SYBASE®

Adaptive Server Enterprise 12.5.2
新機能ガイド

Adaptive Server® Enterprise
12.5.2

ドキュメント ID：DC20137-01-1252-01

改訂：2004 年 4 月

マニュアルの注文

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

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

目次

はじめに

対象読者

このマニュアルは、Adaptive Server® バージョン 12.5.2 を使用している Sybase® システム管理者およびデータベース所有者を対象としています。このバージョンの Adaptive Server に含まれる新しい機能について説明します。

Adaptive Server バージョン 12.5.2 は上書き可能なリリースであり、Adaptive Server の現在のバージョンに対して新しいバイナリ・コードをインストールすることができます。詳細については、カバー・レターを参照してください。

このマニュアルの内容

このマニュアルの内容は、次のとおりです。

- 「第 1 章 ステートメント・キャッシュ」では、キャッシュした SQL コードの保存に使用するステートメント・キャッシュについて説明します。
- 「第 2 章 XML サービス」では、XML サービス機能に対する変更点と拡張機能について説明します。
- 「第 3 章 Real Time Messaging Services」では、Real Time Messaging Services オプションについて説明します。
- 「第 4 章 Web Services Consumer」では、Web Services オプションについて説明します。
- 「第 5 章 IPv6 のサポート」では、インターネット・プロトコル・バージョン 6 (IPv6) の実装について説明します。
- 「第 6 章 拡張型全文検索機能の変更」では、拡張型全文検索 (EFTS) の新しい機能について説明します。
- 「第 7 章 セキュリティの変更」では、新しいセキュリティ機能について説明します。
- 「第 8 章 圧縮データベース・ダンプの作成」では、dump コマンドの圧縮パラメータについて説明します。このパラメータでデータベース・ダンプを圧縮できます。
- 「第 9 章 パスワードで保護されたデータベース・ダンプの作成」では、dump コマンドと load database コマンドの password パラメータについて説明します。このパラメータでデータベース・ダンプをパスワードで保護できます。
- 「第 10 章 Veritas 2.1 でフェールオーバを使用できるように Linux Adaptive Server を設定する」では、Veritas Cluster Server 上の Adaptive Server をフェールオーバ用に設定する方法について説明します。

- 「第 11 章 32 ビット版 Linux でのラージ・メモリ・サポート」では、32 ビット版 Red Hat Enterprise Linux 3.0 (RHEL 3) および Red Hat Advanced Server Linux 2.1 用のラージ・メモリ・サポートについて説明します。
- 「第 12 章 グローバル変数、コマンド、ストアド・プロシージャの変更」では、新規および変更されたグローバル変数、関数、コマンド、ストアド・プロシージャについて説明します。

関連マニュアル

Sybase Adaptive Server Enterprise には次のマニュアルが用意されています。

- 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。

  『リリース・ノート』の最新版 (英語版) にはインターネットからアクセスできます。この製品の CD-ROM がリリースされたあとに追加された重要な製品情報やマニュアル情報を確認する場合は、Sybase Technical Library を参照してください。
- 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。
- 『Adaptive Server Enterprise 新機能ガイド』－ Adaptive Server バージョン 12.5.1 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する変更についても説明しています。
- 『ASE Replicator ユーザーズ・ガイド』－ プライマリ・サーバから 1 つ以上のリモートの Adaptive Server に対して基本的な複写を行うための Adaptive Server の ASE Replicator 機能の使用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』－ 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へ接続するための Adaptive Server コンポーネント統合サービス機能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『Adaptive Server Enterprise 設定ガイド』－ Adaptive Server の特定の設定作業を行う方法について説明しています。
- 『EJB Server ユーザーズ・ガイド』－ EJB Server を使用して Adaptive Server で Enterprise JavaBeans を展開、実行する方法について説明しています。
- 『トラブルシューティング&エラー・メッセージ・ガイド』－ 発生頻度の高いエラー・メッセージとシステムの問題について、解決方法を説明しています。
- 『Enhanced Full-Text Search Specialty Data Store ユーザーズ・ガイド』－ Verity で全文検索機能を使用して Adaptive Server Enterprise のデータを検索する方法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、SQL Server® と 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 でジョブをインストールして設定する方法、および作成してスケジュールする方法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、SQL Server と 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。
- 『パフォーマンス＆チューニング・ガイド』－ Adaptive Server で最高のパフォーマンスを実現するためのチューニング方法について説明しています。このマニュアルは以下の 4 冊に分かれています。
  - 『基本』－ Adaptive Server のパフォーマンスに関する問題の理解と調査の基本について説明しています。
  - 『ロック』－ さまざまなロック・スキームを使用して Adaptive Server のパフォーマンスを向上させる方法について説明しています。
  - 『オプティマイザと抽象プラン』－ 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。
  - 『モニタリングと分析』－ 統計を取得および使用してパフォーマンスを監視および最適化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のマニュアルです。
- 『ASE リファレンス・マニュアル』－ 詳細な Transact-SQL® 情報を記載しています。このマニュアルは以下の 4 冊に分かれています。
  - 『ビルディング・ブロック』－ Transact-SQL のデータ型、関数、グローバル変数、式、識別子とワイルドカード、予約語。
  - 『コマンド』－ Transact-SQL のコマンド。
  - 『プロシージャ』－ Transact-SQL のシステム・プロシージャ、カタログ・ストアド・プロシージャ、システム拡張ストアド・プロシージャ、dbcc ストアド・プロシージャ。

    - 『テーブル』－ Transact-SQL のシステム・テーブルと dbcc テーブル。
- 『システム管理ガイド』－ サーバとデータベースを管理するための高度な情報について説明しています。このマニュアルでは、物理的なリソース、セキュリティ、ユーザ・データベース、システム・データベースの管理方法、および文字セットの変換、言語の国際化、ソート順の指定方法についての手順とガイドラインを説明しています。
- 『システム・テーブル・ダイヤグラム』－ 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。印刷版のみが用意されています。
- 『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。このマニュアルでは、データベース管理システムの操作に慣れていない方のために、テキストブック形式で説明しています。また、pubs2 と pubs3 サンプル・データベースについても説明しています。
- 『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。
- 『高可用性システムにおける Sybase フェールオーバの使用』－ Sybase のフェールオーバ機能を使用して、Adaptive Server を高可用性システムのコンパニオン・サーバとして設定する方法について説明しています。
- 『ASE ユーティリティ・ガイド』－ オペレーティング・システム・レベルで実行される isql および bcp などの、Adaptive Server のユーティリティ・プログラムについて説明しています。
- 『Web Services ユーザーズ・ガイド』－ Adaptive Server 用の Web Services 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。
- 『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 の DTM XA インタフェースを使用する方法について説明しています。
- 『Adaptive Server Enterprise における XML Services 』－ データベースに XML 機能を導入する、Sybase ネイティブの XML プロセッサと Sybase Java ベースの XML のサポートについて、また XML サービス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。

その他の情報ソース

Sybase Getting Started CD、Sybase Technical Library CD、Technical Library 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・ガイドが収録されています。また、その他のマニュアルや、Technical Library CD には含まれない更新情報が収録されることもあります。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)。
- Technical Library CD には製品マニュアルが入っており、この CD は製品のソフトウェアに同梱されています。DynaText リーダー (Technical Library CD に収録) を使用すると、この製品に関する技術情報に簡単にアクセスできます。

  Technical Library のインストールと起動の方法については、マニュアル・パッケージに含まれている『Technical Library Installation Guide』を参照してください。
- Technical Library Product Manuals Web サイトは、Technical Library CD の HTML バージョンで、標準の Web ブラウザを使っ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。

Web 上の Sybase 製品の動作確認情報

Sybase Web サイトの技術的な資料は頻繁に更新されます。

❖ **製品認定の最新情報にアクセスする**

1. Web ブラウザで Technical Documents を指定します。(http://www.sybase.com/support/techdocs/)
2. 左側のナビゲーション・バーから [Products] を選択します。
3. 製品リストから製品名を選択し、[Go] をクリックします。
4. [Certification Report] フィルタを選択し、時間枠を指定して [Go] をクリックします。
5. [Certification Report] のタイトルをクリックして、レポートを表示します。

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

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

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

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

Sybase EBF とソフトウェア・メンテナンス

❖ **EBF とソフトウェア・メンテナンスの最新情報にアクセスする**

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

2 [EBFs/Maintenance] を選択します。すでに Web アカウントをお持ちの場合はユーザ名とパスワードを要求されますので、各情報を入力します。Web アカウントをお持ちでない場合は、新しいアカウントを作成します。サービスは無料です。

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。

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。

表記の規則

このマニュアルの本文では、次のようにファイル名とディレクトリ名を斜体で表記します。

- Windows NT の場合：*%SYBASE%¥bin*
- UNIX プラットフォームの場合：*$SYBASE*

注意 UNIX の場合は *$SYBASE* を、また Windows NT の場合は *%SYBASE%* を、使用している Sybase インストール・ドライブおよびディレクトリに置き換えてください。

表 1 は、このマニュアルで使用されている表記 (フォントと構文) の規則をまとめたものです。

表 1: このマニュアルのフォントと構文の規則

要素	例
コマンド名、コマンドのオプション名、データベース名、データ型、ユーティリティ名、ユーティリティのフラグ、キーワードは、**太字の** Helvetica で表記する。	dsedit
変数 (ユーザが入力する値を表す語) は斜体で表記する。	`select` *column_name* `from` *table_name* `where` *search_conditions*
カッコはコマンドの一部として入力する。	`compute row_aggregate (column_name)`
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 (「カンマ」参照)。	`{cheese, sauce}` **注意** コマンドには中カッコは入力しない。
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。	`[anchovies, pineapple, bell_peppers]` **注意** コマンドには角カッコは入力しない。
縦線は、複数のオプションのうち 1 つだけを選択できることを意味する。	`{cash | check | credit}` **注意** コマンドには中カッコは入力しない。
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。	`[extra_cheese, avocados, sour_cream]` **注意** コマンドには角カッコは入力しない。

省略記号 (...) は、直前の要素を必要な回数だけ繰り返し指定できることを意味する。	`buy` *thing* `= price [cash | check | credit]`
	`[,` *thing* `= price [cash | check | credit] ]...`
	• この例では、製品 (*thing*) を少なくとも 1 つ購入 (buy) し、価格 (price) を指定する必要がある。
	• 支払方法を角カッコの中から 1 つ選択できる。
	• さらに、他の製品を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) を指定する。

構文では、すべてのオプションを含むユーティリティ構文を通常のフォントで表記する。ただし、フラグとオプション (-v) は通常のフォントで、またユーザが指定する値 (*username*) は斜体で表記する。	charset [-P*password*] [-S*server*] [-I*interface*] *sort_order* | *charset*

コンピュータからの出力例は Courier フォントで表記する。	`pub_id pub_name city state`
	`------ --------- ------- -----`
	`0736 New Age Books Boston MA`
	`0877 Binnet & Hardley Washington DC`
	`(2 rows affected)`

不明な点があるときは

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

第 1 章

ステートメント・キャッシュ

この章では、キャッシュした SQL 文の保存に使用するステートメント・キャッシュについて説明します。Adaptive Server は、受信した SQL とキャッシュしている SQL を比較し、一致する場合はすでに保存している SQL プランを実行します。これにより、アプリケーションは、同じ文を繰り返し実行するたびにクエリをコンパイルする必要がなくなります。

ステートメント・キャッシュの設定

ステートメント・キャッシュを使用すると、Adaptive Server はアドホック SQL 文のテキストを保管できます。Adaptive Server は、新たに受信したアドホック SQL 文とキャッシュしている SQL 文を比較し、一致する場合は最初の実行時にキャッシュしたプランを使用します。この方法により、Adaptive Server では、すでにプランがある SQL 文を再コンパイルする必要がなくなります。

ステートメント・キャッシュはサーバワイドなリソースであり、プロシージャ・キャッシュ・メモリ・プールのメモリを割り付け、消費します。ステートメント・キャッシュのサイズは、statement cache size 設定パラメータを使用して動的に設定します。構文は次のとおりです。

```
sp_configure "statement cache size", size_of_cache
```

size_of_cache は 2K ページ単位のサイズです。たとえば、ステートメント・キャッシュを 5000 × 2K ページに設定するには、次のように入力します。

```
sp_configure "statement cache size", 5000
```

詳細については、「statement cache size 設定パラメータ」(8 ページ) を参照してください。

ステートメント・キャッシュのメモリを設定するときは次の点に注意してください。

- プロシージャ・キャッシュ・メモリ・プールに割り付けられるメモリ量は、statement cache size 設定パラメータと procedure cache size 設定パラメータの合計です。ステートメント・キャッシュ・メモリはプロシージャ・キャッシュ・メモリ・プールから使用されます。前の例では、プロシージャ・キャッシュ・メモリ・プールのサイズが 5000 × 2K ページ分増えます。
- statement cache size を使用すると、キャッシュされる SQL テキストとプランによってプロシージャ・キャッシュ・メモリの量が制限されます。つまり、Adaptive Server は、statement cache size 設定パラメータで設定した量を超えるメモリをステートメント・キャッシュに使用することはできません。
- statement cache size 設定パラメータで割り付けられたメモリも含め、プロシージャ・キャッシュ・メモリすべてはストアド・プロシージャに使用でき、キャッシュされた文は LRU ベースで置換されます。
- max memory 設定パラメータは、ステートメント・キャッシュの設定と同じ量だけ増やしてください。つまり、ステートメント・キャッシュのサイズを 100 × 2K ページに設定した場合は、これと同じ量だけ max memory を増やします。
- statement cache size 設定パラメータを使用してステートメント・キャッシュを設定している場合、set statement cache を使用してセッション・レベルでステートメント・キャッシュの無効と有効を切り替えることができます。ステートメント・キャッシュは、サーバ・レベルで設定されている場合、セッション・レベルではデフォルトで有効になります。
- キャッシュされた文はそれぞれ 1 つのオブジェクト記述子を使用するため、number of open databases 設定パラメータを使用してオブジェクト記述子の数を増やす必要があります。キャッシュできる SQL 文の数を推定するには、「ステートメント・キャッシュのサイズ設定」(4 ページ) を参照してください。

アドホック・クエリの処理

Adaptive Server によるステートメント・キャッシュを使用したアドホック SQL 文の処理は、次の手順で行われます。

1 Adaptive Server は文を解析します。

キャッシュすべき文である場合 (「キャッシュする条件」(4 ページ) を参照)、Adaptive Server は文のハッシュ値を計算します。このハッシュ値を使用して、ステートメント・キャッシュ内で一致する文を検索します (「文の一致基準」(3 ページ) を参照)。

- ステートメント・キャッシュ内で一致する文が見つかった場合は、手順 4 に進みます。
- 一致する文が見つからない場合は、手順 2 に進みます。

2　Adaptive Server は、この SQL 文のテキストをキャッシュします。

3　Adaptive Server は、SQL 文をライトウェイト・ストアド・プロシージャでラップし、ローカル変数が含まれる場合にはそれをプロシージャ・パラメータに変更します。この時点では、ライトウェイト・プロシージャの内部表現はプランにコンパイルされません。

4　Adaptive Server は、SQL 文を、対応するライトウェイト・プロシージャの execute 文に変換します。

- キャッシュにプランがない場合は、プロシージャをコンパイルしてプランをキャッシュします。ローカル変数に割り当てられたランタイム値を使用してプランをコンパイルします。
- プランが存在していても無効である場合は、手順 3 に戻り、キャッシュされている SQL 文のテキストを使用します。

5　Adaptive Server はプロシージャを実行します。

文の一致基準

Adaptive Server は、アドホック SQL 文とキャッシュした文の照合に、SQL テキストや、ログイン (特に両方のユーザが sa_role を持っている場合)、ユーザ ID、データベース ID、セッション状態の設定を使用します。関連するセッション状態は、次に示す set コマンド・パラメータの設定で構成されます。

- forceplan
- jtc
- parallel_degree
- prefetch
- quoted_identifier
- sort_merge
- table count
- transaction isolation level
- chained (トランザクション・モード)

これらのパラメータの設定内容により、キャッシュした文に対して Adaptive Server が生成するプランの動作が決まります。set コマンドとパラメータの詳細については、『ASE リファレンス・マニュアル』を参照してください。

注意　ステートメント・キャッシュを有効にする場合は、set chained on/off をバッチ内に設定してください。

キャッシュする条件

- 現在、Adaptive Server では、少なくとも 1 つのテーブル参照を含む select 文、update 文、delete 文、insert select 文がキャッシュされます。
- abstract plan dump パラメータまたは abstract plan load パラメータが有効になっている場合、文はキャッシュされません。
- select into 文、カーソル文、動的文、単純な insert 文 (insert select を除く) や、ストアド・プロシージャ、ビュー、トリガ内の文はキャッシュされません。また、テンポラリ・テーブルを参照する文や、BLOB データ型として送信される言語パラメータを含む文もキャッシュされません。許容サイズを超える文もキャッシュされません。さらに、if exists または if not exists という条件句に含まれる select 文もキャッシュされません。

ステートメント・キャッシュのサイズ設定

キャッシュされる文は、SQL テキストの長さによって異なりますが、それぞれ約 1K のステートメント・キャッシュ用メモリが必要です。キャッシュされるプランは、それぞれ少なくとも 2K のプロシージャ・キャッシュ・メモリが必要です。必要なステートメント・キャッシュ・メモリを推定するには、キャッシュする各文ごとの次の値を合計します。

- SQL 文の長さ (バイト単位で 256 の倍数に切り上げる)。
- 約 100 バイトのオーバヘッド。
- プロシージャ・キャッシュ内のプランのサイズ。このサイズは、キャッシュされる文のみを含むストアド・プロシージャ・プランのサイズと同じです。キャッシュされた 1 つの文を複数のユーザが同時に使用する場合、プランが複製されることがあります。

ステートメント・キャッシュのモニタリング

sp_sysmon は、文のキャッシュとストアド・プロシージャの実行についてレポートします。ステートメント・キャッシュは次のカウンタを使用してモニタされます。

- Statements Found in Cache － クエリ・プランが再使用された回数。キャッシュのヒット回数が少ない場合、ステートメント・キャッシュが小さすぎることを示している場合があります。
- Statements Not Found － 繰り返された SQL 文が存在しないことを示します。statements found in cache と statements not found の合計が、処理対象になる発行された SQL 文の総数です。
- Statements Cached － キャッシュ内の SQL 文の数。通常、これは Statements Not Found と同数です。statements cached の値の方が小さい場合は、ステートメント・キャッシュがアクティブな文で一杯になっています。

- Statements Dropped − キャッシュされずに削除された文の数。この値が大きい場合、プロシージャ・キャッシュ・メモリ量が不十分であるか、ステートメント・キャッシュ・サイズの設定が小さすぎることがあります。
- Statements Restored − SQL テキストから再生成されたクエリ・プランの数。値が大きい場合は、プロシージャ・キャッシュ・サイズが不十分です。
- Statements Not Cached − ステートメント・キャッシュが有効になっていれば Adaptive Server がキャッシュするはずであった文の数。ただし、Statements Not Cached では、何種類の文がキャッシュされなかったかはわかりません。

次に、sp_sysmon のサンプル出力を示します。

```
Procedure Cache Management        per sec      per xact       count  % of total
--------------------------  ------------  ------------  ----------  ----------
  Procedure Requests                 6.6           3.7          33        n/a
  Procedure Reads from Disk          1.0           0.6           5      15.2%
  Procedure Writes to Disk           0.4           0.2           2       6.1%
  Procedure Removals                 2.6           1.4          13        n/a
  Procedure Recompilations           0.8           0.4           4        n/a

  Recompilations Requests:
   Execution Phase                   0.6           0.3           3      75.0%
   Compilation Phase                 0.2           0.1           1      25.0%
   Execute Cursor Execution          0.0           0.0           0       0.0%
   Redefinition Phase                0.0           0.0           0       0.0%

  Recompilations Reasons:
   Table Missing                     0.6           0.3           3        n/a
   Temporary Table Missing           0.2           0.1           1        n/a
   Schema Change                     0.0           0.0           0        n/a
   Index Change                      0.0           0.0           0        n/a
   Isolation Level Change            0.2           0.1           1        n/a
   Permissions Change                0.0           0.0           0        n/a
   Cursor Permissions Change         0.0           0.0           0        n/a

  SQLStatement Cache:
   Statements Cached                 0.0           0.0           0        n/a
   Statements Found in Cache         0.7           0.0           2        n/a
   Statements Not Found              0.0           0.0           0        n/a
   Statements Dropped                0.0           0.0           0        n/a
   Statements Recompiled             0.3           0.0           1        n/a
   Statements Not Cached             1.3           0.0           4        n/a
```

ステートメント・キャッシュの消去

dbcc purgesqlcache を実行すると、ステートメント・キャッシュからすべての SQL 文が削除されます。現在実行している文は削除されません。

構文情報については、「dbcc の変更」(11 ページ) を参照してください。

dbcc purgesqlcache を実行するには sa_role が必要です。

dbcc purgesqlcache を実行すると次のメッセージが出力されます。

```
dbcc purgesqlcache
DBCC の実行が完了しました。DBCC がエラー・メッセージを表示した場合、システム管理者 (SA) の権限を持つユーザに連絡してください。
```

文の要約の出力

dbcc prsqlcache を実行すると、ステートメント・キャッシュ内の文の要約が出力されます。*oid* オプションを指定すると、出力する文のオブジェクト ID を指定できます。また、*printopt* オプションでは、トレースの説明を出力するか (0 を指定) または showplan オプションを出力するか (1 を指定) を指定できます。*oid* または *printopt* に値を指定しないで dbcc prsqlcache を実行すると、ステートメント・キャッシュの内容全体が表示されます。

構文情報については、「dbcc の変更」(11 ページ) を参照してください。

dbcc prsqlcache を実行するには sa_role が必要です。

次の例は、キャッシュ内のすべての文の情報を出力します。

```
dbcc prsqlcache
Start of SSQL Hash Table at 0xfc67d830
Memory configured: 1000 2k pages            Memory used: 18 2k pages
Bucket# 625 address 0xfc67ebb8

SSQL_DESC 0xfc67f9c0
ssql_name *ss1248998166_0290284638ss*
ssql_hashkey 0x114d645e ssql_id 1248998166
ssql_suid 1             ssql_uid 1      ssql_dbid 1
ssql_status 0x28        ssql_parallel_deg 1
ssql_tab_count 0        ssql_isolate 1  ssql_tranmode 0
ssql_keep 0             ssql_usecnt 1   ssql_pgcount 8
SQL TEXT: select * from sysobjects where name like "sp%"

Bucket# 852 address 0xfc67f2d0
SSQL_DESC 0xfc67f840
ssql_name *ss1232998109_1393445479ss*
ssql_hashkey 0x530e4a67 ssql_id 1232998109
ssql_suid 1             ssql_uid 1      ssql_dbid 1
ssql_status 0x28        ssql_parallel_deg 1
ssql_tab_count 0        ssql_isolate 1  ssql_tranmode 0
ssql_keep 0             ssql_usecnt 1   ssql_pgcount 3
SQL TEXT: select  name from systypes where allownulls = 0
```

```
End of SSQL Hash Table
```

DBCC の実行が完了しました。DBCC がエラー・メッセージを表示した場合、システム管理者 (SA) の権限を持つユーザに連絡してください。

また、次の例のように、特定のオブジェクト ID の情報を得ることができます。

```
dbcc prsqlcache (1232998109, 0)
SSQL_DESC 0xfc67f840
ssql_name *ss1232998109_1393445479ss*
ssql_hashkey 0x530e4a67 ssql_id 1232998109
ssql_suid 1             ssql_uid 1      ssql_dbid 1
ssql_status 0x28        ssql_parallel_deg 1
ssql_tab_count 0        ssql_isolate 1  ssql_tranmode 0
ssql_keep 0             ssql_usecnt 1   ssql_pgcount 3
SQL TEXT: select  name from systypes where allownulls = 0
```

DBCC の実行が完了しました。DBCC がエラー・メッセージを表示した場合、システム管理者 (SA) の権限を持つユーザに連絡してください。

次の例は、showplan の出力のために printopt パラメータに 1 を指定しています。

```
dbcc prsqlcache (1232998109, 1)
SSQL_DESC 0xfc67f840
ssql_name *ss1232998109_1393445479ss*
ssql_hashkey 0x530e4a67 ssql_id 1232998109
ssql_suid 1             ssql_uid 1       ssql_dbid 1
ssql_status 0x28        ssql_parallel_deg 1
ssql_tab_count 0        ssql_isolate 1  ssql_tranmode 0
ssql_keep 0             ssql_usecnt 1   ssql_pgcount 3
SQL TEXT: select  name from systypes where allownulls = 0

文 1 (1 行目 ) のクエリ・プラン。

     STEP 1
     クエリのタイプは SELECT です。
         FROM TABLE
            systypes
     ネストした繰り返し
     テーブル・スキャンです。
     前方スキャン

     テーブルの最初に位置付けます。
     データ・ページに対して I/O サイズ 2 キロバイトを使用しています。
     データ・ページに対する LRU でのバッファ置換方式
     DBCC の実行が完了しました。DBCC がエラー・メッセージを表示した場合、
     システム管理者 (SA) の権限を持つ
      ユーザに連絡してください。
```

statement cache size 設定パラメータ

statement cache size

要約	
デフォルト値	0
有効な値	キャッシュのサイズ (2K ページ単位)
ステータス	動的
表示レベル	包括
必要な役割	システム管理者

statement cache size パラメータを使用すると、プロシージャ・キャッシュ・メモリのサーバ割り付けが増え、プロシージャ・キャッシュ・プールのうち文のキャッシュに使用されるメモリ量が制限されます。ステートメント・キャッシュ機能はサーバワイドに有効になります。

```
statement cache size size_of_cache
```

注意 ステートメント・キャッシュを有効にする場合は、set chained on/off をバッチ内に設定してください。

キャッシュされた文はライトウェイト・ストアド・プロシージャに変換されるため、文のキャッシュではオープンしているオブジェクト記述子がさらに必要になります。

キャッシュのためのメモリの設定

Adaptive Server の設定において、メモリは最も重要な設定オプションです。メモリは、さまざまな設定パラメータ、プロシージャ・キャッシュ、ステートメント・キャッシュ、データ・キャッシュによって消費されます。システム・パフォーマンスを高めるには、さまざまな設定パラメータとキャッシュの値を適切に設定することが重要です。

システム起動時に割り付けられるメモリの合計は、Adaptive Server のすべての設定に必要なメモリの合計です。この値は、読み込み専用の設定パラメータ total logical memory から取得され、Adaptive Server によって計算されます。設定パラメータ max memory には、total logical memory 以上の値を指定する必要があります。max memory は、Adaptive Server の使用に対応できるメモリの量を示しています。

デフォルトでは、total logical memory の値に基づいてサーバの起動時にメモリが割り付けられます。ただし、設定パラメータ allocate max shared memory が設定されている場合は、max memory の値に基づいてメモリが割り付けられます。設定パラメータ allocate max shared memory を使用すると、システム管理者は Adaptive Server で使用できる最大メモリをサーバの起動時に割り付けることができます。

メモリ設定の重要な点は次のとおりです。

- システム管理者は、Adaptive Server に使用できる共有メモリのサイズを決定し、max memory をこの値に設定する必要がある。
- 設定パラメータ allocate max shared memory を起動時と実行時にオンにすると、最小限の共有メモリ・セグメントを使用してすべての共有メモリを max memory まで割り付けることができる。多数の共有メモリ・セグメントを使用すると、特定のプラットフォームでパフォーマンスが低下するという欠点がある。共有メモリ・セグメントの最適な数については、オペレーティング・システムのマニュアルを参照。割り付けられた共有メモリ・セグメントは、サーバが再起動されるまで解放できない。
- max memory と total logical memory の差分は、プロシージャ・キャッシュ、ステートメント・キャッシュ、データ・キャッシュ、または他の設定パラメータに使用できる追加メモリになる。

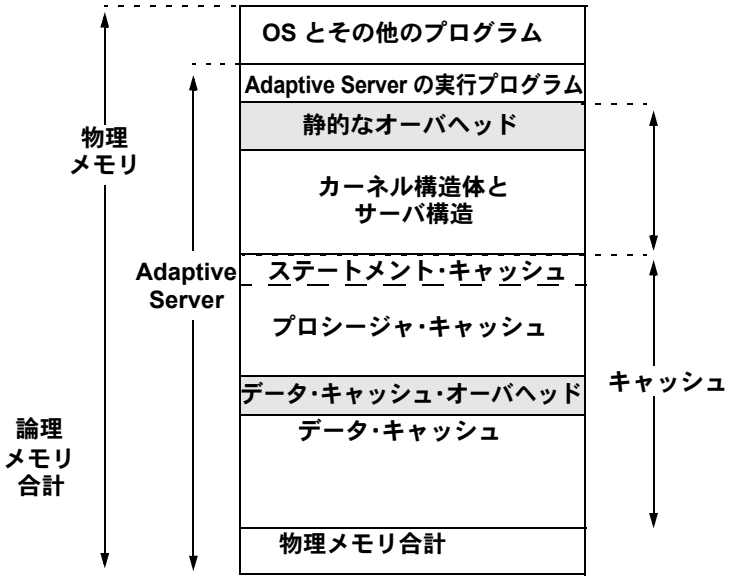
  起動時に Adaptive Server によって割り付けられるメモリの量は、total logical memory または max memory によって決まる。この値が大きすぎると、次の問題が発生する可能性がある。

  - マシンの物理リソースが不十分な場合は、Adaptive Server が起動しないことがある。
  - Adaptive Server が起動しても、オペレーティング・システム・ページのフォールト・レートが著しく上昇し、オペレーティング・システムを再設定する必要が生じることがある。
  - よりワイドな文字リテラルの処理。この処理では、Adaptive Server で文字列ユーザ・データにメモリを割り付ける必要がある。Adaptive Server では、最大可能サイズのバッファを静的に割り付けるのではなく、メモリを動的に割り付ける。つまり、必要に応じてローカル・バッファにメモリを割り付け、大きなバッファが不要な場合でも、常に最大サイズのメモリをこれらのバッファに割り付ける。このようなメモリ管理要求によって、Adaptive Server がワイド文字データを処理するときのパフォーマンスの低下を最小限に抑えることができる。

- Adaptive Server で 1 つのテーブルから 1000 以上のカラムを処理したり、ストアド・プロシージャへの 10000 以上の引数を処理したりする必要がある場合、こうしたオブジェクトのさまざまな内部データ構造に合わせて Adaptive Server を設定してメモリを割り付ける必要がある。繰り返し実行される小さいタスクの数が増えると、大量の項目を処理するクエリのパフォーマンスが低下することがある。このパフォーマンスは、カラムとストアド・プロシージャ引数の数を増やすと向上する。
- メモリを動的に割り付けると、サーバのパフォーマンスがわずかに低下する。
- Adaptive Server でより大きい論理ページ・サイズを使用する場合は、すべてのディスク I/O はより大きい論理ページ・サイズに基づいて行われる。たとえば、Adaptive Server で 8K 論理ページ・サイズを使用する場合、8K ブロック単位でディスクからデータが取得される。この結果、I/O スループットが増加する。ただし、スループットの量は最終的にはコントローラの I/O 帯域幅によって制限される。

他のすべての必要なメモリ領域が満たされたあとに残ったメモリは、プロシージャ・キャッシュ、ステートメント・キャッシュ、データ・キャッシュに使用できます。図 1-1 は、メモリの配分を示します。

図 1-1: Adaptive Server のメモリの使い方



コマンドの変更

この項では、ステートメント・キャッシュの設定とモニタリングに対応するためのコマンドの変更点について説明します。

dbcc の変更

dbcc に prsqlcache パラメータと purgesqlcache パラメータが追加されています。prsqlcache パラメータを使用すると、キャッシュされている SQL 文の要約が出力されます。purgesqlcache パラメータを使用すると、現在使用中の文を除くステートメント・キャッシュのすべてのエントリが消去されます。

dbcc の構文の一部は次のとおりです。

```
dbcc prsqlcache[oid, printopt]
dbcc purgesqlcache
```

各パラメータの意味は、次のとおりです。

- prsqlcache － キャッシュされている SQL 文の要約を出力します。
- *oid* － 要約を出力するエントリのオブジェクト ID。その他の出力内容は、*printopt* 変数の値によって制御されます。*oid* の値を 0 に設定すると、キャッシュされているすべての SQL 文の要約が出力され、*printopt* の値は無視されます。
- *printopt* － *oid* が有効なオブジェクト ID に設定されているときに prsqlcache の出力を制御します。*printopt* 変数には 0 または 1 を設定できます。*printopt* に 1 を設定すると showplan の出力が表示されます。
- purgesqlcache － 現在使用中の文を除くすべての SQL 文をステートメント・キャッシュから削除します。

dbcc prsqlcache の例については 「ステートメント・キャッシュの消去」(6 ページ)、dbcc purgesqlcache の例については 「文の要約の出力」(6 ページ) を参照してください。

set の変更

ステートメント・キャッシュに対応するため、set statement cache パラメータが追加されています。

set の構文の一部は次のとおりです。

```
set statement_cache [on | off]
```

パラメータ

statement_cache － セッション・レベルでステートメント・キャッシュを有効または無効にします。SA 権限を使用して statement cache size を 0 以外の値に設定すると、そのセッションでは、セッション・レベルでステートメント・キャッシュの有効と無効を切り替えることができます。サーバ・レベルでステートメント・キャッシュが設定されている場合は、セッション・レベルではデフォルトで有効になります。

第 2 章

XML サービス

この章では、XML サービス機能に対する変更点と拡張機能について説明します。

概要

XML サービスでは、主に次の 3 つの領域が強化されています。

- xmlextract 組み込み関数と xmltest 述部に対する XML 問い合わせ言語の拡張
  - XPath 文字列関数のサポート
    - toupper(...)
    - tolower(...)
    - normalize-space(...)
    - concat(...)
  - カッコで囲んだ XPath 式のサポート
  - XPath 規格への収束
    - 規格以外のサブスクリプト式の削除
    - “//” 演算子の解釈

- select 文の for xml 句でサポートされるデータ型の拡張
  - text、image、binary、varbinary
  - java.lang.String
- xmlextract、xmlparse、xmlrepresentation でサポートされるデータ型の拡張
  - xmlextract － 結果の XML フラグメントを java.lang.String として返すことができます。
  - xmlparse － 解析済みドキュメントを varbinary データ型として返します。
  - xmlrepresentation － varbinary 型 XML ドキュメントを受け付けることができます。
- Java-XML サンプル・コードの拡張
  - SQL データを階層構造の XML ドキュメントにマップする Java ベースの ForXmlTree 関数
  - 階層構造の XML ドキュメントから SQL テーブルを抽出する Java ベースの OpenXml 関数
  - SQL クエリで複数の結果セットを返す Java ベースの SQLX マッピング関数

XML 問い合わせ言語の機能拡張

この項では、XML 問い合わせ言語の機能拡張である XPath の演算子と関数について説明します。「XPath の一般的な構文」(14 ページ) と「XPath 文字列関数：一般的なガイドライン」(15 ページ) では、サポートされている一般的な BNF の構文と XML 関数の使用方法の一般的なガイドラインについて説明しています。

XPath の一般的な構文

XML サービスでは、次の XPath 構文をサポートしています。機能拡張部分は太字で示しています。Adaptive Server バージョン 12.5.2 では、*primary_expr* の構文が拡張され、*function_call* の構文が追加されています。

```
xpath::= or_expr
or_expr::= and_expr | and_expr TOKEN_OR or_expr
and_expr::= union_expr | union_expr TOKEN_AND and_expr
union_expr::= intersect_expr
    | intersect_expr TOKEN_UNION union_expr
intersect_expr::= comparison_expr
    | comparison_expr TOKEN_INTERSECT intersect_expr
comparison_expr::= range_exp
```

```
    | range_expr general_comp comparisonRightHandSide
general_comp::= TOKEN_EQUAL  | TOKEN_NOTEQUAL
    | TOKEN_LESSTHAN  | TOKEN_LESSTHANEQUAL
    | TOKEN_GREATERTHAN  | TOKEN_GREATERTHANEQUAL
range_expr::= unary_expr  | unary_expr TOKEN_TO unary_expr
unary_expr::= TOKEN_MINUS path_expr
    | TOKEN_PLUS path_expr
    | path_expr
comparisonRightHandSide::=  literal
path_expr::= relativepath_expr  | TOKEN_SLASH
    | TOKEN_SLASH   relativepath_expr
    | TOKEN_DOUBLESLASH   relativepath_expr
relativepath_expr::= step_expr
    | step_expr TOKEN_SLASH relativepath_expr
    | step_expr TOKEN_DOUBLESLASH relativepath_expr
step_expr::= forward_step predicates
    | primary_expr predicates
    | predicates
primary_expr::= literal | function_call | (xpath)
function_call::=
    tolower([xpath])
    | toupper([xpath])
    | normalize-space([xpath])
    | concat([xpath [,xpath]...])
forward_step::= abbreviated_forward_step
abbreviated_forward_step::= name_test
    | TOKEN_ATRATE name_test
    | TOKEN_PERIOD
name_test::= q_name  | wild_card | text test
text_test ::= TOKEN_TEXT TOKEN_LPAREN TOKEN_RPAREN
literal::= numeric_literal  | string_literal
wild_card::= TOKEN_ASTERISK
q_name::= TOKEN_ID
string_literal::= TOKEN_STRING
numeric_literal::= TOKEN_INT  | TOKEN_FLOATVAL|
    | TOKEN_MINUS TOKEN_INT
    | TOKEN_MINUSTOKEN_FLOATVAL
predicates::=
    | TOKEN_LSQUARE expr TOKEN_RSQUARE predicates
    | TOKEN_LSQUARE expr TOKEN_RSQUARE
```

XPath 文字列関数：一般的なガイドライン

Adaptive Server バージョン 12.5.2 では、XML 問い合わせ言語の xmlextract と xmltest の機能が拡張され、次の XPath 文字列関数をサポートするようになりました。

- toupper
- tolower
- normalize-space
- concat

この項では、XPath 式で関数を使用する際の一般的なガイドラインを示します。このガイドラインは前述のすべての関数に適用されます。以下の例はすべて tolower を使用します。この関数は 1 つの引数を小文字で返します。

ステップ式を使用するところでは、どこにでも関数呼び出しを使用できます。

例 1

XPath クエリの最上位として使用される関数は、最上位関数呼び出しと呼ばれます。次のクエリは、最上位関数呼び出しとして使用される tolower を示します。

```
select xmlextract
('tolower(//book[title]="Seven Years in Trenton"]//first-name)', text_doc)
from sample_docs where name_doc='bookstore'
----------------------------------------
joe
```

最上位関数呼び出しのパラメータには絶対パス式を指定します。つまり、パラメータはスラッシュ (/) またはスラッシュ 2 つ (//) で始めます。

例 2

関数呼び出しのパラメータには、述部を含む複雑な XPath 式を指定できます。また、ネストした関数呼び出しにすることもできます。

```
select xmlextract
('//book[normalize-space(tolower(title))="seven years in trenton"]/author',
text_doc)
from sample_docs where name_doc='bookstore'

----------------------------------------
<author>
     <first-name>Joe</first-name>
     <last-name>Bob</last-name>
     <award>Trenton Literary Review
     Honorable Mention</award>
</author>
```

例 3

関数を相対ステップとして使用できます。相対ステップは相対関数呼び出しとも呼ばれます。次のクエリは、相対関数呼び出しとして使用される tolower を示します。

```
select xmlextract
( '//book[title="Seven Years in Trenton"]//tolower(first-name)', text_doc)
from sample_docs where name_doc='bookstore'

-------------------------------------

joe
```

この例は、相対関数のパラメータには相対パス式を指定する必要があることを示します。つまり、スラッシュ (/) またはダブル・スラッシュ (//) で始めることはできません。

例 4

最上位関数および相対関数は、どちらもパラメータにリテラルを使用できます。次に例を示します。

```
select xmlextract( 'tolower("aBcD")' ,text_doc),
    xmlextract( '/bookstore/book/tolower("aBcD")', text_doc)
from sample_docs where name_doc='bookstore'

 --------  ----------

 abcd      abcd
```

例 5

文字列関数は、そのパラメータのテキストに作用します。つまり、text() が暗黙的に適用されます。たとえば、次のクエリは *first-name* 要素を XML フラグメントとして返します。

```
select xmlextract
( '//book[title="Seven Years in Trenton"]//firstname', text_doc)
from sample_docs where name_doc='bookstore'

------------------------------

<first-name>Joe</first-name>
```

次のクエリは、*first-name* の XML フラグメントのテキストを返します。

```
select xmlextract
( '//book[title="Seven Years in Trenton"]//first-name/text()', text_doc)
from sample_docs where name_doc='bookstore'

------------------------------

Joe
```

次のクエリは *first-name* 要素に tolower を適用します。この関数は要素のテキストに暗黙的に作用します。

```
select xmlextract
('//book[title="Seven Years in Trenton"] //tolower(first-name)', text_doc)
from sample_docs where name_doc='bookstore'

---------------------------------------------

joe
```

次の例は、明示的にパラメータとして XML 要素のテキストを渡します。これは前の例と同じ結果になります。

```
select xmlextract
( '//book[title="Seven Years in Trenton"]//tolower(first-name/text())',
text_doc)
from sample_docs where name_doc='bookstore'

---------------------------------------

joe
```

例 6

パスの 1 つのステップとして相対関数呼び出しを適用します。そのパスが評価されると XML ノードのシーケンスが生成され、各ノードの相対関数呼び出しが実行されます。その結果、関数呼び出し結果のシーケンスが生成されます。たとえば、次のクエリは *first_name* ノードのシーケンスを生成します。

```
select xmlextract( '/bookstore/book/author/first-name', text_doc)
from sample_docs where name_doc='bookstore'
---------------------------------
<first-name>Joe</first-name><first-name>Mary</first-name>
<first-name>Toni</first-name>
```

次のクエリは、前のクエリの最終ステップを toupper 呼び出しに置き換えて、両方の関数呼び出し結果のシーケンスを生成します。

```
select xmlextract('/bookstore/book/author/toupper(first-name)', text_doc)
from sample_docs where name_doc='bookstore'
---------------------------------
JOEMARYTONI
```

ここで、concat を使用して関数結果のシーケンスを区切ることができます。「concat」(20 ページ) の例を参照してください。

例 7

tolower、toupper、normalize-space のパラメータはどれも 1 つです。相対関数呼び出しにこれらの関数を指定するときにパラメータを省略すると、現在のノードが暗黙のパラメータになります。たとえば、次の例は明示的にパラメータを指定した tolower を示します。

```
select xmlextract
('//book[title="Seven Years in Trenton"]//tolower(first-name)', text_doc)
from sample_docs where name_doc='bookstore'
-----------------------------------------------------
joe
```

次の例は、同じクエリですが、パラメータを暗黙的に指定しています。

```
select xmlextract
('//book[title="Seven Years in Trenton"]//first-name/tolower()', text_doc)
from sample_docs where name_doc='bookstore'
-------------------------------------------
joe
```

相対関数呼び出しが複数のノードに適用される場合も、呼び出しにパラメータを暗黙的に指定できます。次に例を示します。

```
select xmlextract('//book//first-name/tolower()', text_doc)
from sample_docs where name_doc='bookstore'
-------------------------------------------
joemarymarytoni
```

関数

この項では、XML サービスを強化する個々の関数について説明します。

tolower と toupper

説明　tolower は小文字で、toupper は大文字で引数を返します。

構文　tolower(*string-parameter*)
toupper(*string-parameter*)

例　この例では、toupper を使用して引数の値を大文字で返します。

```
select xmlextract
('//book[title="Seven Years in Trenton"]//toupper(first-name)', text_doc)
from sample_docs where name_doc='bookstore'
---------------------------------------
JOE
```

normalize-space

説明　引数の値を返すとき、次の 2 つの変更を行います。

- 先頭または末尾にあるスペース文字を削除する。
- 先頭文字以外の 2 つ以上連続するスペース文字をすべて 1 つのスペース文字に置き換える。

構文　normalize-space(*string-parameter*)

例

この例は、先頭と末尾にスペース文字があり、改行文字とタブ文字が埋め込まれているパラメータに normalize-space を適用します。

```
select xmlextract
('normalize-space(" Normalize  space   example. ")', text_doc)
from sample_docs where name_doc='bookstore'
------------------------
Normalize space example.
```

スペース文字や大文字小文字の使用状況が不明な値をテストするとき、XPath 述部で normalize-space と tolower または toupper を使用すると便利です。次の述部は、*title* 要素での大文字小文字やスペース文字の使用状況には影響されません。

```
select xmlextract
('//magazine[normalize-space(tolower(title)="tracking trenton")]//price',
text_doc)
from sample_docs where name_doc='bookstore'
-------------------------
<price>55</price>
```

concat

説明

concat は引数の値を連結した文字列を返します。0 個以上のパラメータを取ります。

構文

concat(*string-parameter* [,*string-parameter*]...)

例

concat は xmlextract の 1 回の呼び出しで複数の要素を返すことができます。たとえば、次のクエリは *first-name* 要素と *last-name* 要素の両方を返します。

```
select xmlextract('//author/concat(first-name, last-name)', text_doc)
from sample_dcs where name_doc='bookstore'
----------------------------------------
JoeBobMaryBobToniBob
```

また、concat を使用して、結果をフォーマットしたり、区切ったりすることもできます。次に例を示します。

```
select xmlextract
('//author/concat(",first(",first-name, ")-last(",last-name, ") ")' ,  text_doc)
from sample_docs where name_doc='bookstore'
-----------------------------------------------
first(Joe)-last(Bob) first(Mary)-last(Bob) first(Toni)-last(Bob)
```

カッコで囲んだ式

Adaptive Server 12.5.2 ではカッコで囲んだ式をサポートしています。「XML 問い合わせ言語の機能拡張」(14 ページ) では、XPath におけるカッコで囲んだ式の一般的な構文について説明しています。以降の各項では、カッコをサブスクリプトや union に使用する方法について説明します。

カッコとサブスクリプト

サブスクリプトは直前にある式に適用されます。パス内の複数の式をグループ化するには、カッコを使用します。この項の例は、カッコをサブスクリプトとともに使用する方法を示します。

サブスクリプトを使用しない次の一般的な例は、book 要素内のすべてのタイトルを返します。

```
select xmlextract('/bookstore/book/title', text_doc) from
sample_docs where name_doc='bookstore'
------------------------------
<title>Seven Years in Trenton</title>
<title>History of Trenton</title>
<title>Tracking Trenton</title>
<title>Treanton Today, Trenton Tomorrow</title>
<title>Whos Who in Trenton</title>
```

最初のタイトルのみを示すには、“[1]” サブスクリプトを使用した次のクエリを入力できます。

```
select xmlextract
('/bookstore/book/title[1]', text_doc)
from sample_docs where name_doc='bookstore'
-----------------------------------------------
<title>Seven Years in Trenton</title>
<title>History of Trenton</title>
<title>Tracking Trenton</title>
<title>Treanton Today, Trenton Tomorrow</title>
<title>Whos Who in Trenton</title>
```

しかし、上のクエリは書店 (bookstore) にある最初の本のタイトルではなく、それぞれの本 (book) の最初のタイトルを返します。同様に、“[2]” サブスクリプトを使用する次のクエリでは、書店にある 2 番目の本のタイトルではなく、それぞれの本の 2 番目のタイトルを返します。本にはタイトルが 1 つしかないため、結果は空になります。

```
select xmlextract
('/bookstore/book/title[2]', text_doc)
from sample_docs where name_doc='bookstore'
--------------------------------
NULL
```

以上のクエリは書店ではなく本の i 番目のタイトルを返します。これは、サブスクリプト演算 (および述部一般) が直前の項目に適用されるためです。本の 2 番目のタイトルではなく書店全体の 2 番目の本のタイトルを返すには、サブスクリプトの適用対象の要素をカッコで囲みます。次に例を示します。

```
select smlextract
('(/bookstore/booktitle)[2]', text_doc)
from sample_docs where name_doc='bookstore'
-------------------------------------------
<title>History of Trenton</title>
```

パスはすべてカッコでグループ化できます。次に例を示します。

```
select xmlextract('(//title)[2]', text_doc)
from sample_docs where name_doc='bookstore'
-------------------------------------------
<title>History of Trenton</title>
```

カッコと union

カッコを使用して 1 つのステップ内の演算をグループ化することもできます。たとえば、次のクエリは書店にあるすべての本のタイトルを返します。

```
select xmlextract('/bookstore/book/title', text_doc) from
sample_docs where name_doc='bookstore'
---------------------------------------
<title>Seven Years in Trenton</title>
<title>History of Trenton</title>
<title>Trenton Today, Trenton Tomorrow</title>
<title>Who's Who in Trenton</title>
```

上のクエリは本のタイトルしか返しません。雑誌 (magazine) のタイトルを返すには、クエリを次のように変更します。

```
select xmlextract('/bookstore/magazine/title' ,text_doc)
from sample_docs where name_doc='bookstore'
---------------------------------------
<title>Tracking Trenton</title>
```

書店にあるすべての商品のタイトルを返すには、次のようにクエリを変更します。

```
select xmlextract('/bookstore/*/title', text_doc)
from sample_docs where name_doc='bookstore'
-----------------------------------
<title>Seven Years in Trenton</title>
<title>History of Trenton</title>
<title>Tracking Trenton</title>
<title>Trenton Today, Trenton Tomorrow</title>
<title>Whos Who in Trenton</title>
```

書店に本と雑誌以外の商品がある場合 (カレンダ、新聞など)、union (垂直線) 演算子を使用し、クエリ・パスをカッコで囲んで、本と雑誌のタイトルのみを問い合わせることができます。次に例を示します。

```
select xmlextract('/bookstore/(book|magazine)/title', text_doc)
from sample_docs where name_doc='bookstore'
---------------------------------
<title>Seven Years in Trenton</title>
<title>History of Trenton</title>
<title>Tracking Trenton</title>
<title>Trenton Today, Trenton Tomorrow</title>
<title>Whos Who in Trenton</title>
```

XPath 規格のサポートの変更

Adaptive Server 12.5.2 では、xmlextract 関数と xmltest 述部でサポートされている XPath 言語が次のように変更されました。

- 規格以外のサブスクリプト式の削除
- "//" 演算子の解釈の変更

これらの変更によって、xmlextract と xmltest は XPath 規格に準拠するようになりました。

規格以外のサブスクリプト式の削除

Adaptive Server 12.5.2 では、Adaptive Server 12.5.1 でサポートされていた次の 2 つの機能がサポートされなくなりました。

- 負の値を持つサブスクリプト式。次に例を示します。

  ```
  "/bookstore/book[-2]
  ```

- 最初の数値が後の数値より大きいサブスクリプト範囲。次に例を示します。

  ```
  "/bookstore/book[5 to 3]
  ```

“//” 演算子の解釈の変更

XPath の “//” 演算子は任意の数の任意のステップを示します。したがって、次の 2 つのクエリ式は同じことを表します。

```
/bookstore//first-name
```

```
/bookstore/(* | */* | */*/* | */*/*/* | */*/*/*/* )/first-name
```

Adaptive Server 12.5.2 では、XPath 規格を反映して “//” 演算子の解釈が変更されました。

著者の名前が “Mary” であるすべての本のタイトルを返す述部を考えてみます。このクエリは次のように記述することができます。

```
(a) /bookstore/book[author/first-name = "Mary"]/title
```

Adaptive Server 12.5.1 では、次のように “//” 演算子を使用して述部で first-name を参照することもできました。

```
(b) /bookstore/book[//first-name = "Mary"]/title
```

Adaptive Server 12.5.1 では、先頭の “//” 演算子を現在の book 要素に含まれるすべての first-name 要素への相対参照と解釈しました。

しかし、XPath 規格では、先頭の “//” 演算子はドキュメント全体のすべての first-name 要素を参照する絶対参照であると指定されています。現在の book に含まれている first-name 要素を参照するには、“//” 演算子の前に "." 演算子を加え、現在のコンテキストを示します。

```
(c) /bookstore/book[.//first-name = "Mary"]/title
```

Adaptive Server 12.5.2 では “//” 演算子を XPath 規格に従って解釈します。Adaptive Server 12.5.1 で (b) 形式で記述したクエリは、Adaptive Server 12.5.2 では (c) 形式で記述してください。

(b) 形式で記述したクエリを Adaptive Server 12.5.2 で使用すると、例外が発生します。

```
select xmlextract( '/bookstore/book[//first-name = "Mary"]/title', text_doc)  from
sample_docs where name_doc='bookstore'
--------------------------------------------------
Msg 14833, Level 16, State 0:
Line 1: Absolute paths inside a filter operator are not supported.
```

このようなクエリは、ASE 12.5.2 の形式である (c) 形式で記述してください。

```
select xmlextract( '/bookstore/book[.//first-name ="Mary"]/title', text_doc)
from sample_docs where name_doc='bookstore'
--------------------------------------------------
<title>History of Trenton</title>
```

xmlextract、*xmlparse*、*xmlrepresentation* でサポートされるデータ型の拡張

Adaptive Server 12.5.2 では、次の関数でサポートするデータ型が拡張されています。

- xmlextract
- xmlparse
- xmlrepresentation

xmlextract の変更

xmlextract の returns 句で java.lang.String をサポートするようになりました。

```
returns_type ::= [,] returns { varchar[(integer)] | text | image |
    java.lang.String }
```

xmlparse の変更

xmlparse にオプションの returns 句が追加されました。この句を使用すると、返される解析済み XML ドキュメントのデータ型を指定できます。

```
xmlparse_call ::=
xmlparse(general_string_expression[options_parameter] [returns_type])
options_parameter ::= [,] option option_string
returns type ::= [,] returns {image | binary | varbinary[(integer)]}
```

returns 句を省略した場合、デフォルトは returns image です。

xmlrepresentation の変更

xmlrepresentation では、image データ型に加え、XML ドキュメント・オペランドに対する binary と varbinary のデータ型をサポートするようになりました。つまり、xmlrepresentation は *image* 型、*binary* 型、または *varbinary* 型のパラメータを調べ、パラメータに解析済み XML データがあるかどうかを示す整数値を返すことができます。

for xml のマッピングでサポートされるデータ型の拡張

SQL の select 文の for xml 句は、クエリの結果セットを SQLX フォーマットの XML ドキュメントにマップします。構文と使用方法については、『Adaptive Server Enterprise における XML Services』の第 4 章にある「for xml 句」を参照してください。

Adaptive Server 12.5.2 では、select...for xml 文で次のデータ型を指定できるようになりました。

- binary と varbinary
- image
- text
- java.lang.String

binary 型、varbinary 型、image 型のデータは、生成された XML ドキュメントで hex 値または base64 値で表されます。値の型を指定するには、『Adaptive Server Enterprise における XML Services』の第 5 章にある「SQLX オプションの定義」で説明されているように、オプション “binary = { hex | base64 }” を使用します。

例

次の 2 つの例は、binary 型の値 0x123abc を含む SQLX 結果セットの異なるバージョンを示します。

次の例は hex 値を指定しています。

```
select 0x123abc for xml option 'binary=hex'
------------------------------
<resultset xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <row>
                <C1>123abc</C1>
        </row></resultset>
```

次の例は base64 値を指定しています。

```
select 0x123abc for xml option 'binary=base64'
--------------------------------------
<resultset xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
>       <row>
                <C1>Ejq8</C1>
        </row>
```

Java 関数を使用して階層構造の XML ドキュメントと SQL データをマップする

Adaptive Server 12.5.2 では、SQL テーブルまたは結果セットと階層構造の XML ドキュメントとの間でデータをマップする次の 2 つの新しいクライアント指向の Java ベース XML 関数をサポートしています。

- ForXmlTree － SQL テーブルまたは結果セットの集合をツリー構造の XML ドキュメントにマップします。
- OpenXml － ツリー構造の XML ドキュメントの繰り返しデータを SQL テーブルに抽出します。

以降の各項では、サンプル・データを使用して、ForXmlTree と OpenXml の使用方法の概要と例を示します。詳細については、*$SYBASE/$SYBASE_ASE/sample/XML/xml-util.{doc, pdf}* を参照してください。

サンプル・データとツリー構造の XML 表現

SQL データはテーブルに格納され、外部キー・カラムとプライマリ・キー・カラムによってテーブル間のツリー構造関係が形成されます。このようなデータを XML で記述する場合、ツリー構造関係は通常ネストされた要素で表されます。

たとえば、表 2-1 に示すデータを含むテーブルを考えてみます。

表 2-1: サンプル・テーブル

テーブル・データ
depts(dept_id, dept_name)
emps(emp_id, emp_name, dept_id)
emp_phones(emp_id, phone_no
projects(project_id, dept_id)

表 2-1 のデータをツリー構造の XML で表すと、次のようになります。

```
<sample xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<depts>
<dept>
      <dept_id>D123</dept_id>
      <dept_name>Main</dept_name>
       <emps>
           <emp>
               <emp_id>E123</emp_id>
               <emp_name>Alex Allen</emp_name>
               <salary>912.34</salary>
               <phones>
                   <phone><phone_no>510.555.1987</phone_no></phone>
                   <phone><phone_no>510.555.1876</phone_no></phone>
                    <!-- other phone elements for this emp -->
               </phones>
          <!--  Other emp elements for this dept  -- >
```

```
        </emps>

        <projects>
             <project>
                  <project_id>PABC</project_id>
                  <budget>598.76</budget>
             </project>
             <!--  Other project elements for this dept - ->
         </projects>
     </dept>
    <!--  other dept elements for this set of depts. -->
</depts>
</sample>
```

ForXmlTree を使用して階層構造の XML に SQL データをマップする

新しい Java ベースの関数 ForXmlTree は、SQL テーブルまたは結果セットの集合をツリー構造の XML ドキュメントにマップします。この関数は Adaptive Server 12.5.1 で導入された SQL の select コマンドの for xml 句を基にしています。

select...for xml は次のタスクを実行します。

- 1 つの SQL 結果セットを 1 つの XML ドキュメントにマップする。
- SQL 結果セットから XML への直接マッピングを生成する。たとえば select が、各ローに 20 のカラムを含む 1000 ローの結果セットを返す場合、for xml が返す XML ドキュメントには 1000 個のロー要素があり、各要素に 20 個のカラム要素が含まれます。

新しい Java ベースの関数 ForXmlTree には次のような特徴があります。

- SQL サーバ、クライアントのコマンド・ライン、クライアントまたはサーバの Java アプリケーションのいずれかで呼び出すことができる。
- 結果セットの集合を 1 つのツリー構造の XML ドキュメントにマップする。
- 出力ツリーとそのツリーの各ノードに含める SQL データを記述する *<forxmltree>* 指定引数が必要である。
- ツリー構造の出力 XML ドキュメントの各ノードに for xml 形式の XML データ・マッピングを生成する。

このため、ForXmlTree は 2 次元の for xml マッピングができると考えることができます。たとえば、ForXmlTree に次の *<forxmltree>* を入力すると、「サンプル・データとツリー構造の XML 表現」(27 ページ) に示した XML ドキュメントが生成されます。

```
1)  <!-- A forxmltree spec for depts-emps-phones-projects, with aggregation -->
2)  <forxmltree treename="sample">
3)  <node> <!-- The node element for depts -->
4)    <query>  select * from depts order by dept_id  </query>
5)    <options> tablename=depts rowname=dept </options>
```

```
6)     <link  variablename="@dept_id" columnname="dept_id" type="char(11)"  />
7)     <node>  <!-- The node element for emps, under depts -->
8)          <query>
9)             select emp_id, emp_name, salary from emps e
10)             where e.dept_id = @dept_id order by emp_id
11)         </query>
12)         <options> tablename=emps rowname=emp   </options>
13)         <link variablename="@emp_id" columnname="emp_id"  type="char(6)"  />
14)           <node> <!-- The node element for phones, under emps -->
15)               <query>
16)                  select phone_no from emp_phones ep where ep.emp_id = @emp_id
17)               </query>
18)               <options> tablename=phones rowname=phone  </options>
19)            </node>  <!-- End the node for phones -->
20)     </node>  <!-- End the node for emps --
21)     <node>  <!-- The node element for projects, under dept -->
22)         <query>
23)             select project_id, budget from projects p
24)             where p.dept_id = @dept_id order by project_id
25)         </query>
26)         <options> tablename=projects rowname=project  </options>
27)     </node>  <!-- End the node for projects -->
28) </node>  <!-- End the node for depts -->
29) </forxmltree>
```

OpenXml を使用して階層構造の XML を SQL にマップする

「ForXmlTree を使用して階層構造の XML に SQL データをマップする」(28 ページ)で説明した ForXmlTree 関数は SQL テーブルまたは結果セットの集合を階層構造の XML ドキュメントにマップします。OpenXml 関数はこれとは逆の処理を行い、入力 XML ドキュメントから SQL テーブルのデータを抽出します。

OpenXml は Adaptive Server 12.5.1 で導入された xmlextract 関数に似ています。この関数は特定の XML ドキュメントから指定したデータ値を抽出します。xmlextract では XML ドキュメントと 1 つの XPath クエリ式を指定し、XPath クエリを XML ドキュメントに適用した結果を返します。

新しい Java ベースの OpenXml 関数には次のような特徴があります。

- クライアントのコマンド・ラインまたはクライアントの Java アプリケーションから呼び出すことができる。SQL サーバでの使用は想定されていません。
- 特定の XML ドキュメントを示す引数と、目的の出力ローを抽出する XPath クエリと各出力ローの必要なカラムを抽出する Xpath クエリを指定するオプション・セットを示す引数が必要である。

したがって、OpenXml は 2 次元の xmlextract と見なすことができます。

OpenXml は次のどちらか一方または両方のアクションを行います。

- SQL テーブルを作成し、抽出したデータを移植する SQL スクリプトを生成する。
- そのスクリプトを実行し、抽出したデータで SQL テーブルを作成する。

以下の例は、「サンプル・データとツリー構造の XML 表現」(27 ページ) に示した XML ドキュメントが *example-document.xml* に格納されていると想定しています。

例 8

次の例は、XML ドキュメントから depts、emps、emp_phones、projects の各テーブルを抽出する 4 つのクライアント・コマンド・ライン呼び出しを示します。

```
java jcs.xmlutil.OpenXml -i "file:example-document.xml"  ¥
     -r "file:depts.opt"  -o "depts.sql"

java jcs.xmlutil.OpenXml -i "file:example-document.xml"  ¥
       -r "file:emps.opt"  -o "emps.sql"

java jcs.xmlutil.OpenXml -i "file:example-document.xml"  ¥
       -r "file:emp-phones.opt"  -o "emp-phones.sql"

java jcs.xmlutil.OpenXml -i "file:example-document.xml"  ¥
       -r "file:projects.opt"  -o "projects.sql"
```

例 9

この例は、例 8 に示したコマンド・ラインで呼び出されるオプションの内容を示します。これらのオプションは、OpenXml 呼び出しで抽出するデータと、その格納先の SQL テーブルを指定します。

```
-- Content of input file "depts.opt
"tablename='depts_ext'
rowpattern='//dept'
columns=
  '   dept_id char( 4 ) "/@dept_id"
      dept_name varchar(50) "/@dept_name" '

-- Content of input options file "emps.opt"
tablename='emps_ext'
rowpattern='//dept/emps/emp'
columns=
  '   emp_id char( 4 ) "/emp_id/text()"
      emp_name varchar(50) "/emp_name/text()"
       dept_id char(4) "/../../@dept_id"
       salary dec(7,2) "/salary/text()"

'-- Content of input options file "emp-phones.opt"
```

```
tablename='emp_phones_ext'
rowpattern='/sample/dept/emps/emp/phone'
columns=  '   emp_id char( 4 ) "/../emp_id/text()"
       phone_no varchar(20) "/@phone_no" '

--Content of input options file "projects.opt"
tablename='projects_ext'
rowpattern='//dept/projects/project'
columns=
  '   project_id char( 4 ) "/project_id/text()"
      dept_id char(4) "/../../@dept_id"
      budget dec(7,2) "/budget/text()" '
```

例 10

この例は、最初の OpenXml 呼び出しで生成される SQL スクリプトを示します。このスクリプトは、テーブルを作成し、depts テーブルの抽出データをそのテーブルに移植します。例 8 に示した後続の OpenXml 呼び出しはそれぞれ、emps、emp_phones、projects のデータに対する同様のスクリプトを生成します。

```
-- output file depts.sql

create table depts_ext
    (dept_id char( 4 ) null, dept_name varchar(50) null )

insert into depts_ext values('D123',   'Main')

insert into depts_ext values('D234',  'Auxiliary')

insert into depts_ext values('D345',  'Repair')
```

複数の結果セット・クエリに対する Java SQLX マッピング

select ... for xml 文と Java ベースの SQLX マッピング関数は 1 つの SQL 結果セットを SQLX フォーマットの 1 つの XML ドキュメントにマップします。Adaptive Server 12.5.2 が提供する新しい Java ベースの SQLX マッピング関数 forxmlmultiplej は、SQL クエリの複数の結果セットを 1 つの XML ドキュメントにマップします。

forxmlmultiplej

説明

SQL クエリの結果セットを XML ドキュメントにマップします。複数の結果セットをマップできます。

構文

```
forxmlmultiplej_function ::=
    forxmlmultiplej(sql_query_expression, option_string)
```

オプション

sql_query_expression と *option_string* については、『Adaptive Server Enterprise における XML Services』の「第 4 章 XML マッピング関数」にある「forxmlj、forxmldtdg、forxmlschemaj、forxmlallj」を参照してください。

使用法

- *sql_query_expression* は複数の結果セットを返すことができ、SQL の print コマンドを含めることができます。
- forxmlmultiplej の例と詳細については、*$SYBASE/$SYBASE_ASE/sample/XML/Using-SQLX-mappings.htm* で、複数の結果セットについての項目を参照してください。

第 3 章

Real Time Messaging Services

Adaptive Server Enterprise 12.5.2 には、Real Time Data Services (RTDS) オプション・パッケージによるメッセージング機能が含まれています。このオプションによって、メッセージング・システムおよびデータベースとの対話を行うアプリケーションの開発が簡単になります。

RTDS を使用すると、Adaptive Server データベースでのトランザクション (データ変更) を取得して、それらを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ことができます。このようなデータ変更 (イベント) は、TIBCO Enterprise™ for JMS によって提供される JMS (Java Messaging Service) メッセージ・バスを介してアプリケーションに渡されます。メッセージ・プロバイダとの間のメッセージの送信と受信には、Adaptive Server 提供の Transact-SQL を使用します。

Real Time Messaging Services オプションの詳細については、『Real Time Data Service ユーザーズ・ガイド』を参照してください。

キューとの間のメッセージの送受信

Transact-SQL アプリケーションは、msgsend 関数と msgrecv 関数を使用して JMS キューとの間でメッセージを送受信できます。メッセージ本文すなわちペイロードは、アプリケーションを使用して構成するか、リレーショナル・テーブルの文字データまたはバイナリ・データを含むことができます。リレーショナル・データまたは Adaptive Server アプリケーションからメッセージ・プロパティの値を構成し、メッセージを送信するときにメッセージ・プロパティを指定することができます。

Adaptive Server アプリケーションでは JMS キューからのメッセージの読み込みを処理できます。また、そのメッセージをリレーショナル・テーブルに挿入できます。読み込み操作を実行するとき、メッセージ・セレクタを指定して特定のメッセージをフィルタ処理することもできます。読み込みメッセージのメッセージ・プロパティは Adaptive Server アプリケーションで個別に処理できます。

トピックとのメッセージのパブリッシュとコンシューム

Transact-SQL アプリケーションは、msgpublish 関数と msgconsume 関数を使用して JMS トピックとの間でメッセージをパブリッシュまたはコンシュームできます。まず sp_msgadmin 'register' を使用してサブスクリプションを登録します。これにより、msgpublish、msgconsume、msgsubscribe、msgunsubscribe が参照できる名前が作成されます。サブスクリプションは、持続的または非持続的として登録できます。

- 持続的なサブスクリプションは、メッセージ・コンシューマ・アプリケーションが接続されていないときでもメッセージ・コンシューマのためにメッセージを保持します。メッセージを保持するのは、Adaptive Server ではなくメッセージ・プロバイダです。
- 非持続的なサブスクリプションは、コンシューマ・アプリケーションがメッセージ・プロバイダに接続されているときのみメッセージを保持します。

メッセージ・セレクタを指定して受け取るメッセージを制御すると、必要なメッセージのみを読み込むようにできます。

msgsubscribe を使用すると、Adaptive Server アプリケーションがメッセージを処理できるようになるまでメッセージを保持するように JMS プロバイダに通知できます。また、msgunsubscribe を使用すると、このサブスクリプションのメッセージがアプリケーションで不要になったことを JMS プロバイダに通知できます。msgunsubscribe では、持続的なサブスクリプションを JMS プロバイダから削除することもできます。読み込まれたメッセージのメッセージ・プロパティは、アプリケーションで個別に処理できます。

例

以下の例は、Transact-SQL メッセージング・インタフェースの概要を示します。

例 1 この例は、メッセージをキューに送信します。

```
select msgsend('hello world',
'tibco_jms:tcp://my_jms_host:7222?queue=queue.sample 'MESSAGE PROPERTY'
'city=Detroit')
```

例 2 この例は、キューからメッセージを読み込みます。フィルタを使用する場合と使用しない場合を示します。

```
select msgrecv('tibco_jms:tcp://my_jms_host:7222?queue=queue.sample')
select msgrecv
('tibco_jms:tcp://my_jms_host:7222?queue=queue.sample 'MESSAGE SELECTOR'
'city=Detroit')
```

例 3　この例は、メッセージをトピックにパブリッシュします。

```
sp_msgadmin register, subscription, sub1,
'tibco_jms:tcp://my_jms_host:7222?topic=topic.sample',
select msgpublish('hello world', 'sub1' 'MESSAGE PROPERTY' 'city=Boston')
```

例 4　この例は、トピックのメッセージをコンシュームします。

```
select msgconsume('sub1')
```

例 5　この例は、プロパティの処理を示します。

```
select msgconsume('sub1')
declare @pcount integer
declare @curr integer
declare @pname varchar(100)
select @curr=1
select @pcount = msgpropcount()
while(@curr<=@pcount)
begin
select @name=msgpropname(@curr)
select msgproptype(@pname)
select msgpropvalue(@pname)
select @curr=@curr+1
end
```

第 4 章

Web Services Consumer

Web サービスは、ネットワーク接続を介して利用できる独立したモジュール方式のアプリケーションです。Web サービスを使用すると、SOAP (Simple Object Access Protocol)、WSDL (Web Services Description Language)、HTTP、XML (Extensible Markup Language) のオープン規格に準拠するため、パフォーマンスは劣化しますが、相互運用性が向上します。

実装に使用されているプログラミング言語にかかわらず、さまざまなプラットフォームやオペレーティング・システムから Web サービスを利用できるため、異なるアプリケーションでのデータ共有機能が大幅に強化されます。限定された特定のタスクを扱う各種 Web サービスを使用して、既存のソフトウェアを安全で管理された環境に公開することにより、企業の統合を動的かつ加速的に進めることができます。Web サービスは、リモート・アプリケーションを起動するための標準化された手段を提供することによって、インフラストラクチャに必要なコードの量を減らします。また、Web サービスは、公開されたインタフェース (WSDL) からユーザが実装を取り出せるようにすることによって、サービス指向アーキテクチャ (SOA) の構築に必要なツールを提供します。

Adaptive Server Enterprise Web Services のコンポーネント

Adaptive Server Enterprise Web Services は、Web Services Producer と Web Services Consumer の 2 つのコンポーネントから構成されます。どちらのコンポーネントも Adaptive Server から独立して動作し、同じライセンスである ASE_WEBSERVICES によって有効になります。

- Web Services Producer は、クライアント・アプリケーションから SOAP を使用して Adaptive Server の SQL とストアド・プロシージャにアクセスできるようにするコンポーネントです。

  たとえば、ユーザが作成したアプリケーションからファイアウォールを通して Adaptive Server にアクセスする必要がある場合、そのアプリケーションは Web Services Producer を使用することによって、HTTP/HTTPS と SOAP を使ってインターネット経由で Transact-SQL にアクセスできます。

  Web Services Producer の詳細については、『Web Services ユーザーズ・ガイド』の「第 2 章 Web Services Producer について」を参照してください。

- Web Services Consumer は、 Adaptive Server 12.5.2 で新たに追加されたコンポーネントです。Web Services Consumer コンポーネントは、実行時にほかのアプリケーションの Web サービスを Adaptive Server のプロキシ・テーブルにマップすることによって、Adaptive Server がそれらの外部 Web サービスにアクセスできるようにします。

  たとえば、あるユーザが Web サービスのデータと Adaptive Server のリレーショナル・データを統合する必要があるとします。Web Services Consumer は、Web サービスをプロキシ・テーブルに動的にマップしたり、Web サービスの出力を Adaptive Server の結果セットにマップする手段を提供します。この機能によって、ユーザはストアド・プロシージャ、トリガ、またはビューで Transact-SQL を使用した Web サービスの出力の取得および操作ができます。

  Web Services Consumer の詳細については、『Web Services ユーザーズ・ガイド』の「第 3 章 Web Services Consumer について」を参照してください。

sp_webservices

sp_webservices ストアド・プロシージャは Web Services の Consumer コンポーネントで使用されるプロキシ・テーブルの作成と管理を行います。sp_webservices には次のオプションがあります。

- add − プロキシ・テーブルを作成する。
- list − WSDL ファイルにマップされたプロキシ・テーブルをリストする。
- modify − タイムアウト、ユーザ名、パスワードの設定を変更する。
- remove − WSDL ファイルにマップされたプロキシ・テーブルを削除する。
- help − sp_webservices の使用方法を表示する。

sp_webservices の詳細については、『Web Services ユーザーズ・ガイド』の「第 5 章 Adaptive Server Enterprise Web Services の使用方法」を参照してください。

第 5 章

IPv6 のサポート

この章では、IPv6 サポートの実装について説明します。

IPv6 を認識する Adaptive Server の起動

Adaptive Server を IPv6-aware (認識) にするには、トレース・フラグ 7841 を使用して Adaptive Server を起動します。これによって、Adaptive Server が IPv6 の可用性を判別できるようになり、IPv6-aware (認識) になります。

注意 IPv6 は 32 ビット版と 64 ビット版の Sun Solaris プラットフォームで使用できます。使用しているプラットフォームでの IPv6 を有効化しているネットワークの設定と管理の方法については、オペレーティング・システムのマニュアルを参照してください。

IPv6 の用語

IPv6 の移行プロセス

IPv6 のインストールと設定は、ユーザに対して透過的に行われます。役に立つ用語を次に示します。

- IPv4-only (専用) ノード － IPv4 のみを実装するノード。IPv4-only (専用) ノードは、ネーム・サービス・データベースに IPv4 アドレスしかありません。
- IPv6-only (専用) ノード － IPv6 のみを実装するノード。IPv6-only (専用) ノードは、ネーム・サービス・データベースに IPv6 アドレスしかありません。

- デュアル・ノード − IPv4 と IPv6 の両方を実装するノード。IPv4-only (専用) ノードをアップグレードするときはデュアル・ノードにアップグレードすることが想定されます。
- IPv6-enabled (有効化) ノード − デュアル・ノードを実装し、少なくとも 1 つの IPv6 インタフェースを持つノード。

IPv6 アドレス指定の用語

- リンクローカル・アドレス − 1 つのリンク経由だけで使用できる IPv6 アドレス。
- サイトローカル・アドレス − 1 つのサイト内だけで使用できる IPv6 アドレス。
- グローバル・アドレス − グローバルなインターネットにわたって使用できる IPv6 アドレス。

IPv6 アプリケーションのタイプ

- IPv6-unaware (非認識) − IPv6 アドレスを処理できないアプリケーション。
- IPv6-aware (認識) − IPv4 アドレスを持たないノードと通信できるアプリケーション。API が実際のアドレスの内容とフォーマットを隠す場合など、これはアプリケーションに対して透過的になることがあります。
- IPv6-enabled (有効化) − IPv6-aware (認識) の特徴を持ち、さらに IPv6 の一部の機能を利用できるアプリケーション。
- IPv6-required (要求) − IPv6 の機能を必要とし、IPv4 経由では動作しないアプリケーション。

第 6 章

拡張型全文検索機能の変更

この章では、Adaptive Server バージョン 12.5.2 の EFTS (Enhanced Full-Text Search：拡張型全文検索) 機能の変更点について説明します。

注意 HP Tru64 用の EFTS は、既存の 12.5.1 バージョンの EFTS です。

インストールの変更

Adaptive Server バージョン 12.5.2 の EFTS にはいくつかの変更や機能拡張が実装されています。これらの変更のため、Adaptive Server バージョン 12.5.2 の EFTS にアップグレードするときは次の点に注意してください。

- 以前のバージョンの EFTS で作成されたコレクションはこのバージョンの EFTS とは互換性がありません。
- 最新バージョンの EFTS にアップグレードする前に既存のコレクションを削除する必要があります。
- アップグレードの後でコレクションを再作成する必要があります。

バージョン 12.5.2 の EFTS は *EFTS-12_5_2* という名前のディレクトリにインストールされ、既存の *EFTS-12_5* ディレクトリは上書きされません。古いコレクションを削除したら、*SYBASE.csh* ファイルと *SYBASE.sh* ファイルを編集して、SYBASE_FTS 環境変数の値が *EFTS-12_5_2* ディレクトリを指すように変更します。

スタイル・ファイルの新しいディレクトリ

EFTS によって使用される Verity スタイル・ファイルを含むディレクトリが変更されました。EFTS では、*$SYBASE/$SYBASE_FTS/verity/common/style* に含まれるスタイル・ファイルは使用されなくなりました。

現在、EFTS では、*$SYBASE/$SYBASE_FTS/verity/common/styles/txtsvr* に含まれるスタイル・ファイルが使用されるようになりました。

新機能

Adaptive Server バージョン 12.5.2 では、EFTS の多数の新機能が導入されています。

停止を行うためのパーミッション

EFTS を停止できるのは sa_role を持つユーザのみです。

新しい疑似カラム *total_docs*

Adaptive Server バージョン 12.5.2 では、EFTS は検索基準と一致するドキュメントの合計数を示す整数値を返すようになりました。この値は、total_docs と呼ばれる新しい疑似カラムに返されます。

これは、返される結果の数を制限する max_docs カラムを使用するときにも役立ちます。

最大 16000 バイトをサポートする *index_any* 句

以前のバージョンの Adaptive Server では、EFTS は 255 バイトを上回る index_any 句をサポートしていませんでした。Adaptive Server バージョン 12.5.2 の EFTS では、最大 16000 バイトの index_any 句がサポートされています。

プロキシ・テーブルのカラム定義は、EFTS に送信される句のサイズには影響しないため変更されていません。

プライマリ・キー

Adaptive Server バージョン 12.5.2 以降では、プライマリ・キーを text id カラムとして使用できます。IDENTITY カラムまたは適切なプライマリ・キーを含むテーブルに対してテキスト・インデックスを作成できます。単一の decimal、numeric、int、smallint、または tinyint カラムに定義されたプライマリ・キーも対象になります。decimal カラムと numeric カラムは、位取りが 0 であることが必要です。

テキスト・インデックスを作成するとき、EFTS はソース・テーブルの IDENTITY カラムを最初に探し、これをテキスト・インデックスの ID カラムとして使用します。IDENTITY カラムが見つからない場合、EFTS は、テキスト・インデックスの ID カラムとして使用できる適切なプライマリ・キーを探します。

第 7 章

セキュリティの変更

この章では、Adaptive Server バージョン 12.5.2 に実装されているセキュリティの変更点について説明します。

概要

Adaptive Server のセキュリティ機能は次の 4 つのカテゴリに分けられます。

- 識別と認証 (I&A)
- アクセス制御
- 責任
- 暗号化技術

Adaptive Server 12.5.2 では、これら 4 つのカテゴリすべてで変更が行われています。

識別と認証

I&A は、Adaptive Server でユーザを明確に識別するために使用される機能です。ユーザが識別されてから、アクセス制御メカニズムとユーザの責任が実施されます。

Adaptive Server 12.5.2 でサポートされる I&A の新機能と拡張機能は次のとおりです。

- 機能拡張された Kerberos
- LDAP ユーザ認証
- PAM ユーザ認証
- 機能拡張されたログイン制御

Kerberos

Kerberos は、シークレット・キー暗号法を使用するネットワーク認証プロトコルであり、これによってクライアントがネットワーク接続経由でサーバに ID を証明できます。ユーザがオペレーティング・システムにログインしたとき、または認証プログラムを実行することにより、ユーザ・クレデンシャルが取得されます。このクレデンシャルは、認証を実行するときに各アプリケーションによって使用されます。ユーザは 1 回ログインすれば各アプリケーションにログインする必要はありません。

Adaptive Server 12.5.2 では、Kerberos は次のようにサポートされます。

- 次のプラットフォームでは CyberSafe Kerberos ライブラリが使用されます。
  - Sun Solaris 32 ビット
  - Sun Solaris 64 ビット (Adaptive Server バージョン 12.5.2 で新規に対応)
  - Windows
  - AIX 32 ビット
- 次のプラットフォームでは MIT Kerberos ライブラリ・バージョン 1.3.1 が使用されます (Adaptive Server バージョン 12.5.2 で新規に対応)。
  - Sun Solaris 32 ビット
  - Sun Solaris 64 ビット
  - Linux 32 ビット
- 次のプラットフォームではネイティブ・ライブラリが使用されます (Adaptive Server バージョン 12.5.2 で新規に対応)。
  - Sun Solaris 32 ビット
  - Sun Solaris 64 ビット
  - Linux 32 ビット

注意 Kerberos セキュリティ・オプションを有効にするには、“Security and directory services” パッケージである ASE_SECDIR が必要です。

Kerberos の互換性

表 7-1 は、各種 Kerberos がサポートされるプラットフォームを示します。

表 7-1: Adaptive Server バージョン 12.5.2 における Kerberos の相互運用性

ハードウェア・プラットフォーム	KDC サーバ	GSS クライアント
Solaris 32	CSF、AD、MIT	CSF、MIT、ネイティブ
Solaris 64	CSF、AD、MIT	CSF、MIT、ネイティブ
Linux 32	CSF、AD、MIT	MIT、ネイティブ
Windows 32	CSF、AD	CSF
AIX 32	CSF	CSF

この相互運用性の表では次の略称を使用しています。

- CSF － CyberSafe 社
- AD － Microsoft Active Directory
- MIT － MIT バージョン 1.3.1

Kerberos の設定

設定プロセスは、使用する Kerberos の種類に関係なく共通です。Kerberos を設定するには、次の手順を実行します。

1　サードパーティ製 Kerberos ソフトウェアを設定して、Kerberos 管理ユーザを作成します。これには、次の処理を行います。

- Kerberos クライアント・ソフトウェアを、Open Client Server クライアントまたは Adaptive Server が稼働するマシンにインストールします。次のクライアント・パッケージは動作が確認されています。
  - CyberSafe TrustBroker 4.0
  - MIT Kerberos バージョン 1.3.1
- Kerberos KDC サーバを別の専用マシンにインストールします。

注意　CyberSafe TrustBroker 4.0、MIT Kerberos v.1.3.1、Microsoft Windows Active Directory の KDC は、Adaptive Server とともに使用できることが確認されています。

- Kerberos サーバに、管理権限を持つ管理者アカウントを作成します。このアカウントは、後のクライアント作業 (クライアント・マシンでのプリンシパルの作成など) で使用します。

注意 この後の手順は Kerberos クライアント・マシンで実行します。

2 Adaptive Server の Kerberos プリンシパル *ase120srv* または *ase120srv@MYREALM* を追加します。

3 プリンシパル *ase120srv@MYREALM* の *keytab* ファイルを抽出し、次のようにファイルとして保存します。

```
/krb5/v5srvtab
```

次の UNIX の例では、CyberSafe または MIT Kerberos で利用可能なコマンド・ライン・ツール kadmin を使用します。Kerberos とユーザの管理を支援する GUI ツールもあります。

```
CyberSafe Kadmin:
% kadmin aseadmin
Principal - aseadmin@MYREALM
Enter password:
Connected to csfA5v01 in realm ASE.
Command: add ase120srv
Enter password:
Re-enter password for verification:
Principal added.
Command: ext -n ase120srv
Service Key Table File Name (/krb5/v5srvtab):
Key extracted.
Command: quit
Disconnected.
```

運用環境では、*keytab* ファイルへのアクセスを制御してください。*keytab* ファイルの読み込みを許可されているユーザは、使用しているサーバになり代わるサーバを作成できます。

chmod と chgrp を使用して、*/krb5/v5srvtab* を次のように設定してください。

```
-rw-r----- 1 root sybase 45 Feb 27 15:42 /krb5/v5srvtab
```

Active Directory を KDC として使用するときは、Domain Controller にログインしてユーザと Adaptive Server プリンシパルを追加します。Active Directory ユーザーとコンピュータ・ウィザードを使用して、ユーザとプリンシパルを作成できます。

Adaptive Server で使用する *keytab* ファイルを抽出するには、ktpass というオプション・ツールが必要です。これは、Microsoft サポート ツール・パッケージに含まれています。

Active Directory を使用する場合、ktpass による *keytab* の抽出は、プリンシパルの作成とは別に実行します。Adaptive Server の *keytab* ファイルは、Windows では CyberSafe プログラム・ファイルと同じ場所にあります。たとえば、CyberSafe ソフトウェアが C ドライブにインストールされている場合、Adaptive Server の *keytab* ファイルは *c:¥Program Files¥CyberSafe¥v5srvtab* に格納されると考えられます。

4　ユーザ “sybuser1” の Kerberos プリンシパルを “sybuser1@MYREALM” として追加します。

5　Adaptive Server を起動し、isql を使用して “sa” としてログインします。この後の手順で、Kerberos セキュリティ・サービスを使用するための Adaptive Server パラメータを設定し、ユーザのログイン・アカウントを作成します。この手順は Windows マシンでも UNIX マシンでも同じです。

- 設定パラメータ use security services を 1 に変更します。

  ```
  1> sp_configure 'use security services', 1
  ```

- ユーザ “sybuser1” のために新しいログインを追加してから、ユーザを追加します。

  ```
  1> sp_addlogin sybuser1, password
  ```

6　Adaptive Server を停止し、管理ファイルと接続設定ファイルを変更します。

- UNIX プラットフォームでは、*$SYBASE/* 内に *interfaces* ファイルがあり、次のようなエントリが含まれています。

  ```
  ase120srv
        master tli tcp myhost 2524
        query tli tcp myhost 2524
        secmech 1 .3.6.1.4.1.897.4.6.6
  ```

  Windows プラットフォームでは、*$SYBASE/ini* 内に *sql.ini* ファイルがあり、次のように同様のサーバ・エントリが含まれています。

  ```
  [ase120srv]
  master=TCP,myhost,2524
  query=TCP,myhost,2524
  secmech=1.3.6.1.4.1.897.4.6.6
  ```

- UNIX プラットフォームでは、*$SYBASE/$SYBASE_OCS/config/* に *libtcl.cfg* ファイルまたは *libtcl64.cfg* ファイルがあります。SECURITY セクションに、CyberSafe Kerberos クライアント・ライブラリに関する次のようなエントリが含まれます。

  ```
  [SECURITY]
  csfkrb5=libskrb.so secbase=@MYREALM
  libgss=/krb5/lib/libgss.so
  ```

64 ビット版 CyberSafe Kerberos クライアント・ライブラリのエントリは次のようになります。

```
[SECURITY]
csfkrb5=libskrb64.so secbase=@MYREALM libgss= ¥
/krb5/appsec-rt/lib/64/libgss.so
```

MIT Kerberos クライアント・ライブラリを使用するマシンでは、エントリは次のようになります。

```
[SECURITY]
csfkrb5=libskrb.so secbase=@MYREALM
libgss=/opt/mitkrb5/lib/libgssapi_krb5.so
```

OS 提供のネイティブ・ライブラリを使用するマシン (Linux など) では、エントリは次のようになります。

```
[SECURITY]
csfkrb5=libskrb.so secbase=@MYREALM
libgss=/usr/kerberos/lib/libgssapi_krb5.so
```

Windows NT では、*%SYBASE%¥%SYBASE_OCS%¥ini¥libtcl.cfg* ファイルに次のようなエントリが含まれます。

```
[SECURITY]
csfkrb5=libskrb secbase=@MYREALM
libgss=C:¥WinNT¥System32¥gssapi32.dll
```

注意 使用する GSS API ライブラリを指定する `libgss=<gss shared object path>` に注目してください。これは、Adaptive Server 12.5.2 と Open Client Server 12.5.1 の新機能です。複数のバージョンの Kerberos Client ライブラリが 1 台のマシンにインストールされている場合は特に、使用するライブラリのロケーションを明確に指定することが重要です。

- また、*$SYBASE/$SYBASE_OCS/config/* の *objectid.dat* を調べて、*[secmech]* セクションに *csfkrb5* のエントリがあることを確認します。

  ```
  [secmech]
  1.3.6.1.4.1.897.4.6.6 = csfkrb5
  ```

7 環境変数を使用して、*keytab* ファイル、Kerberos 設定ファイル、レルム設定ファイルのデフォルト・ロケーションを無効にできます。これは Kerberos 固有の動作であり、すべてのプラットフォームで同様に機能するとはかぎりません。

たとえば、CyberSafe UNIX プラットフォームでは、CSFC5KTNAME 環境変数を使用して *keytab* ファイルを指定できます。

```
% setenv CSFC5KTNAME /krb5/v5srvtab
```

MIT Kerberos でこれに相当する環境変数は KRB5_KTNAME です。

これらの環境変数の詳細については、各ベンダのマニュアルを参照してください。

アプリケーションによっては、ダイナミック・ライブラリ検索パスの環境変数も変更する必要があります。UNIX で一般的に使用される環境変数は LD_LIBRARY_PATH です。Windows では通常、PATH が DLL のロケーションを指すように設定されています。アプリケーションでサードパーティのオブジェクトを正しくロードするには、これらの環境変数を変更する必要があります。たとえば、次のコマンドを使用すると、CyberSafe 32 ビット版の *libgss.so* 共有オブジェクトのロケーションが C シェル環境の検索パスに追加されます。

```
% set path = ( /krb5/lib $path )
```

8　Adaptive Server を再起動します。起動時に次のログ・メッセージが表示されます。

```
00:00000:00000:2001/07/25 11:43:09.91 server
Successfully initialized the security mechanism
'csfkrb5'. The SQL Server will support use of this
security mechanism.
```

9　isql を使用して UNIX ユーザ “sybuser1” として次のように接続します (-U 引数と -P 引数は使用しません)。

```
% $SYBASE/$SYBASE_OCS/bin/isql -Sase120srv -V
1>...
```

次のように暗号化オプションを使用することもできます。

```
$SYBASE/$SYBASE_OCS/bin/isql -Sase120srv 坊 c
```

LDAP ユーザ認証

LDAP では認証を外部で行います。LDAP を使用している場合、認証の決定は、ユーザに代わって Adaptive Server が指定の LDAP サーバに正常にバインドできるかどうかに基づいて行われます。LDAP サーバにバインドするとき、Adaptive Server は指定の LDAP URL から抽出された識別名 (DN) を使用します。

注意　LDAP が有効な場合、パスワード管理は LDAP サービス・プロバイダに委任されます。

Adaptive Server バージョン 12.5.2 では、LDAP 認証ユーザは、すでに Adaptive Server に有効なログインとして存在するユーザであることが必要です。LDAP 認証ユーザの新規 Adaptive Server ログインを自動的に作成するには、次のように入力します。

```
sp_maplogin, LDAP, NULL, "create login"
```

または、LDAP 認証ユーザを既存の Adaptive Server ユーザにマップすることもできます。次に例を示します。

```
sp_maplogin NULL, "externuser", "aseuser"
```

詳細については、「sp_maplogin を使用したログインのマッピング」(59 ページ)を参照してください。

ユーザの DN を検索するための代替アルゴリズム

Adaptive Server バージョン 12.5.2 では、ユーザの DN を検索するために次の処理を行う代替アルゴリズムが追加されています。

- 汎用ログインを使用して LDAP にバインドする方法
- ユーザのログイン ID を含む検索文字列を使用して LDAP ディレクトリを検索する方法
- 検索で返された最初のオブジェクトから DN を抽出する方法

Adaptive Server は、この DN に加えてログイン・パケットのパスワードを使用して LDAP にバインドします。

sp_ldapadmin に追加された新しいオプション

新しい DN 検索アルゴリズムをサポートするために次の新しいオプションが sp_ldapadmin に追加されています。

- Adaptive Server が管理検索に使用するアカウントをユーザが指定できるようにするには、次のオプションを使用します。

```
sp_ldapadmin set_access_acct,
account_distinguished_name, account_password
```

account_distinguished_name の最大長は 255 文字です。*account_password* の最大長は 64 文字で、ディスクに格納されるときは 3DES を使用して暗号化されます。

- 代替認証アルゴリズムを指定するには、次を使用します。

```
sp_ldapadmin set_dn_lookup_url,
           <LDAP URL string for distinguished name lookup>
```

set_dn_lookup_url を NULL 以外の値に設定した場合は、ログインを LDAP Directory Server で認証するために代替アルゴリズムが使用されます。

URL 文字列の最大長は 255 文字です。これを使用して、ログイン名に関連する識別名を検索します。DN を取得するように LDAP URL 文字列の属性名を設定してください。デフォルトは `entrydn` です。

複数のオブジェクトが返された場合は、最初のオブジェクトだけが使用されます。ユーザ・ログインの認証は、プライマリまたはセカンダリ LDAP URL に指定された Directory Server に DN をバインドすることで実行されます。LDAP URL を設定するには、set_primary_url と set_secondary_url を発行します。

次に例を示します。

```
sp_ldapadmin set_access_acct, "cn=admin,
ou=People, dc=mycompany, dc=com", "admin_password"

sp_ldapadmin set_dn_lookup_url,
"ldap://myhost:398/ou=People,dc=mycompany,
dc=com?entrydn?sub?uid=*"

sp_ldapadmin set_primary_url, "ldap://myhost:389/"
```

sp_ldapadmin の詳細については、「第 12 章 グローバル変数、コマンド、ストアド・プロシージャの変更」を参照してください。

次の例は、Windows 2000 コントローラのデフォルトの Microsoft Active Directory スキーマを使用しています。

```
1> sp_ldapadmin set_access_acct, 'cn=aseadmin, cn=Users,
dc=mycompany, dc=com', aseadmin secret password
2> go
1> sp_ldapadmin set_dn_lookup_url,
'ldap://mydomainhostname:389/cn=Users,dc=mycompany,
dc=com?distinguishedName?sub?samaccountname=*'
2> go
1> sp_ldapadmin set_primary_url,ldap://mydomainhostname:389/
2> go
```

“aseadmin” というユーザ名が Active Directory サーバに追加され、ユーザが検出されたツリーとオブジェクトに対する読み込みアクセスが付与されています。このユーザを認証するために、“distinguishedName” で指定された LDAP 属性が取得され、使用されます。フィルタで属性 “samaccountname=*” の検索が指定されています。ワイルドカード (*) は Adaptive Server ログイン・レコードの名前で置き換えられます。

たとえば、“samaccountname=jqpublic” は、“cn=John Q. Public, cn=Users,dc=mycompany, dc=com” という値の DN 属性 “distinguishedName” を Adaptive Server に返します。Adaptive Server はこの文字列を使用して ldap://mydomainhostname:389 にバインドします。バインドが成功すると、認証が成功します。

PAM (Pluggable Authentication Module) のサポート

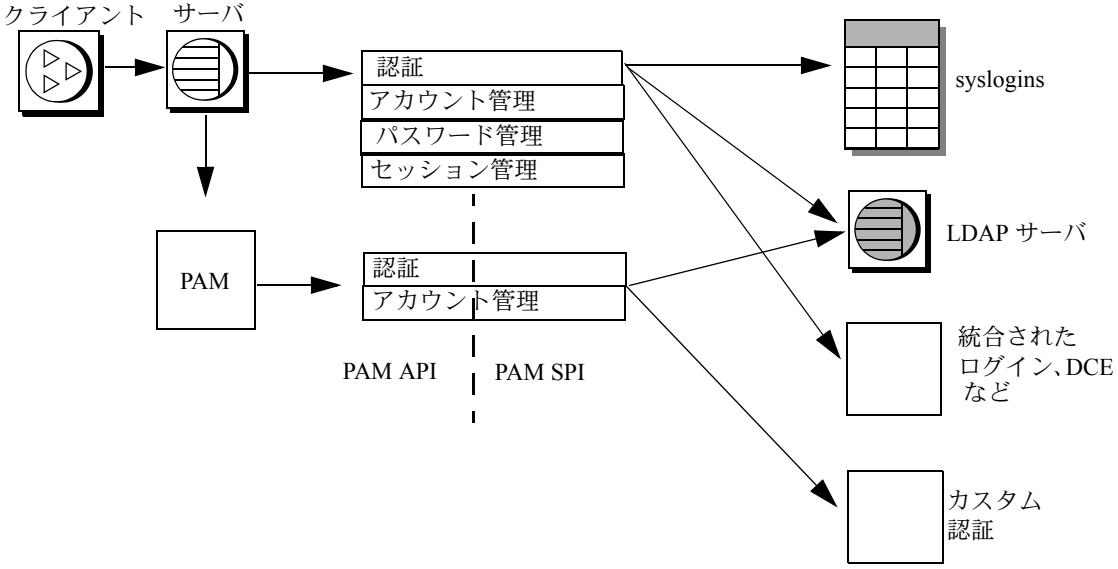
Adaptive Server バージョン 12.5.2 では、PAM (Pluggable Authentication Modules) のサポートが導入されました。これにより、認証を必要とするアプリケーションを変更することなく、複数の認証サービス・モジュールをまとめて使用できるようになります。

PAM により Adaptive Server が Sun や Linux のオペレーティング・システムにさらに密接に統合され、ユーザ・アカウントや認証メカニズムの管理が単純化されます。PAM を使用して密接な統合を実現することで、総保有コスト (TCO) が削減されます。また、ユーザが独自の認証モジュールや許可モジュールをカスタマイズまたは作成できる利点もあります。

注意 現在 PAM がサポートされているプラットフォームは Linux と Solaris です。PAM ユーザ認証の詳細については、各オペレーティング・システムのマニュアルを参照してください。

図 7-1 は、PAM の仕組みを示します。

図 7-1: PAM アーキテクチャ



Adaptive Server は、ログイン・パケットから取得したログイン名とクレデンシャルを PAM API に渡します。PAM は、オペレーティング・システムの設定ファイルの指定に従ってサービス・プロバイダ・モジュールをロードし、認証プロセスを完了するための関数を呼び出します。

Adaptive Server での PAM の有効化

Adaptive Server バージョン 12.5.2 では、PAM 認証ユーザは、すでに Adaptive Server に有効なログインとして存在するユーザであることが必要です。PAM 認証ユーザの新規 Adaptive Server ログインを自動的に作成するには、次のように入力します。

```
sp_maplogin, PAM, NULL, "create login"
```

または、PAM 認証ユーザを既存の Adaptive Server ユーザにマップすることもできます。次に例を示します。

```
sp_maplogin NULL, "externuser", "aseuser"
```

詳細については、「sp_maplogin を使用したログインのマッピング」(59 ページ)を参照してください。

使用する PAM モジュールの決定

Linux と Solaris には定義済みの PAM モジュールがあります。これらのモジュールの 1 つを使用することも、独自のモジュールを作成することもできます。独自のモジュールを作成する場合は、オペレーティング・システムのマニュアルに記載されている PAM モジュールの作成に関する指示に従ってください。

注意 PAM モジュールを作成する場合は、RFC 86.0 “Unified Login With Pluggable Authentication Modules (PAM)” に準拠する必要があります。Adaptive Server では、RFC の認証管理モジュールがサポートされています。アカウント管理、セッション管理、またはパスワード管理のモジュールはサポートされていません。

オペレーティング・システム・ファイルの設定

PAM サポートを有効にするには、各オペレーティング・システムを次のように設定します。

- Solaris では、*/etc/pam.conf* に次の行を追加します。

```
ase auth    required /user/lib/security/$ISA/pam_unix.so.1
```

- Linux では、*/etc/pam.d/ase* という新しいファイルを作成して次の行を入力します。

```
auth       requried /lib/security/pam_unix.so
```

これらのエントリの作成方法の詳細については、オペレーティング・システムのマニュアルを参照してください。

同一マシンでの 32 ビット・サーバと 64 ビット・サーバの実行

$ISA は、命令セット・アーキテクチャを示す環境変数です。これを使用すると、32 ビット版と 64 ビット版のライブラリをともに使用できます。

Solaris の 32 ビット・マシンでは $ISA は空文字列に置き換えられ、64 ビット・マシンでは文字列 “sparcv9” に置き換えられます。

32 ビット版と 64 ビット版の両方が必要な場合は、32 ビット版 PAM モジュールを任意のディレクトリに格納し、64 ビット版 PAM モジュールをそのディレクトリのサブディレクトリに格納します。

pam.conf のエントリは次のようになります。

```
$ ls /usr/lib/security/pam_whatever.so.1
pam_whatever.so.1 -> /wherever/pam_whatever_32bits.so.1

$ ls /usr/lib/security/sparcv9/pam_whatever.so.1
pam_whatever.so.1 -> /wherever/pam_whatever_64bits.so.1

ase     auth    required
/usr/lib/security/$ISA/pam_whatever.so.1
```

注意 *pam.conf* に指定できる変数は $ISA のみです。

PAM ユーザ認証のための Adaptive Server の設定

enable pam user auth は、PAM ユーザ認証サポートを有効にする新しい設定パラメータです。次のように設定できます。

```
sp_configure "enable pam user auth", 0 | 1 | 2
```

パラメータの意味は次のとおりです。

- 0 － PAM 認証を無効にします。これがデフォルト値です。
- 1 － Adaptive Server が最初に PAM 認証を試行し、失敗した場合は syslogins 認証を実行するように指定します。
- 2 － PAM 認証のみを使用できるように指定します。

注意 PAM が有効な場合、パスワード管理は PAM サービス・プロバイダに委任されます。

機能拡張されたログイン制御

Adaptive Server バージョン 12.5.2 では、認証を制御する新しい方法がいくつか導入されています。

“authenticate with” オプション

認証メカニズムはログイン時に定義されます。enable pam user auth は PAM を有効にし、enable ldap user auth は LDAP を有効にします。sp_modifylogin と sp_addlogin に対して新しいオプションを使用して、ログインで特定の認証プロセスを使用するように強制することもできます。

sp_modifylogin

sp_modifylogin の新しいオプション authenticate with には次の値を指定します。

- ASE － syslogin パスワードを使用する Adaptive Server 内部認証を使用します。
- LDAP － LDAP サーバによる外部認証を使用します。
- PAM － PAM による外部認証を使用します。
- ANY － デフォルトのユーザ認証メソッド。ユーザに対して ANY 認証を指定すると、Adaptive Server は外部認証メカニズムが定義されているかどうかを調べます。定義されている場合は、そのメカニズムが使用されます。定義されていない場合は、ASE 認証が使用されます。

  Adaptive Server は次の順序で外部認証メカニズムを調べます。

  - LDAP
  - PAM
  - PAM と LDAP がどちらも有効になっていない場合は、syslogins による ASE 認証が使用されます。

  これらのオプションのいずれかが有効な場合はそのメカニズムが使用され、他は試行されません。このため、LDAP と PAM の両方が有効な場合、ANY 認証のユーザに対して PAM は試行されません。

ログインの authenticate with を設定できるのは、sso_role を持つシステム・セキュリティ担当者だけです。

次に例を示します。

```
sp_modifylogin "nightlyjob", "authenticate with", "ASE"
sp_displaylogin "nightlyjob"
```

これによって次のような出力が表示されます。

```
Suid: 1234
Loginname: nightlyjob
Fullname: Batch Login
Default Database: master
[...]

Date of Last Password Change: Oct 2 2003 7:38 PM
Password expiration interval: 0
Password expired: N
Minimum password length: 6
Maximum failed logins: 0
Current failed login attempts:
Authenticate with: ASE
```

sp_addlogin

sp_addlogin では、認証メカニズムを定義する新しいパラメータ @auth_mech を使用できます。

構文は次のとおりです。

```
sp_addlogin login, passwd [, defdb]
          [, deflanguage] [, fullname] [, passwdexp]
         [, minpwdlen] [, maxfailedlogins] [, auth_mech]
```

auth_mech には、sp_modify login "authenticate with" オプションと同じ値を指定できます。

次の例では、グローバル認証メカニズムを無効にして個々のユーザの設定ができます。

```
sp_addlogin mylogin, mypassword, @auth_mech = ASE
```

sp_displaylogin

sp_displaylogin を使用すると、指定の認証メカニズムがある場合はそれが出力に表示されるようになりました。次に例を示します。

```
1> sp_dislaylogin mylogin
2> go
```

これによって次のような出力が表示されます。

```
Suid: 1234
Loginname: mylogin
Fullname: My Full Name
Default Database: master
Default Language:
Auto Login Script:
Configured Authorization:
Locked: NO
Date of Last Password Change: Oct 2 2003 7:38PM
```

```
Password exiration interval: 0
Password expired: N
Minimum password length: 6
Maximum failed logins: 0
Current failed login attempts:
Authenticate with: ASE
```

sp_maplogin を使用したログインのマッピング

次の構文で sp_maplogin を使用してログインをマップできます。

```
sp_maplogin (authentication_mech | null),
(client_username | null), (action | login_name | null)
```

パラメータの意味は次のとおりです。

- *authentication_mech* は、sp_modifylogin の authenticate with オプションに指定できる有効な値の 1 つです。
- *client_username* は外部ユーザ名です。このユーザ名には、オペレーティング・システム名、LDAP サーバのユーザ名、または PAM ライブラリが認識できる任意の名前を指定できます。null 値を指定すると、すべてのログイン名が有効になります。
- *action* には、create login または drop を指定します。create login を使用すると、ログインが認証されると同時にログインが作成されます。drop はログインを削除するときに使用します。
- *login_name* は、syslogins にすでに存在する Adaptive Server ログインです。

次の例は、外部ユーザ “jsmith” を Adaptive Server ユーザ “guest” にマップします。認証が行われると、“jsmith” は “guest” の権限を得ます。監査ログイン・レコードには、*client_username* と Adaptive Server ユーザ名の両方が表示されます。

```
sp_maplogin NULL, "jsmith", "guest"
```

次の例は、PAM で認証されたすべての外部ユーザについて、ログインが存在しない場合は新規ログインを作成するように Adaptive Server に指示します。

```
sp_maplogin PAM, NULL, "create login"
```

sp_helpmaplogin

sp_helpmaplogin はマップ情報を表示します。パラメータを指定しない場合、すべてのログイン・マップ情報が表示されます。次のように指定すると、出力をクライアント・ユーザ名または認証メカニズムの特定のセットに限定できます。

```
sp_helpmaplogin [ (authentication_mech | null),
(client_username | null) ]
```

たとえば、sp_helpmaplogin を発行すると次のような出力が返されます。

```
sp_helpmaplogin
authentication   client name   login name
--------------   -----------   ------------------
NULL                  jsmith              guest
PAM                   NULL                create login
```

新しいグローバル変数 @@authmech

Adaptive Server バージョン 12.5.2 には、新しいグローバル変数 *@@authmech* が含まれています。これは読み込み専用のグローバル変数であり、ユーザの認証に使用するメカニズムを設定します。

たとえば、Adaptive Server でフェールオーバ対応の PAM ユーザ認証が有効になっており (`enable pam user auth = 1`)、Joe が ANY 認証の外部ユーザであり、LDAP ユーザ認証が無効になっているとします。Joe がログインすると、Adaptive Server は PAM ユーザ認証で Joe を認証しようとします。

Joe の PAM でのユーザ認証が失敗すると、Adaptive Server は ASE 認証 (syslogins) を使用して Joe を認証します。これが成功するとログインできます。

@@authmech グローバル変数の現在の値は次のとおりです。

```
1> select @@authmech
2> go

---------------------------------
ase
```

次に、Adaptive Server が必ず PAM ユーザ認証を使用するように設定されており (`enable pam user auth = 2`)、Joe が有効なユーザとして PAM に追加されているとします。この場合、Joe がログインするときの *@@authmech* の値は次のとおりです。

```
1> select @@authmech
2> go

---------------------------------
pam
```

システム・テーブルの変更

syslogins の status カラム (16 ビットの整数データ型) には、次のように新しいビットが割り当てられました。

表 7-2: syslogins テーブルの status カラム

10 進数	16 進数	Status
1	0x1	6 文字以下のパスワードまたは null。
2	0x2	アカウントはロックされている。
4	0x4	パスワードの有効期限が切れている。
8	0x8	ユーザが RepSrv 権限を持っている。
16	0x10	OMNI：自動接続モードが有効になっている。
32	0x20	Adaptive Server 内部認証メカニズム (syslogins) を使用する。
64	0x40	LDAP 外部認証のみを使用する。
128	0x80	PAM 外部認証のみを使用する。

アクセス制御

この項では、Adaptive Server バージョン 12.5.2 でのアクセス制御の変更点について説明します。

set proxy の細分性の向上

以前のバージョンの Adaptive Server では、set proxy を使用してサーバ・ユーザ ID を他のサーバ・ログインに切り替えることはできましたが、ターゲット・ログインの役割に基づいて set proxy の使用を制限することはできませんでした。set proxy を付与されたユーザは、他の任意のサーバ・ユーザになることが可能でした。

Adaptive Server バージョン 12.5.2 では、set proxy...restricted role を付与することによって、ID を切り替えたときに特定の役割を取得できないように制限できます。

set proxy の詳細については、「第 12 章 グローバル変数、コマンド、ストアド・プロシージャの変更」を参照してください。

管理コマンドの付与と取り消し

Adaptive Server バージョン 12.5.2 では、update statistics、delete statistics、truncate table の各コマンドに対する、ユーザ、役割、グループのパーミッションを付与または取り消すことができます。テーブル所有者も、暗黙の grant によってパーミッションを付与できます。具体的には、update statistics、delete statistics、truncate table をストアド・プロシージャに追加してから、そのストアド・プロシージャの実行パーミッションをユーザまたは役割に付与します。

詳細については、「第 12 章 グローバル変数、コマンド、ストアド・プロシージャの変更」を参照してください。

システム・カタログのパーミッションの制限

Adaptive Server バージョン 12.5.2 では、grant コマンドと revoke コマンドに default permissions パラメータが追加されています。このパラメータによって、特定のシステム・テーブルに対してデフォルト・パーミッションを付与または取り消すことができます。

詳細については、「第 12 章 グローバル変数、コマンド、ストアド・プロシージャの変更」を参照してください。

責任

audit_event_name は、監査イベントの説明を返す新しい関数です。

詳細については、「第 12 章 グローバル変数、コマンド、ストアド・プロシージャの変更」を参照してください。

暗号化

Adaptive Server では、クライアントとの通信を保護するために暗号化技術が使用されています。バージョン 12.5.2 以降の Adaptive Server では、FIPS 認定の SSL 暗号化アルゴリズムが使用されます。Adaptive Server バージョン 12.5.2 では、パスワードを使用してバックアップを保護することもできます。

FIPS 認定の SSL 暗号化アルゴリズム

SSL は、クレジット・カード番号、株式売買、銀行取引などの機密情報を、インターネット上で安全に転送するための規格です。パブリック・キー暗号法に依存します。

現在 SSL 実装では、FIPS 140-2 認定アルゴリズムが使用されています。これらのアルゴリズムは次のもので利用可能になっています。

- Windows 対応の FIPS-140-2 認定 Crypto モジュール (Certicom の SB/GSE － NIST 証明書 #316、検証日 2003 年 5 月 13 日および 2003 年 6 月 30 日)
- Certicom の SB Crypto-C による、Solaris 32 ビット版および 64 ビット版プラットフォーム対応の FIPS 認定アルゴリズム

パスワードで保護されたバックアップ

Adaptive Server 12.5.2 の dump と load database には、データベース・ダンプをパスワードで保護するための password パラメータが含まれています。

dump database コマンドの password パラメータを使用すると、データベース・ダンプを不正なロードから保護できます。データベース・ダンプの作成時に password パラメータを指定した場合には、データベースのロード時にもこのパスワードを指定する必要があります。

詳細については、「第 9 章 パスワードで保護されたデータベース・ダンプの作成」を参照してください。

第 8 章

圧縮データベース・ダンプの作成

Adaptive Server バージョン 12.5.2 の dump には、データベース・ダンプを圧縮するための compression パラメータが含まれています。

データベース・ダンプの圧縮

dump コマンドの compression パラメータを使用すると、アーカイブされたデータベースに必要な領域を減らすことができます。compression API を使用している以前のバージョンの dump database では、データベース・ダンプはローカル・ファイルにしか圧縮できませんでした。Adaptive Server 12.5.2 で compression パラメータを使用すると、リモート・マシンにダンプを圧縮できます。

データベース・ダンプをロードするときに圧縮レベルを指定する必要はありません。ただし、load with listonly=full を発行すると、ダンプが作成されたときの圧縮レベルを判別できます。

dump database の構文の一部は次のとおりです。

```
dump database database_name to file_name [ with compression = compress_level]]
```

各パラメータの意味は、次のとおりです。

- *database_name* − データのコピー元であるデータベースの名前です。データベース名は、リテラル、ローカル変数、またはストアド・プロシージャのパラメータとして指定できます。
- *file_name* − ダンプ・ファイルの名前です。名前は 17 字までで、オペレーティング・システムのファイル命名規則に従っていなければなりません。
- *compress_level* − 1 ～ 9 の数値です。9 が最高の圧縮レベルです。圧縮レベルにはデフォルトはありません。*compress_level* を指定しないとダンプは圧縮されません。

たとえば、次の例は pubs2 データベースを圧縮レベル 4 で “remotemachine” という名前のリモート・マシンに圧縮します。

```
dump database pubs2 to "/Syb_backup/mydb.db" at remotemachine
with compression = "4"
```

第 9 章

パスワードで保護されたデータベース・ダンプの作成

Adaptive Server 12.5.2 の dump と load database には、データベース・ダンプをパスワードで保護するための password パラメータが含まれています。

パスワード保護を使用したデータベースのダンプとロード

dump database コマンドの password パラメータを使用すると、データベース・ダンプを不正なロードから保護できます。データベース・ダンプの作成時に password パラメータを指定した場合には、データベースのロード時にもこのパスワードを指定する必要があります。

パスワード保護に対応する dump database コマンドと load database コマンドの構文の一部は次のとおりです。

```
dump database database_name to file_name [ with passwd = password ]
load database database_name from file_name [ with passwd = password ]
```

各パラメータの意味は、次のとおりです。

- *database_name* − ダンプまたはロードするデータベースの名前です。
- *file_name* − ダンプ・ファイルの名前です。
- *password* − 不正なユーザからダンプ・ファイルを保護するために指定するパスワードです。

パスワードは 6 ～ 30 文字以内で指定します。5 文字以下または 31 文字以上のパスワードを指定すると Adaptive Server によってエラー・メッセージが発行されます。データベースをロードするときに誤ったパスワードを発行すると、Adaptive Server からエラー・メッセージが発行され、コマンドは失敗します。

たとえば、次の例はパスワード “bluesky” を使用して pubs2 データベースのデータベース・ダンプを保護します。

```
dump database pubs2 to "/Syb_backup/mydb.db" with passwd = "bluesky"
```

このデータベース・ダンプをロードするときには同じパスワードを使用する必要があります。

```
load database pubs2 from "/Syb_backup/mydb.db" with passwd = "bluesky"
```

パスワードと以前のバージョンの Adaptive Server

パスワード保護に対応する dump コマンドと load コマンドを使用できるのは、Adaptive Server バージョン 12.5.2 以降のみです。Adaptive Server バージョン 12.5.2 のダンプに password パラメータを使用した場合、そのダンプを以前のバージョンの Adaptive Server にロードしようとすると失敗します。

パスワードと文字セット

ダンプをロードできるサーバは同じ文字セットを使用しているサーバのみです。たとえば、ASCII 文字セットを使用するサーバから ASCII 以外の文字セットを使用するサーバにダンプをロードしようとすると、ASCII のパスワードの値は ASCII ではないパスワードと異なるためロードが失敗します。

ユーザが入力したパスワードは、Adaptive Server のローカル文字セットに変換されます。ASCII 文字は通常は文字セット間で値の表現が同じであるため、ユーザのパスワードが ASCII 文字セットであれば、dump と load のパスワードはすべての文字セットで認識されます。

第 10 章

Veritas 2.1 でフェールオーバを使用できるように Linux Adaptive Server を設定する

この章では、Veritas Cluster Server (VCS) バージョン 2.1 上の Linux Adaptive Server をフェールオーバ用に設定する方法について説明します。

Veritas のユーザ・マニュアルを読み、Veritas のクラスタについてよく理解してから、この章の手順を実行してください。

ハードウェアとオペレーティング・システムの稼働条件

高可用性を実現するには、次のハードウェアとシステム・コンポーネントが必要です。

- CPU やメモリなどのリソースに関して同様に設定されている、2 台の同種のネットワーク・システム。Adaptive Server は、RedHat Enterprise Linux 2.1 Advanced Server と VCS バージョン 2.1 での VCS をサポートしています。

  設定と管理を容易にする VCS GUI (グラフィカル・ユーザ・インタフェース) もインストールします。
- 2 つのシステムが、高可用性の設定が行われている Adaptive Server のデータベースを格納する共有マルチホスト・ディスクにアクセスできるようにします。
- ディスクの管理と DiskGroup や Volume などのリソースの作成を行うために Veritas Volume Manager 3.2 をインストールします。
- メディア障害に対処するためのサードパーティ・ベンダのミラーリングを使用します。

- サービス・グループをシステムごとに作成します。サービス・グループは固有のサービスを提供する一連のリソースです。高可用性の設定が行われている Adaptive Server のサービスを提供するために、サービス・グループには Adaptive Server 用の DiskGroup、Volume、Mount、IP、NIC などのリソースが含まれている必要があります。サービス・グループのサンプルとリソース依存を図 10-1 (71 ページ) に示します。サービス・グループの作成方法とリソースの追加方法の詳細については、『Veritas Cluster Server User's Guide』を参照してください。

  注意 各サービス・グループは少なくとも 2 つのリソースを含み、そのうちの 1 つが VCS 2.1 のリソース・タイプ *HAase* であることが必要です。このリソース・タイプが他のリソースに依存するように、クラスタ・コマンドを使用してリソース依存を確立します。

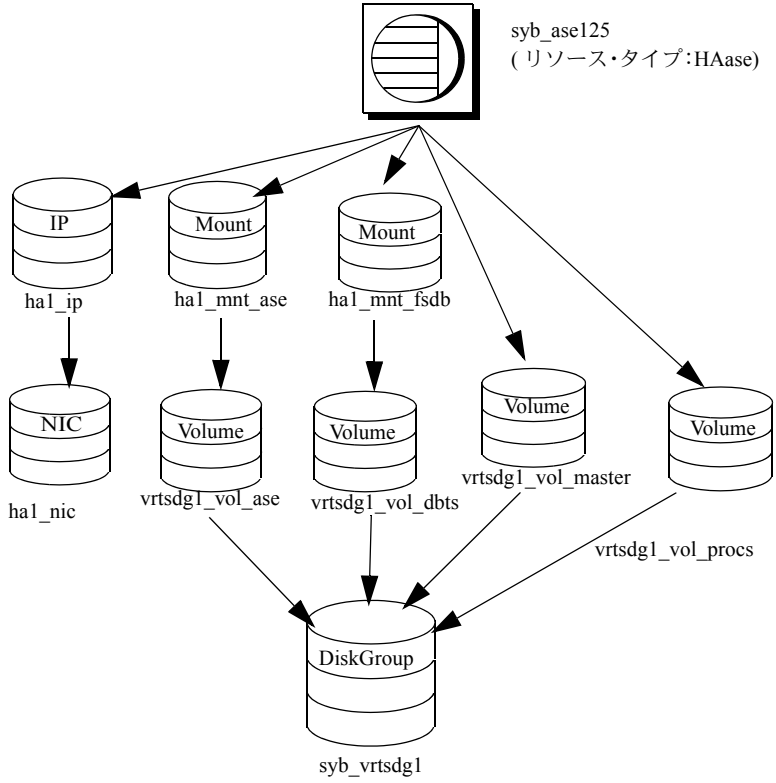
- 両方のノード上に、パブリックとプライベートの両ネットワークを設定します。

プラットフォームに固有の高可用性ソフトウェアのインストールについては、ご使用のハードウェアとソフトウェアのマニュアルを参照してください。

図 10-1 のサービス・グループの設定では、1 つの DiskGroup である *syb_vrtsdg1* 上に 4 つのボリュームが作成されています。ボリュームは、Adaptive Server インストール用に 1 つ、ファイル・システム上に作成されるデータベース用に 1 つと、後の 2 つはロー・デバイスに作成されるデータベース用です。2 つのマウント・リソースは、ボリューム・リソースの上のレイヤにあるタイプ vxfs のファイル・システム (Veritas ファイル・システム) 用に作成されています。タイプ *HAase* のリソース *syb_ase125* は、Adaptive Server のインストール環境であり、マウント・リソースの上に位置します。また、*syb_ase125* はリソース IP も必要とし、これはさらにパブリック・ネットワーク・アクセス用のリソース NIC を必要とします。

サービス・グループ *SybASE* はプライマリ・ノードで動作し、もう 1 つのサービス・グループ *SybASE2* (図 10-1 には含まれていない) は同様の設定でセカンダリ・ノードで動作します。

図 10-1: Veritas Cluster Server で動作するサービス・グループのサンプル



高可用性サブシステムで動作するように Adaptive Server を準備する

この項では、Adaptive Server を高可用性システムで動作させるための作業について説明します。

Adaptive Server のインストール

プライマリ・サーバとセカンダリ・サーバの両方をインストールします。共有ディスクまたはローカル・ディスクのどちらにでもインストールできます。プライマリ・コンパニオンは、新しくインストールした Adaptive Server でも、旧バージョンの Adaptive Server からアップグレードして既存のデータベースやユーザなどを受け継いだものでもかまいません。セカンダリ・コンパニオンは、新しくインストールした Adaptive Server である必要があり、すべてのユーザ・ログインやデータベース名がクラスタ内でユニークであることを確保するため、ユーザ・ログインまたはユーザ・データベースを持つことができません。フェールオーバ用の設定を完了したら、セカンダリ・コンパニオンにユーザ・ログインやデータベースを追加できます。

ローカル・ディスクにインストールする場合は、すべてのデータベース・デバイスがマルチホスト・ディスク上に作成されていることを確認してください。

Adaptive Server のインストールと設定については、使用しているプラットフォームの『インストール・ガイド』を参照してください。

両方の Adaptive Server のエントリを *interfaces* ファイルに追加する

プライマリ・コンパニオンとセカンダリ・コンパニオンの *interfaces* ファイルには、この両方のコンパニオンのエントリが必要です。たとえば、この章の例で使用されているサーバの *interfaces* ファイルには、MONEY1 と PERSONNEL1 の両方のエントリがあります。*interfaces* ファイル内のサーバ・エントリには、sysservers に指定されているネットワーク名を使用してください。*interfaces* ファイルへのエントリの追加については、使用しているプラットフォームの『インストール・ガイド』を参照してください。

フェールオーバ中にクライアント接続を行うために *interfaces* ファイルにエントリを追加する

フェールオーバしたコンパニオンにクライアントが再接続できるようにするには、*interfaces* ファイルに行を追加します。デフォルトでは、クライアントはサーバ・エントリの query 行にリストされているポートに接続します。サーバのフェールオーバが原因でこのポートが使用できない場合は、クライアントはサーバ・エントリの *hafailover* 行で指定されているサーバに接続します。次に示すのは、MONEY1 という名前のプライマリ・コンパニオン用と PERSONNEL1 という名前のセカンダリ・コンパニオン用の *interfaces* ファイルの例です。

```
MONEY1
master tcp ether MONEY 9678
query tcp ether MONEY 9678
hafailover PERSONNEL1

PERSONNEL1
master tcp ether PERSONNEL 9679
query tcp ether PERSONNEL 9679
```

interfaces ファイルにエントリを追加するには、dsedit を使用します。interfaces エントリがすでに存在する場合は、フェールオーバに使用できるように変更します。

dsedit については、『ASE ユーティリティ・ガイド』を参照してください。

sybha 実行プログラム

sybha 実行プログラムによって、Adaptive Server High Availability Basis Services ライブラリは、各プラットフォームの高可用性クラスタ・サブシステムと対話できるようになります。Adaptive Server High Availability Basis Services ライブラリは、*$SYBASE/ASE-12_5/bin* にある *sybha* を呼び出します。*sybha* は、所有権とパーミッションを変更することにより実行可能になります。*$SYBASE/ASE-12_5/install* の *sybhauser* という名前のファイルも編集する必要があります。*sybhauser* には、そのクラスタに対してシステム管理者権限を持つユーザのリストがあります。クラスタに対するシステム管理者権限を持つユーザの数を制限することを強くおすすめします。

root 権限で、次の作業を行います。

1　*sybhagrp* という新しいグループを追加します。このグループを */etc/group* ファイルに追加することも、または NIS マップに追加することもできます。このグループに sybase ユーザを追加します (これは *$SYBASE* ディレクトリを所有するユーザです)。サーバの起動時に sybase ユーザがデータ・サーバを実行します。複数のサーバを実行し、各サーバの *$SYBASE* ディレクトリを異なるユーザが所有する場合は、これらのユーザをすべてこのグループに追加してください。

2　*$SYBASE/$SYBASE_ASE/bin* ディレクトリに変更します。

```
cd $SYBASE/$SYBASE_ASE/bin
```

3 *sybha* の所有権を root に変更します。

```
chown root sybha
```

4 *sybha* プログラムのグループを *sybhagrp* に変更します。

```
chgrp sybhagrp sybha
```

5 *sybha* のファイル・パーミッションを 4550 に修正します。

```
chmod 4550 sybha
```

6 *$SYBASE/$SYBASE_ASE/install* ディレクトリに変更します。

```
cd $SYBASE/ASE-12_5/install
```

7 sybase ユーザを *sybhauser* ファイルに追加します。追加するユーザは、Adaptive Server のログインではなく、UNIX の形式のログイン ID である必要があります。次に例を示します。

```
sybase
coffeecup
spooner
venting
howe
```

8 *sybhauser* の所有権を root に変更します。

```
chown root sybhauser
```

9 *sybhauser* のファイル・パーミッションを修正します。

```
chmod 600 sybhauser
```

新しいデフォルト・デバイスの作成

デフォルトでは、新しくインストールされた Adaptive Server のデフォルト・デバイスは master です。つまり、データベース (フェールオーバによって使用されるプロキシ・データベースも含む) を作成すると、そのデータベースは自動的にマスタ・デバイス上に作成されます。ユーザ・データベースをマスタ・デバイスに追加すると、システム障害時に、マスタ・デバイスをリストアすることが困難になります。そのため、マスタ・デバイスには極力余分なユーザ・データベースを置かないようにします。それには、disk init コマンドを使用し、新しいデバイスを作成します。sp_diskdefault を使用して新しいデバイスをデフォルトに指定してから、Adaptive Server をフェールオーバ用のコンパニオンとして設定します。

たとえば、money_default1 という名前の新しいデフォルト・デバイスを MONEY1 Adaptive Server に追加するには、次のように入力します。

```
sp_diskdefault money1_default1, defaulton
```

特に次のようなコマンドを発行してデフォルトとしての設定を無効にしないかぎり、マスタ・デバイスも引き続きデフォルト・デバイスとして設定されています。

```
sp_diskdefault master, defaultoff
```

disk init と sp_diskdefault の詳細については、『Adaptive Server Enterprise リファレンス・マニュアル』を参照してください。

sysservers へのローカル・サーバの追加

sp_addserver を使用して、ローカル・サーバを sysservers に追加します。サーバには *interfaces* ファイルで指定したネットワーク名を付けます。たとえば、MONEY1 というコンパニオンが、*interfaces* ファイルで指定されている MONEY1 というネットワーク名を使う場合は、次のように入力します。

```
sp_addserver MONEY1, local, MONEY1
```

Adaptive Server をリブートして、変更内容を有効にしてください。

sysservers へのセカンダリ・コンパニオンの追加

セカンダリ・コンパニオンを sysservers にリモート・サーバとして追加します。

```
sp_addserver server_name
```

デフォルトでは、Adaptive Server は srvid の値が 1000 のサーバを追加します。この変更は Adaptive Server をリブートしなくても有効になります。

ha_role の割り当て

sp_companion を実行するには、両方の Adaptive Server で ha_role を持っていることが必要です。ha_role を割り当てるには、isql から次のコマンドを発行します。

```
sp_role "grant", ha_role, sa
```

Adaptive Server の変更を有効にするために、ログアウトしてからログインし直す必要があります。

HA ストアド・プロシージャのインストール

注意　両方のサーバを *interfaces* ファイルに追加してから、高可用性ストアド・プロシージャをインストールしてください。この作業を行わずに *installhasvss* を実行した場合、すべてのシステム・ストアド・プロシージャを再インストールしなければなりません。

installhasvss スクリプトにより、次の作業が行われます。

- フェールオーバに必要なストアド・プロシージャ (sp_companion など) のインストール
- *SYB_HACMP* サーバの sysservers へのインストール

installhasvss を実行するには、システム管理者権限が必要です。

installhasvss は *$SYBASE/ASE-12_5/scripts* にあります。*installhasvss* を実行するには、次のように入力します。

```
$SYBASE/$SYBASE_OCS/bin/isql -Usa -Ppassword -Sservername <
$SYBASE/ASE-12_5/scripts/installhasvss
```

installhasvss は、ストアド・プロシージャや SYB_HACMP サーバを作成するときにメッセージを表示します。

設定パラメータの確認

次の設定パラメータを有効にしてから、Adaptive Server をフェールオーバ用に設定します。

- enable CIS － コンポーネント統合サービス (CIS) を有効にします。この設定パラメータは、デフォルトで有効です。
- enable xact coordination － 分散トランザクション管理 (DTM) を有効にします。この設定パラメータは、デフォルトで有効です。
- enable HA － 高可用性システムで、Adaptive Server をコンパニオンとして機能させます。enable HA は、デフォルトでは無効です。Adaptive Server をリブートして、変更内容を有効にしてください。また、このパラメータを実行すると、高可用性システムで Adaptive Server を起動したというメッセージがエラー・ログに書き込まれます。Adaptive Server でフェールオーバを使用するには、ASE_HA ライセンス・オプションを購入する必要があります。ASE_HA ライセンスの有効化については、使用しているプラットフォームのインストール・ガイドを参照してください。

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

マスタ・ログへのスレッショルドの追加

フェールオーバ、フェールバック、プロキシ・データベースの作成などでは、ログが集中的に使用されます。適度なログ領域がないと、これらの処理が失敗する可能性があります。スレッショルドをマスタ・ログに追加していない場合は、追加してください。

1　ダンプ・トランザクションが発生する前に、master データベースのログに対して sp_thresholdaction を定義して実行し、残りのページ数にスレッショルドを設定します。Sybase では sp_thresholdaction を提供していません。このシステム・プロシージャの作成については、『ASE リファレンス・マニュアル』を参照してください。

2　それぞれのスレッショルドをマスタ・ログ・セグメントに置いて、セグメントが満杯にならないようにします。

```
sp_addthreshold "master", "logsegment", 250, sp_thresholdaction
```

3　プライマリ・コンパニオンをリブートして、この静的なパラメータを有効にします。

Veritas サブシステムを Sybase フェールオーバ用に設定する

この項は、高可用性サブシステムがすでにインストールされていることを前提としています。Veritas Cluster Server 高可用性サブシステムのインストールと使用については、『VCS User's Guide』と『VCS User Guide』を参照してください。

VCS バージョン 2.1 でのエージェントのインストール

次の手順に従って、エージェントをクラスタの各ノードにインストールします (次のコマンドを実行するには root パーミッションが必要です)。

1　*$SYBASE/$SYBASE_ASE/install/veritas/HAase* ディレクトリに変更します。

```
cd $SYBASE/$SYBASE_ASE/install/veritas/HAase
```

2　次のインストール・スクリプトを実行します。

```
perl installHAase.pl
```

インストール・スクリプトにより、次の作業が行われます。

- *HAase* リソース・タイプ・ファイル *HaaseTypes.cf* をローカル・システムの */etc/VRTSvcs/conf/config/* にコピーする。
- ディレクトリ */opt/VRTSvcs/bin/HAase* がまだ作成されていない場合は新しく作成する。

- ローカル・システムの */opt/VRTSvcs/bin/HAase/* に次のエージェント・バイナリとスクリプトをコピーする。
  - *HAaseAgent*
  - *online*
  - *offline*
  - *clean*
  - *sybhautil.pm*
  - *attr_changed*

VCS バージョン 2.1 でのリソース・タイプの設定

VCS バージョン 2.1 にリソース・タイプをインストールするには、次の手順にを実行します。

Adaptive Server ログイン・ファイルの作成

システム管理者とフォールト・モニタ用に追加したユーザの Adaptive Server ログイン情報が入っているファイルを作成します。この情報のテンプレートを含むサンプル・ファイルは、*$SYBASE/$SYBASE_ASE/install/veritas/HAase/ase_login_file* にあります。

このファイルは 2 行で構成されています。1 行目はシステム管理者のログインとパスワードで、2 行目はモニタのユーザ・ログインとパスワードです。

```
login-type<tab>login string
login-type<tab>login string
```

login-type と *login string* は `tab` 文字で区切ります。

注意 別のロケーションにある別のファイルを使用する場合は、*HAase* リソースの設定時に、リソース拡張プロパティ *Dataserver_login_file* にフル・パスを指定します。

---

login-type のデフォルト値は `normal` です。*login string* の値は、*login-name/password* という形式で指定します。次に例を示します。

```
normal     sa/sa-password
normal     probe-user/probe-password
```

セキュリティ上の理由から、read および write のアクセス・パーミッションを root に限定して、*ase_login_file* を安全に保護する必要があります。セキュリティを保持するには、次のコマンドを実行します。

```
chmod 400 ase_login_file
chown root ase_login_file
chgrp  sys  ase_login_file
```

注意 パスワードは空にしないことを強くおすすめします。空のパスワードを使用すると、エージェント・スクリプトが警告メッセージを生成します。

リソース・タイプのインポート

HAase リソース・タイプをインポートするには次の 2 つの方法があります。

- クラスタ GUI ツールを使用して新しいリソース・タイプをインポートする方法。詳細については、『VCS User Guide』を参照してください。
- コマンド・ラインでクラスタ・コマンド hatype と haattr を使用して、新しいリソース・タイプを手動でインポートする方法。詳細については、『VCS User Guide』を参照してください。

エージェントの起動

エージェントは次のいずれかの方法で起動できます。

- Veritas クラスタを再起動する方法。
- クラスタ・コマンドを使用して、手動でエージェントを起動する方法。

中断が起きないため、2 番目の方法を使用することをおすすめします。エージェントを手動で起動するには、次の手順に従います。

1　*HAase* エージェントのステータスを *haagent* ユーティリティで調べます。

```
#haagent -display HAase
#Agent   Attribute  Value
HAase     AgentFile
HAase     Faults     0
HAase     Running    No
HAase      Started      No
```

2　*haagent* ユーティリティを使用してホスト *myhost* で *HAase* エージェントを起動します。

```
# haagent -start HAase -sys myhost
VCS:10001:Please look for messages in the log file
```

3 *HAase* エージェントのステータスを *haagent* ユーティリティで調べます。

```
# haagent -display HAase
#Agent    Attribute   Value
HAase     AgentFile
HAase     Faults      0
HAase     Running     Yes
HAase      Started       Yes
```

リソースの追加

各サービス・グループに 1 つのリソースを含める必要があります。次の表は、リソースの属性を示します。

プロパティ	データ型、次元、デフォルト	説明
Sybase_home	string、スカラ、null	Adaptive Server インストール環境のホーム・ディレクトリ。Adaptive Server インストール環境の SYBASE 環境変数と同じ値です。
Dataserver_name	string、スカラ、null	設定時に提供される Adaptive Server の名前。
Backup_server_name	string、スカラ、null	設定時に提供される Backup Server の名前。
Monserver_name	string、スカラ、null	設定時に提供される Monitor Server の名前。
Textserver_name	string、スカラ、null	設定時に提供される全文検索サーバの名前。
Secondary_companion_name	string、スカラ、null	'sp_companion configure' コマンドの実行時に設定されるセカンダリ・コンパニオン・サーバの名前。
Dataserver_login_file	string、スカラ、null	現在のデータ・サーバのログイン情報が入っているファイルの絶対パス。このファイルは 2 行で構成されています。1 行目はシステム管理者のログインとパスワードで、2 行目は HA エージェント・モニタが完全プローブに使用するユーザ・ログインとパスワードです。
RUN_server_file	string、スカラ、null	代替 *RUN_server* ファイルの絶対パス。デフォルトの *$SYBASE/$SYBASE_ASE/install/RUN_SERVER* を上書きします。
Thorough_probe_cycle	int、スカラ、3	完全プローブが実行される前の「単純な」プローブの数。

プロパティ	データ型、次元、デフォルト	説明
Thorough_probe_script	string、スカラ、null	フォールト・モニタ・プログラムが完全プローブの実行に使用する SQL スクリプトが入っている代替ファイルの絶対パス。null に設定されている場合、エージェントはデフォルトの SQL コマンドを使用します。 セキュリティ上の理由から、このファイルの書き込みアクセスは *$SYBASE* ディレクトリの所有者のみに限定してください。 **注意**　この値は *HAase* リソースでは無視されます。
Debug	Boolean、スカラ、0	1 (真) に設定されている場合、モニタはデバッグ・メッセージを *$VCS_LOG/log/HAase_A.log* に記録し、他のスクリプトはデバッグ・メッセージを *$VCS_LOG/log/engine_A.log* に記録します。メッセージ番号の範囲は 2,000,001 以上です。
Log_max_size	int、スカラ、5000000	*$VCS_LOG/log/HAase_A.log* ファイルの最大サイズ。
Failback_strategy	string、スカラ、null	今後のために予約済み。
HA_config	boolean、スカラ、0	今後のために予約済み。
Cmpstate	boolean、スカラ、0	今後のために予約済み。

注意　*$VCS_LOG* のデフォルト値は */var/VRTSvcs* です。

次の表は、*HAase* インスタンスの設定例を示します。

属性	値
Sybase_home	*/release/rel125*
Dataserver_name	Money1
Backup_server_name	
Monserver_name	
Textserver_name	
Secondary_companion_name	
Dataserver_login_file	*/release/rel125/ASE-12_5/install/MONEY1_login*
RUN_server_file	*/release/rel125/ASE-12_5/install/RUN_MONEY1*
Thorough_probe_cycle	3
Thorough_probe_script	
Debug	

属性	値
Log_max_size	5000000
Failback_strategy	
HA_config	0
Cmpstate	0

各サービス・グループの *HAase* のインスタンスの設定

HAase のインスタンスは、次のいずれかの方法で設定できます。

- クラスタ GUI ツールを使用してリソース・タイプのインスタンスを設定する方法。詳細については、『VCS User Guide』を参照してください。
- クラスタ・コマンドを使用して新しいリソースを手動で追加し、その属性を設定する方法。この方法はこの後で説明します。

  次に示すのは、前述の表の設定を使用するリソース *syb_ase125* を追加するためのクラスタ・コマンドです (サービス・グループ *SybASE* の設定は、図 10-1 (71 ページ) にあります)。

  a リソース・タイプを追加します (この例は *HAase* リソース・タイプを使用しています)。

```
#hares -add syb_ase125 HAase SybASE
VCS:10245:Resource added
NameRule and Enabled attributes must be set before agent monitors
# hares -modify syb_ase125 Dataserver_name MONEY1
# hares -modify syb_ase125 RUN_server_file  /release/rel125/ASE-
12_5/install/RUN_MONEY1
# hares -modify syb_ase125 Log_max_size 5000000
# hares -modify syb_ase125 Dataserver_login_file /release/rel125/ASE-
12_5/install/MONEY1_login
# hares -modify syb_ase125 Sybase_home  /release/rel125
# hares -modify syb_ase125 Thorough_probe_cycle 3
```

  b リソース *syb_ase125* のステータスをモニタするようにエージェントを設定します。

```
# hares -modify syb_ase125 Enabled 1
```

注意 新しいリソースをサービス・グループに追加したら、リソース・タイプと、その他のストレージリソースやネットワーク・アクセス・リソースのアクセスの間にリソース依存を確立します。

次のクラスタ・コマンドを使用して、*syb_ase125* と、タイプが *Mount*、*Volume*、*IP* のリソースとの間にリソース依存を確立します。

```
# hares -link syb_ase125 ha1_mnt_ase
# hares -link syb_ase125 ha1_mnt_fsdb
# hares -link syb_ase125 vrtsdg1_vol_master
# hares -link syb_ase125 vrtsdg1_vol_procs
#hares -link syb_ase125 ha1_ip
```

詳細については、図 10-1 を参照してください。

フェールオーバ用コンパニオン・サーバの設定

この項では、高可用性システムで Adaptive Server をプライマリ・コンパニオンとセカンダリ・コンパニオンとして設定する手順を示します。

VCS バージョン 2.1 での HA モニタ用のユーザとログインの追加

HAase リソースと関連付けられた各データ・サーバに対して、モニタ用に特別なユーザとログインを作成します。isql を使ってデータ・サーバに接続し、次のコマンドを発行します。

```
sp_addlogin probe_ase, password

sp_adduser probe_ase
```

次に例を示します。

```
sp_addlogin joe, joe_password

sp_adduser joe
```

注意 システム管理者は、Adaptive Server の設定時に、プローブに使用するユーザとログインにより、ほかの目的に使用できる接続の合計数が実際には 1 つ減ることを考慮してください。

Adaptive Server に接続しているアプリケーションは、使用可能なすべてのユーザ接続を利用できるため、probe_user が Adaptive Server にログインできないことがあります。これが原因で、Adaptive Server でフェールオーバが発生することもあります。このような状況にならないように十分なユーザ接続数を設定してください。

モニタのログイン情報の格納の詳細については、「Adaptive Server ログイン・ファイルの作成」(78 ページ) を参照してください。

do_advisory を指定して *sp_companion* を実行する

十分なリソースを持つセカンダリ・コンパニオンを設定し、フェールオーバ中に両方のサーバの作業を実行できるようにします。セカンダリ・コンパニオンは、正常なクラスタ・オペレーションを妨げる属性を持っている場合があります。たとえば、プライマリ・コンパニオンとセカンダリ・コンパニオンがともにユーザ・ログイン数 250 用に設定されていると、フェールオーバ中、セカンダリ・コンパニオンには、発生する可能性のあるユーザ・ログインの半分を処理するリソースしかないことになります。したがって、MONEY1 と PERSONNEL1 は、どちらもユーザ・ログイン数 500 用に設定する必要があります。

sp_companion do_advisory オプションを使用すると、プライマリ・コンパニオンとセカンダリ・コンパニオンの両方の設定オプションを確認して、クラスタ・オペレーションを必ず正常に行うことができます。また、sp_companion do_advisory は、変更する必要のある設定オプションを通知します。

sp_companion do_advisory オプションの詳細については、『高可用性システムにおける Sybase フェールオーバの使用』の「第 6 章 do_advisory の実行」を参照してください。

HA エージェントの確認

Adaptive Server はさまざまなクラスタ・ソフトウェアをサポートできるため、sp_companion には、実行中の HA エージェントを問い合わせるための show_cluster オプションと、HA エージェントを設定するための set_cluster オプションが含まれています。

Veritas Cluster Server サブシステムを実行している場合は、sp_companion を使用してエージェントを指定してください。Adaptive Server は、別のエージェントが指定されないかぎり、Solaris プラットフォームで Sun Cluster ソフトウェアが実行されていると想定します。VCS バージョン 2.1 では、エージェントを *VCS-HAase* に設定します。

構文は次のとおりです。

```
sp_companion companion_server_name, [show_cluster]
sp_companion companion_server_name, [set_cluster ["VCS-HAase"]]
```

次の例では、Adaptive Server は Sun Cluster 2.2 用のデフォルトの HA エージェントを使用しています。

```
sp_companion MONEY1, show_cluster
The default cluster is: SC-2.2
The current cluster is set to default.
Supported cluster systems for SunOS:
SC-2.2
```

```
VCS-Sybase
SC-3.0
VCS-HAase
```

Veritas クラスタ用の *HAase* エージェントを使用するように Adaptive Server を変更するには、次のように指定します。

```
sp_companion MONEY1, set_cluster, "VCS-HAase"

The current cluster is set to VCS-HAase
```

これで、Adaptive Server は VCS サブシステム用の *HAase* エージェントを使用するようになります。

注意　Adaptive Server が VCS システムのノーマル・コンパニオン・モード用に設定された場合には、別の HA エージェント・タイプに変更しないでください。

非対称型設定の設定

2 つの Adaptive Server が非対称型設定に設定されます。セカンダリの Adaptive Server から、次のコマンドを発行します。

```
sp_companion "primary_server_name", configure, with_proxydb, login_name, password
```

上記のパラメータの意味は、次のとおりです。

- *primary_server_name* － *interfaces* ファイルのエントリと sysservers に定義されているプライマリ Adaptive Server の名前。
- *with_proxydb* － システム・データベース以外のすべてのデータベースに対して、プロキシ・データベースがセカンダリ・コンパニオン上に作成されることを示しています。その後、データベースが追加されると、プロキシ・データベースも作成されます。
- *login_name* － このクラスタ・オペレーションを行っているユーザの名前 (ha_role が必要)。
- *password* － このクラスタ・オペレーションを行っているユーザのパスワード。

次の例では、MONEY1 という名前の Adaptive Server をプライマリ・コンパニオンとして設定します (セカンダリ・コンパニオン PERSONNEL1 からコマンドを発行します)。

```
sp_companion "MONEY1", configure, null, "Think2Odd", "password"
サーバ 'PERSONNEL1' は有効で、クラスタ設定されています。
ステップ：サーバ 'PERSONNEL1' からサーバ 'MONEY1' へのアクセスが検証されました。
サーバ 'MONEY1' は有効で、クラスタ設定されています。
ステップ：サーバ 'MONEY1' からサーバ 'PERSONNEL1' へのアクセスが検証されました。
(1 row affected)
(1 row affected)
```

```
(1 row affected)
(1 row affected)
...
(1 row affected)
(1 row affected)
(1 row affected)
ステップ：コンパニオン・サーバの設定チェックが成功しました。
ステップ：サーバのハンドシェイクが成功しました。
ステップ：マスタ・デバイスはコンパニオンからアクセス可能です。
ステップ：クラスタ設定にサーバ 'PERSONNEL1' と 'MONEY1' が追加されました。
ステップ：サーバ設定の初期化が成功しました。
Step: Synchronizing Application Specific information from companion server
Step: Synchronizing Roles from companion server
Step: Synchronizing Login Roles from companion server
Step: Synchronizing Remote Logins from companion server
Step: Synchronizing Groups in sysusers from companion server
Step: Synchronizing Sysattributes from companion server
Step: Synchronizing server logins from companion server
Step: Synchronizing server-wide privs from companion server
Step: User information synchronization succeeded.
ステップ：サーバは、通常のコンパニオン・モードで設定されました。
```

sp_companion の使用時にユーザ・データベースがすでに存在している場合は、次のようなメッセージが表示されます。

```
Step: Created proxy database ‘pubs2’
 Step: Proxy status for database has been set. Please Checkpoint the database 'pubs2'
ステップ：サーバは、通常のコンパニオン・モードで設定されました。”
 Starting companion watch thread
```

非対称型設定の詳細については、『高可用性システムにおける Sybase フェールオーバの使用』の「第 3 章 非対称型と対称型の設定」を参照してください。

注意 前述の sp_companion configure コマンドの *login_name* と *password* を null にすることはできません。sp_companion configure が正常に実行されたら、オペレーティング・システムによって新しいファイル */etc/VRTSvcs/conf/config/ha_companion.remote_server_name* が作成されます。このファイルの読み込みおよび書き込みアクセスを、サーバを起動するユーザだけに限定してください。そうしない場合、セキュリティに問題が発生する可能性があります。

対称型設定の設定

非対称型フェールオーバを使用できるようにコンパニオンを設定した後、対称型設定に設定できます。対称型設定では、両方のサーバがプライマリ・コンパニオンとしても、セカンダリ・コンパニオンとしても機能します。

プライマリ・コンパニオンから sp_companion を発行して、対称型設定に設定します。非対称型設定の場合と同じ構文を使用します。sp_companion の構文については、「非対称型設定の設定」(85 ページ) を参照してください。

次は、MONEY1 という Adaptive Server を、セカンダリ・コンパニオンとして PERSONNEL1 という Adaptive Server に追加する例です (このコマンドはプライマリ・コンパニオン MONEY1 から発行します)。

```
sp_companion "PERSONNEL1", configure, with_proxydb, null, sa, Think2Odd
サーバ 'MONEY1' は有効で、クラスタ設定されています。
ステップ：サーバ 'MONEY1' からサーバ 'PERSONNEL1' へのアクセスが検証されました。
サーバ 'PERSONNEL1' は有効で、クラスタ設定されています。
ステップ：サーバ 'PERSONNEL1' からサーバ 'MONEY1' へのアクセスが検証されました。
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
.....
ステップ：コンパニオン・サーバの設定チェックが成功しました。
ステップ：サーバのハンドシェイクが成功しました。
ステップ：マスタ・デバイスはコンパニオンからアクセス可能です。
ステップ：クラスタ設定にサーバ "MONEY1" と "PERSONNEL1" が追加されました。
ステップ：サーバ設定の初期化が成功しました。
Step: Synchronizing server logins from companion server
Step: Synchronizing remoteserver from companion server
Step: Synchronizing roles from companion server
Step: Synchronizing server-wide privs from companion server
Step: User information syncup succeeded
ステップ：サーバは、通常のコンパニオン・モードで設定されました。
```

注意　前述の sp_companion configure コマンドの *login_name* と *password* を null にすることはできません。sp_companion configure が正常に実行されたら、オペレーティング・システムによって新しいファイル */etc/VRTSvcs/conf/config/ha_companion.remote_server_name* が作成されます。このファイルの読み込みおよび書き込みアクセスを、サーバを起動するユーザだけに限定してください。そうしない場合、セキュリティに問題が発生する可能性があります。

Sybase フェールオーバの管理

この項では、Sybase のフェールオーバの使い方を説明します。

フェールオーバ中

プライマリ・ノードがセカンダリ・ノードにフェールオーバすると、プライマリ・ノードでオンラインのサービス・グループがセカンダリ・ノードに切り替えられます。この時点で、Adaptive Server のバイナリを除くすべてのリソースが、セカンダリ・ノード上でオンラインになります。セカンダリ・ノードの Adaptive Server がこれらのリソースを引き継ぎます。

注意 あるサービス・グループがプライマリ・ホストからセカンダリ・ホストにフェールオーバすると、セカンダリ・ホストの Adaptive Server がそのグループのすべてのリソースを引き継ぎますが、フェールオーバしたグループの Adaptive Server は起動されません。

プライマリ・コンパニオンへのフェールバック

フェールバックでは、当初プライマリ・ノードに所属していたサービス・グループを、セカンダリ・ノードからプライマリ・ノードに戻して、そのグループをオンラインにします。

フェールバックを開始するには、次の手順に従います。

1 プライマリ・ノードがサービス・グループを引き継ぐ準備ができたら、セカンダリ・コンパニオンから次のコマンドを発行します。

```
sp_companion primary_companion_name,
prepare_failback
```

primary_companion_name は、プライマリ・コンパニオンの名前です。このコマンドは、プライマリ・ノードのサービス・グループを、セカンダリ・ノードからプライマリ・ノードに戻します。たとえば、プライマリ・コンパニオン MONEY1 をフェールバックするには、次のコマンドをセカンダリ・コンパニオン PERSONNEL1 から発行します。

```
sp_companion "MONEY1", prepare_failback
ステップ：プライマリ・データベースは、セカンダリで停止されました。
ステップ：プライマリ・データベースは現在のセカンダリから削除されました。
ステップ：プライマリ・デバイスは現在のセカンダリから解放されました。
ステップ：プライマリ・サーバの準備フェイルバックが正常に完了しました。
```

2　コマンド・ラインで次のコマンドを発行して、プライマリ・ノードのサービス・グループが正常にプライマリ・ノードに切り替えられたことを確認します。

```
hastatus -group service_group_name
```

このコマンドを使用すると、プライマリ・ノードのサービス・グループのステータスが表示されます。

3　ノーマル・コンパニオン・モードを再開するには、次をプライマリ・コンパニオンから発行します。

```
sp_companion secondary_companion_name, resume
```

secondary_companion_name は、セカンダリ・コンパニオン・サーバの名前です。たとえば、プライマリ・コンパニオン MONEY1 のノーマル・コンパニオン・モードを再開するには、次のように入力します。

```
sp_companion "PERSONNEL1", resume
```

注意　sp_companion resume を発行しないと、フェールオーバ・プロパティ (たとえば、isql -Q) が設定されたクライアントを Adaptive Server に接続できません。sp_companion prepare_failback を発行してからクライアントを再接続しようとすると、sp_companion resume を発行するまでクライアントはハングします。

ノーマル・コンパニオン・モードのサスペンド

サスペンド・モードでは、一時的にプライマリ・コンパニオンがセカンダリ・コンパニオンにフェールオーバできなくなります。ノーマル・コンパニオン・モードからサスペンド・モードに切り替えるには、次の手順に従います。

1　root 権限で、hares を使用して、プライマリ・ノードの *Sybase* リソースの属性 *Critical* を 0 に変更します。構文は次のとおりです。

```
hares -modify name_of_Sybase_resource Critical 0
```

たとえば、プライマリ・コンパニオン MONEY1 の *Sybase* リソース *syb_ase125* の属性 *Critical* を変更するには、次のコマンドを発行します。

```
hares -modify syb_ase125 Critical 0
```

(hares コマンドの詳細については、『Veritas Cluster Server User's Guide』を参照してください。)

2 ノーマル・コンパニオン・モードをサスペンドします。セカンダリ・コンパニオンから、以下を発行します。

```
sp_companion companion_name, suspend
```

たとえば、プライマリ・コンパニオン MONEY1 を保守のためにサスペンドするには、セカンダリ・コンパニオン PERSONEL1 に接続して、次を発行します。

```
sp_companion MONEY1, suspend
```

ノーマル・コンパニオン・モードの再開

サスペンド・モードからノーマル・コンパニオン・モードに戻るには、次の手順に従います。

1 両方のコンパニオンが実行中であることを確認します。root 権限で、次のコマンドを発行します。

```
hastatus
```

2 プライマリ・ノードの *Sybase* リソースの *Critical* 属性を 1 に変更します。root 権限で、次のコマンドを発行します。

```
hares -modify name_of_Sybase_resource Critical 1
```

たとえば、プライマリ・コンパニオン MONEY1 の *Sybase syb_ase125* リソースの *Critical* 属性を変更するには、次のコマンドを発行します。

```
hares -modify syb_ase125 Critical 1
```

3 ノーマル・コンパニオン・モードを再開します。セカンダリ・コンパニオンから、以下を発行します。

```
sp_companion primary_companion_name, resume
```

たとえば、プライマリ・コンパニオン MONEY1 のノーマル・コンパニオン・モードを再開するには、以下を入力します。

```
sp_companion MONEY1, resume
```

注意 sp_companion resume を発行しないと、フェールオーバ・プロパティ (たとえば isql -Q) が設定されたクライアントを接続できません。sp_companion prepare_failback を発行してからクライアントを再接続しようとすると、クライアントは sp_companion resume を発行するまでハングします。

コンパニオン・モードの削除

コンパニオン・モードを削除するには、以下を発行します。

```
sp_companion companion_name, "drop"
```

コンパニオン・モードを削除したら、二度と元に戻すことはできません。したがって、削除する場合は、高可用性システムをフェールオーバさせる前にコンパニオン・サーバを再設定し、Sybase のフェールオーバに装備されている機能をすべて保持する必要があります。しかし、その後も、コンパニオン・サーバは引き続き HA エージェントによってモニタされます。最初に Adaptive Server をモニタするエージェントを無効にしてから、コンパニオン・モードを削除してください。次のコマンドを発行します。

```
hares -modify Sybase_resource_name Enabled 0
```

コンパニオン・モードを削除するには、sp_companion ... drop を発行します。

たとえば、プライマリ・コンパニオン MONEY1 とのコンパニオン関係を削除するには、セカンダリ・コンパニオン PERSONNEL1 に接続して、次のコマンドを発行します。

```
sp_companion "MONEY1", "drop"
```

Veritas クラスタでのフェールオーバのトラブルシューティング

この項では、一般的なエラーに関するトラブルシューティング情報について説明します。

- Adaptive Server のデバッグをオンにします。トレース・フラグ 2205 を使用して、高可用性関連のデバッグ情報を取得します。次の isql セッションは、デバッグをオンにして、メッセージをコンソールにリダイレクトします。

  ```
  dbcc traceon(2205)
  dbcc traceon(3604)
  ```

- エラーが報告されたら、最初にエラー・ログをチェックします。VCS システム・ログ */var/VRTSvcs/log/engine_A.log* では、ID が 2,000,000 より大きいすべてのエラー・メッセージが *HAase* エージェントのエラー・メッセージです。

- VCS エラー・ログは */var/VRTSvcs/log/log_name.log* にあります。このうち、*engine_A.log* は重要な情報源です。

  システム・エラー・ログは、*/var/log/syslog* にあります。

- システムの情報を調べるには、次の監視ツールの使用をおすすめします。
  - *hagui* − GUI ツール。
  - *hastatus* − コマンド・ライン・ツール。
  - VCS システムでのイベントについて警告するトリガ・スクリプト (*injeopardy*、*preonline*、*postonline*、*postoffline*、*resnotoff*、*resfault*、*sysoffline*、*violation*)。
- あるサービス・グループがプライマリ・ホストからセカンダリ・ホストにフェールオーバすると、セカンダリ・ホストの Adaptive Server がそのグループのすべてのリソースを引き継ぎますが、フェールオーバしたグループの Adaptive Server は起動されず、セカンダリ・ホストの HAase リソースに「障害が発生している」ことが VCS によって示される可能性があります。セカンダリ・ホストで次のコマンドを使用して、フェールオーバ後にステータスをクリアしてください。

  ```
  hares -clear sybase_res_name -sys
  secondary_host_name
  ```

失敗した *prepare_failback* からのリカバリ

フェールバック中に、prepare_failback がセカンダリ・コンパニオンで正常に実行されたにもかかわらず、プライマリ・コンパニオンがブートしない場合は、次の手順に従います。

1 プライマリ・コンパニオンのエラー・ログとクラスタ・エラー・ログをチェックし、サーバが起動しなかった原因を特定し、問題を解決します。

2 リソース・タイプの FAULTED ステータスをクリアするには、次のコマンドを発行します。

```
hares -clear HAase_res_name
```

3 root 権限で以下を発行し、プライマリの論理ホストをセカンダリ・ノードに戻します。

```
hagrp -switch primary_service_group -to
secondary_host_name
```

4 セカンダリ・コンパニオンにログインして、次のコマンドを発行します。

```
dbcc ha_admin ("", "rollback_failback")
```

両方のコンパニオン・サーバが、フェールオーバ・モードに戻ります。dbcc ha_admin の詳細については、『高可用性システムにおける Sybase フェールオーバの使用』の「高可用性システムの dbcc オプション」(342 ページ) を参照してください。

5 セカンダリ・コンパニオン上で sp_companion...prepare_failback を再発行します。

ログのロケーション

これらのログの情報を使用して、高可用性サブシステムをデバッグします。

- Adaptive Server エラー・ログ (RUNSERVER ファイルで定義)
- Veritas クラスタ・ログ (*/var/VRTSvcs/log/engine_A.log*)
- オペレーティング・システム・メッセージ (*/var/log/syslog*)
- VCS バージョン 2.1 の *HAase* エージェント・ログ (*/var/VRTSvcs/log/HAase_A.log*)

第 11 章

32 ビット版 Linux でのラージ・メモリ・サポート

概要

32 ビット版 Enterprise Linux オペレーティング・システムでの Adaptive Server のラージ・メモリ・サポートによって、Adaptive Server で使用できるメモリ容量が 2.7GB から最大 64GB に増えました。Adaptive Server で使用できるメモリ容量の増加により、サーバがディスクにアクセスする回数が大幅に減少し、パフォーマンスが向上します。

Adaptive Server のラージ・メモリ・サポートは、Linux 32 ビット版オペレーティング・システムの共有メモリ・ファイル・システム (*shmfs*) とメモリ・マップ・ファイル (*mmap*) 機能を利用しています。ラージ・メモリ・サポートを有効にすると、Adaptive Server 設定ファイルに指定したサイズの *shmfs* ファイルが Adaptive Server によって作成されます。このファイルは、Red Hat Advanced Server Linux 2.1 (AS 2.1) オペレーティング・システムでは最大 16GB、Red Hat Enterprise Linux 3.0 (RHEL 3) オペレーティング・システムでは最大 64GB にできます。

Adaptive Server は、メモリ・マップ・ファイルの領域をセカンダリ・データ・キャッシュとして使用します。プライマリ・データ・キャッシュのサイズは最大 2.7GB しかありません。これは、Linux でアドレス指定可能な最大メモリから必要なオーバヘッドを除いたサイズです。

Adaptive Server は、新しいページを常にプライマリ・データ・キャッシュに配置します。プライマリ・キャッシュが満杯になると、Adaptive Server はページをセカンダリ・キャッシュに再配置して新しいページのための領域を確保します。Adaptive Server が特定のページを検索するときは、まずプライマリ・データ・キャッシュを探し、次にセカンダリ・データ・キャッシュを探します。ページがセカンダリ・キャッシュで検出されると、Adaptive Server は仮想アドレス・ウィンドウを *shmfs* のその部分にマップして、読み取り可能なプライマリ・キャッシュにページをコピーします。

セカンダリ・キャッシュには次のような特徴があります。

- グローバルである。つまり、すべての名前付きプライマリ・キャッシュで共有される。
- サーバ・ページ・サイズと同じサイズのページだけを置換できる。つまり、サーバ・ページ・サイズが 4K で、ページ・サイズがそれぞれ 4K と 16K の 2 つのプールがある場合、セカンダリ・キャッシュは 4K のプールからのページのみを保持する。
- ウォッシングとチェックポイントのみの書き込みをサポートする。
- ウォッシュ・サイズは 20% に固定。プライマリ・キャッシュのプール用のウォッシュ・サイズは考慮されない。ウォッシュはセカンダリ・キャッシュでのみ行われる。
- インデックスの周回と OAM の周回のチューニングはサポートしていない。
- “log only” または “relaxed lru” とマークされたキャッシュのページは保持しない。これは、ログ・ページは通常はディスクから読み取らないことと、キャッシュが “relaxed lru” とマークされるのはオブジェクトがプライマリ・キャッシュにすべてキャッシュされたときのみであることから、セカンダリ・キャッシュを使用してもメリットがないためである。

ラージ・メモリ・サポートの設定

ラージ・メモリ・サポートを設定するには、次の手順を実行します。

1 オペレーティング・システムで *shmfs* を設定します。

2 Adaptive Server にセカンダリ・データ・キャッシュを設定します。

オペレーティング・システムの設定

オペレーティング・システムのコマンドを使用して、サーバに *shmfs* を設定してサイズを指定します。

たとえば、Linux AS 2.1 または Linux RHEL 3 に 8GB の *shmfs* を設定するには、次のように入力します。

```
mount -t shm shmfs -o size=8g /dev/shm
```

注意 Sybase ユーザが */dev/shm* に対する read と write のパーミッションを持っていることを確認してください。

詳細については、Linux オペレーティング・システムのマニュアルを参照してください。

Adaptive Server の設定

Adaptive Server にセカンダリ・キャッシュを設定するには、extended cache size 設定パラメータに値を入力します。extended cache size の値が 0 (デフォルト) の場合、Adaptive Server はセカンダリ・キャッシュを使用しません。

セカンダリ・キャッシュを有効にするには、セカンダリ・キャッシュのサイズを 2K ページ単位で指定します。たとえば、sp_configure を使用して 1048576 × 2K ページ (約 2GB) のセカンダリ・キャッシュを作成するには、次のように入力します。

```
sp_configure "extended cache size", 1048576
```

セカンダリ・キャッシュを有効にすると、Adaptive Server によってファイル */dev/shm/<server_name><pid_number>.shm* が作成されます。

セカンダリ・データ・キャッシュが有効な場合、Adaptive Server の動作は次のようになります。

- Adaptive Server が停止したとき、または extended cache size の値が 0 にリセットされたときは、必ずセカンダリ・キャッシュが削除される。
- Adaptive Server が起動するときは必ずセカンダリ・キャッシュが作成される。
- デフォルト値は 0。extended data cache の値が 0 の場合、Adaptive Server はセカンダリ・データ・キャッシュを使用しない。

セカンダリ・データ・キャッシュのサイズの変更

セカンダリ・データ・キャッシュのサイズは変更できます。

セカンダリ・キャッシュのサイズを増やす

セカンダリ・データ・キャッシュのサイズを *shmfs* で宣言したサイズに増やすには、sp_configure を使用します。これは動的に変更されるため、Adaptive Server を再起動する必要はありません。たとえば、キャッシュ・サイズを 8GB に増やすには、次のように入力します。

```
sp_configure "extended cache size", 4194304
```

セカンダリ・キャッシュのサイズを、オペレーティング・システムで設定された *shmfs* のサイズよりも大きくするには、次の手順を実行します。

1 セカンダリ・キャッシュを無効にします。次のように入力します。

```
sp_configure "extended data cache", 0
```

2 */dev/shm* のマウントを解除します。

3 *shmfs* を目的のサイズでマウントします。

詳細については、「オペレーティング・システムの設定」(97 ページ) を参照してください。

4 目的のサイズでセカンダリ・データ・キャッシュを有効にします。

詳細については、「Adaptive Server の設定」(97 ページ) を参照してください。

セカンダリ・キャッシュのサイズを減らす

セカンダリ・キャッシュのサイズを減らすには 2 つの方法があります。Adaptive Server のリブートが必要な静的な方法と、リブートが必要ない動的な方法です。

セカンダリ・データ・キャッシュのサイズを静的に減らすには、次の手順を実行します。

1 sp_configure を使用して、セカンダリ・データ・キャッシュのサイズを目的のサイズに再設定します。たとえば、キャッシュ・サイズを 2GB に増やすには、次のように入力します。

```
sp_configure "extended cache size", 1048576
```

2 Adaptive Server をリブートします。

セカンダリ・データ・キャッシュのサイズを動的に減らすには、次の手順を実行します。

1 現在のセカンダリ・キャッシュを無効にします。次のように入力します。

```
sp_configure "extended data cache", 0
```

2 目的のサイズのセカンダリ・キャッシュを再作成します。次に例を示します。

```
sp_configure "extended cache size", 1048576
```

システム・ストアド・プロシージャの変更

ユーザ・インタフェースは変わっていませんが、ラージ・メモリ・サポートのために次のシステム・ストアド・プロシージャが変更されました。

- sp_configure
- sp_helpconfig
- sp_sysmon

sp_configure の変更

sp_configure で設定できる Adaptive Server パラメータのリストに extended cache size パラメータが追加されました。extended cache size を使用して、セカンダリ・キャッシュのサイズを指定します。extended cache size を 0 (デフォルト値) に設定すると、セカンダリ・キャッシュが無効になります。

例については、「Adaptive Server の設定」(97 ページ) を参照してください。

sp_helpconfig の変更

sp_helpconfig は、設定パラメータに関する情報を表示します。sp_helpconfig を使用して、セカンダリ・データ・キャッシュを有効にした場合のオーバヘッドを計算します。

たとえば、2GB (1048576 × 2K ページ) のセカンダリ・キャッシュのオーバヘッドを計算するには、次のように入力します。

```
sp_helpconfig "extended cache size", "1048576"
```

sp_sysmon の変更

sp_sysmon は、Adaptive Server のパフォーマンスに関する情報を表示します。sp_sysmon を使用して、セカンダリ・データ・キャッシュのパフォーマンスに関する情報を表示します。この情報は、Data Cache Management セクションに表示されます。

次に例を示します。

```
Cache Statistics Summary (All Caches)
-------------------------------------
                                  per sec    per xact    count    # of total
                                  -----------------------------------------
...


Secondary Cache Search Summary
       Total Cache Hits          152448.5     432.2    15286912      100.0 %
      Total Cache Misses            26.6      28.0        1597        0.0 %
--------------------------      --------    -------  ---------
    Total Cache Searches         152475.2     450.3    15288509
```

extended cache size 設定パラメータ

extended cache size 設定パラメータは、設定ファイルの Cache Manager セクションにあります。次に例を示します。

```
[Cache Manager]
    number of oam trips = DEFAULT
    number of index trips = DEFAULT
    memory alignment boundary = DEFAULT
    global async prefetch limit = DEFAULT
    global cache partition number = 4
    extended cache size = 4194304
```

キャッシュ・マネージャ

extended cache size

要約	
デフォルト値	0
値の範囲	0 ～ 共有メモリ・ファイル・システム (*shmfs*) のサイズ (2K ページ単位)
ステータス	動的 – キャッシュ・サイズを増やすとき 静的 – キャッシュ・サイズを減らすとき
表示レベル	包括
必要な役割	システム管理者

extended cache size パラメータは、32 ビット版 Enterprise Linux オペレーティング・システムのグローバルなセカンダリ・データ・キャッシュのサイズを指定します。セカンダリ・データ・キャッシュを指定すると、Adaptive Server で使用できるメモリの容量が増え、ディスクへのアクセス回数が減少してパフォーマンスが向上します。セカンダリ・データ・キャッシュがない場合、Linux で Adaptive Server に使用できるメモリは 2.7GB に制限されています。

セカンダリ・データ・キャッシュを作成するには、まずオペレーティング・システムのコマンドを使用して共有メモリ・ファイル・システム (*shmfs*) を作成します。AS 2.1 オペレーティング・システムでは最大 16GB、RHEL 3 オペレーティング・システムでは最大 64GB を指定できます。次の場合には、Adaptive Server によってセカンダリ・キャッシュ・ファイルが作成されます。

- extended cache size に 0 よりも大きな値を設定して、セカンダリ・キャッシュが有効になっている場合。
- Adaptive Server が起動し、extended cache size が 0 よりも大きい場合。

extended cache size の値が 0 の場合、Adaptive Server はセカンダリ・データ・キャッシュを使用したり作成することはありません。

第 12 章

グローバル変数、コマンド、ストアド・プロシージャの変更

この章では、Adaptive Server バージョン 12.5.2 のグローバル変数、関数、コマンド、ストアド・プロシージャで、新規のものおよび変更されたものについて説明します。

新しいグローバル変数

表 12-1: 新しいグローバル変数

グローバル変数	意味
@@monitors_active	sp_sysmon によって表示されるメッセージ数を減らす。
@@authmech	ユーザの認証に使用するメカニズムを指定する読み込み専用変数。

Real Time Messages Services オプションでもいくつかのグローバル変数が追加されています。詳細については、『Real Time Data Service ユーザーズ・ガイド』を参照してください。

新しい設定パラメータ

この項では、Adaptive Server バージョン 12.5.2 の新しい設定パラメータについて説明します。

Real Time Messages Services オプションでも設定パラメータが追加されています。詳細については、『Real Time Data Service ユーザーズ・ガイド』を参照してください。

histogram tuning factor

要約	
デフォルト値	1 (オフ)
値の範囲	1 ～ 100
ステータス	動的
表示レベル	中間
必要な役割	システム管理者

histogram tuning factor は、Adaptive Server が update statistics、update index statistics、update all statistics、create index について 1 つのヒストグラムで分析するステップ数を制御します。

次の例では、Adaptive Server は 30 個の値を含む 20 ステップの中間ヒストグラムを生成します。

```
sp_configure 'histogram tuning factor',20
update statistics tab using 30 values
```

Adaptive Server は、次の条件に従って、ヒストグラムを分析して結果のヒストグラムに圧縮します。

- 最初のステップは変更せずにコピーする。
- 頻度の高いステップは変更せずにコピーする。
- 連続した範囲のステップは 1 つの結果ステップにまとめる。このため、まとめられたステップの総ウェイトは値の 30 分の 1 を超えない。

sysstatistics 内の最終的なヒストグラムは次のようになります。

- 範囲ステップは 30 ステップの update statistics と同様に生成され、高頻度の範囲はヒストグラムが 600 ステップで作成された場合のように分離されます。
- 結果ヒストグラムの合計ステップ数は、30 ～ 600 の値になります。
- 均等に分散したデータの場合、値は 30 にかぎりなく近づきます。
- テーブルでの「頻度」が高い値は、ヒストグラムでのステップ数が多くなります。
- 1 つのカラムに異なる値がわずかしかない場合は、それらすべての値が高頻度セルとして表示されることがあります。

histogram tuning factor を使用してステップ数を 600 に増やしても同じ結果が得られますが、バッファやプロシージャ・キャッシュでより多くのリソースを使用することになります。

historgram tuning factor を使用すると、ヒストグラムで使用されるリソースを最小限に抑えられます。リソース使用量を増やすのは、最適化のために適切な場合のみです。たとえば、カラムのデータの分布に一貫性がない場合、または重複する値がカラムに多数存在する場合です。このような場合には、最大 600 のヒストグラム・ステップが使用されます。ただし、ほとんどの場合はデフォルト値 (上記の 30) が使用されます。

number of dump threads

要約	
デフォルト値	無効
値の範囲	1 (無効、並列なし) ～ 8 (完全に並列)
ステータス	動的
表示レベル	中間
必要な役割	システム管理者

number of dump threads は、Adaptive Server がメモリ・ダンプを実行するために生成するスレッド数を制御します。ダンプ・スレッド数を適切な値にすると、メモリ・ダンプ中にエンジンが停止する時間を短縮できます。

メモリ・ダンプのスレッド数を決めるときは次の点を考慮してください。

- マシンのファイル・システム・キャッシュにメモリ・ダンプ全体を格納するのに十分な空きメモリがある場合は値 8 を使用します。
- マシンに十分な空きメモリがあるかどうかわからない場合は、ダンプ・スレッド数の値は、I/O システムの速度、ディスクの速度、コントローラのキャッシュ、ダンプ・ファイルがいくつかのディスクで作成された論理ボリューム・マネージャに存在するかどうかなど、多くの要因によって決まります。
- メモリ・ダンプの実行中にエンジンを停止しない場合は、次に説明するように値 1 (並列処理なし) を使用します。

  Adaptive Server がメモリ・ダンプを実行するとき、作成されるファイル数は、割り当てたメモリ・セグメントの合計数と設定したスレッド数を掛けた値になります。Adaptive Server は、別々のスレッドを使用して別々のファイルに書き込みます。ジョブが完了すると、エンジンが再起動され、これらのファイルがターゲット・ダンプ・ファイルにマージされます。このため、共有メモリを並列でダンプするときにかかる時間は逐次処理よりも長くなります。
- メモリのダンプ中にエンジンを停止する場合は、1 以外の値を使用すると、メモリ・ダンプ時のエンジンの停止時間を短縮できます。

新しいソート順

Adaptive Server バージョン 12.5.2 には、次のソート順が追加されています。

表 12-2: 新しいソート順

言語またはスクリプト	文字セット	ソート順
チェコ語およびスロバキア語	cp852、iso88592、cp1250	dictionary、nocase、noaccents
西欧語	cp1252	dictionary、nocase、nocasepref、noaccents、espdict、espnocs、espnoac

関数、コマンド、ストアド・プロシージャの変更

次に、Adaptive Server バージョン 12.5.2 の関数、コマンド、ストアド・プロシージャの変更点を示します。

Real Time Messages Services オプションと Web Services オプションでもストアド・プロシージャが追加されています。詳細については、『Real Time Data Service ユーザーズ・ガイド』と『Web Services ユーザーズ・ガイド』を参照してください。

変更されたコマンド

union

union の片側に指定できるサブクエリの最大数が 16 から 50 に増えました。

dbcc checkcatalog

dbcc checkcatalog は、sysindexes の一貫性検査を実行するようになりました。エラーがある場合にエラーを修正するための fix パラメータが追加されました。構文の一部は次のとおりです。

```
dbcc checkcatalog [(database_name[, fix])
```

fix は dbcc が検出した sysindexes エラーを修正するかどうかを指定します。checkcatalog のデフォルト・モードではエラーは修正されません。fix オプションを使用するには、データベースをシングルユーザ・モードに切り替える必要があります。新しい sysindexes チェックでは、以前のバージョンのサーバの dbcc checkcatalog では発生しなかった新しいエラーが発生する可能性があります。

kill

Adaptive Server バージョン 12.5.2 では、kill コマンドに statusonly パラメータが追加されました。kill ...statusonly は、ロールバック・ステータスであるサーバ・プロセス ID (spid) の進捗状況についてレポートします。指定した spid は強制終了されません。statusonly レポートには、ロールバックの完了率と完了までにかかる推定時間 (秒単位) が表示されます。構文は次のとおりです。

kill *spid* with statusonly

spid は、停止するプロセスの番号です。

たとえば、次の例は spid 番号 13 のロールバック・プロセスについてレポートします。

```
kill 13 with statusonly
spid: 13 トランザクション・ロールバックが進行中です。推定ロールバック完了 :17% 推定残り時間 :13 秒
spid: 13 トランザクション・ロールバックが進行中です。推定ロールバック完了 :29% 推定残り時間 :9 秒
spid: 13 トランザクション・ロールバックが進行中です。推定ロールバック完了 :40% 推定残り時間 :8 秒
spid: 13 トランザクション・ロールバックが進行中です。推定ロールバック完了 :47% 推定残り時間 :7 秒
spid: 13 トランザクション・ロールバックが進行中です。推定ロールバック完了 :55% 推定残り時間 :6 秒
spid: 13 トランザクション・ロールバックが進行中です。推定ロールバック完了 :65% 推定残り時間 :5 秒
spid: 13 トランザクション・ロールバックが進行中です。推定ロールバック完了 :73% 推定残り時間 :4 秒
spid: 13 トランザクション・ロールバックが進行中です。推定ロールバック完了 :76% 推定残り時間 :3 秒
spid: 13 トランザクション・ロールバックが進行中です。推定ロールバック完了 :83% 推定残り時間 :2 秒
spid: 13 トランザクション・ロールバックが進行中です。推定ロールバック完了 :94% 推定残り時間 :0 秒
```

kill...statusonly を発行したときに、指定した spid のロールバックがすでに完了している場合、または指定した spid がロールバックされていない場合は、kill...statusonly から次のメッセージが返されます。

ステータス・レポートを取得できません。KILL spid:*nn* is not in progress.

select * の追加コメント

select * を使用するクエリの記述は、選択したカラムの記述を収容するように拡張されました。次に例を示します。

```
create procedure myproc as select * from authors
```

この例は、syscomments に次の内容を生成します。

```
 create procedure myproc as
--Adaptive Server は、次の文ですべての '*' 要素を拡張しています。
select authors.au_id, authors.au_lname, authors.au_fname, authors.phone, autho
rs.address, authors.city, authors.state, authors.country, authors.posta
lco
```

grant と *revoke* の変更

この項では、Adaptive Server バージョン 12.5.2 の grant コマンドと revoke コマンドの変更点について説明します。

grant および set proxy の fipsflagger に対する警告の発行

set fipsflagger オプションが有効になっているときに grant dbcc と set proxy を発行すると、次の警告が発行されます。

```
行番号 %1! の SQL 文に ANSI 以外のテキスト があります。DBCC を使用したために、エラーが発生しました。
```

システム・テーブルとストアド・プロシージャへのデフォルト・パーミッションの付与

Adaptive Server バージョン 12.5.2 の installmaster または installmodel では、一部のシステム・テーブル (下記) のデフォルト・パーミッションは付与されません。代わりに、Adaptive Server が新しいデータベースを構築するときに、これらのシステム・テーブルのデフォルト・パーミッションが割り当てられます。

Adaptive Server バージョン 12.5.2 では、grant コマンドと revoke コマンドに default permissions パラメータが追加されています。このパラメータによって、下記のシステム・テーブルのデフォルト・パーミッションを付与または取り消すことができます。構文の一部は次のとおりです。

```
grant default permissions on system tables
revoke default permissions on system tables
```

default permissions on system tables は、任意のデータベースからこのコマンドを発行するときに、次のシステム・テーブルのデフォルト・パーミッションの付与または取り消しを指定します。

- sysalternates
- syscolumns
- syscomments
- sysdepends
- sysindexes
- syskeys
- sysobjects
- sysprocedures
- sysprotects
- syssegments
- systypes
- sysusers

- syslogs
- sysconstraints
- sysreferences
- sysusermessages
- sysattributes
- systabstats
- sysxtypes
- sysjars
- systhresholds
- syspartitions
- sysstatistics
- sysqueryplans

システム・テーブルのデフォルト・パーミッションでは次の変更も行われています。

- public から sysobjects(audflags) パーミッションを取り消す。
- sysobjects のパーミッションを sso_role に付与する。

このコマンドを master データベースから実行すると、次のシステム・テーブルのデフォルト・パーミッションが付与または取り消されます。

- sysdatabases
- sysdevices
- syslocks
- sysmessages
- sysprocesses
- systransactions
- sysusages
- sysconfigures
- syscurconfigs
- syslanguages
- syscharsets
- sysservers
- systimeranges

- sysresourcelimits
- syslogins
- sysremotelogins

このコマンドでは次の変更も行われています。

- public から sysdatabases(audflags) の select を取り消す。
- public から sysdatabases(deftabaud) の select を取り消す。
- public から sysdatabases(defvwaud) の select を取り消す。
- public から sysdatabases(defpraud) の select を取り消す。
- sso_role に sysdatabases の select を付与する。
- public から syslogins(password) の select を取り消す。
- public から syslogins(audflags) の select を取り消す。
- sso_role に syslogins の select を付与する。

update statistics、*delete statistics*、*truncate table* のパーミッションの付与と取り消し

Adaptive Server バージョン 12.5.2 では、update statistics、delete statistics、truncate table の各コマンドに対する、ユーザ、役割、グループのパーミッションを付与または取り消すことができます。テーブル所有者も、暗黙の grant によってパーミッションを付与できます。具体的には、update statistics、delete statistics、truncate table をストアド・プロシージャに追加してから、そのプロシージャの実行パーミッションをユーザまたは役割に付与します。

update statistics のパーミッションをカラム・レベルで付与または取り消すことはできません。sysroles、syssrvroles、sysloginroles の各セキュリティ・テーブルに対して update statistics または delete statistics を実行するには、sso_role が必要です。

デフォルトでは、sa_role を持つユーザは、sysroles、syssrvroles、sysloginroles 以外のシステム・テーブルに対して update statistics と delete statistics を実行するパーミッションがあり、この権限を他のユーザに渡すこともできます。

grant と revoke の構文の一部は次のとおりです。

grant [truncate table | update statistics | delete statistics] on *table_name* to {*user_name* | *role_name*}

revoke [truncate table | update statistics | delete statistics] on *table_name* from {*user_name* | *role_name*}

grant all を発行して、update statistics、delete statistics、truncate table のパーミッションを付与することもできます。

たとえば、次の例は、ユーザ “harry” が authors テーブルに対して truncate table と updates statistics を使用できるようにします。

```
grant truncate table on authors to harry
grant update statistics on authors to harry
```

次の例は、“harry” の authors テーブルに対する truncate table 権限と update statistics 権限を取り消します。

```
revoke truncate table on authors from harry
revoke update statistics on authors from harry
```

次の例は、ユーザ “billy” が authors テーブルに対して delete statistics コマンドを使用できるようにします。

```
grant delete statistics on authors to billy
```

次の例は、ユーザ “billy” の authors テーブルに対する delete statistics 権限を取り消します。

```
revoke delete statistics on authors from billy
```

次の例は、oper_role を持つすべてのユーザに truncate table、update statistics、delete statistics の各権限を付与します (ユーザ “billy” と “harry” は、oper_role を持っている場合、これらのコマンドを authors に対して実行できるようになります)。

```
grant truncate table on authors to oper_role
grant update statistics on authors to oper_role
grant delete statistics on authors to oper_role
```

次の例は、oper_role を持つすべてのユーザの truncate table、update statistics、delete statistics の各権限を取り消します。

```
revoke truncate table on authors from oper_role
revoke update statistics on authors from oper_role
revoke delete statistics on authors from oper_role
```

ユーザ “billy” と “harry” は、これらのコマンドを authors に対して実行できなくなります。

また、ストアド・プロシージャを使用して、truncate table、delete statistics、update statistics のパーミッションを暗黙に付与することもできます。たとえば、“billy” が authors テーブルを所有している場合、billy は次を実行すると、authors に対して truncate table と update statistics を実行する権限を “harry” に付与できます。

```
create procedure sproc1
as
truncate table authors
update statistics authors
go
grant execute on sproc1 to harry
go
```

また、ストアド・プロシージャを使用してカラム・レベルで update statistics と delete statistics のパーミッションを暗黙に付与することもできます。

注意 update statistics を実行するパーミッションをユーザに付与すると、付与されたユーザはコマンドのバリエーション (update all statistics、update partition statistics、update index statistics、update statistics *table* など) を実行するパーミッションも取得します。たとえば、次の例は、authors テーブルに対して update statistics のすべてのバリエーションを実行するパーミッションを “billy” に付与します。

```
grant update statistics on authors to billy
```

update statistics を実行するパーミッションをユーザから取り消すと、そのコマンドのバリエーションを実行するパーミッションも取り消すことになります。

---

update statistics のバリエーション (update index statistics など) のパーミッションを個別に付与することはできません。つまり、次のようなコマンドは発行できません。

```
grant update all statistics to harry
```

ただし、ストアド・プロシージャを作成して、これらのコマンドをどのユーザが実行するかを制御することができます。たとえば、次の例は、authors テーブルに対して update index statistics を実行するパーミッションを “billy” に付与します。

```
create proc sp_ups as
update index statistics on authors
go
revoke update statistics on authors from billy
go
grant execute on sp_ups to billy
```

delete statistics のパーミッションをカラム・レベルで付与または取り消すことはできません。

Adaptive Server は、その他のグローバルな監査として truncate table を監査しますが、update statistics の監査は行いません。truncate table と update statistics の両方について明確な監査証跡を保持するためには、上記のように実行パーミッションをユーザに付与するストアド・プロシージャに両方のコマンドを含めることをおすすめします。

次の条件が当てはまり、かつユーザが update statistics、delete statistics、または truncate table コマンドを発行した場合、コマンドが失敗してエラー番号 10330 が生成されます。

- ユーザがテーブルを所有していない。
- ユーザが sa_role を持っていない。
- ユーザが、テーブルの所有者であるユーザになる setuser を使用したデータベースの所有者ではない。

- ユーザが、update statistics、delete statistics、または truncate table 権限を付与されていない。

grant set proxy コマンド

以前のバージョンの Adaptive Server では、set proxy を使用してサーバ・ユーザ ID を他のサーバ・ログインに切り替えることはできましたが、ターゲット・ログインの役割に基づいて set proxy の使用を制限することはできませんでした。set proxy を付与されたユーザは、他の任意のサーバ・ユーザになることが可能でした。

Adaptive Server バージョン 12.5.2 では、set proxy...restricted role を付与することによって、ID を切り替えたときに特定の役割を取得できないように制限できます。

set proxy の構文は次のとおりです。

```
grant set proxy to user_or_role_list
[restricted role role_list | all | system]
```

パラメータ

- *user_or_role_list* − ターゲット・ログインに対して制限する役割のリスト。付与対象者とターゲット・ログインの両方が、このリストのすべての役割を持っていることが必要です。そうでない場合はコマンドが失敗します。
- *all* − 付与対象者が持つすべての役割をターゲット・ログインに付与します。
- *system* − 付与対象者がターゲット・ログインと同じシステム役割の組み合わせを持つようにします。

例

- 例 1：この例は、set proxy をユーザ “joe” に付与しますが、joe が ID を、sa、sso、または admin の役割を持つユーザに切り替えることは制限します (ただし、joe がすでにこれらの役割を持っている場合は、これらの役割を持つユーザに対して set proxy を実行できます)。

  ```
  grant set proxy to joe
  restricted role sa_role, sso_role, admin_role
  ```

  “joe” が admin_role を持つユーザ (この例では Our_admin_role) に ID を切り替えようとした場合、joe が admin_role を持っていないかぎりコマンドは失敗します。

  ```
  set proxy Our_admin_role
  Msg 10368, Level 14, State 1:
  Server 's', Line 2: 自分にない役割がターゲット・ログインに含まれ、その使用を制限されているために、Set session 権限のパーミッションが拒否されました。
  ```

  “joe” が admin_role を付与された後でコマンドを再試行すると成功します。

  ```
  grant role admin_role to joe
  set proxy Our_admin_role
  ```

- 例 2：ID を切り替えるときに “joe” に新しい役割が付与されないようにします。

```
grant set proxy to joe
restricted role all
```

“joe” は、自分と同じ (またはより重要度の低い) 役割を持つユーザにしか set proxy を付与できません。

- 例 3: set proxy を使用するときに Joe が新しいシステム役割を取得できないようにします。

```
grant set proxy to joe
restricted role system
```

joe が持っていないシステム役割をターゲット・ログインが持っていると、set proxy は失敗します。

使用法

- grant set proxy を使用すると、役割の制限を段階的に拡大できます。たとえば、最初に sa_role を制限し、次に sso_role を制限することができます。

```
grant set proxy to joe
restrict role sa_role
grant set proxy to joe
restrict role sso role
```

- 個々の役割の制限を解除することはできません。すべての役割からパーミッションを取り消すには、次のクエリに示すように set proxy を取り消す必要があります。

```
select distinct user_name(p.uid), b.name, p.grantor,
                 Restricted_role=case
convert(tinyint,substring(isnull(p.columns,0x1),1,1)) & 1
                       when 1 then
                            "None"
                       else
                            isnull(role_name(c.number - 1), "System
"+convert(char,c.number))
                                  end
    from sysprotects p, master.dbo.spt_values b, master.dbo.spt_values c
    where
          convert(tinyint,substring(isnull(p.columns,0x1), c.low,1)) &
c.high = 0
         and c.type = "P" and c.number <= 1024 and c.number >0 and
p.action = 167
         and b.type = "T"
         and b.number = (p.protecttype + 204)
          and role_name(c.number - 1) is not null
```

optdiag

optdiag は、ユーザが指定したサンプリング率で収集される統計を示す sampling percent last used を表示します。

update と *delete* の変更

12.5.2 より前のバージョンの Adaptive Server では、クエリで、union all 句を含むビューに対して update や delete を使用すると、ワーク・テーブルを使用せずに解析され、誤った結果が得られることがありました。Adaptive Server 12.5.2 では、union all 句を含むビューに対して update や delete を使用するクエリは、必ず tempdb のワーク・テーブルを使用して解析されます。ただし、この動作が原因でパフォーマンスが低下する場合があります。

dbcc complete_xact への追加

外部トランザクション・コーディネータが機能しない状況で、システム管理者は dbcc complete_xact を使用して分散トランザクションをコミットまたはロールバックできます。以前のバージョンの Adaptive Server では、トランザクションは「準備」された状態でないかぎり自発的に完了できませんでした。トランザクション・コーディネータでは、トランザクションをコミットするために 2 フェーズ・コミット・プロトコルが使用されていました。ただし、トランザクションのコミットに 1 フェーズ・コミット・プロトコルを使用するのが最適な場合もあります。

Adaptive Server 12.5.2 では、dbcc complete_xact コマンドに 1pc パラメータが含まれています。1pc を使用すると、外部トランザクション・マネージャによって (通常の 2 フェーズ・コミット・プロトコルではなく) 1 フェーズ・コミット・プロトコルでの最適化の対象として完了の調整が行われていたトランザクションが自発的に完了します。このようなトランザクションを自発的にコミットするには、トランザクションが “done” 状態 (sp_transactions でレポートされる) であることが必要です。

dbcc complete_xact の構文の一部は次のとおりです。

```
dbcc complete_xact("<xid>", "commit", "1pc")
```

注意　トランザクションを自発的にコミットする前に、システム管理者は調整を行っているトランザクション・マネージャが分散トランザクションをコミットしたかどうかをあらゆる方法で判別する必要があります。

次の例は、sp_transactions を使用して、「準備」ステータスでなかったために自発的にコミットしなかった 1 フェーズ・コミット・トランザクションの名前を判別します。次に、トランザクションを正常にコミットするための 1pc パラメータの使用方法を示します。

```
sp_transactions
xactkey                          type     coordinator     starttime
state            connection      dbid       spid      loid
failover        srvnname      namelen
xactname
------------------------------  ---------- -----------    --------------------
--------------- ----------      ------      ------    -----------
----------      -----------   ------------------
-------------
0xbc0500000b00000030c316480100 External       XA         Feb  2 2004  1:07PM
Done-Detached     Detached         1         0        2099
Resident Tx     NULL             88
28_u7dAc31Wc380000000000000000000000000000000001HFpfSxkDM000FU_00003M00

00Y_:SYBBEV0A_LRM
(1 row affected)

(return status = 0)
```

このトランザクションをコミットしようとして次のように指定したとします。

```
dbcc
complete_xact("28_u7dAc31Wc380000000000000000000000000000000001HFpfSxkDM000FU
_00003M0000Y_:SYBBEV0A_LRM", "commit"))
```

この場合、Adaptive Server によって次のエラー・メッセージが発行されます。

```
Msg 3947, Level 16, State 1:
Server 'PISSARRO_1251_P', Line 1:
操作に関連する発見的なコンパイルに失敗しました。詳細については、エラー・
ログを参照してください。
DBCC の実行が完了しました。DBCC がエラー・メッセージを表示した場合、シス
テム管理者 (SA) の権限を持つユーザに連絡してください。
```

トランザクションは “done” 状態であるため、トランザクションがコミットされたことを確認してから、1 フェーズ・コミット・プロトコルでの最適化を使用してトランザクションを自発的に完了できます。次のように dbcc complete_xact(“1pc”) パラメータを使用すると、このトランザクションをコミットできます。

```
dbcc
complete_xact("28_u7dAc31Wc380000000000000000000000000000000001HFpfSxkDM
000FU_00003M0000Y_:SYBBEV0A_LRM", "commit", "1pc")
DBCC の実行が完了しました。DBCC がエラー・メッセージを表示した場合、システム管理者 (SA) の権限を持つ
ユーザに連絡してください。
```

dbcc forget_xact コマンドを使用すると systransactions からトランザクションを削除できます。

```
dbcc forget_xact("28_u7dAc31Wc380000000000000000000000000000000001HFpfSxkDM0
00FU_00003M0000Y_:SYBBEV0A_LRM")
DBCC の実行が完了しました。DBCC がエラー・メッセージを表示した場合、システム管理者 (SA) の権限を持つ
ユーザに連絡してください。
```

sp_transactions を再び実行すると、前のトランザクションは表示されません。

```
sp_transactions
xactkey                          type     coordinator     starttime
state            connection      dbid       spid      loid
failover        srvnname      namelen
xactname
------------------------------  ---------- -----------    --------------------
--------------- ----------      ------      ------    -----------
-------------
(0 row affected)
```

create procedure (SQLJ) の変更

以前のバージョンの Adaptive Server では、SQLJ プロシージャ・パラメータのデフォルト値は定義できませんでした。Adaptive Server 12.5.2 ではデフォルト値を定義できるため、パラメータ値を指定せずに SQLJ プロシージャを実行できます。

この create procedure (SQLJ) の新機能は、create procedure の既存の動作と同じようになります。

構文

```
create procedure [owner.]sql_procedure_name
    ( [ [ in | out | inout ] sql_parameter_name
        sql_datatype [( length )] | (precision[, scale]) ]
        [=default]
    [ , [ in | out | inout ] sql_parameter_name
        sql_datatype [( length )] | (precision[, scale]) ]
         [=default] ... ] )
...
```

新しいオプション

default

プロシージャのパラメータに対するデフォルト値を定義します。デフォルトが定義されている場合は、パラメータを指定しなくてもプロシージャを実行できます。デフォルトは定数でなければなりません。プロシージャが like キーワードの付いたパラメータ名を使用している場合は、デフォルトにワイルドカード文字 (%、_、[ ]、^) を使用することもできます。

デフォルトに NULL を指定することもできます。プロシージャ定義には、パラメータ値が NULL の場合に何らかの動作をするように指定できます。

例

このプロシージャは、常に 10 よりも大きい値を返します。

```
create procedure my_max(a int = 10, b int = 10)
language java parameter style java
external name 'java.lang.Math.ma'

exec my_max
(return status = 10)
```

```
exec my_max 8
(return status = 10)
```

Transact-SQL の create procedure の例も参照してください。

ストアド・プロシージャの変更

セキュリティ強化のためのストアド・プロシージャの変更

次のストアド・プロシージャにはセキュリティ機能が追加されています。

- sp_modifylogin
- sp_displaylogin
- sp_addlogin

詳細については、「機能拡張されたログイン制御」(57 ページ) を参照してください。

sp_monitor

以前のバージョンの Adaptive Server では、sp_monitor にはパラメータがありませんでした。Adaptive Server バージョン 12.5.2 では、この項で説明するパラメータが追加されています。

注意 sp_monitor の新しいパラメータを使用する前に、それらのオプションを有効にするために必要なモニタリング・テーブルと関連するストアド・プロシージャを設定してください。これらは *installmontables* スクリプトに含まれています。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「モニタリング・テーブルのインストール」を参照してください。

説明

Adaptive Server についての統計情報を表示します。

構文

```
sp_monitor [ connection, [cpu | diskio | elapsed time ] ]  [event, [spid ] ]
[procedure, [ dbname, [ procname, [, summary | detail ] ] ] ]  [ enable ]
[ disable ] [ statement, [ cpu | diskio | elapsed time ] ]
[ help], [ connection | statement | procedure | event ] ]
```

パラメータ

connection

各接続の情報を表示します。connection は次のモニタリング・テーブルを使用します。

- monProcessSQLText
- monProcessActivity

cpu | diskio | elapsed time

これらのパラメータによって sp_monitor connection の出力が順序付けられます。cpu は各接続で消費された CPU 時間を示します。diskio は各接続で実行された物理読み込み回数を示します。elapsed time は各接続の CPU 時間と待機時間の合計を示します。

event

各タスクが待機していたイベントと待機時間の情報を表示します。これは、待機時間 (ミリ秒) の降順で表示されます。event は次のモニタリング・テーブルを使用します。

- monProcessWaits
- monWaitEventInfo

spid

特定のタスクの *spid* を入力すると、そのタスクの event 情報を取得できます。*spid* は数値を引用符で囲んで指定します。

procedure

ストアド・プロシージャに関する次の統計を表示します。

- ProcName － モニタリングされているストアド・プロシージャ。
- DBNAME － ストアド・プロシージャが格納されているデータベース。
- NumExecs － この特定のストアド・プロシージャの概算の実行回数。
- AvgCPUTime － ストアド・プロシージャの実行にかかる平均 CPU 時間。
- AvgPhysicalReads － ストアド・プロシージャによって実行される平均ディスク読み込み回数。
- AvgLogicalReads － ストアド・プロシージャによって実行される平均論理読み込み回数。
- AvgMemUsed_KB － ストアド・プロシージャで使用される平均メモリ容量 (KB 単位)。

procedure は monSysStatement モニタリング・テーブルを使用します。

dbname

指定されたデータベースのプロシージャに関する情報を表示します。

procname

指定されたプロシージャに関する情報を表示します。

summary | detail

プロシージャのすべてのインスタンスの平均値を示す要約情報か、ストアド・プロシージャの各インスタンスの詳細情報を表示します。

enable

sp_monitor の新しいオプションを有効にします。モニタリングを開始するために必要な設定パラメータをオンにします。

disable

モニタリングを無効にします。

statement

sp_monitor statement は各文の情報を表示します。文は次のモニタリング・テーブルを使用します。

- monProcessSQLText
- monProcessStatement

cpu | diskio | elapsed time

これらのパラメータによって sp_monitor statement の出力が順序付けられます。cpu は各文で消費された cpu 時間を示します。diskio は各文で実行された物理読み込み回数を示します。elapsed time は各文の CPU 時間と待機時間の合計を示します。

help

sp_monitor の構文と例を表示します。

例

例 1 次の例は、接続に関する情報の表示方法を示します。

```
1> sp_monitor "connection"
2> go
spid    LoginName    ElapsedTime  LocksHeld  SQLText
----    ---------    -----------  ---------  -----------------
12      sa           90300        2          exec get_employee_salaries
27      sa           17700        1          exec get_employee_perks
```

デフォルトでは、出力は ElapsedTime の降順でソートされます。

例 2 次の例では、物理読み込みの実行回数が最も多い接続が示されます。

```
1> sp_monitor "connection","diskio"
2> go
spid  LoginName  Physical_Reads  LocksHeld  SQLText
----  ---------  --------------  ---------  -------------------------
12    sa         117             2          exec get_employee_salaries
27    sa         1               0          exec get_employee_perks
```

例 3 次の例は、各文に関する情報を表示します。

```
1> sp_monitor "statement"
2> go
spid   LoginName   ElapsedTime   SQLText
----   ---------   -----------   -------------------------
12     sa          100           exec get_employee_salaries
```

例 4　次の例は、各タスクが待機していたイベントとその待機時間を、待機時間の降順で表示します。

```
1> sp_monitor "event"
2> go
SPID   WaitTime    Description
------ ----------- ----------------------------------------
6      108200      hk: pause for some time
29     108200      waiting for incoming network data
10     107800      waiting while allocating new client socket
15     17100       waiting for network send to complete
14     5900        waiting for CTLIB event to complete
14     400         waiting for disk write to complete
7      200         hk: pause for some time
7      100         waiting on run queue after yield
12     100         waiting for network send to complete
```

例 5　次の例は、spid 14 のイベント・データを表示します。

```
1> sp_monitor "event","14"
2> go
WaitTime    Description
----------- ----------------------------------
9000 waiting for CTLIB event to complete
600 waiting for disk write to complete
200 waiting for disk write to complete
100 waiting on run queue after yield
100 wait for buffer write to complete
```

例 6　次の例は、最近実行されたプロシージャの要約を平均経過時間の降順で表示します。これは現在の状態ではなく履歴モニタリング情報です。

```
1> sp_monitor "procedure"
2> go

Average Procedure Statistics
============================

ProcName      DBName    AvgElapsedTime   AvgCPUTime  AvgWaitTime AvgPhysicalReads
AvgLogicalReads AvgPacketsSent NumExecs
-------------------------------------------------------------------------------
neworder_remote   tpcc   1833     16    1083     26   96    0     6
neworder_local    tpcc   1394     13    1181     31   122   0     38
tc_startup        tpcc   1220     3     1157     0    3     0     59
delivery          tpcc   1000     0     800      23   49    0     2
```

使用法

- sp_monitor は、システムに典型的な負荷がかかっているときに実行します。
- 通常、次の順序でプロシージャを実行します。
  - sp_monitor enable を実行する。
  - sp_monitor オプションを呼び出す。
  - モニタリングが終了したら sp_monitor disable を実行する。
- sp_monitor procedure を使用する場合、返されるロー数が非常に多くなることがあります。その場合は、detail オプションではなく summary オプションを使用することもできます。アクティブなシステムではこのコマンドが完了するまでにしばらく時間がかかることがあります。

パーミッション

sp_monitor を実行するには mon_role パーミッションが必要です。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「モニタリング・テーブル」を参照してください。

sp_dropalias の force パラメータ

sp_dropalias には、オプションのパラメータ force が含まれます。これを使用すると、データベース・オブジェクトを所有しているエイリアスも削除できます。以前のバージョンの sp_dropalias では、エイリアスを削除するには、エイリアスが所有するオブジェクトを前もって削除することが必要でした。

たとえば、force パラメータを使用すると、プロシージャ namelist を所有するエイリアス “harry” を削除できます。エイリアスが Adaptive Server によって削除されますが、次の警告メッセージが発行されます。

```
sp_dropalias harry, force
警告： DB にオブジェクト所有のログイン 'harry' のエイリアスを削除。これらのオブジェクトにほかの DB からアクセス時 / データベースへの参照がある時、エラー発生することあり。
エイリアス・ユーザは削除されました。

(return status = 0)
```

sp_audit

sp_audit を使用すると、拡張ストアド・プロシージャを監査できます。

sp_helptext

sp_helptext は、syscomments のソース・テキストを表示するときに末尾のスペースをトランケートします。

新しい関数、コマンド、ストアド・プロシージャ

次に、Adaptive Server バージョン 12.5.2 に新たに追加された関数、コマンド、ストアド・プロシージャを示します。

audit_event_name

監査イベントの説明を返します。

構文　audit_event_name(*event_id*)

パラメータ　*event_id*

監査イベントの番号です。

例　例 1

テーブル作成イベントの監査証跡を問い合わせます。

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

例 2

現在の監査イベント値を取得します。監査の値と説明のリストについては、表 12-3 を参照してください。

```
create table #tmp(event_id int, description
      varchar(255))
go
declare @a int
select @a=1
while (@a<120)
begin
      insert #tmp values (@a, audit_event_name(@a))
      select @a=@a + 1
end
select * from #tmp
go
-----------------------------------------
event_id    description
----------    -------------------
      1 Ad hoc Audit Record
      2 Alter Database
      ...
      104 Create Index
      105 Drop Index
```

使用法

表 12-3 は、105 個の監査イベントの ID と名前を示します。

表 12-3: 監査イベント

監査番号と説明		
1 特定の監査レコード	36 致命的なエラー	71 ビューの更新
2 データベースの変更	37 致命的でないエラー	72 NULL
3 テーブルの変更	38 ストアド・プロシージャの実行	73 監査の有効化
4 BCP 入力	39 トリガの実行	74 監査の無効化
5 NULL	40 grant コマンド	75 NULL
6 デフォルトのバインド	41 テーブルの挿入	76 SSO パスワードの変更
7 メッセージのバインド	42 ビューの挿入	77 テーブルの切り替え
8 ルールのバインド	43 データベースのロード	78 監査オプションの変更
9 データベースの作成	44 トランザクションのロード	79 NULL
10 テーブルの作成	45 ログイン	80 役割チェックの実行
11 プロシージャの作成	46 ログアウト	81 DBCC コマンド
12 トリガの作成	47 revoke コマンド	82 設定
13 ルールの作成	48 RPC 入力	83 データベースのオンライン化
14 デフォルトの作成	49 RPC 出力	84 setuser コマンド
15 メッセージの作成	50 サーバのブート	85 UDR コマンド
16 ビューの作成	51 サーバの停止	86 組み込み関数
17 データベースへのアクセス	52 curread の変更	87 ディスクの解放
18 テーブルの削除	53 curwrite の変更	88 set SSA コマンド
19 ビューの削除	54 更新モードの変更	89 kill/terminate コマンド
20 ディスクの初期化	55 役割のオン／オフ	90 connect コマンド
21 ディスクの修復	56 NULL	91 参照
22 ディスクの再初期化	57 NULL	92 コマンド・テキスト
23 ディスクのミラーリング	58 監査テーブルのトランケート	93 JCS install コマンド
24 ディスクのミラーリング解除	59 NULL	94 JCS remove コマンド
25 ディスクの再ミラーリング	60 NULL	95 管理者アカウントのロック解除
26 データベースの削除	61 監査テーブルへのアクセス	96 quiesce database コマンド
27 テーブルの削除	62 テーブルの選択	97 SQLJ 関数の作成
28 プロシージャの削除	63 ビューの選択	98 SQLJ 関数の削除
29 トリガの削除	64 テーブルのトランケート	99 SSL の管理
30 ルールの削除	65 信頼されているプロシージャの実行	100 ディスクのサイズ変更
31 デフォルトの削除	66 信頼されているトリガの実行	101 データベースのマウント
32 メッセージの削除	67 デフォルトのバインド解除	102 データベースのマウント解除
33 ビューの削除	68 ルールのバインド解除	103 login コマンド
34 データベースのダンプ	69 メッセージのバインド解除	104 インデックスの作成
35 トランザクションのダンプ	70 テーブルの更新	105 インデックスの削除

規格

ANSI SQL － 準拠レベル：Transact-SQL 拡張機能

パーミッション

すべてのユーザが audit_event_name を実行できます。

参照　select、sp_audit

sp_ldapadmin

説明　LDAP URL 検索文字列を作成またはリストします。あるいは、LDAP URL 検索文字列かログインを確認します。

構文

```
sp_ldapadmin { set_primary_url, 'ldapurl' |
        set_secondary_url, { 'ldapurl' | null } |
        set_access_acct, account_distinguished_name, account_password
        set_dn_lookup_url, ldapurl
        list_urls | check_url, 'ldapurl' |
        check_login, 'login_name' }

ldapurl::=ldap://host:port/node/?attributes?base | one | sub?filter
```

パラメータ

set_primary_url, '*ldapurl*'
　指定された検索文字列 *ldapurl* を作成します。正確に 1 つのプライマリ検索文字列を作成できます。

set_secondary_url, { '*ldapurl*' | null }
　指定されたセカンダリ検索文字列 *ldapurl* を作成するか、またはセカンダリ検索文字列を作成しません。正確に 1 つのセカンダリ検索文字列を作成できます。

set_access_acct
　アカウント、識別名、パスワード情報を使用して管理検索を実行できます。

account_distintuished_name
　ディレクトリ・サービス・アカウントの識別名です。

distintuished_name
　識別名は、階層の特定レベルのエントリをユニークに特定する名前であり、そのエントリの階層型ツリーのルートまでたどれるパスを表します。

account_password
　識別名に関連付けられているパスワードです。

set_dn_lookup_url
　識別名を検索する LDAP URL を指定します。この URL を設定すると、Adaptive Server で代替認証アルゴリズムが使用されます。

ldapurl
　ユーザに関連付けられている識別名を検索する URL 文字列です。デフォルトの属性名は *entrydn* です。

list_urls
　LDAP URL 検索文字列を表示します。

check_url, '*ldapurl*'
　LDAP URL 検索文字列を確認します。ユーザ・アカウントがあるかどうかも確認できますが、ユーザの認証は行いません。

check_login, *login_name*
既存の LDAP URL 検索文字列のユーザ・アカウントを確認します。ユーザの認証は行いません。

host
LDAP サーバのホスト名です。

port
LDAP サーバのポート番号です。

node
検索を開始するオブジェクト階層内でのノードを指定します。

attributes
結果リストで返す属性のリストです。属性リストは、LDAP サーバによって異なることがあります。

base | one | sub
検索条件を修飾します。base は、ベース・ノードの検索を指定します。one は、node で指定されたノードとその 1 つ下のレベルのノードの検索を指定します。sub は、node で指定されたノードとその下位レベルのすべてのノードの検索を指定します。

filter
認証する属性を指定します。フィルタは、“uid=*,” のように簡潔にしたり、“(uid=*)(ou=*group*)” のように複雑したりすることもできます。構文は LDAP サーバに応じて異なり、ログイン名の記述にワイルドカード (*) を使用します。

例

例 1 LDAP SunONE Directory Server 用の LDAP URL 検索文字列を作成します。

```
sp_ldapadmin set_primary_url,'ldap://voyager:389/
    ou=People,dc=MyCompany,dc=com??sub?uid=*'
```

検索文字列で特定するのは、ホスト名 “voyager” で受信しているディレクトリ・サーバ、ポート番号 389 (デフォルトの LDAP プロトコル・ポート)、検索を開始するベース・ノードが組織単位 (ou) “People” 内にあること、ドメインが “MyCompany.com” であることです。この検索文字列は、フィルタ uid=* に一致するすべての属性を返します。Adaptive Server は、ワイルドカードを認証対象となる Adaptive Server のログイン名に置換します。

例 2 例 1 で記述した基準を使用して OpenLDAP 2.0.25 で定義された、LDAP URL 検索文字列を作成します。

```
sp_ldapadmin set_primary_url,'ldap://voyager:389/
    dc=MyCompany,dc=com??sub?cn=*'
```

例 3 セカンダリ LDAP URL 検索文字列を null に設定します。このため、フェールオーバとセカンダリ LDAP サーバは指定されません。

```
sp_ldapadmin set_secondary_url, null
```

例 4 複合フィルタを使用して LDAP URL 検索文字列を作成します。

```
sp_ldapadmin set_primary_url, 'ldap://voyager:389/
    ou=people,dc=siroe,dc=com??sub?(&(uid=*)
    (ou=accounting))
```

例 5 検索のためのアクセス・アカウントを指定します。

```
sp_ldapadmin set_access_acct, 'cn=admin, ou=People, dc=mycompany, dc=com', 'admin
secret password'
```

例 6 アカウントの識別名を検索する URL を指定します。この例では、entrydn は、値が識別名である属性の名前を指定します。ユーザ ID についてはワイルドカード検索 (uid=*) を使用します。

```
sp_ldapadmin set_dn_lookup_url, 'ldap://myhost:389/ou=People,dc=
mycompany,dc=com?entrydn?sub?uid=*'
```

例 7 次の例は、アクティブ・ディレクトリ機能を LDAP ディレクトリ・サーバとして使用します (Adaptive Server バージョン 12.5.2 以降でのみ可能)。代替認証アルゴリズムを使用する一連のステップを構成し、ユーザの識別名を検索するために必要なアカウント情報を設定します。

1　アカウント情報を設定します。

```
sp_ldapadmin set_access_acct, 'cn=Admin Account, cn=Users, dc=mycompany, dc=com',
'Admin Account secret password'
```

2　プライマリ URL を設定します。

```
sp_ldapadmin set_primary_url,  'ldap://hostname:389/'
```

3　識別名の URL 検索を設定します。

```
sp_ldapadmin set_dn_lookup_url, 'ldap://hostname:389/cn=Users,dc=
mycompany,dc=com?distinguishedName?one?samaccountname=*'
```

4　クライアント接続のアカウント名を指定します。

```
% isql -Umylogin -Pmypassword
```

ユーザが Adaptive Server にログインするとき、アカウント名が LDAP 検索 (識別名の URL 検索で指定) で返される識別名と照合されて認証され、さらにクライアント接続パスワードが認証されます。

5　URL フィルタ・パラメータの *samaccountname* 属性名は、Windows と Adaptive Server 両方のアカウント・ログインです。次の検索は *distinguishedName* パラメータを返します。

```
'CN=MyGivenname MySurname,CN=Users,DC=mycompany,DC=com'
```

この *distinguishedName* パラメータは、LDAP サーバをバインドしてユーザを認証するために使用されます。

使用法

- LDAP ベンダが検索文字列の構文を決めています。どの場合でも、検索文字列はユーザをユニークに特定する属性名を指定します。形式は「属性 = ワイルドカード」で、たとえば “cn=*” のようになります。
- 複合フィルタの最初の属性では、相対識別名 (たとえば、`...sub?(uid=*)(ou=group)`) を定義する必要があります。そうしないと、認証に失敗します。
- set_access_acct パラメータの識別名とパスワードを NULL に設定すると、LDAP サーバで許可されている場合は set_access_account が匿名でバインドします。
- アカウント検索文字列の主な機能は、LDAP 検索を実行し、ユーザの完全識別名を探して返すことです。
- set_dn_lookup_url が複数の一致結果を検出した場合は、最初の結果だけが認証に使用されます。
- *account distinguished name* パラメータの最大長は 255 文字です。
- *account password* パラメータの最大長は 64 文字です。
- set_dn_lookup_url 検索で複数の属性値が返されると、最初の値だけが認証バインドに使用されます。
- 検索文字列が追加されると、Adaptive Server は、検索文字列中に有効な LDAP URL 構文が使用され, 実在のノードを参照していることを確認します。有効な文字列が確実に予測した値を返すようにするには、Adaptive Server の設定時に検索文字列を慎重に選択し確認します。
- セカンダリ URL 検索文字列は、別の LDAP サーバへのフェールオーバを有効にします。LDAP サーバが非アクティブになるか、検索文字列が無効にならないかぎり、Adaptive Server は、プライマリ URL 検索文字列を使用します。LDAP サーバが非アクティブになったり、検索文字列が無効になったりした場合、Adaptive Server は、認証にセカンダリ URL 検索文字列を使用します。

パーミッション

sp_ldapadmin を実行できるのはシステム・セキュリティ担当者だけです。

dbcc stackused

説明

データベースの一貫性チェッカ (dbcc) は、データベースの論理的および物理的な一貫性の検査を行い、統計、計画、修復機能を提供します。

構文

dbcc stackused

パラメータ

stackused

サーバが最初に起動されてから使用されたスタック・メモリの最大量をレポートします。

例

```
dbcc stackused
```

使用法

- 過去に使用されたスタック・メモリ量は、将来のニーズの可能性を示すだけです。過去の使用量を上回るスタック・メモリが必要になる場合もあります。dbcc statckused を定期的に実行して、現在のスタック・メモリ使用量を調べてください。

規格

SQL92 － 準拠レベル：Transact-SQL 拡張機能

パーミッション

dbcc stackused を実行できるのはシステム管理者だけです。

索引

記号

数字

A

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

あ

か

き

く

こ

さ

し

す

せ

そ

た

ち

て

と

の

は

ひ

ふ

ま

め

も

や

ゆ

ろ

わ