

SYBASE®

コンポーネント統合サービス・ユーザーズ・ガイド

Adaptive Server® Enterprise

15.5

ドキュメント ID : DC36501-01-1550-01

改訂 : 2009 年 10 月

Copyright © 2010 by Sybase, Inc. All rights reserved.

このマニュアルは Sybase ソフトウェアの付属マニュアルであり、新しいマニュアルまたはテクニカル・ノートで特に示されないかぎり、後続のリリースにも付属します。このマニュアルの内容は予告なしに変更されることがあります。このマニュアルに記載されているソフトウェアはライセンス契約に基づいて提供されるものであり、無断で使用することはできません。

このマニュアルの内容を弊社の書面による事前許可を得ずに、電子的、機械的、手作業、光学的、またはその他のいかなる手段によっても、複製、転載、翻訳することを禁じます。

マニュアルの注文

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

Sybase の商標は、Sybase trademarks ページ (<http://www.sybase.com/detail?id=1011207>) で確認できます。Sybase およびこのリストに掲載されている商標は、米国法人 Sybase, Inc. の商標です。® は、米国における登録商標であることを示します。

Java および Java 関連の商標は、米国およびその他の国における Sun Microsystems, Inc. の商標または登録商標です。

Unicode と Unicode のロゴは、Unicode, Inc. の登録商標です。

IBM および Tivoli は、International Business Machines Corporation の米国およびその他の国における登録商標です。

このマニュアルに記載されている上記以外の社名および製品名は、当該各社の商標または登録商標の場合があります。

Use, duplication, or disclosure by the government is subject to the restrictions set forth in subparagraph (c)(1)(ii) of DFARS 52.227-7013 for the DOD and as set forth in FAR 52.227-19(a)-(d) for civilian agencies.

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

目次

はじめに	vii
第 1 章 概要	1
第 2 章 コンポーネント統合サービスの概要	5
基本概念	5
アクセス・メソッド	6
サーバ・クラス	6
オブジェクト・タイプ	6
リモート・サーバへのインタフェース	7
プロキシ・テーブル	8
create table コマンドの使用法	8
create existing table コマンドの使用法	9
create proxy_table コマンドの使用法	10
プロキシ・テーブルとしてのリモート・プロシージャ	11
サーバの制限	14
カスケード・プロキシ・テーブル	17
プロキシ・データベース	18
ユーザ・プロキシ・データベース	18
システム・プロキシ・データベース	21
ファイル・システム・アクセス	23
セキュリティの考慮事項	23
ディレクトリ・アクセス	24
下位ディレクトリへの再帰	27
ファイル・アクセス	27
リモート・サーバ	29
サーバ・クラス ASEnterprise	30
サーバ・クラス ASAnywhere	30
サーバ・クラス ASIQ	30
サーバ・クラス direct_connect	30
サーバ・クラス sds	30
サーバ・クラス RPCServer	31

接続管理	31
interfaces ファイルを使用しないでリモート・サーバに接続する	32
LDAP ディレクトリ・サービス	32
SSL とのセキュア通信	33
セキュリティの問題	34
リモート・サーバ・ログイン	35
外部ログインのマッピング	36
リモート・サーバ接続フェールオーバー	37
リモート・サーバの機能	38
クエリ処理	38
処理手順	38
RPC 処理とコンポーネント統合サービス	43
サイト・ハンドラとアウトバウンド RPC	44
コンポーネント統合サービスとアウトバウンド RPC	44
RPC のテキスト・パラメータ	46
XJS/390 でのテキスト・パラメータのサポート	47
分散トランザクション管理	48
サーバ・クラスと ASTC	48
DTM 対応サーバ	49
Pre-DTM サーバ	49
strict DTM enforcement	50
enable xact coordination	50
コンポーネント統合サービスを有効にする	50
トランザクション指向 RPC	50
トランザクション管理の制限	51
Adaptive Server に対する Adaptive Server update statistics	51
制限事項	52
Adaptive Server 以外のバックエンドに対する update statistics	52
データベースにおける Java	53
@@textsize	53
@@stringsize	53
Java クラスのカラムの制約	54
エラー・メッセージ	54
Java ADT (抽象データ型)	54
データ型	55
Unicode のサポート	56
データ型の変換	58
text および image データ型	59
設定とチューニング	63
sp_configure の使用	63
ステータスのグローバル変数	67

第 3 章	SQL リファレンス	69
	dbcc コマンド	69
	dbcc のオプション	70
	トレース・フラグ	70
	関数	72
	コンポーネント統合サービスでの関数のサポート	72
	集合関数	72
	データ型変換関数	73
	日付関数	73
	数学関数	73
	セキュリティ関数	74
	文字列関数	75
	システム関数	76
	text 関数および image 関数	77
	Transact-SQL コマンド	77
	alter table	80
	case	80
	connect to...disconnect	80
	create existing table	81
	create index	87
	create table	88
	delete	89
	drop index	90
	fetch	91
	insert	91
	readtext	92
	select	93
	truncate table	94
	update	95
	update statistics	96
	writetext	97
	パススルー・モード	97
	connect to	97
	sp_autoconnect	99
	sp_passthru	100
	sp_remotesql	100
	引用符付き識別子のサポート	101
	区切り識別子のサポート	102
	auto identity オプション	102
	トリガ	102

付録 A	チュートリアル	103
	コンポーネント統合サービスの使用開始	103
	リモート・サーバを追加するには	103
	2つのリモート・テーブルのジョイン.....	105
付録 B	トラブルシューティング	109
	コンポーネント統合サービスにアクセスする際の問題.....	109
	コンポーネント統合サービス使用中の問題.....	110
	リモート・サーバにアクセスできない	110
	リモート・オブジェクトにアクセスできない	113
	リモート・オブジェクトからデータを取り出す際の問題	113
	不明な点があるときは	116
索引		119

はじめに

対象読者

このマニュアルは、Sybase® Adaptive Server® Enterprise のシステム管理者、データベース管理者、ユーザを対象にしています。

このマニュアルの内容

このマニュアルでは、コンポーネント統合サービスの設定方法と使用方法について説明します。このマニュアルの内容は次のとおりです。

- 「[第 1 章 概要](#)」では、コンポーネント統合サービスの概要について説明します。
- 「[第 2 章 コンポーネント統合サービスの概要](#)」では、コンポーネント統合サービスの働きについて説明します。この章では、基本概念と詳細の両方について説明します。
- 「[第 3 章 SQL リファレンス](#)」では、コンポーネント統合サービスを利用する Transact-SQL® コマンドについて説明します。
- 「[第 A 章 チュートリアル](#)」は、初めてお使いになる方がコンポーネント統合サービスを起動、実行する際の手助けとなるチュートリアルです。
- 「[第 B 章 トラブルシューティング](#)」では、コンポーネント統合サービスに関する問題が起こった場合のトラブルシューティングについて説明します。

関連マニュアル

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

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

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

- 使用しているプラットフォームの『インストール・ガイド』— すべての Adaptive Server および関連する Sybase 製品のインストール、アップグレード、設定の手順について説明しています。
- 『新機能ガイド』— Adaptive Server の新しい機能について説明しています。また、新しい機能をサポートするためのシステム変更や、既存のアプリケーションに影響を与える可能性がある変更についても説明しています。

-
- 『Active Messaging ユーザーズ・ガイド』 – Active Messaging を使用して、Adaptive Server Enterprise データベースでトランザクション (データ変更) を取得し、外部アプリケーションにイベントとしてリアルタイムで渡す方法について説明しています。
 - 『コンポーネント統合サービス・ユーザーズ・ガイド』 – コンポーネント統合サービスを使用して、リモートの Sybase データベースおよび Sybase 以外のデータベースに接続する方法について説明しています。
 - 使用しているプラットフォームの『設定ガイド』 – 特定の設定作業の手順について説明しています。
 - 『用語解説』 – Adaptive Server マニュアルで使用されている技術用語について説明しています。
 - 『Historical Server ユーザーズ・ガイド』 – Historical Server を使用して、Adaptive Server のパフォーマンス情報を入手する方法について説明しています。
 - 『Adaptive Server Enterprise における Java』 – Adaptive Server データベースで Java クラスをデータ型、関数、ストアド・プロシージャとしてインストールして使用する方法について説明しています。
 - 『Job Scheduler ユーザーズ・ガイド』 – コマンド・ラインまたはグラフィカル・ユーザ・インタフェース (GUI) を使用して、ローカルまたはリモートの Adaptive Server でジョブのインストール、設定、作成、スケジュールを行う方法について説明しています。
 - 『マイグレーション技術ガイド』 – 別のバージョンの Adaptive Server にマイグレートするための方法とツールについて説明しています。
 - 『Monitor Client Library プログラマーズ・ガイド』 – Adaptive Server のパフォーマンス・データにアクセスする Monitor Client Library アプリケーションの記述方法について説明しています。
 - 『Monitor Server ユーザーズ・ガイド』 – Monitor Server を使用して、Adaptive Server のパフォーマンス統計を取得する方法について説明しています。
 - 『モニタリング・テーブル・ダイアグラム』 – モニタリング・テーブルと、そのエンティティの関係をポスター形式で図解しています。フル・サイズのダイアグラムは印刷版だけで参照できます。コンパクト版は PDF 形式で参照できます。

- 『パフォーマンス&チューニング・シリーズ』 – Adaptive Server で最高のパフォーマンスを実現するためのチューニング方法について説明しています。
 - 『基本』 – Adaptive Server のパフォーマンスに関する問題の理解と調査の基本について説明しています。
 - 『統計的分析によるパフォーマンスの向上』 – Adaptive Server で統計情報がどのように保存され、表示されるかについて説明しています。また、`set statistics` コマンドを使用して、サーバの統計情報を分析する方法について説明しています。
 - 『ロックと同時実行制御』 – ロック・スキームを使用してパフォーマンスを向上させる方法と、同時実行性を最小限に抑えるようにインデックスを選択する方法について説明しています。
 - 『sp_sysmon による Adaptive Server の監視』 – `sp_sysmon` を使用してパフォーマンスをモニタリングする方法について説明しています。
 - 『モニタリング・テーブル』 – Adaptive Server のモニタリング・テーブルに統計情報や診断情報を問い合わせる方法について説明しています。
 - 『物理データベースのチューニング』 – データの物理的配置、データに割り付けられた領域、テンポラリ・データベースの管理方法について説明しています。
 - 『クエリ処理と抽象プラン』 – オプティマイザがクエリを処理する方法と、抽象プランを使用してオプティマイザのプランの一部を変更する方法について説明しています。
- 『クイック・リファレンス・ガイド』 – コマンド、関数、システム・プロシージャ、拡張システム・プロシージャ、データ型、ユーティリティの名前と構文の包括的な一覧表を記載したポケット版 (PDF 版は通常サイズ) のマニュアルです。
- 『リファレンス・マニュアル』 – 詳細な Transact-SQL 情報を記載しています。
 - 『ビルディング・ブロック』 – データ型、関数、グローバル変数、式、識別子とワイルドカード、予約語について説明しています。
 - 『コマンド』 – コマンドについて説明しています。
 - 『プロシージャ』 – システム・プロシージャ、カタログ・ストア・プロシージャ、システム拡張ストア・プロシージャ、`dbcc` ストアド・プロシージャについて説明しています。
 - 『テーブル』 – システム・テーブル、モニタリング・テーブル、`dbcc` テーブルについて説明しています。

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

- 『Unified Agent および Agent Management Console』 – Unified Agent について説明しています。Unified Agent は、分散 Sybase リソースを管理、モニタ、制御するためのランタイム・サービスを提供します。
- 『ユーティリティ・ガイド』 – オペレーティング・システム・レベルで実行される `isql` および `bcp` などの、Adaptive Server のユーティリティ・プログラムについて説明しています。
- 『Web Services ユーザーズ・ガイド』 – Adaptive Server 用の Web サービスの設定、使用、トラブルシューティング方法について説明しています。
- 『XA インタフェース統合ガイド for CICS, Encina, TUXEDO』 – X/Open XA トランザクション・マネージャを備えた Sybase DTM XA インタフェースを使用する方法について説明しています。
- 『Adaptive Server Enterprise における XML サービス』 – データベースに XML 機能を導入する、Sybase ネイティブの XML プロセッサと Sybase Java ベースの XML のサポートについて、また XML サービスに準拠したクエリとマッピング用の関数について説明しています。

その他の情報

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

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

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

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

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

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

Sybase Web サイトの技術的な資料は頻繁に更新されます。

❖ **EBF とソフトウェア・メンテナンスの最新情報にアクセスする**

- 1 Web ブラウザで **Sybase Support Page** を指定します。
(<http://www.sybase.com/support>)
- 2 [EBFs/Maintenance] を選択します。MySybase のユーザ名とパスワードを入力します。
- 3 製品を選択します。
- 4 時間枠を指定して [Go] をクリックします。EBF/Maintenance リリースの一覧が表示されます。

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

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

❖ **コンポーネント認定の最新情報にアクセスする**

- 1 Web ブラウザで **Availability and Certification Reports** を指定します。
(<http://certification.sybase.com/>)
- 2 [Search By Base Product] で製品ファミリとベース製品を選択するか、[Search by Platform] でプラットフォームとベース製品を選択します。
- 3 [Search] をクリックして、入手状況と認定レポートを表示します。

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

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

- 1 Web ブラウザで **Technical Documents** を指定します。
(<http://www.sybase.com/support/techdocs/>)
- 2 [MySybase] をクリックし、MySybase プロファイルを作成します。

Sybase EBF とソフトウェア・メンテナンス

❖ EBF とソフトウェア・メンテナンスの最新情報にアクセスする

- 1 Web ブラウザで Sybase Support Page を指定します。
(<http://www.sybase.com/support>)
- 2 [EBFs/Maintenance] を選択します。MySybase のユーザ名とパスワードを入力します。
- 3 製品を選択します。
- 4 時間枠を指定して [Go] をクリックします。EBF/Maintenance リリースの一覧が表示されます。

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

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

表記規則

次の項では、このマニュアルで使用されている表記について説明します。

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

表 1 に構文の規則を示します。

表 1: このマニュアルでのフォントと構文規則

要素	例
コマンド名、プロシージャ名、ユーティリティ名、その他のキーワードは sans serif フォントで表記する。	<code>select</code> <code>sp_configure</code>
データベース名とデータ型は sans serif フォントで表記する。	<code>master</code> データベース
ファイル名、変数、パス名は斜体で表記する。	システム管理ガイド <i>sql.ini</i> ファイル <i>column_name</i> <i>\$\$SYBASE/ASE</i> ディレクトリ
変数 (ユーザが入力する値を表す語) がクエリまたは文の一部である場合は Courier フォントの斜体で表記する。	<code>select column_name</code> <code>from table_name</code> <code>where search_conditions</code>
カッコはコマンドの一部として入力する。	<code>compute row_aggregate (column_name)</code>

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

- 次は、オプション句のあるコマンドの構文の例です。

```
sp_dropdevice [device_name]
```

複数のオプションを持つコマンドの例を示します。

```
select column_name  
from table_name  
where search_conditions
```

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

- Transact-SQL コマンドの使用例は次のように表記します。

```
select * from publishers
```

- 次は、コンピュータからの出力例です。

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

(3 rows affected)

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

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

アクセシビリティ機能

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

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

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

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

不明な点があるときは

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



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ます。
- **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します。
- 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します。「[第 2 章 コンポーネント統合サービスの概要](#)」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。

トピック	ページ
基本概念	5
プロキシ・テーブル	8
プロキシ・データベース	18
ファイル・システム・アクセス	23
リモート・サーバ	29
クエリ処理	38
RPC 処理とコンポーネント統合サービス	43
分散トランザクション管理	48
Adaptive Server 以外のバックエンドに対する update statistics	52
データベースにおける Java	53
データ型	55
設定とチューニング	63

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 の場合に使用します。これがデフォルトのサーバ・クラスです。
- *ASAnywhere* – サーバが Adaptive Server Anywhere バージョン 6.0 以降の場合に使用します。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 に対して使用してください。
- *ASIQ* – サーバが Adaptive Server IQ バージョン 12.5 以降の場合に使用します。
- *local* – ローカル・サーバ。1 つだけ存在できます。
- *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します。Microsoft SQL Server、DB2、Oracle、または Informix にアクセスするには、DirectConnect サーバを使用します。
- *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します。
- *RPCServer* – サーバが RPC のみ処理できることを示します。SQL 文、トランザクション制御文などは処理できません。

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・テーブルです。これがデフォルトのタイプです。
- **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す。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います。

- **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・プロシージャです。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います。
- **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す。
- **directory** – オブジェクトは、ファイル・システム・ディレクトリです。

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「[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・サーバへのパズスルー・モード接続を確立します。パズスルー・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。このパズスルー・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](#)」(6 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}] 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ます。
- 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ます。カラム名 (大文字と小文字が区別されます)、カラムの型と長さ、カラムの NULL 属性が一致しなければなりません。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りません。
- 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します。これにより、サーバ用語のインデックスとキーが定義されて、クエリ・オプティマイザがこのテーブルに存在する可能性のあるすべてのインデックスを考慮できるようになります。

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** テーブルを定義する。

```
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する。

```
update statistics authors
```

- 5 クエリを実行して、設定をテストする。

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

create proxy_table コマンドの使用法

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**、**syscolumnms**、**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ません。
- 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ります。リストの正確さは保証されません。
- 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ます。これらのカラムは、パラメータ・カラムと呼ばれます。次に例を示します。

```
create existing table t1
(
    a int,
    b int,
    c int,
    _p1 int null,
    _p2 int null
)
external procedure
at "SYBASE.sybssystemprocs.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ていないことになっていますが、クエリで参照できます。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、探索指数をリモート・プロシージャに渡します。
- 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ります。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たりする場合、その値は `NULL` になります。

- 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ます。“or” 述部に含まれない場合、通常は SARG です。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ません。

```
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ります。
 - 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はあってはならない(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ません。
- 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ません。

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。その代わりに、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します。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します。

```
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ません。次に例を示します。

```
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ます。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ード・プロシージャ (または RPC) に渡すことができます。
- 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 カラムです (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 に制限されています (null カラムも可変長と見なされます)。
- 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ます。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します。

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 バイトです。
- 識別子名は最大 255 バイトです。

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、コマンドはアポートされます。

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ポートされ、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)TM 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します。

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ます。

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ます。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りません。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}`) を使用します。

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` にのみ当てはまります) 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ります。

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みリモート・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します。`master.dbo.sys.servers.srvname` に存在している必要があります。
- `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んでいます。
- `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す。指定すると、使用が制限される場合があります。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ります。ほかのユーザが所有しているオブジェクトのプロキシ・テーブルは作成しないでください。

`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・サイズを見積ります。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ます。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ます。

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・テーブルを作成します。システム・テーブル用のプロキシ・テーブルは作成されません。
- プロキシ・テーブルのすべてのパーミッションを“public”に付与します。
- “guest” ユーザをプロキシ・データベースに追加します。
- リモート・サイトからデータベース・ユーザをインポートします (Adaptive Server の場合)。
- create table パーミッションを“public”に付与します。
- 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します。ステータスを設定するには、*master.dbo.sysdatabases.status3* (0x0001, DBT3_USER_PROXYDB) でステータス・フィールドを設定します。

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。その代わりに、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ません。
- 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ません。`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ます。
- 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ます。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ません。
- 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します。`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ません。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ます。

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、カスタム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ります。

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ぬことがあります。

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ます。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ます。
- **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ます。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ます。
- **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ます。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・テーブルが作成または削除されます。

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ます。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・サーバに転送されます。
- **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ます。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ます。
- **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ません。

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 **sybssystemprocs** データベースで探して実行しようとします。**sybssystemprocs** 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せん。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。

カラム名	データ型	説明
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ます。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ます。
- **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ます。意味のあるカラムは、**filename**、**filetype**、**content** のみです。その他のカラムは、**insert** 文には指定しないでください。指定しても無視されます。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** カラムは無視されます。

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ます。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す。次に例を示します。

```
/* 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ます。

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ます。
- **writetext** — **readtext** コマンドを使用すると、ファイルの内容を修正できます。

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 frame (
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時、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ます。`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ます。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ます。最後のカラムは改行文字で終了します。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します。
- `bcp in` と `bcp out` の代わりに使用できる SQL 構文を提供できます。`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ます。
- `select` 文を使用してファイルの内容を照会し、必要に応じて探索指数や関数を使用してローを特定できます。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ます。

```
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 つしかないため、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 `sys.servers` 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]]
```

各オブジェクトの意味は、次のとおりです。

- `server_name` は、サーバを識別するための名前です。
- `server_class` は、サーバのタイプです。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、以降の項で説明します。デフォルトは、サーバ・クラス `ASEnterprise` です。

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す。この名前は、`server_name` と同じ場合もあれば異なる場合もあります。`network_name` は、`physical name` と呼ばれることもあります。デフォルトは、`server_name` と同じ名前です。

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します。
- `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します。
- `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します。
- `SRV_RPC - CIS` からの RPC を処理します。RPC の処理により結果セットが生成される場合があります。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 によって転送されます。

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ません。
- Replication Agent スレッドでこの構文が認識されません。

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します。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・ファイルを使用します。
- 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ます。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します。

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。

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：

```
%SYBASE%%$$SYBASE_ASE%certificates#servername.txt
```

servername は現在の Adaptive Server の名前です。

セキュリティの問題

リモートの Adaptive Server に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 関数を使用されます。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ます。
- `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ます。encrypt キーワードは `create existing table` の `column_list` では使用できません。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します。
- 暗号化カラムがある場所では `create table` は使用できません。
- `alter table` は、プロキシ・テーブルの暗号化カラムでは使用できません。
- `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ます。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します。

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。

- `enctype` – ディスク上のデータ型
- `enclen` – 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ます。

- 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ます。
- 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ます。
- 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します。

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ー

`interfaces` ファイル (または LDAP ディレクトリ・サービス) でフェールオーバー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ー・サーバへの接続を自動的にフェールオーバーします。

次の設定手順を実行した後で、フェールオーバー用としてリモート・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、フェールオーバー・サーバを指定する。

たとえば、サーバ S2 をサーバ S1 のフェールオーバー・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 フェールオーバーの使用』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ます。
- 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します。

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 文を生成する場合があります。

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・ 옵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 **ASEnterprise** 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)。

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、算術オーバーフローが発生します。

`sp_addserver` を使用して新しいサーバを `syssservers` 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・ハンドラが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 `syservers` 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](#)」(35 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します。

```
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します。

```
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經由し、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まれます。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ます。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します。
- 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ます。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ます。
- 接続要求は、通常のクライアント接続としてリモート・サーバに表示されます。`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ません。`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` モードは使用できません)、Adaptive Server のリモート・サーバ管理機能に影響があります。

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 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_parameter 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` 句に含まれる検索指数から派生します。`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ます。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ます。
- 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ます。

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_rpc** 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ります。
- 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ます。
- 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ます。

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なります。

```
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」\(55 ページ\)](#) 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。

@@stringize

`@@stringize` 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 stringize 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 `@@stringize` 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。詳細については、「[トレース・フラグ](#)」(70 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ています。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ーヘッドにより、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 コマンドはアポートされます。

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 定数値の変換に係る。
- 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に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) はできません。
- ローカル変数として使用できません。
- order by 句、compute 句、group by 句内では使用できません。
- インデックス内での使用はできません。
- サブクエリ内での使用はできません。
- where 句内での使用 (キーワード like と併用する場合を除く) はできません。
- ジョインでは使用できません。

@@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 が埋めこまれます (“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。

insert コマンドを使用して text 値または image 値を挿入する場合、データの最大長は 16KB 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ーヘッドは、パフォーマンスに影響を与えます。
- 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をなければなりません。

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ます。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ります。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ます。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 コマンドを使用します。
- 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ります。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 を使用します。
- 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ます。
- 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ります。
- 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す。
- 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ますが、ユニーク・インデックスが必要です。

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ます。*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してください。
- 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します。`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りません。
- *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ます。
 - `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します。
 - `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・テーブルに格納します。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ます。カラム・サイズは、`@@textsize` によって定義されます。この値が十分でない場合、エラーが返されます。
- 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ります。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します。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ります。

- 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ります。
- `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す。

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します。
 - `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 を再起動します。

`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」 \(43 ページ\)](#) 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` の値が含まれます。
- `@@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` の値が含まれます。

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` の現在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。

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。

トピック	ページ
dbcc コマンド	69
関数	72
Transact-SQL コマンド	77
パススルー・モード	97
引用符付き識別子のサポート	101
区切り識別子のサポート	102
auto identity オプション	102
トリガ	102

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、sys.servers 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
```

例：

```
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 (グローバル)
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のロギングを行う (グローバル)
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)
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)
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)
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 <i>type</i> 、 <i>length</i> 、 <i>nullability</i> 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)。
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。

トレース・フラグ	説明
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、整数型も受け入れられます。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
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ます。
- 構文 – コマンドの完全な Transact-SQL 構文の説明が記述されています。
- 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ます。
- 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ます。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ます。
- 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ます。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ます。
- 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ます。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ます。
- 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ます。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ます。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 にも当てはまります。

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します。
- **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、この句も転送されます。

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す。
- **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ます。

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します。

```
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しますが、**alter table** のサポートは必須です。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** を転送します。
- 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ります。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ります。
- サーバ・クラス **direct_connect** は、**bigint**、**unsigned tinyint**、**unsigned smallint**、**unsigned int**、**unsigned bigint** をサポートしていません。
- 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・サーバに送られます。構文機能が DB2 SQL を示す場合は、DB2 データ型が送られます。

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(<i>n</i>)	binary(<i>n</i>)
bit	bit
char	char
date	date
datetime	datetime
decimal(<i>p</i> , <i>s</i>)	decimal(<i>p</i> , <i>s</i>)
float	float
image	image
int	int
money	money
numeric(<i>p</i> , <i>s</i>)	numeric(<i>p</i> , <i>s</i>)
nchar(<i>n</i>)	nchar(<i>n</i>)
nvarchar(<i>n</i>)	nvarchar(<i>n</i>)
real	real
smalldatetime	smalldatetime
smallint	smallint
smallmoney	smallmoney
time	time
timestamp	timestamp
tinyint	tinyint
text	text
unichar	unichar
unitext	unitext
varbinary(<i>n</i>)	varbinary(<i>n</i>)
varchar(<i>n</i>)	varchar(<i>n</i>)

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ます。

`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します。

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(<i>n</i>)	image, binary(<i>n</i>), varbinary(<i>n</i>)。image ではない場合、長さは一致する必要がある。
bit	bit, decimal, float, int, money, numeric, real, smallint, smallmoney, tinyint
char(<i>n</i>)	text, nchar(<i>n</i>), nvarchar(<i>n</i>), char(<i>n</i>), varchar(<i>n</i>), unichar, univarchar。text ではない場合、長さは一致する必要がある。
datetime	datetime, smalldatetime, char, varchar
decimal(<i>p</i> , <i>s</i>)	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(<i>n</i>)	text, nchar(<i>n</i>), nvarchar(<i>n</i>), char(<i>n</i>), varchar(<i>n</i>)。text ではない場合、長さは一致する必要がある。
numeric(<i>p</i> , <i>s</i>)	bit, decimal, float, int, money, numeric, real, smallint, smallmoney, tinyint
nvarchar(<i>n</i>)	text, nchar(<i>n</i>), nvarchar(<i>n</i>), char(<i>n</i>), varchar(<i>n</i>), unichar, univarchar。text ではない場合、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

リモート Adaptive Server データ型	使用できる Adaptive Server データ型
univarchar	char, varchar, unichar, univarchar, text, datetime, smalldatetime
unitext	unitext
varbinary(<i>n</i>)	image, binary(<i>n</i>), varbinary(<i>n</i>). image でない場合、長さは一致する必要がある。
varchar(<i>n</i>)	text, nchar(<i>n</i>), nvarchar(<i>n</i>), char(<i>n</i>), varchar(<i>n</i>), 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

リモート Adaptive Server データ型	使用できる Adaptive Server データ型
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 `unifext` 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します。

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(<i>n</i>)	image, binary(<i>n</i>), varbinary(<i>n</i>). 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(<i>n</i>)	text, nchar(<i>n</i>), nvarchar(<i>n</i>), char(<i>n</i>), varchar(<i>n</i>), unichar, univarchar. text でない場合、長さは一致する必要がある。
datetime	datetime, smalldatetime
decimal(<i>p</i> , <i>s</i>)	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(<i>p</i> , <i>s</i>)	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

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

リモート Adaptive Server Anywhere データ型	使用できる Adaptive Server Anywhere データ型
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ります。
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ります。

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す。
- 動作はサーバ・クラス ASAnywhere の場合と同じになります。

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ます。
- 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ありませんが、表 3-13 に示すように、変換可能である必要があります。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します。

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(<i>n</i>)	image、binary(<i>n</i>)、varbinary(<i>n</i>)。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(<i>n</i>)	text、nchar(<i>n</i>)、nvarchar(<i>n</i>)、char(<i>n</i>) と varchar(<i>n</i>)、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(<i>p</i> , <i>s</i>)	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(<i>n</i>)	text、nchar(<i>n</i>)、nvarchar(<i>n</i>)、char(<i>n</i>) と varchar(<i>n</i>)、unichar、univarchar。長さが一致しない場合、コマンドはアボートする。
numeric (<i>p</i> , <i>s</i>)	bit、decimal、float、int、money、numeric、real、smallint、smallmoney、tinyint
nvarchar(<i>n</i>)	text、nchar(<i>n</i>)、nvarchar(<i>n</i>)、char(<i>n</i>) と varchar(<i>n</i>)、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(<i>n</i>)、nvarchar(<i>n</i>)、char(<i>n</i>)、varchar(<i>n</i>)、unichar、univarchar。text でない場合、長さは一致する必要がある。
univarchar	text、nchar(<i>n</i>)、nvarchar(<i>n</i>)、char(<i>n</i>)、varchar(<i>n</i>)、unichar、univarchar。text でない場合、長さは一致する必要がある。
date	
time	
bigint	UDB および DC/Microsoft は bigint をサポートします。

- データ型の情報は、パラメータに関連する CS_DATAFMT 構造体で渡されます。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ます。
 - datatype* – Adaptive Server データ型を表す CS_Library データ型。たとえば、CS_INT_TYPE などです。
 - 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します(『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 を支援します。

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します。
- 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します。
- 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します。

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・サーバに転送します。

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 に渡します。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します。
- DirectConnect は、bigint、unsigned tinyint、unsigned smallint、unsigned int、unsigned bigint をサポートしていません。
- 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ます。

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(<i>n</i>)	binary(<i>n</i>)
bit	bit
char	char
datetime	datetime
decimal(<i>p</i> , <i>s</i>)	decimal(<i>p</i> , <i>s</i>)
float	float
image	image
int	int
money	money
numeric(<i>p</i> , <i>s</i>)	numeric(<i>p</i> , <i>s</i>)
nchar(<i>n</i>)	nchar(<i>n</i>)
nvarchar(<i>n</i>)	nvarchar(<i>n</i>)
real	real
smalldatetime	smalldatetime

Adaptive Server データ型	デフォルトの DirectConnect データ型
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。
- 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 文に渡します。DirectConnect がサポートしていれば、言語文も使用できます。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ります。

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 *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ます。
- **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ます。**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ます。
- 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・サーバに転送されます。これを、クイックパス・モードと呼びます。**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・モードは使用されません。
- **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します。
- 計算カラムで **insert** コマンドを実行することはできません。

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ます。
- **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ますが、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す。**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・サーバに送信されます。
 - 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ります。

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 文に渡します。DirectConnect がサポートしていれば、言語文も使用できます。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ります。

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ります。
- 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・サーバに転送します。

```
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ます。`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ます。

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` カラムがあり、`blurb 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ます。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・サーバに転送されます。
- 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ます。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 `dboption` オプションを使用して設定される必要があります。

サーバ・クラス
ASAnywhere

- すべての構文がサポートされます。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・サーバに転送されます。
- `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ません。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します。

サーバ・クラス *ASIQ*

- すべての構文がサポートされます。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・サーバに転送されます。

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 により機能を要求します。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 に転送するか決定します。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ます。
- 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ます。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 文に渡します。DirectConnect がサポートしていれば、言語文も使用できます。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ります。
- `select...into` コマンドはサポートされますが、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す。
- `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ません。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します。

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します。
- `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ます。`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ません。代わりに、`writetext` コマンドを使用してください。
- 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ます。

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ます。

```
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、組み込み関数のサポートがあります。

- 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 文に渡します。DirectConnect がサポートしていれば、言語文も使用できます。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ります。

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します。

```
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します。

```
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** コマンドを発行します。
- 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** が必要です。
- 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ません。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ります。

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 の場合と同じです。
- 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ありません。

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、*sys.servers* 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 `sys.servers` 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・サーバからの結果は、出力パラメータに格納することもできます。sp_passthru の構文を次に示します。

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

パラメータ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)` です。
- *command* は、SQL コマンド・バッファで、データ型は `varchar(255)` です。
- *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` 出力です。
- *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)` 出力です。
- *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` 出力です。
- *arg1* – *argn* はオプションのパラメータです。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ります。データ型は異なる場合があります。いずれも出力パラメータでなければなりません。

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、『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す。
- `server_name` は、`varchar(255)` フィールドです。`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、プロシージャはアポートされます。このパラメータは必須です。
- `query_buf1` は、`char` または `varchar` 型のクエリ・バッファで、最大長は 255 バイトです。このパラメータは必須です。

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 fred_s_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 `fred_s_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する必要があります。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 : `quoted identifier`
- 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せん。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。

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・サーバを追加するには
- 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・サーバを追加するには

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

`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 `syssservers` 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]]
```

各パラメータの意味は、次のとおりです。

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

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]
```


各パラメータの意味は、次のとおりです。

- *remote_server* は、リモート・サーバの名前です。*remote_server* に指定するサーバ名は、**master.dbo.syssservers** 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りません。
- *login_name* は、ローカル・サーバに認識されているアカウントです。*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ります。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* アカウントだけです。
- *remote_name* は、*remote_server* に認識されるアカウントです。*remote_server* が稼働しているノード上で有効にしてください。*remote_server* へのログインにはこのアカウントが使用されます。
- *remote_password* は、*remote_name* のパスワードです。

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、`syssservers` 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]
```

各パラメータの意味は、次のとおりです。

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

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
```


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ー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ちます。
- 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。

トピック	ページ
コンポーネント統合サービスにアクセスする際の問題	109
コンポーネント統合サービス使用中の問題	110
不明な点があるときは	116

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してください。

```
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帰ってこない場合は、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します。

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。次の事項を確認します。
 - 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 existed prior to CIS.
```

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ず、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 existed prior to CIS.
```

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す。以降の手順を続行してください。
- 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まれます。
- 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。次に例を示します。

```
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、次のログを取得します。
 - 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

記号

- () (カッコ)
 - SQL 文内 xiii
- , (カンマ)
 - SQL 文内 xiv
- ::= (BNF 表記)
 - SQL 文内 xiv
- @@textsize グローバル変数 59
- [] (角カッコ)
 - SQL 文内 xiv
- { } (中カッコ)
 - SQL 文内 xiv

A

- ASTC サーバ・タイプ 48
- auto identity データベース・オプション 10

B

- Backus Naur Form (BNF) 表記 xiii, xiv
- bcp
 - text および image データ型 61
- BNF 表記、SQL 文内 xiii, xiv

C

- cis connect timeout 設定パラメータ 66
- cis packet size 設定パラメータ 66
- cis rpc handling 設定パラメータ 66
- Client-Library 関数 7
 - ct_send_data 60
 - 接続管理 31
- connect to オプション、付与 98
- connect to コマンド 97, 105
- create existing table 9, 10

- create existing table コマンド 9
 - データ型の変換 9
 - プロキシ・テーブル 80
 - 例 10
- create index コマンド 87
 - リモート・テーブルのクエリ・プラン 88
- create table コマンド
 - クエリ・プラン 89
 - プロキシ・テーブル 88
 - リモート・テーブル 9
- ct_send_data Client-Library 関数 60

D

- dbcc (データベース一貫性チェック) 45
- dbmoretext DB-Library 関数 60
- dbwritetext DB-Library 関数 60
- deallocate cursor コマンド
 - リモート・サーバ 89
- direct_connect サーバ・クラス
 - text および image データ型 62
- DirectCONNECT サーバ 3
- disconnect コマンド 98
- drop database コマンド
 - リモート・サーバ 90
- drop index コマンド
 - リモート・テーブルのクエリ・プラン 90
- drop table コマンド
 - プロキシ・テーブル 90
- DTM 対応サーバ 49

E

- enable cis 設定パラメータ 65

索引

G

grant connect to コマンド 98
grant コマンド
 パススルー接続 98

I

IDENTITY カラム 10
image データ型 59
 値の入力 60
 埋め込み 59
 エラー・ロギング 61
 サーバ・クラス direct_connect 62
 制限 59
 パターン一致 60
 変換 60
 リモート・サーバへのバルク・コピー 61
insert コマンド
 プロキシ・テーブル 91
interfaces ファイル
 リモート・サーバの追加 104

L

LDAP ディレクトリ・サービス 32
like キーワード 60
lock timeout interval 設定パラメータ 45

O

open コマンド 92

P

patindex 文字列関数 60
pre-DTM サーバ 49

R

readtext コマンド
 エラー 60
remcon オプション、dbcc 70

rollback コマンド
 リモート・サーバ 93
RPC 処理 17, 43
RPC の送信 44
rusage オプション、dbcc 70

S

sds サーバ・クラス 30
set コマンド
 リモート・クエリ 94
 「各設定オプション」参照
sp_addexternlogin システム・プロシージャ 104
sp_addserver システム・プロシージャ 104, 106
sp_autoconnect システム・プロシージャ 99
sp_capabilities システム・プロシージャ 38
sp_configure システム・プロシージャ 63
sp_passthru システム・プロシージャ 99
sp_remotelogin システム・プロシージャ 35
sp_remotesql システム・プロシージャ 100
sql.ini ファイル 104
SSL 33
sysconfigures システム・テーブル
 値の更新 64
syssservers システム・テーブル
 リモート・サーバ、コンポーネント
 統合サービス用 29, 104

T

text および image データ型のエラー・ロギング 61
text データ型 59
 値の入力 60
 埋め込み 59
 エラー・ロギング 61
 サーバ・クラス direct_connect 62
 制限 59
 パターン一致 60
 変換 60
 リモート・サーバへのバルク・コピー 61
textsize オプション、set 59
traceon/traceoff オプション、dbcc 70
Transactional RPC 50
transactional_rpc on オプション、set コマンド 50
trusted モード 35

U

- update statistics 52
- update statistics
 - プロキシ・テーブル 51
- update statistics コマンド
 - 完全な分散統計の取得 71
 - 既存のテーブルの定義 9
 - リモート・テーブル 52
- update コマンド
 - リモート・テーブル 94
- using オプション、readtext
 - エラー 60

W

- writetext コマンド
 - リモート・テーブル 96

あ

- アウトバウンド RPC 66
- アウトバウンド RPC の処理 44
- アクセス・メソッド 6

い

- イベント・ロギング 70
- インデックス
 - 更新 71
 - 定義 9
- 引用符付き識別子のサポート 101

え

- エイリアス、ユーザ
 - リモート・ログイン 104

お

- 大文字と小文字の区別
 - SQL xv
- オブジェクト・タイプ 6

か

- カーソル
 - ロー・カウント、設定 66
- 外部ログイン 104
- 外部ログインのマッピング 36
- 角カッコ []
 - SQL 文内 xiv
- 角カッコ。「角カッコ []」参照
- カッコ ()
 - SQL 文内 xiii
- カラム
 - インデックスをプロキシ・テーブル上に作成 87
- カンマ (,)
 - SQL 文内 xiv

き

- 記号
 - SQL 文内 xiii
- 規則
 - Transact-SQL の構文 xiii
 - リファレンス・マニュアル xiii
- 起動時リカバリ、無効化 71
- 規約
 - 「構文」参照

く

- クイックパス・モード 39, 91
- クエリ処理 38
- クエリの最適化 38
- クエリ・プラン 42
 - create table 89

こ

- 更新
 - image データ型 61
 - text データ型 61
 - インデックス 71
- 構文規則、Transact-SQL xiii
- 構文、ネイティブ・データベースの使用。「パススルー・モード」参照
- コピー
 - text および image データ型 61

索引

コンポーネント統合サービス

- 実行 3
 - 設定 3, 103
 - 設定とチューニング 107
 - ユーザ 2
- コンポーネント統合サービスの実行 103
コンポーネント統合サービスの設定 103

さ

- サーバ・クラス 6
- sds 30
 - 「個別のサーバ・クラス名」参照
- サーバ・クラス `direct_connect`
`text` および `image` データ型 62
- サーバ・クラス `sds` 30
- 最適化
- 既存のテーブルの定義 9
 - クイックパス・モード 39, 91
 - リモート・テーブル 52
- 作成
- プロキシ・テーブル 80, 88
 - プロキシ・テーブル上のインデックス 87
- 参照整合性 102

し

- 自動 `identity`
- 自動接続 99
- ジョイン
- リモート・テーブル間 105, 106

す

- スキーマの同期 21

せ

- 制約
- 防止 71
- セキュリティ
- リモート・サーバに関する情報 34
- セキュリティに関する情報 35

接続

- 確認 105
 - 管理 31
 - パーミッション 98
 - 物理、論理 44
 - リモート接続のリスト 70
- 接続管理 31
- 接続の確認 105
- 設定 (サーバ)
- コンポーネント統合サービス 103, 107
- 設定とチューニング 63
- 設定パラメータ
- コンポーネント統合サービス 64, 66

た

- 探索条件
- リモート・テーブル 60

ち

- 中カッコ {}, SQL 文内 xiv
- チューニング
- コンポーネント統合サービス 107

て

- 定義
- インデックス 9
 - テーブル 9, 10
 - リモート・オブジェクト 7
 - リモート・サーバ 7, 103, 105
- ディレクトリ・アクセス 23
- データ型 55
- データ型変換 58
- サーバ・クラス `direct_connect` 87
 - リモート・サーバ 9
- データの整合性
- リモート・テーブル 102
- データベース構文、ネイティブの使用。「パススルー・モード」参照
- データベースにおける Java 53
- テーブル
- 読み込み専用 12
 - リモート、ジョイン 105, 106

テーブル、プロキシ
 定義 9, 10
 トリガ 102

と

統計値
 update statistics 95
 統計のインポート
 プロキシ・テーブル 51
 トランザクション管理 48, 51
 トレース・フラグ 70

は

パーミッション
 パススルー接続 98
 パケット、ネットワーク
 リモート・サーバのサイズ 66
 パススルー接続のパーミッション 98
 パススルー・モード
 connect to コマンド 97, 105
 sp_autoconnect システム・プロシージャ 99
 sp_passthru システム・プロシージャ 100
 sp_remotesql システム・プロシージャ 100
 パターン一致
 text データ型 60
 リモート・テーブル 60
 パフォーマンス
 設定パラメータ 63
 リモート・テーブル 52

ふ

ファイル
 interfaces 104
 sql.ini ファイル 104
 ファイル・アクセス 27
 ファイル・システム・アクセス 23
 物理接続 44
 プロキシ・データベース 18
 プロキシ・テーブル 8
 update statistics 51
 統計のインポート 51
 トリガ 102
 分散トランザクション管理 48

へ

変換、リモート・サーバ・データ型 9
 変数、設定「設定パラメータ」参照

め

メモリ使用率レポート 70

も

モード、trusted/untrusted 35

り

リカバリ
 CIS リカバリの起動時の無効化 71
 リソースの割り付け、sp_configure 63
 リファレンス情報
 CIS に対する Transact-SQL コマンド 77
 リモート・オブジェクト
 定義 7
 リモート・サーバ 29
 interfaces ファイルのエントリ 104
 インタフェース 7
 外部ログインの設定 104
 ジョイン 105, 106
 セキュリティに関する情報 34
 接続の確認 105
 追加 103, 105
 定義 7
 ログイン 7
 リモート・サーバへのインタフェース 7
 リモート接続リスト 70
 リモート・テーブル
 ジョイン 105, 106
 リモート・プロシージャ・コール
 アウトバウンド処理 66
 送信 44
 リモート・ログイン「外部ログイン」参照

索引

れ

- レポート
 - メモリ使用率 70
 - リモート接続 70

ろ

- ローカル・テーブル「プロキシ・テーブル」参照
- ロギング
 - イベント 70
- ログイン
 - 外部 104
 - リモート・サーバ 7
- 論理接続 44

わ

- ワイルドカード文字 60