SYBASE®
An SAP® Company

リファレンス：文とオプション

Sybase IQ 15.4

ドキュメント ID： DC01143-01-1540-01
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、Sybase IQ SQL 文およびデータベース・オプションに関するリファレンス資料を必要としている Sybase® IQ ユーザを対象としています。

言語要素、データ型、関数、システム・プロシージャ、システム・テーブルなど Sybase IQ の他の側面に関するリファレンス資料は、『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』を参照してください。 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。 このリファレンス・マニュアルは、使用可能な SQL 構文、パラメータ、オプションなどの情報を参照するためのものです。 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。

SQL 文

Sybase IQ で利用できる SQL 文について、Embedded SQL または Interactive SQL でのみ使用できるものも含めて説明します。

一般的な SQL 構文要素

ここでは、多くの SQL 構文で使われる言語要素をリストします。

ここで説明する要素の詳細については、「識別子」、「検索条件」、「式」、「文字列」(『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 言語の要素」) を参照してください。

- column-name ― カラム名を表す識別子。
- condition ― 評価によって、TRUE、FALSE、UNKNOWN を判断する式。
- connection-name ― アクティブな接続の名前を表す文字列。
- data-type ― 記憶データ型。
- expression ― 式。
- filename ― ファイル名を指定した文字列。
- host-variable ― 先頭にコロンがあるホスト変数として宣言される C 言語変数の 1 つ。
- indicator-variable ― 通常のホスト変数の直後に置かれた `short int` 型の二次的なホスト変数。インジケータ変数の前にはコロンを付けてください。インジケータ変数は、データベースとの NULL 値の引き渡しに使用されます。
- number ― 後に小数位が続いたり、前に負の記号が置かれたりする、任意のシーケンスに並べられた数字。また、数字の後に‘e’と指数を置くこともできます。次に例を示します。

  ```
  42
  -4.038
  .001
  3.4e10
  1e-10
  ```
- owner ― データベース・オブジェクトを所有するユーザ ID を表す識別子。
- role-name ― 外部キーの役割名を表す識別子。
- savepoint-name ― セーブポイント名を表す識別子。
- search-condition ― TRUE、FALSE、UNKNOWN を評価する条件。

- special-value – 『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 言語の要素」>「特別値」で説明されている特別値の 1 つ。
- statement-label ― ループまたは複合文のラベルを表す識別子。
- table-list ― テーブル名のリスト。相関名が含まれることもあります。詳細については、「FROM 句」を参照してください。
- table-name ― テーブル名を表す識別子。
- userid ― ユーザ名を表す識別子。ユーザ ID の大文字と小文字は区別されません。また、これはデータベースの CASE RESPECT プロパティの設定に影響されません。
- variable-name ― 変数名を表す識別子。

参照：
- FROM 句 (237 ページ)

構文の表記規則

ここでは、SQL 構文の説明に使用する表記規則を示します。

- キーワード ― SQL キーワードはすべて大文字で表記します。ただし、SQL キーワードの大文字と小文字は区別されないため、入力するときはどちらで入力してもかまいません。たとえば、SELECT は Select でも select でも同じです。
- プレースホルダ ― 適切な識別子または式で置き換えられる項目は、*イタリック*で表記します。
- 継続 ― 省略記号 (...) で始まる行は、直前の行から文が継続していることを示します。
- オプション部分 ― 文のオプション指定部分は、角カッコで囲みます。次に例を示します。
  ```
  RELEASE SAVEPOINT [ savepoint-name ]
  ```
  この例では、*savepoint-name* がオプション部分です。角カッコは入力しないでください。
- 繰り返し項目 ― 繰り返し項目のリストは、リストの要素の後ろに省略記号を付けて表します。複数の要素を指定できます。複数の要素を指定する場合、各要素間はカンマで区切る必要があります。次に例を示します。
  ```
  UNIQUE ( column-name [ , ... ] )
  ```
  この例では、カンマで区切って複数回 *column-name* を指定できることを示します。角カッコは入力しないでください。

- 選択肢 ― オプションのうち1つを選択する必要がある場合は、その選択肢を大カッコで囲みます。次に例を示します。
  ```
  [ QUOTES { ON | OFF } ]
  ```
  この場合、ON または OFF のどちらかを選択する必要があります。中カッコは入力しないでください。
- 1つまたは複数のオプション ― 1つまたは複数のオプションを選択する場合は、それぞれをカンマで区切ります。次に例を示します。
  ```
  { CONNECT, DBA, RESOURCE }
  ```

文の適応性インジケータ

一部の文は、どの場合に文を使用できるかを示すインジケータが角カッコで囲まれてタイトルの後ろに付きます。

これらのインジケータは以下のとおりです。

- [ESQL] – Embedded SQL に使用する文です。
- [Interactive SQL] – この文は、Interactive SQL (**dbisql**) でのみ使用できます。
- [SP] – この文は、ストアド・プロシージャまたはバッチで使用します。
- [T-SQL] – この文は、Adaptive Server® Enterprise との互換性を持たせるために実装されています。Transact-SQL フォーマットではないストアド・プロシージャでは、この文を使用できない場合があります。また、Transact-SQL の互換性が問題にならないかぎり、ISO/ANSI SQL 規格に近い代わりの文が推奨される場合もあります。

カッコが 2 組ある場合、その文はどちらの環境でも使用できます。たとえば、[ESQL] [SP] は Embedded SQL でもストアド・プロシージャでも使用できる文であることを意味します。

ALLOCATE DESCRIPTOR 文 [ESQL]

SQL 記述子領域 (SQLDA) に使用する領域を割り付けます。

構文

```
ALLOCATE DESCRIPTOR
                descriptor-name
… [ WITH MAX { integer | host-variable } ]
```

パラメータ

- **descriptor-name：** *– 文字列*

  詳細については、『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 言語の要素」を参照してください。

例

- **例 1** – 次のサンプル・プログラムには、**ALLOCATE DESCRIPTOR** 文の使用例が含まれています。

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>


EXEC SQL INCLUDE SQLCA;


#include <sqldef.h>


EXEC SQL BEGIN DECLARE SECTION;
int         x;
short       type;
int         numcols;
char        string[100];
a_sql_statement_number  stmt = 0;
EXEC SQL END DECLARE SECTION;

int main(int argc, char * argv[])
{
  struct sqlda *      sqlda1;

  if( !db_init( &sqlca ) ) {
      return 1;
  }
    db_string_connect(&sqlca, "UID=dba;PWD=sql;DBF=d:¥¥IQ-15_3¥
¥sample.db");

  EXEC SQL ALLOCATE DESCRIPTOR sqlda1 WITH MAX 25;

  EXEC SQL PREPARE :stmt FROM
        'select * from Employees';
  EXEC SQL DECLARE curs CURSOR FOR :stmt;
  EXEC SQL OPEN curs;

  EXEC SQL DESCRIBE :stmt into sqlda1;
  EXEC SQL GET DESCRIPTOR sqlda1 :numcols=COUNT;
      // how many columns?
  if( numcols > 25 ) {
      // reallocate if necessary
      EXEC SQL DEALLOCATE DESCRIPTOR sqlda1;
```

```
        EXEC SQL ALLOCATE DESCRIPTOR sqlda1
            WITH MAX :numcols;
    }
    type = DT_STRING;   // change the type to string
    EXEC SQL SET DESCRIPTOR sqlda1 VALUE 2 TYPE = :type;
    fill_sqlda( sqlda1 );  // allocate space for the variables

    EXEC SQL FETCH ABSOLUTE 1 curs USING DESCRIPTOR sqlda1;
    EXEC SQL GET DESCRIPTOR sqlda1 VALUE 2 :string = DATA;

    printf("name = %s", string );

    EXEC SQL DEALLOCATE DESCRIPTOR sqlda1;
    EXEC SQL CLOSE curs;
    EXEC SQL DROP STATEMENT :stmt;

    db_string_disconnect( &sqlca, "" );
    db_fini( &sqlca );

    return 0;
}
```

使用法

この文を使用する前に、C 言語のコードの中で次の宣言を行う必要があります。

```
struct sqlda * descriptor_name
```

WITH MAX 句を使用すると、記述子領域の変数の数を指定できます。デフォルトのサイズは 1 です。

さらに、フェッチを行ったり、記述子領域内のデータにアクセスしたりする文を実行する前には、**fill_sqlda** を呼び出して、実際のデータ項目に使用する領域を割り付ける必要があります。

『SQL Anywhere サーバー – プログラミング』の「SQL Anywhere Embedded SQL」>「SQLDA (SQL descriptor area)」を参照してください。

注意： このリファレンスは SQL Anywhere マニュアルにリンクされています。

標準

- SQL—ISO/ANSI SQL 文法のベンダ拡張。
- Sybase—Open Client/Open Server でサポートされています。

参照：

- DEALLOCATE DESCRIPTOR 文 [ESQL] (188 ページ)

ALTER DATABASE 文

以前のバージョンのソフトウェアで作成されたデータベースをアップグレードしたり、jConnect™ for JDBC™ のサポートを追加または削除したりします。この文は Interactive SQL とともに実行します。

構文

```
ALTER DATABASE UPGRADE
      [ PROCEDURE ON ]
      [ JCONNECT { ON | OFF } ]
```

例

- **例 1 –** 次の例は、jConnect サポートを無効にします。

  ```
  ALTER DATABASE UPGRADE JCONNECT OFF
  ```

使用法

ALTER DATABASE 文は、以前のバージョンのソフトウェアで作成されたデータベースをアップグレードします。これは、メジャー・リリースと同様にメンテナンス・リリースにも適用されます。

データベースをアップグレードすると、Sybase IQ によって次の変更が行われます。

- システム・テーブルの現在のバージョンへのアップグレード。
- 新規のデータベース・オプションの追加。
- 現在のバージョンの新機能の有効化。

また、そのデータベースが現在のバージョンのソフトウェアで作成されている場合は、**ALTER DATABASE UPGRADE** を使用して jConnect の機能を容易に追加できます。

注意：

- アップグレード前のバックアップの推奨については、『インストールおよび設定ガイド』を参照してください。
- **ALTER DATABASE UPGRADE** を実行する前に、サーバを起動するときにユーザ接続を制限する方法で起動したことを確認してください。手順とその他のアップグレードに関する注意事項については、プラットフォームに対応した『インストールおよび設定ガイド』の「データベースのアップグレード」を参照してください。
- **iqunload** ユーティリティを使用して、15.0 よりも前のバージョンで作成されたデータベースをアップグレードします。詳細については、プラットフォームに

対応した『インストールおよび設定ガイド』の「データベースのアップグレード」を参照してください。

ALTER DATABASE UPGRADE を使用したら、データベースを停止します。

- **PROCEDURE 句 –** データベースに含まれるすべての dbo 所有プロシージャと sys 所有プロシージャを削除して再作成します。
- **JCONNECT 句 –** Sybase jConnect JDBC ドライバを使用してシステム・カタログ情報にアクセスできるようにするには、**JCONNECT ON** を指定する必要があります。 これによって、jConnect システム・テーブルおよびプロシージャがインストールされます。jConnect システム・オブジェクトを除外するには、**JCONNECT OFF** を指定します。 その場合でも、システム・カタログ情報にアクセスしないかぎり、JDBC を使用できます。 デフォルトでは、jConnect のサポートが含まれています (**JCONNECT ON**)。

関連する動作：

- オートコミット。

標準

- 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。

パーミッション

DBA 権限が必要です。

参照：

- CREATE DATABASE 文 (81 ページ)

ALTER DBSPACE 文

既存の DB 領域に対し、読み書きのモードの変更、サイズの変更、または領域の拡張を行います。

構文

```
ALTER DBSPACE dbspace-name
{ ADD new-file-spec  [, new-file-spec ... ]
| DROP FILE logical-file-name [, FILE logical-file-name ... ]
  |  RENAME TO newname  |  RENAME 'new-file-pathname'
  |  READONLY  |  READWRITE
  |  ONLINE  |  OFFLINE
  |  STRIPING{ ON  |  OFF }
```

```
 | STRIPESIZEKB size-in-KB
ALTER FILE file-name
{ READONLY  |  [ FORCE ] READWRITE }
 | SIZE file-size [ KB | MB | GB | TB ]
 | ADD file-size  [ KB  |  MB  |  GB |  TB  | PAGES ] }
RENAME PATH  'new-file-pathname'
RENAME TO newname
```

パラメータ

- **new-file-spec：** –

  ```
  FILE logical-file-name 'file-path' iq-file-opts
  ```

- **iq-file-opts：** –

  ```
  [ [ SIZE ] file-size ]
  …[ KB | MB  | GB  | TB ] ]
     [ RESERVE reserve-size [ KB | MB | GB | TB ] ]
  ```

例

- **例 1 –** DspHist という名前の DB 領域のモードを、**READONLY** に変更します。

  ```
  ALTER DBSPACE DspHist READONLY
  ```

- **例 2 –** サイズが 500MB のファイル FileHist3 を追加して、DB 領域 DspHist に 500MB を追加します。

  ```
  ALTER DBSPACE DspHist
  ALTER FILE FileHist3 ADD 500MB
  ```

- **例 3 –** UNIX システムで、2 つの 500MB ファイルを DB 領域 DspHist に追加します。

  ```
  ALTER DBSPACE DspHist ADD
  FILE FileHist3 '/History1/data/file3' SIZE 500MB,
  FILE FileHist4 '/History1/data/file4' SIZE 500
  ```

- **例 4 –** DB 領域 IQ_SYSTEM_TEMP のサイズを、2GB だけ増加します。

  ```
  ALTER DBSPACE IQ_SYSTEM_TEMP ADD 2 GB
  ```

- **例 5 –** 2 つのファイルを DB 領域 DspHist から削除します。ファイルは両方とも空である必要があります。

  ```
  ALTER DBSPACE DspHist
  DROP FILE FileHist2, FILE FileHist4
  ```

- **例 6 –** DB 領域 IQ_SYSTEM_MAIN のサイズを、1000 ページだけ増加します (**ADD** のデフォルトはページです)。

  ```
  ALTER DBSPACE IQ_SYSTEM_MAIN ADD 1000
  ```

使用法

ALTER DBSPACE は、読み書きのモードの変更、オンライン／オフライン状態の変更、ファイル・サイズの変更、DB 領域名の変更、ファイルの論理名またはファイ

ル・パス名の変更、または DB 領域ストライピング・パラメータの設定を行います。既存の DB 領域の詳細については、**sp_iqdbspace** プロシージャ、**sp_iqdbspaceinfo** プロシージャ、**sp_iqfile** プロシージャ、**sp_iqdbspaceobjectinfo**、**sp_iqobjectinfo** を実行してください。DB 領域名と dbfile 名では、大文字と小文字は常に区別されません。**CASE RESPECT** を指定してデータベースが作成され、大文字と小文字が区別されるファイルがオペレーティング・システムでサポートされている場合、物理ファイル・パスの大文字と小文字は区別されます。そうでない場合、ファイル・パスの大文字と小文字は区別されません。

DB 領域と dbfile の名前は、引用符で囲まないか、二重引用符で囲みます。dbfile への物理ファイル・パスは、一重引用符で囲みます。

Windows でパスを指定する場合、円記号 (¥) の後に n または x がある場合は円記号を 2 つ重ねます。こうすることで、SQL の文字列の規則に従って、改行文字 (¥n) または 16 進数字 (¥x) として解釈されるのを回避できます。円記号は常にエスケープした方が安全です。

『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「システム・プロシージャ」>「システム・ストアド・プロシージャ」>「sp_iqdbspace プロシージャ」と『システム管理ガイド： 第 1 巻』の「データベース・オブジェクトの管理」>「データの格納」を参照してください。

ADD FILE 句—指定した DB 領域に 1 つまたは複数のファイルを追加します。 各ファイルに対して、dbfile 名と物理ファイル・パスは必須です。また、ユニークでなければなりません。 ファイルは IQ メイン、IQ 共有テンポラリ、または IQ テンポラリの各 DB 領域に追加できます。ファイルを読み込み専用の DB 領域に追加できますが、DB 領域は読み込み専用のままになります。ファイルをマルチプレックス共有テンポラリ DB 領域に追加できますが、読み込み専用モードの場合のみです (**ADD FILE** のデフォルト)。『Sybase IQ Multiplex の使用』を参照してください。

カタログ DB 領域には 1 つのファイルしか含むことができないため、**ADD FILE** をカタログ DB 領域で使用できない場合があります。

DROP FILE 句—指定したファイルを IQ DB 領域から削除します。ファイルは空である必要があります。最後のファイルを指定の DB 領域から削除することはできません。DB 領域に 1 つのファイルしか含まれていない場合は、DROP DBSPACE を代用します。 Rename TO 句—*dbspace-name* に新しい名前を付けます。新しい名前はデータベース内でユニークにします。`IQ_SYSTEM_MAIN`、`IQ_SYSTEM_MSG`、`IQ_SYSTEM_TEMP`、`IQ_SHARED_TEMP`、`SYSTEM` の名前は変更できません。

RENAME 句—1 つのファイルを含んでいる DB 領域のパス名の名前を変更します。**ALTER FILE RENAME PATH** 句と意味的に同じです。DB 領域に複数のファイルが含まれている場合はエラーが返されます。

READONLY 句—IQ_SYSTEM_MAIN、IQ_SYSTEM_TEMP、IQ_SYSTEM_MSG、IQ_SHARED_TEMP、SYSTEM 以外の DB 領域を読み込み専用に変更します。DB 領域に現在割り当てられたオブジェクトへの DML 変更を禁止します。IQ メイン・ストア内の DB 領域に対してのみ使用できます。

READWRITE 句—DB 領域を読み書き用に変更します。DB 領域はオンラインである必要があります。IQ メイン・ストア内の DB 領域に対してのみ使用できます。

ONLINE 句—オフライン DB 領域および関連するすべてのファイルをオンラインにします。IQ メイン・ストア内の DB 領域に対してのみ使用できます。

OFFLINE 句—オンラインの読み込み専用 DB 領域および関連するすべてのファイルをオフラインにします (DB 領域が読み書き用である、既にオフラインになっている、または IQ メイン・ストア内にない場合はエラーが返されます)。IQ メイン・ストア内の DB 領域に対してのみ使用できます。

STRIPING 句—DB 領域のディスク・ストライピングを指定どおりに変更します。ディスク・ストライピングがオンに設定されている場合、データは DB 領域内の各ファイルからラウンド・ロビン方式で割り付けられます。たとえば、最初に書き込みがあったデータベース・ページが最初のファイルへ、2 番目に書き込みがあったページが指定の DB 領域内の次のファイルへ、というようになります。読み込み専用の DB 領域はスキップされます。

STRIPESIZEKB 句—ディスク・ストライピング・アルゴリズムが指定した DB 領域の次のストライプに移動する前に、各ファイルに書き込むデータ量をキロバイト (KB) で指定します。

ALTER FILE READONLY—指定したファイルを読み込み専用に変更します。このファイルは、IQ メイン DB 領域に関連付けられている必要があります。IQ_SHARED_TEMP のファイルは READONLY ステータスに変更することはできません。

ALTER FILE READWRITE—指定した IQ メインまたはテンポラリ・ストア dbfile を読み書き用に変更します。このファイルは、IQ メインまたはテンポラリ DB 領域に関連付けられている必要があります。

ALTER FILE FORCE READWRITE—セカンダリ・ノードに既知のファイル・ステータスの問題がある場合でも、指定した共有テンポラリ・ストア dbfile のステータスを読み書き用に変更します。ファイルは、IQ メイン、共有テンポラリ、またはテンポラリの DB 領域に関連付けられている可能性がありますが、IQ_SYSTEM_MAIN とユーザ・メインの新しい dbfile は読み書き用に作成されるので、この句は、共有テンポラリ DB 領域にのみ影響します。

ALTER FILE SIZE 句—ファイルの新しいサイズを、キロバイト (KB)、メガバイト (MB)、ギガバイト (GB)、テラバイト (TB) の単位で指定します。デフォルトはメガバイトです。DB 領域のサイズは、フリー・リスト (アロケーション・マップ) に十分な余裕があり、DB 領域に十分な領域が確保されていなければ増加できません。DB 領域のサイズを減少できるのは、切り取られる部分が未使用である場合だけです。

ALTER FILE ADD 句—ファイルのサイズを、ページ、キロバイト (KB)、メガバイト (MB)、ギガバイト (GB)、テラバイト (TB) の単位で拡張します。デフォルトは MB です。ファイルのサイズは、フリー・リスト (アロケーション・マップ) に十分な余裕があり、DB 領域に十分な領域が確保されていなければ追加できません。

DB 領域のモードやサイズは、[Sybase Central Dbspaces] ウィンドウでも表示および変更が可能です。

ALTER FILE RENAME PATH 句—指定したファイルに関連付けられたファイルのパス名を変更します。この句はファイルを古いパスの代わりに新しいファイル・パスに関連付けるだけであり、オペレーティング・システムのファイル名を実際に変更するわけではありません。ファイル名を変更するには、オペレーティング・システム経由で実行する必要があります。ファイルのパス名を変更するには、DB 領域がオフラインである必要があります。新しいパスは、DB 領域をオンラインで変更するか、データベースを再起動する場合に使用されます。

新しいパスにアクセスできない場合は、データベースを起動できないため、`IQ_SYSTEM_MAIN` 内でファイルのパス名を変更する必要はありません。`IQ_SYSTEM_MAIN` 内のファイルのパス名を変更する必要がある場合は、ファイルを読み込み専用に設定し、ファイルを空にして削除してから、ファイルに新しいパス名を付けて追加し直します。

ALTER FILE RENAME TO 句—指定したファイルの論理名に新しい名前を付けます。新しい名前はデータベース内でユニークにします。

関連する動作：

- オートコミット。
- 自動チェックポイント。
- モードを **READONLY** に変更すると、DB 領域上のデータベースの内部構造が、読み書き用であるいずれかの DB 領域へ直ちに移動されます。

標準

- 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。

パーミッション

SPACE ADMIN 権限または DBA 権限を持っている必要があります。

参照：

- CREATE DATABASE 文 (81 ページ)
- CREATE DBSPACE 文 (93 ページ)
- DROP 文 (209 ページ)

ALTER DOMAIN 文

ユーザ定義のドメインまたはデータ型の名前を変更します。Java データ型の名前は変更しません。

構文

```
ALTER { DOMAIN | DATATYPE } user-type
RENAME new-name
```

パラメータ

- **new-name：** – 新しいドメイン名を表す識別子。
- **user-type：** – 名前が変更されるユーザ定義のドメインのデータ型。

例

- **例 1 –** 次に、Address ドメインの名前を MailingAddress に変更する場合の例を示します。

  ```
  ALTER DOMAIN Address RENAME MailingAddress
  ```

使用法

ALTER DOMAIN 文は、SYSUSERTYPE システム・テーブルのユーザ定義のドメインまたはデータ型の名前を更新します。『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「システム・テーブルとシステム・ビュー」>「システム・ビュー」>「SYSUSERTYPE システム・ビュー」を参照してください。

そのユーザ定義のドメインまたはデータ型を参照するすべてのプロシージャ、ビュー、またはイベントを再作成する必要があります。そうしないと、それらは以前の名前を参照します。

『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。

パーミッション

DBA 権限を持っているか、そのドメインを作成したデータベース・ユーザである必要があります。

参照：

- CREATE DOMAIN 文 (97 ページ)

ALTER EVENT 文

イベント、または事前定義のアクションを自動化するイベント関連ハンドラの定義を変更します。また、スケジュールされたアクションの定義も変更します。

構文

```
ALTER EVENT event-name
[ DELETE TYPE | TYPE event-type ]
{    WHERE { trigger-condition | NULL }
   | { ADD | [ MODIFY ] | DELETE } SCHEDULE schedule-spec
}
[ ENABLE | DISABLE ]
[ [ MODIFY ] HANDLER compound-statement | DELETE HANDLER }
```

パラメータ

- **event-type：** – **BackupEnd** | **“Connect”** | **ConnectFailed** | **DatabaseStart** | **DBDiskSpace** | **“Disconnect”** | **GlobalAutoincrement** | **GrowDB** | **GrowLog** | **GrowTemp** | **LogDiskSpace** | **“RAISERROR”** | **ServerIdle** | **TempDiskSpace**
- **trigger-condition：** – [**event_condition(** *condition-name*) { **=** | **<** | **>** | **!=** | **<=** | **>=** } *value*]
- **schedule-spec：** – [*schedule-name*] { **STARTTIME***start-time* | **BETWEEN***start-time***AND***end-time* } [**EVERY***period* { **HOURS** | **MINUTES** | **SECONDS** }] [**ON** { (*day-of-week*, …) | (*day-of-month*, …) }] [**STARTDATE***start-date*]
- **event-name | schedule-name：** – *識別子*
- **day-of-week：** – *文字列*
- **value | period | day-of-month：** – *整数*
- **start-time | end-time：** – *時刻*
- **start-date：** – *日付*

使用法

ALTER EVENT では、**CREATE EVENT** で作成されたイベント定義を変更できます。以下の用法が考えられます。

- 開発時にイベント・ハンドラを変更する。
- 開発段階でトリガ条件やスケジュールを指定せずにイベント・ハンドラを定義、テストし、イベント・ハンドラの完成後に **ALTER EVENT** を使って実行条件を追加する。
- イベントを無効にすることにより、イベント・ハンドラを一時的に無効にする。

ALTER EVENT を使用してイベントを変更するときは、イベント名と任意でスケジュール名を指定します。

イベント名は、システム・テーブル SYSEVENT をクエリして、一覧表示します。次に例を示します。

```
SELECT event_id, event_name FROM SYS.SYSEVENT
```

スケジュール名は、システム・テーブル SYSSCHEDULE をクエリして、一覧表示します。次に例を示します。

```
SELECT event_id, sched_name FROM SYS.SYSSCHEDULE
```

イベントには固有のイベント ID が付いています。イベントと関連するスケジュールの対応付けには、SYSEVENT と SYSSCHEDULE の event_id カラムを使用します。

DELETE TYPE 句—イベントとイベント・タイプの関連付けを削除します。

ADD | MODIFY | DELETE SCHEDULE 句—スケジュールの定義を変更します。1 つの **ALTER EVENT** 文で 1 つのスケジュールしか変更できません。

WHERE 句—**WHERE NULL** オプションは条件を削除します。

パラメータの詳細については、「CREATE EVENT 文」を参照してください。

『システム管理ガイド：第 2 巻』の「スケジューリングとイベント処理によるタスクの自動化」も参照してください。

関連する動作：

- オートコミット。

パーミッション

DBA 権限が必要です。

参照：

- BEGIN … END 文 (60 ページ)
- CREATE EVENT 文 (99 ページ)

ALTER FUNCTION 文

既存の関数を変更します。変更した関数全体を **ALTER FUNCTION** 文にインクルードします。

構文

構文 1

```
ALTER FUNCTION [ owner.]function-name function-definition

function-definition : CREATE FUNCTION syntax
```

構文 2

```
ALTER FUNCTION [ owner.]function-name

 SET HIDDEN
```

構文 3

```
ALTER FUNCTION [ owner.]function-name

 RECOMPILE
```

使用法

構文 1—最初の単語を除き、**CREATE FUNCTION** 文の構文とまったく同じです。**CREATE FUNCTION** 文のいずれかのバージョンを変更できます。

関数に対する既存のパーミッションはそのまま維持されます。このため、パーミッションの再割り当ては必要ありません。**DROP FUNCTION** と **CREATE FUNCTION** を実行した場合は、実行パーミッションを再割り当てする必要があります。

構文 2—**SET HIDDEN** を使用すると、関連付けられた関数が暗号化さ れ、判読不能になります。関数は他のデータベースにアンロードおよび再ロードできます。

警告！ SET HIDDEN 設定は、元に戻すことはできません。元のソースを再度必要とする場合は、データベースの外部に保持しておく必要があります。

SET HIDDEN を使用する場合、ストアド・プロシージャ・デバッガを使用したデバッグでは、関数の定義は表示されず、プロシージャ・プロファイリングでも表示されません。

『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、関数、ビューの内容を隠す」を参照してください。

構文 3—**RECOMPILE** を使用して、ユーザ定義の関数を再コンパイルします。関数を再コンパイルする場合は、カタログ内に格納された関数が再解析され、構文が検証されます。関数用に保持されているソースは、再コンパイルによって変更されません。関数を再コンパイルする場合、**SET HIDDEN** 句によって暗号化された定義はそのままの状態で 判読不能になります。

関連する動作：

- オートコミット。

標準

- SQL—ISO/ANSI SQL 文法のベンダ拡張。

パーミッション

関数の所有者であるか、DBA 権限を持っている必要があります。

参照：

- ALTER PROCEDURE 文 (30 ページ)
- CREATE FUNCTION 文 (110 ページ)
- DROP 文 (209 ページ)

ALTER INDEX 文

ベース・テーブルまたはグローバル・テンポラリ・テーブルのインデックスの名前、およびインデックスと外部キーがユーザによって明示的に作成された場合の外部キーの役割名を変更します。

構文

```
ALTER { INDEX index-name
| [ INDEX ] FOREIGN KEY role-name
| [ INDEX ] PRIMARY KEY
| ON [owner.]table-name   { rename-clause | move-clause }
```

パラメータ

- **rename-clause –** : **RENAME TO** | **AS** *new-name*
- **move-clause： – MOVE TO** *dbspace-name*

例

- **例 1 –** プライマリ・キー (c5 の HG) を DB 領域 Dsp4 から Dsp8 に移動します。

```
CREATE TABLE foo (
    c1 INT IN Dsp1,
    c2 VARCHAR(20),
    c3 CLOB IN Dsp2,
    c4 DATE,
    c5 BIGINT,
    PRIMARY KEY (c5) IN Dsp4) IN Dsp3);

CREATE DATE INDEX c4_date ON foo(c4) IN Dsp5;

ALTER INDEX PRIMARY KEY ON foo MOVE TO Dsp8;
```

- **例 2 – DATE** インデックスを `Dsp5` から `Dsp9` に移動します。

```
ALTER INDEX c4_date ON foo MOVE TO Dsp9
```

- **例 3 –** テーブル `jal.mytable` のインデックス `COL1_HG_OLD` の名前を `COL1_HG_NEW` に変更します。

```
ALTER INDEX COL1_HG_OLD ON jal.mytable
RENAME AS COL1_HG_NEW
```

- **例 4 –** テーブル `dba.Employees` の外部キーの役割名 `ky_dept_id` を `emp_dept_id` に変更します。

```
ALTER INDEX FOREIGN KEY ky_dept_id
ON dba.Employees
RENAME TO emp_dept_id
```

使用法

ALTER INDEX 文は、ユーザによって明示的に作成されたインデックスと外部キーのインデックス名と外部キーの役割名を変更します。ベース・テーブルまたはグローバル・テンポラリ・テーブルのインデックスの名前だけが変更できます。キー制約を適用するために作成したインデックスの名前は変更できないという点に注意してください。

ON 句—**ON** 句には、名前を変更するインデックスや外部キーを含むテーブルの名前を指定します。

RENAME [AS | TO] 句—**RENAME** 句には、インデックスや外部キーの役割の変更後の名前を指定します。

MOVE 句—MOVE 句は、指定したインデックス、一意性制約、外部キー、またはプライマリ・キーを指定した DB 領域に移動します。一意性制約または外部キーでは、一意なインデックス名を指定する必要があります。

新しい DB 領域に対する CREATE 権限を持っており、かつテーブル所有者であるか、DBA または SPACE ADMIN 権限を持っている必要があります。

注意： ローカル・テンポラリ・テーブルのインデックスを変更しようとすると、「インデックスが見つかりません。」というエラーが返されます。デフォルト・インデックス (FP) などユーザ作成でないインデックスを変更しようとすると、「インデックスを変更できません。所有者のタイプが USER のベース・テーブルまたはグローバル・テンポラリ・テーブルのインデックスのみを変更できます。」というエラーが返されます。

関連する動作：

- オートコミット。Interactive SQL の、[Results] ペインの [Results] タブがクリアされます。現在の接続に対するすべてのカーソルがクローズします。

標準

- SQL—ISO/ANSI SQL 準拠。
- Sybase—Adaptive Server Enterprise ではサポートされていません。

パーミッション

テーブルを所有しているか、テーブルの REFERENCES パーミッションを持っているか、DBA または RESOURCE 権限を持っている必要があります。**ALTER INDEX MOVE TO** 文については、新しい DB 領域に対する CREATE 権限を持っており、かつテーブル所有者であるか、DBA または SPACE ADMIN 権限を持っている必要があります。

参照：

- ALTER TABLE 文 (37 ページ)
- CREATE INDEX 文 (118 ページ)
- CREATE TABLE 文 (166 ページ)

ALTER LOGICAL SERVER 文

データベース内の既存のユーザ定義による論理サーバの設定を変更します。

構文

```
ALTER LOGICAL SERVER  logical-server-name
    { alter_clause }
```

パラメータ

- **alter_clause： –**

```
{ADD MEMBERSHIP  '(' { ls-member, ... } ')'
| DROP MEMBERSHIP '(' { ls-member, ... } ')'
}
```

- **ls-member： –**

```
FOR LOGICAL COORDINATOR
| mpx-server-name
```

例

- **例 1 –** 次の例は、マルチプレックス・ノードの n1 と n2 を論理サーバ ls1 に追加して、ユーザ定義の論理サーバを変更します。

```
ALTER LOGICAL SERVER ls1 ADD MEMBERSHIP (n1, n2)
```

- **例 2 –** 次の例は、コーディネータの論理メンバシップを追加し、現在のコーディネータ・ノード n1 の名前付きメンバシップを ls1 から削除します。

```
ALTER LOGICAL SERVER ls1 ADD MEMBERSHIP (FOR LOGICAL COORDINATOR)
ALTER LOGICAL SERVER ls1 DROP MEMBERSHIP (n1)
```

使用法

マルチプレックスのみに該当します。

logical-server-name は、既存のユーザ定義の論理サーバ名を参照します。言い換えると、組み込みまたは予約済みの論理サーバ名ではありません。
SYS.ISYSIQLSMEMBER システム・テーブルには、論理サーバ・メンバシップの定義が格納されています。

論理サーバに追加された、または論理サーバから削除されたメンバ・ノードが論理サーバ接続の受け付けを開始または停止するのは、**ALTER LOGICAL SERVER** に対応する TLV ログがそのノードで再生された後のみです。論理サーバの既存の接続は、ノードが論理サーバから削除されてもそのノードで稼動し続けますが、これらの接続の分散処理は停止します。

次の場合には、エラーが返ります。

- **ADD MEMBERSHIP** 句で指定された *ls-member* が、すでに論理サーバのメンバである。
- **DROP MEMBERSHIP** 句で指定された *ls-member* が論理サーバの既存のメンバではない。
- 論理サーバのメンバシップ変更により、メンバシップの重複チェックが失敗する。

パーミッション

DBA 権限または MPX ADMIN 権限を持っている必要があります。

ALTER LOGIN POLICY 文

この文は、2 つの機能を実行します。データベース内の既存のログイン・ポリシーのオプション値を変更する機能と、論理サーバ・アクセスを設定する機能です。

構文

```
ALTER LS POLICY policy-name option-value-list
```

パラメータ

- **alter-clause： –**

  ```
  { { ADD | DROP | SET } LOGICAL SERVER
                          ls-assignment-list
        | policy-option-name = policy-option-value [ LOGICAL SERVER
                          ls-override-list ]
  }
  ```

- **ls-assignment-list： –**

  ```
  { { ls-name, ...} | SERVER | NONE | DEFAULT }
  ```

- **ls-override-list： –**

  ```
  { ls-name, ...}
  ```

- **ls-name： –**

  ```
  { OPEN | user-defined-ls-name }
  ```

- **policy-option-value： –**

  ```
  { UNLIMITED | DEFAULT | value }
  ```

例

- **例 1 –** 「論理サーバへのアクセス許可設定」と「ログイン・ポリシー・オプションの設定」を参照してください。

使用法

マルチプレックスのみに該当します。

「論理サーバへのアクセス許可設定」と「ログイン・ポリシー・オプションの設定」を参照してください。

パーミッション

DBA 権限または USER ADMIN 権限を持っている必要があります。

論理サーバへのアクセス許可設定

ALTER LOGIN POLICY を使用して、論理サーバへのアクセス許可を設定できます。

例 1

ルート・ログイン・ポリシーが論理サーバの ls4 と ls5 へのアクセスを許可し、ログイン・ポリシー lp1 が論理サーバの割り当てなしに存在するとします。次の文は、ログイン・ポリシー lp1 を論理サーバの ls4 と ls5 に実質的に割り当てます。

論理サーバ ls1 をログイン・ポリシー lp1 に割り当てます。

```
ALTER LOGIN POLICY lp1 ADD LOGICAL SERVER ls1
```

例 2

次の文は、論理サーバの ls2 と ls3 にログイン・ポリシー lp1 からアクセスすることを許可します。

```
ALTER LOGIN POLICY lp1 ADD LOGICAL SERVER ls2, ls3
```

例 3

ログイン・ポリシー lp1 を変更して、ls3 と ls4 にのみにアクセスを許可します。

```
ALTER LOGIN POLICY lp1 ADD LOGICAL SERVER ls4
```

```
ALTER LOGIN POLICY lp1 DROP LOGICAL SERVER ls1, ls2
```

または

```
ALTER LOGIN POLICY lp1 SET LOGICAL SERVER ls3, ls4
```

例 4

ログイン・ポリシー lp1 を変更して、すべての論理サーバへのアクセスを拒否します。

```
ALTER LOGIN POLICY lp1 SET LOGICAL SERVER NONE
```

例 5

ログイン・ポリシー lp1 の現在の論理サーバ割り当てを削除し、 ルート・ログイン・ポリシーの論理サーバ割り当てを継承させます。

```
ALTER LOGIN POLICY lp1 SET LOGICAL SERVER DEFAULT
```

使用法
マルチプレックスのみに該当します。

ADD 句、**DROP** 句、または **SET** 句を使用すると、ログイン・ポリシーの論理サーバ割り当てを設定できます。

- **ADD** – 新しい論理サーバ割り当てをログイン・ポリシーに追加します。
- **DROP** – ログイン・ポリシーから既存の論理サーバ割り当てを削除します。
- **SET** – ログイン・ポリシーのすべての論理サーバ割り当てを新しい一連の論理サーバに置き換えます。

ADD 句、**DROP** 句、または **SET** 句のいずれか 1 つのみを使用します。**SERVER**、**NONE**、**DEFAULT** は、**SET** 句でのみ使用します。**ls-assignment list** または **ls-override list** ごとに特定の論理サーバ名を 1 回のみ指定します。

次の場合には、エラーが返ります。

- **ADD** 句で指定された論理サーバが、すでにログイン・ポリシーに割り当てられている。
- **DROP** 句で指定された論理サーバが、ログイン・ポリシーに割り当てられていない。
- 論理サーバ割り当ての変更により、割り当てられている論理サーバ間でメンバシップの重複が発生する。

`SYS.ISYSIQLOGINPOLICYLSINFO` には、論理サーバ割り当ての情報が格納されています。ログイン・ポリシーのログイン・ポリシー・オプションの各論理サーバ上書きについては、対応するローが `ISYSIQLOGINPOLICYLSINFO` に存在します。

ログイン・ポリシー・オプションの設定

ALTER LOGIN POLICY を使用して、ログイン・ポリシー・オプションを設定できます。

例

次の例は、2 つの論理サーバのログイン・ポリシー設定を上書きします。論理サーバ `ls1` で分散クエリ処理を有効にし、論理サーバ `ls2` で最大接続数を増やします。

```
ALTER LOGIN POLICY lp1 dqp_enabled=ON LOGICAL SERVER ls1;
ALTER LOGIN POLICY lp2 max_connections=20 LOGICAL SERVER ls2;
```

使用法
マルチプレックスのみに該当します。

論理サーバ・レベルの上書きは、特定のログイン・ポリシー・オプションが、異なる論理サーバに対して異なる設定を持つことを意味します。
`SYS.ISYSIQLSLOGINPOLICYOPTION` には、論理サーバ上書きのログイン・ポリシー・オプション値が格納されています。ログイン・ポリシーのログイン・ポリ

シー・オプションの各論理サーバ上書きについては、対応するローが ISYSIQLSLOGINPOLICYOPTION に存在します。

注意： 論理サーバ上書き設定は、ログイン・ポリシー・オプションの **max_connections** と **dqp_enabled** にのみ指定できます (**dqp_enabled** オプションは、マルチプレックス・サーバにのみ影響します。「マルチプレックス・リファレンス」>「データベース・オプション」>「dqp_enabled オプション」を参照してください)。ルート・ログイン・ポリシーに論理サーバの上書きを指定することはできません。

表 1 : ログイン・ポリシー・オプション

オプション	説明	値	ROOT ポリシーの初期値	適用対象
dqp_enabled	ON の場合、ログイン・ポリシーに割り当てられているユーザに対して分散クエリ処理が有効になります。	ON、OFF	ON	マルチプレックス・サーバのみ。DBA 権限を持つユーザを含むすべてのユーザ。
locked	このオプションの値が ON の場合、ユーザが新しい接続を確立するのを禁止します。	ON、OFF	OFF	DBA 権限を持っていないユーザのみ。
max_connections	ユーザに許可された同時接続の最大数。	0 – 2147483647	Unlimited	DBA 権限を持っていないユーザのみ。
max_days_since_login	同一ユーザによる前回のログイン時から次回のログイン時までの最大許容日数。	0 – 2147483647	Unlimited	DBA 権限を持っていないユーザのみ。
max_failed_login_attempts	前回の正常なログイン時以降に行った、ユーザ・アカウントがロックアウトされる原因となるログイン試行失敗回数の最大値。	0 – 2147483647	Unlimited	DBA 権限を持っていないユーザのみ。

オプション	説明	値	ROOT ポリシーの初期値	適用対象
max_non_dba_connections	DBA 権限を持っていないユーザが実行できる同時接続の最大数。このオプションは、ルート・ログイン・ポリシーでのみサポートされています。	0 – 2147483647	Unlimited	DBA 権限を持っていないユーザのみ。ルート・ログイン・ポリシーのみをサポート。
password_expiry_on_next_login	このオプションの値が ON に設定されている場合は、ユーザのパスワードは次回ログイン時に有効期限が切れます。	ON、OFF	OFF	DBA 権限を持つすべてのユーザ。
password_grace_time	パスワードの有効期限が切れるまでの日数 (この間は、ログインすることはできますが、デフォルトの post_login プロシージャによって警告が発行されます)。	0 – 2147483647	0	DBA 権限を持つすべてのユーザ。
password_life_time	パスワードの変更が必要になるまでの最大日数。	0 – 2147483647	Unlimited	DBA 権限を持つすべてのユーザ。

注意： Sybase IQ では、**MULTIPLEX SERVER** 上書き句がサポートされなくなりました。ログイン・ポリシー・オプションに対して、**MULTIPLEX SERVER** 上書き句を指定して **ALTER LOGIN POLICY** 文を使用すると、エラーが返ります。

ALTER LS POLICY 文

データベース内の既存のルート論理サーバ・ポリシーの一部またはすべてのオプション値を変更します。

構文

ALTER LS POLICY *policy-name option-value-list*

パラメータ

- **option-value-list： –**
  {*option-name=value*}

例

- **例 1 –** 次の例では、論理サーバ・ポリシーを変更します。

```
ALTER LS POLICY root
ALLOW_COORDINATOR_AS_MEMBER=ON;
```

注意： ALLOW_COORDINATOR_AS_MEMBER は唯一の論理サーバ・ポリシー・オプションで、ルートは唯一の論理サーバ・ポリシーです。論理サーバ・ポリシーは作成できません。

使用法

マルチプレックスのみに該当します。

パーミッション

DBA 権限または MPXADMIN 権限を持っている必要があります。

ALTER MULTIPLEX RENAME 文

マルチプレックスの名前を変更し、マルチプレックス名を SYS.ISYSIQINFO システム・テーブルに格納します。

構文

ALTER MULTIPLEX RENAME *multiplex-name*

使用法

マルチプレックスのみに該当します。

作成されたマルチプレックスには、コーディネータの名前が付けられます。マルチプレックス名は、マルチプレックス・フォルダ内でマルチプレックスを識別するために Sybase Central のみで使用されます。この文は自動的にコミットされます。

パーミッション

DBA 権限または MULTIPLEX ADMIN 権限を持っている必要があります。

ALTER MULTIPLEX SERVER 文

特定のサーバの名前、カタログ・ファイル・パス、役割、またはステータスを変更します。

構文

構文 1：

```
ALTER MULTIPLEX SERVER server-name server-option
```

構文 2：

```
ALTER MULTIPLEX SERVER PRIVATE NULL
```

パラメータ

- **server-option：** –

```
{ RENAME new-server-name
| DATABASE 'dbfile'
| ROLE { WRITER | READER | COORDINATOR }
| STATUS { INCLUDED | EXCLUDED }
| ASSIGN AS FAILOVER SERVER
| host-port-list }
```

- **host-port-list：** – { **HOST** ' *hostname* ' **PORT** *port number* ...} { **PRIVATE HOST** ' *hostname* ' **PORT***port number* ...}

注意： サーバを除外する前に、そのサーバを停止することをおすすめします。除外されるサーバは、停止されていない場合、除外された後に自動的に停止します。また、除外されたサーバをマルチプレックスに再び参加させるには、**ALTER MULTIPLEX SERVER***server-name***STATUS INCLUDED** を実行し、同期を行う必要があります。

例

- **例** – 次の例は、セカンダリ・サーバ `mpx_writer1` を除外します。

```
ALTER MULTIPLEX SERVER mpx_writer1 STATUS EXCLUDED
```

使用法

マルチプレックスのみに該当します。

マルチプレックス・サーバを次のように変更します。

RENAME – 特定のサーバの名前を変更します。サーバは自動的に停止します。次の再起動時に新しい名前が必要となります。

DATABASE – 特定のサーバのカタログ・ファイル・パスを変更します。サーバは自動的に停止し、次回の起動から新しいカタログ・パスを使用します。カタログ・ファイル自体の移動は、ユーザの責任で行う必要があります (Sybase Central に表示されない場合があります)。

ROLE – 特定のサーバの役割を変更します。ユーザは、コーディネータの役割を変更したり、コーディネータでないサーバをコーディネータに変更したりすることはできません。ライタ・ノードの役割がリーダに変更された場合、サーバは停止します。

STATUS – 特定のサーバのステータスを変更します。フェールオーバ・ノードは、そのノードが除外される最後のノードである場合を除き、除外することはできません。除外後に、サーバは自動的に停止します。ノードの追加後は、そのノードを同期し、再起動する必要があります。

ASSIGN – 特定のサーバを新しいフェールオーバ・サーバとして指定します。除外済みのステータスにあるノードは使用できません。ASSIGN AS FAILOVER 句は、他の **ALTER MULTIPLEX SERVER** 句とともに使用できないスタンドアロン句です。

コーディネータが実行中である必要がありますが、**ALTER MULTIPLEX SERVER** コマンドはマルチプレックス内のいずれのサーバからでも実行できます(すべての DDL 文をコーディネータ上で実行することをおすすめします)。役割をリーダからライタに変更する以外のすべての場合に、名前付きサーバは自動的に停止します。

注意： サーバを除外する前に、そのサーバを停止することをおすすめします。除外されるサーバは、停止されていない場合、除外された後に自動的に停止します。また、除外されたサーバをマルチプレックスに再び参加させるには、`ALTER MULTIPLEX SERVER` *`server-name`* `STATUS INCLUDED` を実行し、同期を行う必要があります。

パーミッション

DBA 権限または MULTIPLEX ADMIN 権限を持っている必要があります。

ALTER PROCEDURE 文

既存のプロシージャを修正したものに置き換えます。このとき、修正したプロシージャ全体を **ALTER PROCEDURE** 文にインクルードして、プロシージャに対するユーザ・パーミッションを再割り当てします。

構文

構文 1

```
ALTER PROCEDURE [ owner.]procedure-name procedure-definition
```

構文 2

```
ALTER PROCEDURE [ owner.]procedure-name
REPLICATE { ON | OFF }
```

構文 3

```
ALTER PROCEDURE [ owner.]procedure-name
SET HIDDEN
```

構文 4

```
ALTER PROCEDURE [ owner.]procedure-name
RECOMPILE
```

構文 5

```
ALTER PROCEDURE
[ owner.]procedure-name ( [ parameter, …] )
[ RESULT (result-column, ...)]
EXTERNAL NAME 'external-call' [ LANGUAGE environment-name ] }
```

パラメータ

- **procedure-definition –** 名前の後に **CREATE PROCEDURE** 構文
- **environment-name – JAVA [DISALLOW | ALLOW SERVER SIDE REQUESTS]**
- **external-call –** [*column-name*:]*function-name@library*; ...

使用法

ALTER PROCEDURE 文には、新しいプロシージャ全体を含めます。 **PROC** を **PROCEDURE** の同意語として使用できます。

ALTER PROCEDURE 文は、**CREATE PROCEDURE** 文の構文とまったく同じです。

- **構文 1 – ALTER PROCEDURE** 文の構文は、最初の 1 語を除き、**CREATE PROCEDURE** 文の構文とまったく同じです。 Watcom SQL ダイアレクトと

Transact-SQL ダイアレクトのプロシージャは、**ALTER PROCEDURE** を使用して変更できます。

ALTER PROCEDURE, の場合、プロシージャに対する既存のパーミッションは変更されません。 **CREATE PROCEDURE** に続けて **DROP PROCEDURE** を実行すると、実行パーミッションが再割り当てされます。

- **構文 2 –** Sybase Replication Server を使用してプロシージャを他のサイトに移動する必要がある場合は、プロシージャに REPLICATE ON を設定します。
- **構文 3 – SET HIDDEN** を使用して、関連するプロシージャの定義を難読化し、解読できないようにします。 このプロシージャはアンロードして、他のデータベースに再ロードできます。

  注意： この設定は、元に戻せません。 元のプロシージャ定義をデータベースの外部に保持することをおすすめします。

  SET HIDDEN を使用すると、デバッガを使用したデバッグでは、プロシージャの定義は表示されず、プロシージャ・プロファイリングでも表示されません。

構文 2 と構文 1 を組み合わせることはできません。

- **構文 4 – RECOMPILE** 構文を使用して、ストアド・プロシージャを再コンパイルします。 ストアド・プロシージャを再コンパイルすると、カタログに格納された定義が再解析され、構文が検証されます。 結果セットを生成するものの **RESULT** 句を含まないプロシージャの場合は、データベース・サーバがプロシージャの結果セットの特性を判別しようとし、カタログに情報を格納します。 これは、プロシージャの作成後に、プロシージャの参照先テーブルがカラムの追加、削除、または名前変更するように変更されている場合に役立つことがあります。

  プロシージャの定義は、再コンパイルしても変わりません。**SET HIDDEN** 句で隠された定義を持つプロシージャは再コンパイルできますが、これらの定義は隠されたままになります。
- **構文 5 –** この構文は Java UDF に使用します。

  iq-environment-name: JAVA [DISALLOW | ALLOW SERVER SIDE REQUESTS]:

  DISALLOW がデフォルトです。

  ALLOW は、サーバ側接続が許可されることを示します。

  注意： 必要な場合を除き、**ALLOW** は指定しないでください。**ALLOW** を設定すると、特定の種類の Sybase IQ テーブル・ジョインの速度を低下させます。

  UDF を使用するときは、同じクエリ内で **ALLOW SERVER SIDE REQUESTS** と **DISALLOW SERVER SIDE REQUESTS** の両方を指定しないでください。

ALTER PROCEDURE 文をテーブル UDF に使用する場合、**CREATE PROCEDURE** 文 (外部プロシージャ) と同じ制限事項が適用されます。

標準

- 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。

パーミッション

プロシージャの所有者であるか、DBA 権限を持っている必要があります。オートコミット。

参照：
- CREATE PROCEDURE 文 (137 ページ)

ALTER SERVER 文

リモート・サーバの属性を変更します。

構文

```
ALTER SERVER server-name
[ CLASS 'server-class' ]
[ USING 'connection-info' ]
[ CAPABILITY 'cap-name' { ON | OFF } ]
[ CONNECTION CLOSE [ CURRENT | ALL | connection-id ] ]
```

パラメータ

- **server-class：** – { *ASAJDBC* | *ASEJDBC* | *ASAODBC* | *ASEODBC* | *DB2ODBC* | *MSSODBC* | *ORAODBC* | *ODBC* }
- **connection-info：** – { *machine-name:port-number* [/*dbname*] | *data-source-name* }
- **cap-name：** – サーバ機能の名前

例

- **例 1** – 名前が `ase_prod` である Adaptive Server Enterprise サーバのサーバ・クラスを変更して、Sybase IQ への接続が ODBC ベースになるようにします。 データ・ソース名は `ase_prod` です。

```
ALTER SERVER ase_prod
CLASS 'ASEODBC'
USING 'ase_prod'
```

- **例 2 –** サーバ infodc の機能を変更します。

```
ALTER SERVER infodc
CAPABILITY 'insert select' OFF
```

- **例 3 –** リモート・サーバ rem_test への接続をすべて閉じます。

```
ALTER SERVER rem_test
CONNECTION CLOSE ALL
```

- **例 4 –** 接続 ID 142536 を使用するリモート・サーバ rem_test への接続を閉じます。

```
ALTER SERVER rem_test
CONNECTION CLOSE 142536
```

使用法

ALTER SERVER による変更は、次にリモート・サーバに接続するまで有効になりません。

CLASS 句—**CLASS** 句は、サーバのクラスを変更するために指定されます。サーバ・クラスの詳細については、『システム管理ガイド： 第 2 巻』の「リモート・データへのアクセス」と『システム管理ガイド： 第 2 巻』の「リモート・データ・アクセス用のサーバ・クラス」を参照してください。

USING 句—**USING** 句は、サーバの接続情報を変更します。接続情報の詳細については、「CREATE SERVER 文」を参照してください。

CAPABILITY 句—**CAPABILITY** 句は、サーバの機能の ON と OFF を切り替えます。サーバの機能は、システム・テーブル SYSCAPABILITY に格納されています。サーバ機能の名前は、システム・テーブル SYSCAPABILITYNAME に格納されています。リモート・サーバへの最初の接続が確立されるまで、SYSCAPABILITY テーブルには、そのリモート・サーバのエントリは含まれません。最初の接続時に、Sybase IQ はサーバに対して機能に関する問い合わせを行って、SYSCAPABILITY に結果を格納します。後続の接続では、このテーブルからサーバの機能が取得されます。

通常、サーバの機能を変更する必要はありません。クラス ODBC の汎用サーバの機能を変更しなければならない場合はあります。

CONNECTION CLOSE 句—ユーザがリモート・サーバへの接続を確立した場合、ユーザがローカル・データベースから接続を切断しないかぎり、リモート接続は閉じられません。CONNECTION CLOSE 句を使用すると、リモート・サーバへの接続を明示的に閉じることができます。この句は、リモート接続が非アクティブになるか、不要になった場合に便利です。

次の SQL 文は互いに同じであり、リモート・サーバへの現在の接続を閉じるために使用されます。

```
ALTER SERVER server-name CONNECTION CLOSE
```

```
ALTER SERVER server-name CONNECTION CLOSE CURRENT
```

この構文を使用すると、リモート・サーバへの ODBC 接続と JDBC 接続の両方を閉じることができます。これらのいずれの文を実行する場合も、DBA 権限は必要となりません。

また、接続 ID を指定して特定のリモート ODBC 接続を切断することも、ALL キーワードを指定してすべてのリモート ODBC 接続を切断することもできます。接続 ID または ALL キーワードを指定して JDBC 接続を閉じようとすると、エラーが発生します。*connection-id* によって識別される接続が現在のローカル接続でない場合、ユーザが接続を閉じるには、DBA 権限が必要となります。

関連する動作：

- オートコミット。

標準

- SQL—ISO/ANSI SQL 文法のベンダ拡張。
- Sybase—Open Client/Open Server でサポートされています。

パーミッション

このコマンドを実行するには、DBA 権限が必要です。

参照：

- CREATE SERVER 文 (161 ページ)
- DROP SERVER 文 (218 ページ)

ALTER SERVICE 文

Web サービスを変更します。

構文

```
ALTER SERVICE service-name
[ TYPE 'service-type-string' ]
[ attributes ]
[ AS statement' ]
```

パラメータ

- **属性：[AUTHORIZATION { ON | OFF }] [SECURE { ON | OFF }] [USER user-name | NULL }] [URL [PATH] { PATH] { ON | OFF | ELEMENTS }] [USING**

(SOAP-prefix | NULL }] – *service-type-string*: { **'RAW'** | **'HTML'** | **'XML'** | **'SOAP'** | **'DISH'** }

例

- **例 1 –** Web サーバをすばやく設定するには、**-xs** スイッチを指定してデータベース・サーバを起動し、次の文を実行します。

```
CREATE SERVICE tables TYPE 'HTML'
```

```
ALTER SERVICE tables
AUTHORIZATION OFF
USER DBA
AS SELECT * FROM SYS.ISYSTAB
```

これらの文を実行した後、任意の Web ブラウザを使用して http://localhost/tables の URL を開きます。

使用法

この **ALTER SERVICE** 文を実行すると、データベース・サーバが Web サーバとして機能するようになります。

service-name—Web サービスの名前を変更することはできません。

service-type-string—サービスのタイプを識別します。一覧で示されているサービス・タイプのうちいずれかを指定してください。デフォルト値はありません。

AUTHORIZATION 句—ユーザがサービスに接続する際、ユーザ名とパスワードの指定が必要かどうかを定義します。認証が OFF の場合、AS 句が必要となり、**USER** 句によって 1 人のユーザが識別される必要があります。要求はすべて、そのユーザのアカウントとパーミッションを使用して実行されます。

ON の場合は、すべてのユーザにユーザ名とパスワードの入力が必要になります。オプションで、サービスの使用を許可するユーザを制限することもできます。それには、**USER** 句を使用してユーザ名またはグループ名を入力します。ユーザ名に NULL が指定された場合、認識されているすべてのユーザがサービスを使用できます。

デフォルト値は ON です。実稼働システムでは、AUTHORIZATION を ON にし、ユーザをグループに追加してサービスの使用許可を与えることをおすすめします。

SECURE 句—安全でない接続を受け入れるかどうかを指定します。ON は、HTTPS 接続のみ受け入れることを意味します。HTTP ポートで受け取ったサービス要求は、HTTPS ポートへ自動的にリダイレクトされます。OFF に設定すると、HTTP と HTTPS の接続をどちらも受け入れます。デフォルト値は OFF です。

USER 句—AUTHORIZATION が OFF の場合、このパラメータは必須となり、すべてのサービス要求の実行に使用するユーザ ID が指定されます。AUTHORIZATION

が ON (デフォルト) であれば、この句はオプションとなり、サービスへのアクセスを許可するユーザまたはグループが識別されます。デフォルト値は NULL で、すべてのユーザにアクセス許可が与えられます。

URL 句—URI パスを受け入れるかどうか、また、受け入れるのであれば、どのように処理されるかを定義します。OFF の場合、URI 要求のサービス名の後に何も指定することができません。ON の場合は、URI の残りの部分が、url という名前の変数の値として解釈されます。ELEMENTS は、URI パスの残りの部分がスラッシュ文字で区切られて、最大 10 個の要素のリストになるという意味です。値は、url の末尾に 1 から 10 までの数字を付けた名前を持つ変数に割り当てられます。たとえば、3 つ目までの変数の名前は url1、url2、url3 です。指定された値が 10 に満たなければ、残りの変数は NULL に設定されます。サービス名が / の文字で終わっている場合、URL は OFF に設定する必要があります。デフォルト値は OFF です。

USING 句—この句は DISH サービスにのみ適用されます。パラメータには名前のプレフィクスを指定します。このプレフィクスで始まる名前の SOAP サービスだけが処理されます。

statement—statement が NULL の場合、実行する文を URI に指定する必要があります。それ以外の場合は、指定された SQL 文だけがそのサービス内で実行可能になります。SOAP サービスでは文を指定する必要があります。DISH サービスの場合は指定できません。デフォルト値は NULL です。

実稼働システムでは、実行するすべてのサービスに必ず文を定義することをおすすめします。AUTHORIZATION が ON である場合にかぎり、文は NULL でもかまいません。

RAW—結果セットは、それ以上一切フォーマットされないまま、クライアントへ送信されます。必要なタグをプロシージャ内で明示的に生成すれば、フォーマットされたドキュメントを作成できます。

HTML—文またはプロシージャの結果セットが、テーブルを含む HTML ドキュメントに自動的にフォーマットされます。

XML—結果セットは XML フォーマットであると想定されます。XML フォーマットになっていない場合は、自動的に XML RAW フォーマットに変換されます。

SOAP—要求は、有効な Simple Object Access Protocol (SOAP) 要求であることが必要です。結果セットは、自動的に SOAP 応答としてフォーマットされます。SOAP 規格の詳細については、www.w3.org/TR/SOAP を参照してください。

DISH—SOAP ハンドラ、すなわち DISH を定義します。このサービスは、1 つ以上の SOAP サービスのプロキシとして機能します。使用中は、複数の SOAP サービスへのアクセスを保持し、アクセス可能にするコンテナとして機能します。DISH に含まれる SOAP サービスのそれぞれに、Web Services Description Language

(WSDL) ファイルが自動的に生成されます。内部の SOAP サービスは、共通のプレフィクスで識別されます。この識別子は、**USING** 句で指定する必要があります。

『SQL Anywhere サーバー - プログラミング』の「SQL Anywhere Web サービス」>「HTTP Web サーバとしての SQL Anywhere の使用」も参照してください。

注意： このリファレンスは SQL Anywhere マニュアルにリンクされています。

標準

- 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。

パーミッション

DBA 権限が必要です。

参照：
- CREATE SERVICE 文 (163 ページ)
- DROP SERVICE 文 (219 ページ)

ALTER TABLE 文

テーブルの定義を変更します。

構文

```
ALTER TABLE [ owner.]table-name
{ alter-clause, ... }
```

パラメータ

- **alter-clause：** – **ADD** *create-clause* | **ALTER** *column-name column-alteration* | **ALTER** [**CONSTRAINT** *constraint-name*] **CHECK** (*condition*) | **DROP** *drop-object* | **RENAME** *rename-object* | *move-clause* | **SPLIT PARTITION** *partition-name* **INTO** (*partition-decl-1*, *partition-decl-2*) | **MERGE PARTITION** *partition-name-1* **INTO** *partition-name-2* | **UNPARTITION** | **PARTITION BY RANGE** (*partition-key*) *range-partition-decl*
- **create-clause：** – *column-name column-definition* [*column-constraint*] | *table-constraint* | **PARTITION BY** *partitioning-schema*
- **column-alteration：** – { *alterable-column-attribute* } [*alterable-column-attribute…*] | **ADD** [*constraint-name*] **CHECK** (*condition*) | **DROP** { **DEFAULT** | **CHECK** | **CONSTRAINT** *constraint-name* }

- **alterable-column-attribute：** – [**NOT**] **NULL** | **DEFAULT***default-value* | [**CONSTRAINT***constraint-name*] **CHECK** { **NULL** | (*condition*) }
- **column-constraint：** – [**CONSTRAINT** *constraint-name*] { **UNIQUE** | **PRIMARYKEY** | **REFERENCES***table-name* [(*column-name*)] [*actions*] | **CHECK** (*condition*) | **IQ UNIQUE** (*integer*) }
- **drop-object：** – { *column-name* | **CHECK**| | **CONSTRAINT** *constraint-name* | **UNIQUE** (*index-columns-list*) | **PRIMARY KEY** | **FOREIGN KEY** *fkey-name* | **PARTITION** *range-partition-name* }
- **move-clause：** – { **ALTER** *column-name* **MOVE** { **PARTITION** (*partition-name* **TO** *new-dbspace-name*) | **TO** *new-dbspace-name* } } | **MOVE PARTITION** *partition-name* **TO** *new-dbspace-name* | **MOVE TO** *new-dbspace-name* | **MOVE METADATA TO** *new-dbspace-name*
- **rename-object：** – *new-table-name* | *column-name* **TO** *new-column-name* | **CONSTRAINT** *constraint-name* **TO** *new-constraint-name* | **PARTITION** *partition-name* **TO** *new-partition-name*
- **column-definition：** – *column-namedata-type* [**NOTNULL** | **NULL**] [**IN***dbspace-name*] [**DEFAULT***default-value* | **IDENTITY**]
- **default-value：** – *special-value* | *string* | *global variable* | [-] *number* | (*constant-expression*) | *built-in-function* (*constant-expression*) | **AUTOINCREMENT** | **NULL** | **TIMESTAMP** | **LAST USER** | **USER**
- **special-value：** – **CURRENT** { **DATABASE** | **DATE** | **REMOTE USER** | **TIME** | **TIMESTAMP** | **USER** | **PUBLISHER** }
- **table-constraint：** – [**CONSTRAINT** *constraint-name*] { **UNIQUE** (*column-name* [, …]) | **PRIMARYKEY** (*column-name* [, …]) | *foreign-key-constraint* | **CHECK** (*condition*) }
- **foreign-key-constraint：** – **FOREIGNKEY** [*role-name*] [(*column-name* [, …])] ... **REFERENCES***table-name* [(*column-name* [, …])] ... [*actions*] [
- **rename-object：** – *new-table-name* | *column-name* **TO** *new-column-name* | **CONSTRAINT** *constraint-name* **TO** *new-constraint-name* | **PARTITION** *partition-name* **TO** *new-partition-name*
- **range-partitioning-scheme：** – **RANGE**(*partition-key*) (*range-partition-decl* [,*range-partition-decl* ...])
- **partition-key：** – *column-name*
- **range-partition-decl：** – *partition-name* **VALUES** <= ({*constant* | **MAX** }) [**IN***dbspace-name*]
- **actions：** – [**ON { UPDATE | DELETE** } *action*]
- **action：** – { **RESTRICT** }

例

- **例 1 –** 従業員が勤めている事務所を示す新しいカラムを、`Employees` テーブルに追加します。

```
ALTER TABLE Employees
ADD office CHAR(20)
```

- **例 2 –** `office` カラムを `Employees` テーブルから削除します。

```
ALTER TABLE Employees
DROP office
```

- **例 3 –** `Customers` テーブルにカラムを追加して、各顧客の販売担当を割り当てます。

```
ALTER TABLE Customers
ADD SalesContact INTEGER
REFERENCES Employees (EmployeeID)
```

- **例 4 –** 新しいカラム `CustomerNum` を `Customers` テーブルに追加して、デフォルト値の 88 を割り当てます。

```
ALTER TABLE Customers
ADD CustomerNum INTEGER DEFAULT 88
```

- **例 5 –** `c2`、`c4`、`c5` のインデックス **FP** のみが DB 領域 `Dsp3` から `Dsp6` に移動されます。`c1` のインデックス **FP** は `Dsp1` に残ります。`c3` のインデックス **FP** は、`Dsp2` に残ります。`c5` のプライマリ・キーは、`Dsp4` に残ります。**DATE** インデックス `c4_date` は、`Dsp5` に残ります。

```
CREATE TABLE foo (
        c1 INT IN Dsp1,
        c2 VARCHAR(20),
        c3 CLOB IN Dsp2,
        c4 DATE,
        c5 BIGINT,
        PRIMARY KEY (c5) IN Dsp4) IN Dsp3);

    CREATE DATE INDEX c4_date ON foo(c4) IN Dsp5;
    ALTER TABLE foo
        MOVE TO Dsp6;
```

- **例 6 – FP** インデックス `c1` のみを DB 領域 `Dsp1` から `Dsp7` に移動します。

```
ALTER TABLE foo ALTER c1 MOVE TO Dsp7
```

- **例 7 –** この例では、多数の **ALTER TABLE** 句を使用して、パーティションの移動、分割、名前変更、マージを行う方法を示します。

  分割されたテーブルを作成します。

```
CREATE TABLE bar (
        c1 INT,
        c2 DATE,
        c3 VARCHAR(10))
    PARTITION BY RANGE(c2)
    (p1 VALUES <= ('2005-12-31') IN dbsp1,
     p2 VALUES <= ('2006-12-31') IN dbsp2,
     P3 VALUES <= ('2007-12-31') IN dbsp3,
     P4 VALUES <= ('2008-12-31') IN dbsp4);

    INSERT INTO bar VALUES(3, '2007-01-01', 'banana nut');
```

```
    INSERT INTO BAR VALUES(4, '2007-09-09', 'grape jam');
    INSERT INTO BAR VALUES(5, '2008-05-05', 'apple cake');
```

パーティション p2 を dbsp5 に移動します。

```
ALTER TABLE bar MOVE PARTITION p2 TO DBSP5;
```

パーティション p4 を 2 つのパーティションに分割します。

```
ALTER TABLE bar SPLIT PARTITION p4 INTO
        (P41 VALUES <= ('2008-06-30') IN dbsp4,
         P42 VALUES <= ('2008-12-31') IN dbsp4);
```

次の **SPLIT PARTITION** では、データを移動する必要があるため、エラーが報告されます。既存のローが分割後にすべて同じパーティションにあるとはかぎりません。

```
ALTER TABLE bar SPLIT PARTITION p3 INTO
        (P31 VALUES <= ('2007-06-30') IN dbsp3,
         P32 VALUES <= ('2007-12-31') IN dbsp3);
```

次のエラーが報告されます。

```
No data move is allowed, cannot split partition p3.
```

次の **SPLIT PARTITION** では、パーティションの境界値が変更されるため、エラーが報告されます。

```
ALTER TABLE bar SPLIT PARTITION p2 INTO
        (p21 VALUES <= ('2006-06-30') IN dbsp2,
         P22 VALUES <= ('2006-12-01') IN dbsp2);
```

次のエラーが報告されます。

```
Boundary value for the partition p2 cannot be changed.
```

パーティション p3 を p2 にマージします。高い境界値から低い境界値のパーティションへのマージは使用できないため、エラーが報告されます。

```
ALTER TABLE bar MERGE PARTITION p3 into p2;
```

次のエラーが報告されます。

```
Partition 'p2' is not adjacent to or before partition 'p3'.
```

パーティション p2 を p3 にマージします。

```
ALTER TABLE bar MERGE PARTITION p2 INTO P3;
```

パーティション p1 の名前を p1_new に変更します。

```
ALTER TABLE bar RENAME PARTITION p1 TO p1_new;
```

テーブル bar の分割を解除します。

```
ALTER TABLE bar UNPARTITION;
```

テーブル bar を分割します。このコマンドでは、すべてのローが最初のパーティションに含まれている必要があるため、エラーが報告されます。

```
ALTER TABLE bar PARTITION BY RANGE(c2)
    (p1 VALUES <= ('2005-12-31') IN dbsp1,
     P2 VALUES <= ('2006-12-31') IN DBSP2,
     P3 VALUES <= ('2007-12-31') IN dbsp3,
     P4 VALUES <= ('2008-12-31') IN dbsp4);
```

次のエラーが報告されます。

```
All rows must be in the first partition.
```

テーブル bar を分割します。

```
ALTER TABLE bar PARTITION BY RANGE(c2)
    (p1 VALUES <= ('2008-12-31') IN dbsp1,
     P2 VALUES <= ('2009-12-31') IN dbsp2,
     P3 VALUES <= ('2010-12-31') IN dbsp3,
     P4 VALUES <= ('2011-12-31') IN dbsp4);
```

使用法

ALTER TABLE 文は、以前作成したテーブルのテーブル属性 (カラム定義、制約) を変更します。構文では複数の ALTER 句を使用できますが、1 つの **ALTER TABLE** 文の中では 1 つの table-constraint または column-constraint しか追加、修正、削除できないことに注意してください。

注意： ローカル・テンポラリ・テーブルは変更できませんが、グローバル・テンポラリ・テーブルは、テーブルを使用する接続が 1 つだけの場合には変更できます。

Sybase IQ は **REFERENCES** および **CHECK** 制約を適用します。**ALTER TABLE** 文で追加されるテーブル、またはカラムの検査制約は、ALTER TABLE の処理の中では評価されません。 **CHECK** 制約の詳細については、「CREATE TABLE 文」を参照してください。

ビュー定義に **SELECT** * を使用し、その **SELECT** * で参照されるテーブルを変更する場合は、**ALTER VIEW** *<viewname>* **RECOMPILE** を実行してビュー定義を訂正して、ビューをクエリしたときに予期しない結果が返されるのを防ぐ必要があります。

ADD column-definition [column-constraint]—新しいカラムをテーブルに追加します。NOT NULL を指定するには、テーブルが空であることが必要です。**IDENTITY** カラムまたは **DEFAULT AUTOINCREMENT** 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。カラムにデフォルトの IDENTITY 値が指定されていれば、新しいカラムのすべてのローに連続する値が入力されます。また、1 つのカラム・キーに、外部キー制約をカラム制約として追加できます。**IDENTITY/DEFAULT AUTOINCREMENT** カラムの値は、テーブル内の各ローをユニークに識別します。**IDENTITY/DEFAULT AUTOINCREMENT** カラムには、挿入や更新の際に自動

的に生成される連続した数値が格納されます。**DEFAULT AUTOINCREMENT** カラムは、**IDENTITY** カラムとも呼ばれます。**IDENTITY/DEFAULT AUTOINCREMENT** を使用するカラムは、整数データ型のいずれか、または位取りが 0 の真数値型であることが必要です。カラム制約と IDENTITY/DEFAULT AUTOINCREMENT カラムの詳細については、「CREATE TABLE 文」を参照してください。

注意： 外部キー制約を、12.4.3 またはそれ以前のバージョンの Sybase IQ で作成された強制力のないプライマリ・キーに追加することはできません。

ALTER column-name column-alteration—カラムの定義を変更します。次のような修正が可能です。

- SET DEFAULT default-value—テーブルの既存のカラムのデフォルト値を変更します。この作業では **MODIFY** 句も使用できますが、**ALTER** は ISO/ANSI SQL に準拠しているのに対して **MODIFY** は準拠していません。デフォルト値を変更しても、テーブルの既存の値は変更されません。
- DROP DEFAULT—テーブルの既存のカラムのデフォルト値を削除します。この作業では **MODIFY** 句も使用できますが、**ALTER** は ISO/ANSI SQL に準拠しているのに対して **MODIFY** は準拠していません。デフォルトを削除しても、テーブルの既存の値は変更されません。
- ADD—名前付き制約または **CHECK** 条件をカラムに追加します。新しい制約または条件は、それを定義した後でテーブルに対して実行される処理のみに適用されます。テーブルの既存の値は、新しい制約や条件を満すかどうかの検証を受けません。
- CONSTRAINT column-constraint-name—オプションのカラム制約名を指定すると、後で、カラム制約全体を修正するのではなく、制約を個別に修正または削除できます。
- [CONSTRAINT constraint-name] CHECK (condition)—この句は、カラムの **CHECK** 制約を追加するときに使用します。
- SET COMPUTE (expression)—計算カラムに関連付けられた式を変更します。この文が実行されると、カラムの値は再計算され、新しい式が無効な場合は文が失敗します。
- DROP COMPUTE—計算カラムから非計算カラムに変更します。この文はテーブル内の既存の値を変更しません。

DROP partition 句—指定したパーティションを削除します。ローおよびパーティション定義が削除されます。最後のパーティションは削除できません。これは、分割されたテーブルが非分割テーブルに変換されるためです (分割されたテーブルをマージするには、**UNPARTITION** 句を代わりに使用します)。次に例を示します。

```
CREATE TABLE foo (c1 INT, c2 INT)
    PARTITION BY RANGE (c1)
    (P1 VALUES <= (100) IN dbsp1,
     P2 VALUES <= (200) IN dbsp2,
     P3 VALUES <= (MAX) IN dbsp3
```

```
        ) IN dbsp4);
    LOAD TABLE ….
    ALTER TABLE DROP PARTITION P1;
```

ADD table-constraint—テーブルに制約を追加します。シングルカラムまたはマルチカラムのキーに、外部キー制約をテーブル制約として追加することもできます。テーブル制約の詳細については、「CREATE TABLE 文」を参照してください。

PRIMARY KEY を指定する場合、テーブルには **CREATE TABLE** 文または別の **ALTER TABLE** 文で作成したプライマリ・キーがあってはなりません。

注意： テーブルまたはカラムの制約は **MODIFY** (変更) できません。制約を変更するには、古い制約を **DELETE** (削除) し、新しい制約を **ADD** (追加) します。

DROP column-name—カラムをテーブルから削除します。カラムがマルチカラム・インデックス、一意性制約、外部キー、またはプライマリ・キーに含まれている場合は、インデックス、制約またはキーを削除してからカラムを削除してください。このようにするとカラムを参照する **CHECK** 制約は削除されません。IDENTITY/DEFAULT AUTOINCREMENT カラムを削除できるのは、テーブルの **IDENTITY_INSERT** が OFF に設定され、かつテーブルがローカル・テンポラリ・テーブルでない場合だけです。

DROP CHECK—テーブルのすべての検査制約を削除します。テーブル検査制約とカラム検査制約の両方が対象となります。

DROP CONSTRAINT constraint-name—テーブルまたは指定したカラムの名前付き制約を削除します。

DROP UNIQUE (column-name,…)—指定したカラムにおいて一意性制約を削除します。一意性制約を参照する外部キー (プライマリ・キーではなく) も削除します。関連する外部キー制約がある場合は、エラーが報告されます。**ALTER TABLE** を使用して、プライマリ・キーを参照するすべての外部キーを削除した後でなければ、プライマリ・キー制約を削除することはできません。

DROP PRIMARY KEY—プライマリ・キーを削除します。このテーブルのプライマリ・キーを参照するすべての外部キーも削除します。関連する外部キー制約がある場合は、エラーが報告されます。プライマリ・キーに強制力がない場合、そのプライマリ・キーに強制力のない外部キー制約が存在すると、**DELETE** はエラーを返します。

DROP FOREIGN KEY role-name—特定の役割名を持つ、該当テーブルの外部キー制約を削除します。その外部キー制約に対して自動的に作成された、ユニークでない **HG** インデックスは削除されません。**DROP INDEX** 文を使用して、**HG** インデックスを明示的に削除してください。

DROP PARTITION—パーティション P1 内のローと P1 のパーティション定義を削除します。カラム c1 に値 99 を含んでいる新しいローが挿入されると、DB 領域 dbsp2 内のパーティション p2 に配置されます。

RENAME new-table-name—テーブル名を *new-table-name* に変更します。古いテーブル名を使用しているアプリケーションがある場合は、修正が必要になります。また、古いテーブル名と同じ名前を自動的に割り当てられた外部キーの名前は、変更しません。

RENAME column-name TO new-column-name—カラムの名前を *new-column-name* に変更します。古いカラム名を使用しているアプリケーションがある場合は、修正が必要になります。

RENAME constraint-name TO new-constraint-name—制約の名前を *new-constraint-name* に変更します。古い制約名を使用しているアプリケーションがある場合は、修正が必要になります。

ALTER TABLE 文は、他の接続で現在使用中のテーブルに影響を及ぼす場合は実行できません。**ALTER TABLE** は処理に時間がかかり、この文の処理中は、同じテーブルを参照する要求がサーバで処理されません。

ALTER Column MOVE TO—指定したカラムを非分割テーブルの新しい DB 領域に移動します。**ALTER** *Column* **MOVE TO** 句は、分割テーブルに対しては要求できません。**ALTER** *Column* **MOVE PARTITION** 句は、指定したパーティションのカラムを指定した DB 領域に移動します。

MOVE PARTITION—指定したパーティションを新しい DB 領域に移動します。

MOVE TO—テーブルが新しい DB 領域にマッピングされると、そのテーブルと同じ DB 領域に存在するカラム、インデックス、一意性制約、プライマリ・キー、外部キー、メタデータなどのすべてのテーブル・オブジェクトを新しい DB 領域に移動します。

各テーブル・オブジェクトは 1 つの DB 領域にのみ置くことができます。どのタイプの **ALTER MOVE** も、移動中はテーブルへの変更をすべてブロックします。

MOVE TABLE METADATA—テーブルのメタデータを新しい DB 領域に移動します。分割されたテーブルでは、**MOVE TABLE METADATA** 句は、パーティション間で共有されるメタデータも移動します。

DBA 権限または SPACE ADMIN 権限があるか、新しい DB 領域に対する CREATE 権限があり、かつテーブル所有者であるか、またはテーブルの alter パーミッションを持っている必要があります。

SPLIT PARTITION—指定したパーティションを 2 つのパーティションに分割します。パーティションはデータを移動しない場合にかぎり、分割できます。分割さ

れるパーティションのすべての既存のローは、分割後に 1 つのパーティションに残す必要があります。*partition-decl-1* の境界値は、*partition-name* の境界値より小さく、*partition-decl-2* の境界値は、*partition-name* の境界値と一致する必要があります。2 つの新しいパーティションにはそれぞれ異なる名前を指定できます。新しい名前が指定されていない場合、古い *partition-name* は 2 番目のパーティションのみに使用できます。

MERGE PARTITION—*partition-name-1* を *partition-name-2* にマージします。2 つのパーティションが隣接しており、データが同じ DB 領域にある場合は、パーティションをマージできます。低いパーティションの値を持つパーティションを高いパーティションの値を持つパーティションにマージする場合にのみ、パーティションをマージできます。サーバは、パーティションのマージ先の DB 領域に対する **CREATE** パーミッションをチェックしないことに注意してください。隣接するパーションを作成する方法の例については、「CREATE TABLE 文」の例 3 を参照してください。

UNPARTITION—分割されたテーブルからパーティションを削除します。各カラムは 1 つの DB 領域に配置されます。サーバは、すべてのパーティションのデータの移動先の DB 領域に対する **CREATE** パーミッションをチェックしないことに注意してください。**ALTER TABLE UNPARTITION** は、データベースのアクティビティをすべてブロックします。

PARTITION BY—未分割のテーブルを分割します。すべての既存のローが最初のパーティションに属する場合は、未分割のテーブルを分割できます。最初のパーティションには、カラムまたはテーブルの DB 領域とは別の DB 領域を指定できます。ただし、既存のローは移動されません。代わりに、カラム／パーティションに適した DB 領域が、既存のカラム用に `SYS.ISYSIQPARTITIONCOLUMN` に保存されます。最初のパーティション用に指定した DB 領域には、最初のパーティション用に後で追加されるデフォルトまたは最大 identity カラムのみが保存されます。

RENAME PARTITION—既存のパーティション名を新しいパーティション名に変更します。

関連する動作：

- オートコミット。**ALTER** と **DROP** オプションは現在の接続に対するすべてのカーソルをクローズします。**dbisql** データ・ウィンドウもクリアします。
- **ALTER TABLE** 操作の始めでチェックポイントを実行します。
- カラムまたはテーブルを変更すると、その変更したカラムを参照するストアド・プロシージャ、ビューなどは機能しなくなります。

標準

- SQL—ISO/ANSI SQL 文法のベンダ拡張。
- Sybase—一部の句は、Adaptive Server Enterprise でサポートされています。

パーミッション

MOVE 構文の場合は、次のいずれかに該当する必要があります。

- DBA 権限または SPACE ADMIN 権限を持っている。
- 新しい DB 領域に対する CREATE パーミッションがあり、かつテーブル所有者である。
- テーブルに対する ALTER パーミッションを持っている。

MOVE 以外の構文の場合は、次のいずれかに該当する必要があります。

- DBA 権限を持っている。
- 新しい DB 領域に対する CREATE パーミッションがあり、かつテーブル所有者である。
- テーブルに対する ALTER パーミッションを持っている。

テーブルへの排他的なアクセスが必要である。

参照：

- CREATE TABLE 文 (166 ページ)
- DROP 文 (209 ページ)
- IDENTITY_INSERT オプション (458 ページ)

ALTER TEXT CONFIGURATION 文

テキスト設定オブジェクトを変更します。

構文

以下を参照してください。

使用法

構文と詳細な説明については、『Sybase IQ の非構造化データ分析の概要』を参照してください。

ALTER TEXT INDEX 文

TEXT・インデックスの定義を変更します。

構文

以下を参照してください。

使用法

構文と詳細な説明については、『Sybase IQ の非構造化データ分析の概要』を参照してください。

ALTER USER 文

ユーザ設定を変更します。

構文

構文 1

ALTER USER *user-name* [**IDENTIFIED BY** *password*] [**LOGIN POLICY** *policy-name*] [**FORCE PASSWORD CHANGE** { **ON** | **OFF** }]

構文 2

ALTER USER *user-name* [**RESET LOGIN POLICY**]

例

- **例 1** – ユーザ `SQLTester` を変更します。パスワードは "welcome" に設定されます。`SQLTester` ユーザは `Test1` ログイン・ポリシーに割り当てられ、パスワードの有効期限は次回ログイン時に切れません。

  ```
  ALTER USER SQLTester
  IDENTIFIED BY welcome
  LOGIN POLICY Test1
  FORCE PASSWORD CHANGE OFF
  ```

  ユーザがこのコマンドを実行するには、USER ADMIN 権限と PERMS ADMIN 権限または DBA 権限が必要です。PERMS ADMIN 権限はパスワードの変更に必要であり、USER ADMIN 権限はログイン・ポリシーの変更に必要です。

使用法

user--name—ユーザ名。

IDENTIFIED BY 句—ユーザのパスワードを入力する句。

policy-name—ユーザに割り当てるログイン・ポリシーの名前。**LOGIN POLICY** 句が指定されていない場合、変更は行われません。

FORCE PASSWORD CHANGE 句—ユーザがログイン時に新しいパスワードを指定する必要があるかどうかを制御します。この設定は、ポリシー内の PASSWORD_EXPIRY_ON_NEXT_LOGIN オプション設定を上書きします。

RESET LOGIN POLICY 句—ユーザのログイン設定をログイン・ポリシー内の元の値に戻します。これによって、通常は、ユーザのログイン失敗回数が最大値を超えている、または前回ログイン以降に経過した日数が最大値を超えているために、暗示的に設定されたロックがすべてクリアされます。ログイン・ポリシーをリセットすると、ユーザは MAX_FAILED_LOGIN_ATTEMPTS または MAX_DAYS_SINCE_LOGIN などのログイン・ポリシー・オプションの制限を超えているためにロックされているアカウントにアクセスできます。

ログイン・ポリシーの管理の詳細については、『SQL Anywhere サーバー – データベース管理』の「データベースの設定」>「ユーザー ID、権限、パーミッションの管理」>「ログインポリシーの管理」を参照してください。

注意： このリファレンスは SQL Anywhere マニュアルにリンクされています。

マルチプレックスに使用する強化された **ALTER LOGIN POLICY** 構文は、『Sybase IQ Multiplex の使用』で説明します。

ユーザ ID とパスワードは、次のようには指定できません。

- 最初の文字をスペース、一重引用符または二重引用符にする。
- 最後の文字をスペースにする。
- セミコロンを含める。

PASSWORD_EXPIRY_ON_NEXT_LOGIN 値を ON に設定すると、このログイン・ポリシーに割り当てられたすべてのユーザのパスワードは、次回ログイン時にすぐに期限切れになります。次回ログイン時にユーザにパスワードを変更させるようにするには、**ALTER USER** と **LOGIN POLICY** 句を使用します。

標準

- 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。

パーミッション

ユーザは自分自身のパスワードを変更できます。他のユーザのパスワードを変更するには、DBA 権限または PERMS ADMIN 権限が必要です。**ALTER USER LOGIN**

POLICY、**FORCE PASSWORD CHANGE**、または **RESET LOGIN POLICY** を使用するには、DBA 権限または USER ADMIN 権限が必要です。

参照：

- COMMENT 文 (72 ページ)
- CREATE LOGIN POLICY 文 (132 ページ)
- CREATE USER 文 (182 ページ)
- DROP LOGIN POLICY 文 (215 ページ)
- DROP USER 文 (221 ページ)
- GRANT 文 (247 ページ)
- REVOKE 文 (331 ページ)
- ALTER LOGIN POLICY 文 (22 ページ)

ALTER VIEW 文

ビューの定義を修正したものに置き換えます。

構文

構文 1 –ビューの構造を変更します。

```
ALTER VIEW
… [ owner.]view-name [ ( column-name [ , … ] ) ]
… AS select-statement
… [ WITH CHECK OPTION ]
```

構文 2 – ビューの属性を変更します。

```
ALTER VIEW
… [ owner.]view-name
… { SET HIDDEN | RECOMPILE | DISABLE | ENABLE }
```

使用法

AS—**CREATE VIEW** 文と同じ目的と構文。「CREATE VIEW 文」を参照してください。

WITH CHECK OPTION—**CREATE VIEW** 文と同じ目的と構文。「CREATE VIEW 文」を参照してください。

SET HIDDEN—ビューと句の定義を難読化し、このビューを Sybase Central などのビューから見えないようにします。ビューを明示的に参照することは可能です。

警告！ SET HIDDEN 操作を、元に戻すことはできません。

RECOMPILE—ビューのカラム定義を再作成します。機能的には **ENABLE** 句と同じですが、無効になっていないビューで使用できるという点で異なります。

DISABLE—データベース・サーバでビューを使用できなくします。

ENABLE—無効になったビューを有効にします。これによって、データベース・サーバはビューのカラム定義を再作成します。ビューを有効にする前は、そのビューが使用するビューをすべて有効にする必要があります。

ビューを変更すると、ビューに対する既存のパーミッションはそのまま維持されます。このため、パーミッションの再割り当ては必要ありません。**ALTER VIEW** 文を使用する代わりに、**DROP VIEW** を使用してビューを削除したり、**CREATE VIEW** を使用してビューを再作成したりすることもできます。この操作を実行する場合は、ビューのパーミッションを再割り当てする必要があります。

構文 1 を使用してビューの変更を完了したら、データベース・サーバはビューを再コンパイルします。実行した変更のタイプによっては、依存するビューがある場合は、データベース・サーバはそれらのビューを再コンパイルしようとします。従属ビューに影響を及ぼす変更を加えると、従属ビューが無効になることがあり、従属ビューの定義の変更も必要になることがあります。

警告！ ビューを定義する **SELECT** 文にアスタリスク (*) が付いており、基になるテーブルからカラムが追加または削除されている場合は、ビュー内のカラムの数が変化する可能性があります。ビュー・カラムの名前とデータ・タイプも変更される場合があります。

構文 1—ビューの構造を変更します。テーブルを変更する場合、変更は個々のカラムのみに制限できますが、これとは異なり、ビューの構造を変更する場合は、ビューを作成する場合と同じように、ビューの定義全体を新しい定義に置き換える必要があります。ビューの構造を定義するのに使用するパラメータの説明については、「CREATE VIEW 文」を参照してください。

構文 2—ビューの定義が非表示になっているかどうかなど、ビューの属性を変更します。

SET HIDDEN を使用すると、ビューを別のデータベースにアンロードしたり、再ロードしたりできます。デバッガを使用したデバッグでは、ビューの定義は表示されず、プロシージャ・プロファイリングでも表示されません。非表示のビュー定義を変更するには、ビューを削除してから、**CREATE VIEW** 文を使用して再作成します。

DISABLE 句を使用すると、ビューはデータベース・サーバでクエリに応答するために使用できなくなります。ビューを無効にしても、ビューの定義はデータベースに残りますが、それ以外の点では、ビューを削除するようなものです。ビューを無効にすると、依存するビューも無効になります。したがって、**DISABLE** 句は、無効になっているビューだけでなく、同様に無効になっている従属ビューへの排他的なアクセスも必要とします。

『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、関数、ビューの内容を隠す」と『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「システム・プロシージャ」>「システム・ストアド・プロシージャ」>「sa_dependent_views プロシージャ」を参照してください。

データベース・サーバがビューの依存性を処理する方法の詳細な説明については、『SQL Anywhere サーバー – SQL の使用法』の「データベースオブジェクトの使用」>「ビューの操作」>「ビューの依存性」を参照してください。

注意： このリファレンスは SQL Anywhere マニュアルにリンクされています。

関連する動作：

- オートコミット。
- プロシージャとトリガはすべてメモリからアンロードされるため、ビューを参照するプロシージャとトリガは新しいビューの定義を反映します。定期的にビューを変更する場合は、プロシージャとトリガをアンロードしたり、ロードしたりすると、パフォーマンスが低下する可能性があります。

標準

- 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。

パーミッション

ビューの所有者であるか、DBA 権限を持っている必要があります。

参照：

- CREATE VIEW 文 (186 ページ)
- DROP 文 (209 ページ)
- 無効な従属ビューの特定と修正 (51 ページ)

無効な従属ビューの特定と修正

基本となるテーブルに対する変更によって無効になった従属ビューがないかどうかをチェックし、存在する場合には修正します。

基本となるテーブルが変更されると、通常は、データベース・サーバはビューを自動的に再コンパイルしてビューの有効性を維持します。ただし、テーブルの変更により、ビュー定義によって参照されているものが削除されたり、大幅に変更されたりすると、従属ビューが無効になることがあります。たとえば、ビュー定義で参照されているカラムが削除されると、従属ビューは無効になります。
ビュー定義を修正し、ビューを手動で再コンパイルします。

1. **sa_dependent_views** を実行して、従属ビューのリストを取得します。
2. テーブルを変更する DDL 操作を実行します。サーバは従属ビューを自動的に無効にし、DDL が完了したらそれらの再コンパイルを試行します。
3. **sa_dependent_views** によってリストされるすべてのビューが無効になっていることを確認します。たとえば、**SELECT * FROM myview** のような単純なテストを実行します。
4. ビューが無効な場合は、その問題を解決するためにビュー定義の変更が必要になることもあります。実行した DDL 変更に対するビュー定義を検証し、必要な変更を行います。**ALTER VIEW RECOMPILE** を実行して、ビュー定義を修正します。
5. 修正したビューをテストして、機能することを確認します。たとえば、**SELECT * FROM myview** のような単純なテストを実行します。

sa_dependent_views は、特定のテーブルまたはビューのすべての従属ビューのリストを返します。『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「システム・プロシージャ」>「システム・ストアド・プロシージャ」>「sa_dependent_views プロシージャ」および『SQL Anywhere サーバー - SQL リファレンス』>「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_dependent_views システムプロシージャー」を参照してください。

データベース・サーバがビューの依存性を処理する方法の詳細な説明については、『SQL Anywhere サーバー – SQL の使用法』の「データベースオブジェクトの使用」>「ビューの操作」>「ビューの依存性」を参照してください。

注意： これらのリファレンスは SQL Anywhere マニュアルにリンクされています。

参照：

- ALTER VIEW 文 (49 ページ)

BACKUP 文

1 つまたは複数のアーカイブ・デバイスに Sybase IQ データベースをバックアップします。

構文

```
BACKUP DATABASE
[ backup-option… ]
TO archive_device [ archive-option... ]
… [ WITH COMMENT string ]
```

パラメータ

- **backup-option： –**

```
{ READWRITE FILES ONLY |
READONLY dbspace-or-file [, … ] }
CRC { ON | OFF }
ATTENDED { ON | OFF }
BLOCK FACTOR integer
{ FULL | INCREMENTAL | INCREMENTAL SINCE FULL }
VIRTUAL { DECOUPLED |
                    ENCAPSULATED 'shell_command' }
WITH COMMENT comment
```

- **dbspace-or-file： –**

```
{ DBSPACES identifier-list | FILES identifier-list }
```

- **identifier-list： –** *識別子* [, …]
- **archive-option： – SIZE***integer***STACKER***integer*

例

- **例 1 –** 次の UNIX の例は、`iqdemo` データベースを、Sun Solaris プラットフォームのテープ・デバイスである `/dev/rmt/0` と `/dev/rmt/2` にバックアップします。Solaris の場合は、デバイス名に続けて *n* の文字を入力すると、"no rewind on close" 機能が指定されます。**BACKUP** では、お使いの UNIX プラットフォームに適した命名規則を使用して、必ずこの機能を指定してください (Windows はこの機能をサポートしていません)。次の例では、前回のフル・バックアップ以降にデータベースに加えられた全変更をバックアップします。

```
BACKUP DATABASE
INCREMENTAL SINCE FULL
TO '/dev/rmt/0n' SIZE 10000000
TO '/dev/rmt/2n' SIZE 15000000
```

注意： サイズの単位はキロバイト (KB) です。ただし、ほとんどの場合、1GB 未満のサイズは不適切です。この例では、指定サイズは 10GB と 15GB です。

- **例 2 –** 次の **BACKUP** コマンドは、読み込み専用ファイルと DB 領域を指定します。

```
BACKUP DATABASE READONLY DBSPACES dsp1
TO '/dev/rmt/0'
```

```
BACKUP DATABASE READONLY FILES dsp1_f1, dsp1_f2
TO 'bkp.f1f2'
```

```
BACKUP DATABASE READONLY DBSPACES dsp2, dsp3
READONLY FILES dsp4_f1, dsp5_f2
TO 'bkp.RO'
```

使用法

BACKUP コマンドの実行時に、多くの読み込みユーザと書き込みユーザが IQ データベースをオープンして使用している場合があります。 その場合、BACKUP は読み込み専用ユーザとして動作し、Sybase IQ のテーブル・レベルのバージョン管理機能を使用してデータの一貫性を維持します。

BACKUP は開始の前に **CHECKPOINT** を暗黙的に発行し、その後、データベースを記述したカタログ・テーブル (およびカタログ・ストアに追加したすべてのテーブル) をバックアップします。この最初のフェーズ中、Sybase IQ ではデータベースのメタデータの変更 (カラムやテーブルの追加や削除など) は行えません。したがって、後でバックアップの **RESTORE** を行っても、最初の **CHECKPOINT** までしかリストアできません。

BACKUP コマンドでは、フル・バックアップまたはインクリメンタル・バックアップの指定が可能です。インクリメンタル・バックアップには、2 種類の選択肢があります。**INCREMENTAL** は最後の **BACKUP** (インクリメンタルまたはフル) 以降に変更またはコミットしたブロックのみをバックアップします。**INCREMENTAL SINCE FULL** は最後のフル・バックアップ以降に変更したブロックをすべてバックアップします。最初のタイプのインクリメンタル・バックアップは、**BACKUP** コマンドのバックアップ量が小さく実行速度が速くなりますが、**RESTORE** コマンドの実行速度が遅く複雑になります。後のタイプのインクリメンタル・バックアップは、その反対になります。その理由は、最初のタイプからは通常、フル・バックアップ・アーカイブごとに N セットのインクリメンタル・バックアップ・アーカイブが生成されるからです。リストアが必要な場合は、最初にフル・バックアップ・アーカイブを **RESTORE** してから、各インクリメンタル・アーカイブを正しい順序でリストアしてください (どの順序で必要になるかは、Sybase IQ に記録されます)。2 番目のタイプでは DBA はフル・バックアップ・アーカイブと最後のインクリメンタル・アーカイブをリストアするだけで済みます。

インクリメンタル仮想バックアップは、**BACKUP** 文の **VIRTUAL DECOUPLED** パラメータと **VIRTUAL ENCAPSULATED** パラメータによってサポートされます。

テーブルスペースの OS レベル・コピーを実行して、1 つまたは複数の読み込み専用 DB 領域の仮想バックアップを行うことができますが、仮想バックアップは IQ システム・テーブル内にバックアップを記録するので、仮想バックアップ文を使用することをおすすめします。『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「システム・テーブルとシステム・ビュー」>「システム・ビュー」>「SYSIQBACKUPHISTORY システム・ビュー」と『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「システム・テーブルとシステム・ビュー」>「システム・ビュー」>「SYSIQBACKUPHISTORYDETAIL システム・ビュー」を参照してください。

READWRITE FILES ONLY を **FULL**、**INCREMENTAL**、**INCREMENTAL SINCE FULL** と一緒に使用すれば、バックアップをデータベース内の読み込み／書き込みファイルのセットのみに制限できます。読み込み／書き込み DB 領域／ファイルは IQ DB 領域である必要があります。

READWRITE FILES ONLY を **INCREMENTAL** バックアップまたは **INCREMENTAL SINCE FULL** バックアップに使用すると、依存バックアップ以降に変更した読み込み専用の DB 領域または dbfile のデータはバックアップされません。**INCREMENTAL** バックアップまたは **INCREMENTAL SINCE FULL** バックアップに **READWRITE FILES ONLY** が指定されていない場合は、依存バックアップ以降に変更した、読み込み／書き込みと読み込み専用 DB 領域の両方のすべてのデータベース・ページがバックアップされます。

CRC 句 – ブロックごとの 32 ビット周期の冗長性検査をアクティブにします (ハードウェアで使用可能なエラー検出もすべてアクティブにします)。この句を指定すると、その後に **RESTORE** 操作を行った場合に、バックアップで計算された数字が検証されるため、両方のコマンドのパフォーマンスに影響を及ぼします。デフォルトは ON です。

ATTENDED 句 – この句は、テープ・デバイスにバックアップを作成する場合にのみ適用されます。**ATTENDED ON** (デフォルト値) を指定した場合、テープ・ドライブが介入を必要とするときに、**BACKUP** 文を発行したアプリケーションにメッセージが送信されます。これは、たとえば、新しいテープが必要になった場合に発生します。OFF を指定すると、**BACKUP** は新しいテープを要求するメッセージを表示しません。追加のテープが必要で、OFF が指定されている場合、Sybase IQ はエラーを送信し **BACKUP** コマンドをアボートします。ただし、自動スタッカ・ドライブがテープを切り替えるための時間を考慮して、短い遅延が含まれています。

BLOCK FACTOR 句 – 同時に書き込むブロック数を指定します。値は 0 より大きくしてください。そうしないと、Sybase IQ からエラー・メッセージが生成されます。デフォルトは (小さいサイズの固定テープ・ブロックを使用するため)、UNIX システムの場合は 25、Windows システムの場合は 15 です。この句はバッファが使用するメモリを効率よく制御します。実際のメモリ量は、この値にブロック・サイズを掛けて、データベースからのデータの抽出に使用するスレッド数を掛けたものです。**BLOCK FACTOR** を最低でも 25 に設定することをおすすめします。

FULL 句 – フル・バックアップを指定します。データベースで使用中の全ブロックをアーカイブ・デバイスに保存します。これがデフォルトです。

INCREMENTAL 句 – インクリメンタル・バックアップを指定します。最後のすべての種類のバックアップ以降に変更された全ブロックをアーカイブ・デバイスに保存します。

キーワード **INCREMENTAL** は、**READONLY FILES** では使用できません。

INCREMENTAL SINCE FULL 句 – インクリメンタル・バックアップを指定します。最後のフル・バックアップ以降に変更された全ブロックをアーカイブ・デバイスに保存します。

VIRTUAL DECOUPLED 句 – 分離仮想バックアップを指定します。バックアップを完了させるには、分離仮想バックアップが終了した後に IQ の DB 領域をコピーし、非仮想のインクリメンタル・バックアップを実行する必要があります。

VIRTUAL ENCAPSULATED 句 – カプセル化仮想バックアップを指定します。'shell-command' 引数には、カプセル化仮想バックアップの中で実行されるシェル・コマンドを、文字列または文字列を含む変数で指定します。シェル・コマンドは、バックアップ処理の中で、IQ ストアのシステム・レベルのバックアップを実行します。

TO 句 – バックアップで使用する archive_device の名前を一重引用符で囲んで指定します。archive_device はアーカイブ・ファイル用のファイル名またはテープ・ドライブ・デバイス名です。複数のアーカイブ・デバイスを使用する場合、それぞれに別々の **TO** 句を使用して指定します (カンマで区切ったリストは使用できません)。アーカイブ・デバイスは異なるものでなければなりません。出力デバイスに関して Sybase IQ が試みる並行処理の量は、**TO** 句の数によって決まります。

BACKUP と **RESTORE** によって、指定したすべてのアーカイブ・デバイスからの IQ データの取得とそれらへの書き込みが並行して行われます。その後、カタログ・ストアが最初のデバイスに書き込まれます。バックアップ時間とリストア時間を短縮するには、優れた並行処理を行う必要があります。

Sybase IQ では、最大 36 のハードウェア・デバイスのバックアップがサポートされています。バックアップ時間を短縮するには、コアごとに 1 つまたは 2 つのデバイスを指定して、ハードウェアおよび IO の競合を回避します。**BACKUP** コマンドに **SIZE** パラメータを設定して各バックアップ・デバイスに複数のファイルを作成することを防止し、**BACKUP** コマンドの **BLOCK FACTOR** 句に使用する値を検討します。

古いファイルを移動するか、*archive_device* 名やパスに以前と異なるものを指定しないかぎり、**BACKUP** は既存のアーカイブ・ファイルを上書きします。

バックアップ API の DLL 版では、アーカイブ・デバイスのオープン時に DLL に渡す引数を指定できます。サード・パーティの実装の場合は、archive_device 文字列は次のフォーマットになります。

```
'DLLidentifier::vendor_specific_information'
```

具体例：

```
'spsc::workorder=12;volname=ASD002'
```

archive_device 文字列の最大長は 1023 バイトです。*DLLidentifier* 部分は 1 から 30 バイトの長さで、英数字とアンダースコアの文字だけが使用できます。文字列の *vendor_specific_information* 部分は、内容の検査なしでサード・パーティの実装に渡されます。サード・パーティ版を使用している場合、**BACKUP** コマンドの **SIZE** 句または **STACKER** 句は指定しないでください。この情報は文字列の *vendor_specific_information* 部分にコード化する必要があるためです。

注意： この構文を Sybase IQ で使用して動作確認されているサード・パーティ製品は、ごく一部に限られます。使用上のその他の指示や制約については、『リリース・ノート』を参照してください。サード・パーティ製品をこの方法で使用して Sybase IQ データベースをバックアップする場合は、その製品が動作確認されたものであるかどうかを事前に確認してください。『リリース・ノート』または Technical Documents Web サイト (http://www.sybase.com/support/techdocs/) の Sybase IQ 製品に対する Sybase Certification Reports を参照してください。

バックアップ API の Sybase 版の場合、テープ・デバイス名またはファイル名以外の情報を指定する必要はありません。ディスク・デバイスの場合は、SIZE 値も指定してください。指定しなければ、Sybase IQ は作成される各ディスク・ファイルが 2GB (UNIX の場合) または 1.5GB (Windows の場合) よりも大きくならないものと想定します。特定の UNIX システムのテープ・デバイスを指定する Sybase API DLL 用のアーカイブ・デバイスの例を示します。

```
'/dev/rmt/0'
```

SIZE 句—出力デバイスごとのテープまたはファイルの最大容量を指定します (テープの終了マーカを正確に検出できないプラットフォームもあります)。対応するデバイスで使用するボリュームは、この値より小さくしないでください。この値は、テープとディスク・ファイルの両方に適用されますが、サード・パーティのデバイスには適用されません。

単位はキロバイト (KB) です。ただし、ほとんどの場合、1GB 未満は不適切です。たとえば、3.5GB テープの場合は 3500000 を指定します。デフォルトはプラットフォームと媒体によって異なります。バックアップ・ファイルの最終的なサイズは正確ではありません。これは、バックアップでは、大きなデータ・ブロック単位で書き込まれるためです。

SIZE パラメータは、出力デバイス単位で指定します。**SIZE** はデバイスごとのバイト数を制限しません。**SIZE** はファイル・サイズを制限します。出力デバイスごとに異なる **SIZE** パラメータを指定できます。

バックアップ中、指定されたデバイスに書き込まれる情報量が **SIZE** パラメータに指定された値に達した場合、**BACKUP** は次のいずれかの方法で処理します。

- ファイル・システム・デバイスの場合、**BACKUP** は現在のファイルを閉じて別のファイルを同じ名前で作成します。このファイル名には、たとえば

`bkup1.dat1.1`、`bkup1.dat1.2`、`bkup1.dat1.3` のように、昇順で次の番号が付加されます。
- テープ・ユニット・デバイスの場合、**BACKUP** は現在のテープを閉じます。このとき、ユーザは別のテープをマウントする必要があります。

追加のテープが必要になった場合のマウント処理、またはディスクにバックアップで必要になる領域が十分にあるかを確認する作業は、ユーザが自分で行ってください。

複数のデバイスを指定すると、**BACKUP** は情報をすべてのデバイスに配布します。

表 2 : BACKUP のデフォルト・サイズ

プラットフォーム	SIZE のデフォルト (テープ)	SIZE のデフォルト (ディスク)
UNIX	なし	2GB
Windows	1.5GB **SIZE** には 64 の倍数を指定してください。それ以外の値は、64 の倍数まで切り捨てられます。	1.5GB

STACKER 句—デバイスの自動的なロードを指定し、ロード時のテープ数を指定します。この値は、スタッカのテープの位置ではありません。テープの位置は 0 になります。 **ATTENDED** が OFF で **STACKER** が ON に設定されている場合、Sybase IQ は事前に定義された時間を待機してから、次のテープの自動的なロードを許可します。 **SIZE** 句を使用して指定されたテープ数は、バックアップ・データを格納する十分な領域があるかどうかを判定するために使用されます。 この句は、サード・パーティのメディア管理デバイスには使用しないでください。

WITH COMMENT 句—アーカイブ・ファイルとバックアップ・ヒストリ・ファイルに記録するコメント (オプション) を指定します。最大長は 32KB です。値を指定しなければ、NULL 文字列が格納されます。

BACKUP では、他にも以下の点に注意してください。

- **BACKUP** ではロー・デバイスをアーカイブ・デバイスとして使用することはできません。
- Windows システムがサポートするのは、テープ・デバイスに対する固定長の I/O 操作のみです (この制限の詳細については、『インストールおよび設定ガイド』を参照)。 Windows はテープ・パーティションをサポートしていますが、Sybase IQ はテープ・パーティションを使用しません。このため、別のアプリケーションを使用して **BACKUP** 用のテープをフォーマットしないでください。Windows のテープ・デバイスの命名方式は非常に簡単です。1 番目のテープ・デバイスは *¥¥.¥tape0*、2 番目は *¥¥.¥tape1* というように命名されます。

警告！ バックアップ (およびその他のほとんどの状況) では、文字列中に出現する ¥n、¥x または ¥¥ のうちの最初の ¥ は、Sybase IQ によりエスケープ文字として処理されます。 このためバックアップ用のテープ・デバイスを指定する場合は、Windows の命名規則に従って必要な場所に ¥ を 2 つずつ使用しなければなりません。 たとえば、バックアップに使用する 1 番目の Windows テープ・デバイスを '¥¥¥¥.¥¥tape0'、2 番目を '¥¥¥¥.¥¥tape1' とします。 追加すべき ¥ を省略したり、テープ・デバイス名の入力を間違えたり、システム上にないテープ・デバイス名を入力したりすると、Sybase IQ はそれをディスク・ファイル名と解釈します。

- Sybase IQ は、テープを使用する前にリワインドしません。**BACKUP** または **RESTORE** の操作で使用したテープの開始ポイントが正しいことを必ず確認してから、テープをデバイスに挿入してください。ただし、リワインディング・デバイスでテープを使用している場合は、Sybase IQ がリワインドを実行します。
- **BACKUP** や **RESTORE** の操作中にアーカイブ・デバイスを開くことができず (メディアのロードが必要な場合など)、**ATTENDED** パラメータが ON の場合、Sybase IQ は 10 秒間待機してから再試行します。 デバイスが正常に挿入されるか、[Ctrl] キーを押しながら [C] キーを押して操作を終了しないかぎり、無制限に再試行されます。
- [Ctrl] キーを押しながら [C] キーを押すと、 **BACKUP** は失敗し、データベースはバックアップ開始前の状態に戻ります。
- RAID デバイスなどでディスク・ストライピングを使用している場合は、ストライプしたディスクは単一のデバイスとして処理されます。
- SQL Anywhere データベースをリカバリする場合のその他のオプションについては、『SQL Anywhere サーバー – データベース管理』の「データベースの保守」>「バックアップとデータリカバリ」を参照してください。

『システム管理ガイド：第 1 巻』の「データのバックアップ、リカバリ、アーカイブ」も参照してください。

関連する動作：

- オートコミット。

標準

- SQL—ISO/ANSI SQL 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。

パーミッション

データベースの所有者であるか、DBA 権限を持っている必要があります。DBA 権限を持たないユーザには、OPERATOR 権限が必要です。

参照：

- RESTORE 文 (321 ページ)

BEGIN ... END 文

SQL 文をグループ化します。

構文

```
[ statement-label : ]
… BEGIN [ [ NOT ] ATOMIC ]
… [ local-declaration ; … ]
… statement-list
… [ EXCEPTION [ exception-case … ] ]
… END [ statement-label ]
```

パラメータ

- **local-declaration：** – { *variable-declaration* | *cursor-declaration* | *exception-declaration* | *temporary-table-declaration* }
- **variable-declaration：** – **DECLARE** *variable-name* [, …] *data-type* [{ **=** | **DEFAULT**} *initial-value*]
- **initial-value：** – *special-value* | *string* | [-] *number* | (*constant-expression*) | *built-in-function* (*constant-expression*) | **NULL**
- **special-value：** – **CURRENT** { **DATABASE** | **DATE** | **PUBLISHER** | **TIME** | **TIMESTAMP** | **USER** | **UTC TIMESTAMP** } | **USER**

例

- **例 1 –** プロシージャの本文は複合文です。

```
CREATE PROCEDURE TopCustomer (OUT TopCompany CHAR(35), OUT
TopValue INT)
BEGIN
  DECLARE err_notfound EXCEPTION FOR
    SQLSTATE '02000' ;
  DECLARE curThisCust CURSOR FOR
    SELECT CompanyName, CAST(
        sum(SalesOrderItems.Quantity *
        Products.UnitPrice) AS INTEGER) VALUE
    FROM Customers
        LEFT OUTER JOIN Salesorders
        LEFT OUTER JOIN SalesOrderItems
        LEFT OUTER JOIN Products
    GROUP BY CompanyName ;
  DECLARE ThisValue INT ;
  DECLARE ThisCompany CHAR(35) ;
  SET TopValue = 0 ;
  OPEN curThisCust ;
```

```
 CustomerLoop:
 LOOP
   FETCH NEXT curThisCust
     INTO ThisCompany, ThisValue ;
   IF SQLSTATE = err_notfound THEN
     LEAVE CustomerLoop ;
   END IF ;
   IF ThisValue > TopValue THEN
     SET TopValue = ThisValue ;
     SET TopCompany = ThisCompany ;
   END IF ;
 END LOOP CustomerLoop ;


CLOSE curThisCust ;
END
```

使用法

プロシージャまたはトリガの本文は複合文です。複合文は、プロシージャまたはトリガ内の制御文の中でも使用できます。

複合文によって 1 つまたは複数の SQL 文をグループ化して、1 つの単位として処理できます。 複合文は **BEGIN** で始まり、**END** で終わります。**BEGIN** のすぐ後で、複合文は複合文の中だけで有効なローカル宣言を行うことができます。 複合文には、変数、カーソル、テンポラリ・テーブル、または例外に対するローカル宣言を含めることができます。 ローカル宣言は、その複合文またはその中にネストされた複合文の中にあるどの文からでも参照できます。 ローカル宣言は、複合文の中から呼び出される他のプロシージャには作用しません。

終了の *statement-label* を指定する場合は、開始の *statement-label* と一致させる必要があります。**LEAVE** 文を使用すると、複合文の後にある最初の文から実行を再開できます。プロシージャの本文である複合文は、プロシージャまたはトリガの名前と同じ暗黙のラベルを持っています。

「アトミック」な文とは、完全に実行されたか、まったく実行されなかったかのいずれかの文です。 たとえば、何千ものローを更新する **UPDATE** 文では、たくさんのローを更新した後にエラーが発生することがあります。 文が完了しない場合は、すべての変更が元の状態に戻ります。 同様に、**BEGIN** 文をアトミックに指定すると、文は完全に実行されるか、またはまったく実行されません。

initial-value を指定すると、変数はその値に設定されます。**initial-value** を指定しないと、**SET** 文によって異なる値が割り当てられるまで、変数には NULL 値が含まれます。

initial-value を指定する場合、そのデータ型は、*data-type* によって定義された型に一致する必要があります。

複合文と例外処理の詳細については、『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。

標準

- SQL—ISO/ANSI SQL 文法のベンダ拡張。
- Sybase—Adaptive Server Enterprise によってサポートされています。複合文中のすべての文がサポートされているわけではありません。
  BEGIN キーワードと **END** キーワードは、Transact-SQL では必要ありません。
  BEGIN と **END** を Transact-SQL で使用して、文のグループを 1 つの複合文にまとめることができます。**IF ... ELSE** のような制御文は単一の SQL 文の動作だけに影響しますが、複合文にまとめれば、グループ全体の動作を制御できます。
  ATOMIC キーワードは、Adaptive Server Enterprise でサポートされていません。
  Transact-SQL では、**DECLARE** 文は **BEGIN** のすぐ後に続く必要はなく、宣言したカーソルまたは変数は複合文の間だけ存在します。互換性を保つために、複合文の冒頭で変数を宣言してください。

パーミッション

なし

参照：

- DECLARE LOCAL TEMPORARY TABLE 文 (199 ページ)
- DECLARE CURSOR 文 [ESQL] [SP] (191 ページ)
- LEAVE 文 (271 ページ)
- RESIGNAL 文 (320 ページ)
- SIGNAL 文 (361 ページ)

BEGIN PARALLEL IQ ... END PARALLEL IQ 文

CREATE INDEX 文をグループ化して、同時に実行できるようにします。

構文

```
... BEGIN PARALLEL IQ

            statement-list
... END PARALLEL IQ
```

パラメータ

- **statement-list – CREATE INDEX** 文のリスト

例

- **例 1** – 次の文はアトミックに実行されます。1 つのコマンドが失敗すると、文全体がロールバックします。

```
BEGIN PARALLEL IQ
    CREATE HG INDEX c1_HG on table1 (col1);
    CREATE HNG INDEX c12_HNG on table1 (col12);
    CREATE LF INDEX c1_LF on table1 (col1);
    CREATE HNG INDEX c2_HNG on table1 (col2);
END PARALLEL IQ
```

使用法

BEGIN PARALLEL IQ ... END PARALLEL IQ 文により、単一の DDL 文であるかのように **CREATE INDEX** 文のグループを実行し、複数の IQ テーブルのインデックスを同時に作成できます。この文の実行中、ユーザは他の DDL 文を発行できません。

文のリスト内に複数のテーブルを指定できます。細分化レベルはカラム・レベルです。すなわち、同一カラムの複数のインデックスが逐次的に実行されます。

注意： この文は **TEXT** インデックスをサポートしません。

関連する動作：

- オートコミット。

標準

- 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。この文に含まれる文のサポートについては、「CREATE INDEX 文」を参照してください。

パーミッション

なし

参照：

- CREATE INDEX 文 (118 ページ)

BEGIN TRANSACTION 文 [T-SQL]

この文を使用して、ユーザ定義のトランザクションを開始します。

注意： **BEGIN TRANSACTION** は T-SQL 構成体であるため、有効な T-SQL コマンドのみを含める必要があります。T-SQL コマンドと T-SQL 以外のコマンドを組み合わせることはできません。

構文

BEGIN TRAN[**SACTION**] [*transaction-name*]

例

- **例 1 –** 次のバッチは、**@@trancount** の連続値を 0、1、2、1、0 でレポートします。値はサーバ・ウィンドウに出力されます。

```
PRINT @@trancount
BEGIN TRANSACTION
PRINT @@trancount
BEGIN TRANSACTION
PRINT @@trancount
COMMIT TRANSACTION
PRINT @@trancount
COMMIT TRANSACTION
PRINT @@trancount
```

@@trancount の値は、発行された明示的な **BEGIN TRANSACTION** 文を数える以上の目的には使用しないでください。

Adaptive Server Enterprise が暗黙的にトランザクションを開始すると、@@trancount 変数は 1 に設定されます。Sybase IQ の場合、トランザクションが暗黙的に開始されても、@@trancount 値は 1 に設定されません。 そのため、(現在のトランザクションがあるときでも) **BEGIN TRANSACTION** 文が始まる前は、Sybase IQ の @@trancount 変数の値は 0 で、Adaptive Server Enterprise (連鎖モード) の場合の値は 1 になります。

BEGIN TRANSACTION 文で始まるトランザクションの場合、最初の **BEGIN TRANSACTION** 文の後の @@trancount の値は、Sybase IQ と Adaptive Server Enterprise の両方で 1 になります。トランザクションが別の文で暗黙的に開始し、その後、**BEGIN TRANSACTION** 文が実行された場合、**BEGIN TRANSACTION** 文の後、@@trancount の値は Sybase IQ と Adaptive Server Enterprise の両方で 2 になります。

使用法

オプションのパラメータ *transaction-name* はこのトランザクションに割り当てられた名前です。これは有効な識別子である必要があります。トランザクション名は、ネストされた **BEGIN/COMMIT** 文または **BEGIN/ROLLBACK** 文の最も外側のペアでのみ使用してください。

BEGIN TRANSACTION 文をトランザクション内で実行すると、トランザクションのネスト・レベルが 1 つ増加します。ネスト・レベルは **COMMIT** 文で減少します。トランザクションがネストされているときは、最も外側の **COMMIT** だけがデータベースへの変更を確定します。

Adaptive Server Enterprise と Sybase IQ には、それぞれ 2 つのトランザクション・モードがあります。

デフォルトの Adaptive Server Enterprise トランザクション・モードは非連鎖モードと呼ばれ、明示的な **BEGIN TRANSACTION** 文が実行されてトランザクションを起動しないかぎり、各文を個々にコミットします。 反対に、ISO SQL/2003 互換の連鎖モードは、明示的 **COMMIT** が実行されるときか、オートコミットする文 (データ定義文など) が実行されるときにだけトランザクションをコミットします。

BEGIN TRANSACTION 文 [T-SQL] の詳細については、『SQL Anywhere サーバー - SQL リファレンス』の「SQL 文」>「SQL 文」>「BEGIN TRANSACTION 文 [T-SQL]」を参照してください。

注意： このリファレンスは SQL Anywhere マニュアルにリンクされています。

chained データベース・オプションを設定してモードを制御できます。Sybase IQ の ODBC と Embedded SQL 接続のデフォルト設定は ON で、この場合、Sybase IQ は連鎖モードで動作します (ODBC ユーザは AutoCommit ODBC の設定もチェックする必要があります)。 TDS 接続のデフォルトは OFF です。

非連鎖モードでは、データ検索文か修正文の前にトランザクションが暗黙のうちに開始されます。これらの文には、次のようなものがあります。**DELETE**、**INSERT**、**OPEN**、**FETCH**、**SELECT**、**UPDATE**。トランザクションの終了は、**COMMIT** 文または **ROLLBACK** 文を使って明示的に行います。

トランザクション内で chained オプションを変更することはできません。

注意： ストアド・プロシージャを呼び出すときは、目的のトランザクション・モードで正しく動作することを確認してください。

現在のネスト・レベルはグローバル変数 @@trancount に入っています。@@trancount 変数の値は、最初の **BEGIN TRANSACTION** 文が実行される前は 0 であり、@@trancount の値が 1 のときに実行された **COMMIT** だけがデータベースへの変更を確定できます。

トランザクションまたはセーブポイント名のない **ROLLBACK** 文は、常に文を最も外側の **BEGIN TRANSACTION** (明示的または暗黙的) 文にロールバックし、トランザクション全体をキャンセルします。

標準

- SQL—ISO/ANSI SQL 文法のベンダ拡張。
- Sybase—Adaptive Server Enterprise でサポートされています。

パーミッション

なし

参照：

- COMMIT 文 (75 ページ)
- ROLLBACK TRANSACTION 文 [T-SQL] (336 ページ)
- SAVE TRANSACTION 文 [T-SQL] (338 ページ)
- ISOLATION_LEVEL オプション (467 ページ)

CALL 文

プロシージャを起動します。

構文

構文 1

```
[ variable = ] CALL procedure-name ( [ expression ] [ , … ] )
```

構文 2

```
[ variable = ] CALL procedure-name ( [ parameter-name = expression ]
[ , … ] )
```

例

- **例 1 –** `sp_customer_list` プロシージャを呼び出します。このプロシージャはパラメータがなく、結果セットを返します。

  ```
  CALL sp_customer_list()
  ```

- **例 2 –** 次の **dbisql** の例は、指定した ID を持つ顧客からの注文の数を返すプロシージャを作成し、結果を保持する変数を作成し、プロシージャを呼び出し、結果を表示します。

  ```
  CREATE PROCEDURE OrderCount (IN CustomerID INT, OUT Orders INT)
  BEGIN
  ```

```
SELECT COUNT("DBA".SalesOrders.ID)
INTO Orders
FROM "DBA".Customers
KEY LEFT OUTER JOIN "DBA".SalesOrders
WHERE "DBA".Customers.ID = CustomerID ;
END
go
-- Create a variable to hold the result
CREATE VARIABLE Orders INT
go

-- Call the procedure, FOR customer 101
-- -----------------------------
CALL OrderCount ( 101, Orders)
go
--------------------------------
--  Display the result
SELECT Orders FROM DUMMY
go
```

使用法

CALL は、あらかじめ **CREATE PROCEDURE** 文を使用して作成されたプロシージャを呼び出します。プロシージャが完了すると、INOUT または OUT パラメータ値がコピーし直されます。

引数リストは、引数の位置によって、またはキーワード・フォーマットを使用して指定できます。位置で指定する場合、引数はプロシージャのパラメータ・リスト内の対応するパラメータと一致します。キーワードでは、引数は指定したパラメータと一致します。

プロシージャ引数には、**CREATE PROCEDURE** 文のデフォルト値が割り当てられます。パラメータが見つからない場合は、デフォルト値が割り当てられます。また、デフォルトが設定されていない場合、パラメータが NULL に設定されます。

プロシージャの内部では、プロシージャが結果セットを返すときに **DECLARE** 文で **CALL** 文を使用できます。

注意： **CALL** SQL 文では、テーブル UDF は参照できません。

プロシージャは **RETURN** 文を使って整数値 (ステータス・インジケータとして) を返すことができます。この戻り値を変数に保存するには、代入演算子として等号を使います。

```
CREATE VARIABLE returnval INT ;
returnval = CALL proc_integer ( arg1 = val1, ... )
```

注意： この文を使用して関数を呼び出すことはできなくなりました。関数を呼び出すには、代入文を使用して関数を呼び出し、その結果を変数に割り当てます。例を示します。

```
DECLARE varname INT;
SET varname=test( );
```

標準

- 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。サポートされている代替機能については、「EXECUTE 文 [ESQL]」を参照してください。

パーミッション

プロシージャの所有者であるか、プロシージャの EXECUTE パーミッションまたは DBA 権限を持っている必要があります。

参照：

- CREATE PROCEDURE 文 (137 ページ)
- EXECUTE 文 [ESQL] (223 ページ)
- GRANT 文 (247 ページ)

CASE 文

複数のケースに基づいて実行パスを選択します。

構文

```
CASE value-expression
…WHEN [ constant | NULL ] THEN statement-list …
… [ WHEN [ constant | NULL ] THEN statement-list ] …
…ELSE statement-list
… END
```

例

- **例 1 – CASE** 文を使用する次のプロシージャは、デモ・データベースの Products テーブルにリストされている製品を、シャツ、帽子、ショート・パンツ、不明のいずれかに分類します。

```
CREATE PROCEDURE ProductType (IN product_id INT, OUT type
CHAR(10))
  BEGIN
  DECLARE prod_name CHAR(20) ;
  SELECT name INTO prod_name FROM "GROUPO"."Products"
  WHERE ID = product_id;
  CASE prod_name
  WHEN 'Tee Shirt' THEN
    SET type = 'Shirt'
```

```
  WHEN 'Sweatshirt' THEN
    SET type = 'Shirt'
  WHEN 'Baseball Cap' THEN
    SET type = 'Hat'
  WHEN 'Visor' THEN
    SET type = 'Hat'
  WHEN 'Shorts' THEN
    SET type = 'Shorts'
  ELSE
    SET type = 'UNKNOWN'
  END CASE ;
  END
```

使用法

CASE 文は制御文であり、これを使用して SQL 文のリストから式の値に対応する文を選択して実行できます。

WHEN 句が *value-expression* の値に対して存在する場合、**WHEN** 句の中の *statement-list* が実行されます。適切な **WHEN** 句が存在せず、**ELSE** 句が存在する場合、**ELSE** 句の中の *statement-list* が実行されます。**END** の後に記述されている最初の文から実行が再開されます。

注意： ANSI 標準には、使用可能な **CASE** 文の形式が 2 つあります。Sybase IQ も両方の形式に対応していますが、**CASE** を述部内で使用する場合は、パフォーマンスを考慮してここに示されている形式を使用してください。

SQL Anywhere との互換性を保つために、もう一方の形式 (これも ANSI 構文です) を使用する必要がある場合は、『SQL Anywhere サーバー - SQL リファレンス』>「SQL 文」>「SQL 文」>「CASE 文」の **CASE** 文の構文 2 を参照してください。

注意： **CASE** 文の構文と CASE 式の構文を混同しないでください。

CASE 式の詳細については、『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 言語の要素」>「式」を参照してください。

標準

- 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。

パーミッション

なし

参照：

- BEGIN … END 文 (60 ページ)

CHECKPOINT 文

データベースのチェックポイントを実行します。

構文

```
CHECKPOINT
```

使用法

CHECKPOINT 文を記述すると、データベース・サーバでチェックポイントが実行されます。チェックポイントは、データベース・サーバが内部アルゴリズムに従って自動実行もできます。アプリケーションは通常、**CHECKPOINT** を発行する必要がありません。チェックポイントの詳細については、『システム管理ガイド：第 1 巻』の「データのバックアップ、リカバリ、アーカイブ」を参照してください。

標準

- SQL—ISO/ANSI SQL 文法のベンダ拡張。
- Sybase—Adaptive Server Enterprise でサポートされています。

パーミッション

データベースのチェックポイントを実行するには、DBA 権限または OPERATOR 権限が必要です。

CLEAR 文 [Interactive SQL]

Interactive SQL (**dbisql**) の開いている結果セットをすべて閉じます。

構文

```
CLEAR
```

使用法

開いている結果セットをすべて閉じ、[SQL 文] ウィンドウ枠の内容は変更せずに保持します。

関連する動作：

CLEAR 文は、クリアしたデータに関連付けられているカーソルを閉じます。

標準

- SQL—ISO/ANSI SQL 文法のベンダ拡張。
- Sybase—なし。

パーミッション

なし

参照：

- EXIT 文 [Interactive SQL] (229 ページ)

CLOSE 文 [ESQL] [SP]

カーソルをクローズします。

構文

CLOSE *cursor-name*

パラメータ

- **cursor-name：** – { *identifier* | *host-variable* }

例

- **例 1 –** 次の例は Embedded SQL のカーソルをクローズします。

```
EXEC SQL CLOSE employee_cursor;
EXEC SQL CLOSE :cursor_var;
```

- **例 2 –** カーソルを使用します。

```
CREATE PROCEDURE TopCustomer (OUT TopCompany CHAR(35), OUT
TopValue INT)
BEGIN
  DECLARE err_notfound EXCEPTION
  FOR SQLSTATE '02000' ;
  DECLARE curThisCust CURSOR FOR
    SELECT CompanyName,
      CAST(       sum(SalesOrderItems.Quantity *
      Products.UnitPrice) AS INTEGER) VALUE
    FROM Customers
    LEFT OUTER JOIN SalesOrders
    LEFT OUTER JOIN SalesOrderItems
    LEFT OUTER JOIN Products
  GROUP BY CompanyName ;
```

```
  DECLARE ThisValue INT ;
  DECLARE ThisCompany CHAR(35) ;
  SET TopValue = 0 ;
  OPEN curThisCust ;
  CustomerLoop:
  LOOP
    FETCH NEXT curThisCust
    INTO ThisCompany, ThisValue ;
    IF SQLSTATE = err_notfound THEN
      LEAVE CustomerLoop ;
    END IF ;
    IF ThisValue > TopValue THEN
      SET TopValue = ThisValue ;
      SET TopCompany = ThisCompany ;
      END IF ;
   END LOOP CustomerLoop ;
   CLOSE curThisCust ;
END
```

使用法

この文は指定したカーソルをクローズします。

標準

- SQL—ISO/ANSI SQL 文法のベンダ拡張。
- Sybase—Adaptive Server Enterprise でサポートされています。

パーミッション

カーソルを事前にオープンしておく必要があります。

参照：

- DECLARE CURSOR 文 [ESQL] [SP] (191 ページ)
- OPEN 文 [ESQL] [SP] (301 ページ)
- PREPARE 文 [ESQL] (309 ページ)

COMMENT 文

データベース・オブジェクトに関するコメントをシステム・テーブルに保存します。

構文

```
COMMENT ON
{ COLUMN [ owner.]table-name.column-name
| DBSPACE dbspace-name
| EVENT event-name
```

```
| EXTERNAL ENVIRONMENT environment-name
| EXTERNAL OBJECT object-name
| FOREIGN KEY [owner.]table-name.role-name
| INDEX [ [owner.]table.]index-name
| INTEGRATED LOGIN integrated-login-id
| JAVA CLASS java-class-name
| JAVA JAR java-jar-name
| KERBEROS LOGIN "client-Kerberos-principal"
| LOGIN POLICY policy-name
| MATERIALIZED VIEW [owner.]materialized-view-name
| PROCEDURE [owner.]table-name
| SERVICE web-service-name
| TABLE [ owner.]table-name
| TRIGGER [[ owner.]table-name.]trigger-name
| USER userid
| VIEW [ owner.]view-name }
IS comment
```

パラメータ

- **comment：** – { *string* | **NULL** }
- **environment-name：** – **JAVA | PERL | PHP | CLR | C_ESQL32 | C_ESQL64 | C_ODBC32 | C_ODBC64**

例

- **例 1** – Employees テーブルにコメントを追加します。

```
COMMENT
ON TABLE Employees
IS "Employee information"
```

- **例 2** – Employees テーブルからコメントを削除します。

```
COMMENT
ON TABLE Employees
IS NULL
```

使用法

COMMENT 文を使用すると、データベース内のオブジェクトにコメントを設定できます。**COMMENT** 文は、ISYSREMARK システム・テーブルにある注釈を更新します。コメントは、NULL に設定すると削除できます。インデックスまたはトリガに関するコメントの所有者は、インデックスまたはトリガが定義されているテーブルの所有者になります。

COMMENT ON DBSPACE、**COMMENT ON JAVA JAR**、**COMMENT ON JAVA CLASS** 文を使用すると、SYS.ISYSREMARK システム・テーブルに Remarks カラムを設定できます。コメントは、NULL に設定して削除します。

ローカル・テンポラリ・テーブルに対するコメントは追加できません。

注意： 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。

標準

- 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。

パーミッション

コメントされるデータベース・オブジェクトの所有者であるか、または DBA 権限を持っている必要があります (**DBSPACE** 句を使用してこの文を発行するには、DBA 権限または SPACE ADMIN 権限を持っている必要があります)。

COMMENT ON LOGICAL SERVER 文

ユーザ定義の論理サーバにコメントを追加します。

構文

```
COMMENT ON LOGICAL SERVER logical-server-name IS 'comment'
```

例

- **例 –** 次の例は、ユーザ定義の論理サーバ *ls1* についてのコメントを作成します。

```
COMMENT ON LOGICAL SERVER ls1 IS ‘ls1: Primary
Logical Server’;
```

使用法

マルチプレックスのみに該当します。

パーミッション

DBA 権限または MPX ADMIN 権限を持っている必要があります。

COMMIT 文

データベースへの変更を保存します。またはユーザ定義のトランザクションを終了します。

構文

構文 1

```
COMMIT [ WORK ]
```

構文 2

```
COMMIT TRAN[SACTION ] [ transaction-name ]
```

例

- **例 1 –** 現在のトランザクションをコミットします。

```
COMMIT
```

- **例 2 –** 次の Transact-SQL バッチは、`@@trancount` の連続した値を、0、1、2、1、0 としてレポートします。

```
PRINT @@trancount
BEGIN TRANSACTION
PRINT @@trancount
BEGIN TRANSACTION
PRINT @@trancount
COMMIT TRANSACTION
PRINT @@trancount
COMMIT TRANSACTION
PRINT @@trancount
go
```

使用法

構文 1—**COMMIT** 文はトランザクションを終了し、このトランザクション中に行ったすべての変更をデータベースに確定します。

データ定義文は、自動的にコミットします。詳細については、各 SQL 文の「関連する動作」を参照してください。

データベース・サーバが無効な外部キーを検知すると、**COMMIT** 文は失敗します。その場合、無効な外部キーがあることによってトランザクションを終了できなくなります。通常、外部キー整合性の検査はそれぞれのデータ操作オペレーションごとに行います。ただし、データベース・オプション `WAIT_FOR_COMMIT` が ON に設定されているか、または特定の外部キーが **CHECK ON COMMIT** 句によって定義されている場合、データベース・サーバは **COMMIT** 文が実行されるまで整合性のチェックを遅延します。

構文 2—**BEGIN TRANSACTION** 文と **COMMIT TRANSACTION** 文をペアで使用すると、ネストされたトランザクションを作成できます。ネストされたトランザクションはセーブポイントに似ています。ネストされたトランザクション・セットの最も外側で実行した文がデータベースへの変更を保存します。トランザクション内で実行した場合、**COMMIT TRANSACTION** 文はトランザクションのネスト・レベルを 1 つ減らします。トランザクションがネストされているときは、最も外側の **COMMIT** だけがデータベースへの変更を確定します。

オプションのパラメータ *transaction-name* はこのトランザクションに割り当てられた名前です。この名前は、有効な識別子でなければなりません。トランザクション名は、ネストされた **BEGIN/COMMIT** 文または **BEGIN/ROLLBACK** 文の最も外側のペアでのみ使用してください。

さまざまなオプションを使用して、**COMMIT** 文の動作を詳細に制御できます。「COOPERATIVE_COMMIT_TIMEOUT オプション」、「COOPERATIVE_COMMITS オプション」、「DELAYED_COMMITS オプション」、「DELAYED_COMMIT_TIMEOUT オプション」を参照してください。**Commit** 接続プロパティを使用して、現在の接続のコミット数を取得できます。

関連する動作：

- **WITH HOLD** を指定してオープンしたもの以外のすべてのカーソルをクローズします。
- **ON COMMIT PRESERVE ROWS** を指定して宣言した場合を除いて、この接続上にある、宣言したテンポラリ・テーブルのすべてのローを削除します。

標準

- SQL—ISO/ANSI SQL 準拠。
- Sybase—Adaptive Server Enterprise によってサポートされています。構文 2 は、ISO/ANSI SQL 文法の Transact-SQL 拡張です。

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。

参照：

- BEGIN TRANSACTION 文 [T-SQL] (64 ページ)
- CONNECT 文 [ESQL] [Interactive SQL] (77 ページ)
- DISCONNECT 文 [Interactive SQL] (208 ページ)
- ROLLBACK 文 (334 ページ)
- SAVEPOINT 文 (337 ページ)
- SET CONNECTION 文 [ESQL] [Interactive SQL] (354 ページ)

- COOPERATIVE_COMMIT_TIMEOUT オプション (426 ページ)
- COOPERATIVE_COMMITS オプション (427 ページ)
- DELAYED_COMMITS オプション (442 ページ)
- DELAYED_COMMIT_TIMEOUT オプション (442 ページ)

CONFIGURE 文 [Interactive SQL]

Interactive SQL (**dbisql**) 設定ウィンドウをアクティブにします。

構文

```
CONFIGURE
```

使用法

dbisql 設定ウィンドウには、すべての **dbisql** オプションの現在の設定が表示されます。データベース・オプションは表示されず、変更を加えることもできません。

[Permanent] を選択すると、オプションはデータベースの `SYSOPTION` テーブルに書き込まれ、データベース・サーバは自動的に **COMMIT** を実行します。
[Permanent] を選択しないで、[OK] をクリックすると、オプションは一時的に設定され、現在のデータベース接続の間だけに有効になります。

標準

- 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。

パーミッション

なし

参照：

- SET OPTION 文 (356 ページ)

CONNECT 文 [ESQL] [Interactive SQL]

データベースへの接続を確立します。

構文

構文 1

```
CONNECT
… [ TO engine-name ]
…[ DATABASE database-name ]
…[ AS connection-name ]
…[ USER ] userid [ IDENTIFIED BY ]
```

構文 2

```
CONNECT USING connect-string
```

パラメータ

- **engine-name：** – 識別子、文字列、またはホスト変数
- **database-name：** – 識別子、文字列、またはホスト変数
- **connection-name：** – 識別子、文字列、またはホスト変数
- **userid：** – 識別子、文字列、またはホスト変数
- **password：** – 識別子、文字列、またはホスト変数
- **connect-string：** – 有効な接続文字列、またはホスト変数

例

- **例 1** – 次は、Embedded SQL 内での **CONNECT** の使用例です。

  ```
  EXEC SQL CONNECT AS :conn_name
  USER :userid IDENTIFIED BY :password;
  EXEC SQL CONNECT USER "dba" IDENTIFIED BY "sql";
  ```

- **例 2** – 次は、**dbisql** 内での **CONNECT** の使用例です。

  - **dbisql** からデータベースに接続します。ユーザ ID とパスワードの入力を要求するプロンプトが表示されます。

    ```
    CONNECT
    ```

  - **dbisql** から DBA としてデフォルト・データベースに接続します。パスワードの入力が求められます。

    ```
    CONNECT USER "DBA"
    ```

  - **dbisql** から DBA としてデモ・データベースに接続します。

    ```
    CONNECT
    TO <machine>_iqdemo
    USER "DBA"
    IDENTIFIED BY sql
    ```

    ここで、*<machine>_iqdemo* はエンジン名です。

  - 接続文字列を使用して **dbisql** からデモ・データベースに接続します。

    ```
    CONNECT
    USING 'UID=DBA;PWD=sql;DBN=iqdemo'
    ```

使用法

CONNECT 文は、*engine-name* で識別されるサーバ上で実行している、*database-name* で識別されるデータベースへの接続を確立します。

Embedded SQL の動作 — Embedded SQL では、*engine-name* が指定されていない場合、デフォルトのローカル・データベース・サーバが使用されます (最初に起動するデータベース・サーバ)。 *database-name* を指定していない場合、指定したサーバ上の最初のデータベースが使用されます。

WHENEVER 文、**SET SQLCA** 文、一部の **DECLARE** 文はコードを生成しないので、ソース・ファイル内の **CONNECT** 文の前に置いてもかまいません。それ以外の場合は、**CONNECT** 文が正常に実行されるまで、どのような文も使用できません。

ユーザ ID とパスワードを使用して、動的 SQL 文ごとにパーミッションをチェックします。デフォルトでは、パスワードの大文字と小文字は区別されますが、ユーザ ID では区別しません。

接続アルゴリズムの詳細については、『システム管理ガイド：第 1 巻』の「Sybase IQ の接続」>「Sybase IQ の接続の確立方法」を参照してください。

DBISQL の動作—**CONNECT** 文でデータベースまたはサーバを指定しない場合、**dbisql** はデフォルトのサーバとデータベースには接続しないで、現在のデータベースとの接続を継続します。サーバ名を指定せずにデータベース名を指定した場合、**dbisql** は現在のサーバ上の指定したデータベースに接続しようとします。データベース・ファイル名ではなく、**-n** データベース・スイッチで定義されたデータベース名を指定する必要があります。データベース名を指定せずにサーバ名を指定した場合、**dbisql** は指定したサーバ上のデフォルト・データベースに接続します。たとえば、データベースへの接続中に次のバッチを実行すると、同じデータベースに 2 つのテーブルが作成されます。

```
CREATE TABLE t1( c1 int );
CONNECT DBA IDENTIFIED BY sql;
CREATE TABLE t2 (c1 int );
```

CONNECT 文が正常に実行されるまで、他のデータベース文を使用できません。

ユーザ ID とパスワードを使用して、SQL 文ごとにパーミッションをチェックします。パスワード、またはユーザ ID とパスワードが指定されていない場合、足りない情報を入力するようにプロンプトが表示されます。デフォルトでは、パスワードの大文字と小文字は区別されますが、ユーザ ID では区別しません。

複数の接続を管理するときにも、現在の接続という概念が使用されます。CONNECT 文が正常に実行されると、その新しい接続が現在の接続になります。別の接続に切り替えるには、**SET CONNECTION** を使用します。**CONNECT** 文を実行

しても、既存の接続は (仮にあったとしても) クローズされません。接続を削除するには、**DISCONNECT** を使用します。

静的 SQL 文は **SQLPP** 文の行に *-I* オプションで指定したユーザ ID とパスワードを使用します。*-I* オプションが指定されていない場合は、**CONNECT** 文のユーザ ID とパスワードを静的 SQL 文でも使用します。

パスワードが不要な接続—DBA 権限を使用してユーザ ID に接続している場合は、パスワードを指定しなくても別のユーザ ID に接続できます(**dbtran** の出力には、この機能が必要です)。たとえば、DBA として Interactive SQL からデータベースに接続する場合は、パスワードがなくても次の文を使用して接続できます。

```
CONNECT other_user_id
```

Embedded SQL では、パスワードの代わりにホスト変数を使い、ホスト変数の値を NULL ポインタに設定することにより、パスワードがなくても接続できます。

AS 句—オプションで **AS** 句を指定して、接続に名前を付けることができます。接続に名前を付けると、同じデータベースへの複数の接続、あるいは同じまたは異なるデータベース・サーバへの複数の接続が、すべて同時に行えるようになります。それぞれの接続には、固有のトランザクションがあります。ただし、2 つの異なる接続から同じデータベース内の同じレコードを修正しようした場合など、トランザクション間でロックの競合が起こることもあります。

構文 2—*connect-string* は **keyword**=*value* 形式のパラメータ設定リストであり、一重引用符で囲む必要があります。

標準

- SQL—ISO/ANSI SQL 文法のベンダ拡張。
- Sybase—Open Client Embedded SQL は **CONNECT** 文に対して異なる構文をサポートします。

パーミッション

なし

参照：

- DISCONNECT 文 [Interactive SQL] (208 ページ)
- GRANT 文 (247 ページ)
- SET CONNECTION 文 [ESQL] [Interactive SQL] (354 ページ)

CREATE DATABASE 文

複数のオペレーティング・システム・ファイルから構成されるデータベースを作成します。

構文

```
CREATE DATABASE db-name
… [ [ TRANSACTION ] { LOG ON [ log-file-name ]
      [ MIRROR mirror-file-name ] } ]
… [ CASE { RESPECT | IGNORE } ]
… [ PAGE SIZE page-size ]
… [ COLLATION collation-label[( collation-tailoring-string ) ] ]
… [ ENCRYPTED [ TABLE ] {algorithm-key-spec | OFF } ]
… { … [ BLANK PADDING ON ]
… [ JCONNECT { ON | OFF } ]
… [ IQ PATH iq-file-name ]
… [ IQ SIZE iq-file-size ]
… [ IQ PAGE SIZE iq-page-size ]
… [ BLOCK SIZE block-size ]
… [ IQ RESERVE sizeMB ]
… [ TEMPORARY RESERVE sizeMB ]
… [ MESSAGE PATH message-file-name ]
… [ TEMPORARY PATH temp-file-name ]
… [ TEMPORARY SIZE temp-db-size ]
… [ DBA USER userid ]
… [ DBA PASSWORD password ]
```

パラメータ

- **db-name | log-file-name | mirror-file-name | iq-file-name | message-file-name | temp-file-name：** – '*file-name*'
- **page-size：** – { **4096** | **8192** | **16384** | **32768** }
- **iq-page-size：** – { **65536** | **131072** | **262144** | **524288** }
- **block-size：** – { **4096** | **8192** | **16384** | **32768** }
- **collation-label：** – *文字列*
- **collation-tailoring-string：** – *keyword=value*
- **algorithm-key-spec：** – **ON** | [**ON**] **KEY** *key* [**ALGORITHM** *AES-algorithm*] | [**ON**] **ALGORITHM** *AES-algorithm* **KEY** *key* | [**ON**] **ALGORITHM** '**SIMPLE**'
- **AES-algorithm：** – '**AES**' | '**AES256**' | '**AES_FIPS**' | '**AES256_FIPS**'
- **key：** – *quoted string*

例

- **例 1 –** 次の Windows の例は、mydb という名前の Sybase IQ データベースを、関連する mydb.db、mydb.iq、mydb.iqtmp、mydb.iqmsg の各ファイルと共に C:¥s1¥data ディレクトリに作成します。

```
CREATE DATABASE 'C:¥¥s1¥¥data¥¥mydb'
BLANK PADDING ON
IQ PATH 'C:¥¥s1¥¥data'
IQ SIZE 2000
IQ PAGE SIZE 65536
```

- **例 2 –** 次の UNIX コマンドは、**IQ PATH** と **TEMPORARY PATH** 用のロー・デバイスで Sybase IQ データベースを作成します。 IQ のデフォルトのページ・サイズである 128KB が適用されています。

```
CREATE DATABASE '/s1/data/bigdb'
IQ PATH '/dev/md/rdsk/bigdb'
MESSAGE PATH '/s1/data/bigdb.iqmsg'
TEMPORARY PATH '/dev/md/rdsk/bigtmp'
```

- **例 3 –** 次の Windows コマンドは、**IQ PATH** 用のロー・デバイスで Sybase IQ データベースを作成します。 ロー・デバイス名には円記号を 2 つ続けて指定します (Windows の仕様)。

```
CREATE DATABASE 'company'
IQ PATH '¥¥¥¥.¥¥E:'
JCONNECT OFF
IQ SIZE 40
```

- **例 4 –** 次の UNIX の例では、AES 暗号化アルゴリズムとキー "is!seCret" を使用して、高度に暗号化された Sybase IQ データベースを作成しています。

```
CREATE DATABASE 'marvin.db'
BLANK PADDING ON
CASE RESPECT
COLLATION 'ISO_BINENG'
IQ PATH '/filesystem/marvin.main1'
IQ SIZE 6400
IQ PAGE SIZE 262144
TEMPORARY PATH '/filesystem/marvin.temp1'
TEMPORARY SIZE 3200
ENCRYPTED ON KEY 'is!seCret' ALGORITHM 'AES'
```

使用法

指定した名前と属性で IQ データベースを作成します。Sybase IQ データベースの作成には **IQ PATH** 句が必須です。この句が指定されていない場合は、標準の SQL Anywhere データベースが作成されます。

IQ PATH オプションを省略すると、次のどのオプションを指定してもエラーが生成されるようになります。**IQ SIZE**、**IQ PAGE SIZE**、**BLOCK SIZE**、**MESSAGE PATH**、**TEMPORARY PATH**、**TEMPORARY SIZE**。

Sybase IQ は IQ データベース作成時に、IQ データベースを構成する、異なる種類のデータを格納するための 4 つのデータベース・ファイルを自動生成します。 各ファイルは dbspace に対応し、その論理名を使って Sybase IQ はデータベース・ファイルを識別します。 各ファイルを次に示します。

- *db-name.db* は、カタログ DB 領域 SYSTEM を保持するファイルです。データベースや追加したすべての標準 SQL Anywhere データベース・オブジェクトを記述するシステム・テーブルやストアド・プロシージャが格納されます。.db 拡張子を付けずに指定すると、Sybase IQ で自動的に追加されます。 この最初の DB 領域には、カタログ・ストアが格納されており、後で、DB 領域を追加してカタログ・ストアのサイズを大きくすることが可能です。 ロー・パーティション上では作成できません。
- *db-name.iq* は、IQ テーブルとインデックスを格納するメイン・データ DB 領域 IQ_SYSTEM_MAIN を保持するファイルのデフォルト名です。**IQ PATH** 句を使用して異なるファイル名を指定できます。この最初の DB 領域には、IQ ストアが格納されています。

  警告！ IQ_SYSTEM_MAIN は、 IQ db_identity ブロック、IQ チェックポイント・ログ、コミットされた各トランザクションとチェックポイントが設定されたアクティブな各トランザクションの IQ ロールフォワード／ロールバック・ビットマップ、インクリメンタル・バックアップ・ビットマップ、フリーリスト・ルート・ページといった、データベースがオープンするのに必要なすべての構造体を含んでいる特別な DB 領域です。IQ_SYSTEM_MAIN は、データベースがオープンしているときは常にオンラインになっています。

  管理者は、特に、これらのテーブルが小さく、重要なテーブルである場合に、IQ_SYSTEM_MAIN にユーザ・テーブルを作成することを許可できます。ただし、データベースを作成したらすぐに、管理者が 2 つ目のメイン DB 領域を作成し、DB 領域 IQ_SYSTEM_MAIN における CREATE 権限をすべてのユーザから取り消し、新しいメイン DB 領域に対する CREATE 権限を選択したユーザに付与して、PUBLIC.default_dbspace を新しいメイン DB 領域に設定する方法の方がより一般的です。

- *db-name.iqtmp* は、イニシャル・テンポラリ DB 領域 IQ_SYSTEM_TEMP を保持するファイルのデフォルト名です。特定のクエリによって生成したテンポラリ・テーブルを格納します。このファイルの必要サイズはクエリのタイプとデータ量に応じて異なります。**TEMPORARY PATH** 句を使用して異なるファイル名を指定できます。この最初の DB 領域には、テンポラリ・ストアが格納されています。
- *db-name.iqmsg* は、メッセージ・トレース DB 領域 IQ_SYSTEM_MSG を格納するファイルのデフォルト名です。**MESSAGE PATH** 句を使用して異なるファイル名を指定できます。

これらのファイルのほかに、IQ データベースにはトランザクション・ログ・ファイル (db-name.log) が含まれます。また、トランザクション・ログ・ミラー・ファイルを含む場合もあります。

ファイル名と **CREATE DATABASE** 文：

ファイル名 (*db-name*、*log-file-name*、*mirror-file-name*、*iq-file-name*、*message-file-name*、*temp-file-name*) は、オペレーティング・システム・ファイル名を含む文字列です。リテラル文字列として、一重引用符で囲んでください。

- Windows でパスを指定する場合、円記号 (¥) の後に n または x がある場合は円記号を 2 つ重ねます。こうすることで、SQL の文字列の規則に従って、改行文字 (¥n) または 16 進数字 (¥x) として解釈されるのを回避できます。円記号は常にエスケープした方が安全です。次に例を示します。

```
CREATE DATABASE 'c:¥¥sybase¥¥mydb.db'
LOG ON 'e:¥¥logdrive¥¥mydb.log'
JCONNECT OFF
IQ PATH 'c:¥¥sybase¥¥mydb'
IQ SIZE 40
```

- パスを指定しない場合、または相対パスを指定する場合は、以下のことに注意してください。
  - カタログ・ストア・ファイル (*db-name.db*) は、サーバの作業ディレクトリを基準に作成されます。
  - IQ ストア、テンポラリ・ストア、メッセージ・ログの各ファイルは、カタログ・ストアと同じディレクトリ内に作成されるか、カタログ・ストアを基準とする相対位置に作成されます。

相対パス名の使用をおすすめします。

警告！ データベース・ファイル、テンポラリ DB 領域ファイル、トランザクション・ログ・ファイルは、データベース・サーバと同じ物理マシンに配置する必要があります。データベース・ファイルとトランザクション・ログ・ファイルをネットワーク・ドライブに置かないでください。ただし、トランザクション・ログは、そのミラー・ファイルとは別のデバイスに置く必要があります。

UNIX システムでは、指定するファイルのパス名を含む間接的なポインタであるシンボリック・リンクを作成できます。 シンボリック・リンクを相対パス名として使用できます。データベース・ファイル名のシンボリック・リンクを作成することには、以下の利点があります。

- 実際のデバイス名は一般に意味を持たないが、ロー・デバイスへのシンボリック・リンクには、意味のある名前を付けることができる。
- シンボリック名を使用すると、バックアップ以降に別のディレクトリへ移動されたデータベース・ファイルをリストアする際の問題を回避できる。

シンボリック・リンクを作成するには、**ln -s** コマンドを使用します。次に例を示します。

```
ln -s /disk1/company/iqdata/company.iq company_iq_store
```

このリンクを作成しておけば、完全修飾パス名を使用する代わりに **CREATE DATABASE** または **RESTORE** などのコマンドでシンボリック・リンクを指定できます。

データベースや DB 領域を作成する際、パスはすべての DB 領域ファイルでユニークである必要があります。**CREATE DATABASE** コマンドに指定したパスとファイル名が、これらの 2 つのストアで一致する場合は、エラーが返されます。

注意： マルチプレックス・データベースを作成する方法については、『Sybase IQ Multiplex の使用』を参照してください。

次のいずれかの方法で、ユニークなパスを作成できます。

- ファイルごとに異なる拡張子を指定する (`mydb.iq` と `mydb.iqtmp` など)。
- 異なるファイル名を指定する (`mydb.iq` と `mytmp.iq` など)。
- 異なるパス名 (`/iqfiles/main/iq` と `/iqfiles/temp/iq` など) または異なるロー・パーティションを指定する。
- データベースの作成時に、**TEMPORARY PATH** を省略する。この場合、テンポラリ・ストアは、デフォルトの名前と拡張子 `dbname.iqtmp` でカタログ・ストアと同じパスに作成されます。*dbname* は、データベース名です。

警告！ UNIX プラットフォームでは、データベースの一貫性を保つためには別のファイルにリンクするファイル名を指定する必要があります。Sybase IQ は、リンクされたファイルが指す位置を検出できません。 コマンド内でファイル名が異なっていても、同じオペレーティング・システム・ファイルをポイントしていないか確認してください。

CREATE DATABASE の句とオプション：

TRANSACTION LOG—トランザクション・ログは、データベース・サーバがデータベースに対して行われた全変更のログを取るファイルです。トランザクション・ログはシステム・リカバリで重要な役割を果たします。**TRANSACTION LOG** 句を指定しない場合、またはファイルのパスを省略した場合、`.db` ファイルと同じディレクトリに格納されます。ただし、`.db` と `.iq` とは異なる物理デバイス上に格納する必要があります。ロー・パーティション上では作成できません。

MIRROR—トランザクション・ログ・ミラーは、トランザクション・ログの同一コピーであり、通常、ユーザ・データの保護強化のために異なるデバイス上で管理されます。 デフォルトでは、Sybase IQ はミラーリングされたトランザクション・ログを使用しません。 トランザクション・ログ・ミラーを使用する場合、ファイル名を指定する必要があります。 相対パスを使用すると、トランザクショ

ン・ログ・ミラーはカタログ・ストア (db-name.db) のディレクトリに相対して作成されます。 トランザクション・ログのミラー・コピーを常に作成することをおすすめします。

CASE—**CASE RESPECT** を使用して作成したデータベースでは、比較と文字列操作で、対象となるすべての値の大文字と小文字が区別されます。カラム、プロシージャ、ユーザ ID などのデータベース・オブジェクトの名前は対象外です。**CASE** の指定に関係なく、DB 領域名は常に大文字と小文字が区別されます。

デフォルト (RESPECT) では、すべての比較において大文字と小文字が区別されます。**CASE RESPECT** を使用すれば、**CASE IGNORE** よりもパフォーマンスが高くなります。

テーブルに挿入した文字列は、データベースが大文字と小文字を区別するかどうかに関係なく、常に入力された大文字と小文字がそのまま格納されます。文字列 **Value** は、文字データ型カラムに挿入される場合、常に大文字の V と残りの文字が小文字でデータベースに格納されます。**SELECT** 文は、**Value** として文字列を返します。ただし、データベースが大文字と小文字を区別しない場合は、すべての比較で **Value**、**value**、**VALUE** などは同じものとして扱われます。IQ サーバは大文字と小文字を任意に組み合わせた結果を返すので、大文字と小文字を区別しない (**CASE IGNORE**) データベースで、大文字と小文字を区別する結果は期待できません。

たとえば、次のテーブルとデータがあるとします。

```
CREATE TABLE tb (id int NOT NULL,
                 string VARCHAR(30) NOT NULL);
INSERT INTO tb VALUES (1, ‘ONE’);
SELECT * FROM tb WHERE string = ‘oNe’;
```

SELECT の結果は "oNe" (**WHERE** 句で指定) となることがあり、必ずしも "ONE" (データベースに格納される形) になるとはかぎりません。

同じように、次の結果は

```
SELECT * FROM tb WHERE string = ‘One’;
```

"One" になることがあり、次の結果は

```
SELECT * FROM tb WHERE string = ‘ONe’;
```

"ONe" になることがあります。

すべてのデータベースは、少なくとも 1 つの次のユーザ ID と、

```
DBA
```

パスワードを指定して作成されます。

```
sql
```

新しいデータベースでは、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。ユーザ ID は **CASE RESPECT** 設定に影響されません。

PAGE SIZE—カタログ・テーブルを含むデータベースの SQL Anywhere セグメントのページ・サイズは、4096、8192、16384、または 32768 バイトです。 通常、デフォルトの 4096 (4KB) を使用します。データベースが大きい場合は、このデフォルトよりも大きいページ・サイズが必要になることがあり、その結果、パフォーマンスが高くなる場合があります。 それより小さい値ではデータベースがサポートできるカラム数が制限される可能性があります。 4096 よりも小さいページ・サイズを指定すると、Sybase IQ によってページ・サイズは 4096 になります。

データベースの起動時に、ページ・サイズを現在のサーバのページ・サイズより大きくすることはできません。サーバのページ・サイズは、最初に起動されたデータベースによって決まるか、またはサーバのコマンド・ラインで **-gp** コマンド・ライン・オプションを指定して設定します。

すべての文で、コマンド・ラインの長さはカタログ・ページ・サイズに制限されます。デフォルトの 4KB で十分なことがほとんどですが、かなりの数の DB 領域を参照する **RESTORE** コマンドなど、非常に長いコマンドを含めるために、大きな **PAGE SIZE** 値が必要となるケースもまれにあります。多数のテーブルまたはビューを伴うクエリを実行するには、ページ・サイズを大きくする必要が生じる場合もあります。

デフォルトのカタログ・ページ・サイズは 4KB なので、ページ・サイズが 1024 である `utility_db` などのデータベースに接続する場合にのみ問題が発生します。この制限によって、多くの DB 領域を参照する **RESTORE** コマンドが失敗する可能性があります。この問題を回避するには、SQL コマンド・ラインの長さがカタログ・ページ・サイズ以上にならないようにします。

または、`-gp 32768` を指定してエンジンを開始して、カタログ・ページ・サイズを増加させます。

COLLATION—データベース内の文字データ型のソートと比較で使用される照合順。 この照合によって、使用しているコード化 (文字セット) の文字比較とソート順に関する情報が返されます。 **COLLATION** 句を指定していない場合、Sybase IQ はオペレーティング・システムの言語とコード化に基づいて照合を選択します。

ほとんどのオペレーティング・システムでは、デフォルトの照合順である ISO_BINENG を使用して、最適なパフォーマンスを発揮します。ISO_BINENG では、照合順序は ASCII 文字セットの文字順序と同じです。すべての大文字がすべての小文字に先行します (たとえば、'A' も 'B' も 'a' に先行します)。

照合は、サポートされる照合リストから選択できます。Sybase IQ サーバで作成された SQL Anywhere データベースでは、照合に UCA (Unicode Collation Algorithm) を

使用することもできます。 UCA が指定されている場合は、**ENCODING** 句も指定します。 **ENCODING** 句の詳細については、『SQL Anywhere サーバー - SQL リファレンス』の「SQL 文」>「SQL 文」>「CREATE DATABASE 文」を参照してください。

Sybase IQ では IQ データベースの UCA ベースの照合はサポートされていません。UCA ベースの照合が IQ データベースの **CREATE DATABASE** 文で指定されている場合は、"`UCA collation is not supported`" というエラーがサーバによって返され、データベースを作成できません。

照合は慎重に選択することが重要です。照合をデータベースの作成後に変更できません。照合の選択の詳細については、『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。

文字列のソートや比較を詳細に制御することを目的に、照合の適合化オプション (*collation-tailoring-string*) を指定することもできます。これらのオプションは、キーワード=値ペアの形式で、カッコで囲んで指定して、その後ろに照合名を記述します。

注意： 照合の適合化オプションの中には、Sybase IQ サーバで作成された SQL Anywhere データベース用の UCA 照合を指定する場合にサポートされているものもあります。 その他の照合と Sybase IQ の場合は、大文字と小文字の区別の適合化オプションのみがサポートされます。 また、照合の適合化オプションで作成したデータベースは、15.0 より前のデータベース・サーバでは起動できません。

「Sybase IQ で使用する照合の適合化オプション」には、Sybase IQ データベースの照合の適合化オプション (*collation-tailoring-string*) で指定できるキーワード、別の形式、値が含まれています。

表 3 : Sybase IQ で使用する照合の適合化オプション

キーワード	照合	別の形式	指定できる値
CaseSensitivity	サポートされているすべての照合	CaseSensitive、Case	• **respect** 大文字と小文字を区別します。UCA 照合では、UpperFirst に相当します。その他の照合では、respect の値は照合自体によって異なります。 • **ignore** 文字の大文字小文字の違いは無視されます。 • **UpperFirst** 常に、大文字を先にソートします (Aa)。 • **LowerFirst** 常に、小文字を先にソートします (aA)。

SQL Anywhere データベースの UCA 照合を指定する場合にサポートされている照合の適合化オプションの構文と完全なリストについては、『SQL Anywhere サー

バー - SQL リファレンス』の「SQL 文」>「SQL 文」>「CREATE DATABASE 文」を参照してください。

注意： このリファレンスは SQL Anywhere マニュアルにリンクされています。

ENCRYPTED—暗号化を使用すると、物理データベース・ファイルに格納されたデータを判読不能にすることができます。 データベース全体を暗号化するには、**CREATE DATABASEENCRYPTED** キーワード (**TABLE** キーワードなし) を使用します。 **ENCRYPTED TABLE** 句は、SQL Anywhere テーブルのテーブル暗号化のみを有効にする場合に使用します。 テーブル・レベルの暗号化は、Sybase IQ テーブルではサポートされていません。 「テーブルの暗号化を有効にする」とは、以降に **ENCRYPTED** 句を使用して作成されるテーブルや変更されるテーブルは、データベースの作成時に指定した設定を使用して暗号化されるということです。

データベースとテーブルの暗号化には 単純と強力という 2 つのレベルがあります。

- 単純な暗号化は、難読化 (obfuscation) と同レベルです。データは判読不能になりますが、暗号化の専門知識を持つ者であれば、データを解読できる可能性があります。単純暗号化の場合は、 **CREATE DATABASE** 句の **ENCRYPTED ON ALGORITHM 'SIMPLE'** または **ENCRYPTED ALGORITHM 'SIMPLE'** を指定するか、アルゴリズムやキーを指定せずに **ENCRYPTED ON** 句を指定します。
- 強力な暗号化は、128 ビットのアルゴリズムとセキュリティ・キーを使用して達成されます。データは判読不能で、キーがなければ事実上解読できません。強力な暗号化では、 **CREATE DATABASE** 句の **ENCRYPTED ON ALGORITHM** を使用して 128 ビットまたは 256 ビットの AES アルゴリズムを指定し、**KEY** 句を指定して暗号化キーを指定します。キーの値には、少なくとも 16 文字の長さを持ち、大文字と小文字を含み、数字、文字、特殊文字を使用したものを選ぶことをおすすめします。
  この暗号化キーは、データベースを起動するたびに入力する必要があります。

警告！ 暗号化キーをなくさないでください。キーは安全な場所に格納してください。キーを失うと、データベースにまったくアクセスできなくなります。そのうえ、修復する方法もありません。

暗号化は、データベースの作成時にのみ指定できます。暗号化を既存のデータベースに導入するには、完全にアンロードした後でデータベースを再作成し、すべてのデータを再ロードする必要があります。

アルゴリズムを指定せずに **ENCRYPTED** 句を使用すると、デフォルトの AES が使用されます。デフォルトでは、暗号化は OFF です。

BLANK PADDING—デフォルトでは、後続ブランクは比較の場合無視されます (`BLANK PADDING` ON)。Embedded SQL プログラムは文字配列にフェッチされた文字列を埋め込みます。このオプションは、ISO/ANSI SQL 規格との互換性を保つために用意されています。

たとえば、次の 2 つの文字列は、BLANK PADDING ON を指定して作成されたデータベースでは等価です。

```
'Smith'
'Smith   '
```

注意：CREATE DATABASE では、BLANK PADDING OFF はサポートされなくなりました。

JCONNECT—Sybase jConnect for JDBC ドライバを使用してシステム・カタログ情報にアクセスできるようにするには、jConnect サポートをインストールします。**JCONNECT** を **OFF** に設定すると、jConnect システム・オブジェクトが除外されます (デフォルトは **ON**)。その場合でも、システム情報にアクセスしないかぎり、JDBC を使用できます。

IQ PATH—Sybase IQ データが入っているメイン・セグメント・ファイルのパス名。 オペレーティング・システム・ファイル、または I/O デバイスのロー・パーティションを指定できます (各プラットフォーム用の『インストールおよび設定ガイド』にロー・パーティションを指定する場合のフォーマットについて記載されています)。Sybase IQ は指定したパス名に基づいたタイプを自動的に検出します。相対パスを使用すると、ファイルはカタログ・ストア (.db ファイル) のディレクトリに相対して作成されます。

IQ SIZE—**IQ PATH** 句で指定するロー・パーティションまたはオペレーティング・システム・ファイルのいずれかのサイズ (MB 単位)。 ロー・パーティションの場合、**IQ SIZE** を指定しないで、デフォルトを常に使用してください。デフォルトを使用することで、Sybase IQ はロー・パーティション全体を使用できます。**IQ SIZE** に値を指定する場合は、値を I/O デバイスのサイズに一致させてください。そうしないと、Sybase IQ はエラーを返します。 オペレーティング・システム・ファイルの場合は、次の表の最小値から、最大 4TB までの値を指定できます。

オペレーティング・システム・ファイルのデフォルト・サイズは **IQ PAGE SIZE** に応じて次のように異なります。

表 4 : IQ ストア・ファイルとテンポラリ・ストア・ファイルのデフォルトと最小サイズ

IQ PAGE SIZE	IQ SIZE のデフォルト	TEMPORARY SIZE のデフォルト	明示的な IQ SIZE の最小値	明示的な TEMPORARY SIZE の最小値
65536	4096000	2048000	4MB	2MB
131072	8192000	4096000	8MB	4MB
262144	16384000	8192000	16MB	8MB

IQ PAGE SIZE	IQ SIZE のデフォルト	TEMPORARY SIZE のデフォルト	明示的な IQ SIZE の最小値	明示的な TEMPORARY SIZE の最小値
524288	32768000	16384000	32MB	16MB

IQ PAGE SIZE—(IQ のテーブルとインデックスを含む) データベースの Sybase IQ セグメントのページ・サイズ (バイト単位)。 値は 65536 から 524288 バイトまでの 2 の累乗です。 デフォルトは 131072 (128KB) です。 サイズにこれ以外の値を指定すると、その値より大きい次の値に変更されます。 IQ ページ・サイズは、データベースのデフォルト I/O 転送ブロック・サイズと最大データ圧縮比を決定します。

パフォーマンスを高めるために、次の最小 IQ ページ・サイズの使用をおすすめします。

- 64KB (**IQ PAGE SIZE 65536**)。最も大きいテーブルのロー数が 10 億以下のデータベース、または合計サイズが 8TB より小さいデータベース。これは新規データベースの最小サイズです。32 ビット・プラットフォームでは、IQ ページ・サイズを 64KB にすると最高のパフォーマンスが得られます。
- 128KB (**IQ PAGE SIZE 131072**)。最も大きいテーブルに含まれるローの数が 10 億より多く 40 億より少ないか、合計サイズが 8TB 以上になる可能性がある、64 ビット・プラットフォーム上のデータベース。128KB はデフォルトの IQ ページ・サイズです。
- 256KB (**IQ PAGE SIZE 262144**)。最も大きいテーブルに含まれるローの数が 40 億よりも多いか、合計サイズが 8TB 以上になる可能性がある、64 ビット・プラットフォーム上のデータベース。

VARCHAR のワイド・データのカラム (255 ～ 32,767 バイトのカラム) を複数含むなど、非常に大きなテーブルでは、さらに大きい **IQ PAGE SIZE** が必要になる可能性もあります。

BLOCK SIZE—データベースの Sybase IQ セグメントの I/O 転送ブロック・サイズ (バイト単位)。 **IQ PAGE SIZE** よりも小さい、4096 から 32768 までの 2 の累乗の値を指定してください。それ以外の値をサイズに指定すると、その値より大きい次のサイズに変更されます。 デフォルト値は、**IQ PAGE SIZE** 句の値に応じて異なります。 ほとんどのアプリケーションでは、デフォルト値を使用するのが最適です。 デフォルト以外の値を指定する場合は、『パフォーマンス＆チューニング・ガイド』の「システム・リソースの管理」を参照してください。

IQ RESERVE—領域のサイズをメガバイト単位で指定して、メイン IQ ストア (IQ_SYSTEM_MAIN DB 領域) 用に領域を確保します。これにより、dbfile のサイズを後で増やすことが可能になります。*sizeMB* パラメータには、0 よりも大きい任意の数値を指定できます。DB 領域を作成した後で、このサイズを変更することはできません。

IQ RESERVE を指定すると、データベースの内部 (フリー・リスト) 構造により多くの領域が使用されるようになります。予約サイズが大きすぎると、内部構造体が必要とする領域が、指定されたサイズよりも大きくなり、エラーになります。

TEMPORARY RESERVE 句—領域のサイズをメガバイト単位で指定して、テンポラリ IQ ストア (IQ_SYSTEM_TEMP DB 領域) 用に領域を確保します。これにより、dbfile のサイズを後で増やすことが可能になります。*sizeMB* パラメータには、0 よりも大きい任意の数値を指定できます。DB 領域を作成した後で、このサイズを変更することはできません。

TEMPORARY RESERVE を指定すると、データベースの内部 (フリー・リスト) 構造により多くの領域が使用されるようになります。予約サイズが大きすぎると、内部構造体が必要とする領域が、指定されたサイズよりも大きくなり、エラーになります。

注意： データベースがバックアップからリストアされると、テンポラリ DB 領域用に確保された領域とモードは破棄されます。

MESSAGE PATH—Sybase IQ メッセージ・トレース・ファイルを含むセグメントのパス名。 オペレーティング・システム・ファイルを指定する必要があります。 メッセージ・ファイルはロー・パーティション上には作成できません。 相対パスを指定するか、パスを省略した場合、メッセージ・ファイルは .db ファイルのディレクトリに相対して作成されます。

TEMPORARY PATH—特定のクエリが生成したテンポラリ・テーブルが入っているテンポラリ・セグメント・ファイルのパス名。オペレーティング・システム・ファイル、または I/O デバイスのロー・パーティションを指定できます (各プラットフォーム用の『インストールおよび設定ガイド』にロー・パーティションを指定する場合のフォーマットについて記載されています)。Sybase IQ は指定したパス名に基づいたタイプを自動的に検出します。 相対パスを指定するか、パスを省略した場合、テンポラリ・ファイルは .db ファイルのディレクトリに相対して作成されます。

TEMPORARY SIZE—**TEMPORARY PATH** 句で指定するロー・パーティションまたはオペレーティング・システム・ファイルのいずれかのサイズ (メガバイト単位)。ロー・パーティションの場合、**TEMPORARY SIZE** を指定しないで、Sybase IQ がロー・パーティション全体を使用できるデフォルトを常に指定してください。 オペレーティング・システム・ファイルのデフォルトは、常に **IQ SIZE** 値の 2 分の 1 です。IQ ストアがロー・パーティションにあり、テンポラリ・ストアがオペレーティング・システム・ファイルである場合、デフォルトの **TEMPORARY SIZE** は IQ ストアのロー・パーティションの半分のサイズになります。

DBA USER—DBA 権限のあるデフォルトのユーザ・アカウントのユーザ名。 この句を指定しない場合、Sybase IQ によってデフォルトの dba ユーザ ID が作成されます。

DBA PASSWORD—DBA 権限のあるデフォルトのユーザ・アカウントのパスワード。

関連する動作：

- オートコミット。

標準

- SQL—ISO/ANSI SQL 文法のベンダ拡張。
- Sybase—Adaptive Server Enterprise は、**CREATE DATABASE** 文を用意していますが、オプションは異なります。

パーミッション

この文を実行するのに必要なパーミッションは、**-gu** オプションを使用してサーバ・コマンド・ラインで設定されます。デフォルトの設定は、DBA 権限を必要とします。

サーバを実行中のアカウントには、ファイルが作成されたディレクトリの書き込みパーミッションが必要です。

参照：

- CREATE DBSPACE 文 (93 ページ)
- DROP DATABASE 文 (213 ページ)

CREATE DBSPACE 文

IQ メイン・ストアまたはカタログ・ストアに使用する新しい DB 領域および関連する dbfile を作成します。

構文

構文 1

カタログ・ストア DB 領域のみ (SQL Anywhere (SA) DB 領域) に使用します。

```
CREATE DBSPACE dbspace-name AS file-path CATALOG STORE
```

構文 2

IQ DB 領域に使用します。

```
CREATE DBSPACE dbspace-name USING file-specification
[ IQ STORE ] iq-dbspace-opts
```

パラメータ

- **file-specification：** – { *single-path-spec* | *new-file-spec* [, ...] }
- **single-path-spec：** – *'file-path'* | *iq-file-opts*
- **new-file-spec：** – **FILE** *logical-file-name* | *'file-path' iq-file-opts*
- **iq-file-opts：** – [[**SIZE**] *file-size*] …[**KB** | **MB** | **GB** | **TB**]] [**RESERVE** *size* …[**KB** | **MB** | **GB** | **TB**]]
- **iq-dbspace-opts：** – [**STRIPING**] {ON | OFF}] …[**STRIPESIZEKB***sizeKB*]

例

- **例 1** – UNIX システム上の 2 つのファイルを使用して、DspHist という名前の DB 領域を IQ メイン・ストア用に作成します。各ファイルのサイズは 1GB で、500MB まで増やすことができます。

```
CREATE DBSPACE DspHist USING FILE
FileHist1 '/History1/data/file1'
SIZE 1000 RESERVE 500,
FILE FileHist2 '/History1/data/file2'
SIZE 1000 RESERVE 500;
```

- **例 2** – DspCat2 という名前の 2 番目のカタログ DB 領域を作成します。

```
CREATE DBSPACE DspCat2 AS
'catalog_file2'
CATALOG STORE;
```

- **例 3** – EmpStore1 という名前の IQ メイン DB 領域を IQ ストア用に作成します (3 つの別の構文を例示しています)。

```
CREATE DBSPACE EmpStore1
USING FILE EmpStore1
'EmpStore1.IQ' SIZE 8 MB IQ STORE;
```

```
CREATE DBSPACE EmpStore1
USING FILE EmpStore1
'EmpStore1.IQ' 8 IQ STORE;
```

```
CREATE DBSPACE EmpStore1
USING FILE EmpStore1
'EmpStore1.IQ' 8;
```

使用法

CREATE DBSPACE は、IQ メイン・ストアまたはカタログ・ストアに使用する新しい DB 領域を作成します。追加する DB 領域を最初の DB 領域とは別のディスク・デバイス上に置き、1 台の物理デバイスの容量を超えるストアを作成することも可能です。

構文 1 は、カタログ・ストアに使用する DB 領域を作成します。DB 領域と dbfile の論理名は同じです。カタログ・ストアにある各 DB 領域には、ファイルは 1 つしかありません。

new-file-spec は、IQ メイン・ストアに使用する DB 領域を作成します。IQ メイン・ストアでは、1 つまたは複数の dbfile を指定できます。各ファイルに対して、dbfile 名と物理ファイル・パスは必要かつユニークでなければなりません。

DB 領域名と dbfile 名の大文字と小文字は常に区別されません。物理ファイル・パスについては、データベースが **CASE RESPECT** の場合はオペレーティング・システムで大文字と小文字が区別され、**CASE IGNORE** の場合は区別されません。

IQ テンポラリ・ストア用の DB 領域を作成することはできません。
IQ_SYSTEM_TEMP という名前の 1 つのテンポラリ DB 領域は、新しいデータベースを作成する場合や 15.3 よりも前のバージョンの Sybase IQ で作成されたものをアップグレードする場合に作成されます。 **ALTER DBSPACE ADD FILE** 構文を使用すると、IQ_SYSTEM_TEMP DB 領域にファイルを追加できます。

RESERVE 句—将来、DB 領域のサイズを増加できるように、予約領域のサイズをキロバイト (KB)、メガバイト (MB)、ギガバイト (GB)、テラバイト (TB) の単位で指定します。*size* パラメータには、0 より大きい任意の値を指定できます。デフォルトはメガバイトです。DB 領域および dbfile を作成した後で、この予約サイズは変更できません。

RESERVE を指定すると、データベースの内部 (フリー・リスト) 構造により多くの領域が使用されるようになります。予約サイズが大きすぎると、内部構造体が必要とする領域が、指定されたサイズよりも大きくなり、エラーになります。

デフォルトで作成されたファイルの名前と種類については、「CREATE DATABASE 文」を参照してください。

注意： マルチプレックス・データベースの DB 領域の作成方法については、『Sybase IQ Multiplex の使用』を参照してください。

次のいずれかの方法で、ユニークなパスを作成できます。

- ファイルごとに異なる拡張子を指定する (mydb.iq など)。
- 異なるファイル名を指定する (mydb2.iq など)。
- 異なるパス名 (/iqfiles/main/iq など) または異なるロー・パーティションを指定する。

警告！ UNIX プラットフォームでは、データベースの一貫性を保つために、別のファイルにリンクするファイル名を指定してください。Sybase IQ は、リンクされたファイルが指す位置を検出できません。 コマンド内でファイル名が異なっていても、同じオペレーティング・システム・ファイルをポイントしていないか確認してください。

dbspace-name と *dbfile-name* は DB 領域と dbfile の内部名です。*filepath* は dbfile の実際のオペレーティング・システム・ファイル名であり、必要に応じてその前にパスを付けます。明示的なディレクトリ指定がない `filepath` は、データベースのカタログ・ストアと同じディレクトリに作成されます。相対ディレクトリはカタログ・ストアに相対となります。

SIZE 句—*filepath* で指定したオペレーティング・システム・ファイルのサイズを 0 から 4 テラバイトまでの間で指定します。デフォルトはストア・タイプとブロック・サイズに応じて異なります。IQ メイン・ストアの場合、デフォルトのバイト数は 1000 * ブロック・サイズになります。カタログ・ストアに **SIZE** 句を指定することはできません。

SIZE に 0 の値を指定すると、最小サイズの DB 領域が作成されます。これは、IQ メイン・ストアでは 8MB です。

ロー・パーティションの場合は、**SIZE** を明示的に指定しないでください。Sybase IQ は自動的にこのパラメータを最大ロー・パーティション・サイズに設定し、それ以外のサイズが指定されるとエラーを返します。

STRIPESIZEKB 句—ディスク・ストライピング・アルゴリズムが指定した DB 領域の次のストライプに移動する前に、各ファイルに書き込むデータ量をキロバイト (KB) で指定します。

ストライピングまたはストリップ・サイズを指定しない場合は、`DEFAULT_DISK_STRIPING` オプションと `DEFAULT_KB_PER_STRIPE` オプションのデフォルト値が適用されます。

データベースは、データベース作成時に作成した最初の DB 領域を含め、(32KB - 1) までの DB 領域を持つことが可能です。ただし、使用しているオペレーティング・システムで、データベースごとのファイル数が制限される可能性もあります。

『システム管理ガイド：第 1 巻』の「データベース・オブジェクトの管理」を参照してください。

関連する動作：

- オートコミット。
- 自動チェックポイント。

標準

- 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。

パーミッション

DBA 権限または SPACE ADMIN 権限を持っている必要があります。

参照：

- CREATE DATABASE 文 (81 ページ)
- DROP 文 (209 ページ)

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。

参照：

- DROP 文 (209 ページ)

CREATE EVENT 文

イベントとそのハンドラ (事前定義のアクションを自動化する) を定義します。また、スケジュールされたアクションも定義します。

構文

```
CREATE EVENT event-name
[ TYPE event-type
        [ WHERE trigger-condition [ AND trigger-condition ], ...]
    | SCHEDULE schedule-spec, … ]
…[ ENABLE | DISABLE ]
…[ AT { CONSOLIDATED | REMOTE | ALL } ]
…[ HANDLER
       BEGIN
…
   END ]
```

パラメータ

- **event-type： – BackupEnd | "Connect"** | **ConnectFailed** | **DatabaseStart** | **DBDiskSpace** | **"Disconnect"** | **GlobalAutoincrement** | **GrowDB** | **GrowLog** | **GrowTemp | IQMainDBSpaceFree** | **IQTempDBSpaceFree** | **LogDiskSpace** | **"RAISERROR"** | **ServerIdle** | **TempDiskSpace**
- **trigger-condition： – event_condition(** *condition-name*) { **=** | **<** | **>** | **!=** | **<=** | **>=** } *value*
- **schedule-spec： –** [*schedule-name*] { **STARTTIME***start-time* | **BETWEEN***start-time***AND***end-time* } [**EVERY***period* { **HOURS** | **MINUTES** | **SECONDS** }] [**ON** { (*day-of-week*, …) | (*day-of-month*, …) }] [**STARTDATE***start-date*]
- **event-name | schedule-name： –** *識別子*
- **day-of-week： –** *文字列*
- **day-of-month | value | period： –** *整数*
- **start-time | end-time： –** *時刻*
- **start-date： –** *日付*

例

- **例 1 –** この例は、毎日午前 1 時に自動インクリメンタル・バックアップを実行するよう、データベース・サーバに指示します。

```
CREATE EVENT IncrementalBackup
SCHEDULE
START TIME '1:00AM' EVERY 24 HOURS
HANDLER
```

```
BEGIN
  BACKUP DATABASE INCREMENTAL
  TO 'backups/daily.incr'
END
```

- **例 2 –** この例は、システム・ストアド・プロシージャ **sp_iqspaceused** を 10 分おきに呼び出し、その返り値である現在の日付と時刻、データベースへの現在の接続数、メイン IQ ストアとテンポラリ IQ ストアの使用に関する現在の情報をテーブルに格納するよう、データベース・サーバに指示します。

```
CREATE TABLE mysummary(dt DATETIME,
  users INT, mainKB UNSIGNED BIGINT,
  mainPC UNSIGNED INT,
  tempKB UNSIGNED BIGINT,
  tempPC UNSIGNED INT) ;
```

```
CREATE EVENT mysummary
  SCHEDULE sched_mysummary
    START TIME '00:01 AM' EVERY 10 MINUTES
  HANDLER
  BEGIN
    DECLARE mt UNSIGNED BIGINT;
    DECLARE mu UNSIGNED BIGINT;
    DECLARE tt UNSIGNED BIGINT;
    DECLARE tu UNSIGNED BIGINT;
    DECLARE conncount UNSIGNED INT;

    SET conncount = DB_PROPERTY('ConnCount');
    CALL SP_IQSPACEUSED(mt,mu,tt,tu);

        INSERT INTO mysummary VALUES( NOW(),
    conncount, mu, (mu*100)/mt, tu,
    (tu*100)/tt );
    END;
```

- **例 3 –** トランザクション・ログ・ファイルが格納されているデバイス上の空きディスク領域が 30% を下回った場合に、サーバ・ログにメッセージを記録します。ただし、ハンドラが実行されるのは、300 秒に 1 回だけです。

```
CREATE EVENT LowTxnLogDiskSpace
TYPE DBDiskSpace
WHERE event_condition( 'DBFreePercent' ) < 30
AND event_condition( 'Interval' ) >= 300
HANDLER
BEGIN
message 'Disk space for Transaction Log is low.';
END;
```

その他の例については、『システム管理ガイド：第 2 巻』の「スケジューリングとイベント処理によるタスクの自動化」>「イベントのトリガ条件の定義」を参照してください。

使用法

イベントは、次の 2 とおりの方法で使用できます。

- スケジュール・アクション – データベース・サーバが、スケジュールされた時間に一連のアクションを実行します。この機能を使用して、バックアッ プ、妥当性チェック、レポート用テーブルに記録するためのクエリの実行などをスケジュールできます。
- イベント処理アクション – データベース・サーバが、事前に定義されたイベントが発生したときに一連のアクションを実行します。処理可能なイベントは、ディスク領域の制限 (指定されたパーセンテージ) を超えるデータが入力されたとき、サーバがアイドル状態にあるときなどです。

イベント定義は、2 つの異なる要素で構成されます。トリガ条件には、ディスクに定義されたしきい値を超えるデータが入力されたなどの出来事を指定します。スケジュールは時間の設定です。それぞれがトリガ条件として機能します。トリガ条件を満たすと、イベント・ハンドラが実行されます。イベント・ハンドラには、複合文 (**BEGIN... END**) の中に指定された 1 つまたは複数のアクションが含まれています。

トリガ条件やスケジュールを指定しない場合は、明示的な **TRIGGER EVENT** 文だけがイベントをトリガします。開発時には **TRIGGER EVENT** を使用してイベント・ハンドラを開発したり、テストしたりし、テストが完了してからスケジュールや **WHERE** 句を追加することもできます。

イベント・エラーが起こると、データベース・サーバ・コンソールにログがとられます。

イベント・ハンドラがトリガされると、サーバはイベントをトリガさせた接続 ID (イベント・ハンドラが **EVENT_PARAMETER** 関数を使用する際に使用可能) などのコンテキスト情報を作成します。

注意： イベント内では結果セットを返す文を使用することはできませんが、イベントがストアド・プロシージャを呼び出してその結果をテンポラリ・テーブルに挿入するようにすることはできます。『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」>「データベースからデータをエクスポートする方法」>「データ抽出機能」>「データ抽出オプションの有効化」>「抽出処理の制約事項」を参照してください。

CREATE EVENT – *event-name* は識別子です。イベントを作成したユーザがそのイベントの作成者となり、イベント・ハンドラはその作成者のパーミッションで実行されます。これは、ストアド・プロシージャが実行される場合と同じです。他のユーザが所有するイベントを作成することはできません。

イベント名は、システム・テーブル SYSEVENT をクエリして、一覧表示できます。次に例を示します。

```
SELECT event_id, event_name FROM SYS.SYSEVENT
```

TYPE – *event-type* には、リストで示したシステム定義のイベント・タイプのうちいずれかを指定します。イベント・タイプの大文字と小文字は区別されません。*event-type* がイベントをトリガする条件を指定するには、**WHERE** 句を使用します。

- DiskSpace イベント・タイプ—データベースに DiskSpace タイプのいずれかに対するイベント・ハンドラがある場合、データベース・サーバは関連するファイルが含まれる各デバイスの利用可能な領域を 30 秒ごとにチェックします。

  データベースが別々のドライブに複数の DB 領域を持っている場合、**DBDiskSpace** は各ドライブをチェックし、その中で最小の空き領域に基づいて動作します。

  LogDiskSpace イベント・タイプは、トランザクション・ログとミラーリング・トランザクション・ログのロケーションをチェックし、最小の空き領域に基づいてレポートします。
- Globalautoincrement イベント・タイプ—このイベントはテーブルにおける GLOBAL AUTOINCREMENT のデフォルト値が、その範囲の最大値の 1 パーセントに満たない場合に起動されます。このハンドラのアクションでは一般に、**GLOBAL_DATABASE_ID** オプションの新しい値が要求されます。

  EVENT_CONDITION 関数と **RemainingValues** をこのイベント・タイプの引数として使用できます。
- ServerIdle イベント・タイプ—データベースに **ServerIdle** タイプのイベント・ハンドラがある場合、サーバは 30 秒おきにサーバのアクティビティをチェックします。

WHERE 句—トリガ条件には、イベントを起動する条件を定義します。たとえば、トランザクション・ログを格納するディスクの使用量が 80% を超えたときにアクションを起こす場合は、次のトリガ条件を使用します。

```
... WHERE event_condition( 'LogDiskSpacePercentFree' ) < 20 ...
```

EVENT_CONDITION 関数には、イベント・タイプに有効な引数を指定してください。

複数の **AND** 条件を使用して **WHERE** 句を作成できますが、**OR** などの条件を使用することはできません。

有効な引数の詳細については、『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 関数」>「アルファベット順の関数リスト」>「EVENT_CONDITION 関数 [システム]」を参照してください。

SCHEDULE – スケジュールされたアクションをいつ開始するかを指定します。時間を並べて指定すると、イベント・ハンドラに定義されたアクションに対するトリガ条件のセットとして機能します。

指定したイベントとそのハンドラに、複数のスケジュールを作成できます。これにより、複雑なスケジュールの実装が可能になります。スケジュールが複数あれ

ばスケジュール名の指定が必要になりますが、スケジュールを 1 つだけ指定することも選択できます。

スケジュール名は、システム・テーブル SYSSCHEDULE をクエリして、一覧表示できます。次に例を示します。

```
SELECT event_id, sched_name FROM SYS.SYSSCHEDULE
```

イベントには固有のイベント ID が付いています。イベントと関連するスケジュールの対応付けには、SYSEVENT と SYSSCHEDULE の event_id カラムを使用します。

反復しないようにスケジュールしたイベントが渡された場合、そのスケジュールは削除されますが、イベント・ハンドラは削除されません。

スケジュールされたイベントの時間は、スケジュールが作成されたときに計算され、イベント・ハンドラの実行が終了したときに再度計算されます。次回のイベント時間は、イベントのスケジュールを調べ、次の (将来の) スケジュール時間を検出することで計算されます。イベント・ハンドラに対して、9:00 から 5:00 までの間、1 時間おきに実行するよう指示し、1 回の実行に 65 分間かかる場合、9:00、11:00、1:00、3:00、5:00 に実行されます。重ねて実行させるには、複数のイベントを作成する必要があります。

スケジュール定義のサブ句には、次のものがあります。

- START TIME – 1 日の間で、イベントはこのスケジュールされた時間よりも後にスケジュールされます。**START DATE** を指定すると、**START TIME** はその日の時間となります。**START DATE** が指定されなければ、**START TIME** はその日 (指定の時刻が過ぎていなければ) とそれ以降の毎日の時間となります。
- BETWEEN … AND – 1 日の間でこの時間範囲にない時間に、スケジュールされた時間がくることはありません。**START DATE** が指定された場合は、その日がくるまでスケジュールされた時間になりません。
- EVERY – 継続的にスケジュールされたイベントの間隔。スケジュールされたイベントは、その日の **START TIME** より後、または **BETWEEN …AND** で指定された範囲内でのみ発生します。
- ON – スケジュールされたイベントを開始する日を並べて指定します。デフォルトは毎日です。曜日または日にちで指定します。
  曜日は Monday、Tuesday などのように指定します。Mon、Tue などのように省略形を使用することもできます。データベース・サーバは Sybase IQ でサポートされるどの言語で書かれた日にちの名前でも (または、省略形でも) 認識します。
  日付は、0 ～ 31 の整数で指定します。0 は月の末日を表します。
- START DATE – スケジュールされたイベントは、この日以降に開始されます。デフォルトは現在の日付です。

スケジュールされたイベント・ハンドラが終了するたびに、次のスケジュールされた時間と日付が計算されます。

1. **EVERY** 句を使用した場合は、次のスケジュールされた時刻が現在の日付にあり、**BETWEEN ...AND** で指定された範囲の終わりより前であるかどうかを調べます。そうであれば、それが次のスケジュールされた時刻となります。
2. 次のスケジュールされた時間がその日のうちにこない場合は、次回イベントが実行される日が検出されます。
3. その日の **START TIME**、または **BETWEEN ... AND** の起点が検出されます。

ENABLE | DISABLE – デフォルトでは、イベント・ハンドラは有効です。**DISABLE** を指定すると、スケジュールされた時刻に達したりトリガ条件が発生しても、イベント・ハンドラは起動しません。**TRIGGER EVENT** 文によって、無効なイベント・ハンドラが実行されることはありません。

AT – イベントをリモートのデータベース、または [SQLRemote] 設定で統合されたデータベースで実行する場合は、この句を使用してイベントを処理するデータベースを制限できます。デフォルトでは、すべてのデータベースがイベントを実行します。

HANDLER – イベントはそれぞれにハンドラを 1 つ持ちます。ストアド・プロシージャの本体と同様、ハンドラも複合文です。ただし、いくつか相違点があります。複合文内で **EXCEPTION** 句を使用してエラーを処理できます。ただし、ストアド・プロシージャ内で使用可能な **ON EXCEPTION RESUME** 句は使用できません。

『システム管理ガイド：第 2 巻』の「スケジューリングとイベント処理によるタスクの自動化」も参照してください。

関連する動作：

- オートコミット。
- イベント・ハンドラのアクションは、実行中にエラーが検出されなければコミットされ、検出されればロールバックされます。

標準

- 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。

パーミッション

DBA 権限が必要です。

イベント・ハンドラは別個の接続を使用して、イベント所有者のパーミッションで実行されます。DBA 以外のパーミッションで実行するには、イベント・ハンドラの内部からプロシージャを呼び出します。この場合、プロシージャはその所有

者のパーミッションで実行されます。この別個の接続は、パーソナル・データベース・サーバでの接続数の制限 (10 まで) にはカウントされません。

参照：

- ALTER EVENT 文 (15 ページ)
- BEGIN … END 文 (60 ページ)
- COMMENT 文 (72 ページ)
- DROP 文 (209 ページ)
- TRIGGER EVENT 文 (368 ページ)

CREATE EXISTING TABLE 文

リモート・サーバ上の既存のテーブルを表す新しいプロキシ・テーブルを作成します。

構文

```
CREATE EXISTING TABLE [owner.]table_name
[ ( column-definition, … ) ]
AT 'location-string'
```

パラメータ

- **column-definition：** – *column-namedata-type* [**NOT NULL**]
- **location-string：** – *remote-server-name*.[*db-name*].[*owner*].*object-name* | *remote-server-name*;[*db-name*];[*owner*];*object-name*

例

- **例 1** – リモート・サーバ `server_a` にある `nation` テーブルのプロキシ・テーブル `nation` を作成します。

  ```
  CREATE EXISTING TABLE nation
  ( n_nationkey int,
    n_name char(25),
    n_regionkey int,
    n_comment char(152))
  AT 'server_a.db1.joe.nation'
  ```

- **例 2** – リモート・サーバ `server_a` にある `blurbs` テーブルに対して `blurbs` という名前のプロキシ・テーブルを作成します。Sybase IQ は、リモート・テーブルから取得したメタデータからカラム・リストを導出します。

  ```
  CREATE EXISTING TABLE blurbs
  AT 'server_a.db1.joe.blurbs'
  ```

- **例 3 –** Sybase IQ リモート・サーバ `remote_iqdemo_srv` にある `Employees` テーブルのプロキシ・テーブル `rda_employee` を作成します。

```
CREATE EXISTING TABLE rda_employee
AT 'remote_iqdemo_srv..dba.Employees'
```

使用法

CREATE EXISTING TABLE は、**CREATE TABLE** 文の変形です。**EXISTING** キーワードを **CREATE TABLE** で使用すると、テーブルがすでにリモートに存在していて、そのメタデータが Sybase IQ にインポートされるように指定できます。 これにより、リモート・テーブルはユーザに対して可視なエンティティとして作成されます。Sybase IQ はテーブルを作成する前に、テーブルが外部の場所に存在することを確認します。

プロキシ・テーブルとして使用するテーブルに、30 文字よりも長い名前を付けることはできません。

オブジェクト (ホスト・データ・ファイルまたはリモート・サーバ・オブジェクトのどちらか) が存在しない場合、この文は拒否されてエラー・メッセージが出力されます。

ホスト・データ・ファイルまたはリモート・サーバ・テーブルのインデックス情報が抽出されて、システム・テーブル sysindexes のローを作成するために使用されます。これにより、サーバ用語のインデックスとキーが定義されて、クエリ・オプティマイザがこのテーブルに存在する可能性のあるすべてのインデックスを考慮できるようになります。

参照整合性制約は、必要に応じてリモート・ロケーションに渡されます。

カラム定義を指定しない場合、Sybase IQ は、リモート・テーブルから取得するメタデータからカラム・リストを導出します。カラム定義を指定した場合、Sybase IQ は、そのカラム定義を検証します。Sybase IQ では、カラム名、データ型、長さ、null プロパティについて、次の点をチェックします。

- カラム名が一致しなければなりません (大文字小文字は無視されます)。
- **CREATE EXISTING TABLE** 文のデータ型は、リモート・ロケーションのカラムのデータ型と一致するか、またはそのデータ型に変換可能でなければなりません。たとえば、ローカル・カラムのデータ型が `NUMERIC` として定義されているのに対して、リモート・カラムのデータ型が `MONEY` である場合があります。データ型が一致しないテーブル、またはその他の不整合が存在するテーブルから選択した場合、エラーが発生する可能性があります。
- 各カラムの NULL プロパティがチェックされます。ローカル・カラムの NULL プロパティがリモート・カラムの NULL プロパティと同じでない場合、警告メッセージが出力されますが、文はアボートしません。

- 各カラムの長さがチェックされます。CHAR、VARCHAR、BINARY、DECIMAL、NUMERIC の各カラムの長さが一致しない場合は、警告メッセージが出力されますが、コマンドはアボートしません。**CREATE EXISTING** 文には、実際のリモート・カラム・リストのサブセットだけをインクルードできます。
- **AT** は、リモート・オブジェクトのロケーションを指定します。 **AT** 句は、デリミタとしてセミコロン (;) をサポートします。 セミコロンがロケーション文字列のどこかにある場合、そのセミコロンはフィールド・デリミタです。 セミコロンがない場合は、ピリオドがフィールド・デリミタです。 ピリオドを使用すると、データベースおよび所有者の各フィールドにファイル名と拡張子を使用できます。 セミコロンのフィールド・デリミタは、現在サポートされていないサーバ・クラスで主に使用されていますが、ピリオドもフィールド・デリミタとして機能する状況ではセミコロンも使用できます。 たとえば、次の文は、テーブル proxy_a1 をリモート・サーバ myasa の SQL Anywhere データベース mydb にマッピングします。

```
CREATE EXISTING TABLE
proxy_a1
AT 'myasa;mydb;;a1'
```

シンプレックス環境では、同じノード上でリモート・テーブルを参照するプロキシ・テーブルを作成することはできません。マルチプレックス環境では、マルチプレックス内で定義されたリモート・テーブルを参照するプロキシ・テーブルを作成することはできません。

たとえば、シンプレックス環境で、同じノード上で定義されたベース・テーブル Employees を参照するプロキシ・テーブル proxy_e を作成しようとすると、**CREATE EXISTING TABLE** 文が拒否され、エラー・メッセージが表示されます。マルチプレックス環境では、マルチプレックス内で定義されたリモート・テーブル Employees を参照する任意のノード (コーディネータまたはセカンダリ) からプロキシ・テーブル proxy_e を作成する場合に、**CREATE EXISTING TABLE** 文が拒否されます。

『システム管理ガイド：第 2 巻』の「リモート・データへのアクセス」と『システム管理ガイド： 第 2 巻』の「リモート・データ・アクセス用のサーバ・クラス」を参照してください。

標準

- SQL—ISO/ANSI SQL 準拠。
- Sybase—Open Client/Open Server でサポートされています。

パーミッション

RESOURCE 権限が必要です。別のユーザのテーブルを作成するには、DBA 権限が必要です。

参照：
- CREATE TABLE 文 (166 ページ)

CREATE EXTERNLOGIN 文

リモート・サーバとの通信に使用する代替ログイン名とパスワードを割り当てます。

構文

```
CREATE EXTERNLOGIN login-name
TO remote-server
REMOTE LOGIN remote-user
[ IDENTIFIED BY remote-password ]
```

例

- **例 1 –** サーバ **sybase1** に接続するときに、ローカル・ユーザ **DBA** をパスワード 4TKNOX が設定されたユーザ **sa** にマッピングします。

```
CREATE EXTERNLOGIN dba
TO sybase1
REMOTE LOGIN sa
IDENTIFIED BY 4TKNOX
```

使用法

CREATE EXTERNLOGIN による変更は、次にリモート・サーバに接続するまで有効になりません。

デフォルトでは、Sybase IQ は、クライアントに代わってリモート・サーバに接続するときは、常にそのクライアントの名前とパスワードを使用します。**CREATE EXTERNLOGIN** は、リモート・サーバとの通信に使用される代替ログイン名とパスワードを割り当てます。 パスワードは暗号形式で内部に格納されます。 *remote_server* は、ISYSSERVER システム・テーブルのエントリによって、ローカル・サーバに認識させる必要があります。 詳細については、「CREATE SERVER 文」を参照してください。

CREATE EXTERNLOGIN 文でリモート・ログインを作成し、**CREATE SERVER** 文でリモート・サーバを定義すると、ユーザがどのような状況でもログインとパスワードを使用できるように、**INSERT...LOCATION** の外部ログインとパスワードが設定されます。 これによって、ログインまたはパスワードにアクセスできないことが原因で発生する可能性のあるエラーを回避できます。この方法で、リモート・サーバに接続することをおすすめします。

注意： 現在の接続のユーザ ID とパスワードを使用している場合に、ユーザがパスワードを変更すると、リモート・サーバで新しいパスワードが有効になる前に、

サーバを停止して再起動する必要が生じます。**CREATE EXTERNLOGIN** を使用して作成したリモート・ログインは、デフォルト・ユーザ ID のパスワードが変更されても影響を受けません。

パスワードの自動有効期限を使用するサイトでは、外部ログインのために、定期的なパスワードの更新を計画する必要があります。

CREATE EXTERNLOGIN は、トランザクション内では使用できません。

login-name—ローカル・ユーザ・ログイン名を指定します。統合化ログインを使用する場合、*login-name* は Windows ユーザ ID のマッピング先となるデータベース・ユーザです。

TO—**TO** 句は、リモート・サーバの名前を指定します。

REMOTE LOGIN—**REMOTE LOGIN** 句は、ローカル・ユーザ *login-name* に対して、*remote-server* 上の対応するユーザ・アカウントを指定します。

IDENTIFIED BY— **IDENTIFIED BY** 句は、*remote-password* が *remote-user* のパスワードになるように指定します。**IDENTIFIED BY** 句を省略すると、NULL のパスワードがリモート・サーバに送信されます。**IDENTIFIED BY " "** (空の文字列) を指定すると、送信されたパスワードは空の文字列を持ちます。

remote-user と *remote-password* の組み合わせは、*remote-server* 上で有効でなければなりません。

関連する動作

- オートコミット。

標準

- SQL—ISO/ANSI SQL 準拠。
- Sybase—Open Client/Open Server でサポートされています。

パーミッション

外部ログインを追加または変更できるのは、DBA または USER ADMIN アカウントだけです。

参照：

- DROP EXTERNLOGIN 文 (214 ページ)
- INSERT 文 (258 ページ)
- CREATE SERVER 文 (161 ページ)

CREATE FUNCTION 文

新しい関数をデータベースに作成します。

構文

構文 1

```
CREATE [ OR REPLACE ] [ TEMPORARY ] FUNCTION [ owner.]function-name
( [ parameter, … ] )
 RETURNS data-type routine-characteristics
 [ SQL SECURITY { INVOKER | DEFINER } ]
 { compound-statement
  | AS tsql-compound-statement
  | external-name }
```

構文 2

```
CREATE FUNCTION [ owner.]function-name ( [ parameter, … ] )
 RETURNS data-type
 URL url-string
 [ HEADER header-string ]
 [ SOAPHEADER soap-header-string ]
 [ TYPE { 'HTTP[:{ GET | POST } ] ' | 'SOAP[:{ RPC | DOC } ]' } ]
 [ NAMESPACE namespace-string ]
 [ CERTIFICATE certificate-string ]
 [ CLIENTPORT clientport-string ]
 [ PROXY proxy-string ]
```

パラメータ

- **url-string：** – ' { **HTTP** | **HTTPS** | **HTTPS_FIPS** }://[*user*:*password*@]*hostname*[:*port*][/*path*] '
- **parameter：** – **IN***parameter-name data-type* [**DEFAULT** *expression*]
- **routine-characteristics：** – **ONEXCEPTIONRESUME** | [**NOT**] **DETERMINISTIC**
- **tsql-compound-statement：** – *sql-statement sql-statement …*
- **external-name：** – **EXTERNALNAME***library-call* | **EXTERNALNAME***java-call* **LANGUAGEJAVA**
- **library-call：** – '[*operating-system*:]*function-name@library*; …'
- **operating-system：** – **UNIX**
- **java-call：** – '[*package-name*.]*class-name.method-name method-signature*'
- **method-signature：** – ([*field-descriptor*, ….]) *return-descriptor*
- **field-descriptor and return-descriptor：** – **Z** | **B** | **S** | **I** | **J** | **F** | **D** | **C** | **V** | **[** *descriptor* | **L** *class-name*;

例

- **例 1 –** 次の例は、文字列 `firstname` と文字列 `lastname` を連結します。

```
CREATE FUNCTION fullname (
  firstname CHAR(30),
  lastname CHAR(30) )
RETURNS CHAR(61)
BEGIN
  DECLARE name CHAR(61);
  SET name = firstname || ' ' || lastname;
  RETURN (name);
END
```

fullname 関数は、次の例のように使用します。

- 2 つの提供された文字列からフル・ネームを戻します。

```
SELECT fullname ('joe','smith')
```

fullname('joe', 'smith')
joe smith

- 全従業員の名前をリストします。

```
SELECT fullname (givenname, surname)
FROM Employees
```

fullname (givenname, surname)
Fran Whitney
Matthew Cobb
Philip Chin
Julie Jordan
Robert Breault
...

- **例 2 –** 次の例では、Transact-SQL 構文を使用しています。

```
CREATE FUNCTION DoubleIt ( @Input INT )
RETURNS INT
AS
DECLARE @Result INT
SELECT @Result = @Input * 2
RETURN @Result
```

文 `SELECT DoubleIt( 5 )` は、`10` の値を返します。

- **例 3 –** 次の文は、Java で記述された外部関数を作成します。

```
CREATE FUNCTION dba.encrypt( IN name char(254) )
RETURNS VARCHAR
```

```
EXTERNAL NAME
'Scramble.encrypt (Ljava/lang/String;)Ljava/lang/String;'
LANGUAGE JAVA
```

使用法

CREATE FUNCTION 文は、データベース内でユーザ定義関数を作成します。所有者名を指定することにより、別のユーザが使用する関数を作成できます。パーミッションがあれば、ユーザ定義関数も他の非集合関数とまったく同じように使用できます。

CREATE FUNCTION—パラメータの名前は、データベース識別子の規則に従っている必要があります。 有効な SQL データ型を持たなくてはなりません。また、この引数が関数へ値を提供する式であることを示すキーワード **IN** を先頭に付ける必要があります。

CREATE OR REPLACE FUNCTION を指定すると、新しい関数が作成されるか、同じ名前の既存の関数が置き換えられます。関数が置き換えられると、関数の定義は変更されますが、既存のパーミッションは保持されます。

OR REPLACE 句をテンポラリ関数で使用することはできません。

関数を実行する場合は、必ずしもすべてのパラメータを指定する必要はありません。**CREATE FUNCTION** 文の中にデフォルト値がある場合、不明のパラメータにデフォルト値を割り当てます。引数が呼び出し元から提供されておらず、デフォルトが設定されていない場合、エラーが返されます。

TEMPORARY (**CREATETEMPORARYFUNCTION**) を指定すると、作成した接続でのみ参照できる関数になり、接続を削除すると関数も自動的に削除されます。テンポラリ関数を明示的に削除することもできます。テンポラリ関数に対して **ALTER**、**GRANT**、または **REVOKE** は実行できません。また他の関数とは異なり、テンポラリ関数はカタログやトランザクション・ログに記録されていません。

テンポラリ関数は、作成者 (現在のユーザ) のパーミッションで実行されます。また作成者のみが所有できます。そのため、テンポラリ関数を作成するときは所有者を指定しないでください。

読み込み専用のデータベースに接続するときに、テンポラリ関数の作成と削除を行うことができます。

SQL SECURITY—関数が INVOKER (関数を呼び出しているユーザ)、または DEFINER (関数を所有しているユーザ) として実行されているかを指定します。デフォルトは DEFINER です。

SQL SECURITY INVOKER が指定されていると、プロシージャを呼び出す各ユーザに対して注釈を付ける必要があるため、必要なメモリも多くなります。また、**SQL SECURITY INVOKER** が指定されていると、名前解決も INVOKER として実行さ

れます。したがって、適切な所有者を使用して、すべてのオブジェクト名 (テーブルやプロシージャなど) の条件を満たすように注意してください。

compound-statement—**BEGIN** と **END** で囲まれ、セミコロンで区切られた SQL 文のセット。「BEGIN … END 文」を参照してください。

tsql-compound-statement—ひとまとまりの Transact-SQL 文。『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「他の Sybase データベースとの互換性」>「Transact-SQL のプロシージャ言語の概要」>「Transact-SQL のバッチの概要」と「CREATE PROCEDURE 文 [T-SQL]」を参照してください。

EXTERNAL NAME—**EXTERNAL NAME** 句を使用する関数は、外部ライブラリにある関数への呼び出しのラッパです。**EXTERNAL NAME** を使用する関数では、**RETURNS** 句の後に他の句を置くことはできません。ライブラリ名にはファイル拡張子が付く場合があります。この拡張子は通常、Windows では `.dll`、UNIX では `.so` です。拡張子が付いていなければ、ソフトウェアがプラットフォーム固有のデフォルトのファイル拡張子をライブラリに付加します。

EXTERNAL NAME 句はテンポラリ関数ではサポートされていませんので、注意してください。『SQL Anywhere サーバー – プログラミング』の「SQL Anywhere 外部呼び出しインターフェイス」を参照してください。

注意： このリファレンスは SQL Anywhere マニュアルにリンクされています。

EXTERNAL NAME LANGUAGE JAVA—**LANGUAGE JAVA** 句が付いた **EXTERNAL NAME** を使用する関数は、Java メソッドのラッパです。Java プロシージャの呼び出しについては、「CREATE PROCEDURE 文」を参照してください。

ON EXCEPTION RESUME—Transact-SQL に似たエラー処理です。「CREATE PROCEDURE 文」を参照してください。

NOT DETERMINISTIC—**NOT DETERMINISTIC** として指定された関数は、クエリから呼び出されるたびに再評価されます。このように指定されない関数では、パフォーマンスを高めるために結果がキャッシュされ、クエリの評価中その関数が同じパラメータで呼び出されるたびに再利用されます。

実行時に、データベースのデータを変更するような関数には、 **NOT DETERMINISTIC** を宣言してください。たとえば、プライマリ・キー値を生成し、**INSERT … SELECT** 文で使用される関数は、次のように **NOT DETERMINISTIC** として宣言してください。

```
CREATE FUNCTION keygen( increment INTEGER )
RETURNS INTEGER
NOT DETERMINISTIC
BEGIN
  DECLARE keyval INTEGER;
  UPDATE counter SET x = x + increment;
```

```
  SELECT counter.x INTO keyval FROM counter;
  RETURN keyval
END
INSERT INTO new_table
SELECT keygen(1), ...
FROM old_table
```

特定の入力パラメータに対して常に同じ値を返す関数は、**DETERMINISTIC** として宣言できます。

NOT DETERMINISTIC が宣言されていないかぎり、すべてのユーザ定義関数は決定的として扱われます。DETERMINISTIC 関数は同じパラメータに対して一定の結果を返すため、二次的な影響を心配する必要がありません。つまり、データベース・サーバは同じ関数を同じパラメータで 2 回連続して呼び出したとしても、返される結果が同じであると想定するため、クエリのセマンティクスにおいて厄介な二次的影響が生じることはなくなります。

注意： CLR 外部環境、ESQL 外部環境、ODBC 外部環境、Perl 外部環境、PHP 外部環境のユーザ定義関数は、カタログ・ストア (SQL Anywhere) によって処理されます。 Sybase IQ のパフォーマンス機能は使用されません。 Java のユーザ定義関数は、Sybase IQ によって処理されるため、パフォーマンスが向上します。

特定のケースで、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 データ型を VARCHAR と同じように扱います。

ユーザ定義関数を修正したり、定義をスクランブルすることによって関数の内容を隠したりするには、**ALTER FUNCTION** 文を使用します。

URL—**HTTP** または **SOAP** Web サービス・クライアント関数を定義する場合にのみ使用します。Web サービスの URL を指定します。オプションのユーザ名とパスワードのパラメータは、**HTTP** 基本認証に必要なクレデンシャルとして機能します。**HTTP** 基本認証は、ユーザとパスワードの情報を base-64 でエンコードし、**HTTP** 要求の "Authentication" ヘッダーに渡します。

Web サービス・クライアント関数の場合、**SOAP** 関数と **HTTP** 関数の戻り型は、VARCHAR などの文字データ型の 1 つです。返される値は、**HTTP** 応答の本文です。**HTTP** ヘッダー情報は含まれません。 ステータス情報などの詳細情報が必要な場合は、関数の代わりにプロシージャを使用します。

パラメータ値は要求の一部として渡されます。使用される構文は、要求のタイプにより異なります。**HTTP:GET** では、パラメータは URL の一部として渡されます。**HTTP:POST** 要求では、値が要求の本体として配置されます。**SOAP** 要求へのパラメータは、常に要求本文にバンドルされます。

HEADER—**HTTP** サービス・クライアント関数を作成する場合は、この句を使用して、**HTTP** 要求ヘッダーのエントリを追加または変更します。**HTTP** ヘッダーに指定できるのは印字可能な ASCII 文字のみで、大文字と小文字は区別されません。この句の使用法の詳細については、「CREATE PROCEDURE 文」の **HEADER** 句を参照してください。

HTTP ヘッダーの使用法の詳細については、『SQL Anywhere サーバー – プログラミング』の「HTTP Web サービス」>「Web クライアントを使用した Web サービスへのアクセス」>「Web クライアントアプリケーションの開発」>「Web クライアント関数とプロシージャーの要件と推奨事項」>「HTTP 要求ヘッダーの管理」を参照してください。

注意： このリファレンスは SQL Anywhere マニュアルにリンクされています。

SOAPDHEADER—**SOAP** Web サービスを関数として宣言する場合は、この句を使用して 1 つ以上の **SOAP** 要求ヘッダー・エントリを指定します。 **SOAP** ヘッダーは、静的定数として宣言したり、代入パラメータ・メカニズムを使用して動的に設定したりできます (hd1、hd2 などに **IN**、**OUT**、または **INOUT** パラメータを宣言)。Web サービス関数では、1 つ以上の **IN** モード代入パラメータを定義できますが、**INOUT** または **OUT** 代入パラメータは定義できません。この句の使用法の詳細については、『SQL Anywhere サーバー – SQL リファレンス』>「SQL 文」>「SQL 文」>「CREATE PROCEDURE 文 (Web クライアント)」の **SOAPHEADER** 句を参照してください。

注意： このリファレンスは SQL Anywhere マニュアルにリンクされています。

TYPE—Web サービス要求を行う場合に使用するフォーマットを指定します。**SOAP** が指定されている場合、または type 句が含まれていない場合は、デフォルトのタイプである **SOAP**:**RPC** が使用されます。**HTTP** は **HTTP**:**POST** を暗黙的に指定します。**SOAP** 要求は常に XML 文書として送信されるため、**HTTP**:**POST** が **SOAP** 要求を送信するのに常に使用されます。

NAMESPACE—**SOAP** クライアント関数にのみ適用されます。この句は **SOAP**:**RPC** 要求と **SOAP**:**DOC** 要求の両方に通常必要なメソッド・ネームスペースを識別します。要求を処理する **SOAP** サーバは、このネームスペースを使用して、**SOAP** 要求メッセージ本文内のエンティティの名前を解釈します。ネームスペースは、Web サービス・サーバから使用できる **SOAP** サービスの WSDL 記述から取得できます。デフォルト値は、プロシージャの URL のオプションのパス・コンポーネントの直前までです。

CERTIFICATE—安全な (HTTPS) 要求を行うには、HTTPS サーバで使用される証明書にクライアントがアクセスできる必要があります。必要な情報は、セミコロンで区切られたキー／値のペアの文字列で指定されます。証明書はファイルに置か

れ、file キーを使用して提供されるファイルの名前、または証明書全体を文字列に配置できますが、両方は配置できません。次のキーを使用できます。

キー	省略形	説明
file		証明書のファイル名
certificate	cert	証明書自体
company	co	証明書で指定された会社
unit		証明書で指定された会社の部門
name		証明書で指定された共通名

証明書は、HTTPS サーバに対する直接の要求、または安全でないサーバから安全なサーバにリダイレクトされる可能性がある要求に対してのみ必要です。

CLIENTPORT—HTTP クライアント・プロシージャが TCP/IP を使用して通信するポート番号を識別します。この句は、ファイアウォールを介する通信のためのものであり、このような通信にのみおすすめします。ファイアウォールは TCP/UDP ポートに従ってフィルタします。単一のポート番号、ポート番号の範囲、または両方の組み合わせを指定できます。たとえば CLIENTPORT '85,90-97' を指定できます。

『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」>「ネットワーク通信のパラメータ」>「ClientPort パラメータ (CPort)」を参照してください。

PROXY—プロキシ・サーバの URL を指定します。クライアントがプロキシを介してネットワークにアクセスする場合に使用します。プロシージャがプロキシ・サーバに接続し、そのプロキシ・サーバを介して Web サービスに要求を送信することを示します。

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 ではサポートされていません。

パーミッション

RESOURCE 権限が必要です。

Java 関数を含む外部関数には、DBA 権限が必要です。

参照：

- ALTER FUNCTION 文 (17 ページ)
- BEGIN … END 文 (60 ページ)
- CREATE PROCEDURE 文 (137 ページ)
- DROP 文 (209 ページ)
- RETURN 文 (330 ページ)

CREATE FUNCTION 文 (Java UDF)

データベース内に新しい外部 Java テーブル UDF 関数を作成します。

構文

```
CREATE [ OR REPLACE | TEMPORARY ] FUNCTION [ owner.]function-name
( [ parameter, … ] )
 RETURNS data-type routine-characteristics
 [ SQL SECURITY { INVOKER | DEFINER } ]
 { compound-statement
  | AS tsql-compound-statement
  | EXTERNAL NAME 'java-call' LANGUAGE JAVA }
```

パラメータ

- **parameter – IN***parameter-name**data-type* [**DEFAULT***expression*]
- **routine-characteristics – ONEXCEPTIONRESUME** | [**NOT**] **DETERMINISTIC**
- **tsql-compound-statement** – *sql-statement sql-statement …*
- **environment-name – JAVA [ALLOW | DISALLOW SERVER SIDE REQUESTS]**
- **java-call –** '[*package-name.*]*class-name.method-name**method-signature*'
- **method-signature –** ([*field-descriptor*, ….]) *return-descriptor*
- **field-descriptor and return-descriptor – Z** | **B** | **S** | **I** | **J** | **F** | **D** | **C** | **V** | **[** *descriptor* | **L** *class-name***;**

例

- **例 1 –** 次の文は、Java で記述された外部関数を作成します。

```
CREATE FUNCTION dba.encrypt( IN name char(254) )
RETURNS VARCHAR
EXTERNAL NAME
'Scramble.encrypt (Ljava/lang/String;)Ljava/lang/String;'
LANGUAGE JAVA
```

使用法

戻り値のデータ型として LONG BINARY と LONG VARCHAR は使用できません。

EXTERNAL NAME LANGUAGE JAVA—**LANGUAGE JAVA** 句が付いた **EXTERNAL NAME** を使用する関数は、Java メソッドのラッパです。

iq-environment-name: JAVA [ALLOW/ *DISALLOW SERVER SIDE REQUESTS* **]:**

DISALLOW がデフォルトです。

ALLOW は、サーバ側接続が許可されることを示します。

注意： 必要な場合を除き、**ALLOW** は指定しないでください。**ALLOW** を設定すると、特定の種類の Sybase IQ テーブル・ジョインの速度を低下させます。

UDF を使用するときは、同じクエリ内で **ALLOW SERVER SIDE REQUESTS** と **DISALLOW SERVER SIDE REQUESTS** の両方を指定しないでください。

標準

- SQL—ISO/ANSI SQL 準拠。
- Sybase—Adaptive Server Enterprise ではサポートされていません。

パーミッション

RESOURCE 権限が必要です。

外部 Java 関数には、DBA 権限が必要です。

CREATE INDEX 文

指定したテーブル、またはテーブルのペアにインデックスを作成します。

構文

```
CREATE [ UNIQUE ] [ index-type ] INDEX [ IF NOT EXISTS ] index-name
…ON [ owner.]table-name
… ( column-name [ , column-name ] …)
…|[ WITH NULLS NOT DISTINCT ]
…[ { IN | ON } dbspace-name ]
…[ NOTIFY integer ]
…[ DELIMITED BY ‘separators-string ‘ ]
…[ LIMIT maxwordsize-integer ]
```

パラメータ

- **index-type：** – { **CMP** | **HG** | **HNG** | **LF** | **WD** | **DATE** | **TIME** | **DTTM** }

例

- **例 1** – projected_earnings カラムと current_earnings カラムに比較インデックスを作成します。この 2 つのカラムは、精度と位取りが同じである 10 進のカラムです。

```
CREATE

CMP INDEX proj_curr_cmp
ON sales_data
( projected_earnings, current_earnings )
```

- **例 2** – SalesOrderItems テーブルの ID カラムに High_Group インデックスを作成します。このインデックスのデータ・ページは、DB 領域 Dsp5 から割り付けられます。

```
CREATE

HG INDEX id_hg
ON SalesOrderItems
( ID ) IN Dsp5
```

- **例 3** – SalesOrderItems テーブルの ProductID カラムに High_Group インデックスを作成します。

```
CREATE HG INDEX item_prod_hg
ON Sales_OrderItems
( ProductID)
```

- **例 4** – 通知メッセージを出さないで、SalesOrderItems テーブルの同じ ProductID カラムに Low_Fast インデックスを作成します。

```
CREATE LF INDEX item_prod
ON SalesOrderItems
( ProductID)
 NOTIFY 0
```

- **例 5** – earnings_report テーブルに **WD** インデックスを作成します。文字列のデリミタに、スペース、コロン、セミコロン、ピリオドを指定しています。文字列の長さは 25 に制限されます。

```
CREATE WD INDEX earnings_wd
ON earnings_report_table(varchar)
DELIMITED BY ‘ :;.’
LIMIT 25
```

- **例 6** – SalesOrders テーブルの OrderDate カラムに **DTTM** インデックスを作成します。

```
CREATE DTTM INDEX order_dttm
ON SalesOrders
( OrderDate )
```

使用法

CREATE INDEX 文は、指定したテーブルの指定カラム上でインデックスを作成します。一度インデックスを作成すると、**DROP INDEX** 文を使用して削除するとき以外は、SQL 文中で再び参照されることはありません。

Sybase IQ テーブルのカラムの場合、**HG** (High_Group)、**HNG** (High_Non_Group)、**LF** (Low_Fast)、**WD** (Word)、**DATE**、**TIME**、または **DTTM** (Datetime) を *index-type* に指定できます。 *index-type* を指定しない場合、**HG** インデックスがデフォルトで作成されます。

IQ テーブルの 2 つのカラム間の関係にインデックスを作成するには、**CMP** (Compare) の *index-type* を指定します。2 つのカラムのデータ型、精度、位取りは同じであることが必要です。CHAR、VARCHAR、BINARY、または VARBINARY のカラムの場合、精度は 2 つのカラムの幅が同じであることを意味します。

クエリ速度をできるだけ高速にするために、どのインデックス・タイプを選択するべきかは、次の条件に応じて異なります。

- カラム内のユニークな値の数
- クエリ内での当該カラムの使用方法
- 使用可能なディスク領域量

『システム管理ガイド：第 1 巻』には、インデックス・タイプの詳細と、データに合ったインデックス・タイプの決定方法が説明されています。

IQ テーブルの 1 つのカラムに複数のインデックスを指定できますが、各インデックスは異なるインデックス・タイプでなければなりません。**CREATE INDEX** を使用して、重複するインデックス・タイプを追加することはできません。現在のクエリまたはその一部において利用可能なインデックスの中で最も高速なものが、Sybase IQ によって選択されます。 ただし、追加する各インデックス・タイプによって、そのテーブルの必要領域量が大幅に増加することがあります。

column-name—インデックスを作成するカラムの名前を指定します。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。相関名の詳細については、「FROM 句」を参照してください)。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。

UNIQUE を省略すると、指定可能なインデックスは **HG** だけになります。外部キーにはユニークでない **HG** インデックスが必要で、複合外部キーにはユニークでない複合 **HG** インデックスが必要です。マルチカラムの複合キーの場合、ユニークでもユニークでなくても、**HG** インデックスの幅の最大値は 5300 バイトです。CHAR または VARCHAR のデータは、それが複合キーまたはシングルカラムの **HG**、**LF**、**HNG**、**DATE**、**TIME**、または **DTTM** のインデックスの一部である場合は、255 バイトを超えてはなりません。

UNIQUE—**UNIQUE** 属性を使用すると、インデックスの全カラムに同じ値を持つローが、テーブル内に 2 つ格納されるのを防止できます。各インデックス・キーはユニークであるか、少なくとも 1 つのカラムに NULL が入っている必要があります。複数のカラムを持つユニークな **HG** インデックスは作成できますが、それ以外のインデックス・タイプを使用して、マルチカラム・インデックスを作成することはできません。**UNIQUE** は、**CMP**、**HNG**、**WD**、**DATE**、**TIME**、または **DTTM** のインデックス・タイプを使用して指定することはできません。

Sybase IQ では、カラム定義で NULL 値の使用が許可され、かつ制約 (プライマリ・キーまたは一意性) が設定されていない場合は、ユーザが作成したユニークなマルチカラム HG インデックスのデータ値の中で NULL を使用できます。 詳細については、「注意事項」の「マルチカラム・インデックス」を参照してください。

UNIQUE を指定し、**WITH NULLS NOT DISTINCT** を指定しなかった場合、各インデックス・キーはユニークであるか、少なくとも 1 つのカラムに NULL が入っている必要があります。たとえば、'a', NULL と 'a', NULL という 2 つのエントリは、それぞれユニークと見なされます。

UNIQUE...**WITH NULLS NOT DISTINCT** を指定した場合、インデックス・キーは NULL 値にかかわらず、ユニークである必要があります。たとえば、'a', NULL と 'a', NULL という 2 つのエントリは、同じであると見なされ、ユニークとは見なされません。

IF NOT EXISTS—指定された名前のオブジェクトがすでに存在する場合、変更は行われず、エラーは返されません。

IN—インデックスの配置を指定します。IN 句を省略した場合は、テーブルが作成される DB 領域にインデックスが作成されます。インデックスは常に、そのインデックスのテーブルと同じ型の DB 領域 (IQ ストアまたはテンポラリ・ストア) に配置されます。 インデックスをロードすると、その種類のデータベース・ファイルで、利用可能な領域のあるものすべてにデータが分配されます。Sybase IQ は指定された *dbspace-name* がそのインデックスに適切であることを確認します。 テンポラリ・テーブル上のインデックスに対して `IQ_SYSTEM_MAIN` または他のメイン DB 領域を指定した場合や、その逆の場合には、エラーが発生します。 **CREATE DATABASE**...**CASE IGNORE** または **CASE RESPECT** の指定に関係なく、DB 領域名は常に大文字と小文字が区別されません。

DELIMITED BY—カラムの文字列を複数のワードに解析して、そのカラムの **WD** インデックスに格納するためのセパレータを指定します。 この句が省略されるか、値に空の文字列が指定された場合、Sybase IQ はデフォルトのセパレータのセットを使用します。 デフォルトのセパレータのセットは、デフォルトの照合順序 (ISO-BINENG) にあわせて作成されています。 7 ビットの ASCII 文字 (7 ビットの ASCII 英数字ではありません) で構成され、ハイフンと一重引用符は含まれませ

ん。ハイフンと一重引用符は、デフォルトではワードの一部となります。 デフォルトのセパレータのセットには 64 文字が含まれます。 たとえば、カラムの値が次の文字列だとします。

```
The cat is on the mat
```

また、データベースが **CASE IGNORE** の設定でデフォルトのセパレータを使用して作成されたとすると、この文字列から **WD** インデックスに次のワードが格納されます。

```
cat is mat on the
```

DELIMITED BY 句と **LIMIT** 句を複数指定した場合は、エラーにはなりませんが、それぞれ最後に指定した句だけが使用されます。

separators-string—separators-string には、データベースの作成時に使用した照合順序に含まれる文字を 0 個以上並べて指定する必要があります。separators-string の文字は、それぞれセパレータとして扱われます。separators-string に文字が指定されなければ、デフォルトのセパレータのセットが使用されます(セパレータには使用する照合順序に含まれる単一の文字を指定してください)。separators-string に 256 を超える文字 (セパレータ) を指定することはできません。

タブをデリミタに指定するには、separators-string に <TAB> の文字を入力するか、タブ文字を表す 16 進の ASCII コード (¥x09) を使用します。"¥t"と入力すると、¥ と t が指定されます。改行をデリミタに指定するには、<RETURN> の文字か、16 進の ASCII コード (¥x0a) を入力します。

たとえば、`DELIMITED BY ' :;.¥/t'` 句では、次の 7 つのセパレータが指定されます。`space : ; . ¥ / t`

表 5 : タブと改行をデリミタに指定

デリミタ	DELIMITED BY 句に使用する separators-string
`tab`	`' '` (<TAB> を入力) または `'¥x09'`
`newline`	`' '` (<RETURN> を入力) または `'¥x0a'`

LIMIT—**WD** インデックスの作成時にのみ使用できます。**WD** インデックスで許可される、ワードの最大長を指定します。解析中にこれよりも長いワードが検出されるとエラーが発生します。デフォルトは 255 バイト。指定可能な値の最小値は 1 で、最大値は 255 です。**CREATE INDEX** 文に指定された、またはデフォルトで定義された最大ワード長がカラム幅を超える場合は、使用された最大ワード長がカラム幅まで自動的に削減されます。最大ワード長を短く設定すると、挿入、削除、更新に必要な領域と時間が節約できます。空のワード (セパレータを 2 つ続けて入力) は自動的に無視されます。**WD** インデックスを作成すると、そのカラムへ挿入

されるデータはすべて、作成時に定義したセパレータと最大ワード・サイズで解析されるようになります。セパレータや最大ワード・サイズを、インデックスが作成された後で変更することはできません。

NOTIFY—*n* 個のレコードが正常にインデックスに追加された後で、通知メッセージを表示します。メッセージは標準出力デバイスに送信されます。メッセージには、メモリの使用状況、データベース領域、使用中のバッファ数に関する情報が含まれます。デフォルトは 100,000 レコードです。**NOTIFY** をオフにするには、NOTIFY を 0 に設定します。

注意

- インデックの所有権—**CREATE INDEX** 文ではインデックスの所有者を指定できません。インデックスが定義されているテーブルの所有者が、自動的にそのインデックスの所有者になります。各所有者に対して、インデックス名はユニークである必要があります。
- ビューにインデックスはない—ビューに対してインデックスを作成することはできません。
- インデックス名—各インデックス名は特定のテーブルに対しユニークである必要があります。
- テーブルの排他的使用—**CREATE INDEX** は、この文が別の接続によって現在変更中であるテーブルに影響を与える場合は実行できません。ただし、インデックスを追加しているテーブルに対して、クエリを実行することは可能です。
- CHAR カラム—**WD** インデックスを作成すると、そのカラムへ挿入されるデータはすべて、セパレータと最大ワード・サイズで解析されるようになります。このセパレータと最大ワード・サイズは、インデックスの作成後に変更できません。

  CHAR カラムを使用する場合は、少なくとも空白を区切り文字の 1 つに指定するか、デフォルトの区切り文字セットを使用することを推奨します。Sybase IQ では、CHAR カラムに最大カラム幅になるまで空白が埋め込まれます。 カラムに文字データのほかにブランクが含まれる場合は、**WD** インデックスが作成されたデータへのクエリで、誤った結果が返されることがあります。 たとえば CompanyName というカラムに、セパレータで区切られた 2 つの単語が含まれているとします。ただし、2 つ目の単語には次のように空白が埋め込まれています。

  ```
  ‘Concord’ ‘Farms                                ’
  ```

  このとき、ユーザが次のクエリを入力したとします。

  ```
  SELECT COUNT(*)FROM Customers WHERE CompanyName contains (‘Farms’)
  ```

  パーサはこの文字列を次のように解釈します。

  ```
  ‘Farms           ’
  ```

  次の文字列に一致するとは解釈しないため、

```
'Farms'
```

1 ではなく 0 を返します。この問題を防ぐには、CHAR カラムではなく VARCHAR カラムを使用します。

- データ型
  - **CREATE INDEX** を使用して、BIT データを格納するカラムにインデックスを作成することはできません。
  - 255 バイトよりも大きい CHAR および VARCHAR のデータに作成可能なインデックスは、デフォルト・インデックス、**CMP** インデックス、または **WD** インデックスのみです。
  - LONG VARCHAR データに対して作成できるのは、デフォルトのインデックス・タイプと **WD** インデックス・タイプのみです。
  - 255 バイトよりも大きい BINARY および VARBINARY のデータに作成できるのは、デフォルト・インデックス、**CMP** インデックス、および **TEXT** インデックス・タイプのみです。
  - FLOAT、REAL、DOUBLE のデータを持つカラムに **HNG** インデックスや **CMP** インデックスを作成することはできません。
  - **TIME** インデックスを作成できるのは、TIME データ型を持つカラムだけです。
  - **DATE** インデックスを作成できるのは、DATE データ型を持つカラムだけです。
  - **DTTM** インデックスを作成できるのは、DATETIME か TIMESTAMP のデータ型を持つカラムだけです。
- マルチカラム・インデックス—複数のカラムから、ユニークなまたはユニークでない **HG** インデックスを作成できます。Sybase IQ では、外部キーを構成するカラム・セットに対して、ユニークでない **HG** インデックスが自動的に作成されます。
  複数のカラムを持つことができるインデックス・タイプは、**HG** と **CMP** だけです。複数のカラムから、ユニークな **HNG** または **LF** のインデックス、**DATE**、**TIME**、または **DTTM** のインデックスは作成できません。
  マルチカラムを連結したキーの最大幅は 5KB (5300 バイト) です。連結可能なカラムの数は、5KB に収まるカラムの数によって決まります。CHAR または VARCHAR のデータは、それがシングルカラムの **HG**、**LF**、**HNG**、**DATE**、**TIME**、または **DTTM** のインデックスの複合キーの一部である場合は、255 バイトを超えてはなりません。
  ベース・テーブルのマルチカラム・インデックスは、そのベース・テーブルを使用して作成されたジョイン・インデックスに複写されません。
  マルチカラム・インデックスに対する **INSERT** には、インデックスのすべてのカラムを含める必要があります。

ORDER BY 句にカラムが 1 つだけある場合は、マルチカラム **HG** インデッ クスを使用するとクエリが速く実行されます。次に例を示します。

```
SELECT abs (x) from t1
ORDER BY x
```

上記の例では、**HG** インデックスはソート順に *x* を縦方向に射影します。

クエリ・パフォーマンスを高めるには、マルチカラム **HG** インデックスを使用して、次の状況にある **SELECT** または **ORDER BY** 句にある複数のカラム (**ROWID** を含めることも可能) に対して、**ORDER BY** 操作を実行します。

- すべての射影されたカラムとすべての順序カラム (**ROWID** を除く) がインデックス内に存在する。
- 順序キーが先頭の **HG** カラムに順に一致する。

複数のマルチカラム **HG** インデックスが上記の条件を満たしている場合は、個別カウント数が最も低いインデックスが使用されます。

クエリに **ORDER BY** 句が含まれており、**ORDER BY** カラム・リストが (**SELECT** リストに参照されるすべてのカラムがマルチカラム・インデックス内にある) マルチカラム・インデックスのプレフィクスである場合、マルチカラム・インデックスは次のように縦方向の射影を使用します。

```
SELECT x,z,y FROM T
ORDER BY x,y
```

SELECT リストのベース・カラム上に式が存在し、すべての式で参照されたカラムがすべてマルチカラム・インデックスにある場合、クエリは次のようにマルチカラム・インデックスを使用します。

```
SELECT power(x,2), x+y, sin(z) FROM T
ORDER BY x,y
```

上記の 2 つの例以外にも、**ROWID()** 関数が **SELECT** リストの式にある場合は、マルチカラム・インデックスが使用されます。次に例を示します。

```
SELECT rowid()+x, z FROM T
ORDER BY x,y,z
```

上記の 3 つの例以外にも、**ROWID()** が **ORDER BY** リストの終わりにあり、そのリストのカラム (**ROWID()** のリストを除く) が正確な順序でマルチカラム・インデックスを使用している場合は、マルチカラム・インデックスがクエリに対して使用されます。次に例を示します。

```
SELECT z,y FROM T
ORDER BY x,y,z,ROWID()
```

Sybase IQ では、カラム定義で NULL 値の使用が許可され、かつ制約 (プライマリ・キーまたは一意性) が設定されていない場合は、ユーザが作成したユニークなマルチカラム **HG** インデックスのデータ値の中で NULL を使用できます。この機能の規則を次に示します。

- NULL は未定義の値として扱われる。

- ユニークなインデックス・カラムで、複数のローに NULL 値を設定できる。
  1. シングル・カラム・インデックスで、インデックス・カラムの複数のローに NULL 値を設定できる。
  2. マルチカラム・インデックスで、1 つまたは複数のインデックス・ カラムの複数のローに NULL 値を設定できる。ただし、インデッ クスのユニークさが確保できるように他のカラムが非 NULL 値の場合に限る。
  3. マルチカラム・インデックスで、インデックスに関与しているすべてのカラムで複数のローに NULL 値を設定できる。

  次にこれらの規則の例を示します。次のようにして作成されたテーブル `table1` があるとします。

```
CREATE TABLE table1
(c1 INT NULL, c2 INT NULL, c3 INT NOT NULL);
```

  NULL 値が使用できるカラムに、ユニークなシングル・カラムの **HG** インデックスを作成します。

```
CREATE UNIQUE HG INDEX c1_hg1 ON table1 (c1);
```

  上記の規則 1 に従って、複数のローのインデックス・カラムに NULL 値を挿入します。

```
INSERT INTO table1(c1,c2,c3) VALUES (NULL,1,1);
INSERT INTO table1(c1,c2,c3) VALUES (NULL,2,2);
```

  NULL 値が使用できるカラムに、ユニークなマルチカラムの **HG** インデックスを作成します。

```
CREATE UNIQUE HG INDEX c1c2_hg2 ON table1(c1,c2);
```

  上記の規則 2 に従って、インデックスのユニークさを確保する必要があります。次の **INSERT** は成功しません。なぜなら、ロー 1 とロー 3 のマルチカラム・インデックス `c1c2_hg2` に同じ値が入るためです。

```
INSERT INTO table1(c1,c2,c3) VALUES (NULL,1,3);
```

  しかし、次の **INSERT** 操作は、規則 1 と規則 3 に従って成功します。

```
INSERT INTO table1(c1,c2,c3) VALUES (NULL,NULL,3);
INSERT INTO table1(c1,c2,c3) VALUES (NULL,NULL,4);
```

  マルチカラム・インデックスでのユニークさが保持されています。
  次の **UPDATE** 操作は成功します。これは、規則 3 によって、複数のローでマルチカラム・インデックスのすべてのカラムに NULL 値を設定できるためです。

```
UPDATE table1 SET c2=NULL WHERE c3=1
```

  マルチカラム **HG** インデックスが一意性制約に制御されている場合は、インデックスに関与しているカラムには NULL 値を設定できません。
- インデックスの並行作成—**BEGIN PARALLEL IQ ... END PARALLEL IQ** 文を使用して、複数の IQ テーブルに対する **CREATE INDEX** 文をグループにまとめること

で、それらの文を 1 つの DDL 文であるかのように実行できます。詳細については、「BEGIN PARALLEL IQ … END PARALLEL IQ 文」を参照してください。

警告！ コミットされていないデータを含むローカル・テンポラリ・テーブルで **CREATE INDEX** コマンドを使用するとエラーになり、次のエラー・メッ セージが表示されます。`"Local temporary table, <tablename>, must be committed in order to create an index."` ローカル・テンポラリ・テーブルのデータをコミットしてからインデックスを作成してください。

『システム管理ガイド： 第 1 巻』の「Sybase IQ インデックス」も参照してください。

関連する動作

- オートコミット。

標準

- SQL—ISO/ANSI SQL 文法のベンダ拡張。
- Sybase—Adaptive Server Enterprise には、Sybase IQ よりも複雑な **CREATE INDEX** 文があります。Adaptive Server Enterprise 構文は Sybase IQ で使用できますが、一部の句やキーワードは無視されます。 Adaptive Server Enterprise**CREATE INDEX** 文の完全な構文については、『Adaptive Server Enterprise リファレンス・マニュアル、第 2 巻： コマンド』を参照してください。

Adaptive Server Enterprise インデックスは、*「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」*のどちらかになります。クラスタード・インデックスを使用すると、ほとんどの場合ノンクラスタード・インデックスよりも速くデータを検索できます。 テーブルごとに 1 つだけクラスタード・インデックスが許可されます。

Sybase IQ は、クラスタード・インデックスをサポートしていません。
CLUSTERED と NONCLUSTERED のキーワードは SQL Anywhere で許可されますが、Sybase IQ では無視されます。*index-type* が指定されない場合、Sybase IQ は指定のカラムに対して HG インデックスを作成します。

Sybase IQ では、DESC キーワードを指定できません。

インデックス名は、Sybase IQ でも Adaptive Server Enterprise でも、指定されたテーブル内でユニークである必要があります。

パーミッション

インデックスを作成するには、指定した DB 領域で DBA 権限または RESOURCE 権限が必要かつ、CREATE 権限が必要になります。

参照：

- BEGIN PARALLEL IQ … END PARALLEL IQ 文 (62 ページ)
- CREATE JOIN INDEX 文 (128 ページ)
- DROP 文 (209 ページ)
- INDEX_PREFERENCE オプション (462 ページ)
- FROM 句 (237 ページ)

CREATE JOIN INDEX 文

ジョイン・インデックスを作成します。ジョイン・インデックスは、特定のカラムを介してジョイン済みのテーブルのグループを定義し、ジョイン操作でそのテーブルを使用するクエリのパフォーマンスを改善します。

構文

```
CREATE JOIN INDEX join-index-name FOR  join-clause
IN dbspace-name
```

パラメータ

- **join-clause：** – [( ] *join-expressionjoin-typejoin-expression* [**ON** *search-condition*] [ )]
- **join-expression：** – { *table-name* | *join-clause* }
- **join-type：** – [**NATURAL**] **FULL** [**OUTER**] **JOIN**
- **search-condition：** – [( ] *search-expression* [**AND***search-expression*] [ )]
- **search-expression：** – [( ] [*table-name.*] *column-name* = [*table-name.*] *column-name* [ )]

例

- **例 1** – `Departments` のプライマリ・キー、`Employees` の外部キーである `DepartmentID` カラムを使用し、`Departments` テーブルと `Employees` テーブルとのジョイン・インデックスを作成します。

  ```
  CREATE JOIN INDEX emp_dept_join
  FOR Departments FULL OUTER JOIN Employees
  ON Departments.DepartmentID = Employees.DepartmentID
  ```

- **例 2** – 将来のデータの割り付けがデフォルトの DB 領域から行われるテーブル `t1` と `t2` を作成し、将来のデータ割り付けが DB 領域 `Dsp6` から行われるジョイン・インデックス `t1t2` を作成します。

  ```
  CREATE TABLE t1(c1 int, c2 char(5));
  CREATE TABLE t2(c1 int, c3 char(5));
  ```

```
CREATE JOIN INDEX t1t2 FOR t1
   FULL OUTER JOIN t2 ON t2.c1=t1.c1 IN Dsp6;
```

使用法

CREATE JOIN INDEX は、指定したテーブルの指定カラム上でジョイン・インデックスを作成します。一度ジョイン・インデックスを作成すると、**DROP JOIN INDEX** を使用して削除する場合や **SYNCHRONIZE JOIN INDEX** を使用して同期する場合を除いて、再び参照することはありません。

この文がサポートするのは **FULL OUTER;** タイプのジョインのみです。**OUTER** キーワードはオプションです。

IN—ジョイン・インデックスの配置を指定します。**IN** 句を省略した場合、Sybase IQ は (オプションの DEFAULT_DBSPACE で指定された) デフォルトの DB 領域にジョイン・インデックスを作成します。

ON—参照するのは、2 つのテーブルのカラムだけです。1 つのカラム・セットは左のサブツリーにある単一のテーブルに属し、もう 1 つのカラム・セットは右のサブツリーにあるテーブルに属しています。サポートされている述部は、等価ジョイン述部のみです。Sybase IQ では、変数が 1 つの述部、カラム内での比較、不等ジョインは使用できません。

ジョイン・インデックスのカラムのデータ型、精度、位取りは同じであることが必要です。

マルチパート・キーを指定するには、2 つのテーブルを結び付ける述部を 2 つ以上含めて、その述部を論理演算子 AND で連結します。 分離した **ON** 句はサポートされていません (すなわち、Sybase IQ ではジョイン述部に論理 OR を使用できません)。 また、**ON** 句は標準の **WHERE** 句を許可しないため、エイリアスを指定することはできません。

ON 句の代わりに NATURAL キーワードを使用できます。NATURAL ジョインは名前によってカラムをペアにする等価ジョインです。NATURAL ジョインが複数のテーブル・ペアに影響する述部を生成する場合、**CREATE JOIN INDEX** はエラーを返します。NATURAL または **ON** のいずれかを指定できますが、両方の指定はできません。

CREATE JOIN INDEX 文はテーブル内でプライマリ・キー対外部キーの関係を探し、1 対多関係の方向を決定します (1 対 1 関係の方向は重要ではありません)。 プライマリ・キーは常に「1」であり、外部キーは常に「多」です。 そのような情報が定義されていない場合、Sybase IQ は左のサブツリーを「1」、右のサブツリーを「多」とみなします。 実際はこれと逆である場合、**CREATE JOIN INDEX** はエラーを返します。

注意： どのジョインでも、クエリの最適化は基礎となるプライマリ・キーに大いに依存します。 このときに外部キーは必要ありません。しかし、外部キーを使用

するとメリットがあります。Sybase IQ は、プライマリ・キーと外部キーの関係を確認するためにロードを設定している場合は外部キーを使用します。

ジョイン・インデックス・テーブルは、Sybase IQ ベース・テーブルである必要があります。テンポラリ・テーブル、リモート・テーブル、またはプロキシ・テーブルにすることはできません。

ベース・テーブルのマルチカラム・インデックスは、そのベース・テーブルを使用して作成されたジョイン・インデックスに複写されません。

スタージョイン・インデックスは、スターの中心にある 1 つのテーブルが、複数のテーブルにジョインされて 1 対多の関係になるインデックスです。スタージョイン・インデックスを定義するには、シングルカラムのキーとプライマリ・キーを定義し、**CREATE JOIN INDEX** 文でキー・ジョインの構文を使用する必要があります。Sybase IQ は、ジョインに複数のジョイン・キー・カラムを使用するスタージョイン・インデックスをサポートしません。

注意： グループ内の他のユーザに対し、データベースの「ジョイン仮想テーブル」へのパーミッションを明示的に与える必要があります。与えない場合、ユーザはジョイン内のテーブルを操作できません。 ジョイン仮想テーブルに対する権限の付与については、『システム管理ガイド：第 1 巻』の「Sybase IQ インデックス」>「ジョイン・インデックスの使用」>「ジョイン・インデックス内のテーブルでの挿入と削除」を参照してください。

『システム管理ガイド：第 1 巻』の「Sybase IQ インデックス」も参照してください。

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 ではサポートされていません。

パーミッション

DBA 権限または RESOURCE 権限を持ち、ジョインの影響を受けるすべてのテーブルの所有者であり、DB 領域で CREATE パーミッションを持っている必要があります。

参照：

- CREATE INDEX 文 (118 ページ)
- CREATE TABLE 文 (166 ページ)

CREATE LOGICAL SERVER 文

ユーザ定義の論理サーバを作成します。

構文

```
CREATE LOGICAL SERVER logical-server-name [MEMBERSHIP
    '(' { ls-member, ...} ')']
```

パラメータ

- **ls-member： – FOR LOGICAL COORDINATOR** | *mpx-server-name*

例

- **例 1 –** 次の例は、メンバとして 3 つのマルチプレックス・ノードを含む、ユーザ定義の論理サーバ *ls1* を作成します。

```
CREATE LOGICAL SERVER ls1 MEMBERSHIP ( n1, n2, n3 )
```

使用法

マルチプレックスのみに該当します。

カタログには、論理サーバとそのメンバシップの定義が格納されています。コーディネータとの論理的なメンバシップを定義するには、MEMBERSHIP 句に FOR LOGICAL COORDINATOR を指定します。

論理サーバの作成時にメンバを指定しないと、論理サーバは空で作成されます。

注意： OPEN 論理サーバや SERVER 論理サーバなどに対する暗黙的な論理サーバ・メンバシップ定義が格納されることはありません。

SYS.ISYSLOGICALSERVER システム・テーブルには、論理サーバについての情報が格納されています。

SYS.ISYSLOGICALMEMBER システム・テーブルには、論理サーバ・メンバシップ定義についての情報が格納されています。

logical-server-name には、次を除く任意のユーザ定義の識別子を指定できます。

- OPEN
- SERVER
- NONE
- DEFAULT

- COORDINATOR
- ALL

ルート論理サーバ・ポリシーの ALLOW_COORDINATOR_AS_MEMBER オプションを ON から OFF に変更しても、カタログに格納されているメンバシップ情報に影響はありません。論理サーバ設定の有効性に影響するのみです。

ALLOW_COORDINATOR_AS_MEMBER が OFF に設定されている場合でも、マルチプレックス・サーバ名を指定するか、FOR LOGICAL COORDINATOR 句を使用して、現在のコーディネータへの論理サーバのメンバシップを定義できます。メンバシップ定義はカタログに格納されますが、マルチプレックス・サーバがコーディネータとして動作している間は非アクティブです。

パーミッション

DBA 権限または MULTIPLEX ADMIN 権限を持っている必要があります。

CREATE LOGIN POLICY 文

ログイン・ポリシーをデータベースに作成します。

構文

CREATE LOGIN POLICY *policy-name policy-options*

パラメータ

- **policy-options：** – *policy-option* [*policy-option*...]
- **policy_option：** – *policy-option-name* =*policy-option-valuepolicy-option-value*={ **UNLIMITED** | *legal-option-value* }

例

- **例 1 –** 次の例では、`Test1` のログイン・ポリシーを作成します。このログイン・ポリシーでは、パスワードは無期限で、アカウントがロックされるまでに許容されるユーザ・パスワードの入力回数が最大 5 回に設定されています。

```
CREATE LOGIN POLICY Test1
password_life_time=UNLIMITED
max_failed_login_attempts=5;
```

使用法

policy-name—ログインポリシーの名前。

policy-option-name—ログイン・ポリシー・オプションの名前。オプションを指定しない場合は、ルート・ログイン・ポリシーから取得した値が適用されます。

policy-option-value—ログイン・ポリシー・オプションに割り当てられた値。UNLIMITED と指定すると、制限は設けられません。

ポリシー・オプションを指定しない場合は、ルート・ログイン・ポリシーからログイン・ポリシーの値が取得されます。

表 6 : ログイン・ポリシー・オプション

オプション	説明	値	ROOT ポリシーの初期値	適用対象
locked	このオプションの値が ON の場合、ユーザが新しい接続を確立するのを禁止します。	ON、OFF	OFF	DBA 権限を持っていないユーザのみ。
max_connections	ユーザに許可された同時接続の最大数。	0 – 2147483647	Unlimited	DBA 権限を持っていないユーザのみ。
max_days_since_login	同一ユーザによる前回のログイン時から次回のログイン時までの最大許容日数。	0 – 2147483647	Unlimited	DBA 権限を持っていないユーザのみ。
max_failed_log-in_attempts	前回の正常なログイン時以降に行った、ユーザ・アカウントがロックアウトされる原因となるログイン試行失敗回数の最大値。	0 – 2147483647	Unlimited	DBA 権限を持っていないユーザのみ。
max_non_dba_con-nections	DBA 権限を持っていないユーザが実行できる同時接続の最大数。このオプションは、ルート・ログイン・ポリシーでのみサポートされています。	0 – 2147483647	Unlimited	DBA 権限を持っていないユーザのみ。ルート・ログイン・ポリシーのみをサポート。
password_expiry_on_next_login	このオプションの値が ON に設定されている場合は、ユーザのパスワードは次回ログイン時に有効期限が切れます。	ON、OFF	OFF	DBA 権限を持つすべてのユーザ。

オプション	説明	値	ROOT ポリシーの初期値	適用対象
password_grace_time	パスワードの有効期限が切れるまでの日数 (この間は、ログインすることはできますが、デフォルトの post_login プロシージャによって警告が発行されます)。	0 – 2147483647	0	DBA 権限を持つすべてのユーザ。
password_life_time	パスワードの変更が必要になるまでの最大日数。	0 – 2147483647	Unlimited	DBA 権限を持つすべてのユーザ。

パーミッション

DBA 権限または USER ADMIN 権限を持っている必要があります。

CREATE MESSAGE 文 [T-SQL]

PRINT 文と **RAISERROR** 文で使用するために、ユーザ定義メッセージを SYSUSERMESSAGES システム・テーブルに追加します。

構文

```
CREATE MESSAGE message-number
... AS 'message-text'
```

使用法

CREATE MESSAGE はメッセージ番号をメッセージ文字列に関連付けます。このメッセージ番号は **PRINT** 文と **RAISERROR** 文で使用できます。

- message_number—追加するメッセージのメッセージ番号。ユーザ定義メッセージのメッセージ番号は 20000 以上にしてください。
- message_text—追加するメッセージのテキスト。最大長は、255 バイトです。**PRINT** 文と **RAISERROR** 文はメッセージ内のプレースホルダを認識して出力します。1 つのメッセージに 20 個までのユニークなプレースホルダを任意の順序で指定することができます。このプレースホルダは、メッセージ・テキストがクライアントに送信されるときに、メッセージに続く引数のフォーマットされた内容に置き換えられます。
  メッセージを別の文法構文の言語に変換するときに引数の順番を変えられるように、プレースホルダには番号が付けられます。引数のプレースホルダは

"%nn!"、すなわちパーセント記号 (%)、1 ～ 20 の整数、感嘆符 (!) の順に表示されます。整数は引数リスト内の引数の位置を表し、たとえば "%1!" は 1 番目の引数、"%2!" は 2 番目の引数などのように表されます。

sp_addmessage の *language* 引数に対応するパラメータはありません。

関連する動作

- オートコミット。

標準

- SQL—ISO/ANSI SQL 文法のベンダ拡張。
- Sybase—**CREATE MESSAGE** の機能は、Adaptive Server Enterprise の **sp_addmessage** プロシージャによって提供されています。

パーミッション

RESOURCE 権限が必要です。

参照：

- PRINT 文 [T-SQL] (311 ページ)
- RAISERROR 文 [T-SQL] (315 ページ)

CREATE MULTIPLEX SERVER 文

マルチプレックス・サーバを作成します。

構文

変数宣言：

```
CREATE MULTIPLEX SERVER server-name DATABASE 'dbfile
' host-port list [ ROLE {  READER |  WRITER  } ] [  STATUS | {
INCLUDED |  EXCLUDED } ]
```

パラメータ

- **host-port-list： –** {[**PRIVATE**] **HOST** ' *hostname* ' **PORT** *port number* }

例

- **例 1 –** 「マルチプレックス・サーバの追加 (Interactive SQL)」を参照してください。

使用法

マルチプレックスのみに該当します。

サーバ・パスに UNIX のソフト・リンク (シンボリック・リンク) を使用することを計画している場合、ソフト・リンクを作成してから **CREATE MULTIPLEX SERVER** を実行します。新しいサーバを起動する場合、データベース・ファイル・パスは、サーバを作成したときに指定したデータベース・ファイル・パスに一致している必要があります。

サーバ起動オプション **-n** の規則に従って、マルチプレックス・サーバの名前 (*server-name*) を選択します。『ユーティリティ・ガイド』の「第 1 章 データベース・サーバの実行」>「データベース・サーバの起動」を参照してください。

最初のマルチプレックス・サーバを作成すると、コーディネータ・ノードとセカンダリ・ノードのローが SYS.ISYSIQMPXSERVER に追加されます。トランザクション・ログは、この操作を 2 つの異なる **CREATE MULTIPLEX SERVER** コマンドとして記録します。1 つはコーディネータ・ノード用で、他の 1 つはセカンダリ・ノード用です。

SYS.ISYSIQMPXSERVER システム・テーブルには、HOST ホスト名 PORT ポート名のペアが host:port[;host:port…] の形式で connection_info 文字列に格納されています。

注意： マルチプレックス・サーバを実行中のコンピュータに、異なるネットワーク・アドレスにマップされている複数の冗長ネットワーク・カードがある場合は、複数の host:port ペアを使用してください。

DATABASE、host-port list、ROLE、STATUS の各句は、どのような順序で指定してもかまいません。デフォルトの ROLE は READER です。デフォルトの STATUS は INCLUDED です。

host-port-list のキーワード PRIVATE は、特定の HOST PORT ペアがプライベート相互接続用であることを指定します。MIPC 用の個別のプライベート相互接続を使用することにより、可用性が高く、パフォーマンスに優れたネットワーク設定が実現できます。Sybase IQSybase IQ は、プライベート・ポートを自動的に開きます。サーバを起動するために使用される host-port-list にリストする必要はありません。 すべてのパブリック・ポートとプライベート・ポートは、競合を避けるために、ユニークなポート番号を必要とします。

サーバの追加時にはコーディネータが実行中である必要がありますが、**CREATE MULTIPLEX SERVER** コマンドはマルチプレックス内のいずれのサーバからでも実行できます。

この文は自動的にコミットされます。

パーミッション

DBA 権限または MULTIPLEX ADMIN 権限を持っている必要があります。

CREATE PROCEDURE 文

データベースに新しいユーザ定義の SQL プロシージャを作成します。

外部プロシージャ・インタフェースを作成する方法については、「CREATE PROCEDURE 文 (外部プロシージャ)」を参照してください。

構文

```
CREATE [ OR REPLACE | TEMPORARY ] PROCEDURE [ owner.]procedure-name
( [ parameter, …] ) {
[ RESULT ( result-column, …) | NO RESULT SET ]
[ SQL SECURITY { INVOKER | DEFINER } ]
[ ON EXCEPTION RESUME ] compound statement | AT location-string
```

パラメータ

- **parameter：** – *parameter_mode*_parameter-name_*data-type* [**DEFAULT** *expression*] | **SQLCODE** | **SQLSTATE**
- **parameter_mode：** – **IN** | **OUT** | **INOUT**
- **result-column：** – *column-name*_data-type_

例

- **例 1** – 次のプロシージャは、CASE 文を使用してクエリ結果を分類します。

```
CREATE PROCEDURE ProductType (IN product_id INT, OUT type
CHAR(10))
BEGIN
  DECLARE prod_name CHAR(20) ;
  SELECT name INTO prod_name FROM "GROUPO"."Products"
  WHERE ID = product_id;
  CASE prod_name
  WHEN 'Tee Shirt' THEN
    SET type = 'Shirt'
  WHEN 'Sweatshirt' THEN
    SET type = 'Shirt'
  WHEN 'Baseball Cap' THEN
    SET type = 'Hat'
  WHEN 'Visor' THEN
    SET type = 'Hat'
  WHEN 'Shorts' THEN
    SET type = 'Shorts'
  ELSE
    SET type = 'UNKNOWN'
```

```
  END CASE ;
END
```

- **例 2 –** 次のプロシージャはカーソルのロー上でカーソルとループを使用して、単一の値を返します。

```
CREATE PROCEDURE TopCustomer (OUT TopCompany CHAR(35), OUT
TopValue INT)
BEGIN
  DECLARE err_notfound EXCEPTION
     FOR SQLSTATE '02000' ;
  DECLARE curThisCust CURSOR FOR
  SELECT CompanyName, CAST(       sum(SalesOrderItems.Quantity *
  Products.UnitPrice) AS INTEGER) VALUE
  FROM Customers
  LEFT OUTER JOIN SalesOrders
  LEFT OUTER JOIN SalesorderItems
  LEFT OUTER JOIN Products
  GROUP BY CompanyName ;


  DECLARE ThisValue INT ;
  DECLARE ThisCompany CHAR(35) ;
  SET TopValue = 0 ;
  OPEN curThisCust ;
  CustomerLoop:
  LOOP
    FETCH NEXT curThisCust
    INTO ThisCompany, ThisValue ;
    IF SQLSTATE = err_notfound THEN
      LEAVE CustomerLoop ;
    END IF ;
    IF ThisValue > TopValue THEN
      SET TopValue = ThisValue ;
      SET TopCompany = ThisCompany ;
      END IF ;
  END LOOP CustomerLoop ;
  CLOSE curThisCust ;
END
```

使用法

CREATE PROCEDURE はデータベースにプロシージャを作成します。DBA 権限を持つユーザは、**owner** を指定することにより他のユーザのプロシージャを作成できます。プロシージャは **CALL** 文で呼び出します。

注意： ストアド・プロシージャは、 ISO/ANSI SQL と T-SQL という 2 とおりの方法で作成できます。たとえば、**CREATE PROCEDURE** 構文を使用する場合、**BEGIN TRANSACTION** は T-SQL でしか使用できません。ストアド・プロシージャを作成する際は、2 種類の構文を混ぜて使わないでください。「CREATE PROCEDURE 文 [T-SQL]」を参照してください。

CREATE PROCEDURE—永続的なストアド・プロシージャまたはテンポラリ (TEMPORARY) ストアド・プロシージャを作成できます。PROC を PROCEDURE の同義語として使用できます。

パラメータ名は、カラム名などの他のデータベース識別子に関するルールに従って付ける必要があり、さらに有効な SQL データ型である必要があります。『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。キーワードには次の意味があります。

パラメータは、IN、OUT、INOUT のキーワードのいずれかをプレフィクスにできます。キーワードを指定しない場合、パラメータはデフォルトで INOUT になります。キーワードには次の意味があります。

- IN—このパラメータは、プロシージャに値を渡す式です。
- OUT—このパラメータは、プロシージャから値を与えられる変数です。
- INOUT—このパラメータはプロシージャに値を与え、プロシージャから新しい値を受け取ることがある変数です。

CALL を使用してプロシージャを実行する場合、必ずしもすべてのパラメータを指定する必要はありません。**CREATE PROCEDURE** 文の中にデフォルト値がある場合、不明のパラメータにデフォルト値を割り当てます。 **CALL** に引数が指定されておらず、デフォルトも設定されていない場合には、エラーが発生します。

SQLSTATE と **SQLCODE** は、プロシージャが終了するときに、**SQLSTATE** 値または **SQLCODE** 値を出力する特別なパラメータです (**OUT** パラメータです)。**SQLSTATE** パラメータと **SQLCODE** パラメータを指定しても、指定しなくても、**SQLSTATE** と **SQLCODE** の特別値をプロシージャ・コールのすぐ後でチェックして、プロシージャのリターン・ステータスをテストできます。

SQLSTATE と **SQLCODE** の特別値は、その次の SQL 文によって変更されます。**SQLSTATE** または **SQLCODE** をプロシージャ引数として与えると、リターン・コードは変数の中に格納されます。

CREATE OR REPLACE PROCEDURE を指定すると、新しいプロシージャが作成されるか、同じ名前の既存のプロシージャが置き換えられます。この句によって、プロシージャの定義は変更されますが、既存のパーミッションは維持されます。**OR REPLACE** をテンポラリ・プロシージャで使用することはできません。また、置き換えられるプロシージャがすでに使用されている場合は、エラーが返されます。

CREATE TEMPORARY PROCEDURE を指定すると、作成した接続でのみ参照できる関数になり、接続を削除すると関数も自動的に削除されます。テンポラリ・ストアド・プロシージャは明示的に削除することもできます。テンポラリ・ストアド・プロシージャに対して **ALTER**、**GRANT**、または **REVOKE** は実行できません。また他の関数とは異なり、テンポラリ・ストアド・プロシージャはカタログやトランザクション・ログに記録されていません。

テンポラリ・プロシージャは、作成者 (現在のユーザ) または指定された所有者のパーミッションで実行されます。次の場合に、テンポラリ・プロシージャの所有者を指定できます。

- テンポラリ・プロシージャが永続的なストアド・プロシージャ内に作成されている。
- テンポラリ・プロシージャと永続的なプロシージャの所有者が同じである。

テンポラリ・プロシージャの所有者を削除するには、最初にテンポラリ・プロシージャを削除します。

テンポラリ・ストアド・プロシージャを作成および削除できるのは、読み込み専用データベースに接続されている場合、つまり、外部プロシージャにすることができない場合です。

たとえば、次のテンポラリ・プロシージャは `CustRank` というテーブルがあればそれを削除します。この例では、プロシージャで、テーブル名がユニークであり、プロシージャの作成者がテーブルの所有者を指定しなくても参照できることを想定しています。

```
CREATE TEMPORARY PROCEDURE drop_table( IN @TableName char(128) )
BEGIN
    IF EXISTS  ( SELECT * FROM SYS.SYSTAB WHERE
    table_name = @TableName )
    THEN EXECUTE IMMEDIATE
    'DROP TABLE "' || @TableName || '"';
    MESSAGE 'Table "' || @TableName ||
    '" dropped' to client;
    END IF;
END;
CALL drop_table( 'CustRank' )
```

RESULT—結果セットのカラムの数と型を宣言します。**RESULT** キーワードに続くカッコで囲まれたリストは、結果カラムの名前と型を定義します。**CALL** 文が記述されていると、この情報を Embedded SQL **DESCRIBE** または ODBC **SQLDescribeCol** が返します。許容されるデータ型については、『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。

プロシージャから返される結果セットの詳細については、『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。

一部のプロシージャでは、実行方法によっては複数の結果セットが生成される場合があります。たとえば、次のプロシージャは、2 カラムを返す場合も、1 カラムを返す場合もあります。

```
CREATE PROCEDURE names( IN formal char(1))
BEGIN
  IF formal = 'n' THEN
    SELECT GivenName
    FROM Employees
```

```
  ELSE
    SELECT Surname,GivenName
    FROM Employees
  END IF
END
```

これらの結果セット・プロシージャは **RESULT** 句を指定しないで記述するか、Transact-SQL で記述します。これらの使用には、次の制約があります。

- Embedded SQL—正しい形式の結果セットを取得するには、結果セットのカーソルが開かれてからローが返されるまでの間に、プロシージャ・コールを記述 (**DESCRIBE**) します。**DESCRIBE** 文に、**CURSOR** *cursor-name* 句が必要です。
- ODBC、OLE DB、ADO.NET—ODBC アプリケーションでは、変数結果セット・プロシージャを使用できます。結果セットの記述は、ドライバまたはプロバイダによって実行されます。
- Open Client アプリケーション—Open Client アプリケーションでは、変数結果セット・プロシージャを使用できます。

プロシージャが返す結果セットが 1 つだけの場合は、**RESULT** 句を使用する必要があります。この句を使用すると、カーソルがオープンした後で ODBC と Open Client アプリケーションが結果セットを記述し直すのを防ぐことができます。

複数の結果セットを処理するには、ODBC はプロシージャが定義した結果セットではなく、現在実行中のカーソルを記述する必要があります。したがって、ODBC はいつもプロシージャ定義の **RESULT** 句内で定義されているカラム名を記述するわけではありません。この問題を回避するには、結果セットを生成する **SELECT** 文でカラム・エイリアスを使用します。

NO RESULT SET—プロシージャが結果セットを返さないことを宣言します。これは、プロシージャが結果セットを返さないことを外部環境が認識する必要がある場合に便利です。

SQL SECURITY—INVOKER (プロシージャを呼び出しているユーザ) または DEFINER (プロシージャを所有しているユーザ) のどちらとしてプロシージャが実行されるかを定義します。デフォルトは DEFINER です。

SQL SECURITY INVOKER を指定すると、プロシージャを呼び出す各ユーザに対して注釈を付ける必要があるため、追加のメモリが使用されます。また、名前解決も INVOKER として実行されます。そのため、適切な所有者名で、すべてのオブジェクト名 (テーブル、プロシージャなど) を修飾します。たとえば、`user1` が次のプロシージャを作成したとします。

```
CREATE PROCEDURE user1.myProcedure()
   RESULT( columnA INT )
   SQL SECURITY INVOKER
   BEGIN
       SELECT columnA FROM table1;
   END;
```

user2 がこのプロシージャを実行しようとして、user2.table1 テーブルが存在しない場合、テーブルのルックアップでエラーが発生します。さらに、user2.table1 が存在する場合は、意図した user1.table1 ではなくそのテーブルが使用されます。この状況を防ぐため、文ではテーブル参照を修飾してください (table1 だけでなく user1.table1 とします)。

ON EXCEPTION RESUME を使用すると、プロシージャは **ON_TSQL_ERROR** オプションの設定に応じたアクションを実行します。**ON_TSQL_ERROR** を **CONDITIONAL** (デフォルト) に設定すると、次の文がエラーを処理する場合は実行が継続され、そうでない場合は終了します。

エラー処理文には次の文があります。

- **IF**
- **SELECT** @variable =
- **CASE**
- **LOOP**
- **LEAVE**
- **CONTINUE**
- **CALL**
- **EXECUTE**
- **SIGNAL**
- **RESIGNAL**
- **DECLARE**
- **SET VARIABLE**

ON EXCEPTION RESUME 句では、明示的なエラー処理コードを使用しないでください。

「ON_TSQL_ERROR オプション [TSQL]」を参照してください。

AT location-string—*location-string* で指定されたリモート・プロシージャのために、現在のデータベースに「プロキシ・ストアド・プロシージャ」を作成します。**AT** 句は、*location-string* 内のフィールド・デリミタとしてセミコロン (;) をサポートします。セミコロンがない場合は、ピリオドがフィールド・デリミタです。ピリオドを使用すると、データベースおよび所有者の各フィールドにファイル名と拡張子を使用できます。

リモート・プロシージャが出力変数で返すことのできる文字数は最大 254 です。

リモート・プロシージャで結果セットが返される可能性がある場合は (すべての場合で返されなくとも)、ローカル・プロシージャ定義に RESULT 句が含まれている必要があります。

リモート・サーバについては、「CREATE SERVER 文」を参照してください。リモート・プロシージャの使用については、『システム管理ガイド： 第 2 巻』の

「リモート・データへのアクセス」>「Sybase IQ とリモート・データ」>「リモート・プロシージャ・コール (RPC)」を参照してください。

注意： プロシージャが削除されて作成されると、Sybase IQ 12.6 より前に作成されたデータベースは、最終的に proc_id の上限である 32767 に達し、**CREATE PROCEDURE** が Sybase IQ 12.6 で "`Item already exists`" エラーを返す場合があります。対処方法については、『システム管理ガイド： 第 1 巻』の「トラブルシューティングのヒント」>「状況別の解決策」>「リソースの問題」>「不十分なプロシージャ識別子」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—Transact-SQL の **CREATE PROCEDURE** 文は異なります。
- SQLJ—Java 結果セットの構文拡張は、推奨される SQLJ1 規格に指定されています。

パーミッション

テンポラリ・プロシージャを作成する場合を除き、RESOURCE 権限を持っている必要があります。外部プロシージャの場合、または別のユーザ用にプロシージャを作成する場合は、DBA 権限を持っている必要があります。

参照：

- プロシージャ内でのテンポラリ・テーブルの参照 (144 ページ)
- BEGIN … END 文 (60 ページ)
- CALL 文 (66 ページ)
- CREATE PROCEDURE 文 [T-SQL] (144 ページ)
- CREATE PROCEDURE 文 (外部プロシージャ) (146 ページ)
- CREATE SERVER 文 (161 ページ)
- DROP 文 (209 ページ)
- EXECUTE IMMEDIATE 文 [ESQL] [SP] (226 ページ)
- GRANT 文 (247 ページ)
- RAISERROR 文 [T-SQL] (315 ページ)
- ON_TSQL_ERROR オプション [TSQL] (500 ページ)

プロシージャ内でのテンポラリ・テーブルの参照

プロシージャ間でテンポラリ・テーブルを共有すると、テーブル定義が矛盾している場合に問題が発生する場合があります。

たとえば、procA と procB という 2 つのプロシージャがあり、その両方がテンポラリ・テーブル temp_table を定義して、sharedProc という名前の別のプロシージャを呼び出すとします。procA と procB のどちらもまだ呼び出されていないため、テンポラリ・テーブルはまだ存在しません。

ここで、procA の temp_table の定義が procB の定義と少し異なるとします。両方とも同じカラム名と型を使用していますが、カラムの順序が異なります。

procA を呼び出すと、予期した結果が返されます。一方で、procB を呼び出すと、異なる結果が返されます。

これは、procA が呼び出されたときに temp_table が作成され、その後で sharedProc が呼び出されたためです。sharedProc が呼び出されたときに、内部の **SELECT** 文が解析されて検証され、その後で、別の **SELECT** 文が実行されたときに再び使用できるように、解析された文の表現がキャッシュされます。キャッシュされたバージョンは、procA のテーブル定義のカラムの順序を反映しています。

procB を呼び出すと temp_table が再作成されますが、カラムの順序が異なります。procB が sharedProc を呼び出すと、データベース・サーバは **SELECT** 文のキャッシュされた表現を使用します。そのため、結果が異なります。

次のいずれかを実行すると、このような状況の発生を防ぐことができます。

- このような方法で使用されるテンポラリ・テーブルは、一致するように定義する。
- 代わりにグローバル・テンポラリ・テーブルを使用することを検討する。

CREATE PROCEDURE 文 [T-SQL]

Adaptive Server Enterprise 互換の方法で、プロシージャを作成します。

構文

次の Transact-SQL の **CREATE PROCEDURE** 文のサブセットは Sybase IQ でサポートされています。

```
CREATE [ OR REPLACE ] PROCEDURE [ owner.]procedure_name
… [ [ ( ] @parameter_name data-type [ = default ] [ OUTPUT ] [ , … ]
[ ) ] ]
…[ WITH RECOMPILE ]
```

```
… AS
… statement-list
```

使用法

Transact-SQL の文と Sybase IQ SQL の文の相違点：

- **CREATE OR REPLACE PROCEDURE** を指定すると、新しいプロシージャが作成されるか、同じ名前の既存のプロシージャが置き換えられます。 この句によって、プロシージャの定義は変更されますが、既存のパーミッションは維持されます。 また、置き換えられるプロシージャがすでに使用されている場合は、エラーが返されます。
- @ をプレフィクスとする変数名—"@" は Transact-SQL 変数名であることを示します。Sybase IQ では、変数に有効なすべての識別子を使用でき、@ のプレフィクスも使用可能です。
- 入出力パラメータ—Sybase IQ のプロシージャ・パラメータは **IN**、**OUT**、または **INOUT** で指定されますが、Transact-SQL のプロシージャ・パラメータはデフォルトの **INPUT** パラメータか **OUTPUT** パラメータで指定されます。 Sybase IQ で **INOUT** または **OUT** で宣言されるパラメータは、Transact-SQL では **OUTPUT** で宣言する必要があります。
- パラメータのデフォルト値—Sybase IQ のプロシージャ・パラメータは、キーワード DEFAULT を使用してデフォルト値を設定します。Transact-SQL では等号記号 (=) を使用してデフォルト値を設定します。
- 返される結果セット—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を行った環境に返されます。

```
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 Sybase IQ のプロシージャ：

```
CREATE PROCEDURE showdept(in deptname
      varchar(30) )
RESULT ( lastname char(20), firstname char(20))
ON EXCEPTION RESUME
BEGIN
  SELECT Employees.SurName, Employees.GivenName
  FROM Departments, Employees
  WHERE Departments.DepartmentName = deptname
  AND Departments.DepartmentID =
        Employees.DepartmentID
END
```

- プロシージャの本文—Transact-SQL プロシージャの本文は、AS キーワードをプレフィクスとして付けた Transact-SQL 文のリストです。 Sybase IQ プロシージャの本文は、**BEGIN** と **END** キーワードで囲まれた複合文です。

注意： ストアド・プロシージャは、 T-SQL と SQL/92 という 2 とおりの方法で作成できます。たとえば、**CREATE PROCEDURE** 構文を使用する場合、**BEGIN TRANSACTION** は T-SQL でしか使用できません。ストアド・プロシージャを作成する際は、2 種類の構文を混ぜて使わないでください。

関連する動作

- オートコミット。

標準

- SQL—ISO/ANSI SQL 文法の Transact-SQL 拡張。
- Sybase—Sybase IQ は、Adaptive Server Enterprise の **CREATE PROCEDURE** 文の構文のサブセットをサポートしています。
  Transact-SQL の **WITH RECOMPILE** オプション句があっても無視されます。SQL Anywhere は、データベース起動後に初めてプロシージャを実行したときに、常にプロシージャを再コンパイルし、コンパイルしたプロシージャをデータベースが停止するまで保管します。
  プロシージャのグループはサポートされません。

パーミッション

RESOURCE 権限が必要です。

参照：

- CREATE PROCEDURE 文 (137 ページ)

CREATE PROCEDURE 文 (外部プロシージャ)

ネイティブ・プロシージャまたは外部プロシージャへのインタフェースを作成します。

SQL プロシージャの作成方法については、「CREATE PROCEDURE 文」を参照してください。 テーブル UDF または TPF の作成方法については、「CREATE PROCEDURE 文 (テーブル UDF)」を参照してください。

構文

```
CREATE [ OR REPLACE ] PROCEDURE [ owner.]procedure-name ( [ parameter, …] )
```

```
[ RESULT ( result-column, …) | NO RESULT SET ]
[   DYNAMIC RESULT SETS integer-expression ]
 [ SQL SECURITY { INVOKER | DEFINER } ]
[ EXTERNAL NAME 'external-call' [ LANGUAGE environment-name ]
```

パラメータ

- **parameter：** – *parameter_modeparameter-namedata-type* [**DEFAULT***expression*] | **SQLCODE** | **SQLSTATE**
- **parameter_mode：** – **IN** | **OUT** | **INOUT**
- **result-column：** – *column-namedata-type*
- **environment-name：** – **C_ESQL32** | **C_ESQL64** | **C_ODBC32** | **C_ODBC64** | **CLR** | **JAVA** | **PERL** | **PHP**

使用法

プロシージャの本文は複合文で構成されます。複合文の詳細については、「BEGIN … END 文」を参照してください。

注意： ストアド・プロシージャは、 ISO/ANSI SQL と T-SQL という 2 とおりの方法で作成できます。たとえば、**CREATE PROCEDURE** 構文を使用する場合、**BEGIN TRANSACTION** は T-SQL でしか使用できません。ストアド・プロシージャを作成する際は、2 種類の構文を混ぜて使わないでください。「CREATE PROCEDURE 文 [T-SQL]」を参照してください。

CREATE PROCEDURE はデータベースにプロシージャを作成します。DBA 権限を持つユーザは、owner を指定することにより他のユーザのプロシージャを作成できます。プロシージャは **CALL** 文で呼び出します。

ストアド・プロシージャによって結果セットが返された場合は、出力パラメータを設定したり、戻り値を返したりすることもできません。

複数のプロシージャからテンポラリ・テーブルを参照しているときに、テンポラリ・テーブルの定義が矛盾していて、テーブルを参照している文がキャッシュされている場合、問題が発生する可能性があります。

さまざまなプログラミング言語で記述された外部プロシージャまたはネイティブ・プロシージャを呼び出す永続的なストアド・プロシージャを作成できます。**PROC** を **PROCEDURE** の同意語として使用できます。

- **パラメータ名** – パラメータ名は、カラム名など他のデータベース識別子に関するルールに従って付けてください。 パラメータ名は、有効な SQL データ型である必要があります。 詳細については、『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。

パラメータは、**IN**、**OUT**、または **INOUT** のキーワードのいずれかをプレフィクスにできます。 これらの値のどれも指定しない場合、パラメータはデフォルトで INOUT になります。 キーワードには次の意味があります。

- IN—このパラメータは、プロシージャに値を渡す式です。
- OUT—このパラメータは、プロシージャから値を与えられる変数です。
- INOUT—このパラメータはプロシージャに値を与え、プロシージャから新しい値を受け取ることがある変数です。

注意： TABLE パラメータは、INOUT または OUT として宣言できません。「CREATE PROCEDURE 文 (テーブル UDF)」を参照してください。

CALL を使用してプロシージャを実行する場合、必ずしもすべてのパラメータを指定する必要はありません。 **CREATE PROCEDURE** 文の中にデフォルト値がある場合、不明のパラメータにデフォルト値を割り当てます。 **CALL** 文に引数が指定されておらず、デフォルトも設定されていない場合には、エラーが発生します。

注意： テーブル UDF を呼び出すことはできません。「CREATE PROCEDURE 文 (テーブル UDF)」を参照してください。

SQLSTATE と **SQLCODE** は、プロシージャが終了するときに、**SQLSTATE** 値または **SQLCODE** 値を出力する特別なパラメータ (**OUT** パラメータ) です。 **SQLSTATE** パラメータと **SQLCODE** パラメータを指定しても、指定しなくても、**SQLSTATE** と **SQLCODE** の特別値をプロシージャ・コールのすぐ後でチェックして、プロシージャのリターン・ステータスをテストできます。

SQLSTATE と **SQLCODE** の特別値は、その次の SQL 文によって変更されます。 **SQLSTATE** または **SQLCODE** をプロシージャ引数として与えると、リターン・コードは変数の中に格納されます。

- **OR REPLACE – OR REPLACE** (**CREATE OR REPLACE PROCEDURE**) を指定すると、新しいプロシージャが作成されるか、同じ名前の既存のプロシージャが置き換えられます。 この句によって、プロシージャの定義は変更されますが、既存のパーミッションは維持されます。置き換えられるプロシージャがすでに使用されている場合は、エラーが返されます。

  TEMPORARY 外部呼び出しプロシージャは作成できません。

  プロシージャから返される結果セットの詳細については、『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。
- **RESULT –** 結果セットのカラムの数と型を宣言します。**RESULT** キーワードに続くカッコで囲まれたリストは、結果カラムの名前と型を定義します。 **CALL** 文が記述されていると、この情報を Embedded SQL **DESCRIBE** または ODBC **SQLDescribeCol** が返します。 許容されるデータ型については、『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。

Embedded SQL (LANGUAGE C_ESQL32、LANGUAGE C_ESQL64) または ODBC (LANGUAGE C_ODBC32、LANGUAGE C_ODBC64) の外部関数を呼び出すプロシージャでは、0 または 1 つの結果セットが返される場合があります。

Perl または PHP (LANGUAGE PERL、LANGUAGE PHP) の外部関数を呼び出すプロシージャでは、結果セットが返されることはありません。 データベース・サーバによってロードされたネイティブ関数を呼び出すプロシージャでは、結果セットが返されることはありません。

CLR または Java (LANGUAGE CLR、LANGUAGE JAVA) の外部関数を呼び出すプロシージャでは、0、1 つ、またはそれ以上の結果セットが返される場合があります。

- **NO RESULT SET –** プロシージャが結果セットを返さないことを宣言します。これは、プロシージャが結果セットを返さないことを外部環境が認識する必要がある場合に便利です。
- **DYNAMIC RESULT SETS –** LANGUAGE CLR 呼び出しおよび LANGUAGE JAVA 呼び出しでこの句を使用します。 この句は、LANGUAGE を指定した場合にのみ意味を持ちます。RESULT 句を指定した場合、DYNAMIC RESULT SETS はデフォルトで 1 に設定されます。RESULT 句を指定しない場合、DYNAMIC RESULT SETS はデフォルトで 0 に設定されます。Perl または PHP (LANGUAGE PERL、LANGUAGE PHP) の外部関数を呼び出すプロシージャでは、結果セットが返されることはない点に注意してください。データベース・サーバによってロードされたネイティブ関数を呼び出すプロシージャでは、結果セットが返されることはありません。
- **SQL SECURITY –** INVOKER (プロシージャを呼び出しているユーザ) または DEFINER (プロシージャを所有しているユーザ) のどちらとしてプロシージャが実行されるかを定義します。デフォルトは DEFINER です。 外部呼び出しの場合は、この句によって、外部環境での非修飾オブジェクト参照の所有権のコンテキストが確立されます。

  SQL SECURITY INVOKER が指定されていると、プロシージャを呼び出す各ユーザに対して注釈を付ける必要があるため、必要なメモリも多くなります。また、SQL SECURITY INVOKER が指定されていると、名前解決も INVOKER として実行されます。 したがって、適切な所有者を使用して、すべてのオブジェクト名 (テーブルやプロシージャなど) の条件を満たすように注意してください。 たとえば、`user1` が次のプロシージャを作成したとします。

```
CREATE PROCEDURE user1.myProcedure()
    RESULT( columnA INT )
    SQL SECURITY INVOKER
    BEGIN
```

```
        SELECT columnA FROM table1;
    END;
```

user2 がこのプロシージャを実行しようとして、user2.table1 テーブルが存在しない場合、テーブルのルックアップでエラーが発生します。 さらに、user2.table1 が存在する場合は、意図した user1.table1 ではなくそのテーブルが使用されます。 この状況を防ぐため、文ではテーブル参照を修飾してください (table1 だけでなく user1.table1 とします)。

- **EXTERNAL NAME LANGUAGE JAVA –** EXTERNAL NAME LANGUAGE 'native-call' *native-call*:[*operating-system*:]*function-name@library*; ...

  LANGUAGE JAVA 句が付いた **EXTERNAL NAME** を使用するプロシージャは、Java メソッドのラッパです。

  operating-system： UNIX—LANGUAGE 属性を指定しないで **EXTERNAL NAME** 句を使用したプロシージャでは、C などのプログラミング言語で記述されたネイティブ関数へのインタフェースが定義されます。ネイティブ関数は、データベース・サーバによってそのアドレス空間にロードされます。

  ライブラリ名にはファイル拡張子が付く場合があります。この拡張子は通常、Windows では .dll、UNIX では .so です。 拡張子が付いていなければ、ソフトウェアがプラットフォーム固有のデフォルトのファイル拡張子をライブラリに付加します。 形式の例を次に示します。

  ```
  CREATE PROCEDURE mystring( IN instr LONG VARCHAR )
  EXTERNAL NAME
  'mystring@mylib.dll;Unix:mystring@mylib.so';
  ```

  プラットフォーム固有のデフォルトを使用して、前述の **EXTERNAL NAME** 句をより簡単に記述する方法を次に示します。

  ```
  CREATE PROCEDURE mystring( IN instr LONG VARCHAR )
  EXTERNAL NAME 'mystring@mylib';
  ```

  関数が含まれているライブラリは、呼び出されると、データベース・サーバのアドレス空間にロードされます。 ネイティブ関数は、サーバの一部として実行されます。 この場合、関数によって障害が引き起こされると、データベース・サーバは終了します。 このため、LANGUAGE 属性を使用して外部環境で関数をロードし、実行することをおすすめします。 外部環境で関数によって障害が引き起こされた場合、データベース・サーバは動作し続けます。

  ネイティブ・ライブラリの呼び出しについては、『SQL Anywhere サーバー - プログラミング』の「SQL Anywhere 外部呼び出しインターフェイス」を参照してください。

注意： このリファレンスは SQL Anywhere マニュアルにリンクされています。

EXTERNAL NAME LANGUAGE 'c-call' LANGUAGE { C_ESQL32 | C_ESQL64 | C_ODBC32 | C_ODBC64 } *c-call*:

[*operating-system*:]*function-name@library*; ...

operating-system: **UNIX**

コンパイルされたネイティブ C 関数をデータベース・サーバ内ではなく外部環境で呼び出すには、**EXTERNAL NAME** 句の後に C_ESQL32、C_ESQL64、C_ODBC32、または C_ODBC64 のいずれかを指定した LANGUAGE 属性を続けて、ストアド・プロシージャまたは関数を定義します。

LANGUAGE 属性を指定すると、関数が含まれているライブラリが外部プロセスによってロードされ、外部関数がその外部プロセスの一部として実行されます。この場合、関数によって障害が引き起こされても、データベース・サーバは動作し続けます。

プロシージャ定義の例：

```
CREATE PROCEDURE ODBCinsert(
IN ProductName CHAR(30),
IN ProductDescription CHAR(50)
)
NO RESULT SET
EXTERNAL NAME 'ODBCexternalInsert@extodbc.dll'
LANGUAGE C_ODBC32;
```

『SQL Anywhere サーバー – プログラミング』の「SQL Anywhere 外部環境のサポート」>「ESQL 外部環境と ODBC 外部環境」を参照してください。

注意： このリファレンスは SQL Anywhere マニュアルにリンクされています。

EXTERNAL NAME 'clr-call' LANGUAGE CLR *clr-call* : *dll-name*:: *function-name* (*param-type-1*, ...)

operating-system: **UNIX**

.NET 関数を外部環境で呼び出すには、プロシージャ・インタフェースを **EXTERNAL NAME** 句で定義し、それに続いて LANGUAGE CLR 属性を指定します。

CLR ストアド・プロシージャまたは関数の動作は、SQL ストアド・プロシージャまたは関数と同じです。ただし、プロシージャまたは関数のコードは C# または Visual Basic などの .NET 言語で記述され、その実行はデータベース・サーバの外側 (つまり別の .NET 実行ファイル内) で行われます。

プロシージャ定義の例：

```
CREATE PROCEDURE clr_interface(
IN p1 INT,
IN p2 UNSIGNED SMALLINT,
```

```
OUT p3 LONG VARCHAR)
NO RESULT SET
EXTERNAL NAME 'CLRlib.dll::CLRproc.Run( int, ushort, out
string )'
LANGUAGE CLR;
```

『SQL Anywhere サーバー – プログラミング』の「SQL Anywhere 外部環境のサポート」>「CLR 外部環境」を参照してください。

注意： このリファレンスは SQL Anywhere マニュアルにリンクされています。

EXTERNAL NAME 'perl-call' LANGUAGE CLR *perl-call*:

< **file=***perl-call* > **$sa_perl_return=***perl-sub* (**$sa_perl_arg0**, ...)

Perl 関数を外部環境で呼び出すには、**EXTERNAL NAME** 句の後に LANGUAGE PERL 属性を続けて、プロシージャのインタフェースを定義します。

Perl のストアド・プロシージャまたは関数は、SQL のストアド・プロシージャまたは関数と同じように動作します。ただし、例外として、プロシージャまたは関数のコードは Perl で記述され、プロシージャまたは関数の実行は、データベース・サーバの外部 (つまり、Perl 実行プログラム・インスタンス内) で行われます。

プロシージャ定義の例：

```
CREATE PROCEDURE PerlWriteToConsole( IN str LONG VARCHAR)
NO RESULT SET
EXTERNAL NAME '<file=PerlConsoleExample>
WriteToServerConsole( $sa_perl_arg0 )'
LANGUAGE PERL;
```

『SQL Anywhere サーバー – プログラミング』の「SQL Anywhere 外部環境のサポート」>「PERL 外部環境」を参照してください。

注意： このリファレンスは SQL Anywhere マニュアルにリンクされています。

EXTERNAL NAME 'perl-call' LANGUAGE PHP <**file=***php-file*> **print** *php-func*(**$argv[1]**, ...)

PHP 関数を外部環境で呼び出すには、**EXTERNAL NAME** 句の後に LANGUAGE PHP 属性を続けて、プロシージャのインタフェースを定義します。

PHP のストアド・プロシージャまたは関数は、SQL のストアド・プロシージャまたは関数と同じように動作します。ただし、例外として、プロシージャまたは関数のコードは PHP で記述され、プロシージャまたは関数の実行は、データベース・サーバの外部 (つまり、PHP 実行プログラム・インスタンス内) で行われます。

プロシージャ定義の例：

```
CREATE PROCEDURE PHPPopulateTable()
NO RESULT SET
EXTERNAL NAME '<file=ServerSidePHPExample>
ServerSidePHPSub()'
LANGUAGE PHP;
```

『SQL Anywhere サーバー – プログラミング』の「SQL Anywhere 外部環境のサポート」>「PHP 外部環境」を参照してください。

注意： このリファレンスは SQL Anywhere マニュアルにリンクされています。

EXTERNAL NAME java-call LANGUAGE JAVA '*java-call*' [*package-name.*] *class-name.method-name* (*method-signature*

method-signature: ([*field-descriptor*, ...]) *return-descriptor*

Java メソッド・シグニチャは、パラメータの型と戻り値の型のコンパクトな文字表現です。

Java メソッドを外部環境で呼び出すには、**EXTERNAL NAME** 句の後に LANGUAGE JAVA 属性を続けて、プロシージャのインタフェースを定義します。

Java インタフェースのストアド・プロシージャまたは関数は、SQL のストアド・プロシージャまたは関数と同じように動作します。ただし、例外として、プロシージャまたは関数のコードは Java で記述され、プロシージャまたは関数の実行は、データベース・サーバの外部 (つまり、Java 仮想マシン内) で行われます。

プロシージャ定義の例：

```
CREATE PROCEDURE HelloDemo( IN
name LONG VARCHAR )
NO RESULT SET
EXTERNAL NAME 'Hello.main([Ljava/lang/String;)V'
LANGUAGE JAVA;
```

『SQL Anywhere サーバー – プログラミング』の「SQL Anywhere 外部環境のサポート」>「Java 外部環境」を参照してください。

注意： このリファレンスは SQL Anywhere マニュアルにリンクされています。

表 7 : Java の Field-descriptor と Return-descriptor

フィールド・タイプ	Java データ型
B	byte
C	char

フィールド・タイプ	Java データ型
D	double
F	float
I	int
J	long
L*class-name*;	クラス *class-name* のインスタンス。クラス名は完全に修飾された名前で、ドットを / に置き換えたものとします。 たとえば、**java/lang/String** のようになります。
S	short
V	void
Z	boolean
[	配列の次元ごとに 1 つ使用

例を示します。

```
double some_method(
  boolean a,
  int b,
  java.math.BigDecimal c,
  byte [][] d,
  java.sql.ResultSet[] d ) {
}
```

この例では、次のシグニチャを得られます。

```
'(ZILjava/math/BigDecimal;[[B[Ljava/sql/ResultSet;)D'
```

関連する動作

- オートコミット。

標準

- SQL—ISO/ANSI SQL 準拠。
- Sybase—Transact-SQL の **CREATE PROCEDURE** 文は異なります。
- SQLJ—Java 結果セットの構文拡張は、推奨される SQLJ1 規格に指定されています。

パーミッション

テンポラリ・プロシージャを作成する場合を除き、RESOURCE 権限を持っている必要があります。外部プロシージャの場合、または別のユーザ用にプロシージャを作成する場合は、DBA 権限を持っている必要があります。

参照：

- ALTER PROCEDURE 文 (30 ページ)
- BEGIN … END 文 (60 ページ)
- CALL 文 (66 ページ)
- CREATE PROCEDURE 文 (137 ページ)
- CREATE PROCEDURE 文 [T-SQL] (144 ページ)
- DROP 文 (209 ページ)
- EXECUTE IMMEDIATE 文 [ESQL] [SP] (226 ページ)
- GRANT 文 (247 ページ)

CREATE PROCEDURE 文 (Java UDF)

外部 Java テーブル UDF へのインタフェースを作成します。

外部プロシージャの場合の **CREATE PROCEDURE** 文のリファレンス情報は、別のトピックで説明しています。テーブル UDF の場合の **CREATE PROCEDURE** 文のリファレンス情報は、別のトピックで説明しています。

クエリで Sybase IQ テーブルを参照する場合は、カタログ・ストア・テーブルのみを参照するクエリとは異なる構文およびパラメータが適用されることに注意してください。

Java テーブル UDF は、**FROM** 句のみでサポートされます。

構文

1 つ以上の Sybase IQ テーブルを参照するクエリ：

```
CREATE[ OR REPLACE ] PROCEDURE
[ owner.]procedure-name ( [ parameter, …] )
[ RESULT (result-column, ...)]
[ SQL SECURITY { INVOKER | DEFINER } ]
EXTERNAL NAME 'java-call'  [ LANGUAGE Java ] }
```

カタログ・ストア・テーブルのみを参照するクエリ：

```
CREATE[ OR REPLACE ] PROCEDURE
[ owner.]procedure-name ( [ parameter, …] )
[  RESULT (result-column, ...)]
| NO RESULT SET
[ DYNAMIC RESULT SETS integer-expression ]
[ SQL SECURITY { INVOKER | DEFINER } ]
EXTERNAL NAME 'java-call' [ LANGUAGE Java ] }
```

パラメータ

- **parameter –** 1 つ以上の Sybase IQ テーブルを参照するクエリ：

  [**IN***parameter_modeparameter-namedata-type* [**DEFAULT***expression*]

カタログ・ストア・テーブルのみを参照するクエリ：

[**IN** | **OUT** | **INOUT**] *parameter_modeparameter-namedata-type* [**DEFAULT***expression*]

- **result-column –** column-name data-type
- **JAVA – [ALLOW | DISALLOW SERVER SIDE REQUESTS]**
- **java-call – '[package-name.]class-name.method-name method-signature'**

使用法

Java テーブル関数の場合、結果セットは 1 つのみ許可されます。 Java テーブル関数が Sybase IQ テーブルとジョインされる場合、または Sybase IQ テーブルのカラムが Java テーブル関数に渡される引数の場合、結果セットは 1 つのみサポートされます。

Java テーブル関数が **FROM** 句内の唯一の項目の場合、*N* 個の結果セットが許可されます。

JAVA: [ALLOW | DISALLOW SERVER SIDE REQUESTS]:

DISALLOW がデフォルトです。

ALLOW は、サーバ側接続が許可されることを示します。

注意： 必要な場合を除き、**ALLOW** は指定しないでください。**ALLOW** を設定すると、特定の種類の Sybase IQ テーブル・ジョインの速度を低下させます。プロシージャ定義を **ALLOW** から **DISALLOW** に変更した場合、またはその逆の変更を行った場合、新しく接続するまで変更は認識されません。

UDF を使用するときは、同じクエリ内で **ALLOW SERVER SIDE REQUESTS** と **DISALLOW SERVER SIDE REQUESTS** の両方を指定しないでください。

標準

- SQL—ISO/ANSI SQL 準拠。
- Sybase—Transact-SQL の **CREATE PROCEDURE** 文は異なります。
- SQLJ—Java 結果セットの構文拡張は、推奨される SQLJ1 規格に指定されています。

パーミッション

テンポラリ・プロシージャを作成する場合を除き、RESOURCE 権限を持っている必要があります。 DBA 権限を持つユーザは、owner を指定することにより他のユーザの UDF を作成できます。 外部 UDF を作成する場合、および別のユーザの外部 UDF を作成する場合、DBA 権限が必要です。

CREATE PROCEDURE 文 (テーブル UDF)

外部のテーブル・ユーザ定義関数 (テーブル UDF: User-Defined Function) へのインタフェースを作成します。テーブル UDF を使用する正規のライセンスを取得しておく必要があります。

テーブル UDF は、a_v4_extfn API を使用して定義します。a_v3_extfn または a_v4_extfn API を使用しない外部プロシージャの場合の **CREATE PROCEDURE** 文のリファレンス情報は、別のトピックで説明しています。Java UDF の場合の **CREATE PROCEDURE** 文のリファレンス情報は、別のトピックで説明しています。

構文

```
CREATE [ OR REPLACE ] PROCEDURE
[ owner.]procedure-name ( [ parameter[, …]] )
| RESULT  result-column [, …] )
[ SQL SECURITY { INVOKER | DEFINER } ]
EXTERNAL NAME 'external-call'
```

パラメータ

- **parameter：** – [**IN**] *parameter-namedata-type* [**DEFAULT** *expression*]
  | [**IN**] *parameter-nametable-type*
- **table-type：** – TABLE(*column-namedata-type* [, ...])
- **external-call：** – [*column-name*:]*function-name@library*; ...

使用法

CREATE PROCEDURE 文はデータベースにプロシージャを作成します。 DBA 権限を持つユーザは、owner を指定することにより他のユーザのプロシージャを作成できます。

ストアド・プロシージャによって結果セットが返された場合は、出力パラメータを設定したり、戻り値を返したりすることもできません。

複数のプロシージャからテンポラリ・テーブルを参照しているときに、テンポラリ・テーブルの定義が矛盾していて、テーブルを参照している文がキャッシュされている場合、問題が発生する可能性があります。 プロシージャ内でテンポラリ・テーブルを参照する場合は注意が必要です。

CREATE PROCEDURE 文を使用して、SQL とは異なるプログラミング言語で実装される外部テーブル UDF を作成できます。 ただし、テーブル UDF の制限事項を確認してから、外部 UDF を作成してください。

スカラ・パラメータ、結果カラム、TABLE パラメータのカラムのデータ型は、有効な SQL データ型である必要があります。

パラメータ名は、カラム名など他のデータベース識別子に関するルールに従って付けてください。 パラメータ名は、有効な SQL データ型である必要があります。

TPF は、スカラ・パラメータと単一の TABLE パラメータの混在をサポートしています。TABLE パラメータでは、UDF によって処理される入力ロー・セットのスキーマを定義します。 TABLE パラメータの定義には、カラム名とカラムのデータ型が含まれます。

```
TABLE(c1 INT, c2 CHAR(20))
```

上記の例は、データ型が INT と CHAR(20) の 2 つのカラム、c1 と c2 を持つスキーマを定義しています。 UDF によって処理される各ローは、2 つの値を持つタプルである必要があります。 Table パラメータには、スカラ・パラメータとは異なり、デフォルト値を割り当てることはできません。

- **IN キーワード –** パラメータには、キーワード IN をプレフィクスとして付けることができます。
  - IN—パラメータは、スカラ・パラメータに値を渡すオブジェクトか、または UDF に渡される TABLE パラメータの値のセットです。

  注意： TABLE パラメータは、INOUT または OUT として宣言できません。TABLE パラメータは 1 つのみ指定できます (その場所は重要ではありません)。
- **OR REPLACE – OR REPLACE** (**CREATE OR REPLACE PROCEDURE**) を指定すると、新しいプロシージャが作成されるか、同じ名前の既存のプロシージャが置き換えられます。 この句によって、プロシージャの定義は変更されますが、既存のパーミッションは維持されます。置き換えられるプロシージャがすでに使用されている場合は、エラーが返されます。
- **RESULT –** 外部 UDF の結果セットのカラム名とそのデータ型を宣言します。カラムのデータ型は、有効な SQL データ型である必要があります (たとえば、結果セット内のカラムは、データ型として TABLE を持つことはできません)。結果内のデータ・セットが TABLE を暗黙的に指定します。 外部 UDF は、TABLE 型の結果セットのみを持つことができます。

  注意： TABLE は出力値ではありません。テーブル UDF は、その結果セット内に LONG VARBINARY または LONG VARCHAR データ型を持つことはできません。ただし、パラメータ化されたテーブル関数 (TPF: Table Parameterized Function) は、その結果セット内にラージ・オブジェクト (LOB: Large Object) データを持つことができます。

  TPF は LOB データを生成できませんが、結果セット内に LOB データ型のカラムを持つことはできます。ただし、出力内に LOB データを含める唯一の方法は、入力テーブルから出力テーブルにカラムを渡すことです。記述属性

EXTFNAPIV4_DESCRIBE_COL_VALUES_SUBSET_OF_INPUT を使用すると、入力テーブルから出力テーブルにカラムを渡すことができます。詳細は、サンプル・ファイル tpf_blob.cxx を参照してください。

プロシージャから返される結果セットの詳細については、『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。

- **SQL SECURITY –** INVOKER (UDF を呼び出しているユーザ) または DEFINER (UDF を所有しているユーザ) のどちらとしてプロシージャが実行されるかを定義します。 デフォルトは DEFINER です。

  SQL SECURITY INVOKER が指定されていると、プロシージャを呼び出す各ユーザに対して注釈を付ける必要があるため、必要なメモリも多くなります。また、SQL SECURITY INVOKER が指定されていると、名前解決も INVOKER として実行されます。 したがって、適切な所有者を使用して、すべてのオブジェクト名 (テーブルやプロシージャなど) の条件を満たすように注意してください。 たとえば、user1 が次のプロシージャを作成したとします。

  ```
  CREATE PROCEDURE user1.myProcedure()
      RESULT( columnA INT )
      SQL SECURITY INVOKER
      BEGIN
          SELECT columnA FROM table1;
      END;
  ```

  user2 がこのプロシージャを実行しようとして、user2.table1 テーブルが存在しない場合、テーブルのルックアップでエラーが発生します。 さらに、user2.table1 が存在する場合は、意図した user1.table1 ではなくそのテーブルが使用されます。 この状況を防ぐため、文ではテーブル参照を修飾してください (table1 だけでなく user1.table1 とします)。

- **EXTERNAL NAME –** EXTERNAL NAME *'external-call'*: *external-call*: [*operating-system*:]*function-name@library*; ....

  外部 UDF には、C などのプログラミング言語で記述された関数へのインタフェースを定義する **EXTERNAL NAME** 句が必要です。関数は、データベース・サーバによってそのアドレス空間にロードされます。

  ライブラリ名にはファイル拡張子が付く場合があります。この拡張子は通常、Windows では .dll、UNIX では .so です。 拡張子が付いていなければ、ソフトウェアがプラットフォーム固有のデフォルトのファイル拡張子をライブラリに付加します。 形式の例を次に示します。

  ```
  CREATE PROCEDURE mystring( IN instr CHAR(255),
      IN input_table TABLE(A INT) )
      RESULT (CHAR(255))
  EXTERNAL NAME
  'mystring@mylib.dll;Unix:mystring@mylib.so'
  ```

プラットフォーム固有のデフォルトを使用して、前述の EXTERNAL NAME 句をより簡単に記述する方法を次に示します。

```
CREATE PROCEDURE mystring( IN instr CHAR(255),
    IN input_table TABLE(A INT) )
    RESULT (CHAR(255))
EXTERNAL NAME 'mystring@mylib'
```

標準

- SQL—ISO/ANSI SQL 準拠。
- Sybase—Transact-SQL の **CREATE PROCEDURE** 文は異なります。
- SQLJ—Java 結果セットの構文拡張は、推奨される SQLJ1 規格に指定されています。

パーミッション

テンポラリ・プロシージャを作成する場合を除き、RESOURCE 権限を持っている必要があります。 DBA 権限を持つユーザは、owner を指定することにより他のユーザの UDF を作成できます。 外部 UDF を作成する場合、および別のユーザの外部 UDF を作成する場合、DBA 権限が必要です。

CREATE SCHEMA 文

データベース・ユーザ用のテーブル、ビュー、パーミッションとそれらに関連するパーミッションのコレクションであるスキーマを作成します。

構文

```
CREATE SCHEMA AUTHORIZATION userid
... [ { create-table-statement
| create-view-statement
| grant-statement } ] …
```

使用法

userid には、現在の接続のユーザ ID を指定する必要があります。別のユーザに対するスキーマを作成することはできません。ユーザ ID の大文字小文字は区別されません。

CREATE SCHEMA 文に含まれるいずれかの文にエラーが発生すると、**CREATE SCHEMA** 文全体がロールバックされます。

CREATE SCHEMA 文を使用すると、個別の **CREATE** 文と **GRANT** 文を 1 つにまとめて一度に処理できます。データベース内に SCHEMA データベース・オブジェクトは作成されません。オブジェクトを削除するには、個別の **DROP TABLE** 文または

DROP VIEW 文を使用します。パーミッションを取り消すには、付与した各パーミッションごとに **REVOKE** 文を使用する必要があります。

注意： アクティブなマルチプレックスでは、**CREATE SCHEMA** 文は無効です。

個々の **CREATE** 文または **GRANT** 文は、文デリミタで区切りません。文デリミタは **CREATE SCHEMA** 文自身の末尾を区切ります。

それぞれの **CREATE** 文または **GRANT** 文は、まずオブジェクトを作成してから、それにパーミッションを付与するという順番に並べます。

1 ユーザに対して複数のスキーマを作成することはおすすめできません。また、今後のバージョンではサポートされなくなる可能性があります。

関連する動作

- オートコミット。

標準

- SQL—ISO/ANSI SQL 準拠。
- Sybase—Sybase IQ では、**CREATE SCHEMA** 文内での **REVOKE** 文の使用はサポートされていません。また、Transact-SQL のバッチまたはプロシージャ内での使用も許可されていません。

パーミッション

RESOURCE 権限が必要です。

参照：

- CREATE TABLE 文 (166 ページ)
- CREATE VIEW 文 (186 ページ)
- GRANT 文 (247 ページ)

CREATE SERVER 文

サーバを ISYSSERVER テーブルに追加します。

構文

```
CREATE SERVER server-name
              CLASS 'server-class'
USING 'connection-info'
[ READ ONLY ]
```

パラメータ

- **server-class：** – { **ASAJDBC** | **ASEJDBC** | **ASAODBC** | **ASEODBC** | **DB2ODBC** | **MSSODBC** | **ORAODBC** | **ODBC** }
- **connection-info：** – { *machine-name:port-number* [*/dbname*] | *data-source-name* }

例

- **例 1 –** JDBC ベースの Adaptive Server Enterprise サーバ用のリモート・サーバを `ase_prod` の名前で作成します。このマシン名は "banana" でポート番号は 3025 です。

```
CREATE SERVER ase_prod
CLASS 'asejdbc'
USING 'banana:3025'
```

- **例 2 –** "apple" というマシンに、名前が `testasa` で、ポート番号 2638 で受信する SQL Anywhere のリモート・サーバを作成します。

```
CREATE SERVER testasa
CLASS 'asajdbc'
USING 'apple:2638'
```

- **例 3 –** Oracle サーバ用の `oracle723` という名前のリモート・サーバを作成します。この ODBC データ・ソース名は "oracle723" です。

```
CREATE SERVER oracle723
CLASS 'oraodbc'
USING 'oracle723'
```

使用法

CREATE SERVER 文は、Sybase IQ カタログからリモート・サーバを定義します。

サーバ・クラスの詳細とサーバを構成する方法については、『システム管理ガイド： 第 2 巻』の「リモート・データ・アクセス用のサーバ・クラス」を参照してください。

USING 句—JDBC ベースのサーバ・クラスを使用する場合、**USING** 句は *hostname:port-number [/dbname]* となります。ここで、

- hostname—リモート・サーバを実行するマシンです。
- portnumber—リモート・サーバが受信している TCP/IP のポート番号です。Sybase IQ と SQL Anywhere のデフォルトのポート番号は 2638 です。
- dbname—SQL Anywhere リモート・サーバでは、*dbname* を指定しない場合、デフォルト・データベースが使用されます。 Adaptive Server Enterprise の場合、デフォルトは **master** データベースです。*dbname* を使用しない場合は、他の方法 (**FORWARD TO** 文など) を使って別のデータベースを指定します。

詳細については、『システム管理ガイド：第 2 巻』の「リモート・データ・アクセス用のサーバ・クラス」>「JDBC ベースのサーバ・クラス」を参照してください。

ODBC ベースのサーバ・クラスを使用する場合、**USING** 句は *data-source-name* になります。data-source-name は ODBC データ・ソース名です。

READ ONLY—**READ ONLY** 句は、リモート・サーバが読み込み専用データ・ソースであることを指定します。 更新の要求は Sybase IQ によって拒否されます。

関連する動作

- オートコミット。

標準

- SQL—ISO/ANSI SQL 準拠。
- Sybase—Open Client/Open Server でサポートされています。

パーミッション

このコマンドを実行するには、DBA 権限が必要です。

参照：

- ALTER SERVER 文 (32 ページ)
- DROP SERVER 文 (218 ページ)

CREATE SERVICE 文

データベース・サーバが Web サーバとして機能することを許可します。

構文

```
CREATE SERVICE service-name
              TYPE service-type-string
[ attributes ] [
AS statement ]
```

パラメータ

- **attributes：** – [**AUTHORIZATION** { **ON** | **OFF** }] [**SECURE** { **ON** | **OFF** }] [**USER** { *user-name* | **NULL** }] [**URL** [**PATH**/] { **ON** | **OFF** | **ELEMENTS** }] [**USING** { *SOAP-prefix* | **NULL** }]
- **service-type-string：** – { '**RAW** ' | '**HTML** ' | '**XML** ' | '**SOAP** ' | ' **DISH** ' }

例

- **例 1 –** Web サーバをすばやく設定するには、`-xs` スイッチを指定してデータベース・サーバを起動し、次の文を実行します。

```
CREATE SERVICE tables TYPE 'HTML'
AUTHORIZATION OFF    USER DBA
```

```
AS SELECT * FROM SYS.ISYSTAB
```

この文を実行したら、Web ブラウザを使用して URL `http://localhost/tables` を開きます。

使用法

CREATE SERVICE 文を使用すると、データベース・サーバは Web サーバとして動作します。新しいエントリが `SYSWEBSERVICE` システム・テーブルに作成されます。

service-name—Web サービス名には、英数字または "/"、"-"、"_"、"."、"!"、"~"、"*"、"'"、"("、")" を自由に並べて指定できます。ただし、最初の文字をスラッシュ (/) にすること、名前にスラッシュを 2 つ以上続けて使用することはできません。

service-type-string—サービスのタイプを識別します。一覧で示されているサービス・タイプのうちいずれかを指定してください。デフォルト値はありません。

AUTHORIZATION 句—ユーザがサービスに接続する際、ユーザ名とパスワードの指定が必要かどうかを定義します。認証が OFF の場合、**AS** 句が必要となり、**USER** 句によって 1 人のユーザが識別される必要があります。要求はすべて、そのユーザのアカウントとパーミッションを使用して実行されます。

ON の場合は、すべてのユーザにユーザ名とパスワードの入力が必要になります。オプションとして、**USER** 句を使用しユーザ名またはグループ名を指定することにより、ユーザに対しサービスの使用許可を制限することもできます。ユーザ名に NULL が指定された場合、認識されているすべてのユーザがサービスを使用できます。

デフォルト値は ON です。Sybase は、実稼働システムでは AUTHORIZATION を ON にし、ユーザをグループに追加してサービスの使用許可を与えることをおすすめします。

SECURE 句—安全でない接続を受け入れるかどうかを指定します。ON は、HTTPS 接続のみ受け入れることを意味します。HTTP ポートで受け取ったサービス要求は、HTTPS ポートへ自動的にリダイレクトされます。OFF に設定すると、HTTP と HTTPS の接続をどちらも受け入れます。デフォルト値は OFF です。

USER 句—AUTHORIZATION が OFF の場合、このパラメータは必須となり、すべてのサービス要求の実行に使用するユーザ ID が指定されます。AUTHORIZATION

が ON (デフォルト) であれば、この句はオプションとなり、サービスへのアクセスを許可するユーザまたはグループが識別されます。デフォルト値は NULL で、すべてのユーザにアクセス許可が与えられます。

URL 句—URI パスを受け入れるかどうか、また、受け入れるのであれば、どのように処理されるかを定義します。OFF の場合、URI 要求のサービス名の後に何も指定することができません。ON の場合は、URI の残りの部分が、*url* という名前の変数の値として解釈されます。**ELEMENTS** は、URI パスの残りの部分がスラッシュ文字で区切られて、最大 10 個の要素のリストになるという意味です。値は、url の末尾に 1 から 10 までの数字を付けた名前を持つ変数に割り当てられます。たとえば、3 つ目までの変数の名前は url1、url2、url3 です。指定された値が 10 に満たなければ、残りの変数は NULL に設定されます。サービス名が / の文字で終わっている場合、URL は OFF に設定する必要があります。デフォルト値は OFF です。

USING 句—DISH サービスにのみ適用されます。パラメータには名前のプレフィクスを指定します。このプレフィクスで始まる名前の SOAP サービスだけが処理されます。

statement—statement が NULL の場合、実行する文を URI に指定する必要があります。それ以外の場合は、指定された SQL 文だけがそのサービス内で実行可能になります。statement は SOAP サービスには必須ですが、DISH サービスでは無視されます。デフォルト値は NULL です。

実稼働システムでは、実行するすべてのサービスに必ず文を定義することをおすすめします。AUTHORIZATION が ON である場合にかぎり、文は NULL でもかまいません。

RAW—結果セットは、それ以上一切フォーマットされないまま、クライアントへ送信されます。必要なタグをプロシージャ内で明示的に生成すれば、フォーマットされたドキュメントを作成できます。後ほど一例を示します。

HTML—文またはプロシージャの結果セットが、テーブルを含む HTML ドキュメントに自動的にフォーマットされます。

XML—結果セットは XML フォーマットであると想定されます。XML フォーマットになっていない場合は、自動的に XML RAW フォーマットに変換されます。

SOAP—要求は、有効な Simple Object Access Protocol (SOAP) 要求であることが必要です。結果セットは、自動的に SOAP 応答としてフォーマットされます。SOAP 規格の詳細については、www.w3.org/TR/SOAP を参照してください。

DISH—SOAP ハンドラ、すなわち DISH を定義します。このサービスは、1 つ以上の SOAP サービスのプロキシとして機能します。使用中は、複数の SOAP サービスへのアクセスを保持し、アクセス可能にするコンテナとして機能します。DISH に含まれる SOAP サービスのそれぞれに、Web Services Description Language

(WSDL) ファイルが自動的に生成されます。内部の SOAP サービスは、共通のプレフィクスで識別されます。この識別子は、**USING** 句で指定する必要があります。

Web サービスの使い方については、『SQL Anywhere サーバー - プログラミング』の「HTTP Web サービス」>「HTTP Web サーバとしての SQL Anywhere の使用」を参照してください。

注意： このリファレンスは SQL Anywhere マニュアルにリンクされています。

標準

- SQL—ISO/ANSI SQL 準拠。
- Sybase—Adaptive Server Enterprise ではサポートされていません。

パーミッション

DBA 権限が必要です。

参照：

- ALTER SERVICE 文 (34 ページ)
- DROP SERVICE 文 (219 ページ)

CREATE TABLE 文

データベースまたはリモート・サーバに新しいテーブルを作成します。

構文

```
CREATE [ { GLOBAL | LOCAL } TEMPORARY ] TABLE
  [ IF NOT EXISTS ] [ owner. ]table-name
… ( column-definition [ column-constraint ] …
[ , column-definition [ column-constraint ] …]
[ , table-constraint ] … )
|[WITH NULLS NOT DISTINCT]
…[ IN dbspace-name ]
…[ ON COMMIT { DELETE | PRESERVE } ROWS
| NOT TRANSACTIONAL ]
[ AT location-string ]
[PARTITION BY range-partitioning-scheme ]
```

パラメータ

- **column-definition：** – *column-name**data-type* [[**NOT**] **NULL**] [**IN** *dbspace-name*] [**DEFAULT***default-value* | **IDENTITY**] [**PARTITION** (*partition-name***IN** *dbspace-name* [, ...])]

- **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**
- **column-constraint：** – [**CONSTRAINT** *constraint-name*] { { **UNIQUE** | **PRIMARYKEY** | **REFERENCES** *table-name* [(*column-name*)] [*action*] } [**IN** *dbspace-name*] | **CHECK** (*condition*) | **IQ UNIQUE** (*integer*) }
- **table-constraint：** – [**CONSTRAINT** *constraint-name*] { { **UNIQUE** (*column-name* [, *column-name*] …) | **PRIMARYKEY** (*column-name* [, *column-name*] …) } [**IN** *dbspace-name*] | *foreign-key-constraint* | **CHECK** (*condition*) | **IQ UNIQUE** (*integer*) }
- **foreign-key-constraint：** – **FOREIGNKEY** [*role-name*] [(*column-name* [, *column-name*] …)] … **REFERENCES** *table-name* [(*column-name* [, *column-name*] …)] … [*action*] [**IN** *dbspace-name*]
- **action：** – **ON** { **UPDATE** | **DELETE** { **RESTRICT** }
- **location-string：** – { *remote-server-name*.[*db-name*].[*owner*].*object-name* | *remote-server-name*;[*db-name*];[*owner*];*object-name* }
- **range-partitioning-scheme：** – **RANGE**(*partition-key*) (*range-partition-decl* [,*range-partition-decl* ...])
- **partition-key：** – *column-name*
- **range-partition-decl：** – *partition-name* **VALUES** <= ({*constant-expr* | **MAX** }) [**IN** *dbspace-name*]

例

- **例 1** – 5 つのカラムを含む SalesOrders2 という名前のテーブルを作成します。カラム FinancialCode、OrderDate、ID のデータ・ページは DB 領域 Dsp3 にあります。整数カラム CustomerID のデータ・ページは DB 領域 Dsp1 にあります。CLOB カラム History のデータ・ページは DB 領域 Dsp2 にあります。プライマリ・キー (ID の HG) のデータ・ページは DB 領域 Dsp4 にあります。

```
CREATE TABLE SalesOrders2 (
FinancialCode CHAR(2),
CustomerID int IN Dsp1,
History CLOB IN Dsp2,
OrderDate TIMESTAMP,
ID BIGINT,
PRIMARY KEY(ID) IN Dsp4
) IN Dsp3
```

- **例 2** – 4 つのカラムを含むテーブル fin_code2 を作成します。カラム code、type、id のデータ・ページはデフォルトの DB 領域にあり、データベース・オプション DEFAULT_DBSPACE の値によって決まります。CLOB カラム

description のデータ・ページは DB 領域 Dsp2 にあります。外部キー fk1 (c1 の HG) のデータ・ページは DB 領域 Dsp4 にあります。

```
CREATE TABLE fin_code2 (
code INT,
type CHAR(10),
description CLOB IN Dsp2,
id BIGINT,
FOREIGN KEY fk1(id) REFERENCES SalesOrders(ID) IN Dsp4
)
```

- **例 3 –** テーブル t1 を作成します。ここで、パーティション p1 はパーティション p2 に、パーティション p2 はパーティション p3 に隣接します。

```
CREATE TABLE t1 (c1 INT, c1 INT)
PARTITION BY RANGE(c1),
(p1 VALUES <= (0), p2 VALUES <= (10), p3 VALUES <= (100))
```

- **例 4 –** 6 つのカラムと 3 つのパーティションを含む、分割されたテーブル bar を作成し、データを日付に基づいたパーティションにマッピングします。

```
CREATE TABLE bar (
        c1 INT IQ UNIQUE(65500),
        c2 VARCHAR(20),
        c3 CLOB PARTITION (P1 IN Dsp11, P2 IN Dsp12,
           P3 IN Dsp13),
        c4 DATE,
        c5 BIGINT,
        c6 VARCHAR(500) PARTITION (P1 IN Dsp21,
           P2 IN Dsp22),
        PRIMARY KEY (c5) IN Dsp2) IN Dsp1
        PARTITION BY RANGE (c4)
        (P1 VALUES <= ('2006/03/31') IN Dsp31,
         P2 VALUES <= ('2006/06/30') IN Dsp32,
         P3 VALUES <= ('2006/09/30') IN Dsp33
        ) ;
```

各パーティションのデータ・ページ割り付け：

パーティション	DB 領域	カラム
P1	Dsp31	c1、c2、c4、c5
P1	Dsp11	c3
P1	Dsp21	c6
P2	Dsp32	c1、c2、c4、c5
P2	Dsp12	c3
P2	Dsp22	c6
P3	Dsp33	c1、c2、c4、c5、c6

パーティション	DB 領域	カラム
P3	Dsp13	c3
P1、P2、P3	Dsp1	c1 および他の共有データのルックアップ・ストア
P1、P2、P3	Dsp2	プライマリ・キー (c5 の HG)

- **例 5** – 図書データベース用にテーブルを作成し、図書情報を保持します。

```
CREATE TABLE library_books (
isbn CHAR(20)      PRIMARY KEY IQ UNIQUE (150000),
copyright_date     DATE,
title              CHAR(100),
author             CHAR(50)
)
```

- **例 6** – 図書データベース用にテーブルを作成し、貸出された図書の情報を保持します。

```
CREATE TABLE borrowed_book (
date_borrowed DATE NOT NULL,
date_returned DATE,
book       CHAR(20)
           REFERENCES library_books (isbn),
CHECK( date_returned >= date_borrowed )
)
```

- **例 7** – リモート・サーバ SERVER_A にテーブル t1 を作成し、リモート・テーブルにマッピングされるプロキシ・テーブル t1 を作成します。

```
CREATE TABLE t1
( a  INT,
  b  CHAR(10))
AT 'SERVER_A.db1.joe.t1'
```

- **例 8** – カラム c1 に特殊定数 LAST USER のデフォルト値を含むテーブル tab1 を作成します。

```
CREATE TABLE tab1(c1 CHAR(20) DEFAULT LAST USER)
```

使用法

所有者名を指定することにより、別のユーザが使用するテーブルを作成できます。**GLOBAL TEMPORARY** または **LOCAL TEMPORARY** が指定されていない場合、テーブルはベース・テーブルと呼ばれます。指定すると、テーブルはテンポラリ・テーブルとなります。

作成されたグローバル・テンポラリ・テーブルは、ベース・テーブルと同様にデータベース内に存在し、**DROP TABLE** 文によって明示的に削除されるまでデータベース内に残ります。テンポラリ・テーブル内のローは、ローを挿入した接続だけが参照できます。同じ、または異なるアプリケーションからの複数の接続が、同じテンポラリ・テーブルを同時に使用することもできます。このとき、それぞ

れの接続で参照できるのは自身のローだけです。この接続は、最初にグローバル・テンポラリ・テーブルを参照し、存在していれば、そのテーブルのスキーマを継承します。接続が終了すると、テンポラリ・テーブルのローは削除されます。

ローカル・テンポラリ・テーブルを作成するときは、所有者を指定しないでください。たとえば、テンポラリ・テーブルの作成時に、`CREATE TABLE dbo.#temp(col1 int)` のように所有者を指定すると、ベース・テーブルが間違って作成されます。

その接続に同じ名前のローカル・テンポラリ・テーブルがある場合、ベース・テーブルまたはグローバル・テンポラリ・テーブルを作成しようとしても失敗します。これは、新しいテーブルを *owner.table* がユニークに識別できないからです。

ただし、既存のベース・テーブルまたはグローバル・テンポラリ・テーブルとしてなら、ローカル・テンポラリ・テーブルを同じ名前で作成できます。テーブル名への参照は、ローカル・テンポラリ・テーブルにアクセスします。ローカル・テンポラリ・テーブルが最初に解決されるからです。

次のシーケンス例を見てみましょう。

```
CREATE TABLE t1 (c1 int);
INSERT t1 VALUES (9);

CREATE LOCAL TEMPORARY TABLE t1 (c1 int);
INSERT t1 VALUES (8);

SELECT * FROM t1;
```

返される結果は `8` です。ローカル・テンポラリ・テーブルが接続により削除されるまで、`t1` に対する参照はいずれも、ローカル・テンポラリ・テーブル `t1` を参照します。

プロシージャの完了後も保持されるテーブルを作成する場合、**DECLARE LOCAL TEMPORARY TABLE** 文ではなく **CREATE LOCAL TEMPORARY TABLE** 文をプロシージャに使用します。**CREATE LOCAL TEMPORARY TABLE** 文を使用して作成されたローカル・テンポラリ・テーブルは、明示的に削除するか接続が終了するまで保持されます。

また、**CREATE LOCAL TEMPORARY TABLE** を使用して **IF** 文で作成されたローカル・テンポラリ・テーブルも、**IF** 文の完了後に保持されます。

テンポラリ・テーブルを使用して、ジョイン・インデックスを作成することはできません。

ジョイン・インデックスの一部となっているベース・テーブルは更新しないでください。この操作は禁止されており、次のエラーが返されます。

```
-1000102 Cannot update table %2 because it is defined in one or more
join indexes
```

Sybase IQ では、Sybase IQ テーブルのテーブル・レベルの暗号化の **CREATE TABLEENCRYPTED** 句はサポートされていません。 ただし、**CREATE TABLEENCRYPTED** 句は、Sybase IQ データベースの SQL Anywhere テーブルではサポートされています。

IF NOT EXISTS—指定された名前のオブジェクトがすでに存在する場合、変更は行われず、エラーは返されません。

WITH NULLS NOT DISTINCT 句を指定できるのは、インデックスが **UNIQUE** であると宣言していて、インデックス・キー内の NULL がユニークでないと指定できる場合に限られます。詳細については、**UNIQUE** 句を参照してください。

IN—テーブルが作成されるデータベース・ファイル (DB 領域) を指定します。この句で SYSTEM を指定し、永久テーブルまたはテンポラリ・テーブルをカタログ・ストアに置くことができます。これ以外の **IN** 句の使用はすべて無視されます。この句を使用して IQ テーブルを特定の DB 領域に置くことはできません。デフォルトでは、すべての永久テーブルはメイン IQ ストアに、すべてのテンポラリ・テーブルはテンポラリ IQ ストアに配置されます。グローバル・テンポラリ・テーブルとローカル・テンポラリ・テーブルを IQ ストアに置くことは絶対にできません。

column-definition 句、*column-constraint* 句、*table-constraint* 句、*foreign-key* 句にある **IN** 句は、オブジェクトが作成される DB 領域を指定します。 **IN** 句を省略した場合、Sybase IQ はテーブルが割り当てられている DB 領域にオブジェクトを作成します。

DB 領域の詳細については、「CREATE DBSPACE 文」を参照してください。

ON COMMIT—テンポラリ・テーブルに対してのみ使用できます。デフォルトで、テンポラリ・テーブルのローは **COMMIT** のときに削除されます。

マルチプレックス・グローバル・テンポラリ・テーブルの句の動作については、『Sybase IQ Multiplex の使用』の「マルチプレックス・トランザクション」>「DDL コマンド」>「役割制限」>「ローの保持」を参照してください。

NOT TRANSACTIONAL—テンポラリ・テーブルに対してのみ使用できます。**NOT TRANSACTIONAL** を使用して作成されたテーブルは、**COMMIT** や **ROLLBACK** の対象になりません。

NOT TRANSACTIONAL 句を使用すると、状況次第では処理を高速にすることができます。その理由は、非トランザクション・テンポラリ・テーブルに対して処理を行う場合、エントリがロールバック・ログに記録されないからです。たとえば、テンポラリ・テーブルを使用するプロシージャが、間に **COMMIT** 文や **ROLLBACK** 文を挟まずに繰り返し呼び出される場合、**NOT TRANSACTIONAL** が有用です。

CREATE TABLE 文に続くカッコの内のリストには、次に示す句が入りますが、順序の指定はありません。

AT—*location-string* によって指定されたリモート・ロケーションにテーブルを作成するために使用されます。作成されたローカル・テーブルは、リモート・ロケーションにマッピングされるプロキシ・テーブルです。プロキシ・テーブルとして使用するテーブルには、30 文字以下の名前を付けてください。**AT** 句は、デリミタとしてセミコロン (;) をサポートします。セミコロンが *location-string* 文字列内のどこかにある場合、そのセミコロンはフィールド・デリミタです。セミコロンがない場合は、ピリオドがフィールド・デリミタです。ピリオドを使用すると、データベースおよび所有者の各フィールドにファイル名と拡張子を使用できます。

セミコロンのフィールド・デリミタは、現在サポートされていないサーバ・クラスで主に使用されていますが、ピリオドもフィールド・デリミタとして機能する状況ではセミコロンも使用できます。たとえば、次の文は、テーブル `proxy_a` をリモート・サーバ `myasa` の SQL Anywhere データベース `mydb` にマッピングします。

```
CREATE TABLE proxy_a1
AT 'myasa;mydb;;a1'
```

外部キー定義は、リモート・テーブルでは無視されます。リモート・テーブルを参照するローカル・テーブルの外部キー定義も無視されます。プライマリ・キー定義は、サーバがプライマリ・キーをサポートする場合、リモート・サーバに送信されます。

シンプレックス環境では、同じノード上でリモート・テーブルを参照するプロキシ・テーブルを作成することはできません。マルチプレックス環境では、マルチプレックス内で定義されたリモート・テーブルを参照するプロキシ・テーブルを作成することはできません。

たとえば、シンプレックス環境で、同じノード上で定義されたベース・テーブル `Employees` を参照するプロキシ・テーブル `proxy_e` を作成しようとすると、**CREATE TABLE ... AT** 文が拒否され、エラー・メッセージが表示されます。マルチプレックス環境では、マルチプレックス内で定義されたリモート・テーブル `Employees` を参照する任意のノード (コーディネータまたはセカンダリ) からプロキシ・テーブル `proxy_e` を作成する場合に、**CREATE TABLE .... AT** 文が拒否されます。

column-definition—テーブル内のカラムを定義します。使用可能なデータ型については、『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。同じテーブルの 2 つのカラムが、同じ名前を持つことはありません。NOT NULL を指定する場合、またはカラムが **UNIQUE** または **PRIMARY KEY** の制約を受ける場合、カラムに NULL 値を含めることはできません。最大 45,000 のカラムを作成可能ですが、1 つのテーブルに 10,000 を超えるカラムを作成すると、パフォーマンスの低下を招くおそれがあります。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) です。

- DEFAULT default-value—テーブルのカラムを定義するとき、**CREATE TABLE** 文 (および **ALTER TABLE** 文) で、DEFAULT キーワードを使用して、カラムのデフォルト値を指定できます。カラムに DEFAULT 値が指定されている場合、カラムの値を指定しないすべての **INSERT** 文 (または **LOAD** 文) でカラムの値としてこの DEFAULT 値が使用されます。

  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。
- DEFAULT AUTOINCREMENT—DEFAULT AUTOINCREMENT カラムの値は、テーブル内の各ローをユニークに識別します。 この種のカラムは、Adaptive Server Enterprise との互換性を考慮して IDENTITY カラムとも呼ばれます。 IDENTITY/DEFAULT AUTOINCREMENT カラムには、挿入または更新時に自動的に生成される連続した数値が格納されます。 IDENTITY/DEFAULT AUTOINCREMENT を使用する場合、カラムは整数データ型のいずれか、または真数型で、位取りを 0 にする必要があります。カラムの値は NULL でもかまいません。 テーブル名は、所有者の名前で修飾して指定する必要があります。

  ON でテーブルへの挿入を行います。IDENTITY/DEFAULT AUTOINCREMENT カラムへの値が指定されなければ、カラム内のどの値よりも大きいユニークな値が生成されます。カラムへ格納する値を INSERT に指定すれば、その値が使用されます。指定した値がそのカラムの現在の最大値よりも小さい場合、指定した値は後続の挿入に対して生成する値の起点として使用されます。
  ローを削除しても IDENTITY/AUTOINCREMENT カウンタはデクリメントされません。ローを削除することによって生じる隔たりは、挿入の際に明示的に値を割り当てる以外に修正する方法がありません。IDENTITY/AUTOINCREMENT カラムへの挿入を実行するには、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。

  以下に、IDENTITY カラムを持つテーブルを作成し、データを明示的に追加する例を示します。

```
CREATE TABLE mytable(c1 INT IDENTITY);
SET TEMPORARY OPTION IDENTITY_INSERT = "DBA".mytable;
INSERT INTO mytable VALUES(5);
```

  最大値よりも小さいロー番号を明示的に挿入すると、後続のローで明示的に割り当てなくても、その最大値より 1 大きい値に自動的にインクリメントされます。

  @@identity グローバル変数を調べれば、カラムに直前に挿入された値を知ることができます。
- IDENTITY—AUTOINCREMENT のデフォルトを Transact-SQL 互換で使用するための代替方法です。 Sybase IQ では、**IDENTITY** 句または **DEFAULT AUTOINCREMENT** 句を使用して IDENTITY カラムを作成します。

table-constraint—データベース内のデータの整合性を保証します。整合性制約には次の 4 つのタイプがあります。

- UNIQUE 制約 (一意性制約)—1 つまたは複数のカラムによってテーブル内の各ローがユニークに識別されるように指定します。テーブル内の 2 つのローは、指定されたすべてのカラム中に同じ値を持つことはできません。1 つのテーブルに複数の一意性制約が存在することがあります。
- PRIMARY KEY 制約 (プライマリ・キー制約)—一意性制約 (**UNIQUE** 制約) と同じですが、プライマリ・キー制約はテーブルに 1 つしか作成できない点が異なります。*プライマリ・キー制約と一意性制約を同じカラムに指定することはできません*。プライマリ・キーは通常、特定のローにもっとも適した識別子を識別します。たとえば、顧客番号は顧客テーブルのプライマリ・キーです。
- FOREIGN KEY 制約 (外部キー制約)—これは、一方のカラム・セットに対する値を制限し、他方のテーブルのプライマリ・キー制約または一意性制約の値と一致させます。たとえば、外部キー制約を使用して、invoice テーブルの顧客番号が customer テーブルの顧客番号と確実に一致するようにします。

  注意： ローカル・テンポラリ・テーブルに対しては外部キー制約を作成できません。グローバル・テンポラリ・テーブルは **ON COMMIT PRESERVE ROWS** で作成してください。
- CHECK 制約 (検査制約)—任意の条件を検証できます。たとえば、検査制約を使用して `Gender` カラムに Male と Female の値しか含まれないようにすることができます。テーブル内のどのローも、制約に違反することは許されません。**INSERT** 文または **UPDATE** 文によって、ローが制約に違反することになる場合、操作は許可されず、この文の結果は取り消されます。
  カラムのチェック制約に記述され、先頭に '@' の記号が付くカラム識別子は、実際のカラム名のプレースホルダです。したがって、次のように記述された文は、

  ```
  CREATE TABLE t1(c1 INTEGER CHECK (@foo < 5))
  ```

  次の文とまったく同じになります。

  ```
  CREATE TABLE t1(c1 INTEGER CHECK (c1 < 5))
  ```

  テーブルの検査制約に記述され、先頭に '@' の記号が付くカラム識別子はプレースホルダではありません。

ある文によって整合性制約に違反するデータベースへの変更が生じた場合、その文は事実上実行されず、エラーがレポートされます (「事実上」とは、エラーが検出されるより前にこの文が行った変更がすべて取り消されることを示します)。

Sybase IQ はそのカラムの **HG** インデックスを作成することにより、シングル・カラム一意性制約 (**UNIQUE** 制約) を強制します。

注意： `BIT` データ型のカラムに、一意性制約 (**UNIQUE** 制約) またはプライマリ・キー制約 (**PRIMARY KEY** 制約) を定義することはできません。また、`BIT` データ型

のカラムのデフォルトでは NULL 値を使用できませんが、カラムを明示的に定義して、NULL 値を使用できるように変更できます。

column-constraint—カラムに格納可能な値を制限します。カラムとテーブルの制約によって、データベース内のデータの整合性が保証されます。文の実行が制約の違反を引き起こす場合、実行が中断され、エラーが検出されるよりも前にその文によって追加された変更が取り消されてエラーがレポートされます。カラム制約は、対応するテーブル制約の省略形です。たとえば、次の文は同じです。

```
CREATE TABLE Products (
    product_num integer UNIQUE
)
CREATE TABLE Products (
    product_num integer,
    UNIQUE ( product_num )
)
```

通常、カラム制約を使用するのは、制約がテーブル内で複数のカラムを参照しない場合です。複数のカラムを参照する場合は、テーブル制約を使用する必要があります。

IQ UNIQUE 制約—この制約はカラムのみに指定できます。**IQ UNIQUE** はカラムのカーディナリティを定義し、インデックスを内部的に最適化するのに使用されます。 デフォルト値は 0 です。0 の場合、IQ にデフォルト・インデックスを最適化するための情報を提供しません。 **IQ UNIQUE** 制約は、異なるカラムの予想数 (すなわち、ユニークな値の数) が 65536 以下である場合に適用してください。これにより、Sybase IQ でこのカラムのデータの記憶領域を最適化できるようになります。

`MINIMIZE_STORAGE` オプションを ON (新規データベースのデフォルトは OFF) にすると、新しく作成されるすべてのカラムに IQ UNIQUE 255 を指定することになるため、ユニークな値が 65536 を超えるカラムを除いて、IQ UNIQUE を指定する必要がなくなります。関連情報については、『システム管理ガイド： 第 1 巻』の「データベース・オブジェクトの管理」>「テーブルの管理」>「テーブル作成のガイドライン」>「記憶領域とクエリ・パフォーマンスの最適化」を参照してください。

整合性制約

UNIQUE または UNIQUE (column-name, …)—テーブル内の 2 つのローは、指定されたすべてのカラム中に同じ値を持つことはできません。1 つのテーブルに複数の一意性制約が存在することがあります。

「*一意性制約*」と「*ユニーク・インデックス*」には違いがあります。ユニーク・インデックスのカラムには NULL を格納できますが、一意性制約のカラムには格納できません。外部キーはプライマリ・キーまたは一意性制約を持つカラムを参照できますが、ユニーク・インデックスのカラムは NULL の複数のインスタンスが格納されている可能性があるため参照できません。

PRIMARY KEY または PRIMARY KEY (column-name, …) —テーブルのプライマリ・キーは、リストしたカラムで構成されます。プライマリ・キーに指定されたどのカラムにも NULL 値を格納することはできません。Sybase IQ では、テーブル内の各ローが必ず、ユニークなプライマリ・キー値を持ちます。 1 つのテーブルが持てる PRIMARY KEY は 1 つだけです。

(**PRIMARY KEY** の後にカラム・リストが続く)2 番目のフォームを使用する場合、カラムのリスト順ではなく定義順にカラムを含むプライマリ・キーが作成されます。

カラムが **PRIMARY KEY**、**FOREIGN KEY**、または **UNIQUE** に指定されると、Sybase IQ によってそのカラムに自動的に High_Group インデックスが作成されます。 マルチカラム・プライマリ・キーの場合、このインデックスは個々のカラムではなく、プライマリ・キーに対して作成されます。パフォーマンスを高めるため、各カラムに **HG** または **LF** を指定したインデックスを個別に作成してください。

REFERENCES primary-table-name [(primary-column-name)]—この句はカラムを、プライマリ・テーブルのプライマリ・キーまたは一意性制約に対する外部キーとして定義します。 通常、外部キーは一意性制約のためのものではなく、プライマリ・キーのためのものです。 プライマリ・カラム名を指定する場合、この名前は一意性制約またはプライマリ・キーの制約を受けるプライマリ・テーブルのカラム名と一致する必要があります。また、この制約は、その 1 カラムだけで構成される必要があります。 それ以外の場合、外部キーは第 2 のテーブルのプライマリ・キーを参照します。 プライマリ・キーと外部キーは、データ型と精度、位取り、符号が同じである必要があります。 シングルカラムの外部キーには、ユニークでないシングルカラムの **HG** インデックスだけが作成されます。 マルチカラム外部キーに対しては、Sybase IQ はユニークでない複合 **HG** インデックスを作成します。ユニークまたはユニークでない **HG** インデックスのマルチカラム複合キーの最大幅は 1KB です。

テンポラリ・テーブルは、ベース・テーブルを参照する外部キーを持つことはできません。また、ベース・テーブルも、テンポラリ・テーブルを参照する外部キーを持つことはできません。ローカル・テンポラリ・テーブルは、外部キーを持つことも、外部キーによって参照されることもできません。

FOREIGN KEY [role-name] [(...)] REFERENCES primary-table-name [(...)]—この句は、別のテーブルのプライマリ・キーまたは一意性制約を参照する外部キーを定義します。通常、外部キーは一意性制約のためのものではなく、プライマリ・キーのためのものです(この説明では、この他方のテーブルをプライマリ・テーブルと呼びます)。

プライマリ・テーブル・カラム名が指定されていない場合、プライマリ・テーブルのカラムは、テーブルのプライマリ・キーの中のカラムになります。外部キー・カラム名が指定されていない場合、外部キー・カラムは、プライマリ・テーブルの中のカラムと同じ名前になります。外部キー・カラム名が指定されて

いる場合は、プライマリ・キーのカラム名を指定する必要があります。これらのカラム名は、リスト内の位置に応じて一対になります。

プライマリ・テーブルと外部キー・テーブルが同じでない場合、参照されるキーには一意性制約またはプライマリ・キー制約を定義する必要があります。参照されるキーと外部キーは、カラムの数、データ型、符号、精度、位取りが同じでなくてはなりません。

ローの外部キーの値は、外部キー内の 1 つまたは複数のカラムで、null を許可する外部キー・カラムに null が格納されている場合を除き、プライマリ・テーブルのいずれかのローに格納された候補キー値として出現する必要があります。

明示的に定義されない外部キーのカラムは、プライマリ・テーブルの対応するカラムと同じデータ型で自動的に作成されます。これらの自動的に作成されたカラムは外部テーブルのプライマリ・キーの一部にはなりません。したがって、プライマリ・キーと外部キーの両方の中で使われるカラムは、明示的に作成する必要があります。

role-name は外部キーの名前です。*role-name* の主な機能は、同じテーブルに対する 2 つの外部キーを区別することです。*role-name* が指定されていない場合、role-name は次のように割り当てられます。

1. テーブル名と同じ *role-name* を含む外部キーが存在しない場合、テーブル名が *role-name* として割り当てられます。
2. テーブル名がすでに使用されている場合、*role-name* は、0 が埋め込まれた、テーブルに固有の 3 桁の数字と結合されたテーブル名になります。

参照整合性アクションは、データベース内で外部キー関係を維持するために実行されるアクションを定義します。データベース・テーブルからプライマリ・キー値が変更されたり削除されると、それに対応して何らかの修正が必要となる外部キー値が他のテーブルに存在する可能性があります。**ON DELETE** 句の後に続けて **RESTRICT** 句を指定できます。

RESTRICT—データベースのどこかに対応する外部キーがあるにもかかわらず、プライマリ・キー値を更新または削除しようとすると、エラーになります。外部キーを更新して、候補キーに一致しない新しい値を作成しようとすると、エラーになります。この動作はデフォルトです。ただし、オプションで参照整合性に違反するローを拒否するように **LOAD** を指定した場合を除きます。これにより、文レベルでの参照整合性が確保されます。

アクションを何も指定しないで **CHECK ON COMMIT** を使用すると、**RESTRICT** は **DELETE** のアクションに対して適用されます。Sybase IQ は **CHECK ON COMMIT** をサポートしません。

グローバル・テンポラリ・テーブルは、ベース・テーブルを参照する外部キーを持つことはできません。また、ベース・テーブルも、グローバル・テンポラリ・テーブルを参照する外部キーを持つことはできません。ローカル・テンポラリ・

テーブルは、外部キーを持つことも、外部キーによって参照されることもできません。

CHECK (条件)—条件に合わないローは許可されません。**INSERT** 文によって、ローがこの条件を満たさなくなる場合、操作は許可されず、この文の効果は取り消されます。

変更は、条件が FALSE の場合にのみ拒否されます。条件が UNKNOWN の場合、変更は許可されます。 CHECK 条件は、Sybase IQ では強制されません。 TRUE、FALSE、UNKNOWN の各条件については、『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 言語の要素」>「NULL 値」および『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 言語の要素」>「検索条件」を参照してください。

注意： 孤立した外部キーがないと確実にわかっている場合を除き、Sybase IQ に参照整合性 (すなわち、外部キーとプライマリ・キーの関係) を定義しないことをおすすめします。

リモート・テーブル

外部キー定義は、リモート・テーブルでは無視されます。リモート・テーブルを参照するローカル・テーブルの外部キー定義も無視されます。プライマリ・キー定義は、サーバがプライマリ・キーをサポートする場合、リモート・サーバに送信されます。

PARTITION BY RANGE—分割カラム内で指定した値の範囲に応じてローを分割するように指定します。

partition-key clause の *column-name* は、分割キーのカラムを指定します。Sybase IQ では、1 つの分割キーのカラムがサポートされています。

range-partition-decl clause の *partition-name* は、テーブル・ローを保管する新しいパーティションの名前を指定します。パーティション名は、テーブル上にあるパーティション・セット内でユニークである必要があります。**partition_name clause** の指定は必須です。

VALUE 句—範囲分割の基準用に各パーティションに対して包括的な上限を指定します。ユーザは、各ローが 1 つのパーティションのみに分配されるように、各範囲分割の分割基準を指定する必要があります。NULL は分割カラムに使用でき、NULL を分割キー値に含んだローは最初のテーブル分割に属します。ただし、NULL をバインド値に指定することはできません。最初のパーティションには、下限 (MIN 値) は設定されていません。分割キーの最初のカラムにある NULL セルのローは、最初のパーティションに移動します。最後のパーティションでは、包括的な上限または MAX を指定できます。最後のパーティションの上限値が MAX でない場合は、最後のパーティションの上限値よりも大きい分割キーの値を含んだローをロードまたは挿入すると、エラーが生成されます。

MAX—無制限の上限を示し、最後のパーティションに対してのみ指定できます。

IN—*partition-decl* では、パーティションのローが存在する DB 領域を指定します。

範囲分割されたテーブルの分割キーとバインド値には、次の制限が適用されます。

- パーティション・バインドは定数式でなく、定数として指定する必要があります。
- パーティション・バインドは、パーティションの作成順に応じて、昇順で指定する必要があります。つまり、2 番目のパーティションの上限は最初のパーティションよりも高く指定する必要があります。
  さらに、パーティション・バインドの値は、対応する分割キー・カラムのデータ型と互換でなければなりません。たとえば、VARCHAR と CHAR は互換性があります。
- バインド値に対応する分割キーのカラムとは異なるデータ型が指定されていると、Sybase IQ はバインド値を分割キーのカラムのデータ型に変換します。ただし、次の場合は例外となります。
- 明示的な変換は使用できません。この例では、INT から VARCHAR に明示的に変換しようとしてエラーが生成されます。
  ```
  CREATE TABLE Employees(emp_name VARCHAR(20))  PARTITION BY
  RANGE(emp_name) (p1 VALUES <=(CAST (1 AS VARCHAR(20))),  p2 VALUES
  <= (CAST (10 AS VARCHAR(20)))
  ```
- データ・ロスにつながる暗黙的な変換は使用できません。この例では、パーティション・バインドは分割キー型と互換性がありません。丸めを前提で処理を行うとデータ・ロスにつながる可能性があり、エラーが生成されます。
  ```
  CREATE TABLE emp_id (id INT) PARTITION BY RANGE(id) (p1 VALUES <=
  (10.5), p2 VALUES <= (100.5))
  ```
- この例では、パーティション・バインドと分割キーのデータ型の間には互換性があります。バインド値は FLOAT 値に直接変換されます。丸め処理は必要なく、変換はサポートされています。
  ```
  CREATE TABLE id_emp (id FLOAT) PARTITION BY RANGE(id) (p1 VALUES
  <= (10),  p2 VALUES <= (100))
  ```
- 非バイナリ・データ型からバイナリ・データ型に変換することはできません。たとえば、次の変換は実行できずに、エラーが返されます。
  ```
  CREATE TABLE newemp (name BINARY) PARTITION BY RANGE(name)  (p1
  VALUES <= ("Maarten"),  p2 VALUES <= ("Zymmerman")
  ```
- NULL を範囲分割テーブルで境界として使用することはできません。
- 分割キーの最初のカラムのセル値が NULL と評価された場合、ローは最初のパーティションに挿入されます。 Sybase IQ は、1 つのカラムの分割キーのみをサポートしているため、分割キー内に NULL が含まれていると、ローは最初のパーティションに分配されます。

『システム管理ガイド： 第 1 巻』の「データベース・オブジェクトの管理」>「テーブルの管理」>「テーブル作成のガイドライン」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 文法のベンダ拡張。
  次にベンダ拡張を示します。
  - { **IN** | **ON** } *dbspace-name* 句
  - **ON COMMIT** 句
  - いくつかのデフォルト値
- Sybase—Adaptive Server Enterprise でサポートされますが、いくつか相違があります。
  - テンポラリ・テーブル—**CREATE TABLE** 文中のテーブル名にプレフィクスとしてシャープ記号 (#) を付けて、テンポラリ・テーブルを作成できます。 このようなテンポラリ・テーブルは、Sybase IQ が宣言するテンポラリ・テーブルであり、現在の接続でしか使用できません。 宣言されたテンポラリ・テーブルの詳細については、「DECLARE LOCAL TEMPORARY TABLE 文」を参照してください。
  - 物理的配置—テーブルの物理的な配置は、Sybase IQ と Adaptive Server Enterprise では方法が異なります。 Adaptive Server Enterprise でサポートされている **ON** *segment-name* 句は、Sybase IQ でもサポートされていますが、*segment-name* は IQ の DB 領域を参照します。
  - 制約—Sybase IQ は、名前付き制約と名前付きデフォルトをサポートしませんが、制約とデフォルトの定義をデータ型定義にカプセル化できるユーザ定義データ型はサポートします。 また、**CREATE TABLE** 文の明示的なデフォルトと CHECK 条件もサポートします。
  - NULL デフォルト—Adaptive Server Enterprise のカラムは、デフォルトで NOT NULL に設定されていますが、Sybase IQ でのデフォルト設定は NULL であり、NULL 値が許可されています。 この設定は、`ALLOW_NULLS_BY_DEFAULT` オプションを使用して制御できます。
    「ALLOW_NULLS_BY_DEFAULT オプション [TSQL]」を参照してください。
    データ定義文を転送可能にするには、NULL または NOT NULL を明示的に指定します。

パーミッション

RESOURCE 権限が必要です。別のユーザのテーブルを作成するには、DBA 権限が必要です。IQ メイン・ストア DB 領域でベース・テーブルを作成するには、指定した DB 領域で DBA 権限または RESOURCE 権限が必要かつ、CREATE 権限が必要になります。

参照：

- ALLOW_NULLS_BY_DEFAULT オプション [TSQL] (403 ページ)
- ALTER TABLE 文 (37 ページ)
- CREATE DBSPACE 文 (93 ページ)
- CREATE INDEX 文 (118 ページ)
- DECLARE LOCAL TEMPORARY TABLE 文 (199 ページ)
- DROP 文 (209 ページ)
- MINIMIZE_STORAGE オプション (490 ページ)

CREATE TEXT CONFIGURATION 文

テキスト設定オブジェクトを作成します。

構文

以下を参照してください。

使用法

構文と詳細な説明については、『Sybase IQ の非構造化データ分析の概要』を参照してください。

CREATE TEXT INDEX 文

TEXT インデックスを作成します。

構文

以下を参照してください。

使用法

構文と詳細な説明については、『Sybase IQ の非構造化データ分析の概要』を参照してください。

CREATE USER 文

ユーザを作成します。

構文

```
CREATE USER  user-name  [ IDENTIFIED BY password ]
[ LOGIN POLICY policy-name ]
[ FORCE PASSWORD CHANGE { ON | OFF } ]
```

例

- **例 1 –** 次の例では、パスワード welcome を使用して、SQLTester という名前のユーザを作成しています。SQLTester ユーザは Test1 ログイン・ポリシーに割り当てられ、パスワードは次回ログイン時に有効期限切れになります。

  ```
  CREATE USER SQLTester IDENTIFIED BY welcome
  LOGIN POLICY Test1
  FORCE PASSWORD CHANGE ON;
  ```

- **例 2 –** 次の例では、MyGroup という名前のグループを作成しています。

  ```
  CREATE USER MyGroup;
  GRANT GROUP TO MyGroup;
  ```

使用法

user-name—ユーザ名。

IDENTIFIED BY 句—ユーザのパスワードを入力する句。

policy-name—ユーザに割り当てるログイン・ポリシーの名前。**LOGIN POLICY** が指定されていない場合、変更は行われません。

FORCE PASSWORD CHANGE 句—ユーザがログイン時に新しいパスワードを指定する必要があるかどうかを制御します。この設定は、ポリシー内の PASSWORD_EXPIRY_ON_NEXT_LOGIN オプション設定を上書きします。

ユーザに対してパスワードを指定する必要はありません。パスワードのないユーザは、データベースに接続できません。これは、グループを作成し、他のユーザはグループ・ユーザ ID を使用してデータベースに接続させないようにする場合に便利です。ユーザ ID には、有効な ID を使用します。

ユーザ ID とパスワードは、次のようには指定できません。

- 最初の文字をスペース、一重引用符または二重引用符にする。
- 最後の文字をスペースにする。

- セミコロンを含める。

パスワードには有効な識別子、または一重引用符で囲まれた文字列 (最大 255 バイト) を指定できます。パスワードでは大文字と小文字を区別します。パスワードは 7 ビットの ASCII 文字で指定することをおすすめします。データベース・サーバで、他の文字をクライアントの文字セットから UTF-8 に変換できない場合は、正しく指定されないことがあります。

VERIFY_PASSWORD_FUNCTION オプションは、パスワード規則 (1 桁以上の長さであることなど) の実装に使用できる関数を指定します。パスワード検証関数を使用する場合、**GRANT CONNECT** 文に複数のユーザ ID とパスワードを指定することはできません。詳細については、「VERIFY_PASSWORD_FUNCTION オプション」と「GRANT 文」を参照してください。

また、『SQL Anywhere サーバー - データベース管理』の「データベースの設定」>「ユーザー ID、権限、パーミッションの管理」>「ログインポリシーの管理」も参照してください。

注意： このリファレンスは SQL Anywhere マニュアルにリンクされています。

標準

- 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。

パーミッション

DBA 権限または USER ADMIN 権限が必要です。

参照：

- COMMENT 文 (72 ページ)
- CREATE LOGIN POLICY 文 (132 ページ)
- DROP LOGIN POLICY 文 (215 ページ)
- DROP USER 文 (221 ページ)
- GRANT 文 (247 ページ)
- PASSWORD_EXPIRY_ON_NEXT_LOGIN オプション (503 ページ)
- VERIFY_PASSWORD_FUNCTION オプション (552 ページ)
- ALTER LOGIN POLICY 文 (22 ページ)

CREATE VARIABLE 文

SQL 変数を作成します。

構文

```
CREATE
[OR REPLACE]
VARIABLE
identifier data-type
[{= | DEFAULT}
initial-value]
```

パラメータ

- **initial-value：** – *special-value* | *string* | [-] *number* | (*constant-expression*) | *built-in-function* (*constant-expression*) | **NULL**
- **special-value：** – **CURRENT** { **DATABASE** | **DATE** | **PUBLISHER** | **TIME** | **TIMESTAMP** | **USER** | **UTC TIMESTAMP** } | **USER**

例

- **例 1 –** 次のコードは、データベースに大きなテキスト値を挿入します。

```
EXEC SQL BEGIN DECLARE SECTION;
char buffer[5000];
EXEC SQL END DECLARE SECTION;
EXEC SQL CREATE VARIABLE hold_blob VARCHAR;
EXEC SQL SET hold_blob = '';
for(;;) {
    /* read some data into buffer ... */
    size = fread( buffer, 1, 5000, fp );
    if( size <= 0 ) break;
    /* add data to blob using concatenation
    Note that concatenation works for binary
    data too! */
    EXEC SQL SET hold_blob = hold_blob || :buffer;
}
EXEC SQL INSERT INTO some_table VALUES ( 1, hold_blob );
EXEC SQL DROP VARIABLE hold_blob;
```

使用法

CREATE VARIABLE 文は、指定したデータ型の新しい変数を作成します。 *initial-value* を指定すると、変数はその値に設定されます。*initial-value* を指定しないと、**SET** 文によって異なる値が割り当てられるまで、変数には NULL 値が含まれます。

OR REPLACE 句を指定すると、すでに存在する名前付き変数は削除され、その定義が置き換えられます。**OR REPLACE** 句は、SQL スクリプトの **VAREXISTS** 関数の代わりとして使用できます。

変数は、カラム名を使用できる場所なら SQL 式のどこででも使うことができます。カラム名が変数と同じ名前で存在する場合、その変数の値が使用されます。

変数は、カラム名を使用できる場所なら SQL 式のどこででも使うことができます。名前の決定は次のように実行されます。

- クエリの SELECT リストで指定した任意のエイリアスと一致。
- 任意の参照先テーブルのカラム名と一致。
- 名前が変数であると仮定。

変数は現在の接続に属し、データベースから切断するか、**DROP VARIABLE** 文を使用すると消去されます。 変数は、他の接続からは参照できません。変数は **COMMIT** または **ROLLBACK** 文の影響を受けません。

CREATE VARIABLE 文で作成された変数は、その文が **(BEGIN...END)** 文の内部で発行されたとしても、接続が存続する間は存続します。**(BEGIN...END)** 文の内部 (ストアド・プロシージャ内など) でのみ存続する変数を作成するには、**DECLARE** を使用します。

変数は、Embedded SQL プログラムから **INSERT** または **UPDATE** 文の大きなテキストまたはバイナリ・オブジェクトを作成するときに役立ちます。

プロシージャとトリガのローカル変数は、複合文の中で宣言されます。『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」>「制御文」>「複合文の使用」を参照してください。

initial-value を指定する場合、そのデータ型は、*data-type* によって定義された型に一致する必要があります。

『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。

標準

- 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。

パーミッション

なし

参照：

- BEGIN … END 文 (60 ページ)

- DECLARE 文 (190 ページ)
- DROP VARIABLE 文 (222 ページ)
- SET 文 [ESQL] (350 ページ)

CREATE VIEW 文

データベースにビューを作成します。ビューを使用すると、格納されている方法とは異なる形でデータを参照できます。

構文

```
CREATE [ OR REPLACE ] VIEW
… [ owner.]view-name [ ( column-name [ , … ] ) ]
… AS select-without-order-by
… [ WITH CHECK OPTION ]
```

例

- **例 1 –** 男性従業員のみの情報を表示するビューを作成します。このビューはベース・テーブルと同じカラム名を持ちます。

  ```
  CREATE VIEW male_employee
  AS SELECT *
  FROM Employees
  WHERE Sex = 'M'
  ```

- **例 2 –** 従業員とその所属部署を表示するビューを作成します。

  ```
  CREATE VIEW emp_dept
  AS SELECT Surname, GivenName, DepartmentName
  FROM Employees JOIN Departments
  ON Employees.DepartmentID = Departments.DepartmentID
  ```

使用法

owner を指定することにより、別のユーザが使用するビューを作成できます。別のサーバのビューを作成するには、DBA 権限が必要です。

SELECT 文、**DELETE** 文、**UPDATE** 文、**INSERT** 文では、ビュー名をテーブル名の代わりに使用できます。ただし、ビューは物理的にはテーブルとしてデータベースの中に存在しません。ビューは使用するたびに抽出されます。ビューは、**CREATE VIEW** 文で指定した **SELECT** 文の結果として派生的に作成されます。ビューの中で使用するテーブル名は、テーブルの所有者のユーザ ID によって修飾します。そうしないと、別のユーザ ID ではテーブルを見つけることができなかったり、間違ったテーブルが取得される可能性があります。

OR REPLACE 句 (CREATE OR REPLACE VIEW) を指定すると、新しいビューが作成されるか、同じ名前の既存のビューが置き換えられます。**OR REPLACE** 句を使用

した場合、既存のパーミッションは保持されます。ただし、ビューに定義された **INSTEAD OF** トリガは削除されます。

ビューのカラムには、column name リスト内で指定された名前が付けられます。column name リストが指定されていない場合、ビューのカラムは select リスト項目にある名前が付けられます。select リスト項目にある名前を使用するには、項目が単純なカラム名であるか、エイリアス名が指定されている必要があります (「SELECT 文」 を参照してください)。IDENTIY/AUTOINCREMENT カラムをビューから追加または削除することはできません。

ビューを定義する **SELECT** 文が **GROUP BY** 句と集合関数を含まないか、または **UNION** 操作を伴わない場合は、ビューを更新できます。ビューを更新すると、基になっているテーブルも更新されます。

view-name—識別子です。デフォルトの所有者は現在のユーザ ID です。

column-name—ビューのカラムには *column-name* リスト内で指定された名前が付けられます。column name リストが指定されていない場合、ビューのカラムには select リスト項目にある名前が付けられます。select リスト項目にある名前を使用するには、項目が単純なカラム名であるか、エイリアス名が指定されている必要があります (「SELECT 文」を参照してください)。

AS—ビューの基となっている **SELECT** 文に、**ORDER BY** 句、**SELECT** リストのサブクエリ、TOP 条件または FIRST 条件を含めることはできません。**GROUP BY** 句を含めたり、**UNION** にすることはできます。

WITH CHECK OPTION—**SELECT** 文で定義されたビューの基準を満たさないビューへの更新や挿入がすべて拒否されます。 ただし、Sybase IQ では現在、このオプションは無視されます (互換性を考慮して構文はサポートされています)。

関連する動作

- オートコミット。

標準

- SQL—ISO/ANSI SQL 文法のベンダ拡張。
- Sybase—Adaptive Server Enterprise でサポートされています。

パーミッション

RESOURCE 権限、およびビュー定義内のテーブルに対する SELECT パーミッションが必要です。

参照：

- CREATE TABLE 文 (166 ページ)

- DROP 文 (209 ページ)
- SELECT 文 (339 ページ)

DEALLOCATE DESCRIPTOR 文 [ESQL]

SQL 記述子領域に関連付けられているメモリを解放します。

構文

```
DEALLOCATE DESCRIPTOR
descriptor-name:
string
```

例

- **例 1 –** 「ALLOCATE DESCRIPTOR 文 [ESQL]」を参照してください。

使用法

データ項目、インジケータ変数、構造体自体など、記述子領域に関連付けられているすべてのメモリを解放します。

標準

- SQL—ISO/ANSI SQL 文法のベンダ拡張。
- 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

参照：

- ALLOCATE DESCRIPTOR 文 [ESQL] (5 ページ)
- SET DESCRIPTOR 文 [ESQL] (355 ページ)

宣言セクション [ESQL]

Embedded SQL プログラムでホスト変数を宣言します。ホスト変数を使用して、データベースとデータを交換します。

構文

```
EXEC SQL BEGIN DECLARE SECTION;
... C declarations
EXEC SQL END DECLARE SECTION;
```

例

- **例 1 –**

```
EXEC SQL BEGIN DECLARE SECTION;
char *emp_lname, initials[5];
int dept;
EXEC SQL END DECLARE SECTION;
```

使用法

宣言セクションは、単に **BEGIN DECLARE SECTION** 文と **END DECLARE SECTION** 文で囲まれた C 変数宣言のセクションです。 宣言セクションによって、SQL プリプロセッサはホスト変数として使われる C 変数を認識します。 すべての C 宣言が宣言セクションの中で有効なわけではありません。『SQL Anywhere サーバー - プログラミング』の「Embedded SQL」>「Embedded SQL のプログラミングテクニック」を参照してください。

注意： このリファレンスは SQL Anywhere マニュアルにリンクされています。

標準

- SQL—ISO/ANSI SQL 文法のベンダ拡張。

パーミッション

なし

参照：

- BEGIN … END 文 (60 ページ)

DECLARE 文

複合文 **(BEGIN... END)** 内に SQL 変数を宣言します。

構文

```
DECLARE
variable_name  [ , … ]
data-type  [{
=
| DEFAULT}
initial-value]
```

パラメータ

- **initial-value：** – *special-value* | *string* | [-] *number* | (*constant-expression*) | *built-in-function* (*constant-expression*) | **NULL**
- **special-value：** – **CURRENT** { **DATABASE** | **DATE** | **PUBLISHER** | **TIME** | **TIMESTAMP** | **USER** | **UTC TIMESTAMP** } | **USER**

例

- **例 1** – 次のバッチは、**DECLARE** 文の使用法を示し、メッセージをサーバ・ウィンドウに表示します。

```
BEGIN
  DECLARE varname CHAR(61);
  SET varname = 'Test name';
  MESSAGE varname;
END
```

使用法

プロシージャの本文で使用される変数は、**DECLARE** 文を使用して宣言できます。複合文で宣言した変数は、その複合文の間持続します。また、変数は、複合文内でユニークである必要があります。

プロシージャの本文は複合文で、変数は **BEGIN** のすぐ後で宣言する必要があります。Transact-SQL のプロシージャやトリガでは、このような制限はありません。

initial-value を指定すると、変数はその値に設定されます。**initial-value** を指定しないと、**SET** 文によって異なる値が割り当てられるまで、変数には NULL 値が含まれます。

initial-value を指定する場合、そのデータ型は、*data-type* によって定義された型に一致する必要があります。

標準

- SQL—ISO/ANSI SQL 準拠。
- Sybase—Adaptive Server Enterprise でサポートされています。
  - Adaptive Server Enterprise との互換性を維持するには、変数名の前に @ を付けてください。
  - Adaptive Server Enterprise では、プロシージャまたはトリガで宣言される変数はプロシージャまたはトリガの間存在します。 Sybase IQ では、複合文内で宣言した変数は、(Sybase IQ SQL または Transact-SQL 複合文で宣言されたかにかかわらず) その複合文の間だけ存在します。

パーミッション

なし

参照：

- BEGIN … END 文 (60 ページ)

DECLARE CURSOR 文 [ESQL] [SP]

カーソルを宣言します。カーソルはクエリの結果を操作する主要な手段です。

構文

```
DECLARE cursor-name
[   SCROLL
  | NO SCROLL
  | DYNAMIC SCROLL
]
CURSOR FOR
{ select-statement
| statement-name

| USING variable-name }
```

パラメータ

- **cursor-name：** – 識別子
- **statement-name：** – 識別子 | ホスト変数
- **column-name-list：** – 識別子
- **variable-name：** – 識別子

例

- **例 1 –** 次の例は、Embedded SQL 内のスクロール・カーソルを宣言する方法を示します。

```
EXEC SQL DECLARE cur_employee SCROLL CURSOR
FOR SELECT * FROM Employees;
```

- **例 2 –** 次の例は、Embedded SQL 内の準備文のためのカーソルを宣言する方法を示します。

```
EXEC SQL PREPARE employee_statement
FROM 'SELECT emp_lname FROM Employees';
EXEC SQL DECLARE cur_employee CURSOR
FOR employee_statement ;
```

- **例 3 –** 次の例は、ストアド・プロシージャのカーソルの使用法を示します。

```
BEGIN
  DECLARE cur_employee CURSOR FOR
      SELECT emp_lname
      FROM Employees;
  DECLARE name CHAR(40);
  OPEN cur_employee;
  LOOP
    FETCH NEXT cur_employee INTO name;
    ...
  END LOOP;
  CLOSE cur_employee;
END
```

使用法

DECLARE CURSOR 文は、**SELECT** 文または **CALL** 文に対して指定された名前を持つカーソルを宣言します。

SCROLL—SCROLL に宣言されたカーソルは、**FETCH** 文の NEXT、PRIOR、FIRST、LAST、ABSOLUTE、RELATIVE のオプションをサポートします。SCROLL カーソルでは、カーソルが開いている間、結果セットにある任意のローをフェッチすることができます。

NO SCROLL—NO SCROLL に宣言されたカーソルでは、**FETCH NEXT** と **FETCH ABSOLUTE (0)** のシーク操作しか使用しないため、結果セット内の移動が前方向に制限されます。

DYNAMIC SCROLL—DYNAMIC SCROLL に宣言されたカーソルは、**FETCH** 文の NEXT、PRIOR、FIRST、LAST、ABSOLUTE、RELATIVE のオプションをサポートします。DYNAMIC SCROLL カーソルでは、カーソルが開いている間、結果セットにある任意のローをフェッチすることができます。

一度カーソルがローを離れたら、そのローに戻ることはできないため、sensitivity の制限はこのカーソルにはありません。 したがって、NO SCROLL カーソルが要

求されると、Sybase IQ は最も効率的なカーソル、すなわち asensitive カーソルを提供します。

FOR statement-name—**PREPARE** 文を使用して文に名前を付けます。カーソルを宣言できるのは、準備された **SELECT** または **CALL** に対してだけです。

FOR READ ONLY—FOR READ ONLY に宣言されたカーソルは、位置付け **UPDATE** や位置付け **DELETE** の処理に使用できません。**READ ONLY** は **FOR** 句のデフォルト値です。

FOR READ ONLY として宣言されたカーソルからは、最初に **FETCH** を実行したときのテーブルのバージョンではなく、カーソルをオープンしたときにカーソルの宣言に使用されるテーブルのバージョンが表示されます。

たとえば、カーソルがフェッチされるとき、テーブルからフェッチできるローは 1 つだけです。

```
CREATE TABLE t1 ( c1 INT );
INSERT t1 VALUES ( 1 );

BEGIN
DECLARE t1_cursor CURSOR FOR SELECT * FROM t1
FOR READ ONLY;
OPEN t1_cursor;
INSERT t1 VALUES ( 2 );
FETCH T1_CURSOR;
END
```

FOR UPDATE—FOR UPDATE に宣言されたカーソルでは、カーソルの結果セットを更新できます。更新可能なカーソルには、asensitive 動作だけがサポートされます。その他の sensitivity はすべて無視されます。

カーソルが開くと、更新のために開かれたすべてのテーブルに排他テーブル・ロックがかけられます。 Sybase IQ では、テーブルを変更する文が一度に 1 つしか許可されないため、同じトランザクション内で更新用に開かれたテーブルに対するスタンドアロンの **LOAD TABLE** 文、**UPDATE** 文、**INSERT** 文、**DELETE** 文、**TRUNCATE** 文は許可されません。 特定のテーブルで、一度に開くことができる更新可能なカーソルは 1 つだけです。

更新可能なカーソルは、Open Client に対する場合を除き、スクロール可能です。

OF column-name-list—更新可能として定義されたカーソル結果セット (*select-statement* によって指定) のカラムのリストです。

USING variable-name—ストアド・プロシージャやユーザ定義関数の変数に、カーソルを宣言できます。この変数はカーソルの **SELECT** 文を含む文字列です。変数は **DECLARE** が処理されるときに使用可能でなくてはなりません。したがって、次のいずれかの方法で定義してください。

- プロシージャへのパラメータ。次に例を示します。

```
create function get_row_count(in qry varchar)
returns int
begin
    declare crsr cursor using qry;
    declare rowcnt int;

    set rowcnt = 0;
    open crsr;
    lp: loop
        fetch crsr;
        if SQLCODE <> 0 then leave lp end if;
        set rowcnt = rowcnt + 1;
    end loop;
    return rowcnt;
end
```

- 変数に値が割り当てられた後、別の **BEGIN...END** の内部にネストされる。次に例を示します。

```
create procedure get_table_name(
  in id_value int, out tabname char(128))

begin
  declare qry varchar;

  set qry = 'select table_name from SYS.ISYSTAB ' ||
        'where table_id=' || string(id_value);
  begin
    declare crsr cursor using qry;

    open crsr;
    fetch crsr into tabname;
    close crsr;
 end
end
```

Embedded SQL

PREPARE 文を使用して文に名前を付けます。カーソルを宣言できるのは、準備された **SELECT** または **CALL** に対してだけです。

更新可能なカーソルのサポート

Sybase IQ での更新可能なカーソルのサポートは、SQL Anywhere での更新可能なカーソルのサポートと似ています。 カーソルのタイプとカーソルの操作の詳細については、『SQL Anywhere サーバー - プログラミング』の「アプリケーションでの SQL の使用」>「カーソルの概要」を参照してください。

注意： このリファレンスは SQL Anywhere マニュアルにリンクされています。

Sybase IQ でサポートされるカーソルの sensitivity は 1 種類です。この sensitivity は、基礎データへの変更が可視であることを基準に決められます。 Sybase IQ のカーソルはすべて asensitive です。つまり、結果セットのメンバシップ、順序、ま

たは値に反映される変更は、カーソルを通して確認できるか、まったく反映されないかのどちらかです。

asensitive カーソルを使用すると、位置付け **UPDATE** 文と位置付け **DELETE** 文による変更がカーソル結果セット内で可視になります。ただし、クライアント・サイドのキャッシュによってこれらの変更の確認が妨げられた場合は確認できません。挿入されたローは確認できません。

ローが更新され、開いているカーソルの **WHERE** 句の要件を満たさなくなった場合でも、そのローは可視のままです。

カーソルの使用には、効率と一貫性のトレードオフが常に伴います。asensitive カーソルの場合、パフォーマンスは効率化されますが、その代償として一貫性が損なわれます。

Sybase IQ では、1 つのテーブルに対する更新可能なカーソルがサポートされます。

`LONG VARCHAR` データ型と `LONG BINARY` データ型は、更新可能なカーソルではサポートされません。Sybase IQ の `LONG VARCHAR` データ型と `LONG BINARY` データ型については、『Sybase IQ の非構造化データ分析の概要』を参照してください。

スカラ・ユーザ定義関数とユーザ定義の集合関数は、更新可能なカーソルではサポートされません。

次に、Sybase IQ における更新可能なカーソルに対してサポートされるクエリの仕様を示します。

- select リスト内の式は、更新されるカラムに機能的に依存しないカラムに対するものである。
- 任意のサブクエリが asensitive の動作を持つ。すなわち、サブクエリから参照されるデータへの変更は、カーソル結果セット内で可視にならない。
- **ORDER BY** 句については、**ORDER BY** を指定したカラムも更新可能だが、結果セットが再度順序付けされることはない。
- カラムは次の要件を満たす。
  - カラムで CAST が行われない。
  - **SELECT** 句のベース・テーブルのベース・カラムである。
  - **SELECT** 句のカラムを参照する式や関数が存在しない。また、select リスト内でカラムが重複しない (`SELECT c1, c1` など)。
  - **FOR UPDATE OF***column-name-list* 句が指定された場合、ベース・テーブルのベース・カラムは、この句にリストされたカラムに制限される。

Sybase IQ では、結果セットのローとベース・テーブルのローの、1 対 1 の関係を妨げる演算子を含むクエリに対して、更新可能なカーソルを使用することはできません。具体的には、以下が挙げられます。

- **SELECT DISTINCT**
- **UNION** が指定された演算子
- **GROUP BY** が指定された演算子
- **SET** 関数が指定された演算子
- **OLAP** 関数が定義された演算子。ただし、**RANK()** を除く

カーソルの結果セット内のローを更新する **SET** 句に指定可能なカラムと式については、「UPDATE (位置付け) 文 [ESQL] [SP]」の説明を参照してください。

Sybase IQ で更新可能なカーソルへの挿入がサポートされるのは、NULL が扱えないカラムと IDENTITY でないカラムがすべて選択され、かつ更新可能である場合だけです。

Sybase IQ では、**COMMIT** と **ROLLBACK** を、開いている更新可能なカーソルの内部で実行することはできません。Sybase IQ は、更新可能なカーソル内部での **ROLLBACK TO SAVEPOINT** をサポートします。

カーソルを開いた後でエラーが発生すると、このカーソルをとおして実行されたすべての処理がロールバックされます。

更新可能なカーソルの制限

次のケースでは、宣言されたカーソルは読み込み専用となり、更新可能になりません。

- パス名に `TEMP_EXTRACT_NAME1` オプションが設定され、データ抽出機能が有効になっている。
- ジョイン・インデックスとして宣言されるか、ジョイン・インデックスに含まれる。
- `ANSI_CLOSE_CURSORS_ON_ROLLBACK` が OFF に設定されている。
- CHAINED が OFF に設定されている。
- 文が **INSERT SELECT** または **SELECT INTO** である。
- 複数のテーブルが含まれる。
- 更新可能なカラムが存在しない。

Sybase IQ が更新可能なカーソルを要求されて設定に失敗した場合は、`.iqmsg` ファイルを参照して関連情報を調べてください。

更新可能カーソルと ODBC に関しては、制限事項があります。次の ODBC 関数を使用して一度に更新、削除、または挿入できる最大ロー数またはレコード数は、65535 です。

- **SQLSetPos** (**SQL_UPDATE**、**SQL_DELETE**、**SQL_ADD**)

- **SQLBulkOperations** (**SQL_ADD**、**SQL_UPDATE_BY_BOOKMARK**、**SQL_DELETE_BY_BOOKMARK**)

影響されるロー数を制御する文の属性の最大値が UNSIGNED SMALL INT の最大値 (65535) に制限されるという、実装固有の制限事項があります。

```
SQLSetStmtAttr(HANDLE,SQL_ATTR_ROW_ARRAY_SIZE, VALUE,0)
```

更新可能なカーソルの違い

Sybase IQ の更新可能カーソルには、ANSI SQL3 標準の動作と比較して次の相違点があります。

- ホールド・カーソルの更新がコミット時にクローズする。
- Sybase IQ はカーソルが開いたときにテーブルをロックする。
- 更新、削除、挿入の処理は、カーソルが閉じたとき、最初に削除、次に更新、最後に挿入の順序で適用される。

『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「システム・プロシージャ」>「システム・ストアド・プロシージャ」>「sp_iqcursorinfo プロシージャ」を参照してください。

標準

- SQL—ISO/ANSI SQL 文法のベンダ拡張。
- 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

参照：

- CALL 文 (66 ページ)
- DELETE (位置付け) 文 [ESQL] [SP] (203 ページ)
- FETCH 文 [ESQL] [SP] (230 ページ)
- OPEN 文 [ESQL] [SP] (301 ページ)
- PREPARE 文 [ESQL] (309 ページ)
- SELECT 文 (339 ページ)
- UPDATE (位置付け) 文 [ESQL] [SP] (376 ページ)

DECLARE CURSOR 文 [T-SQL]

Adaptive Server Enterprise と互換性がある方法でカーソルを宣言します。

構文

```
DECLARE cursor-name
… CURSOR FOR select-statement
…[ FOR { READ ONLY | UPDATE } ]
```

使用法

Sybase IQ がサポートする **DECLARE CURSOR** 構文は、Adaptive Server Enterprise ではサポートされません。 **DECLARE CURSOR** 構文の詳細については、「DECLARE CURSOR 文 [ESQL] [SP]」を参照してください。

『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「システム・プロシージャ」>「システム・ストアド・プロシージャ」>「sp_iqcursorinfo プロシージャ」を参照してください。

標準

- SQL—FOR UPDATE オプションと FOR READ ONLY オプションは、ISO/ANSI SQL 文法の Transact-SQL 拡張機能です。
- Sybase—Adaptive Server Enterprise**DECLARE CURSOR** 文の機能の中には、Sybase IQ でサポートされないものがいくつかあります。
  - Sybase IQ 言語では、**DECLARE CURSOR** をプロシージャまたはバッチ内の BEGIN キーワードの直後に置く必要があります。 Transact-SQL 言語では、このような制限はありません。
  - Adaptive Server Enterprise では、カーソルがプロシージャまたはバッチ内で宣言されている場合、そのプロシージャまたはバッチの間存在します。Sybase IQ では、カーソルを複合文内で宣言した場合、(Sybase IQ と Transact-SQL のどちらの複合文で宣言されているかにかかわらず) カーソルは複合文の間だけ存在します。

パーミッション

なし

参照：

- DECLARE CURSOR 文 [ESQL] [SP] (191 ページ)

DECLARE LOCAL TEMPORARY TABLE 文

ローカル・テンポラリ・テーブルを宣言します。

構文

```
DECLARE LOCAL TEMPORARY TABLE table-name
… ( column-definition [ column-constraint ] …
[ , column-definition [ column-constraint ] … ]
[ , table-constraint ] … )
…[ ON COMMIT { DELETE | PRESERVE } ROWS
NOT TRANSACTIONAL]
```

例

- **例 1 –** 次の例は、Embedded SQL でのローカル・テンポラリ・テーブルの宣言方法を示します。

```
EXEC SQL DECLARE LOCAL TEMPORARY TABLE MyTable (
  number INT
  );
```

- **例 2 –** 次の例は、ストアド・プロシージャでのローカル・テンポラリ・テーブルの宣言方法を示します。

```
BEGIN
  DECLARE LOCAL TEMPORARY TABLE TempTab (
    number INT
  );
  ...
END
```

使用法

DECLARE LOCAL TEMPORARY TABLE 文はテンポラリ・テーブルを宣言します。

ローカル・テンポラリ・テーブルとそのローは、テーブルを作成し、ローを挿入した接続からしか見えません。デフォルトで、テンポラリ・テーブルのローは **COMMIT** のときに削除されます。

複合文内で宣言したローカル・テンポラリ・テーブルは、複合文内に存在します。それ以外の場合、宣言したローカル・テンポラリ・テーブルは接続の終了時まで存在します。

column-definition、*column-constraint*、*table-constraint*、NOT TRANSACTIONAL 句の定義については、「CREATE TABLE 文」を参照してください。テンポラリ・テーブルへのデータの選択方法の例については、「SELECT 文」を参照してください。

ローカル・テンポラリ・テーブルを作成すると、それが明示的であるかどうかにかかわらず、テンポラリ・テーブルが存在している間は別のテンポラリ・テーブ

ルを同じ名前で作成できません。たとえば、次のように入力すると、ローカル・テンポラリ・テーブルが自動的に作成されます。

```
select * into #tmp from table1
```

または次のように宣言して、ローカル・テンポラリ・テーブルを明示的に作成することもできます。

```
declare local temporary table foo
```

その後、#tmp または foo に対して SELECT INTO を実行するか、#tmp または foo をもう一度宣言すると、#tmp または foo がすでに存在することを示すエラーが表示されます。

ローカル・テンポラリ・テーブルを宣言するときは、所有者を指定しないでください。同じセッションの複数の **DECLARE LOCAL TEMPORARY TABLE** 文で同じ owner.table を指定すると、構文エラーが報告されます。たとえば、次の文を同じセッション内で実行すると、エラーがレポートされます。

```
DECLARE LOCAL TEMPORARY TABLE user1.temp(col1 int);
DECLARE LOCAL TEMPORARY TABLE user1.temp(col1 int);
```

所有者名を省略した場合は、エラー "Item temp already exists" がレポートされます。

```
DECLARE LOCAL TEMPORARY TABLE temp(col1 int);
DECLARE LOCAL TEMPORARY TABLE temp(col1 int);
```

その接続に同じ名前のローカル・テンポラリ・テーブルがある場合、ベース・テーブルまたはグローバル・テンポラリ・テーブルを作成しようとしても失敗します。これは、新しいテーブルを *owner.table* がユニークに識別できないからです。

ただし、既存のベース・テーブルまたはグローバル・テンポラリ・テーブルとしてなら、ローカル・テンポラリ・テーブルを同じ名前で作成できます。テーブル名への参照は、ローカル・テンポラリ・テーブルにアクセスします。ローカル・テンポラリ・テーブルが最初に解決されるからです。

次のシーケンス例を見てみましょう。

```
CREATE TABLE t1 (c1 int);
INSERT t1 VALUES (9);

DECLARE LOCAL TEMPORARY TABLE t1 (c1 int);
INSERT t1 VALUES (8);

SELECT * FROM t1;
```

返される結果は 8 です。ローカル・テンポラリ・テーブルが接続により削除されるまで、t1 に対する参照はいずれも、ローカル・テンポラリ・テーブル t1 を参照します。

ローカル・テンポラリ・テーブルに対して、**ALTER TABLE** 文や **DROP INDEX** 文を使用することはできません。

ローカル・テンポラリ・テーブルに対して、**sp_iqindex**、**sp_iqtablesize**、**sp_iqindexsize** のストアド・プロシージャを使用することはできません。

標準

- SQL—ISO/ANSI SQL 文法のベンダ拡張。
- Sybase—Adaptive Server Enterprise は **DECLARE TEMPORARY TABLE** をサポートしません。

パーミッション

なし

参照：

- CREATE TABLE 文 (166 ページ)
- SELECT 文 (339 ページ)

DELETE 文

データベースからローを削除します。

構文

```
DELETE [ FROM ] [ owner.]table-name
…[ FROM table-list ]
…[ WHERE search-condition ]
```

例

- **例 1 –** データベースから従業員番号 105 の従業員を削除します。

  ```
  DELETE
  FROM Employees
  WHERE EmployeeID = 105
  ```

- **例 2 –** `FinancialData` テーブルから、1993 年より前のデータをすべて削除します。

  ```
  DELETE
  FROM FinancialData
  WHERE Year < 1993
  ```

- **例 3 –** `Customers` テーブルにすでに名前がある場合、その名前をすべて `Contacts` テーブルから削除します。

```
DELETE
FROM Contacts
FROM Contacts, Customers
WHERE Contacts.Surname = Customers.Surname
AND Contacts.GivenName = Customers.GivenName
```

使用法

DELETE 文は、指定したテーブルから、検索条件を満たすすべてのローを削除します。**WHERE** 句を指定しない場合、指定したテーブルからすべてのローが削除されます。

ビューを定義する **SELECT** 文が **FROM** 句の中でテーブルを 1 つだけ持ち、**GROUP BY** 句と集合関数を含まないか、または **UNION** 操作を伴わない場合は、**DELETE** 文をビュー上で使用できます。

DELETE 文内のオプションの 2 番目の **FROM** 句は、ジョインに基づいてローを削除できるようにします。2 番目の **FROM** 句がある場合、**WHERE** 句はこの 2 番目の **FROM** 句のローを修飾します。ローは、最初の **FROM** 句内で指定したテーブルから削除されます。

テーブルに対する **DELETE** は、**SYNCHRONIZE JOIN INDEX** コマンドを介してそのテーブルを参照するどのジョイン・インデックスに対しても影響します。パフォーマンス上の理由から、ジョイン・インデックスを同期させる前に、可能なかぎり多くの削除を行います。

注意： ジョイン仮想テーブルに対して **DELETE** 文を使用することはできません。ジョイン仮想テーブルから削除しようとすると、エラーがレポートされます。

相関名の解析

次の文は、相関名を使用する 2 つの **FROM** 句を持つ **DELETE** 文内のテーブル名が、あいまいになる可能性があることを示しています。

```
DELETE
FROM table_1
FROM table_1 AS alias_1, table_2 AS alias_2
WHERE ...
```

テーブル `table_1` は、最初の **FROM** 句の中では相関名を使用せずに識別され、2 番目の **FROM** 句内では相関名を使用して識別されます。この場合、最初の FROM 句の `table_1` が、2 番目の FROM 句では `alias_1` と識別されます。この文の中に **table_1** のインスタンスは 1 つだけです。

これは、同じ文の中で相関名を使用する方法と使用しない方法の両方を使用してテーブルを識別する場合には、テーブルの 2 つのインスタンスが考えられるという、一般規則の例外です。

次の例を考えます。

```
DELETE
FROM table_1
FROM table_1 AS alias_1, table_1 AS alias_2
WHERE ...
```

ここでは、2 番目の **FROM** 句に table_1 のインスタンスが 2 つあります。この場合、どちらのインスタンスで最初の **FROM** 句を識別するのかが判断できません。相関名の一般的な規則を適用して、最初の **FROM** 句の table_1 は、2 番目の FROM 句のいずれのインスタンスによっても識別されません。この文の中には table_1 のインスタンスが 3 つあります。

標準

- SQL—ISO/ANSI SQL 文法のベンダ拡張。
- Sybase—ベンダ拡張を含み、Adaptive Server Enterprise でサポートされています。

パーミッション

テーブルに対する DELETE パーミッションが必要です。

参照：

- FROM 句 (237 ページ)
- INSERT 文 (258 ページ)
- SYNCHRONIZE JOIN INDEX 文 (367 ページ)
- TRUNCATE TABLE 文 (369 ページ)

DELETE (位置付け) 文 [ESQL] [SP]

カーソルの現在の位置にあるデータを削除します。

構文

```
DELETE [ FROM table-spec ]
WHERE CURRENT OF cursor-name
```

パラメータ

- **cursor-name：** – *識別子* | *hostvar*
- **table-spec：** – [*owner.*]*correlation-name*
- **owner：** – *識別子*

例

- **例 1 –** データベースから現在のローを削除します。

```
DELETE WHERE CURRENT OF cur_employee
```

使用法

この形式の **DELETE** 文は、指定されたカーソルの現在のローを削除します。現在のローとは、直前にカーソルからフェッチされたローのことです。

ローが削除されるテーブルは次のように決定されます。

- **FROM** 句が含まれていない場合、カーソルは 1 つのテーブルのみに配置できる。
- カーソルがジョイン・クエリ用の場合は (ジョインがあるビューの使用を含む)、**FROM** 句を使用する必要がある。指定したテーブルの現在のローだけが削除される。ジョインに含まれた他のテーブルは影響を受けない。
- **FROM** 句を含め、テーブル所有者を指定しない場合は *table-spec* がどの相関名に対しても最初に一致する。
  - 相関名がある場合、*table-spec* は相関名で識別される。
  - 相関名がない場合、*table-spec* はカーソルのテーブル名として明確に識別できるようにする必要がある。
- **FROM** 句が含まれ、テーブル所有者が指定されている場合、テーブル仕様値はカーソルのテーブル名として明確に識別できるようにする必要がある。

位置付け **DELETE** 文はビューでカーソルを開くときに使用できます。ただし、ビューが更新可能である場合にかぎられます。

位置付け **DELETE** 文による変更は、カーソル結果セット内で確認できます。ただし、クライアント・サイドのキャッシュによって、これらの変更の確認が妨げられた場合を除きます。

『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「システム・プロシージャ」>「システム・ストアド・プロシージャ」>「sp_iqcursorinfo プロシージャ」を参照してください。

標準

- SQL—`ANSI_UPDATE_CONSTRAINTS` オプションが OFF に設定されている場合、更新可能なカーソルの範囲に ISO/ANSI SQL 文法のベンダ拡張が含まれる可能性があります。
- Sybase—Embedded SQL の使用は、Open Client/Open Server でサポートされています。 プロシージャとトリガの使用は、SQL Anywhere でサポートされます。

パーミッション

カーソルで使用するテーブルの DELETE パーミッションが必要です。

参照：

- DECLARE CURSOR 文 [ESQL] [SP] (191 ページ)
- INSERT 文 (258 ページ)
- UPDATE 文 (372 ページ)
- UPDATE (位置付け) 文 [ESQL] [SP] (376 ページ)

DESCRIBE 文 [ESQL]

データベースから取り出したデータを格納するために必要なホスト変数、またはデータベースにデータを渡すために使用されるホスト変数に関する情報を取得します。

構文

```
DESCRIBE
...[ USER TYPES ]
...[ { ALL | BIND VARIABLES FOR | INPUT
| OUTPUT | SELECT LIST FOR } ]
...[ { LONG NAMES [ long-name-spec ] | WITH VARIABLE RESULT } ]
...[ FOR ] { statement-name | CURSOR cursor-name }
...INTO sqlda-name
```

パラメータ

- **long-name-spec：** – { **OWNER.TABLE.COLUMN** | **TABLE.COLUMN** | **COLUMN** }
- **statement-name：** – 識別子 | ホスト変数
- **cursor-name：** – 宣言されたカーソル
- **sqlda-name：** – 識別子

例

- **例 1** – 次の例は、**DESCRIBE** 文の使用法を示します。

```
sqlda = alloc_sqlda( 3 );
EXEC SQL DESCRIBE OUTPUT
  FOR employee_statement
  INTO sqlda;
if( sqlda->sqld  >  sqlda->sqln ) {
  actual_size = sqlda->sqld;
  free_sqlda( sqlda );
  sqlda = alloc_sqlda( actual_size );
  EXEC SQL DESCRIBE OUTPUT
    FOR employee_statement
```

```
    INTO sqlda;
}
```

使用法

DESCRIBE は、指定した SQLDA を設定し、指定した文に対して **OUTPUT (SELECT LIST** と同じ) または **INPUT (BIND VARIABLES)** を記述します。

INPUT の場合、**DESCRIBE BIND VARIABLES** は SQLDA の中にデータ型を設定しません。アプリケーションがデータ型を設定する必要があります。ALL キーワードを使用すると、1 つの SQLDA の中に **INPUT** と **OUTPUT** を記述できます。

文名を指定する場合、同じ文名で **PREPARE** 文を使用して、文を事前に作成しておく必要があります。また、事前に SQLDA を割り当てておく必要があります (「ALLOCATE DESCRIPTOR 文 [ESQL]」を参照してください)。

カーソル名を指定する場合、カーソルを事前に宣言し、オープンしておく必要があります。デフォルトのアクションでは、**OUTPUT** を記述します。**OUTPUT** を持つのは、**SELECT** 文と **CALL** 文だけです。それ以外の文または動的なカーソルではないカーソルでは、**DESCRIBE OUTPUT** は、SQLDA の **sqld** フィールドを 0 に設定して出力がないことを示します。

USER TYPES—**USER TYPES** 句を持つ **DESCRIBE** 文は、カラムのユーザ定義データ型に関する情報を返します。通常、このような **DESCRIBE** は、以前の **DESCRIBE** が DT_HAS_USERTYPE_INFO のインジケータを返したときに行われます。

返される情報は、**USER TYPES** キーワードを付けない **DESCRIBE** の場合と同じですが、**sqlname** フィールドがカラム名の代わりにユーザ定義データ型の名前を保持する点が異なります。

DESCRIBE が **LONG NAMES** 句を使用する場合、**sqldata** フィールドがこの情報を保持します。

SELECT—**DESCRIBE OUTPUT** は、それぞれの select リスト項目に対する SQLDA でのデータ型と長さを入力します。名前フィールドにも、select リスト項目の名前が入ります。select リスト項目に対してエイリアスが指定されている場合、名前はそのエイリアスになります。エイリアスが指定されていない場合、select リスト項目から派生する名前が付けられます。項目が単純なカラム名である場合は、その名前を使用します。そうでない場合は、式の部分文字列を使用します。また、**DESCRIBE** は、select リスト項目の数を SQLDA の **sqld** フィールドに格納します。

記述されている文が 2 つ以上の **SELECT** 文の **UNION** である場合、**DESCRIBE OUTPUT** に対して返されるカラム名は、最初の **SELECT** 文に対して返されるカラム名と同じです。

CALL—**DESCRIBE OUTPUT** 文は、プロシージャ内の **INOUT** パラメータまたは **OUT** パラメータごとに、SQLDA でのデータ型、長さ、名前を入力します。また、

DESCRIBE OUTPUT は、**INOUT** パラメータまたは **OUT** パラメータの数を SQLDA の **sqld** フィールドに格納します。

CALL (結果セット)—**DESCRIBE OUTPUT** 文は、プロシージャ定義の各 RESULT カラムに対する SQLDA でのデータ型、長さ、名前を入力します。また、**DESCRIBE OUTPUT** は、結果カラムの数を SQLDA の **sqld** フィールドに格納します。

INPUT—バインド変数は、データベースが文を実行するときにアプリケーションによって提供される値です。バインド変数は、文に対するパラメータと考えることができます。**DESCRIBE INPUT** は、SQLDA の名前フィールドにバインド変数名を入力します。また、**DESCRIBE INPUT** は、バインド変数の数を SQLDA の **sqld** フィールドに格納します。

DESCRIBE は、SQLDA の中のインジケータ変数を使って情報を追加します。DT_PROCEDURE_IN と DT_PROCEDURE_OUT は、**CALL** 文を記述するときに、インジケータ変数の中に設定されるビットです。DT_PROCEDURE_IN は **IN** パラメータまたは **INOUT** パラメータを示し、DT_PROCEDURE_OUT は **INOUT** パラメータまたは **OUT** パラメータを示します。プロシージャの RESULT カラムは、両方のビットをクリアします。DESCRIBE **OUTPUT** の後、これらのビットは結果セットを持っている文 (**OPEN**、**FETCH**、**RESUME**、**CLOSE** を使用する必要があります) と持っていない文 (**EXECUTE** を使用する必要があります) を区別するのに使用できます。**DESCRIBE INPUT** は、バインド変数が **CALL** 文に対する引数であるときに、適切に DT_PROCEDURE_IN と DT_PROCEDURE_OUT を設定するだけです。**CALL** 文の引数である式内のバインド変数は、ビットを設定します。

DESCRIBE ALL を使用すると、データベース・サーバに対して 1 回要求するだけで **INPUT** と **OUTPUT** を記述できます。マルチユーザ環境では、これがパフォーマンスによい影響を与えます。**INPUT** 情報が最初に SQLDA に格納され、次に **OUTPUT** 情報が格納されます。**sqld** フィールドには、**INPUT** と **OUTPUT** 変数の総数が入ります。インジケータ変数内の DT_DESCRIBE_INPUT ビットは **INPUT** 変数に対して設定され、**OUTPUT** 変数に対してはクリアされます。

- **長いカラム名の取り出し – LONG NAMES** 句は、文またはカーソルに対応するカラム名を取得するために用意されています。 この句がなければ、カラム名の長さは 29 文字に制限されます。 この句を指定すると、任意の長さの名前がサポートされます。

  LONG NAMES を使用した場合、カーソルからフェッチを行う場合と同様に、長い名前は SQLDA の SQLDATA フィールドに格納されます。 これ以外のフィールド (SQLLEN、SQLTYPE など) にはどれも入力されません。SQLDA は **FETCH** SQLDA のようにセットアップしてください。つまり、各カラムには、文字型のデータのエントリを 1 つ含めます。

長い名前のデフォルトの仕様は、TABLE.COLUMN です。

- **変数結果セットの記述 – WITH VARIABLE RESULT** 文は、複数の結果セットと異なる数または型のカラムを持つプロシージャの記述に使用します。

  WITH VARIABLE RESULT を使用する場合、記述後にデータベース・サーバによって SQLCOUNT 値に次のいずれかの値が設定されます。

  - 0—結果セットは変更される場合があります。各 **OPEN** 文の後でプロシージャ呼び出しを再度記述してください。
  - 1—結果セットは固定です。 再度記述する必要はありません。

SQLDA 構造体の使用法の詳細については、『SQL Anywhere サーバー - プログラミング』の「Embedded SQL」>「SQLDA (SQL descriptor area)」を参照してください。

注意： このリファレンスは SQL Anywhere マニュアルにリンクされています。

標準

- SQL—いくつかの句は、ISO/ANSI SQL 文法のベンダ拡張です。
- Sybase—いくつかの句は Open Client/Open Server でサポートされています。

パーミッション

なし

参照：

- ALLOCATE DESCRIPTOR 文 [ESQL] (5 ページ)
- DECLARE CURSOR 文 [ESQL] [SP] (191 ページ)
- OPEN 文 [ESQL] [SP] (301 ページ)
- PREPARE 文 [ESQL] (309 ページ)

DISCONNECT 文 [Interactive SQL]

データベースとの接続を削除します。

構文

```
DISCONNECT [ { connection-name | CURRENT | ALL } ]
```

パラメータ

- **connection-name：** – 識別子、文字列、またはホスト変数

例

- **例 1 –** 次の文は、Embedded SQL 内での **DISCONNECT** の使用法を示します。

```
EXEC SQL DISCONNECT :conn_name
```

- **例 2 –** 次の文は、**dbisql** から **DISCONNECT** を使用し、すべての接続を切断する方法を示します。

```
DISCONNECT ALL
```

使用法

DISCONNECT 文はデータベース・サーバとの接続を切断し、データベース・サーバが使用するすべてのリソースを解放します。切断しようとする接続の名前が **CONNECT** 文上で指定されている場合、その名前を使用して接続を指定できます。**ALL** を指定すると、すべてのデータベース環境へのアプリケーションの接続すべてが切断されます。デフォルトは **CURRENT** で、現在の接続が切断されます。

接続が切断されると、暗黙の **ROLLBACK** が実行されます。

標準

- SQL—ISO/ANSI SQL 準拠。
- 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

参照：

- CONNECT 文 [ESQL] [Interactive SQL] (77 ページ)
- SET CONNECTION 文 [ESQL] [Interactive SQL] (354 ページ)

DROP 文

データベースからオブジェクトを削除します。

構文

```
DROP
{ DBSPACE dbspace-name
| { DATATYPE [ IF EXISTS ]
| DOMAIN [ IF EXISTS ] } datatype-name
| EVENT [ IF EXISTS ] event-name
| INDEX [ IF EXISTS ] [ [ owner].table-name.]index-name
| JOIN INDEX [ owner.]join-index-name
| MESSAGE message-number
```

```
| TABLE [ IF EXISTS ]  [ owner.]table-name
| VIEW [ IF EXISTS ]  [ owner.]view-name
| PROCEDURE [ IF EXISTS ]  [ owner.]procedure-name
| FUNCTION [ IF EXISTS ]  [ owner.]function-name }
```

例

- **例 1 –** データベースから Departments テーブルを削除します。

  ```
  DROP TABLE Departments
  ```

- **例 2 –** データベースから emp_dept ビューを削除します。

  ```
  DROP VIEW emp_dept
  ```

使用法

DROP は、指定したデータベース構造の定義を削除します。構造が DB 領域である場合、DB 領域を削除する前に、DB 領域内のデータを含むすべてのテーブルを削除または移動する必要があります。これ以外の構造の場合は自動的に移動されます。構造がテーブルである場合、テーブル内のデータは削除プロセスの一部として自動的に削除されます。同様に、テーブルのすべてのインデックスとキーも、**DROP TABLE** によって削除されます。ただし、そのテーブルを使用するジョイン・インデックスが存在する場合はそのテーブルを削除することはできません。最初に **DROP JOIN INDEX** を使用して、ジョイン・インデックスを削除する必要があります。

IF EXISTS 句は、存在しないデータベース・オブジェクトをこの **DROP** 文が削除しようとしたときにエラーを返さないようにする場合に使用します。

DROP INDEX では、明示的に作成されたインデックスがすべて削除されます。一意性制約または外部キー制約、または関連付けられたプライマリ・キーがない場合は、暗黙的に作成されたインデックスだけが削除されます。

ユニークでない **HG** インデックスに **DROP INDEX** を実行した場合、強制力のない外部キーが関連付けられているとエラーになります。

警告！ DBO ユーザが所有するビューを削除しないでください。このようなビューを削除したり、テーブルを変更したりすると、問題が起きる可能性があります。

DROP TABLE、**DROP INDEX**、**DROP JOIN INDEX**、**DROP DBSPACE** は、この文が別の接続によって現在使用中であるテーブルに影響を与える場合は、実行できません。

DROP TABLE は、プライマリ・テーブルに外部キー制約 (強制力のない外部キー制約を含む) が関連付けられている場合は処理されません。

DROP TABLE は、テーブルに IDENTITY カラムがあって IDENTITY_INSERT がこのテーブルに設定されている場合も処理されません。そのテーブルを削除するには、IDENTITY_INSERT をクリア、すなわち空の文字列 (' ') に設定するか、別のテーブル名に設定します。

外部キーは、ユニークでないシングルまたはマルチカラムの **HG** インデックスを持つことができます。プライマリ・キーは、ユニークなシングルまたはマルチカラムの **HG** インデックスを持つことができます。既存の外部キー制約、プライマリ・キー制約、一意性制約に対して自動的に作成された **HG** インデックスを削除することはできません。DBA が別のユーザに属するジョイン・インデックスを削除する場合、そのジョイン・インデックス名を、所有者名で修飾する必要があります。

初期の DB 領域は、SYSTEM、IQ_SYSTEM_MAIN、IQ_SYSTEM_TEMP、IQ_SYSTEM_MSG の 4 つです。これらの初期の DB 領域を削除することはできませんが、複数の DB 領域を含んだ IQ メイン・ストアまたはカタログ・ストアからは DB 領域を削除できます。ただし、少なくとも 1 つの DB 領域が読み書きモードのままである必要があります。

DB 領域内のテーブルを削除しなければ、DB 領域を削除することはできません。DB 領域にユーザ・データが残っていればエラーが返されます。これ以外の構造は、DB 領域が削除されるとき自動的に移動されます。DB 領域を削除できるのは、DB 領域を読み込み専用に指定した後だけです。

注意： コマンドによって使用された後の DB 領域には、コマンド使用後のどの時点のデータも含まれている可能性があるので、その DB 領域に対する **DROP DBSPACE** は処理されません。

DB 領域を変更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「データベース・オブジェクトの管理」>「データの格納」を参照してください。

DROP PROCEDURE は、プロシージャが別の接続で使用中の場合は処理されません。

DROP DATATYPE は、データ型がテーブル内で使用されている場合、処理されません。データ型を削除するには、ユーザ定義データ型を持つすべてのカラムのデータ型を変更する必要があります。**DROP DATATYPE** よりも **DROP DOMAIN** を使用することをおすすめします。これは、**DROP DOMAIN** が ANSI/ISO SQL3 の草案で使用されている構文であるためです。

関連する動作

- オートコミット。**dbisql** のデータ・ウィンドウをクリアします。**DROP TABLE** と **DROP INDEX** は、現在の接続のすべてのカーソルをクローズします。
- ローカル・テンポラリ・テーブルは例外です。削除されると、コミットは実行されません。

『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「システム・プロシージャ」>「システム・ストアド・プロシージャ」>「sp_iqdbspace プロシージャ」と『システム管理ガイド： 第 1 巻』の「データベース・オブジェクトの管理」を参照してください。

標準

- SQL—ISO/ANSI SQL 準拠。
- Sybase—Adaptive Server Enterprise でサポートされています。

パーミッション

DROP DBSPACE の場合は、DBA 権限または SPACE ADMIN 権限が必要です。また、このデータベースに他の接続がないことが必要です。

その他の場合は、オブジェクトの所有者になるか、DBA 権限が必要です。

グローバル・テンポラリ・テーブルは、このテンポラリ・テーブルを参照したすべてのユーザが切断されるまで削除できません。

DROP INDEX の場合、DBA 以外のユーザが DBA が所有するベース・テーブルのインデックスを削除するには、完全に修飾されたインデックス名を指定する必要があります。DBA または適切な権限を持つユーザは、完全に修飾された名前を使用しなくても、DBA 以外のユーザが所有するテーブルのインデックスを削除できます。

DROP CONNECTION 文

すべてのユーザのデータベースへの接続を削除します。

構文

```
DROP
CONNECTION
connection-id
```

例

- **例 1 –** ID 番号 4 の接続を削除します。

```
DROP CONNECTION 4
```

使用法

DROP CONNECTION は、データベースへの接続を削除してデータベースからユーザを切断します。現在の接続は削除できません。最初に別の接続を作成してから、最初の接続を削除する必要があります。

接続の *connection-id* は、**connection_property** 関数を使用し、接続番号を要求して取得します。次の文は、現在の接続の接続 ID を返します。

```
SELECT connection_property( 'number' )
```

標準

- 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。

パーミッション

DBA 権限または OPERATOR 権限が必要です。

参照：

- CONNECT 文 [ESQL] [Interactive SQL] (77 ページ)

DROP DATABASE 文

データベースとそれに関連する DB 領域セグメント・ファイルを削除します。

構文

DROP DATABASE *db-filename* [**KEY** *key-spec*]

パラメータ

- **key-spec：** – 大文字と小文字を混ぜた英数字、および特殊文字で構成される文字列。キーをコマンド・シェルによる解読または変更から保護するために必要です。

例

- **例 1 –** データベース `mydb` を削除します。

  ```
  DROP DATABASE 'mydb.db'
  ```
- **例 2 –** 次の文では、暗号化されたデータベース `marvin.db` を削除します。このデータベースは、キー `is!seCret` を使用して作成されています。

  ```
  DROP DATABASE 'marvin.db' KEY 'is!seCret'
  ```
- **例 3 –** 次の UNIX の例では、データベース `temp.db` を `/s1/temp` ディレクトリから削除します。

  ```
  DROP DATABASE '/s1/temp/temp.db'
  ```

使用法

DROP DATABASE は、カタログ・ストア・ファイルを削除する前に、IQ ストアとテンポラリ・ストアに関連付けられたすべてのデータベース・セグメント・ファイルを削除します。

データベースは、停止してから削除してください。接続パラメータ **AUTOSTOP=no** が使用されている場合、**STOP DATABASE** 文の実行が必要になる場合があります。

`db-filename` には、**CREATE DATABASE** を使用してデータベースに定義したデータベース・ファイル名を指定します。 **CREATE DATABASE** コマンドで、この値にディレクトリ・パスを指定した場合は、**DROP DATABASE** でもディレクトリ・パスを指定する必要があります。指定しないと、Sybase IQ は、サーバ・ファイルが格納されているデフォルト・ディレクトリ内のデータベース・ファイルを探します。

DatabaseStart イベントが定義された IQ データベースを、**DROP DATABASE** 文を実行して削除することはできません。

標準

- 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。

パーミッション

データベース・サーバのコマンド・ライン・オプション **-gu** を使用して、必要なパーミッションを設定します。デフォルトの設定は、DBA 権限を必要とします。

参照：

- CREATE DATABASE 文 (81 ページ)
- STOP DATABASE 文 [Interactive SQL] (365 ページ)

DROP EXTERNLOGIN 文

Sybase IQ システム・テーブルから外部ログインを削除します。

構文

```
DROP EXTERNLOGIN login-name
TO remote-server
```

例

- **例 1 –**

```
DROP EXTERNLOGIN dba TO sybase1
```

使用法

DROP EXTERNLOGIN による変更は、次にリモート・サーバに接続するまで有効になりません。

DROP EXTERNLOGIN は、Sybase IQ システム・テーブルから外部ログインを削除します。

login-name—ローカル・ユーザ・ログイン名を指定します。

TO—リモート・サーバ名を指定します。このサーバのローカル・ユーザの代替ログイン名とパスワードが、削除される外部ログインです。

関連する動作

- オートコミット。

標準

- SQL—ISO/ANSI SQL 準拠。
- Sybase—Open Client/Open Server でサポートされています。

パーミッション

DBA 権限または USER ADMIN 権限を持っている必要があります。

参照：
- CREATE EXTERNLOGIN 文 (108 ページ)

DROP LOGIN POLICY 文

ログイン・ポリシーをデータベースから削除します。

構文

```
DROP LOGIN POLICY
policy-name
```

例

- **例 1 –** 次の例では、`Test11` のログイン・ポリシーを作成してから削除します。

```
CREATE LOGIN POLICY Test11;
DROP LOGIN POLICY Test11 ;
```

使用法

DROP LOGIN POLICY 文は、ユーザに割り当てられたポリシーを削除しようとすると失敗します。**ALTER USER** 文を使用してユーザのポリシー割り当てを変更するか、**DROP USER** を使用してユーザを削除できます。

パーミッション

DBA 権限または USER ADMIN 権限を持っている必要があります。

参照：

- ALTER USER 文 (47 ページ)
- CREATE LOGIN POLICY 文 (132 ページ)
- DROP USER 文 (221 ページ)
- ALTER LOGIN POLICY 文 (22 ページ)

DROP LOGICAL SERVER 文

ユーザ定義の論理サーバを削除します。

構文

DROP LOGICAL SERVER *logical-server-name*

例

- **例 1 –** 次の例は、ユーザ定義の論理サーバ *ls1* を削除します。

```
DROP LOGICAL SERVER ls1
```

使用法

マルチプレックスのみに該当します。

Sybase IQ は、論理サーバの削除時に、次のカタログ変更を内部で実行します。

- 論理サーバのすべてのメンバシップ定義を削除します。
- 削除対象の論理サーバへの明示的な割り当てを含む各ログイン・ポリシーから論理サーバの割り当てを削除します。 削除対象の論理サーバがログイン・ポ

リシーに割り当てられている唯一の論理サーバの場合、Sybase IQ はそのログイン・ポリシーの論理サーバ割り当てを NONE に設定します。
- 論理サーバ・エントリを ISYSIQ.LOGICALSERVER から削除します。

パーミッション

DBA 権限または MULTIPLEX ADMIN 権限を持っている必要があります。

DROP MULTIPLEX SERVER 文

マルチプレックスからサーバを削除します。

構文

DROP MULTIPLEX SERVER {*server-name*} [*drop_mpx_server_clause*]

パラメータ

- **drop_mpx_server_clause： –** { **WITH DROP MEMBERSHIP** | **WITH DROP LOGICAL SERVER** }

例

- **例 1 –**

```
DROP MULTIPLEX SERVER writer1
```

使用法

マルチプレックスのみに該当します。

削除する前に各マルチプレックス・サーバを停止することをおすすめします。この文は自動的にコミットされます。

削除されるサーバが、停止されていない場合は、この文の実行後に自動的に停止します。

最後のセカンダリ・サーバを削除すると、マルチプレックスはシンプレックスに変換されます。マルチプレックス内の最後のセカンダリ・サーバの削除後に、コーディネータは自動的に停止します。必要に応じて再起動する必要があります。マルチプレックスがシンプレックスに変換されるときに適用される一連のルールの詳細については、セクション 3.3.1 を参照してください。

WITH DROP MEMBERSHIP 句 – **DROP MULTIPLEX SERVER** は、削除するマルチプレックス・サーバに、1 つまたは複数の論理サーバ・メンバシップが存在すると、エ

ラーで失敗します。**WITH DROP MEMBERSHIP** 句を指定すると、論理サーバはそのすべてのメンバシップとともに削除されます。

句 **WITH DROP LOGICAL SERVER** – 最後のセカンダリ・サーバを削除する場合、1 つまたは複数のユーザ定義の論理サーバが存在すると、**DROP MULTIPLEX SERVER** コマンドは失敗します。**WITH DROP LOGICAL SERVER** 句を指定すると、最後のセカンダリ・サーバはすべてのユーザ定義の論理サーバとともに削除されます。

注意： **WITH DROP LOGICAL SERVER** 句は、最後のセカンダリ・サーバを削除する場合のみ有効です。他の場合は、エラーが報告されます。

パーミッション

DBA 権限または MULTIPLEX ADMIN 権限を持っている必要があります。

DROP SERVER 文

Sybase IQ システム・テーブルからリモート・サーバを削除します。

構文

DROP SERVER *server-name*

例

- **例 1 –**

```
DROP SERVER ase_prod
```

使用法

DROP SERVER を正常に実行するには、リモート・サーバに定義されたプロキシ・テーブルをすべて削除しておく必要があります。

関連する動作

- オートコミット。

標準

- SQL—ISO/ANSI SQL 準拠。
- Sybase—Open Client/Open Server でサポートされています。

パーミッション

DBA アカウントだけがリモート・サーバを削除できます。

参照：
- CREATE SERVER 文 (161 ページ)

DROP SERVICE 文

Web サービスを削除します。

構文

```
DROP SERVICE service-name
```

例

- **例 1 –** `tables` という名前の Web サービスを削除するには、次の文を実行します。

  ```
  DROP SERVICE tables
  ```

使用法

DROP SERVICE は Web サービスを削除します。

『SQL Anywhere サーバー - プログラミング』の「HTTP Web サービス」>「HTTP Web サーバとしての SQL Anywhere の使用」も参照してください。

注意： このリファレンスは SQL Anywhere マニュアルにリンクされています。

標準

- SQL—ISO/ANSI SQL 準拠。
- Sybase—Adaptive Server Enterprise ではサポートされていません。

パーミッション

DBA 権限が必要です。

参照：
- ALTER SERVICE 文 (34 ページ)
- CREATE SERVICE 文 (163 ページ)

DROP STATEMENT 文 [ESQL]

文のリソースを解放します。

構文

```
DROP
STATEMENT [ owner.]statement-name
```

パラメータ

- **statement-name：** – 識別子またはホスト変数

例

- **例 1 –**

```
EXEC SQL DROP STATEMENT S1;
EXEC SQL DROP STATEMENT :stmt;
```

使用法

DROP STATEMENT は、指定した準備文によって使われているリソースを解放します。これらのリソースは、**PREPARE** 文を実行して割り付けられます。通常、データベース接続が解放されるまで、リソースは解放されません。

標準

- SQL—ISO/ANSI SQL 文法のベンダ拡張。
- Sybase—Open Client/Open Server でサポートされていません。

パーミッション

文が準備されている必要があります。

参照：

- PREPARE 文 [ESQL] (309 ページ)

DROP TEXT CONFIGURATION 文

テキスト設定オブジェクトを削除します。

構文

以下を参照してください。

使用法

構文と詳細な説明については、『Sybase IQ の非構造化データ分析の概要』を参照してください。

DROP TEXT INDEX 文

データベースから **TEXT**・インデックスを削除します。

構文

以下を参照してください。

使用法

構文と詳細な説明については、『Sybase IQ の非構造化データ分析の概要』を参照してください。

DROP USER 文

ユーザを削除します。

構文

DROP USER *user-name*

例

- **例 1 –** データベースからユーザ `SQLTester` を削除します。

```
DROP USER SQLTester;
```

使用法

user-name—削除するユーザ名。

『SQL Anywhere サーバー - データベース管理』の「データベースの設定」>「ユーザー ID、権限、パーミッションの管理」>「ログインポリシーの管理」を参照してください。

注意： このリファレンスは SQL Anywhere マニュアルにリンクされています。

標準

- SQL—ISO/ANSI SQL 準拠。
- Sybase—Adaptive Server Enterprise ではサポートされていません。

パーミッション

DBA 権限または USER ADMIN 権限を持っている必要があります。

参照：
- CREATE LOGIN POLICY 文 (132 ページ)
- CREATE USER 文 (182 ページ)
- DROP LOGIN POLICY 文 (215 ページ)
- GRANT 文 (247 ページ)
- ALTER LOGIN POLICY 文 (22 ページ)

DROP VARIABLE 文

SQL 変数を削除します。

構文

```
DROP
VARIABLE
identifier
```

使用法

DROP VARIABLE 文は、**CREATE VARIABLE** 文を使用して作成した SQL 変数を削除します。データベース接続を解放すると、変数は自動的に削除されます。大きいオブジェクト用に変数がよく使用されます。したがって、使い終わった変数を削除したり、NULL に設定したりすると、リソース (主にディスク領域) が大幅に解放される場合があります。

IF EXISTS 句は、存在しないデータベース・オブジェクトをこの **DROP** 文が削除しようとしたときにエラーを返さないようにする場合に使用します。

標準

- 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。

パーミッション

なし

参照：

- CREATE VARIABLE 文 (184 ページ)
- SET 文 [ESQL] (350 ページ)

EXECUTE 文 [ESQL]

SQL 文を実行します。

構文

構文 1

```
EXECUTE statement-name
... [ { USING DESCRIPTOR sqlda-name | USING host-variable-list } ]
... [ { INTO DESCRIPTOR into-sqlda-name | INTO into-host-variable-
list ]
... [ ARRAY :nnn } ]
```

構文 2

```
EXECUTE IMMEDIATE statement
```

パラメータ

- **statement-name：** – 識別子またはホスト変数
- **sqlda-name：** – 識別子
- **into-sqlda-name：** – 識別子
- **statement：** – 文字列またはホスト変数

例

- **例 1 – DELETE** を実行します。

```
EXEC SQL EXECUTE IMMEDIATE
'DELETE FROM Employees WHERE EmployeeID = 105';
```

- **例 2 –** 準備された **DELETE** 文を実行します。

```
EXEC SQL PREPARE del_stmt FROM
'DELETE FROM Employees WHERE EmployeeID = :a';
EXEC SQL EXECUTE del_stmt USING :employee_number;
```

- **例 3 –** 準備されたクエリを実行します。

```
EXEC SQL PREPARE sel1 FROM
'SELECT Surname FROM Employees WHERE EmployeeID = :a';
EXEC SQL EXECUTE sel1 USING :employee_number INTO :emp_lname;
```

使用法

構文 1 では、事前に準備された指定の文を動的に実行します。 要求に対して情報を提供するホスト変数プレースホルダ (バインド変数) が動的な文にある場合は、次のどちらかを実行してください。1 つは、*sqlda-name* を使用して C 変数を指定します。この変数は SQLDA へのポインタで、SQLDA には文の中に出現するすべてのバインド変数に必要な記述子が格納されています。もう 1 つは、バインド変数を *host-variable-list* に指定します。

オプションの **ARRAY** 句を提供された **INSERT** 文と一緒に使用して、ワイド挿入ができます。ワイド挿入は複数のローを同時に挿入するため、パフォーマンスが改善されることがあります。 値 **nnn** は、挿入するローの数です。 SQLDA には、nnn * (ロー当たりのカラム数) の数だけ変数を指定してください。 最初のローは SQLDA の変数 0 ～ 変数 (ロー当たりのカラム数) -1 に入り、以後のローも同様です。

SELECT 文または **CALL** 文からの **OUTPUT** は、変数リストの変数の中、または指定した SQLDA が記述するプログラム・データ領域の中のいずれかに入ります。 **OUTPUT** (select リストまたはパラメータ) と、ホスト変数リストまたは SQLDA 記述子配列のいずれかとの対応は、1 対 1 です。

EXECUTE と **INSERT** 文と一緒に使用する場合、挿入されたローは 2 番目の記述子に返されます。 たとえば、プライマリ・キーの値を生成するオートインクリメント・プライマリ・キーを使用するとき、**EXECUTE** 文によって、即座にローを再フェッチしてそのローに割り当てられているプライマリ・キーの値を決定するためのメカニズムが提供されます。

構文 2 は、バインド変数または出力がない文に対して **PREPARE** と **EXECUTE** を行うための省略形です。 文字列またはホスト変数の中にある SQL 文を即座に実行し、完了時に削除します。

EXECUTE 文は、準備可能な任意の SQL 文に使用できます。 データベースから多数のローを返す **SELECT** 文または **CALL** 文にはカーソルを使用します。

注意： EXECUTE 文では、テーブル UDF は参照できません。

INSERT 文、**UPDATE** 文、または **DELETE** 文が正常に実行されると、SQLCA (SQLCOUNT) の *sqlerrd[2]* フィールドに、操作の影響を受けるロー数が入ります。

標準

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase—Open Client/Open Server でサポートされています。

パーミッション

実行される文に対するパーミッションを確認します。

参照：

- DECLARE CURSOR 文 [ESQL] [SP] (191 ページ)
- PREPARE 文 [ESQL] (309 ページ)

EXECUTE 文 [T-SQL]

Adaptive Server Enterprise に互換性のある、**CALL** 文の代替としてプロシージャを呼び出します。

構文

```
EXECUTE [ @return_status = ] [owner.]procedure_name
... { [ @parameter-name = ] expression
| [ @parameter-name = ] @variable [ output ] } ,...
```

例

- **例 1 –** プロシージャ `p1` を作成します。

```
CREATE PROCEDURE p1( @var INTEGER = 54 )
AS
PRINT 'on input @var = %1! ', @var
DECLARE @intvar integer
SELECT @intvar=123
SELECT @var=@intvar
PRINT 'on exit @var = %1!', @var;
```

  - 次の文では、パラメータに 23 を入力してプロシージャを実行します。 Open Client アプリケーションと接続している場合は、クライアント・ウィンドウに **PRINT** メッセージが表示されます。 ODBC または Embedded SQL アプリケーションと接続している場合は、データベース・サーバ・ウィンドウに PRINT メッセージが表示されます。

```
EXECUTE p1 23
```

  - 次の文を使用してプロシージャを実行することもできます。複数のパラメータがあるときに便利です。

```
EXECUTE p1 @var = 23
```

- 次の文では、パラメータにデフォルト値を使用してプロシージャを実行します。

```
EXECUTE p1
```

- 次の文を使用してプロシージャを実行し、リターン・ステータスを検査するために変数の戻り値を格納します。

```
EXECUTE @status = p1 23
```

使用法

EXECUTE を使用してストアド・プロシージャを実行します。オプションとしてプロシージャ・パラメータを指定し、出力値を検索し、ステータス情報を返します。

EXECUTE は、Transact-SQL との互換性のために用意されていますが、Transact-SQL と Sybase IQ のいずれのバッチとプロシージャにも使用できます。

注意： EXECUTE 文では、テーブル UDF は参照できません。

パーミッション

プロシージャの所有者であり、プロシージャに対する EXECUTE パーミッションか DBA 権限を持っていることが必要です。

参照：

- CALL 文 (66 ページ)

EXECUTE IMMEDIATE 文 [ESQL] [SP]

動的に構成された文をプロシージャ内から実行できるようにします。

構文

構文 1

```
EXECUTE IMMEDIATE [ execute-option ] string-expression
```

```
execute-option:
WITH QUOTES [ ON | OFF ]
| WITH ESCAPES { ON | OFF }
| WITH RESULT SET { ON | OFF }
```

構文 2

```
EXECUTE ( string-expression )
```

例

- **例 1 –** 次のプロシージャでは、パラメータにテーブル名を指定してテーブルを作成します。**EXECUTE IMMEDIATE** 文は 1 行で指定してください。

```
CREATE PROCEDURE CreateTableProc(
            IN tablename char(30)
            )
BEGIN
 EXECUTE IMMEDIATE 'CREATE TABLE ' || tablename ||
' ( column1 INT PRIMARY KEY)'
END;
```

プロシージャを呼び出し、mytable というテーブルを作成します。

```
CALL CreateTableProc( 'mytable' )
```

使用法

EXECUTE IMMEDIATE は、プロシージャから実行可能な文の範囲を拡張します。これにより、プロシージャに渡されるパラメータを使用して構成された文など、動的に準備される文の実行が可能になります。

文内のリテラル文字列は、一重引用符で囲む必要があります。また、**PREPARE** 文や **EXECUTE IMMEDIATE** 文にある既存のどの文の名前とも同じでないことが必要です。文は 1 行に記述してください。

EXECUTE IMMEDIATE で実行される文から参照できるのはグローバル変数だけです。

Transact-SQL のストアド・プロシージャの内部では、構文 2 だけが使用できます。

WITH QUOTES—**WITH QUOTES** または **WITH QUOTES ON** を指定すると、文字列式にあるすべての二重引用符が、識別子のデリミタとみなされます。**WITH QUOTES** を指定しない場合、または **WITH QUOTES OFF** を指定する場合、文字列式内の二重引用符は、QUOTED_IDENTIFIER オプションの現在の設定に応じて処理されます。

WITH QUOTES は、実行される文がストアド・プロシージャに渡されるオブジェクト名を使って構成されるが、オブジェクト名に二重引用符が必要であり、プロシージャが呼び出されるとき QUOTED_IDENTIFIER が OFF に設定されている、といった場合に便利です。

詳細については、「QUOTED_IDENTIFIER オプション [TSQL]」を参照してください。

WITH ESCAPES—**WITH ESCAPES OFF** 句を使用すると、文字列式にあるすべてのエスケープ・シーケンス (**¥n**、**¥x**、**¥¥** など) が無視されます。たとえば、2 つの連続する円記号は、円記号 1 つに変換されるのではなく、2 つの円記号のまま残ります。デフォルト設定は、WITH ESCAPES ON です。

WITH ESCAPES OFF を使用すると、円記号を含んだファイル名を参照したり、動的に構成された文の実行を簡単にしたりできます。

場合によっては、*string-expression* 内のエスケープ・シーケンスが、**EXECUTE IMMEDIATE** を実行する前に変換されることもあります。たとえば、複合文は実行される前に解析されますが、**WITH ESCAPES** の設定とは関係なく、この解析中にエスケープ・シーケンスの変換が行われます。このような場合には、**WITH ESCAPES OFF** を指定することで、それ以上変換が行われるのを防げます。次に例を示します。

```
BEGIN
DECLARE String1 LONG VARCHAR;
DECLARE String2 LONG VARCHAR;
EXECUTE IMMEDIATE
 'SET String1 = ''One backslash: ¥¥¥¥ ''';
 EXECUTE IMMEDIATE WITH ESCAPES OFF
 'SET String2 = ''Two backslashes: ¥¥¥¥ ''';
 SELECT String1, String2
END
```

WITH RESULT SET—**WITH RESULT SET ON** を指定すると、**EXECUTE IMMEDIATE** 文に結果セットを返させることができます。この句を使うと、外部のプロシージャが結果セットを返すものとしてマーク付けされます。この句を指定しない場合、プロシージャを呼び出したとき文が結果セットを生成しなければエラーが返されます。

注意： デフォルト・オプションは **WITH RESULT SET OFF** で、文の実行時に結果セットが生成されないことを意味します。

関連する動作

なし。ただし、関連する動作としてのオートコミットを伴うデータ定義文である場合は、そのコミットが起こります。

標準

- SQL—ISO/ANSI SQL 準拠。
- 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。文は、プロシージャの所有者のパーミッションを使用して実行されます。プロシージャを呼び出すユーザのパーミッションは使いません。

参照：

- BEGIN … END 文 (60 ページ)
- CREATE PROCEDURE 文 (137 ページ)

- QUOTED_IDENTIFIER オプション [TSQL] (518 ページ)

EXIT 文 [Interactive SQL]

Interactive SQL を終了する。

構文

```
{ EXIT | QUIT | BYE } [ return-code ]
```

return-code: *number* | *connection-variable*

例

- – テーブル T にローが存在する場合は Interactive SQL の戻り値を 1 に設定し、テーブル T にローがない場合は 0 に設定します。

```
CREATE VARIABLE rowCount INT;
CREATE VARIABLE retcode INT;
SELECT COUNT(*) INTO rowCount FROM T;
IF( rowCount > 0 ) THEN
    SET retcode = 1;
ELSE
    SET retcode = 0;
END IF;
EXIT retcode;
```

注意： 次の文は指定できません。これは、**EXIT** が (SQL 文ではなく) Interactive SQL 文であり、Interactive SQL 文を他の SQL ブロック文に含めることはできないためです。

```
CREATE VARIABLE rowCount INT;
SELECT COUNT(*) INTO rowCount FROM T;
IF( rowCount > 0 ) THEN
    EXIT 1;    //  <-- not allowed
ELSE
    EXIT 0;    //  <-- not allowed
END IF;
```

使用法

Interactive SQL を Windows プログラムとして実行する場合、またはコマンド・プロンプト (バッチ)・モードで実行しているときに Interactive SQL を終了する場合、Interactive SQL ウィンドウを閉じます。どちらの場合でも、データベース接続も閉じられます。COMMIT_ON_EXIT オプションが ON に設定されている場合は、データベース接続を閉じる前に Interactive SQL が自動的に **COMMIT** 文を実行します。このオプションが OFF に設定されている場合は、Interactive SQL は暗黙の **ROLLBACK**

を実行します。デフォルトでは、COMMIT_ON_EXIT オプションは ON に設定されます。

オプションのリターン・コードをバッチ・ファイルで使用すると、Interactive SQL コマンド・ファイルにコマンドの成功または失敗が示されます。デフォルトのリターン・コードは 0 です。

関連する動作

- オプション COMMIT_ON_EXIT が ON (デフォルト) に設定されている場合は自動的にコミットを実行し、そうでない場合、この文は暗黙のロールバックを実行します。
- Windows オペレーティング・システムでは、オプションの戻り値を ERRORLEVEL として使用できます。

標準

- SQL—ISO/ANSI SQL 文法のベンダ拡張。
- Sybase—Adaptive Server Enterprise では使用できません。

パーミッション

なし

参照：

- SET OPTION 文 (356 ページ)

FETCH 文 [ESQL] [SP]

カーソルを再配置し、そこからデータを取り出します。

構文

```
FETCH
{ NEXT | PRIOR | FIRST | LAST
| ABSOLUTE row-count | RELATIVE row-count }
... cursor-name
... { [ INTO host-variable-list ]
| USING
              DESCRIPTOR sqlda-name
| INTO variable-list }
... [ PURGE ] [ BLOCK n ] [ ARRAY fetch-count ]
... INTO variable-list
... IQ CACHE row-count
```

パラメータ

- **cursor-name：** – 識別子またはホスト変数
- **sqlda-name：** – 識別子
- **host-variable-list：** – インジケータ変数を含む場合がある
- **row_count：** – 数値またはホスト変数
- **fetch-count：** – 整数またはホスト変数

例

- **例 1** – 次は、Embedded SQL の例です。

```
EXEC SQL DECLARE cur_employee CURSOR FOR
SELECT EmployeeID, Surname FROM Employees;
EXEC SQL OPEN cur_employee;
EXEC SQL FETCH cur_employee
INTO :emp_number, :emp_name:indicator;
```

- **例 2** – 次は、プロシージャの例です。

```
BEGIN
    DECLARE cur_employee CURSOR FOR
        SELECT Surname
        FROM Employees;
    DECLARE name CHAR(40) ;
    OPEN cur_employee;
    LOOP
        FETCH NEXT cur_employee into name ;
         .
         .
         .
    END LOOP
    CLOSE cur_employee;
END
```

使用法

FETCH は、指定したカーソルからローを 1 つ取り出します。

ARRAY 句を使用すると、ワイド・フェッチができます。複数のローを同時に取り出せるため、パフォーマンスが向上します。

カーソルを事前にオープンしておく必要があります。

SELECT 文の結果から 1 つのローを、変数リストの変数に格納します。select リスト対ホスト変数リストは、1 対 1 に対応します。

SELECT の結果から 1 つまたは複数のローを、変数リストの変数、または指定した SQLDA が記述するプログラム・データ領域のどちらかに格納します。select リスト対ホスト変数リスト、または select リスト対 SQLDA 記述子配列は、どちらも 1 対 1 に対応します。

INTO 句はオプションです。これを指定しない場合、**FETCH** はカーソルだけを配置します (次の段落を参照してください)。

オプションの配置パラメータを指定すると、カーソルを移動してからローをフェッチできます。デフォルトは **NEXT** です。これを指定すると、カーソルを 1 つ次のローに進めてから、ローをフェッチします。**PRIOR** 句を指定すると、カーソルを 1 つ前のローに戻してから、ローをフェッチします。

RELATIVE 配置を使用すると、フェッチする前にいずれかの方向へ指定した数のローだけカーソルを移動できます。正の数は前進を示し、負の数は後退を示します。したがって、**NEXT** は **RELATIVE 1** と等しく、**PRIOR** は **RELATIVE -1** と等しくなります。**RELATIVE 0** は、このカーソル上の最後の FETCH 文と同じローを取り出します。

ABSOLUTE 配置パラメータを使用すると、特定のローに移動できます。ゼロは最初のローの前の位置を示します。詳細については、『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。

1 は先頭のロー、2 は 2 番目のロー、のようになります。負の数を使用して、カーソルの最後からの絶対位置を指定します。-1 はカーソルの最後のローを示します。**FIRST** は **ABSOLUTE 1** の省略形です。**LAST** は **ABSOLUTE -1** の省略形です。

注意： Sybase IQ では、**FIRST**、**LAST**、**ABSOLUTE**、負数の **RELATIVE** のオプションを他の DBMS 製品ほど効率よく処理できないため、これらのオプションを使用するとパフォーマンスが低下します。

OPEN は、まず、先頭のローの前にカーソルを配置します。

FOR READ ONLY として宣言されたカーソルからは、最初に **FETCH** を実行したときのテーブルのバージョンではなく、カーソルをオープンしたときにカーソルの宣言に使用されるテーブルのバージョンが表示されます。

FETCH 文に指定された配置パラメータがカーソル位置の許容範囲外である場合は、SQLE_NOTFOUND が警告されます。

IQ CACHE 句には、FIFO キューにバッファされるローの最大数を指定します。**IQ CACHE** に値を指定しなければ、`CURSOR_WINDOW_ROWS` データベース・オプションの値が使用されます。`CURSOR_WINDOW_ROWS` のデフォルト設定は 200 です。

Embedded SQL での FETCH 文と OPEN 文の使用

次の句は、Embedded SQL の中だけで使用します。

- **USING DESCRIPTOR***sqlda-name*
- **INTO***host-variable-list*
- **PURGE**

- **BLOCK***n*
- **ARRAY** *fetch-count*
- *cursor-name* オプションと *row-count* オプションでの *host-variable* の使用

C 言語のソース・コード内では、**DECLARE CURSOR** が **FETCH** 文の前になるようにしてください。また、**FETCH** の前に **OPEN** 文を実行します。カーソル名の代わりにホスト変数を使用している場合は、**DECLARE** 文が実際にコードを生成するため、**FETCH** の前に DECLARE 文を実行する必要があります。

マルチユーザ環境では、ローが複数のクライアントから一度にフェッチされる可能性があります。これはブロック・フェッチまたはマルチロー・フェッチと呼ばれます。最初のフェッチによって、複数のローがサーバから送り返されます。クライアントは、これらのローをバッファに格納します。後に続くフェッチは、サーバへ新しい要求を行わないで、これらのバッファからローを取り出します。

BLOCK 句によって、アプリケーションでフェッチ可能なローの個数に関する情報をクライアントとサーバに与えます。0 という特別な値は、要求がサーバに送信されてシングル・ローが返却されることを示します (ロー・ブロックはありません)。

PURGE 句によって、クライアントはすべてのローのバッファをフラッシュし、次にフェッチ要求をサーバに送信します。このフェッチ要求はローのブロックを返すことに注意してください。

フェッチ時に SQLSTATE_NOTFOUND 警告が返される場合は、SQLCA (SQLCOUNT) の *sqlerrd[2]* フィールドに、カーソル位置の許容範囲を超えてフェッチを試みたときのローの数が入ります。カーソルを、最初のローの前または最後のローの後にあるローに位置付けることができます。ローが見つからなくても位置が有効な場合は、値は 0 です。たとえば、カーソル位置が最後のローのときに **FETCH RELATIVE 1** を実行した場合です。実行しようとするフェッチがカーソルの最後よりも後の場合、値は正になります。実行しようとするフェッチがカーソルの先頭よりも前の場合、値は負になります。

FETCH 文が正常に実行されると、SQLCA (SQLIOCOUNT) の *sqlerrd[1]* フィールドは、フェッチの実行に必要な入出力操作の回数だけ増加します。このフィールドは、実際にはデータベース文が発行されるたびに増加します。

Embedded SQL でワイド・フェッチを使用するには、コードに次のような **FETCH** 文を含めます。

```
EXEC SQL FETCH . . . ARRAY nnn
```

ただし、*ARRAY nnn* は **FETCH** 文の最後の項目です。フェッチ回数を示す *nnn* にはホスト変数も使用できます。SQLDA には、nnn * (ロー当たりのカラム数) の数だけ変数を指定してください。最初のローは SQLDA の変数 0 ～ 変数 (ロー当たりのカラム数) -1 に入り、以後のローも同様です。

サーバは、フェッチしたレコードの数を SQLCOUNT に返します。エラーがない場合は、0 より大きい SQLCOUNT を常に返します。古いバージョンのサーバはローを 1 つだけ返し、SQLCOUNT は 0 に設定されます。エラーではなくて SQLCOUNT が 0 の場合、有効なローが 1 つフェッチされたことを示します。

標準

- SQL—ISO/ANSI SQL 準拠。
- Sybase—Adaptive Server Enterprise でサポートされています。

パーミッション

カーソルをオープンしている必要があります。また、ユーザには、カーソル宣言内で参照するテーブルの SELECT パーミッションが必要です。

参照：

- DECLARE CURSOR 文 [ESQL] [SP] (191 ページ)
- OPEN 文 [ESQL] [SP] (301 ページ)
- PREPARE 文 [ESQL] (309 ページ)
- CURSOR_WINDOW_ROWS オプション (428 ページ)

FOR 文

カーソル内の各ローに対して一度ずつ、文リストの実行を繰り返します。

構文

```
[ statement-label: ]
FOR for-loop-name
              AS cursor-name [ cursor-type ] CURSOR
   { FOR statement
... [ { FOR { UPDATE cursor-concurrency | FOR READ ONLY } ]
    | USING variable-name }
  DO statement-list
              END FOR [ statement-label ]
```

パラメータ

- **cursor-type：** **– NO SCROLL** | **DYNAMIC SCROLL** | **SCROLL** | **INSENSITIVE** | **SENSITIVE**
- **cursor-concurrency：** **– BY** { **VALUES** | **TIMESTAMP** | **LOCK**]
- **variable-name：** *– 識別子*

例

- **例 1 –** 次のコードは、**FOR** ループの使用法を示します。

```
FOR names AS curs CURSOR FOR
SELECT Surname
FROM Employees
DO
    CALL search_for_name( Surname );
END FOR;
```

使用法

FOR 文は、カーソル内の各ローに対して一度ずつ SQL 文のリストを実行できる制御文です。

FOR 文は、カーソルの **DECLARE** と、カーソルの結果セットのそれぞれのカラムに対する変数の **DECLARE** があり、その後には、カーソルから 1 つのローをフェッチしてローカル変数に格納し、カーソル内の各ローに対して一度だけ *statement-list* を実行するループが続く複合文と同じです。

cursor-type パラメータとその他の例については、『SQL Anywhere サーバー - SQL リファレンス』の「SQL 文」>「SQL 文」>「FOR 文」を参照してください。

注意： このリファレンスは SQL Anywhere マニュアルにリンクされています。

ローカル変数の名前とデータ型は、カーソル内で使用する *statement* に基づいて宣言されます。**SELECT** 文の場合、データ型は select リスト中の式のデータ型です。名前は、それらが存在する select リスト項目のエイリアスです。そうでない場合、カラムの名前になります。select リスト項目が単一のカラム参照を持たない場合は、エイリアスを持たせる必要があります。**CALL** 文の場合、名前とデータ型はプロシージャ定義内の **RESULT** 句から取得されます。

LEAVE 文を使用して、**END FOR** に続く最初の文から実行を再開できます。終了の *statement-label* を指定する場合は、開始の *statement-label* と一致させる必要があります。

標準

- SQL—ISO/ANSI SQL 準拠。
- Sybase—Adaptive Server Enterprise ではサポートされていません。

パーミッション

なし

参照：

- DECLARE CURSOR 文 [ESQL] [SP] (191 ページ)
- FETCH 文 [ESQL] [SP] (230 ページ)
- LEAVE 文 (271 ページ)
- LOOP 文 (297 ページ)

FORWARD TO 文

ネイティブな構文をリモート・サーバに送信します。

構文

構文 1

```
FORWARD TO server-name { sql-statement }
```

構文 2

```
FORWARD TO [ server-name ]
```

例

- **例 1 –** 次に、リモート・サーバ ase_prod とのパススルー・セッションの例を示します。

```
FORWARD TO aseprod
SELECT * from titles
SELECT * from authors
FORWARD TO
```

使用法

FORWARD TO を使用すると、パススルー接続に必要なサーバを指定できます。 この文は、次の 2 つの目的で使用できます。

- 文をリモート・サーバに送信する (構文 1)
- 一連の文をリモート・サーバに送信するために、Sybase IQ をパススルー・モードに設定する (構文 2)

ユーザの代わりに *server-name* への接続を確立する場合、サーバは次のものを使用します。

- **CREATE EXTERNLOGIN** を使用するリモート・ログイン・エイリアス・セット
- リモート・ログイン・エイリアスを設定しない場合は、Sybase IQ との通信に使用する名前とパスワード

指定のサーバに接続できない場合は、その理由を示すメッセージがユーザに返されます。

文が要求されたサーバに渡されると、結果はすべて、クライアント・プログラムで認識できるフォームに変換されます。

server-name は、リモート・サーバの名前を示します。

sql-statement は、リモート・サーバのネイティブ SQL 構文によるコマンドを示します。コマンドまたはコマンド・グループをカッコ ({}) または一重引用符で囲みます。

server_name を指定して、**FORWARD TO** クエリに文を指定しないと、セッションはパススルー・モードに入り、後続のすべてのクエリはリモート・サーバに直接渡されます。パススルー・モードをオフにするには、*server_name* を指定せずに **FORWARD TO** を発行してください。

注意： **FORWARD TO** 文はサーバ・ディレクティブであり、ストアド・プロシージャ、トリガ、イベント、またはバッチ内では使用できません。

関連する動作

- リモート接続は、**FORWARD TO** セッションの間、AUTOCOMMIT (非連鎖) モードに設定されます。**FORWARD TO** 文の前に保留中であった作業はすべて自動的にコミットされます。

標準

- SQL—ISO/ANSI SQL 準拠。
- 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

参照：

- CREATE EXTERNLOGIN 文 (108 ページ)
- CREATE SERVER 文 (161 ページ)

FROM 句

SELECT 文に必要なデータベース・テーブルまたはビューを指定します。

構文

```
... FROM table-expression [, …]
```

```
table-expression :
    table-name
    | view-name
    | procedure-name
    | common-table-expression
    | (subquery) [[  AS]  derived-table-name [column_name, ...)]]
    | derived-table
    | join-expression
    | ( table-expression, ... )
    | openstring-expression
    | apply-expression
    | contains-expression
    | dml-derived-table
```

```
table-name :
    [ userid.]table-name ]
    [ [ AS ] correlation-name ]
    [ FORCE INDEX ( index-name ) ]
```

```
view-name :
    [ userid.]view-name [ [ AS ] correlation-name ]
```

```
procedure-name :
    [  owner, ] procedure-name ([  parameter, ...])
    [  WITH( column-name datatype,)]
    [ [   AS] correlation-name ]

parameter :
scalar-expression | table-parameter

table-parameter :
TABLE(select-statement) [ OVER (table-parameter-over)]

table-parameter-over :
[ PARTITION BY {ANY| NONE| table-expression } ]
[ ORDER BY { expression | integer } [ ASC | DESC ] [, …] ]
```

```
derived-table :
    ( select-statement )
    [ AS ] correlation-name [ ( column-name, ... ) ]
```

```
join-expression :
    table-expression join-operator table-expression
    [ ON join-condition ]
```

```
join-operator :
    [ KEY | NATURAL ] [ join-type ] JOIN
    | CROSS JOIN
```

```
join-type :
    INNER
    | LEFT [ OUTER ]
    | RIGHT [ OUTER ]
    | FULL [ OUTER ]
```

```
openstring-expression :
    OPENSTRING ( { FILE | VALUE } string-expression )
```

```
    WITH ( rowset-schema )
    [ OPTION ( scan-option ...  ) ]
    [ AS ] correlation-name
```

```
apply-expression :
    table-expression { CROSS | OUTER } APPLY table-expression
```

```
contains-expression :
    { table-name  | view-name } CONTAINS ( column-name [,...], contains-
query ) [ [ AS ] score-correlation-name ]
```

```
rowset-schema :
    column-schema-list
    | TABLE [owner.]table-name [ ( column-list ) ]
```

```
column-schema-list :
    { column-name user-or-base-type |  filler( ) } [ , ... ]
```

```
column-list :
    { column-name | filler( ) } [ , ... ]
```

```
scan-option :
    BYTE ORDER MARK { ON | OFF }
    | COMMENTS INTRODUCED BY comment-prefix
    | DELIMITED BY string
    | ENCODING encoding
    | ESCAPE CHARACTER character
    | ESCAPES { ON | OFF }
    | FORMAT { TEXT  | BCP  }
    | HEXADECIMAL { ON | OFF }
    | QUOTE string
    | QUOTES { ON | OFF }
    | ROW DELIMITED BY string
    | SKIP integer
    | STRIP { ON | OFF | LTRIM | RTRIM | BOTH }
```

```
contains-query :
    string
```

```
dml-derived-table :
    ( dml-statement  ) REFERENCING ( [ table-version-names  | NONE ] )
```

```
dml-statement :
    insert-statement
    delete-statement
    update-statement
    merge-statement
```

```
table-version-names :
    OLD [ AS ] correlation-name [ FINAL [ AS ] correlation-name ]
    | FINAL [ AS ] correlation-name
```

例

- **例 1 –** 次は、有効な **FROM** 句です。

  ```
  ...
  FROM Employees
  ```

```
...
...
FROM Employees NATURAL JOIN Departments
...
...
FROM Customers
KEY JOIN SalesOrders
KEY JOIN SalesOrderItems
KEY JOIN Products
...
```

- **例 2 –** 次のクエリは、クエリ内での抽出テーブルの使用法を示します。

```
SELECT Surname, GivenName, number_of_orders
FROM Customers JOIN
     ( SELECT CustomerID, count(*)
       FROM SalesOrders
        GROUP BY CustomerID )
     AS sales_order_counts ( CustomerID,
                             number_of_orders )
ON ( Customers.ID = sales_order_counts.cust_id )
WHERE number_of_orders > 3
```

使用法

SELECT 文には、文で使用するテーブルを指定するためのテーブル・リストが必要です。

注意： ここでの説明はテーブルに関するものですが、特にことわりがないかぎりビューにも当てはまります。

FROM テーブル・リストは、指定した全テーブルからのすべてのカラムで構成する結果セットを作成します。 最初に、コンポーネント・テーブルのすべてのローの組み合わせが結果セットの中に入ります。次に、組み合わせの数は、通常、ジョイン条件か **WHERE** 条件、またはその両方によって減ります。

- **SCALAR –** *scalar-parameter* は、有効な SQL データ型を持つ任意のオブジェクトです。
- **TABLE – TABLE** パラメータは、テーブル、ビュー、または共通 *table-expression* 名を使用して指定できます。これらのオブジェクトは、**TABLE** パラメータの外でも使用されている場合、オブジェクトの新しいインスタンスとして扱われます。

  次のクエリは、有効な **FROM** 句を示します。同一のテーブル T への 2 つの参照が、同一テーブル T の異なる 2 つのインスタンスとして扱われています。

  ```
  SELECT * FROM T, my_proc(TABLE(SELECT T.Z, T.X FROM T)
  OVER(PARTITION BY T.Z));
  ```

  パラメータ化されたテーブル関数 (TPF) の例—次のクエリは、有効な **FROM** 句を示します。

```
SELECT * FROM R, SELECT * FROM my_udf(1);
SELECT * FROM my_tpf(1, TABLE(SELECT c1, c2 FROM t))
   (my_proc(R.X, TABLE T OVER PARTITION BY T.X)) AS XX;
```

サブクエリを使用して **TABLE** パラメータを定義する場合は、次の制約があります。

- **TABLE** パラメータが **IN** タイプである。
- **PARTITION BY** または **ORDER BY** 句が、抽出テーブルおよび外部参照のカラムを参照している。 *expression-list* 内の式は、**TABLE** 入力パラメータの K 番目のカラムを参照する整数 *K* にすることができます。

注意： テーブル UDF は、SQL 文の **FROM** 句の中でのみ参照できます。

- **PARTITION BY – PARTITION BY** 句は、実行エンジンが関数の呼び出しを実行する方法を論理的に指定します。 実行エンジン側では、その関数をパーティションごとに呼び出し、関数側では、各呼び出し内でパーティション全体を処理する必要があります。

  また、**PARTITION BY** は、関数の呼び出しごとに 1 パーティション分のデータを正確に処理できるよう、入力データのパーティション分割方法も指定します。 関数は、パーティションの数と同じ回数だけ呼び出される必要があります。 TPF の場合、実行時に、サーバと UDF 間の動的なネゴシエーションを通じて、並行処理の特性が確立されます。 TPF を並行で実行できる場合、入力パーティションが *N* 個あるとすると、関数は *M* 回インスタンス化できます。ここで *M <=N* です。 関数の各インスタンスは、2 回以上呼び出すことができ、呼び出しごとにパーティションが 1 つ消費されます。

  注意： 実行エンジンは、任意のパーティション順で関数を呼び出すことができ、関数は、パーティション順に関係なく、同じ結果セットを返すと仮定されます。 関数の 2 つの呼び出し間で、パーティションを分割することはできません。

  PARTITION BY*expression-list* または **PARTITION BY ANY** 句ごとに指定できる **TABLE** 入力パラメータは、1 つのみです。 これら以外の場所で **TABLE** 入力パラメータを使用する場合は、**PARTITION BY NONE** 句を明示的または暗黙的に指定する必要があります。
- **ORDER BY – ORDER BY** 句は、各パーティション内の入力データが *expression-list* を基準にソートされていると実行エンジンによって想定されることを指定します。 UDF は、各パーティションがこの物理プロパティを持っていると想定します。 パーティションが 1 つのみの場合、入力データ全体が、**ORDER BY** 指定に基づいて順序付けられます。**ORDER BY** 句は、**PARTITION BY NONE** 句が指定されているか、または **PARTITION BY** 句が指定されていない任意の **TABLE** 入力パラメータに指定できます。

全文検索を可能にする、**FROM** 句内で使用される *contains-expression* 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。

- **ジョイン** – *join-type* キーワードには次のようなものがあります。

表 8 : FROM 句の join-type のキーワード

join-type のキーワード	説明
CROSS JOIN	2 つのソース・テーブルの直積 (クロス積) を返す
NATURAL JOIN	2 つのテーブルの間で、同じ名前を持つすべての対応するカラムの等価性を比較する (等価ジョインの特殊なケースであり、カラムの長さとデータ型は同じ)
KEY JOIN	最初のテーブルの外部キーの値と、2 つ目のテーブルのプライマリ・キーの値が同じであるものに制限する
INNER JOIN	結果テーブルから、両方のテーブルに一致するローを持つロー以外をすべて破棄する
LEFT OUTER JOIN	左側のテーブルからの一致しないローは残すが、右側のテーブルからの一致しないローは破棄する
RIGHT OUTER JOIN	右側のテーブルからの一致しないローは残すが、左側のテーブルからの一致しないローは破棄する
FULL OUTER JOIN	左と右の両方のテーブルからの一致しないローを維持する

カンマ形式のジョインとキーワード形式のジョインを、**FROM** 句に混在させないでください。同じクエリを、それぞれいずれかの形式を使用した 2 とおりの方法で記述できます。 ANSI 構文のキーワード形式の方が望ましいとされています。

次のクエリには、カンマ形式のジョインが使用されています。

```
SELECT *
  FROM Products pr, SalesOrders so, SalesOrderItems si
  WHERE pr.ProductID = so.ProductID
    AND pr.ProductID = si.ProductID;
```

同じクエリを、望ましいとされるキーワード形式のジョインで記述すると次のようになります。

```
SELECT *
  FROM Products pr INNER JOIN SalesOrders so
    ON (pr.ProductID = so.ProductID)
  INNER JOIN SalesOrderItems si
    ON (pr.ProductID = si.ProductID);
```

ON 句は内部ジョイン、左ジョイン、右ジョイン、フル・ジョインのデータをフィルタします。 クロス・ジョインには **ON** 句を付けません。内部ジョインで

は、**ON** 句は **WHERE** 句と同じですが、 外部ジョインでは **ON** 句と **WHERE** 句は違います。 外部ジョインの **ON** 句は、クロス積のローをフィルタし、その結果に NULL で拡張された一致しないローを含めます。 その後、**WHERE** 句によって、外部ジョインで生成された一致するローと一致しないローからローを取り除きます。 一致しないローのうち、必要なものまで **WHERE** 句の述部で取り除いてしまわないように注意が必要です。

外部ジョインの **ON** 句の中に、サブクエリを使用することはできません。

Transact-SQL 互換のジョインの記述方法については、『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「他の Sybase データベースとの互換性」を参照してください。

userid を指定すると、別のユーザが所有するテーブルを修飾できます。 ユーザ ID を指定しない場合は、デフォルトで、現在のユーザの所属グループが所有するテーブルを参照します。

この SQL 文の場合にのみ、相関名を使用してテーブルに一時的な名前が付けられます。 カラムはテーブル名で修飾されている必要がありますが、カラムを参照するときにテーブル名が長くて入力が煩わしい場合、この相関名を使うと便利です。 相関名は、同じクエリで複数回、同じテーブルを参照するときに、テーブル・インスタンス間で区別するためにも必要です。 相関名を指定しない場合は、テーブル名を現在の文の相関名として使用します。

テーブル式の中で同じテーブルに対して同じ相関名を 2 回使用する場合、そのテーブルは、1 回だけリストされたものとして扱われます。 次に例を示します。

```
SELECT *
FROM SalesOrders
KEY JOIN SalesOrderItems,
SalesOrders
KEY JOIN Employees
```

`SalesOrders` テーブルの 2 つのインスタンスは、1 つのインスタンスとして扱われます。これは、次のものと等しくなります。

```
SELECT *
FROM SalesOrderItems
KEY JOIN SalesOrders
KEY JOIN Employees
```

一方、次の文では、`Person` テーブルの 2 つのインスタンスとして扱われ、異なる相関名 HUSBAND と WIFE を使います。

```
SELECT *
FROM Person HUSBAND, Person WIFE
```

FROM 句に、1つまたは複数のテーブルかビューの代わりに **SELECT** 文を指定すると、ビューを作成しなくても、グループ上でグループを、またはグループでジョインを使用できます。 **SELECT** 文のこのような使用法は、抽出テーブルと呼ばれます。

ジョイン・カラムのパフォーマンスを高めるために、データ型は似ているものを使用してください。

- **パフォーマンスに関する考慮事項** – クエリによっては、Sybase IQ で最大 16 から 64 までのテーブルを、オプティマイザをオンにした状態で **FROM** 句に指定できます。ただし、非常に複雑なクエリで、16 ～ 18 よりも多いテーブルを **FROM** 句に指定すると、パフォーマンスが損なわれる可能性があります。

注意： **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、特に構文と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。

FROM 句を必要としないクエリがある場合は、**FROM iq_dummy** 句を追加すると、強制的に Sybase IQ で処理させることができます。この `iq_dummy` は、ユーザが自分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。

標準

- SQL—ISO/ANSI SQL 準拠。
- Sybase—**JOIN** 句は、Adaptive Server Enterprise の一部のバージョンではサポートされていません。代わりに、**WHERE** 句を使用してジョインを作成してください。

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。

参照：

- DELETE 文 (201 ページ)
- SELECT 文 (339 ページ)

GET DESCRIPTOR 文 [ESQL]

記述子領域内の変数に関する情報を取り出します。または、記述子領域内の変数から実際のデータを取り出します。

構文

```
            GET DESCRIPTOR
            descriptor-name
... { hostvar = COUNT } | VALUE
            n
            assignment [,…] }
```

パラメータ

- **assignment：** – *hostvar* = { **TYPE** | **LENGTH** | **PRECISION** | **SCALE** | **DATA** | **INDICATOR** | **NAME** | **NULLABLE** | **RETURNED_LENGTH** }

例

- **例 1 –** 使用例については、「ALLOCATE DESCRIPTOR 文 [ESQL]」を参照してください。

使用法

値 *n* には、情報を取り出す記述子領域内の変数を指定します。

GET DESCRIPTOR ... DATA の実行時に型検査を実行して、ホスト変数と記述子変数が同じデータ型を持つようにします。`LONG VARCHAR` と `LONG BINARY` では、**GET DESCRIPTOR ... DATA** はサポートされていません。

エラーが発生した場合は、SQLCA に返されます。

標準

- SQL—ISO/ANSI SQL 準拠。
- 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

参照：

- ALLOCATE DESCRIPTOR 文 [ESQL] (5 ページ)

- DEALLOCATE DESCRIPTOR 文 [ESQL] (188 ページ)
- SET DESCRIPTOR 文 [ESQL] (355 ページ)

GOTO 文 [T-SQL]

ラベルの付いた文に制御を分岐します。

構文

```
label:
GOTO label
```

例

- **例 1 –** 次の Transact-SQL バッチは、サーバ・ウィンドウに "yes" のメッセージを 4 回表示します。

```
declare @count smallint
select @count = 1
restart:
    print 'yes'
    select @count = @count + 1
    while @count <=4
     goto restart
```

使用法

Transact-SQL のプロシージャまたはバッチに含まれる任意の文に、ラベルを付けることができます。ラベル名の末尾にコロン (:) を付けて有効な識別子とします。**GOTO** 文では、コロンは使用できません。

標準

- SQL—ISO/ANSI SQL 準拠。
- Sybase—Adaptive Server Enterprise は、GOTO 文をサポートします。

パーミッション

なし

GRANT 文

特定のユーザにパーミッションを付与し、新しいユーザ ID を作成します。

構文

構文 1 – 権限の付与

```
GRANT authority, …
  TO userid, …
```

```
authority:
 BACKUP
| DBA
| GROUP
| MEMBERSHIP IN GROUP userid [, …]
| MULTIPLEX ADMIN
| OPERATOR
| PERMS ADMIN
| PROFILE
| READCLIENTFILE
| READFILE
| [ RESOURCE | ALL ]
| SPACE ADMIN
| USER ADMIN
| VALIDATE
| WRITECLIENTFILE
```

構文 2 – グループ・ステータスまたはグループのメンバシップの付与

```
GRANT { GROUP | MEMBERSHIP IN GROUP userid, … }
  TO userid, …
```

構文 3 – データベース・オブジェクト・パーミッションの付与

```
GRANT permission, …
 ON [ owner.]table-name
 TO userid [, …]
 [ WITH GRANT OPTION ]
 [ FROM userid ]
```

```
permission:
  ALL [ PRIVILEGES ]
| ALTER
| DELETE
| INSERT
| REFERENCES [ ( column-name [, …] ) ]
| SELECT [ ( column-name [, …] ) ]
| UPDATE [ ( column-name, … ) ]
```

構文 4 – 実行パーミッションの付与

```
GRANT EXECUTE ON [ owner.]procedure-name
 TO userid [, …]
```

構文 5 – 統合化ログインの付与

```
GRANT INTEGRATED LOGIN TO user_profile_name [, …]
 AS USER userid
```

構文 6 – Kerberos ログインの付与

```
GRANT KERBEROS LOGIN TO client-Kerberos-principal, …
 AS USER userid
```

構文 7 – 接続パーミッションの付与

```
GRANT CONNECT TO userid [, …]
 IDENTIFIED BY password [, …]
```

構文 8 – DB 領域への作成パーミッションの付与

```
GRANT CREATE ON dbspace_name
 TO userid [, …]
```

例

- **例 1 –** データベースに 2 つの新しいユーザを作成します。

  ```
  GRANT
  CONNECT TO Laurel, Hardy
  IDENTIFIED BY Stan, Ollie
  ```

- **例 2 –** Laurel というユーザに `Employees` テーブルに関するパーミッションを付与します。

  ```
  GRANT
  SELECT, INSERT, DELETE
  ON Employees
  TO Laurel
  ```

- **例 3 –** ユーザ Hardy に **Calculate_Report** プロシージャの実行を許可します。

  ```
  GRANT
  EXECUTE ON Calculate_Report
  TO Hardy
  ```

- **例 4 –** DB 領域 *DspHist* の `CREATE` パーミッションをユーザの Lawrence と Swift に付与します。

  ```
  GRANT
  CREATE ON DspHist
  TO LAWRENCE, SWIFT
  ```

- **例 5 –** DB 領域 `DspHist` の `CREATE` 権限をユーザの `Fiona` と `Ciaran` に付与します。

  ```
  GRANT CREATE ON DspHist TO Fiona, Ciaran
  ```

使用法

GRANT 文を使って、データベース・パーミッションを個々のユーザ ID とグループに付与します。ユーザとグループを作成したり削除したりする場合にも使用します。

GRANT authority 句 – 次の権限のいずれかをユーザに付与します。

- **BACKUP** 権限 – データベースをバックアップする権限を付与します。『SQL Anywhere サーバー - SQL リファレンス』の「SQL 文」>「SQL 文」>「GRANT 文」を参照してください。
- **DBA** 権限 – データベース管理者権限を与えられたユーザは、すべてのことができます。通常、この権限はデータベース管理者のために予約されています。
- **MULTIPLEX ADMIN** 権限 – ユーザが、マルチプレックス・サーバの作成や削除などのマルチプレックス管理タスクを実行できるようにします。 詳細については、『Sybase IQ Multiplex の使用』の「マルチプレックス・サーバの管理」>「管理用権限の使用」>「MULTIPLEX ADMIN 権限」を参照してください。
- **OPERATOR** 権限 – ユーザが、データベースのチェックポイントとバックアップ、接続の削除、システムのモニタリングを実行できるようにします。詳細については、『システム管理ガイド： 第 1 巻』の「ユーザ ID とパーミッションの管理」を参照してください。
- **PERMS ADMIN** 権限 – ユーザが、データ・パーミッション、グループ、権限、パスワードを管理できるようにします。詳細については、『システム管理ガイド： 第 1 巻』の「ユーザ ID とパーミッションの管理」を参照してください。
- **PROFILE** 権限 – プロファイリングと診断操作を実行する権限をユーザに付与します。『SQL Anywhere サーバー - SQL リファレンス』の「SQL 文」>「SQL 文」>「GRANT 文」を参照してください。
- **READCLIENTFILE** 権限 – データのロード時などに、クライアント・コンピュータ上のファイルから読み取る権限をユーザに付与します。 『SQL Anywhere サーバー - SQL リファレンス』の「SQL 文」>「SQL 文」>「GRANT 文」を参照してください。
- **READFILE** 権限 – ユーザが、OPENSTRING 句を使用してファイルに対して **SELECT** 文を実行できるようにします。 『SQL Anywhere サーバー - SQL リファレンス』の「SQL 文」>「SQL 文」>「GRANT 文」を参照してください。
- **RESOURCE** 権限 – ユーザが、テーブル、ビュー、ストアド・プロシージャなどのデータベース・オブジェクトを作成できるようにします。 構文 1 では、**ALL** は Adaptive Server Enterprise に互換性のある **RESOURCE** と同じです。
- **SPACE ADMIN** 権限 – ユーザが DB 領域を管理できるようにします。詳細については、『システム管理ガイド： 第 1 巻』の「ユーザ ID とパーミッションの管理」を参照してください。
- **USER ADMIN** 権限 – ユーザが、ユーザ、外部ログイン、ログイン・ポリシーを管理できるようにします。詳細については、『システム管理ガイド： 第 1 巻』の「ユーザ ID とパーミッションの管理」を参照してください。
- **VALIDATE** 権限 – ユーザが、データベースの検証、テーブルとインデックスの検証、チェックサムの検証など、さまざまな **VALIDATE** 文でサポートされている検証操作を実行できるようにします。 また、ユーザは Sybase Central の検証

ユーティリティ (**dbvalid**) とデータベース検証ウィザードを使用できるようになります。『SQL Anywhere サーバー - SQL リファレンス』の「SQL 文」>「SQL 文」>「GRANT 文」を参照してください。

- **WRITECLIENTFILE** 権限 - データのアンロード時などに、クライアント・コンピュータ上のファイルに書き込む権限をユーザに付与します。『SQL Anywhere サーバー - SQL リファレンス』の「SQL 文」>「SQL 文」>「GRANT 文」を参照してください。

注意： これらのリファレンスは SQL Anywhere マニュアルにリンクされています。

GROUP 句 – ユーザがメンバを持てるようにします。詳細については、『システム管理ガイド：第 1 巻』の「ユーザ ID とパーミッションの管理」を参照してください。

MEMBERSHIP IN GROUP 句 – ユーザがグループからテーブル・パーミッションを継承し、テーブル名を修飾しなくても、グループが作成したテーブルを参照できるようにします。

特定のユーザが特定のテーブル、ビュー、またはプロシージャにアクセスできないようにするには、そのオブジェクトのパーミッションを持つグループのメンバからそのユーザを除外します。

GRANT permission 句 – 個々のテーブルまたはビューに対するパーミッションを付与します。テーブル・パーミッションを個々に指定することも、**ALL** を指定して一度に 6 つのパーミッションをすべて付与することもできます。**WITH GRANT OPTION** を指定すると、指定したユーザ ID にも、他のユーザ ID に同じパーミッションを **GRANT** (付与) するパーミッションが与えられます。

- **ALL** パーミッション – 構文 3 では、以下のすべてのパーミッションを付与します。
- **ALTER** パーミッション – ユーザは **ALTER TABLE** 文を使用してこのテーブルを変更できます。このパーミッションをビューには適用できません。
- **DELETE** パーミッション – ユーザはテーブルまたはビューからローを削除できます。
- **INSERT** パーミッション – ユーザは指定のテーブルまたはビューにローを挿入できます。
- **REFERENCES** パーミッション – ユーザは、指定したテーブルのインデックスを作成できます。また、指定したテーブルを参照する外部キーを作成できます。カラム名を指定する場合は、ユーザは指定したカラムだけを参照できます。カラムの REFERENCES パーミッションは、ビューに対して付与できません。テーブルのみに付与できます。
- **SELECT** パーミッション – ユーザはビューまたはテーブルの情報を参照できます。カラム名を指定すると、ユーザは指定したカラムだけを参照できます。カ

ラムの SELECT パーミッションは、ビューに対して付与できません。テーブルのみに付与できます。

- **UPDATE** パーミッション – ユーザはビューまたはテーブルの情報を更新できます。カラム名を指定すると、ユーザは指定したカラムだけを更新できます。カラムの UPDATE パーミッションは、ビューに対しては付与されません。テーブルだけに対して付与されます。テーブルを更新するには、ユーザはそのテーブルに対する SELECT パーミッションと UPDATE パーミッションを保有している必要があります。

たとえば、`Employees` テーブルに対する SELECT パーミッションと UPDATE パーミッションをユーザ `Laurel` に付与するには、次のように入力します。

```
GRANT
SELECT, UPDATE ( street )
ON Employees
TO Laurel
```

EXECUTE ON 句 – プロシージャを実行するパーミッションを付与します。

INTEGRATED LOGIN TO 句 – 1 つまたは複数の Windows ユーザ・プロファイルと既存のデータベース・ユーザ ID の間に、明示的な統合化ログインを作成します。これによって、ローカル・マシンにログインできたユーザは、ユーザ ID またはパスワードを指定しなくてもデータベースに接続できます。

KERBEROS LOGIN TO 句 – 1 つまたは複数の Kerberos プリンシパルから既存のデータベース・ユーザ ID に認証済み Kerberos ログイン・マッピングを作成します。これによって、Kerberos に正常にログインしたユーザ (有効な Kerberos TGT (Ticket Granted Ticket) を持ったユーザ) は、ユーザ ID またはパスワードを入力しないで、データベースに接続できます。 『SQL Anywhere サーバー - SQL リファレンス』の「SQL 文」>「SQL 文」>「GRANT 文」を参照してください。

CONNECT TO 句 – 新しいユーザを作成します。また、**GRANT CONNECT** を使用して、任意のユーザが自分のパスワードを変更することもできます。

注意： ユーザを作成するには、**CREATE USER** 文を使用することをおすすめします。詳細については、「CREATE USER 文」を参照してください。

パスワードとして空の文字列を持つユーザを作成するには、次のように入力します。

```
GRANT CONNECT TO userid IDENTIFIED BY ""
```

DBA 権限または PERMS ADMIN 権限がある場合は、次のコマンドを使用して既存の任意のユーザのパスワードを変更できます。

```
GRANT CONNECT TO userid IDENTIFIED BY password
```

また、同じコマンドを使用して新しいユーザを追加できます。このため、新しいユーザを追加するときに既存のユーザのユーザ ID を誤って入力すると、その既存ユーザのパスワードを変更することになります。 これは正常な動作とみなされるため、警告は発生しません。 この動作は、バージョン 12 より前の Sybase IQ とは異なっています。

こうした操作を実施しないようにするには、システム・プロシージャの **sp_addlogin** と **sp_adduser** を使用してユーザを追加します。 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 と同じく、既存のユーザ ID を追加しようとするとエラーが発生します。

注意： ユーザ ID の追加と削除を行うには、**GRANT** または **REVOKE** ではなくシステム・プロシージャを使用します。

パスワードのないユーザを作成するには、次のように入力します。

```
GRANT CONNECT TO userid
```

ユーザ ID の大文字小文字は区別されません。

パスワードのないユーザは、データベースに接続できません。これは、グループ・ユーザ ID への接続をすべて拒否する場合のグループ作成に便利です。

パスワードを有効な識別子にする必要があります。『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 言語の要素」>「識別子」を参照してください。識別子の最大長は、255 バイトです。データベース・オプション VERIFY_PASSWORD_FUNCTION に空の文字列以外が設定されている場合、**GRANT CONNECT TO** *userid* **IDENTIFIED BY** *password* 文はそのオプションの値で識別される関数を呼び出します。その関数が NULL を返す場合は、パスワードが規則に従っていることを示します。VERIFY_PASSWORD_FUNCTION オプションが設定されている場合、**GRANT CONNECT** 文には *userid* と *password* をそれぞれ 1 つだけ指定できます。詳細については、「VERIFY_PASSWORD_FUNCTION オプション」を参照してください。

次の名前は、データベース・ユーザ ID とパスワードでは無効です。

- 空白文字または一重引用符や二重引用符で始まる名前
- 空白文字で終わる名前
- セミコロンを含む名前

CREATE ON 句 – 指定した DB 領域での CREATE パーミッションを指定のユーザやグループに付与します。

詳細については、「REVOKE 文」を参照してください。

関連する動作：

- オートコミット。

注意： これらのリファレンスは SQL Anywhere マニュアルにリンクされています。

標準

- SQL—構文 3 は初級レベル機能です。構文 4 は永続的ストアド・モジュール機能です。その他の構文は、ISO/ANSI SQL 文法のベンダ拡張です。
- Sybase—構文 1 と構文 3 は Adaptive Server Enterprise でサポートされています。セキュリティ・モデルは Adaptive Server Enterprise と Sybase IQ では異なるため、他の構文も異なります。

パーミッション

- 構文 1 と構文 2 の場合は、次の条件のいずれかを満たす必要があります。
  - DBA 権限を任意のユーザに付与するには、DBA 権限を持っている必要がある。
  - GRANT GROUP、GRANT MEMBERSHIP IN GROUP を実行したり、他の権限を任意のユーザに付与したりするには、DBA 権限または PERMS ADMIN 権限を持っている必要がある。
- 構文 3 の場合は、次の条件のいずれかを満たす必要があります。
  - テーブルを作成している。
  - **GRANT OPTION** を使用して、テーブルに対するパーミッションが付与されている。
  - DBA 権限または PERMS ADMIN 権限を持っている。
- 構文 4 の場合は、次の条件のいずれかを満たす必要があります。
  - プロシージャを作成している。
  - DBA 権限または PERMS ADMIN 権限を持っている。
- 構文 5 の場合は、DBA 権限または USER ADMIN 権限が必要です。
- 構文 6 の場合は、DBA 権限または USER ADMIN 権限が必要です。
- 構文 7 の場合は、次の条件のいずれかを満たす必要があります。
  - 新しいユーザを作成する場合は、DBA 権限または USER ADMIN 権限を持っている。
  - 自分のパスワードを変更しようとしている。
  - 別のユーザのパスワードを変更する場合は、DBA 権限または PERMS ADMIN 権限を持っている。

  別のユーザのパスワードを変更する場合は、そのユーザがデータベースに接続していないことが必要です。
- 構文 8 の場合は、DBA 権限または SPACE ADMIN 権限が必要です。

参照：

- CREATE USER 文 (182 ページ)
- REVOKE 文 (331 ページ)
- VERIFY_PASSWORD_FUNCTION オプション (552 ページ)

IF 文

SQL 文を条件付きで実行します。

構文

```
IF search-condition THEN statement-list
... [ ELSEIF search-condition THEN statement-list ]...
... [ ELSE statement-list ]
... END IF
```

例

- **例 1 –** 次のプロシージャは、**IF** 文の使用法を示します。

```
CREATE PROCEDURE TopCustomer (OUT TopCompany CHAR(35), OUT
TopValue INT)
BEGIN
    DECLARE err_notfound EXCEPTION
    FOR SQLSTATE '02000' ;
    DECLARE curThisCust CURSOR FOR
    SELECT CompanyName, CAST(     sum(SalesOrderItems.Quantity *
    Products.UnitPrice) AS INTEGER) VALUE
    FROM Customers
    LEFT OUTER JOIN SalesOrders
    LEFT OUTER JOIN SalesOrsderItems
    LEFT OUTER JOIN Product
    GROUP BY CompanyName ;


    DECLARE ThisValue INT ;
    DECLARE ThisCompany CHAR(35) ;
    SET TopValue = 0 ;
    OPEN curThisCust ;
    CustomerLoop:
    LOOP
        FETCH NEXT curThisCust
        INTO ThisCompany, ThisValue ;
        IF SQLSTATE = err_notfound THEN
            LEAVE CustomerLoop ;
        END IF ;
        IF ThisValue > TopValue THEN
            SET TopValue = ThisValue ;
            SET TopCompany = ThisCompany ;
        END IF ;
    END LOOP CustomerLoop ;
```

```
    CLOSE curThisCust ;
END
```

- **例 2 –** 次のプロシージャ内の複合文は、**ELSEIF** 文の使用法を示します。

```
BEGIN
 DECLARE X INT;
 SET X = 1;
 IF X = 1 THEN
  PRINT '1';
  ELSEIF X = 2 THEN
   PRINT '2';
  ELSE
   PRINT 'something else';
 ENDIF
END
```

使用法

IF 文を使用すると、*search-condition* が TRUE である SQL 文の最初のリストを、条件付きで実行できます。

いずれの *search-condition* も TRUE でなくて **ELSE** 句が存在する場合は、**ELSE** 句にある *statement-list* が実行されます。いずれの *search-condition* も TRUE でなくて **ELSE** 句も存在しない場合は、式は NULL 値を返します。

END IF の後に記述されている最初の文から実行が再開されます。

IF 文の内部の **SELECT** 文から返された 1 つの値を変数と比較する場合は、最初に **SELECT** の結果を別の変数に割り当てる必要があります。

注意： **IF** 文の構文と IF 式の構文を混同しないでください。

IF 文はネストできません。

IF 式の詳細については、『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 言語の要素」>「式」を参照してください。

標準

- SQL—ISO/ANSI SQL 準拠。
- Sybase—Transact-SQL の **IF** 文は、構文が異なります。

パーミッション

なし

参照：

- BEGIN … END 文 (60 ページ)

IF 文 [T-SQL]

Sybase IQ の **IF** 文の代わりとして、Transact-SQL の文を条件付きで実行します。

構文

```
            IF expression
... statement
... [ ELSE [ IF expression ] statement ]...
```

例

- **例 1 –** 次の例は、Transact-SQL の **IF** 文の使用法を示します。

```
IF (SELECT max(id) FROM sysobjects) < 100
  RETURN
ELSE
  BEGIN
    PRINT 'These are the user-created objects'
    SELECT name, type, id
    FROM sysobjects
    WHERE id < 100
END
```

- **例 2 –** 次の例は、Transact-SQL の **ELSEIF** 文の使用法を示します。

```
BEGIN
 DECLARE @X INT
 SET @X = 1
 IF @X = 1
  PRINT '1'
  ELSEIF @X = 2
   PRINT '2'
  ELSE
   PRINT 'something else'
END
```

使用法

Transact-SQL の **IF** 条件と **ELSE** 条件は、それぞれ単一の SQL 文または複合文 (キーワード **BEGIN** と **END** の間) の実行を制御します。

Sybase IQ の **IF** 文とは異なり、Transact-SQL の **IF** 文には **THEN** がありません。 また、Transact-SQL の IF 文には **ELSEIF** や **END IF** のキーワードもありません。

IF 文の内部の **SELECT** 文から返された 1 つの値を変数と比較する場合は、最初に **SELECT** の結果を別の変数に割り当てる必要があります。

注意： **IF** 文はネストできません。

標準

- SQL—ISO/ANSI SQL 文法の Transact-SQL 拡張。
- Sybase—Adaptive Server Enterprise は、Transact-SQL の **IF** 文をサポートします。

パーミッション

なし

INCLUDE 文 [ESQL]

SQL ソース言語プリプロセッサによってスキャンされたソース・プログラムにファイルをインクルードします。

構文

```
INCLUDE
filename
```

パラメータ

- **filename：** – 識別子

使用法

INCLUDE 文は、C プリプロセッサの **#include** 指令と非常によく似ています。

ただし、SQL プリプロセッサは指定のファイルを読み込んで、その内容を出力 C ファイルに挿入します。したがって、インクルード・ファイルに SQL プリプロセッサが必要とする情報がある場合は、Embedded SQL の **INCLUDE** 文を使用して、その情報をインクルードしてください。

具体的には、 SQLCA と SQLDA という 2 つのファイル名が認識されます。Embedded SQL を使用する C プログラムは、すべての Embedded SQL 文の前に次の文を指定する必要があります。

```
EXEC SQL INCLUDE SQLCA;
```

この文を、C 言語プログラムの静的変数宣言を指定できる場所に入れます。多くの Embedded SQL 文では、SQLCA INCLUDE 文の位置に、SQL プリプロセッサによって宣言された変数 (プログラマには見えない) が必要です。SQLDA を使用している場合は、SQLDA ファイルをインクルードしてください。

標準

- SQL—ISO/ANSI SQL 準拠。
- 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

INSERT 文

現在のデータベースの別の場所から、単一のロー (構文 1) または選択されたロー (構文 2) をテーブルに挿入します。別のデータベースから選択されたローをテーブルに挿入することもできます (構文 3)。

構文

構文 1

```
INSERT [ INTO ] [ owner.]table-name [ ( column-name [, …] ) ]
... VALUES ( [ expression | DEFAULT,… ) ]
or
INSERT [ INTO ] [ owner.]table-name DEFAULT VALUES
```

構文 2

```
INSERT [ INTO ] [ owner.]table-name [ ( column-name [, …] ) ]
... insert-load-options insert-select-load-options
... select-statement
```

構文 3

```
INSERT [ INTO ] [ owner.]table-name[ ( column-name [, …] ) ]
... insert-load-options insert-select-load-options
LOCATION 'servername.dbname'
[ location-options ]
... { { select-statement } | ‘select statement’ }
```

パラメータ

- **insert-load-options：** – [**LIMIT***number-of-rows*] [**NOTIFY***number-of-rows*] [**SKIP***number-of-rows*] [**START ROW ID***number*]
- **insert-select-load-options：** – [**WORD SKIP***number*] [**IGNORE CONSTRAINT***constrainttype* [, …]] [**MESSAGE LOG** ‘*string*’ **ROW LOG** ‘*string*’ [**ONLY LOG***logwhat* [, …]]] [**LOG DELIMITED BY** ‘*string*’]
- **constrainttype：** – { **CHECK***integer* | **UNIQUE***integer* | **NULL***integer* | **FOREIGN KEY***integer* | **DATA VALUE***integer* | **ALL***integer* }

- **logwhat：** – { **CHECK** | **ALL** | **NULL** | **UNIQUE** | **DATA VALUE** | **FOREIGN KEY** | **WORD** }
- **location-options：** – [**ENCRYPTED PASSWORD**] [**PACKETSIZE***packet-size*] [**QUOTED_IDENTIFIER** { **ON** | **OFF** }] [**ISOLATION LEVEL** { **READ UNCOMMITTED** | **READ COMMITTED** | **SERIALIZABLE** }]

例

- **例 1 –** データベースに Eastern Sales 部を追加します。

```
INSERT INTO Departments
(DepartmentID, DepartmentName, DepartmentHeadID)
VALUES (600, 'Eastern Sales', 501)
```

- **例 2 –** テーブル `dept_head` に、部長とその部の名称を入力します。

```
INSERT INTO dept_head (name, dept)
  NOTIFY 20
  SELECT Surname || ' ' || GivenName
  AS name,
  dept_name
FROM Employees JOIN Departments
  ON EmployeeID= DepartmentHeadID
```

- **例 3 –** リモート・サーバ `detroit` 上の Sybase IQ データベース `iqdet` にある `lineitem` テーブルの `l_shipdate` カラムと `l_orderkey` カラムのデータを、現在のデータベースにある `lineitem` テーブルの対応するカラムに挿入します。

```
INSERT INTO lineitem
  (l_shipdate, l_orderkey)
  LOCATION 'detroit.iqdet'
  PACKETSIZE 512
  ' SELECT l_shipdate, l_orderkey
FROM lineitem '
```

- **例 4 (カタログ・ストアのみ) –** この例では、テーブルに 3 つのローを挿入します。2 つ以上の値リストの挿入がサポートされているのは、カタログ・ストア (SQL Anywhere) テーブルのみです。この構文を IQ テーブルで使用すると、エラーになります。

```
INSERT INTO T (c1,c2,c3)
VALUES (1,10,100), (2,20,200), (3,30,300);
```

- **例 5 (カタログ・ストアのみ) –** この例の INSERT 文は、3 つのローを 4 つのカラムから構成されるテーブルに挿入します。このテーブルでは、各カラムがデフォルト値を持ちます。2 つ以上の値リストの挿入がサポートされているのは、カタログ・ストア (SQL Anywhere) テーブルのみです。この構文を IQ テーブルで使用すると、エラーになります。

```
INSERT INTO T ()
VALUES (), (), ();
```

使用法

構文 1 を使用すると、指定された式の値で、1 つのローを挿入できます。カラム名のリストを指定しないと、作成順 (**SELECT *** を使って取り出すときと同じ順序) に値がテーブル・カラムの中に挿入されます。ローは、テーブルの任意の位置に挿入されます(リレーショナル・データベースでは、テーブルは順序付けられません)。

構文 2 を使用すると、ユーザは、通常とまったく同じ **SELECT** 文の結果を使用したテーブルへの大量挿入を実行できるようになります。**SELECT** 文に **ORDER BY** 句が含まれていない場合、任意の順序で挿入が行われます。select リストのカラムは、通常、カラム・リストに指定したカラムと一致しているか、カラムの作成順に並んでいます。

注意： **NUMBER(*)** 関数は、**INSERT** 文の構文 2 を使用してプライマリ・キーを生成する場合に役立ちます。詳細については、『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 関数」を参照してください。

構文 3 の **INSERT...LOCATION** は構文 2 の変形であり、Adaptive Server Enterprise または Sybase IQ のデータベースからデータを挿入できます。 **LOCATION** 句で指定される *servername.dbname* は、**FROM** 句のテーブルのリモート・サーバとデータベースを識別します。構文 3 を使用するには、接続中の Adaptive Server Enterprise または Sybase IQ のリモート・サーバが、ローカル・マシン上の Sybase Open Client の `interfaces` ファイルまたは `sql.ini` ファイルに記述されている必要があります。

構文 3 を使用したクエリでは、挿入可能なローの数は 2147483647 までです。

SELECT 文を区切るには、中カッコまたは一重引用符を使用できます。

注意： 中カッコは、ODBC 標準ではエスケープ・シーケンスの開始と終了を表すため、ODBC または Sybase Central のコンテキストではエラーが発生する可能性があります これに対処するには、一重引用符を使用して **SELECT** 文をエスケープします。

ローカルの Sybase IQ サーバは、**LOCATION** 句で指定したサーバとデータベースに接続します。 リモート・テーブルに対するクエリの結果が返され、ローカル・サーバによって結果が現在のデータベースに挿入されます。 **LOCATION** 句でサーバ名を指定しない場合、ローカル・サーバにある現在のデータベース以外は選択できなくなるため、Sybase IQ はどのデータベース名が指定されたとしてもこれを無視します。

Sybase IQ がリモート・サーバに接続するときに、**CREATE EXTERNLOGIN** でリモート・ログインが作成されており、**CREATE SERVER** 文でリモート・サーバが定義されている場合、**INSERT...LOCATION** は現在の接続のユーザ ID のリモート・ログイ

ンを使用できます。リモート・サーバが定義されていないか、現在の接続のユーザ ID に対するリモート・ログインが作成されていない場合、Sybase IQ は現在の接続のユーザ ID とパスワードを使用して接続します。

注意： 現在の接続のユーザ ID とパスワードを使用している場合に、ユーザがパスワードを変更すると、リモート・サーバで新しいパスワードが有効になる前に、サーバを停止して再起動する必要が生じます。**CREATE EXTERNLOGIN** を使用して作成したリモート・ログインは、デフォルト・ユーザ ID のパスワードが変更されても影響を受けません。

CREATE EXTERNLOGIN 文でリモート・ログインを作成し、**CREATE SERVER** 文でリモート・サーバを定義すると、ユーザがどのような状況でもログインとパスワードを使用できるように、**INSERT...LOCATION** の外部ログインとパスワードが設定されます。 これによって、ログインまたはパスワードにアクセスできないことが原因で発生する可能性のあるエラーを回避できます。この方法で、リモート・サーバに接続することをおすすめします。

たとえば、ユーザ `russid` が Sybase IQ データベースに接続して、次の文を実行します。

```
INSERT local_SQL_Types LOCATION ‘ase1.ase1db’
{SELECT int_col FROM SQL_Types};
```

サーバ `ase1` にはユーザ `ase1user` とそのパスワード `sybase` があります。テーブル `SQL_Types` の所有者は、`ase1user` です。リモート・サーバは、IQ サーバでは次のように定義されます。

```
CREATE SERVER ase1 CLASS ‘ASEJDBC’
USING ‘system1:4100’;
```

外部ログインは、IQ サーバでは次のように定義されます。

```
CREATE EXTERNLOGIN russid TO ase1 REMOTE LOGIN ase1user IDENTIFIED BY
sybase;
```

INSERT...LOCATION は、ユーザ ID `ase1user` とユーザ `russid` に対するパスワード `sybase` を使用して、リモート・サーバ `ase1` に接続します。

ENCRYPTED PASSWORD パラメータを指定すると、リモート・サーバへの接続時に、Open Client ライブラリによるデフォルトのパスワード暗号化の使用を指定できるようになります。**ENCRYPTED PASSWORD** が指定され、リモート・サーバが Open Client ライブラリのデフォルトのパスワード暗号化をサポートしない場合、無効なユーザ ID またはパスワードが使用されたことを示すエラーがレポートされます。

Sybase IQ をリモート・サーバとして使用した場合、TDS のパスワード暗号化がサポートされます。 Sybase IQ サーバは、クライアントが送信した、暗号化されたパ

スワードのある接続を受け入れます。パスワードの暗号化を設定するための接続プロパティについては、Software Developer's Kit 15.5 の Open Server 15.5 にある『Open Client Client-Library/C リファレンス・マニュアル』>「Client-Library トピックス」>「セキュリティ機能」>「Adaptive Server Enterprise のセキュリティ機能」>「セキュリティ・ハンドシェイク： 暗号化されたパスワード」を参照してください。

注意： パスワード暗号化には Open Client 15.0 が必要です。TDS のパスワード暗号化には Open Client 15.0 ESD #7 以降が必要です。

Sybase IQ サーバが、暗号化されたパスワードのある jConnect 接続を受け入れるようにするには、jConnect **ENCRYPT_PASSWORD** 接続プロパティを true に設定します。

PACKETSIZE パラメータには、TDS パケット・サイズをバイト数で指定します。デフォルトの TDS パケット・サイズは、ほとんどのプラットフォームで 512 バイトです。アプリケーションで大量のテキストまたはバルク・データをネットワーク経由で受信する場合は、パケット・サイズを大きくすることでパフォーマンスが格段に向上できる可能性があります。

packet-size には、512 の倍数で、デフォルトのネットワーク・パケット・サイズと同じ値か、ネットワーク・パケット・サイズのデフォルト値から最大値までの値を指定してください。ネットワーク・パケット・サイズの最大値およびデフォルト値は、512 から 524288 バイトまでの 512 の倍数です。 ネットワーク・パケット・サイズの最大値は常に、デフォルトのネットワーク・パケット・サイズ以上になります。 ネットワーク・パケット・サイズの詳細については、『Adaptive Server Enterprise システム管理ガイド 第 1 巻』を参照してください。

INSERT...LOCATIONPACKETSIZE*packet-size* が指定されないか、ゼロに指定された場合は、プラットフォームのデフォルトのパケット・サイズ値が使用されます。

INSERT...LOCATION が Sybase IQ サーバとリモートの Sybase IQ サーバまたは Adaptive Server Enterprise サーバとの間でデータを転送する場合、**PACKETSIZE** に 512 バイト以外の値を指定しても、**INSERT...LOCATION** TDS **PACKETSIZE** パラメータの値は常に 512 バイトになります。

注意： 不正なパケット・サイズ (512 の倍数でない 933 など) を指定すると、Open Client **ct_connect** "Connection failed" エラーが表示されて接続の試行が失敗します。接続の試行が失敗すると、必ず "Connection failed" という一般的なメッセージが返されます。 Adaptive Server Enterprise のエラー・ログを調べれば、接続失敗の原因について、より詳細な情報を入手できます。

QUOTED_IDENTIFIER パラメータを使用すると、リモート・サーバで `QUOTED_IDENTIFIER` オプションの設定を指定できます。デフォルト設定は

'OFF' です。**SELECT** 文の識別子がすべて二重引用符で囲まれている場合のみ (次の例では 'c1')、**QUOTED_IDENTIFIER** を 'ON' にします。

```
INSERT INTO foo
LOCATION 'ase.database'
QUOTED_IDENTIFIER ON {select "c1" from xxx};
```

ISOLATION LEVEL パラメータを使用すると、リモート・サーバへの接続用の独立性レベルを指定できます。

独立性レベル	特性
READ UNCOMMITTED	• 独立性レベル 0。 • 書き込みロックの有無にかかわらず、ローで読み込みが許可される。 • 読み込みロックは適用されない。 • 同時トランザクションがローを変更しないこと、またはローに対しての変更がロールバックされないことは保証されない。
READ COMMITTED	• 独立性レベル 1。 • 書き込みロックのないローでは読み込みのみ許可される。 • 現在のローでの読み込みに対してのみ読み込みロックが取得されて保持されるが、カーソルがローから移動すると解放される。 • トランザクション中にデータが変更されないという保証はない。
SERIALIZABLE	• 独立性レベル 3。 • 書き込みロックのない結果内のローに対しては読み込みのみ許可される。 • カーソルが開いているときに取得された読み込みロックは、トランザクションの終了時まで保持される。

『SQL Anywhere サーバー - SQL の使用法』の「トランザクションと独立性レベルの使用」>「独立性レベルと一貫性」を参照してください。

注意： このリファレンスは SQL Anywhere マニュアルにリンクされています。

Sybase IQ は Adaptive Server Enterprise のデータ型である TEXT をサポートしませんが、長さが 255 バイトを超える IQ の CHAR または VARCHAR カラムから **INSERT...LOCATION** (構文 3) を実行できます。また、ASE データベース・カラムのデータ型である TEXT からも実行可能です。 ASE の TEXT カラムと IMAGE カラムは、Sybase IQ が内部変換をサポートしていれば、他の Sybase IQ データ型のカラムに挿入できます。 デフォルトで、リモート・データ・カラムが 2GB を超える場合、Sybase IQ では超えた分のカラムの値が自動的にトランケートされます。

警告！ Sybase IQ では、Adaptive Server Enterprise の UNICHAR、UNIVARCHAR、UNITEXT の各データ型をサポートしていません。 **INSERT...LOCATION** コマンドを使用して、ISO_BINENG 照合で UNICHAR または UNITEXT のカラムから CHAR または CLOB カラムに変換する操作が、エラーなく実行される場合があります。その場合は、カラム内のデータの整合性が失われている可能性があります。 この場合にエラーが報告されるのは、変換に失敗した場合にかぎられます。

非構造化データ分析オプションのラージ・オブジェクト機能を使用するには、正規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 の非構造化データ分析の概要』を参照してください。

注意： **INSERT...LOCATION** を使用して **VARBINARY** カラムから選択されたデータを挿入する場合は、リモート・データベースの **ASE_BINARY_DISPLAY** を OFF に設定してください。

INSERT...LOCATION (構文 3) は **SELECT** 文での変数の使用をサポートしません。

ビューに挿入できるのは、ビューを定義する **SELECT** 文が **FROM** 句の中でテーブルを 1 つだけ持ち、**GROUP BY** 句と集合関数を含まないか、または **UNION** 演算を伴わない場合です。

テーブルに挿入した文字列は、データベースが大文字と小文字を区別するかどうかに関係なく、常に入力された大文字と小文字がそのまま格納されます。したがって、文字列 "Value" をテーブルに挿入すると、V は大文字、残りの英字は小文字で、常にデータベースに格納されます。**SELECT** 文は、*Value* として文字列を返します。ただし、データベースが大文字と小文字を区別しない場合は、すべての比較で *Value*、*value*、*VALUE* は同じものとして扱われます。さらに、単一カラムのプライマリ・キーにエントリ Value がある場合は、プライマリ・キーがユニークでなくなるので、**INSERT** 文による value の追加は拒否されます。

ユーザが **INSERT ... LOCATION** 文を実行すると、Sybase IQ は言語、照合順、文字セット、日付と時刻の形式を判断するために必要なローカライゼーション情報ロードします。 データベースがプラットフォームのデフォルト以外のロケールを使用している場合は、ローカル・クライアントに環境変数を設定して、Sybase IQ が正しい情報をロードするようにしてください。

環境変数 LC_ALL を設定すると、Sybase IQ はその値をロケール名として使用します。 LC_ALL が設定されていない場合、Sybase IQ は LANG 環境変数の値を使用します。 どちらの環境変数も設定されていない場合は、Sybase IQ はロケール・ファイルにあるデフォルトのエントリを使用します。 例については、『システム管理ガイド： 第 1 巻』の「各国語と文字セット」>「INSERT...LOCATION 文のロケールの設定」を参照してください。

DEFAULT VALUES 句と **VALUES** 句を使用すると、挿入する値を指定できます。**CREATE TABLE** 文で指定されたデフォルトのカラム値を挿入するには、**DEFAULT VALUES** を指定します。**DEFAULT VALUES** を指定することは、実質的には次の明示的な構文を指定することと同じです。

```
INSERT [INTO} <tablename>
VALUES(default, default, ..., default)
```

ここで、デフォルトのエントリの数は、テーブル内のカラムの数と同じです。次に例を示します。

```
INSERT INTO table1 DEFAULT VALUES
```

INSERT VALUES(DEFAULT ...) 句を使用して、NULL カラムに挿入することもできます。

LIMIT オプションは、クエリからテーブルに挿入するローの最大数を指定します。制限なしのデフォルトは 0 です。最大値は 2GB -1 です。

NOTIFY オプションは、指定した個数のローがテーブルに正常に挿入されるたびに、メッセージが通知されるように指定します。デフォルトは 100,000 ローごとです。

SKIP オプションを使用して、この挿入の場合に入力テーブルの開始時にスキップするローの数を定義します。デフォルトは 0 です。

START ROW ID オプションは、IQ テーブル内の挿入を開始するローのレコード識別番号を指定します。デフォルトでは、テーブルに領域がある場合、新しいローが挿入され、各挿入により新しいローが開始されます。

LOAD TABLE コマンドと **INSERT** コマンドの **START ROW ID** 句は、分割されたテーブルでは使用できません。

*insert-select-load-options***WORD SKIP**、**IGNORE CONSTRAINT**、**MESSAGE LOG**、**ROW LOG**、**LOG DELIMITED BY**、*constrainttype* パラメータと *logwhat* パラメータについては、「LOAD TABLE 文」を参照してください。

マルチカラム・インデックスに対する **INSERT** には、インデックスのすべてのカラムを含める必要があります。

Sybase IQ は、**INSERT...VALUES**、**INSERT...SELECT**、および **INSERT...LOCATION** のカラムの DEFAULT 値をサポートします。 カラムに DEFAULT 値が指定されている場合、カラムの値を指定しないすべての **INSERT** 文 (または **LOAD** 文) でカラムの値としてこの DEFAULT 値が使用されます。

挿入に伴うカラムの 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。

プロシージャまたは関数で **COMMIT** 文、**ROLLBACK** 文、または一部の **ROLLBACK TO SAVEPOINT** 文を使用している場合は、ストアド・プロシージャまたは関数から **INSERT** を使用することはできません。詳細については、『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」>「制御文」>「アトミックな複合文」と『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」>「プロシージャでのトランザクションとセーブポイント」を参照してください。

SELECT…FROM の結果は、**INSERT…SELECT…FROM** の結果と少し異なる場合があります。これは、`DOUBLE` や `NUMERIC` などの不正確なデータ型の内部データ変換によって、挿入時に最適化が行われるためです。より正確な結果が必要な場合は、より高い精度の `DOUBLE` または `NUMERIC` のデータ型としてカラムを宣言します。

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」>「INSERT 文の使用」を参照してください。

標準

- SQL—ISO/ANSI SQL 準拠。
- Sybase—Adaptive Server Enterprise でサポートされています (*insert-load-options* は除く)。

パーミッション

テーブルに対する INSERT パーミッションが必要です。

参照：

- CREATE EXTERNLOGIN 文 (108 ページ)
- DELETE 文 (201 ページ)
- LOAD TABLE 文 (273 ページ)
- SYNCHRONIZE JOIN INDEX 文 (367 ページ)

INSTALL JAVA 文

データベース内で Java クラスを使用できるようにします。

構文

```
INSTALL JAVA [ install-mode ] [ JAR jar-name ] FROM source
```

パラメータ

- **install-mode：** – { **NEW** | **UPDATE** }
- **source：** – { **FILE***filename* | **URL***url-value* }

例

- **例 1 –** 次の文は、クラスのファイル名とロケーションを指定して、ユーザの作成した "Demo" という名前の Java クラスをインストールします。

  ```
  INSTALL JAVA NEW
  FROM FILE 'D:¥JavaClass¥Demo.class'
  ```

  インストール後、クラスはその名前で参照されます。その元のファイル・パスの位置は使用されなくなります。たとえば、次の文は前の文でインストールしたクラスを使用します。

  ```
  CREATE VARIABLE d Demo
  ```

  Demo クラスがパッケージ sybase.work のメンバの場合は、次の例のように、完全に修飾されたクラス名を使用してください。

  ```
  CREATE VARIABLE d sybase.work.Demo
  ```

- **例 2 –** 次の文は、zip ファイルに含まれるすべてのクラスをインストールし、それらを JAR ファイル名でデータベース内に関連付けます。

  ```
  INSTALL JAVA
  JAR 'Widgets'
  FROM FILE 'C:¥Jars¥Widget.zip'
  ```

  zip ファイルのロケーションは保持されません。クラスは完全に修飾されたクラス名 (パッケージ名とクラス名) を参照しなければなりません。

使用法

- **インストール・モード –** インストール・モードに **NEW** を指定するには、参照される Java クラスが、現在インストールされているクラスの更新ではなく、新しいクラスである必要があります。 データベース内に同じ名前のクラスがすでにある場合、インストール・モードに **NEW** を指定すると、エラーになります。

  インストール・モードに **UPDATE** を指定すると、指定したデータベースにすでにインストールされている Java クラスの代わりとして使用する Java クラスを、参照される Java クラスに含めることができます。

  クラスまたは JAR を更新するには、DBA 権限と、ディスク上のファイルで使用できる新バージョンのコンパイルされたクラス・ファイルまたは JAR ファイルが必要です。

  新しい定義を使用するのは、クラスのインストール後に設定された新しい接続か、クラスのインストール後最初にそのクラスを使用する接続だけです。Java

VM がクラス定義をロードすると、クラス定義は接続が閉じるまでメモリに保存されます。

現在の接続で Java クラスまたはクラスを基にしたオブジェクトを使用している場合、新しいクラス定義を使用するには接続をいったん切断し、その後再接続する必要があります。

インストール・モードが省略されると、デフォルトは **NEW** です。

- **JAR –** これを指定する場合、*file-name* または *text-pointer* には、JAR ファイル、または JAR を含むカラムを指定する必要があります。 JAR ファイルには、通常 `.jar` または `.zip` の拡張子が付きます。

  インストールされた JAR および zip ファイルは、圧縮または展開できます。ただし、Sun JDK **jar** ユーティリティによって作成された JAR ファイルはサポートされていません。 それ以外の zip ユーティリティによって作成されたファイルはサポートされています。

  jar オプションを指定する場合、JAR は、含まれているクラスがインストールされた後で JAR として保持されます。 この JAR は、これらの各クラスに関連付けられた JAR です。 **JAR** オプションによってデータベースにインストールされている JAR のセットは、データベースの「保持された JAR」と呼ばれます。

  保持された JAR は、**INSTALL** 文と **REMOVE** 文によって参照されます。保持された JAR は、他の Java-SQL クラスの使用には影響を与えません。SQL システムは、他のシステムによって、指定されたデータに関連付けられたクラスを要求された場合に、保持された JAR を使用します。 要求されたクラスに、関連付けられている JAR がある場合、SQL システムは、個々のクラスではなく、その JAR を提供します。

  jar-name は長さが 255 バイトまでの文字列値です。*jar-name* は、後続の **INSTALL** 文、**UPDATE** 文、**REMOVE** 文に保持された JAR を識別するのに使用します。
- **source –** インストールする Java クラスのロケーションを指定します。

  file-name でサポートされるフォーマットは、`'c:¥libs¥jarname.jar'` や `'/usr/u/libs/jarname.jar'` のような完全に修飾されたファイル名と、データベース・サーバの現在の作業ディレクトリを基準にした相対ファイル名です。

  filename には、クラス・ファイルまたは JAR ファイルを指定する必要があります。

  各クラスのクラス定義は、最初にそのクラスを使用するときに各接続の VM でロードされます。クラスを **INSTALL** (インストール) すると、接続の VM が暗黙的に再起動されます。 このため、**INSTALL** 文の *install-mode* が **NEW** または **UPDATE** かに関わらず、すぐに新しいクラスにアクセスできます。

これ以外の接続では、新しいクラスがロードされるのは、次回 VM がそのクラスに初めてアクセスするときです。 クラスがすでに VM によってロードされている場合、VM がその接続のために (たとえば、**STOP JAVA** や **START JAVA** を使用して) 再起動されるまで、接続からはその新しいクラスは見えません。

標準

- 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。

パーミッション

- **INSTALL** 文を実行するには、DBA 権限が必要です。
- すべてのインストールされたクラスは、すべてのユーザがすべての方法で参照できます。

参照：

- REMOVE 文 (319 ページ)

IQ UTILITIES 文

Sybase IQ データベースのバッファ・キャッシュに関する統計情報を収集します。

構文

```
IQ UTILITIES { MAIN | PRIVATE }
[ INTO ] table-name
{ START MONITOR ['monitor-options']
| STOP MONITOR }
```

パラメータ

- **monitor-options：** – { **-summary** | { **-append | -truncate** } **-bufalloc** | **-cache** | **-cache_by_type** | **-contention** | **-debug** | **-file_suffix***suffix*| **-io** | **-interval***seconds* | **-threads** }...

例

- **例 1 –** 次の例はバッファ・キャッシュ・モニタを開始し、IQ テンポラリ・バッファ・キャッシュのアクティビティを記録します。

```
IQ UTILITIES PRIVATE INTO monitor START MONITOR '-cache -interval
20'
```

使用法

START MONITOR は IQ バッファ・キャッシュ・モニタを開始します。**START** と **STOP MONITOR** では、*table_name* にダミー・テーブルを使用します。モニタリング専用のテーブルを持つのが理想的ですが、任意の IQ ベース・テーブルまたは IQ テンポラリ・テーブルを指定できます。結果は **MAIN** バッファ・ キャッシュの場合は `dbname.connection#-main-iqmon` テキスト・ファイルに格納され、**PRIVATE** (Temp) バッファ・キャッシュの場合は `dbname.connection#-temp-iqmon` に格納されます。同じデータベースと同じ接続番号からモニタを再実行すると、以前の結果を上書きします。モニタの出力ファイルのディレクトリ位置を設定するには、`MONITOR_OUTPUT_DIRECTORY` オプションを使用します。

monitor-options は結果の内容と頻度を定義します。複数回指定でき、それらを引用符で囲む必要があります。

- **-summary** メインとテンポラリ (プライベート) の両方のバッファ・キャッシュの概要情報を表示します。このオプションがデフォルトです。
- **-append | -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します。デフォルトでは、トランケートされます。
- **-bufalloc** ソート、ハッシュ、ビットマップなどのオブジェクト用にバッファ・キャッシュ内の領域を予約する、メインまたはテンポラリ・バッファ・アロケータ情報を表示します。
- **-cache** メイン・バッファ・キャッシュまたはテンポラリ・バッファ・キャッシュのアクティビティの詳細を表示します。
- **-cache_by_type** IQ ページ・タイプごとにメイン・バッファ・キャッシュまたはテンポラリ・バッファ・キャッシュのアクティビティの詳細を表示します。主に、Sybase 製品の保守契約を結んでいるサポート・センタに情報を提供するために使用します。
- **-contention** 多くの重要なバッファ・キャッシュとメモリ・マネージャ・ロックを表示します。
- **-debug** 同じ情報を扱う標準表示モードがあるかどうかに関係なく、パフォーマンス・モニタで使用可能な情報をすべて表示します。主に、Sybase 製品の保守契約を結んでいるサポート・センタに情報を提供するために使用します。
- **-file_suffix***suffix*<dbname>.<connid>-<main_or_temp>-<suffix> の名前で、モニタリング出力ファイルを作成します。デフォルトは `iqmon` です。
- **-io** メインまたはテンポラリのバッファ・キャッシュの I/O 比率とデータ圧縮率を表示します。
- **-interval** レポート間隔を秒単位で指定します。デフォルトは 60 秒ごとです。最小値は 2 秒ごとです。
- **-threads** 処理スレッド情報を表示します。

詳細については、

- 『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「システム・プロシージャ」>「システム・ストアド・プロシージャ」>「sp_iqsysmon プロシージャ」を参照してください。
- モニタ結果の例については、『パフォーマンス＆チューニング・ガイド』の「パフォーマンスのモニタリングとチューニング」を参照してください。
- **IQ UTILITIES** の高度な機能を利用した、Sybase IQ のシステム・ストアド・プロシージャの機能を拡張するプロシージャ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。

標準

- 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。

パーミッション

なし

参照：

- MONITOR_OUTPUT_DIRECTORY オプション (492 ページ)

LEAVE 文

複合文または **LOOP** を抜けて実行を続けます。

構文

```
LEAVE
statement-label
```

例

- **例 1 –** 次のコードは、**LEAVE** 文を使用してループを抜ける方法を示します。

```
SET i = 1;
lbl:
LOOP
  INSERT
  INTO Counters ( number )
  VALUES ( i ) ;
  IF i >= 10 THEN
    LEAVE lbl ;
  END IF ;
```

```
  SET i = i + 1
END LOOP lbl
```

- **例 2 –** 次のコードは、ネストされたループ内で **LEAVE** を使用します。

```
outer_loop:
LOOP
  SET i = 1;
  inner_loop:
  LOOP
    ...
    SET i = i + 1;
    IF i >= 10 THEN
      LEAVE outer_loop
    END IF
  END LOOP inner_loop
END LOOP outer_loop
```

使用法

LEAVE は、指定したラベルの複合文または指定したラベルのループを抜けるための制御文です。複合文またはループの後に記述されている最初の文から実行が再開されます。

プロシージャの本文である複合文は、プロシージャの名前と同じ暗黙のラベルを持っています。

標準

- SQL—ISO/ANSI SQL 準拠。
- Sybase—Adaptive Server Enterprise ではサポートされていません。BREAK 文は、Transact-SQL 互換のプロシージャと同様の機能を提供します。

パーミッション

なし

参照：

- BEGIN … END 文 (60 ページ)
- FOR 文 (234 ページ)
- LOOP 文 (297 ページ)

LOAD TABLE 文

外部ファイルからデータベース・テーブルにデータをインポートします。

構文

```
LOAD [ INTO ] TABLE [ owner.]table-name
... ( load-specification [, …] )
...   { FROM | USING [ CLIENT ] FILE }
{ 'filename-string' | filename-variable } [, …]
... [ CHECK CONSTRAINTS { ON | OFF } ]
... [ DEFAULTS { ON | OFF } ]
... [ QUOTES
            OFF ]
... ESCAPES
            OFF
... [ FORMAT { ascii | binary | bcp } ]
... [ DELIMITED
            BY 'string' ]
... [ STRIP { ON | OFF | RTRIM } ]
... [ WITH CHECKPOINT { ON | OFF } ]
... [ BYTE ORDER { NATIVE | HIGH | LOW } ]
... [ LIMIT number-of-rows ]
... [ NOTIFY number-of-rows ]
... [ ON FILE ERROR { ROLLBACK | FINISH | CONTINUE } ]
... [ PREVIEW { ON | OFF } ]
... [ ROW DELIMITED BY 'delimiter-string' ]
... [ SKIP number-of-rows ]
... [ HEADER SKIP number [ HEADER DELIMITED BY 'string' ] ]
... [ WORD SKIP number ]
... [ START ROW ID number ]
... [ ON PARTIAL INPUT ROW { ROLLBACK | CONTINUE } ]
... [ IGNORE CONSTRAINT constrainttype [, …] ]
... [ MESSAGE LOG 'string' ROW LOG 'string' [ ONLY LOG logwhat [, …] ]
... [ LOG DELIMITED BY 'string' ]
```

パラメータ

- **load-specification：** – { *column-name* [*column-spec*] | **FILLER** (*filler-type*) }
- **column-spec：** – { **ASCII** (*input-width*) | **BINARY** [**WITH NULL BYTE**] | **PREFIX** { **1** | **2** | **4** } | '*delimiter-string*' | **DATE** (*input-date-format*) | **DATETIME** (*input-datetime-format*) | **ENCRYPTED** (*data-type* '*key-string*' [, '*algorithm-string*']) | **DEFAULT***default-value* } [**NULL** ({ **BLANKS** | **ZEROS** | '*literal*', …})]
- **filler-type：** – { *input-width* | **PREFIX** { **1** | **2** | **4** } | '*delimiter-string*' }
- **constrainttype：** – { **CHECK***integer* | **UNIQUE***integer* | **NULL***integer* | **FOREIGN KEY***integer* | **DATA VALUE***integer* | **ALL***integer* }
- **logwhat：** – { **CHECK** | **ALL** | **NULL** | **UNIQUE** | **DATA VALUE** | **FOREIGN KEY** | **WORD** }

例

- **例 1 –** 1 つのファイルのデータを Windows 上の `Products` テーブルにロードします。`Description` カラムと `Color` カラムの後に続くカラム・デリミタとして、タブを使用します。

```
LOAD TABLE Products
( ID ASCII(6),
FILLER(1),
Name   ASCII(15),
FILLER(1),
Description   '¥x09',
Size   ASCII(2),
FILLER(1),
Color   '¥x09',
Quantity   PREFIX 2,
UnitPrice   PREFIX 2,
FILLER(2) )
FROM 'C:¥¥mydata¥¥source1.dmp'
QUOTES OFF
ESCAPES OFF
BYTE ORDER LOW
NOTIFY 1000
```

- **例 2 –** クライアント・コンピュータにあるファイル `a.inp` からデータをロードします。

```
LOAD TABLE t1(c1,c2,filler(30))
USING CLIENT FILE 'c:¥¥client-data¥¥a.inp'
QUOTES OFF ESCAPES OFF
IGNORE CONSTRAINT UNIQUE 0, NULL 0
MESSAGE LOG 'c:¥¥client-data¥¥m.log'
ROW LOG 'c:¥¥client-data¥¥r.log'ONLY LOG UNIQUE
```

- **例 3 –** 2 つのファイルからデータを UNIX システム上の `product_new` テーブル (NULL 値の使用可) にロードします。タブ文字がデフォルト・カラム・デリミタで、改行文字がロー・デリミタです。

```
LOAD TABLE product_new
( id,
name,
description,
size,
color   '¥x09'   NULL( 'null', 'none', 'na' ),
quantity   PREFIX 2,
unit_price   PREFIX 2 )
FROM '/s1/mydata/source2.dump',
'/s1/mydata/source3.dump'
QUOTES OFF
ESCAPES OFF
FORMAT ascii
DELIMITED BY '¥x09'
```

```
ON FILE ERROR CONTINUE
ROW DELIMITED BY '¥n'
```

- **例 4 –** ワード長の制限違反を 10 回は無視し、11 回目に新しいエラーを表示してロードをロールバックします。

```
load table PTAB1(
       ck1          ','  null ('NULL') ,
       ck3fk2c2       ','  null ('NULL') ,
       ck4          ','  null ('NULL') ,
       ck5          ','  null ('NULL') ,
       ck6c1         ','  null ('NULL') ,
       ck6c2         ','  null ('NULL') ,
       rid          ','  null ('NULL')  )
FROM 'ri_index_selfRI.inp'
       row delimited by '¥n'
       LIMIT 14   SKIP 10
       IGNORE CONSTRAINT UNIQUE 2, FOREIGN KEY 8
       word skip 10 quotes off escapes off strip
       off
```

- **例 5 – FORMAT BCP** ロード・オプションを使用して、データを **BCP** 文字ファイル `bcp_file.bcp` からテーブル `t1` にロードします。

```
LOAD TABLE t1 (c1, c2, c3)
FROM ‘bcp_file.bcp’
FORMAT BCP
...
```

- **例 6 – DEFAULT** ロード・オプションを使用してデフォルト値 12345 を `c1` にロードし、`LoadConst04.dat` ファイルのデータを `c2` と `c3` にロードします。

```
LOAD TABLE t1 (c1 DEFAULT ‘12345 ’, c2, c3, filler(1))
FROM ‘LoadConst04.dat’
STRIP OFF
QUOTES OFF
ESCAPES OFF
DELIMITED BY ‘,’;
```

- **例 7 – FORMAT BCP** ロード・オプションを使用して `bcp_file.bcp` ファイルのデータを `c1` と `c2` にロードし、`c3` に値 10 を設定します。

```
LOAD TABLE t1 (c1, c2, c3 DEFAULT ‘10’)
FROM ‘bcp_file.bcp’
FORMAT BCP
QUOTES OFF
ESCAPES OFF;
```

- **例 8 –** 次のコードは、データ・ファイルの先頭の 1 つのヘッダー・ローを無視します。ヘッダー・ローは、'&&' で区切られています。

```
LOAD TABLE
...HEADER SKIP 1 HEADER DELIMITED by '&&'
```

- **例 9 –** 次のコードは、データ・ファイルの先頭の 2 つのヘッダー・ローを無視します。各ヘッダー・ローは、'¥n' で区切られています。

```
LOAD TABLE
...HEADER SKIP 2
```

使用法

LOAD TABLE 文を使用すると、ASCII ファイルまたはバイナリ・データからデータベース・テーブルに大量の挿入を効率よく行うことができます。

LOAD TABLE オプションによって、整合性制約違反があった場合のロード動作を制御したり、違反についての情報をログに適宜記録したりできます。

テンポラリ・テーブル上で **LOAD TABLE** を使用できます。ただし、**ON COMMIT PRESERVE ROWS** 句を使用してテンポラリ・テーブルを宣言しておく必要があります。そうしないと、次の **COMMIT** によってロードされたローが削除されます。

複数のファイルを指定してデータをロードすることもできます。**FROM** 句で、各 `filename-string` をカンマで区切って指定します。 リソースの制約があるため、Sybase IQ ではすべてのデータがロードされる保証はありません。 リソースの割り付けに失敗した場合、そのロード・トランザクション全体がロールバックされます。 ファイルは 1 つずつ読み込まれ、**FROM** 句で指定された順に処理されます。 **SKIP** または **LIMIT** の値は、ロードの開始時に適用され、各ファイルには適用されません。

注意： マルチプレックス・データベースをロードする場合、ファイル名に絶対パス (完全に修飾されたパス) を使用します。相対パス名は使用しないでください。

LOAD TABLE は、ラージ・オブジェクト (LOB) データのロードをサポートしています。 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 は、ASCII データとバイナリ・データからのロード、および固定長と可変長の両方のフォーマットをサポートしています。 これらのフォーマットをすべて処理するには、*load-specification* を指定し、ソース・ファイルの各「カラム」またはフィールドに想定されるデータの種類を Sybase IQ に知らせる必要があります。 *column-spec* を使用すると、次のフォーマットを定義できます。

- 固定長バイトの ASCII。*input-width* 値は、各レコードの入力フィールドの固定幅のバイト数を示す整数値です。
- PREFIX のバイト数 (1、2、または 4) を使用して入力の長さを指定するバイナリまたはバイナリ以外のフィールド。
  PREFIX 句に関連する 2 つの部分があります。
  - プレフィクス値 – 常にバイナリ値です。
  - 関連データ・バイト – 常に文字フォーマットです。バイナリ・フォーマットとなることはありません。

TEMP_EXTRACT_BINARY オプションを ON に設定し、データ抽出機能を使用してデータがアンロードされた場合は、バイナリ・データをロードするとき、各カラムに **BINARY WITH NULL BYTE** パラメータを使用する必要があります。

- セパレータで区切られた可変長文字列。ターミネータを 16 進の ASCII 文字として指定できます。*delimiter-string* には、印刷可能な文字列と印刷されない文字を表す 8 ビットの 16 進 ASCII コードを自由に組み合わせて、4 文字までの文字列を指定できます。たとえば、次のように指定します。
  - ターミネータとなるタブを表す '¥x09'。
  - null ターミネータ '¥x00' ("C" の文字列と同様に表示されないターミネータ)。
  - ターミネータとなる改行文字の '¥x0a'。特殊文字の組み合わせである '¥n' を使用して改行を表すこともできます。

  注意： delimiter-string には 1 ～ 4 文字を指定できますが、**DELIMITED BY** 句には 1 文字しか指定できません。**BCP** では、10 文字までのデリミタが使用できます。
- ASCII 文字としての DATE または DATETIME 文字列。Sybase IQ がサポートする日付データ型と日付時刻データ型のうち、いずれか一致するフォーマットを使用して、文字列の *input-date-format* または *input-datetime-format* を定義してください。 日付値には **DATE**、日付と時刻の値には **DATETIME** を使用します。

表 9 : 日付と時刻のフォーマット

オプション	意味
yyyy または YYYY yy または YY	年を表す。デフォルトは現在の年。
mm または MM	月を表す。 1 桁の月は、月の先頭に 0 またはブランクを付ける。たとえば、5 月は '05' となる。DATE 値には月を含める。 たとえば、入力した **DATE** 値が '1998' の場合、エラーになる。 たとえば、'03' のように月のみを入力した場合、Sybase IQ はデフォルトの年と日を適用し、'1998-03-01' に変換する。
dd または DD jjj または JJJ	日付を表す。デフォルトの日付は 01。1 桁の日付は、日付の先頭に 0 を付ける。たとえば、最初の日は '01' となる。J または j は年のユリウス日付 (1 ～ 366) を示す。
hh HH	時間を示す。時間は 24 時間表記に基づく。1 桁の時刻は、その前に 0 または空白を付ける。たとえば、午前 1 時は '01' となる。'00' は有効な値で、午前 0 時 (深夜) を表す。
nn	分を示す。1 桁の分は、分の前に 0 を付ける。たとえば、8 分は '08' となる。
ss[.ssssss]	秒数とコンマ以下の秒数を示す。
aa	午前または午後を示す。

オプション	意味
pp	必要な場合にのみ、午後を示す (このオプションはリリース 12.0 より前の Sybase IQ とは互換性がない。以前は、"pp" は "aa" と同義)。
hh	Sybase IQ は、分と秒をゼロとみなす。 たとえば、入力した **DATETIME** 値が '03' の場合、Sybase IQ は '03:00:00.0000' に変換する。
hh:nn または hh:mm	Sybase IQ は秒をゼロとみなす。 たとえば、入力した時刻値が '03:25' の場合、Sybase IQ は '03:25:00.0000' に変換する。

表 10 : DATE と DATETIME フォーマットのオプションの例

入力データ	フォーマット仕様
12/31/98	**DATE** ('MM/DD/YY')
19981231	**DATE** ('YYYYMMDD')
123198140150	**DATETIME** ('MMDDYYhhnnss')
14:01:50 12-31-98	**DATETIME** ('hh:mm:ss MM-DD-YY')
18:27:53	**DATETIME** ('hh:mm:ss')
12/31/98 02:01:50AM	**DATETIME** ('MM/DD/YY hh:mm:ssaa')

Sybase IQ には、一般的な日付、時刻、および日付時刻フォーマット用に、ロードの最適化が組み込まれています。 ロードするデータがこれらのフォーマットのいずれかに当てはまる場合は、正しいフォーマットを使用することでロード時間を大幅に削減できます。 これらのフォーマットの一覧、および日付と日付時刻データをロードする時のパフォーマンスの最適化の詳細については、『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。

date/time フィールドを ASCII 固定幅フィールド (上記参照) として指定し、FILLER(1) オプションを使用してカラム・デリミタをスキップすることもできます。日付と時刻のデータを指定する方法の詳細については、『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」> 「日付と時刻のデータ型」または『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。

column-spec の NULL の部分は、テーブルのカラムにデータをロードするときに、特定の入力値を NULL として処理する方法を示します。NULL として処理される文字には、BLANKS、ZEROS、または定義したその他のリテラルのリストなどがあります。NULL 値を指定するか、またはソース・ファイルから NULL 値を読み込む場合は、ロード先カラムに NULL を格納できる必要があります。

ZEROS は次のように解釈されます。入力データ (ASCII の場合は変換前) がすべてバイナリのゼロ (文字のゼロではない) の場合にかぎり、セルは NULL に設定されます。

- 入力データが文字のゼロの場合は、次のようになります。
  1. NULL (ZEROS) を指定しても、セルに NULL がセットされることはない。
  2. NULL ('0') を指定すると、セルに NULL がセットされる。
- 入力データがバイナリのゼロ (全ビットがクリア) の場合は、次のようになります。
  1. NULL (ZEROS) を指定すると、セルに NULL がセットされる。
  2. NULL ('0') を指定しても、セルに NULL がセットされることはない。

たとえば、**LOAD** 文に `col1 date('yymmdd') null(zeros)` が記述され、日付が 000000 である場合は、000000 を DATE(4) に変換できないことを示すエラーが表示されます。データが 000000 である場合に、**LOAD TABLE** 文で `col1` に NULL 値が挿入されるようにするには、NULL 句を `null('000000')` のように記述するか、データをバイナリのゼロに修正して NULL (ZEROS) を使用する必要があります。

VARCHAR のセルの長さが 0 で、そのセルが NULL でない場合、長さ 0 のセルが作成されます。 その他のデータ型では、セルの長さがゼロの場合、Sybase IQ によって NULL が挿入されます。 これは ANSI 準拠の動作です。ANSI 以外で長さゼロの文字データを扱うには、**NON_ANSI_NULL_VARCHAR** データベース・オプションを設定します。

DEFAULT オプションを使用して、デフォルトのロード・カラム値を指定します。カラムにテーブル・スキーマで定義されたデフォルト値がない場合でも、デフォルト値をカラムにロードできるようになります。この機能によってロード時間をさらに柔軟に変化させることができます。

- **LOAD TABLE** 文で指定されたデフォルト値を使用するには、**LOAD TABLE DEFAULTS** オプションを ON に設定したください。**DEFAULTS** オプションが OFF の場合、指定されたロード・デフォルト値は使用されず、代わりに NULL 値がカラムに挿入されます。
- **LOAD TABLE** コマンドには、**LOAD TABLE** コマンドで指定されたファイルからロードされる必要があるカラムを少なくとも 1 つ以上含めます。そうしないと、エラーが報告され、ロードは実行されません。
- 指定されたロード・デフォルト値は、『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。**LOAD TABLE DEFAULT** オプションは、ロード・デフォルト値として **AUTOINCREMENT**、**IDENTITY**、**GLOBAL AUTOINCREMENT** をサポートしていません。

- **LOAD TABLE DEFAULT***default-value* の文字セットは、データベースと同じである必要があります。
- **LOAD TABLE DEFAULT** 句で指定されたロード・デフォルト値では、デフォルト値の暗号化はサポートされていません。
- 指定されたロード・デフォルト値の評価が原因で発生した制約違反は、テーブルに挿入されたローごとにカウントされます。

load-specification でもう 1 つの重要な部分は、**FILLER** オプションです。このオプションを指定すると、ソース入力ファイル内の指定したフィールドをスキップします。たとえば、ローの末尾の文字や入力ファイルのフィールド全体を、テーブルに追加する必要がない場合があります。**FILLER** では、*column-spec* 定義と同様に、ASCII 固定長バイト、セパレータで区切った可変長文字列、PREFIX バイトを使用するバイナリ・フィールドを指定できます。

filename-string は、文字列としてサーバに渡されます。このため、文字列は他の SQL 文字列と同じフォーマット要件に従います。特に、次の点に注意が必要です。

- Windows システムのディレクトリ・パスを示すには、円記号 (¥) を 2 つの円記号で表してください。したがって、ファイル `c:¥temp¥input.dat` から `Employees` テーブルにデータをロードする文は、次のようになります。

```
LOAD TABLE Employees
FROM 'c:¥¥temp¥¥input.dat' ...
```

- パス名はデータベース・サーバと相対的です。クライアント・アプリケーションとは相対的ではありません。別のコンピュータのデータベース・サーバ上で文を実行している場合、ディレクトリ名が参照されるのは、そのサーバ・マシンのディレクトリであり、クライアント・マシンのディレクトリではありません。

次のリストは、文を句ごとに説明します。

USING— **USING FILE** は、サーバから 1 つまたは複数のファイルをロードします。この句は、**FROM***filename* 句を指定することと同義です。**USING CLIENT FILE** は、クライアントから 1 つまたは複数のファイルのバルク・ロードを行います。クライアント側のファイルの文字セットは、サーバ照合と同じである必要があります。Sybase IQ はファイルをファイル・リスト順に処理します。 各ファイルは処理時に読み込みモードでロックされ、その後で、ロックが解除されます。 クライアント側のバルク・ロードには、ディスク領域、メモリ、またはネットワークをモニタリングするためのデーモンが余分に必要になるなどの、管理作業にかかるオーバヘッドは発生しません。

ラージ・オブジェクトのバルク・ロードを行う場合は、**USING CLIENT FILE** 句をプライマリ・ファイルとセカンダリ・ファイルの両方に適用します (非構造化データ分析オプションを持っている場合は、『Sybase IQ の非構造化データ分析の概要』を参照してください)。

クライアント側のロード中に、**IGNORE CONSTRAINT** ログ・ファイルがクライアント・ホスト上で作成されます。ログ・ファイルの作成中にエラーが発生すると、操作がロールバックされます。

クライアント側バルク・ロードは、Command Sequence プロトコルを使用する Interactive SQL と ODBC/JDBC クライアントによってサポートされています。TDS プロトコルを使用するクライアントによってはサポートされていません。ネットワーク上のデータ・セキュリティには、トランスポート・レイヤ・セキュリティを使用します。クライアント側バルク・ロードを使用できるユーザを制御するには、セキュア機能 **(-sf)** サーバ起動スイッチ、`ALLOW_READ_CLIENT_FILE` データベース・オプション、`READCLIENTFILE` アクセス制御を使用します。

『SQL Anywhere サーバー - SQL の使用法』の「リモートデータとバルクオペレーション」>「データのインポートとエクスポート」>「クライアントコンピューター上のデータへのアクセス」>「クライアント側データセキュリティ」および『SQL Anywhere サーバー - SQL の使用法』の「リモートデータとバルクオペレーション」>「データのインポートとエクスポート」>「クライアントコンピューター上のデータへのアクセス」を参照してください。

注意： これらのリファレンスは SQL Anywhere マニュアルにリンクされています。

LOAD TABLE FROM 句は今後廃止されますが、サーバ上に存在するファイルを指定するのに使用できます。

次に、クライアント・コンピュータ上の `a.inp` ファイルからデータをロードする例を示します。

```
LOAD TABLE t1(c1,c2,filler(30))
USING CLIENT FILE 'c:¥¥client-data¥¥a.inp'
QUOTES OFF ESCAPES OFF
IGNORE CONSTRAINT UNIQUE 0, NULL 0
MESSAGE LOG 'c:¥¥client-data¥¥m.log'
ROW LOG 'c:¥¥client-data¥¥r.log'
ONLY LOG UNIQUE
```

CHECK CONSTRAINTS—このオプションのデフォルトは ON です。**CHECK CONSTRAINTS ON** を指定すると、チェック制約が評価されます。ユーザはそれを無視することもログに記録することもできます。

CHECK CONSTRAINTS OFF に設定すると、Sybase IQ はすべてのチェック制約違反を無視します。 これは、たとえばデータベースの再構築中などに便利です。 テーブルにユーザ定義関数を呼び出すチェック制約があり、その関数がまだ作成されていない場合、このオプションを OFF に設定しなければ再構築が失敗します。

このオプションは、次のオプションに対して相互に排他的です。これらのオプションのいずれかが同じロードに指定されていると、エラーが起こります。

- **IGNORE CONSTRAINT ALL**
- **IGNORE CONSTRAINT CHECK**
- **LOG ALL**
- **LOG CHECK**

DEFAULTS—**DEFAULTS** オプションが ON (デフォルト) でカラムにデフォルト値がある場合、その値が使用されます。**DEFAULTS** オプションが OFF の場合は、カラム・リストに含まれないすべてのカラムに NULL が割り当てられます。

DEFAULTS オプションの設定は、AUTOINCREMENT を含むすべてのカラムの DEFAULT 値に適用されます。

ロードと挿入に伴うカラムの 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。

QUOTES—このパラメータは省略可能であり、デフォルトは ON です。**QUOTES** が ON の場合、**LOAD TABLE** は入力文字列が引用符文字に囲まれていることを想定します。引用符文字はアポストロフィ (一重引用符) または二重引用符のいずれかです。文字列の中で最初に出てくるこのような文字は、文字列の引用符文字として処理されます。文字列データは、対応する引用符文字で終わっている必要があります。

QUOTES ON の場合、カラム・デリミタ文字またはロー・デリミタ文字をカラム値の一部とすることができます。先行する引用符文字と終端の引用符文字は、値の一部とはみなされず、ロードされるデータ値から取り除かれます。

QUOTES ON が指定されている値の中に引用符文字を含めるには、2 つの引用符を使用する必要があります。たとえば、次の行では 3 番目のカラムの値に一重引用符が含まれています。

```
‘123 High Street, Anytown’, ‘(715)398-2354’,’’’’
```

STRIP を ON (デフォルト) に指定すると、後続ブランクを削除してから値を挿入します。後続ブランクが削除されるのは、引用符で囲まれていない文字列だけです。引用符で囲まれた文字列は、後続ブランクを保持します。先行ブランクまたは TAB 文字は、**QUOTES** の設定が ON の場合にのみ削除されます。

データ抽出機能には、引用符を処理するオプション (TEMP_EXTRACT_QUOTES、TEMP_EXTRACT_QUOTES_ALL、TEMP_EXTRACT_QUOTE) があります。IQ テーブルにロードされるデータを抽出する場合に、デフォルトの ASCII 抽出で文字列フィールドにカラム・デリミタまたはロー・デリミタが含まれているときは、抽出に TEMP_EXTRACT_BINARY オプションを使用し、**LOAD TABLE** に **FORMAT binary** オプションと **QUOTES OFF** オプションを使用します。

制限事項：

- **QUOTES ON** オプションは、カラム・デリミタがある ASCII フィールドにのみ適用されます。
- **QUOTES ON** の場合、カラム・デリミタまたはロー・ターミネータの最初の文字に一重引用符または二重引用符は使用できません。
- **QUOTES** オプションは、その設定に関係なく、セカンダリ・ファイルからのバイナリ・ラージ・オブジェクト (BLOB) またはキャラクタ・ラージ・オブジェクト (CLOB) のデータのロードには適用されません。開始引用符または終了引用符は、CLOB データの一部としてロードされます。引用符で囲まれている 2 つの連続した引用符は、**QUOTES ON** オプションを使用すると 2 つの連続した引用符としてロードされます。
- Adaptive Server Enterprise**BCP** は **QUOTES** オプションをサポートしていません。すべてのフィールド・データは、**QUOTES OFF** 設定の場合と同様にコピーされます。**QUOTES ON** が Sybase IQ **LOAD TABLE** 文のデフォルト設定であるため、**BCP** 出力から Sybase IQ テーブルに ASE データをインポートする場合は **QUOTES OFF** を指定する必要があります。

例外：

- **LOAD TABLE** で、引用符で囲まれたフィールドの終了引用符文字の後に空白でない文字がある場合、次のエラーがレポートされてロード操作はロールバックされます。
```
Non-SPACE text found after ending quote character for an enclosed
field. SQLSTATE: QTA14    SQLCODE: -1005014L
```
- **QUOTESON** で、カラム・デリミタの最初の文字として一重引用符または二重引用符が指定された場合、エラーがレポートされてロード操作は失敗します。
```
Single or double quote mark cannot be the 1st character of column
delimiter or row terminator with QUOTES option ON. SQLSTATE:
QCA90    SQLCODE: -1013090L
```

QUOTES オプションの使用例については、『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」>「LOAD TABLE 文を使用したバルク・ロード」を参照してください。

ESCAPES—**ESCAPES** が ON (デフォルト) の場合に、入力フィールドの *column-spec* 定義を省略すると、データベース・サーバは円記号に続く文字を特別な文字として認識、解釈します。 改行文字は ¥n という組み合わせとして、他の文字はタブ文字の ¥x09 のような 16 進の ASCII のコードとしてデータに含まれます。 2 つの円記号 (¥) は 1 つの円記号として解釈されます。 Sybase IQ では、**ESCAPES** は OFF に設定する必要があります。

FORMAT—Sybase IQ は ASCII とバイナリの入力フィールドをサポートします。 このフォーマットは通常、上記の *column-spec* で定義します。 カラムに対してこの定義を省略した場合、デフォルトで Sybase IQ はこのオプションで定義したフォー

マットを使用します。 入力行は、**ascii** (デフォルト) または **binary** フィールド、1 行あたり 1 ロー、カラム・デリミタ文字で区切った値を持つものとされます。

FORMAT BINARY と **BINARY** のカラム指定句を使用した **LOAD TABLE** 文で読み込むことができるデータ・ファイルを生成するために、Sybase IQ で使用されるバイナリ・フォーマットの詳細については、『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」>「BINARY ロード形式」を参照してください。

Sybase IQ は、**LOAD TABLE** コマンドへの入力として BCP 文字ファイルのデータも受け入れます。

- **LOAD TABLE FORMAT BCP** 文を使用して Sybase IQ テーブルにロードされる BCP データは、**-c** オプションを使用して、プラットフォームを問わないファイル・フォーマットでエクスポート (**BCP OUT**) する必要があります。
- **FORMAT BCP** の場合、**LOAD TABLE** 文のデフォルトのカラム・デリミタは <タブ> であり、デフォルトのロー・ターミネータは <改行文字> です。
- **FORMAT BCP** では、ローの最後のカラムはカラム・デリミタではなくロー・ターミネータで終了します。カラム・デリミタがロー・ターミネータの前にある場合、データの一部として扱われます。
- ロード指定で最後のカラムではないカラムのデータは、カラム・デリミタだけを使用して区切ります。最後のカラムではないカラムで、ロー・ターミネータがカラム・デリミタの前にある場合、カラム・データの一部として扱われます。
- カラム・デリミタは **DELIMITED BY** 句で指定できます。**FORMAT BCP** の場合、デリミタの長さは 10 文字以下である必要があります。デリミタの長さが 10 を超える場合はエラーが返されます。
- **FORMAT BCP** では、ロード指定にカラム名、**NULL**、**ENCRYPTED** のみを含めることができます。ロード指定にこれ以外のオプションが指定された場合は、エラーが返されます。
  たとえば、次の **LOAD TABLE** ロード指定は有効です。

  ```
  LOAD TABLE x( c1, c2 null(blanks), c3 ) FROM 'bcp_file.bcp' FORMAT
  BCP ...
  ```

  ```
  LOAD TABLE x( c1 encrypted(bigint,'KEY-ONE','aes'), c2, c3 ) FROM
  'bcp_file.bcp' FORMAT BCP ...
  ```

  LOAD TABLE ENCRYPTED 句の詳細については、『Sybase IQ による高度なセキュリティ』を参照してください。

DELIMITED BY—*column-spec* 定義でカラムのデリミタを省略した場合は、デフォルトのカラム・デリミタ文字であるカンマが使用されます。1 文字の ASCII 文字、または 16 進の文字表現を入力することにより、別のカラム・デリミタを指定できます。**DELIMITED BY** 句の構文は次のようになります。

```
... DELIMITED BY '¥x09' ...
```

デリミタとして改行文字を使用する場合は、特殊文字の組み合わせである '¥n' またはその ASCII 値である '¥x0a' を指定できます。column-spec の *delimiter-string* には 4 文字まで指定できますが、**DELIMITED BY** 句には 1 文字しか指定できません。

STRIP—**STRIP** 句では、引用符で囲まれていない値を挿入する前に、後続ブランクを削除するかどうかを指定します。**LOAD TABLE** コマンドでは、次の **STRIP** キーワードを指定できます。

- STRIP OFF—後続ブランクを削除しません。
- STRIP RTRIM—後続ブランクを削除します。
- STRIP ON—廃止されました。**STRIP RTRIM** に相当します。

STRIP を ON (デフォルト) に指定すると、Sybase IQ は後続ブランクを削除してから値を挿入します。 これは VARCHAR データの場合にのみ効果的です。**STRIP OFF** は後続ブランクを保持します。

後続ブランクが削除されるのは、引用符で囲まれていない文字列だけです。引用符で囲まれた文字列は、後続ブランクを保持します。 ブランクを区別する必要がない場合は、後続スペースをすべて削除する代わりに、**FILLER** オプションを使用して、削除するバイト数をより詳細に指定できます。Sybase IQ では、**STRIP OFF** はさらに効率的で、後続ブランクの処理は ANSI 規格に準拠します (CHAR データでは、常にブランクが埋め込まれます。したがって、この **STRIP** オプションが有効なのは、VARCHAR データの場合だけです)。

STRIP オプションは、可変長の非バイナリ・データだけに適用され、ASCII 固定幅の挿入には適用されません。たとえば、次のようなスキーマを想定します。

```
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・データの後続ブランクは常に削除されます。

WITH CHECKPOINT—このオプションは、Sybase IQ データベースの SQL Anywhere テーブルをロードする場合のみ役立ちます。

この句は、チェックポイントを実行するかどうかを指定するために使用します。デフォルト設定は OFF です。この句を ON に設定すると、文が正常に完了し、ロギングされた後にチェックポイントが発行されます。接続がコミットを実行してから次のチェックポイントを実行するまでにサーバに障害が発生した場合、リカバリを正常に完了するには、テーブルにロードするために使用されたデータ・ファイルが存在する必要があります。ただし、**WITH CHECKPOINT ON** が指定されており、以降でリカバリが必要な場合、リカバリ時にデータ・ファイルは必要ありません。

データベースが破損したため、バックアップを使用して現在のログ・ファイルを適用する必要がある場合は、この句に何が指定されているかに関係なく、データ・ファイルが必要となります。

警告！ データベース・オプション `CONVERSION_ERROR` を OFF に設定すると、不正なデータが、エラーのレポートなしにロードされることがあります。**WITH CHECKPOINT ON** を指定しない場合、データベースのリカバリが必要となったときには、`CONVERSION_ERROR` が ON (デフォルト値) であれば、リカバリが失敗することがあります。テーブルをロードするときには、`CONVERSION_ERROR` を ON に設定し、**WITH CHECKPOINT ON** を指定することをおすすめします。

詳細については、「CONVERSION_ERROR オプション [TSQL]」を参照してください。

Sybase IQ データの自動リカバリの詳細については、『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。

BYTE ORDER—読み込み時のバイトの順序を指定します。このオプションはすべてのバイナリ入力フィールドに該当します。 何も定義されなければ、このオプションは無視されます。Sybase IQ は通常バイナリ・データを、自分が動作しているコンピュータのネイティブ・フォーマットで読み込みます (デフォルトは **NATIVE** です)。または、次のように指定できます。

- **HIGH** マルチバイトの値が上位バイト優先である場合に指定します (Sun、IBM AIX、HP などのビッグ・エンディアン・プラットフォーム用)。
- **LOW** マルチバイトの値が下位バイト優先である場合に指定します (Windows などのリトル・エンディアン・プラットフォーム用)。

LIMIT—テーブルに挿入するローの最大数を指定します。制限なしのデフォルトは 0 です。ロー数の最大値は 2^{31} - 1 (2147483647) です。

NOTIFY—指定した個数のローがテーブルに正常に挿入されるたびに、メッセージが通知されるように指定します。デフォルトは 100,000 ローごとです。このオプションの値は、`NOTIFY_MODULUS` データベース・オプションの値を上書きします。

ON FILE ERROR—入力ファイルが存在しないか、またはファイルを読み込むパーミッションが不正であるために、ファイルを開くことができない場合の Sybase IQ の動作を指定します。 次のいずれかを指定できます。

- **ROLLBACK** トランザクション全体をアボートします (デフォルト)。
- **FINISH** すでに完了している挿入処理を完了して、ロード操作を終了します。
- **CONTINUE** エラーを返すが、該当するファイルのみを省略してロード操作を継続します。

ON FILE ERROR 句は 1 つしか使用できません。

PREVIEW—各カラムの開始位置、名前、データ型など、ロード先テーブルへの入力のレイアウト情報を表示します。Sybase IQ はロード処理の開始時にこの情報を表示します。 ログ・ファイルにログを書き込んでいる場合は、この情報もログに書き込みます。

ROW DELIMITED BY—入力レコードの末尾を指定する文字列を最大 4 バイトで指定します。このオプションが使用できるのは、ロー内の全フィールドが次のいずれかである場合だけです。

- カラム・ターミネータで区切られている場合
- DATE または DATETIME の *column-spec* オプションで定義されている場合
- ASCII 固定長フィールド

入力フィールドにバイナリ・データが格納されている場合は、このオプションは使用できません。このオプションを指定すると、ロー・ターミネータにより、不足したフィールドが NULL に設定されます。すべてのローに同じロー・デリミタがあり、すべてのカラム・デリミタと区別する必要があります。ロー・デリミタ文字列およびフィールド・デリミタ文字列に、互いの初期サブセットを指定することはできません。たとえば、フィールド・デリミタとして "*" を、ロー・デリミタとして "*#" を指定することはできませんが、フィールド・デリミタとして "#" を、ロー・デリミタとして "*#" を指定することはできます。

ローにデリミタがない場合は、Sybase IQ はエラーを返し、ロード・トランザクション全体をロールバックします。 唯一の例外はファイルの最終レコードです。この場合、そのローがロールバックされて、警告メッセージが返されます。Windows では、通常ロー・デリミタは改行文字とそれに続く復帰文字によって指定されます。このオプションまたは **FILLER** では、これらの文字を、上記で説明されている *delimiter-string* として指定しなければならない場合があります。

SKIP—このロードの場合、入力テーブルの開始時にスキップするロー数を定義します。スキップするローの最大数は 2^{31} - 1 (2147483647) です。デフォルトは 0 です。

HEADER SKIP…HEADER DELIMITED BY—**LOAD TABLE** 操作でスキップする、ヘッダー・ローを含むデータ・ファイルの先頭行数を指定します。指定した数の

ローがスキップされるまで、すべての **LOAD TABLE** カラム指定と他のロード・オプションは無視されます。

- スキップする行数には、0 以上の数を指定する必要があります。
- 行は、**HEADER DELIMITED BY** 句で指定した 1 ～ 4 文字のデリミタ文字で区切られます。デフォルトの **HEADER DELIMITED BY** 文字は、'¥n' 文字です。
- **HEADER DELIMITED BY** 文字の最大長は 4 文字です。文字列の長さが 5 文字以上または 1 未満の場合はエラーが返ります。
- ゼロ以外の **HEADER SKIP** 値を指定すると、**HEADER DELIMITED BY** デリミタを含むすべてのデータは、**HEADER SKIP** 句で指定した数のデリミタが検出されるまで無視されます。
- 指定されている数のローがスキップされるまで、すべての **LOAD TABLE** カラム指定と他のロード・オプションは無視されます。指定されている数のローがスキップされると、**LOAD TABLE** カラム指定と他のロード・オプションが残りのデータに適用されます。
- 「ヘッダー」バイトは、データの開始位置でのみ無視されます。**USING** 句で複数のファイルが指定されていると、**HEADER SKIP** は、指定されている数のヘッダー・ローがスキップされるまで、最初のファイルの最初のローから始まるデータだけを無視します。それらのローが後続のファイルにあっても同じです。**LOAD TABLE** は、実際のデータの解析を開始すると、ヘッダーを検索しません。
- **HEADER SKIP** で指定されている数のローをスキップする前に、**LOAD TABLE** がすべての入力データを処理しても、エラーは報告されません。

WORD SKIP—ワード・インデックスの作成時に、指定された制限よりも長いデータがあった場合にロードを続行できます。

ワードが許可されている最大長を超えたためにローがロードされない場合、警告メッセージが `.iqmsg` ファイルに書き込まれます。WORD サイズ制限の違反は、オプションで MESSAGE LOG ファイルに記録され、拒否されたローは、ROW LOG ファイルに記録されます。これらのファイルは **LOAD TABLE** 文で指定されます。

- このオプションが指定されていない場合、**LOAD TABLE** は、指定された制限を超えた最初のワードでエラーをレポートしてロールバックします。
- *number* は、「許可されている最大の長さを超える語はサポートされていません」のエラーを無視する回数を指定します。
- 0 (ゼロ) は制限がないことを意味します。

START ROW ID—Sybase IQ テーブル内の挿入を開始するローのレコード識別番号を指定します。

LOAD TABLE コマンドと **INSERT** コマンドの **START ROW ID** 句は、分割されたテーブルでは使用できません。

ON PARTIAL INPUT ROW—ロード中に部分入力ローがあった場合のアクションを指定します。次のいずれかを指定できます。

- **CONTINUE** 警告を発行し、ロード操作を続行します。これがデフォルトです。
- **ROLLBACK** ロード操作全体をアボートし、エラーを報告します。

```
Partial input record skipped at EOF. SQLSTATE: QDC32    SQLSTATE:
-1000232L
```

IGNORE CONSTRAINT—ロード中に発生した CHECK、UNIQUE、NULL、DATA VALUE、または FOREIGN KEY 整合性制約違反を無視するかどうかを指定します。また、違反をいくつ無視してからロールバックを開始するかを決める違反の最大数を指定します。*constrainttype* の指定に応じて、次のような動作となります。

- CHECK *limit*—*limit* に 0 を指定すると、CHECK 制約違反は無制限に無視されます。CHECK が指定されなければ、最初に CHECK 制約違反が発生した時点で **LOAD** 文がロールバックされます。*limit* が 0 でなければ、CHECK 制約違反が *limit* +1 回発生した時点でロードがロールバックされます。
- UNIQUE *limit*—*limit* に 0 を指定すると、UNIQUE 制約違反は無制限に無視されます。*limit* が 0 でなければ、UNIQUE 制約違反が *limit* +1 回発生した時点でロードがロールバックされます。
- NULL *limit*—*limit* に 0 を指定すると、NULL 制約違反は無制限に無視されます。*limit* が 0 でなければ、NULL 制約違反が *limit* +1 回発生した時点でロードがロールバックされます。
- FOREIGN KEY *limit*—*limit* に 0 を指定すると、FOREIGN KEY 制約違反は無制限に無視されます。*limit* が 0 でなければ、FOREIGN KEY 制約違反が *limit* +1 回発生した時点でロードがロールバックされます。
- DATA VALUE *limit*—データベース・オプションに CONVERSION_ERROR = ON を指定すると、エラーがレポートされて文がロールバックされます。*limit* に 0 を指定した場合、DATA VALUE 制約違反 (データ型変換エラー) は無制限に無視されます。*limit* が 0 でなければ、DATA VALUE 制約違反が *limit* +1 回発生した時点でロードがロールバックされます。
- ALL *limit*—データベース・オプションに `CONVERSION_ERROR = ON` が指定された場合、エラーがレポートされて文がロールバックされます。*limit* に 0 を指定した場合、すべての整合性制約違反は無制限に無視されます。*limit* が 0 以外の場合は、UNIQUE、NULL、DATA VALUE、FOREIGN KEY の整合性制約違反を無視した数の累計が *limit* の値を超えた時点で、ロードはロールバックされます。たとえば、次の **IGNORE CONSTRAINT** オプションを指定したとします。

  ```
  IGNORE CONSTRAINT NULL 50, UNIQUE 100, ALL 200
  ```

  この場合、整合性制約違反の合計数は 200 を超えることができません。同時に、NULL は 50、UNIQUE は 100 を超えてはなりません。これらのいずれかの制限を超えた時点で、**LOAD TABLE** 文はロールバックされます。

注意： 1 つのローに、整合性制約違反が複数ある場合もあります。それぞれの整合性制約違反が、当該の種類の違反としてカウントされ、件数が制限に近づきます。

無視された整合性制約違反をロギングする場合は、**IGNORE CONSTRAINT** オプションの limit は 0 以外の値に設定することを強くおすすめします。ロギングする違反の数が多すぎると、ロードのパフォーマンスに影響します。

IGNORE CONSTRAINT 句に CHECK、UNIQUE、NULL、または FOREIGN KEY を指定しない場合、これらのいずれかのタイプの整合性制約違反が最初に起きた時点でロードがロールバックします。

IGNORE CONSTRAINT 句で DATA VALUE が指定されていない場合は、この種類の整合性制約違反が最初に見つかった時点で、ロードはロールバックされます。ただし、データベース・オプション `CONVERSION_ERROR = OFF` が設定されている場合は除きます。`CONVERSION_ERROR = OFF` が指定されていると、すべての DATA VALUE 制約違反に警告がレポートされ、ロードが継続されます。

ロードが完了すると、整合性制約違反に関する情報メッセージが `.iqmsg` ファイルに記録されます。このメッセージには、ロード中に発生した整合性制約違反の数と、スキップされたローの数が含まれます。

MESSAGE LOG—整合性制約違反に関する情報を記録するログ・ファイルの名前と、ログに記録する違反のタイプを指定します。ロードの開始と完了を示すタイムスタンプは、**MESSAGE LOG** ファイルと **ROW LOG** ファイルの両方に記録されます。**MESSAGE LOG** と **ROW LOG** は両方とも必ず指定します。そうしないと、整合性違反についての情報はロギングされません。

- **ONLY LOG** 句が指定されない場合、整合性制約の違反についての情報はログに記録されません。ロードの開始と完了を示すタイムスタンプのみがロギングされます。
- **ONLY LOG** 句に指定されたすべてのタイプの整合性制約の違反について、またはキーワード WORD が指定されている場合のすべてのワード・インデックス長制限の違反についての情報が、ログに記録されます。
- 制約違反がログに記録される場合、整合性制約違反が起きるたびに、**MESSAGE LOG** ファイルに必ず 1 行の情報が生成されます。

  MESSAGE LOG ファイルの行数 (報告されるエラーの数) が、**IGNORE CONSTRAINT** オプションの制限を超えることがあります。並行して動作する複数のスレッドによってロードが実行されるからです。制約違反の数が指定された制限を超えたことを、複数のスレッドがレポートする場合もあります。
- 制約違反がログに記録される場合、特定のローに対して、(整合性制約違反がいくつ発生しようと) 情報が **ROW LOG** ファイルに必ず 1 行で記録されます。

MESSAGE LOG ファイルに記録された個々のエラーの数と、**ROW LOG** ファイルに記録されたローの数は一致しない場合があります。両者の行数の違いは、**MESSAGE LOG** のところで説明した、ロードの並行処理によるものです。

- **MESSAGE LOG** ファイルと **ROW LOG** ファイルには、ロー・パーティションや名前付きパイプは使用できません。
- **MESSAGE LOG** ファイルまたは **ROW LOG** ファイルがすでに存在している場合は、新しいログ情報がそのファイルに追加されます。
- **MESSAGE LOG** ファイルまたは **ROW LOG** ファイルに無効なファイル名を指定すると、エラーが発生します。
- **MESSAGE LOG** ファイルまたは **ROW LOG** ファイルに同じファイル名を指定すると、エラーが発生します。

IGNORE CONSTRAINT オプションと **MESSAGE LOG** オプションをさまざまに組み合わせて指定すると、次の表に示すロギング・アクションがそれぞれ実行されます。

表 11 : LOAD TABLE のロギング・アクション

IGNORE CONSTRAINT の指定の有無	MESSAGE LOG の指定	アクション
あり	あり	無視されたすべての整合性制約違反が記録される。また、ユーザが limit に指定した違反もロールバック前に記録される。
なし	あり	最初の整合性制約違反がロールバック前に記録される。
あり	なし	何もロギングされない。
なし	なし	何もロギングされない。最初の整合性制約違反でロールバックが実行される。

注意： 無視された整合性制約違反をロギングする場合は、**IGNORE CONSTRAINT** オプションの limit はゼロ以外の値に設定することを強くおすすめします。1 つのローに複数の整合性制約違反がある場合、**MESSAGE LOG** ファイルには、各違反がそれぞれ別個の行として記録されます。ロギングする違反の数が多すぎると、ロードのパフォーマンスに影響します。

LOG DELIMITED BY—**ROW LOG** ファイルのデータ値を区切るセパレータを指定します。デフォルトのセパレータはカンマです。

MESSAGE LOG ファイルと **ROW LOG** ファイルの内容とフォーマットの詳細については、『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」>「LOAD TABLE 文を使用したバルク・ロード」を参照してください。

Sybase IQ では、**FORMAT BCP** が **LOAD TABLE** 句として指定された場合でも、エラー・メッセージを返さなくなりました。さらに、次の状態が確認され、適切なエラー・メッセージが返されます。

- 指定されたロード形式が **ASCII**、**BINARY**、または **BCP** のいずれでもない場合、Sybase IQ はメッセージ「LOAD フォーマットをサポートするのは ASCII、BCP、および BINARY だけです。」を返します。
- **LOAD TABLE** のカラム指定にカラム名、**NULL**、または **ENCRYPTED** 以外のものが含まれている場合、Sybase IQ はエラー・メッセージ「LOAD ... FORMAT BCP に対する無効なロード指定。」を返します。
- **FORMAT BCP** ロードのカラム・デリミタまたはロー・ターミネータのサイズが、10 文字を超えた場合、Sybase IQ はメッセージ「デリミタ '%2' 1 ～ %3 文字の長さである必要があります。」を返します (ここで、%3 には 10 が入ります)。

  FORMAT BCP と **FORMAT ASCII** で発生する可能性があるエラーまたは警告の状態に対応するメッセージは、どちらのフォーマットでも同じです。
- 指定されるロード・デフォルト値が **AUTOINCREMENT**、**IDENTITY**、または **GLOBAL AUTOINCREMENT** の場合、エラー「デフォルト値 %2 は LOAD のデフォルト値として使用できません。%1」が報告されます。
- **LOAD TABLE** 指定に、指定されたファイルからロードする必要があるカラムが含まれていない場合、エラー "The LOAD statement must contain at least one column to be loaded from input file." が報告され、**LOAD TABLE** 文がロールバックします。
- ロード時に **TEXT** インデックスがあるテキスト・ドキュメントで、単語の数が最大数の制限を超えると、エラー "Text document exceeds maximum number of terms. Support up to 4294967295 terms per document." がレポートされます。

標準

- SQL—ISO/ANSI SQL 文法のベンダ拡張。
- Sybase—なし。

パーミッション

LOAD TABLE 文の実行に必要なパーミッションは、次のように、データベース・サーバの **-gl** コマンド・ライン・オプションによって異なります。

- **-gl** オプションが ALL に設定されている場合、テーブルの所有者であるか、DBA 権限または ALTER パーミッションが必要。

- **-gl** オプションが DBA に設定されている場合、DBA 権限が必要。
- **-gl** オプションが NONE に設定されている場合、**LOAD TABLE** は許可されない。

詳細については、『ユーティリティ・ガイド』の「start_iq データベース・サーバ起動ユーティリティ」>「start_iq サーバ・オプション」の **-gl** コマンド・ライン・オプションを参照してください。

また、**LOAD TABLE** ではテーブルに書き込みロックを必要とします。

参照：

- INSERT 文 (258 ページ)
- LOAD_ZEROLENGTH_ASNULL オプション (475 ページ)
- NON_ANSI_NULL_VARCHAR オプション (496 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 12 : 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
VARCHAR、CHARACTER VARYING	幅 (32K – 1) バイト	(32K – 1) バイト	(32K – 1) バイト

LOCK TABLE 文

他の同時トランザクションが指定の時間内にテーブルにアクセスしたり変更したりするのを防ぎます。

構文

```
LOCK TABLE table-list [ WITH HOLD ] IN { SHARE | WRITE
                | EXCLUSIVE } MODE [   WAIT time ]
```

パラメータ

- **table-list：** – [*owner.*] *table-name* [, [*owner.*] *table-name*, …]

  time:

  文字列

例

- **例 1 –** 次の文は、Customers と Employees の各テーブルが 5 分 3 秒以内に使用できるようになる場合、それぞれのテーブルの WRITE ロックを取得します。

```
LOCK TABLE Customers, Employees IN WRITE MODE WAIT
'00:05:03'
```

- **例 2 –** Customers と Employees の各テーブルの WRITE ロックが使用できるようになるか、または割り込みが発生するまで、いつまでも待ちます。

```
LOCK TABLE Customers, Employees IN WRITE MODE WAIT
```

使用法

table-name—テーブルは、ビューではなくベース・テーブルである必要があります。 WRITE モードが有効なのは、IQ ベース・テーブルの場合だけです。**LOCK TABLE** はテーブル・リストにあるすべてのテーブルをロックするか、1 つもロックしないかのどちらかです。 SQL Anywhere テーブルのロックを取得する場合、または SHARE か EXCLUSIVE のどちらかのロックを取得するときに指定できるのは、1 つのテーブルだけです。 *table-name* の解析には、標準の Sybase IQ オブジェクト修飾ルールが使用されます。関連する詳細については、『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 言語の要素」>「識別子」および『システム管理ガイド：第 1 巻』の「データベース・オブジェクトの管理」>「テーブルの管理」>「テーブル作成のガイドライン」>「テーブル・タイプ」を参照してください。

WITH HOLD—この句が指定された場合、そのロックは接続の終了まで継続します。この句が指定されない場合、そのロックは現在のトランザクションがコミットまたはロールバックされたときに解放されます。

SHARE—他の同時トランザクションによるテーブルの変更を防ぎますが、読み込みアクセスは許可します。このモードでは、他のトランザクションが間接的に、または **LOCK TABLE** を使用して明示的に修正中のローをロックしていないかぎり、テーブルのデータを修正できます。

WRITE—他のトランザクションによるテーブルのリストの変更を防ぎます。 接続の一番外側のトランザクションを無条件にコミットします。 トランザクションが使用するスナップショット・バージョンは、**LOCK TABLE IN WRITE MODE** 文ではなく、Sybase IQ が処理する次のコマンドの実行により確立されます。

ジョイン・インデックスに参加している IQ テーブルの WRITE モード・ロックは、次の対象もロックします。

- WRITE モードのジョイン・インデックス階層のトップ・テーブル (X がトップ・テーブル以外の場合)

- 対応するジョイン仮想テーブル (JVT)

WRITE モード・ロックは、トランザクションがコミットされるかロールバックされるとき、または接続が切断されたときに解放されます。

EXCLUSIVE—他のトランザクションによるテーブルへのアクセスを防ぎます。このモードでは、他のトランザクションはテーブルに対してクエリの発行、あらゆる修正、その他のあらゆる操作を実行できません。 テーブル `t` が **LOCK TABLE t IN EXCLUSIVE MODE** によって明示的にロックされている場合、デフォルトのサーバの動作では `t` のロー・ロックを獲得しません。 `SUBSUME_ROW_LOCKS` オプションを OFF に設定すると、この動作を無効にできます。 『SQL Anywhere サーバー - データベース管理』の「データベースの設定」>「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」>「subsume_row_locks オプション」を参照してください。

注意： このリファレンスは SQL Anywhere マニュアルにリンクされています。

コーディネータ上の IQ メイン・ストアのテーブルに対して **LOCK TABLE** 文を実行しても、セカンダリ・サーバ上の接続からこれらのテーブルへのアクセスは影響を受けません。次に例を示します。

コーディネータ接続では、次のコマンドを発行します。

```
LOCK TABLE coord1 WITH HOLD IN EXCLUSIVE MODE
```

コーディネータ上で **sp_iqlocks** を実行すると、テーブル `coord1` に排他 (E) ロックが含まれていることを確認できます。

セカンダリ・サーバ上の接続に対して **sp_iqlocks** を実行した結果には、テーブル `coord1` に対する排他ロックが示されません。この接続上のユーザは、コーディネータ上のテーブル `coord1` に対する更新を表示できます。

コーディネータ上の他の接続からは、`coord1` に対する排他ロックを表示できます。コーディネータ上の別の接続からテーブル `coord1` を選択しようとすると、`User DBA has the row in coord1 locked.` が返されます。

WAIT *time*— Wait の各種オプションは、全種類のロックの最長ブロック時間を指定します。このオプションはロック・モードが WRITE の場合に必須です。time 引数が指定された場合、サーバは指定時間内に使用できる場合にかぎり、指定されたテーブルをロックします。この time 引数は、*hh:nn:ss:sss* の形式で指定できます。日付部分が指定されても、サーバはそれを無視し、その引数をタイムスタンプに変換します。time 引数が指定されない場合は、WRITE ロックが使用できるようになるか、または割り込みが発生するまで、サーバはいつまでも待ちます。

ビューでの **LOCK TABLE** はサポートされていません。ビューをロックしようとすると、コマンドで指定されたモードにかかわらず、共有のスキーマ・ロックが取

得されます。共有のスキーマ・ロックは、他のトランザクションによるテーブル・スキーマの変更を防ぎます。

Transact-SQL (T-SQL) ストアド・プロシージャ言語は、**LOCK TABLE** をサポートしません。たとえば、次の文は `Syntax error near LOCK` を返します。

```
CREATE PROCEDURE tproc()
AS
BEGIN
COMMIT;
LOCK TABLE t1 IN SHARE MODE
INSERT INTO t1 VALUES(30)
END
```

Watcom-SQL ストアド・プロシージャ言語は、**LOCK TABLE** をサポートします。デフォルトでは、コマンドのデリミタはセミコロン (;) です。次に例を示します。

```
CREATE PROCEDURE tproc()
AS
BEGIN
COMMIT;
LOCK TABLE t1 IN SHARE MODE
INSERT INTO t1 VALUES(30)
END
```

詳細については、『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「システム・プロシージャ」>「システム・ストアド・プロシージャ」>「sp_iqlocks プロシージャ」を参照してください。

標準

- SQL—ISO/ANSI SQL 文法のベンダ拡張。
- Sybase—Adaptive Server Enterprise でサポートされています。**WITH HOLD** 句は、Adaptive Server Enterprise ではサポートされていません。Adaptive Server Enterprise には **WAIT** 句がありますが、これは SQL Anywhere ではサポートされていません。

パーミッション

SHARE モードでテーブルをロックするには、SELECT 権限が必要です。

EXCLUSIVE モードでテーブルをロックするには、そのテーブルの所有者であるか DBA 権限が必要です。

参照：

- SELECT 文 (339 ページ)

LOOP 文

文リストの実行を繰り返します。

構文

```
[ statement-label: ]
... [ WHILE
            search-condition ] LOOP
... statement-list
... END
            LOOP [ statement-label ]
```

例

- **例 1 –** プロシージャ中の **WHILE** のループの例

```
...
SET i = 1 ;
WHILE i <= 10 LOOP
  INSERT INTO Counters( number ) VALUES ( i ) ;
  SET i = i + 1 ;
END LOOP ;
...
```

- **例 2 –** プロシージャ中の指定されたラベルのループの例

```
SET i = 1;
lbl:
LOOP
  INSERT
  INTO Counters( number )
  VALUES ( i ) ;
  IF i >= 10 THEN
    LEAVE lbl ;
  END IF ;
  SET i = i + 1 ;
END LOOP lbl
```

使用法

WHILE 文と **LOOP** 文は制御文です。これを使用して、*search-condition* が TRUE と評価するかぎり、SQL 文のリストを繰り返し実行できます。**LEAVE** 文を使用して、**END LOOP** に続く最初の文から実行を再開できます。

終了の *statement-label* を指定する場合は、開始の *statement-label* と一致させる必要があります。

標準

- SQL—ISO/ANSI SQL 準拠。
- Sybase—Adaptive Server Enterprise ではサポートされていません。**WHILE** 文は、Transact-SQL ストアド・プロシージャにループ機能を提供します。

パーミッション

なし

参照：

- FOR 文 (234 ページ)
- LEAVE 文 (271 ページ)
- WHILE 文 [T-SQL] (381 ページ)

MESSAGE 文

メッセージを表示します。

構文

```
MESSAGE expression, …
[ TYPE { INFO | ACTION | WARNING | STATUS } ]
[ TO { CONSOLE
 | CLIENT [ FOR { CONNECTION conn_id  [ IMMEDIATE ] | ALL } ]
 | [ EVENT | SYSTEM ] LOG }
 [ DEBUG ONLY ] ]
```

パラメータ

- **conn_id：** – *integer*

例

- **例 1 –** データベース・サーバのメッセージ・ウィンドウに、文字列 `The current date and time` と、現在の日付と時刻を表示します。

```
CREATE PROCEDURE message_test ()
BEGIN
MESSAGE 'The current date and time: ', Now();
END;
CALL message_test();
```

- **例 2 –** ODBC にコールバックを登録するには、最初にメッセージ・ハンドラを宣言する必要があります。

```
void SQL_CALLBACK my_msgproc(
  void *    sqlca,
```

```
  unsigned char     msg_type,
  long              code,
  unsigned short    len,
  char*              msg )
{ … }
```

SQLSetConnectAttr 関数を呼び出して、宣言されたメッセージ・ハンドラをインストールします。

```
rc = SQLSetConnectAttr(
  dbc,
  ASA_REGISTER_MESSAGE_CALLBACK,
  (SQLPOINTER) &my_msgproc, SQL_IS_POINTER );
```

使用法

MESSAGE 文は、メッセージを表示します。これには、任意の式を指定できます。句は、メッセージが表示される場所を指定します。

MESSAGE ... TO CLIENT 文を発行するプロシージャは、接続に関連付けられている必要があります。

たとえば、次の例ではイベントが接続の外部で発生するため、メッセージ・ボックスは表示されません。

```
CREATE EVENT CheckIdleTime TYPE ServerIdle
WHERE event_condition( 'IdleTime' ) > 100
HANDLER
BEGIN
  MESSAGE 'Idle engine' type warning to client;
END;
```

一方、次の例では、メッセージがサーバ・コンソールに書き込まれます。

```
CREATE EVENT CheckIdleTime TYPE ServerIdle
WHERE event_condition( 'IdleTime' ) > 100
HANDLER
BEGIN
  MESSAGE 'Idle engine' type warning to console;
END;
```

有効な式には、引用符で囲まれた文字列などの定数、または変数、関数が含まれます。ただし、本来ならクエリも式の 1 つですが、**MESSAGE** 文の出力にクエリを含めることはできません。

FOR 句を使用すると、サーバで検出されたイベントを別のアプリケーションに通知できるため、アプリケーションでイベントを明示的にチェックする必要がなくなります。**FOR** 句を使用した場合、受信者は次回 SQL 文を実行したときにメッセージを受け取ります。受信者が現在 SQL 文を実行中であれば、メッセージは文が完了したときに受け取られます。実行中の文がストアド・プロシージャの呼び出しであれば、メッセージは呼び出しが完了する前に受け取られます。

アプリケーションに、メッセージの送信後ただちに通知を受け取る必要があり、接続が SQL 文を実行していない場合は、2 つ目の接続を使用してください。この

接続は 1 つ以上の **WAITFOR DELAY** 文を実行できます。これらの文は、サーバやネットワークのリソースを膨大に消費することはありませんが (ポーリング方式で実行されるため)、アプリケーションはメッセージの送信後ただちに通知を受け取ることができます。

ESQL クライアントと ODBC クライアントは、メッセージ・コールバック関数をとおしてメッセージを受信します。どちらの場合も、関数の登録が必要です。ESQL メッセージ・ハンドラを登録するには、**db_register_callback** 関数を使用します。

ODBC クライアントの場合は、**SQLSetConnectAttr** 関数を使用してコールバック関数を登録してください。

コールバック関数の使用方法については、『SQL Anywhere サーバー - プログラミング』の「データベースツールインターフェイス (DBTools)」>「データベースツールインターフェイスの使い方」>「コールバック関数の使い方」を参照してください。

注意： このリファレンスは SQL Anywhere マニュアルにリンクされています。

TYPE—**TYPE** 句が有効なのは、メッセージがクライアントに送信される場合だけです。クライアント・アプリケーションで、メッセージの処理方法を指定する必要があります。Interactive SQL は、次のロケーションにメッセージを表示します。

- INFO – メッセージ・ウィンドウ (デフォルト)
- ACTION – [OK] ボタンのあるメッセージ・ボックス
- WARNING – [OK] ボタンのあるメッセージ・ボックス
- STATUS – メッセージ・ウィンドウ枠

TO—メッセージのロード先を指定します。

- CONSOLE – メッセージをデータベース・サーバ・ウィンドウに送信します。**CONSOLE** がデフォルトです。
- CLIENT – メッセージをクライアント・アプリケーションに送信します。アプリケーションではメッセージの処理方法を決める必要があります。また、この決定のベースとなる情報として **TYPE** を使用できます。
- LOG – メッセージを -o オプションで指定されたサーバ・ログ・ファイルに送信します。

FOR—**TO CLIENT** のメッセージの場合、このメッセージに関する通知をどの接続が受け取るかを指定します。

- CONNECTION conn_id – メッセージの受信者の接続 ID を指定します。
- IMMEDIATE – 『SQL Anywhere サーバー - SQL リファレンス』の「SQL 文」>「SQL 文」>「MESSAGE 文」を参照してください。

- ALL – すべてのオープンな接続が、メッセージを受信することを指定します。

DEBUG ONLY—ストアド・プロシージャに追加されたデバッグ・メッセージの有効または無効を、 DEBUG_MESSAGES オプションの設定を変えることによって制御できます。 **DEBUG ONLY** が指定されている場合は、DEBUG_MESSAGES オプションが ON に設定されているときにだけ、**MESSAGE** 文が実行されます。

注意： **DEBUG ONLY** を指定したメッセージは、DEBUG_MESSAGES オプションが OFF に設定されているときコストがかからないため、これらの文は通常、実稼働システムのストアド・プロシージャに残されます。ただし、頻繁に実行されるストアド・プロシージャではあまり使用しないでください。不用意に使用すると、パフォーマンスが多少低下する可能性があります。

標準

- 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。Transact-SQL の **PRINT** 文は、同様な機能を提供し、SQL Anywhere で使用できます。

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。

FOR 句が指定された **MESSAGE** 文を実行するには、DBA 権限が必要です。

参照：

- CREATE PROCEDURE 文 (137 ページ)
- WAITFOR 文 (378 ページ)
- DEBUG_MESSAGES オプション (433 ページ)

OPEN 文 [ESQL] [SP]

事前に宣言したカーソルをオープンし、データベースからの情報にアクセスします。

構文

```
OPEN cursor-name
... [ USING [ DESCRIPTOR { sqlda-name | host-variable [, …] } ] ]
... [ WITH
            HOLD ]
```

パラメータ

- **cursor-name：** – 識別子またはホスト変数
- **sqlda-name：** – 識別子

例

- **例 1 –** 次の例は、Embedded SQL での **OPEN** の使用方法を示します。

```
EXEC SQL OPEN employee_cursor;
```

および

```
EXEC SQL PREPARE emp_stat FROM 'SELECT EmployeeID, Surname FROM
Employees WHERE name like ?'; EXEC SQL DECLARE employee_cursor
CURSOR FOR emp_stat; EXEC SQL OPEN employee_cursor USING :pattern;
```

- **例 2 –** 次は、プロシージャからの例です。

```
BEGIN
DECLARE cur_employee CURSOR FOR
  SELECT Surname
  FROM Employees ;
DECLARE name CHAR(40) ;
OPEN cur_employee;
LOOP
FETCH NEXT cur_employee into name ;
   ...
END LOOP
CLOSE cur_employee;
END
```

使用法

デフォルトで、すべてのカーソルは現在のトランザクションの最後 (**COMMIT** または **ROLLBACK**) で自動的にクローズします。オプションの **WITH HOLD** 句は、次に実行されるトランザクションのためにカーソルを開いたままにします。カーソルは現在の接続が終了するまで、または明示的な **CLOSE** 文が実行されるまでオープンしたままです。接続が終了すると、カーソルは自動的にクローズします。

カーソルは最初のローの前に位置づけられます。詳細については、『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。

FOR READ ONLY として宣言されたカーソルからは、最初に **FETCH** を実行したときのテーブルのバージョンではなく、カーソルをオープンしたときにカーソルの宣言に使用されるテーブルのバージョンが表示されます。

USING DESCRIPTOR sqlda-name、**host-variable**、**BLOCK n** のフォーマットを使用できるのは、Embedded SQL だけです。

カーソル名が識別子または文字列によって指定される場合、C プログラムでは **OPEN** の前に対応する **DECLARE CURSOR** 文を置く必要があります。ホスト変数に

よってカーソル名を指定する場合、**DECLARE CURSOR** 文を **OPEN** 文の前に実行する必要があります。

オプションの **USING** 句には、カーソルが宣言されている **SELECT** 文のプレースホルダ・バインド変数にバインドされるホスト変数を指定します。

OPEN 文が正常に実行された後で、SQLCA (SQLIOESTIMATE) の *sqlerrd[3]* フィールドに、クエリのすべてのローをフェッチするのに必要な入出力操作の数の推定値が格納されます。同様に、SQLCA (SQLCOUNT) の *sqlerrd[2]* フィールドに、カーソル内にある実際のロー数 (0 以上の値)、またはその推定値 (絶対値が推定値である負の数) が入ります。データベース・サーバが、ローをカウントしなくてもこの値を計算できる場合、*sqlerrd[2]* フィールドは実際のローの数になります。

標準

- SQL—ISO/ANSI SQL 準拠。
- Sybase—簡単な **OPEN***cursor-name* 構文は、Adaptive Server Enterprise でサポートされています。 それ以外の句はすべて Adaptive Server Enterprise ストアド・プロシージャでサポートされていません。 Open Client/Open Server は、**USING** 記述子またはホスト変数構文をサポートします。

パーミッション

- **SELECT** 文のすべてのテーブルに SELECT パーミッション、または **CALL** 文のプロシージャに EXECUTE パーミッションが必要です。
- カーソルが **CALL** 文にあるとき、**OPEN** は最初の結果セット (**INTO** 句がない **SELECT** 文) が見つかるまでプロシージャを実行します。プロシージャが完了しても結果セットが見つからない場合は、SQLSTATE_PROCEDURE_COMPLETE 警告が設定されます。

参照：

- CLOSE 文 [ESQL] [SP] (71 ページ)
- DECLARE CURSOR 文 [ESQL] [SP] (191 ページ)
- FETCH 文 [ESQL] [SP] (230 ページ)
- PREPARE 文 [ESQL] (309 ページ)
- RESUME 文 (328 ページ)

OUTPUT 文 [Interactive SQL]

現在のクエリ結果をファイルへ書き込みます。

構文

```
OUTPUT TO filename
[ APPEND ] [ VERBOSE ]
[ FORMAT output-format ]
[ ESCAPE CHARACTER character ]
[ DELIMITED BY string ]
[ QUOTE string [ ALL ] ]
[ COLUMN WIDTHS ( integer, … ) ]
[ HEXADECIMAL { ON | OFF | ASIS } ]
[ ENCODING encoding ]
[ WITH COLUMN NAMES  ]
```

パラメータ

- **output-format：** – **TEXT** | **FIXED** | **HTML** | **SQL** | **XML**
- **encoding：** – *string* または *identifier*

例

- **例 1** – Employees テーブルの内容をテキスト・ファイルに出力します。

  ```
  SELECT * FROM Employees;
  OUTPUT TO employees.txt FORMAT TEXT
  ```

- **例 2** – Employees テーブルの内容を既存のファイルの末尾に出力し、クエリに関するメッセージも同じファイルに追加します。

  ```
  SELECT * FROM Employees;
  OUTPUT TO employees.txt APPEND VERBOSE
  ```

- **例 3** – 改行文字を含む値をエクスポートします。 改行文字は数値 10 に相当し、SQL 文では文字列 "¥x0a" と表現されます。

  HEXADECIMAL ON を指定して、次の文を実行します。

  ```
  SELECT 'line1¥x0aline2'; OUTPUT TO file.txt HEXADECIMAL ON
  ```

  次のテキストを含む 1 行のファイルが取得されます。

  ```
  line10x0aline2
  ```

  HEXADECIMAL OFF を指定して同じ文を実行すると、次の結果が得られます。

  ```
  line1¥x0aline2
  ```

HEXADECIMAL を **ASIS** に設定すると、次の 2 行から成るファイルが取得されます。

```
'line1
line2'
```

ASIS を使用すると 2 行生成されるのは、間に含まれる改行文字が、2 桁の 16 進表現に変換されることも、プレフィクスもなく、エクスポートされるためです。

使用法

OUTPUT 文は、現在のクエリが取り出した情報をファイルにコピーします。

出力フォーマットは、オプションの **FORMAT** 句を使用して指定できます。 **FORMAT** 句が指定されていない場合、Interactive SQL の OUTPUT_FORMAT オプションの設定が使用されます。

現在のクエリは、[結果] ウィンドウ枠の [結果] タブに表示される情報を生成した **SELECT** 文または **LOAD TABLE** 文です。 現在のクエリがない場合、**OUTPUT** 文はエラーをレポートします。

注意： **OUTPUT** 文は、特にクエリやレポートの結果を別のアプリケーションで利用するとき便利ですが、バルク・オペレーションには向いていません。 大量のデータを移動する場合は、**SELECT** 文の **ASCII** と **BINARY** のデータ抽出機能を使用してください。この抽出機能を使用すると、格段に高いパフォーマンスで大量のデータを移動し、出力ファイルを作成してロードに使用できます。

APPEND — このオプションのキーワードを使用すると、クエリの結果は、既存の出力ファイルに記述されている内容を上書きするのではなく、ファイルの末尾に追加されます。 **APPEND** 句を使用しない場合、**OUTPUT** 文はデフォルトで出力ファイルの内容を上書きします。 出力フォーマットが TEXT、FIXED、または SQL の場合に、**APPEND** キーワードが有効です。

VERBOSE — オプションの **VERBOSE** キーワードを指定すると、クエリに関するエラー・メッセージ、データの選択に使用した SQL 文、データそのものが出力ファイルに書き込まれます。 **VERBOSE** を省略すると (デフォルト)、ファイルにはデータのみが書き込まれます。 出力フォーマットが TEXT、FIXED、または SQL の場合に、**VERBOSE** キーワードが有効です。

FORMAT — 次の出力フォーマットを指定できます。

- TEXT — TEXT フォーマットでファイルに出力され、1 行につき 1 ローが書き込まれます。すべての値がカンマで区切られ、文字列はアポストロフィ (一重引用符) で囲まれます。 デリミタと引用符文字列は、**DELIMITED BY** 句と **QUOTE** 句を使用して変更できます。**QUOTE** 句に **ALL** が指定されている場合は、文字

列だけでなく、すべての値が引用符で囲まれます。 TEXT はデフォルトの出力フォーマットです。
この他に 3 つの特別なシーケンスが認識されます。2 文字の文字列 ¥n は改行文字を表し、¥¥ は単一の円記号を示し、¥xDD のようなシーケンスは 16 進コード DD の付いた文字を示します。
文字列の値を返す Java メソッドをエクスポートする場合は、**HEXADECIMAL OFF** 句を使用する必要があります。

- FIXED — それぞれのカラムが固定幅を持つ固定フォーマットで出力されます。 それぞれのカラムの幅は **COLUMN WIDTHS** 句を使って指定できます。 このフォーマットでは、カラム見出しは出力されません。
  COLUMN WIDTHS を省略した場合、各カラムの幅はカラムのデータ型から計算され、そのデータ型の値を保持するのに十分な大きさになります。 ただし、`LONG VARCHAR` と `LONG BINARY` のデータだけは、デフォルトで 32KB になります。
- HTML — Hyper Text Markup Language フォーマットで出力されます。
- SQL — テーブル内の情報を再作成するのに必要な Interactive SQL **INPUT** 文が出力されます。

  注意： Sybase IQ は **INPUT** 文をサポートしません。この出力を使ってデータをロードバックするには、INPUT 文を有効な **LOAD TABLE** (または **INSERT**) 文に編集する必要があります。
- XML — UTF-8 でエンコードされ、DTD が埋め込まれた XML ファイルに出力されます。 バイナリ値は CDATA ブロック内にエンコードされ、バイナリ・データが 2 桁の 16 進文字列で表されます。**LOAD TABLE** 文のファイル・フォーマットに XML を指定することはできません。

ESCAPE CHARACTER — 16 進のコードと記号として格納されている文字に使用するデフォルトのエスケープ文字は、円記号 (¥) です。たとえば、¥x0A は改行文字です。

このデフォルトのエスケープ文字は、**ESCAPE CHARACTER** 句を使用して変更できます。たとえば、感嘆符 (!) をエスケープ文字として使用するには、次のように入力します。

```
... ESCAPE CHARACTER '!'
```

DELIMITED BY — **DELIMITED BY** 句を使用できるのは、TEXT 出力フォーマットの場合のみです。 デリミタ文字列 (デフォルトはカンマ) は、カラムの間に置かれます。

QUOTE — **QUOTE** 句を使用できるのは、TEXT 出力フォーマットの場合のみです。 文字列の値の前後を引用符文字列で囲みます。 デフォルトは一重引用符です。
ALL を **QUOTE** 句に指定すると、引用符文字列は文字列の前後だけでなく、すべての値の前後に置かれます。

COLUMN WIDTHS — **COLUMN WIDTHS** 句を使用して、**FIXED** フォーマット出力のカラム幅を指定します。

HEXADECIMAL — **HEXADECIMAL** 句では、TEXT フォーマットの場合にのみ、バイナリ・データをアンロードする方法を指定します。 この句を **ON** に設定すると、バイナリ・データは **0xabcd** フォーマットでアンロードされます。**OFF** に設定すると、バイナリ・データはアンロード時に (**¥xab¥xcd** のように) エスケープされます。**ASIS** に設定すると、値が制御文字を含んでいても、そのままの値が (エスケープなしで) 書き込まれます。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。

ENCODING — ファイルの書き込み時に使用するコード化を指定します。 **ENCODING** 句を使用できるのは TEXT フォーマットだけです。

encoding を指定しない場合、Interactive SQL はファイルの書き込みに使用するコード・ページを次のリストのように決定します。このリストで先に示すコード・ページの値は、後に示す値よりも優先されます。

- **DEFAULT_ISQL_ENCODING** オプションで指定されたコード・ページ (このオプションが設定されている場合)
- Interactive SQL を実行しているコンピュータのデフォルトのコード・ページ

関連する動作

- Interactive SQL では、現在のクエリの結果だけが [結果] タブに表示されます。前回のクエリの結果は、現在のクエリの結果にすべて置き換えられます。

標準

- SQL — ISO/ANSI SQL 文法のベンダ拡張。
- Sybase — なし。

パーミッション

なし

参照：

- SELECT 文 (339 ページ)
- DEFAULT_ISQL_ENCODING オプション [Interactive SQL] (437 ページ)

PARAMETERS 文 [Interactive SQL]

Interactive SQL (**dbisql**) コマンド・ファイルにパラメータを指定します。

構文

```
PARAMETERS parameter1, parameter2, …
```

例

- **例 1 –** 次の **dbisql** コマンド・ファイルは、2 つのパラメータを使用します。

```
PARAMETERS department_id, file ;
SELECT Surname
FROM Employees
WHERE DepartmentID = {department_id}
>#{file}.dat;
```

使用法

PARAMETERS は、指定したコマンド・ファイルに対するパラメータ数を指定します。また、これらのパラメータに名前を与え、そのコマンド・ファイル内で後から参照できるようにします。

パラメータは、指定したパラメータを置き換えるコマンド・ファイルに、そのパラメータを次のように入れることで参照されます。

```
{parameter1}
```

大カッコとパラメータ名の間には、スペースを入れないでください。

コマンド・ファイルを呼び出すときに、すべての必要なパラメータを指定しないと、**dbisql** は不足しているパラメータの値を要求するメッセージを表示します。

標準

- SQL—ISO/ANSI SQL 文法のベンダ拡張。
- Sybase—なし。

パーミッション

なし

参照：

- READ 文 [Interactive SQL] (316 ページ)

PREPARE 文 [ESQL]

後から実行する文、またはカーソルに使用する文を準備します。

構文

```
PREPARE statement-name
FROM statement [ FOR { READ ONLY | UPDATE [ OF column-name-list ] } ]
... [ DESCRIBE describe-type INTO [ [ SQL ] DESCRIPTOR ] descriptor ]
... [ WITH EXECUTE ]
```

パラメータ

- **statement-name：** – 識別子またはホスト変数
- **statement：** – 文字列またはホスト変数
- **describe-type：** – { **ALL** | **BIND VARIABLES** | **INPUT** | **OUTPUT** | **SELECT LIST** } ... { **LONG NAMES** [[**OWNER.**]**TABLE.**]**COLUMN**] | **WITH VARIABLE RESULT** }

例

- **例 1 –** 簡単なクエリを準備します。

```
EXEC SQL PREPARE employee_statement FROM
'SELECT Surname FROM Employees';
```

使用法

PREPARE 文は、*statement* から SQL 文を準備し、準備した文を *statement-name* と関連付けます。

この statement-name を参照し、文を実行します。または、文が **SELECT** 文の場合、カーソルをオープンします。*statement-name* は、`sqlca.h` ヘッダー・ファイルの中で定義される `a_sql_statement_number` 型のホスト変数であり、自動的にインクルードされます。識別子を *statement-name* で使用する場合、*statement-name* を使用して各モジュールに準備できるのは、それぞれ 1 文だけです。

ホスト変数を *statement-name* に使用する場合は、`short int` 型とする必要があります。`sqlca.h` 内には、`a_sql_statement_number` というこの型の型定義があります。この型は SQL プリプロセッサで認識されるので、**DECLARE** セクションの中で使用できます。**PREPARE** 文の実行中に、データベースによってホスト変数が埋められるので、プログラマが初期化する必要はありません。

FOR UPDATE | FOR READ ONLY 文がカーソルによって使用されている場合の、カーソルの更新可能性を定義します。**FOR READ ONLY** カーソルは、**UPDATE** (位置付け) 操作または **DELETE** (位置付け) 操作には使用できません。**FOR READ ONLY** がデ

フォルトです。**FOR UPDATE** を指定したカーソル要求への応答では、Sybase IQ は value-sensitive カーソルまたは sensitive カーソルを提供します。 insensitive カーソルと asensitive カーソルは更新できません。

DESCRIBE INTO DESCRIPTOR 句を使用すると、準備文は指定した記述子に記述されます。記述タイプとして、**DESCRIBE** 文で許容されるいずれかのものを使用できます。

WITH EXECUTE 句を使用すると、**CALL** 文または **SELECT** 文ではなく、ホスト変数が含まれていない場合にのみ、文が実行されます。正常に実行された後、文はその場で削除されます。文の **PREPARE** と **DESCRIBE** (存在する場合) が正常に終了したのに、文が実行できない場合、警告 SQLCODE 111、SQLSTATE 01W08 が設定され、文は削除されません。

結果セット **DESCRIBE INTO DESCRIPTOR** 句と **WITH EXECUTE** 句を使用すると、クライアントとサーバとの間で必要な通信が少なくなるので、パフォーマンスが改善できる場合があります。

WITH VARIABLE RESULT 句は、異なる数または型のカラムを含む複数の結果セットを持つプロシージャの記述に使用します。

WITH VARIABLE RESULT を使用する場合、記述後にデータベース・サーバによって SQLCOUNT 値に次のいずれかの値が設定されます。

- 0—結果セットが変わる可能性があります。各 **OPEN** 文の後でプロシージャ・コールを再度記述してください。
- 1—結果セットは固定です。再度記述する必要はありません。

次は、PREPARED の対象になる文のリストです。

- **ALTER**
- **CALL**
- **COMMENT ON**
- **CREATE**
- **DELETE**
- **DROP**
- **GRANT**
- **INSERT**
- **REVOKE**
- **SELECT**
- **SET OPTION**

互換性を保つために、**COMMIT** 文、**PREPARE TO COMMIT** 文、**ROLLBACK** 文の準備はサポートされています。ただし、静的 Embedded SQL を使用して、すべてのトランザクション管理操作を行うことをおすすめします。特定のアプリケーション

環境では、これが必要とされるからです。また、他の Embedded SQL システムは、動的トランザクション管理操作をサポートしません。

注意： 文を使用した後は、確実に **DROP** してください。DROP しないと、文が使用したメモリを再使用できません。

関連する動作

- 以前に同じ名前で準備した文が失われます。

標準

- SQL—ISO/ANSI SQL 準拠。
- 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

参照：

- DECLARE CURSOR 文 [ESQL] [SP] (191 ページ)
- DESCRIBE 文 [ESQL] (205 ページ)
- DROP 文 (209 ページ)
- EXECUTE 文 [ESQL] (223 ページ)
- OPEN 文 [ESQL] [SP] (301 ページ)

PRINT 文 [T-SQL]

データベース・サーバのメッセージ・ウィンドウにメッセージを表示します。

構文

```
PRINT format-string [, arg-list]
```

例

- **例 1 –** サーバのメッセージ・ウィンドウにメッセージを表示します。

  ```
  CREATE PROCEDURE print_test
  AS
  PRINT 'Procedure called successfully'
  ```

  この文は、クライアントに "Procedure called successfully" という文字列を返します。

```
EXECUTE print_test
```

- **例 2 – PRINT** 文でのプレースホルダの使用法を示します。これらの文をプロシージャ内で実行します。

```
DECLARE @var1 INT, @var2 INT
SELECT @var1 = 3, @var2 = 5
PRINT 'Variable 1 = %1!, Variable 2 = %2!', @var1, @var2
```

- **例 3 – RAISERROR** を使用して接続を禁止します。

```
CREATE procedure DBA.login_check()
begin
  // Allow a maximum of 3 concurrent connections
  IF( db_property('ConnCount') > 3 ) then
  raiserror 28000
    'User %1! is not allowed to connect -- there are
      already %2! users logged on',
      current user,
      cast(db_property('ConnCount') as int)-1;
    ELSE
  call sp_login_environment;
    end if;
end
go
grant execute on DBA.login_check to PUBLIC
go
set option PUBLIC.Login_procedure='DBA.login_check'
go
```

接続を禁止するための別の方法については、「LOGIN_PROCEDURE オプション」または『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「システム・プロシージャ」>「システム・ストアド・プロシージャ」>「sp_iqmodifylogin プロシージャ」を参照してください。

使用法

PRINT 文は、Open Client アプリケーションまたは JDBC アプリケーションから接続している場合、クライアント・ウィンドウにメッセージを返します。Embedded SQL または ODBC アプリケーションから接続している場合、このメッセージはデータベース・サーバのウィンドウに表示されます。

フォーマット文字列は、オプション引数リスト (arg-list) にある引数のプレースホルダを含むことができます。プレースホルダの形式は *%nn!* で、*nn* は 1 ～ 20 の整数です。

標準

- SQL—ISO/ANSI SQL 文法の Transact-SQL 拡張。
- Sybase—Adaptive Server Enterprise でサポートされています。

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。

参照：

- MESSAGE 文 (298 ページ)
- LOGIN_PROCEDURE オプション (478 ページ)

PUT 文 [ESQL]

指定されたカーソルにローを挿入します。

構文

```
PUT cursor-name [ USING DESCRIPTOR sqlda-name
| FROM hostvar-list ] [ INTO { DESCRIPTOR into-sqlda-name
| into-hostvar-list } ] [ ARRAY :nnn ]
```

パラメータ

- **cursor-name：** – *識別子* または *hostvar*
- **sqlda-name：** – *識別子*
- **hostvar-list：** – インジケータ変数を含む場合がある

例

- **例 1 –** 次の例は、Embedded SQL での **PUT** の使用方法を示します。

```
EXEC SQL PUT cur_employee FROM :EmployeeID, :Surname;
```

使用法

指定されたカーソルにローを挿入します。カラムの値は、最初の SQLDA から取得されます。または、**INSERT** 文に指定したカラム (INSERT カーソル用) または select リストのカラム (SELECT カーソル用) と 1 対 1 で対応するホスト変数リストからカラムの値を取得します。

PUT 文は、**INSERT** 文または **SELECT** 文のカーソルでのみ使用でき、これらの文では **FROM** 句に指定した 1 つのテーブルを参照するか、1 つのベース・テーブルで構成される更新可能ビューを参照します。

SQLDA 内の **sqldata** ポインタが NULL ポインタである場合、そのカラムには値を指定しません。カラムに DEFAULT VALUE が関連付けられていればその値が使用されますが、関連付けられていなければ NULL 値が使用されます。

2 番目の SQLDA またはホスト変数のリストには、**PUT** 文の結果が格納されています。

オプションの **ARRAY** 句を使用すると、ワイド挿入が実行できます。このワイド挿入では複数のローが一度に挿入されるため、パフォーマンスが向上します。値 *nnn* は、挿入するローの数です。SQLDA には、*nnn** (*columns per row*) 変数が含まれている必要があります。最初のローは SQLDA の変数 0 から (*columns per row*) - 1 に格納され、以後のローも同様です。

注意： スクロール (値依存) カーソルでは、挿入されたローは、新しいローが **WHERE** 句の条件を満たし、キーセット・カーソルの移植が完了していない場合に、表示されます。動的なカーソルの場合、挿入されたローは、**WHERE** 句の条件を満たせば表示されます。Insensitive なカーソルは更新できません。

データベースへの LONG VARCHAR 値または LONG BINARY 値の挿入については、「SET 文 [ESQL]」 を参照してください。

関連する動作

- 値依存 (キーセット駆動型) カーソルにローを挿入する場合、挿入されたローは、たとえクエリの **WHERE** 句の条件を満たさなくても、または **ORDER BY** 句が正常な処理としてローを結果セットの別の場所に置く可能性がある場合でも、結果セットの末尾に表示されます。 詳細については、『SQL Anywhere サーバー - プログラミング』の「アプリケーションでの SQL の使用」>「SQL Anywhere のカーソル」>「value-sensitive カーソル」を参照してください。

  注意： このリファレンスは SQL Anywhere マニュアルにリンクされています。

標準

- SQL—ISO/ANSI SQL 準拠。
- Sybase—Open Client/Open Server でサポートされています。

パーミッション

INSERT パーミッションが必要です。

参照：

- DELETE (位置付け) 文 [ESQL] [SP] (203 ページ)
- INSERT 文 (258 ページ)
- SET 文 [ESQL] (350 ページ)
- UPDATE 文 (372 ページ)
- UPDATE (位置付け) 文 [ESQL] [SP] (376 ページ)

RAISERROR 文 [T-SQL]

エラー信号を送り、クライアントにメッセージを送信します。

構文

RAISERROR *error-number* [*format-string*] [, *arg-list*]

例

- **例 1 –** 次の文はエラー 99999 を発行します。この番号は、ユーザ定義エラーの範囲に含まれており、クライアントにメッセージが送信されます。

```
RAISERROR 99999 'Invalid entry for this
column: %1!', @val
```

使用法

RAISERROR 文を使用することにより、ユーザ定義エラーを通知し、クライアントにメッセージを送信することができます。

error-number は 17000 よりも大きい 5 桁の整数です。このエラー番号は、グローバル変数 *@@error* に格納されます。

error-number パラメータと *format-string* パラメータの間には、カンマはありません。カンマの後にある最初の項目は、引数リストの最初の項目として解釈されます。

format-string を指定しないか、空にすると、エラー番号を使用してシステム・テーブルのエラー・メッセージが検索されます。Adaptive Server Enterprise は、17000 ～ 19999 のメッセージを SYSMESSAGES テーブルから取得します。 Sybase IQ では、このテーブルは空のビューであるため、この範囲のエラーに対してはフォーマット文字列を指定してください。 エラー番号が 20000 以上のメッセージは、SYS.SYSUSERMESSAGES テーブルから取得されます。

format-string の最大長は 255 バイトです。 これは、Adaptive Server Enterprise でも同様です。

SQL Server または Adaptive Server Enterprise の **RAISERROR** 文でサポートされる拡張値は、Sybase IQ ではサポートされていません。

フォーマット文字列は、オプション引数リスト (arg-list) にある引数のプレースホルダを含むことができます。プレースホルダの形式は *%nn!* で、*nn* は 1 ～ 20 の整数です。

中間 **RAISERROR** のステータスとコード情報は、プロシージャが終了すると失われます。結果が返されるときに **RAISERROR** で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。アプリケーションでは、別の実行ポイントでグローバル変数 @@error を検査して、中間 **RAISERROR** ステータスを問い合わせることができます。

標準

- SQL—ISO/ANSI SQL 文法の Transact-SQL 拡張。
- Sybase—Adaptive Server Enterprise でサポートされています。

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。

参照：

- CONTINUE_AFTER_RAISERROR オプション [TSQL] (418 ページ)
- ON_TSQL_ERROR オプション [TSQL] (500 ページ)

READ 文 [Interactive SQL]

Interactive SQL (**dbisql**) 文をファイルから読み込みます。

構文

```
READ filename [ parameter ] …
```

例

- **例 1 –**

```
READ status.rpt '160'
```

- **例 2 –**

```
READ birthday.sql [>= '1988-1-1'] [<= '1988-1-30']
```

- **例 3 –**

```
[test1.sql]
PARAMETERS par1, par2;

BEGIN
DECLARE v_par1 int;
DECLARE v_par2 varchar(200);

SET v_par1 = {par1};
SET v_par2 = {par2};

MESSAGE STRING('PAR1 Value: ', v_par1 ) TO CLIENT;
MESSAGE STRING('PAR2 Value: ', v_par2 ) TO CLIENT;
```

```
END;
```

```
(USR1)> READ test1.sql 123 '041028'
PAR1 Value: 123
PAR2 Value: 041028
```

注意： 2つ目のパラメータ値の 041028 は引用符で囲みます。これは、*v_par2*が文字データ型として宣言されているためです。

使用法

READ 文は、指定したファイルから **dbisql** 文のシーケンスを読み込みます。このファイルには、別の **READ** 文を含む有効な **dbisql** 文 (どのような深さまでもネスト可能) を含むことができます。

コマンド・ファイルを探すために、**dbisql** は、最初に現在のディレクトリを検索し、次に環境変数 SQLPATH 内で指定されているディレクトリを検索し、次に環境変数 PATH 内で指定されているディレクトリを検索します。 指定したファイルにファイル拡張子がない場合、**dbisql** は各ディレクトリで拡張子 `.sql` を持つ同じファイル名を検索します。

パラメータは、コマンド・ファイル名の後にリストできます。これらのパラメータは、文ファイルの先頭の **PARAMETERS** 文で指定したパラメータに対応します (「PARAMETERS 文」を参照してください)。**dbisql** は、ソース・ファイルに次の表現がある場合、対応するパラメータを置き換えます。

```
{ parameter-name }
```

ここで、*parameter-name* は、適切なパラメータの名前です。

コマンド・ファイルに渡すパラメータは、識別子、数、引用符付きの識別子、または文字列です。パラメータの周囲を引用符で囲むときは、入れ替えるテキストの中に引用符を入れてください。識別子、数、または文字列 (スペースまたはタブを含む) ではないパラメータは、角カッコ (**[ ]**) で囲みます。これにより、コマンド・ファイル内で任意のテキストを置き換えることができます。

SQL の文字リテラル (文字データなど) をパラメータとして **READ** 文に渡す場合は、引用符で囲んでください。

十分なパラメータがコマンド・ファイルに渡されない場合、**dbisql** は不足しているパラメータの値を要求するメッセージを表示します。

READ 文は **ENCODING** 句もサポートします。この句を使用すると、ファイルの読み込みに使用するコード化を指定できます。詳細については、『SQL Anywhere サーバ ― SQL リファレンス』の「SQL 文」>「SQL 文」>「READ 文 [Interactive SQL]」を参照してください。

注意： このリファレンスは SQL Anywhere マニュアルにリンクされています。

標準

- SQL—ISO/ANSI SQL 文法のベンダ拡張。
- Sybase—なし。

パーミッション

なし

参照：

- DEFAULT_ISQL_ENCODING オプション [Interactive SQL] (437 ページ)
- PARAMETERS 文 [Interactive SQL] (308 ページ)

RELEASE SAVEPOINT 文

現在のトランザクション内でセーブポイントを解放します。

構文

```
RELEASE
SAVEPOINT [ savepoint-name ]
```

使用法

savepoint-name は、現在のトランザクションの **SAVEPOINT** 文で指定された識別子です。*savepoint-name* を省略すると、最新のセーブポイントが解放されます。

セーブポイントの詳細については、『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。セーブポイントを解放しても、どの種類の **COMMIT** も実行されません。現在アクティブなセーブポイントのリストから、セーブポイントが削除されるだけです。

標準

- 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。同様の機能が、ネストされたトランザクションを使用する Adaptive Server Enterprise 互換の方法で使用できます。

パーミッション

現在のトランザクション内の **SAVEPOINT** に対応するパーミッションを持っている必要があります。

参照：

- ROLLBACK TO SAVEPOINT 文 (335 ページ)
- SAVEPOINT 文 (337 ページ)

REMOVE 文

クラス、パッケージ、または JAR ファイルをデータベースから削除します。削除されたクラスは、変数型として利用できなくなります。

構文

```
REMOVE JAVA classes_to_remove
```

パラメータ

- **classes_to_remove：** – { **CLASS***java_class_name* [, *java_class_name*]… | **PACKAGE***java_package_name* [, *java_package_name*]… | **JAR***jar_name* [, *jar_name*]… [**RETAIN CLASSES**] }
- **jar_name：** – *character_string_expression*

例

- **例 1 –** 次の文は、現在のデータベースから "Demo" という名前の Java クラスを削除します。

```
REMOVE JAVA CLASS Demo
```

使用法

クラス、パッケージ、または JAR を削除するためには、それがインストールされていることが必要です。

java_class_name—削除される 1 つまたは複数の Java クラスの名前です。これらのクラスは、現在のデータベースにインストールされている必要があります。

java_package_name—削除される 1 つまたは複数の Java パッケージの名前です。これらのパッケージは現在のデータベース内にあるパッケージ名でなければなりません。

jar_name—最大長 255 の文字列値。

各 *jar_name* は、現在のデータベース内に保持されている *jar_name* と同じでなければなりません。*jar_name* と同じかどうかは、SQL システムの文字列比較規則で決定されます。

JAR...RETAIN CLASSES を指定すると、指定した JAR がデータベースに保持されなくなり、保持されたクラスは関連付けられた JAR を持たなくなります。**RETAIN CLASSES** を指定した場合、これが **REMOVE** 文の唯一のアクションになります。

標準

- 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。同様の機能が、ネストされたトランザクションを使用する Adaptive Server Enterprise 互換の方法で使用できます。

パーミッション

DBA 権限を持っているか、オブジェクトの所有者である必要があります。

RESIGNAL 文

例外条件の信号を送り返します。

構文

```
RESIGNAL [ exception-name ]
```

例

- **例 1 –** 次のコードは、カラムが見つからない (Column Not Found) 場合を除いて、すべての例外をアプリケーションに返します。

```
...
DECLARE COLUMN_NOT_FOUND EXCEPTION
    FOR SQLSTATE '52003';
...
EXCEPTION
WHEN COLUMN_NOT_FOUND THEN
SET message='Column not found' ;
WHEN OTHERS THEN
RESIGNAL ;
```

使用法

例外ハンドラの中で **RESIGNAL** を使用して、まだアクティブな例外を持つ複合文を終了するか、別の指定された例外のレポートを終了できます。例外は、別の例外ハンドラによって処理されるか、またはアプリケーションに返されます。**RESIGNAL** の前の例外ハンドラによる動作は取り消されます。

標準

- SQL—ISO/ANSI SQL 準拠。
- Sybase—Adaptive Server Enterprise ではサポートされていません。Transact-SQL プロシージャのエラー処理は、**RAISERROR** 文を使用して行われます。

パーミッション

なし

参照：

- BEGIN … END 文 (60 ページ)
- SIGNAL 文 (361 ページ)

RESTORE 文

1 つまたは複数のアーカイブ・デバイスから Sybase IQ データベース・バックアップをリストアします。

構文

構文 1

```
RESTORE DATABASE 'db_file'
FROM 'archive_device' [ FROM 'archive_device' ]…
… [ CATALOG ONLY ]
… [ KEY key_spec ]
… [ [ RENAME logical-dbfile-name
            TO 'new-dbspace-path'],...
    | VERIFY [ COMPATIBLE ] ]
```

構文 2

```
RESTORE DATABASE 'database-name'
[ restore-option ... ]
FROM 'archive_device' ...
```

パラメータ

- **db_file：** – リストアするデータベースの相対パスまたは絶対パス。カタログ・ストア・ファイルの場合、元のロケーションか、または新しいロケーションを指定できます。
- **key_spec：** – 大文字と小文字を混ぜた英数字、および特殊文字で構成される引用符文字列。キーをコマンド・シェルによる解読または変更から保護するために必要です。
- **restore-option：** –

```
 READONLY  dbspace-or-file [, … ]
KEY  key_spec
                          RENAME  file-name
                          TO  new-file-path ...
```

例

- **例 1 –** 次の UNIX の例は、iqdemo データベースを、Sun Solaris プラットフォームのテープ・デバイスである /dev/rmt/0 と /dev/rmt/2 からリストアします。Solaris の場合は、デバイス名に続けて *n* の文字を入力すると、"no rewind on close" 機能が指定されます。**RESTORE** でこの機能を指定するには、使用する UNIX プラットフォームに適した命名規則を使用してください(Windows はこの機能をサポートしていません)。

  ```
  RESTORE DATABASE 'iqdemo'
  FROM '/dev/rmt/0n'
  FROM '/dev/rmt/2n'
  ```

- **例 2 –** 次の例は、marvin という名前を持ち、キー *is!seCret* で暗号化されたデータベースをリストアします。

  ```
  RESTORE DATABASE 'marvin'
  FROM 'marvin_bkup_file1'
  FROM 'marvin_bkup_file2'
  FROM 'marvin_bkup_file3'
  KEY 'is!seCret'
  ```

- **例 3 –** 次の例は、**BACKUP** 文と 2 種類の **RESTORE** 文の構文を示します (この例では、説明のために iqdemo データベースのオブジェクトを使用します。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が、この DB 領域はお使いのデータベースには存在しない場合があります)。

  次の **BACKUP** 文があるとします。

  ```
  BACKUP DATABASE READONLY DBSPACES iq_main
  TO '/system1/IQ15/demo/backup/iqmain'
  ```

  DB 領域 iq_main をリストアするには、次のいずれかの **RESTORE** 文を使用します。

  ```
  RESTORE DATABASE 'iqdemo' READONLY DBSPACES iq_main
  FROM '/system1/IQ15/demo/backup/iqmain'
  ```

  または

  ```
  RESTORE DATABASE 'iqdemo'
  FROM '/system1/IQ15/demo/backup/iqmain'
  ```

  選択的バックアップによって、すべての READWRITE DB 領域、または指定した読み込み専用 DB 領域または dbfile がバックアップされます。選択的バックアップは、フル・バックアップまたはインクリメンタル・バックアップのサブタイプです。

注意：

- **READONLY** 選択的バックアップを使用すると、(上記の 2 番目の例のように) このバックアップからすべてのオブジェクトをリストアできます。
- 包括的なバックアップを使用すると、読み込み専用ファイルと DB 領域ファイルを選択的にリストアできます。
- 複数の読み込み専用ファイルと DB 領域の **READONLY** 選択的バックアップを使用すると、読み込み専用ファイルと DB 領域のサブセットを選択的にリストアできます。「パーミッション」を参照してください。
- 読み込み専用バックアップをリストアできるのは、読み込み専用ファイルがバックアップ以降に変更されていない場合に限られます。DB 領域を読み込み／書き込みに再度設定すると、DB 領域が読み込み専用に設定された時点にデータベースの読み込み／書き込みの全部分をリストアする場合を除き、読み込み専用のバックアップは無効になります。
- 選択的または非選択的の両方のバックアップ・サブタイプを使用することをせずに、常に一方のサブタイプのみを使用することをおすすめします。非選択的バックアップから選択的バックアップに、またはその逆に切り替える場合は、すべての変更がバックアップされるようにするために、新しいサブタイプに切り替える前に必ず非選択的フル・バックアップを行う必要があります。

- **例 4 –** 次の例は、書き込み操作を実行せずに、**VERIFY** 句を使用してデータベース・アーカイブを検証する構文を示します。

```
RESTORE DATABASE <database_name.db>
FROM '/sys1/dump/dmp1'
FROM '/sys1/dump/dmp2'
VERIFY
```

検証する場合は、"Database name not unique" エラーの発生を避けるために、異なるデータベース名を指定します。たとえば、元のデータベースが iqdemo.db の場合、代わりに iq_demo_new.db を使用します。

```
RESTORE DATABASE iqdemo_new.db FROM iqdemo.bkp VERIFY
```

使用法

RESTORE コマンドは、DBA によるデータベースへの排他的アクセスが必要です。この排他的アクセスは **-gd** スイッチを DBA に設定して実現できます。このスイッチはサーバ・エンジンの起動時のデフォルトです。

データベースを起動する前に **RESTORE** コマンドを発行します (utility_db データベースに接続する必要があります)。バックアップのタイプに合った **RESTORE** コマンドを指定して完了すると、そのデータベースは使用できる状態になります。データベースは、リストアした最新バックアップ内の最初の暗黙的な **CHECKPOINT** が終わった時の状態になります。これで、**START DATABASE** を指定

すると、他のユーザがリストアしたデータベースにアクセスできるようになります。

RESTORE コマンド全体の最大サイズは、すべての句を含めて 32KB です。

新しいデバイスにリストアするときは、リストアする DB 領域を書き込めるだけの大きさがそのデバイスにあるかどうかを確認します。IQ **RESTORE** はロー・デバイスのサイズをチェックし、DB 領域をリストアするために十分なサイズがなければエラーを返します。詳細については、『システム管理ガイド： 第 1 巻』の「データのバックアップ、リカバリ、アーカイブ」>「データベースのリストア」>「RESTORE 文」>「データベース・ファイルの移動」>「新しいデバイスへのリストア」を参照してください。

BACKUP コマンドでは、フル・バックアップまたはインクリメンタル・バックアップを指定できます。インクリメンタル・バックアップには、2 種類の選択肢があります。**INCREMENTAL** は最後のバックアップ (インクリメンタルまたはフル) 以降に変更またはコミットしたブロックのみをバックアップします。**INCREMENTAL SINCE FULL** は最後のフル・バックアップ以降に変更したブロックをすべてバックアップします。フル・バックアップの **RESTORE** の後に、いずれかのタイプのインクリメンタル・バックアップが続く場合、連続する **RESTORE** コマンド間でデータベースの変更を行うことはできません。この規則により、クラッシュ・リカバリを必要とするデータベース、または変更されたデータベース上で、インクリメンタル・バックアップからの **RESTORE** は実行できません。ただし、そのようなデータベースに対しては、フル・バックアップから **RESTORE** を実行してデータベースを上書きできます。

フル・リストアを開始する前に、 カタログ・ストア・ファイル (デフォルト名 `dbname.db`) とトランザクション・ログ・ファイル (デフォルト名 `dbname.log`) の 2 つのファイルを削除する必要があります。

インクリメンタル・バックアップをリストアする場合、**RESTORE** によってバックアップ・メディア・セットが正しい順序でアクセスされることが保証されます。つまり、最後のフル・バックアップ・テープ・セットを最初にリストアし、次に最初のインクリメンタル・バックアップ・テープ・セット、次に新しいセットというような順序で、最新のインクリメンタル・バックアップ・テープ・セットまで、順にリストアされます。DBA が **INCREMENTAL SINCE FULL** バックアップを作成した場合、必要なのは、フル・バックアップ・テープ・セットと最新の **INCREMENTAL SINCE FULL** バックアップ・テープ・セットだけです。ただし、**INCREMENTAL SINCE FULL** バックアップ以降に作成した **INCREMENTAL** バックアップが存在する場合は、それも使用されます。

Sybase IQ では、リストアの順序が正しいことが保証されます。正しくない場合はエラーが発生します。 リストア中にこれ以外のエラーが発生した場合、データベースは破損 (corrupt)、使用不可 (unusable) とマーク付けされます。そのような破

損したデータベースを修復するには、フル・バックアップと、必要な場合はその後の追加のインクリメンタル・バックアップから **RESTORE** を実行する必要があります これらのバックアップのどれかに破損が発生した可能性があるため、新しいほうのバックアップを無視し、古いほうのセットを使用しなければならない場合があります。

アーカイブ・バックアップから読み込み専用ファイルまたは DB 領域をリストアする場合、**RESTORE** 文の発行時にデータベースが実行しており、管理者がデータベースに接続していてもかまいません。バックアップに含まれている名前は、データベース・システムのテーブル情報と一致していれば、読み取り専用ファイルのパス名と一致している必要はありません。

READWRITE FILES ONLY またはすべてのファイルのバックアップの **FULL**、**INCREMENTAL SINCE FULL**、**INCREMENTAL** のバックアップからリストアするときは、データベースが起動していないことを確認してください。読み取り専用ファイルのバックアップをリストアする際には、データベースが実行中でも実行中でなくてもかまいません。読み込み専用 DB 領域で指定のファイルをリストアする場合は、DB 領域がオフラインである必要があります。読み込み／書き込み DB 領域で読み込み専用ファイルをリストアする場合は、DB 領域はオンラインまたはオフラインのどちらでもかまいません。リストアは読み込み専用ファイルをクローズし、ファイルをリストアしてから、これらのファイルをリストアの終了時に再オープンします。

DB 領域が同じ読み込み専用状態になっているかぎりは、選択的リストアを使用して、読み込み専用 DB 領域をリストアできます。

FROM—リストアする *archive_device* の名前を、一重引用符で囲んで指定します。複数のアーカイブ・デバイスを使用している場合、それぞれに別々の **FROM** 句を使用して指定します。カンマで区切ったリストは使用できません。 アーカイブ・デバイスは異なるものでなければなりません。入力デバイスに関して Sybase IQ が試みる並行処理の量は、**FROM** 句の数によって決まります。

バックアップ／リストア API の DLL 版では、アーカイブ・デバイスのオープン時に DLL に渡す引数を指定できます。サード・パーティの実装の場合は、*archive_device* 文字列は次のフォーマットになります。

```
'DLLidentifier::vendor_specific_information'
```

具体例：

```
'spsc::workorder=12;volname=ASD002'
```

archive_device 文字列の最大長は 1023 バイトです。*DLLidentifier* 部分は 1 から 30 バイトの長さで、英数字とアンダースコアの文字だけが使用できます。文字列の *vendor_specific_information* 部分は、内容の検査なしでサード・パーティの実装に渡されます。

注意： この構文を Sybase IQ で使用して動作確認されているサード・パーティ製品は、ごく一部に限られます。 使用上のその他の指示や制約については、『リリース・ノート』を参照してください。 サード・パーティ製品を使用して Sybase IQ データベースをバックアップする場合は、その製品が動作確認されたものであるかどうかを事前に確認してください。詳細については、『リリース・ノート』、または「Technical Documents」で提供されている Sybase IQ 製品に対する Sybase Certification Reports を参照してください。

バックアップ／リストア API の Sybase の実装の場合、テープ・デバイス名またはファイル名以外の情報を指定する必要はありません。ただし、ディスク・デバイスを使用する場合、バックアップ上で指定したものと同数のアクティブ・デバイスを **RESTORE** 上で指定する必要があります。同数を指定しない場合、バックアップの実行に使用した数と異なる数のリストア・デバイスが存在することになります。UNIX システム用に非リワインディング・テープ・デバイスを指定する Sybase API DLL 用のアーカイブ・デバイスの具体例を示します。

```
'/dev/rmt/0n'
```

CATALOG ONLY—アーカイブ・メディアからバックアップ・ヘッダー・レコードだけをリストアします。

RENAME—1 つ以上の Sybase IQ データベース・ファイルを新しいロケーションにリストアできます。 SYSFILE テーブル内と同じ表記で、移動する各 *dbspace-name* を指定します。また、*new-dbspace-path* を新しいロー・パーティションとして、またはその DB 領域の新しいフル・パス名または相対パス名として指定します。

相対パスを使用しデータベース・ファイルを作成した場合、ファイルはデフォルトでカタログ・ストア・ファイル (SYSTEM DB 領域) に相対的なロケーションにリストアされ、RENAME 句は必要ありません。絶対パスを使用してデータベース・ファイルを作成し、RENAME 句をファイルに指定しない場合、元のロケーションにリストアされます。

RENAME 句の相対パス名は、データベースまたは DB 領域を作成したときと同じように機能します。メイン IQ ストアの DB 領域、テンポラリ・ストアの DB 領域、メッセージ・ログは、db_file (カタログ・ストア) のロケーションに基づいてリストアされます。また、ユーザが作成した IQ ストアの DB 領域は、メイン IQ DB 領域が入っているディレクトリに基づいてリストアされます。

RENAME 句を使用して、カタログ・ストアを保持する SYSTEM DB 領域を移動しないでください。カタログ・ストアと、カタログ・ストアのロケーションに基づいて作成されたファイルの内、**RENAME** 句に指定されていないファイルを移動するには、*db_file* パラメータに新しいロケーションを指定します。

VERIFY [COMPATIBLE]—フル・バックアップ、インクリメンタル・バックアップ、フル・バックアップ以降のインクリメンタル・バックアップ、または仮想

バックアップの指定された Sybase IQ データベース・バックアップ・アーカイブを検証するようサーバに指示します。 バックアップは Sybase IQ バージョン 12.6 以降である必要があります。検証プロセスでは、リストア・プロセスでチェックされても書き込み操作が実行されないエラーと同じエラーがないかどうか、指定されたアーカイブをチェックします。 ステータス・メッセージと検出されたエラーは、すべてサーバ・ログ・ファイルに書き込まれます。

RENAME 句を **VERIFY** 句と一緒に使用することはできません。エラーがレポートされます。

バックアップ検証プロセスは、データベースのホストとは異なるホストで実行できます。**RESTORE VERIFY** を実行するには、DBA、BACKUP、または OPERATOR の権限が必要です。

VERIFY とともに **COMPATIBLE** 句が指定されている場合、インクリメンタル・アーカイブと既存のデータベース・ファイルとの互換性がチェックされます。**RESTORE...VERIFY COMPATIBLE** が呼び出されたシステム上にデータベース・ファイルが存在しない場合は、エラーが返されます。フル・バックアップの検証時に **COMPATIBLE** を指定した場合、このキーワードは無視されます。フル・バックアップのリストア中に、互換性チェックを実行する必要はありません。

フル・バックアップ内の読み込み専用 DB 領域のバックアップを検証するには、データベースとログ・ファイル (`.db` と `.log`) が必要です。これらのファイルがない場合は、**READONLY***dbspace* 句を指定せずに **RESTORE...VERIFY** を実行して、バックアップ全体を検証します。

詳細については、『システム管理ガイド： 第 1 巻』の「データのバックアップ、リカバリ、アーカイブ」>「データベースのリストア」>「RESTORE 文」>「リストア後のデータベースの検証」を参照してください。

注意： バックアップ・アーカイブの検証は、データベース一貫性チェッカ (DBCC) の verify モード (`sp_iqcheckdb 'verify...'`) とは異なります。**RESTORE VERIFY** では、バックアップ・アーカイブの一貫性を検証して、そのアーカイブをリストアできるかどうかを確認するのに対し、DBCC ではデータベースのデータの一貫性を検証します。

`sp_iqcheckdb 'verify...'` を実行してから、バックアップを実行してください。一貫性のないデータベースをバックアップし、同じバックアップ・アーカイブからリストアした場合、**RESTORE VERIFY** で検証の成功が報告された場合でも、データは一貫性のない状態のままになっています。

RESTORE のその他の注意点：

- ディスクへの **RESTORE** では、アーカイブ・デバイスとしてロー・デバイスをサポートしていません。

- Sybase IQ はテープをその使用前にリワインドしません。リワインディング・テープ・デバイス上で、使用後にテープをリワインドします。**RESTORE** を開始する前に各テープを Sybase IQ データの先頭に配置します。
- **BACKUP** や **RESTORE** の処理中にアーカイブ・デバイスをオープンできない場合 (メディアがロードされている必要があるなど) に、ATTENDED オプションが ON になっていると、Sybase IQ は次のテープをドライブに挿入できるよう 10 秒間待機してから再試行します。 デバイスが正常に挿入されるか、[Ctrl] キーを押しながら [C] キーを押して操作を終了しないかぎり、無制限に再試行されます。
- [Ctrl] キーを押しながら、[C] キーを押すと、**RESTORE** は失敗し、データベースがリストア開始前の状態に戻されます。
- ディスク・ストライピングを使用した場合、ストライプしたディスクを単一のデバイスとして処理します。
- `SYSTEM` DB 領域に対する `SYSFILE` システム・テーブルの `file_name` カラムは、リストア中は更新されません。`SYSTEM` DB 領域では、データベース作成時の名前が `file_name` カラムに常に反映されます。`SYSTEM` DB 領域のファイル名は、データベース・ファイルの名前です。

詳細については、『システム管理ガイド： 第 1 巻』の「データのバックアップ、リカバリ、アーカイブ」を参照してください。

標準

- 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。

パーミッション

DBA 権限が必要です。**-gu** サーバ起動オプションが値 **DBA** (デフォルト) に設定されている場合、SPACE ADMIN 権限を持つユーザは、読み込み専用の選択的リストアを実行できます。

参照：

- BACKUP 文 (52 ページ)

RESUME 文

クエリ後にプロシージャを再開します。

構文

構文 1

```
RESUME cursor-name
```

構文 2

```
RESUME [ ALL ]
```

パラメータ

- **cursor-name：** – 識別子
- **cursor-name：** – 識別子またはホスト変数

例

- **例 1 –** Embedded SQL の例を示します。

  ```
  EXEC SQL RESUME cur_employee;
  ```

  および

  ```
  EXEC SQL RESUME :cursor_var;
  ```

- **例 2 – dbisql** の例：

  ```
  CALL sample_proc() ;
  RESUME ALL;
  ```

使用法

RESUME 文は、結果セットを返すプロシージャの実行を再開します。

プロシージャは、次の結果セット (**INTO** 句のない **SELECT** 文) が見つかるまで実行されます。プロシージャが完了しても結果セットが見つからない場合は、SQLSTATE_PROCEDURE_COMPLETE 警告が設定されます。この警告は、**SELECT** 文に対してカーソルを **RESUME** するときにも設定されます。

注意： 構文 1 の **RESUME** 文は **dbisqlc** ではサポートされていますが、**dbisql** (Interactive SQL) や SQL Anywhere JDBC ドライバを使用してデータベースに接続しているときは無効です。

dbisqlRESUME 文 (構文 2) は現在のプロシージャを再開します。**ALL** を指定しない場合、**RESUME** を実行すると、次の結果セットが表示されるか、またはそれ以上結果セットが返されないときは、プロシージャが完了します。

dbisqlRESUME ALL 文は、プロシージャ内のすべての結果セットを表示しないで繰り返し、その後プロシージャを完了します。これは主にプロシージャをテストするときに便利です。

標準

- 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。

パーミッション

カーソルを事前にオープンしておく必要があります。

参照：
- DECLARE CURSOR 文 [ESQL] [SP] (191 ページ)

RETURN 文

関数またはプロシージャを無条件で終了し、オプションで値を返します。
RETURN に続く文は実行されません。

構文

```
RETURN [ ( expression ) ]
```

例

- **例 1 –** 3 つの番号の製品を返します。

```
CREATE FUNCTION product ( a numeric,
                b numeric ,
                c numeric)
RETURNS numeric
BEGIN
  RETURN ( a * b * c ) ;
END
```

- **例 2 –** 3 つの数値の積を計算します。

```
SELECT product (2, 3, 4)
```

```
product (2,3,4)
24
```

- **例 3 –** 次の例では、RETURN 文を使用して、意味のない複雑なクエリを実行するのを避けます。

```
CREATE PROCEDURE customer_products
( in customer_id integer DEFAULT NULL)
RESULT ( id integer, quantity_ordered integer )
BEGIN
  IF customer_id NOT IN (SELECT ID FROM Customers)
  OR customer_id IS NULL THEN
    RETURN
  ELSE
    SELECT ID,sum(
      SalesOrderItems.Quantity )
    FROM Products,
```

```
        SalesOrderItems,
        SalesOrders
    WHERE SalesOrders.CustomerID = customer_id
    AND SalesOrders.ID = SalesOrderItems.ID
    AND SalesOrderItems.ProductID = Products.D
    GROUP BY Products.ID
  END IF
END
```

使用法

expression を指定する場合、*expression* の値を関数またはプロシージャの値として返します。

関数の中では、式を関数の **RETURN** データ型と同じデータ型にしてください。

プロシージャ内では、**RETURN** は Transact-SQL との互換性を確保するためのもので、整数のエラー・コードを返すために使用されます。

標準

- SQL—ISO/ANSI SQL 準拠。
- Sybase—Transact-SQL プロシージャは、RETURN 文を使用して、整数のエラー・コードを返します。

パーミッション

なし

参照：

- BEGIN … END 文 (60 ページ)
- CREATE PROCEDURE 文 (137 ページ)

REVOKE 文

指定したユーザのパーミッションを取り消します。

構文

構文 1

```
REVOKE
{ BACKUP
| CONNECT
| DBA
| GROUP
| INTEGRATED LOGIN
| KERBEROS LOGIN
```

```
| MEMBERSHIP IN GROUP userid [, …]
| MULTIPLEX ADMIN
| OPERATOR
| PERMS ADMIN
| PROFILE
| RESOURCE
| SPACE ADMIN
| USER ADMIN }
| VALIDATE
… FROM userid [, …]
```

構文 2

```
REVOKE
{…ALL [ PRIVILEGES ] | ALTER | DELETE | INSERT
| REFERENCE | SELECT [ ( column-name [, …] ) ] | UPDATE [ ( column-
name, …) ] }
… ON [ owner.]table-name FROM userid [, …]
```

構文 3

```
REVOKE EXECUTE ON [ owner.]procedure-name FROM userid [, …]
```

構文 4

```
REVOKE CREATE ON dbspace-name FROM userid [, …]
```

例

- **例 1 –** ユーザ dave が Employees テーブルに挿入できないようにします。

  ```
  REVOKE INSERT ON Employees FROM dave
  ```

- **例 2 –** ユーザ Jim からリソース・パーミッションを取り消します。

  ```
  REVOKE RESOURCE FROM Jim
  ```

- **例 3 –** ユーザ dave が Employees テーブルを更新できないようにします。

  ```
  REVOKE UPDATE ON Employees FROM dave
  ```

- **例 4 –** ユーザ・プロファイル名 Administrator から統合化ログイン・マッピングを取り消します。

  ```
  REVOKE INTEGRATED LOGIN FROM Administrator
  ```

- **例 5 –** finance グループがプロシージャ **sp_customer_list** を実行できないようにします。

  ```
  REVOKE EXECUTE ON sp_customer_list
  FROM finance
  ```

- **例 6 –** ユーザ ID franw をデータベースから削除します。

  ```
  REVOKE CONNECT FROM franw
  ```

- **例 7 –** DB 領域 DspHist の CREATE 権限をユーザ Smith から取り消します。

  ```
  REVOKE CREATE ON DspHist FROM Smith
  ```

- **例 8 –** DB 領域 *DspHist* の CREATE パーミッションをデータベースのユーザ ID fionat から取り消します。

```
REVOKE CREATE ON DspHist FROM fionat
```

使用法

REVOKE 文を使用すると、**GRANT** 文を使用して付与したパーミッションを削除できます。

構文 1 を使用して特定のユーザ・パーミッション (権限) を取り消します。構文 2 を使用してテーブル・パーミッションを取り消します。構文 3 を使用してプロシージャを実行するパーミッションを取り消します。**REVOKE CONNECT** を使用して、データベースからユーザ ID を削除します。

注意： ユーザ ID の追加と削除を行うには、**GRANT** または **REVOKE** ではなくシステム・プロシージャを使用します。

REVOKE GROUP は、グループのすべてのメンバから自動的にメンバシップを取り消します。

REVOKE CREATE は、指定したユーザ ID から指定した DB 領域の CREATE パーミッションを削除します。

グループ内の特定のユーザに対してパーミッションを取り消すことはできません。特定のユーザが特定のテーブル、ビュー、またはプロシージャにアクセスできないようにするには、そのオブジェクトのパーミッションを持つグループのメンバからそのユーザを除外します。

注意： ユーザがテーブルなどのデータベース・オブジェクトを所有している場合、そのユーザの接続権限を取り消すことはできません。**REVOKE** 文または **sp_dropuser** プロシージャでそれを試行すると、`"Cannot drop a user that owns tables in runtime system."` のようなエラーが返されます。

関連する動作

- オートコミット

標準

- SQL—構文 1 は ISO/ANSI SQL 文法のベンダ拡張です。構文 2 は初級レベル機能です。構文 3 は永続的ストアド・モジュール機能です。
- Sybase—構文 2 と構文 3 は Adaptive Server Enterprise でサポートされています。構文 1 は、Adaptive Server Enterprise ではサポートされていません。ユーザ管理とセキュリティ・モデルは、Sybase IQ と Adaptive Server Enterprise では異なります。

パーミッション

取り消されるパーミッションの付与者であるか、DBA 権限を持っている必要があります。

構文 1 の場合は、REVOKE CONNECT、REVOKE INTEGRATED LOGIN、REVOKE KERBEROS LOGIN には、DBA 権限または USER ADMIN 権限が必要です。REVOKE GROUP、REVOKE (権限、DBA を除く)、REVOKE MEMBERSHIP IN GROUP には、DBA 権限または PERMS ADMIN 権限が必要です。DBA のみが DBA 権限を取り消すことができます。

別のユーザの **CONNECT** パーミッションまたはテーブル・パーミッションを取り消す場合、他のユーザはそのデータベースに接続できません。

構文 2 の場合は、REVOKE、REVOKE ALTER、REVOKE DELETE、REVOKE INSERT、REVOKE REFERENCE、REVOKE SELECT、REVOKE UPDATE には、DBA 権限または PERMS ADMIN 権限が必要です。

構文 3 の場合は、DBA 権限または PERMS ADMIN 権限が必要です。

構文 4 の場合は、DBA 権限または SPACE ADMIN 権限が必要です。

参照：

- GRANT 文 (247 ページ)

ROLLBACK 文

最後の **COMMIT** または **ROLLBACK** の後に加えられた変更を取り消します。

構文

```
ROLLBACK [ WORK ]
```

使用法

ROLLBACK は、作業 (トランザクション) の論理単位を終了し、このトランザクションの間にデータベースに加えられたすべての変更を取り消します。トランザクションは、1 つのデータベース接続上で **COMMIT** 文または **ROLLBACK** 文の間で行われるデータベース作業です。

関連する動作

- WITH HOLD 句を指定してオープンしなかったカーソルをすべてクローズします。

- **ROLLBACK** を実行して、トランザクションによって保持されているロックを開放します。

標準

- SQL—ISO/ANSI SQL 準拠。
- Sybase—Adaptive Server Enterprise でサポートされています。

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。

参照：
- COMMIT 文 (75 ページ)
- ROLLBACK TO SAVEPOINT 文 (335 ページ)

ROLLBACK TO SAVEPOINT 文

SAVEPOINT の後に加えられたすべての変更を取り消します。

構文

```
ROLLBACK
            TO
            SAVEPOINT [ savepoint-name ]
```

使用法

ROLLBACK TO SAVEPOINT 文は、**SAVEPOINT** の作成後に加えられたすべての変更を取り消します。

SAVEPOINT の前に加えられた変更は取り消されず、そのまま残ります。セーブポイントの詳細については、『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。

savepoint-name は、現在のトランザクションの **SAVEPOINT** 文で指定された識別子です。*savepoint-name* を省略すると、最新のセーブポイントが使用されます。指定したセーブポイントの後にあるセーブポイントは自動的に解放されます。

標準

- SQL—ISO/ANSI SQL 文法のベンダ拡張。

- Sybase—セーブポイントは Adaptive Server Enterprise ではサポートされていません。 Adaptive Server Enterprise 互換の方法で同様の機能を実装するには、ネストされたトランザクションを使用します。

パーミッション

現在のトランザクション内の **SAVEPOINT** に対応するパーミッションを持っている必要があります。

参照：

- RELEASE SAVEPOINT 文 (318 ページ)
- ROLLBACK 文 (334 ページ)
- SAVEPOINT 文 (337 ページ)

ROLLBACK TRANSACTION 文 [T-SQL]

SAVE TRANSACTION の後に加えられたすべての変更を取り消します。

構文

ROLLBACK TRANSACTION [*savepoint-name*]

例

- **例 1 –** 次の例では、値 10、20 などを持つ 5 つのローが返されます。DELETE の効果は **ROLLBACK TRANSACTION** 文によって取り消されますが、前の INSERT または UPDATE の効果は取り消されません。

```
BEGIN
    SELECT row_num INTO #tmp
    FROM sa_rowgenerator( 1, 5 )
    UPDATE #tmp SET row_num=row_num*10
    SAVE TRANSACTION before_delete
    DELETE FROM #tmp WHERE row_num >= 3
    ROLLBACK TRANSACTION before_delete
    SELECT * FROM #tmp
END
```

使用法

ROLLBACK TRANSACTION は、**SAVE TRANSACTION** を使用してセーブポイントが確立された後に加えられたすべての変更を取り消します。**SAVE TRANSACTION** の前に加えられた変更は取り消されず、そのまま残ります。

savepoint-name は、現在のトランザクションの **SAVE TRANSACTION** 文で指定された識別子です。*savepoint-name* を省略した場合、未確定の変更すべてがロールバックされます。指定したセーブポイントの後にあるセーブポイントは自動的に解放されます。

『SQL Anywhere サーバー - SQL リファレンス』の「SQL の使用」>「SQL 文」>「ROLLBACK TRANSACTION 文 [T-SQL]」を参照してください。

注意： このリファレンスは SQL Anywhere マニュアルにリンクされています。

標準

- ISO/ANSI SQL 文法のベンダ拡張。

パーミッション

現在のトランザクション内の **SAVE TRANSACTION** に対応するパーミッションを持っている必要があります。

参照：

- BEGIN TRANSACTION 文 [T-SQL] (64 ページ)
- SAVE TRANSACTION 文 [T-SQL] (338 ページ)

SAVEPOINT 文

現在のデータベース内でセーブポイントを確立します。

構文

```
SAVEPOINT [ savepoint-name ]
```

使用法

savepoint-name は **RELEASE SAVEPOINT** 文または **ROLLBACK TO SAVEPOINT** 文の中で使用できる識別子です。

トランザクションが終了すると、すべてのセーブポイントは自動的に解放されます。『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。

トリガまたはアトミックな複合文の実行中に確立されたセーブポイントは、アトミック操作が終了すると自動的に解放されます。

標準

- 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。Adaptive Server Enterprise 互換の方法で同様の機能を実装するには、ネストされたトランザクションを使用します。

パーミッション

なし

参照：

- RELEASE SAVEPOINT 文 (318 ページ)
- ROLLBACK TO SAVEPOINT 文 (335 ページ)

SAVE TRANSACTION 文 [T-SQL]

現在のデータベース内でセーブポイントを確立します。

構文

```
SAVE TRANSACTION [ savepoint-name ]
```

例

- **例 1 –** 次の例では、値 10、20 などを持つ 5 つのローが返されます。DELETE の効果は **ROLLBACK TRANSACTION** 文によって取り消されますが、前の INSERT または UPDATE の効果は取り消されません。

```
BEGIN
     SELECT row_num INTO #tmp
     FROM sa_rowgenerator( 1, 5 )
     UPDATE #tmp SET row_num=row_num*10
     SAVE TRANSACTION before_delete
     DELETE FROM #tmp WHERE row_num >= 3
     ROLLBACK TRANSACTION before_delete
     SELECT * FROM #tmp
END
```

『SQL Anywhere サーバー - SQL リファレンス』の「SQL 文」>「SQL 文」>「SAVE TRANSACTION 文 [T-SQL]」を参照してください。

注意： このリファレンスは SQL Anywhere マニュアルにリンクされています。

使用法

現在のデータベース内でセーブポイントを確立します。*savepoint-name* は **ROLLBACK TRANSACTION** 文の中で使用できる識別子です。トランザクションが終了すると、すべてのセーブポイントは自動的に解放されます。

標準

- SQL—ISO/ANSI SQL 文法のベンダ拡張。

パーミッション

なし

参照：

- BEGIN TRANSACTION 文 [T-SQL] (64 ページ)
- ROLLBACK TRANSACTION 文 [T-SQL] (336 ページ)

SELECT 文

データベースから情報を取り出します。

構文

```
SELECT [ ALL | DISTINCT ] [ FIRST | TOP number-of-rows ] select-list
… [ INTO { host-variable-list | variable-list | table-name } ]
… [ INTO LOCAL TEMPORARY TABLE { table-name } ]
… [ FROM table-list ]
… [ WHERE search-condition ]
… [ GROUP BY [ expression [, …]
      | ROLLUP ( expression [, …] )
      | CUBE ( expression [, …] ) ] ]
… [ HAVING search-condition ]
… [ ORDER BY { expression | integer } [ ASC | DESC ] [, …] ]
… [ row-limitation-option ]
```

パラメータ

- **select-list： –**

  ```
  { column-name
  | expression [ [ AS ] alias-name ]
  | * }
  ```

- ***row-limitation-option: –***

```
LIMIT { [ offset-expression, ] limit-expression
| limit-expression OFFSET offset-expression }
```

- ***limit-expression:*** *– simple-expression*
- ***offset-expression:*** *– simple-expression*
- ***simple-expression:*** *–*

```
integer
| variable
| ( simple-expression )
| ( simple-expression { + | - | * } simple-expression )
```

例

- **例 1 –** システム・カタログのすべてのテーブルとビューをリストします。

```
SELECT tname
FROM SYS.SYSCATALOG
WHERE tname LIKE 'SYS%' ;
```

- **例 2 –** すべての顧客とその顧客からの注文の総額をリストします。

```
SELECT CompanyName,
  CAST( sum(SalesOrderItems.Quantity *
  Products.UnitPrice) AS INTEGER) VALUE
FROM Customers
  LEFT OUTER JOIN SalesOrders
  LEFT OUTER JOIN SalesOrderItems
  LEFT OUTER JOIN Products
GROUP BY CompanyName
ORDER BY VALUE DESC
```

- **例 3 –** 従業員の数をリストします。

```
SELECT count(*)
FROM Employees;
```

- **例 4 –** 次に、Embedded SQL の SELECT 文を示します。

```
SELECT count(*) INTO :size FROM Employees;
```

- **例 5 –** 年、モデル、色別に売り上げ合計をリストします。

```
SELECT year, model, color, sum(sales)
FROM sales_tab
GROUP BY ROLLUP (year, model, color);
```

- **例 6 –** 値引きのあるすべての項目を抽出し、テンポラリ・テーブルに格納します。

```
SELECT * INTO #TableTemp FROM lineitem
WHERE l_discount < 0.5
```

- **例 7 –** 従業員を姓でソートした場合の最初の従業員に関する情報を返します。

```
SELECT FIRST *
FROM Employees
ORDER BY Surname;
```

- **例 8 –** 従業員の姓名を姓でソートした場合の、最初の 5 人の従業員を返します。

```
SELECT TOP 5 *
FROM Employees
ORDER BY Surname;
```

```
SELECT *
FROM Employees
ORDER BY Surname
LIMIT 5;
```

- **例 9 –** 姓で降順にソートした場合の 5 番目と 6 番目の従業員をリストします。

```
SELECT *
FROM Employees
ORDER BY Surname DESC
LIMIT 4,2;
```

- **例 10 –** 従業員の姓名を姓でソートした場合の、最初の 5 人の従業員を返します。

```
CREATE OR REPLACE VARIABLE atop INT = 10;
```

```
SELECT TOP (atop -5) *
FROM Employees
ORDER BY Surname;
```

```
SELECT *
FROM Employees
ORDER BY Surname
LIMIT (atop-5);
```

- **例 11 –** 姓で降順にソートした場合の 5 番目と 6 番目の従業員をリストします。

```
CREATE OR REPLACE VARIABLE atop INT = 10;
```

```
SELECT *
FROM Employees
ORDER BY Surname DESC
LIMIT (atop - 8) OFFSET (atop -2 -3 -1);
```

使用法

SELECT 文を Interactive SQL 内で使用して、データベース内のデータを参照したり、データベースから外部ファイルにデータをエクスポートしたりできます。

SELECT 文は、プロシージャまたは Embedded SQL 内でも使用できます。**SELECT** 文がローを 1 つだけ返す場合は、**INTO** 句のある **SELECT** 文を使ってデータベースから結果を取り出します (**SELECT INTO** で作成されるテーブルは、IDENTITY/AUTOINCREMENT テーブルを継承しません)。複数のローを対象にクエリを実行する場合は、カーソルを使用する必要があります。複数のカラムを選択し、*#table* を使用しなければ、**SELECT INTO** によって永久ベース・テーブルが作成されます。**SELECT INTO** に *#table* を指定すると、カラムの数に関係なく、常にテンポラリ・テーブルが作成されます。テーブルへの **SELECT INTO** で、カラムが 1 つしかない場合は、結果がホスト変数に返されます。

注意： テンポラリ・テーブルに対して **SELECT INTO** を実行するスクリプトとストアド・プロシージャを作成するときは、ベース・カラムではない select リスト項

目を **CAST** 式にラップすることをおすすめします。これにより、テンポラリ・テーブルのカラムのデータ型が目的のデータ型になることが保証されます。

名前が同じで所有者が異なるテーブルには、エイリアスを使用する必要があります。エイリアスを使用しないクエリは、間違った結果を返します。

```
SELECT * FROM user1.t1
WHERE NOT EXISTS
(SELECT *
FROM user2.t1
WHERE user2.t1.col1 = user1.t.col1);
```

正しい結果を返すには、各テーブルに次のようにエイリアスを使用します。

```
SELECT * FROM user1.t1 U1
WHERE NOT EXISTS
(SELECT *
FROM user2.t1 U2
WHERE U2.col1 = U1.col1);
```

variable-list がある **INTO** 句を使用できるのは、プロシージャ内だけです。

SELECT 文内でベース・テーブルまたはビューが許可されている場所に、ストアド・プロシージャ・コールを記述できるようになりました。 ただし、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。 たとえば、**SELECT** 文を使用して、プロシージャから結果セットを返すこともできます。 構文と使用例については、『SQL Anywhere サーバー – SQL リファレンス』の「SQL 文」>「SQL 文」>「FROM 句」を参照してください。

注意： このリファレンスは SQL Anywhere マニュアルにリンクされています。

ストアド・プロシージャ内のテンポラリ・テーブルから選択する処理に影響を及ぼす制限については、『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」>「プロシージャ入門」>「プロシージャの結果を結果セットとして返す」を参照してください。

ALL または DISTINCT—どちらも指定しない場合、**SELECT** 文の句を満たす ALL (全) ローが取り出されます。**DISTINCT** を指定すると、重複した出力ローが除外されます。これは文の結果の射影と呼びます。多くの場合、**DISTINCT** を指定すると文の実行に時間が非常に長くかかるため、**DISTINCT** は必要な場合だけ使用するようにしてください。

DISTINCT を使用する場合、**DISTINCT** パラメータを持つ集合関数を文に含めることはできません。

FIRST または TOP number-of-rows—クエリから返されるローの数を指定します。**FIRST** を指定すると、クエリから選択された最初のローが返されます。**TOP** では、

指定された数のローがクエリから返されます。*number-of-rows* には 1 ～ 2147483647 の値を、整数の定数か整数の変数で指定します。

FIRST と **TOP** は、主に **ORDER BY** 句で使用されます。これらのキーワードが **ORDER BY** 句以外で使用された場合、同じクエリを実行しても、そのたびにオプティマイザが異なるクエリ・プランを選択して、結果が変わる可能性があります。

FIRST と **TOP** は、クエリの最上位レベルの **SELECT** でのみ使用できます。したがって、抽出テーブルやビュー定義では使用できません。**FIRST** や **TOP** をビュー定義で使用すると、クエリがビューで実行されたときに、このキーワードが無視される可能性があります。

FIRST の使用は、ROW_COUNT データベース・オプションを 1 に設定する場合と同じです。**TOP** の使用は、ROW_COUNT オプションをロー数と同じ数に設定する場合と同じです。**TOP** と ROW_COUNT が両方とも設定された場合、**TOP** の値が優先されます。

ROW_COUNT オプションは、グローバル変数、システム関数、またはプロキシ・テーブルを伴うクエリで使用された場合、一貫性のない結果を生成する可能性があります。詳細については、「ROW_COUNT オプション」を参照してください。

select-list—*select-list* は、カンマで区切った式のリストであり、データベースから何を取り出すかを指定します。アスタリスク (*) を指定すると、**FROM** 句に記述された全テーブルの全カラムを選択することを意味します (table-name は、指定したテーブルのすべてのカラムを選択します)。*select-list* には、集合関数と統計関数を使用できます。『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 関数」を参照してください。

注意： Sybase IQ でも、SQL Anywhere や Adaptive Server Enterprise と同様、スカラ・サブクエリ (ネストされた select) を最上位レベルの **SELECT** の select リストで使用できます。 サブクエリを条件値の式の内部 (**CASE** 文の内部など) で使用することはできません。

WHERE 句や **HAVING** 句の述部 (サポートされる述部のタイプのいずれか) でサブクエリを使用することもできます。 ただし、**WHERE** 句または **HAVING** 句の内部であっても、値の式の内部や **CONTAINS** または **LIKE** の述部の内部でサブクエリを使用することはできません。外部ジョインの **ON** 句、または **GROUP BY** 句でサブクエリを使用することはできません。

サブクエリの使用の詳細については、『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 言語の要素」>「式」>「式内のサブクエリ」と『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 言語の要素」>「検索条件」>「探索条件内のサブクエリ」を参照してください。

---

alias-names はクエリを通じて使用でき、エイリアスの式を表します。 エイリアス名も、**SELECT** 文から出力された各カラムの最上部に、Interactive SQL で表示されます。 オプションの *alias-name* が式の後で指定されていなければ、Interactive SQL は式自体を表示します。 カラムのエイリアスにカラム名と同じ名前、または式を使用する場合、名前はテーブル・カラム名でなく、エイリアス・カラムとして処理されます。

INTO host-variable-list—Embedded SQL 内でのみ使用できます。これは **SELECT** 文の結果が移動する場所を指定します。*select-list* 内の各項目に対して、それぞれ 1 つの *host-variable* 項目が必要です。select リスト項目は、順番にホスト変数の中に置かれます。インジケータ・ホスト変数も各 *host-variable* とともに使用でき、プログラムは select リスト項目が NULL であったかどうかを通知できます。

INTO variable-list—使用できるのは、プロシージャ内だけです。これは **SELECT** 文の結果が移動する場所を指定します。select リスト内の各項目に対して、それぞれ 1 つの変数項目が必要です。select リスト項目は、順番に変数の中に置かれます。

INTO table-name—テーブルを作成して、そこにデータを入力するために使用されます。

テーブル名が # で始まる場合は、テンポラリ・テーブルとして作成されます。そうでなければ、テーブルは永久ベース・テーブルとして作成されます。永久テーブルを作成するには、クエリが次のいずれかの条件を満たしている必要があります。

- *select-list* に複数の項目が指定され、**INTO** ターゲットに *table-name* 識別子が 1 つだけ指定されている。または、
- select リストに * が指定され、**INTO** のターゲットが *owner.table* の形で指定されている。

カラムが 1 つである永久テーブルを作成するには、テーブル名を *owner.table* の形で指定します。所有者の指定を省略すると、テンポラリ・テーブルが作成されます。

この文では、テーブルを作成する副作用として実行前に **COMMIT** が行われます。この文の実行には、RESOURCE 権限が必要です。新しいテーブルに対するパーミッションは与えられません。これは、**CREATE TABLE** とそれに続く **INSERT... SELECT** を短縮して定義できる文です。

ストアド・プロシージャまたは関数から **SELECT INTO** を使用することはできません。これは、**SELECT INTO** がアトミック文であり、アトミック文内では **COMMIT**、**ROLLBACK**、または一部の **ROLLBACK TO SAVEPOINT** 文を実行できないためです。詳細については、『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」>「制御文」>「アトミックな複合文」と『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」>「プロシージャでのトランザクションとセーブポイント」を参照してください。

この文を使用して作成されたテーブルには、プライマリ・キーが定義されていません。**ALTER TABLE** を使用してプライマリ・キーを追加できます。プライマリ・キーは、テーブルに UPDATE または DELETE を適用する前に追加してください。そうしないと、これらの操作によって、影響を受けるローのトランザクション・ログにすべてのカラム値が記録されます。

この句は、SQL Anywhere の有効なクエリにしか使用できません。Sybase IQ 拡張機能はサポートされません。

INTO LOCAL TEMPORARY TABLE—ローカル・テンポラリ・テーブルを作成し、そのテーブルにクエリの結果を格納します。この句を使用する場合、テンポラリ・テーブル名の先頭に # を付ける必要はありません。

FROM table-list—*table-list* 内で指定したテーブルとビューからローを取り出します。ジョイン演算子を使用して、ジョインを指定できます。詳細については、「FROM 句」を参照してください。**FROM** 句を指定しない **SELECT** 文を使って、テーブルから取り出せなかった式の値を表示できます。次に例を示します。

```
SELECT @@version
```

これはグローバル変数 @@version の値を表示します。これは次と同じです。

```
SELECT @@version
FROM DUMMY
```

注意： **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、特に構文と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`" 句を追加すると、強制的に Sybase IQ で処理させることができます。この `iq_dummy` は、ユーザが自分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。

WHERE search-condition—**FROM** 句に指定されたテーブルからどのローを選択するかを指定します。これを使用すると、複数のテーブル間のジョインも行えます。これを行うには、**WHERE** 句内に条件を入れます。この WHERE 句は、一方のテーブルのカラムまたはカラム・グループを、他方のテーブルのカラムまたはカラム・グループと関連付けます。両方のテーブルを **FROM** 句内にリストする必要があります。

同じ **CASE** 文を、グループ化したクエリの **SELECT** と **WHERE** 句の両方で使用することはできません。『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 言語の要素」>「検索条件」を参照してください。

Sybase IQ では、サブクエリ述部の分離もサポートされています。 各サブクエリは、**WHERE** 句または **HAVING** 句内で他の述部とともに指定し、AND 演算子または OR 演算子を使用して結合できます。『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 言語の要素」>「検索条件」>「探索条件内のサブクエリ」>「サブクエリ述部の分離」を参照してください。

GROUP BY—カラム、エイリアス名、または関数によってグループ化できます。 **GROUP BY** 式も select リストの前に入れる必要があります。 クエリ結果には、指定したカラム、エイリアス、または関数内の異なる値セットそれぞれに対してローが 1 つずつ入ります。 テーブル・リストの各ロー・グループに対して 1 つのローが出力されるため、出力される各ローをグループと呼ぶことがあります。 **GROUP BY** 句があると、すべての NULL 値は同じものとして扱われます。 集合関数をこれらのグループに適用して、意味のある結果を取得できます。

GROUP BY には、定数を 2 つ以上含める必要があります。グループ化するクエリに定数を選択するために、**GROUP BY** 句に定数を追加する必要はありません。 **GROUP BY** 式に定数が 1 つしか含まれない場合は、エラーが返されてクエリが拒否されます。

GROUP BY を使用する場合、select リスト、**HAVING** 句、**ORDER BY** 句で参照できるのは、**GROUP BY** 句の中で指定した識別子だけです。ただし、 *select-list* と **HAVING** 句は例外で、集合関数を持つことができます。

ROLLUP 演算子—**GROUP BY** 句で **ROLLUP** 演算子を使用すると、さまざまなレベルの詳細を使用して小計を分析できます。非常に詳細なレベルから総計までロールアップする小計を作成します。

ROLLUP 演算子では、グループ化式が順に並べられたリストを引数に指定する必要があります。**ROLLUP** は、最初に **GROUP BY** に指定された標準の集合値を計算します。次に、**ROLLUP** はグループ化を行うカラムのリストを右から左に移動し、より高いレベルの小計を連続して作成します。最後に総計が作成されます。グループ化するカラムの数が *n* 個の場合、**ROLLUP** は *n+1* レベルの小計を作成します。

ROLLUP 演算子には次の制限があります。

- **ROLLUP** 演算子は **GROUP BY** 句で使用可能なすべての集合関数をサポートしますが、**ROLLUP** は現在 **COUNT DISTINCT** と **SUM DISTINCT** をサポートしていません。
- **ROLLUP** は **SELECT** 文でのみ使用できます。**ROLLUP** は **SELECT** サブクエリで使用できません。

- **ROLLUP**、**CUBE**、**GROUP BY** のカラムを同じ **GROUP BY** 句内で組み合わせた複数グループ化の指定は、現在サポートされていません。
- **GROUP BY** のキーに定数式を指定することはできません。

GROUPING に **ROLLUP** 演算子を付けて使用すると、格納されていた NULL 値と **ROLLUP** によって作成されたクエリ結果の NULL 値を区別できます。

ROLLUP 構文：

```
SELECT ... [ GROUPING ( column-name ) ...] ... GROUP BY [ expression [, ...] |
ROLLUP ( expression [, ...] ) ]
```

演算子の式のフォーマットについては、『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 言語の要素」>「式」を参照してください。

GROUPING は、カラム名をパラメータとして受け取り、ブール値を返します。

表 13 : ROLLUP 演算子が指定された GROUPING によって返される値

結果値の種類	GROUPING の戻り値
ROLLUP 処理によって作成された NULL	1 (真)
ローが小計であることを示す NULL	1 (真)
ROLLUP 演算子によっては作成されていない NULL	0 (偽)
格納されていた NULL	0 (偽)

ROLLUP の例については、『システム管理ガイド：第 2 巻』の「OLAP の使用」を参照してください。

CUBE 演算子—**GROUP BY** 句の **CUBE** 演算子は、データを複数の次元でグループ化することでデータを分析します。**CUBE** にはグループ化式 (次元) を順に並べたリストを引数として指定する必要があります。これにより、**SELECT** 文は組み合わせ可能なすべての次元のグループに対して、小計を計算できるようになります。

CUBE 演算子には次の制限があります。

- **CUBE** 演算子は **GROUP BY** 句で使用可能なすべての集合関数をサポートしますが、**CUBE** は現在 **COUNT DISTINCT** と **SUM DISTINCT** をサポートしていません。
- **CUBE** は、逆分散統計関数である **PERCENTILE_CONT** と **PERCENTILE_DISC** を現在はサポートしていません。
- **CUBE** は **SELECT** 文でのみ使用できます。**CUBE** は **SELECT** サブクエリでは使用できません。
- **ROLLUP**、**CUBE**、**GROUP BY** のカラムを同じ **GROUP BY** 句内で組み合わせた複数の **GROUPING** の指定は、現在サポートされていません。

- **GROUP BY** のキーに定数式を指定することはできません。

GROUPING に **CUBE** 演算子を付けて使用すると、格納されていた NULL 値と **CUBE** によって作成されたクエリ結果の NULL 値を区別できます。

CUBE 構文：

```
SELECT … [ GROUPING ( column-name ) …] … GROUP BY [ expression [, …] |
CUBE ( expression [, …] ) ]
```

GROUPING は、カラム名をパラメータとして受け取り、ブール値を返します。

表 14 : CUBE 演算子が指定された GROUPING によって返される値

結果値の種類	GROUPING の戻り値
CUBE 処理によって作成された NULL	1 (真)
ローが小計であることを示す NULL	1 (真)
CUBE 演算子によっては作成されていない NULL	0 (偽)
格納されていた NULL	0 (偽)

クエリ・プランを生成するとき、Sybase IQ オプティマイザは、**GROUP BY CUBE** ハッシュ操作で生成されるグループの合計数を見積もります。
MAX_CUBE_RESULTS データベース・オプションには、実行可能なハッシュ・アルゴリズムに対してオプティマイザが推測する、ローの予想数の上限を設定します。 実際のロー数が MAX_CUBE_RESULT オプションの値を超える場合、オプティマイザはクエリの処理を中止し、"Estimate number: nnn exceed the DEFAULT_MAX_CUBE_RESULT of GROUP BY CUBE or ROLLUP" というエラー・メッセージを表示します。*nnn* は、IQ オプティマイザが見積もった数字です。 MAX_CUBE_RESULT オプションの設定については、「MAX_CUBE_RESULT オプション」を参照してください。

CUBE の例については、『システム管理ガイド：第 2 巻』の「OLAP の使用」を参照してください。

HAVING search-condition—個々のロー値ではなく、グループ値に基づきます。 **HAVING** 句を使用できるのは、文が **GROUP BY** 句を持っているか、select リストが集合関数だけで構成されている場合だけです。**HAVING** 句の中で参照されるカラム名は、**GROUP BY** 句の中に入れるか、または **HAVING** 句の中の集合関数に対するパラメータとして使用する必要があります。

ORDER BY—クエリ結果をソートします。**ORDER BY** リストの各項目には、昇順の場合は ASC、降順の場合は DESC のラベルを付けることができます。指定がない場合は、昇順であるとみなします。式が整数 **n** である場合、クエリ結果は select リストの **n** 番目の項目でソートされます。

Embedded SQL の場合は、データベースから結果を取得し、その値を **INTO** 句でホスト変数に格納するために、**SELECT** 文を使用します。**SELECT** 文は、1 つのローだけを返すようにします。複数のローを対象にクエリを実行する場合は、カーソルを使用する必要があります。

Java のクラスを **SELECT** リストに指定することはできませんが、たとえば、Java のクラスのラッパとして機能する関数または変数を作成し、それを指定することは可能です。

row-limitation 句—ローを制限する句を使用すると、**WHERE** 句を満たすローのサブセットのみを返すことができます。1 度に指定できる *row-limitation* 句は 1 つのみです。この句を指定する場合は、意味のある順にローを並べるために **ORDER BY** 句を指定する必要があります。

LIMIT 引数と **OFFSET** 引数は、ホスト変数、整数定数、または整数変数についての簡単な算術式にすることができます。**LIMIT** 引数は、0 以上の値に評価される必要があります。また、**OFFSET** 引数も、0 以上の値に評価される必要があります。*offset-expression* が指定されていない場合、デフォルトは 0 です。

ローを制限する句である **LIMIT***offset-expression*, *limit-expression* は、**LIMIT***limit-expression***OFFSET***offset-expression* と同じです。

LIMIT キーワードは、デフォルトで無効になっています。**LIMIT** キーワードを有効にするには、RESERVED_KEYWORDS オプションを使用します。

標準

- SQL—ISO/ANSI SQL 準拠。
- Sybase—Adaptive Server Enterprise でサポートされますが、構文にいくつか相違点があります。

パーミッション

指定したテーブルとビューに対する SELECT パーミッションが必要です。

参照：

- CREATE VIEW 文 (186 ページ)
- DECLARE CURSOR 文 [ESQL] [SP] (191 ページ)
- FETCH 文 [ESQL] [SP] (230 ページ)
- FROM 句 (237 ページ)
- MAX_CUBE_RESULT オプション (482 ページ)
- OPEN 文 [ESQL] [SP] (301 ページ)
- UNION 演算 (371 ページ)

- RESERVED_KEYWORDS オプション (519 ページ)
- ROW_COUNT オプション (521 ページ)
- SUBQUERY_CACHING_PREFERENCE オプション (527 ページ)

SET 文 [ESQL]

SQL 変数に値を割り当てます。

構文

```
SET identifier = expression
```

例

- **例 1 –** 次のコードは、データベースに大きなテキスト値を挿入します。

```
EXEC SQL BEGIN DECLARE SECTION;
char buffer[5001];
EXEC SQL END DECLARE SECTION;


EXEC SQL CREATE VARIABLE hold_text VARCHAR;
EXEC SQL SET hold_text = '';
for(;;) {
    /* read some data into buffer ... */
    size = fread( buffer, 1, 5000, fp );
    if( size <= 0 ) break;

    /* buffer must be null-terminated */
    buffer[size] = '¥0';
    /* add data to blob using concatenation */
    EXEC SQL SET hold_text = hold_text || :buffer;
}
EXEC SQL INSERT INTO some_table VALUES ( 1, hold_text );
EXEC SQL DROP VARIABLE hold_text;
```

- **例 2 –** 次のコードは、データベースに大きなバイナリ値を挿入します。

```
EXEC SQL BEGIN DECLARE SECTION;
DECL_BINARY( 5000 ) buffer;
EXEC SQL END DECLARE SECTION;
EXEC SQL CREATE VARIABLE hold_blob LONG BINARY;
EXEC SQL SET hold_blob = '';
for(;;) {
    /* read some data into buffer ... */
    size = fread( &(buffer.array), 1, 5000, fp );
    if( size <= 0 ) break;
    buffer.len = size;


    /* add data to blob using concatenation
```

```
        Note that concatenation works for
        binary data too! */
    EXEC SQL SET hold_blob = hold_blob || :buffer;
}
EXEC SQL INSERT INTO some_table VALUES ( 1, hold_blob );
EXEC SQL DROP VARIABLE hold_blob;
```

使用法

SET 文は、**CREATE VARIABLE** 文を使用してそれまでに作成してある変数に新しい値を割り当てます。

変数は、カラム名を使用できる場所なら SQL 文のどこででも使用できます。識別子と一致するカラム名が存在しない場合、データベース・サーバはその値と一致する変数がないかどうかをチェックし、その値を使います。

変数は、現在の接続にローカルなもので、データベースとの接続を切断したり、**DROP VARIABLE** を使用したりすると自動的に消えます。変数は **COMMIT** または **ROLLBACK** 文の影響を受けません。

変数は、Embedded SQL プログラムから **INSERT** 文または **UPDATE** 文の対象となるサイズの大きなテキストまたはバイナリ・オブジェクトを作成するために必要です。これは、Embedded SQL のホスト変数のサイズが 32,767 バイトに制限されているためです。

『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 言語の要素」>「式」を参照してください。

標準

- SQL—ISO/ANSI SQL 準拠。
- Sybase—サポートなし。Adaptive Server Enterprise では、変数はテーブルを指定しない **SELECT** 文を使用して割り当てます。これは Sybase IQ でもサポートされている Transact-SQL 構文です。**SET** 文は、Adaptive Server Enterprise でデータベース・オプションを設定するのに使用されます。

パーミッション

なし

参照：

- CREATE VARIABLE 文 (184 ページ)
- DROP VARIABLE 文 (222 ページ)

SET 文 [T-SQL]

Adaptive Server Enterprise に互換する方法でデータベース・オプションを設定します。

構文

```
SET option-name option-value
```

使用法

Sybase IQ のデータベース・オプションは、**SET OPTION** 文を使用して設定します。ただし、Sybase IQ は、互換性に対して特に便利なオプションを持つ Adaptive Server Enterprise の **SET** 文もサポートします。

表 15 : Transact-SQL の SET のオプション

オプション名	オプションの値
ANSINULL	**ON** | **OFF**
ANSI_PERMISSIONS	**ON** | **OFF**
CLOSE_ON_ENDTRANS	**ON**
QUOTED_IDENTIFIER	**ON** | **OFF**
ROWCOUNT	*integer*
STRING_RTRUNCATION	**ON** | **OFF**
TRANSACTION ISOLATION LEVEL	**0** | **1** | **2** | **3**

次のオプションは、Adaptive Server Enterprise の場合と同様、Sybase IQ で Transact-SQL の **SET** 文を使用して設定できます。

- SET ANSINULL { ON | OFF }—値を NULL と比較するデフォルトの動作は、Sybase IQ と Adaptive Server Enterprise で異なります。 **ANSINULL** を OFF に設定すると、NULL との比較に対して Transact-SQL との互換性が提供されます。
- SET ANSI_PERMISSIONS { ON | OFF }—カラム参照を含む **DELETE** の実行に必要なパーミッションに関して、Sybase IQ と Adaptive Server Enterprise ではデフォルト動作に違いがあります。 **ANSI_PERMISSIONS** を OFF に設定すると、**DELETE** のパーミッションに対して Transact-SQL との互換性が提供されます。
- SET CLOSE_ON_ENDTRANS { ON }—**CLOSE_ON_ENDTRANS** が ON の場合 (これがデフォルト値であり、許可されている唯一の値)、カーソルはトランザク

ションが終了すると閉じられます。このオプションが ON に設定されていると、**CLOSE_ON_ENDTRANS** は Transact-SQL と互換性のある動作になります。

- SET QUOTED_IDENTIFIER { ON | OFF }—二重引用符内の文字列を識別子 (ON) とリテラル文字列 (OFF) のどちらとして解釈するかを制御します。
- SET ROWCOUNT *integer*—Transact-SQL **ROWCOUNT** オプションは、カーソル用にフェッチされるロー数の上限 (整数) を指定します。カーソルを再配置してフェッチされたローにも適用されます。この上限を超えたフェッチには警告が発せられます。このオプション設定は、**OPEN** 要求に対して、カーソルがフェッチするロー数の推測値が返されるときに考慮されます。

注意： Sybase IQ は、*@@rowcount* グローバル変数をサポートします。 **SELECT** 文、**INSERT** 文、**DELETE** 文、**UPDATE** 文は、**ROWCOUNT** オプション値の影響を受けます。**ROWCOUNT** オプションは、カーソル・オペレーション、**IF** 文、テーブルまたはプロシージャの作成／削除には影響しません。

Sybase IQ では、**ROWCOUNT** が **dbisql** で表示可能なロー数より大きい場合、**dbisql** は追加のフェッチを行ってカーソルを再配置できます。 したがって、実際に表示されるロー数は、要求した数より少なくなることがあります。また、トランケーションの警告のために、ローが再度フェッチされる場合、カウントは正しくない場合があります。
値を 0 にすると、オプションをリセットし、すべてのローを取得します。

- SET STRING_RTRUNCATION { ON | OFF }—SQL 文字列データへの代入時にスペース以外の文字をトランケートする場合のデフォルト動作は、Sybase IQ と Adaptive Server Enterprise では異なります。 **STRING_RTRUNCATION** を ON に設定すると、Transact-SQL 互換の文字列比較が、16 進文字列 (バイナリ・データ型) の比較を含めて可能になります。
- SET TRANSACTION ISOLATION LEVEL { 0 | 1 | 2 | 3 }—現在の接続に対し、ロック独立性レベルを設定します。詳細については、『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」を参照してください。Adaptive Server Enterprise の場合、有効なオプションは 1、2、3 です。 Sybase IQ の場合、有効なオプションは 3 だけです。

さらに、次の **SET** 文は、互換性を考慮して Sybase IQ で許可されていますが、効果はありません。

- SET PREFETCH { ON | OFF }

標準

- SQL—ISO/ANSI SQL 文法の Transact-SQL 拡張。

- Sybase—Sybase IQ は Adaptive Server Enterprise のデータベース・オプションのサブセットをサポートしています。

パーミッション

なし

参照：
- SET OPTION 文 (356 ページ)

SET CONNECTION 文 [ESQL] [Interactive SQL]

アクティブなデータベース接続を変更します。

構文

```
SET CONNECTION [connection-name]
```

パラメータ

- **connection-name：** – 識別子、文字列、またはホスト変数

例

- **例 1 –** 次は、Embedded SQL の例です。

```
EXEC SQL SET CONNECTION :conn_name
```

- **例 2 – dbisql** から、現在の接続を接続名 "conn1" に設定します。

```
SET CONNECTION conn1
```

使用法

現在の接続状態を保存し、再びアクティブな接続になるときにこれを再開します。*connection-name* を省略し、名前のない接続がある場合は、この接続がアクティブな接続になります。

注意： カーソルを Embedded SQL でオープンするとき、カーソルを現在の接続と関連付けます。接続が変更されると、カーソル名にはアクセスできません。カーソルはアクティブなまま配置され、関連付けられている接続が再びアクティブになると、アクセスできるようになります。

標準

- SQL— **dbisql** の使用は ISO/ANSI SQL 文法のベンダ拡張です。Embedded SQL は上級レベル機能です。
- 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

参照：

- CONNECT 文 [ESQL] [Interactive SQL] (77 ページ)
- DISCONNECT 文 [Interactive SQL] (208 ページ)

SET DESCRIPTOR 文 [ESQL]

SQL 記述子領域の変数の記述や、記述子領域へのデータの格納を行います。

構文

```
              SET DESCRIPTOR
              descriptor-name
… { COUNT = { integer | hostvar }
| VALUE
              n
              assignment [, …] }
```

パラメータ

- **assignment：** – { { **TYPE** | **SCALE** | **PRECISION** | **LENGTH** | **INDICATOR** } = { *integer* | *hostvar* } | **DATA** = *hostvar* }

例

- **例 1 –** 「ALLOCATE DESCRIPTOR 文 [ESQL]」を参照してください。

使用法

SET...COUNT を使用すると、記述子領域内にある記述済み変数の数を設定できます。カウント対象の値は、記述子領域を割り付けたときに指定した変数の数を超えることはできません。

値 *n* は、代入の対象となる記述子領域内の変数を指定します。

SET...DATA の実行時に型検査を実行し、ホスト変数と記述子変数が同じデータ型を持つように保証します。

エラーが発生すると、エラー・コードが SQLCA に返されます。

標準

- SQL—ISO/ANSI SQL 準拠。
- 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

参照：

- ALLOCATE DESCRIPTOR 文 [ESQL] (5 ページ)
- DEALLOCATE DESCRIPTOR 文 [ESQL] (188 ページ)

SET OPTION 文

データベース・オプションを変更します。

構文

```
SET [ EXISTING ] [ TEMPORARY ] OPTION
… [ userid. | PUBLIC.]option-name = [ option-value ]
```

パラメータ

- **userid：** – 識別子、文字列、またはホスト変数
- **option-name：** – 識別子、文字列、またはホスト変数
- **option-value：** – ホスト変数 (インジケータ使用可)、文字列、識別子、または数値

例

- **例 1 – DATE_FORMAT** オプションを設定します。

  ```
  SET OPTION public.date_format = 'Mmm dd yyyy'
  ```

- **例 2 – WAIT_FOR_COMMIT** オプションを ON に設定します。

  ```
  SET OPTION wait_for_commit = 'on'
  ```

- **例 3 –** Embedded SQL の例を示します。

```
EXEC SQL SET OPTION :user.:option_name = :value;
EXEC SQL SET TEMPORARY OPTION Date_format = 'mm/dd/yyyy';
```

使用法

SET OPTION 文を使用すると、データベース・サーバの動作や Transact-SQL との互換性に影響を及ぼすオプションを変更できます。オプションの値を設定すると、すべてのユーザまたは個別のユーザの動作を一時的または永続的なスコープで変更できます。

オプションのクラスは次のとおりです。

- 一般的なデータベース・オプション
- Transact-SQL 互換性データベース・オプション

ユーザ ID または PUBLIC ユーザ ID を指定することにより、オプションの設定の対象が、個々のユーザであるか、*userid* で表されるユーザ・グループであるか、またはすべてのユーザがそのメンバであるユーザ・グループの PUBLIC ユーザ ID であるかどうかを決定できます。オプションがグループ・ユーザ ID に適用される場合、オプション設定はグループのメンバには継承されません。つまり、変更はグループ・ユーザ ID だけに適用されます。ユーザ・グループが指定されていない場合、**SET OPTION** 文の発行者である現在ログイン中のユーザ ID に対してオプション変更が適用されます。

たとえば、次の文は、オプション変更を PUBLIC ユーザ ID に適用します。

```
SET OPTION Public.login_mode = standard
```

DBA 権限を持つユーザのみが、PUBLIC ユーザ ID にオプションを設定する権限を持ちます。

Embedded SQL では、データベース・オプションが設定できるのは、一時的にすぎません。

PUBLIC ユーザ ID のオプション値を変更すると、値を設定していないすべてのユーザのオプション値が設定されます。そのオプションに PUBLIC ユーザ ID が設定されていないと、個々のユーザ ID のオプション値は設定されません。

DBA 権限を持っていない場合は、他のユーザのオプションを設定することはできません。

SET OPTION 文を使用して自分のユーザ ID の値を変更できます。自分以外のユーザ ID のオプション値を設定できるのは、DBA 権限を持っている場合のみです。

EXISTING キーワードを使用した場合、個別のユーザ ID のオプション値は、そのオプションにすでに PUBLIC ユーザ ID 設定が行われていないかぎり、設定できません。

SET OPTION 文に **TEMPORARY** キーワードを追加すると、変更の効果の継続期間を変更できます。**TEMPORARY** キーワードを付けなければ、オプションの変更は永久的です。**SET OPTION** を使用して明示的に変更されるまで変わりません。

個別のユーザ ID を使用して **SET TEMPORARY OPTION** 文を実行すると、そのユーザがデータベースにログインしている間、新しいオプション値が適用されます。

PUBLIC ユーザ ID を使用して **SET TEMPORARY OPTION** を使用すると、データベースの実行中はその変更が継続します。データベースが停止すると、PUBLIC ユーザ ID の **TEMPORARY** オプションはその永久値に戻ります。

PUBLIC ユーザ ID のオプションを、永続的ではなく一時的に設定するとセキュリティが向上します。たとえば、LOGIN_MODE オプションが使用可能な場合、データベースは実行中のシステムのログイン・セキュリティを使用します。このオプションを一時的に有効にすると、Windows NT ドメインのセキュリティに依存しているデータベースは、データベースが停止し、ローカル・マシンにコピーされるといった場合でも、危険にさらされることはありません。この場合、LOGIN_MODE オプションを一時的に有効にしてあると、オプションは永久値の Standard に戻ります。このモードでは、統合化ログインを使用できません。

option-value を省略すると、指定したオプション設定がデータベースから削除されます。これが各ユーザ固有のオプション設定の場合は、値は PUBLIC 設定に戻ります。**TEMPORARY** オプションを削除すると、その設定値は永久値に戻ります。

注意： 整数値を指定できるデータベース・オプションでは、小数の *option-value* 設定が常に整数値にトランケートされます。 たとえば、3.8 という値は 3 にトランケートされます。

option-value の最大長は、文字列を設定する場合は 127 バイトです。

警告！ カーソルからローをフェッチしている間のオプション設定変更は、予期しない動作が発生する可能性があるため、サポートされていません。たとえば、カーソルからフェッチしている間に DATE_FORMAT 設定を変えると、結果セットのロー同士で日付フォーマットが異なってしまいます。ローをフェッチしている間にオプション設定を変更しないでください。

各データベース・オプションの詳細については、「データベース・オプション」を参照してください。

標準

- 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。Sybase IQ は、Adaptive Server Enterprise の一部のオプションを、**SET** 文を使用してサポートしています。

パーミッション

ユーザ自身のオプションを設定するには、権限は必要ありません。別のユーザまたは PUBLIC に対してデータベース・オプションを設定するには、DBA 権限が必要です。

参照：
- データベース・オプション (383 ページ)

SET OPTION 文 [Interactive SQL]

Interactive SQL (**dbisql**) オプションを変更します。

構文

構文 1

```
SET [ TEMPORARY ] OPTION
… [ userid. | PUBLIC.]option-name = [ option-value ]
```

構文 2

```
SET PERMANENT
```

構文 3

```
SET
```

パラメータ

- **userid：** – 識別子、文字列、またはホスト変数
- **option-name：** – 識別子、文字列、またはホスト変数
- **option-value：** – ホスト変数 (インジケータ使用可)、文字列、識別子、または数値

使用法

SET PERMANENT (構文 2) は、現在の **dbisql** オプションをすべて SYSOPTION システム・テーブルに格納します。**dbisql** が現在のユーザ ID で起動されるたびに、これらの値が自動的に設定されます。

構文 3 を使用すると、現在のオプション設定がすべて表示されます。**dbisql** またはデータベース・サーバにテンポラリ・オプションがある場合は、それが表示されます。それ以外の場合、永久オプション設定が表示されます。

オプション設定時にオプション名を誤って入力すると、誤った名前が SYSOPTION テーブルに保存されます。誤って入力した名前を SYSOPTION テーブルから削除

するには、オプション PUBLIC でオプション名の後に等号を付け、値を指定しません。

```
SET OPTION PUBLIC.a_mistyped_name=;
```

参照：

- データベース・オプション (383 ページ)

SET SQLCA 文 [ESQL]

デフォルトのグローバル *sqlca* 以外の SQLCA を使用するように、SQL プリプロセッサに通知します。

構文

```
SET
            SQLCA sqlca
```

パラメータ

- **sqlca：** – 識別子または文字列

例

- **例 1 –** Windows DLL の次の関数を表示します。DLL を使用する各アプリケーションは独自の SQLCA を持ちます。

```
an_sql_code FAR PASCAL ExecuteSQL( an_application *app, char
*com )
{
    EXEC SQL BEGIN DECLARE SECTION;
    char *sqlcommand;
    EXEC SQL END DECLARE SECTION;
    EXEC SQL SET SQLCA "&app->.sqlca";
    sqlcommand = com;
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    EXEC SQL EXECUTE IMMEDIATE :sqlcommand;
return( SQLCODE );
}
```

使用法

Embedded SQL 文ごとに、現在の SQLCA ポインタを暗黙のうちにデータベース・インタフェース・ライブラリに渡します。C 言語ソース・ファイルにおいて、この文の後に記述されているすべての Embedded SQL 文は、新しい SQLCA を使います。この文を使用するのは、再入可能コードを記述するときだけです。*sqlca* は、

ローカル変数を参照しなければなりません。グローバルまたはモジュール静的変数は別のスレッドによる修正を受けます。

『SQL Anywhere サーバー – プログラミング』の「Embedded SQL」>「SQLCA (SQL Communication Area)」を参照してください。

注意： このリファレンスは SQL Anywhere マニュアルにリンクされています。

標準

- SQL—ISO/ANSI SQL 文法のベンダ拡張。
- Sybase—Open Client/Open Server でサポートされていません。

パーミッション

なし

SIGNAL 文

例外条件の信号を発信します。

構文

```
SIGNAL
exception-name
```

使用法

SIGNAL を使用すると、例外を発生させることができます。例外が処理される方法の詳細については、『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。

標準

- SQL—ISO/ANSI SQL 準拠。
- Sybase—**SIGNAL** は、Adaptive Server Enterprise ではサポートされていません。

パーミッション

なし

参照：

- BEGIN … END 文 (60 ページ)

- RESIGNAL 文 (320 ページ)

START DATABASE 文 [Interactive SQL]

指定したデータベース・サーバ上でデータベースを起動します。

構文

```
START DATABASE  database-file
… [ AS  database-name ]
… [ ON  engine-name ]
… [ AUTOSTOP { YES | NO } ]
… [ KEY  key ]
```

例

- **例 1 –** UNIX システムで、現在のサーバからデータベース・ファイル `/s1/sybase/sample_2.db` を起動します。

  ```
  START DATABASE '/s1/sybase/sample_2.db'
  ```

- **例 2 –** Windows システムで、データベース・ファイル `c:¥sybase¥sample_2.db` を `sam2` という名前で、サーバ `eng1` から起動します。

  ```
  START DATABASE 'c:¥sybase¥sample_2.db'
  AS sam2
  ON eng1
  ```

使用法

データベース・サーバが作動中である必要があります。ファイルが現在のディレクトリにない場合は、データベース・ファイルに対してフル・パスを指定してください。

START DATABASE 文は、指定したデータベースに **dbisql** を接続しません。接続するには、**CONNECT** 文を発行する必要があります。

database-name を指定しない場合、デフォルト名がデータベースに割り当てられます。このデフォルト名は、データベース・ファイルのルートです。たとえば、ファイル `c:¥sybase¥IQ_15¥demo¥iqdemo.db` 内のデータベースには、`iqdemo` というデフォルト名が付けられます。

engine-name を指定しない場合は、デフォルトのデータベース・サーバが使用されます。デフォルトのデータベース・サーバは、現在作動中のサーバの中で最初に起動したサーバです。

AUTOSTOP 句に対するデフォルトの設定は YES です。**AUTOSTOP** を YES に設定すると、最後の接続が削除されるときに、データベースが自動的にアンロードされます。**AUTOSTOP** を NO に設定すると、データベースはアンロードされません。

データベースが高度に暗号化されている場合は、**KEY** 句を使用して KEY 値 (パスワード) を入力してください。

特定の Sybase IQ データベース・サーバからは、1 つのデータベースのみ起動することをおすすめします。

標準

- SQL—ISO/ANSI SQL 文法のベンダ拡張。
- Sybase—なし。

パーミッション

DBA 権限が必要です。

START ENGINE 文 [Interactive SQL]

データベース・サーバを起動します。

構文

```
START ENGINE AS  engine-name [ STARTLINE  command-string ]
```

例

- **例 1 –** `eng1` という名前のデータベース・サーバを、1 つもデータベースを起動しない状態で起動します。

  ```
  START ENGINE AS eng1
  ```

- **例 2 –** 次の例は、**STARTLINE** 句の使用法を示します。

  ```
  START ENGINE AS eng1 STARTLINE 'start_iq -c 8096'
  ```

使用法

サーバに対してオプションのセットを指定するには、コマンド文字列に **STARTLINE** キーワードを付けます。

有効なコマンド文字列については、『ユーティリティ・ガイド』の「start_iq データベース・サーバ起動ユーティリティ」に記載されているデータベース・サーバのコマンド・ラインの説明を参照してください。

注意： Sybase IQ を適切に動作させるには、サーバ・オプションをいくつか設定する必要があります。 正しいオプションのセットを使用するため、サーバを起動する際は、Sybase Central を使用するか、設定ファイルと **start_iq** コマンドを使用することをおすすめします。

標準

- SQL—ISO/ANSI SQL 文法のベンダ拡張。
- Sybase—なし。

パーミッション

なし

参照：

- STOP ENGINE 文 [Interactive SQL] (366 ページ)

START JAVA 文

Java VM を起動します。

構文

```
START EXTERNAL ENVIRONMENT JAVA
```

例

- **例 1 –** Java VM を起動します。

  ```
  START EXTERNAL ENVIRONMENT JAVA
  ```

使用法

START EXTERNAL ENVIRONMENT JAVA を使用して VM を適宜ロードし、ユーザが Java の機能を使い始めるときに、VM ロード中の一時停止が発生しないようにします。

標準

- SQL—ISO/ANSI SQL 文法のベンダ拡張。
- Sybase—なし。

パーミッション

DBA 権限が必要です。

参照：

- STOP JAVA 文 (366 ページ)

STOP DATABASE 文 [Interactive SQL]

指定したデータベース・サーバ上のデータベースを停止します。

構文

```
STOP DATABASE  database-name
… [ ON  engine-name ]
… [ UNCONDITIONALLY ]
```

例

- **例 1 –** デフォルト・サーバ上のデータベース `sample` を停止します。

```
STOP DATABASE sample
```

使用法

engine-name を指定しないと、すべての作動中のエンジンについて、指定したデータベースが検索されます。

database-name は、データベース起動時の **-n** パラメータ、または **DBN** (**DatabaseName**) 接続パラメータで指定した名前です。この名前は通常、カタログ・ストアを保持するデータベース・ファイルのファイル名から `.db` 拡張子を除いたものですが、ユーザ定義の名前も使用できます。

UNCONDITIONALLY を指定すると、データベースはデータベースへの接続がある場合でも停止します。**UNCONDITIONALLY** を指定しないと、接続がある場合にはデータベースは停止しません。

標準

- SQL—ISO/ANSI SQL 文法のベンダ拡張。
- Sybase—なし。

パーミッション

DBA 権限が必要です。

参照：

- DISCONNECT 文 [Interactive SQL] (208 ページ)
- START DATABASE 文 [Interactive SQL] (362 ページ)

STOP ENGINE 文 [Interactive SQL]

データベース・サーバを停止します。

構文

STOP ENGINE *engine-name* [**UNCONDITIONALLY**]

例

- **例 1 –** `sample` という名前のデータベース・サーバを停止します。

```
STOP ENGINE sample
```

使用法

UNCONDITIONALLY を指定すると、データベース・サーバはサーバへの接続がある場合でも停止します。**UNCONDITIONALLY** を指定しないと、接続がある場合にはデータベース・サーバは停止しません。

標準

- SQL—ISO/ANSI SQL 文法のベンダ拡張。
- Sybase—なし。

パーミッション

なし

参照：

- START ENGINE 文 [Interactive SQL] (363 ページ)

STOP JAVA 文

Java VM に関連付けられているリソースを解放します。

構文

STOP EXTERNAL ENVIRONMENT JAVA

使用法

STOP EXTERNAL ENVIRONMENT JAVA の主な目的は、システム・リソースを経済的に使用することです。

標準

- SQL—ISO/ANSI SQL 文法のベンダ拡張。
- Sybase—なし。

パーミッション

DBA 権限

参照：

- START JAVA 文 (364 ページ)

SYNCHRONIZE JOIN INDEX 文

1 つまたは複数のジョイン・インデックスを、そのベース・テーブルが更新された後に同期します。

構文

SYNCHRONIZE JOIN INDEX [*join-index-name* [, *join-index-name*]…]

例

- **例 1 –** ジョイン・インデックスの `emp_dept_join1` と `emp_dept_join2` を同期します。

  ```
  SYNCHRONIZE JOIN INDEX emp_dept_join1, emp_dept_join2
  ```

使用法

ジョイン・インデックスに関連するベース・テーブルが更新されると、Sybase IQ はそのジョイン・インデックスを使用不可と通知します。 以前にジョイン・インデックスを利用したクエリは、代わりにアドホック・ジョインを実行しますが、パフォーマンスに影響する可能性があります。 **SYNCHRONIZE JOIN INDEX** コマンドを使用すると、ジョイン・インデックスを最新のものにでき、クエリでの使用が可能になります。

注意： ジョイン・インデックスは 2 つのテーブル・カラム間で「1 対多」関係 (つまりプライマリ・キー対外部キー) を定義します。「1」側 (プライマリ・キー) のカラムへの挿入で 1 つまたは複数の重複値が発生する場合、そのジョイン・イン

デックスは無効になり、同期できません。**SYNCHRONIZE JOIN INDEX** がジョイン・インデックスを再び有効にできるようにするには、先に重複値を含むローを削除する必要があります。

ジョインを構成するベース・テーブルのサイズによっては、ジョイン・インデックスの同期に時間がかかる可能性があります。 このコマンドを使用する時期はユーザ側で判断してください。 夜間または週末など、システムの負荷が比較的軽いと思われる時間に、バッチ処理としてスケジュールすることもできます。ジョイン・インデックスをできるだけ早く使用するには、Sybase IQ が一連の挿入や削除をコミットした直後に実行します。 ただし、ジョイン・インデックスの更新時期はテーブルへの更新の順序によって決定されるため、INSERT や DELETE を 1 つ実行するごとにジョイン・インデックスの同期は行わないでください。

SYNCHRONIZE JOIN INDEX を使用すると、複数の *join-index-name* をカンマで区切って指定できます。各ジョイン・インデックスの所有者であるか、または DBA である必要があります。 *join-index-name* を指定しない場合、Sybase IQ はユーザが所有するすべてのジョイン・インデックス (ユーザが DBA である場合はデータベース内にあるすべてのジョイン・インデックス) を同期します。これにより、システムのパフォーマンスが低下する場合があります。

標準

- SQL—ISO/ANSI SQL 文法のベンダ拡張。
- Sybase—なし。

パーミッション

ジョイン・インデックスの所有者であるか、DBA 権限を持っている必要があります。

参照：
- CREATE JOIN INDEX 文 (128 ページ)

TRIGGER EVENT 文

指定されたイベントをトリガします。イベントには、イベント・トリガ用に定義されたイベントと、スケジュールされたイベントがあります。

構文

```
TRIGGER EVENT
event-name [ ( parm = value, ... ) ]
```

使用法

トリガ条件またはスケジュールには、**CREATE EVENT** 文でアクションを関連付けます。**TRIGGER EVENT** を使用すると、スケジュールされた時間でなくても、トリガ条件が発生しなくても、イベント・ハンドラを強制的に実行できます。**TRIGGER EVENT** は、無効なイベント・ハンドラを実行しません。

トリガ条件によってイベント・ハンドラが実行されるとき、データベース・サーバは event_parameter 関数を使用してコンテキスト情報をイベント・ハンドラに渡すことができます。**TRIGGER EVENT** では、これらのパラメータを明示的に指定して、イベント・ハンドラのコンテキストをシミュレートできるようにしています。

イベントをトリガするときは、イベント名を指定します。イベント名は、システム・テーブル SYSEVENT をクエリして、一覧表示できます。次に例を示します。

```
SELECT event_id, event_name FROM SYS.SYSEVENT
```

『システム管理ガイド：第 2 巻』の「スケジューリングとイベント処理によるタスクの自動化」を参照してください。

パーミッション

DBA 権限が必要です。

参照：

- ALTER EVENT 文 (15 ページ)
- CREATE EVENT 文 (99 ページ)

TRUNCATE TABLE 文

テーブルからすべてのローを削除します。ただし、テーブル定義は削除しません。

構文

構文 1

```
TRUNCATE TABLE [ owner.]table-name
```

構文 2

```
TRUNCATE TABLE [ owner .]table [ PARTITION partition-name ]
```

例

- **例 1 –** Sale テーブルからすべてのローを削除します。

```
TRUNCATE TABLE Sale
```

使用法

WHERE 句のない **DELETE** 文と同じです。ただし、**TRUNCATE TABLE** 文の場合は、DELETE 文と違い、個々のローの削除がトランザクション・ログに記録されません。**TRUNCATE TABLE** 文の後、テーブル構造体とインデックスのすべては、**DROP TABLE** 文が発行されるまで存在し続けます。カラム定義と制約はそのまま残り、パーミッションは有効なままです。

TRUNCATE TABLE 文は、データ定義文のように 1 つの文としてトランザクション・ログに記録されます。それぞれの削除されたローは、トランザクション・ログには記録されません。

パーティション句は、トランケートするパーティションを指定します。他のパーティション上のデータには影響はありません。

『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」を参照してください。

標準

- SQL—ISO/ANSI SQL 文法のベンダ拡張。
- Sybase—Adaptive Server Enterprise でサポートされています。

パーミッション

- テーブルの所有者であるか、DBA 権限が必要です。
- テンポラリ・テーブルとベース・テーブルの両方で、他のユーザにテーブルへの読み込みアクセスがある間でも **TRUNCATE TABLE** を実行できます。この動作は、ベース・テーブルのトランケートに排他的アクセスを必要とする SQL Anywhere とは異なります。 他のユーザに読み込みアクセスがある間も **TRUNCATE TABLE** を実行できるのは、Sybase IQ にテーブルのバージョン管理機能があるためです。ただし、これらのユーザが参照するテーブルのバージョンは、読み込み書き込みのトランザクションがいつコミットされるかによって変わります。

参照：

- DELETE 文 (201 ページ)

UNION 演算

2 つ以上の SELECT 文の結果を結合します。

構文

```
select-without-order-by
… UNION [ ALL ] select-without-order-by
… [ UNION [ ALL ] select-without-order-by ]…
… [ ORDER
            BY integer [ ASC | DESC ] [, …] ]
```

例

- **例 1 –** 従業員と顧客のそれぞれの姓すべてをリストします。

```
SELECT Surname
FROM Employees
UNION
SELECT Surname
FROM Customers
```

使用法

複数の **SELECT** 文の結果は、**UNION** を使用して 1 つの大きな結果へと結合できます。コンポーネントの **SELECT** 文には、それぞれの select リストに同じ数の項目を指定します。各文には **ORDER BY** 句を含めることはできません。「FROM 句」を参照してください。

UNION ALL の結果は、複数のコンポーネントの **SELECT** 文を結合した結果です。**UNION** の結果は **UNION ALL** と同じですが、重複ローが削除されている点が異なります。重複ローを削除するには余分な処理が必要なため、可能であれば **UNION** の代わりに **UNION ALL** を使用してください。

2 つの select リスト中の対応する項目が異なるデータ型の場合、Sybase IQ は結果の中から対応するカラムのデータ型を選択し、各コンポーネント **SELECT** 文のカラムを自動的にそれぞれ変換します。

ORDER BY を使用する場合、順序を表すには整数だけを使用してください。これらの整数はソートされるカラムの位置を指定します。

表示されるカラム名は、最初の **SELECT** 文に対して表示されるカラム名と同じです。

注意： **SELECT** 文に定数値と UNION ALL ビューが含まれているが、**FROM** 句が省略されている場合は、エラーを防ぐために `iq_dummy` を使用します。詳細については、「FROM 句」を参照してください。

標準

- SQL—ISO/ANSI SQL 準拠。
- Sybase—Adaptive Server Enterprise でサポートされます。**COMPUTE** 句もサポートされます。

パーミッション

コンポーネントの **SELECT** 文のそれぞれに対して、SELECT パーミッションが必要です。

参照：
- FROM 句 (237 ページ)
- SELECT 文 (339 ページ)

UPDATE 文

単一のテーブル、または単一のテーブルしか含まないビューの既存のローを修正します。

構文

```
UPDATE table
... SET [column-name  = expression, …
... [ FROM table-expression, ]
... [ WHERE search-condition ]
... [ ORDER
            BY expression [ ASC | DESC ] , …]
```

```
FROM table-expression
```

パラメータ

- **table-expression：** – *table-spec* | *table-expression join-type table-spec* [**ON***condition*] | *table-expression*, …

例

- **例 1 –** 従業員 Philip Chin (従業員 129) を販売部からマーケティング部に異動する例を示します。

  ```
  UPDATE Employees
  SET DepartmentID = 400
  WHERE EmployeeID = 129;
  ```
- **例 2 –** マーケティング部 (400) はボーナスを各従業員の基本給の 4% から 6% に増加します。

```
UPDATE Employees
SET bonus = base * 6/100
WHERE DepartmentID =400;
```

- **例 3 –** 従業員全員の給与を、各部のボーナス分だけ昇給します。

```
UPDATE Employees
SET emp.Salary = emp.Salary + dept.bonus
FROM Employees emp, Departments dept
WHERE emp.DepartmentID = dept.DepartmentID;
```

- **例 4 –** 先ほどの昇給を別の方法で行います。

```
UPDATE Employees
SET emp.salary = emp.salary + dept.bonus
FROM Employees emp JOIN Departments dept
ON emp.DepartmentID = dept.DepartmentID;
```

使用法

UPDATE で使用するテーブルは、ベース・テーブルまたはグローバル・テンポラリ・テーブルの場合があります。

注意： ベース・テーブルはジョイン・インデックスの一部であってはなりません。

それぞれ指定したカラムを、等号の右側の式の値に設定します。*column-name* も式の中で使用できます。この場合、古い値を使用します。

FROM 句にジョイン条件を使って複数のテーブルを指定し、指定されたすべてのテーブルのすべてのカラムを、ジョイン条件および **WHERE** 条件でフィルタして返すことができます。

FROM 句で間違ったジョイン条件を使用すると、予測できない結果が返されます。**FROM** 句に 1 対多のジョインを指定し、SET 句でジョインの「多」の側のセルを参照した場合、セルは最初に選択された値で更新されます。言い換えれば、ジョイン条件によって、テーブル内でロー ID ごとに複数のローが更新される場合、更新後の結果は最初に返されたローの値になります。次に例を示します。

```
UPDATE T1
SET T1.c2 = T2.c2
FROM T1 JOIN TO T2
ON T1.c1 = T2.c1
```

テーブル T2 に T2.c1 の値が同じであるローが複数存在する場合、結果は次のようになります。

T2.c1	T2.c2	T2.c3
1	4	3
1	8	1
1	6	4

```
1                    5                    2
```

ORDER BY 句を指定しなければ、`T1.c2` は 4、6、8、9 のいずれかになります。

- `ORDER BY T2.c3` を指定すると、`T1.c2` は 8 に更新されます。
- `ORDER BY T2.c3 DESC` を指定すると、`T1.c2` は 6 に更新されます。

Sybase IQ は、更新を行うテーブルが外部ジョインの null を補う側にある場合は、**UPDATE** 文を拒否します。 具体的には、次のようになります。

- 左外部ジョインでは、ジョインを行うカラムに対して、ジョインの左側のテーブルのローがすべて揃っていなければならない。
- 右外部ジョインでは、ジョインを行うカラムに対して、ジョインの右側のテーブルのローがすべて揃っていなければならない。
- フル外部ジョインでは、ジョインを行うカラムに対して、両方のテーブルのローがすべて揃っていなければならない。

たとえば、次の文では、テーブル `T1` は左外部ジョインの左側にあるため、すべてのローが揃っていなければなりません。

```
UPDATE T1
SET T1.c2 = T2.c4
FROM T1 LEFT OUTER JOIN T2
ON T1.rowid = T2.rowid
```

通常、ローを更新する順序は重要ではありません。ただし、**NUMBER(*)** 関数と一緒に使って、ある指定された順序でローの中の数字を増加させる場合に、順序付けが役に立ちます。**NUMBER(*)** 関数を使用していない場合、**ORDER BY** 句は使用しないでください。**UPDATE** 文では ORDER BY 句がないほうがパフォーマンスが良いからです。

UPDATE 文で **NUMBER(*)** 関数を **SET** 句に使用し、**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、数値は連続的には増加しません。**NUMBER(*)** 関数の詳細については、『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 関数」>「アルファベット順の関数リスト」>「NULLIF 関数 [その他]」を参照してください。

ORDER BY 句を使用すると、**FROM** 句で複数のテーブルがジョインされている場合に **UPDATE** が返す結果を制御できます。

UPDATE に検索条件が指定されている場合、Sybase IQ は **ORDER BY** 句を無視し、構文が ANSI 構文で無効であることを示すエラーを返します。

WHERE 句を指定しない場合、すべてのローが更新されます。 **WHERE** 句を指定すると、Sybase IQ は検索条件を満たすローだけを更新します。

SET 句の左側には、ベース・テーブルのカラムを指定してください。

ビューを定義する **SELECT** 文の中に **GROUP BY** 句や集合関数がないか、または **UNION** 演算を伴わない場合は、ビューを更新できます。ビューに含めるテーブルは 1 つだけです。

テーブルに挿入した文字列は、データベースが大文字と小文字を区別するかどうかに関係なく、常に入力された大文字と小文字がそのまま格納されます。このため、文字列 **Value** で更新した文字データ型カラムは、常に大文字の V と残りの文字が小文字でデータベースに格納されます。**SELECT** 文は、**Value** として文字列を返します。ただし、データベースが大文字と小文字を区別しない場合は、すべての比較で **Value**、**value**、**VALUE** は同じものとして扱われます。IQ サーバは大文字と小文字を任意に組み合わせた結果を返すので、大文字と小文字を区別しない (**CASE IGNORE**) データベースで、大文字と小文字を区別する結果は期待できません。さらに、シングルカラム・プライマリ・キーにすでにエントリ **Value** がある場合、**value** の **INSERT** はできません。プライマリ・キーがユニークでなくなるからです。

更新によってチェック制約に違反が起こる場合は、文全体がロールバックされます。

Sybase IQ は、**SET** 句内でのスカラ・サブクエリをサポートします。次に例を示します。

```
UPDATE r
SET r.o= (SELECT MAX(t.o)
FROM t ... WHERE t.y = r.y),
r.s= (SELECT SUM(x.s)
FROM x ...
WHERE x.x = r.x)
WHERE r.a = 10
```

Sybase IQ は、**UPDATE** 文でカラムの DEFAULT 値をサポートします。カラムに DEFAULT 値がある場合、明示的にそのカラムの値を修正しないすべての **UPDATE** 文で、カラムの値としてこの DEFAULT 値が使用されます。

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。

別の種類の DEFAULT カラムである IDENTITY/AUTOINCREMENT カラムの更新の詳細については、「CREATE TABLE 文」 を参照してください。

標準

- SQL—ISO/ANSI SQL 文法のベンダ拡張。
- Sybase—次の例外を除いて、IQ の **UPDATE** 文の構文は、通常、Adaptive Server Enterprise の **UPDATE** 文の構文 1 と互換性があります。 Sybase IQ では、**FROM** 句にジョイン条件を使用して、複数のテーブルを指定できます。

リモート・テーブルの更新には、CIS でサポートされる Sybase IQ 構文しか使用できません。詳細については、『システム管理ガイド： 第 2 巻』の「リモート・データへのアクセス」と『システム管理ガイド： 第 2 巻』の「リモート・データ・アクセス用のサーバ・クラス」を参照してください。

パーミッション

修正対象のカラムに対する UPDATE パーミッションが必要です。

参照：

- CREATE TABLE 文 (166 ページ)

UPDATE (位置付け) 文 [ESQL] [SP]

カーソルの現在の位置にあるデータを更新します。

構文

```
UPDATE table-list SET set-item, …
WHERE CURRENT OF cursor-name
```

パラメータ

- **cursor-name：** – 識別子 | ホスト変数
- **set-item：** – *column-name* [.*field-name…*] = *scalar-value*

例

- **例 1 –** 次は、**WHERE CURRENT OF** にカーソル名を指定した **UPDATE** 文の例です。

```
UPDATE Employees SET surname = 'Jones'
WHERE CURRENT OF emp_cursor
```

使用法

この形式の **UPDATE** 文は、指定されたカーソルの現在のローを更新します。現在のローとは、直前にカーソルから正しくフェッチされたローのことです。カーソルに対する直前の処理が、位置付け **DELETE** 文であってはなりません。

SET—*set-item* で参照されるカラムは、更新を行うベース・テーブルのカラムであることが必要です。エイリアスを参照したり、ほかのテーブルやビューのカラムを参照することはできません。更新を行うテーブルで、カーソルに相関名が指定されている場合は、**SET** 句にも相関名を使用してください。**SET** 句の右側の式で

は、クエリの **SELECT** 句のカラム、定数、変数、式を参照できます。*set-item* の式に関数や式を含めることはできません。

指定されたクエリの現在のローの位置にあるローに対して、要求されたカラムに指定の値が設定されます。カラムは、指定の開いているカーソルの select リストに指定されていなくてはなりません。

位置付け **UPDATE** 文による変更は、カーソル結果セット内で確認できます。ただし、クライアント・サイドのキャッシュによって、これらの変更の確認が妨げられた場合を除きます。ローが更新され、開いているカーソルの **WHERE** 句の要件を満たさなくなった場合でも、そのローは可視のままです。

WHERE CURRENT OF 句に **ORDER BY** を使用することはおすすめしません。**ORDER BY** を指定したカラムも更新可能ですが、結果セットが再度順序付けされることはありません。そのため、結果が (間違った順番でフェッチされたように見えるため) 正しくないように見えます。

Sybase IQ が **CREATE VIEW... WITH CHECK OPTION** をサポートしないため、位置付け **UPDATE** もこのオプションをサポートしません。 **WITH CHECK OPTION** は、ビューに表示されないローを作成するような更新を許可しません。

位置付け **UPDATE** で、ロー ID カラムを更新することはできません。

Sybase IQ では、結果セット内で同じローを繰り返し更新できます。

『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「システム・プロシージャ」>「システム・ストアド・プロシージャ」>「sp_iqcursorinfo プロシージャ」を参照してください。

標準

- `The range of cursors that can be updated may contain vendor extensions to ISO/ANSI SQL grammar if the ANSI_UPDATE_CONSTRAINTS option is set to OFF.`
- Embedded SQL の使用は Open Client/Open Server でサポートされ、プロシージャとトリガの使用は SQL Anywhere でサポートされています。

パーミッション

修正対象のカラムに対する UPDATE パーミッションが必要です。

参照：

- DECLARE CURSOR 文 [ESQL] [SP] (191 ページ)
- DELETE 文 (201 ページ)
- DELETE (位置付け) 文 [ESQL] [SP] (203 ページ)

- UPDATE 文 (372 ページ)

WAITFOR 文

この文は、指定した時間が経過するか特定の時刻になるまで、現在のコネクションの処理を遅らせます。

構文

```
WAITFOR {
DELAY time | TIME time }
[ CHECK EVERY integer }
[ AFTER MESSAGE BREAK ]
```

パラメータ

- **time：** – 文字列

例

- **例 1 –** 次は、3 秒間待機する例です。

  ```
  WAITFOR DELAY '00:00:03'
  ```

- **例 2 –** 次は、0.5 秒間 (500 ミリ秒) 待機する例です。

  ```
  WAITFOR DELAY '00:00:00:500'
  ```

- **例 3 –** 次は、午後 8 時まで待機する例です。

  ```
  WAITFOR TIME '20:00'
  ```

使用法

WAITFOR 文は、定期的 (デフォルトでは 5 秒おき) に起動して、キャンセルされたかどうか、またはメッセージが受信されたかどうかをチェックします。いずれの処理も確認されなかった場合は、文は待機を継続します。

DELAY を使用すると、処理は特定の期間だけ中断されます。**TIME** を指定すると、サーバの時間が指定の時間になるまで処理がサスペンドします。

現在のサーバの時間が指定の時間を過ぎている場合は、次の日のその時間まで処理がサスペンドします。

WAITFOR は、次の文と同じ目的で使用できます。顧客がデータベースで Java を無効にすることを選択した場合に利用すると便利です。

```
call java.lang.Thread.sleep( <time_to_wait_in_millisecs> )
```

通常は、**WAITFOR TIME** を使用するよりも、スケジュールされたイベントを使用することをおすすめします。これは、スケジュールされたイベントは、専用接続で実行されるためです。

CHECK EVERY 句—このオプション句は、**WAITFOR** 文が起動する頻度を制御します。デフォルトでは、**WAITFOR** は 5 秒ごとに起動します。値はミリ秒単位であり、最小値は 250 ミリ秒です。

AFTER MESSAGE BREAK 句—**WAITFOR** 文を使用して、別の接続からのメッセージを待つことができます。ほとんどの場合、メッセージが受信されると、**WAITFOR** 文を実行したアプリケーションに転送され、**WAITFOR** 文は待機を継続します。**AFTER MESSAGE BREAK** 句が指定されている場合、別の接続からのメッセージが受信されると、**WAITFOR** 文が完了します。メッセージ・テキストはアプリケーションに転送されませんが、MessageReceived 接続プロパティの値を取得してアクセスできます。

関連する動作

- この文の実装では、待機中にワーカ・スレッドが使用されます。このため、サーバのコマンド・ライン・オプション **-gn** で指定したスレッドの 1 つが使用できなくなります。

標準

- SQL—ISO/ANSI SQL 文法のベンダ拡張。
- Sybase—この文は Adaptive Server Enterprise でも実装されています。

パーミッション

なし

参照：

- CREATE EVENT 文 (99 ページ)

WHENEVER 文 [ESQL]

Embedded SQL プログラム内でのエラー処理を指定します。

構文

```
                WHENEVER
{ SQLERROR | SQLWARNING | NOTFOUND }
… { GOTO
                label | STOP | CONTINUE | C code; }
```

パラメータ

- **label：** – 識別子

例

- **例 1 –**

```
EXEC SQL WHENEVER NOTFOUND GOTO done;
```

- **例 2 –**

```
EXEC SQL WHENEVER SQLERROR
    {
        PrintError( &sqlca );
        return( FALSE );
    };
```

使用法

WHENEVER は Embedded SQL C プログラム内の任意の位置に入れることができます。この文はコードを生成しません。プリプロセッサは、それぞれの継続する SQL 文の後にコードを生成します。すべての Embedded SQL 文では、**WHENEVER** 文のソース行から同じエラー条件の次の **WHENEVER** 文、またはソース・ファイルの最後まで、エラー・アクションは有効です。

デフォルト・アクションは **CONTINUE** です。

WHENEVER は、単純なプログラムの中で便利に使用できます。ほとんどの場合、SQLCA (SQLCODE) の `sqlcode` フィールドを直接検査するのがエラー条件を検査する最も簡単な方法です。この場合、**WHENEVER** は使用しません。**WHENEVER** 文が実行されると、プリプロセッサはそれぞれの文の後に *if (SQLCODE)* テストを生成します。

注意： エラー条件は、文がいつ実行されたかではなく、C 言語ソース・ファイルの位置に基づいて効力を持ちます。

標準

- SQL—ISO/ANSI SQL 文法のベンダ拡張。
- 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

WHILE 文 [T-SQL]

文または複合文を繰り返し実行します。

構文

```
WHILE expression
... statement
```

例

- **例 1 –**

```
WHILE (SELECT AVG(unit_price) FROM Products) < 30
BEGIN
    DELETE FROM Products
    WHERE UnitPrice = MAX(UnitPrice)
    IF ( SELECT MAX(UnitPrice) FROM Products ) < 50
        BREAK
END
```

BREAK 文は、最も高い製品の価格が 50 ドル未満の場合、**WHILE** ループをブレイクします。そうでない場合、ループは平均価格が 30 ドルを超えるまで継続します。

使用法

WHILE 条件は、単一の SQL 文と、**BEGIN** と **END** のキーワードで囲まれた複合文にまとめられている SQL 文にのみ影響します。

複合文中での文の実行は、**BREAK** 文と **CONTINUE** 文で制御できます。**BREAK** 文はループを終了し、ループの最後を示す **END** キーワードの後から実行が再開されます。**CONTINUE** 文は、**CONTINUE** の後の文をすべて省略して **WHILE** ループを再開します。

標準

- SQL—ISO/ANSI SQL 文法の Transact-SQL 拡張。
- Sybase—Adaptive Server Enterprise でサポートされています。

パーミッション

なし

参照：

- BEGIN … END 文 (60 ページ)

データベース・オプション

データベース・オプションと Interactive SQL オプションは、データベースの動作をカスタマイズしたり、変更したりします。Sybase IQ のデータベース・オプションは、 一般的なオプション、Transact-SQL 互換性オプション、Interactive SQL オプションの 3 つに分類されます。

データベース・オプションの概要

データベース・オプションは、互換性、エラー処理、同時実行性などさまざまな面からデータベースの動作を制御します。

たとえば、次の目的でデータベース・オプションを使用できます。

- 互換性 – 使用している Sybase IQ データベースがどの程度 Adaptive Server Enterprise と同様の操作をするか、また SQL92 に準拠しない SQL に対してエラーを発生させるかどうかを制御できます。
- エラー処理 – 0 で割ろうとしたとき、またはオーバフロー・エラーなどのエラーが発生したときの処理を制御できます。
- 同時実行性とトランザクション – 同時実行性の程度と COMMIT 動作の詳細をオプションを使用して制御できます。

SET OPTION 文を使用してオプションを設定します。一般的な構文は次のとおりです。

```
SET [ EXISTING ] [ TEMPORARY ] OPTION
... [ userid. | PUBLIC. ]option-name = [ option-value ]
```

ユーザ ID またはグループ名を指定し、そのユーザまたはグループのみにオプションを設定します。すべてのユーザは PUBLIC グループに属します。ユーザ ID またはグループを指定しない場合、**SET OPTION** 文を発行した、現在ログオンしているユーザ ID にオプション変更が適用されます。

たとえば、次の文は、すべてのユーザが属するユーザ・グループである PUBLIC ユーザ ID に変更を適用します。

```
SET OPTION Public.login_mode = standard
```

注意： ユーザまたはグループを指定せずにオプションを TEMPORARY に設定した場合、新しいオプション値は、文を発行した現在ログオン中のユーザ ID にのみ、その接続の間だけ適用されます。PUBLIC グループに対してオプションを TEMPORARY に設定した場合、その変更はデータベースが実行されているかぎり維

持されます。データベースを停止すると、PUBLIC グループに対する TEMPORARY オプションは、その永久値に戻ります。

TEMPORARY キーワードを発行せずにオプションを設定した場合、新しいオプション値は、文を発行したユーザまたはグループの永久値となります。

テンポラリ・オプション値と永久オプション値の詳細については、「データベース・オプションのスコープと継続期間」、「temporary オプション」、「SET OPTION 文」を参照してください。

option-value の最大長は、文字列を設定する場合は 127 バイトです。

注意： 整数値を指定できるデータベース・オプションでは、小数の *option-value* 設定が常に整数値にトランケートされます。 たとえば、3.8 という値は 3 にトランケートされます。

警告！ ローをフェッチしている間にオプション設定を変更しないでください。

参照：

- データベース・オプションのスコープと継続期間 (385 ページ)
- temporary オプション (386 ページ)
- SET OPTION 文 (356 ページ)

現在のオプション設定

オプション設定のリストまたは個々のオプションの値は、**sp_iqcheckoptions**、**sa_conn_properties**、**SET** 文、Sybase Central、SYSOPTIONS システム・ビューを使用して取得できます。

- 接続されているユーザについては、**sp_iqcheckoptions** ストアド・プロシージャが、デフォルト値から変更されたデータベース・オプションの現在の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を変更し、変更されたこれら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 は、デフォルト値から変更されたサーバ起動オプションも一覧表示します。
  DBA が **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 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。

sp_iqcheckoptions ストアド・プロシージャには、パラメータを指定する必要はありません。Interactive SQL で、次のコマンドを実行します。

```
sp_iqcheckoptions
```

『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』を参照してください。

システム・テーブル DBA.SYSOPTIONDEFAULTS には、Sybase IQ および SQL Anywhere オプションのすべての名前とデフォルト値が格納されています。 このテーブルにクエリを実行すると、すべてのオプションのデフォルト値を参照できます。

- 現在の接続のオプション設定は、「接続プロパティ」から取得できます。**sa_conn_properties** システム・プロシージャを使用すると、すべての接続プロパティをリストできます。

  ```
  call sa_conn_properties
  ```
- Interactive SQL では、引数なしで **SET** 文を使用すると、現在のオプション設定がリストされます。

  ```
  SET
  ```
- Sybase Central の場合、データベース上で右クリックし、サブメニューから [オプション] を選択します。
- SYSOPTIONS システム・ビューで、次のクエリを実行します。

  ```
  SELECT *
  FROM SYSOPTIONS
  ```

  このクエリでは、PUBLIC の値すべてと、明示的に設定された USER の値が表示されます。
- 個々の設定は **connection_property** システム関数を使用して確認できます。たとえば、次の文で Ansinull オプションの値がわかります。

  ```
  SELECT connection_property ('Ansinull')
  ```

データベース・オプションのスコープと継続期間

オプションのスコープには 3 つのレベル、 public、user、temporary があります。

temporary オプションは、user と public より優先度が高くなっています。ユーザ・レベルのオプションは public より優先度が高くなっています。ユーザ・レベルのオプションを現在のユーザに対して設定すると、対応する temporary オプションも同じように設定されます。

オプションの中には (COMMIT 動作など)、スコープ内でデータベース全体に影響するものがあります。これらのオプションを設定するためには、DBA パーミッションが必要です。その他のオプション (ISOLATION_LEVEL など) は、現在の接続のみに適用され、特別なパーミッションは必要ありません。

オプション設定の変更がいつ有効になるかは、各オプションによって異なります。RECOVERY_TIME などのグローバル・オプションの変更は、次にサーバを起動したときに有効になります。次のリストは、サーバを再起動したときに有効になるオプションの一部を示します。

サーバの再起動を必要とするデータベース・オプション
CACHE_PARTITIONS
CHECKPOINT_TIME
OS_FILE_CACHE_BUFFERING
OS_FILE_CACHE_BUFFERING_TEMPDB
PREFETCH_BUFFER_LIMIT
PREFETCH_BUFFER_PERCENT
RECOVERY_TIME
SWEEPER_THREADS_PERCENT
WASH_AREA_BUFFERS_PERCENT

現在の接続にのみ影響を与えるオプションは、通常すぐに有効になります。オプションの設定は、たとえばトランザクションの途中でも変更できます。

警告！ カーソルが開いている最中にオプションを変更すると、信頼できない結果につながる場合があります。たとえば、カーソルが開いているときに DATE_FORMAT を変更しても、次のローのフォーマットは変わりません。カーソルが取り出される方法によっては、いくつかのローの処理が終わってから、ユーザの指定どおりに変更が行われる場合があります。

temporary オプション

SET OPTION 文に **TEMPORARY** キーワードを追加すると、変更の持続期間を変更できます。

通常、オプションの変更は永久的です。**SET OPTION** 文を使用して明示的に変更されるまで変わりません。

SET TEMPORARY OPTION 文で指定された新しいオプション値は、現在の接続のみに適用され、現在の接続の間だけ持続します。

SET TEMPORARY OPTION を使用して PUBLIC オプションを設定すると、データベースの実行中はその変更が継続します。データベースが停止すると、PUBLIC ユーザ ID の TEMPORARY・オプションはその永久値に戻ります。

PUBLIC ユーザ ID に対するオプションを設定すると、セキュリティに一時的な利点をもたらします。たとえば、LOGIN_MODE オプションが使用可能な場合、データベースは実行中のシステムのログイン・セキュリティを使用します。LOGIN_MODE を一時的に有効にすると、Windows ドメインのセキュリティに依存しているデータベースは、データベースが停止し、ローカル・マシンにコピーされるといった場合でも、危険にさらされることはありません。この場合、LOGIN_MODE オプションは、永久値に戻ります。永久値は STANDARD で、これは統合ログインが許可されないモードです。

public オプション

PUBLIC ユーザ ID のオプション値を変更すると、値を設定していないすべてのユーザのオプション値が設定されます。

DBA 権限を持つユーザのみが、PUBLIC ユーザ ID にオプションを設定する権限を持ちます。

ただし、そのオプションがすでに PUBLIC ユーザ ID に設定されていないかぎり、個々のユーザ ID にオプション値は設定されません。

オプション設定の削除

option-value を省略すると、指定したオプション設定がデータベースから削除されます。

option-value が各ユーザ固有のオプション設定の場合は、値は PUBLIC 設定に戻ります。TEMPORARY オプションを削除すると、その設定値は永久値に戻ります。

たとえば、次の文を実行すると ANSINULL オプションはデフォルト値に戻ります。

```
SET OPTION ANSINULL =
```

オプション設定時にオプション名を誤って入力すると、誤った名前が SYSOPTION テーブルに保存されます。誤って入力した名前を SYSOPTION テーブルから削除するには、オプション PUBLIC でオプション名の後に等号を付け、値を指定しません。

```
SET OPTION PUBLIC.a_mistyped_name=;
```

たとえば、あるオプションを設定するときに名前を誤って入力した場合、SYSOPTIONS ビューで選択することでそのオプションが保存されたことを確認できます。

```
SET OPTION PUBLIC.a_mistyped_name='ON';
SELECT * FROM SYSOPTIONS ORDER BY 2;
```

user_name	option	設定
PUBLIC	a_mistyped_name	ON
PUBLIC	Abort_On_Error_File	
PUBLIC	Abort_On_Error_Line	0
PUBLIC	Abort_On_Error_Number	0
...		

誤って入力したオプションは、値を設定しないことで削除し、その後、オプションが削除されたことを確認します。

```
SET OPTION PUBLIC.a_mistyped_name=;
SELECT * FROM SYSOPTIONS ORDER BY 2;
```

user_name	option	設定
PUBLIC	Abort_On_Error_File	
PUBLIC	Abort_On_Error_Line	0
PUBLIC	Abort_On_Error_Number	0
...		

PUBLIC オプションを削除した後に、USER オプションを追加しようとすると、次のエラー・メッセージが表示されます。

```
Couldn't execute the statement.
Invalid option 'chained' -- no PUBLIC setting exists
SQLCODE=-200?ODBC 3 State="42000"
Line 1,Column 29
```

PUBLIC オプションをそのデフォルト値にリセットするには、デフォルト値を明示的に設定します。

```
SET OPTION PUBLIC.chained ='ON';
```

オプションの初期設定

ストアド・プロシージャを使用して、ユーザのデータベース・オプションを初期設定できます。

Sybase IQ への接続は、TDS (Tabular Data Stream) プロトコル (Open Client と jConnect™ for JDBC™ 接続) または Sybase IQ プロトコル (ODBC、Embedded SQL) を使用して行われます。

TDS プロトコルと Sybase IQ 固有のプロトコルを両方使用するユーザがいる場合は、ストアド・プロシージャを使用してそれらを初期設定できます。出荷時設定

では、Sybase IQ は、デフォルトの Adaptive Server Enterprise の動作を反映させるため、この方法を使用して Open Client 接続と jConnect 接続を設定しています。

初期設定は、LOGIN_PROCEDURE オプションを使用して制御され、接続が有効であることを確認するすべてのチェック実行後に呼び出されます。
LOGIN_PROCEDURE オプションは、ユーザが接続したときに実行されるストアド・プロシージャを指定します。デフォルト設定では、**sp_login_environment** システム・ストアド・プロシージャが使用されますが異なるストアド・プロシージャを指定することもできます。『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』を参照してください。

sp_login_environment は、接続が TDS によって行われたかどうかを確認します。TDS によって行われている場合は、**sp_tsql_environment** プロシージャを呼び出して、いくつかのオプションに現在の接続に合った新しいデフォルト値を設定します。

参照：
- LOGIN_PROCEDURE オプション (478 ページ)

廃止予定のデータベース・オプション

このリリースで廃止予定のデータベース・オプションの詳細については、『新機能の概要 Sybase IQ15.4』を参照してください。

一般的なデータベース・オプション

一般的なデータベース・オプションは、Transact-SQL 互換性オプションと Interactive SQL オプションを除くすべてのオプションで構成されます。

注意： Sybase 製品の保守契約を結んでいるサポート・センタから、この表に含まれている以外の内部オプションを使うよう指示されることもあります。

表 16 : 一般的なデータベース・オプション

オプション名	指定できる値	デフォルト設定
AGGREGATION_PREFERENCE	-3 ～ 3	0
ALLOW_READ_CLIENT_FILE	ON、OFF	OFF
APPEND_LOAD	ON、OFF	OFF
AUDITING	ON、OFF	OFF

オプション名	指定できる値	デフォルト設定
BIT_VECTOR_PINNABLE_CACHE_PERCENT*	0 – 100	40
BLOCKING	OFF	OFF
BT_PREFETCH_MAX_MISS	0 – 1000	2
BT_PREFETCH_SIZE	0 – 100	10
BTREE_PAGE_SPLIT_PAD_PERCENT	0 - 90	50
CACHE_PARTITIONS	0 ～ 64 の 2 の累乗値	0
CHECKPOINT_TIME	数値 (分単位)	60
CIS_ROWSET_SIZE	整数	50
CONVERSION_MODE	0, 1	0
CONVERT_VARCHAR_TO_1242	ON、OFF	OFF
COOPERATIVE_COMMIT_TIMEOUT	整数	250
COOPERATIVE_COMMITS	ON、OFF	ON
CURSOR_WINDOW_ROWS	20 – 100000	200
DATE_FIRST_DAY_OF_WEEK	0 – 6	0
DATE_FORMAT	文字列	'YYYY-MM-DD'
DATE_ORDER	'YMD'、'DMY'、'MDY'	'YMD'
DBCC_LOG_PROGRESS	ON、OFF	OFF
DBCC_PINNABLE_CACHE_PERCENT	0 – 100	50
DEBUG_MESSAGES	ON、OFF	OFF
DEFAULT_DBSPACE	文字列	'' (空の文字列)
DEFAULT_DISK_STRIPING	ON、OFF	ON
DEDICATED_TASK	ON、OFF	OFF
DEFAULT_HAVING_SELECTIVITY_PPM	0 – 1000000	0
DEFAULT_KB_PER_STRIPE	1 ～ max unsigned bigint	1

オプション名	指定できる値	デフォルト設定
DEFAULT_LIKE_MATCH_SELECTIVITY_PPM	0 – 1000000	150000
DEFAULT_LIKE_RANGE_SELECTIVITY_PPM	1 – 1000000	150000
DELAYED_COMMIT_TIMEOUT	整数	500
DELAYED_COMMITS	OFF	OFF
DISABLE_RI_CHECK	ON、OFF	OFF
EARLY_PREDICATE_EXECUTION	ON、OFF	ON
ENABLE_LOB_VARIABLES	ON、OFF	OFF
EXTENDED_JOIN_SYNTAX	ON、OFF	ON
FORCE_DROP	ON、OFF	OFF
FORCE_NO_SCROLL_CURSORS	ON、OFF	OFF
FORCE_UPDATABLE_CURSORS	ON、OFF	OFF
FP_LOOKUP_SIZE	1 ～ 4096MB	16 MB
FP_LOOKUP_SIZE_PPM	1 – 1000000	2500
FP_PREDICATE_WORKUNIT_PAGES	整数	200
FP_PREFETCH_SIZE	0 – 100	10
FPL_EXPRESSION_MEMORY_KB	0 – 20000	1024
GARRAY_FILL_FACTOR_PERCENT	0 – 1000	25
GARRAY_INSERT_PREFETCH_SIZE	0 – 100	3
GARRAY_PAGE_SPLIT_PAD_PERCENT	0-100	25
GARRAY_RO_PREFETCH_SIZE	0 – 100	10
HASH_PINNABLE_CACHE_PERCENT*	0 – 100	20
HASH_THRASHING_PERCENT	0 – 100	10
HG_DELETE_METHOD	0 – 3	0

オプション名	指定できる値	デフォルト設定
HG_SEARCH_RANGE	整数	10
HTTP_SESSION_TIMEOUT	整数値 (1 〜 525600)	30
IDENTITY_ENFORCE_UNIQUE-NESS	ON、OFF	OFF
IDENTITY_INSERT	文字列	" (空の文字列)
INDEX_ADVISOR	ON、OFF	OFF
INDEX_PREFERENCE	-10 – 10	0
INFER_SUBQUERY_PREDICATES	ON、OFF	ON
IN_SUBQUERY_PREFERENCE	-3 – 3	0
IQGOVERN_MAX_PRIORITY	1 – 3	2
IQGOVERN_PRIORITY	1 – 3	2
IQGOVERN_PRIORITY_TIME	1 〜 1000000 秒	0 (無効)
ISOLATION_LEVEL	0, 1, 2, 3	0
JAVA_LOCATION	文字列	" (空の文字列)
JAVA_VM_OPTIONS	文字列	" (空の文字列)
JOIN_EXPANSION_FACTOR	0 – 100	30
JOIN_OPTIMIZATION	ON、OFF	ON
JOIN_PREFERENCE	-7 – 7	0
JOIN_SIMPLIFICATION_ THRESHOLD	1 – 64	15
LARGE_DOUBLES_ACCUMULATOR	ON、OFF	OFF
LF_BITMAP_CACHE_KB	1 – 8	4
LOAD_ZEROLENGTH_ASNULL	ON、OFF	OFF
LOCKED	ON、OFF	OFF
LOG_CONNECT	ON、OFF	ON
LOG_CURSOR_OPERATIONS	ON、OFF	OFF
LOGIN_MODE	STANDARD、MIXED、INTEGRATED	STANDARD

オプション名	指定できる値	デフォルト設定
LOGIN_PROCEDURE	文字列	sp_login_environment
MAIN_RESERVED_DBSPACE_MB	整数値 >= 200 (MB 単位)	200
MAX_CARTESIAN_RESULT	整数	100000000
MAX_CLIENT_NUMERIC_PRECISION	0 – 126	0
MAX_CLIENT_NUMERIC_SCALE	0 – 126	0
MAX_CONNECTIONS	0 - 2147483647	Unlimited
MAX_CUBE_RESULT	0 – 4294967295	10000000
MAX_CURSOR_COUNT	整数	50
MAX_DAYS_SINCE_LOGIN	0 - 2147483647	Unlimited
MAX_FAILED_LOGIN_ATTEMPTS	0 - 2147483647	Unlimited
MAX_HASH_ROWS	4294967295 以下の整数値	2500000
MAX_IQ_THREADS_PER_CONNECTION	3 – 10000	144
MAX_IQ_THREADS_PER_TEAM	1 – 10000	144
MAX_JOIN_ENUMERATION	1 – 64	15
MAX_NON_DBA_CONNECTIONS	0 – 2147483647	Unlimited
MAX_PREFIX_PER_CONTAINS_PHRASE	0 – 300	1
MAX_QUERY_PARALLELISM	整数	64
MAX_QUERY_TIME	$0 – 2^{32} - 1$	0 (無効)
MAX_STATEMENT_COUNT	整数	100
MAX_TEMP_SPACE_PER_CONNECTION	整数	0
MAX_WARNINGS	整数	$2^{48} - 1$
MINIMIZE_STORAGE	ON、OFF	OFF
MIN_PASSWORD_LENGTH	0 以上の整数	0 文字
MONITOR_OUTPUT_DIRECTORY	文字列	*database directory*

オプション名	指定できる値	デフォルト設定
NOEXEC	ON、OFF	OFF
NON_ANSI_NULL_VARCHAR	ON、OFF	OFF
NOTIFY_MODULUS	整数	100000
ODBC_DISTINGUISH_CHAR_AND_VARCHAR	ON、OFF	OFF
ON_CHARSET_CONVERSION_FAILURE	文字列	IGNORE
OS_FILE_CACHE_BUFFERING	ON、OFF	OFF
PASSWORD_GRACE_TIME	0 – 2147483647	0
PASSWORD_EXPIRY_ON_NEXT_LOGIN	ON、OFF	OFF
PASSWORD_LIFE_TIME	0 – 2147483647	Unlimited
POST_LOGIN_PROCEDURE	文字列	dbo.sa_post_login_procedure
PRECISION	126	126
PREFETCH	ON、OFF	ON
PREFETCH_BUFFER_LIMIT	整数	0
PREFETCH_BUFFER_PERCENT	0 – 100	40
PREFETCH_GARRAY_PERCENT	0 – 100	60
PREFETCH_SORT_PERCENT	0 – 100	20
PRESERVE_SOURCE_FORMAT	ON、OFF	ON
QUERY_DETAIL	ON、OFF	OFF
QUERY_NAME	文字列	'' (空の文字列)
QUERY_PLAN	ON、OFF	ON
QUERY_PLAN_AFTER_RUN	ON、OFF	OFF
QUERY_PLAN_AS_HTML	ON、OFF	OFF
QUERY_PLAN_AS_HTML_DIRECTORY	文字列	'' (空の文字列)

オプション名	指定できる値	デフォルト設定
QUERY_PLAN_TEXT_ACCESS	ON、OFF	OFF
QUERY_PLAN_TEXT_CACHING	ON、OFF	OFF
QUERY_ROWS_RETURNED_LIMIT	整数	0
QUERY_TEMP_SPACE_LIMIT	整数	0
QUERY_TIMING	ON、OFF	OFF
RECOVERY_TIME	数値 (分単位)	2
RETURN_DATE_TIME_AS_STRING	ON、OFF	OFF
ROW_COUNT	整数	0
SCALE	0 – 126	38
SIGNIFICANTDIGITSFORDOU-BLEEQUALITY	0 – 15	0
SORT_COLLATION	Internal、collation_name、または collation_id	Internal
SORT_PINNABLE_CACHE_PER-CENT*	0 – 100	20
SUBQUERY_CACHING_PREFER-ENCE	-3 – 3	0
SUBQUERY_FLATTENING_PER-CENT	0, 1 - 2^{32} -1	100
SUBQUERY_FLATTENING_PREF-ERENCE	-3 – 3	0
SUBQUERY_PLACEMENT_PREFER-ENCE	-1 – 1	0
SUPPRESS_TDS_DEBUGGING	ON、OFF	OFF
SWEEPER_THREADS_PERCENT	1 ～ 40	10
TDS_EMPTY_STRING_IS_NULL	ON、OFF	OFF
TEMP_DISK_PER_STRIPE	整数値 > 0 (KB 単位)	1
TEMP_EXTRACT_APPEND	ON、OFF	OFF
TEMP_EXTRACT_BINARY	ON、OFF	OFF

オプション名	指定できる値	デフォルト設定
TEMP_EXTRACT_COLUMN_DELIM-ITER	文字列	','
TEMP_EXTRACT_DIRECTORY	文字列	'' (空の文字列)
TEMP_EXTRACT_ESCAPE_QUOTES	ON、OFF	OFF
TEMP_EXTRACT_NAME1 – TEMP_EXTRACT_NAME8	文字列	'' (空の文字列)
TEMP_EXTRACT_NULL_AS_EMPTY	ON、OFF	OFF
TEMP_EXTRACT_NULL_AS_ZERO	ON、OFF	OFF
TEMP_EXTRACT_QUOTE	文字列	'' (空の文字列)
TEMP_EXTRACT_QUOTES	ON、OFF	OFF
TEMP_EXTRACT_QUOTES_ALL	ON、OFF	OFF
TEMP_EXTRACT_ROW_DELIMITER	文字列	'' (空の文字列)
TEMP_EXTRACT_SIZE1 – TEMP_EXTRACT_SIZE8	AIX、HP-UX： 0 〜 64GB Sun Solaris、 Linux ： 0 〜 512GB Windows：0 〜 128GB	0
TEMP_EXTRACT_SWAP	ON、OFF	OFF
TEMP_RESERVED_DBSPACE_MB	整数値 >= 200 (MB 単位)	200
TEMP_SPACE_LIMIT_CHECK	ON、OFF	ON
TEXT_DELETE_METHOD	0 – 2	0
TIME_FORMAT	文字列	'HH:NN:SS.SSS'
TIMESTAMP_FORMAT	文字列	'YYYY-MM-DD HH:NN:SS.SSS'
TOP_NSORT_CUTOFF_PAGES	1 – 1000	1
TRIM_PARTIAL_MBC	ON、OFF	OFF
USER_RESOURCE_RESERVATION	整数	1
VERIFY_PASSWORD_FUNCTION	文字列	'' (空の文字列)
WASH_AREA_BUFFERS_PERCENT	1 – 100	20
WAIT_FOR_COMMIT	ON、OFF	OFF
WD_DELETE_METHOD	0 – 3	0

参照：

- Transact-SQL 互換性オプション (397 ページ)
- Interactive SQL オプション (400 ページ)
- アルファベット順のオプション・リスト (402 ページ)

データ抽出オプション

データ抽出機能では、標準インタフェースから 1 つまたは複数のディスク・ファイル、または名前付きパイプに **SELECT** 文の出力をリダイレクトすることにより、データベースからデータを抽出できます。

いくつかのデータベース・オプション (TEMP_EXTRACT_...) は、この機能の制御に使用されます。

『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」>「データベースからデータをエクスポートする方法」>「データ抽出機能」>「抽出オプション」を参照してください。

Transact-SQL 互換性オプション

Transact-SQL 互換性オプションを使用することにより、Sybase IQ の動作に Adaptive Server Enterprise との互換性を持たせることができます。また旧版の動作と ISO SQL92 動作を両方可能にすることもできます。

Adaptive Server Enterprise との互換性をさらに向上させるため、Sybase IQ の **SET OPTION** 文の代わりに Transact-SQL の **SET** 文を使用して、現在の接続の継続中にだけ適用される以下のオプションの値を設定できます。

表 17 : Transact-SQL 互換性オプション

オプション	指定できる値	デフォルト設定
ALLOW_NULLS_BY_DEFAULT	ON、OFF	ON
ANSI_BLANKS*	ON、OFF	OFF
ANSI_CLOSE_CURSORS_ON_ROLLBACK	ON	ON
ANSI_INTEGER_OVERFLOW*		
ANSI_PERMISSIONS	ON、OFF	ON
ANSINULL	ON、OFF	ON
ANSI_SUBSTRING	ON、OFF	ON

オプション	指定できる値	デフォルト設定
ANSI_UPDATE_CONSTRAINTS	OFF、CURSORS、STRICT	CURSORS
ASE_BINARY_DISPLAY	ON、OFF	OFF
ASE_FUNCTION_BEHAVIOR	ON、OFF	OFF
CHAINED	ON、OFF	ON
CLOSE_ON_ENDTRANS	ON	ON
CONTINUE_AFTER_RAISERROR	ON、OFF	ON
CONVERSION_ERROR	ON、OFF	ON
DIVIDE_BY_ZERO_ERROR	ON、OFF	ON
ESCAPE_CHARACTER*	予約済み	予約済み
FIRE_TRIGGERS*	ON、OFF	ON
NEAREST_CENTURY	0 – 100	50
NON_KEYWORDS	カンマで区切られたキーワード・リスト	オフにされているキーワードなし
ON_TSQL_ERROR	STOP、CONTINUE、CONDITIONAL	CONDITIONAL
QUERY_PLAN_ON_OPEN*		
QUOTED_IDENTIFIER	ON、OFF	ON
RI_TRIGGER_TIME*		
SQL_FLAGGER_ERROR_LEVEL	E、I、F、W、OFF、SQL:1992/Entry、SQL:1992/Intermediate、SQL:1992/Full、SQL:1999/Core、SQL:1999/Package、SQL:2003/Core、SQL:2003/Package	OFF
SQL_FLAGGER_WARNING_LEVEL	E、I、F、W、OFF、SQL:1992/Entry、SQL:1992/Intermediate、SQL:1992/Full、SQL:1999/Core、SQL:1999/Package、SQL:2003/Core、SQL:2003/Package	OFF
STRING_RTRUNCATION	ON、OFF	ON

オプション	指定できる値	デフォルト設定
TEXTSIZE*		
TSQL_HEX_CONSTANT*		
TSQL_VARIABLES	ON、OFF	OFF

注意： オプション名の脇にアスタリスク (*) が付いているものは、現在 Sybase IQ ではサポートされていません。

参照：

- 一般的なデータベース・オプション (389 ページ)
- Interactive SQL オプション (400 ページ)
- アルファベット順のオプション・リスト (402 ページ)
- SET 文 [T-SQL] (352 ページ)

Adaptive Server Enterprise との互換性のための Transact-SQL オプション設定

オプションのデフォルト設定のなかには、Adaptive Server Enterprise のデフォルト設定と異なるものがあります。動作の互換性を保証したい場合は、各オプションを明示的に設定してください。

Open Client または JDBC インタフェースを介して接続された場合、現在の接続に Adaptive Server Enterprise との互換性を維持させるために、いくつかのオプションが明示的に設定されています。

設定方法については、『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』を参照してください。

表 18 : ASE 互換性のために明示的に設定される Transact-SQL オプション

オプション	ASE 互換の設定
ALLOW_NULLS_BY_DEFAULT	OFF
ANSINULL	OFF
CHAINED	OFF
CONTINUE_AFTER_RAISERROR	ON
DATE_FORMAT	YYYY-MM-DD
DATE_ORDER	MDY

オプション	ASE 互換の設定
ESCAPE_CHARACTER	OFF
ISOLATION_LEVEL	1
ON_TSQL_ERROR	CONDITIONAL
QUOTED_IDENTIFIER	OFF
TIME_FORMAT	HH:NN:SS.SSS
TIMESTAMP_FORMAT	YYYY-MM-DD HH:NN:SS.SSS
TSQL_VARIABLES	OFF

Interactive SQL オプション

Interactive SQL オプションは、データベースに対する Interactive SQL の対応を変更します。

構文 1

```
SET [ TEMPORARY ] OPTION
... [ userid. | PUBLIC. ]option-name = [ option-value ]
```

構文 2

```
SET PERMANENT
```

構文 3

```
SET
```

パラメータ

```
userid:
identifier, string or host-variable
```

```
option-name:
identifier, string or host-variable
```

```
option-value:
host-variable (indicator allowed), string, identifier,
or number
```

説明
TEMPORARY キーワードが含まれた構文 1 は、複合文の **BEGIN** キーワードと **END** キーワードの間で使用できません。

構文 2 の **SET PERMANENT** は、現在の Interactive SQL オプションすべてを SYSOPTIONS システム・テーブルに格納します。Interactive SQL が現在のユーザ ID で起動されるたびに、これらの値が自動的に設定されます。

構文 3 を使用すると、現在のオプション設定がすべて表示されます。Interactive SQL またはデータベース・サーバに テンポラリ・オプションがある場合は、それが表示されます。それ以外の場合、永久オプション設定が表示されます。。

表 19 : Interactive SQL オプション

オプション	指定できる値	デフォルト設定
DEFAULT_ISQL_ENCODING	識別子または文字列	空の文字列 (システム・コード・ページを使用)
NULLS*	文字列	NULL
ON_ERROR	STOP、CONTINUE、PROMPT、EXIT、NOTIFY_CONTINUE、NOTIFY_STOP、NOTIFY_EXIT	PROMPT
OUTPUT_FORMAT*	ASCII、FIXED、DIF、DBASEII、DBASEIII、FOXPRO、LOTUS、SQL、TEXT、WATFILE	ASCII
OUTPUT_LENGTH*	負でない整数	0 (トランケーションなし)
OUTPUT_NULLS*	文字列	'NULL'
STATISTICS*	0, 3, 4, 5, 6	3
TRUNCATION_LENGTH*	整数	256

注意： オプション名の脇にアスタリスク (*) が付いているものは、現在 Sybase IQ ではサポートされていません。

参照：
- 一般的なデータベース・オプション (389 ページ)
- Transact-SQL 互換性オプション (397 ページ)

- アルファベット順のオプション・リスト (402 ページ)

アルファベット順のオプション・リスト

一般的なオプション、Transact-SQL 互換性オプション、Interactive SQL データベース・オプションの説明。オプションのクラスを示す、角カッコで囲まれたインジケータがオプション名の後に付くこともあります。

データベース・オプションのクラス・インジケータは次のとおりです。

- [Interactive SQL] – このオプションを使用して、データベースに対する Interactive SQL の対応を変更します。
- [TSQL] – このオプションを使用して、Sybase IQ の動作に Adaptive Server Enterprise との互換性を持たせることができます。また旧版の動作と ISO SQL92 動作を両方可能にすることもできます。

参照：
- データベース・オプションの概要 (383 ページ)
- 一般的なデータベース・オプション (389 ページ)
- Transact-SQL 互換性オプション (397 ページ)
- Interactive SQL オプション (400 ページ)

AGGREGATION_PREFERENCE オプション

集合関数を処理する際に使用するアルゴリズムを選択します。

指定できる値
-3 ～ 3

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。

説明
Sybase IQ オプティマイザは、クエリ内の集合関数 (**GROUP BY** 関数、**DISTINCT** 関数、**SET** 関数) に対して、集合関数処理のためのいくつかのアルゴリズムの 1 つを選択します。`AGGREGATION_PREFERENCE` オプションを使用すると、オプティマイザが処理量をもとに決定した使用アルゴリズムを無効にできます。クエリ・エ

ンジンに対してアルゴリズムが適切かどうかを判断するための内部規則を無効にするものではありません。

このオプションは通常、内部のテスト、および、オプティマイザがうまく処理できないクエリの手動チューニングに利用されます。経験豊富な DBA のみ使用してください。AGGREGATION_PREFERENCE を設定するのは、オプティマイザに変更を加えることが必要である場合のみであるため、このオプションを設定する必要がある場合は、Sybase 製品の保守契約を結んでいるサポート・センタにご連絡ください。

表 20 : AGGREGATION_PREFERENCE の値

値	アクション
0	オプティマイザの選択に従う。
1	ソートを使用する集合関数を優先する。
2	IQ のインデックスを使用する集合関数を優先する。
3	ハッシュを使用する集合関数を優先する。
-1	ソートを使用する集合関数を避ける。
-2	IQ のインデックスを使用する集合関数を避ける。
-3	ハッシュを使用する集合関数を避ける。

ALLOW_NULLS_BY_DEFAULT オプション [TSQL]

NULL か NOT NULL かを指定しないで作成された新規カラムに NULL 値を許可するかどうかを制御します。

指定できる値
ON、OFF

デフォルト値
ON

Open Client および JDBC 接続の場合は OFF

説明
ALLOW_NULLS_BY_DEFAULT オプションは、Transact-SQL との互換性を維持するために組み込まれています。

ANSI_CLOSE_CURSORS_ON_ROLLBACK オプション [TSQL]

WITH HOLD 句で開いたカーソルを、**ROLLBACK** を実行するときに閉じるかどうかを制御します。

指定できる値
ON

デフォルト値
ON

説明
ANSI の SQL/3 標準では、トランザクションをロールバックするときにすべてのカーソルが閉じている必要があります。このオプションを使用すると、強制的にこの動作に従うようになります。変更はできません。このオプションより、CLOSE_ON_ENDTRANS オプションが優先されます。

ANSI_PERMISSIONS オプション [TSQL]

DELETE 文と UPDATE 文のパーミッションのチェックを制御します。

指定できる値
ON、OFF

デフォルト値
ON

説明
ANSI_PERMISSIONS を ON にすると、**DELETE** 文と **UPDATE** 文に対する SQL92 パーミッションの要件がチェックされます。 Adaptive Server Enterprise では、このオプションのデフォルト値は OFF です。次の表にその違いを示します。

表 21 : ANSI_PERMISSIONS オプションの効果

SQL 文	ANSI_PERMISSIONS OFF 時に必要なパーミッション	ANSI_PERMISSIONS ON 時に必要なパーミッション
UPDATE	値が設定されているカラムでの UPDATE パーミッション	値が設定されているカラムでの UPDATE パーミッション WHERE 句に指定されたすべてのカラムでの SELECT パーミッション SET 句の右側に指定されたすべてのカラムでの SELECT パーミッション
DELETE	テーブルに対する DELETE パーミッション	テーブルに対する DELETE パーミッション **WHERE 句に指定されるすべてのカラ ムに対する SELECT パーミッション。**

ANSI_PERMISSIONS オプションは、PUBLIC のみに設定できます。個人的な設定は許可されません。

ANSINULL オプション [TSQL]

NULL 値のある = と != の解釈を制御します。

指定できる値
ON、OFF

デフォルト値
ON

説明
ANSINULL を ON にすると、= または != を使用した NULL との比較結果は不定になります。CASE などの他の処理による比較結果も対象に含まれます。

ANSINULL を OFF に設定すると、NULL との比較に対して不定ではない結果を返すことで、Adaptive Server Enterprise との互換性を維持できます。

注意： SQL Anywhere とは異なり、Sybase IQ は NULL 値を含むカラムに対する集合関数に対して "null value eliminated in aggregate function" (SQLSTATE=01003) という警告を生成しません。

ANSI_SUBSTRING オプション [TSQL]

start または length パラメータに負の値が指定されている場合に、SUBSTRING (SUBSTR) 関数の動作を制御します。

指定できる値
ON、OFF

デフォルト値
ON

説明
ANSI_SUBSTRING オプションを ON に設定した場合、**SUBSTRING** 関数は ANSI/ISO SQL/2003 と同じ動作をします。start オフセットが負またはゼロの場合は、文字列の左側に非文字が埋め込まれているものと解釈され、length が負であればエラーが発生します。

このオプションを OFF に設定した場合、**SUBSTRING** 関数の動作は以前のバージョンの Sybase IQ と同じです。 負の start オフセットは、文字列の末尾からのオフセットを意味し、負の length は、対象の部分文字列が開始オフセットから左側に数えて length 文字目の位置で終了することを意味します。 start オフセット 0 を使用することは、start オフセット 1 の場合と同じです。

SUBSTRING 関数では、正以外の start オフセットまたは負の length を使用しないようにしてください。可能なかぎり、SUBSTRING 関数の代わりに **LEFT** 関数または **RIGHT** 関数を使用してください。

例
次の例は、ANSI_SUBSTRING オプションの設定に基づいて **SUBSTRING** 関数によって返される値の違いを示します。

```
SUBSTRING( 'abcdefgh',-2,4 );
    ansi_substring = Off ==> 'gh'
    // substring starts at second-last character
    ansi_substring = On  ==> 'a'
    // takes the first 4 characters of
    // ???abcdefgh and discards all ?
```

```
SUBSTRING( 'abcdefgh',4,-2 );
    ansi_substring = Off ==> 'cd'
    ansi_substring = On  ==> value -2 out of range
    for destination
```

```
SUBSTRING( 'abcdefgh',0,4 );
    ansi_substring = Off ==> 'abcd'
    ansi_substring = On  ==> 'abc'
```

ANSI_UPDATE_CONSTRAINTS オプション

許可する更新の範囲を制御します。

指定できる値
OFF、CURSORS、STRICT

デフォルト値
CURSORS

説明
Sybase IQ には、ANSI SQL 規格で許可されていない更新を可能にするいくつかの拡張機能が用意されています。 これらの拡張機能は、更新を行うための強力で効果的なメカニズムを提供します。 しかし、場合によっては、通常と異なる動作が起きることもあります。 ユーザ側のアプリケーションがこれらの拡張機能の動作を想定していない場合、このような動作によって、更新が失われるなどの異常が発生することがあります。

ANSI_UPDATE_CONSTRAINTS は、SQL92 標準で認められている更新以外も認めるかどうかを制御します。

このオプションを STRICT に設定すると、次の更新が禁止されます。

- JOINS を含んだカーソルの更新
- **ORDER BY** 句に表示されるカラムの更新
- **UPDATE** 文での **FROM** 句の使用

このオプションを CURSORS に設定した場合は、カーソルにだけ同じ制限が加えられます。カーソルが **FOR UPDATE** または **FOR READ ONLY** で開かれている場合を除き、データベース・サーバは SQL92 標準に基づいて更新が許可されるかどうかを判断します。

ANSI_UPDATE_CONSTRAINTS が CURSORS または STRICT の場合、**ORDER BY** 句が含まれるカーソルはデフォルトで **FOR READ ONLY** になります。その他の場合、デフォルトは **FOR UPDATE** です。

例
次のコードの結果は、ANSI_UPDATE_CONSTRAINTS の設定によって変わります。

```
CREATE TABLE mmg (a CHAR(3));
CREATE TABLE mmg1 (b CHAR(3));
```

```
INSERT INTO mmg VALUES ('001');
INSERT INTO mmg VALUES ('002');
INSERT INTO mmg VALUES ('003')
INSERT INTO mmg1 VALUES ('003');
```

```
SELECT * FROM mmg;
SELECT * FROM mmg1;
```

オプション 1： ANSI_UPDATE_CONSTRAINTS を STRICT に設定する。

```
SET OPTION public.Ansi_update_constraints = 'strict';
DELETE MMG FROM MMG1 WHERE A=B;
```

この更新操作は許可されていないというエラーが表示されます。

オプション 2： ANSI_UPDATE_CONSTRAINTS を CURSORS または OFF に設定する。

```
SET OPTION public.Ansi_update_constraints = 'CURSORS';  // or 'OFF'
DELETE mmg FROM mmg1 WHERE A=B;
```

エラーが発生することなく削除が完了します。

参照：

- UPDATE 文 (372 ページ)

ALLOW_READ_CLIENT_FILE オプション

クライアント側データの転送を有効にします。

『SQL Anywhere サーバー – データベース管理』の「データベースの設定」>「データベースオプション」>「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」>「allow_read_client_file オプション」を参照してください。

注意： これらのリファレンスは SQL Anywhere マニュアルにリンクされています。

APPEND_LOAD オプション

改版されたページが使用する領域の節約に役立ちます。

指定できる値
ON、OFF

デフォルト値
OFF

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
APPEND_LOAD オプションは、**LOAD**、**INSERT...SELECT**、**INSERT...VALUES** の各文に適用され、それらの文を次に実行したときに有効になります。

APPEND_LOAD オプションを OFF にすると、削除されたローのロー ID が Sybase IQ で再利用されます。 このオプションを ON にすると、新しいデータはテーブルの末尾に追加されます。

分割されたテーブルと分割されていないテーブルでは、APPEND_LOAD の動作が異なります。分割されたテーブルの各パーティションに、ロー ID の範囲が割当てられます。分割されたテーブルで APPEND_LOAD が ON の場合は、適切なパーティションの最後に新しいローが追加されます。APPEND_LOAD が OFF の場合、削除されたローから最初に使用可能なロー ID と領域が再利用されます。

分割されていないテーブルで APPEND_LOAD が ON の場合は、テーブルのローの最後にある最大ロー ID の後に、新しいローが追加されます。APPEND_LOAD が OFF の場合は、削除されたロー ID が再利用されます。分割されていないテーブルでは、**LOAD** 文または **INSERT** 文の **START ROW ID** 句を使用して挿入を開始するローを指定することにより、ローの挿入場所を制御することもできます。

ASE_BINARY_DISPLAY オプション

Sybase IQ バイナリ・カラムの表示を Adaptive Server Enterprise バイナリ・カラムの表示と一致させるよう指定します。

指定できる値
ON、OFF

デフォルト値
OFF

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
ASE_BINARY_DISPLAY は、**SELECT** 文の出力に影響します。

このオプションは IQ ストアのカラムのみに影響します。変数、カタログ・ストアのカラム、SQL Anywhere カラムには影響しません。このオプションが ON の場合、カラムは判読可能な ASCII 形式で表示されます。たとえば、0x1234567890abcdef と表示されます。 このオプションが OFF の場合、カラムは ASCII ではなくバイナリ出力として表示されます。

バイナリ・データ型でバルク・コピー・オペレーションをサポートするには、**ASE_BINARY_DISPLAY** を OFF に設定します。Sybase IQ は、**LOAD TABLE USING CLIENT FILE** 文を使用したリモート・データのバルク・ロードをサポートします。

参照：
- LOAD TABLE 文 (273 ページ)

ASE_FUNCTION_BEHAVIOR オプション

INTTOHEX と **HEXTOINT** を含む Sybase IQ 関数の出力を Adaptive Server Enterprise 関数の出力と一致させるよう指定します。

指定できる値
ON、OFF

デフォルト値
OFF

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
ASE_BEHAVIOR_FUNCTION が ON の場合、**HEXTOINT** と **INTTOHEX** を含む Sybase IQ データ型変換関数の一部は、Adaptive Server Enterprise 関数の出力と一致する出力を返します。 ASE と Sybase IQ の出力で形式と長さに違いがあるのは、ASE は主にデフォルトとして符号付き 32 ビットを使用し、Sybase IQ は主にデフォルトとして符号なし 64 ビットを使用するためです。

Sybase IQ は、64 ビットの整数をサポートしていません。ASE には 64 ビットの整数データ型がありません。

INTTOHEX 関数と **HEXTOINT** 関数の動作の詳細については、『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』を参照してください。

例
この例で、**HEXTOINT** 関数は、ASE_FUNCTION_BEHAVIOR オプションが ON か OFF かによって異なる値を返します。

ASE_FUNCTION_BEHAVIOR が OFF の場合には、**HEXTOINT** 関数は 4294967287 を返します。

```
select hextoint('fffffff7') from iq_dummy
```

ASE_FUNCTION_BEHAVIOR が ON の場合には、**HEXTOINT** 関数は -9 を返します。

```
select hextoint('fffffff7') from iq_dummy
```

参照：

- CONVERSION_ERROR オプション [TSQL] (419 ページ)

AUDITING オプション [データベース]

データベース内の監査を有効または無効にします。

指定できる値
ON、OFF

デフォルト値
OFF

スコープ
PUBLIC グループのみに設定できます。すぐに有効になります。DBA 権限が必要です。

説明
このオプションは、監査をオンまたはオフにします。

監査とは、データベース内の多くのイベントに関する詳細をトランザクション・ログに記録することです。監査は、パフォーマンスの多少の低下と引き換えに、いくつかのセキュリティ機能を実現します。データベースの監査をオンにすると、トランザクション・ログの使用は停止できません。トランザクション・ログをオフにする前に、監査をオフにする必要があります。監査がオンの場合、データベースは読み取り専用モードで起動できません。

AUDITING オプションを機能させるには、AUDITING オプションを ON に設定し、**sa_enable_auditing_type** システム・プロシージャを使用して監査の対象とする情報のタイプを指定する必要があります。次のいずれかの場合、監査は行われません。

- AUDITING オプションが OFF に設定されている
- 監査オプションが無効になっている

AUDITING オプションを ON に設定し、監査オプションを指定しない場合は、すべてのタイプの監査情報が記録されます。その代わりに、**sa_enable_auditing_type** を使用して、次の任意の組み合わせを記録するように選択できます。それは、パーミッション・チェック、接続試行、DDL 文、public オプション、トリガの組み合わせです。『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』を参照してください。

BIT_VECTOR_PINNABLE_CACHE_PERCENT オプション

永続的ビットベクトル・オブジェクトが維持できる、ユーザの一時メモリの最大パーセンテージを制御します。

指定できる値
0 – 100

デフォルト値
40

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
BIT_VECTOR_PINNABLE_CACHE_PERCENT は、任意の永続的ビットベクトル・オブジェクトをメモリ内に留めることができる、ユーザの一時メモリ割り付けの最大パーセンテージを制御します。デフォルトは 40% です。通常は変更しないでください。

このオプションは、主に 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。BIT_VECTOR_PINNABLE_CACHE_PERCENT の値を変更する場合は、細心の注意が必要です。さまざまなクエリへの影響を分析してから実行してください。

参照：
- HASH_PINNABLE_CACHE_PERCENT オプション (453 ページ)
- SORT_PINNABLE_CACHE_PERCENT オプション (524 ページ)

BLOCKING オプション

ロック競合に対する動作を制御します。

指定できる値
OFF

デフォルト値
OFF

スコープ
個々の接続または PUBLIC グループに設定できます。すぐに有効になります。

説明
BLOCKING が OFF の場合、トランザクションが書き込み処理を試みて、それが別のトランザクションの読み込みロックによってブロックされた場合に、トランザクションにエラーが返されます。

BT_PREFETCH_MAX_MISS オプション

特定のクエリの B ツリー・ページのプリフェッチを継続するかどうかを Sybase IQ が判断する方法を制御します。

指定できる値
0 – 1000

デフォルト値
2

スコープ
個々の接続または PUBLIC グループに設定できます。すぐに有効になります。

説明
Sybase 製品の保守契約を結んでいるサポート・センタから指示があった場合にのみ使用してください。**HG** (High_Group) インデックスを使用するクエリでは、Sybase IQ はプリフェッチが必要ないと判断するまで B ツリー・ページを順にプリフェッチします。 一部のクエリでは、プリフェッチを早く終了することがあります。 BT_PREFETCH_MAX_MISS の値を大きくすると、Sybase IQ はより長くプリフェッチを継続しますが、I/O が不必要に増大する可能性があります。

HG インデックスを使用するクエリの実行速度が予想より遅い場合は、BT_PREFETCH_MAX_MISS の値を徐々に増やしてみます。

最高のパフォーマンスを得るためには、いくつかの設定を試してみる必要があります。大部分のクエリには、1 から 10 までの範囲で値を指定します。

参照：
- BT_PREFETCH_SIZE オプション (413 ページ)
- PREFETCH_BUFFER_LIMIT オプション (506 ページ)

BT_PREFETCH_SIZE オプション

High_Group B ツリーの先読みバッファのサイズを制限します。

指定できる値
0 – 100. 0 に設定すると、B ツリーのプリフェッチが無効になります。

デフォルト値
10

スコープ
個別のユーザにのみ設定可能です。すぐに有効になります。

説明
B ツリーのプリフェッチは、**INSERT**、大規模な **DELETE**、範囲述部、DBCC (データベース一貫性チェッカ) コマンドなどの High_Group インデックスへのすべてのシーケンシャル・アクセスにおいてデフォルトで有効になります。

BT_PREFETCH_SIZE は B ツリー・ページの先読みバッファのサイズを制限します。プリフェッチのサイズを小さくするとバッファの空きが増えますが、パフォーマンスが多少低下します。プリフェッチのサイズを大きくしていくと、これ以上増やしても効果が上がらない値がわかる場合があります。このオプションは、ユニークでない High_Group インデックスで、PREFETCH_GARRAY_PERCENT、GARRAY_INSERT_PREFETCH_SIZE、GARRAY_RO_PREFETCH_SIZE の各オプションとともに使用します。

参照：
- GARRAY_INSERT_PREFETCH_SIZE オプション (451 ページ)
- GARRAY_RO_PREFETCH_SIZE オプション (453 ページ)
- PREFETCH_GARRAY_PERCENT オプション (507 ページ)

BTREE_PAGE_SPLIT_PAD_PERCENT オプション

1 ページあたりのフィル・ファクタを B ツリー構造のページ分割時に決定します。

指定できる値
0 – 90

デフォルト値
50

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できます。すぐに有効になります。

説明
B ツリー構造は、HG、LF、DT、TIME、および DTTM インデックスによって使用されます。B ツリー・ページを分割すると、新しいキーがインデックスに挿入される際の分割を回避するため、指定された割合が空白になります。

インデックスは、ページ・レベルで記憶領域を予約します。この記憶領域は、追加データとして新しいキーが挿入されたときに割り当てることができます。領域を確保するには追加のディスク領域が必要となりますが、インクリメンタルな挿入のパフォーマンスが向上します。インクリメンタルな挿入を行う予定があり、それによってできる新しいローの値が以前のインデックスにまだ存在しない場合は、GARRAY_PAGE_SPLIT_PAD_PERCENT オプションに 0 以外の値を設定すると、インクリメンタルな挿入処理のパフォーマンスが向上します。

インデックスをインクリメンタルに更新する予定がない場合は、このオプションの値を小さくして、ディスク領域を節約できます。

参照：

- GARRAY_FILL_FACTOR_PERCENT オプション (451 ページ)
- GARRAY_PAGE_SPLIT_PAD_PERCENT オプション (452 ページ)

CACHE_PARTITIONS オプション

メインおよびテンポラリ・バッファ・キャッシュに使用するパーティションの数を設定します。

指定できる値
0, 1, 2, 4, 8, 16, 32, 64

0: number_of_cpus/8 を 2 の累乗値に丸めたものをパーティションの数とする (64 が最大)。

1: パーティションは 1 つ。パーティショニングは無効。

2 – 64: パーティションの数。2 の累乗。

デフォルト値
0 (Sybase IQ がパーティション数を自動計算)。

スコープ
PUBLIC グループのみに設定できます。次回、データベース・サーバを起動したときに現在のデータベースで有効になります。

説明
バッファ・キャッシュのパーティショニングは、複数の CPU を搭載したシステムでロック競合を減らし、パフォーマンスを向上させることがあります。 通常は、システム上の CPU の数を基に Sybase IQ が自動計算した値を使用します。 ただし、マルチ CPU 環境でクエリのパフォーマンスが思うように得られない場合は、CACHE_PARTITIONS の値を変更することでパフォーマンスが向上することがあります。 『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」>「ロック管理用ツール」>「ロック競合調査用ツール」を参照してください。

理想的なパーティションの数は、CPU の数とプラットフォームの両方で決まります。環境に最も適した設定を判断するには、いくつかの値で試してみる必要があります。

CACHE_PARTITIONS に設定した値は、メインとテンポラリの両方のバッファ・キャッシュに適用されます。各バッファ・キャッシュにおいて、パーティションの絶対最大数は 64 です。

-iqpartition サーバ・オプションは、サーバ・レベルでパーティションの制限を設定します。サーバの起動時に **-iqpartition** が指定されている場合、これは常に CACHE_PARTITIONS の設定よりも優先されます。『ユーティリティ・ガイド』を参照してください。

パーティションの数は、その他のバッファ・キャッシュの設定には影響しません。また、IQ モニタが収集する統計情報にも影響しません。すべてのパーティションの統計情報は、合計され、1 つの値として報告されます。

例
100 の CPU が搭載されたシステムで CACHE_PARTITIONS を設定しなかった場合、Sybase IQ はパーティションの数を自動的に 16 に設定します。

100 cpus/8 = 12 なので、丸めると 16 になります。

この設定では、メイン・キャッシュに 16 のパーティション、テンポラリ・キャッシュにも 16 のパーティションが作成されます。

100 の CPU を搭載した同じシステムで、パーティション数を明示的に 8 に設定するには、次のように指定します。

```
SET OPTION "PUBLIC".CACHE_PARTITIONS=8
```

CHAINED オプション [TSQL]

BEGIN TRANSACTION 文のないときにトランザクション・モードを制御します。

指定できる値
ON、OFF

Open Client および JDBC 接続の場合は OFF

デフォルト値
ON

説明
Transact-SQL トランザクション・モードを制御します。非連鎖モード (CHAINED = OFF) では、明示的な **BEGIN TRANSACTION** 文でトランザクションが開始された場合でないかぎり、各文は個々にコミットされます。 連鎖モード (CHAINED = ON) では、データ検索文か修正文の前にトランザクションが暗黙的に開始されます。Adaptive Server Enterprise では、デフォルト設定は OFF です。

CHECKPOINT_TIME オプション

チェックポイントを使用しないでデータベース・サーバを実行する最大時間を分単位で設定します。

指定できる値
整数値

デフォルト値
60

スコープ
PUBLIC グループのみに設定できます。このオプションを設定するには、DBA パーミッションが必要です。変更を有効にするには、データベース・サーバを停止し、再起動します。

説明
このオプションは、RECOVERY_TIME とともに使用し、チェックポイントの実行タイミングを決定します。

参照：
- RECOVERY_TIME オプション (519 ページ)

CIS_ROWSET_SIZE オプション

各フェッチに対してリモート・サーバから返されるローの数を設定します。

指定できる値
整数

デフォルト値
50

スコープ
個々の接続または PUBLIC グループに設定できます。リモート・サーバに対して新しい接続が確立された時に有効になります。

説明
このオプションは、ODBC を使用してリモート・データベース・サーバに接続する場合の ODBC **FetchArraySize** 値を設定します。リモート・データ・アクセスの詳細については、『システム管理ガイド： 第 2 巻』を参照してください。

CLOSE_ON_ENDTRANS オプション [TSQL]

トランザクションの終了時にカーソルを閉じるかどうかを制御します。

指定できる値
ON

デフォルト値
ON

説明
CLOSE_ON_ENDTRANS が ON の場合 (これがデフォルト値であり、許可される唯一の値)、カーソルはトランザクションが終了すると閉じられます。これは、Transact-SQL と互換性のある動作です。

CONTINUE_AFTER_RAISERROR オプション [TSQL]

RAISERROR 文に応じて動作を制御します。

指定できる値
ON、OFF

デフォルト値
ON

説明
The **RAISERROR** statement is used within procedures to generate an error. When CONTINUE_AFTER_RAISERROR is set to OFF, the execution of the procedure is stopped when the **RAISERROR** statement is encountered.

CONTINUE_AFTER_RAISERROR が ON の場合、**RAISERROR** 文は実行終了エラーを通知しなくなります。代わりに、**RAISERROR** ステータス・コードとメッセージが格納され、プロシージャが完了すると直前の **RAISERROR** が返されます。**RAISERROR** を引き起こしたプロシージャが別のプロシージャに呼び出された場合、最も外側のプロシージャが終了するまで **RAISERROR** は返されません。

中間 **RAISERROR** の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。リターン時に **RAISERROR** 以外のエラーが発生した場合は、発生したエラー情報が返され、**RAISERROR** 情報は失われます。アプリケーションでは、別の実行ポイントでグローバル変数 @@error を検査して、中間 **RAISERROR** ステータスを問い合わせることができます。

CONTINUE_AFTER_RAISERROR オプションの設定は、ON_TSQL_ERROR オプションが CONDITIONAL (デフォルト値) に設定されている場合にのみ、**RAISERROR** 文の後の動作を制御する目的で使用されます。 ON_TSQL_ERROR オプションを STOP または CONTINUE に設定すると、ON_TSQL_ERROR の設定が CONTINUE_AFTER_RAISERROR の設定よりも優先されます。

参照：

- ON_TSQL_ERROR オプション [TSQL] (500 ページ)

CONVERSION_ERROR オプション [TSQL]

データベースからデータをフェッチするときの、データ型変換障害のレポートを制御します。

指定できる値
ON、OFF

デフォルト値
ON

説明
このオプションでは、データがデータベースからフェッチされるとき、またはデータベースに挿入されるときのデータ型変換障害が、エラー (CONVERSION_ERROR が ON) と警告 (CONVERSION_ERROR が OFF) のどちらとしてデータベースに通知されるかを制御します。

CONVERSION_ERROR を ON に設定すると、SQLE_CONVERSION_ERROR エラーが生成されます。

オプションを OFF に設定すると、警告 SQLE_CANNOT_CONVERT が生成されます。**LOAD** 文でデータ変換を実行する各スレッドは、せいぜい 1 件の警告メッセージを .iqmsg ファイルに書き込むだけです。

変換エラーが警告のみでレポートされた場合、変換できなかった値の代わりに NULL 値が使用されます。Embedded SQL では、インジケータ変数はエラーを出したカラムに -2 を設定します。

CONVERSION_MODE オプション

さまざまな処理で、バイナリ・データ型 (BINARY、VARBINARY、LONG BINARY) と他の非バイナリ・データ型 (BIT、TINYINT、SMALLINT、INT、UNSIGNED INT、BIGINT、UNSIGNED BIGINT、CHAR、VARCHAR、LONG VARCHAR) の間の暗黙の変換を制限します。

指定できる値
0, 1

デフォルト値
0

スコープ
PUBLIC または一時的に設定できます。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。

説明
デフォルト値 0 は、12.7 より前のバージョンでの暗黙の変換動作を維持します。CONVERSION_MODE を 1 に設定すると、**INSERT**、**UPDATE**、クエリでのバイナリ・データ型の非バイナリ・データ型への暗黙の変換が制限されます。バイナリ変換制限モードは、**LOAD TABLE** のデフォルト値と **CHECK** 制約にも適用されます。このオプションを使用して、実質的に意味のない操作である暗号化データの暗黙のデータ型変換を防止できます。

データ型変換の詳細については、『システム管理ガイド：第 1 巻』を参照してください。

カラム暗号化の詳細については、『Sybase IQ による高度なセキュリティ』を参照してください。Sybase IQ Advanced Security オプションの暗号化カラム機能を使用するには、正規のライセンスを取得している必要があります。

暗黙的な変換の制限
CONVERSION_MODE オプションのバイナリ変換制限モード値 1 (CONVERSION_MODE = 1) は、次の操作の暗黙の変換を制限します。

- **LOAD TABLE** (**CHECK** 制約またはデフォルト値が指定されている場合)
- **INSERT...SELECT**、**INSERT...VALUE**、**INSERT...LOCATION**
- 特定の種類の **UPDATE**
- 更新可能カーソルを介する、特定の種類の **INSERT** と **UPDATE**
- 通常、クエリのすべての側面

LOAD TABLE に対する暗黙的なバイナリ変換制限モード

CONVERSION_MODE を 1 に設定すると、暗黙的なバイナリ変換制限モードが、**CHECK** 制約またはデフォルト値を指定した **LOAD TABLE** に適用されます。

例

```
CREATE TABLE t3 (c1 INT,
    csi SMALLINT,
    cvb VARBINARY(2),
    CHECK (csi<cvb));
SET TEMPORARY OPTION CONVERSION_MODE = 1;
```

次の要求は、

```
LOAD TABLE t3(c1 ',', csi ',', cvb ',')
    FROM '/s1/mydata/t3.inp'
    QUOTES OFF ESCAPES OFF
    ROW DELIMITED BY '¥n'
```

次のメッセージで失敗します。

```
"Invalid data type comparison in predicate (t3.csi < t3.cvb),
[-1001013] ['QFA13']"
```

INSERT に対する暗黙的なバイナリ変換制限モード

CONVERSION_MODE を 1 に設定すると、暗黙的なバイナリ変換制限モードが、**INSERT...SELECT**、**INSERT...VALUE**、**INSERT...LOCATION.** に適用されます。

例

```
CREATE TABLE t1 (c1 INT PRIMARY KEY,
    cbt BIT NULL,
    cti TINYINT,
    csi SMALLINT,
    cin INTEGER,
    cui UNSIGNED INTEGER,
    cbi BIGINT,
    cub UNSIGNED BIGINT,
    cch CHAR(10),
    cvc VARCHAR(10),
    cbn BINARY(8),
    cvb VARBINARY(8),
    clb LONG BINARY,
    clc LONG VARCHAR);

CREATE TABLE t2 (c1 INT PRIMARY KEY,
    cbt BIT NULL,
    cti TINYINT,
    csi SMALLINT,
    cin INTEGER,
    cui UNSIGNED INTEGER,
    cbi BIGINT,
    cub UNSIGNED BIGINT,
```

```
    cch CHAR(10),
    cvc VARCHAR(10),
    cbn BINARY(8),
    cvb VARBINARY(8),
    clb LONG BINARY,
    clc LONG VARCHAR);

CREATE TABLE t4 (c1 INT, cin INT DEFAULT 0x31);

SET TEMPORARY OPTION CONVERSION_MODE = 1;
```

次の要求は、

```
INSERT INTO t1(c1, cvb) SELECT 99, cin FROM T2
WHERE c1=1
```

次のメッセージで失敗します。

```
"Unable to convert column 'cvb' to the requested datatype (varbinary)
from datatype (integer). [-1013043] ['QCA43']"
```

UPDATE に対する暗黙的なバイナリ変換制限モード

CONVERSION_MODE を 1 に設定すると、暗黙的なバイナリ変換制限モードが、特定の種類の **UPDATE** に適用されます。

暗黙的なバイナリ変換制限モードが次に適用されます。

- **UPDATE SET VALUE FROM** *expression* (定数の場合も含みます)
- **UPDATE SET VALUE FROM** *other column*
- **UPDATE SET VALUE FROM** *host variable*
- **JOIN UPDATE SET VALUE FROM** *column of other table*

例

次の要求は、

```
UPDATE t1 SET cbi=cbn WHERE c1=1
```

次のメッセージで失敗します。

```
"Unable to implicitly convert column 'cbi' to datatype (bigint) from
datatype (binary). [-1000187] ['QCB87']"
```

更新可能カーソルによる位置付け INSERT と位置付け UPDATE に対する暗黙的なバイナリ変換制限モード

CONVERSION_MODE を 1 に設定すると、暗黙的なバイナリ変換制限モードが、更新可能カーソルによる、特定の種類の **INSERT** と **UPDATE** に適用されます。

暗黙的なバイナリ変換制限モードが次に適用されます。

- **PUT** *cursor-name* **USING** … *host-variable*
- 別のカラムからの位置付け **UPDATE**

- 定数からの位置付け **UPDATE**
- ホスト変数からの位置付け **UPDATE**

クエリに対する暗黙的なバイナリ変換制限モード

CONVERSION_MODE を 1 に設定すると、暗黙的なバイナリ変換制限モードは、通常、クエリのすべての側面に適用されます。

比較演算子

CONVERSION_MODE = 1 の場合、制限は次の演算子に適用されます。

- =、!=、<、<=、>=、<>、!>、!<
- BETWEEN … AND
- IN

これらが、次の句の検索条件で使用されている場合に適用されます。

- **WHERE** 句
- **HAVING** 句
- **CHECK** 句
- ジョインでの **ON** フレーズ
- **IF**/**CASE** 式

例

次のクエリは、

```
SELECT COUNT(*) FROM T1
WHERE cvb IN (SELECT csi FROM T2)
```

次のメッセージで失敗します。

```
"Invalid data type comparison in predicate  (t1.cvb IN (SELECT
t1.csi ...)), [-1001013] ['QFA13']"
```

文字列関数

CONVERSION_MODE = 1 の場合、制限は次の文字列関数に適用されます。

- **CHAR**
- **CHAR_LENGTH**
- **DIFFERENCE**
- **LCASE**
- **LEFT**
- **LOWER**
- **LTRIM**
- **PATINDEX**

- **RIGHT**
- **RTRIM**
- **SIMILAR**
- **SORTKEY**
- **SOUNDEX**
- **SPACE**
- **STR**
- **TRIM**
- **UCASE**
- **UPPER**

例

次のクエリは、

```
SELECT ASCII(cvb) FROM t1 WHERE c1=1
```

次のメッセージで失敗します。

```
"Data exception - data type conversion is not possible. Argument to
ASCII must be string, [-1009145] ['QFA2E']"
```

次の関数には、文字列引数またはバイナリ引数のいずれかを指定できます。CONVERSION_MODE = 1 の場合、制限はデータ型の混在する引数 (1 つが文字列で、もう 1 つがバイナリ) に適用されます。

- **INSERTSTR**
- **LOCATE**
- **REPLACE**
- **STRING**
- **STUFF**

例

次のクエリは、

```
SELECT STRING(cvb, cvc) FROM t1 WHERE c1=1
```

(ここで、カラム cvb は VARBINARY として、カラム cvc は VARCHAR として定義されています)

次のメッセージで失敗します。

```
"Data exception - data type conversion is not possible. Arguments to
STRING must be all binary  or all string, [-1009145] ['QFA2E']"
```

制限は次の文字列関数に適用されません。

- **BIT_LENGTH**
- **BYTE_LENGTH**

- **CHARINDEX**
- **LENGTH**
- **OCTET_LENGTH**
- **REPEAT**
- **REPLICATE**
- **SUBSTRING**

算術演算と関数
CONVERSION_MODE = 1 の場合、制限は算術演算で使用される次の演算子に適用されます。

+, -, *, /

制限はビット処理式で次のビット処理演算子に適用されます。

& (AND)、| (OR)、^ (XOR)

制限は次の関数の整数引数にも適用されます。

- **ROUND**
- "**TRUNCATE**"
- **TRUNCNUM**

例

次のクエリは、

```
SELECT ROUND(4.4, cvb) FROM t1 WHERE C1=1
```

次のメッセージで失敗します。

```
"Data exception - data type conversion is not possible. Second
Argument to ROUND cannot be converted into an integer, [-1009145]
['QFA2E']"
```

さまざまな関数に対する整数引数
CONVERSION_MODE = 1 の場合、制限は次の関数の整数引数に適用されます。

- **ARGN**
- **SUBSTRING**
- **DATEADD**
- **YMD**

例

次のクエリは、

```
SELECT ARGN(cvb, csi, cti) FROM t1 WHERE c1=1
```

次のメッセージで失敗します。

```
"Data exception - data type conversion is not possible. First
Argument to ARGN cannot be converted to an integer, [-1009145]
['QFA2E']"
```

統計関数、集合関数、数値関数
CONVERSION_MODE = 1 の場合、引数として数値式を要求する統計関数、集合関数、数値関数に、それ以上の制限は適用されません。

CONVERT_VARCHAR_TO_1242 オプション

バージョン 12.4.2 以前の VARCHAR データを圧縮フォーマットに変換します。

指定できる値
ON、OFF

デフォルト値
OFF

スコープ
PUBLIC グループのみに設定できます。**sp_iqcheckdb** (どのモードでも) を実行したときに有効になります。

説明
データベースのデータ圧縮率を高めます。特に可変長文字列が多いデータベースに対して有効です。

このオプションの設定直後に、12.4.2 より前のバージョンで作成された VARCHAR カラムに対してのみ **sp_iqcheckdb** を一度だけ実行してください。

COOPERATIVE_COMMIT_TIMEOUT オプション

トランザクション・ログ内の COMMIT エントリがディスクに書き込まれるタイミングを管理します。

指定できる値
整数値 (ミリ秒単位)

デフォルト値
250

スコープ
個々の接続または PUBLIC グループに設定できます。すぐに有効になります。

説明
このオプションは、COOPERATIVE_COMMITS が ON に設定されているときにかぎり有効です。データベース・サーバは、他の接続がログのページを埋めるのを指定されたミリ秒間だけ待ってから、ディスクに書き込みます。デフォルト設定は 250 ミリ秒です。

参照：
- COOPERATIVE_COMMITS オプション (427 ページ)

COOPERATIVE_COMMITS オプション

コミットされた情報がディスクに書き込まれるタイミングを制御します。

指定できる値
ON、OFF

デフォルト値
ON

スコープ
個々の接続または PUBLIC グループに設定できます。すぐに有効になります。

説明
COOPERATIVE_COMMITS を OFF に設定した場合、**COMMIT** された情報はデータベース・サーバがそれを受信するとすぐにディスクに書き込まれ、その後アプリケーションの継続が許可されます。

COOPERATIVE_COMMITS が ON (デフォルト) の場合、データベース・サーバは **COMMIT** (コミット) された情報をディスクにすぐには書き込みません。その代わり、アプリケーションは COOPERATIVE_COMMIT_TIMEOUT オプションで設定されている最大時間だけ該当ページに対する入力を待ち、その後にコミットされた情報をディスクに書き込みます。

COOPERATIVE_COMMITS を ON に設定し、COOPERATIVE_COMMIT_TIMEOUT の設定を大きくすると、ディスク I/O の数が減少することによってデータベース・サーバの全体的なスループットが向上しますが、個々の接続に対するターンアラウンド・タイムは長くなります。

参照：
- COOPERATIVE_COMMIT_TIMEOUT オプション (426 ページ)

CURSOR_WINDOW_ROWS オプション

バッファするカーソル・ローの数を定義します。

指定できる値
20 – 100000

デフォルト値
200

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
アプリケーションがカーソルを開くと、Sybase IQ は FIFO (先入れ先出し) バッファを生成し、クエリによって生成されたデータ・ローを保持します。
CURSOR_WINDOW_ROWS では、バッファされるローの数を定義します。 カーソルが NO SCROLL 以外のモードで開かれた場合、Sybase IQ はクエリの再開前ならば、バッファできるロー数まで後方スクロールを許可します。 NO SCROLL カーソルの場合は後方スクロールが許可されていないため、上記は該当しません。

たとえば、このオプションのデフォルト値のままの場合、最初はクエリ結果セットのロー 1 から 200 までが保持されます。 ここで先頭の 300 ローをフェッチすると、バッファにはロー 101 から 300 までが保持されます。このバッファに対してはほとんどオーバヘッドなしに前方または後方へのスクロールができます。 ロー 101 より前のローまでスクロールすると、Sybase IQ はバッファに必要なローが含まれるまでクエリを再開します。 これはパフォーマンスの浪費なので、アプリケーションの設計時に、できるだけ避けるようにしてください。
CURSOR_WINDOW_ROWS の値を大きくして、スクロール・エリアをできるだけ大きくするという方法もありますが、200 というデフォルト設定は、ほとんどのアプリケーションで十分な値です。

DATE_FIRST_DAY_OF_WEEK オプション

1 週間の始まりを何曜日にするかを指定します。

指定できる値
0 – 6

デフォルト値
0 (日曜日)

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
1 週間の始まりの曜日をこのオプションを使用して指定します。デフォルトでは、日曜日が第 1 日、月曜日が第 2 日、火曜日が第 3 日、のようになります。

表 22 : DATE_FIRST_DAY_OF_WEEK の有効な値

値	第 1 日
0	日曜日
1	月曜日
2	火曜日
3	水曜日
4	木曜日
5	金曜日
6	土曜日

たとえば、DATE_FIRST_DAY_OF_WEEK の値を 3 に変更すると、水曜日が第 1 日、木曜日が第 2 日となります。このオプションの影響を受けるのは、**DOW** 関数と **DATEPART** 関数のみです。

SQL Anywhere のオプション FIRST_DAY_OF_WEEK は、これと同じ機能を持ちますが、割り当てる値が 0 から 6 ではなく 1 から 7 です。1 は月曜日、7 は日曜日 (デフォルト値) です。

DATE_FORMAT オプション

データベースから取得した日付のフォーマットを設定します。

指定できる値
文字列

デフォルト値
'YYYY-MM-DD'。これは ISO 日付フォーマット仕様準拠です。

スコープ
個々の接続または PUBLIC グループに設定できます。すぐに有効になります。

説明
フォーマットは次の記号を組み合わせた文字列です。

表 23 : DATE_FORMAT 文字列で使用される記号

記号	説明
yy	西暦年 2 桁。
yyyy	西暦年 4 桁。
mm	2 桁の月、またはコロンの後の場合 (hh:mm など) は 2 桁の分。
mmm	3 文字で示す月の名前。
mmmm[m...]	月を示す長い文字列—m の数の文字を使用する。指定された m の数の文字で月を示す。ただし、m の数が、月の名前を超えない範囲に限定される。
d	曜日を示す 1 桁の数字 (0 = 日曜、6 = 土曜)。
dd	2 桁の日。
ddd	3 文字で示す曜日。
dddd[d...]	曜日を示す長い文字列—d の数の文字を使用する。指定された d の数の文字で曜日を示す。ただし、d の数が、曜日の名前を超えない範囲に限定される。
jjj	1 から 366 までの年日数。

注意： 日付フォーマット文字列の中でマルチバイト文字は使用できません。データベースの照合順が 932JPN などのマルチバイト照合順であっても、使用できるのは 1 バイト文字だけです。日付フォーマットの文字列でマルチバイト文字を使用するには、連結演算子を使用します。たとえば、'*?* がマルチバイト文字の場合、次のよう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文字列の外に移動します。

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

各記号は、フォーマットされた日付の対応するデータで置き換えられます。数字の出力ではなく文字を表すフォーマット記号は大文字で入力でき、その場合、置き換えられる文字も大文字になります。数字の場合、フォーマット文字列に大文字と小文字を混在させると、出力に先行ゼロが付きません。

記号の大文字、小文字を変更することで、数字の埋め込みを制御できます。大文字か小文字でそろえると (MM、mm、DD、dd) 空いたフィールドに 0 が埋め込まれ

ます。大文字と小文字を混ぜると (Mm、mM、Dd、dD)、0 は埋め込まれません。各表示は必要なスペースだけを使用します。次に例を示します。

```
SELECT dateformat ( cast ('2011/01/01' as date ), 'yyyy/Mm/Dd' )
```

この例では、次の値が返されます。

```
2011/1/1
```

例

DATE_FORMAT 設定と、その設定で 2011 年 5 月 21 日 (土) に次の文が実行された場合の結果を示します。

```
SELECT CURRENT DATE
```

表 24 : DATE_FORMAT の設定

DATE_FORMAT	SELECT CURRENT DATE
yyyy/mm/dd/ddd	2011/05/21/sat
jjj	141
mmm yyyy	may 2011
mm-yyyy	05-2011

参照：

- RETURN_DATE_TIME_AS_STRING オプション (520 ページ)
- TIME_FORMAT オプション (548 ページ)

DATE_ORDER オプション

日付フォーマットの解釈を制御します。

指定できる値

'MDY'、'YMD'、または 'DMY'

デフォルト値

'YMD'これは ISO 日付フォーマット仕様準拠です。

説明

DATE_ORDER は、10/11/12 が 1912 年 10 月 11 日なのか、1910 年 11 月 12 日なのか、それとも 1912 年 11 月 10 日なのかを指定するために使用します。'MDY'、'YMD'、または 'DMY' を指定できます。

DBCC_LOG_PROGRESS オプション

sp_iqcheckdb システム・ストアド・プロシージャの進捗をレポートします。

指定できる値
ON、OFF

デフォルト値
OFF

スコープ
個々の接続または PUBLIC グループに設定できます。次回 **sp_iqcheckdb** を実行したときに有効になります。

説明
DBCC_LOG_PROGRESS が ON の場合、**sp_iqcheckdb** システム・ストアド・プロシージャは進行メッセージを IQ メッセージ・ファイルに送信します。これらのメッセージによって、**sp_iqcheckdb** 操作の進行を追跡できます。

ストアド・プロシージャについては、『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』を参照してください。

例
次は、sp_iqcheckdb 'check database' コマンドの進捗ログの出力例です。

```
IQ Utility Check Database Start CHECK STATISTICS table: tloansf Start
CHECK STATISTICS for field: aqsn_dt Start CHECK STATISTICS processing
index: IQ_IDX_T444_C1_FP Start CHECK STATISTICS processing index:
tloansf_aqsn_dt_HNG Done CHECK STATISTICS field: aqsn_dt
```

次は、sp_iqcheckdb 'allocation table nation' コマンドの進捗ログの出力例です。

```
Start ALLOCATION table: nation Start ALLOCATION processing index:
nationhg1 Done ALLOCATION table: nation Done ALLCOATION processing
index: nationhg1
```

DBCC_PINNABLE_CACHE_PERCENT オプション

sp_iqcheckdb システム・ストアド・プロシージャが使用するキャッシュの割合を制御します。

指定できる値
0 – 100

デフォルト値
50

スコープ
個々の接続または PUBLIC グループに設定できます。次回 **sp_iqcheckdb** を実行したときに有効になります。

説明
sp_iqcheckdb システム・ストアド・プロシージャは、このオプションで指定された固定数のバッファを使用して動作します。**sp_iqcheckdb** のパフォーマンスを最大限に向上させるため、デフォルトでキャッシュの大部分は予約されています。

ストアド・プロシージャについては、『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』を参照してください。

sp_iqcheckdb のトラブルシューティングの詳細については、『システム管理ガイド：第2巻』の「システムのリカバリとデータベースの修復」>「データベースの検証」>「sp_iqcheckdb 実行時のリソースの問題」を参照してください。

DEBUG_MESSAGES オプション

DEBUG ONLY 句を含む **MESSAGE** 文を実行するかどうかを制御します。

指定できる値
ON、OFF

デフォルト値
OFF

説明
このオプションは、DEBUG ONLY 句が指定された **MESSAGE** 文を含むストアド・プロシージャ内のデバッグ・メッセージの動作を制御できるようにします。このオプションはデフォルトで OFF に設定されており、**MESSAGE** 文が実行された場合にデバッグ・メッセージは表示されません。DEBUG_MESSAGES を ON に設定すると、すべてのストアド・プロシージャでデバッグ・メッセージを有効にできます。

注意： DEBUG ONLY を指定したメッセージは、DEBUG_MESSAGES オプションが OFF に設定されているときコストがかからないため、これらの文は通常、実稼働システムのストアド・プロシージャに残されます。ただし、頻繁に実行されるストアド・プロシージャではあまり使用しないでください。不用意に使用すると、パフォーマンスが多少低下する可能性があります。

参照：
- MESSAGE 文 (298 ページ)

DEDICATED_TASK オプション

要求処理のタスクが、1 つの接続からの要求だけを処理するようにします。

指定できる値
ON、OFF

デフォルト値
OFF

スコープ
現在の接続の継続中に、temporary オプションとしてのみ設定できます。このオプションを設定するには、DBA パーミッションが必要です。

説明
DEDICATED_TASK 接続オプションを ON に設定すると、要求処理タスクは、その接続の要求処理専用となります。あらかじめこのオプションを有効にして接続を確立すると、データベース・サーバが応答しなくなった場合に、サーバの状態に関する情報を収集できます。

DEFAULT_DBSPACE オプション

テーブルまたはジョイン・インデックスが作成されるデフォルトの DB 領域を変更します。

指定できる値
DB 領域名を含む文字列

デフォルト値
'' (空の文字列)

スコープ
個々の接続または PUBLIC グループに設定できます。設定は、すぐに有効になります。現在のユーザ以外のグループまたはユーザのオプションを設定するには、DBA パーミッションが必要です。すぐに有効になります。

説明
DEFAULT_DBSPACE は、管理者によるグループまたはユーザのデフォルト DB 領域の設定、またはユーザによる独自のデフォルト DB 領域の設定を可能にします。

SYSTEM を指定するテーブル IN 句が使用されていないかぎり、常に IQ_SYSTEM_TEMP がグローバル・テンポラリ・テーブルに使用されます。その場合、SA グローバル・テンポラリ・テーブルが作成されます。

データベースの作成時に、PUBLIC.DEFAULT_DBSPACE オプションの設定が空、または明示的に IQ_SYSTEM_MAIN に設定されている場合、システム DB 領域である IQ_SYSTEM_MAIN が作成されて暗黙に定義されます。データベースの作成直後に、管理者は第 2 のメイン DB 領域を作成し、DB 領域 IQ_SYSTEM_MAIN の CREATE 権限を PUBLIC から取り消し、新しいメイン DB 領域のために DB 領域の CREATE 権限を特定ユーザまたは PUBLIC に付与し、PUBLIC.DEFAULT_DBSPACE を新しいメイン DB 領域に設定することをおすすめします。次に例を示します。

```
CREATE DBSPACE user_main USING FILE user_main
'user_main1' SIZE 10000;
GRANT CREATE ON user_main TO PUBLIC;
REVOKE CREATE ON IQ_SYSTEM_MAIN FROM PUBLIC;
SET OPTION PUBLIC.DEFAULT_DBSPACE = 'user_main';
```

例

この例では、すべての DB 領域の CONNECT 権限と RESOURCE 権限は、ユーザの usrA と usrB に付与され、各ユーザには特定の DB 領域の CREATE 権限が付与されます。

```
GRANT CONNECT, RESOURCE TO usrA, usrB
   IDENTIFIED BY pwdA, pwdB;
GRANT CREATE ON dbsp1 TO usrA;
GRANT CREATE ON dbsp3 TO usrB;
SET OPTION “usrA”.default_dbspace = ‘dbsp1’;
SET OPTION “usrB”.default_dbspace = ‘dbsp3’;
SET OPTION “PUBLIC”.default_dbspace = dbsp2;

CREATE TABLE “DBA”.t1(c1 int, c2 int);
INSERT INTO t1 VALUES (1, 1);
INSERT INTO t1 VALUES (2, 2);
COMMIT;
```

UsrA が以下を実行します。

```
CREATE TABLE “UsrA”.t1(c1 int, c2 int);
INSERT INTO t1 VALUES (1, 1);
INSERT INTO t1 VALUES (2, 2);
COMMIT;
```

UsrB が以下を実行します。

```
CREATE TABLE “UsrB”.t1(c1 int, c2 int);
INSERT INTO t1 VALUES (1, 1);
INSERT INTO t1 VALUES (2, 2);
COMMIT;
```

DBA が以下を実行します。

```
SELECT Object, DbspaceName, ObjSize
FROM sp_iqindexinfo();
```

sp_iqindexinfo の結果は次のようになります。

```
DBA.t1                              dbsp2        200k
DBA.t1.ASIQ_IDX_T730_C1_FP          dbsp2        288k
DBA.t1.ASIQ_IDX_T730_C2_FP          dbsp2        288k
usrA.t1                             dbsp1        200k
usrA.t1.ASIQ_IDX_T731_C1_FP         dbsp1        288k
usrA.t1.ASIQ_IDX_T731_C2_FP         dbsp1        288k
usrB.t1                             dbsp3        200k
usrB.t1.ASIQ_IDX_T732_C1_FP         dbsp3        288k
usrB.t1.ASIQ_IDX_T732_C2_FP         dbsp3        288k
```

DEFAULT_DISK_STRIPING オプション

すべての DB 領域にディスク・ストライピングの値を設定します。

指定できる値
ON、OFF

デフォルト値
ON

スコープ
PUBLIC グループのみに設定できます。DBA パーミッションが必要です。

説明
デフォルトでは、IQ メイン・ストアにあるすべての DB 領域のディスク・ストライピングは ON になります。**CREATE DBSPACE** がストライピングを指定しない場合、このオプションは、**CREATE DBSPACE** によってのみ使用され、デフォルトのストライピング値を定義します。

参照：

- CREATE DBSPACE 文 (93 ページ)

DEFAULT_HAVING_SELECTIVITY_PPM オプション

ほとんどの **HAVING** 句のデフォルトの選択性の見積もりをオプティマイザに提供します (100 万分の 1 単位)。

指定できる値
0 – 1000000

デフォルト値
0

スコープ
個々の接続または PUBLIC グループに設定できます。すぐに有効になります。

説明
DEFAULT_HAVING_SELECTIVITY_PPM は、**HAVING** 句の選択性を設定し、オプティマイザによる見積もりを上書きします。**HAVING** 句は **GROUP BY** 句、または集合関数だけで構成されている select リストを指定したクエリの結果をフィルタします。DEFAULT_HAVING_SELECTIVITY_PPM がデフォルトの 0 に設定されている場合、オプティマイザは **HAVING** 句によってフィルタされるローの数を推定します。IQ オプティマイザに十分な情報がなく、正確な選択性を決められない場合、一般的な見積もり値の 40% が採用されます。
DEFAULT_HAVING_SELECTIVITY_PPM を使用して、クエリ内のすべての **HAVING** 述部に対するオプティマイザの見積もりを上書きできます。

ユーザは、クエリ内の個々の HAVING 句の選択性を指定することもできます。詳細については、『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』を参照してください。

DEFAULT_ISQL_ENCODING オプション [Interactive SQL]

READ 文と **OUTPUT** 文が使用するコード・ページを指定します。

指定できる値
identifier または *string*

デフォルト値
システム・コード・ページを使用 (空の文字列)

スコープ
現在の接続の継続中に、テンポラリ・オプションとしてのみ設定できます。

説明
DEFAULT_ISQL_ENCODING は、ファイルの読み込み／書き込み時に使用するコード・ページの指定に使用されます。永続的に設定することはできません。デフォルトのコード・ページは、実行中のプラットフォームのデフォルトのコード・ページです。英語版 Windows のマシンでは、デフォルトのコード・ページは 1252 です。

Interactive SQL は、特定の **OUTPUT** 文または **READ** 文で使用するコード・ページを次のリストの順で決定します (先に出てきたコード・ページが優先されます)。

- **OUTPUT** 文または **READ** 文の **ENCODING** 句で指定されているコード・ページ
- DEFAULT_ISQL_ENCODING オプションで指定されたコード・ページ (このオプションが設定されている場合)
- Interactive SQL を実行しているコンピュータのデフォルトのコード・ページ

サポートされているコード・ページのリストについては、『SQL Anywhere サーバ – データベース管理』の「データベースの設定」>「国際言語と文字セット」>「文字セットと照合の参考情報」>「サポートされている照合と代替照合」を参照してください。

また、『SQL Anywhere サーバ – データベース管理』の「データベースの設定」>「国際言語と文字セット」>「文字セットの知識」>「文字セットの知識」を参照してください。

注意： これらのリファレンスは SQL Anywhere マニュアルにリンクされています。

例
コード化を UTF-16 (Unicode ファイルの読み込み用) に設定します。

```
SET TEMPORARY OPTION DEFAULT_ISQL_ENCODING = 'UTF-16'
```

参照：
- OUTPUT 文 [Interactive SQL] (304 ページ)
- READ 文 [Interactive SQL] (316 ページ)

DEFAULT_KB_PER_STRIPE オプション

書き込み操作が次のストライプに移動する前にデータが書き込まれるストライプの量の上限スレッショルドを KB 単位で設定します。

この設定は、IQ メイン・ストアのすべての DB 領域のデフォルト・サイズです。

指定できる値
1 以上の整数

デフォルト値
1

スコープ
PUBLIC グループのみに設定できます。DBA パーミッションが必要です。

説明
デフォルト値の 1KB は、1 ページが圧縮され、圧縮されたページが 1 回の操作でディスクに書き込まれることを意味します。指定したページ・サイズに関係なく、次の操作は、DB 領域の次の dbfile に書き込みます。

次のストライプに移動する前に複数のページを同じストライプに書き込むには、DEFAULT_KB_PER_STRIPE 設定を変更します。たとえば、ページ・サイズが 128KB である場合、DEFAULT_KB_PER_STRIPE を 512KB に設定すると、Sybase IQ はページ書き込みをキューに保留し、圧縮されたページのサイズが 512KB を超えると、ディスクに書き込みます。

CREATE DBSPACE でストライプ・サイズが指定されていない場合、このオプションは、**CREATE DBSPACE** によってのみ使用され、IQ メイン・ストアの DB 領域のデフォルトのストライピング・サイズを定義します。

参照：
- CREATE DBSPACE 文 (93 ページ)

DEFAULT_LIKE_MATCH_SELECTIVITY_PPM オプション

ほとんどの **LIKE** 述部のデフォルトの選択性の見積もりをオプティマイザに提供します (100 万分の 1 単位)。

指定できる値
0 ～ 1000000

デフォルト値
150000

スコープ
個々の接続または PUBLIC グループに設定できます。すぐに有効になります。

説明
DEFAULT_LIKE_MATCH_SELECTIVITY_PPM は、たとえば、LIKE '*string* *%string*' (% はワイルドカード文字) のような **LIKE** 述部のデフォルトの選択性を設定します。

他に選択性に関する情報が提供されておらず、一致文字列が、一連の定数文字と 1 つのワイルドカードで始まっていない場合、オプティマイザはこのオプションを参照します。

カラムに、LF インデックスまたは 1 バイト、2 バイト、3 バイト FP インデックスのいずれかがある場合、オプティマイザは正しい値を得ることができるため、この値を使用する必要はありません。

ユーザが、クエリ内で選択性を指定することもできます。ユーザ指定の条件ヒントの詳細については、『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』を参照してください。

参照：

- DEFAULT_LIKE_RANGE_SELECTIVITY_PPM オプション (440 ページ)
- FP_LOOKUP_SIZE オプション (448 ページ)

DEFAULT_LIKE_RANGE_SELECTIVITY_PPM オプション

先頭の定数 **LIKE** 述部のデフォルトの選択性の見積もりをオプティマイザに提供します (100 万分の 1 単位)。

指定できる値
1000000 ～ 1

デフォルト値
150000

スコープ
個々の接続または PUBLIC グループに設定できます。すぐに有効になります。

説明
DEFAULT_LIKE_RANGE_SELECTIVITY_PPM は、LIKE '*string%*' という形の **LIKE** 述部のデフォルトの選択性を設定します。ここで、一致文字列は一連の定数文字とワイルドカード文字 (%) 1 つで構成されます。選択性に関する情報が提供されていない場合、オプティマイザはこのオプションを参照します。

カラムに、LF インデックスまたは 1 バイト、2 バイト、3 バイト FP インデックスのいずれかがある場合、オプティマイザは正しい値を得ることができるため、この値を使用する必要はありません。

ユーザが、クエリ内で選択性を指定することもできます。ユーザ指定の条件ヒントの詳細については、『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』を参照してください。

参照：

- DEFAULT_LIKE_MATCH_SELECTIVITY_PPM オプション (439 ページ)
- FP_LOOKUP_SIZE オプション (448 ページ)

DEFAULT_PROXY_TABLE_ROW_COUNT オプション

プロキシ・テーブルから返されるローの数のデフォルト見積もりを上書きできます。

指定できる値
0 ～ 4294967295

デフォルト値
200000

スコープ
このオプションを設定するには、DBA パーミッションが必要です。PUBLIC グループに対してのみ一時的に設定できます。

DEFAULT_TABLE_UDF_ROW_COUNT オプション

テーブル UDF (C、C++、または Java テーブル UDF) から返されるローの数のデフォルト見積もりを上書きできます。

指定できる値
0 ～ 4294967295

デフォルト値
200000

スコープ
このオプションを設定するには、DBA パーミッションが必要です。PUBLIC グループに対してのみ一時的に設定できます。

説明
テーブル UDF では、**DEFAULT_TABLE_UDF_ROW_COUNT** オプションを使用して、テーブル UDF が返すローの数の見積もりをクエリ・プロセッサに渡すことができます。これは、Java テーブル UDF の場合、この情報を伝達する唯一の方法です。ただし、C または C++ テーブル UDF の場合、UDF の開発者は、`EXTFNAPIV4_DESCRIBE_PARM_TABLE_NUM_ROWS` 記述パラメータを使用して `describe` フェーズ内でこの情報をパブリッシュすることによって、返される見込みのローの数をパブリッシュすることを考慮する必要があります。`EXTFNAPIV4_DESCRIBE_PARM_TABLE_NUM_ROWS` の値は、**DEFAULT_PROXY_TABLE_UDF_ROW_COUNT** オプションの値を常に上書きします。

DELAYED_COMMIT_TIMEOUT オプション

COMMIT の後でサーバが制御をアプリケーションにいつ戻すかを決定します。

指定できる値
整数値 (ミリ秒単位)

デフォルト値
500

説明
DELAYED_COMMITS は OFF 以外の値をとらないので、このオプションは Sybase IQ からは無視されます。

DELAYED_COMMITS オプション

COMMIT の後でサーバが制御をアプリケーションにいつ戻すかを決定します。

指定できる値
OFF

デフォルト値
OFF。これが ISO の COMMIT 動作に準拠しています。

説明
OFF に設定すると (Sybase IQ では OFF しか使用できません)、アプリケーションは **COMMIT** の結果がディスクに書き込まれるまで待たなければなりせん。 ANSI/ISO の COMMIT の動作を得るには、このオプションを OFF に設定する必要があります。

DISABLE_RI_CHECK オプション

ロード、挿入、更新、削除の操作で参照整合性のチェックを省略することを許可し、パフォーマンスを向上させます。

指定できる値
ON、OFF

デフォルト値
OFF

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
DISABLE_RI_CHECK が ON に設定されている場合、要求中に参照整合性の違反が起きないことをユーザの責任において保証する必要があります。

DIVIDE_BY_ZERO_ERROR オプション [TSQL]

ゼロ除算のレポートを制御します。

指定できる値
ON、OFF

デフォルト値
ON

スコープ
このオプションでは、ゼロ除算をエラーとしてレポートするかどうかを指示します。オプションを ON に設定すると、ゼロ除算は SQLSTATE 22012 のエラーになります。

このオプションを OFF に設定すると、ゼロ除算はエラーになりません。NULL が返されます。

DQP_ENABLED オプション

テンポラリ・データベース・オプション **dqp_enabled** を使用すると、接続レベルで DQP を有効または無効にすることができます。

テンポラリ・データベース・オプション **dqp_enabled** を OFF に設定することによって、現在の接続における DQP を無効にすることができます。このオプションを ON (デフォルト値) に設定することによって現在の接続における DQP を有効にすることができますが、それが可能なのは現在の接続の論理サーバに対するユーザのログイン・ポリシーによって、そのユーザに対して DQP が有効になっている場合のみです。

セカンダリ・サーバを実行するには、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。**dqp_enabled** 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。

EARLY_PREDICATE_EXECUTION オプション

クエリの最適化の前に簡単なローカル述部を実行するかどうかを制御します。

指定できる値
ON、OFF

デフォルト値
ON

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
このオプションが ON (デフォルト値) の場合、オプティマイザは、クエリ・プラン内の "Estimated Result Rows" の値をより正確なものにするために、ジョインの順序付け、ジョイン・アルゴリズムの選択、グループ化アルゴリズムの選択などを行ってクエリを最適化する前に、ローカル・カラムと制約のみを含む述部を検索、準備、実行します。このオプションが OFF の場合、オプティマイザは単純な述部を検索および準備しますが、クエリの最適化の前にこれらを実行することはありません。述部を実行しないと、"Estimated Result Rows" の最終的な値の正確さは劣ります。

EARLY_PREDICATE_EXECUTION オプションは多くのクエリのクエリ・プランを向上させるため、通常は常に ON にしておきます。

EARLY_PREDICATE_EXECUTION オプションが ON の場合、NOEXEC オプションが ON の場合でも、Sybase IQ はクエリ・プランの生成に先立ってすべてのクエリのローカル述部を実行します。生成されたクエリ・プランは実行時プランと同一です。

ルート・ノードに関する次の情報がクエリ・プランに含まれます。

- ローカルの不変の述部を実行するのに使われるスレッド。1 より大きな数の場合、ローカルの不変の述部を平行して実行します。
- Early_Predicate_Execution：オプションが OFF かどうかを示します。
- Time of Cursor Creation： カーソルが作成された時間。

このオプションが実行を制御する単純な述部は、クエリ・プラン内では不変の述部として扱われます。リーフ・ノード上にローカルの不変の述部がある場合、そのリーフ・ノードに関する次の情報がクエリ・プランに含まれます。

- Generated Post Invariant Predicate Rows： ローカルの不変の述部を実行した後の実際の結果。
- Estimated Post Invariant Predicate Rows：ローカルの不変の述部の推定選択性によって算出されます。
- Time of Condition Start：ローカルの不変の述部の実行を開始した時間。
- Time of Condition Done：ローカルの不変の述部の実行が完了した時間。
- Elapsed Condition Time： ローカルの不変の述部を実行する際の経過時間。

ENABLE_LOB_VARIABLES オプション

ラージ・オブジェクト変数のデータ型変換を制御します。

ラージ・オブジェクト変数を使用するには、非構造化データ分析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。 ENABLE_LOB_VARIABLES の構文と詳細な説明については、『Sybase IQ の非構造化データ分析の概要』を参照してください。

EXTENDED_JOIN_SYNTAX オプション

マルチテーブル・ジョインに対してあいまいな構文を持つクエリを許可するか、またはエラーとして報告するかを制御します。

指定できる値
ON、OFF

デフォルト値
ON

説明
このオプションを使用して、NULL 入力テーブルに重複する相関名があるために構文があいまいになっている外部ジョインを含むクエリに対して、構文エラーを報告します。

報告されるのは、次のようなジョイン句があるクエリです。ここで C1 は条件です。

```
( R left outer join T , T  join S on ( C1 ) )
```

EXTENDED_JOIN_SYNTAX を ON に設定すると、このクエリは次のように解釈されます。ここで、C1 と C2 は条件です。

```
( R left outer join T on ( C1 ) ) join S on ( C2 )
```

FORCE_DROP オプション

Sybase IQ が **DROP** コマンドの実行中にデータベースのディスク領域を再利用するのではなくリークすることを指定します。

指定できる値
ON、OFF

デフォルト値
OFF

スコープ
このオプションを設定するには、DBA パーミッションが必要です。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
破損インデックス、ジョイン・インデックス、カラム、またはテーブルを DROP する必要がある場合は、FORCE_DROP オプションを ON に設定してください。DROP 対象のオブジェクト中の不正または不正な可能性があるファイル領域割り付け情報をもとに、フリー・リストが不正に更新されるのを防ぐことができます。破損オブジェクトを DROP した後に、サーバ・スイッチの **-iqfrec** と **-iqdroplks** を使用して、ファイル領域を再利用できます。

オブジェクトを強制削除するときは、DBA 以外にデータベースに接続しているユーザがいないことを確認してください。強制削除の後、サーバをすぐに再起動する必要があります。

Sybase 製品の保守契約を結んでいるサポート・センタから指示があった場合を除き、オブジェクトを強制削除しないでください。

システム・リカバリとデータベース修復のための FORCE_DROP プロシージャについては、『システム管理ガイド：第 1 巻』を参照してください。

FORCE_NO_SCROLL_CURSORS オプション

すべてのカーソルをスクロールしないよう強制的に設定します。

指定できる値
ON、OFF

デフォルト値
OFF

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
デフォルトでは、すべてのカーソルはスクロール可能です。ホスト変数が宣言されていないスクロール・カーソルに対して、Sybase IQ は結果を一時的に保持するためのバッファを生成します。 結果セットの各ローは、後方スクロールを可能にするためにこのバッファに保持されます。

FORCE_NO_SCROLL_CURSORS を ON に設定すると、一時的に使用されるメモリが削減されます。このオプションは、大量 (数百万) のローを取得する場合に役立ちます。ただし、フロントエンド・アプリケーションで後方スクロール・カーソル・オペレーションが頻繁に使用される場合は、このオプションを OFF に設定した方が、クエリの応答が速くなります。

フロントエンド・アプリケーションで後方スクロールがほとんど行われない場合は、FORCE_NO_SCROLL_CURSORS = 'ON' を永続的な PUBLIC オプションに設定します。メモリの節約になるため、クエリのパフォーマンスが向上します。

FORCE_UPDATABLE_CURSORS オプション

更新可能として宣言されていないカーソルを更新可能にするかどうかを制御します。

指定できる値
ON、OFF

デフォルト値
OFF

スコープ
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。DBA パーミッションは必要ありません。すぐに有効になります。

説明
FORCE_UPDATABLE_CURSORS オプションが ON のとき、更新可能として宣言されなかったカーソルを更新できます。このオプションによって、フロントエンドのアプリケーションで **DECLARE CURSOR** 文の **FOR UPDATE** 句を指定しなくても更新可能カーソルを使えるようになります。

FORCE_UPDATABLE_CURSORS は、どうしても必要な場合を除き使用しないようにしてください。

FP_LOOKUP_SIZE オプション

Sybase IQ で使用されるルックアップ・ページの最大数を指定します。ルックアップ FP インデックス、特に FP(3) インデックスの作成に割り付けられたキャッシュの量を制御します。

指定できる値
1MB ～ 4096MB

デフォルト値
16 MB

スコープ
このオプションを設定するには、DBA パーミッションが必要です。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
FP_LOOKUP_SIZE オプションは ルックアップ・ページの最大数を制御します。

FP_LOOKUP_SIZE オプションは public に設定する必要があり、許容される構文は次のとおりです。

```
SET OPTION public.FP_LOOKUP_SIZE = 1
```

次のデータベース・オプションは、3 バイトのインデックスをサポートしています。

- INDEX_ADVISOR
- MINIMIZE_STORAGE
- FP_LOOKUP_SIZE_PPM

次のストアド・プロシージャは、3 バイトのインデックスをサポートしています。

- **sp_iqcheckdb**
- **sp_iqcolumn**
- **sp_iqindexadvice**
- **sp_iqindexmetadata**
- **sp_iqindexsize**
- **sp_iqindex**
- **sp_iqindexfragmentation**
- **sp_iqrebuildindex**

- **sp_iqrowdensity**

参照：
- FP_LOOKUP_SIZE_PPM オプション (449 ページ)
- MINIMIZE_STORAGE オプション (490 ページ)

FP_LOOKUP_SIZE_PPM オプション

Sybase IQ での FP ルックアップ・ストレージ・サイズを制限します (メイン・メモリの 100 万分の 1 単位)。

指定できる値
1000000 ～ 1

デフォルト値
2500

スコープ
このオプションを設定するには、DBA パーミッションが必要です。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
FP_LOOKUP_SIZE_PPM は、すべての FP ルックアップ・インデックスのルックアップ FP インデックス、特に FP(3) インデックス作成に割り付けられたメイン・キャッシュの量を制御します。

ルックアップ・ページの最大数を制御し、この数を制限します (メイン・メモリの 100 万分の 1 単位)。つまり FP_LOOKUP_SIZE_PPM * size of main memory / 1,000,000 の値で、メイン・メモリのサイズは、サーバのスタートアップ・パラメータ **-iqmc** によって指定されたものです。

次のオプションは、3 バイトのインデックスをサポートしています。

- FP_LOOKUP_SIZE
- INDEX_ADVISOR
- MINIMIZE_STORAGE

参照：
- FP_LOOKUP_SIZE オプション (448 ページ)
- MINIMIZE_STORAGE オプション (490 ページ)

FP_PREDICATE_WORKUNIT_PAGES オプション

デフォルト・インデックスで使用される並列性の度合いを指定します。

指定できる値
整数値

デフォルト値
200

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
デフォルト・インデックスの作成時には SUM、RANGE、MIN、MAX、COUNT DISTINCT などの述部が並列的に計算されます。
FP_PREDICATE_WORKUNIT_PAGES は、各スレッドが操作するページ数を指定することで、並列性の度合いを制御します並列度を上げたい場合は、このオプションの値を小さくします。

FPL_EXPRESSION_MEMORY_KB オプション

列挙記憶領域を持つカラムに対する関数式を含むクエリの最適化に使用するメモリを制御します。

指定できる値
0 – 20000

デフォルト値
1024 キロバイト

スコープ
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
FPL_EXPRESSION_MEMORY_KB は、列挙記憶領域を持つカラムに対する関数式を含むクエリの最適化に使用するメモリを制御します。このオプションによって、DBA は、この最適化で使用されるメモリ量を制限し、他の Sybase IQ メモリ要件 (キャッシュなど) とのバランスを取ることができます。 このオプションを 0 に設定すると、最適化が無効になります。

GARRAY_FILL_FACTOR_PERCENT オプション

既存のグループに対するインクリメンタルな挿入を今後行うために予約しておく領域が各 **HG** garray ページに占める割合を指定します。

指定できる値
0 – 1000

デフォルト値
25

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
garray は、各グループを増大することにより、この値によって設定された空き領域の埋め込みを含めようとします。この領域は、既存のインデックス・グループに追加するロー用に使用されます。

HG インデックスに対して、グループごとに決められた大きさの記憶領域を確保できます (ここで、グループは同じ値を持つローのまとまりとして定義されているとします)。領域を確保するには追加のディスク領域が必要となりますが、**HG** インデックスへのインクリメンタルな挿入処理のパフォーマンスが向上します。

HG インデックスに対するインクリメンタルな挿入が今後も発生し、またそれによってできる新しいローの値が以前のインデックスにすでに存在する場合は、このオプションに 0 以外の値を設定すると、インクリメンタルな挿入処理のパフォーマンスが向上します。

インデックスをインクリメンタルに更新する予定がない場合は、このオプションの値を小さくして、ディスク領域を節約できます。

参照：
- GARRAY_PAGE_SPLIT_PAD_PERCENT オプション (452 ページ)

GARRAY_INSERT_PREFETCH_SIZE オプション

プリフェッチに使用するページ数を指定します。

指定できる値
0 – 100

デフォルト値
3

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
このオプションを使用して、**HG** インデックスのあるカラムに対する挿入動作中に実行されるデータベースの先読みページ数を指定します。

Sybase 製品の保守契約を結んでいるサポート・センタからの指示があった場合を除いて、このオプションは使用しないでください。

参照：
- GARRAY_FILL_FACTOR_PERCENT オプション (451 ページ)

GARRAY_PAGE_SPLIT_PAD_PERCENT オプション

1 ページあたりのフィル・ファクタを garray に対するページ分割時に決定し、インクリメンタルな挿入を今後行うために予約しておく領域が各 **HG** garray ページに占める割合を指定します。

指定できる値
0 – 100

デフォルト値
25

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
garray ページの分割は、このパーセンテージを空のままにしようとします。この領域は、新しいインデックス・グループに追加するロー用に使用されます。

HG インデックスは、ページ・レベルで記憶領域を予約できます。この記憶領域は、追加のローが挿入されたときに新しいグループに割り当てることができます。領域を確保するには追加のディスク領域が必要となりますが、**HG** インデックスへのインクリメンタルな挿入処理のパフォーマンスが向上します。

インクリメンタルな挿入を **HG** インデックスに行う予定があり、それによってできる新しいローの値が以前のインデックスにまだ存在しない場合は、GARRAY_PAGE_SPLIT_PAD_PERCENT オプションに 0 以外の値を設定すると、インクリメンタルな挿入処理のパフォーマンスが向上します。

インデックスをインクリメンタルに更新する予定がない場合は、このオプションの値を小さくして、ディスク領域を節約できます。

参照：

- GARRAY_FILL_FACTOR_PERCENT オプション (451 ページ)

GARRAY_RO_PREFETCH_SIZE オプション

プリフェッチに使用するページ数を指定します。

指定できる値
0 – 100

デフォルト値
10

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
このオプションを使用して、**HG** インデックスのあるカラムに対するクエリ動作中に実行されるデータベースの先読みページ数を指定します。

Sybase 製品の保守契約を結んでいるサポート・センタからの指示があった場合を除いて、このオプションは使用しないでください。

HASH_PINNABLE_CACHE_PERCENT オプション

ハッシュ・オブジェクトが保持できるユーザの一時メモリの最大パーセンテージを制御します。

指定できる値
0 – 100

デフォルト値
20

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
HASH_PINNABLE_CACHE_PERCENT オプションは、任意のハッシュ・オブジェクトがメモリ内に保持できる、ユーザの一時メモリ割り付けのパーセンテージを制御します。デフォルトは 20% ですが、複雑なクエリを処理するサイトに対しては 10% と低く設定し、大きな **IN** サブクエリのように、大きなハッシュ・オブジェクトを 1 つだけ実行する単純なクエリを処理するサイトには 50% と高く設定してください。

HASH_PINNABLE_CACHE_PERCENT オプションは、主に 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。この値を変更する場合は、細心の注意が必要です。さまざまなクエリへの影響を分析してから実行してください。

参照：
- BIT_VECTOR_PINNABLE_CACHE_PERCENT オプション (412 ページ)
- SORT_PINNABLE_CACHE_PERCENT オプション (524 ページ)

HASH_THRASHING_PERCENT オプション

ハッシュ・アルゴリズムを使ったクエリを含む文の実行中、文がロールバックされ、エラー・メッセージが返されるまでに許容するハード・ディスク I/O の割合を指定します。

指定できる値
0 – 100

デフォルト値
10

スコープ
個々の接続または PUBLIC グループに設定できます。すぐに有効になります。

説明
ハッシュ・アルゴリズムを使ったクエリが大量のハード・ディスク I/O (メモリからディスクへのページング・バッファ) を引き起こすと、クエリのパフォーマンスが低下し、サーバのパフォーマンスにも悪影響が及びます。
HASH_THRASHING_PERCENT は、文がロールバックされ、エラー・メッセージが返される前に許可するハード・ディスク I/O の割合を制御します。エラー・メッ

セージのテキストは、Hash insert thrashing detected または Hash find thrashing detected のいずれかです。

HASH_THRASHING_PERCENT のデフォルト値は 10% です。この値を大きくするとロールバックするまでのディスクへのページング量が増え、この値を小さくするとページング量が減ります。

過剰なページングの制御と HASH_THRASHING_PERCENT の使用方法の詳細については、『システム管理ガイド：第 1 巻』の「トラブルシューティングのヒント」>「状況別の解決策」>「処理の問題」>「ロードまたはクエリに予想以上の時間がかかる」を参照してください。

参照：

- HASH_PINNABLE_CACHE_PERCENT オプション (453 ページ)

HG_DELETE_METHOD オプション

HG インデックスの削除処理で使用されるアルゴリズムを指定します。

指定できる値
0 – 3

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
このオプションは、削除操作中に **HG** インデックスが使用するアルゴリズムを指定するために使用します。コスト・モデルは、適切な削除アルゴリズムを選択する際に、CPU 関連のコストと I/O 関連のコストを考慮します。コスト・モデルでは以下の要素が考慮されます。

- 削除されたロー
- インデックス・サイズ
- インデックス・データ型の幅
- インデックス・データのカーディナリティ
- 利用可能なテンポラリ・キャッシュ
- マシンに関連する I/O と CPU の特性
- 利用可能な CPU とスレッド

- 参照整合性のコスト

"small" メソッドを強制するには、このオプションを 1 に設定します。"large" メソッドを強制するには、このオプションを 2 に設定します。"midsize" メソッドを強制するには、このオプションを 3 に設定します。

HG_SEARCH_RANGE オプション

HG インデックス内の範囲述部の評価に使用する B ツリー・ページの最大数を指定します。

指定できる値
整数

デフォルト値
10

スコープ
個々の接続または PUBLIC グループに設定できます。すぐに有効になります。

説明
このオプションは、デフォルトの設定が大部分のクエリに適用できます。

このオプションは、オプティマイザが範囲述部に使用される最良のインデックスの検索にかける時間を効率よく制御します。高い値に設定すると、クエリがオプティマイザで費やす時間が増加する可能性がありますが、結果としては範囲述部を解決する最適なインデックスが選択されます。

HTTP_SESSION_TIMEOUT オプション

クライアントが HTTP セッションがタイムアウトになったと判断するまで待機する時間 (分単位) を指定します。

指定できる値
整数値 (0 ～ 525600)

デフォルト値
30

スコープ
DBA 権限が必要です。PUBLIC グループのみに設定できます。

説明
このオプションは、Web サービス・アプリケーションに対してさまざまなセッション・タイムアウト制御を提供します。Web サービス・アプリケーションは、

HTTP セッションを所有する任意の要求内でタイムアウト値を変更できます。ただし、タイムアウト値を変更すると、HTTP セッションがタイムアウトになった場合に、それ以降にキューイングされた要求に影響する場合があります。Web アプリケーションには、存在しなくなった HTTP セッションへのアクセスをクライアントが試行しているかどうかを検出するロジックを含める必要があります。そのためには、**SessionCreateTime** 接続プロパティの値を調べて、タイムスタンプが有効であるかどうかを確認します。対象の HTTP 要求が現在の HTTP セッションに関連付けられていない場合は、**SessionCreateTime** 接続プロパティには空の文字列が設定されています。

参照
『SQL Anywhere サーバー - プログラミング』の「HTTP Web サービス」>「HTTP Web サーバとしての SQL Anywhere の使用」を参照してください。

『SQL Anywhere サーバー - データベース管理』>「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」の **SessionCreateTime** プロパティおよび **http_session_timeout** プロパティを参照してください。

注意： これらのリファレンスは SQL Anywhere マニュアルにリンクされています。

IDENTITY_ENFORCE_UNIQUENESS オプション

すでにプライマリ・キーになっている場合を除き、各 Identity/Autoincrement カラムにユニーク **HG** インデックスを作成します。

指定できる値
ON、OFF

デフォルト値
OFF

スコープ
個々の接続で一時的に設定できます。個々のユーザまたは `PUBLIC` グループに設定できます。すぐに有効になります。

説明
このオプションを ON に設定すると、将来的な identity カラム用に **HG** インデックスが作成されます。このインデックスは、テーブルを使用しているユーザが 1 人しかおらず、テーブルがローカル・テンポラリ・テーブルでない場合に、そのユーザによってのみ削除できます。

参照：
- QUERY_PLAN オプション (511 ページ)

IDENTITY_INSERT オプション

ユーザが IDENTITY または AUTOINCREMENT カラムに値を挿入したり、これらのカラムを更新したりできるようにします。

指定できる値
= '*tablename*'

デフォルト値
'' (空の文字列)

スコープ
個々の接続で一時的に設定できます。個々のユーザまたは PUBLIC グループに設定できます。すぐに有効になります。

注意： ユーザ・レベルのオプションを現在のオプションに対して設定すると、対応する temporary オプションも同じように設定されます。「データベース・オプションのスコープと継続期間」を参照してください。

説明
IDENTITY_INSERT が設定されている場合、挿入と更新が有効になります。挿入または更新を行うカラムを識別するために、テーブルを指定する必要があります。テーブルの所有者でないユーザは、テーブル名を所有者の名前で修飾します。

IDENTITY カラムを持つテーブルを削除するには、IDENTITY_INSERT がそのテーブルに設定されていないことが必要です。

例
たとえば、明示的な挿入をテーブル Employees で行うには、次のように指定します。

```
SET TEMPORARY OPTION IDENTITY_INSERT = 'Employees'
```

オプションをオフにするには、等号と空の文字列を指定します。

```
SET TEMPORARY OPTION IDENTITY_INSERT = ''
```

ユーザ・レベル・オプションが temporary オプションに与える影響 (注意事項を参照してください) を説明するために、データベースに DBA として接続している場合に、次のコマンドを発行したと想定します。

```
SET OPTION IDENTITY_INSERT = 'Customers'
```

オプションの値は、ユーザ DBA に対して恒久的に、現在の接続に対して一時的に、Customers に設定されます。その後で他のユーザが DBA としてデータベー

スに接続した場合も、IDENTITY_INSERT のオプションの値は Customers になります。

参照：
- データベース・オプションのスコープと継続期間 (385 ページ)
- QUERY_PLAN オプション (511 ページ)

INDEX_ADVISOR オプション

1 つまたは複数のクエリのパフォーマンスを向上させる、追加のカラム・インデックスを推奨するメッセージを生成します。

指定できる値
ON、OFF

デフォルト値
OFF

スコープ
個々のユーザまたは PUBLIC グループに一時的(1 回の接続)に設定できます。すぐに有効になります。

説明
ON に設定されていると、インデックス・アドバイザは、Sybase IQ クエリ・プランの一部として、またはクエリ・プランが無効の場合に Sybase IQ メッセージ・ログ・ファイル内の独立したメッセージとして、推奨されるインデックスを表示します。これらのメッセージは、"Index Advisor:" という文字列で始まります。この文字列を検索することで、Sybase IQ メッセージ・ファイルからこれらのメッセージをフィルタできます。 出力は OWNER.TABLE.COLUMN の形式です。

インデックス・アドバイスを累積するには、**INDEX_ADVISOR** と **INDEX_ADVISOR_MAX_ROWS** の両方を設定します。

注意： **INDEX_ADVISOR_MAX_ROWS** が ON に設定されている場合、インデックス・アドバイスは独立したメッセージとして Sybase IQ メッセージ・ファイルに書き込まれません。 ただし、アドバイスは Sybase IQ メッセージ・ファイルのクエリ・プランに引き続き示されます。

表 25 : インデックス・アドバイザ

状況	推奨
HG、**LF**、**HNG**、DATE、TIME、または DATETIME のインデックスが望ましい、単一カラム上のローカル述部	<index-type> インデックスを <col> カラムに追加することを推奨
LF インデックスまたは **HG** インデックスが有用であると思われる単一カラムのジョイン・キー	**LF** インデックスまたは **HG** インデックスをジョイン・キーに追加
HG が存在するが、ユニークな **HG** または **LF** に変更可能な単一カラムの候補キー・インデックス	ジョイン・キー <col> をユニークな **LF** インデックスまたは **HG** インデックスに変更
ジョイン・キーに適切でないデータ型が割り当てられており、適切なデータ型で 1 つのカラムを再生成すると改善されると思われる場合	ジョイン・キー <col1> と <col2> を同じデータ型にする
LF インデックスまたは **HG** インデックスが有用であると思われるサブクエリの述部のカラム	**LF** インデックスまたは **HG** インデックスをサブクエリ・カラム <col> に追加
LF インデックスまたは **HG** インデックスが有用であると思われるグループ化カラム	**LF** インデックスまたは **HG** インデックスをグループ化カラム <col> に作成
単一のテーブルでのカラム間の比較で、2 つのカラムが同じデータ型で、**CMP** インデックスが推奨される場合	**CMP** インデックスを <col1>、<col2> で作成
LF インデックスまたは **HG** インデックスが存在し、異なる値の数が許容するカラムが、**FP** を 1 または 2 バイトの **FP** インデックスに変換することを推奨している場合	<col> を、'optimize storage=on'で再構築する
3 バイト幅のデフォルト・インデックスのルックアップのサポート	FP インデックスを IQ UNIQUE 制約値 65537 で 3 バイト FP インデックスとして再構築する

インデックスを追加することでどれだけのクエリが改善されるか、また、インデックスを作成および維持するメリットがあるかどうかはユーザが判断する必要があります。場合によっては、推奨されたインデックスを追加することで、パフォーマンスが改善するかどうか、改善するとしたらどの程度かを判断できないこともあります。

たとえば、ジョイン・キーとして使われているカラムを例に考えてみます。Sybase IQ は、クエリを実行するためのより良くより速いクエリ・プランを生成するために、**HG** または **LF** インデックスによって提供されるメタデータを多く使用します。 **HG** インデックスまたは **LF** インデックスを持たないジョイン・カラムに

これらのインデックスを追加すると、IQ オプティマイザがより速いジョイン・プランを選択できる可能性が高くなりますが、インデックスを追加してクエリを再実行しないかぎり、新しいインデックスによってクエリのパフォーマンスが変わらないか向上するかを判断するのは非常に困難です。

例
クエリ・プランが OFF に設定されたインデックス・アドバイザの出力

```
I. 03/30 14:18:45. 0000000002 Advice: Add HG or LF index
on DBA.ta.c1 Predicate: (ta2.c1 < BV(1))
```

クエリ・プランが ON に設定されたインデックス・アドバイザの出力

注意： この方法では複数のクエリのインデックス・アドバイザの情報が累計されるため、複数のクエリのアドバイスを中央のロケーションで一定期間にわたって追跡できます。

```
I. 03/30 14:53:24. 0000000008 [20535]: 6    ...#03: Leaf
I. 03/30 14:53:24. 0000000008 [20535]:         Table Name: tb
I. 03/30 14:53:24. 0000000008 [20535]:         Condition 1
(Invariant):
(tb.c3 =tb.c4)
I. 03/30 14:53:24. 0000000008 [20535]:         Condition 1 Index
Advisor:
Add a CMP index on DBA.tb (c3,c4)
```

参照：

- FP_LOOKUP_SIZE オプション (448 ページ)
- INDEX_ADVISOR_MAX_ROWS オプション (461 ページ)
- MINIMIZE_STORAGE オプション (490 ページ)
- QUERY_PLAN オプション (511 ページ)

INDEX_ADVISOR_MAX_ROWS オプション

インデックス・アドバイザによって max_rows に格納されるユニークなアドバイス・メッセージの最大数を設定します。

指定できる値

値	説明
0	インデックス・アドバイスの収集を無効にする最小値
4294967295	許可される最大値

デフォルト値
0

スコープ
(現在の接続に)一時的に設定できます。また、ユーザ／グループ (PUBLIC または DBA など) に対して永続的に設定できます。すぐに有効になります。

説明
INDEX_ADVISOR_MAX_ROWS は、インデックス・アドバイザによって格納されるメッセージの数を制限します。指定した制限値に到達すると、**INDEX_ADVISOR** は新しいアドバイスの保存を停止します。ただし、既存のアドバイス・メッセージのカウントとタイムスタンプの更新は続行します。

```
SET OPTION public.Index_Advisor_Max_Rows = max_rows;
```

参照：
- FP_LOOKUP_SIZE オプション (448 ページ)
- INDEX_ADVISOR オプション (459 ページ)

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 26 : 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。

値	アクション
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。
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。

INFER_SUBQUERY_PREDICATES オプション

オプティマイザによる追加のサブクエリの述部の推測を制御します。

指定できる値
ON、OFF

デフォルト値
ON

スコープ
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。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。

説明
INFER_SUBQUERY_PREDICATES は、単純な等号ジョイン述部の推移閉包によって、既存のサブクエリの述語から追加のサブクエリの述部をオプティマイザに推測させるかどうかを制御します。多くの場合、オプティマイザがこの推測を行ったほうがクエリの実行が速くなります。これには例外もあり、パフォーマンスが向上しない場合もあります。このオプションを使うことが環境において適切かどうかを確認する必要があります。

IN_SUBQUERY_PREFERENCE オプション

IN サブクエリを処理する際に使用するアルゴリズムを選択します。

指定できる値
-3 ～ 3

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
IN サブクエリを処理する際、IQ オプティマイザはいくつかのアルゴリズムから 1 つを選択します。このオプションを使用すると、オプティマイザが処理量をもとに決定した使用アルゴリズムを無効できます。クエリ・エンジンに対してアルゴリズムが適切かどうかを判断するための内部規則を無効にするものではありません。

IN_SUBQUERY_PREFERENCE は通常、内部のテストと、オプティマイザがうまく処理できないクエリの手動チューニングに利用されます。経験豊富な DBA のみ使用してください。このオプションを使用するのは、サブクエリによって生成されるローの数をオプティマイザが少なく見積もりすぎているために、ハッシュ・オブジェクトがスラッシングしている場合のみです。このオプションを設定する前に、まず、失われたインデックスや依存述部をチェックして、誤った見積もりを修正してください。

IN_SUBQUERY_PREFERENCE を設定するのは、オプティマイザに変更を加えることが適切である場合のみであるため、このオプションを設定する必要がある場合は、Sybase 製品の保守契約を結んでいるサポート・センタにご連絡ください。

表 27 : IN_SUBQUERY_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	ソートベースの **IN** サブクエリを優先する。
2	垂直 **IN** サブクエリ (サブクエリがクエリ・プラン内のリーフ・ノードの子) を優先する。
3	ハッシュベースの **IN** サブクエリを優先する。
-1	ソートベースの **IN** サブクエリを回避する。
-2	垂直 **IN** サブクエリを回避する。
-3	ハッシュベースの **IN** サブクエリを回避する。

IQGOVERN_MAX_PRIORITY オプション

許可される IQGOVERN_PRIORITY 設定を制限します。

指定できる値
1 – 3

デフォルト値
2

スコープ
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。設定には DBA パーミッションが必要です。すぐに有効になります。

説明
許可される IQGOVERN_PRIORITY の設定を制限します。この設定は、ユーザのクエリが実行待ちのキューに入れられる順番に影響します。 指定可能な値は、1 (優先度高)、2 (優先度中、デフォルト値)、3 (優先度低) です。ユーザが IQGOVERN_MAX_PRIORITY よりも大きい値を IQGOVERN_PRIORITY に指定すると、Sybase IQ はエラーを返します。

参照：
- IQGOVERN_PRIORITY オプション (466 ページ)

- IQGOVERN_PRIORITY_TIME オプション (466 ページ)

IQGOVERN_PRIORITY オプション

-iqgovern キュー内で待機中の各クエリに優先度を割り当てます。

指定できる値
1 – 3

デフォルト値
2

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
ユーザのクエリが実行待ちのキューに入れられる順番を決定する値を割り当てます。指定可能な値は、1 (優先度高)、2 (優先度中、デフォルト値)、3 (優先度低) のいずれかです。このスイッチは、すべてのユーザが、一時的、ユーザごと、または public に指定できます。優先度の低いクエリは、より優先度の高いクエリがすべて実行されるまで実行されません。

このオプションは、オプション IQGOVERN_MAX_PRIORITY のユーザごとまたはグループごとの値によって制限されます。

参照：
- IQGOVERN_MAX_PRIORITY オプション (465 ページ)
- IQGOVERN_PRIORITY_TIME オプション (466 ページ)

IQGOVERN_PRIORITY_TIME オプション

高優先度のクエリが実行までにキューで待機する時間を制限します。

指定できる値
0 ～ 1,000,000 秒。IQGOVERN_MAX_PRIORITY よりも小さい値である必要があります。

デフォルト値
0 (無効)

スコープ
PUBLIC グループのみに設定できます。DBA パーミッションが必要です。すぐに有効になります。

説明
高優先度 (優先度 1) のクエリが実行までにキューで待機する時間を制限します。この時間が過ぎると、**-iqgovern** の設定で実行が許可されたクエリ数を超過しても、クエリが実行されます。このスイッチを変更するには、グループ DBA のメンバである必要があります。指定できる値は、1 から 1,000,000 秒です。0 (デフォルト値) では、この機能が無効になります。IQGOVERN_PRIORITY_TIME は PUBLIC に設定されることが必要です。

参照：
- IQGOVERN_MAX_PRIORITY オプション (465 ページ)
- IQGOVERN_PRIORITY オプション (466 ページ)

ISOLATION_LEVEL オプション

カタログ・ストア・テーブルのロック独立性レベルを制御します。

指定できる値
0、1、2、または 3

デフォルト値
0

説明
ロック独立性レベルは次のように定義されています。

- 0 – ダーティ・リード、繰り返し不可能読み出し、幻ローが許可されます。
- 1 – ダーティ・リードは許可されません。繰り返し不可能読み出しと幻ローが許可されます。
- 2 – ダーティ・リードは許可されませんが、繰り返し可能読み出しが保証されます。幻ローは許可されます。
- 3 – 逐次化可能です。ダーティ・リードは許可されず、繰り返し可能読み出しは保証され、幻ローは許可されません。

ISOLATION_LEVEL オプションは、カタログ・ストアのテーブルに対する独立性レベルの設定に使用します。Sybase IQ では、IQ ストア内のテーブルに常にレベル 3 を適用します。 レベル 3 が ANSI のレベル 4 に相当します。

JAVA_LOCATION オプション

データベースの Java VM のパスを指定します。

指定できる値
文字列

デフォルト値
空の文字列

スコープ
PUBLIC グループのみに設定できます。DBA 権限が必要です。

説明
デフォルトでは、このオプションには空の文字列が含まれています。この場合、データベース・サーバは、JAVA_HOME 環境変数、パス、およびその他のロケーションで Java VM を検索します。

参照：
- JAVA_VM_OPTIONS オプション (468 ページ)

JAVA_VM_OPTIONS オプション

データベース・サーバが Java VM を起動するときに使用するコマンド・ライン・オプションを指定します。

指定できる値
文字列

デフォルト値
空の文字列

スコープ
PUBLIC グループのみに設定できます。DBA 権限が必要です。

説明
JAVA_VM_OPTIONS では、JAVA_LOCATION オプションで指定された Java VM を起動するときにデータベース・サーバが使用するオプションを指定できます。これらの追加のオプションを使用して、デバッグ用に、または UNIX プラットフォーム上でサービスとして実行するように Java VM を設定できます。32 ビット・モードではなく 64 ビット・モードで Java VM を使用するには、追加のオプションが必要な場合があります。

『SQL Anywhere サーバー - データベース管理』の「データベースの設定」>「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」>「java vm options オプション」を参照してください。

注意： このリファレンスは SQL Anywhere マニュアルにリンクされています。

参照：

- JAVA_LOCATION オプション (468 ページ)

JOIN_EXPANSION_FACTOR オプション

極端に複雑な状況において、オプティマイザによるジョイン結果の見積もりをどの程度抑えるかを制御します。

指定できる値
1 – 100

デフォルト値
30

スコープ
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
このオプションは、特定のジョインへの入力が最低でも 1 つの中間ジョインを経由しており、ジョインされるテーブルから同じローが複数コピーされる可能性がある場合に、ジョイン・オプティマイザによる見積もりサイズがどうなるかを制御します。

ゼロを指定した場合、オプティマイザは、中間拡張ジョインがある場合も、通常と同じ見積もり方法を使います。

この場合、予測されるジョインの結果サイズは最も楽観的に (小さく) なります。

100 を指定した場合、中間拡張ジョインがある場合のオプティマイザによる見積もりはかなり控えめになり、予測されるジョインの結果サイズはもっとも慎重に (大きく) なります。

通常、この値を変更する必要はありません。変更する場合は、JOIN_EXPANSION_FACTOR を temporary オプション、または user オプションとして設定することをおすすめします。

JOIN_OPTIMIZATION オプション

ジョイン順序の最適化を有効または無効にします。

指定できる値
ON、OFF

デフォルト値
ON

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
JOIN_OPTIMIZATION オプションが ON の場合、Sybase IQ はジョイン順序を最適化して中間結果とソートのサイズを小さくし、システム負荷のバランスをとります。このオプションが OFF の場合、ジョイン順序は **SELECT** 文の **FROM** 句のテーブルの順序に従って決定されます。

JOIN_OPTIMIZATION は常に ON に設定してください。

JOIN_OPTIMIZATION オプションは、ジョインの順序を制御しますが、テーブルの順序は制御しません。この差について、4 つのテーブルがある次の **FROM** 句を例に説明します。

```
FROM  A, B, C, D
```

この **FROM** 句は、デフォルトで左側が深いジョイン・プランを作成します。これは、明示的に表すと次のようになります。

```
FROM  (((A, B), C), D)
```

JOIN_OPTIMIZATION が OFF の場合、一連のテーブルにおけるこれらのジョインの順序は **FROM** 句で指定されたとおりになります。つまり、A と B がまずジョインされ、その成果物がテーブル C にジョインされ、最後にこれがテーブル D にジョインされます。このオプションは、各ジョインの左右の深さは制御しません。JOIN_OPTIMIZATION が OFF で上記の **FROM** 句を実行した場合、オプティマイザは次のようなジョイン・プランを作成します。

```
FROM  ((C, (A, B)), D)
```

または

```
FROM  (((B, A), C), D)
```

または

```
FROM  (D, ((A, B), C))
```

いずれの場合でも、まず A と B がジョインされ、その成果物が C にジョインされ、最後にテーブル D にジョインされます。ジョインの順序は同じですが、テーブルが表示される順序が異なります。

JOIN_OPTIMIZATION が OFF の場合は、意図したとおりの順序でジョインが行われるよう、上の例のように **FROM** 句内でカッコを使用する必要があります。C と D をジョインしたものに A と B をジョインするには、カッコを使って次のように指定します。

```
FROM ((A, B), (C, D))
```

上記の **FROM** 句では、最初の例の **FROM** 句と同じ順序でテーブルが表示されていますが、異なる順序でジョインが行われます。

JOIN_OPTIMIZATION は、ジョインのパフォーマンスに関する隠れた問題を診断する場合、または少数の定義済みクエリを手動で最適化する場合のみ OFF に設定します。JOIN_OPTIMIZATION が OFF の場合、クエリは最大で 128 のテーブルをジョインできますが、パフォーマンスに深刻な悪影響が及ぶ可能性があります。

警告！ JOIN_OPTIMIZATION を無効にすると、Sybase IQ はジョインを含むクエリに最良のパフォーマンスを保証できません。クエリのパフォーマンスに関してユーザ自身が全面的に責任を負うことになります。

JOIN_PREFERENCE オプション

ジョイン処理で使用されるアルゴリズムを選択します。

指定できる値
-7 ～ 7

デフォルト値
0

スコープ
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
IQ オプティマイザは、クエリ内のジョインを処理する場合に複数のアルゴリズムの選択肢から 1 つを選択します。JOIN_PREFERENCE を使用すると、オプティマイザが処理量をもとに決定した使用アルゴリズムを無効にできます。クエリ・エンジンに対してアルゴリズムが適切かどうかを判断するための内部規則を無効に

するものではありません。0 以外の値を設定すると、クエリ内のすべてのジョインに影響します。クエリ内のいくつかのジョインから 1 つを選択して修正することはできません。

通常、このオプションは、内部的なテストに使用されます。また、経験豊富な DBA だけが使用してください。

表 28 : JOIN_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	ソート／マージを優先する。
2	ネストされたループを優先する。
3	ネストされたループのプッシュダウンを優先する。
4	ハッシュを優先する。
5	ハッシュ・プッシュダウンを優先する。
6	プリジョインを優先する。
7	ソート／マージのプッシュダウンを優先する。
-1	ソート／マージを回避する。
-2	ネストされたループを回避する。
-3	ネストされたループのプッシュダウンを回避する。
-4	ハッシュを回避する。
-5	ハッシュ・プッシュダウンを回避する。
-6	プリジョインを回避する。
-7	ソート／マージのプッシュダウンを回避する。

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')
```

JOIN_SIMPLIFICATION_THRESHOLD オプション

ジョイン・オプティマイザの単純化が行われる前にジョインされるテーブルの最小数を指定します。

指定できる値
1 – 64

デフォルト値
15

スコープ
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
クエリ・オプティマイザは、ルックアップ・テーブル (非選択的な次元テーブル) と有効な直積であるテーブルを別々に扱うことで、ジョインの順序の最適化を単純化します。単純化の後で、オプティマイザは MAX_JOIN_ENUMERATION によって指定されている範囲内で、残りのテーブルのジョインの順序を最適化します。

このオプションに MAX_JOIN_ENUMERATION の値よりも大きい値を指定しても何も起こりません。

このオプションに MAX_JOIN_ENUMERATION の値よりも小さい値を指定すると、大量のジョインを含むクエリの最適化にかかる時間を短縮できる可能性がありますが、オプティマイザが最適なジョイン・プランを見つけられなくなる可能性もあります。

通常、この値を変更する必要はありません。変更する場合は、JOIN_SIMPLIFICATION_THRESHOLD を temporary レベルまたは ユーザ・レベルのオプションとして設定し、9 以上の値にすることをおすすめします。

参照：

- MAX_JOIN_ENUMERATION オプション (486 ページ)

LARGE_DOUBLES_ACCUMULATOR オプション

浮動小数点の値の **SUM** または **AVG** に使用するアキュムレータの選択を制御します。

指定できる値
ON、OFF

デフォルト値
OFF

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
float 型と double 型データ用の小さなアキュムレータで、1e-20 から 1e20 までのたいへん広い範囲の加算がとても正確に実行できます。上記の範囲外の数値に対しては若干精度が犠牲になります (それでも通常のアプリケーションには十分な精度が出ます)。この小さなアキュムレータを使用すると、大きなアキュムレータを使用した場合に比べて、オプティマイザは簡単に高いパフォーマンスを実現するハッシュを選択できます。大きなアキュムレータはあらゆる float 型と double 型のデータに対する演算が非常に正確ですが、その規模のためハッシュを使用した最適化が選択されないことが多くなります。デフォルトは小さなアキュムレータです。

LF_BITMAP_CACHE_KB オプション

LF インデックスに対するロードで使用されるメモリ量を指定します。

指定できる値
1 – 8

デフォルト値
4

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
LF_BITMAP_CACHE_KB は、**LF** インデックスに対するロードで重複しない各値に対して使用されるヒープ・メモリの大きさ (KB 単位) の指定に使用します。デフォルトは 4KB です。すべての **LF** インデックスの重複しない値の個数の合計が比較的大きい (10,000 以上) テーブルの場合は、ヒープ・メモリの使用が増えて、システムのページ・フォールトが発生し、ロードのパフォーマンスに影響を与える可能性があります。このような事態が発生したら、LF_BITMAP_CACHE_KB の設定値を小さくしてください。

ロード中に各 **LF** インデックスが使用するヒープ・メモリの大きさは、次の式に従って計算します。

```
Heap-memory-used = (lf_bitmap_cache_kb * 1024)
* lf-distinct-count-for-column
```

デフォルト値である 4KB を使用すると、重複しない 1000 の値がある **LF** インデックスはロード中に最大 4MB のヒープ・メモリを使用します。

LOAD_ZEROLENGTH_ASNULL オプション

特定の条件での LOAD 文の動作を指定します。

指定できる値
ON、OFF

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。

デフォルト値
OFF

説明
このオプションは、次の条件での LOAD 文の動作を指定します。

- データ型が CHAR、VARCHAR、LONG VARCHAR、BINARY、VARBINARY、または LONG BINARY であるカラムに、長さゼロのデータ値を挿入している。
  および
- 同じカラムに対して、NULL カラム指定 (NULL(ZEROS) や NULL(BLANKS) など) が行われている。

LOAD_ZEROLENGTH_ASNULL を ON に設定すると、これらの条件が満たされている場合に、長さゼロの値を NULL としてロードします。

LOAD_ZEROLENGTH_ASNULL を OFF に設定すると、オプション NON_ANSI_NULL_VARCHAR の設定によって、長さゼロの値を長さゼロとしてロードします。

参照：
- NON_ANSI_NULL_VARCHAR オプション (496 ページ)
- LOAD TABLE 文 (273 ページ)

LOCKED オプション

ログイン・ポリシーに設定されている場合、そのポリシーを持つユーザによる新しい接続の確立を防止します。

詳細については、『システム管理ガイド：第 1 巻』の「ユーザ ID とパーミッションの管理」を参照してください。

参照：

- CREATE LOGIN POLICY 文 (132 ページ)
- ALTER LOGIN POLICY 文 (22 ページ)

LOG_CONNECT オプション

ユーザ接続のロギングを制御します。

指定できる値
ON、OFF

デフォルト値
ON

スコープ
PUBLIC グループのみに設定できます。すぐに有効になります。

説明
このオプションが ON の場合、ユーザが Sybase IQ データベースに接続、または切断すると IQ メッセージ・ログ (.iqmsg ファイル) にメッセージが表示されます。

注意： ユーザの接続時にこのオプションが OFF (接続のロギングが無効) に設定されており、そのユーザが切断する前に ON にされた場合、メッセージ・ログにはユーザの接続は記録されず、切断だけが記録されます。

LOG_CURSOR_OPERATIONS オプション

カーソル操作のログ動作を制御します。

指定できる値
ON、OFF

デフォルト値
OFF

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
このオプションが ON の場合、カーソルを開いたり閉じたりするたびにメッセージが IQ メッセージ・ログに表示されます。デフォルトは OFF で、通常はこのままにしてください。このオプションを ON にするのは、何か問題があって、Sybase 製品の保守契約を結んでいるテクニカル・サポートに対してデバッグ・データを送る必要があるときだけにしてください。

LOGIN_MODE オプション

データベースの統合化ログインの使用を制御します。

指定できる値
Standard、Mixed、または Integrated

デフォルト値
Standard

スコープ
PUBLIC グループのみに設定できます。すぐに有効になります。

説明
このオプションは、統合化ログインが許可されるかどうかを指定します。値は次のとおりで、大文字と小文字を区別しません。

- Standard – 統合化ログインは使用できません。これがデフォルト設定です。統合化ログインを使用して接続しようとすると、エラーが発生します。
- Mixed – 統合化ログインと標準ログインの両方を使用できます。
- Integrated – この設定では、データベースへのすべてのログインを統合化ログインで行う必要があります。

警告！ LOGIN_MODE を Integrated に設定すると、統合化ログイン・マッピングを付与されているユーザだけに接続が制限されます。ユーザ ID とパスワードを使用して接続しようとすると、エラーが生成されます。唯一の例外は、DBA 権限 (完全な管理権) を持つユーザです。

LOGIN_PROCEDURE オプション

起動時の接続互換性オプションを設定するログイン・プロシージャを指定します。

指定できる値
文字列

デフォルト値
sp_login_environment システム・プロシージャ

スコープ
個々の接続または PUBLIC グループに設定できます。このオプションを設定するには、DBA パーミッションが必要です。すぐに有効になります。

説明
初期設定は、LOGIN_PROCEDURE オプションを使用して制御され、接続が有効であることを確認するすべてのチェック実行後に呼び出されます。
LOGIN_PROCEDURE オプションは、ユーザが接続したときに実行されるストアド・プロシージャを指定します。デフォルト設定では、**sp_login_environment** システム・ストアド・プロシージャが使用されます。異なるストアド・プロシージャを指定できます。LOGIN_PROCEDURE オプションに指定したプロシージャはイベント接続では実行されません。

sp_login_environment は、接続が TDS によって行われたかどうかを確認します。TDS によって接続が行われている場合は、**sp_login_environment** が **sp_tsql_environment** プロシージャを呼び出して、いくつかのオプションに現在の接続に合った新しいデフォルト値を設定します。

LOGIN_PROCEDURE オプションの詳細と例については、『SQL Anywhere サーバー – データベース管理』の「データベースの設定」>「データベースオプション」>「データベースオプションの概要」>「アルファベット順のオプションリスト」>「login_procedure オプション」を参照してください。

注意： このリファレンスは SQL Anywhere マニュアルにリンクされています。

参照：
- オプションの初期設定 (388 ページ)

MAIN_RESERVED_DBSPACE_MB オプション

Sybase IQ が IQ メイン・ストアに予約する領域の量を制御します。

指定できる値
200 またはそれ以上の整数 (メガバイト)

デフォルト値
200。Sybase IQ は、IQ_SYSTEM_MAIN の最新読み書き可能ファイルの最大で 50%、最小で 1% を予約します。

スコープ
PUBLIC グループのみに設定できます。このオプションを設定するには、DBA パーミッションが必要です。すぐに有効になります。予約領域サイズを変更するためにサーバを再起動する必要はありません。

説明
MAIN_RESERVED_DBSPACE_MB を使用すると、セーブポイントの解放操作、コミット操作、チェックポイント操作で使用される小さいが重要なデータ構造体用に Sybase IQ が IQ メイン・ストア内に確保する領域の量を制御できます。 運用データベースでは、この値を 200MB から 1GB の間で設定します。 IQ ページ・サイズや同時接続数が大きくなればなるほど、より多くの領域を予約する必要があります。

予約領域サイズは、IQ_SYSTEM_MAIN の最新読み書き可能ファイルの 1 ～ 50% で計算されます。

MAX_CARTESIAN_RESULT オプション

直積ジョインからのローの数を制限します。

指定できる値
整数値

個々の接続で一時的に設定できます。個々のユーザまたは PUBLIC グループに設定できます。すぐに有効になります。

デフォルト値
100000000

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
MAX_CARTESIAN_RESULT は、直積ジョインを含むクエリからの結果ロー (クエリを生成したときに 1 つまたはそれ以上のジョイン条件が見つからなかった結果) の数を制限するのに使用します。 Sybase IQ が、この制限以下の予想結果を持つ直積ジョインに対するクエリ・プランを見つけることができなかった場合、クエリ

は拒否され、エラーが返されます。 MAX_CARTESIAN_RESULT を 0 に設定すると、直積ジョインの結果ロー数のチェックが無効になります。

MAX_CLIENT_NUMERIC_PRECISION オプション

クライアントに送信される numeric データの最大精度を制御します。

指定できる値
0 – 126

デフォルト値
0

スコープ
すべてのユーザがどのレベルにでも設定できます。このオプションはすぐに有効になります。

説明
Sybase IQ は、計算を行うとき、精度を保証するためにデータ型を適切なサイズに拡大します。 拡大されたデータ型は Open Client のサイズよりも大きい可能性があり、正しく扱えるのは一部の ODBC アプリケーションです。

MAX_CLIENT_NUMERIC_PRECISION がゼロ以外の値の場合、Sybase IQ は numeric の結果カラムがこの値を超過していないかどうかチェックします。 結果カラムが MAX_CLIENT_NUMERIC_PRECISION の制限をオーバーしており、Sybase IQ が指定された精度にキャストすることもできない場合、クエリは次のようなエラーを返します。

```
Data Exception - data type conversion is not possible %1 SQLCODE =
-1001006
```

注意： SQL Anywhere では、数値関数でサポートされる最大値は 255 です。数値関数の精度がサポートされている最大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'`

参照：

- MAX_CLIENT_NUMERIC_SCALE オプション (481 ページ)
- PRECISION オプション (505 ページ)

MAX_CLIENT_NUMERIC_SCALE オプション

クライアントに送信される numeric データの最大位取りを制御します。

指定できる値
0 – 126

デフォルト値
0

スコープ
すべてのユーザがどのレベルにでも設定できます。このオプションはすぐに有効になります。

説明
Sybase IQ は、計算を行うとき、精度を保証するためにデータ型を適切な位取りとサイズに拡大します。 拡大されたデータ型は、最初に定義されたデータのサイズよりも大きくなる可能性があります。 このオプションを設定して、numeric の結果の位取りを指定します。

掛け算、割り算、足し算、引き算、集合関数はすべて、結果が最大精度および最大位取りを超える可能性があります。

たとえば、DECIMAL(88,2) と DECIMAL(59,2) を掛けると、結果には DECIMAL(147,4) が必要です。`MAX_CLIENT_NUMERIC_PRECISION` が 126 の場合、126 桁のみが結果に残ります。`MAX_CLIENT_NUMERIC_SCALE` が 4 の場合、結果は DECIMAL(126,4) として返されます。`MAX_CLIENT_NUMERIC_SCALE` が 2 の場合、結果は DECIMAL(126,2) として返されます。どちらの場合も、オーバフローが起こる可能性があります。

参照：
- MAX_CLIENT_NUMERIC_PRECISION オプション (480 ページ)
- SCALE オプション (522 ページ)

MAX_CONNECTIONS オプション

ユーザに許可される同時接続の最大数を指定します。

詳細については、『SQL Anywhere サーバー – データベース管理』を参照してください。

MAX_CUBE_RESULT オプション

IQ オプティマイザが **GROUP BY CUBE** 操作を行う場合の対象とするロー数の最大値を設定します。

指定できる値
0 – 4294967295

デフォルト値
10000000

スコープ
すべてのユーザがどのレベルにでも設定できます。このオプションはすぐに有効になります。

説明
クエリ・プランを生成するとき、IQ オプティマイザは、**GROUP BY CUBE** ハッシュ操作で生成されるグループの合計数を見積もります。IQ オプティマイザは、**GROUP BY CUBE** 操作にハッシュ・アルゴリズムを使用します。このオプションは、実行可能なハッシュ・アルゴリズムに対してオプティマイザが見積もるロー数に上限を設定します。実際のロー数が MAX_CUBE_RESULT オプションの値を超える場合、オプティマイザはクエリの処理を中止し、`Estimate number: nnn exceeds the default MAX_CUBE_RESULT of GROUP BY CUBE or ROLLUP` というエラー・メッセージを表示します。*nnn* は、IQ オプティマイザが見積もった数字です。

デフォルト値を上書きするには、MAX_CUBE_RESULT をゼロに設定します。このオプションがゼロに設定されている場合、IQ オプティマイザはローの上限をチェックせずにクエリの実行を許可します。MAX_CUBE_RESULT をゼロに設定するとクエリが失敗する可能性があるため、おすすめしません。

MAX_CURSOR_COUNT オプション

各接続が一度に使用できるカーソルの最大数を制限するリソース・ガバナを指定します。

指定できる値
整数

デフォルト値
50

スコープ
個々の接続または PUBLIC グループに設定できます。すぐに有効になります。このオプションを設定するには、対象となる接続がどのようなものであっても DBA パーミッションが必要です。

説明
DBA は指定されたリソース・ガバナを使用して、ユーザが 1 つの接続で使用できるカーソルの数を制限できます。1 つの接続に対してこの制限を超える操作を行うと、制限を超過していることを示すエラーを生成します。

ある接続でストアド・プロシージャが実行される場合、それはプロシージャの所有者のパーミッションに従って実行されます。ただし、プロシージャが使用するリソースは現在の接続から割り当てられます。

MAX_CURSOR_COUNT を 0 (ゼロ) に設定すると、リソース制限を削除できます。

MAX_DAYS_SINCE_LOGIN オプション

同じユーザによる 2 回連続のログイン間における最大日数を指定します。

詳細については、『SQL Anywhere サーバー - データベース管理』の「データベースの設定」>「ユーザー ID、権限、パーミッションの管理」>「ログインポリシーの管理」を参照してください。

注意： このリファレンスは SQL Anywhere マニュアルにリンクされています。

MAX_FAILED_LOGIN_ATTEMPTS オプション

前回の正常なログイン時以降に行われた、アカウントがロックされる前にユーザ・アカウントが失敗したログイン試行回数の最大値を指定します。

詳細については、『SQL Anywhere サーバー - データベース管理』の「データベースの設定」>「ユーザー ID、権限、パーミッションの管理」>「ログインポリシーの管理」を参照してください。

注意： このリファレンスは SQL Anywhere マニュアルにリンクされています。

マルチプレックス環境内の各ノードは、特定のユーザに関するログインの失敗回数と最後にログインが失敗した時刻を維持しています。

Sybase IQ では、MAX_FAILED_LOGIN_ATTEMPTS は、DBA 権限を持つユーザも含めたすべてのユーザに適用されます。DBA 権限を持つユーザ・アカウントの場合、Sybase IQ サーバは、最後にログインが失敗してから 15 分後に自動的にアカウントのロックを解除します。

次の場合、DBA 権限を持つユーザはログインできません。

- そのユーザのログイン・ポリシーで許可されている、ログイン試行の最大失敗回数を超えた場合
  および
- 現在時刻と、そのユーザのログインが最後に失敗した時刻の差が 15 分未満の場合

データベースを再起動すると、DBA 権限を持つユーザは、ログイン試行の失敗回数が、自分のログイン・ポリシーで許可されている最大回数を超えている場合でも、1 度のみログインを試行できます。

MAX_HASH_ROWS オプション

IQ オプティマイザがハッシュ・アルゴリズムを使用する際の対象とするロー数の最大値を設定します。

指定できる値
1 ～ 4294967295 の整数

デフォルト値
2500000

スコープ
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。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。このオプションはすぐに有効になります。

説明
クエリ・プランを生成する際、IQ オプティマイザには、クエリの特定部分の処理用アルゴリズムとして複数の選択肢 (ハッシュ、ソート、インデックス) があります。選択されるアルゴリズムは、処理されるロー数の予測値、または検討の対象とされているクエリの特定部分に含まれているロー数の予測値によって変わります。このオプションは、ハッシュ・アルゴリズムを検討する際の予想ロー数の最大値を設定するのに使用されます。

たとえば、2 つのテーブル間にジョインがあり、両方のテーブルからジョインに入力されるロー数が MAX_HASH_ROWS の値を超えると、オプティマイザはハッシュ・ジョインを選択肢から外します。1 ユーザに対し 50MB 以上のテンポラリ・バッファ・キャッシュ領域があるシステムでは、このオプションにより大きな値を設定することもできます。

MAX_IQ_THREADS_PER_CONNECTION オプション

各接続のスレッド数を制御します。

指定できる値
3 – 10000

デフォルト値
144

スコープ
一時的にも永続的にも設定できます。設定に DBA パーミッションは必要ありません。PUBLIC グループのみに設定できます。すぐに有効になります。

説明
接続上で実行されるコマンドが使用するスレッド数 (システム リソースの量) を制限します。大部分のアプリケーションには、デフォルト値を使用します。

MAX_IQ_THREADS_PER_TEAM オプション

接続内で実行される 1 つの操作 (カラム上の **LIKE** 述部など) を実行するために割り当てられるスレッドの数を制御します。

指定できる値
1 – 10000

デフォルト値
144

スコープ
一時的にも永続的にも設定できます。設定に DBA パーミッションは必要ありません。PUBLIC グループのみに設定できます。すぐに有効になります。

説明
単一の操作が割り当てられるスレッド数 (システム リソースの量) を制限します。この接続で同時に実行されるすべてのチームにおける合計数は、関連オプション MAX_IQ_THREADS_PER_CONNECTION によって制限されます。大部分のアプリケーションには、デフォルト値を使用します。

参照：

- MAX_IQ_THREADS_PER_CONNECTION オプション (485 ページ)

MAX_JOIN_ENUMERATION オプション

オプティマイザによる単純化の適用後、ジョイン順を最適化するテーブルの最大数を制御します。

指定できる値
1 – 64

各 **FROM** 句に指定できるテーブル数は、最大 64 に制限されます。実際には、**FROM** 句内のテーブル数の効果的な制限値はこの値よりもはるかに小さく、テーブル間のジョイン関係の複雑さに部分的に基づいて決定されます。テーブル数の効果的な制限値は、**MAX_JOIN_ENUMERATION** の設定によって制約を受けます。オプティマイザは、**FROM** 句内のジョイン関係のセットを単純化しようとします。これらの単純化を行っても、同時に考慮すべきジョインのセットの数を **MAX_JOIN_ENUMERATION** の現在の設定以下に削減できない場合は、クエリはエラーを返します。

警告！ MAX_JOIN_ENUMERATION の設定をデフォルトを超える 16 より大きくする場合は、特に、オプティマイザが必要とする時間の量が劇的に増加する可能性がある、枝分かれの多いジョイン関係を伴うクエリの場合、注意が必要です。リニア・チェーンのジョイン関係のみを使用するクエリであっても、**MAX_JOIN_ENUMERATION** を 64 に設定すると、最適化に大量の時間がかかることがあります。

デフォルト値
15

スコープ
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
クエリ・オプティマイザは、ルックアップ・テーブル (非選択的な次元テーブル) と有効な直積であるテーブルを別々に扱うことで、ジョインの順序の最適化を単純化します。単純化の後で、オプティマイザは MAX_JOIN_ENUMERATION によって指定されている範囲内で、残りのテーブルのジョインの順序を最適化します。この制限をオーバーすると、エラーが発生し、クエリは拒否されます。ユーザはクエリを単純化するか、上限を上げます。

通常、この値を変更する必要はありません。変更する場合は、MAX_JOIN_ENUMERATION を temporary オプション、または user オプションとして設定することをおすすめします。

MAX_PREFIX_PER_CONTAINS_PHRASE オプション

テキスト検索式で許可するプレフィクス単語の数を指定します。

TEXT インデックスを使用し、全文検索を実行するには、非構造化データ分析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。

MAX_PREFIX_PER_CONTAINS_PHRASE の構文と詳細な説明については、『Sybase IQ の非構造化データ分析の概要』を参照してください。

MAX_QUERY_PARALLELISM オプション

GROUP BY 操作の並列実行数と、**UNION** の分岐の数に上限を設定します。

指定できる値
CPU 数よりも少ないか、多いか、これに等しい整数

デフォルト値
64

スコープ
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
このパラメータは、オプティマイザによってクエリ演算の並列度が制限される上限値を設定します。これは、多くのクエリ・ジョイン、**GROUP BY**、**UNION**、**ORDER BY**、他のクエリ操作の CPU 使用率に影響を与えます。

64 以上の CPU コアを持つシステムでは、システムの CPU コアの総数 (最大 512) 以下の大きな値が有効です。いくつかの値を試し、システムおよびクエリでのこのパラメータの最適な値を決定できます。

64 またはそれ以下の CPU コアを持つシステムでは、過度なシステム時間が検出されないかぎり、この値を下げる必要はありません。検出された場合、この値を下げて、CPU のシステム時間が減少し、クエリの応答時間およびシステム全体のスループットが向上するかを判断します。

MAX_QUERY_TIME オプション

オプティマイザが長すぎるクエリを拒否できるよう、タイム・リミットを設定します。

指定できる値
0 ～ 2^{32} - 1 分

デフォルト値
0 (無効)

スコープ
セッションレベル(temporary)、ユーザ・レベル、PUBLIC に設定できます。

説明
クエリが MAX_QUERY_TIME の設定よりも長い時間実行されると、Sybase IQ はクエリを中止して、ユーザと IQ メッセージ・ファイルにメッセージを送信します。次に例を示します。

```
The operation has been cancelled -- Max_Query_Time exceeded.
```

MAX_QUERY_TIME はクエリのみに適用され、データベースの内容を変更する SQL 文には適用されません。

MAX_STATEMENT_COUNT オプション

1 つの接続で一度に使用できる準備文の最大数を制限するリソース・ガバナを指定します。

指定できる値
整数

デフォルト値
100

スコープ
個々の接続または PUBLIC グループに設定できます。すぐに有効になります。このオプションを設定するには、対象となる接続がどのようなものであっても DBA パーミッションが必要です。

説明
指定したリソース・ガバナを使用して、DBA はユーザが 1 つの接続で使用できる準備文の数を制限できます。1 つの接続に対してこの制限を超える操作を行うと、制限を超過していることを示すエラーを生成します。

ある接続でストアド・プロシージャが実行される場合、それはプロシージャの所有者のパーミッションに従って実行されます。ただし、プロシージャが使用するリソースは現在の接続から割り当てられます。

MAX_STATEMENT_COUNT を 0 (ゼロ) に設定すると、リソース制限を削除できます。

MAX_TEMP_SPACE_PER_CONNECTION オプション

接続ごとに使用されるテンポラリ・ストアの領域を制限します。

指定できる値
整数 (MB の数)

デフォルト値
0 (テンポラリ・ストアの使用には制限なし)

スコープ
このオプションを設定するには、DBA パーミッションが必要です。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
接続ごとの領域を制御することにより、このオプションは、DBA によるロードとクエリの両領域の管理を可能にします。 接続が MAX_TEMP_SPACE_PER_CONNECTION によって指定された実行時クオータを超える場合、Sybase IQ は、現在の文をロールバックし、次のメッセージを IQ メッセージ・ファイルまたはクライアント・ユーザに返します。

```
The current operation has been cancelled:
Max_Temp_Space_Per_Connection exceeded
```

バッファ・キャッシュを満杯にする条件には、読み書きエラー、メインまたは一時領域の不足、メモリ不足が含まれます。Sybase IQ は、これらの状況で最初に検出されたエラーを返し、DBA は適切なソリューションを決定する必要があります。 詳細については、『エラー・メッセージ』と『システム管理ガイド： 第 1 巻』の「トラブルシューティングのヒント」を参照してください。

分散クエリ処理トランザクションでは、Sybase IQ は、共有テンポラリ・ストアの QUERY_TEMP_SPACE_LIMIT オプションと MAX_TEMP_SPACE_PER_CONNECTION オプションの値セットを使用して、分散クエリに参加するすべてのノードによって使用される共有とローカルのテンポラリ領域の総量を制限します。これは、すべての単一クエリは、参加するノード数に関係なく、(IQ_SYSTEM_TEMP と IQ_SHARED_TEMP の DB 領域からの) テンポラリ領域総量制限を超えられないことを意味します。

たとえば、制限値が 100 で参加する 4 つのノードがテンポラリ領域にそれぞれ 25 ユニットを使用する場合、このクエリは制限内となります。ノードにより使用される領域の総量が 100 を超える場合、クエリはロールバックされます。

例
次の文はすべての接続に対して 500GB の制限を設定します。

```
SET OPTION
PUBLIC.MAX_TEMP_SPACE_PER_CONNECTION = 512000
```

次の文はすべての接続に対して 10TB の制限を設定します。

```
SET OPTION
PUBLIC.MAX_TEMP_SPACE_PER_CONNECTION = 10485760
```

次の文はユーザ `wilson` に対して 5,000MB の制限を設定します。

```
SET OPTION
wilson.MAX_TEMP_SPACE_PER_CONNECTION = 5000
```

参照：

- QUERY_TEMP_SPACE_LIMIT オプション (517 ページ)

MAX_WARNINGS オプション

受け取ることができる警告の最大数を制御します。

指定できる値
整数値

デフォルト値
$2^{48} - 1$

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
このオプションは、リジェクトされた値、ローのミスマッチなど、DDL コマンドの実行中に発生する警告の数を制限します。デフォルトでは、受信する数を制限しません。

MINIMIZE_STORAGE オプション

新規作成されたカラムでのディスク領域の使用を最小限に抑えます。

指定できる値
ON、OFF

デフォルト値
OFF

スコープ
PUBLIC グループに設定できます。または一時的に設定できます。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。このオプションはすぐに有効になります。

説明
MINIMIZE_STORAGE が ON の場合、IQ は、1 つのローに対して 1 バイトのディスク領域 (可能な場合) を使用して、新しいカラムの格納を最適化します。デフォルトでは、このオプションは PUBLIC グループで OFF であり、新規作成されたすべてのカラムに対して指定された領域最適化は発生しません。PUBLIC グループでは MINIMIZE_STORAGE が OFF であっても、temporary オプションと user オプションでは ON である場合、そのユーザ ID で作成された新しいカラムでは 1 バイト・ストレージが使用されます。

MINIMIZE_STORAGE を ON に設定することは、新しいカラムに **IQ UNIQUE 255** 句を指定することと同義ですが、後者の場合、特定のデータ型には 1 バイト・ストレージでは足りないことがあります。MINIMIZE_STORAGE を ON に設定すると、65536 を超えるユニークな値を持つカラムを除き、**IQ UNIQUE** を指定する必要はありません。

注意： 65536 を超える IQ UNIQUE の値は、3 バイトのインデックスを作成できます。このような値は、以前に MINIMIZE_STORAGE が ON の場合に作成防止のために使用されました。MINIMIZE_STORAGE が ON の場合に特定の領域最適化を防止する場合、IQ UNIQUE に 16777216 を超える制約値を設定します。

メイン・メモリとカラム数の比が大きい場合、MINIMIZE_STORAGE を ON にすると有効です。そうでない場合、新しいカラムの領域は、通常このオプションを OFF にすると有効になります。

CREATE TABLE または **ALTER TABLE ADD COLUMN** で **IQ UNIQUE** を明示的に指定すると、そのカラムの MINIMIZE_STORAGE オプションの設定が上書きされます。

参照：
- FP_LOOKUP_SIZE オプション (448 ページ)
- INDEX_ADVISOR オプション (459 ページ)

MIN_PASSWORD_LENGTH オプション

データベースの新しいパスワードの長さの最小値を設定します。

指定できる値
0 以上の整数

値はバイト単位です。シングルバイト文字セットの場合、これは文字数と同じになります。

デフォルト値
0 文字

スコープ
PUBLIC グループに設定できます。すぐに有効になります。このオプションを設定するには、DBA パーミッションが必要です。

説明
このオプションを使用すると、DBA は、すべての新しいパスワードの長さに最小値を設定してセキュリティを強化できます。既存のパスワードには影響しません。

例
新しいパスワードの最小長を 6 バイトに設定します。

```
SET OPTION PUBLIC.MIN_PASSWORD_LENGTH = 6
```

MONITOR_OUTPUT_DIRECTORY オプション

IQ バッファ・キャッシュ・モニタの出力ファイルの場所を制御します。

指定できる値
文字列

デフォルト値
データベースと同じディレクトリ

スコープ
PUBLIC グループに設定できます。すぐに有効になります。このオプションを設定するには、DBA パーミッションが必要です。

説明
MONITOR_OUTPUT_DIRECTORY は、モニタ対象や使用するモニタ・モードに関係なく、作成される IQ モニタリング出力ファイルが配置されるディレクトリを制御します。モニタの起動に使われるダミー・テーブルは、テンポラリ・テーブルで

も永久テーブルでもかまいません。どの物理マシン上にあるディレクトリでもかまいません。

すべてのモニタの出力ファイルは、モニタが実行されている期間使用されます。接続が無効になった後は使用されません。出力ファイルは、モニタリングの実行が停止した後も残ります。1つの接続では最大2つまでのパフォーマンス・モニタを同時に実行できます。1つはメイン・キャッシュ用で、もう1つはテンポラリ・キャッシュ用です。1つの接続で連続して何回でもモニタを実行できます。

DBA は、PUBLIC 設定を使用して、すべてのモニタ出力を同じディレクトリに設定することも、ユーザごとのディレクトリに設定することもできます。

例
この例では、モニタ出力用のテンポラリ・テーブルを宣言し、その配置場所を設定し、そこにファイルを送信するモニタをメインおよびテンポラリ・バッファ・キャッシュに対して開始する方法を示します。

注意： この例では、出力ディレクトリ文字列が "/tmp" と "tmp/" の両方に設定されています。末尾のスラッシュ (/) は正しく、インタフェースによってサポートされます。この例は、バッファ・キャッシュ・モニタが永久テーブルを必要とせず、テンポラリ・テーブルを使用できることを示します。

```
declare local temporary table dummy_monitor (dummy_column integer)
```

```
set option Monitor_Output_Directory = "/tmp"
iq utilities main into dummy_monitor start monitor '-debug -interval
2'
```

```
set option Monitor_Output_Directory = "tmp/"
```

```
iq utilities private into dummy_monitor start monitor '-debug -
interval 2'
```

MPX_AUTOEXCLUDE_TIMEOUT オプション

コーディネータ・ノード上でのセカンダリ・ノードの自動除外のタイムアウトを指定します。

この値を 0 にすると、ノードは自動的に除外されません。このオプションは、指定済みのフェールオーバ・ノードには適用されません。 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。MPX_AUTOEXCLUDE_TIMEOUT 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。

MPX_HEARTBEAT_FREQUENCY オプション

ハートビート・スレッドがウェイクアップし、セカンダリ・ノードの接続プールをクリーンアップするまでの間隔を指定します。

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。 MPX_HEARTBEAT_FREQUENCY 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。

MPX_IDLE_CONNECTION_TIMEOUT オプション

セカンダリ・ノード上の接続プールにある未使用の接続がクローズされるまでの待機時間を指定します。

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。
MPX_IDLE_CONNECTION_TIMEOUT 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。

MPX_MAX_CONNECTION_POOL_SIZE オプション

セカンダリ・ノード上の接続プールで許容される接続の最大数を指定します。

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。
MPX_MAX_CONNECTION_POOL_SIZE 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。

MPX_MAX_UNUSED_POOL_SIZE オプション

セカンダリ・ノード上の接続プールでの未使用接続の最大数を指定します。

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。 MPX_MAX_UNUSED_POOL_SIZE 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。

MPX_WORK_UNIT_TIMEOUT オプション

マルチプレックス DQP リーダが、分散された作業単位を別の DQP ワーカ・ノードに再割り当てするまでの時間 (秒単位) です。

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。

MPX_WORK_UNIT_TIMEOUT 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。

NEAREST_CENTURY オプション [TSQL]

文字列から日付への変換における 2 桁の年の解釈を制御します。

指定できる値
0 – 100

デフォルト値
50

説明
NEAREST_CENTURY は、文字列から日付またはタイムスタンプに変換する際の 2 桁の年の処理の制御に使用します。

NEAREST_CENTURY 設定は、ロールオーバ・ポイントとして動作する数値です。この値より小さい 2 桁の年は 20yy に変換され、この値以上の年は 19yy に変換されます。

Adaptive Server Enterprise と Sybase IQ は最も近い世紀を使用するので、*yy* が 50 より小さい年は 20yy に設定されます。

NOEXEC オプション

オプティマイザ・クエリ・プランを生成します。実行はしません。

指定できる値
ON、OFF

デフォルト値
OFF

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
クエリの処理方法を決定する際、IQ オプティマイザは、各クエリをクエリ・エンジンがどう処理するかを記述したマップとして、クエリ・プランを生成します。このオプションが ON の場合、オプティマイザは各クエリのプランをクエリ・エ

ンジンに送る代わりに IQ メッセージ・ファイルに送信します。NOEXEC は、クエリと、クエリを含むコマンドに影響します。

NOEXEC を ON に設定すると、**INSERT...VALUES**、**INSERT...SELECT**、**INSERT...LOCATION**、**SELECT...INTO**、**LOAD TABLE**、**UPDATE**、**TRUNCATE TABLE**、**DELETE**、**SYNCHRONIZE JOIN INDEX**、更新可能なカーソルの各操作が実行できなくなります。

EARLY_PREDICATE_EXECUTION オプションが ON の場合、NOEXEC オプションが ON の場合でも、Sybase IQ はクエリ・プランの生成に先立ってすべてのクエリのローカル述部を実行します。 生成されたクエリ・プランは実行時プランと同一です。

参照：

- EARLY_PREDICATE_EXECUTION オプション (444 ページ)

NON_ANSI_NULL_VARCHAR オプション

挿入／ロード／更新時に長さゼロの VARCHAR データを NULL として扱うかどうかを制御します。

指定できる値
ON、OFF

デフォルト値
OFF

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
NON_ANSI_NULL_VARCHAR は、ロードまたは更新時の長さゼロの VARCHAR データの処理を非 ANSI (Version 12.03.1) 動作に戻す場合に使用します。このオプションを OFF にした場合、長さゼロの VARCHAR データはロード／挿入／更新時に長さゼロとして保存されます。このオプションを ON にした場合、長さゼロの VARCHAR データはロード／挿入／更新時に NULL として保存されます。

NON_KEYWORDS オプション [TSQL]

個々のキーワードをオフにして、識別子として使用できるようにします。

指定できる値
文字列

デフォルト値
'' (空の文字列)

説明
NON_KEYWORDS は、各キーワードを無効にするために使用します。現在データベースにキーワードである識別子がある場合、すべてのアプリケーションまたはスクリプトで識別子を二重引用符で囲むか、NON_KEYWORDS オプションを使用してキーワードを無効にできます。

次の文を記述すると、**TRUNCATE** と **SYNCHRONIZE** はキーワードとして認識されません。

```
SET OPTION NON_KEYWORDS = 'TRUNCATE, SYNCHRONIZE'
```

このオプションによる新規設定が以前の設定に置き換わります。次の文は以前の設定内容をすべてクリアします。

```
SET OPTION NON_KEYWORDS =
```

このオプションを使用すると、オフにしたキーワードを使用する SQL 文は構文エラーが発生します。

NOTIFY_MODULUS オプション

特定のコマンドから発行される通知メッセージのデフォルト周期を制御します。

指定できる値
整数値

デフォルト値
100000

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
このオプションは、通知メッセージを生成する特定のコマンドに対して Sybase IQ が発行する通知メッセージのデフォルト数を設定するのに使用します。 **CREATE INDEX**、**LOAD TABLE**、**DELETE** などのコマンドで **NOTIFY** 句を指定すると、このオプションの設定が上書きされます。**NOTIFY** 句をサポートしない **SYNCHRONIZE JOIN INDEX** などのコマンドでは、このオプションの値が常に使用されます。 デフォルトでは、受け取るメッセージの数を制限しません。

ODBC_DISTINGUISH_CHAR_AND_VARCHAR オプション

Sybase IQ と SQL Anywhere ODBC ドライバが CHAR カラムを表現する方法を制御します。

指定できる値
ON、OFF

デフォルト値
OFF

説明
接続が開かれると、Sybase IQ と SQL Anywhere ODBC ドライバはこのオプションの設定を使用して CHAR カラムの表現方法を決定します。
ODBC_DISTINGUISH_CHAR_AND_VARCHAR が OFF (デフォルト値) に設定されている場合、CHAR カラムは SQL_VARCHAR として表現されます。 このオプションが ON に設定されている場合、CHAR カラムは SQL_CHAR として表現されます。 VARCHAR カラムは、常に SQL_VARCHAR として表現されます。

ON_CHARSET_CONVERSION_FAILURE オプション

文字変換でエラーが起きた場合の対応を制御します。

指定できる値
文字列。指定可能な値については、「説明」を参照してください。

デフォルト値
IGNORE

説明
ON_CHARSET_CONVERSION_FAILURE は、文字変換でエラーが起きた場合の対応を制御します。

文字変換エラー	アクション
IGNORE	エラーも警告も表示しません。
WARNING	置き換えと不正な文字を警告として報告します。不正な文字は変換されません。
ERROR	置き換えと不正な文字をエラーとして報告します。

シングルバイトからシングルバイトへの変換では、置き換えと不正な文字を報告できないため、このオプションは IGNORE に設定する必要があります。

ON_ERROR オプション [Interactive SQL]

Interactive SQL の文の実行中にエラーが起こった場合の動作を制御します。

指定できる値
文字列。指定可能な値については、「説明」を参照してください。

デフォルト値
PROMPT

説明
文の実行時にエラーが起きた場合の対応を制御します。

- STOP – Interactive SQL は、ファイルからの文の実行を停止し、入力用の文ウィンドウに戻ります。
- PROMPT – Interactive SQL は、ユーザに継続したいかどうかを確認するプロンプトを表示します。
- CONTINUE – エラーが表示されますが、Interactive SQL は文の実行を続行します。
- EXIT – Interactive SQL は終了します。
- NOTIFY_CONTINUE – エラーが返され、続行するには [Enter] キーを押すか、[OK] をクリックするよう求めるプロンプトがユーザに表示されます。
- NOTIFY_STOP – エラーが返され、文の実行を中止するには [Enter] キーを押すか、[OK] をクリックするよう求めるプロンプトがユーザに表示されます。
- NOTIFY_EXIT – エラーが返され、Interactive SQL を終了するには [Enter] キーを押すか、[OK] をクリックするよう求めるプロンプトがユーザに表示されます。

.SQL ファイルを実行している場合は、STOP と EXIT のどちらに設定しても同じ結果となります。

ON_TSQL_ERROR オプション [TSQL]

ストアド・プロシージャでのエラー処理を制御します。

指定できる値
文字列。指定可能な値については、「説明」を参照してください。

デフォルト値
CONDITIONAL

説明
ON_TSQL_ERROR は、ストアド・プロシージャでのエラー処理を制御します。

- STOP – エラーの検出と同時に実行を停止します。
- CONDITIONAL – プロシージャが **ON EXCEPTION RESUME** を使用し、エラーのすぐ後の文がエラーを処理する場合は継続します。そうでない場合は終了します。
- CONTINUE – 次の文に関係なく実行が継続されます。 複数のエラーがある場合は、ストアド・プロシージャで最初に検出されたエラーが返されます。 このオプションによる動作は Adaptive Server Enterprise の動作とほとんど同じです。

ON_TSQL_ERROR の CONDITIONAL と CONTINUE の設定は、Adaptive Server Enterprise との互換性のために使用されます。Adaptive Server Enterprise の動作を最も忠実にシミュレートするのは、CONTINUE です。 ただし、特に新規の Transact-SQL ストアド・プロシージャを作成する場合などは、より早い段階でエラーを報告する CONDITIONAL の設定をおすすめします。

Adaptive Server Enterprise との互換性の詳細については、『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』を参照してください。

このオプションが STOP または CONTINUE に設定されている場合、これは CONTINUE_AFTER_RAISERROR オプションの設定を上書きします。ただし、このオプションを CONDITIONAL (デフォルト) に設定した場合は、**RAISERROR** 文の後の動作は CONTINUE_AFTER_RAISERROR オプションの設定によって決まります。

参照：
- CREATE PROCEDURE 文 (137 ページ)
- CREATE PROCEDURE 文 [T-SQL] (144 ページ)
- RAISERROR 文 [T-SQL] (315 ページ)
- CONTINUE_AFTER_RAISERROR オプション [TSQL] (418 ページ)

OS_FILE_CACHE_BUFFERING オプション

IQ メイン DB 領域のファイル・システム・バッファリングの使用を制御します。

指定できる値
ON、OFF

デフォルト値
OFF。デフォルトは新規作成されたデータベースに対してのみ有効です。

スコープ
PUBLIC グループのみに設定できます。変更を有効にするには、データベースを停止し、再起動させてください。このオプションを設定するには、DBA パーミッションが必要です。

説明
OS_FILE_CACHE_BUFFERING を OFF に設定すると、IQ メイン・ストア・ファイルのためにファイル・システム・バッファリングを行いません。ファイル・システム・バッファリングをオフにすると、ファイル・システム・バッファからのデータのコピーはメイン IQ バッファ・キャッシュにキャッシュされます。通常これにより、IQ バッファ・マネージャとオペレーティング・システムのファイル・システム・バッファとのメモリ競合によるページングが減ります。OS_FILE_CACHE_BUFFERING によってページングが減少してパフォーマンスが向上することがありますが、データベースの IQ ページ・サイズがファイル・システムのブロック・サイズより小さい場合 (通常はテストのような場合のみ)、パフォーマンスは低下し、特にマルチユーザの場合に悪くなります。

このオプションを使用するかどうかは、各マシンの環境に応じて実際に動作させてから判断してください。新しい設定を有効にするには、データベースを再起動する必要があります。

このダイレクト I/O パフォーマンス・オプションは、Sun Solaris UFS、Linux、Linux IBM、AIX、Windows ファイル・システムでのみ有効です。このオプションは HP-UX と HP-UXi には影響しません。また、ロー・ディスク上に構築されたデータベースにも影響はありません。Linux では、ダイレクト I/O はカーネル・バージョン 2.6.x でサポートされます。

Linux カーネル・バージョン 2.6 および AIX でダイレクト I/O を有効にするには、環境変数 IQ_USE_DIRECTIO を 1 に設定します。Linux カーネル・バージョン 2.6 および AIX では、ダイレクト I/O はデフォルトで無効になっています。IQ_USE_DIRECTIO は、Sun Solaris と Windows には影響しません。

注意： Sybase IQ は、Linux カーネル・バージョン 2.4 でダイレクト I/O をサポートしていません。 Linux カーネル・バージョン 2.4 で IQ_USE_DIRECTIO 環境変数を設定すると、Sybase IQ サーバは起動しません。 エラー `Error: Invalid Block I/O argument, maybe <pathname> is a directory, or it exceeds maximum file size limit for the platform, or trying to use Direct IO on unsupported OS` が報告されます。

`OS_FILE_CACHE_BUFFERING_TEMPDB` は、IQ テンポラリ・ストア・ファイルのファイル・システム・バッファリングを制御します。

参照：
- OS_FILE_CACHE_BUFFERING_TEMPDB オプション (502 ページ)

OS_FILE_CACHE_BUFFERING_TEMPDB オプション

IQ テンポラリ DB 領域のファイル・システム・バッファリングの使用を制御します。

指定できる値
ON、OFF

デフォルト値
OFF

スコープ
PUBLIC グループのみに設定できます。変更を有効にするには、データベースを停止し、再起動させてください。このオプションを設定するには、DBA パーミッションが必要です。

説明
`OS_FILE_CACHE_BUFFERING_TEMPDB` を OFF に設定すると、IQ テンポラリ・ストア・ファイルのためにファイル・システム・バッファリングを行いません。ファイル・システム・バッファリングをオフにすると、ファイル・システム・バッファからのデータのコピーはテンポラリ IQ バッファ・キャッシュにキャッシュされます。通常これにより、IQ バッファ・マネージャとオペレーティング・システムのファイル・システム・バッファとのメモリ競合によるページングが減ります。`OS_FILE_CACHE_BUFFERING_TEMPDB` によってページングが減少してパフォーマンスが向上することがありますが、データベースの IQ ページ・サイズがファイル・システムのブロック・サイズより小さい場合 (通常はテストのような場合のみ)、パフォーマンスは低下し、特にマルチユーザの場合に悪くなります。

このオプションを使用するかどうかは、各マシンの環境に応じて実際に動作させてから判断してください。新しい設定を有効にするには、データベースを再起動する必要があります。

このダイレクト I/O パフォーマンス・オプションは、Sun Solaris UFS、Linux、Linux IBM、AIX、Windows ファイル・システムでのみ有効です。このオプションは HP-UX と HP-UXi には影響しません。また、ロー・ディスク上に構築されたデータベースにも影響はありません。Linux では、ダイレクト I/O はカーネル・バージョン 2.6.x でサポートされます。

Linux カーネル・バージョン 2.6 および AIX でダイレクト I/O を有効にするには、環境変数 IQ_USE_DIRECTIO を 1 に設定します。Linux カーネル・バージョン 2.6 および AIX では、ダイレクト I/O はデフォルトで無効になっています。IQ_USE_DIRECTIO は、Sun Solaris と Windows には影響しません。

注意： Sybase IQ は、Linux カーネル・バージョン 2.4 でダイレクト I/O をサポートしていません。Linux カーネル・バージョン 2.4 で IQ_USE_DIRECTIO 環境変数を設定すると、Sybase IQ サーバは起動しません。エラー `Error: Invalid Block I/O argument, maybe <pathname> is a directory, or it exceeds maximum file size limit for the platform, or trying to use Direct IO on unsupported OS` が報告されます。

`OS_FILE_CACHE_BUFFERING` は、IQ メイン・ストア・ファイルのファイル・システム・バッファリングを制御します。

参照：

- OS_FILE_CACHE_BUFFERING オプション (501 ページ)

PASSWORD_EXPIRY_ON_NEXT_LOGIN オプション

ユーザにログイン・ポリシーが割り当てられ、ポリシーにおけるこのオプションが ON に設定されている場合、ユーザのパスワードは次のログイン時に有効期限切れとマーク付けされます。

詳細については、『SQL Anywhere サーバー – SQL リファレンス』の「SQL 文」>「SQL 文」>「CREATE LOGIN POLICY 文」を参照してください。

注意： このリファレンスは SQL Anywhere マニュアルにリンクされています。

PASSWORD_GRACE_TIME オプション

ログインは可能であるにもかかわらず、デフォルトのポスト・ログイン・プロシージャによって警告が発行される、パスワードの有効期間切れ前の日数を指定します。

詳細については、『SQL Anywhere サーバー – SQL リファレンス』の「SQL 文」>「SQL 文」>「CREATE LOGIN POLICY 文」を参照してください。

注意： このリファレンスは SQL Anywhere マニュアルにリンクされています。

PASSWORD_LIFE_TIME オプション

パスワードの変更が必要になるまでの最大日数を指定します。

詳細については、『SQL Anywhere サーバー – SQL リファレンス』の「SQL 文」>「SQL 文」>「CREATE LOGIN POLICY 文」を参照してください。

注意： このリファレンスは SQL Anywhere マニュアルにリンクされています。

POST_LOGIN_PROCEDURE オプション

結果セットにユーザが正常にログインした直後にクライアント・アプリケーションにより表示されるメッセージが記載されたログイン・プロシージャを指定します。

指定できる値
文字列

デフォルト値
dbo.sa_post_login_procedure

スコープ
個々の接続または PUBLIC グループに設定できます。このオプションを設定するには、DBA パーミッションが必要です。すぐに有効になります。

説明
デフォルトのポスト・ログイン・プロシージャ **dbo.sa_post_login_procedure** は、ユーザが正常にログインした直後に実行されます。

DBA 権限がある場合、ログイン後の処理をカスタマイズできます。そのためには、プロシージャを新規作成して、その新しいプロシージャを呼び出すよう POST_LOGIN_PROCEDURE を設定します。通常のオペレーション中には **dbo.sa_post_login_procedure** を編集しないでください。使用するすべてのデータベースで、カスタマイズされたポスト・ログイン・プロシージャを作成してください。

ポスト・ログイン・プロシージャは、クライアント・アプリケーション Interactive SQL、Interactive SQL Classic、Sybase Central 用の IQ プラグインをサポートしています。

参照：

- LOGIN_PROCEDURE オプション (478 ページ)

PRECISION オプション

カタログ・ストアのクエリに限定して、10 進数の算術の結果に使用できる最大桁数を指定します。

指定できる値
126

デフォルト値
126

スコープ
PUBLIC 設定のみが可能です。

説明
精度は、小数点の左右の合計桁数です。PRECISION 値は 126 固定です。SCALE オプションは、カタログ・ストア上のクエリの計算結果が PRECISION で指定された最大値にトランケートされた場合、小数点以下の最大桁数を指定するのに使用します。

注意： SQL Anywhere では、数値関数でサポートされる最大値は 255 です。数値関数の精度がサポートされている最大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'`

参照：
- SCALE オプション (522 ページ)
- MAX_CLIENT_NUMERIC_PRECISION オプション (480 ページ)

PREFETCH オプション

フェッチの ON と OFF を切り替えることができます。また、ALWAYS 値を使用して、カーソル結果 (SENSITIVE カーソル タイプとプロキシ・テーブルに関わるカーソルに関するものを含む) をプリフェッチすることもできます。

指定できる値
ON、OFF、ALWAYS

デフォルト値
ON

スコープ
個々の接続または PUBLIC グループに設定できます。すぐに有効になります。

説明
カタログ・ストアの場合のみ、PREFETCH はローがクライアント・アプリケーションで使用できるようになる前にクライアントにフェッチできるかどうかを制御します。一度にいくつものローをフェッチすると、クライアント・アプリケーションが一度に 1 つのローを要求した場合でも (たとえば、カーソルのローをループする場合)、応答時間を短縮し、データベースへの要求の数を制限して全体的なスループットを向上させます。

PREFETCH の設定は、Open Client 接続と JDBC 接続、そして IQ ストアに無視されます。

PREFETCH_BUFFER_LIMIT オプション

プリフェッチに使用するメモリ量を指定します。

指定できる値
整数

デフォルト値
0

スコープ
PUBLIC グループのみに設定できます。このオプションを設定するには、DBA パーミッションが必要です。変更を有効にするには、データベース・サーバを停止し、再起動します。

説明
PREFETCH_BUFFER_LIMIT は、Sybase IQ がプリフェッチ (データベース・ページの先読み) に使用できるキャッシュ・ページの数を定義します。

Sybase 製品の保守契約を結んでいるサポート・センタからの指示があった場合を除いて、このオプションは使用しないでください。

参照：
- PREFETCH_BUFFER_PERCENT オプション (507 ページ)

PREFETCH_BUFFER_PERCENT オプション

プリフェッチに使用するメモリをパーセントで指定します。

指定できる値
0 – 100

デフォルト値
40

スコープ
PUBLIC グループのみに設定できます。このオプションを設定するには、DBA パーミッションが必要です。変更を有効にするには、データベース・サーバを停止し、再起動します。

説明
PREFETCH_BUFFER_PERCENT is an alternative to PREFETCH_BUFFER_LIMIT, as it specifies the percentage of cache available for use in prefetching.

Sybase 製品の保守契約を結んでいるサポート・センタからの指示があった場合を除いて、このオプションは使用しないでください。

参照：

- PREFETCH_BUFFER_LIMIT オプション (506 ページ)

PREFETCH_GARRAY_PERCENT オプション

HG インデックスに挿入できるプリフェッチ・リソースをパーセントで指定します。

指定できる値
0 – 100

デフォルト値
60

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
PREFETCH_SORT_PERCENT と同様、このオプションは、**HG** インデックスに挿入するときに使用できるプリフェッチ・リソースのパーセント量を指定するのに使用します。

Sybase 製品の保守契約を結んでいるサポート・センタからの指示があった場合を除いて、このオプションは使用しないでください。

PREFETCH_SORT_PERCENT オプション

ソート・オブジェクトが使用するプリフェッチ・リソースをパーセントで指定します。

指定できる値
0 – 100

デフォルト値
20

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
PREFETCH_SORT_PERCENT は、1 つのソート・オブジェクトが使用するプリフェッチ・リソースをパーセンテージで指定するのに使用します。この値を大きくすると、シングルユーザの場合の挿入と削除のパフォーマンスが向上しますが、マルチユーザ動作に対して悪影響を及ぼす可能性があります。

Sybase 製品の保守契約を結んでいるサポート・センタからの指示があった場合を除いて、このオプションは使用しないでください。

PRESERVE_SOURCE_FORMAT オプション [database]

プロシージャ、ビュー、イベント・ハンドラのオリジナルのソース定義をシステム・ファイル内に保存するかどうかを制御します。保存する場合は、フォーマットされたソースが SYSTABLE、SYSPROCEDURE、SYSEVENT のカラムのソースに保存されます。

指定できる値
ON、OFF

デフォルト値
ON

スコープ
PUBLIC 設定のみが可能です。

説明
PRESERVE_SOURCE_FORMAT が ON の場合、サーバは、プロシージャ、ビュー、イベントの **CREATE** 文と **ALTER** 文からフォーマットされたソースを保存し、オリジナルのソース定義を適切なシステム・テーブルの source カラムに配置します。

フォーマットされていないソース・テキストは同じシステム・テーブル内の proc_defn カラムと view_defn カラムに格納されます。ただし、これらの定義は Sybase Central で読みやすく表示することはできません。フォーマットされた source カラムでは、希望どおりのスペース、コメント、大文字／小文字が設定された形式で定義を表示できます。

このオプションをオフにすると、オブジェクト定義を保存するために使用されるデータベース領域を減らすことができます。このオプションは、PUBLIC グループのみに設定できます。

QUERY_DETAIL オプション

追加のクエリ情報をクエリ・プランの Query Detail セクションに含めるかどうかを指定します。

指定できる値
ON、OFF

デフォルト値
OFF

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
QUERY_DETAIL と QUERY_PLAN (または QUERY_PLAN_AS_HTML) が両方オンになっていると、Sybase IQ はクエリ・プランの作成時にクエリに関する追加情報を表示します。 QUERY_PLAN と QUERY_PLAN_AS_HTML が OFF の場合は、このオプションは無視されます。

QUERY_PLAN が ON (デフォルト値) の場合、QUERY_DETAIL も ON である場合は特に、メッセージ・ログ・ラッピングまたはメッセージ・ログ・アーカイブを有効にしてメッセージ・ログ・ファイルが満杯にならないようにしてください。メッセージ・ログ・ラッピングの詳細については、『システム管理ガイド：第 1 巻』を参照してください。

参照：

- QUERY_PLAN オプション (511 ページ)
- QUERY_PLAN_AS_HTML オプション (512 ページ)

QUERY_NAME オプション

クエリ・プランで実行されるクエリに名前を付けます。

指定できる値
引用符で区切った文字列 (最大 80 字)

デフォルト値
'' (空の文字列)

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
QUERY_NAME オプションには、次のように、引用符で区切られた任意の文字列 (80 字以下) を指定できます。次に例を示します。

```
set temporary option Query_Name = 'my third query'
```

このオプションが設定されていると、.iqmsg ファイルまたは .html ファイルに送信されるクエリ・プランの先頭近くに、次のような行が含まれます。

```
Query_Name:  'my third query'
```

スクリプト内の各クエリの前にこのオプションが別の値に設定されている場合、特定のクエリに対して正しいクエリ・プランを選択するのがずっと容易になります。クエリ名は HTML クエリ・プランのファイル名に追加されます。このオプションが、クエリに対してその他の影響を及ぼすことはありません。

QUERY_PLAN オプション

追加のクエリ・プランが Sybase IQ メッセージ・ファイルに印刷されるかどうかを指定します。

指定できる値
ON、OFF

デフォルト値
ON

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
このオプションが ON になっている場合、Sybase IQ は、IQ メッセージ・ファイルのテキスト・クエリ・プランを作成します。 これらのクエリ・プランは、クエリ・ツリー・トポロジおよび最適化と実行についての詳細を表示します。 OFF の場合、このようなメッセージは生成されません。情報は、<dbname>.iqmsg ファイルに送信されます。

参照：
- QUERY_DETAIL オプション (509 ページ)
- QUERY_PLAN_AFTER_RUN オプション (511 ページ)
- QUERY_PLAN_AS_HTML オプション (512 ページ)

QUERY_PLAN_AFTER_RUN オプション

クエリの実行完了後に、クエリ・プラン全体を表示します。

指定できる値
ON、OFF

デフォルト値
OFF

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
QUERY_PLAN_AFTER_RUN がオンになっていると、クエリの実行が完了するとクエリ・プランが表示されます。これにより、クエリ・プランに、クエリの各ノードに渡された実際のロー数などの情報を追加できます。

このオプションが機能するためには、QUERY_PLAN オプションが ON (デフォルト値) である必要があります。このオプションを QUERY_DETAIL と組み合わせて使用して、クエリ・プラン・レポートに追加の情報を生成することもできます。

参照：
- QUERY_DETAIL オプション (509 ページ)
- QUERY_PLAN オプション (511 ページ)
- QUERY_PLAN_AS_HTML オプション (512 ページ)

QUERY_PLAN_AS_HTML オプション

Web ブラウザで表示できるグラフィカルなクエリ・プランを HTML フォーマットで生成します。

指定できる値
ON、OFF

デフォルト値
OFF

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
QUERY_PLAN_AS_HTML は、グラフィカルなクエリ・プランを HTML フォーマットで生成します。

このオプションを設定する場合、クエリ・プランに関連付けられているクエリを識別できるよう、各クエリに QUERY_NAME オプションも設定してください。

Sybase IQ は、.iqmsg ファイルと同じディレクトリの、次のような名前のファイルにクエリ・プランを書き込みます。

user-name_query-name_YYYYMMDD_HHMMSS_query-number.html

たとえば、ユーザ DBA がテンポラリ・オプション QUERY_NAME を 'Query_1123' に設定した場合、2011 年 5 月 18 日の午前 8 時 30 分ちょうどに作

成されたファイルは DBA_Query_1123_20110518_083000_1.html という名前になります。既存のファイルを上書きしないよう、日付、時間、およびユニークな値がファイル名に自動的に付加されます。

注意： この機能を利用する場合、.iqmsg とログ・ファイルが拡張できるだけの余裕があるかどうか、ディスク領域の利用状況を監視してください。メッセージ・ログ・ラッピングまたはメッセージ・ログ・アーカイブを有効にしてメッセージ・ログ・ファイルが満杯にならないようにしてください。

メッセージ・ログ・ラッピングの詳細については、『システム管理ガイド：第 1 巻』を参照してください。

QUERY_PLAN_AS_HTML は、QUERY_PLAN オプションの設定とは関係なく機能します。つまり、QUERY_PLAN_AS_HTML が ON の場合、QUERY_PLAN が ON であっても OFF であっても、HTML フォーマットのクエリ・プランが得られます。

広く利用されているブラウザの新しいバージョンではこの機能がサポートされています。一部のブラウザでは、非常に複雑なクエリに対して生成されたプランで表示の問題が発生することもあります。

参照：
- QUERY_NAME オプション (510 ページ)
- QUERY_PLAN オプション (511 ページ)
- QUERY_PLAN_AFTER_RUN オプション (511 ページ)

QUERY_PLAN_AS_HTML_DIRECTORY オプション

Sybase IQ が HTML のクエリ・プランを書き込むディレクトリを指定します。

指定できる値
ディレクトリのパス名を含む文字列

デフォルト値
'' (空の文字列)

スコープ
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。このオプションを設定するには、DBA パーミッションが必要です。すぐに有効になります。

説明
QUERY_PLAN_AS_HTML オプションが ON で、
QUERY_PLAN_AS_HTML_DIRECTORY オプションでディレクトリが指定されている場合、Sybase IQ は指定されたディレクトリに HTML のクエリ・プランを書き込

みます。 このオプションでは、HTML クエリ・プランがサーバ・ディレクトリ外で作成できるため、セキュリティが強化されます。
QUERY_PLAN_AS_HTML_DIRECTORY オプションが使用されない場合、クエリ・プランはデフォルト・ディレクトリ (.iqmsg ファイルのディレクトリ) に送信されます。

QUERY_PLAN_AS_HTML オプションが ON で、
QUERY_PLAN_AS_HTML_DIRECTORY に存在しないディレクトリが指定されている場合、Sybase IQ は HTML クエリ・プランを保存せず、エラーが返されます。 この場合、クエリの実行は続けられ、メッセージが IQ メッセージ・ファイルにログ記録されます。DBA はこの記録によって、HTML クエリ・プランが書き込まれなかったことを確認できます。 指定されたディレクトリ・パスまたはディレクトリのパーミッションが適切でない場合、"Error opening HTML Query plan: *file-name*" が .iqmsg ファイルに書き込まれます。

例
例として、/system1/users/DBA/html_plans というディレクトリを作成し、このディレクトリに適切なパーミッションを設定します。次に、このオプションを設定してクエリを実行します。

```
SET TEMPORARY OPTION QUERY_PLAN_AS_HTML = ‘ON’;
SET TEMPORARY OPTION QUERY_PLAN_AS_HTML_DIRECTORY = ‘/system1/users/
DBA/html_plans’;
SELECT col1 FROM tab1;
```

HTML クエリ・プランが、指定されたディレクトリ /system1/users/DBA/html_plans に書き込まれます。

参照：
- QUERY_PLAN_AS_HTML オプション (512 ページ)

QUERY_PLAN_TEXT_ACCESS オプション

Interactive SQL クライアントからクエリ・プランにアクセスすること、または SQL のさまざまな関数を使用してプランを取得することの許可／禁止を切り替えます。

指定できる値
ON、OFF

デフォルト値
OFF

スコープ
このオプションを変更するには、DBA パーミッションが必要です。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
QUERY_PLAN_TEXT_ACCESS オプションが ON の場合、Interactive SQL クライアントからクエリ・プランを表示、保存、出力できます。オプションが OFF の場合、クエリ・プランはキャッシュされないので、クエリ・プランが関係する他のデータベース・オプションが Interactive SQL クライアントのクエリ・プランの表示に影響を与えることはありません。次のエラー・メッセージが表示されます。

```
No plan available. The database option QUERY_PLAN_TEXT_ACCESS is OFF.
```

参照：
- QUERY_DETAIL オプション (509 ページ)
- QUERY_PLAN_AFTER_RUN オプション (511 ページ)
- QUERY_PLAN_AS_HTML オプション (512 ページ)
- QUERY_PLAN_TEXT_CACHING オプション (515 ページ)
- OUTPUT 文 [Interactive SQL] (304 ページ)

QUERY_PLAN_TEXT_CACHING オプション

ユーザが実行するクエリに対して Sybase IQ が IQ プランを生成、キャッシュするかどうかを指定できるようにします。

指定できる値
ON、OFF

デフォルト値
OFF

スコープ
DBA パーミッションがなくても、このオプションを変更できます。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
IQ クエリ・プランのサイズは異なり、クエリが複雑な場合は非常に大きくなることがあります。表示するプランを Interactive SQL クライアントにキャッシュするには、高いリソース要件が必要とされることがあります。
QUERY_PLAN_TEXT_CACHING オプションを使用すると、プランをキャッシュで

きます。このオプションを OFF にすると (デフォルト)、クエリ・プランはそのユーザの接続のためにはキャッシュされません。

注意： QUERY_PLAN_TEXT_ACCESS を OFF にすると、QUERY_PLAN_TEXT_CACHING の設定にかかわらず、そのユーザからの接続ではクエリ・プランはキャッシュされなくなります。

参照：

- QUERY_DETAIL オプション (509 ページ)
- QUERY_PLAN_AFTER_RUN オプション (511 ページ)
- QUERY_PLAN_AS_HTML オプション (512 ページ)
- QUERY_PLAN_TEXT_ACCESS オプション (514 ページ)
- OUTPUT 文 [Interactive SQL] (304 ページ)

QUERY_ROWS_RETURNED_LIMIT オプション

結果セットの予想サイズにもとづき、クエリを拒否するためのローのスレッショルドを設定します。

指定できる値
整数値

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 は、結果ローの予想サイズが QUERY_ROWS_RETURNED_LIMIT の値より大きいクエリを受け取ると、次のようなメッセージを生成してクエリを拒否します。

```
Query rejected because it exceeds resource:
Query_Rows_Returned_Limit
```

このオプションが 0 の場合 (デフォルト)、制限はなくなり、出力中のロー数によってクエリが拒否されることはありません。

QUERY_TEMP_SPACE_LIMIT オプション

クエリが拒否される前のテンポラリ領域の予測最大容量を指定します。

指定できる値
整数値

デフォルト値
0 (制限なし)

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 はテンポラリの結果領域がこのオプションの値より大きいと予測されたクエリを受け取ると、次のようなメッセージとともにそのクエリを拒否します。

```
Query rejected because it exceeds total space resource limit
```

ゼロに設定すると (デフォルト)、クエリによるテンポラリ・ストアの使用に制限はありません。

ユーザは自分の環境でこのオプションを上書きし、全テンポラリ・ストアを満杯にする可能性のあるクエリを実行できます。暴走クエリによってテンポラリ・ストアがいっぱいになるのを防止するために、DBA は MAX_TEMP_SPACE_PER_CONNECTION オプションを設定できます。MAX_TEMP_SPACE_PER_CONNECTION オプションは、クエリのみではなく、すべての DML 文のためにテンポラリ・ストアの使用を監視したり、制限したりします。

分散クエリ処理トランザクションでは、Sybase IQ は、共有テンポラリ・ストアの QUERY_TEMP_SPACE_LIMIT オプションと MAX_TEMP_SPACE_PER_CONNECTION オプションの値セットを使用して、分散クエリに参加するすべてのノードによって使用される共有とローカルのテンポラリ領域の総量を制限します。これは、すべての単一クエリは、参加するノード数に関係なく、(IQ_SYSTEM_TEMP と IQ_SHARED_TEMP の DB 領域からの) テンポラリ領域総量制限を超えられないことを意味します。

たとえば、制限値が 100 で参加する 4 つのノードがテンポラリ領域にそれぞれ 25 ユニットを使用する場合、このクエリは制限内となります。ノードにより使用される領域の総量が 100 を超える場合、クエリはロールバックされます。

参照：

- MAX_TEMP_SPACE_PER_CONNECTION オプション (489 ページ)

QUERY_TIMING オプション

特定のタイミング統計を収集して、クエリ・プランに表示するかどうかを決定します。

指定できる値
ON、OFF

デフォルト値
OFF

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
このオプションは、サブクエリのタイミング統計の収集などのクエリ・エンジンの反復的な機能を制御するのに使用します。短い相関サブクエリでは、各サブクエリの実行についてタイミングを取っているとクエリの実行速度が遅くなるため、このパラメータは通常は OFF (デフォルト値) にしておきます。

クエリ・タイミングは、クエリ・プランの詳細で、一連のタイムスタンプとして表されます。これらのタイムスタンプは、クエリ演算フェーズ (Conditions、Prepare、Fetch、Complete) に対応しています。HTML と Interactive SQL クエリ・プランは、クエリのタイミングをタイムラインとして図示します。

QUOTED_IDENTIFIER オプション [TSQL]

二重引用符内の文字列の解釈を制御します。

指定できる値
ON、OFF

Open Client 接続の場合は OFF。

デフォルト値
ON

説明
QUOTED_IDENTIFIER は、二重引用符内の文字列を識別子 (ON) とリテラル文字列 (OFF) のどちらとして解釈するかを制御します。このオプションは、Transact-SQL との互換性を維持するために組み込まれています。

Sybase Central と Interactive SQL は、QUOTED_IDENTIFER が OFF に設定されている場合、一時的に ON にします。その場合、この変更を通知するメッセージが表示されます。変更は Sybase Central または Interactive SQL 接続のみで有効です。JDBC ドライバも QUOTED_IDENTIFIER を一時的に ON に設定します。

RECOVERY_TIME オプション

データベース・サーバがシステム障害からの回復にかける最長時間を分単位で設定します。

指定できる値
整数値 (分単位)

デフォルト値
2

スコープ
PUBLIC グループのみに設定できます。サーバ再起動時に有効になります。

説明
このオプションは CHECKPOINT_TIME オプションとともに使用し、チェックポイントの実行タイミングを決定します。

ヒューリスティックを使用して、直前のチェックポイントから後に実行された操作をもとにリカバリ時間を計測します。そのため、リカバリ時間は正確ではありません。

参照：
- CHECKPOINT_TIME オプション (417 ページ)

RESERVED_KEYWORDS オプション

デフォルトで無効になっている個々のキーワードをオンにします。

指定できる値
文字列

デフォルト値
空の文字列

説明
このオプションは、PUBLIC グループのみに設定できます。ユーザ設定およびテンポラリ設定には設定できません。

このオプションは、デフォルトで無効になっている個々のキーワードをオンにします。LIMIT キーワードのみをオンにできます。

例
次の文を使用すると、LIMIT キーワードがキーワードとして認識されます。

```
SET OPTION RESERVED_KEYWORDS = 'LIMIT';
```

SET、OPTION、OPTIONS キーワードはオンにできません。単語がキーワードとして認識されるかどうかは、次の優先度順で決まります。

- SQL Anywhere の予約語リストに表示されている。
- IRESERVED_KEYWORDS オプションによってオンに設定されている。
- NON_KEYWORDS オプションによってオフに設定されている。

このオプションを設定するごとに、以前の設定が置き換えられます。次の文は以前の設定内容をすべてクリアします。

```
SET OPTION RESERVED_KEYWORDS = ;
```

RETURN_DATE_TIME_AS_STRING オプション

クエリされた日付、時間、またはタイムスタンプの値をクライアントに渡す方法を制御します。

指定できる値
ON、OFF

デフォルト値
OFF

スコープ
現在の接続の継続中に、temporary オプションとしてのみ設定できます。

説明
RETURN_DATE_TIME_AS_STRING は、日付、時刻、タイムスタンプの値をアプリケーションに返すときに、date または time データ型とするか文字列とするかを指定します。

このオプションを ON に設定すると、TIMESTAMP_FORMAT、DATE_FORMAT、または TIME_FORMAT のオプションの設定を保持するために、サーバは日付、時刻、またはタイムスタンプの値を文字列に変換してからクライアントに送信します。

Sybase Central と Interactive SQL は、RETURN_DATE_TIME_AS_STRING オプションを自動的に ON にします。

参照：
- DATE_FORMAT オプション (429 ページ)
- TIME_FORMAT オプション (548 ページ)
- TIMESTAMP_FORMAT オプション (549 ページ)

ROW_COUNT オプション

クエリから返されるロー数を制限します。

指定できる値
整数値

デフォルト値
0 (返されるロー数の制限なし)

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
このランタイム・オプションにゼロでない値が設定されている場合、指定された数のローの処理が終わるとクエリ処理が停止されます。

このオプションは、キーワード **SELECT** を持つ文のみに影響を与えます。**UPDATE** 文と **DELETE** 文には影響を与えません。

SELECT 文の **FIRST** キーワードと **TOP** キーワードでも、クエリが返すローの数を制限できます。**FIRST** の使用は、ROW_COUNT データベース・オプションを 1 に設定する場合と同じです。**TOP** の使用は、ROW_COUNT をロー数と同じ数に設定する場合と同じです。**TOP** と ROW_COUNT が両方とも設定された場合、**TOP** の値が優先されます。

ROW_COUNT オプションは、グローバル変数、システム関数、またはプロキシ・テーブルを伴うクエリで使用された場合、非決定的な結果を生成する可能性があります。これらのクエリは、CIS (Component Integrated Services) を使用して部分的に実行されます。このような場合は、ROW_COUNT を設定する代わりに **SELECT TOP** *n* を使用するか、グローバル変数をローカル変数に設定して、そのローカル変数をクエリで使用します。

参照：
- QUERY_ROWS_RETURNED_LIMIT オプション (516 ページ)
- SELECT 文 (339 ページ)

SCALE オプション

カタログ・ストア上のクエリにかぎり、計算結果が最大 PRECISION にトランケートされる場合の、小数点以下の最小桁数を指定します。

指定できる値
整数値 (最大 126)

デフォルト値
38

スコープ
PUBLIC のみに設定できます。

説明
カタログ・ストア上のクエリにかぎり、計算結果が最大 PRECISION にトランケートされる場合の、小数点以下の最小桁数を指定します。

乗算、除算、加算、減算、集合関数はすべて、結果が最大精度を超える可能性があります。

参照：
- MAX_CLIENT_NUMERIC_SCALE オプション (481 ページ)
- PRECISION オプション (505 ページ)

SIGNIFICANTDIGITSFORDOUBLEEQUALITY オプション

2 つの複雑な算術式の等価性テストでは、指数表記で小数点の右側に有効桁数を指定する方法が使われます。

指定できる値
0 – 15

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
double 型のデータは 10 進数 (base 10) ではなく 2 進数 (base 2) で格納されるので、この設定では、使用されるおおよその 10 進有効桁数を指定します。0 に指定するとすべての桁が使用されます。

たとえば、SIGNIFICANTDIGITSFORDOUBLEEQUALITY を 12 に設定すると、次の 2 つの数値は等しいとみなされます。13 に設定すると、等しくないとみなされます。

- 1.23456789012345
- 1.23456789012389

SIGNIFICANTDIGITSFORDOUBLEEQUALITY は、2 つの複雑な算術式の等価性テストに影響しますが、インデックスによって行われるものには影響しません。

SORT_COLLATION オプション

ORDER BY 式に対して **SORTKEY** 関数の暗黙の使用を許可します。

指定できる値
Internal、照合名または照合 ID

デフォルト値
Internal

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
SORT_COLLATION の値が Internal である場合、**ORDER BY** 句は変更されません。

このオプションの値を、有効な照合名または照合 ID に設定すると、**ORDER BY** 句の文字列式は、**SORTKEY** 関数が呼び出されたものとして扱われます。

関数の詳細については、『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』を参照してください。

例
ソート照合をバイナリに設定します。

```
SET TEMPORARY OPTION sort_collation='binary';
```

ソート照合をバイナリに設定すると、次のクエリを変換します。

```
SELECT Name, ID
FROM Products
ORDER BY Name, ID;
SELECT Name, ID
FROM Products
ORDER BY 1, 2;
```

クエリは以下のように変換されます。

```
SELECT Name, ID
FROM Products
ORDER BY SORTKEY(Name, 'binary'), ID;
```

SORT_PINNABLE_CACHE_PERCENT オプション

現在使用可能なバッファに対してソート・オブジェクトが保持する最大パーセンテージを設定します。

指定できる値
0 – 100

デフォルト値
20

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
大規模なソートに対して大きな値を設定すると、ソートに必要なマージ処理の数を減らすことができる場合があります。一方で、大きな値を設定すると同じシステム上で実行される他のユーザのソートやハッシュ処理に影響を与える可能性があります。このオプションを変更する場合、不適切な値を設定するとかえってパフォーマンスを低下させるおそれがあるため、いくつかの設定を試したうえでパフォーマンス向上に最適な値を判断する必要があります。
SORT_PINNABLE_CACHE_PERCENT ではデフォルト値を使用することをおすすめします。

このオプションは、主に 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。SORT_PINNABLE_CACHE_PERCENT の値を変更する場合は、細心の注意が必要です。

SQL_FLAGGER_ERROR_LEVEL オプション [TSQL]

指定された標準の一部ではない SQL コードへの応答動作を制御します。

指定できる値

- OFF
- SQL:1992/Entry
- SQL:1992/Intermediate
- SQL:1992/Full
- SQL:1999/Core
- SQL:1999/Package
- SQL:2003/Core
- SQL:2003/Package

デフォルト値
OFF

説明
指定された標準の一部ではない SQL コードをエラーとして通知します。たとえば、SQL:2003/Package を指定すると、データベース・サーバは、上級 SQL/2003 構文でない構文を通知します。

以前のバージョンの Sybase IQ との互換性を確保するため、次の表に示されている値も受け入れられ、指定されているとおりにマップされます。

表 29 : SQL_FLAGGER_ERROR_LEVEL の互換値

値	アクション
E	初級レベル SQL92 構文でない構文を通知する。SQL:1992/Entry に対応する。
I	中級レベル SQL92 構文でない構文を通知する。SQL:1992/Intermediate に対応する。
F	上級レベル SQL92 構文でない構文を通知する。SQL:1992/Full に対応する。
W	サポートされている構文をすべて許可する。OFF に対応する。

SQL_FLAGGER_WARNING_LEVEL オプション [TSQL]

指定された規格の一部ではない SQL への応答を制御します。

指定できる値

- OFF

- SQL:1992/Entry
- SQL:1992/Intermediate
- SQL:1992/Full
- SQL:1999/Core
- SQL:1999/Package
- SQL:2003/Core
- SQL:2003/Package

デフォルト値
OFF

説明
警告として、指定された標準の一部ではない SQL コードをエラーとして通知します。たとえば、SQL:2003/Package を指定すると、データベース・サーバは、上級 SQL/2003 構文でない構文を通知します。

デフォルトの動作 OFF では、警告の通知がオフになります。

以前のバージョンの Sybase IQ との互換性を確保するため、次の表に示されている値も受け入れられ、指定されているとおりにマップされます。

表 30 : SQL_FLAGGER_WARNING_LEVEL の互換値

値	アクション
E	初級レベル SQL92 構文でない構文を通知する。SQL:1992/Entry に対応する。
I	中級レベル SQL92 構文でない構文を通知する。SQL:1992/Intermediate に対応する。
F	上級レベル SQL92 構文でない構文を通知する。SQL:1992/Full に対応する。
W	サポートされている構文をすべて許可する。OFF に対応する。

STRING_RTRUNCATION オプション [TSQL]

INSERT または **UPDATE** によって CHAR 文字列または VARCHAR 文字列がトランケートされるときに、エラーを通知するかどうかを指定します。

指定できる値
ON、OFF

デフォルト値
ON

説明
トランケートされた文字がスペースだけからなる場合、例外は発生しません。ON は SQL92 動作に準拠します。`STRING_RTRUNCATION` を OFF に設定すると、例外

は発生せず、文字列は何も通知されずにトランケートされます。このオプションの値が ON のときにエラーが発生すると、**ROLLBACK** が起こります。

Sybase IQ 15.0 より前では、このオプションはデフォルトで OFF になっています。下位互換性のために OFF に設定しても問題はありません。 ただし、トランケーションによってデータ・ロスが発生する可能性のある文を識別するには、ON 設定の方がよい場合があります。

SUBQUERY_CACHING_PREFERENCE オプション

相関サブクエリの述部の処理に使用するアルゴリズムを制御します。

指定できる値
-3 ～ 3

値	アクション
1	最初のサブクエリの述部にはソートベースの処理を使用する。同じ順序付けキーのない他のサブクエリ述部は、ハッシュ・テーブルを使用して、サブクエリ結果をキャッシュに入れるように処理されます。
2	許可される場合、ハッシュ・テーブルを使用して、すべてのサブクエリ述部の結果をキャッシュに入れる。使用可能なテンポラリ・キャッシュがサブクエリの結果のすべてに対応できない場合、パフォーマンスは低下します。
3	以前のサブクエリの結果の 1 つをキャッシュに入れる。**SORT** と **HASH** は使用しません。
0	オプティマイザの選択に従う。
-1	**SORT** の使用を回避する。許可される場合、IQ オプティマイザは **HASH** を選択します。
-2	**HASH** の使用を回避する。許可される場合、IQ オプティマイザは **SORT** またはキャッシュの値を選択します。
-3	キャッシュの値の使用を回避する。許可される場合、IQ オプティマイザは **HASH** または **SORT** を選択します。

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
相関サブクエリの述部では、IQ オプティマイザは、外部参照とサブクエリの実行コストを減らすサブクエリの結果をキャッシュに入れることができます。SUBQUERY_CACHING_PREFERENCE を使用すると、オプティマイザが処理量をもとに決定した使用アルゴリズムを無効にできます。クエリ・エンジンに対してアルゴリズムが適切かどうかを判断するための内部規則を無効にするものではありません。

ゼロ以外の値を設定すると、クエリ内の各サブクエリの述部に影響します。クエリ内のサブクエリの述部の 1 つを選択し、それにゼロ以外の値を使用することはできません。

『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 言語の要素」>「検索条件」>「探索条件内のサブクエリ」>「サブクエリ述部の分離」を参照してください。

SUBQUERY_CACHING_PREFERENCE は、通常、経験豊富な DBA によってのみ内部テストに使用されます。**IN** サブクエリには適用されません。

参照：
- IN_SUBQUERY_PREFERENCE オプション (464 ページ)

SUBQUERY_FLATTENING_PERCENT オプション

ユーザは、オプティマイザがスカラ・サブクエリのジョインへの変換を決定するスレッショルドを変更できます。

指定できる値
0 ～ (2^{32} -1)

値	アクション
0	オプティマイザ・コスト・モデルの決定に従う。
1 ～ (2^{32} -1)	フラット化する参照の割合。

デフォルト値
100

スコープ
このオプションは、相関スカラ・サブクエリのみに適用されます。SUBUERY_FLATTENING_PERCENT を設定するために、DBA パーミッションは必要ありません。このオプションは、任意のユーザによってどのレベルにでも設定され、すぐ有効になります。SUBUERY_FLATTENING_PERCENT をデフォルト以外の値に設定すると、クエリ内のすべてのスカラ・サブクエリ述部は影響を受け

ます。クエリ内の 1 つのスカラ・サブクエリ述部を選択し、それに対してこのオプションを使用することはできません。

説明
Sybase IQ クエリ・オプティマイザは、相関スカラ・サブクエリを等価のジョイン操作に変換して、クエリのパフォーマンスを向上させることができます。SUBQUERY_FLATTENING_PERCENT オプションでは、この最適化が実行されるスレッショルドをユーザが調整できます。

SCALAR_FLATTENING_PERCENT は、スカラ・クエリ内にある独自の予想外部値に対する内部値の割合をパーセントで表します。予想割合が 100% に近づくと、ジョインとしてのサブクエリの評価コストは、個別のインデックスによる検査よりも低くなります。予想された内部値が予想された外部値よりも少ないという保証はないため、100% より大きな値を設定できます。

参照：
- SUBQUERY_FLATTENING_PREFERENCE オプション (529 ページ)

SUBQUERY_FLATTENING_PREFERENCE オプション

スカラまたは **EXISTS** サブクエリをジョインに変換 (フラット化) する場合、ユーザはオプティマイザの決定を無効にできます。

指定できる値
-3 ～ 3

値	アクション
-3	**EXISTS** とスカラ・サブクエリのジョイン操作へのフラット化を回避する。
-2	スカラ・サブクエリのジョイン操作へのフラット化を回避する。
-1	**EXISTS** サブクエリのジョイン操作へのフラット化を回避する。
0	IQ オプティマイザによるサブクエリのフラット化の決定を許可する。
1	可能であれば、**EXIST** のフラット化のコストを無視する。
2	可能であれば、スカラのフラット化のコストを無視する。
3	**EXISTS** とスカラ・サブクエリのコストを無視する。

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。SUBQUERY_FLATTENING_PREFERENCE は、任意のユーザによってどのレベルにおいても設定できます。このオプションはすぐに有効になります。オプションをゼロ以外の値に設定すると、クエリ内のすべてのサブクエリ述部は影響を受けます。クエリ内の 1 つのサブクエリ述部を選択し、それに対してこのオプションを使用することはできません。

説明
Sybase IQ オプティマイザは、相関スカラ・サブクエリ、**EXISTS** サブクエリ、または **NOT EXISTS** サブクエリを等価のジョイン操作に変換して、クエリのパフォーマンスを向上させることができます。この最適化は、*「サブクエリのフラット化」*と呼ばれます。SUBQUERY_FLATTENING_PREFERENCE オプションでは、使用するアルゴリズムの選択時にオプティマイザのコスト決定を無効にできます。

SUBQUERY_FLATTENING_PREFERENCE をゼロ (IQ オプティマイザがサブクエリのフラット化を決定できる) に設定することは、Sybase IQ の以前のリリースで、廃止予定の FLATTEN_SUBQUERIES オプションを ON に設定することと同じです。

参照：
- SUBQUERY_FLATTENING_PERCENT オプション (528 ページ)

SUBQUERY_PLACEMENT_PREFERENCE オプション

クエリ・プラン内での、相関サブクエリの述語演算子の配置を制御します。

指定できる値
-1 ～ 1

値	アクション
-1	クエリ・プランのできるかぎり下のほうの場所を選択し、クエリ内でのサブクエリの実行をできるだけ早める。
0	オプティマイザの選択に従う。
1	クエリ・プランのできるかぎり上のほうの場所を選択し、クエリ内でのサブクエリの実行をできるだけ遅らせる。

デフォルト値
0

スコープ
すべてのスコープ、ユーザに設定でき、すぐに有効になります。

説明
IQ オプティマイザは、クエリ内の相関サブクエリの演算子を、そのクエリのプラン内のどこに配置するかを選択できます。SUBQUERY_PLACEMENT_PREFERENCE を使用すると、オプティマイザが処理量をもとに決定した配置場所を無効にできます。場所が有効かどうかを確認する内部ルールを無効にすることはありません。一部のクエリでは、有効な選択肢が 1 つしかないこともあります。0 以外の値を設定すると、クエリ内のすべての相関サブクエリの述語に影響します。クエリ内のいくつかのサブクエリから 1 つを選択してその配置場所を修正することはできません。

通常、このオプションは、内部的なテストに使用されます。また、経験豊富な DBA だけが使用してください。

このオプションでは、ほとんどの場合、デフォルトの設定が最適です。まれに、Sybase 製品の保守契約を結んでいるサポート・センタがこの値を変更するよう指示することがあります。

SUPPRESS_TDS_DEBUGGING オプション

TDS デバッグ情報をサーバ画面に表示するかどうかを決定します。

指定できる値
ON、OFF

デフォルト値
OFF

説明
サーバの起動時に **-z** オプションが指定されていれば、TDS プロトコルのデバッグ情報を含むデバッグ情報がサーバ画面に表示されます。

SUPPRESS_TDS_DEBUGGING オプションは、サーバ画面に表示される TDS 関連のデバッグ情報を制限します。このオプションが OFF (デフォルト値) に設定されている場合、TDS デバッグ情報がサーバ画面に表示されます。

SWEEPER_THREADS_PERCENT オプション

バッファ・キャッシュのクリーンアップに使用する Sybase IQ スレッドの割合を指定します。

指定できる値
1 – 40

デフォルト値
10

スコープ
PUBLIC グループのみに設定できます。このオプションを設定するには、DBA パーミッションが必要です。変更を有効にするには、データベース・サーバを停止し、再起動します。

説明
Sybase IQ は処理スレッドのうち数パーセントをスイーパ・スレッドとして使用します。 これらのスイーパ・スレッドは、メインおよびテンポラリ・バッファ・キャッシュのダーティ・ページをクリーンにします。

IQ モニタの **-cache** レポート内の GDirty カラムには、LRU バッファが "dirty" (変更済み) 状態で取り込まれた回数が表示されます。GDirty が一定時間 0 より大きい数であれば、SWEEPER_THREADS_PERCENT または WASH_AREA_BUFFERS_PERCENT の値を大きくする必要がある場合があります。

このオプションでは、ほとんどの場合、デフォルトの設定が最適です。まれに、Sybase 製品の保守契約を結んでいるサポート・センタがこの値を大きくするよう指示することがあります。

参照：
- WASH_AREA_BUFFERS_PERCENT オプション (554 ページ)

TDS_EMPTY_STRING_IS_NULL オプション [database]

TDS 接続で、空の文字列を NULL として返すか、ブランク文字を 1 つ含む文字列として返すかを制御します。

指定できる値
ON, OFF

デフォルト値
OFF

説明
デフォルトでは、TDS_EMPTY_STRING_IS_NULL は OFF に設定され、空の文字列はブランク文字を 1 つ含む文字列として TDS 接続に返されます。このオプションが ON の場合、空の文字列は NULL 文字列として TDS 接続に返されます。非 TDS 接続では、空の文字列と NULL 文字列が区別されます。

TEMP_EXTRACT_APPEND オプション

データ抽出機能で抽出されたローを、出力ファイルの末尾に追加するよう指定します。

指定できる値
ON、OFF

デフォルト値
OFF

スコープ
個別の接続に設定可能です。すぐに有効になります。

説明
このオプションは、データ抽出機能で抽出されたローを、出力ファイルの末尾に追加するよう指定します。 書き込み／実行パーミッションのあるディレクトリに出力ファイルを作成し、Sybase IQ の起動に使用したユーザ名 (**sybase** など) にそのディレクトリと出力ファイルの書き込みパーミッションを設定します。必要に応じて、出力ファイルのパーミッションを他のユーザに与えることもできます。出力ファイルの名前は、TEMP_EXTRACT_NAME1 オプションで指定します。ファイルが存在しない場合、データ抽出機能によって出力ファイルが作成されます。

TEMP_EXTRACT_APPEND は、TEMP_EXTRACT_SIZEn オプションと互換性がありません。抽出追加出力ファイルのサイズを制限しようとすると、Sybase IQ からエラーが表示されます。

データ抽出機能と抽出オプションの利用の詳細については、『システム管理ガイド：第1巻』の「データのインポートとエクスポート」>「データベースからデータをエクスポートする方法」>「データ抽出機能」を参照してください。

参照：
- TEMP_EXTRACT_NAMEn オプション (537 ページ)

TEMP_EXTRACT_BINARY オプション

TEMP_EXTRACT_SWAP オプションと組み合わせて、データ抽出機能で実行する抽出の種類を指定します。

指定できる値
ON、OFF

デフォルト値
OFF

スコープ
個別の接続に設定可能です。すぐに有効になります。

説明
このオプションは、TEMP_EXTRACT_SWAP オプションとともに、データ抽出機能で実行する抽出の種類を指定します。

表 31 : 抽出の種類に対する抽出オプション

抽出の種類	TEMP_EXTRACT_BINARY	TEMP_EXTRACT_SWAP
バイナリ	ON	OFF
バイナリ／スワップ	ON	ON
ASCII	OFF	OFF

抽出の種類のデフォルトは ASCII です。

データ抽出機能と抽出オプションの利用の詳細については、『システム管理ガイド：第1巻』の「データのインポートとエクスポート」>「データベースからデータをエクスポートする方法」>「データ抽出機能」を参照してください。

参照：
- TEMP_EXTRACT_SWAP オプション (545 ページ)

TEMP_EXTRACT_COLUMN_DELIMITER オプション

ASCII 抽出のデータ抽出機能の出力で使用するカラムのデリミタを指定します。

指定できる値
文字列

デフォルト値
','

スコープ
個別の接続に設定可能です。すぐに有効になります。

説明
TEMP_EXTRACT_COLUMN_DELIMITERを使用して、データ抽出機能の出力で使用する、カラム間のデリミタを指定します。ASCII 抽出の場合、デフォルトではカンマでカラムの値を区切ります。文字列は、デフォルトでは引用符がついていません。

デリミタには 1 バイトから 4 バイト必要で、マルチバイト照合順を使用している場合はその照合順でデリミタが有効である必要があります。デリミタとして使用する文字列は、データ出力文字列内で使われないものにしてください。

ASCII 抽出で、このオプションが空の文字列 " に設定されている場合は、抽出されたデータはカラム・デリミタなしの固定幅の ASCII で書き込まれます。数値とバイナリ・データ型は、*n* ブランクのフィールドに右揃えされます。ここで *n* は、その型の値を表すのに必要な最大バイト数を表します。文字データ型は、*n* ブランクのフィールドで左揃えされます。

注意： 固定幅の ASCII 抽出での最小のカラム幅は、NULL 値の場合に "NULL" という文字列を格納できるよう、4 バイトです。たとえば、抽出されたカラムが CHAR(2) で、TEMP_EXTRACT_COLUMN_DELIMITER が空の文字列 " に設定されている場合、抽出されたデータの後に 2 つのスペースが付きます。

データ抽出機能と抽出オプションの利用の詳細については、『システム管理ガイド：第 1 巻』の「データのインポートとエクスポート」>「データベースからデータをエクスポートする方法」>「データ抽出機能」を参照してください。

参照：

- TEMP_EXTRACT_QUOTE オプション (541 ページ)
- TEMP_EXTRACT_QUOTES オプション (542 ページ)
- TEMP_EXTRACT_QUOTES_ALL オプション (542 ページ)
- TEMP_EXTRACT_ROW_DELIMITER オプション (543 ページ)

TEMP_EXTRACT_DIRECTORY オプション

ユーザがデータ抽出機能を利用できるかどうかを制御します。一時抽出ファイルが配置されたディレクトリを制御し、TEMP_EXTRACT_NAMEn オプションで指定されたディレクトリ・パスを上書きします。

指定できる値
文字列

デフォルト値
" (空の文字列)

スコープ
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。このオプションを設定するには、DBA パーミッションが必要です。このオプションはすぐに有効になります。

説明
ユーザの TEMP_EXTRACT_DIRECTORY オプションが FORBIDDEN (大文字と小文字を区別しない) という文字列に設定されている場合、そのユーザはデータ抽出を実行できません。このユーザがデータ抽出機能を使用しようとすると次のようなエラーになります。`You do not have permission to perform Extracts.`

PUBLIC グループの TEMP_EXTRACT_DIRECTORY が FORBIDDEN に設定されている場合は、どのユーザもデータ抽出を実行できません。

TEMP_EXTRACT_DIRECTORY が有効なディレクトリ・パスに設定されている場合、一時抽出ファイルはそのディレクトリに配置され、TEMP_EXTRACT_NAMEn オプションで指定されたパスは上書きされます。

TEMP_EXTRACT_DIRECTORY が無効なディレクトリ・パスに設定されている場合は、次のようなエラーが発生します。`Files does not exist File: <invalid path>`

TEMP_EXTRACT_DIRECTORY が空白の場合、一時抽出ファイルは、TEMP_EXTRACT_NAMEn で指定されたディレクトリに配置されます。TEMP_EXTRACT_NAMEn でパスが指定されていない場合、抽出ファイルはデフォルトでサーバ起動ディレクトリに配置されます。

このオプションは、大きなデータ抽出ファイルの作成を、ユーザが書き込みアクセスを持つディレクトリに制限することで、より充実したセキュリティを実現し、ディスク管理を制御しやすくします。

データ抽出機能と抽出オプションの利用の詳細については、『システム管理ガイド：第1巻』の「データのインポートとエクスポート」>「データベースからデータをエクスポートする方法」>「データ抽出機能」を参照してください。

参照：
- TEMP_EXTRACT_NAMEn オプション (537 ページ)

TEMP_EXTRACT_ESCAPE_QUOTES オプション

ASCII 抽出のデータ抽出機能の出力で、引用符が含まれたフィールドのすべての引用符をエスケープするかどうかを指定します。

指定できる値
ON、OFF

デフォルト値
OFF

スコープ
個別の接続に設定可能です。すぐに有効になります。

説明
TEMP_EXTRACT_QUOTE がデフォルトのままか、値 "" (二重引用符) に設定されており、**TEMP_EXTRACT_BINARY** が OFF であり、**TEMP_EXTRACT_QUOTES** または **TEMP_EXTRACT_QUOTES_ALL** が ON である場合を除き、このオプションは無視されます。

参照：
- TEMP_EXTRACT_BINARY オプション (533 ページ)
- TEMP_EXTRACT_QUOTES オプション (542 ページ)
- TEMP_EXTRACT_QUOTES_ALL オプション (542 ページ)

TEMP_EXTRACT_NAMEn オプション

データ抽出機能で使用される出力ファイルまたは名前付きパイプの名前を指定します。オプションは TEMP_EXTRACT_NAME1 から TEMP_EXTRACT_NAME8 の 8 つです。

指定できる値
文字列

デフォルト値
" (空の文字列)

スコープ
個別の接続に設定可能です。すぐに有効になります。

説明
TEMP_EXTRACT_NAME1 から TEMP_EXTRACT_NAME8 は、抽出機能のデータに使用する出力ファイルの名前を指定します。これらのオプションは順番に使用する必要があります。たとえば、TEMP_EXTRACT_NAME1 と TEMP_EXTRACT_NAME2 の両方のオプションがすでに設定されていないかぎり、TEMP_EXTRACT_NAME3 を設定しても無効となります。

この中で最も重要なオプションは TEMP_EXTRACT_NAME1 です。
TEMP_EXTRACT_NAME1 がデフォルト値 (空の文字列 ") に設定されている場合、抽出処理は使用できず、リダイレクトされる出力もありません。抽出処理を使用可能にするには、TEMP_EXTRACT_NAME1 にパス名を設定してください。すると、そのパス名のファイルに対して抽出が始まります。パス名には、他で使用されて

いないファイルを選択してください。TEMP_EXTRACT_NAME1 オプションを TEMPORARY として設定することをおすすめします。

TEMP_EXTRACT_NAME1 は、TEMP_EXTRACT_APPEND オプションが ON に設定されている場合には、出力ファイル名の指定にも使用されます。 この場合、**SELECT** 文を実行する前に、Sybase IQ の起動に使用されたユーザ名 (**sybase** など) に対し、指定されたファイル、またはこのファイルを含むフォルダの書き込みパーミッションを設定します。 追加モードの場合は、データ抽出機能は、抽出されたローをファイルの末尾に追加します。ファイルに既に書き込まれているデータへは上書きしません。 出力ファイルが存在しない場合、データ抽出機能によってファイルが作成されます。

警告！ 既存のファイルのパス名を指定し、TEMP_EXTRACT_APPEND オプションが OFF (デフォルト) に設定されている場合は、ファイルの内容が上書きされます。これは、ウィークリー・レポートなどのファイルであれば問題ありませんが、データベース・ファイルに対して行ってはなりません。

TEMP_EXTRACT_NAME2 から TEMP_EXTRACT_NAME8 までのオプションは、TEMP_EXTRACT_NAME1 に追加して、複数の出力ファイルを指定するために使用できます。

抽出先のディスク・ファイルまたは名前付きパイプが 1 つだけの場合は、TEMP_EXTRACT_NAME2 から TEMP_EXTRACT_NAME8 と TEMP_EXTRACT_SIZE1 と TEMP_EXTRACT_SIZE8 の各オプションはデフォルト値のままにします。

TEMP_EXTRACT_NAME1 が設定されているとき、次の操作は実行できません。

- ジョインのトップ・テーブルに対する **LOAD**、**DELETE**、**INSERT**、または **INSERT...LOCATION**
- **SYNCHRONIZE JOIN INDEX** (明示的に発行した場合、または **CREATE JOIN INDEX** の処理過程で実行された場合)
- **INSERT...SELECT**

さらに、データ抽出機能の次の制限に留意してください。

- 抽出は、IQ ストアに格納されたデータに対してのみ実行できます。
- 抽出は、システム・テーブルやデータベース間ジョインに対しては実行できません。
- 抽出は、ユーザ定義関数やシステム関数を使用するクエリに対しては実行できません。ただし、システム関数の **suser_id()** と **suser_name()** は例外です。
- Interactive SQL を **-q** (クワイエット・モード) オプション付きで実行し、データ抽出コマンドがコマンド・ファイル内にある場合、まず Interactive SQL オプション [複数の結果セットを表示] を設定し、これを永続的なものにする必要があります。このオプションが設定されていない場合、出力ファイルは作成されません。

[複数の結果セットを表示] オプションを設定するには、Interactive SQL ウィンドウの [ツール] - [オプション] を選択し、[複数の結果セットを表示] ボックスをクリックして [永続化する] をクリックします。

TEMP_EXTRACT_NAMEn オプションを使用して指定されたディレクトリ・パスは、TEMP_EXTRACT_DIRECTORY オプションを使用して上書きできます。

データ抽出機能と抽出オプションの利用の詳細については、『システム管理ガイド：第1巻』の「データのインポートとエクスポート」>「データベースからデータをエクスポートする方法」>「データ抽出機能」を参照してください。

参照：

- TEMP_EXTRACT_APPEND オプション (533 ページ)
- TEMP_EXTRACT_DIRECTORY オプション (535 ページ)
- TEMP_EXTRACT_SIZEn オプション (544 ページ)

TEMP_EXTRACT_NULL_AS_EMPTY オプション

ASCII 抽出のデータ抽出機能の出力で使用する null 値の表示方法を指定します。

指定できる値
ON、OFF

デフォルト値
OFF

スコープ
個別の接続に設定可能です。すぐに有効になります。

説明
TEMP_EXTRACT_NULL_AS_EMPTY は、ASCII 抽出のデータ抽出機能の出力で使用する null 値の表示方法を制御します。TEMP_EXTRACT_NULL_AS_EMPTY オプションが ON に設定されている場合は、すべてのデータ型に対して、null 値は '' (空の文字列) で表現されます。

上記の引用符は、抽出出力ファイルには含まれないので注意してください。
TEMP_EXTRACT_NULL_AS_EMPTY オプションを OFF にすると、null 値の表示に常に文字列 'NULL' が使用されます。デフォルトは OFF です。

データ抽出機能と抽出オプションの利用の詳細については、『システム管理ガイド：第1巻』の「データのインポートとエクスポート」>「データベースからデータをエクスポートする方法」>「データ抽出機能」を参照してください。

TEMP_EXTRACT_NULL_AS_ZERO オプション

ASCII 抽出のデータ抽出機能の出力で使用する null 値の表示方法を指定します。

指定できる値
ON、OFF

デフォルト値
OFF

スコープ
個別の接続に設定可能です。すぐに有効になります。

説明
TEMP_EXTRACT_NULL_AS_ZERO は、ASCII 抽出のデータ抽出機能の出力で使用する null 値の表示方法を制御します。TEMP_EXTRACT_NULL_AS_ZERO が ON に設定されているとき、null 値は次のように表示されます。

- 算術型には '0'
- CHAR と VARCHAR の文字型には " (空の文字列)
- 日付には " (空文字列)
- 時刻には " (空文字列)
- タイムスタンプには " (空文字列)

上記の引用符は、抽出出力ファイルには含まれないので注意してください。
TEMP_EXTRACT_NULL_AS_ZERO オプションを OFF にすると、null 値の表示に常に文字列 'NULL' が使用されます。デフォルトは OFF です。

注意： Sybase IQ 12.5 では、テーブル内の CHAR カラムまたは VARCHAR カラムからの ASCII 抽出を行うと、出力ファイルに少なくとも 4 文字が返されます。Sybase IQ は、null 値が含まれるカラム内のローに NULL という単語を書き込む必要があるため、これは、TEMP_EXTRACT_NULL_AS_ZERO が OFF に設定されている場合に必要です。 TEMP_EXTRACT_NULL_AS_ZERO が ON の場合は、4 文字分の領域を予約する必要はありません。

Sybase IQ 12.6 以降では、TEMP_EXTRACT_NULL_AS_ZERO が ON に設定されている場合は、ASCII 抽出で CHAR または VARCHAR カラムに対してファイルに書き込まれる文字数は、そのカラムの文字数に等しくなります。4 文字未満のカラムの場合でも同じです。

データ抽出機能と抽出オプションの利用の詳細については、『システム管理ガイド：第 1 巻』の「データのインポートとエクスポート」>「データベースからデータをエクスポートする方法」>「データ抽出機能」を参照してください。

TEMP_EXTRACT_QUOTE オプション

TEMP_EXTRACT_QUOTES オプションまたは TEMP_EXTRACT_QUOTES_ALL オプションが ON に設定されているときに、ASCII 抽出のデータ抽出機能の出力のフィールドを囲むための引用符として使用する文字列を指定します。

指定できる値
文字列

デフォルト値
'' (空の文字列)

スコープ
個別の接続に設定可能です。すぐに有効になります。

説明
このオプションは、デフォルト値が適切でない場合に、ASCII 抽出のデータ抽出機能の出力のフィールドを囲むための引用符として使用する文字列を指定します。TEMP_EXTRACT_QUOTE は、TEMP_EXTRACT_QUOTES オプションと TEMP_EXTRACT_QUOTES_ALL オプションと併せて使用されます。

TEMP_EXTRACT_QUOTE オプションで指定する引用符文字列には、ローとカラムのデリミタと同じ制限があります。 このオプションのデフォルトは空の文字列で、Sybase IQ はこれを一重引用符に変換します。

TEMP_EXTRACT_QUOTE オプションに指定された文字列には 1 ～ 4 バイト必要で、マルチバイト照合順を使用している場合はその照合順でデリミタが有効でなければなりません。使用する文字列は、データ出力文字列内で使われないものにしてください。

データ抽出機能と抽出オプションの利用の詳細については、『システム管理ガイド：第1巻』の「データのインポートとエクスポート」>「データベースからデータをエクスポートする方法」>「データ抽出機能」を参照してください。

参照：

- TEMP_EXTRACT_COLUMN_DELIMITER オプション (534 ページ)
- TEMP_EXTRACT_QUOTES オプション (542 ページ)
- TEMP_EXTRACT_QUOTES_ALL オプション (542 ページ)
- TEMP_EXTRACT_ROW_DELIMITER オプション (543 ページ)

TEMP_EXTRACT_QUOTES オプション

ASCII 抽出のデータ抽出機能の出力で、文字列フィールドを引用符で囲むよう指定します。

指定できる値
ON、OFF

デフォルト値
OFF

スコープ
個別の接続に設定可能です。すぐに有効になります。

説明
このオプションは、ASCII 抽出のデータ抽出機能の出力で、文字列フィールドを引用符で囲むよう指定します。デフォルト値が適切でない場合、引用符として使用する文字列は TEMP_EXTRACT_QUOTE オプションで指定されます。

データ抽出機能と抽出オプションの利用の詳細については、『システム管理ガイド：第1巻』の「データのインポートとエクスポート」>「データベースからデータをエクスポートする方法」>「データ抽出機能」を参照してください。

参照：
- TEMP_EXTRACT_COLUMN_DELIMITER オプション (534 ページ)
- TEMP_EXTRACT_QUOTES_ALL オプション (542 ページ)
- TEMP_EXTRACT_ROW_DELIMITER オプション (543 ページ)

TEMP_EXTRACT_QUOTES_ALL オプション

ASCII 抽出のデータ抽出機能の出力で、すべてのフィールドを引用符で囲むよう指定します。

指定できる値
ON、OFF

デフォルト値
OFF

スコープ
個別の接続に設定可能です。すぐに有効になります。

説明
TEMP_EXTRACT_QUOTES_ALL は、ASCII 抽出のデータ抽出機能の出力で、すべてのフィールドを引用符で囲むよう指定します。デフォルト値が適切でない場合、引用符として使用する文字列は TEMP_EXTRACT_QUOTE オプションで指定されます。

データ抽出機能と抽出オプションの利用の詳細については、『システム管理ガイド：第1巻』の「データのインポートとエクスポート」>「データベースからデータをエクスポートする方法」>「データ抽出機能」を参照してください。

参照：
- TEMP_EXTRACT_COLUMN_DELIMITER オプション (534 ページ)
- TEMP_EXTRACT_QUOTES オプション (542 ページ)
- TEMP_EXTRACT_QUOTES_ALL オプション (542 ページ)
- TEMP_EXTRACT_ROW_DELIMITER オプション (543 ページ)

TEMP_EXTRACT_ROW_DELIMITER オプション

ASCII 抽出のデータ抽出機能の出力で使用するローのデリミタを指定します。

指定できる値
文字列

デフォルト値
" (空の文字列)

スコープ
個別の接続に設定可能です。すぐに有効になります。

説明
TEMP_EXTRACT_ROW_DELIMITER を使用して、データ抽出機能の出力で使用する、ロー間のデリミタを指定します。ASCII 抽出では、デフォルトでは、UNIX プラットフォームでは改行、Windows プラットフォームでは復帰／改行の組み合わせでローが終了します。

デリミタには 1 バイトから 4 バイト必要で、マルチバイト照合順を使用している場合はその照合順でデリミタが有効である必要があります。 デリミタとして使用する文字列は、データ出力文字列内で使われないものにしてください。
TEMP_EXTRACT_ROW_DELIMITER オプションのデフォルトは空の文字列です。Sybase IQ は、このオプションのデフォルトの空の文字列を、UNIX プラットフォームでは改行文字、Windows プラットフォームでは復帰／改行文字に変換します。

データ抽出機能と抽出オプションの利用の詳細については、『システム管理ガイド：第1巻』の「データのインポートとエクスポート」>「データベースからデータをエクスポートする方法」>「データ抽出機能」を参照してください。

参照：
- TEMP_EXTRACT_COLUMN_DELIMITER オプション (534 ページ)
- TEMP_EXTRACT_QUOTES オプション (542 ページ)
- TEMP_EXTRACT_QUOTES_ALL オプション (542 ページ)

TEMP_EXTRACT_SIZEn オプション

データ抽出機能で使用される、対応する出力ファイルの最大サイズを指定します。

オプションは TEMP_EXTRACT_SIZE1 から TEMP_EXTRACT_SIZE8 の 8 つです。

デフォルト値
0

スコープ
個別の接続に設定可能です。すぐに有効になります。

説明
TEMP_EXTRACT_SIZE1 から TEMP_EXTRACT_SIZE8 は、データ抽出機能で使用される、対応する出力ファイルの最大サイズを指定します。
TEMP_EXTRACT_SIZE1 は TEMP_EXTRACT_NAME1 で指定された出力ファイルの最大サイズ、TEMP_EXTRACT_SIZE2 は TEMP_EXTRACT_NAME2 で指定された出力ファイルの最大サイズ、というように指定します。

なお、データ抽出サイズのオプションのデフォルト値は 0 です。Sybase IQ はこのデフォルト値を、次の値に変換します。

デバイス・タイプ	サイズ
ディスク・ファイル	AIX と HP-UX：0 ～ 64GB Sun Solaris と Linux：0 ～ 512GB Windows の場合：0 ～ 128GB
テープ*	524288KB (0.5GB)
その他	9007199254740992KB (8192 ペタバイト、「無制限」)

*テープ・デバイスは現時点ではサポートされていません。

JFS2 など、大規模なファイル・システムでデフォルト値よりも大きなファイル・サイズをサポートしている場合、TEMP_EXTRACT_SIZEn をファイル・システム

が許可する値に設定します。たとえば、ITB をサポートするには、次のように設定します。

```
TEMP_EXTRACT_SIZE1 = 1073741824 KB
```

抽出先のディスク・ファイルまたは名前付きパイプが 1 つだけの場合は、TEMP_EXTRACT_NAME2 から TEMP_EXTRACT_NAME8 と TEMP_EXTRACT_SIZE1 と TEMP_EXTRACT_SIZE8 の各オプションはデフォルト値のままにします。

TEMP_EXTRACT_SIZE*n* は、TEMP_EXTRACT_APPEND オプションと互換性がありません。抽出追加出力ファイルのサイズを制限しようとすると、Sybase IQ からエラーが表示されます。

データ抽出機能と抽出オプションの利用の詳細については、『システム管理ガイド：第1巻』の「データのインポートとエクスポート」>「データベースからデータをエクスポートする方法」>「データ抽出機能」を参照してください。

参照：

- TEMP_EXTRACT_NAMEn オプション (537 ページ)

TEMP_EXTRACT_SWAP オプション

TEMP_EXTRACT_BINARY オプションと組み合わせて、データ抽出機能で実行する抽出の種類を指定します。

指定できる値
ON、OFF

デフォルト値
OFF

スコープ
個別の接続に設定可能です。すぐに有効になります。

説明
このオプションは、TEMP_EXTRACT_BINARY オプションとともに、データ抽出機能で実行する抽出の種類を指定します。

表 32 : 抽出の種類に対する抽出オプション

抽出の種類	TEMP_EXTRACT_BINARY	TEMP_EXTRACT_SWAP
バイナリ	ON	OFF
バイナリ／スワップ	ON	ON

抽出の種類	TEMP_EXTRACT_BINARY	TEMP_EXTRACT_SWAP
ASCII	OFF	OFF

抽出の種類のデフォルトは ASCII です。

データ抽出機能と抽出オプションの利用の詳細については、『システム管理ガイド：第1巻』の「データのインポートとエクスポート」>「データベースからデータをエクスポートする方法」>「データ抽出機能」を参照してください。

参照：

- TEMP_EXTRACT_BINARY オプション (533 ページ)

TEMP_RESERVED_DBSPACE_MB オプション

Sybase IQ がテンポラリ IQ ストアに予約する領域の量を制御します。

指定できる値
200 またはそれ以上の整数 (メガバイト)

デフォルト値
200。Sybase IQ は、`IQ_SYSTEM_TEMP` の最新読み書き可能ファイルの最大で 50%、最小で 1% を予約します。

スコープ
`PUBLIC` グループのみに設定できます。このオプションを設定するには、DBA パーミッションが必要です。すぐに有効になります。予約領域サイズを変更するためにサーバを再起動する必要はありません。

説明
`TEMP_RESERVED_DBSPACE_MB` では、セーブポイントの解放、コミット、およびチェックポイント操作で使用される、小さいが重要なデータ構造体のために、テンポラリ IQ ストア内で予約される領域の量を制御します。 運用データベースでは、この値を 200MB から 1GB の間で設定します。 IQ ページ・サイズや同時接続数が大きくなればなるほど、より多くの領域を予約する必要があります。

予約領域サイズは、`IQ_SYSTEM_TEMP` の最新読み書き可能ファイルの 1 ～ 50% で計算されます。

TEMP_SPACE_LIMIT_CHECK オプション

カタログ・ストアのテンポラリ領域を、接続ごとにチェックします。

指定できる値
ON または OFF (制限チェックなし)

デフォルト値
ON

スコープ
PUBLIC グループのみに設定できます。DBA 権限が必要です。

説明
TEMP_SPACE_LIMIT_CHECK が ON の場合、データベース・サーバは、接続が使用するカタログ・ストアのテンポラリ・ファイル領域の容量をチェックします。このオプションが OFF に設定されている場合に接続がクオータ以上のテンポラリ・ファイル領域を要求すると、致命的なエラーが発生することがあります。このオプションが ON に設定されている場合に接続がクオータ以上のテンポラリ・ファイル領域を要求すると、要求は失敗し、"Temporary space limit exceeded" というエラーが返されます。

接続のテンポラリ・ファイルのクオータを決定するには、テンポラリ・ファイルの最大サイズ、そしてアクティブなデータベース接続の 2 つの要素が使用されます。テンポラリ・ファイルの最大サイズは、ファイルの現在のサイズと、そのファイルが含まれるパーティションの空き領域を合計したサイズです。制限チェックがオンになっているとき、テンポラリ・ファイルが最大サイズの 80% を超えている場合に接続リクエストがテンポラリ・ファイル領域をリクエストすると、サーバは接続がクオータをオーバーしていないかどうかチェックします。このとき、最大テンポラリ・ファイル領域をアクティブな接続数で割ったサイズ以上の領域を使用している接続はすべて失敗します。

注意： このオプションは、IQ テンポラリ・ストア領域とは無関係です。IQ テンポラリ領域の増大を制限するには、QUERY_TEMP_SPACE_LIMIT オプションと MAX_TEMP_SPACE_PER_CONNECTION オプションを参照してください。

テンポラリ・ファイルに使用できる領域に関する情報は、**sa_disk_free_space system** プロシージャを使用すると取得できます。 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_disk_free_space システムプロシージャー」を参照してください。

注意： このリファレンスは SQL Anywhere マニュアルにリンクされています。

例
100MB の空き領域を持ち、他にアクティブなファイルがないドライブ上で、テンポラリ・ファイルと共にデータベースは起動されます。つまり、利用可能なテンポラリ・ファイル領域は 100MB です。DBA は次の文を発行できます。

```
SET OPTION PUBLIC.TEMP_SPACE_LIMIT_CHECK = 'ON'
```

テンポラリ・ファイル領域が 80MB に満たない場合、サーバは以前と変わらず動作します。80MB に達すると、新しい動作が発生する可能性があります。10 個のクエリが実行中で、テンポラリ・ファイルのサイズを増やす必要がある場合を考えてみましょう。あるクエリが、テンポラリ・ファイル領域のうち 8MB 以上を使用していることをサーバが検出すると、そのクエリは失敗します。

TEXT_DELETE_METHOD オプション

TEXT インデックスの削除処理で使用されるアルゴリズムを指定します。

TEXT インデックスを使用するには、非構造化データ分析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。 TEXT_DELETE_METHOD の構文と詳細な説明については、『Sybase IQ の非構造化データ分析の概要』を参照してください。

TIME_FORMAT オプション

データベースから取り出した時刻に対して使用するフォーマットを設定します。

指定できる値
HH、NN、MM、SS を使用してコロンで区切って作られた文字列

デフォルト値
'HH:NN:SS.SSS'

Open Client および JDBC 接続の場合のデフォルトも、HH:NN:SS.SSS に設定されます。

説明
フォーマットは次の記号を組み合わせた文字列です。

- hh – 2 桁の時間 (24 時間表示)
- nn – 2 桁の分
- mm – コロンの後の場合は、2 桁の分 (hh:mm など)
- ss[.s...s] – 2 桁の秒と、オプションで小数

各記号は、フォーマットされた日付の対応するデータで置き換えられます。数字の出力ではなく文字を表すフォーマット記号は大文字で入力でき、その場合、置き換えられる文字も大文字になります。数字の場合、フォーマット文字列に大文字と小文字を混在させると、出力に先行ゼロが付きません。

フォーマット文字列の中でマルチバイト文字は使用できません。データベースの照合順が 932JPN などのマルチバイト照合順であっても、使用できるのは 1 バイト文字だけです。

参照：
- DATE_FORMAT オプション (429 ページ)

- RETURN_DATE_TIME_AS_STRING オプション (520 ページ)

TIMESTAMP_FORMAT オプション

データベースから取り出したタイプスタンプに対して使用するフォーマットを設定します。

指定できる値
下記の記号を組み合わせた文字列

デフォルト値
'YYYY-MM-DD HH:NN:SS.SSS'

説明
フォーマットは次の記号を組み合わせた文字列です。

表 33 : TIMESTAMP_FORMAT の文字列記号

記号	説明
yy	2 桁の年。
yyyy	4 桁の年。
mm	2 桁の月、またはコロンの後の場合 (hh:mm など) は 2 桁の分。
mmm	月の名前を示す 3 文字。
mmmm[m...]	月を示す長い文字列—m の数の文字を使用する。指定された m の数の文字で月を示す。ただし、m の数が、月の名前を超えない範囲に限定される。
dd	2 桁の日。
ddd	曜日の名前を示す 3 文字。
dddd[d...]	曜日を示す長い文字列—d の数の文字を使用する。指定された d の数の文字で曜日を示す。ただし、d の数が、曜日の名前を超えない範囲に限定される。
hh	2 桁の時間。
nn	2 桁の分。
ss.SSS	秒 (ss) と小数 (SSS)。最大で小数点第 6 位まで。小数点以下 6 桁のタイムスタンプをサポートしないプラットフォームもある。
aa	必要であれば a.m. または p.m. (12 時間表記)。

記号	説明
pp	必要であれば p.m. (12 時間表記)。

各記号は、フォーマットされた日付の対応するデータで置き換えられます。数字の出力ではなく文字を表すフォーマット記号は大文字で入力でき、その場合、置き換えられる文字も大文字になります。数字の場合、フォーマット文字列に大文字と小文字を混在させると、出力に先行ゼロが付きません。

フォーマット文字列の中でマルチバイト文字は使用できません。データベースの照合順が 932JPN などのマルチバイト照合順であっても、使用できるのは 1 バイト文字だけです。

参照：

- DATE_FORMAT オプション (429 ページ)
- RETURN_DATE_TIME_AS_STRING オプション (520 ページ)

TOP_NSORT_CUTOFF_PAGES オプション

TOP N アルゴリズムを選択した場合の結果サイズのスレッショルドを設定します。

指定できる値
1 – 1000

デフォルト値
1

説明
TOP_NSORT_CUTOFF_PAGES オプションは、このスレッショルドをページ数で設定します。このとき、**TOP** 句と **ORDER BY** 句の両方を含んでいるクエリを評価すると、順序リストに基づく処理からソートベースの処理にアルゴリズムが切り替わります。順序リストに基づく処理のパフォーマンスが向上するのは、**TOP N** の値が結果ローの数より小さい場合です。ソートベースの処理のパフォーマンスが向上するのは、**TOP N** の値を大きくした場合です。

また、TOP_NSORT_CUTOFF_PAGES の値を大きくして、ソートベースの処理を防ぐことによって、パフォーマンスを向上できる場合もあります。

参照：

- SELECT 文 (339 ページ)

TRIM_PARTIAL_MBC オプション

部分的なマルチバイト文字データの自動削除を有効にします。

指定できる値
ON、OFF

デフォルト値
OFF

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。PUBLIC グループに設定できます。すぐに有効になります。

説明
シングルバイトとマルチバイト文字の両方を含む照合で、一貫したデータ・ロードを実現します。TRIM_PARTIAL_MBC が ON の場合、次のように処理されます。

- 部分的なマルチバイト文字は、CHAR カラムにロードされるときにブランクで置き換えられます。
- 部分的なマルチバイト文字は、VARCHAR カラムにロードされるときにトランケートされます。

TRIM_PARTIAL_MBC が OFF のときは、通常の CONVERSION_ERROR セマンティックが適用されます。

参照：
- CONVERSION_ERROR オプション [TSQL] (419 ページ)

TSQL_VARIABLES オプション [TSQL]

@ 符号を、Embedded SQL ホスト変数名のプレフィクスとして使用できるかどうかを制御します。

指定できる値
ON、OFF

Open Client および JDBC 接続の場合は ON

デフォルト値
OFF

説明
TSQL_VARIABLES を ON に設定すると、Embedded SQL のホスト変数名のプレフィクスとしてコロンの代わりに @ 符号を使用できます。これは、主に Open Server Gateway との互換性を保つために実装されています。

USER_RESOURCE_RESERVATION オプション

現在のユーザ数に対するメモリの使用を調整します。

指定できる値
整数

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。

デフォルト値
1

説明
Sybase IQ は、開いたカーソルの数を追跡して、メモリを割り付けます。 特定の状況においては、このオプションを使用することにより、製品を現在使用していると Sybase IQ が判断している現在のカーソル数の最小値を調整し、テンポラリ・キャッシュから割り付けるメモリをさらに節約できます。

このオプションは、慎重な分析の結果、実際に必要であると判断された場合にのみ設定する必要があります。このオプションを設定する場合は、Sybase 製品の保守契約を結んでいるサポート・センタにお問い合わせください。

VERIFY_PASSWORD_FUNCTION オプション

パスワード・ルールの実装に使用できる、ユーザ指定の認証の関数を指定します。

指定できる値
文字列

スコープ
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。このオプションを設定するには、DBA パーミッションが必要です。このオプションはすぐに有効になります。

デフォルト値
" (空の文字列)。(**GRANT CONNECT** で呼び出される関数はありません。)

説明
VERIFY_PASSWORD_FUNCTION オプション値が有効な文字列に設定されている場合、**GRANT CONNECT TO** *userid* **IDENTIFIED BY** *password* 文によって、オプション値で指定された関数が呼び出されます。

ユーザが関数を上書きするのを防ぐために、オプション値は、*owner.function_name* の形式で指定する必要があります。

関数は、次の 2 つのパラメータを取ります。

- *user_name* VARCHAR(128)
- *new_pwd* VARCHAR(255)

VARCHAR(255) 型の値を返します。

注意： ユーザがプロシージャ・デバッガを使用してステップ実行できないように、関数に対して **ALTER FUNCTION** *function-name* **SET HIDDEN** を実行します。

VERIFY_PASSWORD_FUNCTION オプションが設定されている場合、**GRANT CONNECT** 文に複数のユーザ ID とパスワードを指定することはできません。

例
次の例では、この文でユーザ名とは異なるパスワードを要求する関数を作成します。

```
CREATE FUNCTION DBA.f_verify_pwd
( user_name varchar(128),
new_pwd varchar(255) )
RETURNS varchar(255)
BEGIN
-- enforce password rules
IF new_pwd = user_name then
RETURN('Password cannot be the same as the user name' );
END IF;
-- return success
RETURN( NULL );
END;
ALTER FUNCTION DBA.f_verify_pwd set hidden;
GRANT EXECUTE on DBA.f_verify_pwd to PUBLIC;
SET OPTION PUBLIC.VERIFY_PASSWORD_FUNCTION = 'DBA.f_verify_pwd';
```

オプションをオフにするには、空の文字列を指定します。

```
SET OPTION PUBLIC.VERIFY_PASSWORD_FUNCTION = ''
```

参照：
- ALTER FUNCTION 文 (17 ページ)
- GRANT 文 (247 ページ)

WASH_AREA_BUFFERS_PERCENT オプション

ウォッシュ・マーカを超えるバッファ・キャッシュの割合を指定します。

指定できる値
1 – 100

デフォルト値
20

スコープ
PUBLIC グループのみに設定できます。このオプションを設定するには、DBA パーミッションが必要です。変更を有効にするには、データベース・サーバを停止し、再起動します。

説明
Sybase IQ のバッファ・キャッシュは、長い MRU/LRU チェーンとして構成されています。 ウォッシュ・マーカを超える領域は、ダーティ・ページをディスクにクリーンアップ (書き込み) するために使用されます。

IQ モニタの **-cache** レポート内の Gdirty カラムには、LRU バッファが "dirty" (変更済み) 状態で取り込まれた回数が表示されます。GDirty が一定時間 0 より大きい数であれば、SWEEPER_THREADS_PERCENT または WASH_AREA_BUFFERS_PERCENT の値を大きくする必要がある場合があります。

このオプションでは、ほとんどの場合、デフォルトの設定が最適です。まれに、Sybase 製品の保守契約を結んでいるサポート・センタがこの値を大きくするよう指示することがあります。

参照：
- SWEEPER_THREADS_PERCENT オプション (531 ページ)

WAIT_FOR_COMMIT オプション

データが操作されるときに、いつ外部キー整合性をチェックするかを決定します。

指定できる値
ON、OFF

デフォルト値
OFF

スコープ
個々の接続または PUBLIC グループに設定できます。すぐに有効になります。

説明
このオプションを ON に設定すると、データベースは次の **COMMIT** 文まで外部キー整合性をチェックしません。OFF の場合は、CHECK ON COMMIT オプションで作成されていないすべての外部キーが、挿入、更新、削除時にチェックされます。

WD_DELETE_METHOD オプション

WD インデックスの削除処理で使用されるアルゴリズムを指定します。

指定できる値
0 – 3

- 0: 削除方法はコスト・モデルにより選択される。コスト・モデルが選択するのは、ミッド・デリートとラージ・デリートのどちらか一方だけ。
- 1: スモール・デリートが強制される。削除されるローの数が、テーブルの全ロー数に比べて非常に少ないときは、スモール・デリートが便利。スモール・デリートはインデックスにランダムにアクセス可能、大きいデータセットでキャッシュがスラッシングされる。
- 2: ラージ・デリートが強制される。このアルゴリズムは、削除するロー検索のため全インデックスをスキャンする。削除されるローの数が、テーブルの全ロー数に比べてかなり多いときは、ラージ・デリートが便利。
- 3: ミッド・デリートが強制される。ミッド・デリートはインデックスに順番にアクセスするスモール・デリートの変形版であり、一般にスモール・デリートより高速。

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
WD_DELETE_METHOD は、**WD** インデックスの削除処理で使用されるアルゴリズムを指定します。このオプションを設定しないか 0 に設定した場合、削除方法はコスト・モデルにより選択されます。コスト・モデルは、適切な削除アルゴリズムを選択する際に、CPU 関連のコストと I/O 関連のコストを考慮します。コスト・モデルでは以下の要素が考慮されます。

- 削除されたロー
- インデックス・サイズ
- インデックス・データ型の幅
- インデックス・データのカーディナリティ
- 利用可能なテンポラリ・キャッシュ
- マシンに関連する I/O と CPU の特性
- 利用可能な CPU とスレッド

例
次の文では、**WD** インデックスからのラージ・デリートが強制されます。

```
SET TEMPORARY OPTION WD_DELETE_METHOD = 2
```

索引

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

あ

い

え

お

か

き

く

こ

さ

し

す

せ

た

て

と

は

ひ

ふ

ろ

わ