SYBASE®
An SAP® Company

コンポーネント統合サービス・ユーザーズ・ガイド

Adaptive Server® Enterprise

15.7

ドキュメント ID：DC36501-01-1570-01
改訂：2011 年 9 月

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

目次

第 1 章

概要

コンポーネント統合サービスとは、Adaptive Server® の機能を拡張し、相互運用性の強化を実現する機能です。

また、ロケーションの透過性を提供し、機能補正を行います。

コンポーネント統合サービスを使用すると、Adaptive Server が個々のクライアント・アプリケーションにエンタープライズ・データの同一のビューを提示できます。これをロケーションの透過性といいます。ユーザは、あたかもローカル・データであるかのように、エンタープライズ全体の異機種ソースのデータにアクセスできます。

機能補正は、コンポーネント統合サービスが Transact-SQL の全機能をエミュレートし、実際のデータが必要なときにのみデータ・ソースと対話できるようにする機能です。この機能を使用すると、データ・ソースが Transact-SQL の特定の機能をサポートしているかどうかにかかわらず、Transact-SQL の全機能をあらゆるデータ・ソースに適用できます。この機能には、たとえば、組み込み関数や Java 関数などがあります。コンポーネント統合サービスでは、これらの関数によって処理されるデータがこれらの関数をサポートできない外部ソースに派生している場合でも、文中でこれらの関数を使用できるようにします。

コンポーネント統合サービスは、Adaptive Server Anywhere、Adaptive Server IQ および各種 DirectConnect インタフェースとともに、エンタープライズ内のあらゆるデータベース管理システムへの透過的なアクセスを可能にし、Adaptive Server のアクセス範囲を拡張します。このように Adaptive Server Enterprise による透過的アクセスの範囲が拡大することにより、Enterprise Portal コンポーネントで以下の操作が容易になります。

- あらゆる場所から入手したデータにアクセスし、そのデータを動的コンテンツとして Web ページに表示する。
- 異機種間にまたがるトランザクションを実行する。
- Adaptive Server／コンポーネント統合サービス・システム・カタログに格納されているグローバル・メタデータによって提供される単一のビューを通じて、エンタープライズ全体を表示する。

コンポーネント統合サービスを使用すると、異なるサーバ上に存在する Sybase® データベースと Sybase 以外のデータベースの両方にアクセスできます。これらの外部データ・ソースでは、Adaptive Server や Oracle などのデータベース・システムのホスト・データ・ファイル、テーブル、ビュー、RPC (リモート・プロシージャ・コール) が使用できます。

コンポーネント統合サービスによって、次のことが可能になります。

- あたかもローカル・データであるかのようにリモート・サーバ内のテーブルにアクセスできる。
- 複数のリモート・サーバや異機種サーバにあるテーブル間でジョインを実行できる。たとえば、Oracle データベース管理システム (DBMS) と Adaptive Server 間や、複数の Adaptive Server 間でテーブルをジョインできる。
- select into 文を使用して、あるテーブルの内容を、サポートされているリモート・サーバ上の新しいテーブルに転送できる。
- 異機種のデータ・ソース間で参照整合性を維持する。
- コンポーネント統合サービスのパススルー・モードを使用して、ネイティブ・リモート・サーバの機能にアクセスできる。

コンポーネント統合サービスは、複数のデータ・ソースやレガシー・データにアクセスする必要があるすべての方にご利用いただけます。また、1 つのサーバから別のサーバにデータをマイグレートする必要がある方もご利用になれます。

1 台のサーバを使用して、外部の複数のサーバ上のデータにアクセスすることがよくあります。コンポーネント統合サービスは、外部サーバのロケーションに関係なくデータを管理します。データ管理は、クライアント・アプリケーションから透過的です。

コンポーネント統合サービスを EnterpriseConnect™ と MainframeConnect™ と組み合わせることで、次のような多様なデータ・ソースに透過的にアクセスできます。

- Oracle
- Informix
- Microsoft SQL Server
- Adaptive Server Enterprise
- Adaptive Server Anywhere
- Adaptive Server IQ
- メインフレーム・データ
  - ADABAS
  - IDMS
  - IMS
  - VSAM

コンポーネント統合サービスを起動するには、次の手順に従います。

- 目的の外部データ・ソース (Oracle、Informix、Microsoft SQL Server など) にアクセスするために、DirectConnect サーバまたはゲートウェイをインストールする。
- リモート・オブジェクトにアクセスするためのサーバを設定する。「第 2 章 コンポーネント統合サービスの概要」を参照。

第 2 章

コンポーネント統合サービスの概要

この章では、コンポーネント統合サービスの使用方法について説明します。設定されるコンポーネント統合サービス・オプションを使用して Adaptive Server がどのように動作するかを理解するのに役立ちます。

基本概念

コンポーネント統合サービスの最大の特長は、ローカル・テーブルのようにリモート・テーブル (または外部テーブル) にアクセスできることです。コンポーネント統合サービスは、テーブル内にあるすべてのデータがローカルに格納されているかのように、テーブルをクライアント・アプリケーションに提示します。リモート・テーブルは、メタデータを保管しているローカルのプロキシ・テーブルにマップされます。リモート・テーブルに関わるクエリが実行されると、内部では格納領域のロケーションが判別され、リモートのロケーションにアクセスしてデータを取り出します。

リモート・データを取り出すのに使用するアクセス・メソッドは、外部オブジェクトの次の 2 つの属性によって決定されます。

- リモート・オブジェクトに関連付けられているサーバ・クラス
- オブジェクト・タイプ

ロケーションを透過的にするには、まずテーブルをそれぞれに対応する外部ロケーションにマップする必要があります。

アクセス・メソッド

アクセス・メソッドは、サーバと外部オブジェクト間のインタフェースを形成します。各サーバ・クラスには、クラスとオブジェクト・タイプが同一の Adaptive Server とリモート・サーバ間のすべての対話を処理する個別のアクセス・メソッドがあります。

サーバ・クラス

sp_addserver を使用してサーバを追加するときは、各サーバにサーバ・クラスを割り当てます。サーバ・クラスは、リモート・サーバと対話するためのアクセス・メソッドを指定します。サーバ・クラスを次に示します。

- *ASEnterprise* − サーバが Adaptive Server の場合に使用する。これがデフォルトのサーバ・クラスである。
- *ASAnywhere* − サーバが Adaptive Server Anywhere バージョン 6.0 以降の場合に使用する。このサーバ・クラスは、Adaptive Server IQ 12.5 より前のバージョンの Adaptive Server IQ に対して使用する。
- *ASIQ* − サーバが Adaptive Server IQ バージョン 12.5 以降の場合に使用する。
- *local* − ローカル・サーバ。1 つだけ存在できる。
- *direct_connect* − サーバが DirectConnect™ サーバのインタフェース要件に準拠する Open Server™ アプリケーションであることを示す。Microsoft SQL Server、DB2、Oracle、または Informix にアクセスするには、DirectConnect サーバを使用する。
- *sds* − サーバが Specialty Data Store のインタフェースの要件に準拠することを示す。
- RPCServer − サーバが RPC のみ処理できることを示す。SQL 文、トランザクション制御文などは処理できない。

オブジェクト・タイプ

サーバは、複数のオブジェクト・タイプを、それがローカル・テーブルであるかのようにクライアント・アプリケーションに提示します。サポートしているオブジェクト・タイプを次に示します。

- table － 任意クラスのリモート・サーバ内にあるオブジェクトはリレーショナル・テーブルである。これがデフォルトのタイプである。
- view － 任意クラスのリモート・サーバ内にあるオブジェクトはビューである。コンポーネント統合サービスは、ビューをインデックスのないローカル・テーブルのように扱う。
- remote procedure － 任意クラスのリモート・サーバ内にあるオブジェクトはリモート・プロシージャである。コンポーネント統合サービスは、リモート・プロシージャから返された結果セットを読み込み専用テーブルとして扱う。
- file － オブジェクトは、ファイル・システム内の個別ファイルである。
- directory － オブジェクトは、ファイル・システム・ディレクトリである。

リモート・サーバへのインタフェース

サーバとリモート・サーバ間のインタフェースは、Open Client ソフトウェアの Client-Library™ によって処理されます。インタフェースを実装するのに使用される Client-Library 機能は、コンポーネント統合サービスが対話するサーバ・クラスによって異なります。

たとえば、サーバ・クラスが *direct_connect* の場合、カーソル要求や動的要求などの数多くの機能が使用されます。

サーバがリモート・サーバと対話できるようにするには、次の設定を行います。

- directory services へのリモート・サーバの追加
- リモート・サーバの定義
- リモート・サーバのログイン情報
- リモート・オブジェクトの定義

ディレクトリ・サービス

コンポーネント統合サービスを使用してリモート・テーブルにアクセスする前に、LDAP ディレクトリ・サービスか、*interfaces* ファイル (Windows プラットフォームの場合は *sql.ini*) にアクセスします。リモート・テーブルへのアクセスの詳細については、「接続管理」(31 ページ) を参照してください。ディレクトリ・サービスの設定方法については、使用しているプラットフォームの設定マニュアルを参照してください。コンポーネント統合サービスのユーザを対象にした基本チュートリアルの「付録 A チュートリアル」も参照してください。

リモート・サーバの定義

リモート・サーバは、ストアド・プロシージャ sp_addserver を使用して定義します。この手順は、『ASE リファレンス・マニュアル』で説明しています。

リモート・サーバへのログイン

リモート・サーバを設定したら、ログイン情報を指定します。デフォルトでは、コンポーネント統合サービスは、クライアントとしてリモート・サーバに接続するときは必ず Adaptive Serve のクライアントの名前とパスワードを使用します。ただし、このデフォルトは sp_addexternlogin を使用して上書きできます。システム管理者は、この方法を使用して、リモート・サーバに接続する各ユーザのユーザ名とパスワードを定義できます。

connect to *server_name* を使用して、サーバの設定が正しいことを確認できます。このコマンドは、リモート・サーバへのパススルー・モード接続を確立します。パススルー・モードを使用することで、クライアントは、リモート・サーバとネイティブ構文で通信できるようになります。このパススルー・モードは、disconnect コマンドを発行するまで有効です。

リモート・オブジェクトの定義

リモート・サーバを設定しても、そのリモート・サーバ内にあるオブジェクトとローカル・オブジェクト (プロキシ・テーブル) 間のマッピングを確立するまで、それらのオブジェクトにはテーブルとしてアクセスできません。

リモート・サーバ上に新しいテーブルを作成したり、リモート・サーバ内にある既存のオブジェクトのスキーマを定義したりできます。どちらの手順も似ています。

プロキシ・テーブル

プロキシ・テーブルは、ロケーションの透過性を実現する上で非常に重要なオブジェクトです。プロキシ・テーブルは、リモート・オブジェクトを指すメタデータを格納するローカル・テーブルです。リモート・オブジェクトの詳細については、「オブジェクト・タイプ」(7 ページ) を参照してください。リモート・テーブルはプロキシ・テーブルにマップされ、ローカル・テーブルのように表示されます。

この方法の詳細については、「第 3 章 SQL リファレンス」を参照してください。

create table コマンドの使用方法

create table コマンドは、プロキシ・テーブルとリモート・テーブルを次の構文で同時に作成します。

create table table_name (column_list) [[external {table | file}] at “pathname”]]

リモート・ロケーションは、at pathname 句で指定されます。create table では、外部オブジェクト・タイプの table と file を使用できます。各カラムのデータ型は、変換されずにリモート・サーバに渡されます。

create existing table コマンドの使用方法

create existing table コマンドは、既存のテーブル (プロキシ・テーブル) を定義できます。このオプションの構文は、create table コマンドの構文と似ており、次のようになります。

```
create existing table table_name (column_list)
[[external {table | procedure | file}] at pathname]
```

このコマンドを受け取ったときのサーバの動作は、create table コマンドを受け取ったときの動作とかなり異なります。リモート・ロケーションに新しいテーブルが作成されません。その代わりに、テーブルのマッピングがチェックされ、基本となるオブジェクトの存在が確認されます。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、このコマンドは拒否されてエラー・メッセージが出されます。

オブジェクトが存在する場合、その属性が取得され、システム・テーブル sysobjects、syscolumns、sysindexes を更新するのに使用されます。

- 既存のオブジェクトの性質が判断される。
- リモート・サーバ・オブジェクト (RPC 以外) の場合は、テーブルまたはビューにあるカラム属性が、*column_list* で定義されているものと比較される。カラム名 (大文字と小文字が区別される)、カラムの型と長さ、カラムの NULL 属性が一致しなければならない。カラムの型と長さについては、少なくとも変換可能でなければならない。
- ホスト・データ・ファイルまたはリモート・サーバ・テーブルからインデックス情報を抽出し、システム・テーブル sysindexes のローを作成するために使用する。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る。

既存のテーブルが正しく定義できたら、テーブルに対して update statistics コマンドを発行します。これにより、クエリ・オプティマイザは、インデックス選択とジョイン順序に関して高度な選択を行えるようになります。

データ型の変換

create table コマンドまたは create existing table コマンドを使用する場合は、認識されている Adaptive Server データ型を使用してすべてのデータ型を指定します。リモート・サーバ・テーブルが異機種のサーバのクラスにある場合は、データが検索されるときに、リモート・テーブルのデータ型は指定された Adaptive Server 型に自動的に変換されます。変換できなかった場合、create table コマンドまたは create existing table コマンドを使用してテーブルの作成や定義を行うことはできません。

リモート・テーブル定義の例

次の例は、リモートの Adaptive Server テーブル authors を定義するのに必要な手順を、サーバの定義から順に説明しています。

1 SYBASE という名前のサーバを定義する。サーバ・クラスは *ASEnterprise* で、interfaces ファイル内での名前は SYBASE です。

```
exec sp_addserver SYBASE, ASEnterprise, SYBASE
```

2 リモート・ログイン・エイリアスを定義する。*username* と *password* が両方のサーバで等しい場合、この手順は任意です。ユーザ “sa” は、リモート・サーバ SYBASE に、ユーザ “sa”、パスワード “timothy” として認識されています。

```
exec sp_addexternlogin SYBASE, sa, sa, timothy
```

3 リモート authors テーブルを定義します。

```
create existing table authors
(
au_id        varchar(11)      not null,
au_lname     varchar(40)      not null,
au_fname     varchar(20)      not null,
phone        char(12)         not null,
address      varchar(40)      null,
city         varchar(20)      null,
state        char(2)          null,
country      varchar(12)      null,
postalcode   char(10)         null
)
EXTERNAL TABLE at "SYBASE.pubs2.dbo.authors"
```

4 クエリ・オプティマイザが正しい選択を行えるように、テーブル内の統計情報を更新します。

```
update statistics authors
```

5 クエリを実行して、設定をテストします。

```
select * from authors where au_lname = 'Carson'
```

create proxy_table コマンドの使用方法

create proxy_table コマンドの使用には、カラム・リストは必要ありません。コンポーネント統合サービスは、リモート・テーブルから取得するメタデータからカラム・リストを生成します。

オブジェクトが存在する場合は、create proxy_table によって sysobjects、syscolumns、sysindexes が更新されます。

プロキシ・テーブルとしてのリモート・プロシージャ

create existing table 文にオプションの句を追加して、リモート・オブジェクトがテーブルではなく、実際にはストアド・プロシージャ (またはほかのプロシージャ) であることを示すことができます。この句を指定しないと、リモート・オブジェクトはテーブルまたはビューであると見なされます。

```
create existing table t1
    (
                    column_1    int,
                    column_2    int
    )
    EXTERNAL PROCEDURE AT "SYBASE.mydb.dbo.p1"
```

リモート・オブジェクトのタイプがプロシージャの場合は、処理上で次のような違いが生じます。

- このタイプのオブジェクトではインデックスが作成されない。
- リモート・プロシージャの結果セットの記述と一致するカラム・リストを提供する必要がある。リストの正確さは保証されない。
- アンダースコア ('_') で始まるカラム名は、リモート・プロシージャの結果セットに含まれないカラムを指定するのに使用できる。これらのカラムは、パラメータ・カラムと呼ばれる。次に例を示す。

  ```
  create existing table t1
  (
      a           int,
      b           int,
      c           int,
      _p1         int null,
      _p2         int null
  )
  external procedure
  at “SYBASE.sybsystemprocs.dbo.myproc”

  select a, b, c from t1
  where _p1 = 10 and _p2 = 20
  ```

- この例では、パラメータ・カラム *_p1* および *_p2* が結果セットに含まれないことになっているが、クエリで参照できる。コンポーネント統合サービスは *@p1* および *@p2* という名前のパラメータを介して、探索引数をリモート・プロシージャに渡す。
- パラメータのカラムが select リストに含まれていて、これがリモート・プロシージャにパラメータとして渡される場合、その値は where 句に指定される値と等しくなる。パラメータ・カラムが where 句になかったり、リモート・プロシージャにパラメータとして渡せなかったが、select リストに含まれていたりする場合、その値は NULL になる。

- Adaptive Server クエリ・プロセッサが検索可能な引数 SARG と見なした場合、パラメータ・カラムは、リモート・プロシージャにパラメータとして渡すことができる。“or” 述部に含まれない場合、通常は SARG である。たとえば、次のクエリでは、パラメータ・カラムはパラメータとして使用されない。

  ```
  select a, b, c from t1
  where _p1 = 10 OR _p2 = 20
  ```

- create existing table 文にパラメータ・カラムを定義する場合は、次のルールに従う必要がある。
  - パラメータ・カラムは NULL を許可する必要がある。
  - パラメータ・カラムは、通常の結果カラムの前にあってはならない (カラム・リストの最後になければならない)。

  リモート・プロシージャの定義をローカル・テーブルとして許可することで、コンポーネント統合サービスは、リモート・プロシージャの結果セットを「仮想テーブル」として扱えるようになります。「仮想テーブル」は、ソートしたり、ほかのテーブルとジョインしたり、insert/select 構文を使ってほかのテーブルに挿入できます。ただし、仮想テーブルは読み込み専用と見なされます。
  - *procedure* タイプのテーブルに対して、delete、update、または insert コマンドは発行できない。
  - 仮想テーブルに対して、create index、truncate table、または alter table コマンドは発行できない。

タイプが procedure のオブジェクトがサーバ内で定義されている場合、オブジェクトが格納されているリモート・サーバにクエリは発行されません。その代わり、コンポーネント統合サービスは RPC を発行し、RPC からの結果を読み込み専用のテーブルとして扱います。

例

```
create existing table rtable
 (  col1    int,
    col2    datetime,
    col3    varchar(30)
 )
external procedure at “SYBASE...myproc”

select * from rtable
```

このクエリを発行すると、コンポーネント統合サービスは *myproc* という名前の RPC をサーバ SYBASE に送信します。ローの結果は、ほかのテーブルからの結果と同じように扱われます。つまり、ソート、ほかのテーブルとのジョイン、グループ化、ほかのテーブルへの挿入などの操作が行えます。

RPC パラメータは、結果セットを制限する引数を表している必要があります。パラメータを指定せずに RPC を発行した場合、オブジェクトの結果セット全体が返されます。パラメータを指定して RPC を発行した場合は、各パラメータにより結果セットは制限されます。次に例を示します。

```
select * from rtable where col1 = 10
```

このクエリでは、@col1 という名前の 1 つのパラメータが RPC とともに送信されます。これの値は 10 です。

コンポーネント統合サービスは、リモート・サーバにできるだけ多くの探索引数を渡そうとしますが、実行される SQL 文によっては、コンポーネント統合サービスは結果セットの計算を独自に行います。各パラメータは、= 演算子のように、完全一致の検索を表します。

RPC に送信するパラメータを定義するルールを次に示します。RPC をコンポーネント統合サービスのオブジェクトとして使用する場合は、開発時に次のルールを考慮してください。

- コンポーネント統合サービスは、where 句内の = 演算子をパラメータとして送信する。たとえば、このクエリでは、コンポーネント統合サービスは 2 つのパラメータを送信する。

  ```
  select * from rpc1 where a = 3 and b = 2
  ```

  パラメータ a の値は 3 で、パラメータ b の値は 2 です。RPC は、カラム a の値が 3 で、カラム b の値が 2 である結果ローだけを返します。

- 完全な探索条件が指定されていない場合、コンポーネント統合サービスは where 句、または where 句の一部をパラメータとして送信しない。次に例を示す。

  ```
  select * from rpc1 where a = 3 or b = 2
  ```

  or 句があるため、コンポーネント統合サービスは、a または b のパラメータを送信しません。

  別の例を示します。

  ```
  select * from rpc1 where a = 2 and b < 3
  ```

  コンポーネント統合サービスは、*a* と *b* のパラメータを送信し、*b* に 3 より小さい値を含むローを通すようにフィルタします。

サーバの制限

Adaptive Server では、ページ・サイズを 2K、4K、8K、または 16K バイトに設定できます。また、char/binary カラムに対する 255 バイトの制限も廃止されています。Adaptive Server では、char、varchar、univarchar、unichar、binary、varbinary の各データ型の拡張サイズをサポートしています。新しい制限は、サーバのページ・サイズによって異なります。ページ・サイズによって、新しい制限は次のようになります。

表 2-1: 新しい制限

ページ・サイズ	カラムの最大サイズ
2048	2048
4096	4096
8192	8192
16384	16384

上記のサイズはおおよそのサイズです。基本ルールでは、やはり 1 つのローを 1 ページに収めることができる大きさを最大サイズとします。これらの制限は、テーブルの作成時に指定されたロック・スキームによっても異なります。プロキシ・テーブルの大半が、全ページをロックするデフォルトのロック・スキームによって作成されると考えられます。

- Transact-SQL の変数とパラメータの長さに対する制限 － char、varchar、binary、varbinary の各変数のサイズが拡大され、所定サーバでの同じデータ型のカラムの最大サイズに等しくなっている。これにより、長さが現在の制限 (255 バイト) を超える変数をストアド・プロシージャ (または RPC) に渡すことができる。
- 1 テーブルあたりのカラム数に対する制限 － 1 テーブルあたり最大 1024 カラム (1 ページに収まるカラムの最大数)。ただし、可変長カラムは 254 に制限されている (null カラムも可変長と見なされる)。
- インデックスの幅に対する制限 － Adaptive Server のインデックスの全幅は、サーバのページ・サイズに応じて、以前のバージョンよりも大きくすることができる。表 2-2 に、ページ・サイズに応じたインデックスの最大幅を示す。

表 2-2: インデックスの最大幅

ページ・サイズ	インデックスの幅
2048	600
4096	1250
8192	2600
16384	5300

- 1 インデックスあたりのカラム数に対する制限 － 1 インデックスあたり 31 カラム。
- テーブル名、カラム名、インデックス名は、最大 255 バイト。
- 識別子名は最大 255 バイト。

create new proxy table

create table では、上記のように長さが拡大された、char、varchar、binary、varbinary の各データ型のカラムを指定できます。これらのデータ型と長さは、テーブルが作成されるリモート・サーバに転送されます。

create existing proxy table

create existing table コマンドでも、長さが 255 バイト以上のカラムを指定できます。これにより、コンポーネント統合サービスは、以前には text や image カラムとして取り扱わなければならなかったカラムを、char、varchar、binary または varbinary カラムとしてリモート・データベースで取り扱うことができます。

それでもカラム・サイズの不一致エラーが発生することがあります。たとえば、カラム長が 5000 バイトのテーブルがリモート・データベースに格納され、create existing table を処理する Adaptive Server が最大 1900 バイトまでのカラムしかサポートしていない場合には、サイズの不一致エラーが発生します。この場合、カラムを text または image カラムとして再指定する必要があります。

また、プロキシ・テーブルのカラム・サイズがリモート・テーブル内の対応するカラムのサイズを超えている場合には、サイズの不一致エラーが検出され、コマンドがアボートされます。

create proxy_table

create proxy_table は、リモート・サーバからメタデータをインポートし、インポートされたメタデータからカラム・リストを取り出して、カラム情報を内部 create existing table に変換します。カラムのメタデータを取得するときには、リモート DBMS タイプから Adaptive Server Enterprise タイプに変換する必要があります。

リモートの char、varchar、binary、または varbinary カラムのサイズがローカル・サーバで許容される最大値を超えている場合は、長さが最大サイズ (ページ・サイズによって異なる) にトランケートされます。サイズが 16K バイトを超える場合、char 型または varchar 型は text 型に変換され、binary 型または varbinary 型は image 型に変換されます。

alter table

alter table がプロキシ・テーブルを処理する場合、このコマンドはまずローカルで処理され、次にリモート・サーバに転送されて実行されます。リモートでの実行に失敗すると、ローカルの変更が取り消され、コマンドはアボートされます。

リモート・サーバはコマンドを適切に処理しなければなりません。さもないと、エラーが発生します。エラーが生成された場合は、このコマンドのコンポーネント統合サービス側がアボートされ、ロールバックされます。

select、*insert*、*delete*、*update*

プロキシ・テーブルがデータ操作言語 (DML) のオペレーションに関与している場合、コンポーネント統合サービスは大きなカラムの値を処理します。コンポーネント統合サービスは、次のいずれかの方式で DML を処理します。

- TDS (Tabular Data Stream)™ 言語コマンド － SQL 文全体をリモート・サーバに転送できる場合、コンポーネント統合サービスは、CT-Library ct_command (CS_LANG_CMD) によって生成された TDS 言語コマンドを使用して転送する。

  言語バッファのテキストは、255 バイトを超える long char または binary 値のデータを格納でき、リモート・サーバは、これらのコマンド・バッファの解析を処理しなければなりません。

- TDS 動的コマンド － コンポーネント統合サービスが SQL 文全体をリモート・サーバに転送できない場合 (たとえば、コンポーネント統合サービスが文の機能補正を行わなければならない場合)、insert、update または delete は、TDS 動的コマンドを使用し、必要に応じて CT-Library 関数 ct_dynamic (CS_PREPARE_CMD、CS_EXECUTE_CMD、CS_DEALLOC_CMD) によってパラメータを設定することで処理できる。

  この動的コマンドのパラメータは、CS_LONGCHAR_TYPE でも、CS_LONGBINARY_TYPE でもかまいません。

- TDS カーソル・コマンド － 機能補正を実行しなければならない場合、CT-Library カーソル操作によって select、update、delete に対するプロキシ・テーブルの処理を行うことができる。たとえば、プロキシ・テーブルを更新するときに from 句に複数のテーブルが存在する場合、コンポーネント統合サービスは、複数のデータ・ソースからローをフェッチし、条件に合うローごとに対象のテーブルに対し update を適用しなければならないことがある。この場合、コンポーネント統合サービスは、ct_cursor ({CS_DECLARE_CMD, CS_OPEN_CMD, CS_CURSOR_UPDATE_CMD, CS_CLOSE_CMD, CS_DEALLOC_CMD}) を使用する。

  カーソルを準備してから、パラメータを指定します。これらのパラメータには、CS_LONGCHAR 型または CS_LONGBINARY 型のパラメータを含めることができます。

- バルク挿入コマンド － select/into コマンドを実行するときにターゲット・サーバがバルク・インタフェースをサポートしている場合 (リモートの Adaptive Server と DirectConnect for Oracle にのみ当てはまる) は、リモート・サーバが、(CS_LONGCHAR、CS_LONGBINARY 値を介して) 255 よりも大きい char 値と binary 値を処理できるようにしておく必要がある。

リモート・サーバからのカラムは、CS_LONGCHAR_TYPE 型または CS_LONGBINARY_TYPE 型としてコンポーネント統合サービスに返されることがあります。

RPC 処理

リモート・サーバに送信される RPC は、CS_LONGCHAR 型と CS_LONGBINARY 型のパラメータを含むことができます。コンポーネント統合サービスの cis_rpc_handling コマンドは、これらの型をサポートします。

バージョン 12.5 より前の Adaptive Server には、長いパラメータを送信できません。旧バージョンの Adaptive Server では、CS_LONGCHAR データまたは CS_LONGBINARY データがサポートされていないためです。コンポーネント統合サービスは、RPC を送信する前にリモート・サーバの TDS 機能を検証し、リモート・サーバがこれらのデータ型を受け付けることができない場合は、エラーが発生します。

sp_tables

Adaptive Server Anywhere または ASIQ のストアド・プロシージャ sp_tables はユーザ・テーブルのみを返します。

カスケード・プロキシ・テーブル

Adaptive Server では、コンポーネント統合サービスの複数のインスタンス間でカスケード・プロキシ・テーブルを設定できます。

これによって循環参照 (2 番目のプロキシ・テーブルが 1 番目のプロキシ・テーブルと同じサーバ上のローカル・テーブルを参照するトランザクション) などの問題が発生することがあります。この問題が発生した場合は、コンポーネント統合サービスによって検出されないアプリケーションのデッドロックが生じる可能性があります。このような問題を避けるようにシステムを設定してください。

プロキシ・データベース

プロキシ・データベースには、ユーザ・プロキシ・データベースとシステム・プロキシ・データベースの 2 種類があります。

ユーザ・プロキシ・データベース

ユーザ・プロキシ・データベースを作成すると、実際のテーブルを含むリモート・ロケーションから、プロキシ・テーブルのメタデータが自動的にインポートされます。このメタデータは、プロキシ・データベース内にプロキシ・テーブルを作成するのに使用されます。

プロキシ・データベースを作成するには、次のコマンドを使用します。

```
create database <dbname>
    [create database options]
    [with default_location = 'pathname'
    [for proxy_update]]
```

with default_location 句を使用すると、新しいテーブルの格納領域のロケーションを指定でき、また for proxy_update 句も指定した場合は、プロキシ・テーブルを自動作成するためのメタデータのインポート元も指定できます。for proxy_update は、データベースをプロキシ・データベースとして設定し、with default_location は、プロキシ・テーブルのインポート元のロケーションを定義します。for proxy_update を使用しないと、with default_location の動作は、sp_defaultloc によって実現される動作と同じになります。したがって、デフォルトの格納領域のロケーションは、新しいテーブルと既存のテーブルを作成するために設定されますが、プロキシ・テーブル定義の自動インポートは、create database コマンドを処理する際に行われません。

パス名の値は、*servername.dbname.owner* というフォーマットの文字識別子で表します。

- *servername* − 必須フィールド。プロキシ・テーブルが参照するオブジェクトを所有しているサーバ名を表す。*master.dbo.sysservers.srvname* に存在している必要がある。
- *dbname* − 省略可能。*servername* 内にあるデータベースの名前で、プロキシ・テーブルが参照するオブジェクトを含む。
- *owner* − 省略可能。プロキシ・テーブルが参照するオブジェクトの所有者名である。指定すると、使用が制限される場合がある。そのため、複数のユーザが *dbname* 内にオブジェクトを所有している場合、所有者を指定すると、そのユーザが所有しているオブジェクトだけが選択の対象となる。ほかのユーザが所有しているオブジェクトのプロキシ・テーブルは作成できない。

default_location を指定しないで for proxy_update を指定すると、エラーが表示されます。

for proxy_update オプションを使用してプロキシ・データベースを作成すると、コンポーネント統合サービス機能が呼び出され、次の処理を行います。

- プライマリ・サーバのデータベースに存在する実際のテーブルやビューを表示する、すべてのプロキシ・テーブルを格納するのに必要なデータベース・サイズを見積る。この見積りは、すべてのプロキシ・テーブルとインデックスを格納するのに必要なデータベースのページ数として提示される。サイズもデータベース・デバイスも指定されなかった場合は、このサイズが使用される。

  注意　on device_name = nn で特定のサイズを指定してデータベースを作成したり、on device_name でサイズを指定せずにデバイス名を指定した場合は、プロキシ・データベースに必要なサイズは計算されません。この場合、ユーザやデータベース管理者によって、このプロキシ・データベースに対して計算されたデフォルト・サイズが上書きされるものと見なされます。

  別の Adaptive Server からメタデータをインポートしている場合は、プロキシ・テーブルが作成される前にリモート・データベース・ユーザがインポートされます。インポートされる各データベース・ユーザには、それぞれに対応するシステム・ユーザ名が syslogins に存在しなければなりません。
- コンパニオン・サーバのデータベースで検出された実際のテーブルやビューを表すすべてのプロキシ・テーブルを作成する。システム・テーブル用のプロキシ・テーブルは作成されない。
- プロキシ・テーブルのすべてのパーミッションを “public” に付与する。
- “guest” ユーザをプロキシ・データベースに追加する。
- リモート・サイトからデータベース・ユーザをインポートする (Adaptive Server の場合)。
- create table パーミッションを “public” に付与する。
- ユーザ・プロキシ・データベースであることを示すように、データベースのステータスを設定する。ステータスを設定するには、*master.dbo.sysdatabases.status*3 (0x0001, DBT3_USER_PROXYDB) でステータス・フィールドを設定する。

データベースの作成後、データベースには、デフォルトのロケーションで検出された各テーブルやビューのプロキシ・テーブルが格納されます。ユーザ・プロキシ・データベースの動作は、以前のデータベースの動作と同じになります。ユーザは、procedure、views、rules、または defaults などの追加のオブジェクトを作成できます。また、プロキシ・テーブルを処理する DDL 文と DML 文は、このマニュアルで説明されているとおりに動作します。

唯一の例外は、alter database コマンドです。このコマンドの構文と機能については、次の項で説明します。

ユーザ・プロキシ・データベース・スキーマの同期

データベース管理者は、プロキシ・データベース内に含まれているプロキシ・テーブルの再同期を強制的に実行しなければならないことがあります。これは、次の alter database コマンドを使用して実行できます。

```
alter database <dbname>
        [alter database options]
        [for proxy_update]
```

ほかのオプションを指定しないで for proxy_update 句を入力した場合、データベースのサイズは拡張されません。その代わり、プロキシ・テーブル (存在する場合) がプロキシ・データベースから削除され、メタデータ (create database ... with default_location = 'pathname' で指定した *pathname* から取得) から再作成されます。

データベースのサイズを拡張するために、create database をほかのオプションとともに使用した場合は、サイズが拡張されてからプロキシ・テーブルが同期されます。

alter database は、単一のリモート・サイトにあるすべてのテーブルの最新で正確なプロキシ表示を、データベース管理者が 1 つのステップで簡単に取得できるように拡張されています。

この再同期は、HA クラスタ環境内のプライマリ・サーバだけではなく、すべての外部データ・ソースでサポートされています。また、データベースは、for proxy_update 句によって作成されていなくてもかまいません。create database コマンドまたは sp_defaultloc を使用してデフォルトの格納領域のロケーションを指定した場合、データベース内のメタデータは、リモートの格納領域のロケーションにあるメタデータと同期できます。

create database や alter database コマンドを使用してプロキシ・データベースを指定すると、次のようになります。

- create database コマンドを使用して指定したデフォルトのロケーションは、alter database コマンドでは変更できない。
- ローカル・テーブルは、プロキシ・データベース内に作成できない。create table コマンドを実行すると、プロキシ・テーブルが作成され、実際のテーブルはデフォルトのロケーションに作成される。
- テーブルのデフォルトのロケーションは、at 'pathname' 構文を使用して、create table コマンドで指定できる。パス名がデフォルトのロケーションと異なる場合、alter database コマンドは、このテーブルのメタデータを同期しない。
- デフォルトのロケーションを変更するには、データベースを削除し、default_location = 'pathname' 句に新しいパス名を指定して再作成する。sp_defaultloc を使用してロケーションを変更した場合、メタデータの同期には新しいロケーションが使用され、以前のロケーションで作成されたプロキシ・テーブルは同期されない。また、それが新しいロケーションにあるテーブル名と重複する場合は、削除されて置き換えられる。

システム・プロキシ・データベース

システム・プロキシ・データベースは、ユーザ・プロキシ・データベースと同じように動作しますが、重要な特長と例外がいくつかあります。システム・プロキシ・データベースは、HA 設定でのみ使用されます。

システム・プロキシ・データベースでは、カスタマ作成のアプリケーションを高可用性クラスタのどのノードでも実行できます。これは、「単一システム・イメージ機能」ということではなく、ほとんどのユーザ作成のアプリケーションがクラスタのいずれのノードでも実行できる環境を意味しています。つまり、データベースとユーザ作成オブジェクトのどちらも両方のノードから参照できるようにする必要があります。

システム・プロキシ・データベースは、参照先のプライマリ・ノード内のデータベースと同じ名前を持ち、アプリケーションをサポートするのに必要なユーザ定義オブジェクトの処理を含みます。プロキシ・テーブルは、プライマリ・データベースに存在するユーザ・テーブルとビューごとに作成され、ストアド・プロシージャは、RPC に変換され、プロキシ・データベースによって参照されるノードに転送されます。

システム・プロキシ・データベースの作成

システム・プロキシ・データベースは、次の場合に自動的に作成されます。

- ストアド・プロシージャ sp_companion ServerName、'configure'、with_proxydb を使用して、HA クラスタを設定する。

  この場合、システム・プロキシ・データベースは、ServerName で指定されたサーバに存在するユーザ・データベースごとに作成されます。

- HA ステータスが MODE_APNC、MODE_SNC、MODE_ASNC のいずれかであるサーバで、create database コマンドを発行する。

システム・プロキシ・データベースの作成が完了すると、コンポーネント統合サービス関数が呼び出され、次のコマンドを実行します。

- grant create table to public － このコマンドを使用してプライマリ・サーバでテーブルを作成すると、システム・プロキシ・データベースでプロキシ・テーブルを作成できるようになる。

現在のデータベースにシステム・プロキシ・データベースがあるときのスキーマの同期

HA クラスタでは、プライマリ・サーバのデータベースを変更した場合、両方のサーバの同期を維持するために変更内容をコンパニオン・サーバに転送しなければならないことがあります。

システム・プロキシ・データベースを持つデータベース内で次の DDL コマンドを実行すると、コンパニオン・サーバに通知され、変更が自動的に同期されます。

- create table と drop table − ローカル・オペレーションが実行されて、ローカル・テーブルが作成または削除される。その後、プロキシ・テーブルが作成または削除できるように、このコマンドがコンパニオン・サーバに転送されて、システム・プロキシ・データベースで実行される。
- create index と drop index − ローカル・オペレーションが実行されて、インデックスが作成または削除される。次に、システム・プロキシ・データベースを所有するサーバに通知され、プロキシ・インデックスが削除されて再作成され、インデックスの変更をプロキシ・テーブル内で表示することができる。
- create view と drop view − ローカル・オペレーションが実行され、ローカル・ビューが作成または削除される。次に、システム・プロキシ・データベースを所有するサーバに通知され、プロキシ・ビューが作成または削除される。

これらのコマンドをシステム・プロキシ・データベース内で実行した場合も同じような動作が発生します。

- create table と drop table − ローカル・プロキシ・テーブルが作成または削除される。次に、プロキシ・テーブルによって参照されているローカル・テーブルを作成または削除できるように、このコマンドがプライマリ・サーバに転送される。
- create index と drop index − プロキシ・テーブルに対するローカル・オペレーションが実行され、インデックスが作成または削除される。次に、プライマリ・データベースを所有しているサーバに通知され、プロキシ・テーブルによって参照されているローカル・テーブルでインデックスが作成または削除される。
- create view と drop view − システム・プロキシ・データベース内では使用できない。

システム・プロキシ・データベース内でのストアド・プロシージャの実行

現在のデータベースがシステム・プロキシ・データベースの場合にシステム・ストアド・プロシージャ要求が発生すると、コンポーネント統合サービスは、最初にストアド・プロシージャをローカルの sybsystemprocs データベースで探して実行しようとします。sybsystemprocs で見つからない場合、コンポーネント統合サービスは master データベースを検索します。プロシージャがシステム・ストアド・プロシージャでない場合、またはシステム・ストアド・プロシージャだがローカルで見つからない場合は、ストアド・プロシージャ要求は RPC に変換され、システム・プロキシ・データベースのデフォルトのロケーションで参照されているサーバに送信されます。

システム・プロキシ・データベースのその他の動作

次の組み合わせでコマンドをシステム・プロキシ・データベース内で実行すると、拒否されてエラーが発生します。

- create procedure と drop procedure
- create view と drop view
- create trigger と drop trigger
- create rule と drop rule
- create default と drop default

使用した場合、「メッセージ 12818、重大度 16：システムが作成したプロキシ・データベースには、この型のオブジェクトを作成できません。」というエラー・メッセージが生成されます。

ファイル・システム・アクセス

注意 現在、プロキシ・テーブルにマップされているディレクトリとファイルのファイル・パスの制限は 255 バイトです。

Adaptive Server では、SQL 言語によってファイル・システムにアクセスできます。ファイル・システム・アクセスを使用すると、ファイル・システム・ディレクトリや個別のファイルにマップされるプロキシ・テーブルを作成できます。

ディレクトリやファイルにマップされるプロキシ・テーブルを作成するには、システム管理者またはシステム・セキュリティ担当者の権限が必要です。

セキュリティの考慮事項

Adaptive Server Enterprise では、システム管理者 (sa) またはシステム・セキュリティ担当者 (sso) の役割が付与されているユーザ以外は、ファイルやディレクトリにマッピングされるプロキシ・テーブルを作成できません。この要件により、(実行時にルート・パーミッションが付与される可能性のある) Adaptive Server Enterprise サーバ・プロセス内からのファイル・システム・データへのアクセスに対するセキュリティの懸念に対処しています。

ディレクトリ・アクセス

プロキシ・テーブルを作成して、ファイル・システム・ディレクトリを参照できます。サポートされる構文は次のとおりです。

```
create proxy_table <table_name>
external directory at "directory pathname[;R]"
```

ディレクトリのパス名は、Adaptive Server Enterprise プロセスが認識して検索できるファイル・システム・ディレクトリを参照する必要があります。作成されるプロキシ・テーブルは、カラム名をディレクトリ内に存在するファイルの属性にマップします。パス名の末尾に拡張子の ';R' (「再帰」を示す) を追加すると、コンポーネント統合サービスは、すべての下位ディレクトリのエントリを含めます。表 2-3 は、このコマンドの実行が正常に完了したときに作成されるプロキシ・テーブルのカラムに関する説明です。

表 2-3: プロキシ・テーブルのカラム

カラム名	データ型	説明
id	numeric(24) ― 32 ビット・マシンの場合 numeric(36) ― 64 ビット・マシンの場合	st_dev と st_ino によって取得した値から成る ID 値を示す。これら 2 つの値は、最初に単一の文字列 (フォーマット：“%d%014ld”) に変換され、次にこの文字列が数値に変換される。
filename	varchar(n)	'pathname' で指定されたディレクトリ、またはパス名の下位のディレクトリ内のファイルの名前。ファイル名の全長 (n) は 255 バイトに制限される。
size	int	正規ファイルの場合 ― ファイルのバイト数を指定する。 ディレクトリの場合 ― ブロック特殊または文字特殊。これは未定義。
filetype	varchar(4)	ファイル・タイプ ― 有効な値は、FIFO (パイプ・ファイル)、DIR (ディレクトリ)、CHRS (文字特殊ファイル)、BLKS (ブロック特殊ファイル)、REG (通常のファイル)、UNKN (その他すべてのファイル・タイプ)。リンクは自動的に展開され、独立したファイル・タイプとして表示されない。
access	char(10)	アクセス・パーミッション。ほぼ標準的な UNIX フォーマット “drwxrwxrwx” で表される。
uid	varchar(n)	ファイル所有者の名前。n の値は、システム定義の L_cuserid で指定され、すべての UNIX システムでは 9 である。Windows システムでは 0 になる。
gid	varchar(n)	所有グループの名前。n の値は、システム定義の L_cuserid で指定され、すべての UNIX システムでは 9 である。Windows システムでは 0 になる。
atime	datetime	ファイル・データが最後にアクセスされた日時。
mtime	datetime	ファイルが最後に修正された日時。
ctime	datetime	ファイルのステータスが最後に変更された日時。
content	image	ファイルの実際の物理的内容 (正規ファイルの場合のみ)。ファイルが正規ファイルでない場合は NULL。

ファイル・システム・ディレクトリにマップされるプロキシ・テーブルは、次の SQL コマンドをサポートできます。

- select − select コマンドを使用すると、ファイルの属性と内容をプロキシ・テーブルから取得できる。テキスト値を処理できる組み込み関数は、content カラム (textptr、textvalid、patindex、pattern) で完全にサポートされる。
- insert − insert コマンドを使用すると、新しいファイルまたはディレクトリを作成できる。意味のあるカラムは、filename、filetype、content のみである。その他のカラムは、insert 文には指定しない。指定しても無視される。ファイル・タイプが *DIR* の場合、新しいディレクトリが作成されるため、content カラムは無視される。

  新しいディレクトリを作成するには、次のように入力します。

  ```
  insert D1 (filename, filetype) values ("newdir", "DIR")
  ```

  新しいファイルを作成するには、次のように入力します。

  ```
  insert D1 (filename, content) values ("newdir/newfile",
  "This is an example.")
  ```

- delete − delete コマンドを使用すると、ファイルまたはディレクトリを削除できる。ディレクトリを削除できるのは、ディレクトリが空の場合のみである。次に例を示す。

  ```
  /* delete the files only */
  delete D1 where filename = 'newdir/newfile'
  /* deletes the directory (if empty) */
  delete D1 where filetype = 'DIR' and filename = 'newdir'
  ```

- update − update コマンドを使用すると、ファイル名だけを変更できる。

注意　一部のファイル・システムでは、削除の後に新しいディレクトリ・エントリを作成して update を実装する場合があるため、更新が制限されていない場合は同じファイル名が複数回更新される場合があります。update の where 句にファイル名を含めて、特定のファイルの更新を修飾することをおすすめします。たとえば、次の文を使用すると、更新が複数回行われます。

```
update t1 set filename=filename + 'old' where filetype
= 'REG'
```

この問題を回避するには、"and filename like "%.c" のような句を追加します。

- readtext − readtext コマンドを使用すると、ファイルの内容を取り出すことができる。
- writetext − readtext コマンドを使用すると、ファイルの内容を修正できる。

これら以外の SQL コマンドは、プロキシ・テーブルを処理できません。

正規ファイルの内容は、Adaptive Server プロセスでファイルにアクセスして読み込むために必要な権限が付与されていて、ファイル・タイプが「通常のファイル」を示している場合にのみ使用できます。このほかの場合、content カラムは常に NULL になります。次に例を示します。

```
select filename, size, content
    from directory_table
    where filename like '%.html'
```

上の構文は、Adaptive Server プロセスでファイルへのアクセス権限が与えられている場合、サフィックス ".html" を持つ正規ファイルの名前、サイズ、内容を返します。アクセス権限が与えられていない場合、content カラムは NULL になります。

ディレクトリのパス名によって参照されるパス名がディレクトリでない場合や、Adaptive Server Enterprise プロセスで検索できない場合、create proxy_table は失敗します。

トレース・フラグ 11206 がオンの場合、ディレクトリの内容に関する情報と、その情報を取得するのに必要なクエリ処理手順を含むメッセージが、エラー・ログに書き込まれます。

下位ディレクトリへの再帰

create proxy_table 文で指定されているパス名に *;R* 拡張子が付いている場合、コンポーネント統合サービスは、パス名の下位ディレクトリをすべて検索し、個々の下位ディレクトリの内容に関する情報を返します。これが完了すると、クエリで返されたファイル名には、パス名に関連するファイルの完全な名前が入ります。つまり、すべての下位ディレクトリの名前がこのファイル名に表示されます。たとえば、パス名が次のように *"/work;R"* と指定されているとします。

```
create proxy_table d1 external directory at "/work;R"
select filename, filetype from d1
```

下位ディレクトリのファイルに関して返される値を表 2-4 に示します。

表 2-4: ファイルの値

ファイル名	ファイル・タイプ
dir1	DIR
dir1/file1.c	REG
dir1/file2.c	REG
dir2	DIR
dir2/file1.c	REG

ファイル・アクセス

Adaptive Server で使用できるもう 1 つのプロキシ・テーブルのクラスでは、ファイル・システム内の個々のファイルに SQL でアクセスできます。サポートされる構文は次のとおりです。

```
create proxy_table <table_name>
    external file at " pathname" [column delimiter “<string>”
```

このコマンドを使用すると、名前が “record” で、タイプが varchar(255) の 1 つのカラムを持つプロキシ・テーブルが作成されます。この場合、ファイルの内容が可読文字で、ファイル内の個々のレコードが改行文字 (¥n) で区切られていることを前提とします。

また、create [existing] table コマンドを使用して、独自のカラム名とデータ型を指定することもできます。

```
create existing table fname (
    column1  int  null,
    column2  datetime null,
    column3  varchar(1024)  null
    etc. etc.
) external file at "pathname" [column delimiter “<string>”
```

カラムは、text、image、Java ADT 以外のどのデータ型でもかまいません。existing キーワードの使用は任意であり、文の処理に影響しません。パス名で参照されているファイルが存在しない場合は、作成されます。ファイルが存在する場合、ファイルの内容は上書きされません。create table コマンドと create existing table コマンドの動作はまったく同じです。

プロキシ・テーブルがファイルにマップされるとき、ファイルとその内容は、次のことを前提とします。

- ファイルは、ディレクトリ・ファイル、ブロック特殊ファイル、または文字特殊ファイルではない。
- Adaptive Server プロセスは、少なくともファイルに読み込みアクセスできる。ファイルが作成される場合、サーバ・プロセスは、ファイルの作成先のディレクトリに書き込みアクセスできる。
- 既存のファイルの内容は、人間が判読できる形式である。
- ファイル内のレコードが改行文字で区切られている。
- サポートされる最大レコード・サイズは 32767 バイトである。
- 個々のカラムは、最後のカラムを除いて、最長 16 バイトの column delimiter 文字列によって区切られている。デフォルトは単一のタブ文字である。
- ファイルの各レコード内のデリミタ値とプロキシ・テーブル内のカラムが対応している。

プロキシ・テーブルがファイルにマップされると、次の操作を実行できます。

- select/into または insert/select を使用して、データベース・テーブルをファイル・システムにバックアップできる。insert 文が処理されると、各カラムは、サーバのデフォルト文字セットの文字に変換される。変換結果はバッファされ、最後のカラムを除くすべてのカラムが 1 個のタブで区切られる。最後のカラムは改行文字で終了する。次に、バッファがファイルに書き込まれ、1 つのローのデータを表示する。
- bcp in と bcp out の代わりに使用できる SQL 構文を提供できる。select/into 文を使用すると、簡単に、テーブルをファイルにバックアップしたり、ファイルの内容をテーブルにコピーしたりできる。
- select 文を使用してファイルの内容を照会し、必要に応じて探索引数や関数を使用してローを特定できる。たとえば、次の文では、Adaptive Server のエラー・ログ・ファイルの個々のレコードを読み込むことができる。

  ```
  create proxy_table errorlog
        external file at "/usr/sybase/ASE15_0/install/errorlog"
  select record from errorlog where record like "%server%"
  ```

  このクエリは、ファイルから like パターンに一致するすべてのローを返します。ローが 255 バイトよりも長い場合は、トランケートされます。次のように入力すると、255 バイトより長いローを指定できます。

  ```
  create existing table errorlog
  (
        record      varchar(512) null
  )
  external file at "/usr/sybase/ASE15_0/install/errorlog"
  ```

  この場合、長さが 512 バイトまでのレコードが返されます。プロキシ・テーブルにはカラムが 1 つしかないので、各カラムの実際の長さは、改行文字の有無によって決まります。

ファイル・アクセスでは、select、insert、truncate table 文しかサポートされません。update と delete は、ファイル・プロキシがこれらのコマンドのターゲットの場合、エラーになります。

値をファイルに挿入するときは、すべてのデータがまず char 値に変換され、次にカラム・デリミタで区切られます。

警告！ truncate table は、ファイル・サイズを 0 に設定します。

トレース・フラグ 11206 は、エラー・ログにメッセージのログを記録するために使用されます。これらのメッセージには、ファイル・アクセスに関与するクエリ処理の段階に関する情報が含まれます。

リモート・サーバ

呼び出すローカル・サーバや各リモート・サーバの sysservers テーブルにエントリを追加するには、sp_addserver を使用してください。sp_addserver の構文は次のとおりです。

```
sp_addserver server_name [,server_class [,network_name]]
```

各要素の意味は次のとおりです。

- *server_name* は、サーバを識別するための名前である。
- *server_class* は、サーバのタイプである。各種サーバでサポートされているサーバ・クラスについては、以降の項で説明する。デフォルトは、サーバ・クラス ASEnterprise である。

  注意 コンポーネント統合サービスでは、サーバ・クラス db2 はサポートされません。

- *network_name* は、interfaces ファイルにおけるサーバ名である。この名前は、*server_name* と同じ場合もあれば異なる場合もある。*network_name* は、*physical name* と呼ばれることもある。デフォルトは、*server_name* と同じ名前である。

  注意 ソート順、および大文字と小文字の区別は、サーバ間で同じである必要があります。

サーバ・クラス ASEnterprise

Adaptive Server は、サーバ・クラス ASEnterprise を使用します。コンポーネント統合サービスは、最初にこのクラスのサーバとの接続を確立するときに Adaptive Server のバージョンを判別し、そのバージョンに基づいてサーバの機能を設定します。

サーバ・クラス ASAnywhere

サーバ・クラスが ASAnywhere のサーバは、Adaptive Server Anywhere のインスタンスです。

- Adaptive Server Anywhere 9.0 以降

サーバ・クラス ASIQ

サーバ・クラスが ASIQ のサーバは、Adaptive Server IQ バージョン 12.5 以降です。

サーバ・クラス *direct_connect*

サーバ・クラスが direct_connect のサーバは、インタフェース仕様 direct_connect に準拠した Open Server ベースのアプリケーションです。

サーバ・クラス direct_connect を使用した Open Server ベースのアプリケーションは、Sybase 以外のすべての外部データ・ソースへのアクセスに適しています。

コンポーネント統合サービスを使用する Adaptive Server は、クライアントや Open Server ベースのアプリケーションとの対話を可能にします。たとえば、あるクライアント・アプリケーションは、ネットワークを介して Oracle や Informix などの DirectConnect と対話する Adapter Server 上で CIS と対話します。この場合、DirectConnect とクライアント・アプリケーション間での直接アクセスも可能です。

サーバ・クラス *sds*

『Adaptive Server Specialty Data Store Developer's Kit』マニュアルに説明されているように、サーバ・クラスが sds のサーバは、Specialty Data Store™ のインタフェース要件に準拠しています。Specialty Data Store は、Adaptive Server とのインタフェースとしてユーザが設計する Open Server アプリケーションです。

サーバ・クラス RPCServer

RPCServer を指定して設定されたサーバは、RPC 以外のものを処理できません。このクラスのサーバは分散トランザクションに関与できないため、コンポーネント統合サービスは、このクラスで設定されたサーバに SQL 文を送信しません。

RPC を Backup Server または XP Server に送信するには、sp_serveroption negotiated logins と server logins を有効にする必要があります。

一般にこのクラスのサーバは、カスタマ作成の Open Server アプリケーションで、カスタマイズしたオペレーションを実行する、または 1 つ以上の RPC の結果生成されたデータを Adaptive Server アプリケーションで使用できるようにするためのものです。この種類のアプリケーションに必要な Open Server のイベント・ハンドラは、次のとおりです。

- SRV_CONNECT − CIS またはクライアント・アプリケーションからのログイン要求を処理および認証する。
- SRV_DISCONNECT − CIS またはクライアント・アプリケーションからの切断要求を処理する。
- SRV_ATTENTION − CIS またはクライアント・アプリケーションからのキャンセル要求を処理する。

- SRV_RPC － CIS からの RPC を処理する。RPC の処理により結果セットが生成される場合がある。この結果セットは、CIS が RPC を代行転送した ASE クライアントに対して、CIS によって転送される。

このサーバ・クラスを使用して、RPC にマップする CIS プロキシ・テーブルをサポートする Open Server アプリケーションを作成することができます。

```
create existing table myRPCtable
      (
            <column description(s)
      )
      external procedure at 'myRpcServerName...rpcname'
```

接続管理

クライアントのためにリモート・サーバに接続する場合、コンポーネント統合サービスは Client-Library 関数を使用します。あるクライアントのリモート・サーバへの最初の接続が確立されると、その接続はクライアントがコンポーネント統合サービスから切断されるまでオープンの状態を維持します。

注意　接続は、コンポーネント統合サービスの機能を初めて使用するときに 1 つの Adaptive Server エンジンに関係付けられます。PSS フラグの POMNI_AFFINITY_SET が設定され、自動的にクリアされません。

interfaces ファイルを使用しないでリモート・サーバに接続する

ディレクトリ・サービスの *ldap* ファイルまたは *interfaces* ファイルの対応するエントリを使用しないで、リモート・サーバへの接続を確立できます。このとき、コンポーネント統合サービスは CT-Library の接続プロパティ CS_SERVERADDR を使用します。このプロパティでは次の形式でサーバを指定できます。

```
"hostname.domain.com:99999"
"hostname:99999"
"255.255.255.255:99999"
```

99999 はポート番号です。*hostname* には簡易名、IP アドレス、または完全なドメイン名を指定します。

ネット名引数を付けた sp_addserver を使用して、次のフォーマットで名前を入力します。

```
sp_addserver S1, ASEnterprise, "myhost.sybase.com:11222"
```

または

```
sp_addserver S1, ASEnterprise, "192.123.321.101:11222"
```

ネット名のこの使用方法には、次のような制限があります。

- Adaptive Server サイト・ハンドラでこの構文が認識されない。
- Replication Agent スレッドでこの構文が認識されない。

interfaces ファイルまたは LDAP サーバが設定されていても、この構文が使用されると、CT-Library はディレクトリ・サービスで接続情報を検索しようとしません。

SSL が設定されており、サーバ・ドキュメントの SSL セクションへのポインタがある場合、オプションとして次の SSL 構文を使用できます。

```
"hostname.domain.com:99999:SSL"
"hostname:99999:SSL"
"255.255.255.255.99999:SSL"
```

Adaptive Server で SSL を設定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「第 19 章データの機密保持」を参照してください。

LDAP ディレクトリ・サービス

LDAP ディレクトリ・サービスは、クライアントとサーバの両方で interfaces ファイルを使用することがもはや不要であることを意味します。Adaptive Server では、サーバ情報を取得するために LDAP サービスをサポートしており、コンポーネント統合サービスがサーバ情報を取得します。リモート・サーバへの接続が試行されると、sp_addserver のネット名引数にコロン (:) が含まれないかぎり、コンポーネント統合サービスは interfaces ファイルまたは LDAP サーバを参照するように Open Client ソフトウェアに指示します。

コンポーネント統合サービスは、設定ファイル (*libtcl.cfg*) で LDAP サービスが指定されているときにのみ LDAP サービスを使用します。*libtcl.cfg* のファイル・パスは、*$SYBASE/$SYBASE_OCS/config/libtcl.cfg* (64 ビット・アプリケーションの場合は *$SYBASE/$SYBASE_OCS/config/libtcl64.cfg*) です。

注意 LDAP サーバが *libtcl.cfg* に指定されていると、サーバ情報は LDAP サーバからのみアクセスできるようになり、Adaptive Server とコンポーネント統合サービスは (従来の) interfaces ファイルをすべて無視します。

SSL とのセキュア通信

SSL を使用すると、コンポーネント統合サービスから、SSL プロトコルをサポートする任意の数のリモート・サーバへのセキュア接続を確立できます (Adaptive Server といくつかの DirectConnect)。

コンポーネント統合サービスは SSL 接続を次のように処理します。

- trusted ルート・ファイルのロケーションを設定する。現在のサーバが SSL に対応している場合、コンポーネント統合サービスのアウトバウンド接続は、常に Adaptive Server Enterprise と同じ trusted ルート・ファイルを使用する。
- 現在のサーバが SSL に対応している場合、SSL 対応リモート・サーバからのチャレンジへの応答に使用される Open Client コールバックを定義するために、接続のプロパティが設定される。現在のサーバが SSL に対応していない場合、SSL 対応リモート・サーバへのコールバックのための接続は失敗する。

trusted ルート・ファイル

trusted ルート・ファイルには、システムに適切に追加されたときにローカル・サーバが trusted と見なす他のサーバの証明書があります。$SYBASE_CERT が定義されている場合、trusted ルート・ファイルはローカル・サーバ (Adaptive Server) からアクセスでき、次のロケーションにあります。

```
$SYBASE_CERT/trusted.txt
```

$SYBASE_CERT が定義されていない場合は、次のようになります。

```
$SYBASE/$SYBASE_ASE/certificates/servername.txt
```

UNIX プラットフォームの場合：

```
%SYBASE%¥%SYBASE_ASE%¥certificates¥servername.txt
```

Windows プラットフォームの場合：

servername は現在の Adaptive Server の名前です。

Kerberos を使用したセキュア通信

Kerberos のネットワーク・ベースのユーザ認証機能とは、Kerberos システムを使用して認証された Kerberos クライアントが Kerberos 認証をサポートするアプリケーションに接続できるシングル・サインオン機能です。格納された一元化パスワードが 1 つあれば、Kerberos をサポートするアプリケーションに接続するためにパスワードを指定する必要はありません。

Adaptive Server がサポートする Kerberos バージョン 5 には、クレデンシャル委任やチケット転送と呼ばれる機能があります。これらを利用すると、サーバ接続時に Kerberos クライアントによるクレデンシャル委任が可能になり、今後、その他のサーバに接続したときに Kerberos クライアントの代わりにサーバが Kerberos の認証を開始できるようになります。

クレデンシャル委任機能は、現在 MIT Kerberos GSSAPI ライブラリのバージョン 4.x 以降のみで認定されています。クライアントは、Adaptive Server に接続する前に、(UNIX システムの kinit -f オプションを使用して) Kerberos システムから委任可能なクレデンシャルを取得しなければなりません。

Adaptive Server に接続されている Kerberos クライアントは、Kerberos クレデンシャル委任機能を使用した CIS を介したリモート Adapter Server への一般的な分散クエリ処理要求で、Adaptive Server のリモート・プロシージャ・コールを要求できます。Kerberos 認証は、サイト・ハンドラベースのリモート・サーバ接続ではサポートされていません。

Kerberos 統一化ログインを使用するため、システム・セキュリティ担当者は次のコマンドを使用して、リモート Adaptive Server の CIS の Kerberos セキュリティ・メカニズムを有効にすることができます。

```
sp_serveroption [server, optname, optvalue]
```

たとえば、Kerberos メカニズムを使用して現在のログイン・ユーザが認証されると、ローカル・サーバ S1 で実行される次のコマンドにより、リモート・サーバ S2 への接続について Kerberos 認証が有効になります。

```
sp_serveroption s2, "security mechanism", csfkrb5
```

Kerberos セキュリティ・サービス

リモート Adaptive Server との接続について Kerberos セキュリティ・メカニズムが有効になると、Kerberos が提供する 1 つまたは複数の次のセキュリティ・サービスが使用できます。

- メッセージの機密保持

  データはネットワーク上で暗号化され、不正な開示から保護されます。

- メッセージの整合性

  トランスポート時に通信が改ざんされていないかどうかを検証します。

- 相互認証

クライアントとサーバの身元を検証します。リモート接続を開始しているローカル・サーバは、Adaptive Server を対象とするすべてのリモート接続要求の相互認証を要求できます。これにより、クライアントはリモート・サーバの身元を検証できます。

注意 Kerberos が提供するオプション・セキュリティ・サービスは、デフォルトでは有効ではありません。

メッセージの機密保持

ローカル・サーバ S1 で実行される次のコマンドは、すべての接続のメッセージの機密性を Kerberos 認証を使用したリモート・サーバ S2 に設定します。

```
sp_serveroption s2, "use message confidentiality", true
```

メッセージの整合性

ローカル・サーバ S1 で実行される次のコマンドは、すべての接続のメッセージの整合性を Kerberos 認証を使用したリモート・サーバ S2 に設定します。

```
sp_serveroption s2, "use message integrity", true
```

メッセージ認証

ローカル・サーバ S1 で実行される次のコマンドは、すべての接続の相互認証を Kerberos 認証を使用したリモート・サーバ S2 に設定します。

```
sp_serveroption s2, "mutual authentication", true
```

リモート Adaptive Server Kerberos プリンシパル名の設定

Adaptive Server のプリンシパル名は、デフォルトのサーバ名です。Adaptive server のプリンシパル名とサーバ名は同じでなくても構わないため、システム・セキュリティ担当者はそれぞれのリモート・サーバのサーバ・プリンシパル名を指定できます。

次のコマンドで、リモート・サーバ S2 のリモート Adaptive Server プリンシパル名を指定します。

```
sp_serveroption S2, "server principal", ase2@myrealm.com
```

コンポーネント統合サービスのリモート・プロシージャ・コールの設定

CIS は永続的な Client-Library 接続を使用して RPC 要求が処理されます。CIS は、RPC の要求先のサーバへの Client-Library 接続がクライアントにすでに確立されているかどうかを判断し、アウトバウンド RPC を処理します。接続が存在しない場合は確立します。

CIS RPC 処理メカニズムを有効にするには、設定オプションの cis rpc handling を 1 に設定します。有効にならない場合、この機能を使用するには、Kerberos のユーザは現在のセッションの CIS RPC を一時的に有効にする必要があります。

次のコマンドは、現在のログイン・セッションの CIS RPC 処理を有効にします。

```
set cis_rpc_handling on
```

コンポーネント統合サービスのリモート・プロシージャ・コールのセキュリティの設定

CIS 接続へのすべてのタイプの Adaptive Server で Kerberos 認証を有効にする方法を次に示します。

次の例では、user1 は Adaptive Server S1 にログインしてリモート Adaptive Server S2 に RPC を要求する Kerberos のユーザです。

1 interfaces ファイルまたはディレクトリ・サービスに、両方のサーバ S1 および S2 のエントリと、Kerberos セキュリティ・メカニズムの secmech 行を追加します。

2 Kerberos ユーザのログインが存在しない場合は追加します。

```
create login user1 with password pwuser1
```

3 設定オプションをオンにして、セキュリティ・メカニズムの使用を有効にします。

```
sp_configure "use security services", 1
```

4 ローカル・サーバ S1 で、リモート・サーバ S2 への CIS の Kerberos 認証を有効にします。

注意 リモート・サーバ S2 は S1から CIS コマンド要求のみを受け取ると仮定します。ただし、S2 も他のサーバに対して CIS コマンドを要求でき、Kerberos 認証を有効にする必要がある場合、S2 には同様の設定が必要になります。

a ローカル・サーバ S1 で、リモート・サーバ S2 を追加します。

```
sp_addserver S2
```

b S2 へのアウトバウンド RPC 要求のため、S1 で Kerberos セキュリティ・メカニズムを有効にします。次のコマンドは、現在のログイン・セッションの CIS RPC 処理を有効にします。

```
sp_serveroption S2, "security mechanism", csfkrb5
```

- リモート・サーバ S2 への RPC 要求のセキュリティ・メカニズムの認証は、上記のコマンドを使用してセキュリティ・メカニズムがリモート・サーバ S2 に設定されている場合にのみ開始される。
- S2 へのセキュリティ・メカニズムが設定されていて、ユーザ 'user1' の転送チケットがローカル・サーバ S1 と使用できない場合、RPC 接続は開始されない。
- セキュリティ・メカニズムがリモート・サーバ S2へのRPC に設定されていない場合、つまり、サーバ・オプション security mechanism が設定されていない場合、リモート・サーバ S2 への RPC のセキュリティ・セッションの確立は開始されない。この場合、CIS は、リモート・サーバ S2 によるパスワードに基づく認証を開始する。外部ログイン / パスワード・マッピングが指定されない場合、CIS はリモート・サーバへの接続時には通常、クライアント名とパスワードを使用する。Kerberos 統一化ログイン認証を使用して Adaptive Server にログインしたクライアント user1 には、ログイン・セッションに関連付けられたパスワードがない。システム管理者は、ユーザ user1 が RPC 要求で S2 に接続できるよう、sp_addexternlogin を使用してログイン・マッピングを設定しなければならない。外部ログインとパスワード・マッピングが user1 に指定されている場合、パスワードは空白になり、リモート・サーバへの CIS 接続に失敗する。
- user1 は、次のようにサーバ S1 を接続してクレデンシャル委任を要求し、S2 のストアド・プロシージャ sp_who の実行を要求できる。

```
isql -Vd -S S1
S2...sp_who
```

- user1 は、次のコマンドを使用して、リモート・サーバ S2 へのパススルー接続を行うことができる。

```
connect to S2
```

- user1 は、次のコマンドを使用して、リモート・サーバ上でクエリを実行できる。

```
sp_remotesql S2, "select @@authmech"
go
sp_remotesql S2, “sp_who”
go
```

セキュリティに関する情報

リモートの Adaptive Server との接続を確立する場合、cis_rpc_handling または set transactional_rpc のいずれかが on になっていると、サイト・ハンドラの代わりに Client-Library 関数が使用されます。この方法では、リモート・サーバが、これらの接続を他のクライアントとの接続と区別できなくなります。そのため、リモート・サーバで設定された、特定のサーバからの接続を許可または禁止するリモート・サーバのセキュリティは有効になりません。

コンポーネント統合サービスが有効になっているもう 1 つの Adaptive Server では、リモート・サーバとの接続に *trusted* モードを使用できません。このため、コンポーネント統合サービスとともに使用する場合は、接続する可能性のあるすべてのユーザ･アカウントを Adaptive Server に設定しなければなりません。

パスワードは暗号化されて内部に保管されます。

CIS での暗号化カラムの使用

デフォルトでは､暗号化および復号化はリモート Adaptive Server によって処理されます。CIS では、リモート Adaptive Server 上に暗号化カラムがないかどうかが一括チェックされます｡リモート Adaptive Server で暗号化がサポートされる場合、CIS によって、暗号化カラムに関連するメタデータでローカルの syscolumns カタログが次のように更新されます。

- create proxy_table を実行すると、リモート・テーブルの任意の暗号化カラムの情報によって syscolumns が自動的に更新される。
- create existing table を実行すると、リモート・テーブルの任意の暗号化カラム・メタデータによって syscolumns が自動的に更新される。encrypt キーワードは create existing table の *column_list* では使用できない。CIS は、リモート・テーブルで暗号化カラムを検出すると、そのカラムが暗号化されていることを自動的にマークする。
- 暗号化カラムがある場所では create table は使用できない。
- alter table は、プロキシ・テーブルの暗号化カラムでは使用できない。
- select into existing を実行すると、ソースからプレーン・テキストが取得され、ターゲットのテーブルに挿入される。その後、ローカル Adaptive Server がプレーン・テキストを暗号化してから、暗号化カラムに挿入する。

次のカラムは、リモート・サーバの syscolumns カタログによって更新されます。

- encrtype － ディスク上のデータ型
- encrlen － 暗号化データの長さ
- status2 － カラムが暗号化されていることを示すステータス・ビット

リモート・サーバ・ログイン

リモート・ログインを完全にサポートできるように、Client-Library にはコンポーネント統合サービスが *server connection* を要求できるようになる新しい接続プロパティがあります。この接続は、受信サーバで (通常のクライアント接続ではなく) サーバ接続として認識されます。これにより、リモート・サーバは、接続がサイト・ハンドラによって作成されたかのように、sysremotelogins を使用して接続を検証できるようになります。

サーバ接続は自動的には有効になりません。代わりに、SSO または DBA が sp_serveroption を実行して要求する必要があります。

```
exec sp_serveroption <server_name>,
    'server login', true | false
```

現在のサーバの @@servername グローバル変数が NULL の場合、server login プロパティを変更することはできません。

server login オプションが true の場合、コンポーネント統合サービスは、Client-Library 接続プロパティを使用して、指定したサーバへの接続を確立します。

クライアント・アプリケーションで指定したリモート・パスワードは、変更されずにリモート・サーバに渡されます。サーバ・ログイン内のリモート・パスワードの設定方法と規則は、サイト・ハンドラと同じです。

これらの接続プロパティは、次の場合だけ設定されます。

- サーバ・オプション server login が true に設定されている。
- リモート・サーバが、サーバ・クラス ASEnterprise で設定されている。
- コンポーネント統合サービス対応サーバにローカル・サーバ名が定義されている (つまり、クエリ select @@servername が NULL 以外を返す)。

trusted モード

trusted モードは、“server logins” がリモート・サーバで設定されている場合にコンポーネント統合サービス接続で使用できます。

Backup Server と XP Server への接続

Adaptive Server 12.5.1 以降、コンポーネント統合サービスは RPC を Backup Server または XP Server に送信できるようになりました。送信するには、次のようにサーバ・オプション negotiated logins を有効にします。

```
exec sp_serveroption server_name, "negotiated logins", true
```

これで、コンポーネント統合サービスが、これらいずれかの Sybase 提供サーバで開始されたログイン・チャレンジに応答できます。

外部ログインのマッピング

コンポーネント統合サービスを呼び出す Adaptive Server のユーザには、リモート・サーバのログイン名とパスワードが必要です。コンポーネント統合サービスがリモート・サーバに接続するために使用するユーザ名とパスワードの組み合わせのデフォルトは、Adaptive Server に接続するためにクライアントが使用するユーザ名とパスワードと同じです。

コンポーネント統合サービスでは、Adaptive Server のログイン名とパスワードと、リモート・サーバのログイン名とパスワードの 1 対 1 のマッピングがサポートされています。たとえば、ストアド・プロシージャ sp_addexternlogin を使用すると、次のように Adaptive Server のユーザ steve、パスワード sybase を、Oracle のログイン名 login1、パスワード password1 にマップできます。

```
sp_addexternlogin Oracle, steve, login1, password1
```

Adaptive Server バージョン 12.5 以降では、次のように多対 1 のマッピングを指定して、Oracle に接続する必要がある Adaptive Server ユーザ全員に同じユーザ名とパスワードを割り当てることができます。

```
sp_addexternlogin Oracle, NULL, login2, password2
```

1 対 1 のマッピングの方が優先度が高いので、ユーザ steve に Oracle の外部ログインが割り当てられている場合は、多対 1 ではなく、1 対 1 のマッピングが使用されます。

このほかに、外部ログインを Adaptive Server の役割に割り当てることができます。この機能により、一定の役割を持つユーザなら誰にでも、所定のリモート・サーバの対応するログイン名／パスワードを次のように割り当てることができます。

```
sp_addexternlogin Oracle, null, login3, password3, rolename
```

役割名は、ユーザ名ではなく役割の名前です。この役割が有効なユーザが Oracle に接続する必要がある場合は、その役割に見合ったログイン名／パスワードが、接続を確立するために使用されます。複数の役割が有効なユーザがリモート・サーバへの接続を確立する場合は、役割ごとに外部ログインのマッピングが検索され、検索された最初のマッピングを使用してログインを確立します。この順序は、sp_activeroles によって表示される順序と同じです。

sp_addexternlogin の一般的な構文は、次のとおりです。

```
sp_addexternlogin
      <servername>,
      <loginname>,
      <external_loginname>,
      <external_password>
      [, <rolename>]
```

<*rolename*> はオプションです。これを指定すると、*loginname* が無視されます。

これらの機能の優先度は次のとおりです。

- 1 対 1 のマッピングが定義されている場合は、1 対 1 のマッピングが使用される。
- 1 対 1 のマッピングが未定義で、役割が有効であり、そのマッピングを見つけることができる場合は、役割のマッピングを使用してリモート接続が確立される。
- 以上のいずれも該当しない場合に多対 1 のマッピングが定義されていれば、多対 1 のマッピングが使用される。
- 以上のいずれも該当しない場合は、Adaptive Server のログイン名とパスワードを使用して接続する。

役割のマッピングが行われてからユーザの役割が (set role によって) 変更された場合、役割のマッピングを使用して行われたリモート・サーバへの接続はいずれも切断されます。

sp_helpexternlogin は更新され、sp_addexternlogin によって追加された各種の外部ログインを表示できるようになりました。sp_helpexternlogin の構文は次のとおりです。

```
sp_helpexternlogin [<servername> [,<loginname> [,<rolename>]]]
```

3 つのパラメータはすべてオプションであり、いずれも NULL でかまいません。

ストアド・プロシージャ sp_dropexternlogin には *<rolename>* 引数も指定できます。*<role name>* を指定した場合、2 番目の引数 *<login name>* は無視されます。

リモート・サーバ接続フェールオーバ

interfaces ファイル (または LDAP ディレクトリ・サービス) でフェールオーバ設定を定義している場合、コンポーネント統合サービスはその設定を利用して、プライマリ・サーバへの接続が失敗した場合にフェールオーバ・サーバへの接続を自動的にフェールオーバします。

次の設定手順を実行した後で、フェールオーバ用としてリモート・サーバを設定できます。

1　新しいサーバ・オプション cis hafailover を有効にする。

```
exec sp_serveroption server_name, 'cis hafailover', true
```

2　ディレクトリ・サービス (interfaces ファイルまたは LDAP サーバのサーバ・エントリ) を修正して、フェールオーバ・サーバを指定する。

たとえば、サーバ S2 をサーバ S1 のフェールオーバ・サーバとして設定でき、その逆も可能です。この次に例を示します。

```
S1
        master tcp ether host1 8000
        query tcp ether host1 8000
        hafailover S2
S2
```

```
master ether host2 9000
query ether host2 9000
hafailover S1
```

CS_HAFAILOVER 接続プロパティの詳細については、『高可用性システムにおける Sybase フェールオーバの使用』の付録 C を参照してください。cis hafailover サーバ・オプションが true の場合、コンポーネント統合サービスは ct_con_props() API を使用してこのプロパティを設定します。

リモート・サーバの機能

Adaptive Server は、クラス sds または direct_connect のリモート・サーバとの接続を初めて確立したときに、sp_capabilities という名前の RPC を発行し、結果セットが返されることを想定します。この結果セットには、リモート・サーバの機能面の能力が記述されているので、コンポーネント統合サービスはリモート・サーバとの対話を調整し、使用可能な機能を利用できます。コンポーネント統合サービスは、その機能に応じて、できるだけ多くの構文をリモート・サーバに転送します。

クエリ処理

この項では、コンポーネント統合サービス内でのクエリ処理について説明します。

処理手順

コンポーネント統合サービスが有効になっている場合に実行されるクエリ処理の手順は、Adaptive Server で実行される手順と似ていますが、次の点が異なります。

- クライアント接続がパススルー・モードで確立された場合、Adaptive Server のクエリ処理はスキップされ、SQL テキストがリモート・サーバに実行のために転送される。
- ローカルのプロキシ・テーブルが参照されている場合、select、insert、delete または update 文を実行するためにサーバへ発行したときに、コンポーネント統合サービスはクエリのパフォーマンスを向上させる追加の手順を実行する。

クエリ処理手順の概要を次に示します。

クエリの解析

SQL パーサは、受信した SQL 文の構文をチェックし、実行のために送信された SQL が Transact-SQL パーサによって認識されなかった場合にはエラーを表示します。

クエリの正規化

クエリの正規化では、SQL 文内で参照されている各オブジェクトが検証されます。クエリの正規化では、文内で参照されているオブジェクトが存在し、そのデータ型が文内の値と互換性があるかを検証します。

例

```
select * from t1 where c1 = 10
```

クエリの正規化の段階では、c1 という名前のカラムを持つテーブル t1 がシステム・カタログに存在することを検証します。また、カラム c1 のデータ型が値 10 と互換性があることも確認します。たとえば、カラムのデータ型が datetime の場合、この文は拒否されます。

クエリの前処理

クエリの前処理は、クエリの最適化の準備をします。コンポーネント統合サービスが元の文と構文的に異なる SQL 文を生成する場合もあります。

前処理は、ビューによって参照されるテーブルでクエリが操作できるよう、ビューの拡張を行います。処理を効率化するために、式を並べ替え、サブクエリを変更する手順が実行される場合があります。たとえば、サブクエリの変換により、いくつかのサブクエリがジョインに変換される場合があります。

決定点

前処理後、コンポーネント統合サービスと標準の Adaptive Server クエリ・オプティマイザのどちらで最適化を処理するかが決定されます。

次の場合、コンポーネント統合サービスは、クイックパス・モードという機能を使用して最適化を処理します。

- SQL 文に示されるテーブルはすべて、単一のリモート・サーバ上にある。
- リモート・サーバは、文に示される構文をすべて処理できる。

  コンポーネント統合サービスは、リモート・サーバのクエリ処理機能をそのサーバ・クラスで判別します。たとえば、コンポーネント統合サービスは、サーバ・クラス sql_server として設定されているどのサーバも、すべての Transact-SQL 構文を処理できると想定します。

  サーバ・クラスが direct_connect のリモート・サーバの場合、コンポーネント統合サービスはサーバへ初めて接続したときに、RPC を発行して、リモート・サーバに機能について問い合わせます。コンポーネント統合サービスは、RPC に対するサーバの応答に基づいて、リモート・サーバに転送する SQL 構文を決定します。

- SQL 文は以下の条件を満たす。
  - select、insert、delete、または update 文である。
  - insert、update、または delete 文の場合、identity または timestamp カラム、または参照制約が存在しない。
  - text または image カラムを含んでいない。
  - compute by 句を含んでいない。
  - for browse 句を含んでいない。
  - select...into 文ではない。
  - カーソルに関連する文ではない (たとえば、where current of cursor を含む fetch、declare、open、close、deallocate、update、または delete 文)。
- リモート・サーバ上で開いているカーソルは、リモート接続に含まれません。

上記の条件を満たしていない場合、クイックパス・モードは使用されず、標準の Adaptive Server クエリ・オプティマイザが最適化を処理します。

コンポーネント統合サービスのプランの生成

クイックパス・モードが使用できる場合、コンポーネント統合サービスは簡略化されたクエリ・プランを生成します。文にプロキシ・テーブルが含まれている場合、Adaptive Server のプラン生成フェーズで処理するよりも、リモート・サーバで処理する方がより速く実行されます。

Adaptive Server の最適化とプランの生成

Adaptive Server の最適化とプランの生成は、クエリ実行の最適なパスを評価し、Adaptive Server にクエリの実行方法を伝えるクエリ・プランを作成します。

クエリ内のテーブルに対して update statistics コマンドを実行した場合、オプティマイザは、ジョイン順序を決定するための十分なデータを持っています。update statistics を実行していない場合は、Adaptive Server のデフォルトが適用されます。

Adaptive Server の最適化の詳細については、『パフォーマンス&チューニング・ガイド：オプティマイザと抽象プラン』の「第 2 章 オプティマイザの概要」を参照してください。

コンポーネント統合サービスのプランの生成

クイックパス・モードが使用できる場合、コンポーネント統合サービスは、文全体がリモート・サーバにプッシュされる簡略化されたクエリ・プランを生成します。

クイックパス・モードが使用できない場合、Adaptive Server オプティマイザは文全体を実行するプランを生成します。次にこのプランは調査され、その一部が選択されてリモート・サーバにプッシュ・オフされます。テーブルのロケーションとリモート・サーバの性能に基づいて、元のプランのできる限り多くの部分がプッシュ・オフされます。能力の高いリモート・サーバの場合、リモート文は元の文に非常に近くなることもあります。送信できないプランの一部を実行するローカルの Adaptive Server がある他のサーバに、必要最小限の文が生成されます。

たとえば、クライアントから次の文が入力されるとします。

```
select a,b from table1 where cos(a) > 0 and sin(b) > 0
```

table1 を所有するリモート・サーバで cos() はサポートしているが、sin() はサポートしていない場合、リモート・サーバに送信される文は次のようになります。

```
select a,b from table1 where cos(a) > 0
```

ローカル・サーバでは、リモート・サーバから返された結果セットに対して sin(b) > 0 のチェックを行うプランが生成されます。

コンポーネント統合サービスのリモート・ロケーション・オプティマイザ

Adaptive Server は、各テーブルの格納領域のロケーションを考慮せずに、複数テーブルのクエリの最適なジョイン順序を含むクエリ・プランを生成します。クエリ内にリモート・テーブルがある場合、コンポーネント統合サービスは、ロケーションを考慮に入れた追加の最適化を実施します。このため、リモートでジョイン部分を実行できるようにジョイン順序の計画が再配列されることがあります。

統計値

プラン選択をインテリジェントに行うには、プロキシ・テーブルも含め、クエリに関わるすべてのテーブルの統計が必要です。統計は、特定のテーブルで update statistics を実行することで取得できます。

update statistics を実行していない場合は、Adaptive Server のデフォルトが適用されます。Adaptive Server の最適化の詳細については、『パフォーマンス＆チューニング・ガイド：オプティマイザと抽象プラン』の「第 2 章 オプティマイザの概要」を参照してください。

プロキシ・テーブルに対するオプティマイザのコスト・モデル

Adaptive Server オプティマイザは、次のシーケンスを実行するために必要な時間を指定する「ネットワーク交換」単位に基づいて、リモート・サーバへのネットワーク・アクセスのコストを組み込んでいます。

- カーソルを開く。
- 50 個のローをフェッチする。
- カーソルを閉じる。

1 回の交換にかかるコストはユーザによって制御され、sp_serveroption により、サーバごとに次のように指定されます (デフォルトは 1000 ミリ秒)。

```
sp_serveroption <servername>, "server cost", "nnnn"
```

nnnn には、オプティマイザがネットワーク・コストを計算するときに 1 回の交換に使用される時間 (ミリ秒) を表す数字を指定します。

注意 サーバ・コストの上限は 32767 です。この上限を超えると、算術オーバフローが発生します。

---

sp_addserver を使用して新しいサーバを sysservers に追加すると、デフォルトのコスト 1000 ミリ秒がそのサーバの sysattributes に保存されます。sp_serveroption を使用すると、所定のサーバでデフォルトと異なるコストを指定できます。sp_helpserver を使用すると、サーバに関連する現在のネットワーク・コストが示されます。

クエリ・プランの実行

テーブルに影響を与えるコマンドは、サーバによってチェックされ、オブジェクトがローカルまたはリモートのどちらの格納領域のロケーションを持っているかが判断されます。格納領域のロケーションがリモートの場合、要求された操作をリモート・オブジェクトに提供できるように、クエリ・プランが実行されたときに適切なアクセス・メソッドが呼び出されます。リモート格納領域のロケーションにマップされているオブジェクトに対しては、以下のコマンドが作用します。

- alter table
- begin transaction
- commit
- create index
- create table
- create existing table
- deallocate table

- declare cursor
- delete
- drop table
- drop index
- execute
- fetch
- insert
- open
- prepare transaction
- readtext
- rollback
- select
- set
- setuser
- truncate table
- update
- update statistics
- writetext

RPC 処理とコンポーネント統合サービス

コンポーネント統合サービスが有効な場合、サイト・ハンドラかコンポーネント統合サービスを使用して、アウトバウンド RPC (リモート・プロシージャ・コール) を処理できます。各メカニズムについては、次の項で説明します。

サイト・ハンドラとアウトバウンド RPC

Adaptive Server では、出力 RPC は、サイト・ハンドラによって送信されます。サイト・ハンドラは、リモート・サーバへの単一の物理接続を通じて複数の要求を多重化します。RPC は、多重ステップ操作の一部として処理されます。

1 接続の確立 − Adaptive Server のサイト・ハンドラによって、リモート・サーバへの単一の物理接続が確立されます。この物理接続を通じて、RPC ごとに論理接続を確立しなければなりません。論理接続のルートは、対象のリモート・サーバのサイト・ハンドラを経由して指定されます。

   これらの接続要求に対する接続確認処理は、通常のクライアント接続での処理とは異なります。まず、リモート・サーバは、接続要求の発信元のサーバがその sysservers テーブルに設定されているかどうかを判断しなければなりません。設定されている場合は、システム・テーブル sysremotelogins がチェックされ、その接続要求の処理方法を決定します。trusted モードが設定されている場合は、パスワードはチェックされません。trusted モードの詳細については、「trusted モード」(39 ページ) を参照してください。

2 RPC の送信 − RPC 要求は、論理接続を通じて送信されます。

3 処理結果 − RPC の結果はすべて、論理接続からクライアントへ中継されます。

4 切断 − 論理接続が終了します。

論理接続の接続と切断を処理するため、サイト・ハンドラの RPC 処理時間が長くなる場合があります。

コンポーネント統合サービスとアウトバウンド RPC

コンポーネント統合サービスが有効な場合、クライアントは、コンポーネント統合サービスにアウトバウンド RPC の処理を要求する 2 つの方法のいずれかを使用できます。

- アウトバウンド RPC をすべてのクライアントのデフォルトとして処理するようにコンポーネント統合サービスを設定するには、次のコマンドを発行する。

  ```
  sp_configure "cis rpc handling", 1
  ```

  この方法で cis rpc handling 設定パラメータを設定すると、すべてのクライアント接続がこの動作を継承し、アウトバウンド RPC 要求はコンポーネント統合サービスによって処理されます。これは、その後のすべての接続に継承されるサーバ・プロパティです。クライアントは、必要に応じて次のコマンドを発行し、デフォルトの Adaptive Server の動作に戻すことができます。

  ```
  set cis_rpc_handling off
  ```

- アウトバウンド RPC を現在の接続でのみ処理するようにコンポーネント統合サービスを設定するには、次のコマンドを発行する。

  ```
  set cis_rpc_handling on
  ```

  このコマンドを使用すると、現在のスレッドでのみ cis rpc handling が有効になり、ほかのスレッドでの動作には影響しません。

cis rpc handling が有効な場合、アウトバウンド RPC 要求は Adaptive Server のサイト・ハンドラを経由しません。この場合、コンポーネント統合サービスを経由し、そこで永続的な Client-Library 接続を使用して RPC 要求が処理されます。このメカニズムを使用して、コンポーネント統合サービスはアウトバウンド RPC を以下のように処理します。

1　RPC の要求先のサーバへの Client-Library 接続がクライアントにすでに確立されているかどうかを判断します。確立されていない場合は、これを確立します。

2　Client-Library 機能を使用して、RPC をリモート・サーバに送信します。

3　リモート・サーバからの結果を、Client-Library 機能を使用して RPC を発行したクライアント・プログラムに中継します。

RPC は、ユーザ定義のトランザクションに組み込めます。実際に、クライアント用にコンポーネント統合サービスが実行するすべての処理は、単一の接続のコンテキスト内で実行できます。このため、RPC をトランザクションの処理単位に組み込むことができ、RPC の実行する処理を、トランザクション内で実行されるほかの処理とともにコミットまたはロールバックできます。

コンポーネント統合サービスを使用してアウトバウンド RPC 要求を処理すると、以下のような効果もあります。

- Client-Library 接続は永続的であるため、後続の RPC 要求はリモート・サーバへの同一の接続を使用できる。このため、最初の RPC 以外は接続と切断をすべて省略できるので、後続の RPC のパフォーマンスが向上する。
- RPC の実行する処理を 1 つのトランザクションに組み込み、トランザクションの他の処理とともにコミットまたはロールバックできる。このトランザクション指向の RPC の動作は、現時点では、RPC を受信するサーバが別の Adaptive Server か、トランザクション指向 RPC をサポートする DirectConnect である場合にのみサポートされる。
- 接続要求は、通常のクライアント接続としてリモート・サーバに表示される。server logins が有効になっていないかぎり、リモート・サーバでは、通常のアプリケーションの接続と区別できない。sysremotelogins に対して確認が行われず、すべての接続で、接続要求前に確立された有効な Adaptive Server のログイン・アカウントが必要なため (この場合、trusted モードは使用できない)、Adaptive Server のリモート・サーバ管理機能に影響がある。

RPC のテキスト・パラメータ

Adaptive Server は、サイズの大きいデータ・チャンクを単一のリモート・プロシージャ・コールで送信できます。この機能では、一定のパラメータをテキスト・ポインタとして処理した後、これらのテキスト・ポインタを参照解除して、関連付けられたテキストの値を取得します。text データは、Adaptive Server では 16K、その他すべてのサーバでは 32K のチャンクにパッケージ化され、RPC のパラメータとして Client-Library に渡されます。

テキスト・ポインタは、binary(16) または varbinary(16) 型のパラメータとして識別されます。各テキスト・ポインタのパラメータで参照されるテキストの値は、RPC が実行されたときに取得され、Adaptive Server では 16K、その他すべてのサーバでは 32K のチャンクに拡張されて、それぞれ CS_LONGCHAR_TYPE 型のパラメータとして Client-Library に渡されます。

この動作は、次の set コマンドによって行われます。

```
set textptr_parameters ON
```

RPC が要求されると (cis_rpc_handling はオンでなければなりません)、テキスト・ポインタがコンポーネント統合サービス・レイヤ内で参照解除され、テキストの値を使用して Client-Library 用のパラメータが作成されます。

この処理を適切に実行するには、テキスト・ポインタの前でパス名の引数を宣言します。この引数によって、テキスト・ポインタの派生元のテーブルが識別されます。次に例を示します。

```
declare @pathname varchar(90)
declare @textptr1 binary(16)
declare @textptr2 binary(16)
select @pathname = "mydatabase.dbo.t1",
            @textptr1 = textptr(c1),
            @textptr2 = textptr(c2)
      from mydatabase.dbo.t1
      where ... (whatever)
set textptr_parameters ON
exec SYBASE...myrpc @pathname, @textptr1, @textptr2
set textptr_parameters OFF
```

‘myrpc’ という名前の RPC がサーバ SYBASE に送信されるとき、*@pathname* パラメータは実際には送信されませんが、textptr の *@textptr1* と *@textptr2* によって参照されるテキスト値の位置を示すために使用されます。

varchar 型のパラメータ *@pathname* は、binary(16) 型のパラメータの直前になければなりません。そうでない場合は、*@textptr1* は通常のパラメータと見なされて、binary(16) 値としてサーバ SYBASE に送信されます。

テキストは 16K または 32K のチャンクに分割されて、それぞれが CS_LONGCHAR_TYPE 型の別のパラメータになります。

@@textsize の現在の値は、無視されます。

このスキームは、リモート・プロシージャにマッピングされたプロキシ・テーブルでも使用できます。次に例を示します。

```
create existing table myrpctable
(
        id int,       -- result column
        crdate datetime, -- result column
        name varchar(30), -- result column
        _pathname varchar(90), -- parameter column
        _textptr1 binary(16), -- parameter column
        _textptr2 binary(16), -- parameter column
) external procedure at 'SYBASE...myrpc'
go
declare @textptr1 binary(16)
declare @textptr2 binary(16)
select @textptr1 = textptr(c1), @textptr2 = textptr(c2)
from mydatabase.dbo.t1 where <whatever>
set textptr_parameters ON
select id, crdate, name
from myrpctable
where_pathname = "mydatabase.dbo.t1" and
        _textptr1 = @textptr1 and
        _textptr2 = @textptr2
```

プロキシ・テーブル myrpctable に対するクエリが処理されると、コンポーネント統合サービスは、'myrpc' という名前の RPC をサーバ 'SYBASE' に送信します。パラメータは、クエリの where 句に含まれる検索引数から派生します。textptr_parameter オプションが ON に設定されているため、前述の RPC の例のように、textptr は CS_LONGCHAR_TYPE 型に拡張されます。

XJS/390 でのテキスト・パラメータのサポート

テキストの大きなブロックを RPC パラメータとして転送できるため、コンポーネント統合サービスが XJS/390 を使用して IBM メインフレームと対話できるようになりました。XJS/390 スクリプト (JavaScript 風の構文) は、Adaptive Server のテーブル (または、プロキシ・テーブル経由でアクセス可能なファイル) 内に保管され、RPC を使ってメインフレームに転送できます。このスクリプトの構文は、XJS/390 機能で解析、実行され、スクリプトの手続き型論理に従って結果セットが生成されます。

次のような機能が有効になります。

- Adaptive Server 内のデータベース・イベントにより、MQ Series メッセージを生成できる。XJS/390 Mscript は、メッセージの生成をサポートするので、RPC をメインフレームに送信して、データベース内でトリガされたイベントに対応してメッセージを生成するよう要求できる。
- コンポーネント統合サービスのユーザは、InfoHub などの他社製ミドルウェアをインストールしなくても、VSAM、IMS、MQ Series などのデータにアクセスできる。

スクリプトを RPC パラメータとして処理するには、XJS/390 のバージョン 2.0 以降が必要です。詳細については、XJS/390 の仕様を参照してください。

分散トランザクション管理

Adaptive Server の分散トランザクション管理は、ローカルの Adaptive Server とコンポーネント統合サービス、およびトランザクションに関係しているすべてのリモート・サーバでのトランザクションのステータスを追跡します。ユーザのアプリケーションがトランザクションをコミットすると、Adaptive Server Transaction Coordinator (ASTC) を使用して、関係しているすべてリモート・サーバにコミットが伝達されます。マルチサイト・トランザクションの管理は ASTC がコンポーネント統合サービスと連携して行います。コンポーネント統合サービスは、各トランザクションに関係する新しいサーバを登録してから、トランザクション調整の制御を ASTC に渡します。ASTC は、コンポーネント統合サービスを呼び出して、トランザクション管理のさまざまなコマンドを実行します。

サーバ・クラスと ASTC

内部的に ASTC は、次のいずれかのタイプと見なします。

- DTM 対応
- Pre-DTM

これらのタイプは、使われる 2 組のコールバックに対応し、表 2-5 で示しているようにサーバ・クラスに対応します。

表 2-5: トランザクション機能

ASTC サーバ・タイプ	コンポーネント統合サービスのサーバ・クラス
DTM 対応	ASEnterprise (12.x 以降) DC/Oracle 12.5 以降
Pre-DTM	ASEnterprise (12.0 より前) ASAnywhere ASIQ その他の DirectConnect sds

注意 分散トランザクションを開始する前に、ローカル・サーバに名前を付ける必要があります。*@@servername* は null にできません。

DTM 対応サーバ

「DTM 対応」のリモート・サーバは、ASTC で有効になる完全な 2 フェーズ・コミット・サービスをサポートしています。このサービスをサポートするサーバでは、ほかのセッションによって開始されたトランザクションを別の接続 (セッション) がコミットまたはロールバックできます。この機能は、コミット・コーディネータ (ASTC) がリモート・サイトに接続し、疑わしいトランザクションをコミットまたはロールバックする必要がある場合に必要です。Adaptive Server 12.0 以降では、DirectConnect for Oracle 12.5 以降と同じくこのサポートがあります。

Pre-DTM サーバ

“pre-DTM” に分類されるリモート・サーバでは、begin tran、commit tran、rollback tran などのトランザクション管理文がサポートされますが、あるセッションが別のセッションで開始されたトランザクションをコミットまたはロールバックする機能はサポートされません。

コンポーネント統合サービスには、pre-DTM サーバでのユーザ・トランザクションを確実に管理するためにさまざまな機能が提供されています。ただし、この機能がどの程度サポートされているかは、サーバに組み込まれているアクセス・メソッドによって異なります。以下で説明している一般的なロジックは、サーバ・クラス ASEnterprise (12.0 より前)、ASAnywhere、ASIQ、direct_connect、sds (Specialty Data Store がトランザクション管理をサポートしている場合) で使用されます。リモート・サーバに関連するトランザクションの管理には、2 フェーズ・コミット・プロトコルを使用します。Adaptive Server は、ほとんどの場合にトランザクションの整合性を確保する方式を実装します。ただし、分散した作業単位が未定の状態で残る可能性があります。2 フェーズ・コミット・プロトコルを使用する場合でも、リカバリ処理は含まれません。ユーザのトランザクションを管理する一般的なロジックは、次のようになっています。

コンポーネント統合サービスは、begin transaction ノーティフィケーション (通知) でリモート・サーバの作業を開始します。トランザクションのコミットの準備が整うと、コンポーネント統合サービスは、トランザクションの一部であったリモート・サーバのそれぞれに prepare transaction ノーティフィケーションを送信します。prepare transaction は、リモート・サーバに ping を実行して、接続がまだ確立されているかどうかを確認します。prepare transaction要求が失敗すると、すべてのリモート・サーバは、現在のトランザクションをロールバックするよう指示されます。prepare transaction 要求がすべて成功すると、サーバはトランザクションに関わるそれぞれのリモート・サーバに、commit transaction 要求を送信します。begin transaction によって開始されたコマンドはすべて、トランザクションを開始できます。その他のコマンドはリモート・サーバに送信され、1 つの、リモートの作業単位として実行されます。

strict DTM enforcement

完全な 2 フェーズ・コミット機能を保証するために、ASTC は strict dtm enforcement という概念を使用します。有効にした場合、トランザクションに pre-DTM サーバを含めようとすると、strict dtm enforcement によりトランザクションはアボートします。

enable xact coordination

ASTC は、設定オプション enable xact coordination を使用します。このオプションを使用することで、リモート・サーバに関するすべてのトランザクションを ASTC で管理できるようになります。デフォルトでは有効になっています。xact coordination を有効にする前に、コンポーネント統合サービスを有効にする必要があります。xact coordination が有効になっている間は、コンポーネント統合サービスを無効にできません。xact coordination を有効にすると、transactional_rpcs は暗黙的に有効になります。

コンポーネント統合サービスを有効にする

ASTC は、リモート・サーバとのすべての通信を処理するのにコンポーネント統合サービスを使用します。ASTC は、デフォルトでは有効になっているため (enable xact coordination)、コンポーネント統合サービスもデフォルトでは有効になっています。

トランザクション指向 RPC

サーバは、現在のトランザクションによって開始された作業単位に RPC を含めることを許可します。

トランザクション指向 RPC を使用する前に、set transactional_rpc on コマンドを発行します。

次に示す構文は、トランザクション内に RPC を含めることをリモート・サーバがサポートできると仮定した場合に、この機能がどのように使用されるかを示しています。

```
begin transaction
 insert into t1 values (1)
 update t2 set c1 = 10
 execute @status = SYBASE.pubs2.dbo.myproc
 if @status = 1
       commit transaction
 else
       rollback transaction
```

この例では、サーバ SYBASE 内でプロシージャ *myproc* によって実行された作業は、begin transaction コマンドで開始された作業単位に含まれています。この例では、成功した場合に、リモート・プロシージャ *myproc* がステータス “1” を返すことを要求しています。アプリケーションは、作業が完全な単位としてコミットされたか、ロールバックされたかを制御します。

RPC を受け取るサーバは、RPC がデータ操作言語 (DML) コマンド (select、insert、delete、update) と同じトランザクションのコンテキストに含むことができなければなりません。これは Adaptive Server でも同じであり、また Sybase がリリースしているほとんどの DirectConnect 製品にも該当すると考えられます。ただし、データベース管理システムには、この機能をサポートできないものがあります。

トランザクション管理の制限

ネストされた begin transaction 文と commit transaction 文がリモート・サーバに関わるトランザクションに含まれている場合は、一番外側の組の文だけが処理されます。begin transaction 文とcommit transaction 文を含む一番内側の組は、リモート・サーバに転送されません。

Adaptive Server に対する Adaptive Server update statistics

update statistics をリモート・サーバのプロキシ・テーブルに対して実行すると、関連するテーブル、インデックス、およびカラムが使用可能であれば、そのテーブル・カタログがローカルの systabstats と sysstatistics にインポートされます。

デフォルトで、プロキシ・テーブルに関する update statistics は必須の統計データをインポートしようとします。しかし、リモート・テーブルに統計データがないか不完全なときは、コンポーネント統合サービス (CIS) が再び以前の統計データ収集機構を使うようになります。

Traceflag 11229 をオンにして CIS を強制的に以前の統計データ収集機構に戻すこともできます。これでデータベースのすべてのデータを取得してから、統計を計算することができます。

注意　これは update statistics がリモート・テーブルで実行されず、統計値が利用できない場合の動作です。

制限事項

キーに関する制限：

- プロキシ・テーブルは、別の Adaptive Server (バージョン 11.9 以降) にマップされている必要がある。
- マップ先が RPC、外部ファイル、またはシステム・テーブルのプロキシ・テーブルは除外される。
- リモート・サーバが Adaptive Server Enterprise version 11.9 以降でないか、そのサーバ・クラスが異なる場合、CIS は以前の機構を使用して統計データの収集を続ける。

Adaptive Server 以外のバックエンドに対する update statistics

update statistics コマンドでは、インデックスのキー値の分散に関する情報が提供され、サーバでクエリを処理するときに最適なインデックスを決定するのに役立ちます。update statistics は、すでにデータを含むテーブル上にインデックスを作成するとき、または再作成するときには、自動的に実行されません。インデックス付きのカラムで大量のデータが追加、変更、削除されるときに、これを使用します。クエリ最適化における重要な要素は、分散統計の正確性です。インデックスのキー値に大きな変更があった場合は、そのインデックスに対して update statistics を再実行してください。

update statistics コマンドは、テーブルの所有者またはシステム管理者だけが発行できます。

構文は次のとおりです。

update statistics *table_name* [*index_name*]

update statistics の実行には多量のリソースが必要なため、指定するテーブルが集中的に使用されていないときに実行してください。update statistics では、データを読み込む間、リモート・テーブルとインデックスをロックします。トレース・フラグ 11209 を使用すると、テーブルはロックされません。

コマンドの実行によってシステムに支障をきたさないように、サイトで都合のよい時間帯に update statistics が自動実行されるように設定できます。詳細については、『パフォーマンス＆チューニング・ガイド：モニタリングと分析』の「第 4 章 パフォーマンス改善のための統計値の使用」を参照してください。

サーバは update statistics コマンドに指定された各インデックスに対し、テーブル・スキャンを実行します。

Transact-SQL ではインデックス名がデータベース内でユニークでなくてもよいため、インデックスが関連付けられているテーブルの名前を指定する必要があります。

update statistics の実行後、sp_recompile を実行し、そのインデックスを使用するトリガとプロシージャで、新しい分散統計が使用されるようにします。

```
sp_recompile authors
```

データベースにおける Java

コンポーネント統合サービスでは、データベース内 Java がサポートされ、リモート・データ・アクセスができます。

ただし、次の制限があります。

- Java は、リモートの Adaptive Server 12.x 以降でのみサポートされる。
- Java は、言語イベントでのみサポートされる (リモート・テーブルには動的 SQL を使用できない)。

リモート・データ・アクセスのために Java を使用する前に、「Java クラス定義」(59 ページ) を参照してください。その後で、Java クラス・ファイルをローカル・サーバにインストールしてから、必要な Java クラス・ファイルをリモート・サーバにインストールしてください。

@@textsize

データは、image データ型フォーマットを使用して直列化 Java オブジェクトとして返され、ローカル・サーバ上で非直列化されます。@@textsize は、直列化オブジェクトを保持できる大きさに設定します。@@textsize のサイズがオブジェクトよりも小さい場合、オブジェクトは切り捨てられ、非直列化できません。

@@stringsize

@@stringsize は、toString() メソッドから返される character データのサイズを指定します。これは、@@textsize と動作がよく似ていますが、Java Object.toString() メソッドから返される char データにのみ適用されます。デフォルト値は 50 です。最大値は 16384 です。値をゼロにすると、デフォルトが使用されます。この値は set コマンドで変更できます。

```
set stringsize n
```

n には 0 ～ 16384 までの整数値を指定します。新しい値は即座にグローバル変数 @@stringsize に反映されます。

Java クラスのカラムの制約

リモート・テーブルの Java カラムに対する制約は、リモート・サーバ上でチェックします。リモート・テーブルの制約をローカル・サーバ上でチェックしようとしても、できません。そのため、Java データ型の制約がチェックされるデータに対して insert、update、delete を実行する場合は、トレース・フラグ 11220 を有効にします。「トレース・フラグ」(74 ページ) を参照してください。

エラー・メッセージ

リモート・データ・アクセスで Java を使用する場合に表示されるメッセージが 2 つあります。

- Error 11275 – 拡張データ型を参照する文には、リモート・サーバに送信されないようにする構文が含まれていました。文を再作成するか、または拡張データ型の参照を削除してください。
- Error 11276 – カラム '<colname>' のオブジェクトを直列化解除できませんでした。おそらくオブジェクトがトランケートされたことが原因です。@@textsize の値が、直列化オブジェクトを収めるのに十分な大きさであることを確認してください。

Java ADT (抽象データ型)

JCS (Java Classes in SQL) は、Adaptive Server の内部で Java オブジェクトを保存し使用する方法です。この実装では、Java オブジェクトと Java 関数をリモート・サーバでサポートするためにコンポーネント統合サービスとの対話が必要です。

コンポーネント統合サービスは、リモート Adaptive Server バージョン 12.0 以降で JCS をサポートします。

オブジェクトは、ローカル・サーバとリモート・サーバの間を直列化フォーマット (オブジェクトを再インスタンス化するために使用されるバイナリ表現) で渡されます。コンポーネント統合サービスは、text および image 処理関数を使用して、直列化オブジェクトを image blob として取り扱い、サーバ間でオブジェクトを渡します。オブジェクトが送信先サーバで再インスタンス化されてから処理が続行されます。

リモート・サーバ上の Java オブジェクトと Java 関数への参照を含むクエリを処理する場合、コンポーネント統合サービスは、できるだけ多くの構文をリモート・サーバに転送しようとします。リモート・サーバに渡すことができないクエリはローカル・サーバで処理され、必要なすべてのリモート・オブジェクトの直列化と非直列化が要求されます。Java オブジェクトの直列化と非直列化のオーバヘッドにより、このようなクエリのパフォーマンスは、同等のローカル・アクセスよりも著しく低下します。

Java オブジェクトをサーバ間で交換しやすくするために、コンポーネント統合サービスは次のコマンドを発行します。

```
set raw_object_serialization ON
```

コマンドの発行先は、Java に対応している各 ASEnterprise サーバです。これによりコンポーネント統合サービスは、リモート・サイトから取得したオブジェクトを容易に非直列化できます。

Java クラス定義

サーバ間でオブジェクトを容易に渡すためには、ローカル・サーバとリモート・サーバでの Java クラス定義が互換性を持っている必要があります。そのため、コンポーネント統合サービスは互換性があると見なし、オブジェクト定義にエラーがあれば、非直列化の際に検出されます。リモート・サーバ上の直列化形式のオブジェクトを使用して、ローカル・サーバでオブジェクトを正しくインスタンス化できる場合、オブジェクトは互換と見なされます。これは、逆の場合も同じです。また、リモートでマップされたオブジェクトと共にローカル・サーバで参照される Java メソッドはいずれもリモート・オブジェクトでも定義する必要があります。

ローカル・サーバとリモート・サーバ上のクラス定義の互換性を確保することは、データベース管理者に委ねられています。互換性のないオブジェクトや無効なメソッド参照は、クエリの要求をキャンセルする非直列化エラーや Java の例外を発生させます。

全般的なパフォーマンスを向上させるには、cis packet size 設定変数の値を大きくして、サーバ間での直列化オブジェクトの引き渡しをより円滑にしてください。直列化オブジェクトは、image データ型によってサーバ間で引き渡され、数バイト～ 2GB の範囲でサイズが変化することがあります。

データ型

この項では、コンポーネント統合サービスがデータ型に関する種々の問題にどのように対処するかについて説明します。

Unicode のサポート

Adaptive Server は Unicode 文字セットを正式にサポートしています。用意されているデータ型は、unichar、univarchar、unitext です。これらは、Unicode で表現された 2 バイト文字を構成します。Adaptive Server は、char データ型や varchar データ型の現在の処理方式に一致する、Unicode データとその他すべてのデータ型間の変換関数を提供します。コンポーネント統合サービスはこれらのデータ型をサポートすることで、Unicode で表現されたすべてのエンタープライズ文字のデータのビューを提示できます。メインフレームやその他すべての外部／旧世代システムからの文字データは、unichar または univarchar 型のカラムを使用してプロキシ・テーブルでカラムを定義するときに Unicode に変換されます。

これらの新しいデータ型は、コンポーネント統合サービスの次の機能に影響します。

create table

create table は、新しい Unicode データ型によって記述されたカラムを含むことができます。作成するテーブルがプロキシ・テーブルの場合、コンポーネント統合サービスは、Unicode データ型名 (unichar、univarchar、unitext) を含むコマンド全体を、新しいテーブルが作成されるリモート・サーバに転送します。リモート・サーバがそのデータ型を処理できない場合は、エラーが発生します。

create existing table

Adaptive Server のカラムのデータ型と長さを、リモート・サーバから取得されたメタデータと比較したときに、プロキシ・テーブルで Unicode データ型が許可されるのは次の場合です。

- リモート・サーバのカラムのデータ型が、同じ長さ (バイト数ではなく文字数) の unichar、unitext、または univarchar である。
- 指定されたカラムのリモート・サーバのデータ型が、char または varchar である。この場合、リモート・サーバからフェッチされたデータでの Unicode への変換と、DML コマンド (select、insert、delete、update) の一部として転送されたデータでの Unicode からデフォルトの Adaptive Server 文字セット (UTF8) への変換は、コンポーネント統合サービスが実行する。
- リモート・サーバの Unicode カラムのデータ型が、binary か、varbinary である。リモート・サーバのカラムの長さは、Unicode カラムの長さの 2 倍でなければならない。コンポーネント統合サービスは、リモート・サーバとの間でデータを転送するときに必要に応じて変換を行う。

プロキシ・テーブルをリモート・テーブルにマッピングするときには、Unicode データ型への他のデータ型のマッピングは許可されません。他のデータ型では、型の不一致エラーが発生します。Unicode カラムを既存の char や varchar カラムにマッピングするプロキシ・テーブルを作成するだけで、旧世代システムからのデータを Unicode に変換できます。

注意 Unicode は、create existing table コマンドを使用して、unitext カラムのみにマッピングできます。

create proxy_table

create proxy_table を使用すると、Adaptive Server のユーザは、プロキシ・テーブルのカラム・リストを指定しないで済みます。代わりに、実際のテーブルが存在するリモート・サーバからインポートされたカラムのメタデータからカラム・リストを得られます。リモート・カラムのデータ型が unichar、unitext、または univarchar であるときにのみ、リモート・サーバからの Unicode カラムはプロキシ・テーブルの Unicode カラムにマッピングされます。

alter table

alter table を使用すると、カラムのデータ型を変更できます。Adaptive Server バージョン 12.5 以降では、カラムのデータ型を Unicode データ型に、または Unicode データ型からほかのデータ型に変更できます。プロキシ・テーブルを処理する場合、このコマンドは再構築され、実際のテーブルを所有するリモート・サーバに転送されます。リモート・サーバ (または DirectConnect) がコマンドを処理できない場合は、エラーが発生して Adaptive Server コマンドはアボートされます。

トレース・フラグ 11221 がオンの場合、alter table はリモート・サーバに転送されず、カラムの追加、削除、変更はプロキシ・テーブルでローカルにのみ実行されます。

alter table コマンドを使用すると、unitext を char、varchar、nchar、nvarchar、unichar、univarchar、binary、varbinary に変更できます。またこれらのすべてのデータ型も unitext に変更できます。

select、*insert*、*update*、および *delete* 文

プロキシ・テーブルが関連する場合、Unicode データ型は、select 文の処理に対して 2 とおりの方法で影響します。1 つは、リモート・サーバに渡される SQL 文の作成に関係し、もう 1 つは、コンポーネント統合サービスが Unicode 以外のデータをフェッチしたときの Unicode へのデータの変換に関係します。

プロキシ・テーブルに関する DML コマンドは、リモート・サーバとの対話時に TDS 言語要求か、TDS カーソル要求を使って処理されます。select 文の where 句に、Unicode のカラムと定数に関連する述部が含まれている場合は、文の処理に言語コマンドを使用するか、カーソル・コマンドを使用するかによって、次の 2 つの方法のいずれかを使用して Unicode 定数を処理する必要があります。

1 TDS 言語 − 言語テキスト・バッファに含めることができるクリア・テキスト値を生成する。これは、言語要求の一部として送信できるクリア・テキスト値への Unicode 定数値の変換に関係する。

2 TDS カーソル − CT-Library カーソル要求のための Unicode パラメータを生成する。パラメータ値は Unicode データでもよく、コンポーネント統合サービスは CS_UNICHAR_TYPE で指定されるパラメータのデータ型を使用する必要がある。

コンポーネント統合サービスは、TDS 言語要求か、TDS 動的要求を使って、プロキシ・テーブルに関する insert コマンドを処理します。

insert コマンドをクイックパス・モードで処理できる場合は、TDS 言語要求が使用されます。クイックパス・モードで処理できない場合、insert は TDS 動的要求を使って処理されます。

言語要求では、select コマンドの場合と同様に、Unicode 値をクリア・テキスト形式に変換し、それらを残りの SQL 文と一緒に送信できるようする必要があります。動的要求では、Unicode データは (ほかのすべてのデータ値と共に) パラメータとして動的コマンドに送信されます。受信側サーバは、CS_UNICHAR_TYPE 型のパラメータを処理すると想定されます。

update コマンドと delete コマンドに関連する問題は、select および insert の場合と同じです。Unicode 値は、残りの SQL 文と一緒に送信するためにクリア・テキスト文字に変換するか、または CS_UNICHAR_TYPE 型のパラメータに変換する必要があります。

データ型の変換

サーバがリモート・ソースからデータを受け取った場合は、ベースとなるアプリケーション (Adaptive Server、Open Server) に関係なく、いつでもデータ型を変換できます。

各カラムのリモート・データ型に応じて、リモート・サーバのネイティブのデータ型からローカル・サーバがサポートしている形式にデータが変換されます。

データ型の変換は、create table コマンド、alter table コマンド、create existing table コマンドの処理時に行われます。データ型の変換は、リモート・サーバのサーバ・クラスに依存します。コマンドの処理時に各サーバ・クラスに対して行われるデータ型の変換を説明した表については、「第 3 章 SQL リファレンス」の create table、alter table、create existing table コマンドを参照してください。

text および *image* データ型

text データ型は、Adaptive Server の論理ページ・サイズによってカラム・サイズが異なる、印刷可能な文字データを格納するために使用します。image データ型は、Adaptive Server の論理ページ・サイズによってバイト数が異なる、16 進コードのバイナリ・データを格納するために使用します。text データ、image データ、unitext データの最大長は、カラムがマップされるリモート・サーバのサーバ・クラスによって定義されます。

注意　コンポーネント統合サービスでの unitext は、Adaptive Server バージョン 15.0 以降でのみサポートされます。

---

text、*image*、および *unitext* カラムの制限

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

- ストアド・プロシージャへのパラメータとしての使用 (set textptr_parameters がオンである場合を除く)。
- ローカル変数として使用できない。
- order by 句、compute 句、group by 句内では使用できない。
- インデックス内での使用はできない。
- サブクエリ内での使用はできない。
- where 句内での使用 (キーワード like と併用する場合を除く) はできない。
- ジョインでは使用できない。

@@textsize の最大バイト数

select 文は、グローバル変数 *@@textsize* に指定されている最大バイト数以内で text データ、image データ、および unitext データを返します。最大バイト数を変更するには、set textsize コマンドを使用します。*@@textsize* の初期値は 32K で、最大値は 2147MB です。

奇数バイトの埋め込み

255 バイト未満の奇数バイトの image 値には、先頭に 0 が埋め込まれます (“0xaaabb” を挿入すると “0x0aaabb” になります)。255 バイトを超える奇数バイトの image では、insert は実行できません。

text および *image* データ型の変換

convert 関数を使用して、text 値を char または varchar に、image 値を binary または varbinary に変換できます。ただし、character データ型と binary データ型の最大長に制限されます。最大長は、Adaptive Server の論理ページ・サイズによって異なります。長さを指定しない場合、変換された値はデフォルトの長さである 30 バイトになります。暗黙的な変換はサポートされていません。

text データと *unitext* データでのパターン一致

text、unitext、varchar、または char カラム内で、指定したパターンの最初のオカレンスの開始位置を検索するには、patindex 関数を使用します。% ワイルドカード文字は、最初の文字または最後の文字を検索する場合を除いて、パターンの前後に指定する必要があります。

like キーワードを使用して、特定のパターンを検索できます。次の例は、texttest テーブルの blurb カラムから、パターン “Straight Talk%” が含まれている各 text データ値を選択します。

```
select blurb from texttest
 where blurb like "Straight Talk%"
```

like キーワードを使用して、unitext カラムを特定のパターンで検索できます。ただし、like 句は unitext カラムで使用すると最適化されません。like による unitext のパターン検索は、デフォルトの Unicode ソート順に影響されます。このソート順は unichar と univarchar データ型の like を使用したパターン検索でも使用されます。

text 値および *image* 値の入力

DB-Library™ 関数 dbwritetext と dbmoretext、および Client-Library 関数 ct_send_data は、text 値、unitext 値、image 値を入力する最も効率的な方法です。

readtext using bytes

text カラムで readtext using bytes コマンドを使用し、サイズとオフセットの組み合わせによりマルチバイト文字の一部が転送されたような場合、エラーが発生します。

bulk copy での text、image および unitext

bulk copy を使用して text 値、unitext 値、および image 値をリモート・サーバにコピーする場合、サーバは値をリモート・サーバに送信する前にデータ・ページに格納する必要があります。値がリモート・サーバに送信されると、データ・ページは解放されます。データ・ページの割り当てと解放は、ロー単位で行われます。これは、次の理由のために重要です。

- データ・ページの割り当てと解放のオーバヘッドは、パフォーマンスに影響を与える。
- データ・ページは、テーブルがあるデータベースに割り当てられるため、データベースは、特定のローに存在する最も大きい text、unitext、image 値のデータ・ページに対応できる十分な大きさでなければならない。

エラー・ロギング

text、unitext、image データ (リモート・サーバでのみ) の処理は、トレース・フラグ 11207 を使用してログできます。

サーバ・クラス *ASEnterprise* での *text* データ、*unitext* データ、*image* データ

- text、unitext、または image カラム内のポインタは、カラムが初期化されたときに割り当てられる。text データ、unitext データ、または image データをカラムに入力する前に、カラムを初期化する必要がある。これにより、リモート・サーバまたは Adaptive Server に、容量が 2K のページが割り当てられる。text カラム、unitext カラム、または image カラムを初期化するには、update コマンドを NULL とともに使用するか、非 NULL の insert コマンドを使用する。
- writetext を使用して text データまたは unitext データを入力する前、または readtext を使用して読み込む前には、text カラム、または unitext カラムを初期化する必要がある。update または非 NULL の insert を使用して、text カラムを初期化してから writetext または readtext を使用する。
- update を使用して、既存の text、unitext、および image データを NULL で置き換えると、リモート・サーバ内に割り当てられている、最初のページを除くすべてのデータ・ページが再利用される。
- 16KB を超える text 値、unitext 値、または image 値を入力するには、writetext、select into、DB-Library 関数、または Client-Library 関数を使用する必要がある。
- readtext は、text、unitext、および image データにアクセスするのに最も効率的な方法である。
- insert select と select into は、text データ、unitext データ、および image データをプロキシ・テーブルに挿入するときに使用できるが、ユニーク・インデックスが必要である。

サーバ・クラス *direct_connect* での *text* データ、*image* データ、*unitext* データ

- DirectConnect サーバは、それぞれ異なるレベルで text および image データをサポートしている。text、unitext、および image のサポートについては、DirectConnect のマニュアルを参照。
- サーバは、カラムの長さに、グローバル変数 *@@textsize* に定義されている長さを使用する。create table を発行する前に、クライアント・アプリケーションは、set textsize を呼び出して *@@textsize* を必要な長さに設定しなければならない。
- text、unitext、および image データ型はサポートしていても、テキスト・ポインタはサポートしていない DirectConnect サーバでは、次の制限が適用される。
  - writetext コマンドはサポートしていない。
  - readtext コマンドはサポートしていない。
  - テキスト・ポインタを使用する Client-Library 関数はサポートしていない。
  - テキスト･ポインタを使用する DB-Library 関数はサポートしていない。
- text、unitext、および image データ型はサポートしていても、テキスト・ポインタをサポートしていない DirectConnect サーバで次の関数を使用する場合は、追加の処理をいくつか実行する。
  - patindex
  - char_length
  - datalength

  テキスト・ポインタがサポートされている場合、サーバは、RPC を DirectConnect サーバに発行してこれらの関数を実行します。
- テキスト・ポインタをサポートしていない DirectConnect サーバは、データを sysattributes システム・テーブルに格納する。データ・ページは、ローあたりのカラムごとに事前に割り当てられる。カラム・サイズは、*@@textsize* によって定義される。この値が十分でない場合、エラーが返される。
- DirectConnect サーバによっては、text データ型に対してパターン一致をサポートしている場合とサポートしていない場合がある。DirectConnect サーバがこのパターン一致をサポートしていない場合、サーバは text 値を内部のデータ・ページにコピーし、パターン一致を内部的に実行する。DirectConnect サーバでパターン一致が実行される場合、パフォーマンスは最高になる。
- 450 バイトを超える text 値、unitext 値、または image 値を入力する場合は、writetext、select into、または insert...select を使用する必要がある。
- select into および insert...select を使用して text 値、unitext 値、または image 値を入力する場合は、テーブルにユニーク・インデックスが必要である。

設定とチューニング

この項では、設定、チューニング、トレース・フラグ、バックアップとリカバリ、セキュリティの問題について説明します。

システム管理者またはデータベース所有者は、サーバの使用について、パフォーマンスを最適化するか、使用クライアント数を指定するかを選択できます。設定時の選択によっては、所定の SQL コマンドについて読み込みと書き込みの合計回数を確認できます。

アプリケーションが正常に実行されると、システム管理者はパフォーマンスを監視する必要があり、場合によってはシステムをカスタマイズしたり、チューニングしたりします。サーバでは、こうした作業のためにツールが提供されます。この項で説明する項目は、次のとおりです。

- sp_configure プロシージャを使用してシステム・パラメータを変更する。
- update statistics を使用してコンポーネント統合サービスで既存のインデックスが最大限に活用されるようにする。
- dbcc コマンドを使用してサーバ・アクティビティを監視する。
- トレース・フラグの設定
- ddlgen および関連するバックアップとリカバリの処理を実行する。
- データベース・サイズの要件を決定する。

sp_configure の使用

sp_configure の設定パラメータによって、リソースの割り付けとパフォーマンスが制御されます。システム管理者は、パフォーマンスをチューニングしたり格納領域の割り付けを再定義したりするために、これらの設定パラメータをリセットできます。システム管理者がパラメータを変更していない場合、サーバはすべてのパラメータにデフォルト値を採用します。

次の手順で、設定パラメータをリセットします。

- sp_configure を実行し、システム・テーブル master..sysconfigures の値フィールドを更新する。
- 任意の静的設定パラメータをリセットした場合、サーバを再起動する。次に動的パラメータの一覧を示す。
    - cis rpc handling
    - cis cursor rows
    - cis bulk insert batch size
    - cis bulk insert array size
    - cis packet size

sysconfigures テーブル

master..sysconfigures システム・テーブルには、すべての設定オプションが保管されています。各パラメータの設定値と実行値の他に、各パラメータで設定可能な最小値と最大値を識別するカラムも含まれます。

ユーザは、sysconfigures の status カラムを更新できません。status が 1 の場合、パラメータは動的なので、設定パラメータに新しい値を指定するとすぐに有効になります。status が 0 の設定パラメータは、サーバを再起動した後に有効になります。

sp_configure をパラメータなしで実行すると、現在使用中の設定パラメータの実行値を表示できます。

設定パラメータの変更

sp_configure を引数なしで実行すると、すべての設定値が表示されます。オプションの名前や値を使用すると、サーバではシステム・テーブル内の該当するオプションの設定値がリセットされます。

sp_configure のオプションの構文の詳細については、『システム管理ガイド』を参照してください。

コンポーネント統合サービスのオプションを調べるには、次のように入力します。

```
sp_configure "Component Integration Services"
```

設定パラメータの現在の値を変更するには、sp_configure を次のように入力します。

```
sp_configure "parameter", value
```

コンポーネント統合サービスの設定パラメータ

次の設定パラメータは、コンポーネント統合サービスに固有のものです。

- enable cis
- enable file access
- enable full-text search
- max cis remote connections
- cis bulk insert batch size
- cis bulk insert array size
- cis cursor rows
- cis packet size
- cis rpc handling

enable cis

次の手順で、このパラメータを sp_configure で使用し、コンポーネント統合サービスを有効にします。

1　Adaptive Server にシステム管理者としてログインし、次のコマンドを発行します。

```
sp_configure "enable cis", 1
```

2　Adaptive Server を再起動する。

sp_configure "enable cis", 0 を発行すると、サーバの再起動後にコンポーネント統合サービスが無効になります。

enable file access

この設定パラメータは、プロキシ・テーブルから外部ファイル・システムへのアクセスを有効にします。

enable full-text search

この設定パラメータは拡張型全文検索サービスを有効にします。ASE_EFTS のライセンスが必要です。

max cis remote connections

ゼロ以外の値は、サーバの初期化中に事前に割り当てられた接続データ構造体の数を示します。デフォルトはゼロです。

cis bulk insert batch size

この設定パラメータでは、ターゲット・テーブルが Adaptive Server 内、またはバルク・コピー・インタフェースをサポートする DirectConnect サーバ内にある場合に、select into を使用して、ソース・テーブルからターゲット・テーブルに単一バッチとしてバルク・コピーするローの数を指定します。

デフォルトの 0 のままにした場合、すべてのローが単一バッチとしてコピーされます。0 以外の場合、このパラメータに指定した数のローがターゲット・テーブルにコピーされた後、コンポーネント統合サービスがターゲット・サーバにバルク・コミットを発行することにより、バッチがコミットされます。

通常のクライアント生成によるバルク・コピー操作 (bcp ユーティリティによる生成など) が受信された場合、バルク・バッチのサイズはクライアント側で制御するため、コンポーネント統合サービスはこの設定パラメータの値を無視します。

cis bulk insert array size

Adaptive Server 間のデータのバルク転送を実行するとき、コンポーネント統合サービスはローを内部的にバッファし、Open Client バルク・ライブラリにそれらをブロックとして転送するよう要求します。配列のサイズは、設定パラメータ cis bulk insert array size によって制御されます。デフォルトは 50 ローです。プロパティが動的なので、サーバをリブートせずに変更できます。

cis cursor rows

この設定パラメータでは、cursor open と cursor fetch 操作で可能なカーソルのローの数を指定します。この値を増やすと、1 つの操作でより多くのローがフェッチされるようになります。これによって処理速度は速くなりますが、メモリがより多く必要になります。デフォルトは 50 です。

cis packet size

この設定パラメータでは、コンポーネント統合サービスとリモート・サーバの間で接続開始時に交換される TDS (Tabular Data Stream™) パケットのサイズを指定します。

多くのシステムでは、デフォルトのパケット・サイズは 512 バイトです。この値は、ほとんどのアプリケーションに適しています。ただし、特に text や image データまたはバルク・データが関係する場合は、パケット・サイズをこれより大きくするとクエリのパフォーマンスが大幅に向上することがあります。

デフォルトよりも大きいパケット・サイズを指定する場合は、ターゲット・サーバで可変長のパケット・サイズを処理できるように設定します。この場合に関連する Adaptive Server の設定パラメータは次のとおりです。

- additional netmem
- maximum network packet size

これらの設定パラメータの詳細については、『システム管理ガイド』を参照してください。

cis rpc handling

このグローバル設定パラメータでは、コンポーネント統合サービスがアウトバウンド RPC 要求をデフォルトで処理するかどうかを指定します。sp_configure “cis rpc handling” 1 を使用してこれを有効にすると、すべてのアウトバウンド RPC がコンポーネント統合サービスで処理されます。sp_configure “cis rpc handling” 0 を使用すると、Adaptive Server のサイト・ハンドラが使用されます。set cis_rpc_handling on を使用してセッション単位にこれを上書きすることができます。グローバル・プロパティが無効になっている場合は、必要に応じてこの機能を有効または無効にできます。

アウトバウンド RPC を処理するために Adaptive Server のサイト・ハンドラを使用する場合とコンポーネント統合サービスを使用する場合の違いについては、「RPC 処理とコンポーネント統合サービス」(47 ページ) を参照してください。

ステータスのグローバル変数

コンポーネント統合サービスのユーザのために次のグローバル変数が追加されています。

- @@cis_rpc_handling
- @@transactional_rpc
- @@textptr_parameters
- @@stringsize
- @@bulkbatchsize － sp_configure または set bulk batch size コマンドで設定された、現在の cis bulk insert batch size の値が含まれる。

- @@bulkarraysize − sp_configure または set bulk array size コマンドで設定された、現在の cis bulk insert array size の値が含まれる。

これらのグローバル変数は、対応する設定パラメータの現在のステータスを示します。たとえば、cis_rpc_handling のステータスを調べるには、次のコマンドを発行します。

```
select @@cis_rpc_handling
```

コマンドを発行すると、0 (オフ) または 1 (オン) を返します。

第 3 章

SQL リファレンス

この章では、コンポーネント統合サービスでサポートされているサーバ・クラスに関するリファレンス項目について説明します。

各サーバ・クラスには一連のユニークな特徴があります。リモート・データ・アクセス用にサーバを設定するには、システム管理者およびプログラマがこの特徴を知っておく必要があります。これらの特徴は、次のとおりです。

- 各サーバ・クラスがサポートするサーバのタイプ
- そのサーバ・クラスに特有のデータ型変換
- サーバ・クラスに適用される Transact-SQL 文の制限

dbcc コマンド

コンポーネント統合サービスで使用される dbcc コマンドはすべて、単一の dbcc エントリ・ポイントとともに使用できます。

dbcc cis の構文は、次のとおりです。

```
dbcc cis ("subcommand"[, vararg1, vararg2...])
```

コンポーネント統合サービスが設定されていない場合や、ロードされていない場合は、コマンドは実行時エラーになります。

dbcc cis コマンドの使用には、制限がありません。

dbcc のオプション

以下の dbcc オプションは、コンポーネント統合サービスに固有のものです。

remcon

remcon を使用すると、すべてのコンポーネント統合サービスのクライアントによって確立されたすべてのリモート接続のリストが表示されます。引数はありません。

srvdes

srvdes を引数なしで使用すると、メモリ内すべての SRVDES 構造の書式付きリストが返されます。引数を指定すると、sysservers で検索した情報とメモリ内 SRVDES のバージョンが同期されます。コマンドのオプションの引数は、次のように指定します。

```
srvdes, [ srvid ]
```

showcaps

showcaps を次のように使用すると、servername のすべての機能について、機能名、ID、値のリストが表示されます。

```
showcaps, servername
```

次に例を示します。

dbcc cis(“showcaps”, “servername”)

トレース・フラグ

システム管理者は、dbcc traceon オプションを使用して、トレース・フラグをコンポーネント統合サービス内で有効にできます。コンポーネント統合サービスでイベントが発生すると、そのロギングはトレース・フラグによって有効になります。各トレース・フラグは、番号によってユニークに識別されます。トレース・フラグには、コンポーネント統合サービスでグローバルなものと、spid に基づく、トレース・フラグを有効にしたユーザのみに反映されるものがあります。dbcc traceoff によって、トレース・フラグは無効になります。

構文は次のとおりです。

dbcc traceon (*traceflag* [, *traceflag*...])

トレース・フラグとその説明を表 3-1 に示します。

表 3-1: コンポーネント統合サービスのトレース・フラグ

トレース・フラグ	説明
11201	クライアントの接続イベント、切断イベント、アテンション・イベントのロギングを行う(global)。イベントはエラー・ログにのみ送信されます。
11202	クライアントの言語、カーソル宣言、動的準備、動的即時実行のテキストのロギングを行う (グローバル)。テキストはエラー・ログにのみ送信されます。
11203	クライアントの RPC イベントのロギングを行う (グローバル)。イベントはエラー・ログにのみ送信されます。
11204	クライアントに送信されるすべてのメッセージのロギングを行う (グローバル)。メッセージはエラー・ログにのみ送信されます。
11205	リモート・サーバとのすべての対話のロギングを行う (グローバル)。対話はエラー・ログにのみ送信されます。
11206	ログ・ファイル／ディレクトリ処理ステップ(グローバル)
11207	text 型と image 型の処理のロギングを行う(グローバル)
11208	create index 文と drop index 文がリモート・サーバに転送されないようにする。sysindexes は、更新される (spid)。
11209	update statistics の使用時、リモート・テーブルから、完全な分散統計ではなくロー・カウントだけを取得する (spid)。
11211	テーブルが create table at location 構文を使用して作成されている場合、drop table 構文がリモート・サーバに転送されないようにする。
11212	テーブル名内のアンダースコア (“_”) をエスケープしない (spid)。
11213	カラムとテーブルの制約を生成しない (spid)。
11214	コンポーネント統合サービスの起動時のリカバリを無効にする (グローバル)
11216	クイックパスを無効にする (spid)。
11217	クイックパスを無効にする (グローバル)
11218	コンポーネント統合サービス・テーブルに関連するカーソルをデフォルトで更新可能にする。
11220	ローカル・サーバ上のリモート・テーブルの制約チェックを無効にする。これにより、二重チェックを防ぐ。このトレース・フラグをオンにすると、制約によってクイックパス・モードでクエリが拒否されることがなくなる (spid)。
11221	オンにすると、リモート・サーバに対して alter table コマンドが無効になる。リモート・テーブルのカラムを変更しないで、ローカル・テーブルの *type*、*length*、*nullability* を変更できる。トレース・フラグ 11221 は、慎重に使用する必要がある。「同期しない」テーブルが発生する可能性がある (spid)。

トレース・フラグ	説明
11223	create existing table または create proxy_table コマンドの実行中に、プロキシ・テーブルのインデックスの作成を無効にする。このフラグをオンにすると、プロキシ・テーブルが参照するリモート・サイトからは、インデックス・メタデータがインポートされない。また、プロキシ・テーブルのインデックスも作成されない。このトレース・フラグを使用する場合は注意が必要。不要になったら無効にする必要がある (グローバル)
11228	RPC へのプロキシ・テーブルのマッピングを無効にする。
11229	Adaptive Server バージョン 12.5.3 より前の方法を使用して統計データを収集するように、コンポーネント統合サービスを設定する。
11299	リモート・サーバへの接続に失敗した場合に、接続情報をログに記録する。

関数

この項では、コンポーネント統合サービスのサーバ・クラスと組み込みの Adaptive Server 関数の互換性について説明します。

コンポーネント統合サービスでの関数のサポート

select、insert、delete、update などの SQL 文に組み込み関数が含まれている場合、コンポーネント統合サービスは関数がリモート・サーバに転送可能であるか、関数をリモート・データを使用してローカル・サーバ内で評価する必要があるかを確定します。

関数を含む文をクイックパス・モードで扱うことができる場合のみ、関数はリモート・サーバに送信されます。

下記の表で、'Y' はサーバ・クラスで関数がサポートされることを示し、'N' はサポートされないことを示します。'C' は、基本となる DBMS の機能によってサポートされるかどうかが異なることを示します (ほとんどの場合、DirectConnects がこれに当てはまります)。

集合関数

集合関数は、クエリ結果に新しいカラムとして表示される計算値を生成します。

表 3-2: サーバ・クラスでの集合関数のサポート

関数	ASE	ASA	ASIQ	dir_con
avg	Y	Y	Y	C
count	Y	Y	Y	C
max	Y	Y	Y	C
min	Y	Y	Y	C
sum	Y	Y	Y	C
count_big	Y	N	N	N

データ型変換関数

データ型変換関数は 1 つのデータ型から別のデータ型に式を変更して、日付／時刻情報の新しい表示フォーマットを指定します。

表 3-3: サーバ・クラスでのデータ型変換関数のサポート

関数	ASE	ASA	ASIQ	dir_con
convert()	Y	Y	Y	C
inttohex()	Y	N	N	N
hextoint()	Y	N	N	N
biginttohex()	Y	N	N	N
hextobigint()	Y	N	N	N

日付関数

日付関数は、datetime や smalldatetime のデータ型の値を操作します。getdate() 関数は常にローカル・サーバによって拡張されます。しかし、この組み込み関数の使用によって、クイックパス・モードの最適化からクエリが削除されることはありません。

表 3-4: サーバ・クラスでの日付関数のサポート

関数	ASE	ASA	ASIQ	dir_con
dateadd	Y	Y	Y	C
datediff	Y	Y	Y	C
datename	Y	Y	N	C
datepart	Y	Y	Y	C

数学関数

数学関数は、数値データの操作に通常必要な値を返します。数値関数の名前は、キーワードではありません。

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

表 3-5: サーバ・クラスでの数学関数のサポート

関数	ASE	ASA	ASIQ	dir_con
abs	Y	Y	Y	C
acos	Y	Y	N	C
asin	Y	Y	N	C
atan	Y	Y	N	C
atn2	Y	Y	N	C
ceiling	Y	Y	Y	C
cos	Y	Y	N	C
cot	Y	Y	N	C
degrees	Y	Y	N	C
exp	Y	Y	N	C
floor	Y	Y	Y	C
log	Y	Y	N	C
log10	Y	Y	N	C
pi	Y	Y	N	C
power	Y	Y	N	C
radians	Y	Y	N	C
rand	Y	Y	Y	C
round	Y	Y	N	C
sign	Y	Y	N	C
sin	Y	Y	N	C
sqrt	Y	Y	Y	C
tan	Y	Y	N	C

セキュリティ関数

セキュリティ関数は、セキュリティ関連情報を返します。

表 3-6: サーバ・クラスでのセキュリティ関数のサポート

関数	ASE	ASA	ASIQ	dir_con
ic_sec_service_on()	N	N	N	N
show_sec_services()	N	N	N	N

文字列関数

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

表 3-7: サーバ・クラスでの文字列関数のサポート

関数	ASE	ASA	ASIQ	dir_con
ascii	Y	Y	N	C
char	Y	Y	N	C
charindex	Y	Y	N	C
char_lengt	Y	Y	N	C
difference	Y	Y	Y	C
lower	Y	Y	Y	C
ltrim	Y	Y	Y	C
patindex	N	N	N	N
replicate	Y	Y	N	C
reverse	Y	N	N	Y
right	Y	Y	Y	C
rtrim	Y	Y	Y	C
soundex	Y	N	Y	C
space	Y	Y	N	C
str	Y	Y	N	C
stuff	Y	Y	N	C
substring	Y	Y	Y	C
upper	Y	Y	Y	C

システム関数

システム関数は、データベースから特別な情報を返します。

表 3-8: サーバ・クラスでのシステム関数のサポート

関数	ASE	ASA	ASIQ	dir_con
col_length	Y	Y	N	C
col_name	Y	Y	N	C
curunreservedpgs	N	N	N	N
data_pgs	N	N	N	N
datalength	Y	Y	N	C
db_id	N	N	N	N
db_name	N	N	N	N
getdate	Y	N	N	N
getutcdate	Y	N	N	N
host_id	N	N	N	N

関数	ASE	ASA	ASIQ	dir_con
host_name	N	N	N	N
index_col	N	N	N	N
isnull	Y	Y	N	N
lct_admin	N	N	N	N
mut_excl_roles	N	N	N	N
object_id	N	N	N	N
object_name	N	N	N	N
proc_role	N	N	N	N
ptn_data_pgs	N	N	N	N
reserved_pgs	N	N	N	N
role_contain	N	N	N	N
role_id	N	N	N	N
role_name	N	N	N	N
rowcnt	N	N	N	N
show_role	N	N	N	N
suser_id	N	Y	Y	N
suser_name	N	Y	Y	N
tsequal	Y	Y	N	N
used_pgs	N	N	N	N
user	Y	Y	Y	N
user_id	Y	Y	Y	N
user_name	Y	Y	Y	N
valid_name	N	N	N	N
valid_user	N	N	N	N

text 関数および image 関数

text 関数と image 関数は、text データと image データを操作します。

表 3-9: サーバ・クラスでの text 関数と image 関数のサポート

関数	ASE	ASA	ASIQ	dir_con
textptr()	Y	Y	N	C
textvalid()	Y	Y	N	C

Transact-SQL コマンド

以降のページでは、Transact-SQL コマンドについてアルファベット順に説明します。このコマンドは、直接または間接的に外部テーブルに影響を与え、その結果としてコンポーネント統合サービスにも影響を与えます。コンポーネント統合サービスに与える影響と、コンポーネント統合サービスによる処理方法が、コマンドごとに説明されています。各コマンドの詳細については、『ASE リファレンス・マニュアル』を参照してください。

コンポーネント統合サービスがリモート・サーバにコマンドの全構文 (select 文のすべての句など) を渡さない場合は、渡される構文についてサーバ・クラスごとに説明します。

各コマンドには、コマンドを説明する次のような項があります。

- 説明 − コマンドの簡単な説明が記述されている。
- 構文 − コマンドの完全な Transact-SQL 構文の説明が記述されている。
- 使用法 − コンポーネント統合サービスによる処理についての、サーバ・クラスに依存しない一般的な説明が記述されている。
- サーバ・クラス ASEnterprise − サーバ・クラス ASEnterprise に固有の処理についての説明が記述されている。ASEnterprise クラスのリモート・サーバに転送される構文も記載されている。
- サーバ・クラス ASAnywhere − サーバ・クラス ASAnywhere に固有の処理についての説明が記述されている。ASAnywhere クラスのリモート・サーバに転送される構文も記載されている。
- サーバ・クラス ASIQ − サーバ・クラス ASIQ に固有の処理についての説明が記述されている。ASIQ クラスのリモート・サーバに転送される構文も記載されている。
- サーバ・クラス direct_connect − サーバ・クラス direct_connect に固有の処理についての説明が記述されている。direct_connect クラスのリモート・サーバに転送される構文も記載されている。このリリースでは、サーバ・クラス direct_connect に当てはまるコメントは、すべてサーバ・クラス sds にも当てはまる。

alter table

サーバ・クラス
ASEnterprise

コンポーネント統合サービスは、クラス ASEnterprise として設定されているサーバに次の構文を転送します。

```
alter table [database.[owner].]table_name
    {add column_name datatype [{identity | null}]
       {[, next_column]}...}
    | [drop column_name [, column_name]}
    | modify column_name [data_type] [NULL] |
       [not null]] [, column_name]}
```

- ユーザが alter table コマンドを使用してカラムを追加すると、コンポーネント統合サービスは各カラムのデータ型をリモート・サーバに型名変換せずに渡す。
- ASEnterprise クラス・サーバの場合だけは、lock 句が元のクエリに含まれていて、Adaptive Server のバージョンが 11.9.2 以降であれば、この句も転送される。

サーバ・クラス *ASAnywhere*

このクラスのサーバによる alter table の処理は、ASEnterprise サーバの場合と同じです。

サーバ・クラス *ASIQ*

- このクラスのサーバによる alter table の処理は、ASEnterprise サーバの場合と同じである。
- text データ型と image データ型は、サーバ・クラス ASIQ で完全にサポートされる。

サーバ・クラス *direct_connect*

- コンポーネント統合サービスは、direct_connect クラスとして設定されているリモート・サーバに次の構文を転送する。

```
alter table [database.[owner].]table_name
 add column_name datatype [{identity | null}]
        {[, next_column]}...
```

- コンポーネント統合サービスは、クラスが direct_connect のサーバからの機能応答を要求するが、alter table のサポートは必須である。コンポーネント統合サービスは、機能応答とは無関係にリモート・サーバに alter table を転送する。
- クラスが direct_connect のサーバの動作は、データベースによって異なる。Transact-SQL 構文を転送されたリモート・データベースがこの構文を処理できるかどうかによって、エラーが発生する場合がある。
- サーバ・クラス direct_connect は、bigint、unsigned tinyint、unsigned smallint、unsigned int、unsigned bigint をサポートしていない。
- リモート・サーバの構文機能が Sybase Transact-SQL を示す場合は、Adaptive Server のデータ型がリモート・サーバに送られる。構文機能が DB2 SQL を示す場合は、DB2 データ型が送られる。

  DirectConnect は、bigint、unsigned tinyint、unsigned smallint、unsigned int、unsigned bigint をサポートしていません。

  これらのデータ型のマッピングを表 3-10 に示します。

表 3-10: alter table コマンド処理時の DirectConnect のデータ型変換

Adaptive Server データ型	DirectConnect デフォルトのデータ型
binary(*n*)	binary(*n*)
bit	bit
char	char
date	date
datetime	datetime
decimal(*p*, *s*)	decimal(*p*, *s*)
float	float
image	image
int	int
money	money
numeric(*p*, *s*)	numeric(*p*, *s*)
nchar(*n*)	nchar(*n*)
nvarchar(*n*)	nvarchar(*n*)
real	real
smalldatetime	smalldatetime
smallint	smallint
smallmoney	smallmoney
time	time
timestamp	timestamp
tinyint	tinyint
text	text
unichar	unichar
unitext	unitext
varbinary(*n*)	varbinary(*n*)
varchar(*n*)	varchar(*n*)

使用法

サーバは alter table コマンドを受け取ると、次の場合にそのコマンドを適切なアクセス・メソッドに渡します。

- コマンドが動作するオブジェクトが、リモートまたは外部格納領域のロケーションに関連付けられている。
- コマンドが、add column 要求で構成されている。制約の追加や削除の要求は、アクセス・メソッドには渡されず、ローカルに処理される。

alter table は、リモート・サーバに言語要求として渡されます。

参照

『ASE リファレンス・マニュアル』の「alter table」

case

サーバ・クラス *ASEnterprise*

case 式が元のクエリ構文にあっても、クエリ・オプティマイザはクイックパス・モードを拒否しません。

サーバ・クラス *ASAnywhere*

case 式が元のクエリ構文にあっても、クエリ・オプティマイザはクイックパス・モードを拒否しません。

サーバ・クラス *ASIQ*

case 式を処理する機能は、このクラスのサーバには設定されていません。case 式が含まれている SQL 文が最適化される場合は、case 式があると、コンポーネント統合サービスのクイックパス最適化によって文が拒否されます。この状態が発生した場合は、リモート・サーバからデータを取り出した後、case 式をローカルの Adaptive Server で評価する必要があります。

サーバ・クラス *direct_connect*

case 式を処理する機能は、RPC sp_capabilities からの結果セットで決まります。direct_connect が case 式を処理できることを示している場合は、クエリの処理にクイックパス・モードが使用されると、コンポーネント統合サービスは case 式を direct_connect に転送します。

参照

『ASE リファレンス・マニュアル』の「case」

connect to...disconnect

サーバ・クラス ASEnterprise

disconnect が発行されると、コンポーネント統合サービスはリモート・サーバに disconnect を転送し、パススルー・モードから切断します。パススルー・モードでない場合、構文エラーが発生する場合があります。しかし、このエラーはコンポーネント統合サービスによって無視され、クライアントには転送されません。

サーバ・クラス *ASAnywhere*

connect または disconnect が発行される場合は、ASAnywhere との対話は発生しません。

サーバ・クラス *ASIQ*

connect または disconnect が発行される場合は、ASIQ との対話は発生しません。

サーバ・クラス direct_connect

クラス direct_connect のサーバを使用して connect が発行された場合は、direct_connect に次の RPC が送信されます。

```
sp_thread_props “passthru mode”, 1
```

disconnect が発行され、パススルー・モード接続が確立されたサーバが direct_connect である場合は、direct_connect に次の RPC が送信されます。

```
sp_thread_props “passthru mode”, 0
```

参照

『リファレンス・マニュアル』の「commit」

create existing table

サーバ・クラス *ASEnterprise*

- 表 3-11 は、リモート Adaptive Server のカラムをローカル・プロキシ・テーブルのカラムにマップする場合に使用できるデータ型を示す。

表 3-11: create existing table での Adaptive Server データ型の変換

リモート Adaptive Server データ型	使用できる Adaptive Server データ型
binary(*n*)	image、binary(*n*)、varbinary(*n*)。image でない場合、長さは一致する必要がある。
bit	bit、decimal、float、int、money、numeric、real、smallint、smallmoney、tinyint
char(*n*)	text、nchar(*n*)、nvarchar(*n*)、char(*n*)、varchar(*n*)、unichar、univarchar。text でない場合、長さは一致する必要がある。
datetime	datetime、smalldatetime、char、varchar
decimal(*p*, *s*)	bit、decimal、float、int、money、numeric、real、smallint、smallmoney、tinyint
float	bit、decimal、float、int、money、numeric、real、smallint、smallmoney、tinyint
image	image
int	bit、decimal、float、int、money、numeric、real、smallint、smallmoney、tinyint
money	bit、decimal、float、int、money、numeric、real、smallint、smallmoney、tinyint
nchar(*n*)	text、nchar(*n*)、nvarchar(*n*)、char(*n*)、varchar(*n*)。text でない場合、長さは一致する必要がある。
numeric(*p*, *s*)	bit、decimal、float、int、money、numeric、real、smallint、smallmoney、tinyint
nvarchar(*n*)	text、nchar(*n*)、nvarchar(*n*)、char(*n*)、varchar(*n*)、unichar、univarchar。text でない場合、長さは一致する必要がある。
real	bit、decimal、float、int、money、numeric、real、smallint、smallmoney、tinyint
smalldatetime	datetime、smalldatetime、charvarchar
smallint	bit、decimal、float、int、money、numeric、real、smallint、smallmoney、tinyint
smallmoney	bit、decimal、float、int、money、numeric、real、smallint、smallmoney、tinyint
text	text、unitext
timestamp	timestamp
tinyint	bit、decimal、float、int、money、numeric、real、smallint、smallmoney、tinyint
unichar	char、varchar、unichar、univarchar、text, datetime、smalldatetime
univarchar	char、varchar、unichar、univarchar、text, datetime、smalldatetime
unitext	unitext
varbinary(*n*)	image、binary(*n*)、varbinary(*n*)。image でない場合、長さは一致する必要がある。

リモート Adaptive Server データ型	使用できる Adaptive Server データ型
varchar(*n*)	text、nchar(*n*)、nvarchar(*n*)、char(*n*)、varchar(*n*)、unichar、univarchar。text でない場合、でない場合、長さは一致する必要がある。
date	
time	
bigint	暗黙的：binary、varbinary、bit、tinyint、smallint、int、decimal、numeric、float、real、money、smallmoney 明示的：char、varchar、unichar、univarchar
unsigned tinyint	暗黙的：binary、varbinary、bit、tinyint、smallint、unsigned smallint、int、unsigned int、bigint、unsigned bigint、decimal、numeric、float、real money、smallmoney 明示的：char、varchar、unichar、univarchar サポートされない：text、image、date、time、datetime、smalldatetime
unsigned smallint	暗黙的：binary、varbinary、bit、tinyint、smallint、unsigned smallint、int、unsigned int、bigint、unsigned bigint、decimal、numeric、float、real money、smallmoney 明示的：char、varchar、unichar、univarchar サポートされない：text、image、date、time、datetime、smalldatetime
unsigned int	暗黙的：binary、varbinary、bit、tinyint、smallint、unsigned smallint、int、unsigned int、bigint、unsigned bigint、decimal、numeric、float、real money、smallmoney 明示的：char、varchar、unichar、univarchar サポートされない：text、image、date、time、datetime、smalldatetime
unsigned bigint	暗黙的：binary、varbinary、bit、tinyint、smallint、unsigned smallint、int、unsigned int、bigint、unsigned bigint、decimal、numeric、float、real money、smallmoney 明示的：char、varchar、unichar、univarchar サポートされない：text、image、date、time、datetime、smalldatetime

注意 コンポーネント統合サービスは、リモート Adaptive Server バージョン 15.0 以降で unitext をサポートします。

サーバ・クラス
ASAnywhere

- 表 3-12 は、リモート Adaptive Server のカラムをローカル・プロキシ・テーブルのカラムにマップする場合に使用できるデータ型を示す。

表 3-12: create existing table での Adaptive Server Anywhere データ型の変換

リモート Adaptive Server Anywhere データ型	使用できる Adaptive Server Anywhere データ型
binary(*n*)	image、binary(*n*)、varbinary(*n*)。image でない場合、長さは一致する必要がある。
bit	bit、decimal、float、int、money、numeric、real、smallint、smallmoney、tinyint
char(*n*)	text、nchar(*n*)、nvarchar(*n*)、char(*n*)、varchar(*n*)、unichar、univarchar。text でない場合、長さは一致する必要がある。
datetime	datetime、smalldatetime
decimal(*p*, *s*)	bit、decimal、float、int、money、numeric、real、smallint、smallmoney、tinyint
float	bit、decimal、float、int、money、numeric、real、smallint、smallmoney、tinyint
image	image
int	bit、decimal、float、int、money、numeric、real、smallint、smallmoney、tinyint
money	bit、decimal、float、int、money、numeric、real、smallint、smallmoney、tinyint
numeric(*p*, *s*)	bit、decimal、float、int、money、numeric、real、smallint、smallmoney、tinyint
real	bit、decimal、float、int、money、numeric、real、smallint、smallmoney、tinyint
smalldatetime	datetime、smalldatetime
smallint	bit、decimal、float、int、money、numeric、real、smallint、smallmoney、tinyint
smallmoney	bit、decimal、float、int、money、numeric、real、smallint、smallmoney、tinyint
text	text
timestamp	timestamp
tinyint	bit、decimal、float、int、money、numeric、real、smallint、smallmoney、tinyint
varbinary(*n*)	image、binary(*n*)、と varbinary(*n*)、unichar、unitext、univarchar。image でない場合、長さは一致する必要がある。
varchar(*n*)	text、nchar(*n*)、nvarchar(*n*)、char(*n*)、varchar(*n*)、unichar、univarchar。text でない場合、長さは一致する必要がある。
date	
time	

リモート Adaptive Server Anywhere データ型	使用できる Adaptive Server Anywhere データ型
bigint	暗黙的：binary、varbinary、bit、tinyint、smallint、unsigned smallint、int、unsigned int、bigint、unsigned bigint、decimal、numeric、float、real money、smallmoney 明示的：char、varchar、unichar、univarchar サポートされない：text、image、date、time、datetime、smalldatetime
unsigned tinyint	暗黙的：binary、varbinary、bit、tinyint、smallint、unsigned smallint、int、unsigned int、bigint、unsigned bigint、decimal、numeric、float、real money、smallmoney 明示的：char、varchar、unichar、univarchar サポートされない：text、image、date、time、datetime、smalldatetime
unsigned smallint	暗黙的：binary、varbinary、bit、tinyint、smallint、unsigned smallint、int、unsigned int、bigint、unsigned bigint、decimal、numeric、float、real money、smallmoney 明示的：char、varchar、unichar、univarchar サポートされない：text、image、date、time、datetime、smalldatetime
unsigned int	暗黙的：binary、varbinary、bit、tinyint、smallint、unsigned smallint、int、unsigned int、bigint、unsigned bigint、decimal、numeric、float、real money、smallmoney 明示的：char、varchar、unichar、univarchar サポートされない：text、image、date、time、datetime、smalldatetime
unsigned bigint	暗黙的：binary、varbinary、bit、tinyint、smallint、unsigned smallint、int、unsigned int、bigint、unsigned bigint、decimal、numeric、float、real money、smallmoney 明示的：char、varchar、unichar、univarchar サポートされない：text、image、date、time、datetime、smalldatetime
nchar(n)	text、nchar(n)、nvarchar(n)、char(n)、varchar(n)、unichar、univarchar。text でない場合、長さは一致する必要がある。
nvarchar(n)	text、nchar(n)、nvarchar(n)、char(n)、varchar(n)、unichar、univarchar。text でない場合、長さは一致する必要がある。

サーバ・クラス *ASIQ*

- text データ型と image データ型は、ASIQ バージョン 12.6 でサポートされ、ライセンスが必要である。
- 動作はサーバ・クラス ASAnywhere の場合と同じになる。

サーバ・クラス *direct_connect*

- RPC sp_columns は、既存のテーブル内のカラムのデータ型を問い合わせる。
- ローカル・カラム・データ型はリモート・カラム・データ型と同じである必要はないが、表 3-13 に示すように、変換可能である必要がある。そうでない場合、カラム型エラーが発生し、コマンドはアボートする。

表 3-13: create existing table での DirectConnect データ型変換

DirectConnect データ型	使用できる Adaptive Server データ型
binary(*n*)	image、binary(*n*)、varbinary(*n*)。長さが一致しない場合、コマンドはアボートする。
binary(16)	timestamp
bit	bit、decimal、float、int、money、numeric、real、smallint、smallmoney、tinyint
char(*n*)	text、nchar(*n*)、nvarchar(*n*)、char(*n*)、varchar(*n*)、unichar、univarchar。長さが一致しない場合、コマンドはアボートする。
datetime	datetime、smalldatetime
decimal(*p, s*)	bit、decimal、float、int、money、numeric、real、smallint、smallmoney、tinyint
float	bit、decimal、float、int、money、numeric、real、smallint、smallmoney、tinyint
image	image
int	bit、decimal、float、int、money、numeric、real、smallint、smallmoney、tinyint
money	bit、decimal、float、int、money、numeric、real、smallint、smallmoney、tinyint
nchar(*n*)	text、nchar(*n*)、nvarchar(*n*)、char(*n*)、varchar(*n*)、unichar、univarchar。長さが一致しない場合、コマンドはアボートする。
numeric (p, s)	bit、decimal、float、int、money、numeric、real、smallint、smallmoney、tinyint
nvarchar(*n*)	text、nchar(*n*)、nvarchar(*n*)、char(*n*)、varchar(*n*)、unichar、univarchar。長さが一致しない場合、コマンドはアボートする。
real	bit、decimal、float、int、money、numeric、real、smallint、smallmoney、tinyint
smalldatetime	datetime、smalldatetime
smallint	bit、decimal、float、int、money、numeric、real、smallint、smallmoney、tinyint
smallmoney	bit、decimal、float、int、money、numeric、real、smallint、smallmoney、tinyint
text	text
timestamp	timestamp、binary(8)、varbinary(8)
unichar	text、nchar(*n*)、nvarchar(*n*)、char(*n*)、varchar(*n*)、unichar、univarchar。text でない場合、長さは一致する必要がある。
univarchar	text、nchar(*n*)、nvarchar(*n*)、char(*n*)、varchar(*n*)、unichar、univarchar。text でない場合、長さは**一致する**必要がある。
date	

DirectConnect データ型	使用できる Adaptive Server データ型
time	
bigint	UDB および DC/Microsoft は bigint をサポートする。

- データ型の情報は、パラメータに関連する CS_DATAFMT 構造体で渡される。CS_DATAFMT 構造の次のようなフィールドが、データ型の情報に含まれる。
  - *datatype* − Adaptive Server データ型を表す CS_Library データ型。たとえば、CS_INT_TYPE など。
  - *usertype* − ネイティブ DBMS データ型。sp_columns は、create existing table コマンドの実行中に、その結果セットの一部としてこのデータ型をコンポーネント統合サービスに戻す (『リファレンス・マニュアル』の「sp_columns」を参照)。Adaptive Server は、このデータ型をパラメータの *usertype* フィールドに入れて返し、データ型変換において DirectConnect を支援する。

使用法

create existing table を受け取ると、このコマンドは、システム・カタログを更新するための、オブジェクトのリモートまたは外部ロケーションからメタデータをインポートする要求として解釈されます。このメタデータのインポートは、リモート・サーバに送信された、オブジェクトが関連付けられている次の 3 つの RPC によって実行されます。

- sp_tables − リモート・オブジェクトが実際に存在することを確認する。
- sp_columns − create existing table に定義されているものと比較するために、リモート・オブジェクトのカラム属性を取得する。
- sp_statistics − ローカルのシステム・テーブル sysindexes を更新するために、インデックス情報を取得する。

参照

『リファレンス・マニュアル』の「create existing table」

create index

サーバ・クラス *ASEnterprise*

コンポーネント統合サービスは、on *segment_name* 句以外のすべてをリモート・サーバに転送します。

サーバ・クラス *ASAnywhere*

コンポーネント統合サービスは、on *segment_name* 句以外のすべてをリモート・サーバに転送します。

サーバ・クラス *ASIQ*

コンポーネント統合サービスは、on *segment_name* 句以外のすべてをリモート・サーバに転送します。

サーバ・クラス *direct_connect*

- 言語機能が “Transact-SQL” に設定されている場合、コンポーネント統合サービスは max_rows_per_page と on *segment_name* 句を除くすべての構文をリモート・サーバに転送する。

使用法

コマンドが動作するオブジェクトがリモートまたは外部格納領域のロケーションに関連付けられている場合、サーバは create index コマンドを受け取ると、それを適切なアクセス・メソッドに渡します。

コマンドは、クラスに適した構文で再構成され、実行のためにリモート・サーバに渡されます。

create index は、リモート・サーバに言語要求として渡されます。

参照

『リファレンス・マニュアル』の「create index」

create table

サーバ・クラス *ASEnterprise*

コンポーネント統合サービスは、各カラムのデータ型を変換しないでリモート・サーバに渡します。

サーバ・クラス *ASAnywhere*

コンポーネント統合サービスは、各カラムのデータ型を変換しないでリモート・サーバに渡します。

サーバ・クラス *ASIQ*

コンポーネント統合サービスは、各カラムのデータ型を変換しないでリモート・サーバに渡します。

サーバ・クラス *direct_connect*

- コンポーネント統合サービスは create table を再構築して、コマンドをターゲットの DirectConnect に渡す。ゲートウェイは、基本の DBMS が認識できるフォームにコマンドを変換する。
- DirectConnect は、bigint、unsigned tinyint、unsigned smallint、unsigned int、unsigned bigint をサポートしていない。
- Adaptive Server データ型は、表 3-14 に示す DirectConnect 構文モードのデータ型に変換される。

表 3-14: create table での DirectConnect データ型変換

Adaptive Server データ型	デフォルトの DirectConnect データ型
binary*(n)*	binary*(n)*
bit	bit
char	char
datetime	datetime
decimal*(p, s)*	decimal*(p, s)*
float	float
image	image
int	int
money	money
numeric*(p, s)*	numeric*(p, s)*
nchar(n)	nchar(n)
nvarchar(n)	nvarchar(n)
real	real
smalldatetime	smalldatetime
smallint	smallint
smallmoney	smallmoney
timestamp	timestamp
tinyint	tinyint
text	text
unichar(n)	unichar
univarchar(n)	char*(n)* (bit データ)
varbinary*(n)*	varbinary*(n)*
varchar*(n)*	varchar*(n)*
date	
time	
bigint	UDB および DC/Microsoft は bigint をサポートします。

使用法

サーバが create table コマンドを受け取ると、コマンドは新しいテーブルの作成要求として解釈されます。サーバは、作成するテーブルのサーバ・クラスに適したアクセス・メソッドを呼び出します。作成するテーブルがリモートの場合、テーブルが作成されます。このコマンドが正常に終了すると、システム・カタログが更新され、オブジェクトは、オブジェクトが作成されたデータベース内のローカル・テーブルとしてクライアントに表示されます。

create table は、サーバ・クラスに適した構文で再構成されます。たとえば、サーバ・クラスが direct_connect で、リモート・サーバが DB2 の場合、コマンドは、リモート・サーバに渡される前に Adaptive Server Anywhere 構文を使用して再構成されます。Adaptive Server 環境に対してユニークなデータ型については、データ型変換が行われます。

サーバ・クラスによっては、サポートできるデータ型とサポートできないデータ型の制限があります。

create table は、リモート・サーバに言語要求として渡されます。

参照

『リファレンス・マニュアル』の「create table」

delete

サーバ・クラス *ASEnterprise*

元のクエリを変更しないまま転送できない場合、コンポーネント統合サービスは方法 2 を使用して削除を行います。

サーバ・クラス *ASAnywhere*

元のクエリを変更しないまま転送できない場合、コンポーネント統合サービスは方法 2 を使用して削除を行います。

サーバ・クラス *ASIQ*

元のクエリを変更しないまま転送できない場合、ASIQ では更新可能なカーソルをサポートしていないため、コンポーネント統合サービスからエラーが出力されます。

サーバ・クラス *direct_connect*

- クラス direct_connect のサーバに転送される構文は、コンポーネント統合サービスが初めてリモートの DirectConnect に接続したときに発生する機能ネゴシエーションに依存する。ネゴシエーション可能な機能には、サブクエリのサポート、group by のサポート、組み込みサポートなどがある。
- コンポーネント統合サービスはデータ値をパラメータとして、カーソルまたは動的 SQL 文に渡す。DirectConnect がサポートしていれば、言語文も使用できる。そのパラメータは Adaptive Server にネイティブなデータ型であり、DirectConnect によってターゲット DBMS に適合したフォーマットに変換する必要がある。

参照

『リファレンス・マニュアル』の「delete」

drop index

サーバ・クラス *ASEnterprise*

コンポーネント統合サービスは、次の drop index 構文をクラス ASEnterprise として設定されているリモート・サーバに転送します。

```
drop index table_name.index_name
```

drop index 構文ではデータベース名を指定できないため、コンポーネント統合サービスはこの文よりも use database コマンドを優先します。

サーバ・クラス *ASAnywhere*

- コンポーネント統合サービスは、次の drop index 構文をクラス ASAnywhere として設定されているリモート・サーバに転送する。

  ```
  drop index table_name.index_name
  ```

  drop index 構文ではデータベース名を指定できないため、コンポーネント統合サービスはこの文よりも use database コマンドを優先します。

サーバ・クラス *ASIQ*

コンポーネント統合サービスは、次の drop index 構文をクラス ASIQ として設定されているリモート・サーバに転送します。

drop index *table_name.index_name*

drop index 構文ではデータベース名を指定できないため、コンポーネント統合サービスはこの文よりも use database コマンドを優先します。

サーバ・クラス *direct_connect*

コンポーネント統合サービスは、次の drop index 構文をクラス direct_connect として設定されているリモート・サーバに転送します。

drop index *table_name.index_name*

使用法

コマンドが動作するオブジェクトがリモートまたは外部格納領域のロケーションに関連付けられている場合、サーバは drop index コマンドを受け取ると、それを適切なアクセス・メソッドに渡します。

drop index は、クラスに適した構文で再構成され、実行のためにリモート・サーバに渡されます。

このコマンドは、リモート・サーバに言語要求として渡されます。

参照

『リファレンス・マニュアル』の「drop index」

fetch

サーバ・クラス *ASEnterprise*

カーソルが読み取り専用である場合、カーソルがオープンされた後で最初の fetch が受信されたときに、コンポーネント統合サービスは言語要求をリモート・サーバに送信します。読み取り専用でない場合、コンポーネント統合サービスは Client-Library を使用してカーソルをリモート・サーバに宣言します。

サーバ・クラス *ASAnywhere*

fetch 文の処理は、ASEnterprise の場合と同じです。

サーバ・クラス *ASIQ*

カーソルが開いた後で最初に fetch が要求されたときに、コンポーネント統合サービスは言語要求をリモート・サーバに送信します。

サーバ・クラス *direct_connect*

コンポーネント統合サービスは、このクラスのサーバを ASEnterprise のサーバと同様に扱います。

参照

close、deallocate cursor、declare cursor、open

『リファレンス・マニュアル』の「fetch」

insert

サーバ・クラス
ASEnterprise

- values キーワードを使用する insert コマンドは、完全にサポートされる。
- select コマンドを使用する insert コマンドは、text と image 以外のすべてのデータ型でサポートされる。text カラムと image カラムは、null 値が含まれる場合のみサポートされる。
- すべての insert テーブルと select テーブルが同じリモート・サーバに常駐する場合、文全体が実行のためにそのリモート・サーバに転送される。これを、クイックパス・モードと呼ぶ。select が select コマンドのクイックパス規則のすべてに従っていない場合、クイックパス・モードは使用されない。
- select テーブルが 1 つのリモート・サーバに常駐し、insert テーブルが別のサーバに常駐する場合、コンポーネント統合サービスはそれぞれのローをソース・テーブルから選択して、ターゲット・テーブルにそのローを挿入する。
- 計算カラムで insert コマンドを実行することはできない。

サーバ・クラス
ASAnywhere

insert 文の処理は、ASEnterprise の場合と同じです。

サーバ・クラス *ASIQ*

insert 文の処理は、ASEnterprise の場合と同じです。

サーバ・クラス
direct_connect

- values キーワードを使用する insert コマンドは、完全にサポートされる。
- select コマンドを使用する insert コマンドは完全にサポートされるが、テーブルに text カラムまたは image カラムがある場合は、テーブルにユニーク・インデックスが必要である。insert コマンドを select コマンドとともに使用するとき、次の場合にコマンド全体がリモート・サーバに送信される。
  - コマンドで参照されるすべてのテーブルがリモート・サーバに常駐する場合
  - DirectConnect からの機能応答が、insert-select コマンドがサポートされていることを示す場合
  - TopN 機能を使用する場合は、order by 句を使用する必要がある。

  両方の条件が満たされない場合、コンポーネント統合サービスはソース・テーブルからそれぞれのローを選択して、ターゲット・テーブルにそのローを挿入します。
- コンポーネント統合サービスはデータ値をパラメータとして、カーソルまたは動的 SQL 文に渡す。DirectConnect がサポートしていれば、言語文も使用できる。そのパラメータは Adaptive Server にネイティブなデータ型であり、DirectConnect によってターゲット DBMS に適合したフォーマットに変換する必要がある。

参照

『リファレンス・マニュアル』の「insert」

readtext

サーバ・クラス *ASEnterprise*

コンポーネント統合サービスは、基本となるテーブルがプロキシ・テーブルである場合に、次の構文をリモート・サーバに転送します。

```
readtext [[database.]owner.]table_name.column_name
 text_pointer offset size
[using {chars | characters}]
```

サーバ・クラス *ASAnywhere*

readtext 文の処理は、ASEnterprise の場合と同じです。

サーバ・クラス *ASIQ*

readtext 文の処理は、ASEnterprise の場合と同じです。

サーバ・クラス *direct_connect*

- DirectConnect がテキスト・ポインタをサポートしていない場合、readtext を送信できず、使用結果はエラーとなる。
- DirectConnect がテキスト・ポインタをサポートしていない場合、コンポーネント統合サービスは次の構文をリモート・サーバに転送する。

  ```
  readtext
   [[database.]owner.]table_name.column_name
  text_pointer offset size
   [using {chars | characters}]
  ```

- readtext コマンドは、text データまたは image データを読み込む必要があるたびに発行される。select コマンドによって select リスト内の text カラムまたは image カラムが参照される場合、または where 句によって text カラムまたは image カラムが参照される場合に、readtext コマンドが呼び出される。

  たとえば、books プロキシ・テーブルがあり、リモート・サーバ foo の books テーブルにマップされているとします。id と name というカラムがあり、blurb という text カラムがあります。次の文を発行します。

  ```
  select * from books
  ```

  このとき、コンポーネント統合サービスは次の構文をリモート・サーバに送信します。

  ```
  select id, name, textptr(blurb) from foo_books
  readtext foo_books.blurb @p1 0 0 using chars
  ```

参照

『リファレンス・マニュアル』の「readtext」

select

サーバ・クラス *ASEnterprise*

- すべての構文がサポートされる。リモート・サーバは、Transact-SQL 構文を処理するために必要なすべての機能を持つと見なされるので、上記以外の select コマンドのすべての要素はクイックパス・モードを使用してリモート・サーバに転送される。

- バルク・コピー転送は、select...into コマンドが発行されて into テーブルがリモート Adaptive Server 上に存在する場合に、データを新しいテーブルにコピーするために使用される。ローカル・データベースとリモート・データベースの両方が select into / bulkcopy に設定された db option オプションを使用して設定される必要がある。

サーバ・クラス *ASAnywhere*

- すべての構文がサポートされる。リモート・サーバは、Transact-SQL 構文を処理するために必要なすべての機能を持つと見なされるので、上記以外の select コマンドのすべての要素はクイックパス・モードを使用してリモート・サーバに転送される。
- select...into フォーマットを使用しており、into テーブルが ASAnywhere インタフェースからアクセスできる場合、バルク挿入は使用されない。代わりに、コンポーネント統合サービスが Client-Library を使用して、パラメータ化された動的 insert コマンドを準備し、コマンドの select 部分によって返される各ローについてそれを実行する。

サーバ・クラス *ASIQ*

- すべての構文がサポートされる。リモート・サーバは、Transact-SQL 構文を処理するために必要なすべての機能を持つと見なされるので、上記以外の select コマンドのすべての要素はクイックパス・モードを使用してリモート・サーバに転送される。

サーバ・クラス *direct_connect*

- コンポーネント統合サービスが初めてクラス direct_connect のサーバへの接続が必要になったときは、DirectConnect により機能を要求する。コンポーネント統合サービスは、応答に基づいて、select コマンドのどの部分を DirectConnect に転送するか決定する。ほとんどの場合、DirectConnect のインタフェース先である DBMS の機能によって決定される。
- クイック・パス・モードでは DirectConnect に文全体を転送できない場合、転送できない機能はコンポーネント統合サービスによって補われる。たとえば、リモート・サーバが order by 句を処理できない場合、クイックパスは使用されず、コンポーネント統合サービスが結果セットをソートする。
- コンポーネント統合サービスはデータ値をパラメータとして、カーソルまたは動的 SQL 文に渡す。DirectConnect がサポートしていれば、言語文も使用できる。そのパラメータは Adaptive Server にネイティブなデータ型であり、DirectConnect によってターゲット DBMS に適合したフォーマットに変換する必要がある。
- select...into コマンドはサポートされるが、テーブルが text または image カラムを持つ場合、そのテーブルにはユニーク・インデックスが必要である。
- select...into フォーマットが使用されて into テーブルが DirectConnect を使用してアクセスされる場合は、バルク挿入は使用されない。代わりに、コンポーネント統合サービスが Client-Library を使用して、動的 insert コマンドを準備し、コマンドの select 部分によって返される各ローについてそれを実行する。

参照

『リファレンス・マニュアル』の「select」

truncate table

サーバ・クラス *ASEnterprise*

コンポーネント統合サービスは、クラス ASEnterprise のサーバに truncate table コマンドを転送します。

サーバ・クラス *ASAnywhere*

コンポーネント統合サービスは、クラス ASAnywhere のサーバに truncate table コマンドを転送します。

サーバ・クラス *ASIQ*

コンポーネント統合サービスは、クラス ASIQ のサーバに truncate table コマンドを転送します。

サーバ・クラス *direct_connect* および *sds*

Transact-SQL 構文は次のように転送されます。

truncate table [[*database*.]*owner*.]*table_name*

参照

『リファレンス・マニュアル』の「truncate table」

update

サーバ・クラス *ASEnterprise*

- コンポーネント統合サービスが文全体をリモート・サーバに渡すことができない場合は、そのテーブルにユニーク・インデックスが存在する。
- update コマンドは、text および image 以外のすべてのデータ型についてすべてサポートされる。text データおよび image データは、text 値または image 値を null に設定する場合以外は、update コマンドで変更できない。代わりに、writetext コマンドを使用する。
- クイックパス・モードが使用されない場合は、データはソース・テーブルから取り出され、指定された update の処理用に作成された別のカーソルを使用して、ターゲット・テーブル内の値が更新される。

サーバ・クラス *ASAnywhere*

update 文の処理は、ASEnterprise の場合の処理と同じです。

サーバ・クラス *ASIQ*

update 文の処理は、ASEnterprise の場合の処理と同じです。

元のクエリを変更しないまま転送できない場合、ASIQ では更新可能なカーソルをサポートしていないため、コンポーネント統合サービスからエラーが出力されます。

サーバ・クラス *direct_connect*

- クラス direct_connect のサーバでは、次の構文がサポートされる。

```
update [[database.]owner.]{table_name | view_name}
 set [[[database.]owner.]{table_name.|view_name.}]
    column_name1 =
        {expression1|NULL|(select_statement)}
    [, column_name2 =
        {expression2|NULL|(select_statement)}]...
    [where search_conditions]
```

この構文に準拠する update コマンドは、リモート・サーバからの使用可能な機能応答がコマンドのすべての要素をサポートしていることを示している場合は、クイックパス・モードを使用します。ネゴシエーション可能な機能には、サブクエリのサポート、group by のサポート、組み込みサポートなどがあります。

- リモート・サーバがコマンドのすべての要素をサポートしていないか、コマンドが from 句を含んでいる場合は、コンポーネント統合サービスは set 句に対する値を取得するためにクエリを発行してから、リモート・サーバに update コマンドを発行する。
- コンポーネント統合サービスはデータ値をパラメータとして、カーソルまたは動的 SQL 文に渡す。DirectConnect がサポートしていれば、言語文も使用できる。そのパラメータは Adaptive Server にネイティブなデータ型であり、DirectConnect によってターゲット DBMS に適合したフォーマットに変換する必要がある。

参照

『リファレンス・マニュアル』の「update」

update statistics

サーバ・クラス
ASEnterprise

- 情報を要求されたテーブルにインデックスがない場合、コンポーネント統合サービスは次のコマンドを発行する。

  ```
  select count(*) from table_name
  ```

  これは、トレース・フラグ 11209 がオンのときに唯一発行されるコマンドでもあります。

- テーブルにインデックスがあり、コマンドでそのインデックスが指定された場合、コンポーネント統合サービスは次のコマンドを発行する。

  ```
  select count(*) from table_name

  select count(*) column_name [,column_name, ...]
   from table_name
   group by column_name [,column_name, ..]
  ```

  カラム名は、インデックスを構成する 1 つまたは複数のカラムを表します。

  たとえば、次のコマンドが発行されたとします。

  ```
  update statistics customers ind_name
  ```

  コンポーネント統合サービスは次のコマンドを発行します。

  ```
  select count(*) from customers

  select count(*) last_name, first_name
   from customers
   group by last_name, first_name
  ```

- テーブルに 1 つまたは複数のインデックスがあり、文にインデックスが指定されていない場合、コンポーネント統合サービスは select count (*) を 1 回だけ発行し、各インデックスに対して select/order by コマンドを発行する。
- リモート・ログインを使用して、プロキシ・テーブルに対して update statistics を実行するには、sa_role が必要である。
- Adaptive Server バージョン 15.0 以降では、プロキシ・テーブルが分割されたテーブルを指している場合、グローバルな統計のみがインポートされる。Adaptive Server バージョン 15.0 のプロキシ・テーブルは分割されていないため、これらは集計された統計値になる。

サーバ・クラス *ASAnywhere*

このサーバ・クラスでの update statistics の処理は、Adaptive Server バージョン 15.0 以前のサーバの場合と同じです。

サーバ・クラス *ASIQ*

このサーバ・クラスでの update statistics の処理は、Adaptive Server バージョン 15.0 以前のサーバの場合と同じです。

サーバ・クラス *direct_connect*

- このサーバ・クラスでの update statistics の処理は、上記のサーバ・クラス ASEnterprise の場合と同じである。
- direct_connect が group by または count(*) 構文を処理できないことを示している場合は、direct_connect に対して統計情報が集められることはない。

参照

『リファレンス・マニュアル』の「update statistics」

writetext

サーバ・クラス *ASEnterprise*

writetext コマンドは、リモート・サーバに対する別の接続を使用して処理されます。

サーバ・クラス *ASAnywhere*

writetext コマンドは、リモート・サーバに対する別の接続を使用して処理されます。

サーバ・クラス *ASIQ*

writetext コマンドは、リモート・サーバに対する別の接続を使用して処理されます。

サーバ・クラス *direct_connect*

DirectConnect がテキスト・ポインタをサポートする場合は、コンポーネント統合サービスは DirectConnect をクラス ASEnterprise のサーバのように扱います。

参照

『ASE リファレンス・マニュアル』の「writetext」

パススルー・モード

ユーザが「パススルー」先のサーバ上でネイティブな操作を実行できるように、コンポーネント統合サービス内でパススルー・モードが提供されています。

たとえば、Oracle サーバ用のパススルー・モードを要求すると、Oracle DBMS にネイティブな Oracle SQL を送信できるようになります。結果は Open Client アプリケーションで使用可能なフォームに変換され、ユーザに返されます。

Transact-SQL パーサとコンパイラは、このモードではスキップされ、ユーザから受け取った各言語バッチは、ユーザがパススルー・モードで接続しているサーバに直接渡されます。各バッチからの結果は、クライアントに返されます。

パススルー・モードは、次のいくつかの方法で使用できます。

- connect to
- sp_autoconnect
- sp_passthru
- sp_remotesql

connect to

connect to コマンドを使用すれば、ユーザは、パススルー接続するサーバを指定できるようになります。このコマンドの構文は次のとおりです。

```
connect to server_name
```

server_name は、*sysservers* テーブルに追加され、サーバ・クラスとネットワーク名が定義されたサーバの名前です。『ASE リファレンス・マニュアル』の sp_addserver を参照してください。

ユーザに代わって *server_name* への接続を確立する場合、サーバは次のいずれかを使用します。

- sp_addexternlogin を使用するリモート・ログイン・エイリアス・セット
- Adaptive Server との通信に使用される名前とパスワード

どちらの場合も、指定したサーバへの接続が確立できないと、その理由を示すメッセージがユーザに返されます。

パススルー接続が確立されると、後続の言語テキストを受け取ったときに Transact-SQL パーサとコンパイラがスキップされます。サーバから受け取ったすべての文は、指定したリモート・サーバに直接渡されます。

注意 一部のデータベース管理システムは、一度に複数の文を認識しません。たとえば、単一の言語テキスト・バッファの一部として複数の select 文を受け取った場合、構文エラーが生成されます。

文が要求されたサーバに渡されると、結果はすべて Open Client インタフェースが認識できるフォームに変換され、クライアント・プログラムに返されます。

パススルー・モードを終了するには、disconnect コマンドまたは disc コマンドを発行します。このクライアントからの後続の言語テキストは、その後 Transact-SQL パーサおよびコンパイラを使用して処理されます。

connect to を使用するためのパーミッションは、システム管理者が明示的に付与する必要があります。構文は次のとおりです。

```
grant connect to user_name
```

connect to を使用するためのパーミッションを取り消す構文を次に示します。

```
revoke connect from user_name
```

connect to パーミッションは、master データベースに格納されています。"public" に対してパーミッションをグローバルに付与または取り消すには、システム管理者が master データベース内のパーミッションを設定します。使用しているデータベースに関係なく、設定はサーバ全体に影響します。システム管理者は、master データベースの有効なユーザに対してだけパーミッションを付与または取り消しできます。

システム管理者は、あらゆるデータベース内の "public" に対して「すべて」のパーミッションを付与または取り消しできます。パーミッションが master データベース内にある場合、上記の「すべて」には connect to コマンドも含まれます。その他のデータベース内にある場合、上記の「すべて」には connect to コマンドは含まれません。

例

システム管理者が、"public" からパーミッションを取り消して、ユーザ "fred" だけが connect to コマンドを実行できるようにしたいとします。"fred" は、master の有効なユーザでなければなりません。これを実行するには、システム管理者は master で次のコマンドを発行します。

```
revoke connect from public
 sp_adduser fred
 grant connect to fred
```

sp_autoconnect

ユーザによっては、特定のサーバへのパススルー接続を常に必要とします。このような場合は、これらのユーザがサーバに接続したときに、パススルー・モードで、指定したリモート・サーバに自動的に接続されるようにコンポーネント統合サービスを設定することができます。この機能を有効または無効にするには、sp_autoconnect を次の構文で使用します。

```
sp_autoconnect server_name, true|false [,loginname]
```

sp_autoconnect を使用する前に、sp_addserver を使用して、*server_name* を sysservers に追加します。

ユーザは、sp_autoconnect を使用して、サーバへの自動接続を要求できますが、他のユーザの自動パススルー接続を有効または無効にできるのはシステム管理者だけです。このため、システム管理者だけが、このプロシージャに 3 番目の引数を指定できます。

2 番目の引数が true の場合、現在のユーザ (または、3 番目の引数で指定したユーザ) の autoconnect 機能は有効になります。2 番目の引数が false の場合、autoconnect は無効になります。

ユーザがサーバに接続すると、syslogins にあるそのユーザの autoconnect ステータスがチェックされます。有効になっている場合、syslogins にある *server_name* (sp_autoconnect によって配置された) も有効性がチェックされます。サーバが有効な場合、ユーザはそのサーバに自動的に接続され、パススルー・ステータスが設定されます。サーバがこのユーザから受け取った後続の言語文は、ユーザが明示的に connect コマンドを入力した場合とまったく同じように処理されます。このユーザは、その後、リモート・サーバへのパススルー・ゲートウェイと同じようにサーバを表示します。

「自動接続」したユーザが disconnect を実行すると、そのユーザは通常どおりサーバに戻ります。

リモート・サーバに到達できない場合、ユーザに “sa” 役割が割り当てられていないかぎり、ユーザはローカル Adaptive Server には接続されません。「ログインに失敗しました。」というエラー・メッセージが返されます。

sp_passthru

sp_passthru によって、ユーザはリモート・サーバに SQL コマンド・バッファを渡すことができます。渡された SQL 文の構文は、バッファを受け取るサーバ・クラスにネイティブな構文であると見なされ、変換や解釈は行われません。リモート・サーバから返される結果は、オプションで出力パラメータに組み込むことができます。sp_passthru の構文を次に示します。

```
sp_passthru server, command, errcode, errmsg, rowcount
 [, arg1, arg2, ... argn]
```

各要素の意味は次のとおりです。

- *server* は、SQL コマンド・バッファを受け取るサーバの名前で、データ型は varchar(255) である。
- *command* は、SQL コマンド・バッファで、データ型は varchar(255) である。
- *errcode* は、リモート・サーバから返されたエラー・コードで、データ型は int **出力**である。
- *errmsg* は、リモート・サーバから返されたエラー・メッセージで、データ型は varchar(1024) 出力である。
- *rowcount* は、コマンド・バッファ内の最後のコマンドの影響を受けたローの数で、データ型は int **出力**である。

- *arg1* − *argn* はオプションのパラメータである。指定した場合、これらの出力パラメータは、コマンド・バッファ内の最後のコマンドから返された最後のローから結果を受け取る。データ型は異なる場合がある。いずれも出力パラメータでなければならない。

例

```
sp_passthru ORACLE, "select date from dual", @errcodeoutput,
@errmsg output, @rowcount output, @oradate output
```

この例では、Oracle サーバから日付が出力パラメータ @oradate に返されます。Oracle のエラーが発生した場合は、エラー・コードが @errcode に格納され、対応するメッセージが @errmsg に格納されます。@rowcount パラメータは 1 に設定されています。

sp_passthru とそのリターン・ステータスの詳細については、『ASE リファレンス・マニュアル』を参照してください。

sp_remotesql

sp_remotesql を使用することで、ネイティブ構文をリモート・サーバに渡せるようになります。プロシージャは、リモート・サーバへの接続を確立し、クエリ・バッファを渡し、結果をクライアントに返します。sp_remotesql の構文を次に示します。

sp_remotesql server_name, *query_buf1*
[, *query_buf2*, ... , *query_buf254*]

パラメータの意味は次のとおりです。

- *server_name* は、sp_addserver を使用して定義したサーバの名前である。
- *server_name* は、varchar(255) フィールドである。*server_name* が定義されていない場合、または使用できない場合、接続は失敗し、プロシージャはアボートされる。このパラメータは必須である。
- *query_buf1* は、char または varchar 型のクエリ・バッファで、最大長は 255 バイトである。このパラメータは必須である。

追加する各バッファは、char または varchar で、最大長は 255 バイトです。指定した場合、これらのオプションの引数は、*query_buf1* の内容とともに 1 つのクエリ・バッファに連結されます。

例

```
sp_remotesql freds_server, "select @@version"
```

この例では、サーバはクエリ・バッファを、select *@@version* 構文を解釈してクライアントにバージョン情報を返す *freds_server* に渡します。サーバは、返された情報を解釈しません。

sp_remotesql とそのリターン・コードの詳細については、『リファレンス・マニュアル』を参照してください。

引用符付き識別子のサポート

引用符付き識別子は、サポートされるリモート・サーバに転送されます。これは、次の set コマンドによって行われます。

```
set quoted_identifier on
```

このスレッド・プロパティを有効にすると、コンポーネント統合サービスは、SQL 文をリモート・サーバに送信する前に識別子に引用符を付けます。

リモート・サーバは、引用符付き識別子をサポートできる必要があります。そのために予約されている sp_capabilities の結果セットには、次の機能があります。

- 機能 ID：135
- 機能名：引用符付き識別子
- 機能値：0 = 未サポート、1 = サポート

この機能の値を提供しない DirectConnect では、この機能はデフォルトで 0 になります。

区切り識別子のサポート

角カッコ識別子の動作は、この識別子を使用するために set quoted_identifier on を設定する必要がない点を除けば、引用符付き識別子と同じです。

auto identity オプション

Adaptive Server の auto identity データベース・オプションを有効にすると、データベース内に作成されたすべてのテーブルに IDENTITY カラムが追加されます。カラム名は、CIS_IDENTITY_COL (プロキシ・テーブルの場合)、または SYB_IDENTITY_COL (ローカル・テーブルの場合) になります。どちらの場合も、syb_identity キーワードを使用してカラムを参照できます。

トリガ

コンポーネント統合サービスでは、プロキシ・テーブルでトリガを使用できます。ただし、その有用性は制限されています。create を使用してプロキシ・テーブルにトリガを作成し、標準の Adaptive Server テーブルの場合と同様にそのトリガを呼び出すことができます。ただし、プロキシ・テーブルでは、image データの前後がログに書き込まれません。これは、insert、update、delete コマンドがリモート・サーバに渡されるからです。ログを参照しなければならない inserted と deleted テーブルにはデータが含まれていません。ユーザは、挿入、削除、または更新されたローを調べることができないため、プロキシ・テーブルでのトリガは制限された値になります。

Adaptive Server バージョン 15.0 以降では、トリガを使用して更新された関数をサポートしていません。

付 録　A

チュートリアル

この付録では、コンポーネント統合サービスを設定し、リモート・サーバにアクセスするための手順を例を交えて説明します。

注意　このチュートリアルでは、pubs2 データベースがインストール済みであることを前提にしています。

コンポーネント統合サービスの使用開始

この項では、サーバを設定してリモート・データ・ソースにアクセスするための手順を説明します。以下が、その説明内容です。

- リモート・サーバの追加
- リモート・オブジェクトのローカル・プロキシ・テーブルへのマッピング
- リモート・テーブル間のジョインの実行

Adaptive Server の起動と停止、ログインの作成、グループの作成、ユーザの追加、パーミッションの付与、パスワードの管理などの日常的なシステム管理タスクについては、Adaptive Server のマニュアルで説明します。

リモート・サーバの追加

サーバを使用して、リモート・サーバのデータにアクセスできます。コンポーネント統合サービスを設定してから、実行してください。

概要

1　リモート・サーバを interfaces ファイルに追加します。

2　リモート・サーバの名前、サーバ・クラス、ネットワーク名をシステム・テーブルに追加します。

3　オプションで、代替ログイン名とパスワードを割り当てます。

リモート・サーバを interfaces ファイルに追加する

dsedit または dscp ユーティリティを使用して、次の *$SYBASE* ディレクトリにある interfaces ファイルを編集します。

- UNIX では、interfaces ファイルの名前は *interfaces*。
- Windows では、interfaces ファイルの名前は *sql.ini*。

interfaces ファイルの詳細については、各プラットフォーム用の『Adaptive Server Enterprise 設定ガイド』を参照してください。

サーバ・エントリをシステム・テーブルに作成する

sp_addserver を使用して、エントリを sysservers テーブルに追加します。sp_addserver によって、ローカル・サーバのエントリと、呼び出される各リモート・サーバのエントリが追加されます。sp_addserver の構文は次のとおりです。

```
sp_addserver server_name [,server_class [,network_name]]
```

各要素の意味は次のとおりです。

- *server_name* は、サーバを識別するための名前である。これはユニークでなければならない。
- *server_class* には、サポートされているサーバ・クラスの 1 つを指定する。デフォルト値は ASEnterprise。server_class が local に設定されている場合は、*network_name* は無視される。
- *network_name* は、interfaces ファイルにおけるサーバ名である。この名前は、*server_name* と同じ場合もあれば異なる場合もある。*network_name* は、物理名として参照されることがある。

例

次の例では、SYBASE という名前のローカル・サーバのエントリと、ASEnterprise サーバ・クラスの CTOSDEMO という名前のリモート・サーバのエントリを作成します。

```
sp_addserver SYBASE, local
sp_addserver CTOSDEMO, ASEnterprise, CTOSDEMO
```

ローカル・サーバの追加後は、Adaptive Server を再起動してください。

代替ログインとパスワードを追加する

sp_addexternlogin を使用して、リモート・サーバとの通信時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。この手順は任意です。sp_addexternlogin の構文は、次のとおりです。

```
sp_addexternlogin remote_server, login_name, remote_name [,
remote_password]
```

各要素の意味は次のとおりです。

- *remote_server* は、リモート・サーバの名前である。*remote_server* に指定するサーバ名は、master.dbo.sysservers テーブルのエントリによって、ローカル・サーバに認識されていなければならない。
- *login_name* は、ローカル・サーバに認識されているアカウントである。*login_name* には、master.dbo.syslogins テーブルに対応するエントリが存在する名前を指定する必要がある。ローカル・ユーザのリモート・アクセスを修正する権限を持つのは、“sa” アカウントと “sso” アカウント、および *login_name* アカウントだけである。
- *remote_name* は、*remote_server* に認識されるアカウントである。*remote_server* が稼働しているノード上で有効にする。*remote_server* へのログインにはこのアカウントが使用される。
- *remote_password* は、*remote_name* のパスワードである。

例

```
sp_addexternlogin FRED, sa, system, sys_pass
```

この例では、“sa” というユーザのために、ローカル・サーバがリモート名 “system” とリモート・パスワード “sys_pass” を使用して、リモート・サーバ FRED にアクセスできます。

```
sp_addexternlogin OMNI1012, bobj, jordan, hitchpost
```

ログイン名 “bobj” でログインしたユーザがリモート・サーバ OMNI1012 にアクセスしたときに、リモート名 “jordan” とリモート・パスワード “hitchpost” が使用されます。“bobj”、“sa”、“sso” アカウントだけが、ログイン名 “bobj” のリモート・ログインを追加または修正する権限を持ちます。

確認、接続

connect to *server_name* コマンドを使用して、設定が正しいことを確認します。connect to では、“sa” が “sa” 以外のユーザに明示的に接続権限を与える必要があります。connect to コマンドによって、リモート・サーバへのパススルー・モード接続が確立されます。このパススルー・モードは、disconnect コマンドを発行するまで有効です。

2 つのリモート・テーブルのジョイン

コンポーネント統合サービスを使用して、リモート・テーブル間でのジョインを実行できます。

リモート・サーバを interfaces ファイルに追加する

dsedit ユーティリティを使用して、interfaces ファイルを編集します。

リモート・サーバを定義する

sp_addserver を使用して、sysservers システム・テーブルにエントリを追加します。呼び出し側のサーバ上には、呼び出される各リモート・サーバのエントリが必要です。sp_addserver の構文は次のとおりです。

```
sp_addserver server_name [,server_class] [,network_name]
```

各要素の意味は次のとおりです。

- *server_name* は、サーバを識別するための名前である。これはユニークでなければならない。
- *server_class* には、サポートされているサーバ・クラスの 1 つを指定する。デフォルト値は sql_server。値が local の場合、*network_name* は無視される。
- *network_name* は、interfaces ファイルにおけるサーバ名である。これは、*server_name* と同じ名前でも、異なる名前でもかまわない。*network_name* を指定しない場合、デフォルト値は *server_name* と同じになる。

例

次の例では、SYBASE という名前のローカル・サーバのエントリと、ASEnterprise クラスの SYBASE という名前のリモート・サーバのエントリを作成します。

```
sp_addserver SYBASE, local
sp_addserver CTOSDEMO, ASEnterprise, SYBASE
```

リモート・テーブルを Adaptive Server にマッピングする

create existing table で、既存の (プロキシ) テーブルを定義できます。このオプションの構文は、create table コマンドの構文と似ており、次のようになります。

```
create proxy_table
    table_name
    at “pathname”
```

サーバがこのコマンドを処理する場合、新しいテーブルは作成されません。その代わりに、テーブルのマッピングがチェックされ、基本となるオブジェクトの有無が確認されます。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、サーバはこのコマンドを拒否して、クライアントにエラー・メッセージを返します。

既存のテーブルを定義したら、そのテーブルに対して update statistics コマンドを発行します。これによって、クエリ・オプティマイザを使用する場合にインデックス選択とジョイン順序が適切に判断されるようになります。

例

リモートの Adaptive Server のテーブル publishers と、ローカル・サーバにマッピングされたサンプルの pubs2 データベース内の titles を示すには、次の手順に従います。

リモート・テーブルのマッピング

上の図に示したマッピングを生成するには、次の手順に従います。

1　SYBASE という名前のサーバを定義します。サーバ・クラスは ASEnterprise で、interfaces ファイル内での名前は SYBASE です。

```
exec sp_addserver SYBASE, ASEnterprise, SYBASE
```

2　リモート・ログイン・エイリアスを定義します。この手順は任意です。ユーザ “sa” は、リモート・サーバ SYBASE に、ユーザ “sa”、パスワード “timothy” として認識されています。

```
exec sp_addexternlogin SYBASE, sa, sa, timothy
```

3　リモート・テーブル publishers を定義します。

```
create proxy_table publishers
at "SYBASE.pubs2.dbo.publishers"
```

4　リモート・テーブル titles を定義します。

```
create proxy_table titles
at "SYBASE.pubs2.dbo.titles"
```

ジョインを実行する

select 文を使用して、ジョインを実行します。

```
select Publisher = p.pub_name, Title = t.title
 from publishers p, titles t
 where p.pub_id = t.pub_id
 order by p.pub_name
```

付 録　B

トラブルシューティング

この付録では、コンポーネント統合サービスの使用中に発生する可能性がある問題に対するトラブルシューティングのヒントを説明します。この付録の目的は次のとおりです。

- Sybase 製品の保守契約を結んでいるサポート・センタに問い合わせる前に問題を解決できるように、エラーの状態について十分な情報を提供する。
- Sybase 製品の保守契約を結んでいるサポート・センタに問い合わせる前にユーザが収集できる情報のリストを提供する。これは問題解決の時間を短縮するのに役立つ。
- コンポーネント統合サービスの理解を深める。

トラブルシューティングには、『ASE トラブルシューティング＆エラー・メッセージガイド』も使用してください。この付録には、コンポーネント統合サービスに関してよく寄せられる質問に対するトラブルシューティングのヒント、およびすべてのエラー・メッセージが 1 行のリカバリ手順とともに記載されています。また、『ASE トラブルシューティング＆エラー・メッセージ・ガイド』には、コンポーネント統合サービスに固有でない SQL Server の問題に関するヒントが掲載されています。

コンポーネント統合サービスにアクセスする際の問題

リモート・オブジェクトにアクセスするコマンドを発行した場合にコンポーネント統合サービスが見つからないと、次のエラー・メッセージが表示されます。

```
cis extension not enabled or installed
```

- 次のコマンドを実行して、enable cis 設定パラメータが 1 に設定されているか確認する。

  ```
  sp_configure "enable cis"
  ```

  sp_configure は、enable cis パラメータに次のローを返します。

```
name           min  max   config value   run value
enable cis     0    1     1              1
```

  “config value” と “run value” はどちらも 1 にしてください。両方の値が 0 の場合、enable cis 設定パラメータを 1 に設定し、サーバを再起動します。次の構文を使用します。

  ```
  sp_configure "enable cis" 1
  ```

  “config value” が 1 で “run value” が 0 の場合、enable cis 設定パラメータは設定されていますが、サーバを再起動しなければ設定が有効になりません。

注意 コンポーネント統合サービスは、Adaptive Server バージョン 12.0 からデフォルトで有効になっています。

コンポーネント統合サービス使用中の問題

この項では、コンポーネント統合サービスの使用中に発生する可能性がある問題の解決方法を説明します。

リモート・サーバにアクセスできない

リモート・サーバにアクセスできない場合、次のエラー・メッセージが返されます。

```
11206 Unable to connect to server server_name.
```

メッセージの前には、次のような Client-Library メッセージのいずれかが表示されます。

```
Requested server name not found
Driver call to connect two endpoints failed
Login failed
```

Client-Library メッセージは、リモート・サーバにアクセスできなかった理由を表します。次の項で、その内容を説明します。

要求されたサーバ名が見つからない

次のメッセージが表示された場合は、該当のサーバが interfaces ファイルで定義されていません。

```
Requested server name not found
11206 Unable to connect to server server_name.
```

sp_addserver を使用してリモート・サーバが追加された場合、interfaces ファイルはチェックされません。このファイルは、初めてリモート・サーバに接続するときにチェックされます。この問題を解決するには、コンポーネント統合サービスによって使用されている interfaces ファイルにリモート・サーバを追加します。

2 つの終了ポイントを接続するためのドライバ呼び出しが失敗した

リモート・サーバが interfaces ファイルで定義されていて、接続要求から応答が返ってこない場合は、次のメッセージが表示されます。

```
Driver call to connect two endpoints failed
11206 Unable to connect to server server_name.
```

- 環境が正しく設定されているか確認する。

  環境をテストするには、isql または類似のツールを使用してリモート・サーバに直接接続します。

  a　コンポーネント統合サービスが動作しているマシンにログインする。

  b　SYBASE 環境変数を、コンポーネント統合サービスの起動時に使用されていたのと同じ場所に設定する。*runserver* ファイル内で -i 引数によって書き換えられていないかぎり、コンポーネント統合サービスは、SYBASE 環境変数によって指定されたディレクトリにある interfaces ファイルを使用します。

  注意 リモート・サーバに接続できなかったときに、コンポーネント統合サービスが使用していたのと同じ条件のテスト環境を確保するため、これらの最初の 2 つの手順は重要です。

  c　isql または類似するツールを使用して、リモート・サーバに直接接続する。

  環境が正しく設定されていても接続が失敗する場合、以降の手順を続行してください。接続できた場合は、コンポーネント統合サービスが使用している環境に問題があることになります。

- リモート・サーバが起動し、動作しているか確認する。

  リモート・サーバがあるマシンにログインして、サーバが動作しているか確認します。サーバが動作している場合、以降の手順を続行します。サーバが動作していない場合、サーバを再起動してもう一度クエリを実行します。

- interfaces ファイルにあるリモート・サーバのエントリが正しいか確認する。
  - interfaces ファイル内のマシン名が、ソフトウェアがロードされているマシンの正しい名前であるか確認する。
  - interfaces ファイルがテキスト・ファイルの場合、master 行と query 行がスペースではなくタブで始まっているか確認する。
  - ポート番号が使用可能かどうかを確認する。*/etc* ディレクトリにある *services* ファイルを見て、ポート番号が他の処理用に予約されていないか確認する。

ログインに失敗した

リモート・サーバにアクセスできるが、ログイン名とパスワードが正しくない場合、次のメッセージが表示されます。

```
Login failed
11206 Unable to connect to server server_name.
```

次のコマンドを実行して、外部からリモート・サーバにログインされていないか確認します。

```
exec sp_helpexternlogin server_name
```

外部ログインが定義されていない場合、コンポーネント統合サービスは Adaptive Server への接続に使うユーザ・ログイン名とパスワードを使用します。たとえば、Adaptive Server に接続したユーザが “sa” アカウントを使用している場合、コンポーネント統合サービスはリモート接続の際に “sa” をログイン名として使用します。リモート・サーバが別の Adaptive Server でないかぎり “sa” アカウントはおそらく存在しないので、sp_addexternlogin を使用して外部ログインを追加する必要があります。

外部ログインが定義されている場合は、ユーザのログイン名が正しいか確認します。リモート・サーバのログインでは、大文字と小文字が区別されます。使用しているユーザ・ログイン名や *externlogins* のエントリで、大文字と小文字が正しいか確認します。

ログイン名が正しい場合は、パスワードに問題がある可能性があります。パスワードを表示することはできません。ユーザ・ログイン名やパスワードに問題がある場合は、次のコマンドを実行して外部ログインを削除し、再定義します。

```
exec sp_dropexternlogin server_name, login_name
go
 exec sp_addexternlogin server_name, login_name, remote_login,
remote_password
go
```

リモート・オブジェクトにアクセスできない

リモート・オブジェクトにアクセスできない場合、次のエラー・メッセージが表示されます。

```
Error 11214  Remote object object does not exist.
```

ローカル・プロキシ・テーブルの定義、またはリモート・サーバのテーブル自体に問題がある可能性があります。

次のことを確認します。

- オブジェクトがコンポーネント統合サービスで定義されているか確認する。

  確認するには、次のコマンドを実行します。

  ```
  sp_help object_name
  ```

  オブジェクトが存在しない場合は、コンポーネント統合サービスでオブジェクトを作成します。

- コンポーネント統合サービスでオブジェクトが定義されている場合、その定義が正しいか確認する。

  テーブル名には、*server.dbname.owner.tablename* という形式の *4* つのパートが含まれていることがあります。*dbname* の部分は、Oracle または InfoHUB のサーバに対しては無効です。

  オブジェクト定義が正しくない場合、sp_dropobjectdef を使用して定義を削除し、sp_addobjectdef を使用して正しく定義します。

- ローカルのオブジェクト定義が正しい場合、リモート・サーバのテーブルを確認する。次の事項を確認する。
  - データベースとテーブルの両方にアクセスするためのパーミッションが設定されているか確認する。
  - データベースに suspect のマークが付いているか確認する。
  - データベースが使用可能か確認する。
  - ネイティブのツール (Oracle 上の SQL*Plus など) を使用してリモート・テーブルにアクセスできるか確認する。

リモート・オブジェクトからデータを取り出す際の問題

リモート・オブジェクト間の不一致に関するエラー・メッセージが返された場合は、コンポーネント統合サービスのオブジェクト定義がリモート・オブジェクト定義と一致していません。次のような場合にこの問題が発生します。

- オブジェクト定義がコンポーネント統合サービスの外部で変更された場合。
- インデックスがコンポーネント統合サービスの外部で追加または削除された場合。

オブジェクトがコンポーネント統合サービスの外部で変更された場合

コンポーネント統合サービスでオブジェクトが定義されると、リモート・サーバで行われたオブジェクトに対する変更は、ローカル・プロキシ・オブジェクト定義に対して行われません。オブジェクトがコンポーネント統合サービス外部で変更された場合、オブジェクトの定義に create existing table と create table のどちらが使用されたかによって、問題解決の手順が異なります。

どちらのメソッドが使用されたか確認するには、次のコマンドを実行します。

```
sp_help object_name
```

オブジェクトが create existing table を介して定義されている場合、次のメッセージが結果セットに返されます。

```
Object created with 'existing' option
```

このメッセージが表示されない場合、オブジェクトは create table を介して定義されています。

create existing table を使用してコンポーネント統合サービスでテーブルが作成されている場合は、次の手順に従います。

1 コンポーネント統合サービスで drop table を使用します。

2 create existing table を使用して、コンポーネント統合サービスでテーブルを再作成します。リモート・サーバ上にあるテーブルの新しいバージョンを使用して、テーブルが作成されます。

create table を使用してコンポーネント統合サービスにテーブルが作成されている場合は、drop table を使用する際にリモート・オブジェクトが削除されます。削除されないようにするには、次の手順に従います。

1 リモート・サーバにあるテーブルの名前を変更して、drop table を使用する際にこのテーブルが削除されないようにします。

2 リモート・サーバ上に、元の名前を使ってテーブルを作成します。

3 コンポーネント統合サービスで drop table を使用して、コンポーネント統合サービスにあるテーブルとリモート・サーバにあるテーブルを削除します。

4 リモート・サーバ上で、手順 1 で保存したテーブルの名前を元の名前に変更します。

5 create existing table を使用して、コンポーネント統合サービスでテーブルを再作成します。

警告！ リモート・サーバ上でテーブルの名前を変更する前にコンポーネント統合サービスで drop table を使用しないでください。リモート・サーバ上のテーブルが削除されます。

コンポーネント統合サービス内でオブジェクトを作成するには、リモート・サーバ上にオブジェクトを作成してから create existing table を実行することをおすすめします。こうすることで、より少ない手順で、リモート・サーバ上のオブジェクトを削除することなく、不一致の問題を解決できます。

コンポーネント統合サービス外部でのインデックスの追加または削除

コンポーネント統合サービスは、コンポーネント統合サービス外部で追加または削除されたインデックスについて関知しません。コンポーネント統合サービスで使用されるインデックスがリモート・サーバで使用されるインデックスと同じものか確認します。コンポーネント統合サービスの使用するインデックスを確認するには、sp_help を使用します。リモート・サーバが使用するインデックスを確認するには、リモート・サーバ上で適切なコマンドを使用します。

インデックスが同じでない場合、オブジェクトの定義に create existing table と create table のどちらを使用してオブジェクトが定義されたかによって、問題解決の手順が異なります。

どちらのメソッドが使用されたか確認するには、次のコマンドを実行します。

```
sp_help object_name
```

オブジェクトが create existing table コマンドを介して定義されている場合、次のメッセージが結果セットに返されます。

```
Object created with 'existing' option
```

このメッセージが表示されない場合、オブジェクトは create table を介して定義されています。

create existing table を使用してオブジェクトが作成されている場合は、次の手順に従います。

1　コンポーネント統合サービスで drop table を使用します。

2　create existing table を使用して、コンポーネント統合サービスでテーブルを再作成します。これで、リモート・テーブルにあるインデックスと一致するようにインデックスが更新されます。

create table がオブジェクトの作成に使用されている場合は、次の手順に従います。

1　drop index を使用して、リモート・テーブルからインデックスを削除します。

2　create index を使用して、コンポーネント統合サービスでインデックスを再作成します。これで、コンポーネント統合サービスとリモート・サーバに同じインデックスが作成されます。

create table によってオブジェクトが定義されている場合は、トレース・フラグ 11208 をオンにする方法もあります。このトレース・フラグをオンにすると、リモート・サーバに create index 文を送信できなくなります。トレース・フラグ 11208 を使用するには、次の手順に従います。

1 トレース・フラグ 11208 をオンにします。

```
dbcc traceon(11208)
```

2 create index を使用してインデックスを作成します。

3 トレース・フラグ 11208 をオフにします。

```
dbcc traceoff(11208)
```

不明な点があるときは

マニュアルやオンライン・ヘルプでは解決できない問題があった場合には、担当者を通して Sybase のサポート・センタまでご連絡ください。より迅速に問題を解決するため、Sybase 製品の保守契約を結んでいるサポート・センタに問い合わせる前に、次のような情報を収集しておいてください。

- リモート・データへのアクセス中に問題が発生した場合、ローカル・テーブルに対して同じスクリプトを実行する。ローカル・テーブルで問題がない場合、これはコンポーネント統合サービスに固有の問題である。以降の手順を続行する。
- 使用している Adaptive Server のバージョンを確認する。

  ```
  select @@version
  ```

- 該当の問題を再現した SQL スクリプトを記録する。テーブルを作成するのに使用したスクリプトも含む。
- クエリの処理プランを検索する。これは、set showplan を使用して生成されます。次に例を示す。

  ```
  set showplan, noexec on
   go
   select au_lname, au_fname from authors
   where au_id = 'A1374065371'
   go
  ```

  このクエリの出力は、次のように表示されます。

```
set showplan, noexec on
go
select au_lname, au_fname from authors where au_id = 'A1374065371'
 go
The Abstract Plan (AP) of the final query execution plan:
( remote_sql )
To experiment with the optimizer behavior, this AP can be modified and then
passed to the optimizer using the PLAN clause:
SELECT/INSERT/DELETE/UPDATE ...
PLAN '( ... )
```

```
QUERY PLAN FOR STATEMENT 1 (at line 1).

1 operator(s) under root


The type of query is SELECT.

ROOT:EMIT Operator

   |LE_REMSCANOP Operator
   |    SELECT "au_lname" , "au_fname" FROM pubs2.dbo."authors"
WHERE "au_
   |    id" = 'A1374065371'
```

noexec オプションでクエリがコンパイルされますが、実行はされません。noexec をオフにするまで、後続のコマンドは実行されません。

- トレース・フラグ 11201 ～ 11205 をオンにしてクエリを実行するときに、イベント・ログを取得する。これらのトレース・フラグは、次のログを取得する。
  - 11201 － クライアント接続、クライアント接続の切断、アテンション・イベント
  - 11202 － クライアントの言語、カーソルの宣言、動的準備、動的即時実行テキスト
  - 11203 － クライアント RPC イベント
  - 11204 － クライアントにルート指定されたメッセージ
  - 11205 － リモート・サーバとの対話
  - 11206 － ログ・ファイルとディレクトリの処理ステップ
  - 11207 － text 型と image 型の処理のロギングを行う

  トレース・フラグをオンにしてスクリプトを実行したら、*$SYBASE/install* ディレクトリ内のエラー・ログにロギングが表示されます。次に例を示します。

```
dbcc traceon (11201,11202,11203,11204,11205)
 go
 select au_lname, au_fname from authors
 where au_id = 'A1374065371'
 go
 dbcc traceoff (11201,11202,11203,11204,11205)
 go
```

エラー・ログの出力は、次のとおりです (各エントリの冒頭に表示されるタイムスタンプは、見やすくするため省略します)。

```
server  TDS_LANG, spid 15: command text:
select au_lname, au_fname from authors where au_id = 'A1374065371'

server  RemoteAccess constructed
server  EXECLANG, spid 15, server huntington0_19442, quickpass statement:

ELECT "au_lname" , "au_fname" FROM pubs2.dbo."authors" WHERE "au_id" =
'A1374065371'

server  BINDCOLS, spid 15: column 1, name au_lname, fmt.type 'CHAR', fmt.maxlen
40, fmt.stat 16, con.type 'VARCHAR', con.maxlen 40

server  BINDCOLS, spid 15: column 2, name au_fname, fmt.type 'CHAR', fmt.maxlen
20, fmt.stat 16, con.type 'VARCHAR', con.maxlen 20
server  BINDCOLS, spid 15: bind array size 50, total memory required is 4304 bytes

server  FETCH   , spid 15: cursor C1; ct_fetch() returned 0 rows; status -204

server RemoteAccess deleted
```

このトレースはグローバルであるため、トレース・フラグをオンにすると、実行されたすべてのクエリのログが取得されます。したがって、ログを取得したら、トレースをオフにしてください。また、サーバを停止し、エラー・ログの名前を変更し、サーバを再起動して、定期的にエラー・ログを消去してください。これで、新しいエラー・ログが作成されます。

索引

L

O

P

R

S

T

U

W

あ

い

え

お

か

き

く

こ

さ

し

す

せ

た

ち

て

と

は

ふ

へ

め

も

り

れ

ろ

わ