **WHERE** 条件
location	イベントの発生が許可されている場所 • C = 統合 (consolidated) • R = リモート (remote) • A = すべて (all)
remarks	コメント文字列。

例
ユーザ定義イベント `e1` に関する情報を表示します。

```
sp_iqevent e1
event_name    event_owner    event_type    enabled    action
e1            DBA           (NULL)         Y          (NULL)
condition    location    remarks (NULL)       A           (NULL)
```

すべてのシステム・イベントに関する情報を表示します。

```
sp_iqevent NULL, NULL, SYSTEM

event_name      event_owner    event_type       enabled   action
ev_iqbegintxn   dbo            IQTLVAvailable   Y         begin call
                                                       dbo.sp_iqlog...
ev_iqmpxcompact dbo          (NULL)           N        begin Declare
                                                        _Catalog...

condition    location    remarks
(NULL)       A           (NULL)
(NULL)       A           (NULL)
```

参照：

- sp_iqcolumn プロシージャ (403 ページ)

- sp_iqconstraint プロシージャ (410 ページ)
- sp_iqdatatype プロシージャ (419 ページ)
- sp_iqhelp プロシージャ (448 ページ)
- sp_iqindex および sp_iqindex_alt プロシージャ (456 ページ)
- sp_iqjoinindex プロシージャ (471 ページ)
- sp_iqpkeys プロシージャ (491 ページ)
- sp_iqprocparm プロシージャ (496 ページ)
- sp_iq_reset_identity プロシージャ (505 ページ)
- sp_iqtable プロシージャ (525 ページ)
- sp_iqview プロシージャ (541 ページ)

sp_iqfile プロシージャ

DB 領域の各 dbfile についての詳細情報を表示します。

構文

```
sp_iqfile [ dbspace-name ]
```

パーミッション

DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

説明

sp_iqfile は、DB 領域の各 dbfile のデータの使用率、プロパティ、タイプを表示します。この情報を使用して、データの移動が必要かどうかを判断できます。また、移動されたデータに関しては、旧バージョンの割り付けが解除されているかどうかを確認できます。

sp_iqfile は、次の情報を表示します。

表 136 : sp_iqfile のカラム

カラム名	説明
DBSpaceName	**CREATE DBSPACE** 文で指定された DB 領域の名前。**CREATE DATABASE...CASE IGNORE** または **CASE RESPECT** の指定に関係なく、DB 領域名の大文字と小文字は常に区別されません。
DBFileName	論理ファイル名。
Path	物理ファイルまたはロー・パーティションの場所。
SegmentType	DB 領域のタイプ (MAIN または TEMPORARY)。

カラム名	説明
RWMode	DB 領域のモード。読み込み／書き込み (RW) または読み込み専用 (RO)。
オンライン	T (オンライン) または F (オフライン)。
使用法	DB 領域のこのファイルで現在使用されている DB 領域の割合。
DBFileSize	ファイルまたはロー・パーティションの現在のサイズ。ロー・パーティションでは、このサイズ値は物理サイズよりも小さくなる場合があります。
予約	DB 領域のこのファイルに追加できる予約領域。
StripeSize	ディスク・ストライピングが有効になっている場合、次のファイルに移動するまでにファイルに書き込まれたデータの量。
BlkTypes	ユーザ・データと内部システム構造が使用している領域。
FirstBlk	ファイルに割り当てられている最初の IQ ブロック番号。
LastBlk	ファイルに割り当てられている最後の IQ ブロック番号。
OkToDrop	ファイルを削除できる場合は ‘Y’、それ以外の場合は ‘N’。

以下に、ブロック・タイプ識別子の値を示します。

表 137 : sp_iqfile のブロック・タイプ

識別子	ブロック・タイプ
A	アクティブなバージョン
B	バックアップ構造
C	チェックポイント・ログ
D	データベースの識別情報
F	フリー・リスト
G	グローバル・フリー・リスト・マネージャ
H	フリー・リストのヘッダ・ブロック
I	インデックス・アドバイスの格納
M	マルチプレックス CM*
O	旧バージョン
T	テーブルの使用

識別子	ブロック・タイプ
U	インデックスの使用
N	カラムの使用
X	チェックポイントでの削除

*マルチプレックス・コミット ID ブロック (実際は 128 ブロック) は、シンプレックス・データベースで使用されていない場合でも、すべての IQ データベースに存在します。

例
DB 領域のファイルに関する情報を表示します。

```
sp_iqfile;
```

```
sp_iqfile;
DBSpaceName,DBFileName,Path,SegmentType,RWMode,Online,
Usage,DBFileSize,Reserve,StripeSize,BlkTypes,FirstBlk,
LastBlk,OkToDrop

'IQ_SYSTEM_MAIN','IQ_SYSTEM_MAIN','/sun1-c1/users/smith/mpx/m/
mpx_db.iq','MAIN','RW','T','21','
2.92G','0B','1K','1H,76768F,32D,19A,185O,128M,34B,32C'
,1,384000,'N'

'mpx_main1','mpx_main1','/sun1-c1/users/smith/mpx/m/
mpx_main1.iq','MAIN','RW','T','1'
,'100M','0B','1K','1H',1045440,1058239,'N'

'IQ_SHARED_TEMP','sharedfile1_bcp','/sun1-c1/users/smith/mpx/m/
f1','SHARED_TEMP','RO','T','0',
'50M','0B','1K','1H',1,6400,'N'

'IQ_SHARED_TEMP','sharedfile2_bcp','/sun1-c1/users/smith/mpx/m/
f2','SHARED_TEMP','RO','T','0',
'50M','0B','1K','1H',1045440,1051839,'N'

'IQ_SYSTEM_TEMP','IQ_SYSTEM_TEMP','/sun1-c1/users/smithmpx/m/
mpx_db.iqtmp','TEMPORARY','RW',
'T','1','2.92G','0B','1K','1H,64F,33A',1,384000,'N'
```

sp_iqhelp プロシージャ

システムおよびユーザ定義のオブジェクトおよびデータ型に関する情報を表示します。

構文

sp_iqhelp [*obj-name*], [*obj-owner*], [*obj-category*], [*obj-type*]

パーミッション
DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

使用法

表 138 : パラメータ

パラメータ	説明
obj-name	オブジェクト名。
obj-owner	オブジェクトの所有者。
obj-category	オブジェクトのカテゴリを指定するオプション・パラメータ。 カラム、制約、およびインデックスは、テーブルに関連付けられており、直接クエリできません。テーブルをクエリすると、そのテーブルと関連するカラム、インデックス、および制約に関する情報が表示されます。 指定したオブジェクト・カテゴリが許可された値ではない場合、“Invalid object category” エラーが返されます。
obj-type	オブジェクトのタイプ。指定できる値は次のとおりです。 • **SYSTEM**： システム・オブジェクト (ユーザ SYS または dbo が所有するオブジェクト) に関する情報のみを表示します。 • **ALL**： すべてのオブジェクトに関する情報を表示します。 デフォルトでは、システム・オブジェクト以外のオブジェクトに関する情報のみが表示されます。指定したオブジェクト・タイプが **SYSTEM** または **ALL** でない場合、“Invalid object type” エラーが返されます。

表 139 : sp_iqhelp obj-category のパラメータ値

object-type パラメータ	内容
“table”	オブジェクトはベース・テーブルです。
“view”	オブジェクトはビューです。
“procedure”	オブジェクトはストアド・プロシージャまたは関数です。
“event”	オブジェクトはイベントです。
“datatype”	オブジェクトはシステムまたはユーザ定義のデータ型です。

sp_iqhelp プロシージャは、パラメータなしで呼び出せます。パラメータを指定しない場合、**sp_iqhelp** はデータベース内のすべての独立したオブジェクト、つまりベース・テーブル、ビュー、ストアド・プロシージャ、関数、イベント、データ型に関する情報を表示します。

最初の 3 つのパラメータのいずれかを指定せずに、その次のパラメータを指定する場合には、指定しないパラメータの位置に NULL を入力する必要があります。たとえば、`sp_iqhelp NULL, NULL, NULL, SYSTEM` および `sp_iqhelp NULL, user1, “table”` とします。

obj-category パラメータは、NULL 以外の場合、一重引用符または二重引用符で囲みます。

sp_iqhelp で指定の記述を満たすオブジェクトがデータベースに見つからない場合、エラー `“No object found for the given description”` が返されます。

表 140 : sp_iqhelp の使用例

構文	出力
sp_iqhelp	データベース内のすべてのユーザ定義テーブル、ビュー、プロシージャ、イベント、およびデータ型に関する要約情報を表示します。
sp_iqhelp t1, u1, “table”	ユーザ `u1` が所有するテーブル `t1` に関する情報と、`t1` に関連付けられているカラム、インデックス、制約に関する情報を表示します。
sp_iqhelp NULL, u1, “view”	ユーザ `u1` が所有するビュー `v1` に関する情報と、`v1` に関連付けられているカラム、インデックス、制約に関する情報を表示します。
sp_iqhelp sp2	プロシージャ `sp2` と `sp2` のパラメータに関する情報を表示します。
sp_iqhelp e1	イベント `e1` に関する情報を表示します。
sp_iqhelp dt1	データ型 `dt1` に関する情報を表示します。
sp_iqhelp NULL, NULL, NULL, SYSTEM	すべてのシステム・オブジェクト (`dbo` または `SYS` が所有するオブジェクト) に関する要約情報を表示します。
sp_iqhelp non_existing_obj	オブジェクト **non_existing_obj** が存在しないため、エラー `“Object ‘non_existing_obj’ not found”` が返されます。

構文	出力
sp_iqhelp NULL, non_existing_user	ユーザ **non_existing_user** が存在しないため、エラー `“User ‘non_existing_user’ not found”` が返されます。
sp_iqhelp t1, NULL, “apple”	**“apple”** は *obj-category* に入力できない値なため、エラー `“Invalid object category ‘apple’”` が返されます。
sp_iqhelp t1, NULL, NULL, “USER”	**“USER”** は *obj-type* に入力できない値なため、エラー `“Invalid object type ‘USER’”` が返されます。

説明

sp_iqhelp ストアド・プロシージャは、IQ データベース内のシステム、ユーザ定義オブジェクト、およびユーザ定義データ型に関する情報を表示します。**sp_iqhelp** でサポートされるオブジェクトは、テーブル、ビュー、カラム、インデックス、ジョイン・インデックス、制約、ストアド・プロシージャ、関数、イベント、データ型です。

1 つ以上のパラメータを指定した場合、指定したパラメータによって結果がフィルタされます。たとえば、*obj-name* を指定した場合、指定のオブジェクトに関する情報のみが表示されます。*obj-owner* を指定した場合、**sp_iqhelp** は指定の所有者が所有するオブジェクトに関する情報のみを返します。パラメータを指定しない場合、**sp_iqhelp** はデータベース内のすべてのユーザ定義テーブル、ビュー、プロシージャ、イベント、データ型に関する要約情報を表示します。

sp_iqhelp プロシージャは、指定のパラメータが複数のオブジェクトまたは 1 つのオブジェクトのいずれに一致するかどうかで、要約情報または詳細情報のいずれかを返します。**sp_iqhelp** の出力カラムは、ストアド・プロシージャ **sp_iqtable**、**sp_iqindex**、**sp_iqview**、**sp_iqconstraint** によって表示されるカラムと同様のものです。

複数のオブジェクトが指定の **sp_iqhelp** パラメータに一致する場合、**sp_iqhelp** はそれらのオブジェクトに関する要約情報を表示します。

表 141 : sp_iqhelp 要約情報

オブジェクト・タイプ	表示されるカラム
ベース・テーブル	table_name、table_owner、server_type、location、table_constraints、remarks
ビュー	view_name、view_creator、view_def、server_type、location、remarks

オブジェクト・タイプ	表示されるカラム
ストアド・プロシージャ	proc_name、proc_creator、proc_defn、replicate、srvid、remarks
function	proc_name、proc_creator、proc_defn、replicate、remarks
event	event_name、event_creator、enabled、location、event_type、action、external_action、condition、remarks
システムおよびユーザ定義のデータ型	type_name、creator、nulls、width、scale、default、check

1 つのオブジェクトが指定の **sp_iqhelp** パラメータに一致する場合、**sp_iqhelp** はオブジェクトに関する詳細情報を表示します。

表 142 : sp_iqhelp 詳細情報

オブジェクト・タイプ	説明	カラム
テーブル	指定のベース・テーブルとそのカラム、インデックス、制約、およびジョイン・インデックス (テーブルがジョイン・インデックスに関与する場合) に関する情報を表示します。	• テーブル・カラム：table_name、table_owner、server_type、location、table_constraints、remarks • カラムのカラム：column_name、domain_name、width, scale、nulls、default、check、pkey、user_type、cardinality、est_cardinality、remarks • インデックス・カラム：index_name、column_name、index_type、unique_index、location、remarks • 制約カラム：constraint_name (役割)、column_name、index_name、constraint_type、foreigntable_name、foreigntable_owner、foreigncolumn_name、foreignindex_name、location • ジョイン・インデックス・カラム：joinindex_name、creator、left_table_name、left_table_owner、left_column_name、join_type、right_table_name、right_table_owner、right_column_name、key_type、valid、remarks

オブジェクト・タイプ	説明	カラム
ビュー	指定のビューとそのカラムに関する情報を表示します。	• ビュー・カラム：view_name、view_creator、view_def、server_type、location、remarks • カラムのカラム：column_name、domain_name、width, scale、nulls、default、check、pkey、user_type、cardinality、est_cardinality、remarks
ストアド・プロシージャ	指定のプロシージャとそのパラメータに関する情報を表示します。	• プロシージャ・カラム：proc_name、proc_creator、proc_defn、replicate、srvid、remarks • パラメータ・カラム： parameter_name、type、width、scale、default、mode
function	指定の関数とそのパラメータに関する情報を表示します。	• 関数カラム：proc_name、proc_creator、proc_defn、replicate、srvid、remarks • パラメータ・カラム： parameter_name、type、width、scale、default、mode
event	指定のイベントに関する情報を表示します。	• イベント・カラム：event_name、event_creator、enabled、location、event_type、action、external_action、condition、remarks
データ型	指定のデータ型に関する情報を表示します。	• データ型のカラム：type_name、creator、nulls、width、scale、default、check

注意： システム・プロシージャのプロシージャ定義 (proc-defn) は暗号化され、ビューに表示されません。

各出力カラムの詳細については、関連するストアド・プロシージャを参照してください。たとえば、このテーブル・カラムの詳細については、**sp_iqtable** プロシージャを参照してください。

例
テーブル `sale` に関する詳細情報を表示します。

```
sp_iqhelp sale
```

```
Table_name Table_owner Server_type Location dbspace_id isPartitioned
 table_constraints
```

```
==========  ===========  ==========  =======  ==
======= =============
sale      DBA         IQ           Main    16387      N

Remarks  table_constraints
=======  ================== (NULL)   (NULL)

column_name domain_name width scale nulls default cardinality
==========  =========== ===== ===== ===== ======= ===========
prod_id     integer     4     0     Y    (NULL)   0
month_num   integer     4     0     Y    (NULL)   0
rep_id      integer     4     0     Y    (NULL)   0
sales       integer     4     0     Y    (NULL)   0

  est_cardinality    isPartitioned    remarks    check
  ===============    =============    =======    =====
  0                  N                (NULL)     (NULL)
  0                  N                (NULL)     (NULL)
  0                  N                (NULL)     (NULL)
  0                  N                (NULL)     (NULL)

index_name           column_name  index_type  unique_index  location
==========           ===========  ========== ===========   ========
ASIQ_IDX_T463_C2_FP  month_num    FP          N             Main
ASIQ_IDX_T463_C1_FP  prod_id      FP          N             Main
ASIQ_IDX_T463_C3_FP  rep_id       FP          N             Main
ASIQ_IDX_T463_C4_FP  sales        FP          N             Main

  remarks
  =======
  (NULL)
  (NULL)
  (NULL)
  (NULL)
```

プロシージャ sp_customer_list に関する詳細情報を表示します。

```
sp_iqhelp sp_customer_list
proc_name    proc_owner    proc_defn
==========  ===========    =========
sp_customer_list    DBA    create procedure DBA.sp_customer_list()
                           result(id integer company_name char(35))
                           begin
                           select id company_name from Customers
                           end

  replicate    srvid    remarks
  =========    =====    =======
  N            (NULL)   (NULL)

parm_name      parm_type  parm_mode  domain_name  width  scale
=========      =========  =========  ===========  =====  =====
id             result     out        integer      4      0
company_name   result     out        char         35     0

  default
```

```
=======
(NULL)
```

参照：

- sp_iqcolumn プロシージャ (403 ページ)
- sp_iqconstraint プロシージャ (410 ページ)
- sp_iqdatatype プロシージャ (419 ページ)
- sp_iqevent プロシージャ (443 ページ)
- sp_iqindex および sp_iqindex_alt プロシージャ (456 ページ)
- sp_iqjoinindex プロシージャ (471 ページ)
- sp_iqpkeys プロシージャ (491 ページ)
- sp_iqprocparm プロシージャ (496 ページ)
- sp_iq_reset_identity プロシージャ (505 ページ)
- sp_iqtable プロシージャ (525 ページ)
- sp_iqview プロシージャ (541 ページ)

sp_iqhelp の Adaptive Server Enterprise との互換性

Sybase IQ**sp_iqhelp** ストアド・プロシージャは、Adaptive Server Enterprise**sp_help** プロシージャと同様のもので、SYSOBJECTS システム・テーブルに挙げられたすべてのデータベース・オブジェクトと、システムおよびユーザ定義のデータ型に関する情報を表示します。

サポートされるオブジェクト型とオブジェクトのネームスペースについて、Sybase IQ のアーキテクチャは Adaptive Server Enterprise とは異なります。 Adaptive Server Enterprise では、すべてのオブジェクト (テーブル、ビュー、ストアド・プロシージャ、ログ、規則、デフォルト、トリガ、検査制約、参照制約、テンポラリ・オブジェクト) が SYSOBJECTS システム・テーブルに保存され、同じネームスペースに属します。 Sybase IQ がサポートするオブジェクト (テーブル、ビュー、ストアド・プロシージャ、イベント、プライマリ・キー、一意性制約、検査制約、参照制約) は異なるシステム・テーブルに保存され、異なるネームスペースに属します。 たとえば、Sybase IQ ではテーブルの名前がイベントまたはストアド・プロシージャの名前と同じであっても構いません。

Sybase IQ と Adaptive Server Enterprise のアーキテクチャの相違のために、Sybase IQ**sp_iqhelp** のサポートするオブジェクト・タイプと構文は、Adaptive Server Enterprise**sp_help** のサポートするオブジェクトと構文とは異なります。ただし、両ストアド・プロシージャが表示するデータベース・オブジェクトに関する情報の種類は、同様のものです。

sp_iqindex および sp_iqindex_alt プロシージャ

インデックスに関する情報をリストします。

構文 1

```
sp_iqindex ( [ table_name ],[column_name ],[table_owner ] )
```

構文 2

```
sp_iqindex [table_name='tablename' ],
[column_name='columnname' ],[table_owner='tableowner' ]
```

構文 3

```
sp_iqindex_alt ( [ table_name ],[column_name ],[table_owner ] )
```

構文 4

```
sp_iqindex_alt [table_name='tablename' ],
[column_name='columnname' ],[table_owner='tableowner' ]
```

パーミッション

DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

使用法

表 143 : パラメータ

パラメータ	説明
構文 1	最初の 2 つのパラメータのいずれかを指定せずに、その次のパラメータを指定する場合は、指定しないパラメータの位置に NULL を指定する必要があります。たとえば、`sp_iqindex NULL,NULL,DBA` および `sp_iqindex Departments,NULL,DBA` とします。
構文 2	パラメータは、どのような順序で指定しても構いません。一重引用符で囲みます。
構文 3 および 4	複数カラムのインデックスがある場合、出力は多少異なります。構文 1 および 2 と同じオプションを使用できます。

説明

データベース内のインデックスについての情報を表示します。いずれかのパラメータを指定すると、そのテーブル、カラム、または指定したユーザが所有するテーブルからのみインデックスが返されます。複数のパラメータを指定すると、

指定されたすべてのパラメータにより結果がフィルタされます。パラメータを指定しない場合、データベース内のすべてのテーブルのすべてのインデックスが返されます。

表 144 : sp_iqindex および sp_iqindex_alt カラム

カラム名	説明
table_name	テーブルの名前。
table_owner	テーブルの所有者
column_name	カラムの名前。複数カラムのインデックスでは複数の名前が表示されます。
index_type	インデックスのタイプの省略形 (**HG**、**LF** など)。
index_name	インデックス名。
unique_index	ユニーク・インデックスの場合は 'U'、そうでない場合は 'N'。
location	TEMP = IQ テンポラリ・ストア、MAIN = IQ ストア、SYSTEM = カタログ・ストア。
remarks	**COMMENT** 文で追加されたユーザ・コメント

sp_iqindex は常にインデックスごとに 1 行を生成します。**sp_iqindex_alt** は、複数カラムのインデックスがある場合、カラムごと、インデックスごとに 1 行を生成します。

参照：

- sp_iqcolumn プロシージャ (403 ページ)
- sp_iqconstraint プロシージャ (410 ページ)
- sp_iqdatatype プロシージャ (419 ページ)
- sp_iqevent プロシージャ (443 ページ)
- sp_iqhelp プロシージャ (448 ページ)
- sp_iqjoinindex プロシージャ (471 ページ)
- sp_iqpkeys プロシージャ (491 ページ)
- sp_iqprocparm プロシージャ (496 ページ)
- sp_iq_reset_identity プロシージャ (505 ページ)
- sp_iqtable プロシージャ (525 ページ)
- sp_iqview プロシージャ (541 ページ)

sp_iqindex および sp_iqindex_alt のプロシージャ例

sp_iqindex および **sp_iqindex_alt** を使用する場合は、この例を参照してください。

次の構文ではいずれも、DepartmentID という名前のカラムのすべてのインデックスが返されます。

```
call sp_iqindex (NULL,'DepartmentID')
```

```
sp_iqindex column_name='DepartmentID'
```

table_name	table_owner	column_name	index_type	index_name	unique_index	location	dbspace_id	remarks
Departments	GROUPO	DepartmentID	**FP**	ASIQ_IDX_T201_C1_FP	N	Main	16387	(NULL)
Departments	GROUPO	DepartmentID	**HG**	ASIQ_IDX_T201_C1_HG	U	Main	16387	(NULL)
Employees	GROUPO	DepartmentID	**FP**	ASIQ_IDX_T202_C5_FP	N	Main	16387	(NULL)

次の構文ではいずれも、テーブル所有者 GROUPO が所有するテーブル Departments の、すべてのインデックスが返されます。

```
sp_iqindex Departments,NULL,GROUPO
```

```
sp_iqindex table_name='Departments',table_owner='DBA'
```

table_name	table_owner	column_name	index_type	index_name	unique_index	location	dbspace_id	remarks
Departments	GROUPO	Department-HeadID	**FP**	ASIQ_IDX_T201_C3_FP	N	Main	16387	(NULL)
Departments	GROUPO	DepartmentID	**FP**	ASIQ_IDX_T201_C1_FP	N	Main	16387	(NULL)

table_name	table_owner	column_name	index_type	index_name	unique_index	location	dbspace_id	remarks
Departments	GROUPO	DepartmentID	**HG**	ASIQ_IDX_T201_C1_HG	U	Main	16387	(NULL)
Departments	GROUPO	Department-Name	**FP**	ASIQ_IDX_T201_C2_FP	N	Main	16387	(NULL)

sp_iqindex_alt の次の構文ではいずれも、カラム `City` を含む `Employees` テーブルのインデックスが返されます。インデックス `emp_loc` は、カラム `City` と `State` の複数カラム・インデックスです。**sp_iqindex_alt** では、複数カラム・インデックスのカラムごとに 1 ローが表示されます。

```
sp_iqindex_alt Employees,City
```

```
sp_iqindex_alt table_name='Employees',                column_
name='City'
```

table_name	table_owner	column_name	index_type	index_name	unique_index	dbspace_id	remarks
Employees	GROUPO	City	**FP**	ASIQ_IDX_T452_C7_FP	N	16387	(NULL)
Employees	GROUPO	City	**HG**	emp_loc	N	16387	(NULL)
Employees	GROUPO	状態	**HG**	emp_loc	N	16387	(NULL)

sp_iqindex の出力は、同じテーブル、同じカラムでも多少異なります。

```
sp_iqindex Employees,City
```

```
sp_iqindex table_name='Employee',column_name='City'
```

table_name	table_owner	column_name	index_type	index_name	unique_index	dbspace_id	location	remarks
Employees	GROUPO	City	**FP**	ASIQ_IDX_T452_C7_FP	N	16387	Main	(NULL)

table_name	table_owner	column_name	index_type	index_name	unique_index	dbspace_id	location	remarks
Employees	GROUPO	City,State	**HG**	emp_loc	N	16387	Main	(NULL)

sp_iqindexadvice プロシージャ

格納されたインデックス・アドバイス・メッセージを表示します。オプションで、アドバイス記憶領域をクリアします。

構文

sp_iqindexadvice ([*resetflag*])

パーミッション

DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

使用法

表 145 : パラメータ

パラメータ	説明
resetflag	呼び出し側からインデックス・アドバイス記憶領域をクリアできます。*resetflag* が 0 以外である場合、最後のローの取得後に、すべてのアドバイスが削除されます。

説明

このプロシージャを使用すると、蓄積されたインデックス・アドバイザのメッセージを SQL でクエリできます。情報は、どのインデックスまたはスキーマの変更が多くのクエリに影響を与えるかを判断するために役立てることができます。

INDEX_ADVISOR カラム：

表 146 : sp_iqindexadvice のカラム

カラム名	説明
Advice	ユニークなアドバイス・メッセージ
NInst	メッセージのインスタンス数
LastDT	アドバイスが生成された最終日時

例

sp_iqindexadvice プロシージャからの出力例を示します。

表 147 : sp_iqindexadvice の出力例

Advice	NInst	LastDT
Add a CMP index on DBA.tb (c2, c3) Predicate:(tb.c2 = tb.c3)	2073	2009-04-07 16:37:31.000
Convert HG index on DBA.tb.c4 to a unique HG	812	2009-04-06 10:01:15.000
Join Key Columns DBA.ta.c1 and DBA.tb.c1 have mismatched data types	911	2009-02-25 20:59:01.000

参照：

- sp_iqcolumnuse プロシージャ (405 ページ)
- sp_iqindexuse プロシージャ (469 ページ)
- sp_iqtableuse プロシージャ (530 ページ)
- sp_iqunusedcolumn プロシージャ (536 ページ)
- sp_iqunusedindex プロシージャ (537 ページ)
- sp_iqunusedtable プロシージャ (538 ページ)
- sp_iqworkmon プロシージャ (547 ページ)

sp_iqindexfragmentation プロシージャ

Sybase IQ インデックスの B-tree、garray、および bitmap 構造内でページ領域が占める割合に関する情報をレポートします。

garray の場合、フィル・パーセンテージの計算では、**GARRAY_FILL_FACTOR_PERCENT** オプションによって制御される garray グループ内の予約領域が考慮されません。

構文

```
dbo.sp_iqindexfragmentation ( 'target ' )
```

```
target: table  table-name | index index-name [...]
```

パーミッション

DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

使用法

表 148 : パラメータ

パラメータ	説明
table-name	ターゲットとなる **table***table-name* では、指定されたテーブル内のデフォルト以外のすべてのインデックスについてレポートされます。
index-name	ターゲットとなる **index***index-name* では、指定されたインデックスについてレポートされます。各 index-name は修飾されたインデックス名です。テーブル内の複数のインデックスを指定することもできますが、指定するインデックスごとに **index** キーワードを繰り返し指定する必要があります。

例

`Customers` テーブル内のユニークでない **HG** インデックスである `cidhg` について、内部インデックスの断片化がレポートされます。

```
dbo.sp_iqindexfragmentation ( 'index customers.cidhg ' )
```

インデックス	インデックス・タイプ	btree ノード・ページ	GARRAY_FILL_FACTOR_PERCENT
dba.customers.cidhg	**HG**	3	75
SQLCODE	0		
フィル・パーセント	btree ページ	garray ページ	bitmap ページ
0 - 10%	0	0	0
11 - 20%	0	0	0
21 - 30%	0	0	0
31-40%	0	0	22
41 - 50%	0	0	0
51 - 60%	0	0	10
61 - 70%	2	0	120
71 - 80%	138	3	64
81 - 90%	24	122	14
91 - 100%	18	1	0

この出力では、ユニークでない **HG** インデックスである cidhg 内の 182 B ツリー・ページのうち、2 ページが 61 ～ 70%、138 ページが 71 ～ 80%、24 ページが 81 ～ 90%、18 ページが 91 ～ 100% 使用されていることがわかります。garray および bitmap ページの利用率も、同様に報告されます。すべてのパーセンテージは、一番近いパーセンテージ・ポイントにトランケートされます。**HG** インデックスでは、GARRAY_FILL_FACTOR_PERCENT オプションの値も表示されます。B-tree を使用するインデックス・タイプでは、ノード (リーフでない) ページの数も表示されます。このインデックス・タイプには、**HG**、**LF**、**WD**、**DATE**、**DTTM** があります。

このインデックスでのストアド・プロシージャの実行中にエラーが発生した場合、SQLCODE は 0 以外の値を取ります。

参照：

- sp_iqindexmetadata プロシージャ (465 ページ)
- sp_iqindexinfo プロシージャ (463 ページ)
- sp_iqindexsize プロシージャ (467 ページ)
- sp_iqrebuildindex プロシージャ (500 ページ)
- sp_iqrowdensity プロシージャ (507 ページ)

sp_iqindexinfo プロシージャ

特定のオブジェクトに使用しているブロックの数を、メインの DB 領域のインデックスごとに表示します。オブジェクトが複数の DB 領域に含まれている場合、**sp_iqindexinfo** は、すべての DB 領域で使用される領域を返します。次に例を示します。

構文

```
sp_iqindexinfo '{ database
| [ table   table-name | index   index-name ] [...] }
[ resources   resource-percent ]'
```

パーミッション

DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

使用法

データベース全体のインデックス情報を要求することも、任意の数のテーブルやインデックス・パラメータを指定して要求することもできます。テーブル名が指定されている場合、**sp_iqindexinfo** はテーブル内のすべてのインデックスに関する情報を返します。インデックス名を指定した場合、そのインデックスに関する情報のみが返されます。

ジョイン・インデックスを名前で指定することはできません。ジョイン・インデックスを表示するには、**database** キーワードを使用します。

指定された *table-name* または *index-name* があいまいな場合、またはオブジェクトが見つからない場合には、エラーが返されます。

マルチプレックス・データベースでは、**sp_iqindexinfo** は、デフォルトで、セカンダリ・ノード上の共有 IQ ストアに関する情報を表示します。個別のテーブルまたはインデックスが指定された場合、表示するストアは自動的に選択されます。

resource-percent は 0 より大きい整数である必要があります。リソースのパーセンテージを使用して CPU の合計使用率を指定すれば、**sp_iqindexinfo** プロシージャによる CPU の使用率を制限できます。

マルチプレックス環境では、このプロシージャを使用できます。詳細については、『Sybase IQ Multiplex の使用』を参照してください。

説明

sp_iqindexinfo では、特定のオブジェクトがどの DB 領域にあるかを DBA に示します。この情報によって、DBA はどの DB 領域を **relocate** モードにしてオブジェクトを移動する必要があるのかを判断できます。

sp_iqindexinfo の結果は、コマンドを実行しているトランザクションから見たバージョンが基準になります。他のバージョンで使用されているブロックは表示されません。

表 149 : sp_iqindexinfo のカラム

カラム名	説明
オブジェクト	テーブル、インデックス、ジョイン・インデックスの名前。
Dbspace_name	DB 領域の名前。
ObjSize	この DB 領域での、このオブジェクトのデータのサイズ。
DBSpPct	このオブジェクトが使用している dbspace のパーセンテージ。
MinBlk	この dbspace で、このオブジェクトが使用する最初のブロック。
MaxBlk	この DB 領域で、このオブジェクトが使用する最後のブロック。DB 領域のサイズを縮小する前に移動が必要なオブジェクトを判断するのに役立ちます。

例

テーブル `t2` に関するインデックス情報を表示します。

```
sp_iqindexinfo 'table t2';
```

オブジェクト	dbspace_name	ObjSize	DBSpPct	MinBlk	MaxBlk
t2	IQ_SYSTEM_MAIN	32K	1	84	107
t2	dbspacedb2	160K	2	1045495	1045556
t2	dbspacedb3	8K	1	2090930	2090930
t2.DBA.ASIQ_IDX_T430_C1_FP	IQ_SYSTEM_MAIN	136K	2	126	321
t2.DBA.ASIQ_IDX_T430_C1_FP	dbspacedb3	152K	2	2091032	2091053
t2.DBA.t2c1hng	dbspacedb2	136K	2	1045537	1045553

参照：

- sp_iqdbspace プロシージャ (424 ページ)
- sp_iqdbspaceinfo プロシージャ (426 ページ)
- sp_iqspaceinfo プロシージャ (511 ページ)
- sp_iqindexmetadata プロシージャ (465 ページ)
- sp_iqindexfragmentation プロシージャ (461 ページ)
- sp_iqindexsize プロシージャ (467 ページ)

sp_iqindexmetadata プロシージャ

指定のインデックスのインデックス・メタデータを表示します。

オプションで、指定したテーブル上のインデックスや、指定した所有者に属するインデックスのみに出力を制限できます。

構文

```
dbo.sp_iqindexmetadata {'index-name'
[ , 'table-name' [ , 'owner-name' ] ] }
```

パーミッション

DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

使用法

テーブル名を指定すると、そのテーブルに属するインデックスに出力が制限されます。所有者名を指定すると、その所有者が所有するインデックスに出力が制限されます。パラメータを省略した場合のデフォルトは NULL です。プロシージャごとに 1 つのインデックスのみを指定できます。

説明
出力の最初のローは、インデックスの所有者名、テーブル名、およびインデックス名です。

それ以降出力されるローは、指定したインデックス・タイプに規定されます。

表 150 : sp_iqindexmetadata の出力ロー

インデックス・タイプ	返されるメタデータ
CMP, DATE, DTTM, TIME	Type、Version
FP	Type、Version、LookupPages、Style、LookupEntries、1stLookupPage、LargeLOBs、SmallLOBs、IQ Unique、LOB Compression (カラムのデータ型が `LONG VARCHAR` または `LONG BINARY` の場合のみ)
HG	Type、Version、Distinct Keys
HNG	Type、Version、BitsPerBlockmap、NumberOfBits
LD	Type、Version<ld>、Version、Distinct Keys
LF	Type、Version、IndexStatus、NumberOfBlockmaps、BitsPerBlockmap、Distinct Keys
WD	Type、Version、KeySize、Delimiters、DelimiterCount、MaxKeyWordLength、PermitEmptyWord

例
次のコマンドを入力すると、**HG** インデックス `hg_index_col54` に関するインデックス情報が表示されます。

```
sp_iqindexmetadata 'hg_index_col54' , 'metal' , 'DBA';
```

'DBA',	'metal'	'hg_index_col54'
'Type',	'HG',	''
'Version',	'2',	''
'Distinct Keys',	'0',	''

参照：

- sp_iqindexfragmentation プロシージャ (461 ページ)
- sp_iqindexinfo プロシージャ (463 ページ)

- sp_iqindexsize プロシージャ (467 ページ)

sp_iqindexsize プロシージャ

指定したインデックスのサイズを返します。

構文

```
sp_iqindexsize [ [ owner.] table.] index_name
```

パーミッション

DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

説明

表 151 : sp_iqindexsize のカラム

カラム名	説明
Username	インデックスの所有者。
Indexname	結果を返すインデックス (テーブル名を含む)。
種類	インデックスのタイプ。
情報	KBytes、Pages、Compressed Pages がレポートされる IQ インデックスのコンポーネント。コンポーネントは、インデックスのタイプによって異なります。たとえば、デフォルト (FP) インデックスには、BARRAY (barray) と Bitmap (bm) コンポーネントが含まれます。Low_Fast (LF) インデックスには、B-tree (bt) と Bitmap (bm) コンポーネントが含まれます。
KBytes	物理オブジェクト・サイズ (キロバイト)。
Pages	メモリ内にオブジェクトを保持するために必要な IQ ページの数。
Compressed Pages	(ディスク上で) オブジェクトを圧縮した場合の IQ ページの数。

インデックスの合計サイズ (バイト、キロバイト) と、Info カラムを返します。Info カラムには、KBytes、Pages、Compressed Pages をレポートする IQ インデックスのコンポーネントが返されます。コンポーネントは、インデックスのタイプによって異なります。たとえば、デフォルト (FP) インデックスには、BARRAY (barray) と Bitmap (bm) コンポーネントが含まれます。Low_Fast (LF) インデックスには、B-tree (bt) と Bitmap (bm) コンポーネントが含まれます。

また、メモリにオブジェクトを保持するために必要なページ数と、(ディスク上で) インデックスを圧縮した場合の IQ ページ数も返します。

このプロシージャには、必ず *index_name* パラメータを指定してください。単一のテーブル内のこのインデックス名のみに結果を絞り込むには、インデックスを指定するときに *owner.table.* を含めます。

例

```
sp_iqindexsize ASIQ_IDX_T452_C19_FP
```

Username	Indexname	種類	情報	KBytes	Pages	Compressed Pages
DBA	Employees.ASIQ_IDX_T452_C19_FP	FP	合計	288	4	2
DBA	Employees.ASIQ_IDX_T452_C19_FP	FP	vdo	0	0	0
DBA	Employees.ASIQ_IDX_T452_C19_FP	FP	bt	0	0	0
DBA	Employees.ASIQ_IDX_T452_C19_FP	FP	garray	0	0	0
DBA	Employees.ASIQ_IDX_T452_C19_FP	FP	bm	136	2	1
DBA	Employees.ASIQ_IDX_T452_C19_FP	FP	barray	152	2	1
DBA	Employees.ASIQ_IDX_T452_C19_FP	FP	dpstore	0	0	0
DBA	Employees.ASIQ_IDX_T452_C19_FP	FP	largelob	0	0	0
DBA	Employees.ASIQ_IDX_T452_C19_FP	FP	txtPst	0	0	0

```
CREATE TEXT INDEX ti ON Employees( Street ) IMMEDIATE
REFRESH;sp_iqindexsize 'ti';
```

Username	Indexname	種類	情報	KBytes	Pages	Compressed Pages
GROUPO	GROUPO.Employees.ti	TEXT	合計	896	12	6

Username	Indexname	種類	情報	KBytes	Pages	Compressed Pages
GROUPO	GROUPO.Employees.ti	TEXT	vdo	0	0	0
GROUPO	GROUPO.Employees.ti	TEXT	bt	304	4	2
GROUPO	GROUPO.Employees.ti	TEXT	garray	152	2	1
GROUPO	GROUPO.Employees.ti	TEXT	bm	136	2	1
GROUPO	GROUPO.Employees.ti	TEXT	barray	152	2	1
GROUPO	GROUPO.Employees.ti	TEXT	dpstore	0	0	0
GROUPO	GROUPO.Employees.ti	TEXT	largelob	0	0	0
GROUPO	GROUPO.Employees.ti	TEXT	txtPst	304	4	2

参照：

- sp_iqindexmetadata プロシージャ (465 ページ)
- sp_iqindexfragmentation プロシージャ (461 ページ)
- sp_iqindexinfo プロシージャ (463 ページ)

sp_iqindexuse プロシージャ

負荷によってアクセスされる二次的な (非 FP) インデックスの使用状況の情報を詳細にレポートします。

構文

```
sp_iqindexuse
```

パーミッション

DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

説明
負荷によってアクセスされる二次的なインデックスごとに、ローが 1 つ表示されます。アクセスされないインデックスは、表示されません。インデックスの使用状況は、オプティマイザ、制約、およびクエリの使用状況別に分割されます。

SYSTEM で作成されたテーブルのインデックスはレポートされません。

表 152 : sp_iqindexuse のカラム

カラム名	説明
IndexName	インデックス名
TableName	テーブル名
所有者	インデックス所有者のユーザ名
UID**	インデックスのユニークな識別子
種類	インデックス・タイプ
LastDT	前回のアクセスの日時
NOpt	メタデータ／ユニーク性のアクセスの数
NQry	クエリ・アクセスの数
NConstraint	ユニークまたは参照の整合性チェックのアクセスの数

**UID はシステムが割り当てた番号であり、インデックスのインスタンスをユニークに識別します (インスタンスはオブジェクト作成時に定義されます)。

例
sp_iqindexuse プロシージャからの出力例を示します。

```
IndexName       TableName  Owner UID Type LastDT              NOpt
 NQry  NConstraint
n_nationkey_hg  nation     DBA   29  HG   20070917
22:08:06~  12    0     12
n_regionkey_hg  nation     DBA   31  HG   20070917
22:08:06~  12    0      0
r_regionkey_hg  region     DBA   47  HG   20070917
22:08:06~  12    0     12
s_suppkey_hg    supplier   DBA   64  HG   20070917
22:08:06~  12    0     12
p_partkey_hg    part       DBA   87  HG   20070917
22:08:06~   6    0      6
s_suppkey_hg    supplier   DBA   64  HG   20070917
22:08:06~  12    0     12
...
```

参照：

- sp_iqcolumnuse プロシージャ (405 ページ)
- sp_iqindexadvice プロシージャ (460 ページ)
- sp_iqtableuse プロシージャ (530 ページ)
- sp_iqunusedcolumn プロシージャ (536 ページ)
- sp_iqunusedindex プロシージャ (537 ページ)
- sp_iqunusedtable プロシージャ (538 ページ)
- sp_iqworkmon プロシージャ (547 ページ)

sp_iqjoinindex プロシージャ

ジョイン・インデックスに関する情報を表示します。

構文

```
sp_iqjoinindex [ left-table-name ], [ left-column-name ], [ left-table-owner ], [ right-table-name ], [ right-column-name ], [ right-table-owner ]
```

パーミッション

DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

使用法

sp_iqjoinindex プロシージャは、パラメータなしで呼び出せます。パラメータを指定しない場合、**sp_iqjoinindex** は IQ ベース・テーブルのすべてのジョイン・インデックスに関する情報を表示します。ジョイン・インデックス・テーブルは常に、IQ ベース・テーブルです。テンポラリ・テーブル、リモート・テーブル、またはプロキシ・テーブルは、ジョイン・インデックス・テーブルとして使用することはできません。

表 153 : パラメータ

パラメータ	説明
left-table-name	ジョイン操作の左側を構成するテーブルの名前。
left-column-name	ジョインの左側の一部を構成するカラムの名前。
left-table-owner	ジョイン操作の左側を構成するテーブルの所有者。
right-table-name	ジョイン操作の右側を構成するテーブルの名前。
right-column-name	ジョインの右側の一部を構成するカラムの名前。
right-table-owner	ジョイン操作の右側を構成するテーブルの所有者。

最初の 5 つのパラメータを指定せずに、その次のパラメータを指定する場合は、指定しないパラメータの位置に NULL を指定する必要があります。たとえば、`sp_iqjoinindex NULL, NULL, NULL, t2, n2, DB'` および `sp_iqjoinindex t1, NULL, NULL, t2` とします。

表 154 : sp_iqjoinindex の使用例

構文	出力
sp_iqjoinindex	すべてのジョイン・インデックスについての情報を表示します。
sp_iqjoinindex t1, NULL, DBA	DBA 所有の `t1` がジョイン演算の左側を構成する、すべてのジョイン・インデックスに関する情報を表示します。
sp_iqjoinindex t2, n1, DBA	ジョインの左側を構成する DBA 所有のテーブル `t2` のカラム `n1` についてのジョイン・インデックス情報を表示します。
sp_iqjoinindex NULL, NULL, DBA, NULL, NULL, DBA	左右のテーブルを DBA が所有する、すべてのジョイン・インデックスに関する情報を表示します。
sp_iqjoinindex NULL, NULL, NULL, t2, NULL, NULL	テーブル `t2` がジョイン演算の右側にある、すべてのジョイン・インデックスに関する情報を表示します。
sp_iqjoinindex t1, n1, DBA, t2, n1, DBA	DBA の所有するテーブル `t1` のカラム `n1` が左側にあり、DBA の所有するテーブル `t2` のカラム `n1` が右側にある、ジョイン・インデックスに関する情報を表示します。
sp_iqjoinindex non_existing_table	テーブル **non_existing_table** が存在しないため、ローは返されません。
sp_iqjoinindex NULL, NULL, non_existing_user	ユーザ **non_existing_user** が存在しないため、ローは返されません。

説明

sp_iqjoinindex ストアド・プロシージャは、データベース内のジョイン・インデックスに関する情報を表示します。1 つ以上のパラメータを指定した場合、指定したパラメータによって結果がフィルタされます。たとえば、*left-table-name* を指定した場合、**sp_iqjoinindex** は該当テーブルがジョインの左側を構成するすべてのジョイン・インデックスを表示します。*left-table-owner* を指定した場合、**sp_iqjoinindex** は左のテーブルが指定の所有者によって所有されているジョイン・インデックスのみを返します。パラメータを指定しない場合、**sp_iqjoinindex** はデータベース内のすべてのジョイン・インデックスに関する情報を表示します。

sp_iqjoinindex プロシージャは、次のカラムに情報を返します。

表 155 : sp_iqjoinindex のカラム

カラム名	説明
joinindex_name	ジョイン・インデックス名。
作成者	ジョイン・インデックスの所有者。
left_table_name	ジョイン操作の左側を構成するテーブルの名前。
left_table_owner	ジョイン操作の左側を構成するテーブルの所有者名。
left_column_name	ジョインの左側の一部を構成するカラムの名前。
join_type	現在サポートしている値は "=" のみ。
right_table_name	ジョイン操作の右側を構成するテーブルの名前。
right_table_owner	ジョイン操作の右側を構成するテーブルの所有者名。
right_column_name	ジョインの右側の一部を構成するカラムの名前。
key_type	次のキーのジョインのタイプを定義します。 • NATURAL： ナチュラル・ジョイン • KEY： キー・ジョイン • ON：左外部ジョイン／右外部ジョイン／フル・ジョイン
有効	このジョイン・インデックスを同期する必要があるかどうかを示します。'Y' の場合は同期の必要はなく、'N' の場合は同期の必要があります。
remarks	コメント文字列。
dbspace_id	指定のジョイン・インデックスが存在する DB 領域の名前。

例

テーブル `t1` がジョイン演算の左側を構成する、ジョイン・インデックスに関する情報を表示します。

```
sp_iqjoinindex t1
joinnidex_name creator left_table_name left_table_owner left_column_
name
join_type right_table_name right_table_owner right_column_name key_t
ype valid dbspace_id remarks
t1_t2_t3_join  DBA     t1              DBA              n1
=         t2              DBA              n1              NATURAL
Y    16387    (NULL)
```

テーブル `t2` がジョイン演算の左側を構成する、ジョイン・インデックスに関する情報を表示します。

```
sp_iqjoinindex t2
joinnidex_name creator left_table_name left_table_owner left_column_
name
join_type right_table_name right_table_owner right_column_name key_t
ype valid dbspace_id remarks
t1_t2_t3_join  DBA     t2              DBA              n1
=         t3              DBA              n1               NATURAL
Y    (NULL) t1_t2_t3_join  DBA     t2             DBA            name
=         t3              DBA              name             NATURAL
Y    16387    ((NULL)
```

DBA の所有するテーブル t2 のカラム name が左側にあり、DBA の所有するテーブル t3 のカラム name が右側にある、ジョイン・インデックスに関する情報を表示します。

```
sp_iqjoinindex t2, name, DBA, t3, name, DBA
joinindex_name creator left_table_name left_table_owner left_column_
name
join_type right_table_name right_table_owner right_column_name key_t
ype valid dbspace_id remarks
t1_t2_t3_join  DBA     t2              DBA              name
=         t3              DBA              name             NATURAL
Y    16387    ((NULL)
```

参照：

- sp_iqcolumn プロシージャ (403 ページ)
- sp_iqconstraint プロシージャ (410 ページ)
- sp_iqdatatype プロシージャ (419 ページ)
- sp_iqevent プロシージャ (443 ページ)
- sp_iqhelp プロシージャ (448 ページ)
- sp_iqindex および sp_iqindex_alt プロシージャ (456 ページ)
- sp_iqpkeys プロシージャ (491 ページ)
- sp_iqprocparm プロシージャ (496 ページ)
- sp_iq_reset_identity プロシージャ (505 ページ)
- sp_iqtable プロシージャ (525 ページ)
- sp_iqview プロシージャ (541 ページ)

sp_iqjoinindexsize プロシージャ

指定されたジョイン・インデックスのサイズを取得します。

構文

sp_iqjoinindexsize (*join_index_name*)

パーミッション
DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

説明
インデックスの合計サイズを、バイト、キロバイト、および Nblocks (IQ ブロック) で返します。また、メモリにジョイン・インデックスを保持するために必要なページ数と、(ディスク上で) ジョイン・インデックスを圧縮した場合の IQ ページ数も返します。このプロシージャには、必ず *join_index_name* パラメータを指定してください。

表 156 : sp_iqjoinindexsize のカラム

カラム名	説明
Username	ジョイン・インデックスの所有者。
JoinIndexName	結果を返すジョイン・インデックス。
Number of Tables	ジョイン・インデックス内のテーブル数。
KBytes	物理オブジェクト・サイズ (KB)。
Pages	メモリ内にオブジェクトを保持するために必要な IQ ページの数。
Compressed Pages	(ディスク上で) オブジェクトを圧縮した場合の IQ ページの数。
NBlocks	IQ ブロックの数

例
```
sp_iqjoinindexsize ( 't1t2' )
```

Username	JoinIndexName	Number of Tables	KBytes	Pages	Compressed Pages	NBlocks
DBA	t1t2	2	13	15	4	26

sp_iqlmconfig プロシージャ

ライセンス管理設定を制御し、ライセンスのタイプとステータスを表示して設定します。

構文 1
```
sp_iqlmconfig 'edition', { 'SE' | 'SA' | 'EE' }
```

表 157 : "edition" パラメータの要約情報

トピック	値
デフォルト値	'EE' (Enterprise Edition)
値の範囲	'SE' (Small Business) 'SA' (Single Application) 'EE' (Enterprise Edition)
ステータス	静的

構文 2

```
sp_iqlmconfig 'license type', { 'CP' | 'DT' | 'SF' | 'AC' | 'BC' |
'CH' | 'DH' | 'SH' | 'AH' | 'BH' }
```

表 158 : "license type" パラメータの要約情報

トピック	値
デフォルト値	'DT' (開発およびテスト)
値の範囲	'AC' (OEM CPU ライセンス) 'AH' (OEM CPU ライセンス・チップ) 'BC' (OEM スタンバイ・ライセンス) 'BH' (OEM スタンバイ・ライセンス・チップ) 'CP' (CPU ライセンス) 'CH' (CPU ライセンス・チップ) 'DH' (開発およびテスト・ライセンス・チップ) 'DT' (開発およびテスト) 'EV' (評価) 'SF' (スタンバイ CPU ライセンス) 'SH' (スタンバイ CPU ライセンス・チップ)
ステータス	静的

構文 3

```
sp_iqlmconfig 'email severity', { 'ERROR' | 'WARNING' |
'INFORMATIONAL' | 'NONE' }
```

NONE は、電子メール通知の無効を指定します。

構文 4

```
sp_iqlmconfig 'smtp host', '<host name>'
```

表 159 : パラメータ

パラメータ	説明
ホスト名	電子メール通知に使用する SMTP ホストを指定します。

構文 5

```
sp_iqlmconfig 'email sender', '<email address>'
```

表 160 : パラメータ

パラメータ	説明
<email address>	電子メール通知に使用する送信者の電子メール・アドレスを指定します。

構文 6

```
sp_iqlmconfig 'email recipients', '<email recipients>'
```

表 161 : パラメータ

パラメータ	説明
<email recipients>	電子メール通知を送信する電子メール・アドレスのカンマで区切られたリストを指定します。

構文 7

```
sp_iqlmconfig
```

パーミッション
DBA パーミッションが必要です。

使用法
起動時に、**sp_iqlmconfig** は、指定のライセンスのエディション・タイプとライセンス・タイプをチェックします。

- 指定のライセンスが見つからない場合、サーバは猶予モードになります。
- 指定のライセンス・タイプは、NULL 以外のエディション値が指定された場合のみ有効になります。
- **sp_iqlmconfig** はパラメータなしで呼び出され (構文 3)、上記のすべての情報に加え、次に示す情報も表示します。
  - 製品エディションまたはライセンス・タイプ
  - 使用しているオプションのライセンス
  - ライセンス・カウント
  - 電子メール情報
  - ライセンスに関する一般情報

参照：
- サーバで使用可能なプロパティ (129 ページ)

sp_iqlocks プロシージャ

IQ ストアとカタログ・ストアの両方に関して、データベース内のロックに関する情報を表示します。

構文

```
sp_iqlocks ([connection,] [[owner.]table_name] max_locks,]
[sort_order])
```

パーミッション

DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

使用法

結果の絞り込みに使用できる、オプションの **sp_iqlocks** パラメータを以下に示します。

表 162 : sp_iqlocks のパラメータ (オプション)

パラメータ	データ型	説明
connection	integer	接続 ID。このオプションを指定すると、指定された接続のロックに関する情報のみが返されます。デフォルトはゼロ (すべての接続についての情報を返す)。
owner.table_name	char(128)	テーブル名。このオプションを指定すると、指定されたテーブル内のロックについての情報のみが返されます。デフォルトは NULL (データベース内のすべてのテーブルについての情報を返す)。owner を指定しなかった場合、プロシージャを呼び出したユーザがテーブルの所有者と見なされます。
max_locks	integer	情報を返すロックの最大数。デフォルトは 0 (すべてのロック情報を返す)。
sort_order	char(1)	情報を返す順序。 • C は接続ごとにソート (デフォルト) • T は table_name でソート

説明

データベース内の現在のロックについての情報を表示します。指定する接続によって、単一の接続内のロック、単一のテーブル内のロック、指定された数のロック、というように結果を絞り込むことができます。

sp_iqlocks は、次の情報を sort_order パラメータに指定されたとおりにソートして表示します。

表 163 : sp_iqlocks のカラム

カラム	データ型	説明
conn_name	VARCHAR(128)	現在の接続の名前。
conn_id	INTEGER	ロックが存在する接続 ID。
user_id	CHAR(128)	この接続 ID に関連付けられているユーザ。
table_type	CHAR(6)	テーブルの種類。これは、テーブルを表す BASE、グローバル一時テーブルを表す GLBTMP、マテリアライズド・ビューを表す MVIEW のいずれかになります。 マテリアライズド・ビューは、IQ カタログ・ストアの SQL Anywhere テーブルでのみサポートされます。
creator	VARCHAR(128)	テーブルの所有者。
table_name	VARCHAR(128)	ロックが存在するテーブル。
index_id	INTEGER	インデックス ID または NULL。

カラム	データ型	説明
lock_class	CHAR(8)	ロックのタイプを示す文字列。 • S – share (共有) • SW – share and write (共有／書き込み) • EW – exclusive and write (排他／書き込み) • E – exclusive (排他) • P – phantom (幻) • A – antiphantom (幻でない) • W – write (書き込み) すべてのロックには、S、E、EW、SW のいずれかと、P、A のいずれかまたは両方が表示されます。幻ロックと幻でないロックには、T または * の修飾子が付きます。 • T ― ロックは逐次スキャンに関連する。 • * ― ロックはすべてのスキャンに関連する。 • *nnn* – インデックス番号。ロックが特定のインデックスに関連する場合。 Sybase IQ は、共有ロックを取得してから書き込みロックを取得します。接続に排他ロックがある場合、共有ロックは表示されません。 書き込みロックの場合は、排他、共有、書き込みのすべてのロックが接続にあれば、EW が表示されます。
lock_duration	CHAR(11)	ロックの期間。Transaction、Position、または Connection の 1 つ。
lock_type	CHAR(9)	ロックを識別する値 (ロック・クラスに依存)。
row_identifier	UNSIGNED BIGINT	ローまたは NULL の識別子。

接続 ID またはテーブルでロックされているユーザのユーザ名を見つけられない場合、**sp_iqlocks** には接続 ID に 0 (ゼロ) が表示され、ユーザ名には `User unavailable` が表示されます。

注意： 排他ロック、幻ロック、幻でないロックは、SQL Anywhere のテーブルでは使用できますが、Sybase IQ テーブルでは使用できません。 カタログ・ストア内のテーブルで明示的にロックを解除しないかぎり、これらの種類のロックが Sybase IQ データベースで起きる (修飾子 T、*、および *nnn* が表示される) ことはありませ

ん。 SQL Anywhere テーブルでのロックについては、『SQL Anywhere サーバー - SQL の使用法』を参照してください。

例

次の例は、**sp_iqlocks** プロシージャ・コールと、Sybase IQ データベースでのその出力を示します。 このプロシージャは、すべてのデフォルト・オプションとともに呼び出されており、すべてのロックが接続ごとにソートされて表示されます。

```
call sp_iqlocks()
```

```
conn_name  conn_id  user_id  table_type  creator  table_name
=========  =======  =======  ==========  =======  ==========
con1       70187172 'mary'   BASE        DBA       t1

index_id              lock_class  lock_duration  lock_type  row_iden
tifier
========              ==========  =============  =========  ========
======
ASIQ_IDX_T452_C19_FP  Table       Position       Table      1
```

sp_iqmodifyadmin プロシージャ

名前付きログイン・ポリシーのオプションを所定の値に設定します。ログイン・ポリシーを指定しないと、オプションがルート・ポリシーに設定されます。マルチプレックスでは、sp_iqmodifyadmin は、マルチプレックス・サーバ名であるオプションのパラメータを指定します。

構文 1

```
call sp_iqmodifyadmin (‘policy_option_name’, ‘value_in’ ,
[’login_policy_name’] )
```

構文 2

```
sp_iqmodifyadmin ‘policy_option_name’, ‘value_in’ ,’login_policy_name ’
```

構文 3

```
sp_iqmodifyadmin  policy_option_name, value_in, login_policy_name
```

構文 4

```
sp_iqmodifyadmin ‘policy_option_name’,
‘value_in’ ,’login_policy_name ’ ,’server_name ’
```

使用法

表 164 : パラメータ

パラメータ	説明
policy_option_name	変更するログイン・ポリシー・オプション。
value_in	ログイン・ポリシー・オプションの新しい値。
login_policy_name	変更に必要なログイン・ポリシー・オプションを持つログイン・ポリシーの名前。

パーミッション
DBA 権限が必要です。

例
lockeduser という名前のポリシーについて、ログイン・オプション `locked` を ON に設定します。

```
call sp_iqmodifyadmin ('locked','on','lockeduser')
```

`Writer1` という名前のマルチプレックス・サーバ上の *lockeduser* というポリシーについて、次のようにログイン・オプション `locked` を ON に設定します。

```
call sp_iqmodifyadmin ('locked','on','lockeduser','Writer1')
```

参照：

- sp_expireallpasswords プロシージャ (381 ページ)
- sp_iqaddlogin プロシージャ (381 ページ)
- sp_iqcopyloginpolicy プロシージャ (415 ページ)
- sp_iqmodifylogin プロシージャ (482 ページ)
- sp_iqpassword プロシージャ (489 ページ)

sp_iqmodifylogin プロシージャ

ユーザをログイン・ポリシーに割り当てます。

構文 1

```
call sp_iqmodifylogin ('userid' [, 'login_policy_name'])
```

構文 2

```
sp_iqmodifylogin 'userid', ['login_policy_name']
```

パーミッション
DBA 権限が必要です。

使用法

表 165 : パラメータ

パラメータ	説明
userid	変更するアカウントの名前を保持する変数。
login_policy_name	(オプション) ユーザを割り当てるログイン・ポリシーの名前を指定します。ログイン・ポリシー名を指定しないと、ユーザがルート・ログイン・ポリシーに割り当てられます。

例
次のように、ユーザ `joe` を `expired_password` というログイン・ポリシーに割り当てます。

```
sp_iqmodifylogin 'joe', 'expired_password'
```

次のように、ユーザ `joe` をルート・ログイン・ポリシーに割り当てます。

```
call sp_iqmodifylogin ('joe')
```

参照：
- sp_expireallpasswords プロシージャ (381 ページ)
- sp_iqaddlogin プロシージャ (381 ページ)
- sp_iqcopyloginpolicy プロシージャ (415 ページ)
- sp_iqmodifyadmin プロシージャ (481 ページ)
- sp_iqpassword プロシージャ (489 ページ)

sp_iqmpxcheckdqpconfig プロシージャ

sp_iqmpxcheckdqpconfig は、現在の接続の DQP 設定をチェックする診断ツールです。DQP に失敗する場合は、**sp_iqmpxcheckdqpconfig** を実行して、DQP 設定に問題があるためにクエリ分散が失敗していないかどうかを確認します。

構文

```
sp_iqmpxcheckdqpconfig
```

パーミッション
このプロシージャの実行に、特別な権限は必要ありません。

説明

表 166 : カラムの説明

カラム名	説明
DiagMsgID	診断メッセージの一意の ID
説明	見つかった DQP 設定の問題を説明する診断メッセージ

表 167 : 診断メッセージ

DiagMsgID	説明
0	DQP 設定に問題はない
1	データベースがシンプレックスである
2	マルチプレックスがシングルノード設定モードで実行されている
3	ログイン・ポリシー・オプションの dqp_enabled が OFF に設定されている
4	dqp_enabled 接続オプションが一時的に OFF に設定されている
5	論理サーバ・コンテキストがメンバ・ノードを 1 つしかサポートしていない
6	論理サーバで指定されたメンバシップが現在無効のため、コーディネータが DQP に参加していない
7	ALLOW_COORDINATOR_AS_MEMBER オプションが OFF に設定されているため、論理サーバで指定されたメンバシップが現在無効になっており、コーディネータが DQP に参加していない
8	IQ_SHARED_TEMP DB 領域に dbfile が存在しない
9	IQ_SHARED_TEMP DB 領域内のすべての dbfile が読み取り専用になっている
10	IQ_SHARED_TEMP DB 領域が動的にオフラインになっている

例

sp_iqmpxcheckdqpconfig プロシージャからの出力例を示します。

```
diagmsgid           description
3                   Login policy option dqp_enabled is set to OFF
5                   Logical server context has only one member node
6                   Coordinator does not participate in DQP since its
                    named membership in the logical server is
                    currently ineffective
7                   Coordinator does not participate in DQP since
                    its logical membership in the logical server
                    is currently ineffective because
                    ALLOW_COORDINATOR_AS_MEMBER option in Root
                    Logical server policy set to OFF
8                   There is no dbfile in IQ_SHARED_TEMP dbspace
```

sp_iqmpxfilestatus プロシージャ

コーディネータ・ノードで実行した場合は、含まれている各セカンダリ・ノード上のあらゆる共有 dbfile のファイル・ステータスが表示されます。

共有 dbfile には、IQ_SYSTEM_MAIN、IQ_SHARED_TEMP、ユーザ DB 領域内のすべてのファイルが含まれています。セカンダリ・ノードで実行した場合は、現在のノードのファイル・ステータスのみが表示されます。セカンダリ・ノードを実行するには、マルチプレックス・グリッド・オプションのライセンスを取得している必要があります。 構文などの詳細については、『Sybase IQ Multiplex の使用』を参照してください。

sp_iqmpxincconnpoolinfo プロシージャ

コーディネータ・ノードで実行した場合、各ノードの INC 接続プール・ステータスが表示されます。セカンダリ・ノードで実行した場合、現在のノードのみの INC 接続プール・ステータスが表示されます。

セカンダリ・ノードを実行するには、マルチプレックス・グリッド・オプションのライセンスを取得している必要があります。 **sp_iqmpxincconnpoolinfo** の構文と詳細な説明については、『Sybase IQ Multiplex の使用』を参照してください。

sp_iqmpxincheartbeatinfo プロシージャ

コーディネータ・ノードで実行した場合、各ノードの INC ハートビート・ステータスが表示されます。セカンダリ・ノードで実行した場合、現在のノードのみの INC ハートビート・ステータスが表示されます。

セカンダリ・ノードを実行するには、マルチプレックス・グリッド・オプションのライセンスを取得している必要があります。 **sp_iqmpxincheartbeatinfo** の構文と詳細な説明については、『Sybase IQ Multiplex の使用』を参照してください。

sp_iqmpxinfo プロシージャ

コーディネータ・ノード上で実行した場合、すべてのノードの完全なマルチプレックス構成情報が表示されます。セカンダリ・ノード上で実行した場合、そのノードのみの完全なマルチプレックス構成情報が表示されます。

セカンダリ・ノードを実行するには、マルチプレックス・グリッド・オプションのライセンスを取得している必要があります。 **sp_iqmpxinfo** の構文と詳細な説明については、『Sybase IQ Multiplex の使用』を参照してください。

sp_iqmpxvalidate プロシージャ

マルチプレックス設定の矛盾をチェックします。

sp_iqmpxvalidate の構文と詳細な説明については、『Sybase IQ Multiplex の使用』を参照してください。

sp_iqmpxversioninfo プロシージャ

このサーバの現在のバージョン情報を表示します。

sp_iqmpxversioninfo の構文と詳細な説明については、『Sybase IQ Multiplex の使用』を参照してください。

sp_iqobjectinfo プロシージャ

データベースのオブジェクトおよびサブオブジェクトのパーティションと DB 領域の割り当てを返します。

構文

```
sp_iqobjectinfo [ owner_name ]  [ , object_name ] [ , object-type ]
```

パーミッション

DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

使用法

表 168 : パラメータ

パラメータ	説明
owner_name	オブジェクトの所有者。指定した場合、**sp_iqobjectinfo** は、指定の所有者を持つテーブルおよびジョイン・インデックスについてのみ出力を表示します。指定しない場合、**sp_iqobjectinfo** は、データベース内のすべてのユーザのテーブルおよびジョイン・インデックスに関する情報を表示します。
object_name	テーブルまたはジョイン・インデックスの名前。指定しない場合、**sp_iqobjectinfo** は、データベース内のすべてのテーブルおよびジョイン・インデックスに関する情報を表示します。
object-type	有効なオブジェクト・タイプは、**table** (デフォルト) または **joinindex** です。 オブジェクト・タイプが table の場合は、引用符で囲んでください。

すべてのパラメータがオプションであり、どのパラメータも他のパラメータの値とは関係なく指定できます。

入力パラメータは **sp_iqobjectinfo** と共に使用することをおすすめします。**sp_iqobjectinfo** の結果にクエリを実行でき、クエリの **WHERE** 句で述部を使用する代わりに、入力パラメータを使用することで、クエリのパフォーマンスが向上します。たとえば、クエリ A は次のように記述します。

```
SELECT COUNT(*) FROM sp_iqobjectinfo() WHERE owner = 'DBA' AND
object_name = 'tab_case510' AND object_type = 'table' AND
sub_object_name is NULL AND dbspace_name = 'iqmain7' AND
partition_name = 'P1'
```

クエリ B は、クエリ A を記述し直して、**sp_iqobjectinfo** 入力パラメータを使用できるようにしたものです。

```
SELECT COUNT(*) FROM sp_iqobjectinfo('DBA','tab_case510','table')
WHERE sub_object_name is NULL AND dbspace_name = 'iqmain7' AND
PARTITION_NAME = 'P1'
```

クエリ B は、クエリ A よりも速く結果を返します。パラメータが **sp_iqobjectinfo** に渡されると、プロシージャはシステム・テーブル内の少数のレコードを比較してジョインします。つまり、クエリ A に比べ作業が少なくなります。クエリ B では、述部はプロシージャ自体に適用され、プロシージャは小さい結果セットを返します。そのため、より少数の述部がクエリに適用されます。

sp_iqobjectinfo ストアド・プロシージャは、*owner_name*、*object_name*、*object_type* の解釈のために、ワイルドカード文字をサポートしています。これは、**LIKE** 句が

クエリ内のパターンを照合するのと同じ方法で、指定のパターンと一致するすべての DB 領域の情報を表示します。

説明
特定のまたはすべてのデータベース・オブジェクト (テーブルおよびジョイン・インデックス・タイプのみ) とそのサブオブジェクトの、すべてのパーティションと DB 領域割り当てを返します。サブオブジェクトは、カラム、インデックス、プライマリ・キー、一意性制約、および外部キーです。

表 169 : sp_iqobjectinfo のカラム

カラム名	説明
owner	オブジェクトの所有者の名前。
object_name	DB 領域にあるオブジェクトの名前 (テーブルおよびジョイン・インデックス・タイプのみ)。
sub_object_name	DB 領域に存在するオブジェクトの名前。
object_type	オブジェクトのタイプ (カラム、インデックス、,プライマリ・キー、一意性制約、外部キー、パーティション、ジョイン・インデックスまたはテーブル)。
object_id	オブジェクトのグローバル・オブジェクト ID。
id	オブジェクトのテーブル ID またはジョイン・インデックス ID。
dbspace_name	オブジェクトが存在する DB 領域の名前。文字列 "[multiple]" は、分割されたオブジェクトの特別なメタ・ローの場合に表示されます。[multiple] ローは、テーブルまたはカラムを説明するために、出力に複数のローが伴うことを示します。
partition_name	指定のオブジェクトのパーティションの名前。

例

注意： 出力内容をわかりやすくするため、次の 2 つの例は `iqdemo` データベース内のオブジェクトを示しています。iqdemo には、`iq_main` というサンプルのユーザ DB 領域が含まれています。ただし、この領域は、ユーザが所有するデータベースには存在しない場合があります。

指定のユーザが所有する特定のデータベース・オブジェクトおよびサブオブジェクトのパーティションおよび DB 領域割り当てに関する情報を表示します。

```
sp_iqobjectinfo GROUPO,Departments
```

```
owner   object_name  sub_object_name               object_type  obj
ect_id  id
```

```
GROUPO  Departments  (NULL)                              table         3
632     738
GROUPO  Departments  DepartmentID                       column        3
633     738
GROUPO  Departments  DepartmentName                     column        3
634     738
GROUPO  Departments  DepartmentHeadID                   column        3
635     738
GROUPO  Departments  DepartmentsKey                     primary
key   83       738
GROUPO  Departments  FK_DepartmentHeadID_EmployeeID  foreign
key   92       738

dbspace_name     partition_name
iq_main          (NULL)
iq_main          (NULL)
iq_main          (NULL)
iq_main          (NULL)
iq_main          (NULL)
iq_main          (NULL)
```

指定のユーザが所有している特定のデータベース・オブジェクトおよびサブオブジェクトのパーティションおよび DB 領域割り当てに関する情報を、*object-type***table** に表示します。

```
sp_iqobjectinfo DBA,sale,'table'
```

```
owner  object_name  sub_object_name  object_type  object_id  id
DBA    sale         (NULL)            table          3698     742
DBA    sale         prod_id           column         3699     742
DBA    sale         month_num         column         3700     742
DBA    sale         rep_id            column         3701     742
DBA    sale         sales             column         3702     742
dbspace_name      partition_name iq_main           (NULL)
iq_main           (NULL) iq_main           (NULL)
iq_main           (NULL) iq_main           (NULL)
```

sp_iqpassword プロシージャ

ユーザのパスワードを変更します。

構文 1

```
call sp_iqpassword ('caller_password', 'new_password' [, 'user_name'])
```

構文 2

```
sp_iqpassword 'caller_password', 'new_password' [, 'user_ name']
```

パーミッション
自分のパスワードを設定する場合には不要ですが、他のユーザのパスワードを設定する場合には DBA 権限または PERMS ADMIN 権限が必要です。

使用法

表 170 : パラメータ

パラメータ	説明
caller_password	自分のパスワード。自分のパスワードを変更する場合は、古い方のパスワードを指定します。DBA または PERMS ADMIN が他のユーザのパスワードを変更する場合、caller_password は DBA または PERMS ADMIN のパスワードです。
new_password	ユーザの新しいパスワードです。
user_name	DBA または PERMS ADMIN がパスワードを変更するユーザのログイン名。自分のパスワードを変更する場合は user_name を指定しません。

説明
ユーザ・パスワードは識別子です。すべてのユーザが **sp_iqpassword** によって自分のパスワードを変更できます。DBA 権限または PERMS ADMIN 権限では、既存のすべてのユーザのパスワードを変更できます。

ユーザの作成には、**CREATE USER** 文を使用する方法が適しています。

識別子の最大長は、128 バイトです。識別子は、次のいずれかの条件に当てはまる場合、二重引用符または角カッコで囲む必要があります。

- 識別子にスペースが含まれている。
- 識別子の最初の文字がアルファベット文字 (以下で説明) ではない。
- 識別子に予約語が含まれている。
- 識別子にアルファベット文字と数字以外の文字が含まれている。
  *アルファベット文字*に含まれるのは、アルファベット、アンダースコア文字 (_)、アットマーク (@)、シャープ記号 (#)、ドル記号 ($) です。データベースの照合順によって、どの文字をアルファベットまたは数字として扱うかが決まります。

例
ログインしたユーザのパスワードを irk103 から exP984 に変更します。

```
sp_iqpassword 'irk103', 'exP984'
```

ログインしているユーザが DBA 権限または PERMS ADMIN 権限を持っているか、またはユーザが joe 自身の場合のみ、ユーザ joe のパスワードを eprr45 から pdi032 に変更します。

```
call sp_iqpassword ('eprr45', 'pdi032', 'joe')
```

参照：

- sp_expireallpasswords プロシージャ (381 ページ)
- sp_iqaddlogin プロシージャ (381 ページ)
- sp_iqcopyloginpolicy プロシージャ (415 ページ)
- sp_iqmodifyadmin プロシージャ (481 ページ)
- sp_iqmodifylogin プロシージャ (482 ページ)

sp_iqpkeys プロシージャ

プライマリ・キーとプライマリ・キー制約に関する情報を、テーブル、カラム、テーブル所有者別に表示します。または、データベース内のすべての Sybase IQ テーブルについて表示します。

構文

```
sp_iqpkeys { [ table-name ], [ column-name ], [ table-owner ] }
```

パーミッション

DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

使用法

表 171 : パラメータ

パラメータ	説明
table-name	ベース・テーブルまたはグローバル・テンポラリ・テーブルの名前。このオプションを指定すると、指定されたテーブルで定義されたプライマリ・キーに関する情報のみが返されます。
column-name	カラムの名前。このオプションを指定すると、指定されたカラムのプライマリ・キーに関する情報のみが返されます。
table-owner	テーブルの所有者。このオプションを指定すると、指定された所有者によって所有されているテーブルのプライマリ・キーに関する情報のみが返されます。

1 つ以上のパラメータを指定できます。最初の 2 つのパラメータのいずれかを指定せずに、その次のパラメータを指定する場合は、指定しないパラメータの位置に NULL を指定する必要があります。パラメータを指定しない場合、データベース内のすべてのテーブルのすべてのプライマリ・キーの説明が表示されます。指定したいずれかのパラメータが無効な場合、出力にローが表示されません。

表 172 : sp_iqpkeys の使用例

構文	出力
sp_iqpkeys sales	テーブル `sales` 上で定義されたプライマリ・キーに関する情報を表示します。
sp_iqpkeys sales, NULL, DBA	`DBA` が所有するテーブル `sales` 上で定義された、プライマリ・キーに関する情報を表示します。
sp_iqpkeys sales, store_id, DBA	`DBA` が所有するテーブル `sales` のカラム `store_id` 上で定義された、プライマリ・キーに関する情報を表示します。
sp_iqpkeys NULL, NULL, DBA	`DBA` が所有するすべてのテーブル上で定義された、プライマリ・キーに関する情報を表示します。

説明
sp_iqpkeys ストアド・プロシージャは、データベース内のベース・テーブルおよびグローバル・テンポラリ・テーブル上のプライマリ・キーに関する次の情報を表示します。

表 173 : sp_iqpkeys のカラム

カラム名	説明
table_name	テーブルの名前。
table_owner	テーブルの所有者
column_name	プライマリ・キーが定義されているカラムの名前。
column_id	カラム ID。
constraint_name	プライマリ・キー制約の名前。
constraint_id	プライマリ・キー制約の ID。

注意： **sp_iqpkeys** ストアド・プロシージャは、Sybase IQ のバージョン 12.6 以降で作成されたデータベースにのみ存在します。

例
テーブル `sales1` のカラムで定義されたプライマリ・キーを表示します。

```
sp_iqpkeys sales1
table_name table_owner column_name column_id constraint_name constra
int_id
sales1         DBA        store_id     1         MA114           114
```

テーブル `sales2` のカラムで定義されたプライマリ・キーを表示します。

```
sp_iqpkeys sales2
table_name table_owner column_name column_id constraint_name constra
int_id
sales2         DBA      store_id,     1,2        MA115           115
                         order_num
```

テーブル sales2 のカラム store_id で定義されたプライマリ・キーを表示します。

```
sp_iqpkeys sales2, store_id
table_name table_owner column_name column_id constraint_name constra
int_id
sales2         DBA      store_id       1         MA115           115
```

参照：

- sp_iqcolumn プロシージャ (403 ページ)
- sp_iqconstraint プロシージャ (410 ページ)
- sp_iqdatatype プロシージャ (419 ページ)
- sp_iqevent プロシージャ (443 ページ)
- sp_iqhelp プロシージャ (448 ページ)
- sp_iqindex および sp_iqindex_alt プロシージャ (456 ページ)
- sp_iqjoinindex プロシージャ (471 ページ)
- sp_iqprocparm プロシージャ (496 ページ)
- sp_iq_reset_identity プロシージャ (505 ページ)
- sp_iqtable プロシージャ (525 ページ)
- sp_iqview プロシージャ (541 ページ)

sp_iqprocedure プロシージャ

システム・プロシージャおよびユーザ定義プロシージャに関する情報を表示します。

構文

sp_iqprocedure [*proc-name*], [*proc-owner*], [*proc-type*]

パーミッション

DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

使用法

表 174 : パラメータ

パラメータ	説明
proc-name	プロシージャの名前です。
proc-owner	プロシージャの所有者。
proc-type	プロシージャのタイプ。指定できる値は次のとおりです。 • **SYSTEM**： システム・プロシージャ (ユーザ SYS または dbo が所有するプロシージャ) に関する情報のみを表示します。 • **ALL**： ユーザ・プロシージャおよびシステム・プロシージャに関する情報を表示します。 • その他の値：ユーザのプロシージャに関する情報を表示します。

sp_iqprocedure プロシージャは、パラメータなしで呼び出せます。パラメータを指定しない場合、ユーザ定義プロシージャ (dbo または SYS が所有していないプロシージャ) に関する情報のみがデフォルトで表示されます。

最初の 2 つのパラメータのいずれかを指定せずに、その次のパラメータを指定する場合は、指定しないパラメータの位置に NULL を指定する必要があります。たとえば、`sp_iqprocedure NULL, NULL, SYSTEM` および `sp_iqprocedure NULL, user1` とします。

表 175 : sp_iqprocedure の使用例

構文	出力
sp_iqprocedure	dbo または SYS によって所有されていない、データベース内のすべてのプロシージャに関する情報を表示します。
sp_iqprocedure sp_test	プロシージャ **sp_test** に関する情報を表示します。
sp_iqprocedure non_existing_proc	プロシージャ **non_existing_proc** が存在しないため、ローは返されません。
sp_iqprocedure NULL, DBA	DBA が所有するすべてのプロシージャに関する情報を表示します。
sp_iqprocedure sp_test, DBA	DBA が所有するプロシージャ **sp_test** に関する情報を表示します。

構文	出力
sp_iqprocedure sp_iqtable	プロシージャ **sp_iqtable** はシステム・プロシージャではありません。 **sp_iqtable** という名前のユーザ定義プロシージャが存在しない場合、ローは返されません (デフォルトでは、ユーザ定義プロシージャのみが返されます)。
sp_iqprocedure sp_iqtable, dbo	**sp_iqtable** プロシージャはユーザ・プロシージャではないため、ローは返されません (デフォルトではユーザ・プロシージャのみが返されます)。
sp_iqprocedure NULL, NULL, SYSTEM	システム・プロシージャ (dbo または SYS が所有するプロシージャ) に関する情報を表示します。
sp_iqprocedure sp_iqtable, NULL, SYSTEM	システム・プロシージャ **sp_iqtable** に関する情報を表示します。
sp_iqprocedure sp_iqtable, dbo, ALL	dbo が所有するシステム・プロシージャ **sp_iqtable** に関する情報を表示します。

説明

sp_iqprocedure ストアド・プロシージャは、データベース内のプロシージャに関する情報を表示します。1 つ以上のパラメータを指定した場合、指定したパラメータによって結果がフィルタされます。たとえば、*proc-name* を指定した場合、指定のプロシージャに関する情報のみが表示されます。*proc-owner* を指定した場合、**sp_iqprocedure** は指定の所有者が所有するプロシージャに関する情報のみを返します。パラメータを指定しない場合、**sp_iqprocedure** はデータベース内のすべてのユーザ定義プロシージャに関する情報を表示します。

sp_iqprocedure プロシージャは、次のカラムに情報を返します。

表 176 : sp_iqprocedure のカラム

カラム名	説明
proc_name	プロシージャの名前。
proc_owner	プロシージャの所有者。
proc_defn	プロシージャの作成に使用するコマンド。隠しプロシージャでは、キーワード 'HIDDEN' が表示されます。
replicate	このプロシージャが Replication Server のインストール環境においてプライマリ・データ・ソースである場合は Y が表示され、それ以外の場合は N が表示されます。

カラム名	説明
srvid	プロシージャがリモート・データベース・サーバ上にある場合にリモート・サーバを示します。
remarks	コメント文字列。

例
ユーザ定義プロシージャ sp_test に関する情報を表示します。

```
sp_iqprocedure sp_test
proc_name    proc_owner    proc_defn       replicate     srvid     r
emarks  sp_test      DBA         create
procedure   N             (NULL)    (NULL)
                    DBA.sp_test(in n1                       integer)
                       begin message‘sp_test’end
```

DBA が所有するすべてのプロシージャに関する情報を表示します。

```
sp_iqprocedure NULL, DBA
proc_name    proc_owner    proc_defn       replicate     srvid     r
emarks  sp_test      DBA         create
procedure   N             (NULL)    (NULL)
                    DBA.sp_test(in n1                       integer)
                       begin message‘sp_test’end
sp_dept     DBA       create procedure   N           (NULL)   (NULL)
                       DBA.sp_dept() begin end
```

sp_iqprocparm プロシージャ

ストアド・プロシージャに関する情報を表示します。結果セット変数と SQLSTATE/SQLCODE エラー値も含まれます。

構文

sp_iqprocparm [*proc-name*], [*proc-owner*], [*proc-type*]

パーミッション
DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

使用法

表 177 : パラメータ

パラメータ	説明
proc-name	プロシージャの名前です。
proc-owner	プロシージャの所有者。

パラメータ	説明
proc-type	プロシージャのタイプ。指定できる値は次のとおりです。 • **SYSTEM**： システム・プロシージャ (ユーザ SYS または dbo が所有するプロシージャ) に関する情報のみを表示します。 • **ALL**： ユーザ・プロシージャおよびシステム・プロシージャに関する情報を表示します。 • その他の値：ユーザのプロシージャに関する情報を表示します。

sp_iqprocparm はパラメータなしで呼び出せます。パラメータを指定しない場合、すべてのユーザ定義プロシージャ (dbo または SYS が所有していないプロシージャ) の入出力パラメータおよび結果パラメータがデフォルトで表示されます。

最初の 2 つのパラメータのいずれかを指定せずに、その次のパラメータを指定する場合は、指定しないパラメータの位置に NULL を指定する必要があります。たとえば、`sp_iqprocparm NULL, NULL, SYSTEM` および `sp_iqprocparm NULL, user1` とします。

表 178 : sp_iqprocparm の使用例

構文	出力
sp_iqprocparm	dbo または SYS によって所有されていない、データベース内のすべてのプロシージャのパラメータを表示します。
sp_iqprocparm sp_test	プロシージャ **sp_test** に関する情報を表示します。
sp_iqprocparm non_existing_proc	プロシージャ **non_existing_proc** が存在しないため、ローは返されません。
sp_iqprocparm NULL, DBA	DBA が所有するすべてのプロシージャのパラメータを表示します。
sp_iqprocparm sp_test, DBA	DBA が所有するプロシージャ **sp_test** のパラメータを表示します。
sp_iqprocparm sp_iqtable	**sp_iqtable** はシステム・プロシージャです。**sp_iqtable** という名前のユーザ定義プロシージャが存在しない場合、ローは返されません (デフォルトでは、ユーザ定義プロシージャのみが返されます)。

構文	出力
sp_iqprocparm sp_iqtable, dbo	**sp_iqtable** プロシージャはユーザ・プロシージャではないため、ローは返されません (デフォルトでは、ユーザ・プロシージャのみが返されます)。
sp_iqprocparm NULL, NULL, SYSTEM	システム・プロシージャ (dbo または SYS が所有するプロシージャ) のパラメータを表示します。
sp_iqprocparm sp_iqtable, NULL, SYSTEM	システム・プロシージャ **sp_iqtable** のパラメータを表示します。
sp_iqprocparm sp_iqtable, dbo, ALL	dbo が所有するシステム・プロシージャ **sp_iqtable** のパラメータを表示します。

説明

sp_iqprocparm ストアド・プロシージャは、ストアド・プロシージャのパラメータに関する情報を表示します。結果セット変数と SQLSTATE/SQLCODE エラー値も含まれます。1 つ以上のパラメータを指定した場合、指定したパラメータによって結果がフィルタされます。たとえば、*proc-name* を指定した場合、指定のプロシージャに対するパラメータの情報のみが表示されます。*proc-owner* を指定した場合、**sp_iqprocparm** は指定の所有者が所有するプロシージャに関する情報を返すだけです。パラメータを指定しない場合、**sp_iqprocparm** はパラメータに関する情報をデータベース内のすべてのユーザ定義プロシージャに表示します。

sp_iqprocparm プロシージャは、次のカラムに情報を返します。

表 179 : sp_iqprocparm のカラム

カラム名	説明
proc_name	プロシージャの名前。
proc_owner	プロシージャの所有者。
parm_name	パラメータの名前。
parm_type	パラメータは、次に示すタイプのいずれかに該当します。 • 標準のパラメータ (変数)。 • 結果変数。結果セットを返すプロシージャで使用されます。 • SQLSTATE エラー値。 • SQLCODE エラー値。

カラム名	説明
parm_mode	パラメータのモード。プロシージャに値を渡すパラメータ、値を返すパラメータ、この両方を行うパラメータ、またはいずれも行わないパラメータ。パラメータ・モードは、次のいずれかです。 • in： パラメータはプロシージャに値を渡します。 • out： パラメータは値を返します。 • inout： パラメータは、値を渡すとともに値を返します。 • NULL：パラメータは値を渡すことも返すこともしません。
domain_name	SYSDOMAIN システム・テーブルに挙げられたパラメータのデータ型の名前。
width	文字列パラメータの長さ、数値パラメータの精度、その他すべてのデータ型の格納バイト数。
scale	数値データ型パラメータでは小数点以下の桁数、その他のデータ型では 0。
default	文字列として保持されたパラメータのデフォルト値。

例
ユーザ定義プロシージャ `sp_test` のパラメータに関する情報を表示します。

```
sp_iqprocparm sp_test
proc_name proc_owner parm_name parm_type parm_mode domain_name widt
h scale  default  sp_test   DBA         ID
normal    in        integer     4     0      (NULL)
```

システム・プロシージャ **sp_iqshowcompression** のパラメータに関する情報を表示します。

```
sp_iqprocparm sp_iqshowcompression, dbo, system
proc_name            proc_owner  parm_name    parm_type  parm_mode
domain_name  width  scale  default
sp_iqshowcompression  dbo         @owner_name  normal     in
char         128    0     (NULL)
sp_iqshowcompression  dbo         @table_name  normal     in
char         128    0     (NULL)
sp_iqshowcompression  dbo         @column_name normal     in
char         128    0     (NULL)
sp_iqshowcompression  dbo         Column       result     out
char         128    0     (NULL)
sp_iqshowcompression  dbo         Compression  result     out
char         3      0     (NULL)
```

参照：
- sp_iqcolumn プロシージャ (403 ページ)
- sp_iqconstraint プロシージャ (410 ページ)

- sp_iqdatatype プロシージャ (419 ページ)
- sp_iqevent プロシージャ (443 ページ)
- sp_iqhelp プロシージャ (448 ページ)
- sp_iqindex および sp_iqindex_alt プロシージャ (456 ページ)
- sp_iqjoinindex プロシージャ (471 ページ)
- sp_iqpkeys プロシージャ (491 ページ)
- sp_iq_reset_identity プロシージャ (505 ページ)
- sp_iqtable プロシージャ (525 ページ)
- sp_iqview プロシージャ (541 ページ)

sp_iqrebuildindex プロシージャ

テーブル上の 1 つ以上のインデックスを、**CREATE TABLE** 文で指定された元の **IQ UNIQUE** の値または新しい **IQ UNIQUE** 値で再構築し、必要な記憶領域やクエリのパフォーマンスを変更します。

デフォルト・インデックス以外のインデックスを再構築するには、インデックス名を指定します。

構文

```
sp_iqrebuildindex  table_name, index_clause
```

パーミッション

このプロシージャを実行するには、EXECUTE パーミッションが必要です。テーブル上のインデックスを再構築するには、そのテーブルの INSERT パーミッションが必要です。

使用法

表 180 : パラメータ

パラメータ	説明
table_name	インデックスの再構築作業が行われるテーブルの、部分的または完全に修飾された名前。テーブルの所有者であるユーザがこのプロシージャを実行する場合、部分的に修飾された名前を使用できます。そうでない場合は、完全に修飾された名前である必要があります。

パラメータ	説明
index_clause	次に示す 1 つ以上の文字列。スペースで区切って指定します。 **column** *column_name [count]* **index** *index_name* 各 *column_name* または *index_name* は、指定されたテーブル上のカラムまたはインデックスを参照するものである必要があります。*column_name* または *index_name* を複数回指定した場合、プロシージャはエラーを返し、インデックスは再構築されません。 *count* は、**IQ UNIQUE** の値を示す非負数です。**CREATE TABLE** 文では、**IQ UNIQUE (count)** が、特定のカラム内にある、重複しない値の数を概算します。重複しない値の数は、クエリの速度と格納領域の要件に影響します。

キーワードである **column** と **index** を指定する必要があります。 これらのキーワードでは大文字と小文字が区別されません。

注意： このプロシージャは **TEXT** インデックスをサポートしません。**TEXT** インデックスを再構築するには、インデックスを削除してから再作成する必要があります。

説明

カラム名を指定した場合、このプロシージャはそのカラムのデフォルト・インデックスを再構築します。インデックス名を指定する必要はありません。 この場合に、カラム名に加えて Sybase IQ によって割り当てられたデフォルト・インデックスの名前を指定すると、エラーが発生します。 *column_name* の後ろの *count* を省略すると、0 (ゼロ) がデフォルトとして使用されます。

デフォルト・インデックスが 1 バイト・インデックスの場合、**sp_iqrebuildindex** は **IQ UNIQUE** に指定された値にかかわらず、これを必ず 1 バイト・インデックスとして再構築します。

1 バイトのデフォルト・インデックスでは、*column_name (count)* に指定された値が 0 であるか、または 256 よりも大きい場合、`SYS.SYSIQCOLUMN` 内の `approx_unique_count` カラムはカラムのカーディナリティの値を使用して更新されます。

sp_iqrebuildindex は、データ型 `LONG VARCHAR` (`CLOB`) のカラム上にある **WD** インデックスを再構築します。

デフォルト・インデックスが 2 バイト・インデックスであり、指定されたカウントが 0 であるか、または 65536 よりも大きい数である場合、カラムのカーディナリティの値によって、デフォルトを 1 バイト・インデックスと 2 バイト・インデックスのどちらで再構築するかが決定されます。このカーディナリティの値は、

SYS.SYSIQCOLUMN の approx_unique_count カラムを更新するのに使用されます。

IQ UNIQUE に 0 以外の値を指定した場合、デフォルト・インデックスは 1 バイト、2 バイト、またはフラット・デフォルト・インデックスとして再構築されます (前述の例外あり)。

IQ UNIQUE の値として 0 を指定した場合や、**IQ UNIQUE** に値を指定しなかった場合は、次のように、MINIMIZE_STORAGE オプションによってインデックスの再構築方法が決まります。

- MINIMIZE_STORAGE オプションを ON に設定すると、まず 1 バイトのデフォルト・インデックスとしてインデックスが再構築され、その後必要に応じて 2 バイトまたはフラットに変換されます。
- MINIMIZE_STORAGE を OFF に設定すると、インデックスはデータ型のデフォルトに従って再構築されます。

例
カラム *Surname* のデフォルト・インデックスを再構築します。

```
sp_iqrebuildindex ‘emp1‘, ‘column dept_id‘
```

または

```
call sp_iqrebuildindex (‘empl1‘, ‘column dept_id‘)
```

カラム c1 のフラット・デフォルト・インデックスを作成します。

```
CREATE TABLE mytable (c1 int IQ UNIQUE 1000000000)
```

デフォルトの 1 バイト・インデックスを 2 バイト・インデックスに変換します。

```
sp_iqrebuildindex ‘mytable‘, ‘column c1 1024‘
```

または

```
call sp_iqrebuildindex (‘mytable‘, ‘column c1 1024‘)
```

注意： サイズの大きい HG インデックス上で **sp_iqrebuildindex** を実行すると、一時的なパフォーマンスの低下が予想されます。

参照：

- sp_iqindexfragmentation プロシージャ (461 ページ)
- sp_iqrowdensity プロシージャ (507 ページ)

sp_iqrename プロシージャ

ユーザが作成したテーブル、カラム、インデックス、制約 (一意性、プライマリ・キー、外部キー、検査)、ストアド・プロシージャ、関数の名前を変更します。

構文

```
sp_iqrename  object-name, new-name [, object-type ]
```

パーミッション

テーブルの所有者か、オブジェクトの DBA 権限または alter パーミッションを持っていることが必要です。オブジェクトへの排他的なアクセスが必要です。

使用法

表 181 : パラメータ

パラメータ	説明
object-name	ユーザの作成したオブジェクトの元の名前。 オプションで、*object-name* には、*owner-name.object-name* のように指定して *owner-name* を含めることができます。*owner-name* は名前を変更するオブジェクトの所有者名です。*owner-name* を指定しない場合、**sp_iqrename** を呼び出しているユーザがオブジェクトの所有者と見なされます。**sp_iqrename** を呼び出しているユーザがオブジェクトの名前を変更するために必要なパーミッションを持つ場合にのみ、オブジェクトの名前が正しく変更されます。 名前を変更するオブジェクトがカラム、インデックス、または制約である場合、オブジェクトが関連付けられているテーブルの名前を指定する*必要があります*。カラム、インデックス、または制約の場合、*object-name* は *table-name.object-name* または *owner-name.table-name.object-name* の形式で指定できます。
new-name	オブジェクトの新しい名前。この名前は、ID の規則に従っており、名前を変更するオブジェクト・タイプに対してユニークであることが必要です。
object-type	名前を変更するユーザ作成オブジェクトのタイプを指定するオプション・パラメータ。つまり、オブジェクト *object-name* のタイプです。*object-type* パラメータは、大文字または小文字のいずれかで指定できます。

object-type パラメータの値を次に示します。

表 182 : sp_iqrename object-type のパラメータ値

object-type パラメータ	内容
column	名前を変更するオブジェクトはカラム
index	名前を変更するオブジェクトはインデックス
constraint	名前を変更するオブジェクトは、一意性、プライマリ・キー、検査、または参照 (外部キー) の制約
procedure	名前を変更するオブジェクトは関数
object-type 指定なし	名前を変更するオブジェクトはテーブル

警告！ sp_iqrename によって名前を変更されているオブジェクト上で、依存オブジェクト (プロシージャ、関数、およびビュー) の定義を適切に変更する必要があります。**sp_iqrename** プロシージャは、従属オブジェクトの定義を自動的には更新しません。これらの定義は手動で変更する必要があります。

説明

sp_iqrename ストアド・プロシージャは、ユーザ作成のテーブル、カラム、インデックス、制約 (一意性、プライマリ・キー、外部キー、検査)、および関数の名前を変更します。

オブジェクト・タイプに対してユニークではない名前を持つオブジェクトの名前を変更しようとすると、**sp_iqrename** はメッセージ「項目はすでに存在します。」を返します。

sp_iqrename は、ビュー、プロシージャ、イベント、またはデータ型の名前を変更できません。メッセージ *object-type* パラメータとして **event** または **datatype** を指定した場合、**sp_iqrename** によって「機能はサポートされていません。」 が返されます。

ALTER TABLE 文と **ALTER INDEX** 文の **RENAME** 句を使用して名前を変更することもできます。『リファレンス： 文とオプション』を参照してください。

例

ユーザ shweta が所有するテーブル titles の名前を books に変更します。

```
sp_iqrename shweta.titles, books
```

テーブル books のカラム id の名前を isbn に変更します。

```
sp_iqrename shweta.books.id, isbn, column
```

テーブル books 上のインデックス idindex の名前を isbnindex に変更します。

```
sp_iqrename books.idindex, isbnindex, index
```

テーブル books 上のプライマリ・キー制約 prim_id の名前を prim_isbn に変更します。

```
sp_iqrename books.prim_id, prim_isbn, constraint
```

sp_iq_reset_identity プロシージャ

指定されたテーブルに関連付けられた Identity/Autoincrement カラムのシード値を、指定された値に設定します。

構文

```
sp_iq_reset_identity  table_name, table_owner, value
```

使用法

table_name、*table owner*、*value* を指定してください。

パーミッション

不要。

説明

Identity/Autoincrement カラムには、自動生成された数字が格納されます。生成された値は、受信データのユニークな識別子です。値は連続したもので、自動生成され、ローがテーブルから削除されても再利用されることはありません。指定されたシード値は、このデフォルトのシード値の代わりとなるもので、データベースのシャットダウンや障害があっても持続します。

参照：

- sp_iqcolumn プロシージャ (403 ページ)
- sp_iqconstraint プロシージャ (410 ページ)
- sp_iqdatatype プロシージャ (419 ページ)
- sp_iqevent プロシージャ (443 ページ)
- sp_iqhelp プロシージャ (448 ページ)
- sp_iqindex および sp_iqindex_alt プロシージャ (456 ページ)
- sp_iqjoinindex プロシージャ (471 ページ)
- sp_iqpkeys プロシージャ (491 ページ)
- sp_iqprocparm プロシージャ (496 ページ)
- sp_iqtable プロシージャ (525 ページ)
- sp_iqview プロシージャ (541 ページ)

sp_iq_reset_identity プロシージャ例

sp_iq_reset_identity を使用する場合は、この例を参照してください。

次の例は、最初のシード値を 50 として Identity カラムを作成します。

```
CREATE TABLE mytable(c1 INT identity)
```

```
call sp_iq_reset_identity('mytable', 'dba', 50)
```

sp_iqrestoreaction プロシージャ

指定された過去の日付の矛盾のない状態にデータベースを戻すために必要なリストア・アクションを示します。

構文

```
sp_iqrestoreaction [ timestamp ]
```

使用法

表 183 : パラメータ

パラメータ	説明
タイムスタンプ	対象の過去の日付を指定します。

パーミッション

DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

説明

データベースを指定のタイムスタンプの矛盾のない状態に戻すことができない場合、sp_iqrestoreaction は、エラーを返します。それ以外の場合、データベースを矛盾のない状態に戻すリストア・アクションを示します。

データベースをリストアできる時点は通常、指定のタイムスタンプの直前に読み書き可能なファイルをバックアップした最後のバックアップ時間と一致します。バックアップは、包括的なファイルまたは読み書き可能なファイルのいずれかに限られます。

出力は、バックアップ時間に基づいた正確な昇順でないことがあります。バックアップ・アーカイブが複数の読み込み専用 dbfile で構成されている場合、複数のロー (同じバックアップ時間とバックアップ ID を持つ) が含まれている可能性があります。

読み込み専用 dbfile または DB 領域を複数回バックアップした場合、リストアには最後のバックアップが使用されます。DB 領域／dbfile の alter ID が、リストアされる最後の読み込み／書き込みバックアップに記録された DB 領域／dbfile の alter ID

と一致していれば、対応するバックアップ時間を指定のタイムスタンプ後にすることもできます。

sp_iqrestoreaction によって、次の結果が返されます。

表 184 : sp_iqrestoreaction のカラム

カラム名	説明
sequence_number	処理を行う順序
backup_id	バックアップ・トランザクションの識別子
backup_archive_list	バックアップのアーカイブ・ファイルのリスト
backup_time	バックアップを行った時間
virtual_type	仮想バックアップの種類："Non-virtual"、"Decoupled"、または "Encapsulated"。
restore_dbspace	空にできます。すべての DB 領域がバックアップ・アーカイブからリストアされることを示します。
restore_dbfile	空にできます。指定の DB 領域のすべての dbfile がバックアップ・アーカイブからリストアされることを示します。
backup_comment	ユーザ・コメント

例
sp_iqrestoreaction の出力例を以下に示します。

```
sequence_number   backup_id   backup_archive_list    backup_time
              1         1192   c:¥¥¥¥temp¥¥¥¥b1       2008-09-23
14:47:40.0
              2         1201   c:¥¥¥¥temp¥¥¥¥b2.inc   2008-09-23
14:48:05.0l
              3         1208   c:¥¥¥¥temp¥¥¥¥b3.inc   2008-09-23
14:48:13.0

virtual_type   restore_dbspace   restore_dbfile   backup_comment
Nonvirtual
Nonvirtual
Nonvirtual
```

sp_iqrowdensity プロシージャ

FP インデックス・レベルのテーブルの内部ローの断片化に関する情報をレポートします。

構文

```
dbo.sp_iqrowdensity ('target ')
```

```
target:(table table-name | (column column-name (...))
```

パーミッション
DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

使用法

パラメータ	説明
table-name	ターゲット・テーブル *table-name* は、指定したテーブル内のすべてのカラムについてレポートします。
column-name	ターゲット・カラム *column-name* は、ターゲット・テーブル内の指定したカラムについてレポートします。ターゲット・カラムは複数指定できますが、キーワードは毎回指定する必要があります。

キーワードである **table** と **column** を指定する必要があります。これらのキーワードでは大文字と小文字が区別されません。

説明
sp_iqrowdensity はデフォルト・インデックス・レベルでのローの断片化を計測します。密度は、既存のテーブルのローに対してインデックスが必要とする最低限のページ数に対する、実際にインデックスが使用するページ数の割合です。このプロシージャは、密度を 0 < *density* < 1 のように示します。たとえば、格納領域として最低 8 ページを必要とするインデックスが実際には 10 ページを占有している場合、密度は .8 です。

レポートされる密度は、デフォルト・インデックスの再作成または再構成によって再利用可能となるディスク・ページの数を示すものではありません。

このプロシージャは、カラムのロー密度を表示するだけで、それ以上のアクションを推奨するものではありません。インデックスの再作成、再構成、再構築を行うかどうかはユーザが判断する必要があります。

例
次のプロシージャは、*SalesOrders* テーブル内のカラム *ID* のロー密度をレポートします。

```
sp_iqrowdensity('column groupo.SalesOrders.ID')
```

Tablename	ColumnName	IndexType	Density
'GROUPO.SalesOrders'	'ID'	'Flat style FP'	'1.0'

参照：
- sp_iqindexfragmentation プロシージャ (461 ページ)

- sp_iqrebuildindex プロシージャ (500 ページ)

sp_iqsharedtempdistrib プロシージャ

マルチプレックスのノードに共有一次領域を分配する方法について、および領域が使用可能か隔離されているかについて診断情報を返します。

セカンダリ・ノードを実行するには、マルチプレックス・グリッド・オプションのライセンスを取得している必要があります。**sp_iqsharedtempdistrib** の構文と詳細な説明については、『Sybase IQ Multiplex の使用』を参照してください。

sp_iqshowpsexe プロシージャ

接続のタスクとリソース使用の優先順位を制御するデータベース・オプションの設定に関する情報を表示します。

構文

```
sp_iqshowpsexe [ connection-id ]
```

パーミッション

DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

使用法

表 185 : パラメータ

パラメータ	説明
connection-id	接続 ID を表す整数。 *connection-id* を指定した場合、**sp_iqshowpsexe** は指定した接続に関する情報のみを返します。*connection-id* を指定しない場合、**sp_iqshowpsexe** はすべての接続に関する情報を返します。 指定の *connection-id* が存在しない場合、**sp_iqshowpsexe** はローを返しません。

説明

sp_iqshowpsexe ストアド・プロシージャは、接続のタスクおよびリソースの使用優先度を管理するデータベース・オプションの設定に関する情報を表示します。これは、データベース管理者がパフォーマンス・チューニングを行うときに役立ちます。

表 186 : sp_iqshowpsexe のカラム

カラム名	説明
connectionid	接続 ID。
application	接続を開いたクライアント・アプリケーションに関する情報。次の **AppInfo** 接続プロパティ情報を含みます。HOST： クライアント・マシン EXE のホスト名 クライアントの実行可能な APPINFO の名前 (Windows のみ)クライアント接続文字列の APPINFO (指定した場合)
userid	接続を開いたユーザのログイン名。
iqgovern_priority	-iqgovern キュー内で待機する各クエリに優先度を割り当てるデータベース・オプション IQGOVERN_PRIORITY の値。デフォルトでは、このオプションは値 2 (MEDIUM) を持ちます。値 1、2、および 3 はそれぞれ、HIGH、MEDIUM、および LOW として表示されます。
max_query_time	オプティマイザで非常に長いクエリを拒否するための制限を設定するデータベース・オプション MAX_QUERY_TIME の値。デフォルトでは、このオプションは無効であり、値 0 を持ちます。
query_row_limit	結果セットの予測サイズに基づいてクエリを拒否するためのロー・スレッショルドを設定する、データベース・オプション QUERY_ROWS_RETURNED_LIMIT の値。デフォルトは 0 で、制限がないことを示します。
query_temp_space_limit	ユーザ・クエリによってテンポラリ IQ DB 領域の使用を制限するデータベース・オプション QUERY_TEMP_SPACE_LIMIT の値 (メガバイト)。デフォルト値は 2,000MB です。
max_cursors	接続が一度に使用できるカーソルの最大数を制限するためにリソース・ガバナを指定するデータベース・オプション MAX_CURSOR_COUNT の値。デフォルト値は 50 です。値 0 は、制限がないことを示します。
max_statements	接続が一度に使用できる準備文の最大数を制限するためにリソース・ガバナを指定するデータベース・オプション MAX_STATEMENT_COUNT の値。デフォルト値は 100 です。値 0 は、制限がないことを示します。

これらのカラム内で参照されているデータベース・オプションの詳細については、『リファレンス： 文とオプション』を参照してください。

注意： **AppInfo** プロパティは、Interactive SQL や Sybase Central などの Open Client または jConnect アプリケーションからは使用できない場合があります。**AppInfo** プロパティを使用できない場合、`application` カラムはブランクになります。

例
接続 ID 2 のタスクとリソース使用の優先順位を制御するデータベース・オプションの設定に関する情報を表示します。

```
sp_iqshowpsexe 2  connectionid    application
           2    HOST=GOODGUY-XP;EXE=C:¥¥Program Files¥¥Sybase¥¥
                IQ-15_3¥¥bin32¥¥dbisqlg.exe;
userid    iqgovern_priority    max_query_time    query_row_limit
DBA       MEDIUM                          0                  0
query_temp_space_limit    max_statements    max_cursors
                  2000                50            100
```

参照：

- CONNECTION_PROPERTY 関数 [システム] (153 ページ)
- sp_iqcontext プロシージャ (412 ページ)

sp_iqspaceinfo プロシージャ

現在のデータベース内の各オブジェクトが使用しているブロック数と、オブジェクトが置かれている DB 領域の名前を表示します。

構文

```
sp_iqspaceinfo [‘main
| [table  table-name | index  index-name] [...] ‘]
```

パーミッション
DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、`EXECUTE` パーミッションが付与される必要があります。

説明
現在のデータベースに関して、オブジェクト名、各オブジェクトが使用するブロック数、DB 領域の名前を表示します。**sp_iqspaceinfo** にはパラメータは不要です。

sp_iqspaceinfo が返した情報は、DB 領域の管理に役立ちます。

マルチプレックス環境では、このプロシージャを使用できます。詳細については、『Sybase IQ Multiplex の使用』を参照してください。

例

iqdemo データベース上で実行される sp_iqspaceinfo ストアド・プロシージャの出力を次に示します。この例では、一部のテーブルやインデックスに対する出力は省略されています。

```
              Name                                 NBlocks    dbspace_name

Contacts                                          19       IQ_SYSTEM_MAIN
SalesOrderItems.DBA.ASIQ_IDX_T205_C5_FP            56      IQ_SYSTEM_MAIN
Contacts.DBA.ASIQ_IDX_T206_C10_FP                  55      IQ_SYSTEM_MAIN
Contacts.DBA.ASIQ_IDX_T206_C1_FP                   61      IQ_SYSTEM_MAIN
...
Contacts.DBA.ASIQ_IDX_T206_C9_FP                   55      IQ_SYSTEM_MAIN
Contacts.DBA.ASIQ_IDX_T206_I11_HG                  19      IQ_SYSTEM_MAIN
Customers                                         20       IQ_SYSTEM_MAIN
Customers.DBA.ASIQ_IDX_T207_C1_FP                  61      IQ_SYSTEM_MAIN
Customers.DBA.ASIQ_IDX_T207_C2_FP                  55      IQ_SYSTEM_MAIN
...
Customers.DBA.ASIQ_IDX_T207_I10_HG                 19      IQ_SYSTEM_MAIN
...
```

参照：

- sp_iqindexinfo プロシージャ (463 ページ)
- sp_iqdbspace プロシージャ (424 ページ)
- sp_iqdbspaceinfo プロシージャ (426 ページ)

sp_iqspaceused プロシージャ

IQ ストア、IQ テンポラリ・ストア、および IQ グローバルと IQ ローカルの共有テンポラリ・ストアの、空き領域と使用領域に関する情報を表示します。

構文

```
sp_iqspaceused(out mainKB unsigned bigint,
                                  out mainKBUsed unsigned bigint,
                                  out tempKB unsigned bigint,
                                    out tempKBUsed unsigned bigint,
out shTempTotalKB unsigned bigint,
                                out shTempTotalKBUsed unsigned bigint,
out shTempLocalKB unsigned bigint,
                                    out shTempLocalKBUsed unsigned
bigint)
```

パーミッション

DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

使用法

sp_iqspaceused は、unsigned bigint の out パラメータとして 8 つの値を返します。このシステム・ストアド・プロシージャは、ユーザ定義のストアド・プロシー

ジャから呼び出し、メインおよびテンポラリ IQ ストアの領域の使用量を確認できます。

説明
sp_iqspaceused は、**sp_iqstatus** によって提供された情報のサブセットを返しますが、計算に使用する SQL 変数内の情報をユーザが返すこともできます。

表 187 : sp_iqspaceused のカラム

カラム名	説明
mainKB	IQ メイン・ストアの領域の合計 (キロバイト)。
mainKBUsed	データベースが使用している IQ メイン・ストア領域のキロバイト数 (セカンダリ・マルチプレックス・ノードは '(Null)' を返す)。
tempKB	IQ テンポラリ・ストアの領域の合計 (キロバイト)。
tempKBUsed	IQ テンポラリ・ストアの領域の合計 (キロバイト)。
shTempTotalKB	IQ グローバル共有テンポラリ・ストアの領域の合計 (キロバイト)。
shTempTotalKBUsed	IQ グローバル共有テンポラリ・ストアの領域の合計 (キロバイト)。(セカンダリ・マルチプレックス・ノードは '(Null)' を返します。)
shTempLocalKB	IQ ローカル共有テンポラリ・ストアの領域の合計 (キロバイト)。
shTempLocalKBUsed	データベースが使用している IQ ローカル共有テンポラリ・ストアの領域 (キロバイト)。

例
sp_iqspaceused には 8 つの出力パラメータが必要です。次の例では、ユーザ定義のストアド・プロシージャ **myspace** を作成します。このストアド・プロシージャは、8 つの出力パラメータを宣言し、**sp_iqspaceused** を呼び出します。

```
create procedure dbo.myspace()
begin
  declare mt unsigned bigint;
  declare mu unsigned bigint;
  declare tt unsigned bigint;
  declare tu unsigned bigint;
  declare gt unsigned bigint;
  declare gu unsigned bigint;
  declare lt unsigned bigint;
  declare lu unsigned bigint;
  call sp_iqspaceused(mt,mu,tt,tu,gt,gu,lt,lu);
  select cast(mt/1024 as unsigned bigint) as mainMB,
         cast(mu/1024 as unsigned bigint) as mainusedMB,
        mu*100/mt as mainPerCent,
         cast(tt/1024 as unsigned bigint) as tempMB,
```

```
          cast(tu/1024 as unsigned bigint) as tempusedMB,
         tu*100/tt as tempPerCent;
          cast(gt/1024 as unsigned bigint) as shTempTotalKB,
          cast(gu/1024 as unsigned bigint) as shTempTotalKBUsed,
         gu*100/gt as globalshtempPerCent;
          cast(lt/1024 as unsigned bigint) as shTempLocalMB,
          cast(lu/1024 as unsigned bigint) as shTempLocalKBUsed,
         lu*100/lt as localshtempPerCent; end
```

sp_iqspaceused の出力を表示するには、プロシージャ **myspace** を実行します。

```
myspace
```

sp_iqstatistics プロシージャ

使用可能な各統計または指定の統計について、シリアル番号、名前、説明、値、および単位指定子を返します。

構文

```
sp_iqstatistics [ stat_name ]
```

使用法

パラメータ	説明
stat_name	(オプション) 統計の名前を指定する VARCHAR パラメータ。

パーミッション

DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

説明

stat_name を指定すると、**sp_iqstatistics** は、指定の統計について 1 つのローを返すか、または名前が無効な場合はローを返しません。パラメータを指定せずに呼び出すと、**sp_iqstatistics** はすべての統計を返します。

結果セット

カラム名	データ型	説明
stat_num	UNSIGNED INTEGER	統計のシリアル番号
stat_name	VARCHAR(255)	統計の名前
stat_desc	VARCHAR(255)	統計の説明
stat_value	LONG VARCHAR	統計の値
stat_unit	VARCHAR(128)	単位指定子

次の統計が返されます。

stat_num	stat_name	stat_desc	stat_unit
0	CpuTotalTime	最後にサーバを起動してから IQ サーバによって消費された合計 CPU 時間 (秒)	Second
1	CpuUserTime	最後にサーバを起動してから IQ サーバによって消費された CPU ユーザ時間 (秒)	Second
2	CpuSystemTime	最後にサーバを起動してから IQ サーバによって消費された CPU システム時間 (秒)	Second
3	ThreadsFree	空き IQ スレッドの数	該当なし
4	ThreadsInUse	使用中の IQ スレッドの数	該当なし
5	MemoryAllocated	割り付けられたメモリ (メガバイト)	MB
6	MemoryMaxAllocated	割り付けられた最大メモリ (メガバイト)	MB
7	MainCacheCurrentSize	メイン・キャッシュの現在のサイズ (メガバイト)	MB
8	MainCacheFinds	メイン・キャッシュの検索要求の総数	該当なし
9	MainCacheHits	メイン・キャッシュのヒットの総数	該当なし
10	MainCachePagesPinned	メイン・キャッシュの固定ページの数	ページ
11	MainCachePagesPinned-Percentage	メイン・キャッシュの固定ページの割合	%
12	MainCachePagesDirtyPer-centage	メイン・キャッシュのダーティ・ページの割合	%
13	MainCachePagesInUsePer-centage	メイン・キャッシュの使用中ページの割合	%
14	TempCacheCurrentSize	テンポラリ・キャッシュの現在のサイズ (メガバイト)	MB
15	TempCacheFinds	テンポラリ・キャッシュの検索要求の総数	該当なし

stat_num	stat_name	stat_desc	stat_unit
16	TempCacheHits	テンポラリ・キャッシュのヒットの総数	該当なし
17	TempCachePagesPinned	テンポラリ・キャッシュの固定ページの数	ページ
18	TempCachePagesPinned-Percentage	テンポラリ・キャッシュの固定ページの割合	%
19	TempCachePagesDirtyPer-centage	テンポラリ・キャッシュのダーティ・ページの割合	%
20	TempCachePagesInUse-Percentage	テンポラリ・キャッシュの使用中ページの割合	%
21	MainStoreDiskReads	メイン・ストアから読み込まれたデータ量 (キロバイト)	KB
22	MainStoreDiskWrites	メイン・ストアから書き込まれたデータ量 (キロバイト)	KB
23	TempStoreDiskReads	メイン・ストアから読み込まれたデータ量 (キロバイト)	KB
24	TempStoreDiskWrites	メイン・ストアから書き込まれたデータ量 (キロバイト)	KB
25	ConnectionsTotalConnec-tions	サーバ起動後の総接続数	該当なし
26	ConnectionsTotalDison-nections	サーバ起動後の総切断数	該当なし
27	ConnectionsActive	アクティブな接続の数	該当なし
28	OperationsWaiting	IQ リソース・ガバナを待機している処理の数	該当なし
29	OperationsActive	IQ リソース・ガバナによって認められたアクティブな同時処理の数	該当なし
30	OperationsActiveLoadTa-bleStatements	アクティブな LOAD TABLE 文の数	該当なし

例
1 つの統計 (総 CPU 時間) を表示します。

```
sp_iqstatistics 'CPUTotalTime'
```

MainCache% のすべての統計を表示します。

```
SELECT * from sp_iqstatistics() WHERE stat_name LIKE 'MainCache%'
```

sp_iqstatus プロシージャ

現在のデータベースについて、さまざまな Sybase IQ ステータス情報を表示します。

構文

```
sp_iqstatus
```

パーミッション

DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

説明

現在のデータベースについて、ステータス情報を表示します。データベース名、作成日、ページ・サイズ、DB 領域セグメントの数、ブロック使用状況、バッファ使用状況、I/O、バックアップ情報などが含まれます。

sp_iqstatus は、メインおよびテンポラリのストアの領域不足状態を表示します。任意のストアが領域不足状態に陥った場合、**sp_iqstatus** はそのストアの領域不足状態表示欄に Y を表示します。

sp_iqspaceused は、**sp_iqstatus** によって提供された同じ情報のサブセットを返しますが、計算に使用する SQL 変数内の情報をユーザが返すこともできます。

接続を切断することによって再利用できる領域を表示するには、**sp_iqstatus** を使用して、返された 2 つのローからの結果を追加します。

```
(DBA)> select * from sp_iqstatus() where name like '%Versions:%'
Execution time: 6.25 seconds
Name                 Value
---------------------------
Other Versions: 2 = 1968Mb
Active Txn Versions: 1 = C:2175Mb/D:2850Mb

(First 2 rows)
```

上記の出力例には、1 つのアクティブな書き込みトランザクションによって 2175MB のデータが作成され、2850MB のデータが破棄されたことが示されています。トランザクションで消費され、まだ解放されていないデータの合計サイズは、4818MB (1968MB + 2850MB = 4818MB) です。

sp_iqstatus は、次のチェックポイントで割り付けが解除されるブロックを示しません。ただし、これらのブロックは、**sp_iqdbspace** 出力にタイプ X として表示されます。

マルチプレックス環境では、このプロシージャを使用できます。詳細については、『Sybase IQ Multiplex の使用』を参照してください。

例

注意： 出力内容をわかりやすくするため、次の例は iqdemo データベース内のオブジェクトを示しています。iqdemo には、iq_main というサンプルのユーザ DB 領域が含まれていますが、この領域はユーザ独自のデータベースには存在しない場合があります。

sp_iqstatus ストアド・プロシージャの出力を次に示します。

```
Sybase IQ (TM)                Copyright (c) 1992-2010 by Sybase, Inc.
                                    All rights reserved.
Version:                          15.3.0/090416/P/MS/Windows/2010/
                                    32bit/2010-04-16 02:11:41
Time Now:                           2010-04-21 13:48:22.319
Build Time:                         2010-04-16 02:15:39
File Format:                        23 on 03/18/1999
Server mode:                        IQ Server
Catalog Format:                     2
Stored Procedure Revision:          1
Page Size:                          131072/8192blksz/16bpp
Number of Main DB Files :           2
Main Store Out Of Space:            N
Number of Local Temp DB Files :     1
Local Temp Store Out Of Space:      N
DB Blocks:  1-3200                  IQ_SYSTEM_MAIN
DB Blocks:  1045440-1055039         iq_main
Local Temp Blocks:  1-1600          IQ_SYSTEM_TEMP
Create Time:                        2009-04-03 11:30:20.674
Update Time:                        2009-04-03 11:34:33.040
Main IQ Buffers:                    255, 32Mb
Temporary IQ Buffers:               191, 24Mb
Main IQ Blocks Used:              5915 of 11200, 52%=46Mb, Max
Block#:105278
Local Temporary IQ Blocks Used:     65 of 800, 8%=0Mb, Max
Block#: 0
Main Reserved Blocks Available:     1600 of 1600, 100%=6Mb
Local Temporary Reserved Blocks Available:     6400 of 6400,
100%=50Mb
IQ Dynamic Memory:                  Current: 69mb, Max: 70mb
Main IQ Buffers:                    Used: 17, Locked: 0
Temporary IQ Buffers:               Used: 4, Locked: 0
Main IQ I/O:                      I: L1581/P14 O: C3/D163/P161
D:34 C:97.1
Temporary IQ I/O:                 I: L6627/P0 O: C1086/D1166/P83
                                    D:1082 C:100.0
Other Versions:                     0 = 0Mb
Active Txn Versions:                0 = C:0Mb/D:0Mb
Last Full Backup ID:                0
Last Full Backup Time:
Last Backup ID:                     0
```

```
Last Backup Type:                          None
Last Backup Time:
DB Updated:                                1
Blocks in next ISF Backup:                 0 Blocks: =0Mb
Blocks in next ISI Backup:                 0 Blocks: =0Mb
DB File Encryption Status:                 OFF
```

`Main IQ I/O` と `Temporary IQ I/O` 出力コードの意味は、次のとおりです。

- `I`：入力項目
- `L`： 読み込まれた論理ページ数 (Finds)
- `P`： 読み込まれた物理ページ数
- `O`： 出力
- `C`：作成されたページ数
- `D`： ダーティ・ページ
- `P`： 物理的書き込み
- `D`： 破損したページ数
- `C`：圧縮率

参照：

- sp_iqtransaction プロシージャ (531 ページ)
- sp_iqversionuse プロシージャ (539 ページ)

sp_iqsysmon プロシージャ

Sybase IQ の複数のコンポーネントをモニタします。モニタ対象には、バッファ・キャッシュ、メモリ、スレッド、ロック、入出力機能、CPU 使用率の管理を含みます。

バッチ・モードでの構文

```
sp_iqsysmon start_monitor
sp_iqsysmon stop_monitor [, “section(s)” ]
or
sp_iqsysmon “time-period” [, “section(s)” ]
```

ファイル・モードでの構文

```
sp_iqsysmon start_monitor, ‘filemode’ [, ”monitor-options” ]
sp_iqsysmon stop_monitor
```

パーミッション

DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、`EXECUTE` パーミッションが付与される必要があります。

バッチ・モードでの使用法

パラメータ	説明
start_monitor	モニタリングを開始します。
stop_monitor	モニタリングを停止し、レポートを表示します。
time-period	モニタリングの期間。HH:MM:SS の形式で指定してください。
section(s)	**sp_iqsysmon** によって表示される 1 つ以上のセクションの省略形。複数のセクションを指定する場合、セクションの省略形をスペースで区切り、そのリストを一重引用符または二重引用符で囲む必要があります。デフォルトでは、すべてのセクションが表示されます。 IQ ストアに関連するセクションの場合、セクションの省略形に "m" または "t" のプレフィクスを付けることによって、それぞれメイン・ストアまたはテンポラリ・ストアを指定できます。プレフィックスを使用しない場合、両方のストアがモニタされます。たとえば、"mbufman" を指定した場合、IQ メイン・ストア・バッファ・マネージャのみがモニタされます。"mbufman tbufman" または "bufman" を指定した場合、メイン・ストアとテンポラリ・ストアの両方のバッファ・マネージャがモニタされます。

表 188 : sp_iqsysmon レポート・セクションの省略形

レポート・セクションまたは IQ コンポーネント	省略形
バッファ・マネージャ	(m/t)bufman
バッファ・プール	(m/t)bufpool
プリフェッチ管理	(m/t)prefetch
フリー・リスト管理	(m/t)freelist
バッファ割り付け	(m/t)bufalloc
メモリ管理	memory
スレッド管理	threads
CPU の使用率	cpu
トランザクション管理	txn
サーバ・コンテキスト統計	server
カタログの統計	catalog

注意： Sybase IQ コンポーネント・ディスク I/O およびロック・マネージャは、現在 **sp_iqsysmon** ではサポートされていません。

ファイル・モードでの使用法

表 189 : パラメータ

パラメータ	説明
start_monitor	モニタリングを開始します。
stop_monitor	モニタリングを停止し、残りの出力をログ・ファイルに書き込みます。
filemode	**sp_iqsysmon** をファイル・モードで実行することを指定します。ファイル・モードでは、モニタリング期間中一定の時間間隔でサンプリングした統計情報が示されます。デフォルトでは、*dbname.connid-iqmon* という名前のログ・ファイルに出力が書き込まれます。**file_suffix** オプションを使用して、出力ファイルのサフィックスを変更します。**file_suffix** オプションの説明については、*monitor_options* パラメータを参照してください。
monitor_options	*monitor _options* 文字列

monitor _options 文字列には 1 つ以上のオプションを含めることがでます。

表 190 : monitor_options 文字列オプション

monitor_options 文字列オプション	説明
-interval*seconds*	レポート間隔を秒単位で指定します。モニタ統計のサンプリング情報が、一定の時間間隔でログ・ファイルに出力されます。**-interval** オプションを指定しない場合のデフォルトは 60 秒間隔です。最小レポート間隔は 2 秒です。このオプションで指定した間隔が無効であるか、2 秒未満である場合、2 秒間隔に設定されます。 最初の記録では、サーバの起動からのカウンタが示されます。それ以降の記録では、前の表示との差が示されます。通常は、パフォーマンスに問題がある期間 (クエリ実行時や特定の時間帯) に 60 秒のデフォルト間隔でモニタを実行すると、有意な結果を得ることができます。間隔が短すぎると、意味のある結果を取得できないことがあります。ジョブ時間に見合った間隔を指定してください。通常は 60 秒で十分です。

monitor_options 文字列オプション	説明
-file_suffix suffix	`dbname.connid-suffix` という名前のモニタリング出力ファイルを作成します。**-file_suffix** オプションを指定しないと、サフィックスはデフォルトで `iqmon` に設定されます。**-file_suffix** オプションを指定した場合で、サフィックスを指定しないか、サフィックスとしてブランクの文字列を指定したときは、サフィックスは使用されません。
-append または **-truncate**	前者は既存の出力ファイルに追加、後者は既存の出力ファイルをトランケートするよう **sp_iqsysmon** に指示します。デフォルトでは、トランケートされます。両方のオプションを指定した場合、文字列内で後ろに指定されているオプションが有効になります。
-section*section(s)*	モニタ・ログ・ファイルに書き込む 1 つ以上のセクションの省略形を指定します。デフォルトでは、すべてのセクションが書き込まれます。ファイル・モードのセクション・リストで指定する省略形は、バッチ・モードで使用する省略形と同じです。複数のセクションを指定する場合、セクションの省略形をスペースで区切る必要があります。 セクションなしで **-section** オプションを指定した場合、どのセクションもモニタされません。無効なセクション省略形は無視され、IQ メッセージ・ファイル内に警告が示されます。

使用構文の例

表 191 : sp_iqsysmon の使用例

構文	結果
sp_iqsysmon start_monitor sp_iqsysmon stop_monitor	バッチ・モードでモニタを開始し、メイン・ストアおよびテンポラリ・ストアのすべてのセクションを表示します。
sp_iqsysmon start_monitor sp_iqsysmon stop_monitor “mbufman mbufpool”	バッチ・モードでモニタを開始し、メイン・ストアのバッファ・マネージャとバッファ・プールの統計情報を表示します。
sp_iqsysmon “00:00:10”, “mbufpool memory”	バッチ・モードでモニタを 10 秒間実行し、10 秒間の経過後に集約した統計情報を表示します。

構文	結果
sp_iqsysmon start_monitor, ‘filemode’, “-interval 5 -sections mbufpool memory” sp_iqsysmon stop_monitor	ファイル・モードでモニタを開始し、メイン・バッファ・プールとメモリ・マネージャの統計情報を 5 秒間隔でログ・ファイルに書き込みます。

説明

sp_iqsysmon ストアド・プロシージャは、Sybase IQ の複数のコンポーネントをモニタします。モニタ対象には、バッファ・キャッシュ、メモリ、スレッド、ロック、入出力機能、CPU の使用率の管理を含みます。

sp_iqsysmon プロシージャは、次の 2 つのモニタリング・モードをサポートします。

- バッチ・モード
  バッチ・モードでは、**sp_iqsysmon** はモニタを開始してから停止するまでの期間、または *time-period* パラメータで指定した期間、モニタ統計情報を収集します。モニタリング期間の経過後、**sp_iqsysmon** は集約した統計情報のリストを表示します。
  バッチ・モードの **sp_iqsysmon** は、Adaptive Server Enterprise プロシージャ **sp_sysmon** と同様に動作します。
- ファイル・モード
  ファイル・モードでは、**sp_iqsysmon** はモニタを開始してから停止するまで、一定の時間間隔でサンプリングした統計情報をログ・ファイルに書き込みます。
  ファイル・モードの最初の記録では、サーバの起動からのカウンタが示されます。それ以降の記録では、前の表示との差が示されます。
  ファイル・モードの **sp_iqsysmon** は **IQ UTILITIES** コマンド **START MONITOR** および **STOP MONITOR** インタフェースと同様のものです。

バッチ・モードの例

10 分後にモニタ情報を出力します。

```
sp_iqsysmon “00:10:00”
```

5 分後に、**sp_iqsysmon** レポートのメモリ・マネージャのセクションのみを出力します。

```
sp_iqsysmon “00:05:00”, memory
```

モニタを開始した後、2 つのプロシージャと 1 つのクエリを実行し、モニタを停止して、レポートのバッファ・マネージャのセクションのみを出力します。

```
sp_iqsysmon start_monitor
go
execute proc1
```

```
go
execute proc2
go
select sum(total_sales) from titles
go
sp_iqsysmon stop_monitor, bufman
go
```

20 分後に、レポートのメイン・バッファ・マネージャおよびメイン・バッファ・プールのセクションのみを出力します。

```
sp_iqsysmon “00:02:00”, “mbufman mbufpool”
```

ファイル・モードの例

モニタを開始してから停止するまで、2 秒ごとに情報をトランケートし、ログ・ファイルに書き込みます。

```
sp_iqsysmon start_monitor, ‘filemode’, ‘-interval 2’ . . .
sp_iqsysmon stop_monitor
```

`dbname.connid-testmon` という名前の ASCII ファイルに対して、メイン・バッファ・マネージャおよびメモリ・マネージャのセクションのみの出力を追加します。データベース `iqdemo` について、ファイル `iqdemo.2-testmon` に結果を書き込みます。

```
sp_iqsysmon start_monitor, ‘filemode’,
“-file_suffix testmon -append -section mbufman memory”
.
.
.
sp_iqsysmon stop_monitor
```

例

バッチ・モードでモニタを 10 秒間実行し、10 秒間の経過後に集約した統計情報を表示します。

```
sp_iqsysmon “00:00:10”, “mbufpool memory”
============================== Buffer Pool (Main)
============================== STATS-
NAME         TOTAL  NONE  BTREEV  BTREEF  BV  VDO  DBEXT  DBID  SORT
MovedToMRU         0      0     0       0       0   0    0      0     0
MovedToWash        0      0     0       0       0   0    0      0     0
RemovedFromLRU     0      0     0       0       0   0    0      0     0
RemovedFromWash    0      0     0       0       0   0    0      0     0
RemovedInScanMode  0      0     0       0       0   0    0      0     0
STORE  GARRAY  BARRAY  BLKMAP  HASH  CKPT  BM  TEST  CMID  RIDCA  LOB
0      0       0       0       0     0     0   0     0     0      0
0      0       0       0       0     0     0   0     0     0      0
0      0       0       0       0     0     0   0     0     0      0
0      0       0       0       0     0     0   0     0     0      0
0      0       0       0       0     0     0   0     0     0      0
STATS-NAME                  VALUE  Pages                    127  (
100.0 %)  InUse                          4     ( 3.1 %)
Dirty                          1     ( 0.8 %)
```

```
Pinned                          0     ( 0.0 %)
Flushes                         0  FlushedBufferCount             0
GetPageFrame                    0  GetPageFrameFailure            0
GotEmptyFrame                   0  Washed                          0
TimesSweepersWoken              0

washTeamSize                 0 WashMaxSize                 26   ( 20.5
%) washNBuffers                      4     ( 3.1 %)
washNDirtyBuffers               1     ( 0.8
%)
                   washSignalThreshold            3      ( 2.4 %)
washNActiveSweepers             0 washIntensity                  1
==============================  Memory Manager
==============================  STATS-NAME                    VALUE
MemAllocated                  43616536     (  42594 KB)
MemAllocatedMax               43735080     (  42710 KB)
MemAllocatedEver              0            (      0 KB)
MemNAllocated                 67079 MemNAllocatedEver           0
MemNTimesLocked               0 MemNTimesWaited
0            (   0.0 %)
```

sp_iqtable プロシージャ

データベース内のテーブルに関する情報を表示します。

構文 1

```
sp_iqtable [ table_name ], [table_owner ], [table_type ]
```

構文 2

```
sp_iqtable [table_name='tablename'],
[table_owner='tableowner' ],[table_type='tabletype' ]
```

パーミッション

DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

使用法：構文 1

最初の 2 つのパラメータのいずれかを指定せずに、その次のパラメータを指定する場合は、指定しないパラメータの位置に NULL を指定する必要があります。たとえば、`sp_iqtable NULL,NULL,TEMP` および `sp_iqtable NULL,dbo,SYSTEM` とします。

注意： 構文 1 では、*table_type* の値である ALL と VIEW を一重引用符で囲む必要があります。

使用法：構文 2

パラメータはどのような順番で指定しても構いません。一重引用符で囲みます。

table_type パラメータに指定できる値を次に示します。

表 192 : sp_iqtable の table_type の値

table_type の値	表示される情報
SYSTEM	システム・テーブル
TEMP	グローバル・テンポラリ・テーブル
VIEW	ビュー
ALL	IQ テーブル、システム・テーブル、ビュー
その他の値	IQ テーブル

説明
いずれかのパラメータを指定すると、そのパラメータに合致するテーブルだけが返されます。 複数のパラメータを指定すると、指定されたすべてのパラメータにより結果がフィルタされます。 パラメータを指定しない場合は、データベース内のすべての Sybase IQ テーブルが返されます。 ローカル・テンポラリ・テーブルの名前を返す方法はありません。

表 193 : sp_iqtable のカラム

カラム名	説明
table_name	テーブル名。
table_type	BASE ― ベース・テーブル。 MAT VIEW - マテリアライズド・ビュー(SA テーブルのみ)。 GBL TEMP - グローバル・テンポラリ・テーブル。 PARTITION - テーブル・パーティション (このテーブルは内部でのみ使用可能であり、Sybase IQ ユーザが使用することはできない)。 VIEW ― ビュー。 JVT ― ジョイン仮想テーブル。
table_owner	テーブルの所有者
server_type	IQ ― IQ ストアで作成されたオブジェクト。 SA ― SA ストアで作成されたオブジェクト。 ビューはすべて SA ストアで作成されます。

カラム名	説明
location	TEMP ― IQ テンポラリ・ストア。 MAIN ― IQ ストア。 SYSTEM ― カタログ・ストア。
dbspace_id	テーブルが存在する DB 領域の名前。
isPartitioned	カラムが、分割されたテーブルに属しており、かつテーブル・パーティションの DB 領域と異なる DB 領域を持つ 1 つ以上のパーティションを持っている場合は 'Y'。カラムのテーブルが分割されていないか、またはカラムの各パーティションがテーブル・パーティションと同じ DB 領域に存在する場合は 'N'。
remarks	**COMMENT** 文で追加されたユーザ・コメント。
table_constraints	テーブルに対する制約。

参照：

- sp_iqcolumn プロシージャ (403 ページ)
- sp_iqconstraint プロシージャ (410 ページ)
- sp_iqdatatype プロシージャ (419 ページ)
- sp_iqevent プロシージャ (443 ページ)
- sp_iqhelp プロシージャ (448 ページ)
- sp_iqindex および sp_iqindex_alt プロシージャ (456 ページ)
- sp_iqjoinindex プロシージャ (471 ページ)
- sp_iqpkeys プロシージャ (491 ページ)
- sp_iqprocparm プロシージャ (496 ページ)
- sp_iq_reset_identity プロシージャ (505 ページ)
- sp_iqview プロシージャ (541 ページ)

sp_iqtable プロシージャ例

sp_iqtable を使用する場合は、この例を参照してください。

次の構文は、いずれもテーブル `Departments` に関する情報を返します。

```
sp_iqtable ('Departments')
```

```
sp_iqtable table_name='Departments'
```

Table_name	Table_type	Table_owner	Server_type	location
Departments	BASE	GROUPO	IQ	Main

dbspace_id	isPartitioned	Remarks	table_constraints
16387	N	(NULL)	(NULL)

次の構文は、いずれも GROUPO が所有するすべてのテーブルを返します。

```
sp_iqtable NULL,GROUPO
```

```
sp_iqtable table_owner='GROUPO'
```

Table_name	Table_type	Table_owner	Server_type	location
Contacts	BASE	GROUPO	IQ	Main
Customers	BASE	GROUPO	IQ	Main
Departments	BASE	GROUPO	IQ	Main
Employees	BASE	GROUPO	IQ	Main
FinancialCodes	BASE	GROUPO	IQ	Main
FinancialData	BASE	GROUPO	IQ	Main
Products	BASE	GROUPO	IQ	Main
SalesOrders	BASE	GROUPO	IQ	Main
SalesOrderItems	BASE	GROUPO	IQ	Main

dbspace_id	isPartitioned	Remarks	table_constraints
16387	N	(NULL)	(NULL)
16387	N	(NULL)	(NULL)
16387	N	(NULL)	(NULL)
16387	N	(NULL)	(NULL)
16387	N	(NULL)	(NULL)
16387	N	(NULL)	(NULL)
16387	N	(NULL)	(NULL)
16387	N	(NULL)	(NULL)

dbspace_id	isPartitioned	Remarks	table_constraints
16387	N	(NULL)	(NULL)

sp_iqtablesize プロシージャ

指定したテーブルのサイズを返します。

構文

```
sp_iqtablesize ( table_owner.table_name )
```

パーミッション

DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

説明

テーブルの合計サイズを、ブロック、キロバイト、Nblocks (IQ ブロック) で返します。また、メモリにテーブルを保持するために必要なページ数と、(ディスク上で) テーブルを圧縮した場合に圧縮された IQ ページ数も返します。このプロシージャには、必ず *table_name* パラメータを指定してください。*table_name* の所有者がこれを実行する場合は、*table_owner* パラメータを指定する必要はありません。

表 194 : sp_iqtablesize のカラム

カラム名	説明
Ownername	所有者の名前
Tablename	テーブルの名前。
Columns	テーブルのカラム数
KBytes	物理テーブル・サイズ (KB)
Pages	メモリ内にテーブルを保持するために必要な IQ ページの数
CompressedPages	(ディスク上で) テーブルを圧縮した場合に圧縮される IQ ページの数
NBlocks	IQ ブロックの数

Pages は、テーブルの IQ ページの合計数です。ページの測定単位は、IQ ページ・サイズです。すべてのメモリ内バッファ (IQ バッファ・キャッシュ内のバッファ) は同じサイズです。

ディスク上の IQ ページは圧縮されています。ディスク上の各 IQ ページは、1 から 16 ブロックを使用します。IQ ページ・サイズが 128KB の場合、IQ ブロック・サ

イズは 8KB です。この場合、ディスク上の個別のページは 8、16、24、32、40、48、56、64、72、80、88、96、104、112、120、128KB のいずれかです。

KBytes の値をページ・サイズで除算すると、ディスク上のページ・サイズの平均がわかります。

注意： Sybase IQ は常に、ブロックではなくページ全体を読み書きします。 たとえば、個別のページが 88K まで圧縮された場合、IQ は 1 回の I/O で 88K を読み書きします。平均的なページは、3 分の 1 から 2 分の 1 に圧縮されます。

NBlocks は、Kbytes を IQ ブロック・サイズで除算したものです。

CompressedPages は、圧縮されたページの数です。 たとえば、Pages が 1000 で、CompressedPages が 992 である場合、1000 ページ中 992 ページが圧縮されたことになります。大部分のページは圧縮されるため、CompressedPages を Pages で割った結果は、通常ほぼ 100% になります。 Sybase IQ は空のページを書き込まないため、空のページは圧縮されません。 IQ ページは、ページの満杯度にかかわらず、高い圧縮率で圧縮されます。

例

```
call sp_iqtablesize ('dba.emp1')
```

Ownername	Tablename	カラム	KBytes	Pages	CompressedPages	NBlocks
DBA	emp1	4	1504	24	14	188

sp_iqtableuse プロシージャ

負荷によってアクセスされるテーブルの使用状況の情報を詳細にレポートします。

構文

```
sp_iqtableuse
```

パーミッション

DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

説明

SYSTEM で作成されたテーブルはレポートされません。

表 195 : sp_iqtableuse のカラム

カラム名	説明
TableName	テーブル名
Owner	テーブル所有者のユーザ名
UID**	テーブルのユニークな識別子
LastDT	前回のアクセスの日時
NRef	クエリ参照の数

**UID はシステムが割り当てた番号であり、テーブルのインスタンスをユニークに識別します (インスタンスはオブジェクト作成時に定義されます)。

参照：

- sp_iqcolumnuse プロシージャ (405 ページ)
- sp_iqindexadvice プロシージャ (460 ページ)
- sp_iqindexuse プロシージャ (469 ページ)
- sp_iqunusedcolumn プロシージャ (536 ページ)
- sp_iqunusedindex プロシージャ (537 ページ)
- sp_iqunusedtable プロシージャ (538 ページ)
- sp_iqworkmon プロシージャ (547 ページ)

sp_iqtransaction プロシージャ

トランザクションとバージョンに関する情報を表示します。

構文

```
sp_iqtransaction
```

パーミッション

DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

説明

sp_iqtransaction は、Sybase IQ トランザクション・マネージャの各トランザクション制御ブロックについて 1 つのローを返します。 Name、Userid、ConnHandle の各カラムは、それぞれ **Name**、**Userid**、**Number** の接続プロパティに対応しています。 ローは TxnID の順に並べられます。

sp_iqtransaction の出力には、実行中のトランザクションを持たない接続に対応するローは含まれません。すべての接続を表示するには、**sp_iqconnection** を使用します。

注意： **sp_iqtransaction** を使用して、他のユーザによるテーブルへの書き込みをブロックしているユーザを確認することもできますが、この場合は **sp_iqlocks** を使用することをおすすめします。

マルチプレックス環境では、このプロシージャを使用できます。詳細については、『Sybase IQ Multiplex の使用』を参照してください。

表 196 : sp_iqtransaction のカラム

カラム名	説明
Name	アプリケーションの名前です。
Userid	接続のユーザ ID。
TxnID	このトランザクション制御ブロックのトランザクション ID。トランザクション ID は、begin transaction の間に割り当てられます。これは、BeginTxn、CmtTxn、および PostCmtTxn メッセージによって `.iqmsg` ファイルに表示されるトランザクション ID、また、データベースが開かれたときにログ記録される Txn ID Seq と同じです。
CmtID	トランザクションがコミットしたときにトランザクション・マネージャによって割り当てられる ID。アクティブなトランザクションでは 0 です。
VersionID	シンプレックス・データベースでは、VersionID としてゼロが表示されます。 マルチプレックス・コーディネータでは、VersionID はアクティブなトランザクションの TxnID、コミット済みのトランザクションの CmtID と同一です。 マルチプレックスのセカンダリ・サーバでは、VersionID は、マルチプレックス・コーディネータでデータベース・バージョンを作成したトランザクションの CmtID です。 これは、マルチプレックス・データベース内のすべてのノードに対してデータベース・バージョンを一意に識別するために、Sybase IQ のメモリ内カタログと IQ トランザクション・マネージャで使用されます。

カラム名	説明
State	トランザクション制御ブロックの状態。この変数は内部の Sybase IQ 実装の詳細を反映するもので、将来的に変更される可能性があります。このマニュアルの作成時点では、トランザクション状態には、NONE、ACTIVE、ROLLING_BACK、ROLLED_BACK、COMMITTING、COMMITTED、APPLIED があります。 NONE、ROLLING_BACK、ROLLED_BACK、COMMITTING、APPLIED は、非常に短い一時的な状態です。 ACTIVE は、トランザクションが読み込み専用であることを示す。 COMMITTED は、トランザクションが完了し、APPLIED への遷移待ちである状態を示します。APPLIED の状態では、すべてのトランザクションに認識されないバージョンは、ガーベジ・コレクションの対象となります。 トランザクションの状態が ROLLED_BACK、COMMITTED、または APPLIED になると、開いているカーソルで保持されているロック以外のロックは所有できなくなります。
ConnHandle	接続の ID 番号。
IQConnID	.iqmsg ファイル内のすべてのメッセージの一部として表示される 10 桁の接続 ID。これは、サーバ・セッション内でユニークな、単純増加する整数です。
MainTableKBCr	このトランザクションによって作成された IQ ストアの領域 (KB 単位)。
MainTableKBDr	このトランザクションによって削除済みの IQ ストアの領域のうち、他のデータベース・バージョン、またはこのトランザクションの他のセーブポイントでこの領域が表示されているためにストア上のディスクに残っている領域の容量 (KB 単位)。
TempTableKBCr	このトランザクションが IQ テンポラリ・テーブルのデータの格納用に作成した IQ テンポラリ・ストアの領域 (KB 単位)。
TempTableKBDr	このトランザクションによって削除された IQ テンポラリ・テーブルの領域のうち、IQ カーソルで表示されているため、またはこのトランザクションの他のセーブポイントが所有しているためにテンポラリ IQ ストアのディスク上に残っている領域の容量 (KB 単位)。

カラム名	説明
TempWorkSpaceKB	ステータスが ACTIVE であるトランザクションでは、このトランザクションが使用中のワークスペースのスナップショットです (ソート、ハッシュ、テンポラリ・ビットマップなど)。この数字は、**sp_iqtransaction** を実行するタイミングによって変わります。たとえば、クエリ・エンジンがテンポラリ・キャッシュに 60MB を作成した場合、クエリ処理が継続中でもその大部分をすぐに解放することがあります。したがって、クエリが完了した後に **sp_iqtransaction** を実行すると、このカラムの数字がずっと小さくなっていることがあります。トランザクションがアクティブでなくなると、このカラムはゼロになります。 ACTIVE なトランザクションでは、このカラムは **sp_iqconnection** の TempWorkSpaceKB カラムと同一です。
TxnCreateTime	トランザクションの開始時刻。すべての Sybase IQ トランザクションは、アクティブな接続が確立されるか、前のトランザクションがコミットまたはロールバックしたときに暗黙的に開始されます。
CursorCount	このトランザクション制御ブロックを参照している、開いた Sybase IQ カーソルの数。 トランザクションが ACTIVE である場合、トランザクション内で作成され、開かれているカーソルの数を示します。 トランザクションのステータスが COMMITTED の場合は、このトランザクション制御ブロックが所有するデータベース・バージョンを参照する HOLD カーソルの数を示します。
SpCount	トランザクション制御ブロック内に存在する、セーブポイント構造の数。セーブポイントは、暗示的に作成および解放されます。したがって、この番号はトランザクション内でセーブポイントを作成したユーザの数を示すものではありません。
SpNumber	トランザクションの、アクティブなセーブポイントの数です。これは実装の詳細を反映しているため、セーブポイントを作成したユーザの数は反映されていない場合があります。
MPXServerName	この値は、アクティブなトランザクションが、ノード間通信 (INC) 接続からのトランザクションであるかどうかを示します。INC 接続からのトランザクションの場合、この値は、そのトランザクションが開始されたマルチプレックス・サーバの名前となります。INC 接続からのトランザクションでない場合、NULL となります。トランザクションがアクティブでない場合は、常に NULL となります。
GlobalTxnID	この値は、現在のトランザクションに関連付けられているグローバル・トランザクション ID を示します。関連付けられているグローバル・トランザクションがない場合、ゼロとなります。

例

sp_iqtransaction の出力の例を次に示します。

```
Name   Userid  TxnID  CmtID VersionID    State    ConnHandle IQConnID
======= ===== ====== ====== ========= ========== ===========
========
red2      DBA  10058  10700     10058  COMMITTED   419740283       14
blue1     DBA  10568      0     10568     ACTIVE   640038605       17
          DBA  10604      0     10604     ACTIVE  2094200996       18
fromSCJ   DBA  10619      0     10619     ACTIVE   954498130       20
blue2     DBA  10634  10677     10634  COMMITTED   167015670       21
ntJava2   DBA  10676      0     10676     ACTIVE  1779741471       24
blue2     DBA  10678      0     10678     ACTIVE   167015670       21
nt1       DBA  10699      0     10699     ACTIVE   710225777       28
red2      DBA  10701      0     10701     ACTIVE   419740283       14
          DBA  16687      0     16687     ACTIVE  1306718536       23

MainTableKBCr      MainTableKBDr    TempTableKBCr TempTableKBDr
============= ================== ================================
        0                    0            65824              0
        0                    0                0              0
        0                    0                0              0
        0                    0                0              0
     3960                  152                0              0
        0                    0                0              0
     2400                 1992                0              0
        0                    0                0              0
        0                    0             2912          22096
        0                    0                0              0

TempWorkSpaceKB TxnCreateTime               CursorCount SpCount
SpNumber
==============  ==================        ==========  ======  ======
       0    2009-06-26 13:17:27.612                1       3       2
  102592    2009-06-26 13:27:28.491                1       1       0
       0    2009-06-26 13:30:27.548                0       1       0
       0    2009-06-26 13:31:27.151                0      24     262
       0    2009-06-26 13:35:02.128                0       0       0
       0    2009-06-26 13:43:58.805                0      39     408
     128    2009-06-26 13:45:28.379                0       1       0
       0    2009-06-26 14:05:15.759                0      42     413
     680    2009-06-26 14:57:51.104                1       2      20
       0    2009-06-26 15:09:30.319                0       1       0

MPXServerName  GlobalTxnID
=============  ===========
        (NULL)           0
        (NULL)           0
        (NULL)           0
        (NULL)           0
        (NULL)           0
        (NULL)           0
        (NULL)           0
        (NULL)           0
```

```
        (NULL)            0
        (NULL)            0
```

参照：

- sp_iqstatus プロシージャ (517 ページ)
- sp_iqversionuse プロシージャ (539 ページ)

sp_iqunusedcolumn プロシージャ

負荷によって参照されなかった IQ カラムをレポートします。

構文

```
sp_iqunusedcolumn
```

パーミッション

DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

説明

SYSTEM で作成されたテーブルまたはローカル・テンポラリ・テーブルのカラムはレポートされません。

表 197 : sp_iqunusedcolumn のカラム

カラム名	説明
TableName	テーブル名
ColumnName	カラム名
Owner	カラム所有者のユーザ名

例

sp_iqunusedcolumn プロシージャからの出力例を示します。

```
TableName              ColumnName             Owner
SalesOrders            ID                     GROUPO
SalesOrders            CustomerID             GROUPO
SalesOrders            OrderDate              GROUPO
SalesOrders            FinancialCode          GROUPO
SalesOrders            Region                 GROUPO
SalesOrders            SalesRepresentative    GROUPO
SalesOrderItems        ID                     GROUPO
SalesOrderItems        LineID                 GROUPO
SalesOrderItems        ProductID              GROUPO
```

```
SalesOrderItems        Quantity              GROUPO
SalesOrderItems        ShipDate              GROUPO
Contacts               ID                    GROUPO
Contacts               Surname               GROUPO
Contacts               GivenName             GROUPO ...
```

参照：

- sp_iqcolumnuse プロシージャ (405 ページ)
- sp_iqindexadvice プロシージャ (460 ページ)
- sp_iqindexuse プロシージャ (469 ページ)
- sp_iqtableuse プロシージャ (530 ページ)
- sp_iqunusedindex プロシージャ (537 ページ)
- sp_iqunusedtable プロシージャ (538 ページ)
- sp_iqworkmon プロシージャ (547 ページ)

sp_iqunusedindex プロシージャ

負荷によって参照されなかった IQ の二次的な (非 FP) インデックスをレポートします。

構文

```
sp_iqunusedindex
```

パーミッション

DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

説明

SYSTEM で作成されたテーブルまたはローカル・テンポラリ・テーブルのインデックスはレポートされません。

表 198 : sp_iqunusedindex のカラム

カラム名	説明
IndexName	インデックス名
TableName	テーブル名
Owner	インデックス所有者のユーザ名
IndexType	インデックス・タイプ

例
sp_iqunusedindex プロシージャからの出力例を示します。

```
IndexName            TableName             Owner  IndexType
ASIQ_IDX_T450_I7_HG  SalesOrders           GROUPO HG
ASIQ_IDX_T450_C6_HG  SalesOrders           GROUPO HG
ASIQ_IDX_T450_C4_HG  SalesOrders           GROUPO HG
ASIQ_IDX_T450_C2_HG  SalesOrders           GROUPO HG
ASIQ_IDX_T451_I6_HG  SalesOrderItems       GROUPO HG
ASIQ_IDX_T451_C3_HG  SalesOrderItems       GROUPO HG
ASIQ_IDX_T451_C1_HG  SalesOrderItems       GROUPO HG
ASIQ_IDX_T452_I11_HG Contacts              GROUPO HG
ASIQ_IDX_T453_I10_HG Contacts              GROUPO HG
ASIQ_IDX_T454_I4_HG  FinancialCodes        GROUPO HG
ASIQ_IDX_T455_I5_HG  FinancialData         GROUPO HG
ASIQ_IDX_T455_C3_HG  FinancialData         GROUPO HG
ASIQ_IDX_T456_I8_HG  Products              GROUPO HG
ASIQ_IDX_T457_I4_HG  Departments           GROUPO HG
ASIQ_IDX_T457_C3_HG  Departments           GROUPO HG
ASIQ_IDX_T458_I21_HG Departments           GROUPO HG
ASIQ_IDX_T458_C5_HG  Departments           GROUPO HG
```

参照：
- sp_iqcolumnuse プロシージャ (405 ページ)
- sp_iqindexadvice プロシージャ (460 ページ)
- sp_iqindexuse プロシージャ (469 ページ)
- sp_iqtableuse プロシージャ (530 ページ)
- sp_iqunusedcolumn プロシージャ (536 ページ)
- sp_iqunusedtable プロシージャ (538 ページ)
- sp_iqworkmon プロシージャ (547 ページ)

sp_iqunusedtable プロシージャ

負荷によって参照されなかった IQ テーブルをレポートします。

構文
sp_iqunusedtable

パーミッション
DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

説明
SYSTEM で作成されたテーブルとローカル・テンポラリ・テーブルはレポートされません。

表 199 : sp_iqunusedtable のカラム

カラム名	説明
TableName	テーブル名
Owner	テーブル所有者のユーザ名

例
次の表は **sp_iqunusedtable** プロシージャからの出力例です。

```
TableName                Owner FinancialCodes          GROUPO
Contacts                 GROUPO Employees               GROUPO
emp1                     DBA SalesOrders              GROUPO
FinancialData            GROUPO Departments             GROUPO
SalesOrderItems          GROUPO Products                GROUP
iq_dummy                 DBA Customers                GROUPO
sale                     DBA
```

参照：
- sp_iqcolumnuse プロシージャ (405 ページ)
- sp_iqindexadvice プロシージャ (460 ページ)
- sp_iqindexuse プロシージャ (469 ページ)
- sp_iqtableuse プロシージャ (530 ページ)
- sp_iqunusedcolumn プロシージャ (536 ページ)
- sp_iqunusedindex プロシージャ (537 ページ)
- sp_iqworkmon プロシージャ (547 ページ)

sp_iqversionuse プロシージャ

IQ メイン・ストアで使用されているバージョンを表示します。

構文
```
sp_iqversionuse
```

パーミッション
DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

説明

sp_iqversionuse システム・ストアド・プロシージャは、テーブルのバージョンが複数あるためにデータベースが記憶領域を過度に使用している場合の、トラブルシューティングに役立ちます。

領域不足状態が発生している場合や、**sp_iqstatus** がマルチプレックス・サーバでのメイン・ブロックの高い使用率を示している場合は、**sp_iqversionuse** を実行して、使用されているバージョン、およびバージョンをリリースすることでリカバリできる容量を確認します。

領域の大きさは範囲として表されます。これは、一般的に他にどのバージョンが解放されたかによって実際の大きさが異なるためです。解放される実際の領域の大きさは、MinKBRelease ～ MaxKBRelease の範囲です。最も古いバージョンでは必ず、MaxKBRelease と MinKBRelease が一致します。

WasReported は、バージョン使用情報がセカンダリ・サーバからコーディネータに送信されたかどうかを示します。 新しいバージョンでは、コーディネータ上での WasReported の初期値は 0 です。 データベース・サーバがコーディネータにバージョン使用情報をレプリケートすると、WasReported が 1 になります。

注意： WasReported カラムは、マルチプレックス設定で使用されます。マルチプレックスの詳細については、『Sybase IQ Multiplex の使用』を参照してください。

表 200 : sp_iqversionuse のカラム

カラム名	説明
VersionID	シンプレックス・データベースでは、VersionID としてゼロが表示されます。 マルチプレックス・コーディネータでは、VersionID はアクティブなトランザクションの TxnID、コミット済みのトランザクションの CmtID と同一です。 マルチプレックスのセカンダリ・サーバでは、VersionID は、マルチプレックス・コーディネータでデータベース・バージョンを作成したトランザクションの CmtID です。 これは、マルチプレックス・データベース内のすべてのノードに対してデータベース・バージョンを一意に識別するために、Sybase IQ のメモリ内カタログと IQ トランザクション・マネージャで使用されます。
Server	このバージョンのユーザが接続するサーバ
IQConnID	このバージョンを使用する接続 ID
WasReported	このバージョンの使用情報をサーバが受信したかどうかを示します。
MinKBRelease	このバージョンが使用されなくなったときに返される領域の最小サイズ

カラム名	説明
MaxKBRelease	このバージョンが使用されなくなったときに返される領域の最大サイズ

例
次の表は **sp_iqversionuse** システム・プロシージャからの出力例です。

```
VersionID Server         IQConnID WasReported
========= ======         ======== ===========
        0 ab2ab_iqdemo          9           0

MinKBRelease  MaxKBRelease
============  ============
           0             0
```

参照：
- sp_iqstatus プロシージャ (517 ページ)
- sp_iqtransaction プロシージャ (531 ページ)

sp_iqview プロシージャ

テーブル内のビューに関する情報を表示します。

構文 1

```
sp_iqview ([view_name],[view_owner],[view_type])
```

構文 2

```
sp_iqview [view_name='viewname'],
[view_owner='viewowner' ],[view_type='viewtype' ]
```

パーミッション
DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

使用法：構文 1

```
sp_iqview NULL,NULL,SYSTEM
```

最初の 2 つのパラメータのいずれかを指定せずに、その次のパラメータを指定する場合は、指定しないパラメータの位置に NULL を指定する必要があります。たとえば、`sp_iqview NULL,NULL,SYSTEM` や `sp_iqview deptview,NULL,'ALL'.` のように指定します。

注意： *view_type* の値 ALL は、構文 1 では一重引用符で囲む必要があります。

使用法：構文 2
パラメータはどのような順番で指定しても構いません。一重引用符で囲みます。

view_type パラメータに指定できる値を次に示します。

表 201 : sp_iqview の view_type の値

view_type の値	表示される情報
SYSTEM	システム・ビュー
ALL	ユーザ・ビューとシステム・ビュー
その他の値	ユーザ・ビュー

説明
いずれかのパラメータを指定すると、指定されたビュー名のビュー、または指定されたユーザが所有するビューのみが返されます。複数のパラメータを指定すると、指定されたすべてのパラメータにより結果がフィルタされます。パラメータを指定しない場合は、データベース内のすべてのユーザ・ビューが返されます。

表 202 : sp_iqview のカラム

カラム名	説明
view_name	ビューの名前
view_owner	ビューの所有者
view_def	**CREATE VIEW** 文で指定されたビューの定義
remarks	**COMMENT** 文で追加されたユーザ・コメント

sp_iqview は、32K 文字より大きいビュー定義をトランケーションなしで返します。

参照：
- sp_iqcolumn プロシージャ (403 ページ)
- sp_iqconstraint プロシージャ (410 ページ)
- sp_iqdatatype プロシージャ (419 ページ)
- sp_iqevent プロシージャ (443 ページ)
- sp_iqhelp プロシージャ (448 ページ)
- sp_iqindex および sp_iqindex_alt プロシージャ (456 ページ)
- sp_iqjoinindex プロシージャ (471 ページ)
- sp_iqpkeys プロシージャ (491 ページ)
- sp_iqprocparm プロシージャ (496 ページ)
- sp_iq_reset_identity プロシージャ (505 ページ)
- sp_iqtable プロシージャ (525 ページ)

sp_iqview プロシージャ例

sp_iqview を使用する場合は、次の例を参照してください。

次の構文は、いずれもビュー ViewSalesOrders に関する情報を返します。

```
call sp_iqview('ViewSalesOrders')
```

```
sp_iqview view_name='ViewSalesOrders'
```

次の構文は、いずれも GROUPO が所有するすべてのビューを返します。

```
sp_iqview NULL,GROUPO
```

```
sp_iqview view_owner='GROUPO'
```

view_name	view_owner	view_def	remarks
ViewSalesOrders	GROUPO	Create views GROUPO , ViewSalesOrders(ID, LineID, ProductID, Quantity, OrderDate, Ship-Date, Region, SalesRepresentativeName	(NULL)

sp_iqwho プロシージャ

現在のすべてのユーザと接続に関する情報、または特定のユーザまたは接続に関する情報を表示します。

構文

```
sp_iqwho [ { connhandle | user-name } [, arg-type ] ]
```

パーミッション

DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

説明

sp_iqwho ストアド・プロシージャは、現在のすべてのユーザと接続に関する情報、または特定のユーザまたは接続に関する情報を表示します。

表 203 : sp_iqwho のカラム

カラム名	説明
ConnHandle	SA 接続ハンドル
IQConnID	Sybase IQ 固有の接続 ID
Userid	接続 "ConnHandle" を開いたユーザの名前

カラム名	説明
BlockedOn	特定の接続をブロックする接続。どの接続もブロックしない場合は 0 です。
BlockUserid	ブロックする接続の所有者。ブロック接続が存在しない場合は NULL。
ReqType	接続を通じて行われた要求のタイプ。コマンドを発行しない場合は DO_NOTHING。
IQCmdType	接続から発行された Sybase IQ コマンドのタイプ。コマンドを発行しない場合は NONE。
IQIdle	この接続を通じて最後の Sybase IQ コマンドを発行してから経過した時間 (秒)。最後に発行した Sybase IQ コマンドが存在しない場合、2000 年 1 月 1 日からの経過時間が表示されます。
SAIdle	この接続を通じて最後の SA 要求を発行してから経過した時間 (秒)。最後に発行した SA コマンドが存在しない場合、2000 年 1 月 1 日からの経過時間が表示されます。
IQCursors	接続内のアクティブなカーソルの数。カーソルが存在しない場合は 0 です。
IQThreads	接続を持つスレッドの数。接続を開くとすぐに、少なくとも 1 つのスレッドが開始するので、IQThreads の最小値は 1 です。
TempTableSpaceKB	テンポラリ・テーブル領域のサイズ (キロバイト)。テンポラリ・テーブル領域が使用されていない場合は 0 です。
TempWorkSpaceKB	テンポラリ作業領域のサイズ (キロバイト)。テンポラリ作業領域が使用されていない場合は 0 です。

表 204 : sp_who カラムと sp_iqwho カラムのマッピング

sp_who カラム	sp_iqwho カラム
fid	ロックの属するファミリ。Sybase IQ には適用されないため省略されます。
spid	ConnHandle、IQConnID
status	IQIdle、SAIdle
loginame	Userid

sp_who カラム	sp_iqwho カラム
orignameユーザのエイリアス。Sybase IQ には適用されないため省略されます。	
hostname	サーバが実行されているホストの名前。現在はサポートされていません。
blk_spid	BlockedOn
dbname	Sybase IQ に対して 1 つのサーバと 1 つのデータベースが存在し、これらが接続ごとに同じであるため、省略されます。
cmd	ReqType、IQCmdType
block_xloid	BlockUserid

使用法

表 205 : パラメータ

パラメータ	説明
connhandle	接続 ID を表す整数。このパラメータを指定した場合、**sp_iqwho** は指定された接続に関する情報のみを返します。指定の接続が開いていない場合、出力にローが表示されません。
user-name	ユーザ・ログイン名を表す char(255) パラメータ。このパラメータを指定した場合、**sp_iqwho** は指定のユーザに関する情報のみを返します。指定のユーザが接続を開いていない場合、出力にローが表示されません。指定のユーザがデータベースに存在しない場合、**sp_iqwho** はエラー・メッセージ ″User *user-name* does not exist″ を返します。
arg-type	*arg-type* パラメータはオプションであり、最初のパラメータが指定されたときにのみ指定できます。*arg-type* の唯一の値は **“user”** です。*arg-type* 値を **“user”** として指定すると、**sp_iqwho** は、最初のパラメータが数値であってもユーザ名として解釈します。**“user”** 以外の値が *arg-type* に指定された場合、**sp_iqwho** は、エラー “Invalid parameter.” を返します。 *arg-type* 値を二重引用符で囲んでください。

パラメータを指定しない場合、**sp_iqwho** では現在アクティブなすべての接続およびユーザに関する情報が表示されます。

最初の **sp_iqwho** パラメータとして、接続ハンドルまたはユーザ名のいずれかを指定できます。パラメータ *connhandle* と *user-name* は排他的であり、オプションで

す。これらのパラメータは、一度に片方だけ指定できます。デフォルトでは、最初のパラメータが数値である場合、パラメータは接続ハンドルと見なされます。最初のパラメータが数値でない場合、ユーザ名と見なされます。

Sybase IQ では数値のユーザ名を使用できます。*arg-type* パラメータは、最初のパラメータの数値をユーザ名として解釈するように **sp_iqwho** に指示します。 次に例を示します。

```
sp_iqwho 1, “user”
```

*arg-type***“user”** を指定すると、**sp_iqwho** は最初のパラメータ 1 を、接続 ID ではなく、ユーザ名として解釈します。`1` という名前のユーザがデータベースに存在する場合、**sp_iqwho** は、ユーザ `1` が開いた接続に関する情報を表示します。

表 206 : sp_iqwho の使用例

構文	出力
sp_iqwho	すべてのアクティブな接続を表示します。
sp_iqwho 3	接続 3 に関する情報を表示します。
sp_iqwho “DBA”	ユーザ `DBA` が開いた接続を表示する。
sp_iqwho 3, “user”	3 をユーザ名として解釈し、ユーザ 3 が開いた接続を表示する。ユーザ 3 が存在しない場合、エラー `“User 3 does not exist”` を返す。
sp_iqwho non-existing-user	エラー `“User non-existing-user does not exist”` を返す。
sp_iqwho 3, “xyz”	エラー `“Invalid parameter: xyz”` を返す。

sp_iqwho プロシージャの例

sp_iqwho を使用する場合は、この例を参照してください。

すべてのアクティブな接続を表示します

```
ConnHandle IQConnID Userid ReqType      IQCmdType            Blocked
On BlockUserid IQCursors
IQThreads IQIdle SAIdle TempTableSpaceKB TempWorkSpaceKB
12         118      DBA    CURSOR_OPEN  IQUTILITYOPENCURSOR  0
    (NULL)      0   1         1      0      0             0
13         119      shweta DO_NOTHING   NONE                0
    (NULL)      0 1    16238757   470     0            0
```

sp_iqwho の Adaptive Server Enterprise との互換性

Sybase IQ**sp_iqwho** ストアド・プロシージャは、Adaptive Server Enterprise**sp_who** プロシージャによって表示されるカラムと同等の Sybase IQ カラムを取り込みます。

いくつかの Adaptive Server Enterprise カラムは、Sybase IQ に適用できないため省略されます。

sp_iqworkmon プロシージャ

負荷モニタリングの使用状況情報の収集処理を制御し、モニタリング収集ステータスをレポートします。**sp_iqworkmon** は、クエリ (**FROM** 句がある SQL 文) の情報だけを収集します。**sp_iqworkmon** は、**INSERT** 文または **LOAD** 文には使用できません。

構文

```
sp_iqworkmon [ 'action' ] [ , 'mode' ]
```

```
action = 'start' , 'stop' , 'status' , 'reset'
```

```
mode = 'index' , 'table' , 'column' , 'all'
```

次に例を示します。

```
sp_iqworkmon 'start' , 'all'
```

引数を 1 つだけ指定する場合、指定できるのは *action* だけです。次に例を示します。

```
sp_iqworkmon 'stop'
```

パーミッション

DBA 権限が必要です。DBA 権限を持たないユーザがこのストアド・プロシージャを実行するには、EXECUTE パーミッションが付与される必要があります。

使用法

表 207 : パラメータ

パラメータ	説明
action の値	適用する制御アクションを指定します。値 *start* を指定すると、指定したモードでモニタリングがすぐに開始します。値 *stop* を指定すると、モニタリングがすぐに停止します。値 *status* (デフォルト値) を指定すると、状態が変わることなく現在のステータスが表示されます。 統計は、*reset* 引数でクリアされるまで、またはサーバが再起動されるまで保持されます。サーバの再起動後、統計の収集は自動的には再開しません。*start* を使用して再開する必要があります。
mode	制御するモニタリングの種類を指定します。INDEX、TABLE、COLUMN の各キーワードは、インデックスの使用状況、テーブルの使用状況、カラムの使用状況のモニタリングをそれぞれ個別に制御します。デフォルトの ALL キーワードでは、すべての使用状況モニタリング機能のモニタリングが同時に制御されます。

sp_iqworkmon を実行すると、常に結果セットも表示されます。特定のモード (インデックスなど) を指定すると、そのモードのローだけが表示されます。

使用状況が収集されるのは、**FROM** 句を含む SQL 文だけです。たとえば、**SELECT**、**UPDATE**、**DELETE** などがあります。

表 208 : sp_iqworkmon のカラム

カラム名	説明
MonMode	table、index、または column
ステータス	started または stopped
Rowcount	現在までに収集されたローの数

例

sp_iqworkmon プロシージャからの出力例を示します。

```
MonMode     Status      Rowcount index       started     15
table       started     10 column      started     31
```

参照：

- sp_iqcolumnuse プロシージャ (405 ページ)

- sp_iqindexadvice プロシージャ (460 ページ)
- sp_iqindexuse プロシージャ (469 ページ)
- sp_iqtableuse プロシージャ (530 ページ)
- sp_iqunusedcolumn プロシージャ (536 ページ)
- sp_iqunusedindex プロシージャ (537 ページ)
- sp_iqunusedtable プロシージャ (538 ページ)

カタログ・ストアド・プロシージャ

カタログ・ストア・ストアド・プロシージャは、データベース・サーバ、データベース、接続プロパティの情報を表にまとめて、結果セットとして返します。

このプロシージャを所有するのは、ユーザ ID dbo です。このプロシージャに対する EXECUTE パーミッションは、PUBLIC グループが持っています。

sa_ansi_standard_packages システム・プロシージャ

SQL 文で使用されている、コア以外の SQL 拡張機能に関する情報を返します。

構文

```
sa_ansi_standard_packages( standard, statement )
```

引数

- ***standard*** **–** コア拡張機能に使用する規格。SQL:1999 または SQL:2003。
- ***statement*** **–** 評価する SQL 文。

備考

文に使用される、コア以外の拡張機能がない場合、結果セットは空です。

パーミッション

なし。

関連する動作

なし。

例

次に、sa_ansi_standard_packages システム・プロシージャを呼び出す例を示します。

```
CALL sa_ansi_standard_packages( 'SQL:2003',
'SELECT *
   FROM ( SELECT o.SalesRepresentative,
   o.Region,
```

```
                  SUM( s.Quantity * p.UnitPrice ) AS total_sales,
                  DENSE_RANK() OVER ( PARTITION BY o.Region,
                                      GROUPING( o.SalesRepresentative )
                                         ORDER BY total_sales DESC ) AS
sales_rank
            FROM Product p, SalesOrderItems s, SalesOrders o
            WHERE p.ID = s.ProductID AND s.ID = o.ID
            GROUP BY GROUPING SETS( ( o.SalesRepresentative,
o.Region ), o.Region ) ) AS DT
   WHERE sales_rank <= 3
   ORDER BY Region, sales_rank');
```

このクエリは、次の結果セットを生成します。

package_id	package_name
T612	Advanced OLAP operations
T611	Elementary OLAP operations
F591	Derived tables
T431	Extended grouping capabilities

sa_audit_string システム・プロシージャ

文字列をトランザクション・ログに追加します。

構文

```
sa_audit_string( string )
```

引数

- ***string*** – トランザクション・ログに追加する文字列。

備考

このシステム・プロシージャでは、監査が有効な場合、トランザクション・ログに格納されている監査情報にコメントが追加されます。この文字列は最大 200 バイトまでです。

パーミッション

DBA 権限

関連する動作

なし。

例

次の例では、sa_audit_string を使用してコメントをトランザクション・ログに追加します。

```
CALL sa_audit_string( 'Auditing test' );
```

sa_checkpoint_execute システム・プロシージャ

チェックポイント中にシェル・コマンドの実行を可能にします。

構文

sa_checkpoint_execute '*shell_commands*'

パラメータ

パラメータ	説明
shell_commands	システム・シェル内で 1 つまたは複数のユーザ・コマンドを実行します。シェル・コメントはシステム・シェルに固有のものです。コマンドはセミコロン (;) で区切られます。

パーミッション

なし。

説明

サーバがクワイス (静止) 状態のとき、シェル・コマンドを実行して、実行中のデータベースをチェックポイント操作中にコピーできます。コピーされたデータベースは、システム障害の後と同じように、開始して通常のリカバリを行うことができます。

sa_checkpoint_execute は、チェックポイントを開始し、チェックポイントの中からシステム・シェルを実行し、ユーザ・コマンドをシェルに渡します。サーバは、シェルが完了するのを待って、データベース・ファイルをコピーするための任意のサイズの時間枠を作成します。チェックポイントの実行中、ほとんどのデータベース・アクティビティは停止します。したがって、シェル・コマンドの実行時間は、ユーザへの応答時間として許容できる範囲の時間に制限されます。

シェル・コマンドが 0 以外のステータスを返すと、**sa_checkpoint_execute** はエラーを返します。

sa_checkpoint_execute を対話型コマンドと共に使用しないでください。使用した場合、サーバは対話型コマンドが強制終了されるまで待機することになります。上書きフラグを設定し、対話型になる可能性のあるシェル・コマンドのプロンプトが表示されないようにしてください。具体的には、**COPY**、**MOVE**、**DELETE** などのコマンドでプロンプトが表示される可能性があります。

sa_checkpoint_execute の本来の用途は、ディスク・ミラーリングと共に使用して、ミラーリングされたデバイスを分割することです。

sa_checkpoint_execute を使用して iqdemo.* ファイルを別のディレクトリにコピーすると、.db ファイルと .log ファイルを除く、すべてのファイルがコピーされます。エラー -910 が返されます。

このエラーは、製品異常ではなく Windows の制限事項です。つまり、Windows のコピー・コマンドでは、データベースによって開かれているカタログ・ファイルをコピーできません。

例

backup というサブディレクトリが作成されていると仮定した場合、次の文は、チェックポイントを発行し、**iqdemo** データベースの全ファイルを backup サブディレクトリにコピーして、チェックポイントを完了します。

```
sa_checkpoint_execute 'cp iqdemo.* backup/'
```

sa_conn_activity システム・プロシージャ

サーバ上の指定したデータベース接続に関して、最後に作成された SQL 文を返します。

構文

```
sa_conn_activity( [ connidparm ] )
```

引数

- ***connidparm*** – 接続 ID 番号を指定する任意の INTEGER パラメータ。

結果セット

カラム名	データ型	説明
Number	INT	接続の ID 番号を返します。
Name	VARCHAR(255)	現在の接続の名前を返します。 テンポラリ接続名の場合、接続名の先頭に INT: が追加されます。
Userid	VARCHAR(255)	接続のユーザ ID を返します。
DBNumber	INT	データベースの ID 番号を返します。
LastReqTime	VARCHAR(255)	指定された接続において最後の要求が開始された時刻を返します。このプロパティは、イベントなどの内部接続の場合は空の文字列を返すことがあります。
LastStatement	LONG VARCHAR	現在の接続で最後に準備された SQL 文を返します。

備考
サーバが情報の収集を指示された場合、sa_conn_activity システム・プロシージャは、各接続について、最後に作成された SQL 文で構成される結果セットを返します。sa_conn_activity を呼び出す前に、データベース・サーバで文の記録をオンにします。文の記録をオンにするには、データベース・サーバの起動時に -zl オプションを指定するか、次の文を実行します。

```
CALL sa_server_option('RememberLastStatement','ON');
```

このプロシージャは、データベース・サーバがビジー状態のときに、各接続について作成された最後の SQL 文の情報を取得するのに便利です。この機能は、要求ロギングの代わりとして使用できます。

connidparm を指定しない場合、データベース・サーバ上で実行されているすべてのデータベースに対するすべての接続について、情報が返されます。*connidparm* が 0 未満の場合、現在の接続のオプション値が返されます。

テナント・データベース分離規則のため、このシステム・プロシージャをクラウドで実行した場合は、現在のテナント・データベースに関する情報のみが返されます。

パーミッション
なし。

関連する動作
なし。

sa_conn_info システム・プロシージャ

接続プロパティ情報をレポートします。

構文

```
sa_conn_info( [ connidparm ] )
```

引数

- ***connidparm*** – 接続 ID 番号を指定する任意の INTEGER パラメータ。

結果セット

カラム名	データ型	説明
Number	INTEGER	接続の ID 番号を返します。

カラム名	データ型	説明
Name	VARCHAR(255)	接続の ID 番号を返します。 テンポラリ接続名の場合、接続名の先頭に `INT:` が追加されます。
Userid	VARCHAR(255)	接続のユーザ ID を返します。
DBNumber	INTEGER	データベースの ID 番号を返します。
LastReqTime	VARCHAR(255)	指定された接続において最後の要求が開始された時刻を返します。このプロパティは、イベントなどの内部接続の場合は空の文字列を返すことがあります。
ReqType	VARCHAR(255)	最後の要求のタイプを返します。接続が接続プーリングによりキャッシュされている場合の ReqType 値は CONNECT_POOL_CACHE です。
CommLink	VARCHAR(255)	接続の通信リンクを返します。これは SQL Anywhere がサポートするネットワーク・プロトコルであり、同一コンピュータ接続の場合は local となります。
NodeAddr	VARCHAR(255)	クライアント／サーバ接続のクライアント・アドレスを返します。
ClientPort	INTEGER	クライアントの TCP/IP ポート番号を返します。接続の種類が TCP/IP でない場合は、0 を返します。
ServerPort	INTEGER	データベース・サーバの TCP/IP ポート番号または 0 を返します。
BlockedOn	INTEGER	現在の接続がブロックされていない場合は 0 を返し、ブロックされている場合はロック競合によってブロックされた接続の接続番号を返します。
LockRowID	UNSIGNED BIGINT	ロックされたローの識別子を返します。 ローに関連付けられているロックで接続が待機していない場合 (つまり、ロックで待機していない場合、または関連付けられているローがないロックで待機している場合)、LockRowID は NULL です。

カラム名	データ型	説明
LockIndexID	INTEGER	ロックされたインデックスの識別子を返します。 ロックが LockTable のテーブルにあるすべてのインデックスに関連付けられている場合、LockIndexID は -1 を返します。インデックスに関連付けられているロックで接続が待機していない場合 (つまり、ロックで待機していない場合、または関連付けられているインデックスがないロックで待機している場合)、LockIndexID は NULL です。
LockTable	VARCHAR(255)	現在、接続がロックを待機している場合は、ロックに関連付けられているテーブルの名前を返します。それ以外の場合、LockTable は空の文字列を返します。
UncommitOps	INTEGER	コミットされていない操作の数を返します。
ParentConnection	INTEGER	データベース・オペレーション (データベースのバックアップや作成など) を実行するためにテンポラリ接続を作成した接続の接続 ID を返します。その他のタイプの接続について、このプロパティは NULL を返します。

備考
接続 ID 番号を指定した場合、sa_conn_info system プロシージャは、指定された接続について、接続プロパティで構成される結果セットを返します。*connidparm* を指定しない場合は、このシステム・プロシージャによって、サーバ上のデータベースに対する現在の接続情報がすべて返されます。*connidparm* が 0 未満の場合、現在の接続のオプション値が返されます。

ブロックの場合には、このプロシージャが返す BlockedOn 値によって、どのユーザがどのユーザによってブロックされているかを調べることができます。sa_locks システム・プロシージャを使用して、ブロックされた接続で保持されているロックを表示できます。

これらプロパティに関する詳細については、次のような例を実行できます。

```
SELECT *, DB_NAME( DBNumber ),
   CONNECTION_PROPERTY( 'LastStatement', Number )
   FROM sa_conn_info( );
```

LockRowID の値は、sa_locks プロシージャの出力でロックを検索するときに使用できます。

LockIndexID の値は、sa_locks プロシージャの出力でロックを検索するときに使用できます。また、LockIndexID の値は、SYSIDX システム・ビューを使用して表示できる ISYSIDX システム・テーブルのプライマリ・キーに対応します。

ロックにはそれぞれ関連付けられたテーブルがあるため、LockTable の値を使用して、ロックで接続が待機しているかどうかを明確に判断できます。

テナント・データベース分離規則のため、このシステム・プロシージャをクラウドで実行した場合は、現在のテナント・データベースに関する情報のみが返されます。

パーミッション
なし。

関連する動作
なし。

例
次の例では、sa_conn_info システム・プロシージャを使用して、サーバに対する全接続の接続プロパティをまとめた結果セットを返します。

```
CALL sa_conn_info( );
```

Number	Name	Userid	DBNumber	...
79		DBA	0	...
46	Sybase Central 1	DBA	0	...
...	...	...	...	...

次の例では、sa_conn_info システム・プロシージャを使用して、テンポラリ接続を作成した接続を示す結果セットが返されます。

```
SELECT Number, Name, ParentConnection FROM sa_conn_info();
```

接続 8 がテンポラリ接続を作成し、そのテンポラリ接続によって CREATE DATABASE 文が実行されました。

```
Number      Name                ParentConnection
------------------------------------------------
1000000048  INT: CreateDB       8
9           SQL_DBC_14675af8    (NULL)
8           SQL_DBA_152d5ac0    (NULL)
```

sa_conn_list システム・プロシージャ

接続 ID を含む結果セットを返します。

構文

```
sa_conn_list ([ connidparm ] [ ,dbidparm])
```

引数

- *connidparm* 接続の ID 番号を指定するため、任意で使用する INTEGER パラメータ。
- *dbidparm* データベースの ID 番号を指定するため、任意で使用する INTEGER パラメータ。

結果セット

カラム名	データ型	説明
Number	INTEGER	接続の ID 番号。

参照

『SQL Anywhere サーバー - SQL リファレンス』の「システムプロシージャー」>「システムプロシージャーのアルファベット順リスト」>「sa_conn_list システムプロシージャー」

注意： 参照先は SQL Anywhere のマニュアルです。

sa_conn_properties システム・プロシージャ

接続プロパティ情報をレポートします。

構文

```
sa_conn_properties ( [ connidparm ] )
```

詳細については、『SQL Anywhere サーバー - SQL リファレンス』の「システムプロシージャー」>「システムプロシージャーのアルファベット順リスト」>「sa_conn_properties システムプロシージャー」を参照してください。

注意： 参照先は SQL Anywhere のマニュアルです。

sa_db_info システム・プロシージャ

データベースのプロパティ情報をレポートします。

構文

```
sa_db_info( [ dbidparm ] )
```

引数

- ***dbidparm*** **–** データベース ID 番号を指定する任意の INTEGER パラメータ。

結果セット

カラム名	データ型	説明
Number	INTEGER	接続の ID 番号を返します。
Alias	VARCHAR(255)	データベース名を返します。
File	VARCHAR(255)	パスを含むデータベース・ルート・ファイル名を返します。
ConnCount	INTEGER	データベースとの接続の数を返します。プロパティ値には、内部処理に使用されている接続は含まれませんが、イベントと外部環境サポートに使用されている接続は含まれます。
PageSize	INTEGER	データベースのページ・サイズ (バイト単位) を返します。
LogName	VARCHAR(255)	パスを含むトランザクション・ログ・ファイル名を返します。

備考
データベース ID を指定した場合、sa_db_info は、指定したデータベースの Number、Alias、File、ConnCount、PageSize、LogName を持つ 1 つのローを返します。

dbidparm を指定しない場合は、すべてのデータベースのプロパティが返されます。

テナント・データベース分離規則のため、このシステム・プロシージャをクラウドで実行した場合は、現在のテナント・データベースに関する情報のみが返されます。

パーミッション
なし。

関連する動作
なし。

例
次の文は、サーバで実行される各データベースのローを返します。

```
CALL sa_db_info( );
```

プロパティ	値
Number	0
Alias	demo
File	
ConnCount	1
PageSize	4096
LogName	C:¥Documents and Settings¥All Users¥Program Data¥SybaseIQ¥demo¥iqdemo.log

sa_db_properties システム・プロシージャ

データベースのプロパティ情報をレポートします。

構文

```
sa_db_properties( [ dbidparm ] )
```

引数

- ***dbidparm*** **–** データベース ID 番号を指定する任意の INTEGER パラメータ。

結果セット

カラム名	データ型	説明
Number	INTEGER	データベース ID 番号。
PropNum	INTEGER	データベース・プロパティの番号。
PropName	VARCHAR(255)	データベース・プロパティ名。
PropDescription	VARCHAR(255)	データベース・プロパティの説明。
Value	LONG VARCHAR	データベース・プロパティの値。

備考

データベース ID を指定した場合、sa_db_properties システム・プロシージャは使用可能な各データベース・プロパティについてデータベース ID 番号と、PropNum、PropName、PropDescription、Value を返します。すべてのデータベース・プロパティと、データベースに関する統計情報の値が返されます。NULL 値を持つ有効なプロパティも返されます。

dbidparm を指定しない場合は、すべてのデータベースのプロパティが返されます。

テナント・データベース分離規則のため、このシステム・プロシージャをクラウドで実行した場合は、現在のテナント・データベースに関する情報のみが返されます。

パーミッション
なし。

関連する動作
なし。

例
次の例では、sa_db_properties() システム・プロシージャを使用して、すべてのデータベースのデータベース・プロパティ情報をまとめた結果セットを返します。

```
CALL sa_db_properties( );
```

Number	PropNum	PropName	...
0	0	ConnCount	...
0	1	IdleCheck	...
0	2	IdleWrite	...
...	...	...	...

sa_disable_auditing_type システム・プロシージャ

特定のイベントの監査を無効にします。

構文

```
sa_disable_auditing_type(' types ')
```

引数

- ***types*** – カンマで区切られた文字列を指定する VARCHAR(128) パラメータ。この文字列には、次の 1 つ以上の値が含まれています。
  - **all** – すべてのタイプの監査を無効にします。
  - **connect** – 成功した接続と失敗した接続の両方の監査を無効にします。
  - **connectFailed** – 失敗した接続の監査を無効にします。
  - **DDL** – DDL 文の監査を無効にします。
  - **options** – パブリック・オプションの監査を無効にします。
  - **permission** – パーミッション・チェック、ユーザ・チェック、SETUSER 文の監査を無効にします。

- **permissionDenied –** 失敗したパーミッション・チェックと失敗したユーザ・チェックの監査を無効にします。
- **triggers –** トリガ・イベントに応じて監査を無効にします。

備考
sa_disable_auditing_type システム・プロシージャを使用すると、1 つ以上の情報カテゴリの監査を無効にできます。

このオプションを all に設定すると、すべての監査を無効にできます。監査を無効にするには、PUBLIC.auditing オプションを Off に設定する方法もあります。

パーミッション
DBA 権限

関連する動作
なし。

例
すべての監査を無効にするには、次の文を実行します。

```
CALL sa_disable_auditing_type( 'all' );
```

sa_disk_free_space システム・プロシージャ

DB 領域、トランザクション・ログ、トランザクション・ログ・ミラー、テンポラリ・ファイルに使用可能な領域に関する情報をレポートします。

構文

```
sa_disk_free_space( [ p_dbspace_name ] )
```

引数

- ***p_dbspace_name* –** DB 領域、トランザクション・ログ・ファイル、トランザクション・ログ・ミラー・ファイル、またはテンポラリ・ファイルの名前を指定する VARCHAR(128) パラメータ。

  log、mirror、または temp という DB 領域がある場合は、キーワードの前にアンダースコアを付けます。たとえば、log という DB 領域が存在する場合は、_log を使用してログ・ファイルに関する情報を取得します。

  SYSTEM を指定すると、メイン・データベース・ファイルに関する情報を取得できます。TEMPORARY または TEMP を指定するとテンポラリ・ファイル、TRANSLOG を指定するとトランザクション・ログ、TRANSLOGMIRROR を指定するとトランザクション・ログ・ミラーに関する情報を、それぞれ取得できます。

結果セット

カラム名	データ型	説明
dbspace_name	VARCHAR(128)	これは、DB 領域名、トランザクション・ログ・ファイル、トランザクション・ログ・ミラー・ファイル、またはテンポラリ・ファイルです。
free_space	UNSIGNED BIGINT	ボリューム上の空きバイト数。
total_space	UNSIGNED BIGINT	DB 領域があるドライブで使用可能な合計ディスク領域。

備考
p_dbspace_name パラメータが指定されていないか NULL の場合、結果セットには DB 領域ごとに 1 つのローと、存在する場合はトランザクション・ログ、トランザクション・ログ・ミラー、テンポラリ・ファイルごとに 1 つのローが含まれます。*p_dbspace_name* が指定されている場合は、1 つまたは 0 個のローが返ります (このような DB 領域が存在しない場合や、log または mirror が指定されていて、ログ・ファイルまたはミラー・ファイルがない場合は 0 です)。

パーミッション
DBA 権限

関連する動作
なし。

例
次の例では、sa_disk_free_space システム・プロシージャを使用して、空き領域についての情報を格納した結果セットを返します。

```
CALL sa_disk_free_space( );
```

dbspace_name	free_space	total_space
system	10952101888	21410402304
translog	10952101888	21410402304
temporary	10952101888	21410402304

sa_enable_auditing_type システム・プロシージャ

監査を有効にし、監査対象のイベントを指定します。

構文

```
sa_enable_auditing_type( 'types' )
```

引数

- ***types*** – カンマで区切られた文字列を指定する VARCHAR(128) パラメータ。この文字列には、次の 1 つ以上の値が含まれています。
  - **all** – すべてのタイプの監査を有効にします。
  - **connect** – 成功した接続と失敗した接続の両方の監査を有効にします。
  - **connectFailed** – 失敗した接続の監査を有効にします。
  - **DDL** – DDL 文の監査を有効にします。
  - **options** – パブリック・オプションの監査を有効にします。
  - **permission** – パーミッション・チェック、ユーザ・チェック、および SETUSER 文の監査を有効にします。
  - **permissionDenied** – 失敗したパーミッション・チェックと失敗したユーザ・チェックの監査を有効にします。
  - **triggers** – トリガ・イベント後の監査を有効にします。

備考

sa_enable_auditing_type は、PUBLIC.auditing オプションと一緒に使用して、特定のタイプの情報の監査を有効にします。

PUBLIC.auditing オプションを On に設定して、監査対象の情報タイプを指定しない場合は、デフォルト設定 (all) が有効になります。この場合、すべてのタイプの監査情報が記録されます。

PUBLIC.auditing オプションを On に設定して、sa_disable_auditing_type を使用してすべてのタイプの監査を無効にすると、監査情報は記録されません。監査を再確立するには、sa_enable_auditing_type を使用して監査対象にする情報のタイプを指定します。

PUBLIC.auditing オプションを Off に設定すると、sa_enable_auditing_type の設定にかかわらず監査情報は記録されません。

パーミッション

DBA 権限

関連する動作

なし。

例
オプションの監査のみを有効にするには、次のようにします。
```
CALL sa_enable_auditing_type( 'options' );
```

sa_eng_properties システム・プロシージャ

データベース・サーバのプロパティ情報をレポートします。

構文
```
sa_eng_properties( )
```

結果セット

カラム名	データ型	説明
PropNum	INTEGER	データベース・サーバ・プロパティの番号。
PropName	VARCHAR(255)	データベース・サーバ・プロパティ名。
PropDescription	VARCHAR(255)	データベース・サーバ・プロパティの説明。
Value	LONG VARCHAR	データベース・サーバ・プロパティの値。

備考
使用可能な各サーバ・プロパティの PropNum、PropName、PropDescription、Value を返します。すべてのデータベース・サーバ・プロパティと、データベース・サーバに関する統計情報の値が返されます。

パーミッション
なし。

関連する動作
なし。

例
次の文は、使用可能な一連のサーバ・プロパティを返します。
```
CALL sa_eng_properties( );
```

PropNum	PropName	...
1	IdleWrite	...
2	IdleChkPt	...
...	...	...

sa_flush_cache システム・プロシージャ

データベース・サーバ・キャッシュ内の現在のデータベースに対するすべてのページを空にします。

構文

```
sa_flush_cache( )
```

備考

データベース管理者は、このプロシージャを使用して現在のデータベースのデータベース・サーバ・キャッシュの内容を空にします。パフォーマンスの計測に使用し、同じ結果が繰り返し得られるようにします。

パーミッション

DBA 権限

関連する動作

なし。

sa_make_object システム・プロシージャ

ALTER 文を実行する前に、オブジェクトのスケルトン・インスタンスが存在することを確認します。

構文

```
sa_make_object(
  objtype,
  objname
  [, owner
  [, tabname ]  ]
)
```

```
objtype:
'procedure'
| 'function'
| 'view'
| 'trigger'
| 'service'
| 'event'
```

引数

- ***objtype*** – 作成されるオブジェクトのタイプを指定する CHAR(30) パラメータ。objtype が `'trigger'` の場合、この引数はトリガが作成されるテーブルの所有者を示します。

- ***objname*** – 作成されるオブジェクトの名前を指定する CHAR(128) パラメータ。
- ***owner*** – 作成されるオブジェクトの所有者を指定する任意の CHAR(128) パラメータ。デフォルト値は CURRENT USER です。
- ***tabname*** – この CHAR(128) パラメータは、objtype が 'trigger' である場合にのみ必要です。トリガを作成するテーブルの名前を指定するときに使用します。

備考
このプロシージャは、データベース・スキーマの作成または変更のために繰り返し実行されるスクリプトで使用すると便利です。このようなスクリプトでは、最初に CREATE 文を実行しますが、その後は ALTER 文を実行することが共通の問題です。このプロシージャを使用すると、オブジェクトが存在するかどうかを確認するためにシステム・ビューにクエリを実行する必要がなくなります。

使用するには、このプロシージャの後にオブジェクト定義全体を含む ALTER 文を実行します。

パーミッション
データベース・オブジェクトの作成または変更には、RESOURCE 権限が必要です。

関連する動作
オートコミット。

例
次の文は、スケルトン・プロシージャ定義が作成されていることを確認し、プロシージャを定義し、プロシージャに対するパーミッションを付与します。これらの命令が記述されたスクリプト・ファイルは、データベースに対して繰り返し実行でき、エラーは起こりません。

```
CALL sa_make_object( 'procedure','myproc' );
ALTER PROCEDURE myproc( in p1 INT, in p2 CHAR(30) )
BEGIN
    // ...
END;
GRANT EXECUTE ON myproc TO public;
```

次の例は、sa_make_object システム・プロシージャを使用して、スケルトン Web サービスを追加します。

```
CALL sa_make_object( 'service','my_web_service' );
```

sa_rowgenerator システム・プロシージャ

指定された開始値と終了値の間のローを格納した結果セットを返します。

構文

```
sa_rowgenerator(
 [ rstart
 [, rend
 [, rstep ] ] ]
)
```

引数

- ***rstart*** **–** 開始値を指定する任意の INTEGER パラメータ。デフォルト値は 0 です。
- ***rend*** **–** *rstart* 以上の終了値を指定する任意の INTEGER パラメータ。デフォルト値は 100 です。
- ***rstep*** **–** シーケンス値の増分を指定する任意の INTEGER パラメータ。デフォルト値は 1 です。

結果セット

カラム名	データ型	説明
row_num	INTEGER	シーケンス番号。

備考

sa_rowgenerator プロシージャをクエリの FROM 句で使用して、番号のシーケンスを生成できます。RowGenerator システム・テーブルを使用する代わりに、このプロシージャを使用できます。次のようなタスクには、sa_rowgenerator を使用できます。

- 結果セット内の既知の数のローについてテスト・データを生成する。
- あらゆる範囲の値に対するローを格納した結果セットを生成する。たとえば、1 か月の毎日についてのローを生成したり、郵便番号の範囲を生成したりできます。
- 結果セット内に指定した数のローを格納するクエリを生成する。これは、クエリのパフォーマンスのテストに役立ちます。

正しい開始値、終了値、正の 0 以外の増分値を指定しないと、ローは返されません。

次の文を使用して、RowGenerator テーブルの動作をエミュレートできます。

```
SELECT row_num FROM sa_rowgenerator( 1, 255 );
```

パーミッション
なし。

関連する動作
なし。

例
次のクエリは、現在の日付ごとに 1 つのローを含む結果セットを返します。

```
SELECT DATEADD( day, row_num-1,
        YMD( DATEPART( year, CURRENT DATE ),
            DATEPART( month, CURRENT DATE ), 1 ) )
    AS day_of_month
    FROM sa_rowgenerator( 1, 31, 1 )
    WHERE DATEPART( month, day_of_month ) =
       DATEPART( month, CURRENT DATE )
    ORDER BY row_num;
```

次のクエリは、郵便番号が (0-9999)、(10000-19999)、...、(90000-99999) の範囲の地域に居住している従業員数を示します。この範囲の一部には従業員がいないため、警告が発生します。

sa_rowgenerator プロシージャを使用すると、ある範囲の郵便番号に従業員がいない場合でも、これらの範囲を生成できます。

```
SELECT row_num AS r1, row_num+9999
 AS r2, COUNT( PostalCode ) AS zips_in_range
FROM sa_rowgenerator( 0, 99999, 10000 ) D LEFT JOIN Employees
   ON PostalCode BETWEEN r1 AND r2
GROUP BY r1, r2
ORDER BY 1;
```

次の例は、データのローを 10 個生成し、それらを NewEmployees テーブルに挿入します。

```
INSERT INTO NewEmployees ( ID, Salary, Name )
SELECT row_num,
   CAST( RAND() * 1000 AS INTEGER ),
   'Mary'
FROM sa_rowgenerator( 1, 10 );
```

次の例は、sa_rowgenerator システム・プロシージャを使用して、すべての整数を含むビューを作成します。この例の値 2147483647 は、サポートされている最大の符号付き整数を表します。

```
CREATE VIEW Integers AS
SELECT row_num AS n
FROM sa_rowgenerator( 0, 2147483647, 1 );
```

次の例は、sa_rowgenerator システム・プロシージャを使用して、0001-01-01 ～ 9999-12-31 の日付を含むビューを作成します。この例の値 3652058 は、0001-01-01

～ 9999-12-31 (サポートされている最初の日付と最後の日付の間) の日数を表します。

```
CREATE VIEW Dates AS
SELECT DATEADD( day, row_num, '0001-01-01' ) AS d
FROM sa_rowgenerator( 0, 3652058, 1 );
```

次のクエリは、1900 年から 2058 年までで、54 週あるすべての年を返します。

```
SELECT DATEADD ( day, row_num, '1900-01-01' ) AS d, DATEPART ( week,
d ) w
FROM sa_rowgenerator ( 0, 63919, 1 )
WHERE w = 54;
```

sa_server_option システム・プロシージャ

サーバの実行中に、サーバ・オプションを上書きします。

構文

```
sa_server_option(
opt,
val
)
```

引数

- ***opt*** **–** サーバ・オプション名を指定する CHAR(128) パラメータ。
- ***val*** **–** サーバ・オプションの新しい値を指定する CHAR(128) パラメータ。

備考

データベース管理者はこのプロシージャを使用して、データベース・サーバを再起動せずにデータベース・サーバ・オプションの一部を一時的に上書きできます。

このプロシージャを使用して変更されるオプション値は、サーバが停止するとデフォルト値にリセットされます。サーバが起動するたびにオプション値を変更する場合は、データベース・サーバの起動時に対応するデータベース・サーバ・オプションがあればそれを指定できます (これらのオプションは次の表で一番右のカラムにリストされています)。

次のオプション設定を変更できます。

オプション名	値の範囲	デフォルト
AutoMultiProgrammingLevel	YES、NO	YES
AutoMultiProgrammingLevelStatistics	YES、NO	NO
CacheSizingStatistics	YES、NO	NO

オプション名	値の範囲	デフォルト
CollectStatistics	YES、NO	YES
ConnsDisabled	YES、NO	NO
ConnsDisabledForDB	YES、NO	NO
ConsoleLogFile	*filename*	
ConsoleLogMaxSize	*file-size* (バイト)	
CurrentMultiProgrammin-gLevel	整数	20
DatabaseCleaner	ON、OFF	ON
DeadlockLogging	ON、OFF、RESET、CLEAR	OFF
DebuggingInformation	YES、NO	NO
DropBadStatistics	YES、NO	YES
DropUnusedStatistics	YES、NO	YES
IdleTimeout	整数 (分)	240
IPAddressMonitorPeriod	整数 (秒)	ポータブル・デバイスの場合は 120、それ以外の場合は 0
LivenessTimeout	整数 (秒)	120
MaxMultiProgrammingLevel	整数	CurrentMultiProgrammingLevel 値の 4 倍
MessageCategoryLimit	整数	400
MinMultiProgrammingLevel	整数	-gtc オプションの値の最小値およびコンピュータ上の論理 CPU の数
OptionWatchAction	MESSAGE、ERROR	MESSAGE
OptionWatchList	カンマで区切られたデータベース・オプションのリスト	
ProcedureProfiling	YES、NO、RESET、CLEAR	NO

オプション名	値の範囲	デフォルト
ProfileFilterConn	*connection-id*	
ProfileFilterUser	*user-id*	
QuittingTime	有効な日付と時刻	
RememberLastPlan	YES、NO	NO
RememberLastStatement	YES、NO	NO
RequestFilterConn	*connection-id*, -1	
RequestFilterDB	*database-id*, -1	
RequestLogFile	`filename`	
RequestLogging	SQL、HOSTVARS、PLAN、PROCEDURES、TRIGGERS、OTHER、BLOCKS、REPLACE、ALL、YES、NONE、NO	NONE
RequestLogMaxSize	*file-size* (バイト)	
RequestLogNumFiles	整数	
RequestTiming	YES、NO	NO
SecureFeatures	*feature-list*	
StatisticsCleaner	ON、OFF	ON
WebClientLogFile	`filename`	
WebClientLogging	ON、OFF	OFF

- **AutoMultiProgrammingLevel –** YES に設定すると、データベース・サーバでマルチプログラミング・レベルが自動的に調整されます。マルチプログラミング・レベルとは、一度にアクティブにできるタスクの最大数を制御するものです。このオプションを NO に設定して、マルチプログラミング・レベルを手動で制御するようにした場合、マルチプログラミング・レベルの初期値、最小値、最大値を設定できます。

- **AutoMultiProgrammingLevelStatistics –** YES に設定すると、マルチプログラミング・レベルの自動調整の統計がデータベース・サーバのメッセージ・ログに表示されます。
- **CacheSizingStatistics –** YES に設定した場合、キャッシュ・サイズが変更されるたびに、データベース・サーバ・メッセージ・ウィンドウにキャッシュ情報を表示します。
- **CollectStatistics –** YES に設定すると、データベース・サーバはパフォーマンス・モニタの統計情報を収集します。
- **ConnsDisabled –** YES に設定すると、データベース・サーバ上のデータベースに対する他の接続は許可されません。
- **ConnsDisabledForDB –** YES に設定すると、その他の接続が現在のデータベースに許可されます。
- **ConsoleLogFile –** データベース・サーバ・メッセージ・ログ情報の記録に使用されるファイル名。空の文字列を指定すると、ファイルへのロギングが停止します。これは SQL 文字列なので、パスの円記号は 2 つ重ねます。
- **ConsoleLogMaxSize –** データベース・サーバ・メッセージ・ログ情報の記録に使用されるファイルの最大サイズ (バイト単位)。データベース・サーバ・メッセージ・ログ・ファイルが、このプロパティまたは -on サーバ・オプションで指定されたサイズに達すると、そのファイルは拡張子 `.old` の名前に変更されます (既存のファイルが存在する場合は、同じ名前で置換されます)。その後、データベース・サーバ・メッセージ・ログ・ファイルが再開されます。
- **CurrentMultiProgrammingLevel –** データベース・サーバのマルチプログラミング・レベルを設定します。
- **DatabaseCleaner –** このオプションの設定は、iAnywhere テクニカル・サポートの指示があった場合を除いて、変更しないでください。
- **DeadlockLogging –** デッドロックのロギングを制御します。値 deadlock_logging もサポートされます。デッドロック・ロギング・オプションは、Sybase Central の [データベースのプロパティ] ウィンドウでも設定できます。次の値がサポートされます。

  - **ON –** デッドロック・ロギングを有効にします。
  - **OFF –** デッドロック・ロギングを無効にしますが、デッドロック・データの表示は可能です。
  - **RESET –** デッドロック・データが存在する場合は、記録済みのデッドロック・データをクリアし、その後、デッドロック・ロギングを有効にします。
  - **CLEAR –** デッドロック・データが存在する場合は、記録済みのデッドロック・データをクリアし、その後、デッドロック・ロギングを無効にします。

  デッドロック・ロギングが有効になると、sa_report_deadlocks システム・プロシージャを使用してデータベースからデッドロック情報を取得することができます。

- **DebuggingInformation –** 診断メッセージなどのメッセージをトラブルシューティングのために表示します。メッセージは、データベース・サーバ・メッセージ・ウィンドウに表示されます。
- **DropBadStatistics –** 自動統計管理で、不適切な推定値を返す統計をデータベースから削除できるようにします。
- **DropUnusedStatistics –** 自動統計管理で、連続 90 日間使用されなかった統計をデータベースから削除できるようにします。
- **IdleTimeout –** minutes で指定された時間の間、要求を送信しなかった TCP/IP 接続を切断します。こうすることで、アクティブではない接続がロックを無制限に維持することが回避されます。
- **IPAddressMonitorPeriod –** 新しい IP アドレスをチェックする時間を秒単位で設定します。最小値は 10、デフォルト値は 0 です。ポータブル・デバイスの場合、デフォルト値は 120 秒です。
- **LivenessTimeout –** 接続が維持されていることを確認するため、クライアント／サーバの TCP/IP ネットワークを介して、定期的に活性パケットが送信されます。ネットワーク・サーバが、活性パケットを検出することなく、LivenessTimeout 時間にわたって実行されると、通信は切断されます。
- **MaxMultiProgrammingLevel –** データベース・サーバのマルチプログラミング・レベルの最大値を設定します。
- **MessageCategoryLimit –** sa_server_messages システム・プロシージャを使用して取り出すことのできるメッセージの最小数を、重大度レベルごとおよびカテゴリごとに設定します。
- **MinMultiProgrammingLevel –** データベース・サーバのマルチプログラミング・レベルの最小値を設定します。
- **OptionWatchAction –** リストにオプションを設定しようとしたときにデータベース・サーバが実行するアクションを指定します。サポートされる値は MESSAGE と ERROR です。OptionWatchAction が MESSAGE に設定されており、OptionWatchList によって指定されるオプションが設定されている場合は、データベース・サーバ・メッセージ・ウィンドウにメッセージが表示され、設定されているオプションがオプション・ウォッチ・リストに入っていることが示されます。

  OptionWatchAction が ERROR に設定されている場合は、オプション・ウォッチ・リストにオプションが入っているためにオプションを設定できないことを示すエラーが返されます。

  このプロパティの現在の設定は、次のクエリを実行して表示できます。

  ```
  SELECT DB_PROPERTY( 'OptionWatchAction' );
  ```
- **OptionWatchList –** 設定したときに通知の対象としたり、データベース・サーバでエラーを返したりするデータベース・オプションのリストを、カンマで区切って指定します。この文字列の長さは 128 バイトに制限されています。デフォルトでは、空の文字列です。たとえば、次のコマンドは、ウォッチするオ

プションのリストに automatic_timestamp、float_as_double、および tsql_hex_constant の各オプションを追加します。

```
CALL sa_server_option( 'OptionWatchList','automatic_timestamp,
float_as_double,tsql_hex_constant' )
```

このプロパティの現在の設定は、次のクエリを実行して表示できます。

```
SELECT DB_PROPERTY( 'OptionWatchList' );
```

- **ProcedureProfiling –** ストアド・プロシージャ、関数、イベント、トリガについてのプロシージャ・プロファイリングを制御します。プロシージャ・プロファイリングは、ストアド・プロシージャ、関数、イベント、トリガの実行所要時間を示します。Sybase Central の [データベースのプロパティ] ウィンドウでも、プロシージャ・プロファイリング・オプションを設定できます。
  - **YES –** 現在の接続先データベースについてプロシージャ・プロファイリングを有効にします。
  - **NO –** プロシージャ・プロファイリングを無効にしますが、プロファイリング・データの表示は可能です。
  - **RESET –** YES または NO の設定は変更しないで、プロファイリング・カウンタを 0 に戻します。
  - **CLEAR –** プロファイリング・カウンタを 0 に戻し、プロシージャ・プロファイリングを無効にします。

    プロファイリングを有効にすると、システム・プロシージャ sa_procedure_profile_summary と sa_procedure_profile を使用して、データベースからプロファイリング情報を取り出すことができます。
- **ProfileFilterConn –** 他の接続がデータベースを使用する処理を阻害することなく、特定の接続 ID に関するプロファイル情報を取得するように、データベース・サーバに指示します。接続フィルタが有効な場合、`SELECT PROPERTY( 'ProfileFilterConn' )` の戻り値は監視されている接続の接続 ID です。ID が指定されていない場合、または接続フィルタが無効な場合、戻り値は -1 です。
- **ProfileFilterUser –** データベース・サーバに、特定のユーザ ID のプロファイリング情報を取得するよう指示します。
- **QuittingTime –** データベース・サーバに、指定された時間にサーバを停止するよう指示します。
- **RememberLastPlan –** 接続で最後に実行された長いテキスト・プランをキャプチャするように、クエリのデータベース・サーバに指示します。この設定は、-zp サーバ・オプションによって制御されます。

  RememberLastPlan が ON の場合は、LastPlanText 接続プロパティの値を問い合わせることで、この接続で最後に実行されたクエリの計画のテキスト表現を取得できます。

```
SELECT CONNECTION_PROPERTY( 'LastPlanText' );
```

- **RememberLastStatement –** データベース・サーバに、サーバ上で実行されている各データベースに関して最後に作成された SQL 文を取得するように指示します。ストアド・プロシージャ・コールの場合、プロシージャ内の文ではなく、最も外側のプロシージャ・コールのみが表示されます。

  RememberLastStatement が ON の場合は、LastStatement 接続プロパティの値を問い合わせることで、接続に関する LastStatement の現在の値を取得できます。

  ```
  SELECT CONNECTION_PROPERTY( 'LastStatement' );
  ```

  クライアントでの文のキャッシュが有効であり、キャッシュされた文が再使用されているとき、このプロパティは空の文字列を返します。

  RememberLastStatement が ON の場合、次の文は指定された接続に対して最後に準備された文を返します。

  ```
  SELECT CONNECTION_PROPERTY( 'LastStatement', connection-id );
  ```

  sa_conn_activity システム・プロシージャは、すべての接続に対して同じ情報を返します。

  警告！ -zl が指定されている場合、または RememberLastStatement サーバ設定がオンになっている場合、ユーザはだれでも sa_conn_activity システム・プロシージャを呼び出すか、LastStatement 接続プロパティの値を取得することにより、他のユーザが最後に準備した SQL 文を見つけることができます。このオプションは注意して使用し、不要な場合はオフにしてください。

- **RequestFilterConn –** 要求ログ情報をフィルタして、特定の接続の情報のみ記録されるようにします。これによって、多くのアクティブな接続または複数のデータベースのあるデータ・サーバを監視するときに、要求ログ・ファイルのサイズを削減できます。次の文を実行して、接続 ID を取得できます。

  ```
  CALL sa_conn_info( );
  ```

  接続 ID を取得した後でロギングする特定の接続を指定するには、次の文を実行します。

  ```
  CALL sa_server_option( 'RequestFilterConn', connection-id );
  ```

  フィルタは、明示的にリセットされるか、データベース・サーバが停止するまで有効なままになります。フィルタをリセットするには、次の文を使用します。

  ```
  CALL sa_server_option( 'RequestFilterConn', -1 );
  ```

- **RequestFilterDB –** 要求ログ情報をフィルタして、特定のデータベースの情報のみ記録されるようにします。これによって、複数のデータベースのあるサーバを監視するときに、要求ログ・ファイルのサイズを削減できます。目的のデータベースに接続しているときに次の文を実行して、データベース ID を取得できます。

  ```
  SELECT CONNECTION_PROPERTY( 'DBNumber' );
  ```

特定のデータベースについての情報のみロギングすることを指定するには、次の文を実行します。

```
CALL sa_server_option( 'RequestFilterDB', database-id );
```

フィルタは、明示的にリセットされるか、データベース・サーバが停止するまで有効なままになります。フィルタをリセットするには、次の文を使用します。

```
CALL sa_server_option( 'RequestFilterDB', -1 );
```

- **RequestLogFile –** 要求情報の記録に使用されるファイルの名前。空の文字列を指定すると、要求ログ・ファイルへのロギングが停止します。要求のロギングが有効でも、要求のログ・ファイルを指定しなかった場合、または空の文字列に設定されている場合、サーバは要求をデータベース・サーバ・メッセージ・ウィンドウにロギングします。SQL 文字列と同じように、パスの円記号は 2 つ重ねます。

  クライアントでの文のキャッシュが有効であるときに、`tracetime.pl` Perl スクリプトを使用してログを分析する場合は、要求ログの取得時に max_client_statements_cached オプションを 0 に設定してクライアントでの文のキャッシュを無効にする必要があります。
- **RequestLogging –** これは、データベース・サーバ・オプション -zr と -zo とともにトラブルシューティングで使用されます。次の値をプラス記号 (+) またはカンマで区切って組み合わせた値です。
  - **PLAN –** 実行プランのロギングを有効にします (短いプラン)。プロシージャの実行プランは、プロシージャ (PROCEDURES) のロギングが有効な場合にも記録されます。
  - **HOSTVARS –** ホスト変数の値のロギングを有効にします。HOSTVARS を指定した場合、SQL にリストされている情報もロギングされます。
  - **PROCEDURES –** プロシージャ内から実行されている文のロギングを有効にします。
  - **TRIGGERS –** トリガ内から実行されている文のロギングを有効にします。
  - **OTHER –** SQL に含まれないその他の要求タイプのロギングを有効にします (FETCH や PREFETCH など)。ただし、OTHER を指定して SQL を指定しない場合、SQL+OTHER を指定した場合と同じです。OTHER を含めると、ログ・ファイルが急速に拡大し、サーバのパフォーマンス低下につながることがあります。
  - **BLOCKS –** 別の接続で接続がブロックされたときと、接続のブロックが解除されたときに表示する詳細のロギングを有効にします。
  - **REPLACE –** ロギングの開始時に、既存の要求ログは同じ名前を持つ新規の (空の) ログで置換されます。それ以外の場合、既存の要求ログが開き、新規エントリがファイルの末尾に追加されます。
  - **ALL –** すべてのサポート情報をロギングします。これは、SQL+PLAN +HOSTVARS+PROCEDURES+TRIGGERS+OTHER+BLOCKS を指定した場合

と同じです。この設定では、ログ・ファイルが急速に拡大し、サーバのパフォーマンス低下につながることがあります。

- **NO または NONE –** 要求ログに対するロギングを無効にします。

このプロパティの現在の設定は、次のクエリを実行して表示できます。

```
SELECT PROPERTY( 'RequestLogging' );
```

- **RequestLogMaxSize –** 要求ログ情報の記録に使用されるファイルのバイト単位での最大サイズ 0 を指定した場合は、要求ロギング・ファイルの最大サイズは適用されず、名前は変更されません。これはデフォルト値です。

  要求ログ・ファイルが、sa_server_option システム・プロシージャまたは -zs サーバ・オプションで指定されたサイズに達すると、そのファイルは拡張子 `.old` が付いた名前に変更されます (既存のファイルが存在する場合は、同じ名前で置換されます)。要求ログ・ファイルが再起動します。
- **RequestLogNumFiles –** 保持する要求ログ・ファイルのコピーの数

  要求ロギングが長時間にわたって有効になっていると、要求ログ・ファイルが大きくなることがあります。-zn オプションを使用すると、保持する要求ログ・ファイルのコピー数を指定できます。
- **RequestTiming –** データベースに、各接続のタイミング情報を保守するように指示します。この機能はデフォルトでオフになっています。オンにすると、データベース・サーバは各接続の累積タイマを保守します。このタイマは、接続が確立した状態でのサーバ・ステータスごとの時間を示すものです。sa_performance_diagnostics システム・プロシージャを使用して、このタイミング情報の概要を取得できます。または、次の接続プロパティを調べて、各値を取得できます。

  ReqCountUnscheduled
  ReqTimeUnscheduled
  ReqCountActive
  ReqTimeActive
  ReqCountBlockIO
  ReqTimeBlockIO
  ReqCountBlockLock
  ReqTimeBlockLock
  ReqCountBlockContention
  ReqTimeBlockContention

  RequestTiming サーバ・プロパティがオンであると、追加のカウンタを保守するため、要求ごとに多少のオーバーヘッドがかかります。
- **SecureFeatures –** すでに動作しているデータベース・サーバの保護機能を有効または無効にできます。*feature-list* は、機能名または機能セットのカンマ区切りリストです。このリストに機能を追加することで、追加した機能の可用性を

制限します。保護機能のリストから項目を削除するには、保護機能名の前にマイナス記号 (-) を指定します。

機能を有効または無効にするために行った変更は、接続で直ちに有効になります。この設定は、sa_server_option システム・プロシージャを実行する接続に影響を与えません。変更を確認するには、切断してから再接続する必要があります。

注意： sa_server_option システム・プロシージャを使用して、機能を有効または無効にするには、データベース・サーバの起動時に -sk オプションを使用してキーを指定してから、secure_feature_key データベース・オプションの値を -sk で指定したキーに設定します (たとえば、`SET TEMPORARY OPTION secure_feature_key = 'j978kls12'` など)。secure_feature_key データベース・オプションを -sk 値に設定すると、保護機能の設定を変更できます。

たとえば、2 つの機能を無効にして 3 番目を有効にするには、次の構文を使用します。

```
CALL sa_server_option('SecureFeatures',
'CONSOLE_LOG,WEBCLIENT_LOG,-REQUEST_LOG' );
```

この文を実行すると、CONSOLE_LOG と WEBCLIENT_LOG が保護機能のリストに追加され、REQUEST_LOG がリストから削除されます。

- **StatisticsCleaner –** 統計クリーナでは、テーブルに対してスキャンを実行して、不適切な推定値を示す統計を修正します。デフォルトでは、統計クリーナはバックグラウンドで実行され、パフォーマンスへの影響は最小限に抑えられます。

  統計クリーナをオフにしても統計ガバナは無効になりませんが、統計クリーナがオフのときには、統計はクエリの実行時にのみ作成または修正されます。
- **WebClientLogFile –** Web サービス・クライアント・ログ・ファイルの名前。Web サービス・クライアント・ログ・ファイルは、-zoc サーバ・オプションや WebClientLogFile プロパティを使用して、ファイル名を設定またはリセットするたびにトランケートされます。これは文字列であるため、パスの円記号は 2 つ重ねます。
- **WebClientLogging –** このオプションは、Web サービス・クライアントのロギングを有効または無効にします。ログに記録される情報には、HTTP の要求と応答のデータが含まれています。ON を指定すると Web サービス・クライアント・ログ・ファイルへのロギングが開始され、OFF を指定するとファイルへのロギングが中止されます。

パーミッション
アプリケーション・プロファイリングまたは要求ロギングに関連する次のオプションには、DBA 権限または PROFILE 権限が必要です。

ProcedureProfiling
ProfileFilterConn
ProfileFilterUser
RequestFilterConn
RequestFilterDB
RequestLogFile
RequestLogging
RequestLogMaxSize
RequestLogNumFiles

この他のすべてのオプションには DBA 権限が必要です。

関連する動作
なし。

例
次の文は、データベース・サーバへの新しい接続を禁止します。

```
CALL sa_server_option( 'ConnsDisabled', 'YES' );
```

次の文は、現在のデータベースへの新しい接続を禁止します。

```
CALL sa_server_option( 'ConnsDisabledForDB', 'YES' );
```

次の文は、すべての SQL 文、プロシージャの呼び出し、プラン、イベントのブロックとブロック解除のロギングを有効にし、新規要求ログの開始を指定します。

```
CALL sa_server_option( 'RequestLogging', 'SQL+PROCEDURES+BLOCKS+PLAN
+REPLACE' );
```

sa_set_http_header システム・プロシージャ

Web サービスによる HTTP 応答ヘッダの設定を許可します。

構文

```
sa_set_http_header(
fldname,
val
)
```

引数

- ***fldname*** – HTTP ヘッダ・フィールドのいずれかの名前を含む文字列を指定する CHAR(128) パラメータ。
- ***val*** – 指定されたパラメータに設定する値を指定する LONG VARCHAR パラメータ。応答ヘッダを NULL に設定すると、そのヘッダが効果的に削除されます。

備考
特別なヘッダ・フィールド @HttpStatus を設定すると、要求によってステータス・コードが返されるように設定されます。ステータス・コードは応答コードとも呼ばれます。たとえば、次のスクリプトはステータス・コードを 404 Not Found に設定します。

```
CALL sa_set_http_header( '@HttpStatus', '404' );
```

3 桁のステータス・コードを指定し、コロンで区切ったテキスト・メッセージを任意で追加することで、ユーザ定義のステータス・メッセージを作成できます。たとえば、次のスクリプトは、メッセージ付きのステータス・コード "999 User Code" を出力します。

```
CALL sa_set_http_header( '@HttpStatus', '999:User Code' );
```

注意： LogOptions プロトコル・オプションを使用してログを取っている場合、ユーザ定義のステータス・テキスト・メッセージはデータベースの文字セットには変換されません。

エラー・メッセージの本文は、自動的に挿入されます。有効な HTTP エラー・コードだけが使用できます。ステータスに無効なコードを設定すると、SQL エラーが発生します。

sa_set_http_header プロシージャは、呼び出されたときに、ヘッダ・フィールドの既存のヘッダ値を必ず上書きします。

データベース・サーバによって自動的に生成された応答ヘッダは削除できます。たとえば、次のコマンドは Expires 応答ヘッダを削除します。

```
CALL sa_set_http_header( 'Expires', NULL );
```

パーミッション
なし。

関連する動作
なし。

例
次の例は、Content-Type ヘッダ・フィールドを text/html に設定します。

```
CALL sa_set_http_header( 'Content-Type', 'text/html' );
```

sa_set_http_option システム・プロシージャ

Web サービスによるプロセス制御用の HTTP オプションの設定を許可します。

構文

```
sa_set_http_option(
optname,
val
 )
```

引数

- ***optname*** – HTTP オプションのいずれかの名前を含む文字列を指定する CHAR(128) パラメータ。

  サポートされるオプションは次のとおりです。

  - **CharsetConversion** – このオプションを使用して、結果セットをデータベースの文字セット・エンコードからクライアントの文字セット・エンコードに自動的に変換するかどうかを制御します。指定できる値は、ON と OFF だけです。デフォルト値は ON です。
  - **AcceptCharset** – このオプションを使用して、応答の文字セット・エンコードに関する Web サーバの優先度を指定します。1 つ以上の文字セット・エンコードを優先度順に指定できます。このオプションの構文は、RFC2616 Hypertext Transfer Protocol の HTTP Accept-Charset 要求ヘッダ・フィールドの仕様に使用する構文に準拠します。

    Web ブラウザなどの HTTP クライアントでは、Accept-Charset 要求ヘッダを使用して、優先度順の文字セット・エンコードのリストを指定できます。必要に応じて、各エンコードには、対応する品質値 (q=*qvalue*) を指定できます。この品質値は、そのエンコードに対するクライアントの優先度を表します。デフォルトでは、品質値は 1 (q=1) です。次に例を示します。

    ```
    Accept-Charset: iso-8859-5, utf-8;q=0.8
    ```

    AcceptCharset HTTP オプション値内のプラス記号 (+) は、現在のデータベースの文字セット・エンコードを表すショートカットとして使用できます。また、プラス記号は、クライアントのリストでもエンコードが指定されている場合に、クライアントによって割り当てられている品質値に関係なく、データベースの文字セット・エンコードを優先する必要があることも示します。

    AcceptCharset HTTP オプション内のアスタリスク (*) は、クライアントとサーバのリストに共通部分がない場合に、クライアントで優先される文字セット・エンコードがサーバでもサポートされていれば、Web サービスでその文字セット・エンコードを使用する必要があることを示します。

応答の送信時には、クライアントと Web サービスの両方で優先される最初の文字セット・エンコードが使用されます。クライアントの優先度順が最初に考慮されます。共通のエンコード優先度が存在しない場合は、Web サービスのリストにアスタリスク (*) がなければ、Web サービスで最も優先されるエンコードが使用されます。アスタリスクがある場合は、クライアントで最も優先されるエンコードが使用されます。

AcceptCharset HTTP オプションを使用しない場合は、クライアントで指定され、サーバでサポートされている最も優先度の高い文字セット・エンコードが使用されます。クライアントで指定されたどのエンコードもサポートされていない場合 (またはクライアントが Accept-Charset 要求ヘッダを送信しない場合) は、データベースの文字セット・エンコードが使用されます。

クライアントが Accept-Charset ヘッダを送信しない場合は、次のいずれかのアクションが実行されます。

- AcceptCharset HTTP オプションが指定されていない場合、Web サーバではデータベースの文字セット・エンコードを使用します。
- AcceptCharset HTTP オプションが指定されている場合、Web サーバでは最も優先度の高い文字セット・エンコードを使用します。

クライアントが Accept-Charset ヘッダを送信する場合は、次のいずれかのアクションが実行されます。

- AcceptCharset HTTP オプションが指定されていない場合、Web サーバではクライアントで優先される文字セット・エンコードのいずれかを使用しようとします (最も優先度の高いエンコードから開始)。クライアントで優先されるどのエンコードも Web サーバでサポートされていない場合、Web サーバではデータベースの文字セット・エンコードを使用します。
- AcceptCharset HTTP オプションが指定されている場合、Web サーバでは両方の優先度リストで共通する最初の文字セット・エンコードを使用しようとします (クライアントで最も優先されるエンコードから開始)。たとえば、クライアントが送信する Accept-Charset ヘッダ・リストのエンコードの優先度順が iso-a、iso-b、iso-c であり、Web サーバでの優先度順が iso-b、iso-a、iso-c の場合、iso-a が選択されます。

  ```
  Web client: iso-a, iso-b, iso-c
  Web server: iso-b, iso-a, iso-c
  ```

  2 つのリストに共通部分がない場合、Web サーバで最初に優先される文字セットが使用されます。次の例では、エンコード iso-d が使用されます。

  ```
  Web client: iso-a, iso-b, iso-c
  Web server: iso-d, iso-e, iso-f
  ```

AcceptCharset HTTP オプションにアスタリスク ('*') が含まれている場合は、クライアントのエンコードが優先されるため、iso-a が使用されます。基本的に、アスタリスクを使用すると、2 つのリストに共通部分がないという状況がなくなります。

理想的な状況は、クライアントと Web サービスの両方でデータベースの文字セット・エンコードが使用される場合です。このような場合、文字セットの変換の必要性がなくなり、Web サーバの応答時間が向上します。

CharsetConversion オプションを OFF に設定している場合、AcceptCharset の処理は実行されないことに注意してください。

- **SessionID –** このオプションを使用して、HTTP セッションの作成、削除、または名前の変更を行います。Web サービスでこのオプションを設定して HTTP セッションを作成するときにはデータベース接続は維持されますが、サーバの再起動時にはセッションは維持されません。すでにセッション・コンテキスト内にある場合、この呼び出しによってセッションの名前が新しいセッション ID に変更されます。NULL 値を使用して呼び出した場合、Web サービスが終了するとセッションは削除されます。

  生成されるセッション・キーは 128 文字に制限され、複数のデータベースがロードされている場合はデータベース間でユニークになります。
- **SessionTimeout –** このオプションを使用して、非アクティブ状態の HTTP セッションを保持する時間を分単位で指定します。このタイムアウト期間は、HTTP 要求で指定のセッションが使用されると常にリセットされます。SessionTimeout を過ぎると、セッションは自動的に削除されます。

- ***val* –** 指定されたオプションに設定する値を指定する LONG VARCHAR パラメータ。

備考
Web サービスを処理する文またはプロシージャ内でこのプロシージャを使用して、オプションを設定します。

Web サービスを介して呼び出されたプロシージャ内から sa_set_http_option が呼び出されたときに、オプションまたはオプション値のどちらかが無効であると、エラーが返されます。

パーミッション
なし。

関連する動作
なし。

例
次の例は、sa_set_http_option を使用して、データベースの文字セット・エンコードに関する Web サービスの優先度を指定する方法を示します。2 番目の選択肢として UTF-8 エンコードが指定されています。アスタリスク (*) は、クライアントで最も優先される文字セット・エンコードが Web サーバでサポートされている場合に、Web サービスでその文字セット・エンコードが使用されることを示しています。

```
CALL sa_set_http_option( 'AcceptCharset', '+,UTF-8,*');
```

次の例は、sa_set_http_option を使用して、Web サービスで使用されている文字エンコードを正しく識別する方法を示します。この例では、Web サーバは 1251CYR データベースに接続されており、キリル語のアルファベットを含む HTML ドキュメントを任意の Web ブラウザに提供するように準備されています。

```
CREATE PROCEDURE cyrillic_html()
RESULT (html_doc XML)
BEGIN
  DECLARE pos INT;
  DECLARE charset VARCHAR(30);
  CALL sa_set_http_option( 'AcceptCharset', 'iso-8859-5, utf-8' );
  SET charset = CONNECTION_PROPERTY( 'CharSet' );
  -- Change any IANA labels like ISO_8859-5:1988
  -- to ISO_8859-5 for Firefox.
  SET pos = LOCATE( charset, ':' );
  IF pos > 0 THEN
    SET charset = LEFT( charset, pos - 1 );
  END IF;
  CALL sa_set_http_header( 'Content-Type', 'text/html; charset=' ||
        charset );
  SELECT  '<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">' ||
    XMLCONCAT(
      XMLELEMENT('HTML',
        XMLELEMENT('HEAD',
          XMLELEMENT('TITLE', 'Cyrillic characters')
        ),
        XMLELEMENT('BODY',
          XMLELEMENT('H1', 'First 5 lowercase Russian letters'),
          XMLELEMENT('P', UNISTR('¥u0430¥u0431¥u0432¥u0433¥u0434'))
        )
      )
    );
END;
CREATE SERVICE cyrillic
TYPE 'RAW'
AUTHORIZATION OFF
USER DBA
AS CALL cyrillic_html();
```

Firefox などの Web ブラウザから Web サービスに配信される、次の Accept-Charset ヘッダを例に、使用する正しい文字セット・エンコードを確立するプロセスにつ

いて説明します。このヘッダは、ブラウザでは ISO-8859-1 および UTF-8 エンコードが優先されるものの、他のエンコードも処理できることを示しています。

```
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
```

転送される Web ページにキリル語の文字が含まれているため、Web サービスでは ISO-8859-1 文字セット・エンコードは受け入れられません。Web サービスでは、sa_set_http_option の呼び出しで指定されているように、ISO-8859-5 または UTF-8 エンコードが優先されます。この例では、UTF-8 エンコードが選択されます (両方で許容されるため)。データベース接続プロパティ 'CharSet' は、どのエンコードが Web サービスで選択されたかを示します。sa_set_http_header プロシージャは、Web ブラウザに対して HTML ドキュメントのエンコードを示すために使用します。

```
Content-Type: text/html; charset=UTF-8
```

Web ブラウザで Accept-Charset が指定されていない場合、Web サービスではデフォルトで最初の優先エンコード ISO-8859-5 が使用されます。sa_set_http_header プロシージャは、HTML ドキュメントのエンコードを示すために使用します。

```
Content-Type: text/html; charset=ISO_8859-5
```

次の例では、ユニークな HTTP セッション識別子を設定します。

```
DECLARE sessionid VARCHAR(30);
DECLARE tm TIMESTAMP;
SET tm = NOW(*);
SET sessionid = 'MySessions_' ||
  CONVERT( VARCHAR, SECONDS(tm)*1000 + DATEPART(millisecond,tm));
CALL sa_set_http_option('SessionID', sessionid);
```

次の例では、HTTP セッションのタイムアウトを 5 分に設定します。

```
CALL sa_set_http_option('SessionTimeout', '5');
```

sa_table_page_usage システム・プロシージャ

データベース・テーブルのページの使用状況をレポートします。

構文

```
sa_table_page_usage( )
```

結果セット

カラム名	データ型	説明
TableId	UNSIGNED INTEGER	テーブル ID。
TablePages	INTEGER	テーブルで使用されるテーブル・ページの数。

カラム名	データ型	説明
PctUsedT	INTEGER	使用されているテーブル・ページ領域の割合。
IndexPages	INTEGER	テーブルで使用されるインデックス・ページの数。
PctUsedI	INTEGER	使用されているインデックス・ページ領域の割合。
PctOfFile	INTEGER	テーブルが使用しているデータベース・ファイル総数の割合。
TableName	CHAR(128)	テーブル名。

備考
結果には、情報ユーティリティで提供される情報と同じ内容が含まれています。progress_messages データベース・オプションが Raw または Formatted に設定されている場合、このシステム・プロシージャの実行中にデータベース・サーバからクライアントに進行状況メッセージが送信されます。

パーミッション
DBA 権限

関連する動作
なし。

sa_validate システム・プロシージャ

データベースの全部または一部を検証します。

構文

```
sa_validate(
  [ tbl_name  [, owner_name ] ]
)
```

引数

- ***tbl_name*** – 検証するテーブルまたはマテリアライズド・ビューの名前を指定する任意の VARCHAR(128) パラメータ。
- ***owner_name*** – 所有者を指定する任意の VARCHAR(128) パラメータ。所有者のみを指定すると、この所有者が所有するすべてのテーブルとマテリアライズド・ビューが検証されます。

パーミッション
DBA 権限

関連する動作
なし。

備考

指定される引数	検証タイプ
なし。	データベース内のすべてのテーブル、マテリアライズド・ビュー、インデックスが検証されます。チェックサムの検証を含め、データベース自体も検証されます。
owner_name	指定されたユーザが所有するすべてのテーブル、マテリアライズド・ビュー、インデックスが検証されます。
tbl_name	現在のユーザが所有している指定のテーブルまたはマテリアライズド・ビュー、およびそのすべてのインデックスが検証されます。
owner_name と *tbl_name*	指定されたユーザが所有している指定のテーブルまたはマテリアライズド・ビュー、およびそのすべてのインデックスが検証されます。

tbl_name と *owner_name* の値は文字列であり、引用符で囲む必要があります。

プロシージャは Messages という 1 つのカラムを返します。検証時にエラーが返されると、カラムに表示されます。エラーなく検証が成功した場合は、カラムに `No error detected` が格納されます。

警告！ テーブルまたはデータベース全体の検証は、データベースに変更を加えている接続がない場合に実行してください。そうしないと、実際に破損していなくても、何らかの形でデータベースが破損したことを示すエラーがレポートされます。

例
次の文は、DBA が所有するテーブルとマテリアライズド・ビューの検証を実行します。

```
CALL sa_validate( owner_name = 'DBA' );
```

sa_verify_password システム・プロシージャ

現在のユーザのパスワードを検証します。

構文

```
sa_verify_password( curr_pswd )
```

引数

- ***curr_pwsd*** – 現在のデータベース・ユーザのパスワードを指定する CHAR(128) パラメータ。

備考
このプロシージャは sp_password で使用されます。パスワードが一致した場合、そのパスワードは受け入れられます。パスワードが一致しなかった場合、エラーが返されます。

パーミッション
なし。

関連する動作
なし。

sp_login_environment システム・プロシージャ

ユーザがログインするときの接続オプションを設定します。

構文

```
sp_login_environment( )
```

備考
sp_login_environment は、デフォルトで login_procedure データベース・オプションによって呼び出されるプロシージャです。

このプロシージャを編集しないことをおすすめします。ログイン環境を変更するには、異なるプロシージャを指すよう login_procedure オプションを変更します。

sp_login_environment プロシージャのテキストを次に示します。

```
CREATE PROCEDURE dbo.sp_login_environment( )
BEGIN
  IF connection_property( 'CommProtocol' ) = 'TDS' THEN
    CALL dbo.sp_tsql_environment( )
  END IF
END;
```

パーミッション
なし。

関連する動作
なし。

sp_remote_columns システム・プロシージャ

リモート・テーブルにあるカラムのリストと、それらのデータ型の説明を生成します。

このシステム・プロシージャを使用するには、サーバを **CREATE SERVER** 文で定義します。

構文

```
sp_remote_columns ( @servername , @tablename [, @table_owner ] [,
@table_qualifier ]] )
```

詳細については、『SQL Anywhere サーバー - SQL リファレンス』の「システムプロシージャー」>「システムプロシージャーのアルファベット順リスト」>「sp_remote_columns システムプロシージャー」を参照してください。

注意： 参照先は SQL Anywhere のマニュアルです。

sp_remote_exported_keys システム・プロシージャ

指定されたプライマリ・テーブルに外部キーを持つテーブルに関する情報を表示します。

このシステム・プロシージャを使用するには、サーバを CREATE SERVER 文で定義します。

構文

```
sp_remote_exported_keys(
  server_name
  , sp_name
  [, sp_owner
  [, sp_qualifier ] ]
)
```

引数

- ***server_name*** – この CHAR(128) パラメータを使用して、リモート・テーブルが格納されているサーバを指定します。
- ***sp_name*** – プライマリ・キーを格納するテーブルを指定する CHAR(128) パラメータ。
- ***sp_owner*** – プライマリ・テーブルの所有者を指定する任意の CHAR(128) パラメータ。
- ***sp_qualifier*** – プライマリ・テーブルを格納するデータベースを指定する任意の CHAR(128) パラメータ。

結果セット

カラム名	データ型	説明
pk_database	CHAR(128)	プライマリ・キー・テーブルがあるデータベース。
pk_owner	CHAR(128)	プライマリ・キー・テーブルの所有者。
pk_table	CHAR(128)	プライマリ・キー・テーブル。
pk_column	CHAR(128)	プライマリ・キー・カラムの名前。
fk_database	CHAR(128)	外部キー・テーブルがあるデータベース。
fk_owner	CHAR(128)	外部キー・テーブルの所有者。
fk_table	CHAR(128)	外部キー・テーブル。
fk_column	CHAR(128)	外部キー・カラムの名前。
key_seq	SMALLINT	キーのシーケンス番号。
fk_name	CHAR(128)	外部キーの名前。
pk_name	CHAR(128)	プライマリ・キーの名前。

備考
このプロシージャは、特定のプライマリ・テーブルに外部キーを持つリモート・テーブルに関する情報を提供します。sp_remote_exported_keys システム・プロシージャの結果セットには、プライマリ・キーと外部キーの両方のデータベース、所有者、テーブル、カラム、名前と、外部キー・カラムの外部キー・シーケンスが含まれます。基本となる ODBC と JDBC 呼び出しのために結果セットが変わる場合もありますが、外部キーのテーブルとカラムに関する情報は常に返されます。

パーミッション
なし。

関連する動作
なし。

例
この例では、システム・プロシージャを呼び出すときに名前付きパラメータを使用し、サーバ asetest で、運用データベースの SYSOBJECTS テーブルに外部キーを持つリモート・テーブルに関する情報を返します。

```
CALL sp_remote_exported_keys(
     @server_name='asetest',
     @sp_name='sysobjects',
     @sp_qualifier='production' );
```

sp_remote_imported_keys システム・プロシージャ

指定された外部キーに対応するプライマリ・キーを持つリモート・テーブルに関する情報を提供します。

このシステム・プロシージャを使用するには、サーバを CREATE SERVER 文で定義します。

構文

```
sp_remote_imported_keys(
server_name
 , sp_name
 [, sp_owner
 [, sp_qualifier ] ]
)
```

引数

- ***server_name*** – 外部キー・テーブルが配置されているサーバを指定する任意の CHAR(128) パラメータ。このパラメータの値は必須です。
- ***sp_name*** – 外部キーを格納するテーブルを指定する任意の CHAR(128) パラメータ。このパラメータの値は必須です。
- ***sp_owner*** – 外部キー・テーブルの所有者を指定する任意の CHAR(128) パラメータ。
- ***sp_qualifier*** – 外部キー・テーブルを含むデータベースを指定する任意の CHAR(128) パラメータ。

結果セット

カラム名	データ型	説明
pk_database	CHAR(128)	プライマリ・キー・テーブルがあるデータベース。
pk_owner	CHAR(128)	プライマリ・キー・テーブルの所有者。
pk_table	CHAR(128)	プライマリ・キー・テーブル。
pk_column	CHAR(128)	プライマリ・キー・カラムの名前。
fk_database	CHAR(128)	外部キー・テーブルがあるデータベース。
fk_owner	CHAR(128)	外部キー・テーブルの所有者。
fk_table	CHAR(128)	外部キー・テーブル。
fk_column	CHAR(128)	外部キー・カラムの名前。
key_seq	SMALLINT	キーのシーケンス番号。

カラム名	データ型	説明
fk_name	CHAR(128)	外部キーの名前。
pk_name	CHAR(128)	プライマリ・キーの名前。

備考
外部キーは、対応するプライマリ・キーを持つ別のテーブル内のローを参照します。このプロシージャを使用すると、特定の外部テーブルに対応するプライマリ・キーを持つリモート・テーブルのリストを取得できます。
sp_remote_imported_keys の結果セットには、プライマリ・キーと外部キーの両方のデータベース、所有者、テーブル、カラム、名前と、外部キー・カラムの外部キー・シーケンスが含まれます。基本となる ODBC と JDBC 呼び出しのために結果セットが変わる場合もありますが、プライマリ・キーのテーブルとカラムに関する情報は常に返されます。

パーミッション
なし。

関連する動作
なし。

例
次の例では、プロシージャを呼び出すときに名前付きパラメータを使用し、asetest サーバの SYSOBJECTS テーブルの外部キーに対応するプライマリ・キーを持つテーブルを返します。

```
CALL sp_remote_imported_keys(
     @server_name='asetest',
     @sp_name='sysobjects',
     @sp_qualifier='production' );
```

sp_remote_primary_keys システム・プロシージャ

リモート・データ・アクセスを使用して、リモート・テーブルのプライマリ・キー情報を表示します。

構文

```
sp_remote_primary_keys( @server_name [, @table_name  [, @table_owner
[, @table_qualifier ] ] ])
```

詳細については、『SQL Anywhere サーバー - SQL リファレンス』の「システムプロシージャー」>「システムプロシージャーのアルファベット順リスト」>「sp_remote_imported_keys システムプロシージャー」を参照してください。

注意： 参照先は SQL Anywhere のマニュアルです。

sp_remote_tables システム・プロシージャ

サーバ上のテーブルのリストを返します。

このシステム・プロシージャを使用するには、サーバを CREATE SERVER 文で定義します。

構文

```
sp_remote_tables(
 server_name
 [, table_name
 [, table_owner
 [, table_qualifier
 [, with_table_type ] ] ] ]
)
```

引数

- ***server_name*** – リモート・テーブルが配置されているサーバを指定する CHAR(128) パラメータ。
- ***table_name*** – リモート・テーブルを指定する任意の CHAR(128) パラメータ。
- ***table_owner*** – リモート・テーブルの所有者を指定する任意の CHAR(128) パラメータ。
- ***table_qualifier*** – *table_name* が格納されているデータベースを指定する CHAR(128) パラメータ。
- ***with_table_type*** – リモート・テーブルのタイプを指定する任意の BIT パラメータ。この引数は、Bit データ型で、0 (デフォルト) と 1 の 2 つの値を指定できます。テーブルのタイプをリストするカラムを結果セットに入れる場合は、値 1 を指定してください。

結果セット

カラム名	データ型	説明
database	CHAR(128)	リモート・データベースの名前。
owner	CHAR(128)	リモート・データベース所有者の名前。
table-name	CHAR(128)	リモート・テーブル。
table-type	CHAR(128)	テーブル・タイプを指定します。値は、リモート・サーバのタイプによって異なります。たとえば、指定できる値として TABLE、VIEW、SYS、GBL TEMP があります。

備考
データベース・サーバを設定するときに、特定のサーバ上で使用可能なリモート・テーブルのリストを取得しておくと役立つ場合があります。このプロシージャは、サーバ上のテーブルのリストを返します。

このプロシージャには 5 つのパラメータを指定できます。テーブル、所有者、またはデータベース名を指定すると、テーブルのリストはその引数に当てはまるものだけに限定されます。

標準と互換性

- **Sybase –** Open Client/Open Server でサポートされています。

パーミッション
なし。

関連する動作
なし。

例
サーバ excel で参照されている ODBC データソースから、使用可能なすべての Microsoft Excel ワークシートのリストを取得します。

```
CALL sp_remote_tables( 'excel' );
```

Adaptive Server Enterprise サーバ asetest の production データベースにある fred が所有するすべてのテーブルのリストを取得します。

```
CALL sp_remote_tables( 'asetest', null, 'fred', 'production' );
```

sp_servercaps システム・プロシージャ

リモート・サーバの機能についての情報を表示します。

このシステム・プロシージャを使用するには、サーバを CREATE SERVER 文で定義します。

構文

```
sp_servercaps( sname )
```

引数

- ***sname*** **–** CREATE SERVER 文で定義されるサーバを指定する CHAR(64) パラメータ。*sname* は、CREATE SERVER 文で使用されるサーバ名と同じである必要があります。

備考
このプロシージャは、リモート・サーバの機能についての情報を表示します。この機能の情報を使用して、どのくらいの SQL 文をリモート・サーバに転送できるかを判断します。サーバの機能をリストする ISYSCAPABILITY システム・テーブルは、最初のリモート・サーバへの接続が確立されるまで設定されません。

標準と互換性

- **Sybase –** Open Client/Open Server でサポートされています。

パーミッション
なし。

関連する動作
なし。

例
リモート・サーバ testasa に関する情報を表示します。

```
CALL sp_servercaps( 'testasa' );
```

sp_tsql_environment システム・プロシージャ

ユーザが jConnect または Open Client アプリケーションから接続するときの接続オプションを設定します。

構文

```
sp_tsql_environment( )
```

備考
sp_login_environment プロシージャは、login_procedure データベース・オプションによって指定されるデフォルトのプロシージャです。新規接続ごとに、login_procedure で指定されたプロシージャが呼び出されます。接続に TDS 通信プロトコルを使用する場合 (つまり、Open Client または jConnect 接続の場合)、今度は sp_login_environment が sp_tsql_environment を呼び出します。

このプロシージャは、デフォルトの Adaptive Server Enterprise の動作との互換性を持つように、データベース・オプションを設定します。

デフォルトの動作を変更したい場合は、新しいプロシージャを作成して、その新しいプロシージャを指すように login_procedure オプションを変更することをおすすめします。

パーミッション
なし。

関連する動作
なし。

例
sp_tsql_environment プロシージャのテキストを次に示します。

```
CREATE PROCEDURE dbo.sp_tsql_environment()
BEGIN
    IF db_property( 'IQStore' ) = 'Off' THEN
        -- SQL Anywhere datastore
        SET TEMPORARY OPTION close_on_endtrans='OFF';
    END IF;
    SET TEMPORARY OPTION ansinull='OFF';
    SET TEMPORARY OPTION tsql_variables='ON';
    SET TEMPORARY OPTION ansi_blanks='ON';
    SET TEMPORARY OPTION chained='OFF';
    SET TEMPORARY OPTION quoted_identifier='OFF';
    SET TEMPORARY OPTION allow_nulls_by_default='OFF';
    SET TEMPORARY OPTION on_tsql_error='CONTINUE';
    SET TEMPORARY OPTION isolation_level='1';
    SET TEMPORARY OPTION date_format='YYYY-MM-DD';
    SET TEMPORARY OPTION timestamp_format='YYYY-MM-DD HH:NN:SS.SSS';
    SET TEMPORARY OPTION time_format='HH:NN:SS.SSS';
    SET TEMPORARY OPTION date_order='MDY';
    SET TEMPORARY OPTION escape_character='OFF';
END
```

Adaptive Server Enterprise のシステム・プロシージャとカタログ・プロシージャ

Adaptive Server Enterprise には、システム・プロシージャとカタログ・プロシージャがあります。これらは、多くの管理機能を実行し、システム情報を取得します。Sybase IQ は、こうしたプロシージャの一部をサポートしています。

システム・プロシージャは、システム・テーブルから情報を取得して更新するための組み込みストアド・プロシージャです。カタログ・ストアド・プロシージャは、システム・テーブルから表形式で情報を取り出します。

注意： これらのプロシージャは Adaptive Server Enterprise やバージョン 12 より前の Sybase IQ と同様の機能を実行しますが、まったく同じというわけではありません。 既存のスクリプトでこれらのプロシージャが使われている場合、動作をチェックしておくことをおすすめします。 ストアド・プロシージャのテキストを表示するには、次を実行します。

```
sp_helptext 'owner.procedure_name'
```

Sybase が提供するすべてのシステム・ストアド・プロシージャの所有者は dbo です。 別のユーザが所有する同じ名前のストアド・プロシージャのテキストを参照するには、そのユーザを指定する必要があります。次に示すのはその例です。

```
sp_helptext 'myname.myprocedure'
```

Adaptive Server Enterprise のシステム・プロシージャ

Sybase IQ でサポートされる Adaptive Server Enterprise システム・プロシージャ

表 209 : Sybase IQ でサポートされる ASE システム・プロシージャ

システム・プロシージャ	説明	パーミッション
sp_addgroup*group-name*	データベースにグループを追加します。	既存のユーザをグループに変更するには、DBA または PERMS ADMIN が必要です。新しいユーザを作成し、それをグループに変更するには、DBA、または USER ADMIN と PERMS ADMIN の両方が必要です。
sp_addlogin*userid, password[, defdb [, deflanguage [, fullname]]]*	データベースに新規ユーザ・アカウントを追加します。	DBA 権限または USER ADMIN 権限が必要です。
sp_addmessage*message-num, message_text [, language]*	ストアド・プロシージャの **PRINT** 呼び出しと **RAISERROR** 呼び出しに使用するユーザ定義メッセージを、SYSUSERMESSAGES に追加する。	DBA 権限または RESOURCE 権限が必要です。
sp_addtype*typename, data-type, [, "identity" | null-type]*	ユーザ定義データ型を作成します。Sybase IQ は IDENTITY カラムをサポートしません。	DBA 権限または RESOURCE 権限が必要です。

システム・プロシージャ	説明	パーミッション
sp_adduser*userid [, name_in_db [, grpname]]*	データベースに新規ユーザを追加します。	新しいユーザを作成するには、DBA 権限または USER ADMIN 権限が必要です。新しいユーザを作成し、それを特定のグループに追加するには、DBA 権限、または USER ADMIN と PERMS ADMIN の両方の権限が必要です。
sp_changegroup*new-group-name, userid*	ユーザのグループを変更する、またはグループにユーザを追加します。	DBA 権限または PERMS ADMIN 権限が必要です。
sp_dboption*[dbname, opt-name, {true | false}]*	データベース・オプションを表示または変更します。	不要。
sp_dropgroup*group-name*	データベースからグループを削除します。	DBA 権限または PERMS ADMIN 権限が必要です。
sp_droplogin*userid*	データベースからユーザを削除します。	DBA 権限または USER ADMIN 権限が必要です。
sp_dropmessage*message-number [, language]*	ユーザ定義メッセージを削除します。	DBA 権限または RESOURCE 権限が必要です。
sp_droptype*typename*	ユーザ定義データ型を削除します。	DBA 権限または RESOURCE 権限が必要です。
sp_dropuser*userid*	データベースからユーザを削除します。	DBA 権限または USER ADMIN 権限が必要です。
sp_getmessage*message-num, @msg-var output [, language]*	**PRINT** 文と **RAISERROR** 文で使用するために、SYSUSERMESSAGES 内に格納されているメッセージ文字列を取得します。	不要。
sp_helptext '*owner.object-name*'	システム・プロシージャまたはビューのテキストを表示します。	不要。

システム・プロシージャ	説明	パーミッション
sp_password*caller_passwd, new_passwd [, userid]*	ユーザ ID のパスワードを追加または変更します。	他のユーザのパスワードを変更するには、DBA 権限、または DBA 権限を持たないユーザの場合は PERMS ADMIN 権限が必要です。これらの権限がない場合も、自身のパスワードは変更できます。

注意： **sp_dropuser** などのプロシージャには、Adaptive Server Enterprise のストアド・プロシージャとの互換性はほとんどありません。Adaptive Server Enterprise (または Sybase IQ 11.x) のストアド・プロシージャを使い慣れている場合は、それらのテキストと Sybase IQ 12 のプロシージャとを比較してから、Interactive SQL でプロシージャを使用してください。 比較には、次のコマンドを使用します。

```
sp_helptext 'owner.procedure_name'
```

Sybase が提供するシステム・ストアド・プロシージャの場合、所有者は常に dbo です。別のユーザが所有する同じ名前のストアド・プロシージャのテキストを参照するには、そのユーザを指定する必要があります。次に示すのはその例です。

```
sp_helptext 'myname.myprocedure'
```

参照：

- ユーザ、グループ、パーミッション (721 ページ)

Adaptive Server Enterprise のカタログ・プロシージャ

Sybase IQ には、sp_column_privileges プロシージャ以外の Adaptive Server Enterprise のカタログ・プロシージャがほとんど実装されています。

また、Sybase IQ には、これらの Adaptive Server Enterprise カタログ・プロシージャの一部と同様の、カスタマイズされたストアド・プロシージャが用意されています。

表 210 : Sybase IQ に実装されている ASE カタログ・プロシージャ

ASE カタログ・プロシージャ	説明	IQ プロシージャ
sp_columns *table-name [, table-owner] [, table-qualifier] [, column-name]*	指定したカラムのデータ型を返します。	

ASE カタログ・プロシージャ	説明	IQ プロシージャ
sp_fkeys*pktable_name [, pktable-owner][, pktable-qualifier] [, fktable-name] [, fktable_owner] [, fktable-qualifier]*	指定したテーブルの外部キー情報を返します。	
sp_pkeys*table-name [, table_owner] [, table_qualifier]*	指定したテーブルのプライマリ・キー情報を返します。	**sp_iqpkeys**
sp_special_columns*table_name [, table-owner] [, table-qualifier] [, col-type]*	テーブルのローをユニークに識別するのに最適なカラム・セットを返します。	
sp_sproc_columns*proc-name [, proc_owner] [, proc-qualifier] [, column-name]*	ストアド・プロシージャの入力パラメータおよびリターン・パラメータ情報を返します。	**sp_iqprocparm**
sp_stored_procedures*[sp-name] [, sp-owner] [, sp-qualifier]*	1 つ以上のストアド・プロシージャの情報を返します。	**sp_iqprocedure**
sp_tables*table-name [, table-owner] [, table-qualifier] [, table-type]*	**FROM** 句に表示できるオブジェクトのリストを返します。	

次の Adaptive Server Enterprise カタログ・プロシージャはサポートされていません。

- `sp_column_privileges`
- `sp_databases`
- `sp_datatype_info`
- `sp_server_info`

SQL Anywhere でサポートされているプロシージャ

Sybase IQ は SQL Anywhere のシステム・プロシージャをサポートします。

詳細については、『SQL Anywhere サーバー - SQL リファレンス』の「システムオブジェクト」>「システムプロシージャー」>「システムプロシージャーのアルファベット順リスト」を参照してください。

sa_get_table_definition プロシージャは、SQL Anywhere テーブルについてのみサポートされます。 このプロシージャを IQ テーブルに対して実行すると、`not implemented for IQ tables` というエラーが返されます。

システム・テーブルとシステム・ビュー

これ以降のトピックでは、Sybase IQ がサポートするシステム・テーブル、システム・ビュー、統合ビュー、互換ビュー、ASE T-SQL 互換ビューについて説明します。

システム・テーブル

すべての Sybase IQ データベースの構造は、多くのシステム・テーブルに記述されています。 システム・テーブルは内部での使用を目的として設計されています。

DUMMY システム・テーブルは、ユーザが直接アクセスできる唯一のシステム・テーブルです。その他のすべてのシステム・テーブルについては、その対応するビューを通じて基本データにアクセスします。

表 211 : システム・テーブルのリスト

システム・テーブル	内部使用の有無
DUMMY	不可
ISYSARTICLE	可
ISYSARTICLECOL	可
ISYSATTRIBUTE	可
ISYSATTRIBUTENAME	可
ISYSCAPABILITY	可
ISYSCHECK	可
ISYSCOLPERM	可
ISYSCOLSTAT	可
ISYSCONSTRAINT	可
ISYSDBFILE	可
ISYSDBSPACE	可
ISYSDBSPACEPERM	可
ISYSDEPENDENCY	可
ISYSDOMAIN	可

システム・テーブル	内部使用の有無
ISYSEVENT	可
ISYSEXTERNENV	可
ISYSEXTERNENVOBJECT	可
ISYSEXTERNLOGIN	可
ISYSFKEY	可
ISYSGROUP	可
ISYSHISTORY	可
ISYSIDX	可
ISYSIDXCOL	可
ISYSIQBACKUPHISTORY	可
ISYSIQBACKUPHISTORYDETAIL	可
ISYSIQDBFILE	可
ISYSIQDBSPACE	可
ISYSIQIDX	可
ISYSIQINFO	可
ISYSIQJOINIDX	可
ISYSIQJOINIXCOLUMN	可
ISYSIQLOGICALSERVER	可
ISYSIQLOGINPOLICYLSINFO	可
ISYSIQLSLOGINPOLICYOPTION	可
ISYSIQLSMEMBER	可
ISYSIQLSPOLICY	可
ISYSIQLSPOLICYOPTION	可
ISYSIQMPXSERVER	可
ISYSIQPARTITIONCOLUMN	可
ISYSIQTAB	可
ISYSIQTABCOL	可
ISYSJAR	可

システム・テーブル	内部使用の有無
ISYSJARCOMPONENT	可
ISYSJAVACLASS	可
ISYSLOGINMAP	可
ISYSLOGINPOLICY	可
ISYSLOGINPOLICYOPTION	可
ISYSMVOPTION	可
ISYSMVOPTIONNAME	可
ISYSOBJECT	可
ISYSOPTION	可
ISYSOPTSTAT	可
ISYSPARTITION	可
ISYSPARTITIONKEY	可
ISYSPARTITIONSCHEME	可
ISYSPHYSIDX	可
ISYSPROCEDURE	可
ISYSPROCPARM	可
ISYSPROCPERM	可
ISYSPROXYTAB	可
ISYSPUBLICATION	可
ISYSREMARK	可
ISYSREMOTEOPTION	可
ISYSREMOTEOPTIONTYPE	可
ISYSREMOTETYPE	可
ISYSREMOTEUSER	可
ISYSSCHEDULE	可
ISYSSERVER	可
ISYSSOURCE	可
ISYSSQLSERVERTYPE	可

システム・テーブル	内部使用の有無
ISYSSUBPARTITIONKEY	可
ISYSSUBSCRIPTION	可
ISYSSYNC	可
ISYSSYNCPROFILE	可
ISYSSYNCSCRIPT	可
ISYSTAB	可
ISYSTABCOL	可
ISYSTABLEPERM	可
ISYSTEXTCONFIG	可
ISYSTEXTIDX	可
ISYSTEXTIDXTAB	可
ISYSTRIGGER	可
ISYSTYPEMAP	可
ISYSUSER	可
ISYSUSERAUTHORITY	可
ISYSUSERMESSAGE	可
ISYSUSERTYPE	可
ISYSVIEW	可
ISYSWEBSERVICE	可

SYS.DUMMY テーブルと IQ_DUMMY テーブルの比較

DUMMY システム・テーブルは、ローを常に 1 つだけ持つテーブルとして提供されています。

これはデータベースから情報を抽出するのに役立ちます。次に、データベースから現在のユーザ ID と今日の日付を取り出す例を示します。

```
SELECT USER, today(*) FROM SYS.DUMMY
```

DUMMY テーブルを使用したクエリを実行するのは、Sybase IQ ではなく SQL Anywhere (カタログ・ストア) です。Sybase IQ データベース内で、ダミー・テーブルを作成できます。例を示します。

```
CREATE TABLE iq_dummy (dummy_col INT NOT NULL);
```

さらに、このテーブルを明示的に使用します。

```
SELECT NOW() FROM iq_dummy;
```

DUMMY システム・テーブル

カラム名	カラム型	カラム制約	テーブル制約
dummy_col	INTEGER	NOT NULL	

DUMMY テーブルは、常に 1 つだけのローを持つ、読み込み専用のテーブルとして提供されています。これはデータベースから情報を抽出するのに役立ちます。次に、データベースから現在のユーザ ID と今日の日付を取り出す例を示します。

```
SELECT USER, today(*) FROM IQ.DUMMY;
```

```
SELECT USER, today(*);
```

dummy_col – このカラムは使用されません。テーブルはカラムなしでは作成できないので、このカラムが存在します。

IQ.DUMMY テーブルからの読み取りコストは、同様のユーザ作成テーブルからの読み取りコストよりも低いコストです。これは、IQ.DUMMY のテーブル・ページにはラッチがないためです。

実行プランは IQ.DUMMY テーブルのスキャンによって構築されるわけではありません。代わりに、IQ.DUMMY への参照がロー・コンストラクタ・アルゴリズムに置き換えられ、これがテーブル参照を仮想化します。これにより、IQ.DUMMY の使用に伴う競合を排除できます。ただし、DUMMY は、テーブル名か相関名またはその両方として、短いプラン、長いプラン、グラフィカルなプランに引き続き表示されます。

システム・ビュー

システム・テーブルの内容を見るには、システム・ビューを使用します。

システム・テーブル内の情報を読みやすい形で表示するために、定義済みシステム・ビューが多数用意されています。

システム・ビューの記述ではその定義も示します。複雑なものもありますが、ビューを使用するには理解する必要はありません。

統合ビュー

統合ビューにはユーザがよく要求する形式でデータが表示されます。

たとえば、統合ビューにはよく一般に必要とされるジョインが用意されています。統合ビューはシステム・ビューとは違って、システム・テーブルからの生データをそのまま表示するビューではありません。たとえば、システム・ビューのカラ

ムの多くは意味のない ID の値ですが、統合ビューでは、それは意味のある名前になっています。

SYSCATALOG や SYSINDEXES などの統合ビューは、Sybase IQ と SQL Anywhere の両方に共通です。 これらの統合ビューおよびその他の統合ビューの定義については、『SQL Anywhere サーバー - SQL リファレンス』の「ビュー」>「統合ビュー」を参照してください。

互換ビュー

互換ビューは廃止されたビューですが、旧バージョンの SQL Anywhere と Sybase IQ との互換性を維持するために提供されています。

できるだけ、互換ビューの代わりにシステム・ビューと統合ビューを使用するようにしてください。互換ビューは Sybase IQ の将来のバージョンから削除される可能性があります。

互換ビューの詳細については、『SQL Anywhere サーバー - SQL リファレンス』の「ビュー」>「互換ビュー」を参照してください。

ASE T-SQL 互換ビュー

Sybase IQ には DBO という特殊なユーザが所有するビューのセットがあります。これは Adaptive Server Enterprise のシステム・テーブルとビューに対応しています。

参照：

- Transact-SQL 互換ビュー (698 ページ)

SYSARTICLE システム・ビュー

SYSARTICLE システムの各ローは、パブリケーション内のアーティクルを示します。このビューの基本となるシステム・テーブルは ISYSARTICLE です。

カラム名	データ型	説明
publication_id	UNSIGNED INT	このアーティクルが含まれるパブリケーション。
table_id	UNSIGNED INT	各アーティクルは、単一のテーブルのカラムとローから構成されます。このカラムには、このテーブルの ID が含まれます。
where_expr	LONG VARCHAR	WHERE 句で定義されたローのサブセットを含むアーティクルの場合、このカラムに探索条件が含まれます。

カラム名	データ型	説明
subscribe_by_expr	LONG VARCHAR	SUBSCRIBE BY 式で定義されたローのサブセットを含むアーティクルの場合、このカラムに式が含まれます。
query	CHAR(1)	データベース・サーバにアーティクル・タイプ情報を示します。
alias	VARCHAR(256)	アーティクルのエイリアス。
schema_change_active	BIT	テーブルとパブリケーションが同期スキーマ変更の一部である場合、1 です。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (publication_id, table_id)
```

```
FOREIGN KEY (publication_id) references SYS.ISYSPUBLICATION
(publication_id)
```

```
FOREIGN KEY (table_id) references SYS.ISYSTAB (table_id)
```

SYSARTICLECOL システム・ビュー

SYSARTICLECOL システム・ビューの各ローは、アーティクル内のカラムを識別します。このビューの基本となるシステム・テーブルは ISYSARTICLECOL です。

カラム名	データ型	説明
publication_id	UNSIGNED INT	カラムが含まれるパブリケーションのユニークな識別子。
table_id	UNSIGNED INT	カラムが属するテーブル。
column_id	UNSIGNED INT	SYSTABCOL システム・ビューのカラム識別子。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (publication_id, table_id, column_id)
```

```
FOREIGN KEY (publication_id, table_id) references SYS.ISYSARTICLE
(publication_id, table_id)
```

```
FOREIGN KEY (table_id, column_id) references SYS.ISYSTABCOL
(table_id, column_id)
```

SYSARTICLECOLS 統合ビュー

SYSARTICLECOLS ビューの各ローは、アーティクル内のカラムを識別します。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSARTICLECOLS"
  as select p.publication_name,t.table_name,c.column_name
    from SYS.ISYSARTICLECOL as ac
      join SYS.ISYSPUBLICATION as p on p.publication_id =
ac.publication_id
      join SYS.ISYSTAB as t on t.table_id = ac.table_id
      join SYS.ISYSTABCOL as c on c.table_id = ac.table_id
      and c.column_id = ac.column_id
```

SYSARTICLES 統合ビュー

SYSARTICLES ビューの各ローは、パブリケーション内のアーティクルを示します。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSARTICLES"
  as select u1.user_name as publication_owner,p.publication_name,
    u2.user_name as table_owner,t.table_name,
    a.where_expr,a.subscribe_by_expr,a.alias
    from SYS.ISYSARTICLE as a
      join SYS.ISYSPUBLICATION as p on(a.publication_id =
p.publication_id)
      join SYS.ISYSTAB as t on(a.table_id = t.table_id)
      join SYS.ISYSUSER as u1 on(p.creator = u1.user_id)
      join SYS.ISYSUSER as u2 on(t.creator = u2.user_id)
```

SYSCAPABILITIES 統合ビュー

SYSCAPABILITIES ビューの各ローは、リモート・データベース・サーバの機能のステータスを示します。このビューは、ISYSCAPABILITY と ISYSCAPABILITYNAME の各システム・テーブルからデータを取得します。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSCAPABILITIES"
  as select
ISYSCAPABILITY.capid,ISYSCAPABILITY.srvid,property('RemoteCapabilit
y',ISYSCAPABILITY.capid) as capname,ISYSCAPABILITY.capvalue
    from SYS.ISYSCAPABILITY
```

SYSCAPABILITY システム・ビュー

SYSCAPABILITY システム・ビューの各ローは、リモート・データベース・サーバの機能のステータスを示します。このビューの基本となるシステム・テーブルは ISYSCAPABILITY です。

カラム名	データ型	説明
capid	INTEGER	SYSCAPABILITYNAME システム・ビューに表示されている機能 ID。
srvid	UNSIGNED INT	SYSSERVER システム・ビューに表示されている、機能が適用されるサーバ。
capvalue	CHAR(128)	機能の値。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (capid, srvid)
```

```
FOREIGN KEY (srvid) references SYS.ISYSSERVER (srvid)
```

SYSCAPABILITYNAME システム・ビュー

SYSCAPABILITYNAME システム・ビューの各ローには、SYSCAPABILITY システム・ビュー内の各機能 ID の名前があります。

カラム名	データ型	説明
capid	INTEGER	機能をユニークに識別する番号。
capname	VARCHAR(32000)	機能の名前。

備考

SYSCAPABILITYNAME システム・ビューは、sa_rowgenerator と次のサーバ・プロパティの組み合わせを使用して定義されます。

RemoteCapability
MaxRemoteCapability

SYSCATALOG 統合ビュー

SYSCATALOG ビューの各ローは、システム・テーブルを示します。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSCATALOG"( creator,
  tname,dbspacename,tabletype,ncols,primary_key,"check",
  remarks )
```

```
  as select u.user_name,tab.table_name,dbs.dbspace_name,
    if tab.table_type_str = 'BASE' then 'TABLE' else
tab.table_type_str endif,
    (select count() from SYS.ISYSTABCOL
      where ISYSTABCOL.table_id = tab.table_id),
    if ix.index_id is null then 'N' else 'Y' endif,
    null,
    rmk.remarks
    from SYS.SYSTAB as tab
     join SYS.ISYSDBSPACE as dbs on(tab.dbspace_id = dbs.dbspace_id)
      join SYS.ISYSUSER as u on u.user_id = tab.creator
     left outer join SYS.ISYSIDX as ix on(tab.table_id = ix.table_id
and ix.index_id = 0)
      left outer join SYS.ISYSREMARK as rmk on(tab.object_id =
rmk.object_id)
```

SYSCHECK システム・ビュー

SYSCHECK システム・ビューの各ローは、テーブル内の名前付き検査制約を定義します。このビューの基本となるシステム・テーブルは ISYSCHECK です。

カラム名	データ型	説明
check_id	UNSIGNED INT	データベースの制約をユニークに識別する番号。
check_defn	LONG VARCHAR	CHECK 式。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (check_id)
```

```
FOREIGN KEY (check_id) references SYS.ISYSCONSTRAINT (constraint_id)
```

SYSCOLAUTH 統合ビュー

SYSCOLAUTH ビューの各ローは、カラムに付与されている一連の権限 (UPDATE、SELECT、または REFERENCES) を示します。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW
"SYS"."SYSCOLAUTH"( grantor,grantee,creator,tname,colname,
  privilege_type,is_grantable )
  as select u1.user_name,u2.user_name,u3.user_name,tab.table_name,
    col.column_name,cp.privilege_type,cp.is_grantable
    from SYS.ISYSCOLPERM as cp
      join SYS.ISYSUSER as u1 on u1.user_id = cp.grantor
      join SYS.ISYSUSER as u2 on u2.user_id = cp.grantee
      join SYS.ISYSTAB as tab on tab.table_id = cp.table_id
      join SYS.ISYSUSER as u3 on u3.user_id = tab.creator
      join SYS.ISYSTABCOL as col on col.table_id = cp.table_id
      and col.column_id = cp.column_id
```

SYSCOLLATION 互換ビュー (旧式)

SYSCOLLATION 互換ビューには、データベースの照合順情報が格納されます。組み込み関数経由で取得でき、カタログには保存されません。このビューの定義を次に示します。

```
ALTER VIEW "SYS"."SYSCOLLATION"
  as select 1 as collation_id,
    DB_PROPERTY('Collation') as collation_label,
    DB_EXTENDED_PROPERTY('Collation','Description') as
collation_name,
    cast(DB_EXTENDED_PROPERTY('Collation','LegacyData') as
binary(1280)) as collation_order
```

SYSCOLLATIONMAPPINGS 互換ビュー (旧式)

SYSCOLLATIONMAPPINGS 互換ビューにはデータベース照合マッピングを持つローが 1 つだけ含まれています。組み込み関数経由で取得でき、カタログには保存されません。このビューの定義を次に示します。

```
ALTER VIEW "SYS"."SYSCOLLATIONMAPPINGS"
  as select DB_PROPERTY('Collation') as collation_label,
    DB_EXTENDED_PROPERTY('Collation','Description') as
collation_name,
    DB_PROPERTY('Charset') as cs_label,
    DB_EXTENDED_PROPERTY('Collation','ASESensitiveSortOrder') as
so_case_label,
    DB_EXTENDED_PROPERTY('Collation','ASEInsensitiveSortOrder') as
so_caseless_label,
    DB_EXTENDED_PROPERTY('Charset','java') as jdk_label
```

SYSCOLPERM システム・ビュー

GRANT 文を使うと、テーブルの各カラムに UPDATE、SELECT、または REFERENCES パーミッションを与えることができます。UPDATE、SELECT、または REFERENCES パーミッションを持つ各カラムは、SYSCOLPERM システム・ビューの各ローに記録されます。このビューの基本となるシステム・テーブルは ISYSCOLPERM です。

カラム名	データ型	説明
table_id	UNSIGNED INT	カラムが属するテーブルのテーブル番号。
grantee	UNSIGNED INT	カラムにパーミッションを与えられたユーザ ID のユーザ番号。パーミッションの grantee が特別な PUBLIC ユーザ ID のユーザ番号の場合、パーミッションはすべてのユーザ ID に与えられます。

カラム名	データ型	説明
grantor	UNSIGNED INT	パーミッションを付与するユーザ ID のユーザ番号。
column_id	UNSIGNED INT	このカラム番号と table_id を使って、パーミッションを付与されたカラムを識別します。
privilege_type	SMALLINT	このカラムの数値は、カラム・パーミッションの種類 (16=REFERENCES、1=SELECT、または 8=UPDATE) を示します。
is_grantable	CHAR(1)	カラムのパーミッションが GRANT OPTION を使って付与されているかどうかを示します。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (table_id, grantee, grantor, column_id, privilege_type)
```

```
FOREIGN KEY (table_id, column_id) references SYS.ISYSTABCOL
(table_id, column_id)
```

```
FOREIGN KEY (grantor) references SYS.ISYSUSER (user_id)
```

```
FOREIGN KEY (grantee) references SYS.ISYSUSER (user_id)
```

SYSCOLSTAT システム・ビュー

SYSCOLSTAT システム・ビューには、オプティマイザが使用するヒストグラムなどのカラムの統計情報が含まれます。このビューの内容を取り出すには、sa_get_histogram ストアド・プロシージャまたはヒストグラム・ユーティリティを使用するのが最適です。このビューの基本となるシステム・テーブルは ISYSCOLSTAT です。

カラム名	データ型	説明
table_id	UNSIGNED INT	このカラムが属するテーブルまたはマテリアライズド・ビューをユニークに識別する番号。
column_id	UNSIGNED INT	table_id とともに使用してカラムをユニークに識別する番号。
format_id	SMALLINT	システムでのみ使用。
update_time	TIMESTAMP	このカラムの統計情報が最後に更新された時刻。
density	FLOAT	カラムの単一の値の平均による選択性の推定。ローに格納されている大きい単一値の選択性は考慮されません。
max_steps	SMALLINT	システムでのみ使用。
actual_steps	SMALLINT	システムでのみ使用。

カラム名	データ型	説明
step_values	LONG BINARY	システムでのみ使用。
frequencies	LONG BINARY	システムでのみ使用。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (table_id, column_id)
```

```
FOREIGN KEY (table_id, column_id) references SYS.ISYSTABCOL
(table_id, column_id)
```

SYSCOLSTATS 統合ビュー

SYSCOLSTATS ビューには、オプティマイザによって使用されるカラム統計が、ヒストグラムとして格納されます。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSCOLSTATS"
  as select u.user_name,t.table_name,c.column_name,
    s.format_id,s.update_time,s.density,s.max_steps,
    s.actual_steps,s.step_values,s.frequencies
    from SYS.ISYSCOLSTAT as s
      join SYS.ISYSTABCOL as c on(s.table_id = c.table_id
      and s.column_id = c.column_id)
      join SYS.ISYSTAB as t on(t.table_id = c.table_id)
      join SYS.ISYSUSER as u on(u.user_id = t.creator)
```

SYSCOLUMN 互換ビュー (旧式)

SYSCOLUMN ビューは、SYSCOLUMN システム・テーブルを提供していた古いバージョンのソフトウェアとの互換性を保つために用意されています。ただし、以前の SYSCOLUMN テーブルは、SYSTABCOL システム・ビューに対応する ISYSTABCOL システム・テーブルで置換されたため、ISYSTABCOL の使用をおすすめします。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSCOLUMN"
  as select b.table_id,
    b.column_id,
    if c.sequence is null then 'N' else 'Y' endif as pkey,
    b.domain_id,
    b.nulls,
    b.width,
    b.scale,
    b.object_id,
```

```
    b.max_identity,
    b.column_name,
    r.remarks,
    b."default",
    b.user_type,
    b.column_type
    from SYS.SYSTABCOL as b
      left outer join SYS.ISYSREMARK as r on(b.object_id =
r.object_id)
     left outer join SYS.ISYSIDXCOL as c on(b.table_id = c.table_id
and b.column_id = c.column_id and c.index_id = 0)
```

SYSCOLUMNS 統合ビュー

SYSCOLUMNS ビューの各ローは、カタログ内の各テーブルとビューのカラム 1 つを示します。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW
"SYS"."SYSCOLUMNS"( creator,cname,tname,coltype,nulls,length,
  syslength,in_primary_key,colno,default_value,
  column_kind,remarks )
  as select
u.user_name,col.column_name,tab.table_name,dom.domain_name,
   col.nulls,col.width,col.scale,if ixcol.sequence is null then 'N'
else 'Y' endif,col.column_id,
    col."default",col.column_type,rmk.remarks
    from SYS.SYSTABCOL as col
      left outer join SYS.ISYSIDXCOL as ixcol on(col.table_id =
ixcol.table_id and col.column_id = ixcol.column_id and
ixcol.index_id = 0)
      join SYS.ISYSTAB as tab on(tab.table_id = col.table_id)
      join SYS.ISYSDOMAIN as dom on(dom.domain_id = col.domain_id)
      join SYS.ISYSUSER as u on u.user_id = tab.creator
      left outer join SYS.ISYSREMARK as rmk on(col.object_id =
rmk.object_id)
```

SYSCOLUMNS ASE 互換ビュー

このビューは DBO というユーザが所有しています。syscolumns の各ローは、あらゆるテーブルとビュー内の各カラムまたはプロシージャ内の各パラメータに対応しています。

参照：

- 各 Adaptive Server Enterprise データベース内のテーブル (699 ページ)
- SYSCOMMENTS ASE 互換ビュー (615 ページ)
- SYSINDEXES ASE 互換ビュー (632 ページ)
- SYSIQOBJECTS ASE 互換ビュー (644 ページ)

- SYSIQVINDEX ASE 互換ビュー (647 ページ)
- SYSOBJECTS ASE 互換ビュー (653 ページ)
- SYSTYPES ASE 互換ビュー (690 ページ)
- SYSUSERS ASE 互換ビュー (696 ページ)

SYSCOMMENTS ASE 互換ビュー

syscomments テーブルには、ビュー、ルール、デフォルト、トリガ、テーブル制約、パーティション、プロシージャ、計算カラム、関数ベース・インデックス・キー、その他の形式のコンパイル済みオブジェクトのそれぞれに対するエントリが含まれています。

このビューは、DBO が所有します。

テキスト・カラムには元の定義文が入っています。テキスト・カラムが 255 バイトを超えるとエントリは複数ローにまたがります。それぞれのオブジェクトは最大 65,025 ローまで使用できます。

参照：

- 各 Adaptive Server Enterprise データベース内のテーブル (699 ページ)
- SYSCOLUMNS ASE 互換ビュー (614 ページ)
- SYSINDEXES ASE 互換ビュー (632 ページ)
- SYSIQOBJECTS ASE 互換ビュー (644 ページ)
- SYSIQVINDEX ASE 互換ビュー (647 ページ)
- SYSOBJECTS ASE 互換ビュー (653 ページ)
- SYSTYPES ASE 互換ビュー (690 ページ)
- SYSUSERS ASE 互換ビュー (696 ページ)

SYSCONSTRAINT システム・ビュー

SYSCONSTRAINT システム・ビューの各ローは、データベース内の名前付き制約を示します。このビューの基本となるシステム・テーブルは ISYSCONSTRAINT です。

カラム名	データ型	説明
constraint_id	UNSIGNED INT	制約のユニークな ID。

カラム名	データ型	説明
constraint_type	CHAR(1)	制約の型。 • **C** – カラム検査制約 • **T** – テーブル制約 • **P** – プライマリ・キー • **F** – 外部キー • **U** – 一意性制約
ref_object_id	UNSIGNED BIGINT	制約の適用対象となるカラム、テーブル、インデックスのオブジェクト ID。
table_object_id	UNSIGNED BIGINT	スクリプトの適用対象となるテーブルのオブジェクト ID。
constraint_name	CHAR(128)	制約名。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (constraint_id)
FOREIGN KEY (ref_object_id) references SYS.ISYSOBJECT (object_id)
FOREIGN KEY (table_object_id) references SYS.ISYSOBJECT (object_id)
UNIQUE Constraint (table_object_id, constraint_name)
```

SYSDBFILE システム・ビュー

SYSDBFILE システム・ビューの各ローは、DB 領域ファイルを示します。このビューの基本となるシステム・テーブルは ISYSDBFILE です。

カラム名	データ型	説明
dbfile_id	SMALLINT	内部でのみ使用。
dbspace_id	SMALLINT	データベースの各 DB 領域ファイルには、ユニークな番号が割り当てられています。システム DB 領域には、すべてのシステム・オブジェクトが含まれ、0 の dbspace_id があります。
dbfile_name	CHAR(128)	DB 領域のファイル名。システムと TEMPORARY 以外の DB 領域では、ファイル名は次の文を使用して変更できます。 `ALTER DBSPACE dbspace RENAME 'new-filename';`
file_name	LONG VARCHAR	DB 領域のユニークなファイル名。これは CREATE TABLE コマンド中で使われます。

カラム名	データ型	説明
lob_map	LONG VARBIT	内部でのみ使用。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (dbfile_id)
```

```
FOREIGN KEY (dbspace_id) references SYS.ISYSDBSPACE (dbspace_id)
```

```
UNIQUE Index (file_name)
```

SYSDBSPACE システム・ビュー

SYSDBSPACE システム・ビューの各ローは、DB 領域ファイルを示します。このビューの基本となるシステム・テーブルは ISYSDBSPACE です。

カラム名	データ型	説明
dbspace_id	SMALLINT	DB 領域を識別するユニークな番号。システム DB 領域には、すべてのシステム・オブジェクトが含まれ、0 の dbspace_id があります。
object_id	UNSIGNED BIGINT	DB 領域のファイル名。システム DB 領域の場合、データベースを作成したときに、値はデータベース・ファイルの名前です。また、この値は参照情報であり、変更できません。その他の DB 領域では、ファイル名は次の文を使用して変更できます。 `ALTER DBSPACE dbspace RENAME 'new-filename';`
dbspace_name	CHAR(128)	DB 領域のユニークなファイル名。これは CREATE TABLE コマンド中で使われます。
store_type	TINYINT	内部でのみ使用。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (dbspace_id)
```

```
FOREIGN KEY (object_id) references SYS.ISYSOBJECT (object_id) MATCH
UNIQUE FULL
```

SYSDBSPACEPERM システム・ビュー

SYSDBSPACEPERM システム・ビューの各ローは、DB 領域ファイルのパーミッションを示します。このビューの元となるシステム・テーブルは ISYSDBSPACEPERM です。

SYSDBSPACEPERM ビューは SQL Anywhere のシステム・ビューです。詳細については、『SQL Anywhere サーバー - SQL リファレンス』の「ビュー」>「システムビュー」>「SYSDBFILE システムビュー」を参照してください。

注意： 参照先は SQL Anywhere のマニュアルです。

SYSDEPENDENCY システム・ビュー

SYSDEPENDENCY システム・ビューの各ローは、2 つのデータベース・オブジェクト間の依存性を示します。このビューの基本となるシステム・テーブルは ISYSDEPENDENCY です。

あるオブジェクトが定義内の別のオブジェクトを参照する場合、この 2 つのデータベース・オプション間に依存性が存在します。たとえば、ビューのクエリ指定がテーブルを参照する場合、ビューはテーブルに依存していると呼びます。データベース・サーバが、テーブル、ビュー、マテリアライズド・ビュー、カラムの依存性を追跡します。

カラム名	データ型	説明
ref_object_id	UNSIGNED BIGINT	参照オブジェクトのオブジェクト ID。
dep_object_id	UNSIGNED BIGINT	参照オブジェクトの ID。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (ref_object_id, dep_object_id)
FOREIGN KEY (ref_object_id) references SYS.ISYSOBJECT (object_id)
FOREIGN KEY (dep_object_id) references SYS.ISYSOBJECT (object_id)
```

SYSDOMAIN システム・ビュー

SYSDOMAIN システム・ビューは、組み込みデータ型に関する情報 (ドメインとも呼びます) を記録します。通常の操作ではビューのコンテンツは変化しません。このビューの基本となるシステム・テーブルは ISYSDOMAIN です。

カラム名	データ型	説明
domain_id	SMALLINT	各データ型に割り当てられたユニークな番号。この番号は変更できません。

カラム名	データ型	説明
domain_name	CHAR(128)	CREATE TABLE コマンドの中に通常見られるデータ型の名前 (CHAR または INTEGER など)。
type_id	SMALLINT	ODBC データ型。この値は、Transact-SQL 互換の dbo.SYSTYPES テーブルにある data_type の値に対応します。
"precision"	SMALLINT	このデータ型を使って格納できる有効桁数。数値でないデータ型に対応するカラム値は NULL です。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (domain_id)
```

SYSEVENT システム・ビュー

SYSEVENT システム・ビューの各ローは、CREATE EVENT で作成されたイベントを示します。このビューの基本となるシステム・テーブルは ISYSEVENT です。

カラム名	データ型	説明
event_id	UNSIGNED INT	各イベントに割り当てられたユニークな番号。
object_id	UNSIGNED BIGINT	データベースのイベントをユニークに識別するイベントの内部 ID。
creator	UNSIGNED INT	イベントの所有者のユーザ番号。ユーザ名は SYSUSER システム・ビューで確認できます。
event_name	VARCHAR(128)	イベント名。
enabled	CHAR(1)	イベントが起動できるかどうかを示します。
location	CHAR(1)	イベントを起動するロケーション。 C = 統合 R = リモート A = すべて
event_type_id	UNSIGNED INT	システム・イベントの場合、イベント型は SYSEVENTTYPE システム・ビューにリストされます。
action	LONG VARCHAR	イベント・ハンドラ定義。難読化された値は、隠されたイベントを示します。
external_action	LONG VARCHAR	システムでのみ使用。
condition	LONG VARCHAR	イベント・ハンドラの起動を制御するのに使用される条件。

カラム名	データ型	説明
remarks	LONG VARCHAR	イベントの注釈。このカラムは ISYSREMARK に由来します。
source	LONG VARCHAR	イベントの元のソース。このカラムは ISYSSOURCE に由来します。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (event_id)
```

```
FOREIGN KEY (creator) references SYS.ISYSUSER (user_id)
```

```
FOREIGN KEY (object_id) references SYS.ISYSOBJECT (object_id) MATCH
UNIQUE FULL
```

```
UNIQUE Index (event_name)
```

SYSEVENTTYPE システム・ビュー

SYSEVENTTYPE システム・ビューは、CREATE EVENT で参照できるシステムのイベント・タイプを定義します。

SYSEVENTTYPE ビューは SQL Anywhere のシステム・ビューです。詳細については、『SQL Anywhere サーバー - SQL リファレンス』の「ビュー」>「システムビュー」>「SYSEVENT システムビュー」を参照してください。

注意： 参照先は SQL Anywhere のマニュアルです。

SYSEXTERNENV システム・ビュー

C/C++ で記述された Embedded SQL と ODBC アプリケーション、Java、Perl、PHP、または Microsoft .NET Framework Common Language Runtime (CLR) に基づく C# や Visual Basic などの言語で記述されたアプリケーションなど、多くの外部ランタイム環境がサポートされています。

SYSEXTERNENV システム・ビューの各ローは、各外部環境を識別して起動するために必要な情報を示します。このビューの基本となるシステム・テーブルは ISYSEXTERNENV です。

カラム名	データ型	説明
object_id	unsigned bigint	外部環境のユニークな識別子。
Name	char(128)	外部環境または言語の名前。
scope	char(1)	外部環境が接続ごとに 1 つ (C) 起動されるか、データベースごとに 1 つ (D) 起動されるかを識別します。

カラム名	データ型	説明
support_result_sets	char(1)	ユーザに結果セットを返す可能性のある外部環境を識別します。
location	long varchar	環境のメイン実行プログラムを検出できるロケーションを識別します。
options	long varchar	外部環境を起動するためにコマンド・ラインで必要なオプションを識別します。
user_id	unsigned int	内部でのみ使用。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (object_id)
```

```
FOREIGN KEY (object_id) references SYS.ISYSOBJECT (object_id) MATCH
UNIQUE FULL
```

```
FOREIGN KEY (user_id) references SYS.ISYSUSER (user_id)
```

```
UNIQUE Index (name)
```

SYSEXTERNENVOBJECT システム・ビュー

C/C++ で記述された Embedded SQL と ODBC アプリケーション、Java、Perl、PHP、または Microsoft .NET Framework Common Language Runtime (CLR) に基づく C# や Visual Basic などの言語で記述されたアプリケーションなど、多くの外部ランタイム環境がサポートされています。

SYSEXTERNENVOBJECT システム・ビューの各ローは、インストールされた外部オブジェクトを示します。このビューの基本となるシステム・テーブルは ISYSEXTERNENVOBJECT です。

カラム名	データ型	説明
object_id	unsigned bigint	外部オブジェクトのユニークな識別子。
extenv_id	unsigned bigint	外部環境のユニークな識別子 (SYSEXTERNENV.object_id)。
owner	unsigned int	外部オブジェクトの作成者／所有者を識別します。
Name	long varchar	INSTALL 文に指定された外部オブジェクトの名前を識別します。
contents	long binary	外部オブジェクトの内容。
update_time	timestamp	オブジェクトが最後に変更 (またはインストール) された時刻を識別します。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (object_id)
```

```
FOREIGN KEY (object_id) references SYS.ISYSOBJECT (object_id) MATCH
UNIQUE FULL
```

```
FOREIGN KEY (extenv_id) references SYS.ISYSEXTERNENV (object_id)
```

```
FOREIGN KEY (owner) references SYS.ISYSUSER (user_id)
```

```
UNIQUE Index (name)
```

SYSEXTERNLOGIN システム・ビュー

SYSEXTERNLOGIN システム・ビューの各ローは、リモート・データ・アクセスの外部ログインを示します。このビューの基本となるシステム・テーブルは ISYSEXTERNLOGIN です。

注意： SYSEXTERNLOGINS システム・テーブルに含まれる前のカタログ・バージョン。このテーブル名は ISYSEXTERNLOGIN ('S' なし) に変更され、このビューの基本となるテーブルになります。

カラム名	データ型	説明
user_id	UNSIGNED INT	ローカル・データベースでのユーザ ID。
srvid	UNSIGNED INT	SYSSERVER システム・ビューにリストされているリモート・サーバ。
remote_login	VARCHAR(128)	このユーザのリモート・サーバのログイン名。
remote_password	VARBINARY(128)	このユーザのリモート・サーバのパスワード。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (user_id, srvid)
```

```
FOREIGN KEY (user_id) references SYS.ISYSUSER (user_id)
```

```
FOREIGN KEY (srvid) references SYS.ISYSSERVER (srvid)
```

SYSFILE 互換ビュー (旧式)

SYSFILE システム・ビューの各ローは、データベースの DB 領域を示します。各データベースは、1 つ以上の DB 領域で構成されます。各 DB 領域は、1 つのオペレーティング・システム・ファイルに対応します。

メイン・データベース・ファイル、テンポラリ・ファイル、トランザクション・ログ・ファイル、トランザクション・ログ・ミラー・ファイルの DB 領域が自動

的に作成されます。トランザクション・ログとトランザクション・ログ・ミラーの DB 領域に関する情報は、SYSFILE システム・ビューには表示されません。

```
ALTER VIEW "SYS"."SYSFILE"
  as select b.dbfile_id as file_id,
    if b.dbspace_id = 0 and b.dbfile_id = 0 then
      db_property('File')
    else
      if b.dbspace_id = 15 and b.dbfile_id = 15 then
        db_property('TempFileName')
      else
        b.file_name
      endif
    endif as file_name,
    a.dbspace_name,
    a.store_type,
    b.lob_map,
    b.dbspace_id
    from SYS.ISYSDBSPACE as a
      join SYS.ISYSDBFILE as b on(a.dbspace_id = b.dbspace_id)
```

SYSFKCOL 互換ビュー (旧式)

SYSFKCOL の各ローは、外部テーブルの外部カラムと、プライマリ・テーブルのプライマリ・カラムの関係を示します。このビューは使用されなくなりました。代わりに SYSIDX と SYSIDXCOL のシステム・ビューを使用してください。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSFKCOL"
  as select a.table_id as foreign_table_id,
    a.index_id as foreign_key_id,
    a.column_id as foreign_column_id,
    a.primary_column_id
    from SYS.ISYSIDXCOL as a
      ,SYS.ISYSIDX as b
    where a.table_id = b.table_id
    and a.index_id = b.index_id
    and b.index_category = 2
```

SYSFKEY システム・ビュー

SYSFKEY システム・ビューの各ローは、システム内の外部キー制約を示します。このビューの基本となるシステム・テーブルは ISYSFKEY です。

カラム名	データ型	説明
foreign_table_id	UNSIGNED INT	外部テーブルのテーブル番号。

カラム名	データ型	説明
foreign_index_id	UNSIGNED INT	外部キーのインデックス番号。
primary_table_id	UNSIGNED INT	プライマリ・テーブルのテーブル番号。
primary_index_id	UNSIGNED INT	プライマリ・キーのインデックス番号。
match_type	TINYINT	制約と一致する型。一致する型には次の型があります。 • **0 –** デフォルトのマッチングを使用 • **1 –** SIMPLE • **2 –** FULL • **129 –** SIMPLE UNIQUE • **130 –** FULL UNIQUE
check_on_commit	CHAR(1)	COMMIT で外部キーが有効であるかどうかが確認されるまで、INSERT と UPDATE 文を待たせるかどうかを示します。
nulls	CHAR(1)	外部キーのカラムが NULL 値を許容するかどうかを示します。この設定は外部キーのカラム中の nulls 設定とは独立していることに注意してください。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (foreign_table_id, foreign_index_id)
```

```
FOREIGN KEY (foreign_table_id, foreign_index_id) references
SYS.ISYSIDX (table_id, index_id)
```

```
FOREIGN KEY (primary_table_id, primary_index_id) references
SYS.ISYSIDX (table_id, index_id)
```

SYSFOREIGNKEY 互換ビュー (旧式)

SYSFOREIGNKEY ビューは、SYSFOREIGNKEY システム・テーブルを提供していた古いバージョンのソフトウェアとの互換性を保つために用意されています。ただし、以前の SYSFOREIGNKEY システム・テーブルは、SYSFKEY システム・ビューに対応する ISYSFKEY システム・テーブルで置換されたため、ISYSFKEY の使用をおすすめします。

外部キーは、外部テーブルとプライマリ・テーブルの 2 つのテーブル間の関係です。外部キーは、SYSFOREIGNKEY の 1 つのローと SYSFKCOL の 1 つまたは複数のローで定義されます。SYSFOREIGNKEY には、外部キーに関する一般的な情

報が入っています。SYSFKCOL は外部キーのカラムを識別して、外部キーの各カラムとプライマリ・テーブルの中のプライマリ・キーのカラムを関連付けます。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSFOREIGNKEY"
  as select b.foreign_table_id,
    b.foreign_index_id as foreign_key_id,
    a.object_id,
    b.primary_table_id,
    p.root,
    b.check_on_commit,
    b.nulls,
    a.index_name as role,
    r.remarks,
    b.primary_index_id,
    a.not_enforced as fk_not_enforced,
    10 as hash_limit
    from(SYS.ISYSIDX as a left outer join SYS.ISYSPHYSIDX as p
on(a.table_id = p.table_id and a.phys_index_id = p.phys_index_id))
      left outer join SYS.ISYSREMARK as r on(a.object_id =
r.object_id)
      ,SYS.ISYSFKEY as b
    where a.table_id = b.foreign_table_id
    and a.index_id = b.foreign_index_id
```

SYSFOREIGNKEYS 統合ビュー

SYSFOREIGNKEYS ビューの各ローは、カタログ内の各テーブルの外部キー 1 つを示します。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSFOREIGNKEYS"( foreign_creator,
  foreign_tname,
  primary_creator,primary_tname,role,columns )
  as select fk_up.user_name,fk_tab.table_name,pk_up.user_name,
    pk_tab.table_name,ix.index_name,
    (select list(string(fk_col.column_name,' IS ',
      pk_col.column_name)
      order by fkc.table_id,fkc.index_id,fkc."sequence")
      from SYS.ISYSIDXCOL as fkc
        join SYS.ISYSTABCOL as fk_col on(
        fkc.table_id = fk_col.table_id
        and fkc.column_id = fk_col.column_id)
        ,SYS.ISYSTABCOL as pk_col
      where fkc.table_id = fk.foreign_table_id
      and fkc.index_id = fk.foreign_index_id
      and pk_col.table_id = fk.primary_table_id
      and pk_col.column_id = fkc.primary_column_id)
```

```
    from SYS.ISYSFKEY as fk
      join SYS.ISYSTAB as fk_tab on fk_tab.table_id =
fk.foreign_table_id
      join SYS.ISYSUSER as fk_up on fk_up.user_id = fk_tab.creator
      join SYS.ISYSTAB as pk_tab on pk_tab.table_id =
fk.primary_table_id
      join SYS.ISYSUSER as pk_up on pk_up.user_id = pk_tab.creator
     join SYS.ISYSIDX as ix on ix.table_id = fk.foreign_table_id and
ix.index_id = fk.foreign_index_id
```

SYSGROUP システム・ビュー

SYSGROUP システム・ビューには、各グループのメンバごとに 1 つのローがあります。このビューは、グループとメンバの多対多の関係を示します。グループには複数のメンバがあり、あるユーザは複数のグループのメンバになれます。このビューの基本となるシステム・テーブルは ISYSGROUP です。

カラム名	データ型	説明
group_id	UNSIGNED INT	グループのユーザ番号。
group_member	UNSIGNED INT	メンバのユーザ番号。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (group_id, group_member)
```

```
FOREIGN KEY group_id (group_id) references SYS.ISYSUSER (user_id)
```

```
FOREIGN KEY group_member (group_member) references SYS.ISYSUSER
(user_id)
```

SYSGROUPS 統合ビュー

SYSGROUPS ビューには、各グループのメンバごとに 1 つのローがあります。このビューは、グループとメンバの多対多の関係を示します。グループには複数のメンバがあり、あるユーザは複数のグループのメンバになれます。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSGROUPS"( group_name,
  member_name )
  as select g.user_name,u.user_name
    from SYS.ISYSGROUP,SYS.ISYSUSER as g,SYS.ISYSUSER as u
   where ISYSGROUP.group_id = g.user_id and ISYSGROUP.group_member =
u.user_id
```

SYSHISTORY システム・ビュー

SYSHISTORY システム・ビューの各ローは、データベースの開始、データベースの調整など、データベースに対するシステム操作を記録します。このビューの基本となるシステム・テーブルは ISYSHISTORY です。

カラム名	データ型	説明
operation	CHAR(128)	データベース・ファイルに対して実行されるオペレーションのタイプ。operation には次のいずれかの値を指定します。 • **INIT** – データベースがいつ作成されたかに関する情報。 • **UPGRADE** – データベースがいつアップグレードされたかに関する情報。 • **START** – 特定のオペレーティング・システム上で特定のバージョンのデータベース・サーバを使用してデータベースがいつ開始されたかに関する情報。 • **LAST_START** – データベース・サーバが最後に開始された時刻に関する情報。LAST_START オペレーションは、LAST_START ローに現在格納されている値とは異なるバージョンのデータベース・サーバか異なるオペレーティング・システム、またはその両方でデータベースが開始されたときに START オペレーションに変換されます。 • **DTT** – DB 領域で実行される 2 番目から最後のディスク転送時間 (DTT: Disk Transfer Time) の調整操作に関する情報。つまり、ALTER DATABASE CALIBRATE 文または ALTER DATABASE RESTORE DEFAULT CALIBRATION 文のどちらかの実行について、2 番目から最後の実行に関する情報です。 • **LAST_DTT** – DB 領域で実行される最新の DTT 調整操作に関する情報。つまり、ALTER DATABASE CALIBRATE 文または ALTER DATABASE RESTORE DEFAULT CALIBRATION 文のどちらかの実行について、最新の実行に関する情報です。 • **LAST_BACKUP** – 最後のバックアップ (バックアップの日時を含みます)、バックアップの種類、バックアップするファイル、バックアップを実行したデータベース・サーバのバージョンに関する情報。

カラム名	データ型	説明
object_id	UNSIGNED INT	DTT と LAST_DTT 以外の操作については、このカラムの値は 0 になります。DTT と LAST_DTT の操作の場合、SYSDBSPACE システム・ビューで定義されている DB 領域の dbspace_id です。
sub_operation	CHAR(128)	DTT と LAST_DTT 以外の操作については、このカラムの値は、空の単一引用符 ('') のセットになります。DTT と LAST_DTT の操作の場合、このカラムには、DB 領域で実行されるサブ操作の種類も含まれます。次のような値があります。 • **DTT_SET –** DB 領域の調整が設定されます。 • **DTT_UNSET –** DB 領域のデフォルト設定が復元されます。
version	CHAR(128)	オペレーションの実行に使用されるデータベース・サーバのバージョンとビルド番号。
platform	CHAR(128)	オペレーションが実行されたオペレーティング・システム。
first_time	TIMESTAMP	特定のオペレーティング・システム上の特定のバージョンのソフトウェアでデータベースが最初に開始された日時。
last_time	TIMESTAMP	特定のオペレーティング・システム上の特定のバージョンのソフトウェアでデータベースが最後に開始された日時。
details	LONG VARCHAR	このカラムは、データベース・サーバの起動に使用されたコマンド・ライン・オプションやデータベースに対して有効になっている機能ビットなどの情報を格納します。この情報は、Sybase 製品の保守契約を結んでいるサポート・センタが使用します。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (operation, object_id, version, platform)
```

SYSIDX システム・ビュー

SYSIDX システム・ビューの各ローは、データベースの論理インデックスを定義します。このビューの基本となるシステム・テーブルは ISYSIDX です。

カラム名	データ型	説明
table_id	UNSIGNED INT	このインデックスが適用されるテーブルをユニークに識別します。

カラム名	データ型	説明
index_id	UNSIGNED INT	テーブル内のインデックスを識別するユニークな番号。
object_id	UNSIGNED BIGINT	データベースのインデックスをユニークに識別するインデックスの内部 ID。
phys_index_id	UNSIGNED INT	論理インデックスの実装に使用する基本となる物理インデックスを識別します。この値は、テンポラリ・テーブルまたはリモート・テーブル上のインデックスに対しては NULL となります。その他の場合、この値は SYSPHYSIDX システム・ビューの物理インデックスの object_id に対応します。
dbspace_id	SMALLINT	インデックスを含むファイルの ID。この値は SYSDBSPACE システム・ビューのエントリに対応します。
file_id	SMALLINT	廃止予定。このカラムは SYSVIEW に存在しますが、基本となるシステム・テーブル ISYSIDX には存在しません。このカラムの内容は dbspace_id と同じであり、互換性のために提供されています。今後は dbspace_id を使用してください。
index_category	TINYINT	インデックスの型。次のような値があります。 • **1** – プライマリ・キー • **2** – 外部キー • **3** – セカンダリ・インデックス (一意性制約を含む) • **4** – テキスト・インデックス
"unique"	TINYINT	インデックスがユニーク・インデックス (1) か、ユニークでないインデックス (4) か、一意性制約 (2) かを示します。ユニーク・インデックスでは、インデックス・カラムの 2 つのローは同じ値を持てません。
index_name	CHAR(128)	インデックス名。
not_enforced	CHAR(1)	システムでのみ使用。
file_id	SMALLINT	システムでのみ使用。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (table_id, index_id)
```

```
FOREIGN KEY (table_id) references SYS.ISYSTAB (table_id)
```

```
FOREIGN KEY (object_id) references SYS.ISYSOBJECT (object_id) MATCH
UNIQUE FULL
```

```
FOREIGN KEY (table_id, phys_index_id) references SYS.ISYSPHYSIDX
(table_id, phys_index_id)
```

```
UNIQUE Index (index_name, table_id, index_category)
```

SYSIDXCOL システム・ビュー

SYSIDXCOL システム・ビューの各ローは、SYSIDX システム・ビューで記述されているインデックスのカラム 1 つを示します。このビューの基本となるシステム・テーブルは ISYSIDXCOL です。

カラム名	データ型	説明
table_id	UNSIGNED INT	インデックスが適用されるテーブルを識別します。
index_id	UNSIGNED INT	カラムが適用されるインデックスを識別します。table_id と index_id は共同で SYSIDX システム・ビューにある 1 つのインデックスを識別します。
sequence	SMALLINT	インデックス内の各カラムには、0 から始まるユニークな番号が割り当てられます。この番号はインデックス内のカラムの相対的な重要度を示します。最も重要なカラムの sequence 番号は 0 になります。
column_id	UNSIGNED INT	インデックスを作成するテーブルのカラムを識別します。table_id と column_id は共同で、SYSCOLUMN システム・ビューで表される 1 つのカラムを識別します。
"order"	CHAR(1)	インデックス内のカラムが昇順 (A) か、降順 (D) かを示すテキスト・インデックスの場合、この値は NULL です。
primary_column_id	UNSIGNED INT	この外部キー・カラムに対応するプライマリ・キーの ID。非外部キーのカラムの場合、値は NULL です。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (table_id, index_id, column_id)
```

```
FOREIGN KEY (table_id, index_id) references SYS.ISYSIDX (table_id,
index_id)
```

```
FOREIGN KEY (table_id, column_id) references SYS.ISYSTABCOL
(table_id, column_id)
```

SYSINDEX 互換ビュー (旧式)

SYSINDEX ビューは、SYSINDEX システム・テーブルを提供していた古いバージョンのソフトウェアとの互換性を保つために用意されています。ただし、以前の SYSINDEX テーブルは、SYSIDX システム・ビューに対応する ISYSIDX システム・テーブルで置換されたため、ISYSIDX の使用をおすすめします。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSINDEX"
  as select b.table_id,
    b.index_id,
    b.object_id,
    p.root,
    b.dbspace_id,
    case b."unique"
    when 1 then 'Y'
    when 2 then 'U'
    when 3 then 'M'
    when 4 then 'N'
    when 5 then 'Y'
    else 'I'
    end as "unique",
    t.creator,
    b.index_name,
    r.remarks,
    10 as hash_limit,
    b.dbspace_id as file_id
    from(SYS.ISYSIDX as b left outer join SYS.ISYSPHYSIDX as p
on(b.table_id = p.table_id and b.phys_index_id = p.phys_index_id))
      left outer join SYS.ISYSREMARK as r on(b.object_id =
r.object_id)
      ,SYS.ISYSTAB as t
    where t.table_id = b.table_id
    and b.index_category = 3
```

SYSINDEXES 統合ビュー

SYSINDEXES ビューの各ローは、データベース内のインデックス1つを示します。このビューの代わりに、SYSIDX と SYSIDXCOL のシステム・ビューを使用することもできます。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSINDEXES"( icreator,
  iname,fname,creator,tname,indextype,
  colnames,interval,level_num )
  as select u.user_name,idx.index_name,dbs.dbspace_name,u.user_name,
    tab.table_name,
    case idx.index_category
    when 1 then 'Primary Key'
    when 2 then 'Foreign Key'
    when 3 then(
      if idx."unique" = 4 then 'Non-unique'
      else if idx."unique" = 2 then 'UNIQUE constraint'
        else if idx."unique" = 5 then 'UNIQUE NULLS NOT DISTINCT'
          else 'UNIQUE'
```

```
          endif
        endif
      endif) when 4 then 'Text Index' end,(select
list(string(c.column_name,
      if ixc."order" = 'A' then ' ASC' else ' DESC' endif) order by
      ixc.table_id asc,ixc.index_id asc,ixc.sequence asc)
      from SYS.ISYSIDXCOL as ixc
        join SYS.ISYSTABCOL as c on(
        c.table_id = ixc.table_id
        and c.column_id = ixc.column_id)
      where ixc.index_id = idx.index_id
      and ixc.table_id = idx.table_id),
    0,0
    from SYS.ISYSTAB as tab
     join SYS.ISYSDBSPACE as dbs on(tab.dbspace_id = dbs.dbspace_id)
      join SYS.ISYSIDX as idx on(idx.table_id = tab.table_id)
      join SYS.ISYSUSER as u on u.user_id = tab.creator
```

SYSINDEXES ASE 互換ビュー

sysindexes には、クラスタード・インデックスごとに 1 つのロー、ノンクラスタード・インデックスごとに 1 つのロー、クラスタード・インデックスのないテーブルごとに 1 つのロー、text カラムや image カラムのあるテーブルごとに 1 つのローが含まれています。

このテーブルには、関数ベース・インデックスまたは計算カラムに作成されたインデックスごとに 1 つのローも格納されています。

このビューは、DBO が所有します。

参照：

- 各 Adaptive Server Enterprise データベース内のテーブル (699 ページ)
- SYSCOLUMNS ASE 互換ビュー (614 ページ)
- SYSCOMMENTS ASE 互換ビュー (615 ページ)
- SYSIQOBJECTS ASE 互換ビュー (644 ページ)
- SYSIQVINDEX ASE 互換ビュー (647 ページ)
- SYSOBJECTS ASE 互換ビュー (653 ページ)
- SYSTYPES ASE 互換ビュー (690 ページ)
- SYSUSERS ASE 互換ビュー (696 ページ)

SYSINFO 互換ビュー (旧式)

SYSINFO ビューは、データベースが作成されたときに定義された、データベースの情報を示します。このテーブルは常に 1 つのローを持ちます。このビューは組み込み関数経由で取得でき、カタログには保存されません。次に SYSINFO ビューの定義を示します。

```
ALTER VIEW "SYS"."SYSINFO"( page_size,
  encryption,
```

```
  blank_padding,
  case_sensitivity,
  default_collation,
  database_version )
  as select db_property('PageSize'),
    if db_property('Encryption') <> 'None' then 'Y' else 'N' endif,
    if db_property('BlankPadding') = 'On' then 'Y' else 'N' endif,
    if db_property('CaseSensitive') = 'On' then 'Y' else 'N' endif,
    db_property('Collation'),
    NULL
```

SYSIQBACKUPHISTORY システム・ビュー

このビューは、ISYSIQBACKUPHISTORYからグループの情報を読みやすい形式で表示します。このビューの各ローは、正常に終了した個々のバックアップ処理を記述します。

SYSIQBACKUP ビューは、カラムのタイプ、サブタイプ、bkp_virtual に対する等価文字列値を示します。

カラム名	カラム・タイプ	カラム制約	説明
bu_id	unsigned bigint	NOT null	操作のチェックポイントのトランザクション識別子。バックアップ処理のバックアップ ID
bu_time	timestamp	NOT null	バックアップ記録に記録されたバックアップ処理が行われた時刻
type	tinyint	NOT null	バックアップのタイプ： 0 = FULL 1 = INCREMENTAL 2 = INCREMENTAL SINCE FULL
selective_type	tinyint	NOT null	バックアップのサブタイプ： 0 = ALL (すべての dbfiles をバックアップ) 1 = READ/WRITE ONLY (すべての読み書きファイルをバックアップ) 2 = READ ONLY (特定の読み取り専用ファイルをバックアップ)
virtual_type	tinyint	NOT null	仮想バックアップの種類 0 = NONE 1 = DECOUPLED 2 = ENCAPSULATED
dependson_id	unsigned bigint	NULL	FULL バックアップには NULL
cmd	long varchar	NOT null	コマンドの完全テキスト
作成者	char(128)	NOT null	バックアップ・コマンドを出したユーザ
バージョン	unsigned int	NOT null	バックアップのバージョン

基本となるシステム・テーブルでの制約

プライマリ・キー (bu_id)

参照：

- sp_iqbackupdetails プロシージャ (383 ページ)

SYSIQBACKUPHISTORYDETAIL システム・ビュー

このビューはバックアップ時にデータベース内に存在するすべての dbfile レコードを記述します。このビューの各ローは、正常に終了した個々のバックアップ処理を記述します。

ISYSIQBACKUPHISTORYDETAIL からのグループ情報を読みやすい形式で表示します。各カラムのカラム制約は NOT NULL です。

カラム名	カラム・タイプ	説明
bu_id	unsigned bigint	操作のチェックポイントのトランザクション識別子。バックアップ処理のバックアップ ID。
dbspace_id	smallint	この dbfile レコードに関連付けられている DB 領域 ID。
dbfile_id	smallint	バックアップ処理の進行中に DB 領域に存在する dbfile ID。
dbspace_rwstatus	char(1)	T は読み書き可能を示します。
dbspace_createid	unsigned bigint	DB 領域を作成したトランザクションのトランザクション ID。
dbspace_alterid	unsigned bigint	DB 領域に RO マークを付けたトランザクションの ID。マークなしの場合は ID を作成します。
dbspace_online	char(1)	T はオンラインを示します。
dbfile_rwstatus	char(1)	T は読み書き可能を示します。
dbfile_createid	unsigned bigint	この dbfile を作成したトランザクションのトランザクション ID。
dbfile_alterid	unsigned bigint	この dbfile の読み書きステータスを最後に変更したトランザクションのトランザクション ID。
is_backed_up	char(1)	このバックアップでその dbfile ファイルがバックアップされたことを示します。
start_block	unsigned bigint	dbfile の開始ブロック。

カラム名	カラム・タイプ	説明
num_blocks	unsigned bigint	dbfile 内の総ブロック数。
num_blocks_backed_up	unsigned bigint	バックアップされた総 IQ ブロック数。
dbspace_name	char(128)	DB 領域名。
dbfile_name	char(128)	dbfile ファイルの論理ファイル名。
dbfile_path	long varchar	ファイルの物理パス。

基本となるシステム・テーブルでの制約

プライマリ・キー (bu_id、dbfile_id)

外部キー (txn_id) が SYS.ISYSBACKUPHISTORY を参照。

SYSIQCOLUMN システム・ビュー (廃止)

SYSIQCOLUMN の代わりに SYSIQTABCOL システム・ビューが使用されます。

参照：

- SYSIQTABCOL システム・ビュー (646 ページ)

SYSIQDBFILE システム・ビュー

ISYSIQDBFILE からのグループ情報を読みやすい形式で表示します。

注意： このビューは廃止になった SYSIQFILE システム・ビューの代わりに使用されます。

カラム名	カラム・タイプ	説明
dbfile_id	small int	dbfile のユニークな ID
start_block	rowid	最初のブロックの番号
block_count	rowid	このファイル (DB 領域) のブロック数
reserve_size	rowid	DB 領域に事前に割り付けられているファイル・システム領域
allocated	char(1)	セグメントが事前割り付けされるか (T)、または自動割り付けされるか (F) を定義

カラム名	カラム・タイプ	説明
data_offset	unsigned int	Sybase IQ のデータが開始する場所の、ロー・パーティションの開始点から見た相対的なバイト・ロケーション
create_time	タイムスタンプ	ファイルの作成日時
last_modified	タイムスタンプ	ファイルの最終変更日時
read_write	char(1)	T は読み書き可能を示します。
オンライン	char(1)	T はオンラインを示します。
create_txn_id	unsigned bigint	その dbfile ファイルを作成したトランザクション ID
alter_txn_id	unsigned bigint	読み書きステータスを最後に変更したトランザクション ID
server_id	unsigned int	マルチプレックス・サーバの名前
file_name	text	IQ DB 領域を開くためにマルチプレックス・サーバが使用する IQ DB 領域名

基本となるシステム・テーブルでの制約

外部キー (server_id) が SYS.ISYSIQMPXSERVER を参照。

ユニーク (server_id, file_name)

参照：

- SYSIQFILE システム・ビュー (廃止) (637 ページ)

SYSIQDBSPACE システム・ビュー

ISYSIQDBSPACE からのグループ情報を読みやすい形式で表示します。

カラム名	カラム・タイプ	説明
dbspace_id	small int	データベース内の各 DB 領域に与えられるユニークな番号 (DB 領域 ID)
last_modified	タイムスタンプ	最後に DB 領域の読み書きステータスが変更された日時
segment_type	char(8)	セグメントのタイプ： Main、Temp、Msg
read_write	char(1)	'T' ― 読み書き可能、'F' ― 読み取り専用
オンライン	char(1)	'T' ― オンライン、'F' ― オフライン

カラム名	カラム・タイプ	説明
create_txn_id	unsigned bigint	DB 領域を作成するトランザクション ID
alter_txn_id	unsigned bigint	読み書きステータスを最後に変更したトランザクション ID
striping_on	char(1)	'T' ― ディスク・ストライピングがオン、'F' ― ディスク・ストライピングがオフ
stripe_size_kb	unsigned int	ディスク・ストライピング・アルゴリズムが次の dbfile に移動する前に、DB 領域の各ファイルに書き込まれるデータ量 (KB)

基本となるシステム・テーブルでの制約

プライマリ・キー (dbspace_id)

外部キー (dbspace_id) が SYS.ISYSDBSPACE(dbspace_id) を参照。

SYSIQFILE システム・ビュー (廃止)

SYSIQFILE の代わりに SYSIQDBFILE システム・ビューが使用されます。

参照：

- SYSIQDBFILE システム・ビュー (635 ページ)

SYSIQIDX システム・ビュー

`ISYSIQIDX` からのグループ情報を読みやすい形式で表示します。`SYSIQIDX` ビューの各ローは IQ インデックスを記述します。

注意： このビューは廃止になった SYSIQINDEX システム・ビューの代わりに使用されます。

カラム名	カラム・タイプ	説明
table_id	unsigned int	このインデックスが適用されるテーブルを識別するユニークなテーブル番号
index_id	unsigned int	個々のテーブルの各インデックスに割り当てられたユニークなインデックス番号
index_type	char(4)	インデックス・タイプ
index_owner	char(4)	インデックスの所有者
max_key	unsigned int	内部用

カラム名	カラム・タイプ	説明
identity_location	hs_vdorecid	内部用
identity_size	unsigned int	内部用
identity_location_size	unsigned int	内部用
link_table_id	unsigned int	内部用
link_index_id	unsigned int	内部用
delimited_by	varchar(1024)	(WD インデックスのみ) カラムの文字列を、そのカラムの WD インデックスに格納する単語に解析するときに使用されるセパレータのリスト
limit	unsigned int	(WD インデックスのみ) WD インデックスの単語の最大長

基本となるシステム・テーブルでの制約

プライマリ・キー (table_id、index_id)

外部キー (table_id、index_id) が SYS.ISYIDX を参照。

外部キー (link_table_id、link_index_id、table_id、index_id) が SYS.ISYSIDX を参照。

SYSIQINFO システム・ビュー

ISYSIQINFO からのグループ情報を読みやすい形式で表示します。

ISYSIQINFO システム・テーブルは、**CREATE DATABASE** を使って Sybase IQ データベースを作成したときに定義された、データベースの特性を示します。 このテーブルに含まれるローの数は常に 1 つのみです。

カラム名	カラム・タイプ	説明
create_time	TIMESTAMP NOT NULL	データベースの作成日時
update_time	TIMESTAMP NOT NULL	前回の更新日時
file_format_version	UNSIGNED INT NOT NULL	このデータベースのファイルのファイル・フォーマット番号
cat_format_version	UNSIGNED INT NOT NULL	このデータベースのカタログ・フォーマット番号
sp_format_version	UNSIGNED INT NOT NULL	このデータベースのストアド・プロシージャ・フォーマット番号

カラム名	カラム・タイプ	説明
block_size	UNSIGNED INT NOT NULL	データベースに指定されているブロック・サイズ
chunk_size	UNSIGNED INT NOT NULL	データベースに指定されているブロック・サイズとページ・サイズを基にした、ページあたりのブロック数
file_format_date	CHAR(10) NOT NULL	ファイル・フォーマット番号の最終更新日付
dbsig	BINARY(136) NOT NULL	カタログが内部で使用
commit_txn_id	unsigned bigint	内部用
rd_commit_txn_id	unsigned bigint	内部用
multiplex name	CHAR(128) NULL	このデータベースがメンバになっているマルチプレックスの名前
last_multiplex_mode	TINYINT NULL	(Sybase IQ 15.3 では使用されないカラム) カタログを読み書きアクセスで最後に開いたサーバのモード。 次の値のいずれかになります。 • 0 – シングル・ノード • 1 – リーダ • 2 – コーディネータ • 3 – ライタ

SYSIQJOINIDX システム・ビュー

ISYSIQJOININDX からのグループ情報を読みやすい形式で表示します。
SYSIQJOININDX ビューの各ローは IQ ジョイン・インデックスを記述します。

注意： このビューは廃止になった SYSIQJOININDEX システム・ビューの代わりに使用されます。

カラム名	カラム・タイプ	説明
joinindex_id	unsigned int	各ジョイン・インデックスには、プライマリ・キーになるユニークな番号が付いています。
jvt_id	unsigned int	内部用。
dbspace_id	smallint	DB 領域の ID。
joinindex_name	char(128)	ジョイン・インデックスの名前を定義します。
joinindex_type	char(12)	内部用。

カラム名	カラム・タイプ	説明
作成者	unsigned int	ジョイン・インデックスを作成したユーザの番号。
join_info_location	hs_vdorecid	内部用。
join_info_loc_size	unsigned int	内部用。
join_info_size	unsigned int	内部用。
block_map	hs_blockmapidentity	内部用。
block_map_size	unsigned int	内部用。
vdo	hs_vdoidentity	内部用。
vdo_size	unsigned_int	内部用。
commit_txn_id	unsigned bigint	内部用。
txn_id	unsigned bigint	内部用。
有効	char(1)	このジョイン・インデックスを同期する必要があるかどうかを示します。Y の場合は同期の必要はなく、N の場合は同期の必要があります。

基本となるシステム・テーブルでの制約

プライマリ・キー (joinindex_id)

外部キー (jvt_id) が SYS.ISYSTAB を参照。

外部キー (dbspace_id) が SYS.ISYSDBSPACE を参照。

外部キー (creator) が SYS.ISYSUSER が参照

ユニーク (jvt_id, commit_txn_id, txn_id)

参照：

- SYSIQJOININDEX システム・ビュー (廃止) (640 ページ)

SYSIQJOININDEX システム・ビュー (廃止)

SYSIQJOININDEX の代わりに SYSIQJOINIDX システム・ビューが使用されます。

参照：

- SYSIQJOINIDX システム・ビュー (639 ページ)

SYSIQJOINIXCOLUMN システム・ビュー

ISYSIQJOINIXCOLUMN からのグループ情報を読みやすい形式で表示します。SYSIQJOINIXCOLUMN ビューの各ローは IQ ジョイン・インデックスを記述します。

```
ALTER VIEW "SYS"."SYSIQJOINIXCOLUMN"
as select * from SYS.ISYSIQJOINIXCOLUMN
```

カラム名	カラム・タイプ	説明
joinindex_id	unsigned bigint	SYSIQJOINIDX のジョイン・インデックス値に対応します。
left_table_id	unsigned int	ジョイン演算の左項に指定する SYSTAB のテーブル値に対応します。
left_column_id	unsigned int	ジョイン演算の左項の一部として指定する SYSTABCOL のカラム値に対応します。
join_type	char(4)	現在サポートしている値は "=" だけです。
right_table_id	unsigned int	ジョイン演算の右項に指定する SYSTAB のテーブル値に対応します。
right_column_id	unsigned int	ジョイン演算の右項の一部として指定する SYSTABCOL のカラム値に対応します。
order_num	unsigned int	内部用。
left_order_num	unsigned int	内部用。
right_order_num	unsigned int	内部用。
key_type	char(8)	キーに対するジョインのタイプを定義します。'NATURAL' はナチュラル・ジョイン、'KEY' はキー・ジョイン、'ON' は左外部ジョイン／右外部ジョイン／フル・ジョインです。
coalesce	char(1)	未使用。

基本となるシステム・テーブルでの制約

プライマリ・キー (joinindex_id, left_table_id, left_column_id, right_table_id, right_column_id)

外部キー (joinindex_id) が SYS.ISYSIQJOINIDX を参照。

外部キー (left_table_id, column_id) が SYS.ISYSTABCOL を参照。

外部キー (right_table_id, column_id) が SYS.ISYSTABCOL を参照。

SYSIQJOINIXTABLE システム・ビュー

ISYSIQJOINIXTABLE からのグループ情報を読みやすい形式で表示します。SYSIQJOINIXTABLE ビューの各ローは IQ ジョイン・インデックスを記述します。

```
ALTER VIEW "SYS"."SYSIQJOINIXTABLE"
as select * from SYS.ISYSIQJOINIXTABLE
```

カラム名	カラム・タイプ	説明
table_id	unsigned int	ジョイン演算に含まれる SYSTAB のテーブル値に対応します。
joinindex_id	unsigned bigint	SYSIQJOINIDX のジョイン・インデックス値に対応します。
アクティブ	unsigned int	ジョイン・インデックスでテーブルが使用される回数を定義します。

基本となるシステム・テーブルでの制約

プライマリ・キー (table_id, joinindex_id)

外部キー (table_id) が SYS.ISYSTAB を参照。

外部キー (joinindex_id) が SYS.ISYSIQJOINIDX を参照。

SYSIQLOGICALSERVER システム・ビュー

ISYSIQLOGICALSERVER テーブルの情報を表示します。このテーブルには、論理サーバの情報、および論理サーバと論理サーバ・ポリシーとの対応情報が格納されています。

詳細については、『Sybase IQ Multiplex の使用』の「マルチプレックス・リファレンス」>「システム・ビュー」>「SYSIQLOGICALSERVER システム・ビュー」を参照してください。

SYSIQLOGINPOLICYLSINFO システム・ビュー

ISYSIQLOGINPOLICYLSINFO テーブルのグループ情報を表示します。このテーブルには、ログイン・ポリシーの論理サーバ割り当て情報が格納されています。

詳細については、『Sybase IQ Multiplex の使用』の「マルチプレックス・リファレンス」>「システム・ビュー」>「SYSIQLOGINPOLICYLSINFO システム・ビュー」を参照してください。

SYSIQLSLOGINPOLICYOPTION システム・ビュー

ISYSIQLOGINPOLICYLSINFO テーブルのグループ情報を表示します。このテーブルには、ログイン・ポリシーの論理サーバ割り当て情報が格納されています。

詳細については、『Sybase IQ Multiplex の使用』の「マルチプレックス・リファレンス」>「システム・ビュー」>「SYSIQLSLOGINPOLICYOPTION システム・ビュー」を参照してください。

SYSIQLSMEMBER システム・ビュー

ISYSIQLSMEMBER テーブルのグループ情報を表示します。このテーブルには、論理サーバのメンバシップ情報が格納されています。

詳細については、『Sybase IQ Multiplex の使用』の「マルチプレックス・リファレンス」>「システム・ビュー」>「SYSIQLSMEMBER システム・ビュー」を参照してください。

SYSIQLSMEMBERS 統合ビュー

このビューは、ユーザ定義論理サーバのすべてのメンバシップを示します。

詳細については、『Sybase IQ Multiplex の使用』の「マルチプレックス・リファレンス」>「システム・ビュー」>「SYSIQLSMEMBERS 統合ビュー」を参照してください。

SYSIQLSLOGINPOLICIES 統合ビュー

このビューは、ログイン・ポリシーのすべての論理サーバ割り当てを示します。

詳細については、『Sybase IQ Multiplex の使用』の「マルチプレックス・リファレンス」>「システム・ビュー」>「SYSIQLSLOGINPOLICIES 統合ビュー」を参照してください。

SYSIQLSPOLICY システム・ビュー

ISYSIQLSPOLICY テーブルのグループ情報を表示します。このテーブルには、論理サーバのポリシーが格納されています。

詳細については、『Sybase IQ Multiplex の使用』の「マルチプレックス・リファレンス」>「システム・ビュー」>「SYSIQLSPOLICY システム・ビュー」を参照してください。

SYSIQLSPOLICYOPTION システム・ビュー

ISYSIQLSPOLICYOPTION テーブルのグループ情報を表示します。このテーブルには、論理サーバのポリシー オプションが格納されています。

詳細については、『Sybase IQ Multiplex の使用』の「マルチプレックス・リファレンス」>「システム・ビュー」>「SYSIQLSPOLICYOPTION システム・ビュー」を参照してください。

SYSIQMPXSERVER システム・ビュー

ISYSIQMPXSERVER テーブルを読みやすい形式で表示します。このテーブルには、特定のマルチプレックス・ノードのメンバシップ・プロパティおよびバージョン・ステータス・データが格納されています。

セカンダリ・ノードを実行するには、マルチプレックス・グリッド・オプションのライセンスを取得している必要があります。 詳細については、『Sybase IQ Multiplex の使用』を参照してください。

SYSIQOBJECTS ASE 互換ビュー

sysiqobjects は、各ローに個々のシステム・テーブル、ユーザ・テーブル、ビュー、プロシージャ、トリガ、イベント、ジョイン・インデックス、制約、ドメイン (sysdomain)、ドメイン (sysusertype)、カラム、インデックスを表示します。このビューは、DBO が所有します。

参照：

- 各 Adaptive Server Enterprise データベース内のテーブル (699 ページ)
- SYSCOLUMNS ASE 互換ビュー (614 ページ)
- SYSCOMMENTS ASE 互換ビュー (615 ページ)
- SYSINDEXES ASE 互換ビュー (632 ページ)
- SYSIQVINDEX ASE 互換ビュー (647 ページ)
- SYSOBJECTS ASE 互換ビュー (653 ページ)
- SYSTYPES ASE 互換ビュー (690 ページ)
- SYSUSERS ASE 互換ビュー (696 ページ)

SYSIQPARTITIONCOLUMN システム・ビュー

ISYSIQPARTITIONCOLUMN からのグループ情報を読みやすい形式で表示します。

```
ALTER VIEW "SYS"."SYSIQPARTITIONCOLUMN"
as select * from SYS.ISYSIQPARTITIONCOLUMN
```

SYSIQPARTITIONCOLUMN ビューの各ローで記述されているパーティションのカラムは、SYSPARTITIONSCHEME ビュー内で記述されている分割されたテーブル

内の SYSIQPARTITION ビュー内で記述されているパーティション内のカラムです。SYSIQPARTITIONCOLUMN では、そのパーティションの DB 領域に保存されていないカラムのパーティションしか記述しません。

カラム名	カラム・タイプ	説明
partitioned_object_id	unsigned bigint	分割されたオブジェクト (テーブル) に割り当てられたユニークな ID
partition_id	unsigned int	分割されたテーブル内のパーティションを識別します。
column_id	unsigned int	カラムの ID。
dbspace_id	smallint	パーティションの指定されたカラムが保存されている DB 領域の ID。

基本となるシステム・テーブルでの制約

プライマリ・キー (partitioned_object_id, partition_id, column_id)

外部キー (partitioned_object_id, partition_id) が SYS.ISYSPARTITION を参照。

外部キー (dbspace_id) が SYS.ISYSDBSPACE を参照。

SYSIQTAB システム・ビュー

ISYSIQTAB からのグループ情報を読みやすい形式で表示します。SYSIQTAB ビューの各ローは IQ テーブルを記述します。

```
ALTER VIEW "SYS"."SYSIQTAB"
as select * from SYS.ISYSIQTAB
```

注意： このビューは廃止になった SYSIQTABLE システム・ビューの代わりに使用されます。

カラム名	カラム・タイプ	説明
table_id	unsigned int	各テーブルには、プライマリ・キーになるユニークな番号 (テーブル番号) が付いています。
block_map	hs_blockmapidentity	内部用。
block_map_size	unsigned int	内部用。
vdo	hs_vdoidentity	内部用。
vdoid_size	unsigned int	内部用。
info_location	hs_vdorecid	未使用。常にゼロ。

カラム名	カラム・タイプ	説明
info_recid_size	unsigned int	未使用。常にゼロ。
info_location_size	unsigned int	未使用。常にゼロ。
commit_txn_id	unsigned bigint	内部用。
txn_id	unsigned bigint	内部用。
join_id	unsigned int	内部用。
update_time	タイムスタンプ	IQ テーブルの最終変更日時。

基本となるシステム・テーブルでの制約

プライマリ・キー (table_id)

参照：
- SYSIQTABLE システム・ビュー (廃止) (647 ページ)

SYSIQTABCOL システム・ビュー

ISYSIQTABCOL からのグループ情報を読みやすい形式で表示します。
SYSIQTABCOL ビューの各ローは IQ テーブル内のカラムを記述します。

```
ALTER VIEW "SYS"."SYSIQTABCOL"
as select * from SYS.ISYSIQTABCOL
```

注意： このビューは廃止になった SYSIQCOLUMN システム・ビューの代わりに使用されます。

カラム名	カラム・タイプ	説明
table_id	unsigned int	このカラムが属するテーブルを識別するユニークなテーブル番号。これは、SYSTAB の table_id カラムに対応します。
column_id	unsigned int	各テーブルでは、カラムに 1 から始まる番号が付きます。select * from table コマンドを実行すると、このカラム番号の順番でカラムが表示されます。
link_table_id	unsigned int	内部用。
link_column_id	unsigned int	内部用。
max_length	unsigned int	カラムで使用できる最大長を示します。
approx_unique_count	rowid	このカラムのユニークな値 (カーディナリティ) の概数。

カラム名	カラム・タイプ	説明
cardinality	rowid	このカラムのユニークな値 (カーディナリティ) の実数。
has_data	char(1)	カラムにデータがあるかどうかを示します (T/F)。
has_original	char(1)	ジョイン・インデックスにオリジナル・データがあるかどうかを示します (T/F)。
original_not_null	char(1)	オリジナル・データがあるジョイン・インデックス・カラムが NOT NULL であったかどうかを示します (T/F)。
original_unique	char(1)	オリジナル・データがあるジョイン・インデックス・カラムが UNIQUE であったかどうかを示します (T/F)。

基本となるシステム・テーブルでの制約

プライマリ・キー (table_id)

参照：

- SYSIQCOLUMN システム・ビュー (廃止) (635 ページ)

SYSIQTABLE システム・ビュー (廃止)

SYSIQTABLE の代わりに SYSIQTAB システム・ビューが使用されます。

参照：

- SYSIQTAB システム・ビュー (645 ページ)

SYSIQVINDEX ASE 互換ビュー

sysiqvindex では、FP 以外の個々のインデックスに 1 つのローが割り当てられます。

このビューは、DBO が所有します。

参照：

- 各 Adaptive Server Enterprise データベース内のテーブル (699 ページ)
- SYSCOLUMNS ASE 互換ビュー (614 ページ)
- SYSCOMMENTS ASE 互換ビュー (615 ページ)
- SYSINDEXES ASE 互換ビュー (632 ページ)
- SYSIQOBJECTS ASE 互換ビュー (644 ページ)
- SYSOBJECTS ASE 互換ビュー (653 ページ)

- SYSTYPES ASE 互換ビュー (690 ページ)
- SYSUSERS ASE 互換ビュー (696 ページ)

SYSIXCOL 互換ビュー (旧式)

SYSIXCOL ビューは、SYSIXCOL システム・テーブルを提供していた古いバージョンのソフトウェアとの互換性を保つために用意されています。ただし、SYSIXCOL システム・テーブルは ISYSIDXCOL システム・テーブルで置換され、SYSIDXCOL システム・ビューに対応しています。SYSIDXCOL システム・ビューの使用をおすすめします。

SYSIXCOL の各ローは、インデックスのカラムを示します。ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSIXCOL"
  as select a.table_id,
    a.index_id,
    a.sequence,
    a.column_id,
    a."order"
    from SYS.ISYSIDXCOL as a
      ,SYS.ISYSIDX as b
    where a.table_id = b.table_id
    and a.index_id = b.index_id
    and b.index_category = 3
```

SYSJAR システム・ビュー

SYSJAR システム・ビューの各ローは、データベースに格納されている JAR ファイルを定義します。このビューの基本となるシステム・テーブルは ISYSJAR です。

カラム名	データ型	説明
jar_id	INTEGER	JAR ファイルを識別するユニークな番号。
object_id	UNSIGNED BIGINT	データベースのインデックスをユニークに識別する JAR ファイルの内部 ID。
creator	UNSIGNED INT	JAR ファイルの作成者のユーザ番号。
jar_name	LONG VARCHAR	JAR ファイルの名前。
jar_file	LONG VARCHAR	データベース内の JAR ファイルの外部ファイル名。
update_time	TIMESTAMP	JAR ファイルが最後に更新された時刻。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (jar_id)
```

```
FOREIGN KEY (object_id) references SYS.ISYSOBJECT (object_id) MATCH
UNIQUE FULL
```

```
UNIQUE Index (jar_name)
```

SYSJARCOMPONENT システム・ビュー

SYSJAR システム・ビューの各ローは、JAR ファイル・コンポーネントを定義します。このビューの基本となるシステム・テーブルは ISYSJARCOMPONENT です。

カラム名	データ型	説明
component_id	INTEGER	コンポーネントの ID を含むプライマリ・キー。
jar_id	INTEGER	JAR の ID 番号を含むフィールド。
component_name	LONG VARCHAR	コンポーネント名。
component_type	CHAR(1)	コンポーネントの型。
contents	LONG BINARY	JAR ファイルのバイト・コード。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (component_id)
```

```
FOREIGN KEY (jar_id) references SYS.ISYSJAR (jar_id)
```

SYSJAVACLASS システム・ビュー

SYSJAVACLASS システム・ビューの各ローは、データベースに格納されている Java クラス 1 つを示します。このビューの基本となるシステム・テーブルは ISYSJAVACLASS です。

カラム名	データ型	説明
class_id	INTEGER	Java クラスのユニークな番号。テーブルのプライマリ・キーでもあります。
object_id	UNSIGNED BIGINT	データベースのインデックスをユニークに識別する Java クラスの内部 ID。
creator	UNSIGNED INT	クラスの作成者のユーザ番号。
jar_id	INTEGER	クラスがある JAR ファイルの ID。
class_name	LONG VARCHAR	Java クラスの名前。
public	CHAR(1)	クラスがパブリック (Y) かプライベート (N) かを示します。

カラム名	データ型	説明
component_id	INTEGER	SYSJARCOMPONENT システム・ビューのコンポーネントの ID。
update_time	TIMESTAMP	クラスが最後に更新された時刻。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (class_id)
```

```
FOREIGN KEY (object_id) references SYS.ISYSOBJECT (object_id) MATCH
UNIQUE FULL
```

```
FOREIGN KEY (creator) references SYS.ISYSUSER (user_id)
```

```
FOREIGN KEY (component_id) references SYS.ISYSJARCOMPONENT
(component_id)
```

SYSLOGINMAP システム・ビュー

SYSLOGINMAP システム・ビューには、統合化ログインまたは Kerberos ログインを使用してデータベースに接続できる各ユーザに 1 つのローが含まれます。セキュリティ上の理由から、このビューの内容は DBA 権限を持つユーザだけが表示できます。このビューの基本となるシステム・テーブルは ISYSLOGINMAP です。

カラム名	データ型	説明
login_mode	TINYINT	ログインのタイプ。統合化ログインの場合は 1、Kerberos ログインの場合は 2。
login_id	VARCHAR(1024)	database_uid にマッピングする統合化ログインのユーザ・プロファイル名または Kerberos 原則。
object_id	UNSIGNED BIGINT	ユニークな識別子。ユーザ ID とデータベースのユーザ ID 間のマッピングに 1 つ。
database_uid	UNSIGNED INT	ログイン ID をマッピングするデータベース・ユーザ ID。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (login_mode, login_id)
```

```
FOREIGN KEY (object_id) references SYS.ISYSOBJECT (object_id) MATCH
UNIQUE FULL
```

```
FOREIGN KEY (database_uid) references SYS.ISYSUSER (user_id)
```

SYSLOGINPOLICY システム・ビュー

このビューの基本となるシステム・テーブルは ISYSLOGINPOLICY です。

カラム名	データ型	説明
login_policy_id	UNSIGNED BIGINT	ログイン・ポリシーのユニークな識別子。
login_policy_name	CHAR(128)	ログイン・ポリシーの名前。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (login_policy_id)
```

```
FOREIGN KEY (login_policy_id) references SYS.ISYSOBJECT (object_id)
```

```
UNIQUE Index (login_policy_name)
```

SYSLOGINPOLICYOPTION システム・ビュー

このビューの基本となるシステム・テーブルは ISYSLOGINPOLICYOPTION です。

カラム名	データ型	説明
login_policy_id	UNSIGNED BIGINT	ログイン・ポリシーのユニークな識別子。
login_option_name	CHAR(128)	ログイン・ポリシーの名前。
login_option_value	LONG VARCHAR	ログイン・ポリシーの作成時の値。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (login_policy_id, login_option_name)
```

```
FOREIGN KEY (login_policy_id) references SYS.ISYSLOGINPOLICY
(login_policy_id)
```

SYSLOGINS ASE 互換ビュー

このビューの所有者は DBO というユーザです。syslogins の各ローは、有効な Adaptive Server Enterprise の各ユーザ・アカウントです。

SYSMVOPTION システム・ビュー

SYSMVOPTION システム・ビューの各ローは、マテリアライズド・ビューまたはテキスト・インデックスを作成した時点のオプション値に関する記述です。オプションの名前は、SYSMVOPTIONNAME システム・ビューにあります。このビューの基本となるシステム・テーブルは ISYSMVOPTION です。

カラム名	データ型	説明
view_object_id	UNSIGNED BIGINT	マテリアライズド・ビューのオブジェクト ID。
option_id	UNSIGNED INT	データベースのオプションを識別するユニークな番号。オプション名を参照するには、SYSMVOPTIONNAME システム・ビューを参照してください。
option_value	LONG VARCHAR	マテリアライズド・ビューが作成された時点のオプション値。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (view_object_id, option_id)
```

```
FOREIGN KEY (view_object_id) references SYS.ISYSOBJECT (object_id)
```

```
FOREIGN KEY (option_id) references SYS.ISYSMVOPTIONNAME (option_id)
```

SYSMVOPTIONNAME システム・ビュー

SYSMVOPTION システム・ビューの各ローには、マテリアライズド・ビューまたはテキスト・インデックスを作成した時点の名前オプション値があります。オプションの値は、SYSMVOPTION システム・ビューにあります。このビューの基本となるシステム・テーブルは ISYSMVOPTIONNAME です。

カラム名	データ型	説明
option_id	UNSIGNED INT	データベースのオプションをユニークに識別する番号。
option_name	CHAR(128)	オプションの名前。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (option_id)
```

```
UNIQUE Index (option_name)
```

SYSOBJECT システム・ビュー

SYSOBJECT システム・ビューの各ローは、データベース・オブジェクトを示します。このビューの基本となるシステム・テーブルは ISYSOBJECT です。

カラム名	データ型	説明
object_id	UNSIGNED BIGINT	データベースのオブジェクトをユニークに識別するインデックスの内部 ID。

カラム名	データ型	説明
status	TINYINT	オブジェクトのステータス。次のような値があります。 • **1 (有効)** – オブジェクトは、データベース・サーバから使用できます。このステータスは、ENABLED と同等です。つまり、オブジェクトを ENABLE にすると、ステータスは VALID に変更されます。 • **2 (無効)** – 内部操作後に、オブジェクトを再コンパイルする試みが失敗しました。たとえば、依存するオブジェクトを変更するスキーマの変更後などです。データベース・サーバは、文で参照されるオブジェクトを再コンパイルする試みを継続します。 • **4 (無効化)** – ユーザがオブジェクトを、明示的に無効化しました。たとえば、ALTER TABLE...DISABLE VIEW DEPENDENCIES 文を使用する場合などです。
object_type	TINYINT	オブジェクトの種類。
creation_time	TIMESTAMP	オブジェクトを作成した日付と時刻。
object_type_str	CHAR(128)	オブジェクトの種類。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (object_id)
```

SYSOBJECTS ASE 互換ビュー

sysobjects には、テーブル、ビュー、ストアド・プロシージャ、拡張ストアド・プロシージャ、ログ、ルール、デフォルト、トリガ、検査制約、参照制約、計算カラム、関数ベース・インデックス・キー、テンポラリ・オブジェクト、その他の形式のコンパイル済みオブジェクトごとに 1 つのローが含まれています。

このビューは、DBO が所有します。

さらに、オブジェクト・タイプが N の場合は、パーティションの条件 ID ごとに 1 つのローが含まれます。

参照：

- 各 Adaptive Server Enterprise データベース内のテーブル (699 ページ)
- SYSCOLUMNS ASE 互換ビュー (614 ページ)
- SYSCOMMENTS ASE 互換ビュー (615 ページ)
- SYSINDEXES ASE 互換ビュー (632 ページ)
- SYSIQOBJECTS ASE 互換ビュー (644 ページ)
- SYSIQVINDEX ASE 互換ビュー (647 ページ)

- SYSTYPES ASE 互換ビュー (690 ページ)
- SYSUSERS ASE 互換ビュー (696 ページ)

SYSOPTION システム・ビュー

SYSOPTION システム・ビューには、データベースに格納されている各オプション設定のローのオプションが含まれます。各ユーザはオプションごとに自分の設定を保存できます。また、PUBLIC のユーザ ID に対する設定は、自分の設定を持たないユーザが使うデフォルトの設定になります。このビューの基本となるシステム・テーブルは ISYSOPTION です。

カラム名	データ型	説明
user_id	UNSIGNED INT	このオプション設定が適用されるユーザ番号。
"option"	CHAR(128)	オプションの名前。
"setting"	LONG VARCHAR	現在のオプションの設定。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (user_id, "option")
```

```
FOREIGN KEY (user_id) references SYS.ISYSUSER (user_id)
```

SYSOPTIONS 統合ビュー

SYSOPTIONS ビューの各ローは、SET コマンドを使用して作成されているオプション 1 つを示します。各ユーザはオプションごとに自分の設定を保存できます。また、PUBLIC ユーザに対する設定は、自分の設定を持たないユーザが使うデフォルトの設定になります。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSOPTIONS"( user_name,"option",setting )
  as select u.user_name,opt."option",opt.setting
    from SYS.ISYSOPTION as opt
      join SYS.ISYSUSER as u on opt.user_id = u.user_id
```

SYSOPTSTAT システム・ビュー

SYSOPTSTAT システム・ビューは、コスト・モデルの調整情報を、ALTER DATABASE CALIBRATE 文で計算して格納します。このビューのコンテンツは内部使用のみです。sa_get_dtt システム・プロシージャ経由でアクセスすることをおすすめします。このビューの基本となるシステム・テーブルは ISYSOPTSTAT です。

カラム名	データ型	説明
stat_id	UNSIGNED INT	システムでのみ使用。
group_id	UNSIGNED INT	システムでのみ使用。
format_id	SMALLINT	システムでのみ使用。
data	LONG BINARY	システムでのみ使用。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (stat_id, group_id, format_id)
```

SYSPARTITION システム・ビュー

ISYSPARTITION からのグループ情報を読みやすい形式で表示します。

SYSPARTITION ビューの各ローは、データベース内の分割された個々のオブジェクト (テーブルやインデックス) です。

```
ALTER VIEW "SYS"."SYSPARTITION"
as select * from SYS.ISYSPARTITION
```

カラム名	カラム・タイプ	説明
partitioned_object_id	unsigned bigint	分割されたオブジェクト (テーブル) に割り当てられたユニークな ID
partition_id	unsigned int	分割されたテーブル内のパーティションを識別します。
partition_object_id	unsigned bigint	各テーブル・パーティションはそれ自体が 1 つのオブジェクトなので、テーブル・オブジェクトまたはインデックス・オブジェクトからユニークな番号が割り当てられます。
partition_values	long varchar	この範囲パーティションの上限です。
位置	unsigned int	パーティションの元の番号。
partition_name	char(128)	パーティションの名前

基本となるシステム・テーブルでの制約

プライマリ・キー (partitioned_object_id, partition_id)

ユニーク (partition_object_id, position)

外部キー (partition_object_id) が SYS.ISYSOBJECT を参照。

外部キー (partitioned_object_id) が SYS.ISYSOBJECT を参照。

SYSPARTITIONKEY システム・ビュー

ISYSPARTITIONKEY からのグループ情報を読みやすい形式で表示します。

SYSPARTITIONKEY ビューの各ローは、データベース内の分割された個々のオブジェクト (テーブルやインデックス) です。

```
ALTER VIEW "SYS"."SYSPARTITIONKEY"
as select * from SYS.ISYSPARTITIONKEY
```

カラム名	カラム・タイプ	説明
partitioned_object_id	unsigned bigint	分割された各オブジェクト (テーブル) にはユニークなオブジェクト番号が付いています。
column_id	unsigned int	カラム ID はパーティション・キーの一部となってテーブル・カラムを識別します。
位置	smallint	パーティション・キー内でのこのカラムの位置。0 から始まります。位置 0 はそのパーティション・キーの 1 番目のカラムです。

基本となるシステム・テーブルでの制約

プライマリ・キー (partitioned_object_id, column_id)

外部キー (partitioned_object_id) が SYS.ISYSOBJECT を参照。

SYSPARTITIONSCHEME システム・ビュー

ISYSPARTITIONSCHEME からのグループ情報を読みやすい形式で表示します。

SYSPARTITIONSCHEME ビューの各ローは、データベース内の分割された個々のオブジェクト (テーブルやインデックス) です。

```
ALTER VIEW "SYS"."SYSPARTITIONSCHEME"
as select * from SYS.ISYSPARTITIONSCHEME
```

カラム名	カラム・タイプ	説明
partitioned_object_id	unsigned bigint	分割された各オブジェクト (テーブル) にはユニークな番号が付いています。
partition_method	tinyint	このテーブルの分割方法。有効な 値は、1 – 範囲分割。Sybase IQ は、範囲分割だけをサポートしています。

カラム名	カラム・タイプ	説明
subpartition_method	tinyint	今後のために予約済み

基本となるシステム・テーブルでの制約

プライマリ・キー (partitioned_object_id)

外部キー (partitioned_object_id) が SYS.ISYSOBJECT を参照。

SYSPHYSIDX システム・ビュー

SYSPHYSIDX システム・ビューの各ローは、データベースの物理インデックスを定義します。このビューの基本となるシステム・テーブルは ISYSPHYSIDX です。

カラム名	データ型	説明
table_id	UNSIGNED INT	インデックスが対応するテーブルのオブジェクト ID。
phys_index_id	UNSIGNED INT	テーブル内の物理インデックスのユニークな番号。
root	INTEGER	データベース・ファイルにおける物理インデックスのルート・ページの位置を識別します。
key_value_count	UNSIGNED INT	インデックス内の個別のキー値の数。
leaf_page_count	UNSIGNED INT	リーフ・インデックス・ページの数。
depth	UNSIGNED SMALLINT	物理インデックスの深さ (レベル数)。
max_key_distance	UNSIGNED INT	システムでのみ使用。
seq_transitions	UNSIGNED INT	システムでのみ使用。
rand_transitions	UNSIGNED INT	システムでのみ使用。
rand_distance	UNSIGNED INT	システムでのみ使用。
allocation_bitmap	LONG VARBIT	システムでのみ使用。
long_value_bitmap	LONG VARBIT	システムでのみ使用。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (table_id, phys_index_id)
```

SYSPROCAUTH 統合ビュー

SYSPROCAUTH ビューの各ローは、プロシージャに付与されている一連の権限を示します。または、SYSPROCPERM システム・ビューを使用することもできます。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSPROCAUTH"( grantee,
  creator,procname )
  as select u1.user_name,u2.user_name,p.proc_name
    from SYS.ISYSPROCEDURE as p
      join SYS.ISYSPROCPERM as pp on(p.proc_id = pp.proc_id)
      join SYS.ISYSUSER as u1 on u1.user_id = pp.grantee
      join SYS.ISYSUSER as u2 on u2.user_id = p.creator
```

SYSPROCEDURE システム・ビュー

SYSPROCEDURE システム・ビューの各ローは、データベース内のプロシージャを示します。このビューの基本となるシステム・テーブルは ISYSPROCEDURE です。

カラム名	データ型	説明
proc_id	UNSIGNED INT	各プロシージャにはユニークな番号 (プロシージャ番号) が割り当てられます。
creator	UNSIGNED INT	プロシージャの所有者。
object_id	UNSIGNED BIGINT	データベースのプロシージャをユニークに識別するプロシージャの内部 ID。
proc_name	CHAR(128)	プロシージャ名。1 人の作成者が、同じ名前のプロシージャを 2 つ持つことはできません。
proc_defn	LONG VARCHAR	プロシージャの定義。
remarks	LONG VARCHAR	プロシージャに関する注記。ISYSREMARK システム・テーブル内に格納されている値。
replicate	CHAR(1)	このプロパティは内部でのみ使用されます。
srvid	UNSIGNED INT	プロシージャがリモート・データベース・サーバ上のプロシージャのプロキシである場合は、そのリモート・サーバを表します。

カラム名	データ型	説明
source	LONG VARCHAR	回避されたプロシージャのソース。この値は、ISYSSOURCE システム・テーブル内に格納されています。
avg_num_rows	FLOAT	プロシージャが FROM 句に表示されるときに、クエリの最適化に使用するために収集される情報。
avg_cost	FLOAT	プロシージャが FROM 句に表示されるときに、クエリの最適化に使用するために収集される情報。
stats	LONG BINARY	プロシージャが FROM 句に表示されるときに、クエリの最適化に使用するために収集される情報。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (proc_id)
```

```
FOREIGN KEY (srvid) references SYS.ISYSSERVER (srvid)
```

```
FOREIGN KEY (object_id) references SYS.ISYSOBJECT (object_id) MATCH
UNIQUE FULL
```

```
FOREIGN KEY (creator) references SYS.ISYSUSER (user_id)
```

```
UNIQUE Index (proc_name, creator)
```

SYSPROCPARM システム・ビュー

SYSPROCPARM システム・ビューの各ローは、データベース内のプロシージャに対するパラメータ 1 つを示します。このビューの基本となるシステム・テーブルは ISYSPROCPARM です。

カラム名	データ型	説明
proc_id	UNSIGNED INT	このパラメータが属するプロシージャをユニークに識別します。
parm_id	SMALLINT	各プロシージャは、パラメータに 1 から順に番号を付けます。パラメータ番号の順は、定義された順になっています。関数の場合、最初のパラメータの内容は関数名であり、その関数の戻り値を表します。

カラム名	データ型	説明
parm_type	SMALLINT	パラメータは、次に示すタイプのいずれかに該当します。 • **0** – 標準のパラメータ (変数) • **1** – 結果変数 — 結果セットを返すプロシージャで使用 • **2** – SQLSTATE エラー値 • **3** – SQLCODE エラー値 • **4** – 関数からの戻り値
parm_mode_in	CHAR(1)	このパラメータが、プロシージャに値を提供するかどうかを示します (IN または INOUT パラメータ)。
parm_mode_out	CHAR(1)	このパラメータが、プロシージャからの値 (IN または INOUT パラメータ) と RESULT 句のカラムのどちらを返すかを示します。
domain_id	SMALLINT	SYSDOMAIN システム・ビューにリストされたデータ型番号から、パラメータのデータ型を識別します。
width	BIGINT	文字列パラメータでは長さ、数値パラメータでは精度、その他のデータ型では記憶領域のサイズをバイトで示します。
scale	SMALLINT	数値データ型の場合、小数点以下の桁数です。その他のデータ型の場合、このカラムの値は 1 です。
user_type	SMALLINT	パラメータのユーザ型 (適用できる場合)。
parm_name	CHAR(128)	プロシージャ・パラメータの名前。
"default"	LONG VARCHAR	パラメータのデフォルト値。参照情報としてのみ表示されます。
remarks	LONG VARCHAR	常に NULL を返します。ODBC ドライバの旧バージョンを新しいパーソナル・データベース・サーバで使用できるようにするために用意されています。
base_type_str	VARCHAR(32767)	パラメータの物理的な型を表す注釈付きの型文字列。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (proc_id, parm_id)
FOREIGN KEY (proc_id) references SYS.ISYSPROCEDURE (proc_id)
FOREIGN KEY (domain_id) references SYS.ISYSDOMAIN (domain_id)
```

```
FOREIGN KEY (user_type) references SYS.ISYSUSERTYPE (type_id)
```

SYSPROCPARMS 統合ビュー

SYSPROCPARMS ビューの各ローは、データベース内のプロシージャに対するパラメータを示します。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSPROCPARMS"( creator,
  procname,parmname,parm_id,parmtype,parmmode,parmdomain,
  length,scale,"default",user_type )
  as select
up.user_name,p.proc_name,pp.parm_name,pp.parm_id,pp.parm_type,
    if pp.parm_mode_in = 'Y' and pp.parm_mode_out = 'N' then 'IN'
    else if pp.parm_mode_in = 'N' and pp.parm_mode_out = 'Y' then
'OUT'
      else 'INOUT'
      endif

endif,dom.domain_name,pp.width,pp.scale,pp."default",ut.type_name
    from SYS.SYSPROCPARM as pp
      join SYS.ISYSPROCEDURE as p on p.proc_id = pp.proc_id
      join SYS.ISYSUSER as up on up.user_id = p.creator
      join SYS.ISYSDOMAIN as dom on dom.domain_id = pp.domain_id
      left outer join SYS.ISYSUSERTYPE as ut on ut.type_id =
pp.user_type
```

SYSPROCPERM システム・ビュー

SYSPROCPERM システム・ビューの各ローは、あるプロシージャを実行するための、ユーザに付与されたパーミッションを示します。パーミッションを付与されたユーザだけがプロシージャを実行できます。このビューの基本となるシステム・テーブルは ISYSPROCPERM です。

カラム名	データ型	説明
proc_id	UNSIGNED INT	プロシージャ番号は、パーミッションが付与されたプロシージャをユニークに特定します。
grantee	UNSIGNED INT	パーミッションを受けるユーザのユーザ番号。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (proc_id, grantee)
```

```
FOREIGN KEY (grantee) references SYS.ISYSUSER (user_id)
```

```
FOREIGN KEY (proc_id) references SYS.ISYSPROCEDURE (proc_id)
```

SYSPROCS 統合ビュー

SYSPROCS ビューには、プロシージャ名または関数名と、そのプロシージャまたは関数について記録されている作成者名およびコメントが表示されます。

ビューを構成するテーブルとカラムは、以下の ALTER VIEW 文で示されます。

```
ALTER VIEW "SYS"."SYSPROCS"( creator,
  procname,remarks )
  as select u.user_name,p.proc_name,r.remarks
    from SYS.ISYSPROCEDURE as p
      join SYS.ISYSUSER as u on u.user_id = p.creator
      left outer join SYS.ISYSREMARK as r on(p.object_id =
r.object_id)
```

SYSPROXYTAB システム・ビュー

SYSPROXYTAB システム・ビューの各ローは、プロキシ・テーブルのリモート・パラメータを示します。このビューの基本となるシステム・テーブルは ISYSPROXYTAB です。

カラム名	データ型	説明
table_object_id	UNSIGNED BIGINT	プロキシ・テーブルのオブジェクト ID。
existing_obj	CHAR(1)	プロキシ・テーブルがリモート・サーバに以前に存在していたかどうかを示します。
srvid	UNSIGNED INT	プロキシ・テーブルと関連付けられたリモート・サーバのユニークな ID。
remote_location	LONG VARCHAR	リモート・サーバに関するプロキシ・テーブルの位置。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (table_object_id)
```

```
FOREIGN KEY (table_object_id) references ISYSOBJECT (object_id)
MATCH UNIQUE FULL
```

```
FOREIGN KEY (srvid) references SYS.ISYSSERVER (srvid)
```

SYSPUBLICATION システム・ビュー

SYSPUBLICATION システム・ビューの各ローはパブリケーションを示します。このビューの基本となるシステム・テーブルは ISYSPUBLICATION です。

カラム名	データ型	説明
publication_id	UNSIGNED INT	パブリケーションをユニークに識別する番号。
object_id	UNSIGNED BIGINT	データベースのパブリケーションをユニークに識別するプロシージャの内部 ID。
creator	UNSIGNED INT	パブリケーションの所有者。
publication_name	CHAR(128)	パブリケーションの名前。
remarks	LONG VARCHAR	パブリケーションに関する注記。ISYSREMARK システム・テーブル内に格納されている値。
type	CHAR(1)	このカラムは使用されません。
sync_type	UNSIGNED INT	パブリケーションの動機の種類。次のような値があります。 • **0 (ログスキャン)** – トランザクション・ログを使用して、最後のアップロード後に変更されたすべての関連データをアップロードする通常のパブリケーションです。 • **1 (スクリプト化されたアップロード)** – このパブリケーションの場合、トランザクション・ログは無視され、アップロードは格納済みプロシージャを使用してユーザが定義します。ストアド・プロシージャに関する情報は、ISYSSYNCSCRIPT システム・テーブルに格納されます。 • **2 (ダウンロード専用)** – これはダウンロードのみのパブリケーションです。データはアップロードされません。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (publication_id)
```

```
FOREIGN KEY (object_id) references SYS.ISYSOBJECT (object_id) MATCH
UNIQUE FULL
```

```
FOREIGN KEY (creator) references SYS.ISYSUSER (user_id)
```

```
UNIQUE Index (publication_name, creator)
```

SYSPUBLICATIONS 統合ビュー

SYSPUBLICATIONS ビューの各ローはパブリケーションを示します。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。+

```
ALTER VIEW "SYS"."SYSPUBLICATIONS"
  as select u.user_name as creator,
    p.publication_name,
    r.remarks,
    p.type,
    case p.sync_type
    when 0 then 'logscan'
    when 1 then 'scripted upload'
    when 2 then 'download only'
    else 'invalid'
    end as sync_type
    from SYS.ISYSPUBLICATION as p
      join SYS.ISYSUSER as u on u.user_id = p.creator
      left outer join SYS.ISYSREMARK as r on(p.object_id =
r.object_id)
```

SYSREMARK システム・ビュー

SYSREMARK システム・ビューの各ローは、オブジェクトの注釈 (コメント) を示します。このビューの基本となるシステム・テーブルは ISYSREMARK です。

カラム	データ型	説明
object_id	UNSIGNED BIGINT	関連する注釈があるオブジェクトの内部 ID。
remarks	LONG VARCHAR	オブジェクトと関連付けられた注釈またはコメント。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (object_id)
```

```
FOREIGN KEY (object_id) references SYS.ISYSOBJECT (object_id) MATCH
UNIQUE FULL
```

SYSREMOTEOPTION システム・ビュー

SYSREMOTEOPTION システム・ビューの各ローは、SQL Remote メッセージ・リンク・パラメータの値を示します。このビューの元となるシステム・テーブルは ISYSREMOTEOPTION です。

このビューには、一般に公開すべきではないデータが含まれている可能性のあるカラムもあります。そのため、このビューへのアクセスは DBA 権限を持つユーザに限定されます。SYSREMOTEOPTION2 ビューを使用すると、このビューのデー

タ (機密データを含む可能性のあるカラムを除く) にパブリック・アクセスできます。

SYSREMOTEOPTION ビューは SQL Anywhere のシステム・ビューです。詳細については、『SQL Anywhere サーバー - SQL リファレンス』の「ビュー」>「システムビュー」>「SYSREMOTEOPTION システムビュー」を参照してください。

注意： 参照先は SQL Anywhere のマニュアルです。

SYSREMOTEOPTION2 統合ビュー

機密データを含まない SYSREMOTEOPTION と SYSREMOTEOPTIONTYPE のカラムを、読みやすい形式で表示します。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSREMOTEOPTION2"
  as select ISYSREMOTEOPTION.option_id,
    ISYSREMOTEOPTION.user_id,
    SYS.HIDE_FROM_NON_DBA(ISYSREMOTEOPTION.setting) as setting
    from SYS.ISYSREMOTEOPTION
```

SYSREMOTEOPTIONS 統合ビュー

SYSREMOTEOPTIONS ビューの各ローは、メッセージ・リンク・パラメータの値に関する記述です。このビューの一部のカラムには、機密データが含まれている可能性があります。このため、このビューにアクセスできるのは DBA 権限を持つユーザに限られます。SYSREMOTEOPTION2 ビューを使用すると、機密データ以外のデータにパブリック・アクセスできます。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSREMOTEOPTIONS"
  as select srt.type_name,
    sup.user_name,
    srot."option",
    SYS.HIDE_FROM_NON_DBA(sro.setting) as setting
    from SYS.ISYSREMOTETYPE as srt
      ,SYS.ISYSREMOTEOPTIONTYPE as srot
      ,SYS.ISYSREMOTEOPTION as sro
      ,SYS.ISYSUSER as sup
    where srt.type_id = srot.type_id
    and srot.option_id = sro.option_id
    and sro.user_id = sup.user_id
```

SYSREMOTEOPTIONTYPE システム・ビュー

SYSREMOTEOPTIONTYPE システム・ビューの各ローは、メッセージ・リンク・パラメータの 1 つを示します。このビューの基本となるシステム・テーブルは ISYSREMOTEOPTIONTYPE です。

カラム	データ型	説明
option_id	UNSIGNED INT	メッセージ・リンク・パラメータの ID 番号。
type_id	SMALLINT	このパラメータを使用するメッセージ・タイプの ID 番号。
"option"	VARCHAR(128)	メッセージ・リンク・パラメータの名前。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (option_id)
```

```
FOREIGN KEY (type_id) references SYS.ISYSREMOTETYPE (type_id)
```

SYSREMOTETYPE システム・ビュー

SYSREMOTETYPE システム・ビューには、SQL Remote に関する情報があります。このビューの基本となるシステム・テーブルは ISYSREMOTETYPE です。

カラム名	データ型	説明
type_id	SMALLINT	SQL Remote によってサポートされているメッセージ・システムのうち、このユーザにメッセージを送信するために使用されているものを識別します。
object_id	UNSIGNED BIGINT	データベースのリモート・タイプをユニークに識別するインデックスの内部 ID。
type_name	CHAR(128)	SQL Remote がサポートするメッセージ・システムの名前。
publisher_address	LONG VARCHAR	リモート・データベース・パブリッシャのアドレス。
remarks	LONG VARCHAR	リモート・タイプに関する注釈。ISYSREMARK システム・テーブル内に格納されている値。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (type_id)
```

```
FOREIGN KEY (object_id) references SYS.ISYSOBJECT (object_id) MATCH
UNIQUE FULL
```

```
UNIQUE Index (type_name)
```

SYSREMOTETYPES 統合ビュー

SYSREMOTETYPES ビューの各ローは、パブリッシャ・アドレスを含め、メッセージ・タイプの 1 つを示します。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSREMOTETYPES"
  as select rt.type_id,rt.type_name,rt.publisher_address,rm.remarks
    from SYS.ISYSREMOTETYPE as rt
      left outer join SYS.ISYSREMARK as rm on(rt.object_id =
rm.object_id)
```

SYSREMOTEUSER システム・ビュー

SYSREMOTEUSER システム・ビューの各ローは、REMOTE パーミッションを持つユーザ ID (サブスクライバ) と、そのユーザとの間で送受信されたメッセージのステータスを表します。このビューの基本となるシステム・テーブルは ISYSREMOTEUSER です。

カラム名	データ型	説明
user_id	UNSIGNED INT	REMOTE パーミッションを持つユーザのユーザ番号。
consolidate	CHAR(1)	ユーザに CONSOLIDATE パーミッション (Y) か REMOTE パーミッション (N) が付与されていることを表します。
type_id	SMALLINT	メッセージ・システムのうち、このユーザにメッセージを送信するために使用されているものを識別します。
address	LONG VARCHAR	メッセージの送信先アドレス。このアドレスは address_type に正しく対応している必要があります。
frequency	CHAR(1)	メッセージを送信する頻度。
send_time	TIME	次にメッセージがこのユーザへ送信される時刻。
log_send	UNSIGNED BIGINT	メッセージは、log_send が log_sent よりも大きいサブスクライバだけに送信されます。
time_sent	TIMESTAMP	このサブスクライバに、最後にメッセージが送信された時刻。
log_sent	UNSIGNED BIGINT	最後に送信されたオペレーションのログ・オフセット。

カラム名	データ型	説明
confirm_sent	UNSIGNED BIGINT	このサブスクライバから最後に確認されたオペレーションに対する、ログ・オフセット。
send_count	INTEGER	メッセージが送信された回数。
resend_count	INTEGER	メッセージがサブスクライバ・データベースに、一度だけ適用されたことを確認するためのカウンタ。
time_received	TIMESTAMP	このサブスクライバから、最後にメッセージが受信された時刻。
log_received	UNSIGNED BIGINT	現在のデータベースで最後に受信されたオペレーションの、サブスクライバ・データベースのログ・オフセット。
confirm_received	UNSIGNED BIGINT	最後に確認メッセージが送信されたオペレーションに対する、サブスクライバ・データベース内のログ・オフセット。
receive_count	INTEGER	メッセージが受信された回数。
rereceive_count	INTEGER	メッセージが現在のデータベースに、一度だけ適用されたことを確認するためのカウンタ。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (user_id)
```

```
FOREIGN KEY (user_id) references SYS.ISYSUSER (user_id)
```

```
FOREIGN KEY (type_id) references SYS.ISYSREMOTETYPE (type_id)
```

```
UNIQUE Index (type_id, address)
```

SYSREMOTEUSERS 統合ビュー

SYSREMOTEUSERS ビューの各ローは、REMOTE パーミッションを持つユーザ ID (サブスクライバ) と、そのユーザとの間で送受信されたメッセージのステータスを表します。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSREMOTEUSERS"
  as select
u.user_name,r.consolidate,t.type_name,r.address,r.frequency,
    r.send_time,
    (if r.frequency = 'A' then null else if r.frequency = 'P' then
        if r.time_sent is null then current timestamp
        else(select min(minutes(a.time_sent,60*hour(a.send_time)
            +minute(seconds(a.send_time,59))))
```

```
            from SYS.ISYSREMOTEUSER as a where a.frequency = 'P'
            and a.send_time = r.send_time)
        endif
      else if current date+r.send_time
         > coalesce(r.time_sent,current timestamp) then
          current date+r.send_time else current date+r.send_time+1
endif
      endif endif) as next_send,
    r.log_send,r.time_sent,r.log_sent,r.confirm_sent,r.send_count,
    r.resend_count,r.time_received,r.log_received,
    r.confirm_received,r.receive_count,r.rereceive_count
    from SYS.ISYSREMOTEUSER as r
      join SYS.ISYSUSER as u on(u.user_id = r.user_id)
      join SYS.ISYSREMOTETYPE as t on(t.type_id = r.type_id)
```

SYSSCHEDULE システム・ビュー

SYSSCHEDULE システム・ビューの各ローは、CREATE EVENT 文の SCHEDULE 句に指定されたイベントの起動時刻を示します。

このビューの元となるシステム・テーブルは ISYSSCHEDULE です。

SYSSCHEDULE ビューは SQL Anywhere のシステム・ビューです。詳細については、『SQL Anywhere サーバー - SQL リファレンス』の「ビュー」>「システムビュー」>「SYSSCHEDULE システムビュー」を参照してください。

注意： 参照先は SQL Anywhere のマニュアルです。

SYSSERVER システム・ビュー

SYSSERVER システム・ビューの各ローは、リモート・サーバを示します。このビューの基本となるシステム・テーブルは ISYSSERVER です。

注意： SYSSERVERS システム・テーブルに含まれる前のカタログ・バージョン。このテーブル名は ISYSSERVER ('S' なし) に変更され、このビューの基本となるテーブルになります。

カラム名	データ型	説明
srvid	UNSIGNED INT	リモート・サーバの識別子。
srvname	VARCHAR(128)	リモート・サーバの名前。
srvclass	LONG VARCHAR	CREATE SERVER 文で指定されたサーバ・クラス。
srvinfo	LONG VARCHAR	サーバ情報。
srvreadonly	CHAR(1)	サーバが読み込み専用かどうか。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (srvid)
```

SYSSOURCE システム・ビュー

SYSSOURCE システム・ビューの各ローには、SYSOBJECT システム・ビューにリストされているオブジェクトのソース・コード (適用できる場合) が含まれます。このビューの基本となるシステム・テーブルは ISYSSOURCE です。

カラム名	データ型	説明
object_id	UNSIGNED BIGINT	ソース・コードが定義されているオブジェクトの内部 ID。
source	LONG VARCHAR	オブジェクトを作成したときに preserve_source_format データベース・オプションが On の場合、このカラムにはオブジェクトの元のソース・コードが含まれます。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (object_id)
```

```
FOREIGN KEY (object_id) references SYS.ISYSOBJECT (object_id) MATCH
UNIQUE FULL
```

SYSSQLSERVERTYPE システム・ビュー

SYSSQLSERVERTYPE システム・ビューには、Adaptive Server Enterprise との互換性に関する情報が含まれます。このビューの基本となるシステム・テーブルは ISYSSQLSERVERTYPE です。

カラム名	データ型	説明
ss_user_type	SMALLINT	Adaptive Server Enterprise ユーザ型。
ss_domain_id	SMALLINT	Adaptive Server Enterprise のドメイン ID。
ss_type_name	VARCHAR(30)	Adaptive Server Enterprise 型名。
primary_sa_domain_id	SMALLINT	対応する SQL Anywhere のプライマリ・ドメイン ID。
primary_sa_user_type	SMALLINT	対応する SQL Anywhere のプライマリ・ユーザ・タイプ。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (ss_user_type)
```

SYSSUBPARTITIONKEY システム・ビュー

このビューは、今後のリリースで使用される予定です。Sybase IQ15.4 はサブパーティションをサポートしません。

SYSSUBSCRIPTION システム・ビュー

SYSSUBSCRIPTION システム・ビューの各ローは、REMOTE パーミッションを持つあるユーザ ID からの、あるパブリケーションに対するサブスクリプションを示します。このビューの基本となるシステム・テーブルは ISYSSUBSCRIPTION です。

カラム名	データ型	説明
publication_id	UNSIGNED INT	ユーザ ID のサブスクリプションが作成されているパブリケーションの識別子。
user_id	UNSIGNED INT	パブリケーションに対して送信されたユーザの ID。
subscribe_by	CHAR(128)	サブスクリプション用の SUBSCRIBE BY 式がある場合、その値。
created	UNSIGNED BIGINT	トランザクション・ログ内の、サブスクリプションが作成されたオフセット。
started	UNSIGNED BIGINT	トランザクション・ログ内の、サブスクリプションが開始されたオフセット。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (publication_id, user_id, subscribe_by)
```

```
FOREIGN KEY (publication_id) references SYS.ISYSPUBLICATION
(publication_id)
```

```
FOREIGN KEY (user_id) references SYS.ISYSUSER (user_id)
```

SYSSUBSCRIPTIONS 統合ビュー

REMOTE パーミッションを持つあるユーザ ID からの、あるパブリケーションに対するサブスクリプションについて、各ローで示します。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSSUBSCRIPTIONS"
  as select p.publication_name,u.user_name,s.subscribe_by,s.created,
    s.started
    from SYS.ISYSSUBSCRIPTION as s
      join SYS.ISYSPUBLICATION as p on(p.publication_id =
s.publication_id)
      join SYS.ISYSUSER as u on u.user_id = s.user_id
```

SYSSYNC システム・ビュー

SYSSYNC システム・ビューには、同期に関する情報が含まれます。このビューの一部のカラムには、機密データが含まれている可能性があります。このため、このビューにアクセスできるのは DBA 権限を持つユーザに限られます。SYSSYNC2 ビューを使用すると、機密データを含む可能性のあるカラムを除いて、このビューのデータにパブリック・アクセスできます。このビューの基本となるシステム・テーブルは ISYSSYNC です。

カラム名	データ型	説明
sync_id	UNSIGNED INT	ユニークにローを識別する番号。
type	CHAR(1)	値は常に D です。
publication_id	UNSIGNED INT	SYSPUBLICATION システム・ビューに入っている publication_id。
progress	UNSIGNED BIGINT	最後に成功したアップロードのログ・オフセット。
site_name	CHAR(128)	ユーザ名。
"option"	LONG VARCHAR	同期オプション。
server_connect	LONG VARCHAR	サーバのアドレスまたは URL。
server_conn_type	LONG VARCHAR	同期時に使用する、TCP/IP などの通信プロトコル。
last_download_time	TIMESTAMP	最後にサーバからダウンロード・ストリームを受信した時刻。
last_upload_time	TIMESTAMP	最後に情報のアップロードが成功した時刻 (サーバで測定)。デフォルトは jan-1-1900 です。
created	UNSIGNED BIGINT	サブスクリプションが作成されたログ・オフセット。
log_sent	UNSIGNED BIGINT	情報をどこまでアップロードしたかを示すログの進行状況。更新されるこのカラムのエントリに対するアップロード確認の受信は必要ありません。
generation_number	INTEGER	ファイルベースのダウンロードで、このサブスクリプションに対して最後に受信した世代番号。デフォルトは 0 です。
extended_state	VARCHAR(1024)	内部でのみ使用。

カラム名	データ型	説明
script_version	CHAR(128)	CREATE SYNCHRONIZATION SUBSCRIPTION 文、ALTER SYNCHRONIZATION SUBSCRIPTION 文、および START SYNCHRONIZATION SCHEMA CHANGE 文によって使用されるスクリプト・バージョンを示します。
subscription_name	CHAR(128)	サブスクリプションの名前。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (sync_id)
```

```
FOREIGN KEY (publication_id) references SYS.ISYSPUBLICATION
(publication_id)
```

```
UNIQUE Index (publication_id, site_name)
```

```
UNIQUE Index (subscription_name)
```

SYSSYNC2 統合ビュー

SYSSYNC2 ビューは、含まれている可能性がある機密データを除き、SYSSYNC システム・ビューにあるデータ (同期に関する情報) へのパブリック・アクセスを提供します。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSSYNC2"
  as select ISYSSYNC.sync_id,
    ISYSSYNC.type,
    ISYSSYNC.publication_id,
    ISYSSYNC.progress,
    ISYSSYNC.site_name,
    SYS.HIDE_FROM_NON_DBA(ISYSSYNC."option") as "option",
    SYS.HIDE_FROM_NON_DBA(ISYSSYNC.server_connect) as
server_connect,
    ISYSSYNC.server_conn_type,
    ISYSSYNC.last_download_time,
    ISYSSYNC.last_upload_time,
    ISYSSYNC.created,
    ISYSSYNC.log_sent,
    ISYSSYNC.generation_number,
    ISYSSYNC.extended_state,
    ISYSSYNC.script_version,
    ISYSSYNC.subscription_name
    from SYS.ISYSSYNC
```

SYSSYNCPUBLICATIONDEFAULTS 統合ビュー

SYSSYNCPUBLICATIONDEFAULTS は、同期に関連するパブリケーションに関連付けられたデフォルトの同期設定のビューです。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSSYNCPUBLICATIONDEFAULTS"
  as select s.sync_id,
    p.publication_name,
    SYS.HIDE_FROM_NON_DBA(s."option") as "option",
    SYS.HIDE_FROM_NON_DBA(s.server_connect) as server_connect,
    s.server_conn_type
    from SYS.ISYSSYNC as s join SYS.ISYSPUBLICATION as p
on(p.publication_id = s.publication_id) where
    s.site_name is null
```

SYSSYNCS 統合ビュー

SYSSYNCS ビューには、同期に関する情報が入っています。このビューの一部のカラムには、機密データが含まれている可能性があります。このため、このビューにアクセスできるのは DBA 権限を持つユーザに限られます。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSSYNCS"
  as select p.publication_name,s.progress,s.site_name,
    SYS.HIDE_FROM_NON_DBA(s."option") as "option",
    SYS.HIDE_FROM_NON_DBA(s.server_connect) as server_connect,
    s.server_conn_type,s.last_download_time,
    s.last_upload_time,s.created,s.log_sent,s.generation_number,
    s.extended_state
    from SYS.ISYSSYNC as s
      left outer join SYS.ISYSPUBLICATION as p
      on p.publication_id = s.publication_id
```

SYSSYNCSCRIPT システム・ビュー

SYSSYNCSCRIPT システム・ビューの各ローは、スクリプト化されたアップロードに関するストアド・プロシージャを識別します。このビューは、SYSSYNCSCRIPTS ビューとほとんど同じですが、このビューの値は未加工形式である点が異なります。

このビューの基本となるシステム・テーブルは ISYSSYNCSCRIPT です。

カラム名	データ型	説明
pub_object_id	UNSIGNED BIGINT	スクリプトが所属するパブリケーションのオブジェクト ID。
table_object_id	UNSIGNED BIGINT	スクリプトの適用対象となるテーブルのオブジェクト ID。
type	UNSIGNED INT	アップロード・プロシージャのタイプ。
proc_object_id	UNSIGNED BIGINT	パブリケーションに使用するストアド・プロシージャのオブジェクト ID。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (pub_object_id, table_object_id, type)
FOREIGN KEY (pub_object_id) references SYS.ISYSOBJECT (object_id)
FOREIGN KEY (table_object_id) references SYS.ISYSOBJECT (object_id)
FOREIGN KEY (proc_object_id) references SYS.ISYSOBJECT (object_id)
```

SYSSYNCSCRIPTS 統合ビュー

SYSSYNCSCRIPTS ビューの各ローは、スクリプト化されたアップロードに関するストアド・プロシージャを識別します。このビューは、SYSSYNCSCRIPT システム・ビューとほとんど同じですが、データが未加工形式ではなく、ユーザが読みやすい形式で表示される点が異なります。

```
ALTER VIEW "SYS"."SYSSYNCSCRIPTS"
  as select p.publication_name,
    t.table_name,
    case s.type
    when 0 then 'upload insert'
    when 1 then 'upload delete'
    when 2 then 'upload update'
    else 'unknown'
    end as type,
    c.proc_name
    from SYS.ISYSSYNCSCRIPT as s
      join SYS.ISYSPUBLICATION as p on p.object_id = s.pub_object_id
      join SYS.ISYSTAB as t on t.object_id = s.table_object_id
      join SYS.ISYSPROCEDURE as c on c.object_id = s.proc_object_id
```

SYSSYNCSUBSCRIPTIONS 統合ビュー

SYSSYNCSUBSCRIPTIONS ビューには、同期のサブスクリプションと関連付けられた同期設定が含まれます。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSSYNCSUBSCRIPTIONS"
  as select s.sync_id,
    p.publication_name,
    s.progress,
    s.site_name,
    SYS.HIDE_FROM_NON_DBA(s."option") as "option",
    SYS.HIDE_FROM_NON_DBA(s.server_connect) as server_connect,
    s.server_conn_type,
    s.last_download_time,
    s.last_upload_time,
    s.created,
    s.log_sent,
    s.generation_number,
    s.extended_state
    from SYS.ISYSSYNC as s join SYS.ISYSPUBLICATION as p
on(p.publication_id = s.publication_id)
    where s.publication_id is not null and
    s.site_name is not null and exists
    (select 1 from SYS.SYSSYNCUSERS as u
      where s.site_name = u.site_name)
```

SYSSYNCUSERS 統合ビュー

同期ユーザに関連付けられた同期設定のビュー。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSSYNCUSERS"
  as select ISYSSYNC.sync_id,
    ISYSSYNC.site_name,
    SYS.HIDE_FROM_NON_DBA(ISYSSYNC."option") as "option",
    SYS.HIDE_FROM_NON_DBA(ISYSSYNC.server_connect) as
server_connect,
    ISYSSYNC.server_conn_type
    from SYS.ISYSSYNC where
    ISYSSYNC.publication_id is null
```

SYSTAB システム・ビュー

SYSTAB システム・ビューの各ローは、データベース内のテーブル 1 つを示します。ビューの追加情報が SYSVIEW システム・ビューにあります。このビューの基本となるシステム・テーブルは ISYSTAB です。

カラム名	データ型	説明
table_id	UNSIGNED INT	各テーブルにはユニークな番号 (テーブル番号) が割り当てられます。
dbspace_id	SMALLINT	テーブルを含む DB 領域を示す値。

カラム名	データ型	説明
count	UNSIGNED BIGINT	テーブルまたはマテリアライズド・ビュー内のロー数。この値は、各チェックポイントが正常処理されているときに更新されます。この数値を使用してデータベース・アクセスの最適化が行われます。非マテリアライズド・ビューの場合、またはリモート・テーブルの場合、カウントは常に 0 です。
creator	UNSIGNED INT	このユーザ番号はテーブルまたはビューの所有者を示します。
table_page_count	INTEGER	基本となるテーブルで使用されるメイン・ページの総数。
ext_page_count	INTEGER	このテーブルで使用される拡張ページの総数。
commit_action	INTEGER	グローバル・テンポラリ・テーブルの場合、0 は、テーブルの作成時に ON COMMIT PRESERVE ROWS 句が指定されたことを示します。1 は、テーブルの作成時に ON COMMIT DELETE ROWS 句が指定されたことを示します (テンポラリ・テーブルのデフォルト動作)。3 は、テーブルの作成時に NOT TRANSACTIONAL 句が指定されたことを示します。非テンポラリ・テーブルの場合、commit_action は常に 0 です。
share_type	INTEGER	グローバル・テンポラリ・テーブルの場合、4 は、テーブルの作成時に SHARE BY ALL 句が指定されたことを示します。5 は、テーブルの作成時に SHARE BY ALL 句が指定されなかったことを示します。非テンポラリ・テーブルの場合、share_type は常に 5 です。これは、非テンポラリ・テーブルの作成時に SHARE BY ALL 句は指定できないためです。
object_id	UNSIGNED BIGINT	テーブルのオブジェクト ID。
last_modified_at	TIMESTAMP	テーブルが最後に修正された日時。このカラムは、チェックポイント時間にのみ更新されます。
last_modified_tsn	UNSIGNED BIGINT	テーブルを修正したトランザクションに割り当てられたシーケンス番号。
file_id	SMALLINT	廃止予定。このカラムは SYSVIEW に存在しますが、基本となるシステム・テーブル ISYSTAB には存在しません。このカラムの内容は dbspace_id と同じであり、互換性のために提供されています。今後は dbspace_id を使用してください。

カラム名	データ型	説明
table_name	CHAR(128)	テーブルまたはビューの名前。1 人の作成者が、同じ名前のテーブルまたはビューを 2 つ以上持つことはできません。
table_type	TINYINT	テーブルまたはビューのタイプ。次のような値があります。 • **1 –** ベース・テーブル • **2 –** マテリアライズド・ビュー • **3 –** グローバル・テンポラリ・テーブル • **4 –** ローカル・テンポラリ・テーブル • **5 –** テキスト・インデックス・ベース・テーブル • **6 –** テキスト・インデックス・グローバル・テンポラリ・テーブル • **21 –** ビュー
replicate	CHAR(1)	この値は内部でのみ使用されます。
server_type	TINYINT	基本となるテーブルのデータの場所。次のような値があります。 • **1 –** ローカル・サーバ • **3 –** リモート・サーバ
tab_page_list	LONG VARBIT	内部でのみ使用。テーブルの情報を含むページ・セット (ビットマップで表示されます)。
ext_page_list	LONG VARBIT	内部でのみ使用。テーブルに関するローの拡張と大規模なオブジェクト (LOB) ページを含むページ・セット (ビットマップで表示されます)。
pct_free	UNSIGNED INT	指定されている場合、テーブルの PCT_FREE の指定、それ以外の場合は NULL。
clustered_index_id	UNSIGNED INT	テーブルのクラスタ化されたインデックスの ID。クラスタ化されたインデックスがない場合、このフィールドは NULL です。
encrypted	CHAR(1)	テーブルまたはマテリアライズド・ビューが暗号化されるかどうか。

カラム名	データ型	説明
table_type_str	CHAR(9)	table_type の読み取り可能な値。次のような値があります。 • **BASE** – ベース・テーブル • **MAT VIEW** – マテリアライズド・ビュー • **GBL TEMP** – グローバル・テンポラリ・テーブル • **VIEW** – ビュー

基本となるシステム・テーブルに関する制約

```
FOREIGN KEY (dbspace_id) references SYS.ISYSDBSPACE (dbspace_id)
FOREIGN KEY (object_id) references SYS.ISYSOBJECT (object_id)
PRIMARY KEY (table_id)
FOREIGN KEY (creator) references SYS.ISYSUSER (user_id)
UNIQUE Index (table_name, creator)
```

SYSTABAUTH 統合ビュー

SYSTABAUTH ビューには、SYSTABLEPERM システム・ビューの情報が表示されますが、より読みやすい形式です。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSTABAUTH"( grantor,
  grantee,screator,stname,tcreator,ttname,
  selectauth,insertauth,deleteauth,
  updateauth,updatecols,alterauth,referenceauth )
  as select u1.user_name,u2.user_name,u3.user_name,tab1.table_name,
    u4.user_name,tab2.table_name,tp.selectauth,tp.insertauth,
    tp.deleteauth,tp.updateauth,tp.updatecols,tp.alterauth,
    tp.referenceauth
    from SYS.ISYSTABLEPERM as tp
      join SYS.ISYSUSER as u1 on u1.user_id = tp.grantor
      join SYS.ISYSUSER as u2 on u2.user_id = tp.grantee
      join SYS.ISYSTAB as tab1 on tab1.table_id = tp.stable_id
      join SYS.ISYSUSER as u3 on u3.user_id = tab1.creator
      join SYS.ISYSTAB as tab2 on tab2.table_id = tp.stable_id
      join SYS.ISYSUSER as u4 on u4.user_id = tab2.creator
```

SYSTABCOL システム・ビュー

SYSTABCOL システム・ビューには、データベースの各テーブルとビューの各カラムのローが含まれます。このビューの基本となるシステム・テーブルは ISYSTABCOL です。

カラム名	データ型	説明
table_id	UNSIGNED INT	カラムが属するテーブルまたはビューのオブジェクト ID。
column_id	UNSIGNED INT	カラムの ID。各テーブルについて、カラムの番号は 1 から開始されます。 column_id 値によって、SELECT * が使用された場合の結果セットのカラムの順序が決定されます。また、INSERT 文にカラム名のリストが指定されない場合も、この値によってカラム順が決定されます。
domain_id	SMALLINT	カラムのデータ型を、SYSDOMAIN システム・ビューにリストされているデータ型番号で示します。
nulls	CHAR(1)	NULL 値をカラムに許可するかどうかを指定します。
width	UNSIGNED INT	文字列カラムでは長さ、数値カラムでは精度、その他のデータ型では格納サイズをバイトで示します。
scale	SMALLINT	NUMERIC または DECIMAL データ型のカラムについて、小数点以下の桁数。文字列のカラムの場合、1 の値は文字長のセマンティックを指定します。0 は、バイト長のセマンティックを指定します。
object_id	UNSIGNED BIGINT	テーブル・カラムのオブジェクト ID。
max_identity	BIGINT	AUTOINCREMENT、IDENTITY、または GLOBAL AUTOINCREMENT カラムの場合、カラムの最大値。
カラム名	CHAR(128)	カラム名。
"default"	LONG VARCHAR	カラムのデフォルト値。この値を指定した場合、INSERT 文が値を指定しないときのみ使われます。
user_type	SMALLINT	ユーザ定義のデータ型を使用してカラムが定義される場合、データ型。
column_type	CHAR(1)	カラムのタイプ (C=計算済みカラム、R=他のカラム)。
compressed	TINYINT	このカラムを圧縮形式で格納するかどうか。

カラム名	データ型	説明
collect_stats	TINYINT	システムが自動的にカラムの統計情報を収集および更新するかどうか。
inline_max	SMALLINT	ローに格納する BLOB の最大バイト数。NULL 値は、デフォルトが適用されたこと、またはカラムは文字型またはバイナリ型ではないことを示します。NULL ではない inline_max 値は、CREATE TABLE 文または ALTER TABLE 文を使用してカラムに指定した INLINE 値に対応します。
inline_long	SMALLINT	BLOB サイズが inline_max 値を超えた場合、ローに格納されている BLOB のバイトを複製した数。NULL 値は、デフォルトが適用されたこと、またはカラムは文字型またはバイナリ型ではないことを示します。NULL ではない inline_long 値は、CREATE TABLE 文または ALTER TABLE 文を使用してカラムに指定した PREFIX 値に対応します。
lob_index	TINYINT	内部的なしきい値サイズ (約 8 データベース・ページ) を超過するカラムの BLOB 値に関するインデックスを構築するかどうか。NULL 値は、デフォルトを適用するか、カラムが BLOB タイプであることを示します。1 の値は、インデックスを構築することを示します。0 の値は、インデックスを構築しないことを示します。NULL ではない lob_index 値は、CREATE TABLE 文または ALTER TABLE 文を使用してカラムに指定した INDEX または NO INDEX 値に対応します。
base_type_str	VARCHAR(32,767)	カラムの物理的な型を表す注釈付きの型文字列。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (table_id, column_id)
```

```
FOREIGN KEY (table_id) references SYS.ISYSTAB (table_id)
```

```
FOREIGN KEY (domain_id) references SYS.ISYSDOMAIN (domain_id)
```

```
FOREIGN KEY (object_id) references SYS.ISYSOBJECT (object_id) MATCH
UNIQUE FULL
```

```
FOREIGN KEY (user_type) references SYS.ISYSUSERTYPE (type_id)
```

SYSTABLE 互換ビュー (旧式)

SYSTABLE ビューは、SYSTABLE システム・テーブルを提供していた古いバージョンのソフトウェアとの互換性を保つために用意されています。ただし、以前

のSYSTABLEテーブルは、SYSTABシステム・ビューに対応するISYSTABシステム・テーブルで置換されたため、SYSTABの使用をおすすめします。

SYSTABLEビューの各ローは、データベース内のテーブル1つを示します。

ビューを構成するテーブルとカラムは、以下のSQL文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSTABLE"
  as select b.table_id,
    b.file_id,
    b.count,
    0 as first_page,
    b.commit_action as last_page,
    COALESCE(ph.root,0) as primary_root,
    b.creator,
    0 as first_ext_page,
    0 as last_ext_page,
    b.table_page_count,
    b.ext_page_count,
    b.object_id,
    b.table_name,
    b.table_type_str as table_type,
    v.view_def,
    r.remarks,
    b.replicate,
    p.existing_obj,
    p.remote_location,
    'T' as remote_objtype,
    p.srvid,
    case b.server_type
    when 1 then 'SA'
    when 2 then 'IQ'
    when 3 then 'OMNI'
    else 'INVALID'
    end as server_type,
    10 as primary_hash_limit,
    0 as page_map_start,
    s.source,
    b."encrypted"
    from SYS.SYSTAB as b
      left outer join SYS.ISYSREMARK as r on(b.object_id =
r.object_id)
      left outer join SYS.ISYSSOURCE as s on(b.object_id =
s.object_id)
      left outer join SYS.ISYSVIEW as v on(b.object_id =
v.view_object_id)
      left outer join SYS.ISYSPROXYTAB as p on(b.object_id =
p.table_object_id)
     left outer join(SYS.ISYSIDX as i left outer join SYS.ISYSPHYSIDX
as ph on(i.table_id = ph.table_id and i.phys_index_id =
ph.phys_index_id))
          on(b.table_id = i.table_id and i.index_category = 1 and
i.index_id = 0)
```

SYSTABLEPERM システム・ビュー

GRANT 文によってパーミッションが与えられると、SYSTABLEPERM システム・ビューに格納されます。このビューの各ローは、1 つのテーブル、パーミッションを与えるユーザ ID (grantor)、そしてパーミッションを与えられるユーザ ID (grantee) に対応します。このビューの基本となるシステム・テーブルは ISYSTABLEPERM です。

カラム名	データ型	説明
stable_id	UNSIGNED INT	パーミッションが適用されるテーブルまたはビューのテーブル番号。
grantee	UNSIGNED INT	パーミッションを受けるユーザ ID のユーザ番号。
grantor	UNSIGNED INT	パーミッションを付与するユーザ ID のユーザ番号。
selectauth	CHAR(1)	SELECT パーミッションが付与されたかどうかを示します。可能な値は Y、N、または G です。これらの値の詳細については、以下の備考部分を参照してください。
insertauth	CHAR(1)	INSERT パーミッションが付与されたかどうかを示します。可能な値は Y、N、または G です。これらの値の詳細については、以下の備考部分を参照してください。
deleteauth	CHAR(1)	DELETE パーミッションが付与されたかどうかを示します。可能な値は Y、N、または G です。これらの値の詳細については、以下の備考部分を参照してください。
updateauth	CHAR(1)	UPDATE パーミッションが、テーブル内のすべてのカラムに付与されたかどうかを示します。可能な値は Y、N、または G です。これらの値の詳細については、以下の備考部分を参照してください。
updatecols	CHAR(1)	UPDATE パーミッションが、基本となるテーブル内の一部のカラムだけに付与されたかどうかを示します。updatecols が "Y" であれば、カラムに UPDATE パーミッションを与える 1 つまたは複数のローが、SYSCOLPERM システム・ビューにあります。
alterauth	CHAR(1)	ALTER パーミッションが付与されたかどうかを示します。可能な値は Y、N、または G です。これらの値の詳細については、以下の備考部分を参照してください。

カラム名	データ型	説明
referenceauth	CHAR(1)	REFERENCE パーミッションが付与されたかどうかを示します。可能な値は Y、N、または G です。これらの値の詳細については、以下の備考部分を参照してください。

備考
与えられるパーミッションには型がいくつかあります。各パーミッションは、次の 3 つのうち 1 つの値を持ちます。

- **N –** No。grantee はこのパーミッションを grantor から付与されていません。
- **Y –** Yes。grantee はこのパーミッションを grantor から付与されています。
- **G –** grantee はこのパーミッションを受けています。また、grantee は同じパーミッションを他のユーザに付与できます。

注意： grantee は同じテーブルに関するパーミッションを、他の grantor から受けることがあります。その場合、この情報は、SYSTABLEPERM システム・ビューの異なるローで見つかります。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (stable_id, grantee, grantor)
FOREIGN KEY (stable_id) references SYS.ISYSTAB (table_id)
FOREIGN KEY (grantor) references SYS.ISYSUSER (user_id)
FOREIGN KEY (grantee) references SYS.ISYSUSER (user_id)
```

SYSTEXTCONFIG システム・ビュー

SYSTEXTCONFIG システム・ビューの各ローは、全文検索機能で使用するテキスト設定オブジェクト 1 つを示します。このビューの基本となるシステム・テーブルは ISYSTEXTCONFIG です。

カラム名	データ型	説明
object_id	UNSIGNED BIGINT	テキスト設定オブジェクトのオブジェクト ID。
creator	UNSIGNED INT	テキスト設定オブジェクトの作成者。

カラム名	データ型	説明
term_breaker	TINYINT	文字列を単語に分割するために使用されるアルゴリズム。値は GENERIC の場合は 0、NGRAM の場合は 1 です。GENERIC の場合、英数字以外の文字で区切られた 1 つまたは複数の英数字の文字列は、単語として扱われます。NGRAM は近似一致または単語の区切りにホワイトスペースを使用しないドキュメントに使用します。
stemmer	TINYINT	内部でのみ使用。
min_term_length	TINYINT	1 つの単語に許容される最小文字数。min_term_length より短い単語は無視されます。 MINIMUM TERM LENGTH 設定は、GENERIC 単語区切りの場合にのみ意味を持ちます。NGRAM テキスト・インデックスの場合、この設定は無視されます。
max_term_length	TINYINT	GENERIC テキスト・インデックスの場合は、1 つの単語に許容される最大長 (文字数)。max_term_length より長い単語は無視されます。 NGRAM テキスト・インデックスの場合は、単語が分解される N-gram の長さ。
collation	CHAR(128)	内部でのみ使用。
text_config_name	CHAR(128)	テキスト設定オブジェクトの名前。
prefilter	LONG VARCHAR	外部事前フィルタ・ライブラリの関数名およびライブラリ名です。
postfilter	LONG VARCHAR	内部でのみ使用。
char_stoplist	LONG VARCHAR	CHAR カラムでの全文検索実行時に無視する単語。これらの単語は、テキスト・インデックスからも省略されます。このカラムは、テキスト設定オブジェクトが default_char から作成されるときに使用されます。
nchar_stoplist	LONG NVARCHAR	NCHAR カラムでの全文検索実行時に無視する単語。これらの単語は、テキスト・インデックスからも省略されます。このカラムは、テキスト設定オブジェクトが default_nchar から作成されるときに使用されます。
external_term_breaker	LONG VARCHAR	外部単語区切りライブラリの関数名およびライブラリ名です。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (object_id)
```

```
FOREIGN KEY (object_id) references SYS.ISYSOBJECT (object_id ) MATCH
UNIQUE FULL
```

```
FOREIGN KEY (creator) references SYS.ISYSUSER (user_id)
```

```
UNIQUE Index (creator, text_config_name)
```

SYSTEXTIDX システム・ビュー

SYSTEXTIDX システム・ビューの各ローは、1 つのテキスト・インデックスを示します。このビューの基本となるシステム・テーブルは ISYSTEXTIDX です。

カラム名	データ型	説明
index_id	UNSIGNED BIGINT	SYSIDX のテキスト・インデックスのオブジェクト ID。
sequence	UNSIGNED INT	内部でのみ使用。
status	UNSIGNED INT	内部でのみ使用。
text_config	UNSIGNED BIGINT	SYSTEXTCONFIG のテキスト設定オブジェクトのオブジェクト ID。
next_handle	UNSIGNED INT	内部でのみ使用。
last_handle	UNSIGNED INT	内部でのみ使用。
deleted_length	UNSIGNED BIGINT	テキスト・インデックスの削除されたインデックス付きの値の合計サイズ。
pending_length	UNSIGNED BIGINT	次回のリフレッシュ時にテキスト・インデックスに追加されるインデックス付きの値の合計サイズ。
refresh_type	TINYINT	再表示タイプ。次のいずれかになります。 • **1 –** MANUAL • **2 –** AUTO • **3 –** IMMEDIATE
refresh_interval	UNSIGNED INT	自動リフレッシュの間隔 (分単位)。
last_refresh	TIMESTAMP	最後に行ったリフレッシュの時刻。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (index_id, sequence)
```

```
FOREIGN KEY (index_id) references SYS.ISYSOBJECT (object_id)
```

```
FOREIGN KEY (text_config) references SYS.ISYSTEXTCONFIG (object_id)
```

SYSTEXTIDXTAB システム・ビュー

SYSTEXTIDXTAB システム・ビューの各ローは、テキスト・インデックスの一部である生成されたテーブルを示します。このビューの基本となるシステム・テーブルは ISYSTEXTIDXTAB です。

カラム名	データ型	説明
index_id	UNSIGNED BIGINT	内部でのみ使用。
sequence	UNSIGNED INT	内部でのみ使用。
table_type	UNSIGNED INT	内部でのみ使用。
table_id	UNSIGNED INT	内部でのみ使用。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (index_id, sequence, table_type)
```

```
FOREIGN KEY (index_id, sequence) references SYS.ISYSTEXTIDX
(index_id, sequence)
```

```
FOREIGN KEY (table_id) references SYS.ISYSTAB (table_id)
```

SYSTRIGGER システム・ビュー

SYSTRIGGER システム・ビューの各ローは、データベース内のトリガ 1 つを示します。このテーブルには、参照トリガ・アクションを持つ外部キー定義に、自動的に作成されるトリガも含まれます (たとえば、ON DELETE CASCADE)。このビューの基本となるシステム・テーブルは ISYSTRIGGER です。

カラム名	データ型	説明
trigger_id	UNSIGNED INT	SYSTRIGGER ビュー内のトリガのユニークな番号です。
table_id	UNSIGNED INT	トリガが所属するテーブルのテーブル ID。
object_id	UNSIGNED BIGINT	データベース内のトリガのオブジェクト ID です。

カラム名	データ型	説明
イベント	CHAR(1)	トリガが起動されるきっかけとなる操作。 • **A –** INSERT、DELETE • **B –** INSERT、UPDATE • **C –** UPDATE COLUMNS • **D –** DELETE • **E –** DELETE、UPDATE • **I –** INSERT • **M –** INSERT、DELETE、UPDATE • **U –** UPDATE
trigger_time	CHAR(1)	イベントに関連するトリガが起動されるタイミング。 • **A –** AFTER (ローレベルのトリガ) • **B –** BEFORE (ローレベルのトリガ) • **I –** INSTEAD OF (ローレベルのトリガ) • **K –** INSTEAD OF (文レベルのトリガ) • **R –** RESOLVE • **S –** AFTER (文レベルのトリガ)
trigger_order	SMALLINT	同じタイプ (insert、update、または delete) の複数のトリガが、同じ時刻 (BEFORE または AFTER トリガのみが該当) に起動するように設定されている場合に、トリガが起動される順序。
foreign_table_id	UNSIGNED INT	参照トリガ・アクション (たとえば ON DELETE CASCADE) を持つ外部キー定義のあるテーブルの ID。foreign_table_id 値は、ISYSIDX.table_id の値を反映します。
foreign_key_id	UNSIGNED INT	foreign_table_id によって参照されるテーブルの外部キーの ID。foreign_key_id 値は、ISYSIDX.index_id の値を反映します。
referential_action	CHAR(1)	外部キーによって定義される動作。この 1 文字の値は、外部キーを作成したときに指定した動作に対応します。 • **C –** CASCADE • **D –** SET DEFAULT • **N –** SET NULL • **R –** RESTRICT
trigger_name	CHAR(128)	トリガ名。1つのテーブルには、同じ名前のトリガは複数存在できません。

カラム名	データ型	説明
trigger_defn	LONG VARCHAR	トリガを作成するのに使ったコマンド。
remarks	LONG VARCHAR	トリガに関する注記。ISYSREMARK システム・テーブル内に格納されている値。
source	LONG VARCHAR	トリガの SQL ソース。この値は、ISYSSOURCE システム・テーブル内に格納されています。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (trigger_id)
```

```
FOREIGN KEY (object_id) references SYS.ISYSOBJECT (object_id) MATCH
UNIQUE FULL
```

```
FOREIGN KEY (table_id) references SYS.ISYSTAB (table_id)
```

```
FOREIGN KEY fkey_index (foreign_table_id, foreign_key_id) references
SYS.ISYSIDX (table_id, index_id)
```

```
UNIQUE Index (table_id, event, trigger_time, trigger_order)
```

```
UNIQUE Index (trigger_name, table_id)
```

```
UNIQUE Index (table_id, foreign_table_id, foreign_key_id, event)
```

SYSTRIGGERS 統合ビュー

SYSTRIGGERS ビューの各ローは、データベース内のトリガ 1 つを示します。このテーブルには、参照トリガ・アクションを持つ外部キー定義に、自動的に作成されるトリガも含まれます (たとえば、ON DELETE CASCADE)。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSTRIGGERS"( owner,
  trigname,tname,event,trigtime,trigdefn )
  as select u.user_name,trig.trigger_name,tab.table_name,
    if trig.event = 'I' then 'INSERT'
    else if trig.event = 'U' then 'UPDATE'
      else if trig.event = 'C' then 'UPDATE'
        else if trig.event = 'D' then 'DELETE'
          else if trig.event = 'A' then 'INSERT,DELETE'
            else if trig.event = 'B' then 'INSERT,UPDATE'
              else if trig.event = 'E' then 'DELETE,UPDATE'
                else 'INSERT,DELETE,UPDATE'
                endif
              endif
            endif
          endif
        endif
```

```
      endif
   endif,if trig.trigger_time = 'B' or trig.trigger_time = 'P' then
'BEFORE'
   else if trig.trigger_time = 'A' or trig.trigger_time = 'S' then
'AFTER'
     else if trig.trigger_time = 'R' then 'RESOLVE'
       else 'INSTEAD OF'
       endif
     endif
   endif,trig.trigger_defn
   from SYS.ISYSTRIGGER as trig
     join SYS.ISYSTAB as tab on(tab.table_id = trig.table_id)
     join SYS.ISYSUSER as u on u.user_id = tab.creator where
   trig.foreign_table_id is null
```

SYSTYPEMAP システム・ビュー

SYSTYPEMAP システム・ビューには、SYSSQLSERVERTYPE システム・ビューの互換性マッピング値があります。このビューの基本となるシステム・テーブルは ISYSTYPEMAP です。

カラム名	データ型	説明
ss_user_type	SMALLINT	Adaptive Server Enterprise ユーザ型を格納します。
sa_domain_id	SMALLINT	対応する SQL Anywhere domain_id を格納します。
sa_user_type	SMALLINT	対応する SQL Anywhere ユーザ型を格納します。
nullable	CHAR(1)	型が NULL 値を許容するかどうか。

基本となるシステム・テーブルに関する制約

```
FOREIGN KEY (sa_domain_id) references SYS.ISYSDOMAIN (domain_id)
```

SYSTYPES ASE 互換ビュー

systypes には、システムが提供するデータ型およびユーザ定義データ型ごとに 1 つのローが含まれています。ドメイン (ルールで定義される) とデフォルトが存在する場合は、それらも含まれています。

このビューは、DBO が所有します。システムが提供するデータ型を示すローは、変更できません。

参照：

- 各 Adaptive Server Enterprise データベース内のテーブル (699 ページ)
- SYSCOLUMNS ASE 互換ビュー (614 ページ)
- SYSCOMMENTS ASE 互換ビュー (615 ページ)
- SYSINDEXES ASE 互換ビュー (632 ページ)
- SYSIQOBJECTS ASE 互換ビュー (644 ページ)

- SYSIQVINDEX ASE 互換ビュー (647 ページ)
- SYSOBJECTS ASE 互換ビュー (653 ページ)
- SYSUSERS ASE 互換ビュー (696 ページ)

SYSUSER システム・ビュー

SYSUSER システム・ビューの各ローは、システム内のユーザを示します。このビューの基本となるシステム・テーブルは ISYSUSER です。

カラム名	データ型	説明
user_id	UNSIGNED INT	ログイン・ポリシーが割り当てられたユーザのユニークな識別子。
object_id	UNSIGNED BIGINT	データベースのユーザのユニークな識別子。
user_name	CHAR(128)	ユーザのログイン名。
password	BINARY(128)	ユーザのパスワード。
login_policy_id	UNSIGNED BIGINT	ログイン・ポリシーのユニークな識別子。
expired_password_on_login	TINYINT	次回のログイン時にユーザのパスワードの有効期限が切れるかどうかを示します。
password_creation_time	TIMESTAMP	ユーザのパスワードが作成された時刻。
failed_login_attempts	UNSIGNED INT	アカウントがロックされるまでにユーザがログインを失敗できる回数。
last_login_time	TIMESTAMP	ユーザが最後にログインした時刻。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (user_id)
```

```
FOREIGN KEY (object_id) references SYS.ISYSOBJECT (object_id) MATCH
UNIQUE FULL
```

```
FOREIGN KEY (login_policy_id) references SYS.ISYSLOGINPOLICY
(login_policy_id)
```

```
UNIQUE Index (user_name)
```

SYSUSERAUTH 互換ビュー (旧式)

SYSUSERAUTH ビューは、古いバージョンのソフトウェアとの互換性を保つために用意されています。代わりに SYSUSERAUTHORITY システム・ビューを使用してください。

SYSUSERAUTH ビューの各ローは、user_id を公開せずに、ユーザに関して記述します。各ユーザは、ユーザ名で識別されます。このビューにはパスワードが表示されるため、PUBLIC 選択パーミッションはありません。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSUSERAUTH"( name,
  password,resourceauth,dbaauth,scheduleauth,user_group )
  as select
SYSUSERPERM.user_name,SYSUSERPERM.password,SYSUSERPERM.resourceauth
,SYSUSERPERM.dbaauth,SYSUSERPERM.scheduleauth,SYSUSERPERM.user_grou
p
    from SYS.SYSUSERPERM
```

SYSUSERAUTHORITY システム・ビュー

SYSUSERAUTHORITY システム・ビューの各ローは、1 つのユーザ ID に付与された権限を示します。このビューの基本となるシステム・テーブルは、ISYSUSERAUTHORITY です。

カラム名	データ型	説明
user_id	UNSIGNED INT	権限が所属するユーザの ID。
auth	VARCHAR(20)	ユーザに付与されている権限。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (user_id, auth)
```

```
FOREIGN KEY (user_id) references SYS.ISYSUSER (user_id)
```

SYSUSERLIST 互換ビュー (旧式)

SYSUSERAUTH ビューは、古いバージョンのソフトウェアとの互換性を保つために用意されています。

SYSUSERLIST ビューの各ローには、user_id とパスワードを公開することなく、ユーザに関して記述します。各ユーザは、ユーザ名で識別されます。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSUSERLIST"( name,
  resourceauth,dbaauth,scheduleauth,user_group )
  as select
SYSUSERPERM.user_name,SYSUSERPERM.resourceauth,SYSUSERPERM.dbaauth,
SYSUSERPERM.scheduleauth,SYSUSERPERM.user_group
    from SYS.SYSUSERPERM
```

SYSUSERMESSAGE システム・ビュー

SYSUSERMESSAGE システム・ビューの各ローには、エラーに対するユーザ定義メッセージが入っています。このビューの基本となるシステム・テーブルは ISYSUSERMESSAGE です。

注意： SYSUSERMESSAGES システム・テーブルに含まれる前のカタログ・バージョン。このテーブル名は ISYSUSERMESSAGE ('S' なし) に変更され、このビューの基本となるテーブルになります。

カラム名	データ型	説明
error	INTEGER	エラー条件に対するユニークな識別番号。
uid	UNSIGNED INT	メッセージを定義するユーザ番号。
description	VARCHAR(255)	エラー条件に対応するメッセージ。
langid	SMALLINT	予約済み。

基本となるシステム・テーブルに関する制約

```
FOREIGN KEY (uid) references SYS.ISYSUSER (user_id)
```

```
UNIQUE Constraint (error, langid)
```

SYSUSEROPTIONS 統合ビュー

SYSUSEROPTIONS ビューには、各ユーザに有効なオプション設定が含まれます。ユーザにオプション設定がない場合、このビューには PUBLIC のオプション設定が表示されます。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSUSEROPTIONS"( user_name,
  "option",setting )
  as select u.user_name,
    o."option",
    isnull((select s.setting
```

```
      from SYS.ISYSOPTION as s
      where s.user_id = u.user_id
      and s."option" = o."option"),
    o.setting)
    from SYS.SYSOPTIONS as o,SYS.ISYSUSER as u
    where o.user_name = 'PUBLIC'
```

SYSUSERPERM 互換ビュー (旧式)

前のバージョンで使用できる権限とパーミッションが表示されるだけなので、このビューは使用されなくなりました。アプリケーションを変更して代わりに SYSUSERAUTHORITY システム・ビューを使用してください。

SYSUSERPERM ビューの各ローは、ユーザ ID 1 つを示します。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSUSERPERM"
  as select b.user_id,
    b.object_id,
    b.user_name,
    b.password,
    if exists(select * from SYS.ISYSUSERAUTHORITY
      where ISYSUSERAUTHORITY.user_id = b.user_id and
ISYSUSERAUTHORITY.auth = 'RESOURCE') then
      'Y' else 'N' endif as resourceauth,
    if exists(select * from SYS.ISYSUSERAUTHORITY
      where ISYSUSERAUTHORITY.user_id = b.user_id and
ISYSUSERAUTHORITY.auth = 'DBA') then
      'Y' else 'N' endif as dbaauth,
    'N' as scheduleauth,
    if exists(select * from SYS.ISYSUSERAUTHORITY
      where ISYSUSERAUTHORITY.user_id = b.user_id and
ISYSUSERAUTHORITY.auth = 'PUBLISH') then
      'Y' else 'N' endif as publishauth,
    if exists(select * from SYS.ISYSUSERAUTHORITY
      where ISYSUSERAUTHORITY.user_id = b.user_id and
ISYSUSERAUTHORITY.auth = 'REMOTE DBA') then
      'Y' else 'N' endif as remotedbaauth,
    if exists(select * from SYS.ISYSUSERAUTHORITY
      where ISYSUSERAUTHORITY.user_id = b.user_id and
ISYSUSERAUTHORITY.auth = 'GROUP') then
      'Y' else 'N' endif as user_group,
    r.remarks
    from SYS.ISYSUSER as b
      left outer join SYS.ISYSREMARK as r on(b.object_id =
r.object_id)
```

SYSUSERPERMS 互換ビュー (旧式)

前のバージョンで使用できる権限とパーミッションが表示されるだけなので、このビューは使用されなくなりました。アプリケーションを変更して代わりに SYSUSERAUTHORITY および SYSUSER システム・ビューを使用してください。

SYSUSERPERMS ビューと同様に、SYSUSERPERMS ビューの各ローはユーザ ID 1 つを示します。ただし、パスワード情報は含まれません。すべてのユーザがこのビューを表示できます。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSUSERPERMS"
  as select
SYSUSERPERM.user_id,SYSUSERPERM.user_name,SYSUSERPERM.resourceauth,
SYSUSERPERM.dbaauth,

SYSUSERPERM.scheduleauth,SYSUSERPERM.user_group,SYSUSERPERM.publish
auth,SYSUSERPERM.remotedbaauth,SYSUSERPERM.remarks
    from SYS.SYSUSERPERM
```

SYSUSERTYPE システム・ビュー

SYSUSERTYPE システム・ビューの各ローには、ユーザ定義のデータ型の説明が格納されます。このビューの基本となるシステム・テーブルは ISYSUSERTYPE です。

カラム名	データ型	説明
type_id	SMALLINT	ユーザ定義データ型のユニークな識別番号。
creator	UNSIGNED INT	データ型の所有者のユーザ番号。
domain_id	SMALLINT	SYSDOMAIN システム・ビューにリストされたデータ型番号で示す、ユーザ定義データ型の元になるデータ型。
nulls	CHAR(1)	ユーザ定義データ型が NULL を許可するかどうか。可能な値は Y、N、または U です。値 U は、NULL 入力属性が指定されていないことを示します。
width	BIGINT	文字列カラムでは長さ、数値カラムでは精度、その他のデータ型では記憶領域のサイズをバイト数で示します。
scale	SMALLINT	数値データ型カラムでは小数点以下の桁数、その他のデータ型では 0 を示します。
type_name	CHAR(128)	データ型の名前。

カラム名	データ型	説明
"default"	LONG VARCHAR	データ型のデフォルト値。
"check"	LONG VARCHAR	データ型の CHECK 条件。
base_type_str	VARCHAR(32767)	ユーザ・タイプの物理的な型を表す注釈付きの型文字列。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (type_id)
FOREIGN KEY (creator) references SYS.ISYSUSER (user_id)
FOREIGN KEY (domain_id) references SYS.ISYSDOMAIN (domain_id)
UNIQUE Constraint (type_name)
```

SYSUSERS ASE 互換ビュー

sysusers には、データベースで許可されたユーザごとに 1 つのロー、およびグループまたは役割ごとに 1 つのローが含まれています。

このビューは、DBO が所有します。

参照：

- 各 Adaptive Server Enterprise データベース内のテーブル (699 ページ)
- SYSCOLUMNS ASE 互換ビュー (614 ページ)
- SYSCOMMENTS ASE 互換ビュー (615 ページ)
- SYSINDEXES ASE 互換ビュー (632 ページ)
- SYSIQOBJECTS ASE 互換ビュー (644 ページ)
- SYSIQVINDEX ASE 互換ビュー (647 ページ)
- SYSOBJECTS ASE 互換ビュー (653 ページ)
- SYSTYPES ASE 互換ビュー (690 ページ)

SYSVIEW システム・ビュー

SYSVIEW システム・ビューの各ローは、データベース内のビューを示します。

ビューの詳細については、「SYSTAB システム・ビュー」を参照してください。このビューの元になっているシステム・テーブルは ISYSVIEW です。

マテリアライズド・ビューの情報を読みやすい形式で表示するには、**sa_materialized_view_info** システム・プロシージャも使用できます。 マテリアライズド・ビューは、IQ カタログ・ストアの SQL Anywhere テーブルでのみサポートされます。

SYSVIEW ビューは SQL Anywhere のシステム・ビューです。詳細については、『SQL Anywhere サーバー - SQL リファレンス』の「ビュー」>「システムビュー」>「SYSVIEW システムビュー」を参照してください。

注意： 参照先は SQL Anywhere のマニュアルです。

SYSVIEWS 統合ビュー

SYSVIEWS ビューの各ローには、ビュー定義など、ビューに関して記述します。

ビューを構成するテーブルとカラムは、以下の SQL 文で示されます。特定のテーブルまたはカラムの詳細については、以下のビュー定義にあるリンクを参照してください。

```
ALTER VIEW "SYS"."SYSVIEWS"( vcreator,
  viewname,viewtext )
  as select u.user_name,t.table_name,v.view_def
    from SYS.ISYSTAB as t
      join SYS.ISYSVIEW as v on(t.object_id = v.view_object_id)
      join SYS.ISYSUSER as u on(u.user_id = t.creator)
```

SYSWEBSERVICE システム・ビュー

SYSWEBSERVICE システム・ビューの各ローには、Web サービスの説明が格納されます。このビューの基本となるシステム・テーブルは ISYSWEBSERVICE です。

カラム名	データ型	説明
service_id	UNSIGNED INT	Web サービスをユニークに識別する番号。
object_id	UNSIGNED BIGINT	Web サービスの ID。
service_name	CHAR(128)	Web サービスに割り当てられた名前。
service_type	VARCHAR(40)	サービスのタイプには、RAW、HTTP、XML、SOAP、DISH などがあります。
auth_required	CHAR(1)	すべての要求に有効なユーザ名とパスワードが必要かどうか。
secure_required	CHAR(1)	HTTP などのセキュリティ保護されていない接続を受け入れるのか、または HTTPS などのセキュリティ保護された接続だけを受け入れるのか。
url_path	CHAR(1)	URL の解釈を制御します。
user_id	UNSIGNED INT	認証が有効の場合、サービス使用のパーミッションを持つユーザまたはユーザ・グループを識別します。認証が無効の場合、要求を処理するときに使用するアカウントを指定します。

カラム名	データ型	説明
parameter	LONG VARCHAR	DISH サービスにインクルードされる SOAP サービスを識別するプレフィクス。
statement	LONG VARCHAR	要求に応答して常に実行される SQL 文。NULL の場合、各要求に含まれる任意の文が代わりに実行されます。DISH 型のサービスでは無視されます。
remarks	LONG VARCHAR	Web サービスに関する注記。ISYSREMARK システム・テーブル内に格納されている値。
enabled	CHAR(1)	Web サービスが、現在、有効または無効のいずれであるかを示します (CREATE SERVICE を参照してください)。

基本となるシステム・テーブルに関する制約

```
PRIMARY KEY (service_id)
```

```
FOREIGN KEY (object_id) references SYS.ISYSOBJECT (object_id) MATCH
UNIQUE FULL
```

```
UNIQUE Constraint (service_name)
```

Transact-SQL 互換ビュー

Adaptive Server Enterprise および Sybase IQ では、両製品の用途の違いを反映して、別々のシステム・カタログを用意しています。

Adaptive Server Enterprise の場合、1 つのサーバ内に 1 つの master データベースが存在します。master データベースは、サーバ内にあるすべてのデータベースに適用される情報を格納する一連のシステム・テーブルで構成されています。 master データベースとともに多数のデータベースが存在し、各データベースにはそれに関連するシステム・テーブルがあります。

Sybase IQ の場合、各データベースは独立して存在し、各自のシステム・テーブルを持っています。 データベースの集合体としてのシステム情報を持つ master データベースはありません。 各サーバは、必要に応じて各データベースを動的にロードおよびアンロードしながら、複数のデータベースを同時に実行できます。

Adaptive Server Enterprise と Sybase IQ のシステム・カタログは異なります。Adaptive Server Enterprise のシステム・テーブルとビューは、特殊なユーザ dbo が所有し、一部は master データベース内に、一部は **sybsecurity** データベース内に、一部は個別のデータベース内に分散されています。一方、Sybase IQ のシステム・テーブルとビューは、特殊なユーザ SYS が所有し、各データベース内に別々に存在しています。

互換性のあるアプリケーションを作成できるように、Sybase IQ には、特殊なユーザ dbo が所有する一連のビューがあります。これらのビューは、Adaptive Server Enterprise のシステム・テーブルおよびビューにそれぞれ対応しています。 構造上の違いのために、特定の Adaptive Server Enterprise テーブルまたはビューの内容が Sybase IQ のコンテキストで無意味になる場合、そのビューはカラム名とデータ型だけの空のビューになります。

これらのトピックは、Adaptive Server Enterprise のシステム・テーブルと、そのシステム・テーブルの Sybase IQ システム・カタログ内での実装を示します。 すべてのテーブルの所有者は、各 DBMS の dbo です。

参照：

- ASE T-SQL 互換ビュー (606 ページ)

各 Adaptive Server Enterprise データベース内のテーブル

Adaptive Server Enterprise のシステム・テーブルのすべてが Sybase IQ システム・カタログに実装されているわけではありません。

表 212 : 各 ASE データベース内のテーブル

テーブル名	説明	データ?	IQ によるサポート?
sysalternates	データベース・ユーザにマップされたユーザごとのロー	いいえ	いいえ
syscolumns	テーブルまたはビューのカラムごとに 1 つのロー、およびプロシージャのパラメータごとに 1 つのロー。Sybase IQ では、クエリするときに所有者名として dbo (つまり、dbo.syscolumns) を使用してください。	可	可
syscomments	ビュー、ルール、デフォルト、プロシージャごとに 1 ロー (複数ローにまたがる可能性あり)、SQL 定義文を表示。	可	可
sysconstraints	テーブルまたはカラムに関連する参照制約および検査制約それぞれにつき 1 つのロー。	いいえ	いいえ
sysdepends	プロシージャまたはビューが参照するプロシージャ、ビュー、またはテーブルごとに 1 ロー。	いいえ	いいえ

テーブル名	説明	データ?	IQ によるサポート?
sysindexes	クラスタード・インデックスまたはノンクラスタード・インデックスごとに 1 ロー。インデックスのないテーブルごとに 1 ロー。さらに text または image データを持つテーブルごとに 1 ロー。Sybase IQ では、クエリするときに所有者名として dbo (つまり、dbo.sysindexes) を使用してください。	可	可
sysiqobjects	システム・テーブル、ユーザ・テーブル、ビュー、プロシージャ、トリガ、イベント、ジョイン・インデックス、制約、ドメイン (sysdomain)、ドメイン (sysusertype)、カラム、インデックスごとに 1 ロー。	可	可
sysiqvindex	FP IQ 以外のインデックスごとに 1 ロー。	可	可
syskeys	ユーザが設定した (Adaptive Server Enterprise によって維持されない) プライマリ・キー、外部キー、または共通キーごとに 1 ロー。	いいえ	いいえ
syslogs	トランザクション・ログ。	いいえ	いいえ
sysobjects	テーブル、ビュー、プロシージャ、ルール、デフォルト、ログ、テンポラリ・オブジェクト (tempdb 内でのみ) ごとに 1 ロー。	互換性のあるデータだけを含む。	可
sysprocedures	ビュー、ルール、デフォルト、プロシージャごとに 1 ロー、内部定義を表示。	いいえ	いいえ
sysprotects	ユーザのパーミッション情報。	いいえ	いいえ
sysreferences	テーブルまたはカラムに宣言された参照整合性制約につき 1 つのロー。	いいえ	いいえ
sysroles	サーバ全体に適用される役割をローカル・データベース・グループにマップ。	いいえ	いいえ
syssegments	各セグメント (ディスクの一部分の集まりに名前を付けたもの) につき 1 つのロー。	いいえ	いいえ
systhresholds	データベースに定義されたスレッショルドにつき 1 つのロー。	いいえ	いいえ
systypes	システムが提供するデータ型またはユーザ定義のデータ型ごとに 1 ロー。	可	可

テーブル名	説明	データ?	IQ によるサポート?
sysusers	データベースに許可されているユーザにつき 1 つのロー。	可	可

参照：

- SYSCOLUMNS ASE 互換ビュー (614 ページ)
- SYSCOMMENTS ASE 互換ビュー (615 ページ)
- SYSINDEXES ASE 互換ビュー (632 ページ)
- SYSIQOBJECTS ASE 互換ビュー (644 ページ)
- SYSIQVINDEX ASE 互換ビュー (647 ページ)
- SYSOBJECTS ASE 互換ビュー (653 ページ)
- SYSTYPES ASE 互換ビュー (690 ページ)
- SYSUSERS ASE 互換ビュー (696 ページ)

Adaptive Server Enterprise master データベース内のテーブル

Adaptive Server Enterprise の master データベース・テーブルのすべてが Sybase IQ システム・カタログに実装されているわけではありません。

表 213 : ASE master データベース・テーブル

テーブル名	説明	データ?	IQ によるサポート?
syscharsets	文字セットまたはソート順ごとに 1 ロー。	いいえ	いいえ
sysconfigures	ユーザが設定できる設定パラメータごとに 1 ロー。	いいえ	いいえ
syscurconfigs	サーバが現在使用中の設定パラメータについての情報。	いいえ	いいえ
sysdatabases	サーバ上のデータベースごとに 1 ロー。	いいえ	いいえ
sysdevices	テープ・ダンプ・デバイス、ディスク・ダンプ・デバイス、データベース用のディスク、データベース用のディスク・パーティションごとに 1 ロー。	いいえ	いいえ
sysengines	現在オンラインのサーバごとに 1 ロー。	いいえ	いいえ

テーブル名	説明	データ?	IQ によるサポート?
syslanguages	サーバが認識している言語 (アメリカ英語を除く) ごとに 1 ロー。	いいえ	いいえ
syslocks	アクティブ・ロックについての情報。	いいえ	いいえ
sysloginroles	システム定義された役割を持つサーバ・ログインごとに 1 ロー。	いいえ	いいえ
syslogins	有効なユーザ・アカウントごとに 1 ロー。	可	可
sysmessages	システム・エラーまたは警告ごとに 1 ロー。	いいえ	いいえ
sysprocesses	サーバ・プロセスについての情報。	いいえ	いいえ
sysremotelogins	リモート・ユーザごとに 1 ロー。	いいえ	いいえ
syssrvroles	サーバ全体に適用される役割ごとに 1 ロー。	いいえ	いいえ
sysservers	リモート・サーバごとに 1 ロー。	いいえ	いいえ
sysusages	データベースに割り当てられたディスクの断片ごとに 1 ロー。	いいえ	いいえ

Adaptive Server Enterprise sybsecurity データベース内のテーブル

Adaptive Server Enterprise の sybsecurity データベース・テーブルは Sybase IQ システム・カタログに実装されていません。

表 214 : ASE sybsecurity データベース・テーブル

テーブル名	説明	データ?	IQ によるサポート?
sysaudits	監査レコードごとに 1 ロー。	いいえ	いいえ
sysauditoptions	グローバル監査オプションごとに 1 ロー。	いいえ	いいえ

他の Sybase データベースとの互換性

この項のトピックを参考にすると、他の Sybase データベースから Sybase IQ への移行が簡単になります。また、これらのトピックは、Adaptive Server Enterprise または SQL Anywhere と互換性のある Sybase IQ アプリケーションの、作成ガイドとしても利用できます。

Sybase IQ の各新バージョンでは互換性を維持するための機能が提供されています。この付録では、Sybase IQ と Adaptive Server Enterprise、および SQL Anywhere を比較します。

SQL Anywhere の概要

Sybase IQ は、SQL Anywhere を拡張したものです。

SQL 構文、関数、オプション、ユーティリティ、プロシージャ、その他の機能のほとんどは、両製品に共通ですが、 両製品には相違点もあるので、 SQL Anywhere のマニュアルに記載されている機能が Sybase IQ でもサポートされているとはかぎりません。

Sybase IQ のマニュアル・セットでは相違点の多くが説明されていますが、すべてをカバーしているわけではありません。*Sybase IQ のマニュアルは常に SQL Anywhere のマニュアルに優先します。*

Transact-SQL サポートの概要

Sybase IQ は、SQL Anywhere と同様に、*Transact-SQL* (Sybase Adaptive Server Enterprise がサポートする SQL のダイアレクト) の大部分のサブセットをサポートしています。

Sybase IQ では、Transact-SQL をサポートすることで、アプリケーションの移植性を実現しています。 アプリケーション、ストアド・プロシージャ、バッチ・ファイルの多くは、Adaptive Server Enterprise と Sybase IQ の両方のデータベースで使用できるように作成できます。

目的は、Adaptive Server Enterprise と Sybase IQ の両方で動作するアプリケーションを作成することです。通常は、既存の Adaptive Server Enterprise アプリケーションを SQL Anywhere または Sybase IQ のデータベー ス上で実行する場合、多少の変更が必要です。

次に、Sybase IQ での Transact-SQL に対するサポート方法を示します。

- SQL 文のほとんどは、Sybase IQ と Adaptive Server Enterprise の間で互換性があります。
- 一部の文、特にプロシージャ言語を使ってプロシージャやバッチ用に書かれた文では、以前のバージョンの Sybase IQ でサポートされた構文に加え、別の Transact-SQL 文もサポートされます。 このような文については、SQL Anywhere と Sybase IQ では 2 種類の SQL ダイアレクトがサポートされています。 この付録では、それらのダイアレクトを Transact-SQL と Watcom-SQL と呼びます。
- 1 つのプロシージャまたはバッチは、Transact-SQL と Watcom-SQL のどちらかの言語のみで実行できます。バッチまたはプロシージャ中では、どちらか 1 つの言語だけで制御文を記述してください。たとえば、ダイアレクトごとに異なったフロー制御文があります。

Sybase IQ は、既存のデータの処理に使用する、Transact-SQL 言語の要素、関数、文の多くをサポートします。

さらに、Sybase IQ では、Transact-SQL のストアド・プロシージャ言語 (**CREATE PROCEDURE** 構文、制御文など) の大部分と、Transact-SQL のデータ定義言語文のほとんどがサポートされています。

それぞれの製品ごとに、サポートされる構造と設定について設計上の相違点があります。 デバイス管理、ユーザ管理、バックアップなどの管理タスクの多くはシステム固有のものです。 ただし、Sybase IQ は Transact-SQL のシステム・テーブルをビューとして提供しますが、Sybase IQ にとって意味を持たないテーブルにはローがありません。 また、Sybase IQ は、一般的な管理タスクの一部を一連のシステム・プロシージャとして提供します。

Adaptive Server Enterprise、SQL Anywhere、Sybase IQ のアーキテクチャ

Adaptive Server Enterprise、SQL Anywhere、Sybase IQ は、それぞれの明確な目的に合わせたアーキテクチャを持つ、相互に補完的な製品です。

Sybase IQ は、データ・ウェアハウスと分析処理専用に設計された、高パフォーマンスの意思決定支援サーバです。SQL Anywhere は、管理作業をほとんど必要とせず、ワークグループや部署単位のサーバとして、さらにはパーソナル・データベースとして適切な製品です。Adaptive Server Enterprise は、トランザクション処理を主眼とした大規模データベース用のエンタープライズレベルのサーバに最適な製品です。

このセクションでは、これら 3 つの製品のアーキテクチャ上の違いを解説します。 また、データベース管理の互換性のために Sybase IQ と SQL Anywhere に含まれている、Adaptive Server Enterprise に似たツールについても説明します。

サーバとデータベース

サーバとデータベースの関係は、Adaptive Server Enterprise と Sybase IQ や SQL Anywhere とでは異なります。

Adaptive Server Enterprise では各データベースはサーバ内に存在し、各サーバは複数のデータベースを持つことができます。 ユーザはサーバに対するログイン権限を持っている場合、サーバに接続できます。さらに、パーミッションがあれば、そのサーバ上のすべてのデータベースに接続できます。 master データベースに保持され、システム全体に適用されるシステム・テーブルには、サーバ上のすべてのデータベースに共通な情報が含まれています。

Sybase IQ には Adaptive Server Enterprise の master データベースに相当するものはありません。その代わり、それぞれのデータベースが独立したエンティティであり、独自のシステム・テーブルを持っています。 ユーザはサーバに対してではなく、個々のデータベースに対する接続権限を与えられます。ユーザが接続するのは、個々のデータベースです。master データベース・レベルで維持され、システム全体に適用されるシステム・テーブルのセットはありません。 各 Sybase IQ データベース・サーバは、動的にデータベースを開始および停止できます。このデータベースに対する個別の接続は、ユーザが保持できます。 1 つのサーバで実行する Sybase IQ データベースは 1 つだけにすることを強くおすすめします。

SQL Anywhere と Sybase IQ は、Transact-SQL のサポートと Open Server のサポートによって、Adaptive Server Enterprise と同様の方法でタスクを実行するツールを提供します。 ただし、これらのツールの厳密な実装方法には違いがあります。

領域の割り付けとデバイス管理

Adaptive Server Enterprise、SQL Anywhere、Sybase IQ は、それぞれの製品の用途を反映して、デバイスの管理とディスク領域の割り付けに、初期設定でもその後の管理でも別々のモデルを使用します。

次に例を示します。

- Adaptive Server Enterprise では、最初に **DISK INIT** を使ってデータベース・デバイス内の領域を割り付け、その後 1 つまたは複数のデータベース・デバイス上に個々のデータベースを作成します。 領域を追加するには、**ALTER DATABASE** を使用する他、スレッショルドを使って自動的に領域を追加することもできます
- Sybase IQ では、最初に **CREATE DATABASE** 文にロー・デバイスをリストアップすることによって、領域を割り付けます。**CREATE DBSPACE** を使用すると、手動で領域を追加できます。領域を自動的に追加することはできませんが、追加のスペースが実際に必要になる前に DBA に警告するイベントを作成することはできます。 Sybase IQ では、ファイル・システム領域も利用できます。

Sybase IQ は、**DISK INIT**、**DISK MIRROR**、**DISK REFIT**、**DISK REINIT**、**DISK REMIRROR**、**DISK UNMIRROR** などの Transact-SQL **DISK** 文をサポートしません。

- SQL Anywhere は、最初の **CREATE DATABASE** 文がロー・デバイスのリストではなく単一のファイル・システム・ファイルを取るという点を除いては、Sybase IQ と同様です。SQL Anywhere では、dbinit という名前のコマンド・ユーティリティを使用してデータベースを初期化できます。Sybase IQ には、このユーティリティの拡張バージョンとして、IQ データベースを初期化する **iqinit** という名前のユーティリティが用意されています。

ディスク管理の詳細については、『システム管理ガイド：第 1 巻』を参照してください。

システム・テーブル、カタログ・ストア、IQ ストア

IQ データベースは、ジョイント・データ・ストアです。

ジョイント・ストアの構成を次に示します。

- カタログ・ストアは、システム・テーブルとストアド・プロシージャを含み、SQL Anywhere と互換性のある一連のテーブル内にあります。
- 永続的な IQ ストアは、一連の Sybase IQ テーブルです。テーブル・データは、インデックスに格納されます。
- テンポラリ・ストアは、データベース・サーバが、ソートやその他の一時的な処理に使用する一連のテンポラリ・テーブルです。

カタログの差異と互換性サポートには次のものが含まれます。

- SQL Anywhere と Sybase IQ は、カタログ (テーブル、カラムなど) に Adaptive Server Enterprise とは異なるスキーマを使用します。
- SQL Anywhere と Sybase IQ には、Adaptive Server Enterprise のシステム・テーブルの対応する部分を模倣する互換ビューが用意されています。ただし、これらのビューを利用するとパフォーマンスに影響します。
- Adaptive Server Enterprise では、データベース所有者 (ユーザ ID dbo) がカタログ・オブジェクトを所有します。
- SQL Anywhere と Sybase IQ では、システム所有者 (ユーザ ID SYS) がカタログ・オブジェクトを所有します。

注意： ユーザ ID dbo は、Sybase IQ が提供する、Adaptive Server Enterprise と互換性のあるシステム・ビューの所有者です。

管理者の役割

Adaptive Server Enterprise は、SQL Anywhere や Sybase IQ より管理者の役割が充実しています。

Adaptive Server Enterprise には、複数の異なった役割が用意されていますが、Adaptive Server Enterprise の複数のログイン・アカウントに同じ役割を与え、1 つのアカウントに複数の役割を持たせることができます。

Adaptive Server Enterprise での役割	説明
システム管理者	特定のアプリケーションに関連していない一般的な管理タスクを担当し、あらゆるデータベース・オブジェクトにアクセスできます。
システム・セキュリティ担当者	セキュリティが問題となる Adaptive Server Enterprise のタスクを担当しますが、データベース・オブジェクトに対する特別なパーミッションは持ちません。
データベース所有者	自分が所有するデータベース内のオブジェクトに対して、フル・パーミッションを持ちます。また、データベースにユーザを追加したり、データベース内でオブジェクトの作成やコマンドの実行を行うパーミッションを他のユーザへ付与したりできます。
データ定義文	**CREATE TABLE** や **CREATE VIEW** などの特定のデータ定義文に対するパーミッションがユーザに付与され、ユーザはデータベース・オブジェクトを作成できるようになります。
オブジェクト所有者	各データベース・オブジェクトには所有者があり、そのオブジェクトにアクセスするパーミッションを他のユーザに与えることができます。オブジェクトの所有者は、自動的にそのオブジェクトに対するすべてのパーミッションを持ちます。

SQL Anywhere/ Sybase IQ での役割	説明
データベース管理者 (DBA 権限)	Adaptive Server Enterprise のデータベース所有者のように、データベース内のすべてのオブジェクト (SYS が所有するオブジェクトは除く) に対してフル・パーミッションを持ち、データベース内でのオブジェクト作成とコマンド実行のパーミッションを他のユーザに付与できます。 デフォルトのデータベース管理者は、ユーザ ID DBA です。

SQL Anywhere/ Sybase IQ での役割	説明
RESOURCE パーミッション	ユーザがデータベース内で任意のオブジェクトを作成することを許可します。 Adaptive Server Enterprise では個々の **CREATE** 文に対してパーミッションを与える方法が使用されますが、RESOURCE パーミッションはその代わりに使用されます。
オブジェクト所有者	Sybase IQ のオブジェクトには、Adaptive Server Enterprise の場合と同じように所有者がいます。 オブジェクト所有者は、パーミッションを付与する権限も含めて、そのオブジェクトに関するすべてのパーミッションを自動的に持ちます。

Adaptive Server Enterprise と Sybase IQ の両方に含まれるデータにスムーズにアクセスするために、適切なパーミッション (Sybase IQ では RESOURCE、Adaptive Server Enterprise では **CREATE** 文ごとのパーミッション) を持つユーザ ID をデータベースに作成し、このユーザ ID からオブジェクトを作成してください。同じユーザ ID を両方の環境で使用すると、オブジェクト名と修飾子が 2 つのデータベースで同一になり、互換性のあるアクセスが可能になります。

データ型

Adaptive Server Enterprise、SQL Anywhere、Sybase IQ では、データ型の取り扱い方法がそれぞれ異なります。

この項では、データ型の互換性情報について説明します。

注意： この項で説明していないデータ型は、現在 3 つの製品すべてでサポートされています。

Bit データ型

Adaptive Server Enterprise、SQL Anywhere、Sybase IQ は、異なる方法で `BIT` データ型をサポートします。

違いは次のとおりです。

- SQL Anywhere は 0 または 1 のみを許可します。
- Adaptive Server Enterprise と Sybase IQ は、integral データ型を `BIT` に暗黙的に変換します。 ゼロ以外の値は 1 (TRUE) として格納されます。

文字データ型

Adaptive Server Enterprise、SQL Anywhere、Sybase IQ は CHAR データ型と VARCHAR データ型を許可していますが、これらのデータ型の取り扱い方法がそれぞれ異なります。

- SQL Anywhere は、空白がパディングされたデータベースにおいても、すべての文字列を VARCHAR として扱います。
- Adaptive Server Enterprise および Sybase IQ は、CHAR (固定長) と VARCHAR (可変長) のデータを区別します。

  Adaptive Server Enterprise では、VARCHAR 値の末尾の空白文字が削除されます。Sybase IQ では、 VARCHAR 値の末尾の空白文字が削除されるかどうかは、データの形式と操作によって決まります。

CHAR または VARCHAR への挿入には次のような違いがあります。

- SQL Anywhere では、integral データ型を CHAR または VARCHAR (暗黙の変換) に挿入することを許可しています。
- Adaptive Server Enterprise と Sybase IQ では、明示的な変換が必要です。

カラムの最大サイズは次のように決まります。

- Adaptive Server Enterprise では、CHAR および VARCHAR は論理ページ・サイズ (2K、4K、8K、16K のいずれか) に依存します。 次に例を示します。
  - ページ・サイズが 2K の場合、カラムの最大サイズは単一のローと同じ大きさ (約 1962 バイト) です。
  - ページ・サイズが 4K の場合、カラムの最大サイズは約 4010 バイトです。
- SQL Anywhere は、CHAR と VARCHAR については最大 32KB-1 バイトを、LONG VARCHAR については最大 2GB をサポートしています。
- SQL Anywhere は名前として LONG VARCHAR とその同意語の TEXT をサポートしていますが、Adaptive Server Enterprise は名前として TEXT のみをサポートし、LONG VARCHAR をサポートしていません。
- Sybase IQ は、CHAR と VARCHAR のサイズとして、最大 32K-1 バイトをサポートします。

  Sybase IQ はさらに、LONGVARCHAR のサイズとして、最大 512TB (IQ ページ・サイズが 128KB の場合) および 2PB (IQ ページ・サイズが 512KB の場合) をサポートしています。Sybase IQ の LONG VARCHAR データ型の詳細については、『Sybase IQ の非構造化データ分析の概要』を参照してください。

- Adaptive Server Enterprise は、NCHAR、NVARCHAR、UNICHAR、UNIVARCHAR のデータ型をサポートします。 N はマルチバイトの文字セットを表し、UNI はシングル・バイトの文字セットを表します。
- SQL Anywhere と Sybase IQ は、Unicode を別のデータ型としてではなく、CHAR データ型と VARCHAR データ型の中でサポートしています。
- Sybase IQ と Adaptive Server Enterprise の互換性を保つため、文字データ型には必ず長さを指定してください。

参照：
- 文字データ型 (75 ページ)

バイナリ・データ型

バイナリ・データ型のサポートは、Adaptive Server Enterprise、SQL Anywhere、Sybase IQ でそれぞれ異なります。

表 215 : バイナリ・データ型の許容されるサイズ

データ型	Adaptive Server Enterprise	SQL Anywhere	Sybase IQ
BINARY	< ページ・サイズ	32KB - 1	255
VARBINARY	< ページ・サイズ	32KB - 1	32KB - 1
LONG BINARY*	未対応	2GB - 1	512TB (IQ ページ・サイズ 128KB) 2PB (IQ ページ・サイズ 512KB)
IMAGE	2GB	2GB - 1	LONG BINARY* を使用してください

*Sybase IQ での LONG BINARY データ型の詳細については、『Sybase IQ の非構造化データ分析の概要』を参照してください。 この機能には、別のライセンスが必要です。

Adaptive Server Enterprise と SQL Anywhere では、バイナリ・データの表示方法が異なります。

- Sybase IQ は、Adaptive Server Enterprise と SQL Anywhere の両方の表示形式をサポートします。
- BINARY フィールドに ‘123’ と入力した場合、SQL Anywhere の表示形式ではバイト単位の '123' になります。Adaptive Server Enterprise の表示形式ではニブル単位の '0x616263' になります。

参照：
- バイナリ・データ型 (84 ページ)
- NEWID 関数 [その他] (253 ページ)
- STRTOUUID 関数 [文字列] (328 ページ)
- UUIDTOSTR 関数 [文字列] (351 ページ)
- 文字データ型 (75 ページ)

date、time、datetime、timestamp データ型

Adaptive Server Enterprise、SQL Anywhere、Sybase IQ はいずれも、日付と時刻のデータを何らかの形式でサポートしていますが、それぞれに違いがあります。

- SQL Anywhere と Sybase IQ は、4 バイトの日付と時刻のデータ型をサポートしています。
- Adaptive Server Enterprise は、バイナリ (8) で実装されているユーザ定義のデータ型 (ドメイン) として、8 バイトの `datetime` データ型と `timestamp` データ型をサポートしています。
- SQL Anywhere と Sybase IQ は、8 バイトの `timestamp` 型と、timestamp として実装されている 8 バイトの `datetime` ドメインをサポートしています。 SQL Anywhere/Sybase IQ の、`datetime` データ型のミリ秒単位の精度は、Adaptive Server Enterprise の精度とは異なります。

それぞれの製品におけるデフォルトの日付表示形式には次のような違いがあります。

- Adaptive Server Enterprise のデフォルトでは、日付を "MM-DD-YYYY" 形式で表示しますが、オプションを設定してこれを変更できます。
- SQL Anywhere と Sybase IQ のデフォルトは、ISO の YYYY-MM-DD という形式ですが、オプションを設定してこれを変更できます。

時刻の変換は次のとおりです。

- Adaptive Server Enterprise では、文字列に格納された時刻を内部時刻に変換する方法が、秒の少数部分がコロンとピリオドのどちらで区切られているかによって変わります。
- SQL Anywhere と Sybase IQ では、デリミタに関係なく同じ方法で時刻が変換されます。

DATETIME カラムに時刻を挿入する場合は、次のようになります。

- Adaptive Server Enterprise と Sybase IQ では、1900 年 1 月 1 日がデフォルトで設定されます。
- SQL Anywhere では、現在の日付がデフォルトです。

Adaptive Server Enterprise データベースから取得した `TIME` 値と DATETIME 値は、**INSERT...LOCATION** を使用して DATETIME カラムがある Sybase IQ テーブルに挿入

すると、変更されます。**INSERT...LOCATION** 文は、DATETIME の精度が 300 分の 1 秒である Open Client を使用します。

たとえば、Adaptive Server Enterprise データベースのテーブル・カラムに、次の値が格納されているとします。

```
2004-11-08 10:37:22.823
```

INSERT...LOCATION を使用してこの値を取得し、Sybase IQ テーブルに格納すると、値は次のようになります。

```
2004-11-08 10:37:22.823333
```

ASE の日時値と時刻値との互換性

INSERT...LOCATION を使用して Adaptive Server Enterprise データベースから取得した `DATETIME` 値または `TIME` 値は、Open Client の日時の精度が原因で、元の値とは異なる値になることがあります。

たとえば、Adaptive Server Enterprise データベースでの `DATETIME` 値が '2004-11-08 10:37:22.823' のとき、**INSERT...LOCATION** を使用して取得した値は '2004-11-08 10:37:22.823333' になります。

BIGTIME と BIGDATETIME のサポート

Sybase IQ は、コンポーネント統合サービス (CIS: Component Integration Services) と **INSERT...LOCATION** 用に Adaptive Server Enterprise (ASE) のデータ型である `BIGTIME` と `BIGDATETIME` をサポートしています。

- Adaptive Server Enterprise とのコンポーネント統合サービス — aseodbc サーバ・クラス・プロキシ・テーブルは、データ型 `BIGTIME` と `BIGDATETIME` のカラムを含む Adaptive Server Enterprise テーブルにマップされます。
  Adaptive Server Enterprise テーブルにマップされるプロキシ・テーブルの作成時に、マッピングが指定されなかった場合、`BIGDATETIME` カラムはデフォルトで `TIMESTAMP` カラムにマップされます。`BIGTIME` カラムはデフォルトで `TIME` カラムにマップされます。
  asejdbc サーバ・クラスは、`BIGTIME` と `BIGDATETIME` のデータ型をサポートしていません。
- **INSERT...LOCATION** — **INSERT...LOCATION** コマンドは、データ型 `BIGTIME` と `BIGDATETIME` のカラムを含む Adaptive Server Enterprise テーブルから Sybase IQ テーブルにデータをロードします。
  Sybase IQ は、Adaptive Server Enterprise のデータ型 `BIGTIME` を Sybase IQ のデータ型 `TIME` に挿入します。

Sybase IQ は、Adaptive Server Enterprise のデータ型 BIGDATETIME を Sybase IQ のデータ型 DATETIME、DATE、TIME、TIMESTAMP に挿入します。

数値データ型

Adaptive Server Enterprise、SQL Anywhere、Sybase IQ では、それぞれデフォルトの精度と位取りが異なります。

- Adaptive Server Enterprise のデフォルトは精度 18、位取り 0 です。
- SQL Anywhere のデフォルトは精度 30、位取り 6 です。
- Sybase IQ のデフォルトは、精度 126、位取り 38 です。これらのデフォルトは TDS や一部のクライアント・ツールには大きすぎるため、Sybase IQ の真数値型では精度と位取りを必ず指定する必要があります。

text データ型

TEXT データのサポートは、Adaptive Server Enterprise、SQL Anywhere、Sybase IQ でそれぞれ異なります。

- Adaptive Server Enterprise は、LONG VARBINARY (SQL Anywhere では LONG BINARY) および TEXT のサイズとして、最大 2GB をサポートします。SQL Anywhere では LONG VARBINARY をカラム・タイプとして使用できませんが、同じ目的で LONG BINARY を使用します。SQL Anywhere は、LONG BINARY および TEXT のサイズとして最大 2GB をサポートします。
- Sybase IQ は、VARCHAR のサイズとして最大 32KB - 1 バイトをサポートします。Sybase IQ はさらに、LONGVARCHAR のサイズとして、最大 512TB (IQ ページ・サイズが 128KB の場合) および 2PB (IQ ページ・サイズが 512KB の場合) をサポートしています。Sybase IQ の LONG VARCHAR データ型の詳細については、『Sybase IQ の非構造化データ分析の概要』を参照してください。

image データ型

IMAGE データのサポートは、Adaptive Server Enterprise、SQL Anywhere、Sybase IQ でそれぞれ異なります。

- Adaptive Server Enterprise と SQL Anywhere は、最大 2GB の IMAGE をサポートしています。
- Sybase IQ は LONGBINARY のサイズとして、最大 512TB (IQ ページ・サイズが 128KB の場合) および 2PB (IQ ページ・サイズが 512KB の場合) をサポートしています。Sybase IQ の LONG BINARY データ型の詳細については、『Sybase IQ の非構造化データ分析の概要』を参照してください。

Java データ型

Adaptive Server Enterprise では、データベースで Java データ型を使用できますが、SQL Anywhere と Sybase IQ では使用できません。

データ定義言語

Adaptive Server Enterprise、SQL Anywhere、Sybase IQ では、データベースおよびデータベース・オブジェクトの作成方法がそれぞれ異なります。

Sybase Central から Transact-SQL と互換性のあるデータベースを作成する

大文字と小文字を区別する設定を使用して、Transact-SQL と互換性のあるデータベースを作成します。この設定によって、Adaptive Server Enterprise がエミュレートされます。

1. データベース作成ウィザードで、[照合順の指定] ページにアクセスします。
2. [文字列比較における大文字と小文字の区別] を ON に設定します。この設定によって、Adaptive Server Enterprise がエミュレートされます。
3. [パスワードで大文字と小文字を区別する] をオンに設定します。

これらの大文字と小文字の区別の設定は、Adaptive Server Enterprise をエミュレートします。

CREATE DATABASE 文を使用して Transact-SQL と互換性のあるデータベースを作成する

Interactive SQL を使用して、Transact-SQL と互換性のあるデータベースを作成します。
次の文を、Interactive SQL などで入力します。

```
CREATE DATABASE 'db-name.db' CASE RESPECT BLANK PADDING ON
```

大文字と小文字の区別

データベースにおける大文字と小文字の区別とは、データ、識別子、パスワードに対する大文字と小文字の区別のことです。

データの大文字と小文字の区別

データの大文字と小文字を区別するかしないかは、インデックス、クエリの結果などに反映されます。

Sybase IQ のデータ比較を行う場合の大文字と小文字の区別については、データベース作成時に決定します。 Sybase IQ データベースではデフォルトで、常に大文字と小文字を区別して、入力されたとおりのデータが保持されますが、比較時には大文字と小文字が区別されます。

Adaptive Server Enterprise が大文字と小文字を区別するかどうかは、Adaptive Server Enterprise システム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・バイト文字セットについては、Adaptive Server Enterprise のソート順を再設定することで、大文字と小文字を区別するかどうかを変更できます。

識別子の大文字と小文字の区別

識別子には、テーブル名、カラム名、ユーザ ID などがあります。

Sybase IQ は、大文字と小文字を区別する識別子をサポートしていません。Adaptive Server Enterprise での、識別子による大文字と小文字の区別は、データの大文字と小文字の区別に従います。

Adaptive Server Enterprise では、ユーザ定義データ型の名前については、大文字と小文字を区別します。 Sybase IQ では、大文字と小文字は区別されません。

ユーザ ID とパスワードの大文字と小文字の区別

パスワードの大文字と小文字の区別は、他の識別子とは異なります。

Sybase IQ と SQL Anywhere の場合、新しく作成されたデータベースでは、データベース内の設定に関係なく、すべてのパスワードの大文字と小文字が区別されます。デフォルトのユーザ ID は DBA で、このユーザのパスワードは小文字の *sql* です。

既存のデータベースを再構築する場合、Sybase IQ と SQL Anywhere でのパスワードの大文字と小文字の区別は、次のように決まります。

- パスワードを最初に入力したのが大文字と小文字を区別しないデータベースだった場合、パスワードの大文字と小文字は区別されません。
- パスワードを最初に入力したのが大文字と小文字を区別するデータベースだった場合、大文字と小文字が区別されるのは、大文字のパスワード、および大文字と小文字が混在したパスワードです。パスワードをすべて小文字で入力した場合、大文字と小文字は区別されません。
- 既存のパスワードでも新しいパスワードでも、変更されると大文字と小文字の区別が有効になります。

Adaptive Server Enterprise では、ユーザ ID およびパスワードの大文字と小文字の区別は、サーバの大文字と小文字の区別に従います。

オブジェクト名の互換性の確保

各データベース・オブジェクトは、特定のネーム・スペース内でユニークな名前を持っている必要があります。

ネーム・スペース外では重複した名前があってもかまいません。Adaptive Server Enterprise の一部のデータベース・オブジェクトは、SQL Anywhere および Sybase IQ とは異なるネーム・スペースを使用します。

テーブル名のユニーク性
テーブル名は、データベース内で一意である必要があります。

- Sybase IQ と SQL Anywhere では、テーブル名は、特定の所有者のデータベース内で一意である必要があります。 たとえば、`user1` と `user2` の両方が `employee` という名前のテーブルを作成できます。この場合は、完全修飾名の `user1.employee` と `user2.employee` によってテーブルを一意に識別できます。
- Adaptive Server Enterprise のテーブル名は、データベース内で一意であり、所有者に対して一意である必要があります。

インデックス名のユニーク性
インデックス名は、テーブル内でユニークである必要があります。3 つの製品ではいずれも、インデックスが作成されたテーブルの所有者が、そのインデックスを所有します。ある 1 つのテーブルではインデックス名はユニークである必要がありますが、どの 2 つのテーブルも、所有者が同じであっても同一のインデックス名を持つことができます。たとえば、3 つの製品ではいずれも、テーブル `t1` と `t2` は、所有者が同じかどうかに関係なく、同じインデックス名を持つことができます。

インデックスと外部キーの名前を変更する
Sybase IQ では、**ALTER INDEX** 文を使って、明示的に作成されたインデックス、インデックスの外部キー・ロール名、外部キーの名前を変更できます。SQL Anywhere では、**ALTER INDEX** 文を使って、インデックス、外部キー・ロール名、外部キーを変更できます。Adaptive Server Enterprise でこれらのオブジェクトの名前を変更することはできません。

CREATE TABLE 文使用時の考慮事項

互換性のあるテーブルを作成するときは、NULL の取り扱い、検査制約、参照整合性、デフォルト値、カラムの識別、計算カラム、テンポラリ・テーブル、テーブルのロケーションの互換性を考慮してください。

カラム内の NULL
互換性のある NULL の取り扱い

- SQL Anywhere と Sybase IQ では、カラムの定義で NOT NULL が宣言されていないかぎり、カラムが null になることを認めます。**ALLOW_NULLS_BY_DEFAULT** データベース・オプションを、Transact-SQL の設定に準じて OFF に設定することで、この動作を変更できます。
- SQL Anywhere では、BIT カラムのみが NULL になれないと想定します。
- Adaptive Server Enterprise では、NULL が宣言されないかぎり、カラムが null であることは認められません。

検査制約
Sybase IQ では、ベース・テーブル、グローバル・テンポラリ・テーブル、ローカル・テンポラリ・テーブル、ユーザ定義のデータ型に検査制約を適用します。ユーザは、検査整合性制約の違反のログを記録して、**LOAD** 文がロールバックするまでに発生を許可する違反の数を指定できます。

Sybase IQ では、ユーザ定義関数、プロキシ・テーブル、非 Sybase IQ テーブルなどで構成された検査制約のように、評価できない検査制約の作成は許可されません。 評価できない制約は、検査制約が定義されているテーブルが、**LOAD** 文、**INSERT** 文、または **UPDATE** 文で最初に使用されたときに検出されます。Sybase IQ では、以下を含む検査制約が許可されません。

- サブクエリ
- データ値のターゲットとして、ホスト言語パラメータ、SQL パラメータ、またはカラムを指定している式
- Set 関数
- 非決定的関数、またはデータを変更する関数の呼び出し

Adaptive Server Enterprise と SQL Anywhere は **CHECK** 制約を適用します。SQL Anywhere はサブクエリを検査制約で許可します。

Sybase IQ がサポートするユーザ定義データ型では、データ型の定義内に制約をカプセル化できます。

参照整合性制約
Sybase IQ では、『システム管理ガイド：第 1 巻』で説明されているように、参照整合性が適用されます。

整合性の適用について次のようなアクションがサポートされています。

- SQL Anywhere は、SET NULL、CASCADE、DEFAULT、RESTRICT のすべての ANSI アクションをサポートしています。
- Adaptive Server Enterprise は、上記のアクションのうち、SET NULL と DEFAULT の 2 つをサポートしています。

  注意： Adaptive Server Enterprise で CASCADE を行うには、参照整合性ではなくトリガを使用します。
- Sybase IQ は RESTRICT アクションのみをサポートしています。
- Sybase IQ は NOT NULL FOREIGN KEY をサポートしていません。
- Sybase IQ には制約があり、カラムが候補キーと外部キーの両方に同時になることはできません。

カラムのデフォルト値
デフォルト値のサポートは、次のようにそれぞれ異なります。

- Adaptive Server Enterprise と SQL Anywhere では、カラムにデフォルト値を指定できます。
- DEFAULT UTC TIMESTAMP をサポートするのは SQL Anywhere だけです。
- Sybase IQ では、カラムにデフォルト値を指定できますが、特殊な値である DEFAULT UTC TIMESTAMP と DEFAULT CURRENT UTC TIMESTAMP は指定できません。Sybase IQ は、`DEFAULT_TIMESTAMP_INCREMENT` データベース・オプションの設定も無視します。

identity カラム
identity カラムのサポートには次のような違いがあります。

- Sybase IQ は、デフォルト値として `IDENTITY` または `DEFAULT AUTOINCREMENT` をサポートします。Sybase IQ では、精度が任意で、位取りが 0 の数値型の identity カラムをサポートしています。カラムは NULL でもかまいません。Sybase IQ の identity カラムは正の数である必要があり、データ型の範囲によって制限されます。Sybase IQ では、1 つのテーブルにつき 1 つの identity カラムをサポートしています。明示的な挿入や更新の場合には、データベース・オプション **IDENTITY_INSERT** をテーブル名に設定する必要があります。`IDENTITY` カラムを持つテーブルを削除するときは、**IDENTITY_INSERT** にそのテーブルを指定しておくことはできません。 identity カラムの追加時には、テーブルにデータが含まれていてもかまいません。 **SELECT INTO** を使用して作成したテーブルには、Identity/Autoincrement カラムがありません。Sybase IQ のビューには、`IDENTITY/DEFAULTAUTOINCREMENT` カラムを含めることはできません。

- SQL Anywhere は AUTOINCREMENT デフォルト値をサポートします。SQL Anywhere は、可能なあらゆる位取りと精度を持つ任意の数値型の identity カラムをサポートしています。 identity カラムの値は、正、負、ゼロのいずれでもかまいませんが、データ型の範囲によって制限されます。SQL Anywhere では、テーブル内で任意の数の identity カラムをサポートしています。明示的な挿入、削除、更新の場合に identity_insert は必要ありません。 identity カラムの追加時には、テーブルは空になっている必要があります。SQL Anywhere では、identity カラムを非 identity カラムに、またその逆に変更できます。 SQL Anywhere ビューでは、AUTOINCREMENT カラムを追加または削除できます。
- Adaptive Server Enterprise では、1 つのテーブルにつき 1 つの identity カラムがサポートされます。Adaptive Server Enterprise の identity カラムは、位取り 0、最大精度 38 の数値データ型に限られます。また、値は正の値である必要があり、データ型の範囲によって制限され、null であってはなりません。Adaptive Server Enterprise では、明示的な挿入と削除には identity_insert が必要ですが、identity カラムの明示的な更新には必要ありません。 identity カラムの追加時に、テーブルにデータが含まれていてもかまいません。Adaptive Server Enterprise ユーザは、identity カラムとして選択される次の値を明示的に設定することはできません。Adaptive Server Enterprise ビューには、IDENTITY/AUTOINCREMENT カラムを含めることはできません。**SELECT INTO** を使用するとき、Adaptive Server Enterprise では、条件によっては、元のテーブルに Identity/Autoincrement カラムが含まれている場合、結果テーブルにもこれらを含めることができます。

計算カラム
計算カラムのサポートには次の違いがあります。

- SQL Anywhere は、計算カラムへのインデックスの設定をサポートしています。
- Adaptive Server Enterprise および Sybase IQ ではサポートされていません。

テンポラリ・テーブル
テンポラリ・テーブルを作成するには、所有者を指定しないで、**CREATE TABLE** 文の中でテーブル名の前にポンド記号 (#) を指定します。 このようなテンポラリ・テーブルは、Sybase IQ の宣言されたテンポラリ・テーブルであり、現在の接続内でのみ使用できます。

テーブルの配置
テーブルの物理的な配置方法は、Adaptive Server Enterprise と Sybase IQ では異なります。Sybase IQ は **ON** *segment-name* 句をサポートしますが、*segment-name* は Sybase IQ の DB 領域を参照します。

CREATE DEFAULT 文、CREATE RULE 文、CREATE DOMAIN 文使用時の考慮事項

Sybase IQ には、ルールを組み込む別の方法が用意されています。

- Adaptive Server Enterprise は、名前付きデフォルトを作成するための、Create Default 文および Create Rule 文をサポートしています。
- SQL Anywhere と Sybase IQ は同じ目的のために **CREATE DOMAIN** 文をサポートしています。

CREATE TRIGGER 文使用時の考慮事項

トリガのサポートは、Adaptive Server Enterprise、SQL Anywhere、Sybase IQ でそれぞれ異なります。

- SQL Anywhere は、ロー・レベルと文レベル両方のトリガをサポートしています。
- Adaptive Server Enterprise は、文レベルのトリガのみをサポートしています。
- Sybase IQ ではトリガをサポートしていません。

注意： トリガは、実質的にはストアド・プロシージャであり、**INSERT**、**UPDATE**、**DELETE** の直前もしくは直後に、同じトランザクションの一部として自動的に実行され、依存する変更 (従業員が別の部署に異動になったときに上司の名前を自動的に更新するなど) を行うために利用できます。また、トリガは、どの修正がデータベースにどの変更をいつ加えたのか、を識別するための監査証跡を書き込むのに利用することもできます。

CREATE INDEX 文使用時の考慮事項

CREATE INDEX 構文は、Adaptive Server Enterprise、SQL Anywhere、Sybase IQ でそれぞれ多少異なります。

- Adaptive Server Enterprise と SQL Anywhere では、次の構文で、クラスタード・インデックスまたはノンクラスタード・インデックスをサポートしています。

```
CREATE [UNIQUE] [CLUSTERED] INDEX name

ON table (column,...)
ON dbspace
```

  Adaptive Server Enterprise でも **NONCLUSTERED** キーワードを使用できます。どちらの製品でもデフォルトは **NONCLUSTERED** です。
- Adaptive Server Enterprise の **CREATE INDEX** 文は SQL Anywhere でも機能しますが、SQL Anywhere は **FILLFACTOR**、**IGNORE_DUP_KEY**、**SORTED_DATA**、**IGNORE_DUP_ROW**、**ALLOW_DUP_ROW** の各キーワードを黙認して無視します。

- SQL Anywhere の **CREATE INDEX** 構文は、Index Consultant で使用する **VIRTUAL** キーワードをサポートしていますが、クエリの実行ではサポートしていません。
- Sybase IQ は特殊なインデックスの種類として、 **LF**、**HG**、**HNG**、**DATE**、**TIME**、**DTTM**、**WD** をサポートします。Sybase IQ では、同じデータ型、精度、位取りを持つ 2 つのカラム間の関係に対する **CMP** インデックスもサポートしています。**CREATE INDEX** 文でインデックスの種類を指定しないかぎり、Sybase IQ はデフォルトで **HG** インデックスを作成します。

```
CREATE [UNIQUE] [type] INDEX name
ON table (column,...)
```

注意： Sybase IQ では、**CREATE JOIN INDEX** もサポートしています。これにより、クエリで定期的かつ頻繁にジョインされるカラムの組み合わせに、ジョイン済みのインデックスを作成できます。

ユーザ、グループ、パーミッション

Adaptive Server Enterprise のユーザとグループのモデルは、SQL Anywhere および Sybase IQ と多少異なります。

Adaptive Server Enterprise では、各ユーザがサーバに接続する場合、サーバに対するログイン ID とパスワード、およびそのサーバ上でアクセスする各データベースに対するユーザ ID が必要です。

SQL Anywhere と Sybase IQ では、ユーザはサーバ・ログイン ID を必要としません。SQL Anywhere と Sybase IQ のすべてのユーザが、データベース用のユーザ ID とパスワードを受け取ります。

ユーザ・グループ
3 つの製品のいずれもユーザ・グループをサポートしているので、複数のユーザに一度にパーミッションを付与できます。ただし、グループの詳細が異なります。

- Adaptive Server Enterprise では、各ユーザがメンバになれるのは 1 つのグループのみです。
- SQL Anywhere と Sybase IQ では、ユーザは複数のグループのメンバになることができ、グループを階層構造にすることができます。

3 つの製品すべてに、デフォルトのパーミッションを定義するための `public` グループがあります。すべてのユーザは、自動的に `public` グループのメンバになります。

データベース・オブジェクトのパーミッション
個々のデータベース・オブジェクトにパーミッションを付与する **GRANT** 文と **REVOKE** 文は、3 つの製品でかなり共通しています。

- いずれの製品でも、SELECT、INSERT、DELETE、UPDATE、REFERENCES のパーミッションをデータベース・テーブルおよびビューに設定でき、UPDATE パーミッションをデータベース・テーブルの特定のカラムに設定できます。たとえば、次の文は 3 つの製品のいずれにおいても有効です。

```
GRANT INSERT, DELETE
ON TITLES
TO MARY, SALES
```

  この文は、TITLES テーブルで **INSERT** 文と **DELETE** 文を使用するパーミッションを、ユーザ MARY と SALES グループに与えます。
- すべての製品で、ストアド・プロシージャに EXECUTE パーミッションを与えることができます。
- Adaptive Server Enterprise ではさらに、次の項目に対する **GRANT** と **REVOKE** がサポートされます。
  - オブジェクト： テーブル内のカラム、ビュー内のカラム、ストアド・プロシージャ
  - ユーザが可能な操作： **CREATE DATABASE**, **CREATE DEFAULT**, **CREATE PROCEDURE**, **CREATE RULE**, **CREATE TABLE**, **CREATE VIEW**
- SQL Anywhere と Sybase IQ でユーザがデータベース・オブジェクトを作成するには、RESOURCE 権限が必要です (Adaptive Server Enterprise でこれにほぼ相当するパーミッションは、データベース所有者が使用する GRANT ALL です)。
- 3 つの製品はいずれも **WITH GRANT OPTION** 句をサポートしています。この句では、パーミッションを付与されたユーザは、そのパーミッションを他のユーザに付与できます。ただし、Sybase IQ の場合、**WITH GRANT OPTION** を **GRANT EXECUTE** 文では使用できません。

データベース全体に適用されるパーミッション
Adaptive Server Enterprise では、データベース全体のユーザ・パーミッションには別のモデルを使用します。

- SQL Anywhere と Sybase IQ では DBA パーミッションを使用して、ユーザに個々のデータベース内での完全な権限を与えます。
- Adaptive Server Enterprise のシステム管理者は、このパーミッションをサーバ上のすべてのデータベースに対して使用できます。 ただし、Sybase IQ データベース上での DBA 権限は、Adaptive Server Enterprise のデータベース所有者のパーミッションとは異なります。Adaptive Server Enterprise のデータベース所有者が、他のユーザが所有するオブジェクトに対するパーミッションを得るには、Adaptive Server Enterprise の **SETUSER** 文を使用します。

ユーザの追加
Adaptive Server Enterprise でユーザを追加するには、 **sp_addlogin** と **sp_adduser** の 2 つのステップが必要です。

SQL Anywhere と Sybase IQ では 1 つのステップでユーザを追加できます。

Sybase IQ ログイン管理のストアド・プロシージャを使用すると、ユーザを追加または削除する必要がなくても、DBA は Sybase IQ ユーザ・アカウントを追加または削除できます。 Sybase IQ ユーザ管理が有効になっていれば、DBA は Sybase IQ のユーザ・アカウントのユーザ接続やパスワードの期限を管理できます。

SQL Anywhere と Sybase IQ では、Adaptive Server Enterprise のシステム・プロシージャを使ってユーザやグループを管理できますが、これらのプロシージャの厳密な構文や機能は同じでない場合があります。

参照：

- Adaptive Server Enterprise のシステム・プロシージャ (597 ページ)

ロード形式

ロード形式のサポートは、Adaptive Server Enterprise、SQL Anywhere、Sybase IQ でそれぞれ異なります。

- Sybase IQ は ASCII、BINARY、および BCP ロード形式を扱います。
- SQL Anywhere では、ASCII と BINARY の他に、dBase、Excel、FoxPro、Lotus ファイル形式をインポートできます。
- Adaptive Server Enterprise は、BCP を通じて ASCII および BINARY のロード形式を扱います。

注意： Sybase IQ と SQL Anywhere の **LOAD** 文の構文は BCP に基づいており、まったく同じ機能を果たします。

Transact-SQL 互換のオプション

Sybase IQ データベース・オプションは、**SET OPTION** 文を使用して設定します。

『リファレンス：文とオプション』の「Transact-SQL 互換性オプション」を参照してください。

データ操作言語

クエリ要件は、Adaptive Server Enterprise、SQL Anywhere、Sybase IQ でそれぞれ異なります。

移植可能な SQL を記述するための一般的なガイドライン

指定した SQL 文を複数のサーバがサポートしている場合でも、デフォルトの動作が各システムで同じであると仮定するのは間違っていることもあります。

次に、互換性のある SQL を記述するときの一般的なガイドラインを示します。

- 複数のデータベース管理システムで使用するために SQL を記述する場合は、SQL 文をできるだけ明示的にします。
- デフォルトの動作を使用しないで、使用可能なオプションをすべて略さずに書きます。
- 演算子の優先順位のデフォルトをそのまま適用する場合でも、カッコを使用して文中での実行順序を明示的に指定します。
- Adaptive Server Enterprise に移植できるように、変数名には @ をプレフィクスとして付ける Transact-SQL の規則に従います。
- **BEGIN** 文のすぐ後で、プロシージャとバッチ内の変数とカーソルを宣言します。Sybase IQ ではこれが必要ですが、Adaptive Server Enterprise ではプロシージャまたはバッチ内のどこでも宣言できます。
- Adaptive Server Enterprise または Sybase IQ の予約語を、データベース内の識別子として使用しないでください。

互換性のあるクエリの記述方法の基準

Sybase IQ と Adaptive Server Enterprise の両方のデータベースで実行されるクエリの記述方法には、基準が 2 つあります。

- クエリ中のデータ型、式、探索条件が互換性を持つこと。
- **SELECT** 文そのものの構文が互換性を持つこと。

Sybase IQ がサポートする、Transact-SQL の **SELECT** 文のサブセットを次に示します。

構文

```
SELECT [ ALL | DISTINCT ] select-list
…[ INTO  #temporary-table-name ]
…[ FROM  table-spec,
…     table-spec, … ]
…[ WHERE  search-condition ]
…[ GROUP
                BY  column-name, … ]
…[ HAVING  search-condition ]
…| [ ORDER
                BY  expression [ ASC | DESC ], … ]     |
   | [ ORDER
                BY  integer [ ASC | DESC ], … ]     |
```

パラメータ

```
 select-list:
{ table-name.* }…
{ * }…
{ expression }…
{ alias-name = expression }…
{ expression as identifier }…
{ expression as T_string }…
```

```
table-spec:
   [ owner. ]table-name
…     [ [ AS ] correlation-name ]
…
```

```
alias-name:
    identifier | 'string' | “string"
```

次の項では、互換性のあるクエリを作成するために注意する必要のある事項を説明します。

参照：

- Transact-SQL プロシージャ内の変数 (736 ページ)

サブクエリのサポート

Sybase IQ でのサブクエリのサポートは、現在、Adaptive Server Enterprise および SQL Anywhere でのサポートとは多少異なります。

Adaptive Server Enterprise と SQL Anywhere は **ON** 句内のサブクエリをサポートしています。Sybase IQ は現在、これをサポートしていません。

サブクエリ内の **UNION** は次のようにサポートされます。

- SQL Anywhere では、**UNION** を相関サブクエリと非相関サブクエリの両方でサポートしています。
- Sybase IQ では、非相関クエリでのみ **UNION** をサポートしています。
- Adaptive Server Enterprise では、どのサブクエリにおいても **UNION** をサポートしていません。

SQL Anywhere は、文法にスカラ値が登場するその他の多くの場面でサブクエリをサポートしています。Adaptive Server Enterprise と Sybase IQ では、サブクエリを指定できる場所について ANSI 標準に従っています。

GROUP BY 句のサポート

GROUP BY ALL のサポートは、Adaptive Server Enterprise、SQL Anywhere、Sybase IQ でそれぞれ異なります。

- Adaptive Server Enterprise は、**WHERE** 句および **HAVING** 句によって除外されたグループも含む、すべてのグループを返す **GROUP BY ALL** をサポートしています。この場合、集約はすべて NULL 値になります。
- SQL Anywhere は、**GROUP BY ALL** Transact-SQL 拡張機能をサポートしていません。

GROUP BY 句の **ROLLUP** と **CUBE** は次のようにサポートされます。

- Sybase IQ と SQL Anywhere は、**GROUP BY** 句内での **ROLLUP** および **CUBE** をサポートします。
- Adaptive Server Enterprise は現在、**ROLLUP** と **CUBE** をサポートしていません。

Adaptive Server Enterprise は、**SELECT** 句内のグループ化されていないカラムの表示をサポートしています。 これは、セマンティックによる拡張グループと呼ばれ、一連の値を返します。Sybase IQ はセマンティックによる拡張グループをサポートしていますが、SQL Anywhere はサポートしていません。 値のリストを返す List() 集約関数をサポートしているのは、SQL Anywhere のみです。

COMPUTE 句のサポート

COMPUTE のサポートは、Adaptive Server Enterprise、SQL Anywhere、Sybase IQ でそれぞれ異なります。

- Adaptive Server Enterprise は、Transact-SQL の **COMPUTE** 句をサポートしています。
- Transact-SQL の **COMPUTE** 句は、ANSI 標準になく、また、この機能はほとんどがサード・パーティのフロントエンド・ツールによって実現されているため、SQL Anywhere と Sybase IQ ではサポートしていません。

WHERE 句のサポート

WHERE 句での Contains() 述部のサポートと、**Like()** 述部での末尾の空白の取り扱いは、Adaptive Server Enterprise、SQL Anywhere、Sybase IQ でそれぞれ異なります。

- Sybase IQ は、文字データ内のワード検索で **Contains()** 述部をサポートしています (MS SQL Server や Verity での Contains と同様)。Sybase IQ は、可能な場合、`WORD` インデックスと `TEXT` インデックスを使用して検索を最適化します。
- Adaptive Server Enterprise は **Contains()** をサポートしません。

Transact-SQL 外部ジョインのサポート

外部ジョインでサポートされている構文は、Adaptive Server Enterprise、SQL Anywhere、Sybase IQ でそれぞれ異なります。

- Adaptive Server Enterprise は、*= および =* の Transact-SQL 外部ジョイン構文をサポートしています。
- SQL Anywhere と Sybase IQ は、Transact-SQL の外部ジョインをサポートしていますが、曖昧になる可能性のある一部の複雑な Transact-SQL 外部ジョインは拒否されます。
- Sybase IQ は、Transact-SQL の連鎖 (ネストした) 外部ジョインをサポートしていません。 このような複数の外部ジョインには、ANSI 構文を使用してください。

注意： Transact-SQL の外部ジョイン構文は SQL Anywhere と Sybase IQ では非推奨です。

ANSI 構文の代替を含む、Transact-SQL 外部ジョインの詳細情報については、MySybase で閲覧可能なホワイト・ペーパー『Semantics and Compatibility of Transact-SQL Outer Joins』を参照してください。 これは SQL Anywhere 用に書かれたものですが、このホワイト・ペーパー内の情報は Sybase IQ にも適用できます。

ANSI ジョインのサポート

ANSI ジョイン構文のサポートは、Adaptive Server Enterprise、SQL Anywhere、Sybase IQ でそれぞれ異なります。

- Sybase IQ は現在、**ON** 句内のサブクエリをサポートしていません。
- Adaptive Server Enterprise と SQL Anywhere は、**ON** 句内でのサブクエリをサポートします。
- ANSI ジョイン構文を使用するクエリでの **FROM** 句内の **CONTAINS** 条件はサポートされますが、最良のパフォーマンスが得られない場合があります。**FROM** 句で **CONTAINS** に外部ジョインを使用するのは、各 **CONTAINS** 句からの "score" カラムが必要な場合のみにしてください。それ以外の場合は、**CONTAINS** を **ON** 条件または **WHERE** 句に移動してください。

完全外部ジョインのサポートは次のとおりです。

- SQL Anywhere と Sybase IQ は **FULL OUTER JOIN** をサポートします。
- Adaptive Server Enterprise は **FULL OUTER JOIN** をサポートしません。

Null 比較のサポート

Adaptive Server Enterprise には、NULL 値を比較する述部を許可する Transact-SQL 拡張機能が用意されています。

たとえば、`{col} = Null` は `{col} Is Null` を意味します。

SQL Anywhere と Sybase IQ は、`ANSINULL` オプションが OFF に設定されていないかぎり、Null 比較に ANSI セマンティックを使用します。OFF に設定されている場合は、比較は Adaptive Server Enterprise 互換で行われます。

注意： SQL Anywhere 8.0 以降では、`TDS_EMPTY_STRING_AS_NULL` のサポートが追加されています。これにより、NULL 値に空の文字列をマッピングする際、Adaptive Server Enterprise との互換性を実現できます。

長さがゼロの文字列のサポート

長さがゼロの文字列の取り扱い方法は、Adaptive Server Enterprise、SQL Anywhere、Sybase IQ でそれぞれ異なります。

- Adaptive Server Enterprise は、長さがゼロの文字列を null 値とみなします。
  Adaptive Server Enterprise で空の文字列を作成するには、空白を 1 つ格納します。
- SQL Anywhere と Sybase IQ は、ANSI セマンティックに従い、長さがゼロの文字列を null ではなく値として扱います。

HOLDLOCK、SHARED、FOR BROWSE のサポート

HOLDLOCK、SHARED、FOR BROWSE の構文は、Adaptive Server Enterprise、SQL Anywhere、Sybase IQ でそれぞれ異なります。

- Adaptive Server Enterprise は、HOLDLOCK、SHARED、FOR BROWSE の構文をサポートします。
- SQL Anywhere は HOLDLOCK をサポートしていますが、SHARED と FOR BROWSE はサポートしていません。
- Sybase IQ はこれらのキーワードをサポートしていません。

SQL 関数のサポート

Sybase IQ は SQL Anywhere や Adaptive Server Enterprise とほぼ同じ関数をサポートしていますが、次のような違いがあります。

- Adaptive Server Enterprise は **PatIndex()** での **USING CHARACTERS | USING BYTES** 構文をサポートしますが、SQL Anywhere および Sybase IQ はサポートしません。
- Adaptive Server Enterprise は **Reverse()** 関数をサポートしますが、SQL Anywhere と Sybase IQ はサポートしません。
- Adaptive Server Enterprise は **Len()** を **Length()** の代わりの構文としてサポートしていますが、SQL Anywhere では代わりとして使用できません。
- Adaptive Server Enterprise は **Square()** と **Str_Replace()** を Microsoft 互換の関数としてサポートしていますが、SQL Anywhere はサポートしていません。
- Sybase IQ は **Str_Replace()** をサポートしています。
- Adaptive Server Enterprise と SQL Anywhere では、変更時刻を確認するための 2 つのタイムスタンプの比較に **TSEQUAL()** をサポートしていますが、Sybase IQ は **TSEQUAL()** をサポートしていません (**TSEQUAL** は Sybase IQ テーブル・レベルのバージョン・モデルには関係しません)。

- Sybase IQ は **ROWID()** をサポートしていますが、Adaptive Server Enterprise と SQL Anywhere はサポートしていません。
- SQL Anywhere と Sybase IQ は、データ型の変換に使用する Adaptive Server EnterpriseAdaptive Server Enterprise の **Convert()** に加えて、**Cast()** をサポートします。

  注意： **Cast()** は ANSI 互換の名前です。
- SQL Anywhere と Sybase IQ は、**Lcase()** と **Ucase()** を **Lower()** と **Upper()** の同意語としてサポートしていますが、Adaptive Server Enterprise はサポートしていません。
- SQL Anywhere と Sybase IQ は **Locate()** 文字列関数をサポートしますが、Adaptive Server Enterprise はサポートしません。
- SQL Anywhere は、文字列をそれぞれのデータ型に変換できるかどうかをテストする **IsDate()** 関数と **IsNumeric()** 関数をサポートしていますが、Adaptive Server Enterprise はサポートしていません。Sybase IQ は **IsDate()** をサポートしています。**IsNumeric** は Sybase IQ で使用できますが、CIS 機能補正のパフォーマンスへの影響を考慮する必要があります。
- SQL Anywhere は、**NEWID**、**STRTOUID**、**UUIDTOSTR** の各関数をサポートしますが、Adaptive Server Enterprise はサポートしません。 これらの関数は Sybase IQ のネイティブ関数なので、CIS 機能補正のパフォーマンスへの影響を考慮する必要はありません。

注意： **SOUNDEX** 文字列関数、**DIFFERENCE** 文字列関数、日付関数の一部など、Sybase IQ と SQL Anywhere では動作が異なる SQL 関数があります。Sybase IQ データベース・オプションの `ASE_FUNCTION_BEHAVIOR` は、一部の Sybase IQ データ型変換関数 (**HEXTOINT**、**INTTOHEX** など) の出力が、Adaptive Server Enterprise の関数の出力と一致するように指定します。

OLAP 関数のサポート

現在、Adaptive Server Enterprise は OLAP 関数をサポートしていません。Sybase IQ と SQL Anywhere ではサポートされます。

Sybase IQ が現在サポートしている OLAP 関数は次のとおりです。

- **Corr()**
- **Covar_Pop()**
- **Covar_Samp()**
- **Cume_Dist**
- **Dense_Rank()**
- **Exp_Weighted_Avg**
- **First_Value**

- **Last_Value**
- **Median**
- **Ntile()**
- **Percent_Rank()**
- **Percentile_Cont()**
- **Percentile_Disc()**
- **Rank()**
- **Regr_Avgx()**
- **Regr_Avgy()**
- **Regr_Intercept()**
- **Regr_R2**
- **Regr_Slope()**
- **Regr_Sxx()**
- **Regr_Sxy()**
- **Regr_Syy()**
- **StdDev()**
- **Stddev_Pop**
- **Stddev_Samp**
- **Var_Pop**
- **Var_Samp**
- **Variance()**
- **Weighted_Avg**

SQL Anywhere はすべての Sybase IQ OLAP 関数をサポートしています。

現在、Adaptive Server Enterprise は OLAP 関数をサポートしていません。

CIS の機能補正では、OLAP 関数はサポートされていません。

注意： OLAP 関数のサポートは、Sybase の製品開発で急速に発展している分野です。

システム関数のサポート

SQL Anywhere と SQL AnywhereSybase IQ は、一部の Adaptive Server Enterprise システム関数をサポートしていません。

SQL Anywhere と Sybase IQ でサポートされない Adaptive Server Enterprise システム関数は次のとおりです。

- **curunreservedpgs()** – DB 領域の空きページ数。
- **data_pgs()** – 個々のテーブルまたはインデックスによって使用されているページ数。

- **host_id()** – サーバ・プロセスの UNIX pid。
- **host_name()** – サーバを実行しているマシンの名前。
- **lct_admin()** – トランザクション・マネージャの「ラストチャンス・スレッショルド」の管理に使用する。
- **reserved_pgs()** – テーブルまたはインデックスに割り当てられたページ数。
- **rowcnt()** – 指定されたテーブル内のロー数。
- **valid_name()** – 名前が、テーブルなどに使用された場合に有効かどうかを示す。
- **valid_user()** – そのユーザが接続パーミッションを持っている場合に TRUE を返す。
- **ptn_data_pgs()** – パーティション内のデータ・ページ数。
- **index_colorder()** – インデックス内のカラムの順番を返す。

ユーザ定義関数のサポート

ユーザ定義関数 (UDF: User Defined Function) のサポートは、Adaptive Server Enterprise、SQL Anywhere、Sybase IQ でそれぞれ異なります。

- SQL Anywhere は、SQL、Java、C での UDF をサポートしています。
- Adaptive Server Enterprise は、Java で書かれた UDF しかサポートしません。
- Sybase IQ では、CIS クエリ分解を通じて UDF をサポートしていますが、パフォーマンスに影響があります。

日付の算術式の異なる解釈

SQL Anywhere と Sybase IQ は、日付に使われる算術式を、さまざまな日付関数の省略形として解釈します。これは、Adaptive Server Enterprise には当てはまりません。

- 日付 +/- 整数は、**Dateadd()** と同義です。
- 日付 - 日付は、**Datediff()** と同義です。
- 日付 + 時刻は、この 2 つからタイムスタンプを作成します。

SELECT INTO 文のサポート

SELECT INTO 文で許可されるテーブルの種類は、Adaptive Server Enterprise、SQL Anywhere、Sybase IQ でそれぞれ異なります。

SELECT INTO 文の例を見てみましょう。

```
select into table1 from table2
```

- Adaptive Server Enterprise では、*table1* が永久テーブル、テンポラリ・テーブル、プロキシ・テーブルのいずれかになります。Adaptive Server Enterprise では **SELECT INTO EXISTING TABLE** もサポートされます。
- SQL Anywhere と Sybase IQ では、*table1* は永久テーブルまたはテンポラリ・テーブルのいずれかになります。 永久テーブルは、select into *table* を実行して複数のカラムを指定した場合にのみ作成されます。 所有者を指定せずに **SELECT INTO***#table* を実行すると、指定したカラムの数にかかわらずテンポラリ・テーブルが作成されます。1 つしかカラムを持たないテーブルで **SELECT INTO** を実行すると、ホスト変数が選択されます。

更新可能なビューのサポート

Adaptive Server Enterprise と SQL Anywhere では、WITH CHECK オプションが要求されていない場合に更新可能となるビュー定義に関しては ANSI よりも自由度が高くなっています。

SQL Anywhere には、SQL92 でサポートされているもののみを更新可能にするか、あるいはもっと自由度の高いルールを設定するかを制御できる **ANSI_UPDATE_CONSTRAINTS** オプションが用意されています。

Sybase IQ は、フラット化できる単一のテーブルのビューでの **UPDATE** のみをサポートしています。Sybase IQ は **WITH CHECK** をサポートしません。

UPDATE と DELETE の FROM 句のサポート

Sybase IQAdaptive Server Enterprise、SQL Anywhere、Sybase IQ はすべて、UPDATE と DELETE での、複数のテーブルがある FROM 句をサポートしています。

Transact-SQL のプロシージャ言語の概要

ストアド・プロシージャ言語は SQL の一部として、ストアド・プロシージャとバッチで使用されます。

SQL Anywhere と Sybase IQ は、SQL92 に基づく Watcom-SQL ダイアレクトに加えて、Transact-SQL ストアド・プロシージャ言語の大部分をサポートしています。

Transact-SQL のストアド・プロシージャの概要

SQL Anywhere と Sybase IQ のストアド・プロシージャ言語は、ISO/ANSI 規格を基準にしているので、Transact-SQL ダイアレクトとは多くの点で異なっています。

概念と機能の多くは似ていますが、構文が異なります。SQL Anywhere と Sybase IQ の Transact-SQL サポートでは、ダイアレクト間の自動変換を可能にすることで、

同じ概念を活用しています。 ただし、プロシージャはどちらかのダイアレクトだけで記述する必要があり、混在させることはできません。

ここでは、Transact-SQL ストアド・プロシージャに対する SQL Anywhere と Sybase IQ のサポートについて、次のような側面から説明します。

- パラメータを引き渡す
- 結果セットを返す
- ステータス情報を返す
- パラメータにデフォルト値を提供する
- 制御文
- エラー処理

Transact-SQL のバッチの概要

Transact-SQL では、バッチは一緒に送信されてグループとして次々に実行される一連の SQL 文です。

バッチはコマンド・ファイルとして保存できます。SQL Anywhere および Sybase IQ の ISQL ユーティリティと、Adaptive Server Enterprise の `isql` ユーティリティは、バッチを対話型で実行するためのよく似た機能を持っています。

プロシージャで使用される制御文はバッチでも使用できます。SQL Anywhere と Sybase IQ は,バッチでの制御文の使用をサポートしています。また、Transact-SQL のように、一連の文を区切りなしで使用できます。この場合、バッチの終わりを示す **GO** 文で終了します。

コマンド・ファイルに保存されているバッチについて、SQL Anywhere と Sybase IQ は、コマンド・ファイルでのパラメータの使用をサポートしています。Adaptive Server Enterprise はパラメータをサポートしません。

プロシージャとバッチ内の SQL 文

Sybase IQ がサポートする SQL 文には、一方のダイアレクトの一部になっていて、もう一方のダイアレクトの一部になっていないものがあります。

そのため、2 つの言語を 1 つのプロシージャやバッチ内で混在させることはできません。これは次のことを意味します。

- Transact-SQL 専用の文は、両方の言語に属する文と共に、バッチまたはプロシージャ内に含めることができます。
- Adaptive Server Enterprise によってサポートされない文は、両サーバによってサポートされる文とともに、バッチまたはプロシージャ内に含めることができます。

- Transact-SQL 専用の文と Sybase IQ 専用の文は、バッチまたはプロシージャ内に混在させることはできません。

セミコロンで区切られていない SQL 文は、Transact-SQL のプロシージャまたはバッチの一部です。個々の文の詳細については、『リファレンス：文とオプション』を参照してください。

Transact-SQL 互換性が改善され、以前は受け入れられていた誤った SQL 構文がエラーで失敗するようになりました。

IF 文内の式サブクエリ

Adaptive Server Enterprise と SQL Anywhere は、式サブクエリによって返されたスカラ値と変数との比較をサポートしています。

次に例を示します。

```
create procedure testIf ()   begin   declare var4 int; set var4 =
10;   if var4 = (select MIN (a_i1) from a) then set      var4 = 100;
end if; end;
```

CASE 文のサポート

Sybase IQ と SQL Anywhere では、CASE 文の許可されている使用方法が異なります。

Adaptive Server Enterprise では、case 式のみがサポートされ、**CASE** 文はサポートされません。

参照：

- 式 (24 ページ)

ロー・レベルのカーソル演算のサポート

Adaptive Server Enterprise、SQL Anywhere、Sybase IQ では、UPDATE および DELETE でのカーソルの使用がサポートされています。

次の例を考えます。

```
UPDATE WHERE CURRENT OF {cursor}
```

```
DELETE WHERE CURRENT OF {cursor}
```

Sybase IQ では、更新可能なカーソルは、asensitive のみ、1 つのテーブル用のみ、連鎖のみです。 更新可能なホールド・カーソルは許可されていません。 Sybase IQ では、更新可能なカーソルはテーブル・ロックされます。

PRINT コマンドのサポート

PRINT コマンドのサポートは、Adaptive Server Enterprise、SQL Anywhere、Sybase IQ でそれぞれ異なります。

PRINT の実行結果はクライアントによって異なります。

- Adaptive Server Enterprise の **PRINT** は、常にクライアントにメッセージを送信します。
- SQL Anywhere と Sybase IQ では、**PRINT** は Open Client と JDBC 接続のクライアントにメッセージを送信します。
- **PRINT** に依存する Adaptive Server Enterprise のストアド・プロシージャは、Sybase IQ では Interactive SQL を使用して機能します。

注意： Sybase IQ ユーザは、iAdaptive Server Anywhere JDBC ドライバ (以前 JDBC-ODBC ブリッジと呼ばれていたもの) ではなく、JDBC で Interactive SQL を使用するようおすすめします。

ストアド・プロシージャの自動変換

Transact-SQL 代替構文のサポートの他に、SQL Anywhere と Sybase IQ では、Watcom-SQL ダイアレクトと Transact-SQL ダイアレクト間での文の変換を支援します。

次に示す関数は、SQL 文に関する情報を返し、SQL 文を自動変換できるようにします。

表 216：自動変換を有効にする関数

関数	説明
SQLDialect(文)	Watcom-SQL または Transact-SQL を返します。
WatcomSQL(文)	その文の Watcom-SQL 構文を返します。
TransactSQL(文)	Transact-SQL 構文を返します。

これらは関数なので、ISQL から **SELECT** 文を使用してアクセスできます。たとえば、次の文は、Watcom-SQL という値を返します。

```
SELECT SqlDialect('select * from Employees')
```

Transact-SQL プロシージャから返される結果セット

SQL Anywhere/Sybase IQ プロシージャと Transact-SQL プロシージャは、異なる結果セットを返します。

SQL Anywhere と Sybase IQ は、**RESULT** 句を使用して返される結果セットを指定します。

Transact-SQL プロシージャでは、最初のクエリのカラム名またはエイリアス名が呼び出し元の環境に返されます。

次の Transact-SQL プロシージャは、Transact-SQL ストアド・プロシージャが結果セットを返す方法を示します。

```
CREATE PROCEDURE showdept (@deptname varchar(30))
AS
    SELECT Employees.Surname, Employees.GivenName
    FROM Departments, Employees
    WHERE Departments.DepartmentName = @deptname
    AND Departments.DepartmentID = Employees.DepartmentID
```

対応する SQL Anywhere または Sybase IQ のプロシージャは次のようになります。

```
CREATE PROCEDURE showdept(in deptname varchar(30))
RESULT ( lastname char(20), firstname char(20))
BEGIN
    SELECT Employees.Surname, Employees.GivenName
    FROM Departments, Employees
    WHERE Departments.DepartmentName = deptname
    AND Departments.DepartmentID = Employee.DepartmentID
END
```

3 つの Sybase クライアント・ツールが複数の結果セットをクライアントに表示する方法には、わずかな違いがあります。

- **isql** は、 1 つのストリームにすべての結果を表示します。
- Interactive SQL は結果セットをそれぞれ別のタブに表示します。ユーザはこの機能を [オプション] メニューで有効にする必要があります。これを永続的な変更とした後、Interactive SQL を再起動するか再接続してください。
- Interactive SQL Classic (廃止) には、連続した各結果セットを表示する **RESUME** が用意されています。

Transact-SQL プロシージャ内の変数

SQL Anywhere と Sybase IQ は、Transact-SQL とは異なる方法で、プロシージャ内の変数に値を割り当てます。

SQL Anywhere と Sybase IQ は SET 文を使用してプロシージャ中の変数に値を割り当てます。

Transact-SQL では、**SELECT** 文と空のテーブル・リストを使用して値を割り当てます。次の単純なプロシージャは、Transact-SQL 構文の働きを示します。

```
CREATE PROCEDURE multiply
                    @mult1 int,
                    @mult2 int,
                    @result int output
AS
SELECT @result = @mult1 * @mult2
```

このプロシージャを呼び出すには、次のようにします。

```
CREATE VARIABLE @product int
go
EXECUTE multiply 5, 6, @product OUTPUT
go
```

プロシージャの実行後、変数 *@product* の値は 30 になります。

変数宣言の順序と持続性にはいくつかの違いがあります。

- Adaptive Server Enterprise では、ストアド・プロシージャ本体のどこでも変数を宣言できます。 変数はプロシージャの実行中持続します。
- SQL Anywhere と Sybase IQ では、複合文の最初 (つまり、**BEGIN...END** ペア内の **BEGIN** の直後) に変数を宣言する必要があります。 変数は、複合文の実行中のみ持続します。

参照：

- 互換性のあるクエリの記述方法の基準 (724 ページ)

Transact-SQL プロシージャでのエラー処理

デフォルトでのプロシージャのエラー処理は、Watcom-SQL 言語と Transact-SQL 言語では異なります。

デフォルトでは、Watcom-SQL 言語のプロシージャはエラーが起こると終了し、呼び出した環境に SQLSTATE 値と SQLCODE 値を返します。

EXCEPTION 文を使って、Watcom-SQL ストアド・プロシージャに明示的なエラー処理を組み込むことができます。または、**ON EXCEPTION RESUME** 文を使って、エラーが発生したら、その次の文から実行を再開するよう、プロシージャに指示することもできます。

Transact-SQL ダイアレクトのプロシージャでエラーが発生した場合は、その次の文から実行が継続されます。グローバル変数 **@@error** には、最後に実行された文のエラー・ステータスが保存されます。文の後ろにあるこの変数をチェックして、プロシージャから強制的に返すことができます。たとえば、次の文は、エラーが起こると終了させます。

```
IF @@error != 0 RETURN
```

プロシージャが実行を終了したときの戻り値で、プロシージャが成功したかどうかがわかります。この戻り値は整数で、次のように指定してアクセスできます。

```
DECLARE @status INT
EXECUTE @status = proc_sample
IF @status = 0
    PRINT 'procedure succeeded'
ELSE
    PRINT 'procedure failed'
```

次のテーブルは、組み込みプロシージャの戻り値とその意味を示します。

表 217 : 組み込みプロシージャの戻り値

値	意味
0	プロシージャが正常に実行された
-1	オブジェクトがない
-2	データ型エラー
-3	プロセスがデッドロックの被害対象として選択された
-4	パーミッション・エラー
-5	構文エラー
-6	その他のさまざまなユーザ・エラー
-7	領域不足などのリソース・エラー
-8	致命的ではない内部の問題
-9	システムの限界に達した
-10	致命的な内部不整合
-11	致命的な内部不整合
-12	テーブルまたはインデックスが破壊されている
-13	データベースが破壊されている
-14	ハードウェア・エラー

RETURN 文は、この表の値以外の、ユーザが意味を定義した整数値を返すこともできます。

プロシージャ内での RAISERROR 文の使用

RAISERROR 文は、ユーザ定義エラーを生成するための Transact-SQL 文です。これは、**SIGNAL** 文と似た機能を持っています。

RAISERROR 文そのものは、プロシージャを終了しませんが、ユーザ定義エラー後の実行を制御するために、**RETURN** 文やグローバル変数 **@@error** のテストと組み合わせることができます。

ON_TSQL_ERROR データベース・オプションを **CONTINUE** に設定すると、**RAISERROR** 文は実行終了時にエラーを通知しません。代わりに、プロシージャが完了し、**RAISERROR** のステータス・コードとメッセージを保存してから、最新の **RAISERROR** を返します。**RAISERROR** を返したプロシージャが他のプロシージャ

から呼び出された場合、最も外側のプロシージャが終了してから **RAISERROR** が返されます。

途中の **RAISERROR** ステータスとコードは、プロシージャが終了すると失われます。**RAISERROR** が返るときにエラーが発生した場合は、エラー情報が返され、**RAISERROR** 情報は失われます。アプリケーションは異なる実行ポイントで **@@error** グローバル変数を検査して、途中の **RAISERROR** ステータスを問い合わせることができます。

Watcom-SQL ダイアレクトでの Transact-SQL に似たエラー処理

Watcom-SQL ダイアレクトのプロシージャが Transact-SQL に似た方法でエラー処理を行うように、設定できます。

次のように、**ON EXCEPTION RESUME** 句を **CREATE PROCEDURE** 文に指定します。

```
CREATE PROCEDURE sample_proc()
ON EXCEPTION RESUME
BEGIN
    ...
END
```

ON EXCEPTION RESUME 句があると、明示的な例外処理コードは実行されません。このため、これらの 2 つの句は一緒に使用しないでください。

SQL Anywhere と Sybase IQ の相違点および共有機能

Sybase IQ と SQL Anywhere には、データベースおよびサーバの起動と管理、サポートされているデータベース・オプション、DDL サポート、DML サポートの点で違いがあります。

詳細については、Sybase IQ を使用している場合は必ずそのマニュアル・セットを参照してください。 SQL Anywhere を使用している場合や、Sybase IQ のマニュアルで SQL Anywhere の具体的な機能を参照している場合は、SQL Anywhere のマニュアル・セットを参照してください。

SQL Anywhere サーバおよびデータベースの起動と管理

Sybase IQ と SQL Anywhere では、データベースおよびサーバの起動と管理が異なります。

- Sybase IQ は、サーバ起動コマンド **start_iq** を使用し、SQL Anywhere のネットワーク・サーバ起動コマンドは使用しません。
- Sybase IQ では、パーソナル・サーバはサポートされていません。
- Sybase IQ は、SQL Anywhere のサーバ用コマンドライン・オプションの多くをサポートしていますが、すべてではありません。 Sybase IQ でサポートされて

いて、SQL Anywhere ではサポートされていないサーバ・オプションもあります。

- Sybase IQ には、サーバをシャットダウンするための **stop_iq** ユーティリティ (UNIX) が用意されています。
- Sybase IQ と SQL Anywhere では、**BACKUP** 文と **RESTORE** 文に使用できる句が異なります。
- Sybase IQ では、マルチプレックス操作用にのみ SQL Remote がサポートされています。

Sybase IQ は、SQL Anywhere のデータベース管理ユーティリティの多くをサポートしていますが、すべてではありません。

- 次に示す SQL Anywhere ユーティリティは、Sybase IQ ではサポート*されません*。 **backup**、**compression**、**console**、**initialization**、**license**、**log transfer**、**log translation**、**rebuild**、**spawn**、**transaction log** オプションの一部 (**-g**、**-il**、**-ir**、**-n**、**-x**、**-z**)、**uncompression**、**unload**、**upgrade**、**write file**。
- Sybase IQ は SQL Anywhere の **validation** ユーティリティをカタログ・ストアでのみサポートしています。 IQ ストアを検証するには、**sp_iqcheckdb** を使用します。

SQL Anywhere のデータベース・オプション

SQL Anywhere のデータベース・オプションの中には、Sybase IQ によってサポートされていないものがあります。DEFAULT_TIMESTAMP_INCREMENT もその 1 つです。

データベース・オプションには、カタログ・ストアのみに適用されるものがあります。これには、`FOR_XML_NULL_TREATMENT`、`ISOLATION_LEVEL`、`PREFETCH`、`PRECISION`、`SCALE` があります。

動作、デフォルト値、または指定可能な値が異なるオプションには、`DELAYED_COMMITS`、`TIME_FORMAT`、`TIMESTAMP_FORMAT` があります。

Sybase IQ にも、SQL Anywhere でサポートされていない多くのオプションがあります。

SQL Anywhere データ定義言語 (DDL) の相違点

前述の DDL の差異に加え、次の点に注意してください。

- **ALTER TABLE** 文の **DELETE/DROP** 句または **PRIMARY KEY** 句では、Sybase IQ は **RESTRICT** アクションを取ります (関連付けられた外部キーがある場合はエラーが報告されます)。SQL Anywhere は常に **CASCADE** アクションを取ります。

- 同様に、Sybase IQ では、関連付けられた外部キー制約がある場合、**DROP TABLE** 文もエラーを報告します。
- Sybase IQ は次の DDL 文をサポートしていません。**CREATE COMPRESSED DATABASE**, **CREATE TRIGGER**, **SETUSER**.
- Sybase IQ では、文レベルの参照整合性をサポートしていますが、トランザクション・レベルの整合性はサポートしていません。トランザクション・レベルの整合性は、SQL Anywhere が **CREATE TABLE** 文の **CHECK ON COMMIT** 句でサポートしています。
- Sybase IQ テーブルは、SQL Anywhere (またはカタログ) テーブルを参照する外部キーを持つことができません。また、SQL Anywhere テーブルは、Sybase IQ テーブルを参照する外部キーを持つことができません。
- Sybase IQ データベースでは、パブリケーションは、SQL Anywhere テーブル上でしか作成できません。
- **CREATE DATABASE** では、大文字と小文字の区別と照合のデフォルトに違いがあります。 Sybase IQ でのデフォルトは CASE RESPECT と ISO_BINENG 照合です。SQL Anywhere では、デフォルトは CASE IGNORE で、照合はオペレーティング・システムの言語と文字セットから判断されます。
- Sybase IQ は、SQL Anywhere でサポートされる **CREATE ENCRYPTED DATABASE** コマンドと **CREATE DECRYPTED DATABASE** コマンドをサポートしません。 詳細については、「Sybase IQ による高度なセキュリティ」を参照してください。

SQL Anywhere データ操作言語 (DML) の相違点

SQL Anywhere の DML オブジェクトおよび構文は、すべてが Sybase IQ でサポートされているわけではありません。

- Sybase IQ は、次の DML 文とプロシージャ文をサポートしていません。**EXPLAIN**, **GET DATA**, **INPUT**, **PREPARE TO COMMIT**, **PUT**, **READTEXT**, **ROLLBACK TRIGGER**, **SYSTEM**, **UNLOAD TABLE**, **VALIDATE TABLE**.

  注意： 一連の抽出オプションは、**UNLOAD TABLE** と類似した操作を実行します。詳細については、『システム管理ガイド：第 1 巻』を参照してください。
- Sybase IQ は **INSERT...LOCATION** 構文をサポートしますが、SQL Anywhere はサポートしません。
- **LOAD TABLE** オプションは、Sybase IQ と SQL Anywhere では異なります。
- Sybase IQ の **OPEN** 文では **BLOCK** 句と **ISOLATION LEVEL** 句をサポートしていません。
- Sybase IQ ではトリガをサポートしていません。
- トランザクション、独立性レベル、チェックポイント、自動生成される COMMIT の使用方法は、カーソル・サポートと同様に、Sybase IQ と SQL Anywhere では異なります。

- Sybase IQ のストアド・プロシージャから **SELECT** を実行する場合、CIS の機能補正によるパフォーマンスへの影響を考慮する必要があります。
- Sybase IQ は、**FROM** 句の *<database name>.<owner>.<table name>* など、Adaptive Server Enterprise の **SELECT** 文内にある完全修飾名のデータベース名の修飾子を無視します。 たとえば、Sybase IQ は、クエリ `SELECT * FROM XXX..TEST` を `SELECT * FROM TEST` と解釈します。

Adaptive Server Enterprise と Sybase IQ の相違点および共有機能

Sybase IQ と Adaptive Server Enterprise では、サポート対象のストアド・プロシージャとビューが異なっています。

詳細については、Sybase IQ を使用している場合は必ずそのマニュアル・セットを参照してください。 Adaptive Server Enterprise を使用している場合や、Sybase IQ のマニュアルで Adaptive Server Enterprise の具体的な機能を参照している場合は、Adaptive Server Enterprise のマニュアル・セットを参照してください。

Adaptive Server Enterprise のストアド・プロシージャ

Sybase IQ は、特定のストアド・プロシージャをサポートしていません。

Sybase IQ では、次の Adaptive Server Enterprise のストアド・プロシージャが使用できなくなりました。

- **sp_addserver**
- **sp_configure**
- **sp_estspace**
- **sp_help**
- **sp_helpuser**
- **sp_who**

Sybase IQ では次のカタログ・プロシージャが使用できなくなりました。

- **sp_column_privileges**
- **sp_databases**
- **sp_datatype_info**
- **sp_server_info**

Adaptive Server Enterprise のシステム・ビュー

Sybase IQ は、特定のビューをサポートしていません。

Sybase IQ では、次の Adaptive Server Enterprise のビューがサポートされなくなりました。

- sysalternates
- sysaudits
- sysauditoptions
- sysconstraints
- syscharsets
- sysconfigures
- syscurconfigs
- sysdatabases
- sysdepends
- sysdevices
- sysengines
- syskeys
- syslanguages
- syslocks
- syslogs
- sysloginroles
- sysmessages
- sysprocedures
- sysprocesses
- sysprotects
- sysreferences
- sysremotelogins
- sysroles
- syssegments
- sysservers
- syssrvroles
- systhresholds
- sysusages

カラム名の違い
Adaptive Server Enterprise の SYSTYPES ビューで使用されるカラム名は "allownulls" です。 Sybase IQ の SYSTYPES ビューで使用されるカラム名は "allownulls" です。

索引

数字

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

Y

あ

い

う

え

お

か

き

く

こ

さ

し

す

と

ね

は

ひ

ふ

ま

め

も

ゆ

ら

り

る

れ

ろ