SYBASE®

移行ガイド

Adaptive Server® Enterprise

バージョン 15.0

ドキュメント ID：DC00065-01-1500-01
改訂：2006 年 9 月

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

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

マニュアルの注文

マニュアルの注文を承ります。ご希望の方は、サイ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 の商標

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

目次

はじめに

このマニュアルでは、Adaptive Server® Enterprise へのマイグレーション方法について説明します。

対象読者

このマニュアルは、Sybase™ の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。

このマニュアルの内容

- 「第 1 章 ビジネス要件の文書化」は、効果的なマイグレーション・プランに必要なビジネス情報を整理するのに役立ちます。
- 「第 2 章 環境の文書化」では、Adaptive Server 運用環境のシステム・ハードウェアとソフトウェアを文書化するためのガイドラインを示します。
- 「第 3 章 マイグレーション・プランの作成」では、マイグレーションの方法と、システム・リソース、アプリケーション、システム管理プロシージャに必要な変更を行うための計画の変更について説明します。
- 「第 4 章 必要なアプリケーションの変更」では、アプリケーションの実行に影響する問題やコーディングの変更が必要な問題について説明します。
- 「第 5 章 データベース管理の変更」では、あらかじめ認識していないと問題が発生することがある Adaptive Server のシステム管理の変更点について説明します。
- 「第 6 章 安定性とパフォーマンスの確認」は、テスト方法の評価とテスト・プランの作成に役立ちます。
- 「付録 A 現在の環境のワークシート」では、マイグレーションの計画に必要な情報を収集するためのガイドラインとサンプル・ワークシートを示します。
- 「付録 B サンプル・マイグレーション作業リスト」では、一般的なマイグレーション、並列マイグレーション、カットオーバ・マイグレーションの完了作業リスト・サンプルを示します。作業リスト・テンプレート・サンプルも示し、段階的カットオーバ・マイグレーションの概要についても説明します。
- 「付録 C マイグレーションの注意事項チェックリスト」では、マイグレーション・プランの作成に役立つチェックリストを示します。
- 「付録 D アップグレード前のチェックリスト」では、アップグレードの準備を最初の作業から最後の作業まで順序どおり行うためのチェックリストを示します。

関連マニュアル

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

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

  『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 15.0 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 Adaptive Server に対して基本的な複写を行うための Adaptive Server の Adaptive Server 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。

- 『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取得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。

- 『Messaging Services ユーザーズ・ガイド』－ Real Time Messaging Services を使用して、TIBCO Java Message Service と IBM WebSphere MQ Messaging Services をすべての Adaptive Server データベース・アプリケーションと統合する方法について説明します。
- 『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』－ パフォーマンスのモニタリングと最適化のために、統計を取得し、使用する方法について説明しています。
- 『クイック・リファレンス・ガイド』－ コマンド、関数、システム・プロシージャ、拡張システム・プロシージャ、データ型、ユーティリティの名前と構文の包括的な一覧表を記載したポケット版 (PDF 版は通常サイズ) のマニュアルです。
- 『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。フル・サイズのダイヤグラムは印刷版だけで参照できます。コンパクト版は PDF 形式で参照できます。
- 『Transact-SQL™ ユーザーズ・ガイド』－ リレーショナル・データベース言語の拡張版である Sybase の Transact-SQL について説明しています。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。
- 『Unified Agent および Agent Management Console』－ Unified Agent について説明します。Unified Agent は、分散 Sybase リソースを管理、モニタ、制御するためのランタイム・サービスを提供します。
- 『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 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 』では、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。

マイグレーション対応マニュアル

Sybase では、マイグレーション・プロセスのすべての段階に対応するマニュアルを提供しています。このマイグレーション・ガイドで説明しているのは、マイグレーション時の問題を回避するためにシステムやアプリケーションに対して行う必要がある最低限の変更のみです。したがって、Sybase の新しいパフォーマンス機能を利用するためには、『新機能ガイド』や他の Sybase マニュアルも参照して、新しい Adaptive Server システムの設計を計画することをおすすめします。

次の表は、マイグレーション・フェーズとそれに対応する Sybase のマニュアルを示します。

ドキュメント	使用フェーズ	対象作業の範囲
• 『Adaptive Server Enterprise 新機能ガイド』 • 『Adaptive Server Enterprise 15.0 へのマイグレーション』	アップグレード前の計画と準備	現在のシステムの評価 マイグレーションの計画 アプリケーションの互換性の確保 DBA の手順の更新
• 『リリース・ノート』	アップグレード前の計画と準備	問題レポート、インストールに関する特別な注意事項、互換性の問題など、アップグレード時の問題を回避するために必要な情報を探す
• 『インストール・ガイド』	アップグレードの準備と実装	アップグレードするシステムの準備 ソフトウェアのインストール アップグレード作業の実行
• 『システム管理ガイド』 • 『パフォーマンス&チューニング・ガイド』	マイグレーション前の計画と準備、アップグレード後のテスト	Adaptive Server 15.0 のシステム設計を計画する システムをモニタ、チューニングして、パフォーマンスを向上させる

Sybase データベース製品の詳細については、http://www.sybase.com/support/manuals を参照してください。

その他の情報

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

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

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

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

- Sybase Product Manuals Web サイトは、SyBooks CD のオンライン版であり、標準の Web ブラウザを使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。

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

- Sybase コンサルティングでは、Adaptive Server 15.0 へのマイグレーションを予定しているお客様にサービスを提供しています。Sybase コンサルティングでは、Sybase が独自に開発した SAFE/EM (Sybase Advanced Framework to Enable Effective Migration) という方法を使用します。このマニュアルで説明するマイグレーション方法は、SAFE/EM 方式に基づいています。

- Sybase では、Adaptive Server 15.0 へのマイグレーションの計画と実行を支援する広範囲のサービスを提供しています。Sybase 開発およびテストを実施したマイグレーション方法を使用することでマイグレーションに必要な時間を節約し、付随するリスクを低減することができます。

- Sybase 製品のトレーニング・コースの詳細については、Sybase 教育サービスのページを参照してください。Adaptive Server バージョン 15.0.1 のコースについては、Database Servers を参照してください。(http://www.sybase.com/education/coursecatalog/databaseservers)

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。

❖ **製品動作確認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. [Certification Report] をクリックします。
3. [Certification Report] フィルタを選択し、製品、プラットフォームおよび時間枠を指定して [Go] をクリックします。
4. [Certification Report] のタイトルをクリックして、レポートを表示します。

❖ **コンポーネント動作確認の最新情報にアクセスする**

1. Web ブラウザで Availability and Certification Reports を指定します。(http://certification.sybase.com/)
2. [Search By Base Product] で製品ファミリとベース製品を選択するか、[Search by Platform] でプラットフォームとベース製品を選択します。
3. [Search] をクリックして、入手状況と動作確認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。MySybase のユーザ名とパスワードを入力します。

3 製品を選択します。

4 時間枠を指定して [Go] をクリックします。EBF/Maintenance リリースの一覧が表示されます。

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

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

表記規則

次の項では、このマニュアルで使用されている表記について説明します。

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

表 1 に構文の規則を示します。

表 1: このマニュアルでのフォントと構文規則

要素	例
コマンド名、プロシージャ名、ユーティリティ名、その他のキーワードは**ゴシック体**フォントで表記する。	select sp_configure
データベース名とデータ型は**ゴシック体**フォントで表記する。	master データベース

要素	例
ファイル名、変数、パス名は斜体で表記する。	*sql.ini* ファイル *column_name* *$SYBASE/ASE* ディレクトリ

変数 (ユーザが入力する値を表す語) がクエリまたは文の一部である場合は等幅 (固定幅) 文字フォントの斜体で表記する。	`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*)
2 つのコロンと等号は、構文が BNF 表記で記述されていることを示す。この記号は入力しない。「～と定義されている」ことを意味する。	`::=`
中カッコは、その中のオプションを 1 つ以上選択しなければならないことを意味する。コマンドには中カッコは入力しない。	`{cash, check, credit}`
角カッコは、オプションを選択しても省略してもよいことを意味する。コマンドには角カッコは入力しない。	`[cash | check | credit]`
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。	`cash, check, credit`
パイプまたは縦線は複数のオプションのうち 1 つだけを選択できることを意味する。	`cash | check | credit`
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ります。支払方法を選択できる。角カッコで囲まれた項目の 1 つを選択する。追加品目を、必要な数だけ購入することもできる。各 buy に対して、購入した製品 (thing)、価格 (price)、オプションで支払方法 (cash、check、credit のいずれか) を指定します。

- 次は、オプション句のあるコマンドの構文の例です。

  sp_dropdevice [*device_name*]

  複数のオプションを持つコマンドの例を示します。

  select *column_name*
  from *table_name*
  where *search_conditions*

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

- Transact-SQL コマンドの使用例は次のように表記します。

  ```
  select * from publishers
  ```

- 次は、コンピュータからの出力例です。

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

(3 rows affected)
```

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

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

アクセシビリティ機能

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

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

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

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

不明な点があるときは

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

第 1 章

ビジネス要件の文書化

この章では、マイグレーションの計画の第一フェーズである、環境の文書化に着手します。この章は、効果的なマイグレーション・プランに必要なビジネス情報を整理するのに役立ちます。

この章の例で使用されるワークシートについては、「付録 A 現在の環境のワークシート」を参照してください。

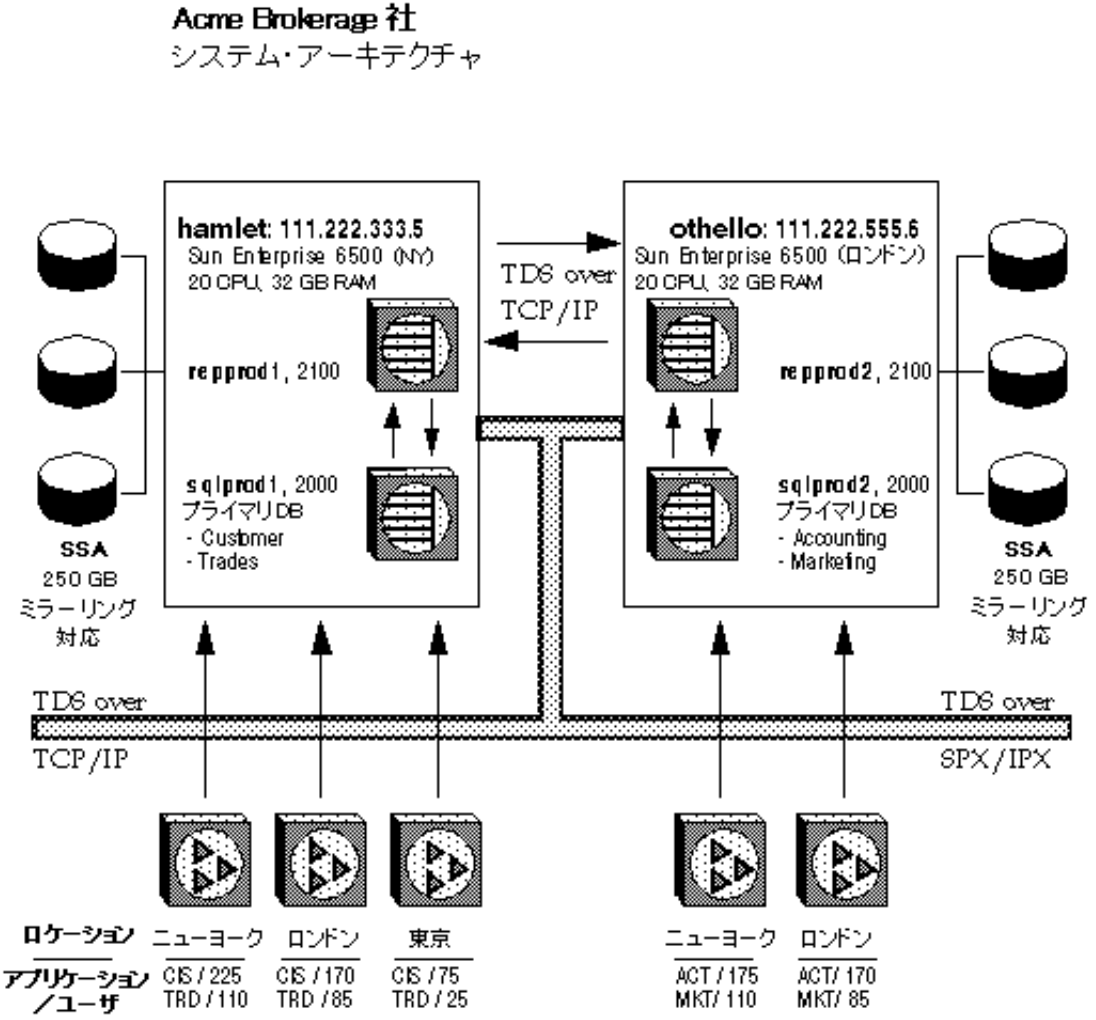
情報フロー図の作成

システムの情報の流れを示す図を作成します。表や説明文でも構いません。この図は、マイグレーション・チームの参考資料になります。この図には次の情報を含めます。

- サーバ (ファイル・サーバ、プリント・サーバ、アプリケーション・サーバなど)。各サーバについて、次の項目を記載します。
  - マシン名
  - IP アドレス
  - Sybase 名とエイリアス
- クライアント
  - アプリケーション
  - ユーザ数
- ネットワーク
  - プロトコル
  - ゲートウェイ
  - ルータ、ブルータ、ブリッジ

たとえば、世界数都市に支店があり、2 つの主要コンピューティング・センタを持つ Acme Brokerage という会社を図式化すると、図 1-1 のようになります。

図 1-1: 証券会社の情報フロー図



図に加えて、または図の代わりに、業務内容の概要を文章にまとめることもできます。

運用に関するビジネス要件の特定

この項では、運用に関するビジネス要件を文書化するための方法について説明します。ここで説明するベースライン要件は、マイグレーションを計画し、成功のための基準を確立する上で役立ちます。

可用性の要件

ユーザがデータベースにアクセスする必要がある時間と、許容できる最大ダウン時間を、次の例のように記録します。

データベース名	稼働時間	最大ダウン時間	コメント
TRD	07:00 ～ 23:00 月～金	5 分	
CIS	07:00 ～ 23:00 月～金	15 分	
ACT	07:00 ～ 21:00 月～土	5 分	
MKT	07:00 ～ 20:00 月～金	30 分	

データベースの変化指標

すべてのデータベースについて、次の項目を記録します。

- データベース・サイズ
- トランザクション・ログの増加状況
- テーブルのロー数と 1 日の変化率 (insert、delete、update の回数)

データベース・ダンプの詳細

ダンプの手順を、回数とデバイスを含め、次の例のように記録します。

データベース名	データベース・ダンプの頻度	使用するダンプ・デバイス	トランザクション・ログ・ダンプの頻度	使用するダンプ・デバイス	コメント
master	毎晩	master_dumpdev			
TRD	毎晩	TRD_tape1 TRD_tape2	15 分ごと	TRD_tape2	
CIS	毎晩	CIS_tape1	15 分ごと	CIS_tape3	
ACT	毎晩	ACT_tape1 ACT_tape2	15 分ごと	ACT_tape3	

メンテナンス手順

この例に示すような表を使用して、データ一貫性チェック・ツールとパフォーマンス・モニタ・ツールの実行スケジュールを記録します。

データベース名	dbcc checkdb と dbcc checktable の実行頻度	dbcc checkalloc と dbcc tablealloc の実行頻度	update statistics の実行頻度	dbcc checkstorage の実行頻度	モニタ・ツールの実行頻度
master	毎晩			毎晩	
TRD	毎週末	毎晩異なるテーブルを順次	毎晩異なるテーブルを順次	毎晩異なるテーブルを順次	1 時間ごと
CIS	毎週末	毎晩異なるテーブルを順次	毎晩異なるテーブルを順次	毎晩異なるテーブルを順次	1 時間ごと
ACT	毎週末	毎晩異なるテーブルを順次	毎晩異なるテーブルを順次	毎晩異なるテーブルを順次	1 時間ごと

サービス・レベルの要件

アプリケーションの詳細とサービス要件について、次の例のように記録します。

アプリケーション名	アプリケーションの種類	アプリケーション言語	クライアント・マシン	同時ユーザ数	アクセス対象のデータベース	可用性の要件 (1 日あたり)	パフォーマンス (平均応答時間)
Trades	負荷の高い OLTP	C	UNIX ワークステーション	220	TRD CIS	ダウン時間 5 分以下	2 秒以下
Customer	負荷の軽い OLTP DSS	Power-builder	PC	470	CIS	ダウン時間 10 分以下	5 秒以下
Accounting	負荷の軽い OLTP DSS バッチ処理	Power-builder	PC	345	ACT CIS	ダウン時間 5 分以下	5 秒以下
Marketing	DSS	C	PC	195	MKT CIS	ダウン時間 30 分以下	120 秒以下

トランザクション・プロファイル

statistics io、showplan、および dbcc 302 と dbcc 310 の両方を使用して、アプリケーション処理の詳細を取得し、トランザクション・プロファイルを記録します。重要なトランザクションについては、showplan と dbcc の出力を保存しておきます。この情報は、「第 6 章 安定性とパフォーマンスの確認」で説明するアップグレード後のテストでベースラインとして使用します。

次のような資料を作成します。

アプリケーション名	プロセス (正確な名前)	処理の種類	正確な優先度	ユーザごとの頻度 (1 時間あたり)	ソース・コード	平均応答時間の要件	最大応答時間の要件	現在の平均／最大応答時間
Trades	addTrade	負荷の高い OLTP	P1	90	ストアド・プロシージャ	2 秒以下	5 秒以下	平均 1 秒、最大 3 秒
Trades	bustTrade	負荷の高い OLTP	P1	10	ストアド・プロシージャ	5 秒以下	10 秒以下	平均 2 秒、最大 8 秒
Trades	reconcileTrades	バッチ処理	P1	1 日 1 回	Embedded SQL/COBOL	30 分以下	60 分以下	平均 25 分、最大 45 分
Trades	listAccounts	負荷の軽い OLTP	P1	180	ストアド・プロシージャ	2 秒以下	5 秒以下	平均 1 秒、最大 2 秒

注意　Adaptive Server バージョン 12.0 以降からアップグレードする場合は、重要なクエリの抽象クエリ・プランを保存しておくことができます。詳細については、『パフォーマンス＆チューニング・ガイド』を参照してください。

トランザクション統計の収集の詳細については、『パフォーマンス＆チューニング・ガイド』を参照してください。また、クエリ処理については、ASE Migration Resources Web ページ (http://sybase.com/support/techdocs/migration) の TechNotes とホワイトペーパーを参照してください。

現在のパフォーマンス測定基準の記録

次のようなパフォーマンス情報をできるだけ多く記録してください。モニタリング機能は、Monitor Server、モニタリング・テーブル、Historical Monitor、sp_sysmon などの形式で提供されています。

- CPU の使用率：

  オペレーティング・システムのモニタと Sybase モニタを使用して、各サーバの「時間枠」(オンライン時やバッチ処理時など) あたりの CPU の平均使用率と最大使用率 (SMP サーバの全 CPU と CPU ごと) を測定します。

- ディスク I/O：
  - オペレーティング・システムのモニタを使用して、ディスク別とコントローラ別の 1 秒あたりの I/O と、サーバごとの「時間枠」あたりの I/O キューの長さを測定します。
  - Sybase モニタを使用して、各サーバの Sybase デバイス別に「時間枠」あたりの毎秒の I/O、読み込み、書き込みの各総数を測定します。
- 同時実行性：
  - Sybase モニタを使用して、ロック競合の平均数を調べます。
- ネットワーク I/O：
  - Sybase モニタを使用して、重要なストアド・プロシージャの平均実行速度を記録します。
  - オペレーティング・システムのモニタを使用して、各サーバネットワーク・インタフェース・カード別に「時間枠」あたりの毎秒のパケット数を測定します。
  - Sybase モニタを使用して、サーバごとに「時間枠」あたりの TDS パケット数 (「送信」パケットと「受信」パケット) を測定します。
  - モニタリング・テーブルを使用して、サーバの状態についての統計情報と分析情報を収集します。
- メモリ：
  - オペレーティング・システムのモニタを使用して、サーバごとに「時間枠」あたりの毎秒のページング／スワッピング速度を調べます。
  - Sybase モニタを使用して、サーバごとに「時間枠」あたりのデータとプロシージャ・キャッシュのヒット率を調べます。

Adaptive Server リリース 15.0 へのアップグレードが完了したら、クエリ測定基準ユーティリティを使用して、以前の抽象プランと現在の抽象プランを比較します。

Sybase のツールとストアド・プロシージャを使用したパフォーマンス要素のモニタリングの詳細については、『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』を参照してください。

その他のビジネス要件の記録

その他の重要な運用に関するビジネス要件を文書化します。たとえば、次のものがあります。

- マイグレートするアプリケーションの優先リスト
- 制約
  - 年末／期末の処理を回避する必要があるかどうか
  - 許容できるダウン時間の量
  - アップグレードを週末に行うかどうか
  - 利用できるスタッフ
  - 他に利用可能なリソース (Replication Server、マシン、ツール、資金など)
- アプリケーションとデータ・サーバの依存性
  - 複数のアプリケーションが同一の Adaptive Server を使用するかどうか
  - 1 つのサーバのすべてのアプリケーションをマイグレートするかどうか
- ベンダの問題 (たとえば、使用するサード・パーティ・アプリケーションが Adaptive Server バージョン 15.0 で動作することが確認されているかどうか)

第 2 章

環境の文書化

この章では、Adaptive Server 運用環境のシステム・ハードウェアとソフトウェアを文書化するためのガイドラインを示します。この情報は、マイグレーションの計画フェーズでリソースの問題を特定するときに使用できます。

この章の例で使用されるワークシートについては、「付録 A 現在の環境のワークシート」を参照してください。

ハードウェアの設定

ハードウェア環境の文書化

- サーバ・ハードウェア全般
- マシンごとの CPU リソース
- ディスクの設定
- ネットワークの設定
- テープの設定

サーバ・ハードウェア全般

各サーバ・マシンについて、以下を記録します。

- 製造元と機種
- ベンダに登録されている顧客 ID

- サポート・センタの情報
  - 電話番号
  - サポート受付時間
  - アカウント管理者の名前と、電話番号またはポケットベルの番号
  - ベンダの Web ページ

マシンごとの CPU リソース

サーバ・マシンごとに次の CPU 情報をリストします。

- プロセッサの総数とその処理速度
- Adaptive Server が使用できるプロセッサの数
- これらのプロセッサを共有するその他の CPU 集中利用プロセス
- 特定の CPU にバインドされたプロセスとスレッドのリスト
- 実行優先度が高いプロセスとスレッドのリスト

ディスクの設定

次の例に示すようなワークシートを使用して、次のディスク I/O 情報を収集します。

- コントローラ・マップ
- ディスク・レイアウト・マップ
- ディスク・パーティション・マップ
- 論理ボリューム・マップ

コントローラの番号	製造元と機種	ファームウェアのリビジョン	サービス期間 (月)	転送率 (KB/ 秒)
0	Sun Fire V440	1.50	9	7500
1	Sun Fire V440	1.00	6	10000

物理デバイス名	製造元と機種	ファームウェアのリビジョン	サービス期間 (月)	コントローラの番号	容量 (MB)	スループット (1 秒あたりの I/O)	転送率 (KB/ 秒)
c0t0d0	Seagate ST43401N	2.15	9	0	2900	80	1500
c0t0d1	Seagate ST43401N	2.15	12	0	2900	80	1000
c0t0d2	Seagate ST43401N	2.15	24	0	2900	80	1600
c0t0d3	Seagate ST43401N	2.00	16	0	2900	80	1200

論理ボリューム名	メンバ・ディスク・パーティション	使用システム (Sybase、UFS)	Sybase デバイス名	ミラー論理デバイス	容量 (MB)	ストライプ幅 (MB)
lv dev1	c0t0d0s3 c0t0d0s4 c0t0d1s3 c0t0d1s4	sybase	TRD ログ	lv dev1 ミラー	500	4
lv dev2	c0t0d0s3 c0t0d0s4 c0t0d1s3 c0t0d1s4	sybase	CIS ログ	lv dev2 ミラー	500	4

物理デバイス名	パーティション番号	使用システム (Sybase、UFS)	デバイス名	OS ミラー・デバイス名	容量 (MB)	シリンダの範囲
c0t0d0	s0	ディスク・ラベル			2	0 ～ 1
	s2	バックアップ				
	s3	スワップ	スワップ		998	2 ～ 501
	s4	sybase	TRD ログ	c1t0s4	500	502 ～ 752
	s5	sybase	CIS ログ	c1t0s5	500	753 ～ 1003
	s6	ufs	/usr		900	1004 ～ 2733
c0t0d2	s0	ディスク・ラベル			2	0 ～ 1
	s2	バックアップ				
	s3	スワップ	スワップ		2900	2 ～ 2733

ネットワークの設定

次の例に示すようなワークシートを使用して、サーバ・マシンとクライアント・マシンのネットワーク・インタフェース・カード情報を記録します。

物理デバイス名	製造元と機種	ファームウェアのリビジョン	サービス期間 (月)	サポートするプロトコル	ネットワーク・アドレス	転送率 (KB/ 秒)
c0t0d0	Sun Fire V480	1.5	9	TCP/IP SPX/IPX	121.222.233.11	7500
c0t0d1	Sun Fire V480	1.00	12	TCP/IP SPX/IPX	121.222.555.33	7500

テープの設定

次の例に示すようなワークシートを使用して、テープまたはその他の記憶メディアの設定を記録します。

物理デバイス名	製造元と機種	ファームウェアのリビジョン	サービス期間 (月)	コントローラの番号	容量 (MB)	転送率 (KB/ 秒)
/dev/rmt/0	Sun Fire E10K	2.15	9	2	2000	500
/dev/rmt/1	Sun Fire E10K	1.00	12	2	2900	500

物理メモリの使用率

サーバ・マシンで実行する主なプロセスをすべてリストし、次に示す式を使用して必要なメモリを計算します。個々のメモリ量を合算して必要なメモリの総計を求めます。

次の表を見本として使用してください。

名前	実行時メモリ使用率の計算
オペレーティング・システム	OS 固有
Adaptive Server	設定ファイルで指定された max memory の値 (Adaptive Server バージョン 12.0 の場合は total memory を使用)
Backup Server	Backup Server の設定ファイルの -m パラメータで指定された max memory の値を追加
Replication Server™	“memory limit” パラメータを参照
ミドルウェア (たとえば Enterprise Connect Data Access)	製品ごとに異なる ― 製品マニュアルを参照
その他のアプリケーション { リストする }	アプリケーションごとに異なる
必要メモリの合計	
取り付けられているメモリの合計	

ソフトウェアの設定

次の各項の説明に従って、ソフトウェア環境を文書化します。

- オペレーティング・システム
- アプリケーション

オペレーティング・システム

次のオペレーティング・システム情報をリストします。

- オペレーティング・システムの名前
- リリース・レベル
- パッチ・レベル
- カーネル設定パラメータ
- スワップ・サイズ
- インストールされている OS 固有のソフトウェア
- インストールされている高可用性ソフトウェア

システム・アップグレードや最新のパッチを入手したり、問題に対処するためのサポートを利用するために、オペレーティング・システムのベンダへの連絡が必要になる場合があります。このような場合に備えて、オペレーティング・システムについて次のサポート・センタ情報を書き留めておきます。

- 電話番号
- サポート受付時間
- テクニカル・アカウント管理者の名前と、電話番号またはポケットベルの番号
- ベンダの Web ページ

アプリケーション

Adaptive Server バージョン 15.0 にマイグレートするアプリケーションのリストを作成します。各アプリケーションについて、以下を記録します。

- データと使用方法についての情報
  - 分散データ
  - データをウェアハウスに格納するか、トランザクション処理に使用するか。トランザクション処理に使用する場合は、データが正確で適切なフォーマットであることが必要です。データをウェアハウスに格納する場合、フォーマットについてはそれほど厳密ではありませんが、データ型変換による差異など、計算上のわずかな違いが発生します。
- アプリケーション・ソース・ファイルのロケーション
- 変更が必要かどうかを評価する対象の SQL コードを含むモジュール (たとえばトリガとストアド・プロシージャ) の種類と数

Sybase の設定

次の各項の説明に従って、Sybase の設定を文書化します。

- 全般情報
- データベース・デバイス
- データベースとセグメント
- ダンプ・デバイス

全般情報

次の Sybase 情報を記録します。

- Adaptive Server とその *$SYBASE* ホーム・ディレクトリ (Windows の場合は *%SYBASE%*)
- コンポーネントとリリース・レベル (EBF を含む)
- サーバのページ・サイズの設定
- データベース環境を再構築するためのスクリプトの名前とロケーション
- サーバのすべての設定値 (*.cfg* ファイルで確認可能)

データベース・デバイス

次の例に示すようにデータベース・デバイス情報を記録します。

データベース・デバイス名	物理デバイス名	ミラー・デバイス名	仮想デバイス番号	サイズ (MB)
TRD_dev1	/dev/rdsk/c0t0d0s3		2	10020
TRD_dev2	/dev/rdsk/c0t1d0s3		3	5020
TRD_log	/dev/rdsk/c0t1d0s4		4	1020
CIS_dev1	/dev/rdsk/c0t1d1s3		5	4020
CIS_log	/dev/rdsk/c0t1d1s4		6	420

データベースとセグメント

すべてのセグメントと、セグメント上にあるオブジェクトをリストします。次の例に示すようなワークシートを使用します。

データベース名	設定されているデータベース・オプション	サイズ (MB)	セグメント名	デバイス名	サイズ (MB)
master	なし	700	default.system.log	master	3
master	なし	500	default.system.log	master	2
master	なし	300	default.system.log	master	1
model	なし	200	default.system.log	master	2
tempdb	select into/bulkcopy	200	default.system.log	master	2
TRD	なし	10000	system、trd_seg1	TRD_dev1	10020
TRD	なし	5000	system、trd_seg1、trd_seg2	TRD_dev2	5020
TRD	なし	1000	log	TRD_log	1020
CIS	なし	4000	system、cis_seg1	CIS_dev1	4020
CIS	なし	400	log	CIS_log	420

ダンプ・デバイス

次の例に示すようにダンプ・デバイス情報を記録します。

データベース・デバイス名	物理デバイス名	メディアの種類	容量 (MB)
Tape dev1	/dev/rmt/0m	4mm	2000
Tape dev2	/dev/rmt/1m	4mm	2000
Tape dev3	/dev/rmt/2m	4mm	2000
Tape dev4	/dev/rmt/3m	4mm	2000
Tape dev5	/dev/rmt/4m	4mm	2000

Adaptive Server のオブジェクト

現在の Adaptive Server に存在するオブジェクトを記録します。

再作成に必要なスクリプトを探すか、スクリプトを作成します。

- サーバ・レベル・オブジェクト
  - データベース・デバイス
  - 設定
  - ログインとセキュリティ
- 次のようなデータベース・レベル・オブジェクト
  - デフォルト、ルール、ユーザ・データ型
  - ユーザ・データベース
  - ユーザ、グループ、エイリアス
  - テーブル、ビュー、ストアド・プロシージャ
  - その他のデータベース・オブジェクト (トリガやインデックスなど)

場合によっては、bcp を実行してデータを抽出し、ロードする必要があります。これらのスクリプトは、テスト環境の設定や新しい運用システムの構築に使用できます。また、リリース・レベルの異なる 2 つのサーバ・システムを使用する場合は、これらのスクリプトが必要になることがあります。

スクリプトがない場合は、スクリプトを複製するか、設定やオブジェクトを複製するために必要な情報にアクセスします。そのためには、次のような方法があります。

- システム・テーブルをクエリします。次のシステム・テーブルに、インストール・スクリプトの作成に必要なオブジェクト情報が含まれています。
  - sysdatabases
  - sysdevices
  - syslogins
  - syspartitions
  - sysobjects
  - sysremotelogins
  - sysservers
  - sysusages
  - sysusers

システム・テーブルとオブジェクトの詳細については、『システム管理ガイド』を参照してください。システム・テーブルを使用してデータベースを再構築する方法については、「Segment Remapping with load database When Moving a Database」(http://www.sybase.com/detail?id=1324) を参照してください。

- システム・ストアド・プロシージャを使用します。現在の Adaptive Server の設定を調べるには、引数を指定しないで sp_configure を実行します。すべての設定パラメータとその値がリストされます。

  ストアド・プロシージャに含まれる SQL コマンドを調べるには、sp_helptext を実行します。

  sp_helpdevice、sp_help、sp_helpdb、sp_helpsegment、sp_helpartiton、sp_helpindex など、サーバのオブジェクトについての情報を取得できるその他のシステム・ストアド・プロシージャの詳細については、『リファレンス・マニュアル：プロシージャ』を参照してください。

- Sybase ツールを使用します。Sybase Central™、PowerDesigner™、WorkSpace などの Sybase ツール、またはサード・パーティのツールを使用して、サーバ・オブジェクトをリバースエンジニアリングできます。

Adaptive Server のパフォーマンス

sp_sysmon を使用して、現在の Adaptive Server のピーク時とアイドル時の使用率についての情報を収集します (一般には 2 ～ 5 分のサンプルで十分ですが、取得するサンプルの量と頻度の両方を判断するためには、当該環境について把握している必要があります)。sp_sysmon の使用方法については、『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「sp_sysmon を利用したパフォーマンスのモニタリング」を参照してください。

また、モニタリング・テーブルを使用してサーバ・パフォーマンス全体を評価することもできます。詳細については、『パフォーマンス&チューニング・ガイド:モニタリングと分析』の「モニタリング・テーブル」を参照してください。

Database Expert などのサード・パーティ・ツールを使用して、現在の Adaptive Server の設定でのクエリ・パフォーマンスについて、ベンチマークの実行と文書化を行うことができます。詳細については、「Analyzing Performance Changes during Adaptive Server Migrations and Upgrades with Sybase Database Expert」(http://www.sybase.com/detail?id=1028449) を参照してください。

サード・パーティ・ツールは、クエリ・レベルとサーバ・レベルのパフォーマンス・データの取得と分析にも使用できます。

第 3 章

マイグレーション・プランの作成

現在のシステムに関するデータを収集したら、マイグレーション・プランを作成することをおすすめします。この章では、マイグレーションの方法と計画について説明します。マイグレーション方法の選択だけでなく、システム・リソースを Adaptive Server バージョン 15.0 で必要なレベルに増やし、アプリケーションとシステム管理プロシージャに必要な変更を行う必要があります。

アップグレード処理を開始する前に、次の点について考慮します。

- Adaptive Server を稼動する前に、まず SySAM ユーティリティを使用してサイト・ライセンスを実装する必要があります。インストール前の計画および SySAM のインストールの情報については、使用するプラットフォームに対応した Adaptive Server の『インストール・ガイド』を参照してください。SySAM の詳細については、『Users guide: Software Asset Management 2.0』を参照してください。また、SySAM とライセンスの問題の詳細については、http://www.sybase.com/sysam も参照してください。
- Adaptive Server のインストール環境がバージョン 11.5.x またはそれよりも古い場合は、まず Adaptive Server バージョン 12.0 にアップグレードしてからバージョン 15.0 にアップグレードすることをおすすめします。
- 最新の緊急バグ修正 (Emergency Bug Fixes: EBF) リリースにアップグレードしてサーバをマイグレートする前に、EBF に含まれている *README* ファイルを参照して、全般的な情報、ロードに関する説明、および個別の修正内容を確認してください。
- Adaptive Server のバージョンによっては、機能を制御するためのシステムの役割が含まれています。ユーザ ID、ログイン ID、役割を制御するためのシステム・アカウントと同様の機能を持つようにユーザまたは役割 ID の値を変更した場合は、新しいバージョンの Adaptive Server で追加されたユーザや役割とこれらの ID が競合しないことを確認してください。

データベースのアップグレード時に、データベースに新しい役割が追加される場合があります。この役割と ID のペアがサーバに既に存在する場合、新しい役割は追加できず、アップグレード全体が失敗します。Adaptive Server をアップグレードする前に、sysusers および sysroles をチェックして、手動で追加したユーザまたは役割 ID が通常の Sybase ユーザの範疇にあることを確かめてください。

- Adaptive Server 15.0 へのアップグレードで dump と load を使用している場合、アップグレードが完了した後で、アップグレードする最初のデータベースをロードする前に、既存のログイン ID と役割情報を、古いバージョンからアップグレード後のサーバへコピーする必要があります。
- アップグレードが完了したら、テーブルを再分割する必要があります。アップグレードを実行する前にこれらのテーブルを手動で分割解除しておくと、アップグレード時の時間と手間を節約できます。これは、分割方式が変更になったテーブルの場合に特に便利です。分割の変更の詳細については、「パーティションの変更」(49 ページ) を参照してください。

アップグレード・プロセスの注意事項

64 ビット・オペレーティング・システムまたは拡張されたページ・サイズへのマイグレーション

一部のプラットフォームでは、64 ビット・バージョンの Adaptive Server を使用できます。プラットフォームによっては、他のバージョンが用意されていないものもあります。

Adaptive Server バージョン 12.5 以降では、2K より大きな論理ページ・サイズを使用するサーバを構築できます。詳細については、『Adaptive Server Enterprise 新機能ガイド』を参照してください。

表 3-1 は、Adaptive Server を両方のケースでマイグレートする手順を示します。

表 3-1: 特殊な事例でのマイグレーション

マイグレーションの種類	使用できる方法とツール
15.0 より前の 32 ビット・サーバから 15.0 の 64 ビット・サーバへ	sqlupgrade ユーティリティを使用できる。 **注意** sqlupgrade ユーティリティを使用できるのは、リリース・レベルを変更する場合だけです。32 ビット・システムから 64 ビット・システムに移行するには、32 ビット・サーバのインストール環境に対して 64 ビット・サーバを起動します。

マイグレーションの種類	使用できる方法とツール
15.0 の 32 ビット・システムから 15.0 の 64 ビット・システムへ	新しい 15.0 インストールを作成し、bcp またはダンプとロードを使用してデータを新しいサーバに移行する。バイナリを手動で置き換えて 64 ビット・システムに変更することも可能。詳細については、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。
15.0 より前の 2K ページ・サーバから 4K、8K、または 16K ページのサーバへ	Adaptive Server のスキーマをあるページ・サイズから別のページ・サイズに変更することは、アップグレードではなくデータベースのマイグレーション。マイグレーションの詳細については、『ユーティリティ・ガイド』の「第 6 章 マイグレーション・ユーティリティ」を参照。

Adaptive Server 12.5.1 でのダンプとロード以外の方法

Adaptive Server 12.5.1 以降をアップグレードする場合で、既存のサーバのアップグレードではなく新しいサーバの構築を予定している場合は、データベースのダンプとロードを行う方法より、次に示す手順の方が早く済みます。Storage Area Network (SAN) 技術を使用している場合には、新しいハードウェアへのマイグレーションを同時に行うときでも、この方法が最も高速です。構築するサーバは、古いサーバの物理デバイス (またはそのコピー) にアクセスできることが必要です。また、データベースは、デバイスと整合していることが必要です。デバイスに複数のデータベースのフラグメントが含まれている場合は、すべてのデータベースを同時に移動またはアップグレードすることが必要なことがあります。

1　テスト中に 12.5.x のデータベースを静止してマニフェスト・ファイルに記録するか、またはサーバからのデータベースのマウントを解除します。

2　15.0 の Adaptive Server が別のホストにある場合は、現在のホストからのディスク・デバイスのマウントを解除して新しいホスト・マシンにマウントするか、または SAN ユーティリティを使用してデバイスをコピーします。

3　新しい Adaptive Server 15.0 のロケーションにマニフェスト・ファイルをコピーします。

4　マニフェスト・ファイルを使用して 12.5.x データベースを Adaptive Server 15.0 サーバにマウントします。場合によっては、サーバ・ロケーションの新しいデバイス・パスを記述し直すことが必要です。

5　online database コマンドを発行します。データベースのアップグレードは、データベースをオンラインにしたときに開始されます。

例

次の例では、testdb というマニフェスト・ファイルを使用して *testdb_manifest.mfst* データベースをアップグレードするものとします。

1 12.5.x データベースを次のように静止します。

```
quiesce database for_upgrd hold testdb
for external dump
to "/opt/sybase/testdb_manifest.mfst"
with override
```

2 ディスク・コピー・コマンド (たとえば dd)、SAN ユーティリティ、または標準のファイル・システム・コマンドを使用してデバイスをコピーします。

3 デバイスのコピーが完了したら、次のように quiesce を解放します。

```
quiesce database for_upgrd release
```

4 これがこのサーバの最終のアップグレードの場合 (テスト用のサーバではない場合) には、サーバを停止して、これ以降は変更されないようにします。

5 デバイスのコピーを新しいホスト・マシンに移動し、必要に応じてマウントします。

6 15.0 のサーバで、次のコマンドを発行して、物理デバイスと論理デバイスのマッピングをリストします。

```
mount database all from "/opt/sybase/testdb_manifest.mfst" with listonly
```

7 手順 6 の結果に基づいて、論理デバイスに対応する新しい物理デバイス・マッピングを確認し、15.0 の Adaptive Server でデータベースを次のようにマウントします。

```
mount database all from "/opt/sybase/testdb_manifest.mfst" using
"/opt/sybase/syb15/data/GALAXY/test_data.dat" = "test_data",
"/opt/sybase/syb15/data/GALAXY/test_log.dat" = "test_log"
```

8 データベースをオンラインにして、アップグレード処理を開始します。

```
online database testdb
```

マイグレーション方法の決定

最適なマイグレーション方法は、作業コスト、業務の種類、データベース・サイズ、使用可能なリソースなどの要素によって決まります。

表 3-1 は、各マイグレーション方法のメリットとデメリットを示します。

表 3-2: 各マイグレーション方法のメリットとデメリット

方法	メリット	デメリット	使用する場合
複写使用の並列マイグレーション	前のバージョンに簡単にフォールバックできる。前のリリースのデータベースを再構築する必要がない。 システム・ダウン時間が最短。	OLTP 環境では複雑になることがある。 Replication Server を設定する必要があり、追加のハードウェアとソフトウェアが必要。	この方法は、24 時間 365 日稼動している大規模な運用データベースに適しており、次のような場合に高い可用性を維持できる。 • リリース・データベースの再構築に時間がかかる場合。 • システムに多数のトランザクションと、サブクエリを含む複雑な Transact-SQL クエリがある場合。
複写なしのカットオーバ	実行に必要なリソースが最小限。	リスクが最大。重要なマイグレーション作業でダウン時間が必要。 運用環境ではリカバリに時間がかかることがある。	この方法は、リソースが限られた環境に適している。大規模な組織では、週末などに十分なダウン時間をスケジュールできる場合のみ有用である。
段階的カットオーバ	リスクが低く、開発オーバヘッドが少ない。 特にテストに有用。	追加のリソース (追加メモリまたはもう 1 つのシステム) が必要になることがある。 アプリケーション・グループやデータベース所有者との緊密な連携が必要。	他の 2 つの方法が適切でない場合は、段階的カットオーバを使用できる。

これらの方法の詳細については、次の各項を参照してください。

- 複写使用の並列マイグレーション
- 複写なしのカットオーバ
- 段階的カットオーバ

注意 このガイドでは、2 つのシステムの並列実行 (両方のシステムを同時に管理する必要がある) やトランザクションの重複 (1 つのフロントエンドで 2 つの並列のバックエンドを駆動する) のような、他の並列マイグレーション方法については説明していません。このようなシステム運用方法にはサイト固有の要素が多く、このマニュアルで詳しく説明するのは有効ではないためです。

可能なかぎり、最初に *test* データベースと *development* データベースをバージョン 15.0 にアップグレードしてください。テストの後で運用システムをアップグレードします。テストの詳細については、「第 6 章 安定性とパフォーマンスの確認」を参照してください。

複写使用の並列マイグレーション

方法

複写使用の並列アップグレードを実行する場合、通常は次の方法が使用されます。

1 Adaptive Server 15.0 の新しいコピーをインストールします。

2 リリース 15.0 より前のデータベースをコピー・インまたはロードします。

3 Replication Server を使用してこれら 2 組のデータベースを管理します。15.0 システムがプライマリ・サーバになるので、15.0 より前のシステムはウォーム・スタンバイとして管理します。

フォールバック

Adaptive Server バージョン 15.0 をオフラインにした後ですべてのユーザが前のバージョンのサーバに再接続するように計画します。必要に応じて TCP/IP アドレスとポートを変更してください。

アプリケーション・テスト・スイートの一環としてフォールバックをテストします (これらは、アプリケーションをテストする一連の作業です)。このテスト・スイートでは、次の両方を実行します。

- Adaptive Server バージョン 15.0 にデータを挿入してください。データを複写して前のサーバで使用できるようにする必要があります。
- フォールバック・スクリプトを実行します。

  データベースの bcp ダンプを毎日作成することを検討してください。フォールバックするために、ダンプを前のサーバにロードします。このとき、次の点を考慮してください。

  - インクリメンタル bcp ダンプをサポートするには、データベースの変更が必要な場合があります。
  - 前のサーバでバージョン 15.0 のバックアップ・ファイルを読み込むことはできません。テーブルを 15.0 より前のリリースに戻すには、bcp または他のスクリプトを作成します。

スキーマ拡張機能を適用しません。

ユーザ・データベースのバックアップのスケジューリング方法については、『システム管理ガイド』を参照してください。

その他のヒントを示します。

- 最初に Replication Server をアップグレードします。
- アプリケーションが適切なサーバを使用するようにしてください。interfaces ファイルと *$DSQUERY* 環境変数の詳細については、使用しているプラットフォームの『設定ガイド』を参照してください。
- クライアントのフォールバック時間を計算に含めるようにします。
- 常時稼働のサイトでは、ロード時間の遅延が同期に影響することがあります。Adaptive Server 15.0 に複写してからサーバを切り替えることを検討してください。

アプリケーション・テスト・スイート

このテスト・スイートは、ユーザが新しいシステムを使用する前に実行してください。

複写使用の並列マイグレーションは高可用性アプリケーションに適した方法であるため、更新が正確かどうかとパフォーマンスに問題がないかをテスト・スイートで確認することが不可欠です。

テストの詳細については、「第 6 章 安定性とパフォーマンスの確認」を参照してください。

注意　検証が成功したら、運用と同じクエリをユーザに入力してもらうことを検討してください。テストは、現実的な運用負荷で実行します。テストは、業務時間後や運用休止時間に行うことをおすすめします。

ブリッジング

マイグレーション中は、ユーザがアクセスしないようにする必要があります。検証テストを厳しくすると、マイグレーション中の問題が少なくなります。

更新が正しく行われたかどうかとパフォーマンスに問題がないかを確認するために、複写環境をテストします。

環境

Adaptive Server バージョン 15.0 で使用される環境は、クエリと複写ロードを処理するために、以前のバージョンよりも高い性能が必要です。詳細については、使用しているプラットフォーム用の Replication Server の『設定ガイド』を参照してください。

リリース 15.0 のメモリ要件は、使用している設定に応じて高くなることを考慮してください。詳細については、以下を参照してください。

- 基本的な RAM 要件については、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。
- 「第 5 章 データベース管理の変更」
- メモリとデータ・キャッシュの設定の詳細については、Adaptive Server の『システム管理ガイド』を参照してください。

- パフォーマンスを高めるためのメモリの設定方法の詳細については、『パフォーマンス&チューニング・ガイド』を参照してください。

注意 運用システムの場合は、オフ時間にパフォーマンス・テスト・スイートを実行してください。

スケジューリング

複写機能、検証とパフォーマンスのテスト・スイート、フォールバック・スクリプトを開発して実行するのは、非常に面倒な作業です。実行環境ですでに複写を使用している場合は、作業が大幅に縮小します。

開発システムについては、リリース 15.0 の問題を考慮して開発スケジュールの期間を少し延ばすことをおすすめします。

運用システムについては、必要に応じて延期またはフォールバックしてください。

複写なしのカットオーバ

方法

同時にすべてのデータベースをリリース 15.0 にアップグレードします。複写なしのカットオーバは、一般的に小規模の組織で開発サーバまたは運用サーバに対して使用されます。十分なダウン時間をスケジュールできる場合は、大規模な組織でも使用できます。

フォールバック

前のデータベースをリストアするために必要な時間に基づいてフォールバックを計画します。たとえば、月曜日の午前 8 時にユーザがシステムを必要とし、リストアに 8 時間かかる場合は、検証テストを日曜日の午前 0 時までに終了する必要があります。

注意 クライアントがフォールバックする時間を計算に含めてください。

アップグレードの前に、dump database または bcp out を使用してフォールバックの準備を行うことができます。

カットオーバの後で、フォールバック時に使用するためのトランザクションを取得する方法を計画します。運用開始後にフォール・バックが必要になった場合は、最後のダンプまたはロード後に発生したすべてのトランザクションをリストアする必要があります。

アプリケーション・テスト・スイート

開発システムでは、簡単な検証が適しています。ただし、運用システムでは、更新が正確かどうかとパフォーマンスに問題がないかどうかの両方をテスト・スイートで確認します。

テストの詳細については、「第 6 章 安定性とパフォーマンスの確認」を参照してください。

注意 可能なかぎり、連休となる週末に検証を行うことをおすすめします。

カットオーバ後のフォールバック

データベースの bcp ダンプを毎日作成することを検討してください。こうすることで、ダンプを前のサーバにロードしてフォールバックできます。このとき、次の点を考慮してください。

- インクリメンタル bcp ダンプをサポートするには、データベースの変更が必要な場合があります。
- 前のサーバでリリース 15.0 のバックアップ・ファイルを読み込むことはできません。テーブルを 15.0 より前のリリースに戻すには、bcp または他のスクリプトを作成する必要があります。
- Adaptive Server 15.0 のスキーマ拡張機能を適用しないでください。

  ユーザ・データベースのバックアップのスケジューリング方法については、『システム管理ガイド』を参照してください。

ブリッジング

マイグレーション中は、ユーザがアクセスしないようにする必要があります。検証テストを厳しくすると、ブリッジングの問題が少なくなります。

環境

リリース 15.0 のメモリ要件は、使用している設定に応じて高くなることを考慮してください。詳細については、以下を参照してください。

- 基本的な RAM 要件については、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。
- 「第 5 章 データベース管理の変更」
- メモリとデータ・キャッシュの設定の詳細については、Adaptive Server の『システム管理ガイド』を参照してください。
- パフォーマンスを高めるためのメモリの設定方法の詳細については、『パフォーマンス＆チューニング・ガイド』を参照してください。

注意　運用システムの場合は、オフ時間にパフォーマンス・テスト・スイートを実行してください。

スケジューリング

開発システムについては、リリース 15.0 の問題を考慮して開発スケジュールの期間を少し延ばすことをおすすめします。

運用システムについては、必要に応じて延期またはフォールバックしてください。

段階的カットオーバ

方法

アプリケーションとデータベースを一度に 1 つずつリリース 15.0 に変更します。

フォールバック

データベースの bcp ダンプを毎日作成することを検討してください。フォールバックする場合は、こうすることでダンプを前のサーバにロードできます。このとき、次の点を考慮してください。

- インクリメンタル bcp ダンプをサポートするには、データベースの変更が必要な場合があります。
- 前のサーバでリリース 15.0 のバックアップ・ファイルを読み込むことはできません。テーブルを前のサーバに戻すには、bcp または他のスクリプトを作成する必要があります。
- 変換が成功するまでリリース 15.0 のスキーマ拡張機能を使用しないでください。

  ユーザ・データベースのバックアップのスケジューリング方法については、『システム管理ガイド』を参照してください。

アプリケーション・テスト・スイート

アプリケーション・テスト・スイートで、更新が正確かどうかとパフォーマンスに問題がないかどうかの両方を確認します。また、次の作業も行ってください。

- 両方のリリースのディレクトリとライブラリを維持します。
- アプリケーションが適切なサーバを使用するようにします。
- 検証が成功したら、運用と同じクエリをユーザに入力してもらうことを検討してください。これは、業務時間後や運用休止時間に行うことをおすすめします。

  テストの詳細については、「第 6 章 安定性とパフォーマンスの確認」を参照してください。

ブリッジング

マイグレーション中は、ユーザがアクセスしないようにする必要があります。検証テストを厳しくすると、ブリッジングの問題が少なくなります。

前のサーバでリリース 15.0 のバックアップ・ファイルを読み込むことはできません。テーブルを 15.0 より前のリリースに戻すには、bcp または他のスクリプトを作成する必要があります。

注意 変換が成功するまでリリース 15.0 のスキーマ拡張機能を使用しないでください。

環境

リリース 15.0 のメモリ要件は、使用している設定に応じて高くなることを考慮してください。詳細については、以下を参照してください。

- 基本的な RAM 要件については、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。

- 「第 5 章 データベース管理の変更」
- メモリとデータ・キャッシュの設定の詳細については、『システム管理ガイド』を参照してください。
- パフォーマンスを高めるためのメモリの設定方法の詳細については、『パフォーマンス＆チューニング・ガイド』を参照してください。

その他のヒントを示します。

- 同様の機能を備えたシステムでパフォーマンス測定を実行します。
- 運用システムの場合は、オフ時間にパフォーマンス・テスト・スイートを実行します。

スケジューリング

開発システムについては、リリース 15.0 の問題を考慮して開発スケジュールの期間を少し延ばすことをおすすめします。

運用システムについては、必要に応じて延期またはフォールバックしてください。

アプリケーションまたはデータベースをオフラインにするときは、必ずユーザに通知してください。

マイグレーション・プランの作成

次の情報を含むプロジェクト・プランを作成します。

- マイグレーション方法 － サイトにとってどの方法が最適か。
- フォールバック － マイグレーションが失敗した場合に何を行うか。作成するプランはサイト固有ですが、一般的な問題については「第 6 章 安定性とパフォーマンスの確認」を参照してください。
- アプリケーション・テスト・スイート － 問題がないことを確認するためにどのような検証とパフォーマンスのテストを実行するか。ガイドラインについては、「第 5 章 データベース管理の変更」を参照してください。
- ブリッジング － マイグレーション時にユーザへの影響を最小限に抑える方法。「第 2 章 環境の文書化」で収集したビジネス要件を参照してください。
- 環境 － 必要な追加リソースと環境の変更。「第 2 章 環境の文書化」で収集した情報に基づいて決定します。
- スケジューリング － マイグレーションにどの程度の時間がかかるか。複雑さのレベルと業務のニーズに基づいて推定します。「第 2 章 環境の文書化」で収集したビジネス要件を参照してください。

さらに、必要に応じてマイグレーション・プランに以下を含めます。

- 作業を時系列に並べて特定の役割に割り当てた作業内訳。「付録 B サンプル・マイグレーション作業リスト」にサンプルがあります。
- アプリケーション変更の仕様。アプリケーションの必要な変更の詳細については、「第 4 章 必要なアプリケーションの変更」を参照してください。

Adaptive Server 環境の構築

システムに最適なマイグレーション方法を決定したら、Adaptive Server リリース 15.0 用の環境の準備を開始します。

- ハードウェア・リソースの更新
- オペレーティング・システムのバージョンと EBF レベルの確認
- Adaptive Server と他の Sybase 製品の相互運用性の確認
- Sybase 製品ダウンロード・センタの使用
- ライセンス環境の実装
- アプリケーションとシステム管理プロシージャの更新
- マイグレーション・スクリプトの作成
- テスト環境の作成。テスト環境の詳細については、「第 6 章 安定性とパフォーマンスの確認」を参照してください。

ハードウェア・リソースの更新

選択したマイグレーション方法に応じてハードウェア・リソースのニーズを評価します。たとえば、複写使用の並列マイグレーションの方法を使用する場合は、セカンダリ・システムをインストールするための追加ディスク領域、高可用性環境のための Replication Server、フォールバック方式のためのディスク・ファーム、追加メモリが必要になることがあります。

使用しているプラットフォームのリリース・ノートに、ハードウェアの稼働条件に関する情報が記載されています。

お使いのプラットフォームの『インストール・ガイド』に記載されている物理メモリの要件を参照します。メモリ要件の詳細については、「第 5 章 データベース管理の変更」を参照してください。

メモリとバックアップの詳細については、Adaptive Server の『システム管理ガイド』を参照してください。

オペレーティング・システムのバージョンと EBF レベルの確認

オペレーティング・システムをチェックし、Adaptive Server リリース 15.0 を実行するための適切なバージョンとレベルであることを確認します。最新のバグ修正が適用されるように最新のオペレーティング・システム・パッチをインストールしてください。

推奨される Sybase EBF (バグ修正) の更新情報については、Adaptive Server Migration Resources の Web ページ (http://sybase.com/support/techdocs/migration) を参照してください。

注意 オペレーティング・システムのアップグレードを実行する必要がある場合は、マイグレーションの前に行ってください。無関係なエラーがマイグレーション・プロセスで発生しないように、新しいシステムをテストして正しく動作することを確認します。

Adaptive Server と他の Sybase 製品の相互運用性の確認

サイトで使用している他の Sybase 製品のバージョンが Adaptive Server 15.0 と互換性を持つようにするには、使用しているプラットフォームのリリース・ノートで製品とプラットフォームの相互運用性のセクションを参照してください。

Adaptive Server 15.0 のシステムの変更 (特にクエリ・プランの出力) に伴い、Adaptive Server 15.0 へのマイグレーションでは DBExpert リリース 15.0 を使用する必要があります。

Sybase 製品ダウンロード・センタの使用

以前は、Sybase ソフトウェアは CD からインストールするのが普通でした。しかし、電子的なソフトウェア配布の普及に伴って、Adaptive Server 15.0 を Sybase 製品ダウンロード・センタ (SPDC) からダウンロードするユーザが大半になっています。ダウンロードしたソフトウェアは CD イメージの形式です。これを CD に書き込んで、必要に応じてこのイメージからインストールすることができます。インストールの問題を回避するには、次の点を考慮します。

- ダウンロード場所には短いパスを使用します (たとえば */sybase/downloads* や *c:¥sybase¥downloads*)。パスにはスペース文字などの特殊文字を使用しないでください。Windows オペレーティング・システムの場合は、*C:¥Program Files¥Sybase¥downloads* などのディレクトリを使用しないでください (“Program” と “Files” の間のスペースが問題です)。不正なパスを使用したり、誤ったバージョンの JRE を使用すると、「クラスが見つかりません」というインストール・エラーが発生します。

- Adaptive Server には、InstallShield ユーティリティに合った適切な JRE が付属しています。JDK や JRE など、他の Java 環境をインストールしている場合で、これらの場所を参照する環境変数がある場合 (たとえば *JAVA_HOME* がある場合や、頻繁にコンパイルする目的で *path* 環境変数に *java* を指定している場合) には、インストールを開始するシェルでこれらの環境変数を無効にしてください。
- UNIX システムで CD イメージを展開する場合には、GNU の gnuzip ユーティリティを使用します。これらのイメージは GNU の zip ユーティリティで圧縮されており、UNIX の標準の圧縮ユーティリティで展開すると、イメージが正しく展開されない場合があります。
- CD イメージのアーカイブ ファイルを抽出するには、GNU の tar ユーティリティを使用します。UNIX の標準の tar ユーティリティを使用すると、ファイルが正しく抽出されない場合があります。
- 多くのハードウェア・ベンダでは、自社の CD ドライブに対する mount で、さまざまなオプションをサポートしています。インストール・ガイドを参照して、使用しているプラットフォームに適合する正しい mount オプションを使用してください。

ライセンス環境の実装

米国政府がサーベンス・オクスリー法を施行する以前から、ライセンス管理とレポートの強力な機能を実装するよう求める声が、多くのお客様から寄せられていました。サーベンス・オクスリー法によって、この重要性はいっそう高まりました。ソフトウェア資産の悪用を含む経済的な不正行為の責任が、企業の CEO に及ぶことになったためです。特に監視の目が厳しい業界の 1 つが銀行業界であること、そして Sybase が金融業界で置かれている立場をふまえて、Sybase は、より厳格なライセンス管理機能を導入しました。

サーベンス・オクスリー法はすべての株式公開企業に適用されます。そこで、SySAM バージョン 2.0 は、ライセンスの法規遵守に企業ユーザが対応しやすいように設計されています。SySAM 2.0 の重要な構成要素の 1 つがレポート機能です。ライセンスの法規遵守について、IT 管理者が監視および企業幹部向けにレポートできるという機能です。ライセンスの使用状況が SySAM ソフトウェアから Sybase にレポートされることはありません。ライセンス契約の範囲を超えて Sybase ソフトウェアを長期にわたって使用する場合は、Sybase の営業担当者との間で調整が必要になります。

SySAM 2.0 の全機能については、『Sybase® ソフトウェア資産管理 2.0 ユーザーズ・ガイド』および使用しているプラットフォーム用の設定ガイドを参照してください。

ライセンス・サーバの数

展開するライセンス・サーバの数は、ハードウェアの地理的な分散状況や、プロジェクトやビジネス単位の区分をどのように処理するかに応じて決まります。広い範囲に分散している組織の場合は、通常は各地域に少なくとも 1 つのライセンス・サーバを配置します。SySAM では複数のライセンス・サーバをサポートできますが、ライセンス・サーバにかかる負荷はきわめて小さいため、複数のライセンス・サーバは技術的には必要ではありません。

ライセンスあたりのサーバ数

SySAM では、ライセンス・サーバがいつでも確実に利用できるように、冗長なライセンス・サーバ・ノードを作成することができますが、これが必要なのは、1 つのライセンス・サーバが長期間にわたって利用できない場合 (たとえば 30 日の猶予期間など) に限られます。Adaptive Server 12.5 では、お客様が各サーバに対するライセンス処理をそれぞれ個別に行いました。Adaptive Server 15.0 では、サーブド・ライセンス (集中化されたロケーションから提供されたライセンスのことで、ネットワーク・ライセンスともいう) のモデルを使用している場合、ライセンスは一元管理されます。サーブド・ライセンス・モデルでは、ライセンス・サーバ・ノードを Sybase に登録するのみで、個々のサーバは登録しません。ライセンス・サーバ・ノードの ID を Sybase に登録する必要があるのは、レポートとしての要件ではなく、ライセンス・サーバ・ノードの ID を使用してライセンス・キーを生成するという理由からです。

独自のイメージを配備する場合

お客様によっては、Sybase 製品のインストール用の配備イメージを独自に作成する方法を選択することがあります。内部的に動作確認した Adaptive Server バージョンとパッチ・レベルを使用し、一連のソフトウェアを “tarball” などの電子的なパッケージに取り込むという方法です。この方法を使用して複数のライセンス・サーバを実装する場合、配備イメージに対して唯一加える変更は、イメージを配備した後で、Adaptive Server がライセンス処理のためにアクセスするライセンス・サーバを指定する必要があるという点だけです。グローバルなインストールでは、配備イメージごとにこの情報を含めることができます。ビルドが異なると、通常はローカライゼーションの要件も異なるためです。ただし、配備されたライセンス・ファイルの先頭行で、ライセンス・サーバの *host_name* を次のように変更する必要があります。

```
SERVER my_license_srvr 00096b138c70
VENDOR SYBASE
USE_SERVER
```

リモート・サーバを使用する場合

リモート・サーバ (特に、ファイアウォールの外部にあるサーバ) を使用する場合、ライセンス・サーバをリモートで配備できますが、その方法には特にメリットはなく、実用的ではありません。代わりに、ライセンス・キーをローカル・ファイルから読み込むアンサーブド・ライセンス・モデルを使用します。アンサーブド・ライセンスでは、各 *hostID* を Sybase に登録する必要があります。OEM の Adaptive Server 配備では、埋め込みシステムには別個のライセンスの実装を使用します。Sybase では、こうしたシステムでアンサーブド・ライセンスを使用する必要性は多くないものと想定しています。

別個にライセンスされるオプションへの対応

以前のバージョンの Adaptive Server では、Java、XML、および XFS (外部ファイル・システムまたはコンテンツ管理) のオプションについて、お客様がライセンス処理を行いました。Adaptive Server バージョン 15.0 には、これらのオプション用のライセンスが含まれています。その他のオプション (高可用性や DTM など) については、別個のライセンスが必要です。Adaptive Server 15.0 を起動したときに、これらのライセンスが含まれているかどうかが自動的にチェックされます。

Developer's Edition の内容

Adaptive Server Developer's Edition には、Sybase が提供するすべての非ロイヤルティ対象オプションが含まれています。ロイヤルティ対象オプションには、拡張型全文検索、Real Time Data Services、および大半のツール (DBXray や Sybase Database Expert など) が含まれます。Developer's Edition のインストール時に、ライセンス・モードの選択を求められたときには、アンサーブド・ライセンス・モデルを選択してください。Developer's Edition ではファイル・ベースのライセンス方式を使用するためです。Developer's Edition では、製品 CD のイメージにキーが含まれているため、ソフトウェアをインストールした後で SySAM によるアクションは必要ありません。

その他の情報

SySAM に関するその他の情報や疑問点については、http://www.sybase.com/sysam を参照してください。または、Sybase 製品の保守契約を結んでいるサポート・センタにお電話いただくか、最寄の Sybase 営業担当者にお問い合わせください。

アプリケーションとシステム管理プロシージャの更新

このマニュアルの次の各章で、Adaptive Server 15.0 のアップグレードの問題について説明しています。

- 「第 4 章 必要なアプリケーションの変更」
- 「第 5 章 データベース管理の変更」

Adaptive Server 15.0 へのマイグレーションでは、システムの問題や予期しない処理結果をもたらす変更がないかどうか、現在のアプリケーションとシステム管理プロシージャを見直す必要があります。

これは、マイグレーション準備の中で最も時間がかかる作業です。ニーズとリソースに応じて最適な方法を選択してください。次のような方法があります。

- アプリケーションを調べて変更するための独自のスクリプトを作成する。
- 15.0 のサーバに関する予約語チェックを実行し、ストアド・プロシージャ sp_checkreswords、sp_procqmode などのツールを使用する。これらのストアド・プロシージャの詳細については、『リファレンス・マニュアル：プロシージャ』を参照してください。
- preupgrade ユーティリティを使用して一般的なアップグレードの問題をチェックします。詳細については、『ユーティリティ・ガイド』を参照してください。

マイグレーション・スクリプトの作成

「第 1 章 ビジネス要件の文書化」の説明に従って、既存のスクリプト、作成したスクリプト、またはリバースエンジニアリングしたスクリプトを使用し、独自の 15.0 Adaptive Server インストールを作成するスクリプトを作成または編集します。次のスクリプトが必要です。

- 新しい Adaptive Server 環境 (データベース・デバイス、設定、ログイン、セキュリティなど) を作成するサーバ・レベルのマイグレーション・スクリプト
- Adaptive Server のユーザ・データベースとデータベース・オブジェクト (テーブル、ビュー、インデックス、トリガ、グループ、ユーザ、パーミッションなど) を作成するデータベース・レベルのマイグレーション・スクリプト

第 4 章

必要なアプリケーションの変更

この章と「第 5 章 データベース管理の変更」では、技術的な問題をアプリケーション開発者に関係するものとデータベース管理者に関係するものに分けて説明します。ただし、多くの問題は一方に限定されないため、役割にかかわらず両方の章を読むことをおすすめします。この章では、アプリケーションの実行に影響する問題やコーディングの変更が必要な問題について説明します。

この章では、システム管理者にとって予期しない動作を引き起こす可能性のある新機能と変更点についてのみ説明します。過去のいくつかのリリースを通じて、Adaptive Server にはこれら以外にも多くの変更が加えられています。変更点と新機能の全リストについては、『Adaptive Server Enterprise 新機能ガイド』を参照してください。

注意 既存のアプリケーションとシステム管理プロシージャのテストおよび変更は、マイグレーション準備の中で最も時間がかかる作業です。このマニュアルでは、これらの変更を行う方法については説明しません。各自のニーズやリソースに合った変更方法を選択してください。たとえば、独自のプランやスクリプトを開発したり、Sybase に相談することもできます。

Adaptive Server Migration Resources Web ページには、マイグレーションに関する TechNote とホワイトペーパーがあります。(http://sybase.com/support/techdocs/migration) また、Sybase 製品の保守契約を結んでいるサポート・センタから、バグやエラー状態などの技術的な問題に関する支援を受けることができます。

現在使用している Adaptive Server のバージョンに関する説明が記載されている最初の項からお読みください。

- Adaptive Server リリース 11.5 のアップグレード
- バージョン 11.9.x からのアップグレード
- リリース 12.0 からのアップグレード
- リリース 12.5 からアップグレードする場合

予約語

予約語は Adaptive Server でのみ使用できます。Adaptive Server に接続しているアプリケーションでは使用できません。多くの場合、Adaptive Server のバージョンが新しくなるたびに、新しいオブジェクトとコマンドに応じて新しい予約語が追加されます。

Adaptive Server をアップグレードする前に、予約語が含まれるすべてのオブジェクト名を変更してください。プロシージャ、Transact- SQL スクリプト、アプリケーションでもこれらの名前を変更してください。変更しないと、アップグレードしたサーバで実行できません。オブジェクト名をチェックするには、アップグレードを開始せずに実行できる sqlupgrade で予約語チェックを使用します。または、古い Adaptive Server に対して *$SYBASE/ASE-15_0/scripts* ディレクトリの *installupgrade* スクリプトを適用し、この Adaptive Server で sp_checkreswords ストアド・プロシージャを実行することもできます。

注意 オブジェクトに含まれる予約語を検出する Sybase プロシージャで、スクリプトとアプリケーションに含まれる予約語を検出することはできません。スクリプトとアプリケーションのチェックは別に行います。

予約語を含む名前は、sp_rename を使用して変更するか、二重引用符で囲むことができます。使用中のプラットフォームに対応するインストール・ガイドの予約語の説明を参照してください。

予約語の完全なリストについては、『リファレンス・マニュアル：ビルディング・ブロック』を参照してください。各リリースで追加された予約語のリストについては、『Adaptive Server Enterprise 新機能ガイド』を参照してください。

Adaptive Server リリース 11.5 のアップグレード

Adaptive Server リリース 15.0 にアップグレードできるのは、Adaptive Server リリース 11.9.x、12.0.x、または 12.5.x だけです。

現在 Adaptive Server リリース 11.5 を実行している場合は、先にリリース 12.0 にアップグレードしてから 15.0 にアップグレードすることをおすすめします。

バージョン 11.9.x からのアップグレード

この項では、次の点について説明します。

- ANSI ジョイン
- クエリ処理の変更

ANSI ジョイン

Adaptive Server リリース 12.0 以降では、ANSI ジョインがサポートされています。ANSI 外部ジョインによって、述語を含むのが on 句なのか where 句なのかがはっきりと指定されるので、ANSI 外部ジョインを使用するようにアプリケーションを書き換えることをおすすめします。前のリリースの Transact-SQL 構文では、はっきりと指定されない場合がありました。

ANSI 構文では、次のいずれかのジョインを作成できます。

- 内部ジョイン ― ジョイン・テーブルに on 句の条件を満たす内部テーブルと外部テーブルのローだけが含まれます。内部ジョインを含むクエリ結果セットは、on 句の条件を満たさない外部テーブルのローの null 入力ローを含みません。ANSI 内部ジョインの構文は次のとおりです。

  ```
  select select_list
  from table1 inner join table2
  on join_condition
  ```

  次に例を示します。

  ```
  select au_id, titles.title_id, title, price
  from titleauthor inner join titles
  on price > 15
  ```

- 外部ジョイン － on 句の条件を満たすかどうかに関係なく、外部テーブルのすべてのローがジョイン・テーブルに含まれます。ローが on 句の条件を満たさない場合、内部テーブルからの値が null 値としてジョイン・テーブルに格納されます。ANSI 外部ジョインの where 句がクエリ結果に含まれるローを限定します。ANSI 構文では、ネストした外部ジョインも作成できます。ANSI 外部ジョインの構文は次のとおりです。

  ```
  select select_list
  from table1 {left | right} [outer] join table2
  on predicate [join restriction]
  ```

  次に例を示します。

  ```
  select au_fname, au_lname, pub_name
  from authors left join publishers
  on authors.city = publishers.city
  ```

クエリ処理の変更

Adaptive Server リリース 12.0 でのクエリ処理とオプティマイザの変更は、11.9.2 での変更に比べると多くありませんが、ほとんどのサイトでパフォーマンスが向上すると考えられます。変更内容は次のとおりです。

- 述部要素変形
- 1 つのクエリで最大 50 個のテーブルのサポート
- 抽象クエリ・プラン
- 最適化時間の増加
- like の最適化の強化
- 具体的 ID

Adaptive Server リリース 12.0 で導入された変更の中には、Adaptive Server リリース 15.0 で導入された変更によって置き換えられているものがあります。これらの変更については、『Adaptive Server Enterprise 新機能ガイド』を参照してください。

Adaptive Server の最近のリリースにおけるオプティマイザの変更点については、『An Introduction to Sybase Adaptive Server Enterprise's Modern Optimizer』(http://my.sybase.com) を参照してください。これは、ISUG Technical Journal で発行されたオプティマイザに関する記事のコレクションで、ダウンロードが可能です。読むためには Acrobat Reader が必要です。

述部要素変形

述部要素変形を使用すると、アクセス・パスが制限されているクエリ (テーブルのローを修飾する探索引数、ジョイン、または or 句が使用される可能性が非常に少ないクエリ) のパフォーマンスが大幅に向上します。

さらに最適化するためには、ジョイン条件、検索句、最適化が可能な or 句に基づいて新しい検索パスを生成します。

述部要素変形は、or (最適化が難しい) で結ばれた述部から最適化できる句を抽出して、and 句 (最適化しやすい) で置き換え、より使用しやすい探索引数をオプティマイザに提供することで、クエリの最適化を促進します。使用可能な探索引数が増えると、オプティマイザが得られる情報が増加して、効果的なプランを選択できる可能性が高くなります。

一部の複雑なクエリでは、完全な直積ジョインは使用されません。

例：

```
select * from lineitem, part
where ((p_partkey = l_partkey and  l_quantity >= 10)
or (p_partkey = l_partkey and l_quantity <= 20) )
```

これが、次のように変換されます。

```
select * from lineitem, part
where ((p_partkey = l_partkey and  l_quantity >= 10)
or (p_partkey = l_partkey and l_quantity <= 20) )
and  (p_partkey = l_partkey)
and  (l_quantity >= 10  or   l_quantity <= 20)
```

等位項を追加することで、オプティマイザで使用可能な探索引数が増加します。述部要素変形によって追加された新しい等位項 (and 句) は、インデックス・アクセス方式 (フィルタ処理の方式) として役立たないと判断されると使用されません。

クエリのセマンティックは変わりません。結果セットも同じです。述部要素変形を無効にすることはできません。これはユーザに対して完全に透過的に行われます。コンパイラの新しいフェーズとして、オプティマイザが開始する直前に実装されます。

述部要素変形が実装されると、トレース・フラグ 302 の出力に追加のコスト・ブロックが表示され、showplan では追加の “Keys are” メッセージが表示されます。

1 つのクエリで最大 50 個のテーブルのサポート

Adaptive Server リリース 12.5.1 以降では、最大 50 個のユーザ・テーブルと 14 個のワーク・テーブルがサポートされています。11.9.2 では 16 個のユーザ・テーブルと 12 個のワーク・テーブルでしたが、これが拡張されました。ただし、参照整合性チェック数の制限は 192 のままであり、使用できるサブクエリの数も 16 個のままです。

注意　Adaptive Server リリース 15.0 では、1 つのクエリで 46 個のワーク・テーブルがサポートされています。

抽象クエリ・プラン

Adaptive Server リリース 12.0 で導入された抽象プランは、次のようなクエリ・プランの変更によるサーバの全体的なパフォーマンスへの影響を防ぐために、システム管理者やパフォーマンス・チューニング担当者が使用できる手段です。

- オプティマイザの選択結果に影響する Adaptive Server ソフトウェアのアップグレード
- クエリ・プランを変更するような新しい Adaptive Server 機能
- 並列度、テーブル分割、またはインデックスなどのチューニング・オプションの変更

Adaptive Server は、sysqueryplans という名前のシステム・テーブルにあるクエリ・テキストを取得したり、このテーブルにクエリの抽象プランを保存したりできます。高速なハッシュ・メソッドを使用すると、受信した SQL クエリと保存されているクエリ・テキストを比較し、一致するものがあった場合、保存されている抽象プランを使用してクエリが実行されます。

抽象クエリ・プランの詳細については、『パフォーマンス&チューニング・ガイド：オプティマイザと抽象プラン』を参照してください。

アップグレード後のパフォーマンスの問題を解決するための抽象クエリ・プランの使用方法については、TechNote『Using and Maintaining Abstract Query Plans』(http://my.sybase.com) を参照してください。

最適化時間の増加

多数のジョイン・キーを含むクエリは、Adaptive Server での最適化の時間が長くなることがあります。

このようなクエリの最適化にかかる時間を短縮する必要がある場合は、クエリの抽象プランを使用することを検討してください。

注意 Adaptive Server バージョン 15.0 では、これらのクエリのパフォーマンスが向上しています。

---

like の最適化の強化

バージョン 12.0 で、探索引数にマイグレートされない like 句のコスト計算が変更され、like 句の先頭にワイルドカードを含むクエリについて、より正確なコスト見積もりを生成する手法が使用されるようになりました。

このために、選択性の見積もりが向上し、より適切なクエリ・プランが生成されるようになりました。以前は、like 句の先頭にワイルドカードが含まれていると、一致データを検索する方法がなかったため、カラムのすべてのローを修飾する (選択性 1.0) と見積もられていました。これは、このような句のコスト計算としては正確な方法ではありませんでした。

次に、バージョン 12.0 サーバのクエリの例を示します。

```
select ... from part, partsupp, lineitem
where l_partkey = p_partkey
and l_partkey = ps_partkey
and p_title like '%Topographic%'
```

like 文字列は、ヒストグラム・セルの境界と比較されます。

パターン一致検索によって like 条件がセル境界で検出されると、一致していると判断されます。

パターン一致が検出されないと、選択性は 1 ／ヒストグラムのステップ数と見なされます。デフォルトの 20 セルを使用すると、ヒストグラム境界値でパターン一致がない場合、選択性は 0.05 になります。

パターン一致がセル境界で検出されると、選択性の見積もりは、パターン一致を含むすべてのセルのウェイトの合計になります。

いずれの場合も、選択性の見積もりは 12.0 より前のバージョンよりも正確です。

この方法は、`like "_abc"` または `like "[ ]abc"` のような like 句を含むクエリにも適用されます。

具体的 ID

Adaptive Server では、具体的 ID を使用して、プロシージャ、ビュー、トリガ、およびそれらが他のデータベースで参照するオブジェクトの間にある所有権の連鎖を検証できます。ユーザがオブジェクトを作成すると、所有者のデータベース・ユーザ ID (UID) とオブジェクト作成者のログイン名の両方が、sysobjects 内のオブジェクトに関連付けられます。この情報によって、どのユーザが所有するオブジェクトであるかが具体的に識別されるため、Adaptive Server は、いつオブジェクトのパーミッションを暗黙的に許可できるかを認識できます。これがデータベース間のクエリを使用する古いアプリケーションに影響を及ぼし、パーミッション・エラーが発生することがあります。

具体的 ID の詳細については、『システム管理ガイド 第 1 巻』の第 17 章「ユーザ・パーミッションの管理」を参照してください。

リリース 12.0 からのアップグレード

この項では、マイグレーションに影響する Adaptive Server バージョン 12.0 での次の変更点について説明します。

Transact-SQL の変更

一般的に、既存の Transact-SQL コマンドの変更はオプションの追加であり、変更されたコマンドが使用されていても、アプリケーションで問題が起きることはありません。変更された Transact-SQL コマンドのリストについては、『Adaptive Server Enterprise 新機能ガイド』を参照してください。

enable xact coordination 設定パラメータ

Adaptive Server リリース 12.0 で enable xact coordination 設定パラメータが導入されました。Adaptive Server バージョン 12.5 では、このパラメータがデフォルトで有効になります。

enable xact coordination の詳細については、『システム管理ガイド 第 1 巻』の第 5 章「設定パラメータ」を参照してください。

select 文の式の数の上限

Adaptive Server バージョン 12.5 には、select 文で使用できる式の数に明示的な上限はありません。この数は、使用可能なシステム・メモリによってのみ制限されます。

ワイド・カラムとデータ・トランケーション

Adaptive Server 12.5 で、ページとカラムのサイズが拡張されました。このために、以前はデータ・トランケーションが行われていた既存のアプリケーションで、予期しない結果が発生する可能性があります。また、拡張されたデータ・サイズを使用する場合は、クライアント・ソフトウェアが新しいサイズを処理できるかどうかを考慮してください。この項では、次の項目の概要について説明します。

- トランケーション動作の変更
- ワイド・カラムとオプティマイザの統計
- col_length() と datalength() でのワイド・カラム

トランケーション動作の変更

12.5 より前のバージョンの Adaptive Server では、カラムの最大長は 255 バイトでした。Adaptive Server では、サーバで使用する論理ページ・サイズによって異なりますが、最大 16294 バイトの char、varchar、binary、varbinary データを使用してカラムを作成できます。このため、以前のバージョンで 255 文字にトランケートされていたデータがトランケートされません。このようなトランケーションに依存していたアプリケーションでは、受け取る結果セットが正しくなくなることがあります。次の例では、col1 と col2 はそれぞれ 200 文字、col3 は 255 文字です。

```
select * from t1 where col1 + col2 = col3
```

col1 と col2 がそれぞれ 200 文字の場合、連結すると 400 文字の文字列になります。以前のバージョンの Adaptive Server ではこれが 255 文字にトランケートされるため、col1 と col2 の合計が col3 と一致します。ただし、Adaptive Server リリース 12.5 では、col1 と col2 を連結すると 400 文字になるため、255 文字の col3 とは一致しません。

文字型とバイナリ型は、最大 16384 バイトの結果を生成できます。この長さを超えるデータはトランケートされます。

ワイド・カラムとオプティマイザの統計

ワイド・カラムに統計を適用するとき、Adaptive Server はデータの先頭の 255 バイトだけを使用します。たとえば、`char(500)` として作成されたカラムにインデックスを設定した場合でも、先頭の 255 バイトだけが sysstatistics のカラム・ヒストグラムで使用されます。unichar のカラムでは、カラムの分散を集計するために先頭の 127 の chars が使用されます。

col_length() と datalength() でのワイド・カラム

col_length() コマンドと datalength() コマンドは、データベースの情報を返す Transact-SQL の組み込み関数です。現在、これらの関数は 225 を超える値を返すことができます。クエリでこれらの関数のいずれかを使用して、結果を変数に代入する場合は、値を保持できる十分なサイズの変数を用意してください。

リリース 12.5 からアップグレードする場合

この項では、次の項目について説明します。

日付と時刻のデータ型

12.5.1 より前のバージョンの Adaptive Server には、日付と時刻のデータ型が datetime と smalldatetime しかありませんでした。Adaptive Server 12.5.1 で、date および time データ型が追加されました。アプリケーションでこれらのデータ型を使用するテーブルにアクセスする場合は、スクリプトとアプリケーションでこれらのデータ型を考慮してください。

クエリとオプティマイザの変更

Adaptive Server 15.0 では、オプティマイザが新しくなっています。このオプティマイザで実行される多くのクエリは、ほとんど変更なしに以前と同等かより効率的に動作するはずです。ただし、サーバを運用環境で使用する前に、以下の項目についてすべてのアプリケーションをテストしてください。

- パーサの変更により、一部のクエリでは、`Syntax error at line #` (メッセージ 156) の代わりに、一般の構文エラー (メッセージ 102) が返される場合があります。
- クエリごとのワーク・テーブルの最大数が 14 から 46 に増加しました。

- クエリで order by 句を明示的に使用しない場合、Adaptive Server 15.0 では結果セットの順序が以前のバージョンと次のように異なります。
  - 以前のリリースでは、order by 句なしでも、group by 句を使用したときに結果セットがソート順で返されていました。グループ化アルゴリズムでクラスタード・インデックスを持つワーク・テーブルが作成され、グループ化はこのワーク・テーブルへの挿入に基づいて行われました。
  - Adaptive Server 15.0 では、グループ化アルゴリズムでハッシュベース方式が使用されており、この方式ではソートされた結果セットが生成されません。ソートされた結果セットを生成するには、クエリを変更して order by 句を追加します。この変更は ANSI SQL 標準に準拠しています。
  - Adaptive Server 15.0 には、ハッシュまたはマージベースの union、ハッシュまたはソートベースの distinct など、関係演算用の代替アルゴリズムがあります。order by を使用しないクエリの結果セットの順序は、実行プランによって決まります。
- クエリ処理エンジンがクエリを最適化するためにより多くの方法を探すため、Adaptive Server 15.0 ではクエリのコンパイル時間が増加しています。ただし、よりコストの低いプランがある場合に最適化時間を短縮できる新しいタイムアウト・メカニズムが検索エンジンに含まれています。
- Adaptive Server 15.0 では enable sort-merge join and JTC 設定オプションが非推奨となっているため、マージ・ジョインを使用するクエリ・エンジンに予期せず遭遇することがあります。クエリ・エンジンで以前のバージョンの動作を強制的に使用することもできますが、新しいメモリ内ソート・オペレーションによって sort merge join のパフォーマンスが向上します。クエリ・オプティマイザは、適切でない場合にはマージ・ジョインを使用しません。sort merge join は、set merge_join 0 コマンドを使用してセッション・レベルで無効にすることも、allrows_oltp 最適化目標を使用して無効にすることもできます。
- Adaptive Server 15.0 では、set store_index 0 コマンドを使用して、再フォーマットをセッション・レベルで無効にすることができます。

  Adaptive Server でログイン・トリガを使用して、アプリケーションを変更せずにセッション・レベルの設定を変更する方法の詳細については、『システム管理ガイド 第 1 巻』の第 17 章「ユーザ・パーミッションの管理」を参照してください。
- Adaptive Server 15.0 では、互換性はあるが異なるデータ型の間のジョインを構成するアルゴリズムが強化されています。探索引数で異なるデータ型が使用されている場合でも、インデックスを使用できます。ただし、トレース・フラグ 291 は使用しないでください。疑似の誤った結果セットが返されます。

- Adaptive Server 15.0 の並列クエリでは set statistics io がサポートされませんが、非並列クエリではサポートされます。

  set statistics plancost on を設定すると、統計の正確な解釈を取得できます。Lava 演算子ツリーの並列アクセス・ノードが表示されます。

Adaptive Server 15.0 でのクエリ処理の詳細については、『Adaptive Server におけるクエリ処理 (QP) について』を参照してください。

サポートされないトレース・フラグ

トレース・フラグは Adaptive Server の動作に影響を与え、問題を修正します。ただし、一部のトレース・フラグは予期しない動作を示し、すべてのトレース・フラグに注意が必要です。多くのトレース・フラグは Adaptive Server リリース 12.5.x に固有のもので、Adaptive Server 15.0 には必要ありません。表 4-1 は、よく使用されるいくつかのトレース・フラグと、それらの Adaptive Server 15.0 との関連性を示します。トレース・フラグの使用を中止する前に、Sybase 製品の保守契約を結んでいるサポート・センタに確認してください。ここに示す以外の目的でトレース・フラグが使用されている場合があるからです。

表 4-1: よく使用されるトレース・フラグ

トレース・フラグ	説明	12.5.x での使用	15.0 での使用	補足説明
291	有効にすると、`col1 <relop> fn(col2)` 形式 (col2 のデータ型が col1 のデータ型より上位にリストされる形式) を使用する述部は、式 `f(col2)` を下位のタイプ (この例では col1) のデータ型にキャストする。	使用する	使用しない	機能は Adaptive Server 15.0 に実装済み
333	min-max 最適化を無効にする。	使用する	使用しない	サポート終了
364	総密度の代わりに範囲密度を使用する。	使用する	使用しない	サポート終了
370	単一テーブルのクエリのテーブル・スキャンの代わりに min-max インデックスを使用する。ジョインの集合の最適化は行わない。	使用する	使用しない	サポート終了
396	単一テーブルのクエリに min-max 最適化を使用する。	使用する	使用しない	サポート終了
526	showplan が有効なときに、セミグラフィカルな実行オペレータ・ツリーを出力する。	使用しない	使用する	機能は Adaptive Server 15.0 に実装済み

ここに示されていないトレース・フラグを使用している場合は、Sybase 製品の保守契約を結んでいるサポート・センタに相談してください。

Adaptive Server 15.0 では、300 番台の診断トレース・フラグ (302、305、308、310、311 など) が廃止されており、今後のリリースでは非推奨となります。これらは、より精度の高い診断と読みやすい出力を提供する showplan オプションに置き換わります。

オプティマイザの判断を調査するためのツールについては、『クエリ・プロセッサ』の「第 1 章 Adaptive Server におけるクエリ処理 (QP) について」を参照してください。

パーティションの変更

Adaptive Server の以前のリリースで分割されたすべてのテーブルは、アップグレード処理中に単一のパーティションに戻されます。データの選択、挿入、削除に使用するコマンドは (テーブルが分割されているかどうかにかかわらず) Adaptive Server リリース 15.0 でも同じですが、プロシージャのパフォーマンスは分割しているテーブルによって異なることがあります。テーブルが単一のパーティションに戻された場合は、テーブルを再分割します。

デフォルトでは、Adaptive Server はラウンドロビン方式を使用してテーブルを分割します。Adaptive Server 15. 0 では、テーブルを分割するための次の方式が追加されています。

- リスト
- 範囲
- ハッシュ

パーティションの詳細については、『Adaptive Server Enterprise 新機能ガイド』と『Transact SQL ユーザーズ・ガイド』を参照してください。

この項では、パーティション使用時の予期しない動作のいくつかの制限について説明します。

パーティションとプライマリ・キーとユニーク・インデックス

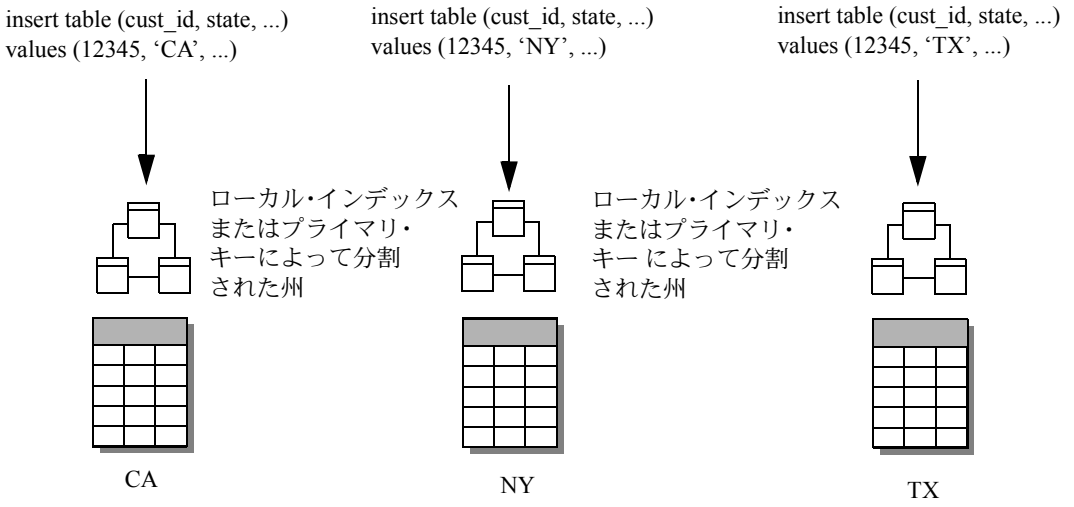
分割キーがローカル・インデックスまたはプライマリ・キー・カラム (またはそのサブセット) と同じでない場合は、プライマリ・キーを含むユニーク・インデックスを強制できません。

ほとんどのユーザはテーブルの分割を検討します。テーブルを分割すると、並列クエリ機能をより効率的、実用的に使用できるようになることや、データベース管理 (DBA) タスクが容易になり、時間と費用の節約になることがその理由です。

DBA タスクに関しては、分割によって予期しない動作が発生する場合があります。ユーザは一般に、日付または曜日 (たとえば、オフセットとして計算されたモジュロ曜日番号) に基づいて DBA タスクのテーブルを分割します。これは通常、古いデータをより高速にアーカイブするために行われます。しかし、場合によってはこの日付がプライマリ・キーにないことや、プライマリ・キーの唯一のカラムであることがあります。州または国とユニーク・キーを使用するなど、テーブルがより低いカーディナリティ除法に従って分割されている場合にも同じ問題が発生します。

たとえば、次のテーブルには米国内の顧客 (cust_id によってユニークに識別される) が含まれており、顧客は販売地域別に分類されています。テーブルは州によって分割されています。

図 4-1: ローカル・インデックスまたはプライマリ・キーを持つ分割されたテーブル



州によって分割されたテーブル

同じ cust_id 値 ("12345") をテーブルに挿入しているにもかかわらず、insert は正常に実行されます。データを別々の state パーティションに挿入しているからです。インデックス・パーティションは独立して機能するため、値を挿入するときに、その値が別のパーティションにすでに存在するかどうかはわかりません。このため、プライマリ・キーを含まないカラム・リストでテーブルを分割したり、分割キーに使用されていないカラムでユニークなローカル・インデックスを作成しようとすると、Adaptive Server が警告を表示します。

パフォーマンス上の理由から、Adaptive Server はユニーク性を強制しません。たとえば、5000 万行のローを持つテーブルでは、インデックスのルート・ノードからデータ・リーフ・ノードを検索するために、プライマリ・キーとノンクラスタード・インデックスにおよそ 7 レベルのインデックス設定が必要です。このテーブルを分割し (値が均等に分散されると仮定)、各パーティションのロー数を 100 万にした場合、5 レベルのインデックス設定が必要です。分割されていないテーブルでは、ユニークな値のチェックで挿入ポイントまで読み込んで、その値を持つローがすでに存在するかどうかを判断するまでにわずか 7 I/O で済みます。しかし、分割されたインデックスの場合は、値のチェックで 50 個すべてのパーティションについて、5 つのすべてのレベルをスキャンするため、合計で 250 I/O が必要になります。

対処方法としては、ローカル・インデックスまたはプライマリ・キー制約 (すべてのプライマリ・キー制約はテーブル・スキーマに従って分割されます) の代わりに、グローバル・ユニーク・インデックスを作成してユニーク性を強制します。グローバル・インデックスは分割されないので、ユニーク性を強制できます。

複数および複合分割キーと範囲分割

範囲分割とハッシュ分割では、分割キーとして最大 31 個のカラムを指定し、複合分割キーを作成できます。ハッシュ・パーティションの場合、分割キーは予期したとおりに動作し、個々のデータ・ローを異なるパーティションに分散する方法を決定します。しかし、範囲パーティションの分割キーは、特に分割キーが数値の場合は予期しない動作をすることがあります。予期しない動作が発生する原因は、Adaptive Server が毎回すべての分割キーを使用してローの格納先を決定する代わりに、適切なパーティションを決定するまでシーケンスで最小限の分割キーしか使用しないためです。次の SQL テキストは、分割キーを決定するためのルールを示します。

```
if key1 < a, then the row is assigned to p1
if key1 = a, then
        if key2 < b or key2 = b, then the row is assigned to p1
if key1 > a or (key1 = a and key2 > b), then
        if key1 < c, then the row is assigned to p2
        if key1 = c, then
            if key2 < d or key2 = d, then the row is assigned to p2
        if key1 > c or (key1 = c and key2 > d), then
            if key1 < e, then the row is assigned to p3
            if key1 = e, then
                if key2 < f or key2 = f, then the row is assigned to p3
                if key2 > f, then the row is not assigned
```

これを要約すると次のようになります。

- *value* が key1 より小さい場合は、現在のパーティションを使用する。
- *value* が key1 と等しい場合は、*value* と key2 を比較する。
- *value* が key1 より大きい場合は、次のパーティション範囲をチェックする。

たとえば、120 万件の顧客情報が含まれるテーブルがあるとします。並列クエリのパフォーマンスを高め、メンテナンスを単純化するためにこのテーブルを分割する場合は、会計四半期と顧客 ID によってテーブルを分割し、必要に応じて四半期ごとにデータをアーカイブすることができます。ただし、テーブルに月と会計四半期のカラムも含まれる場合、四半期は 3 か月ごとに来るので、次の分割方式を使用できます。

```
alter table telco_facts_ptn
   partition by range (month_key, customer_key)
      (p1 values <= (3, 1055000) on part_01,
      p2 values <= (3, 1100000) on part_02,
      p3 values <= (6, 1055000) on part_03,
      p4 values <= (6, 1100000) on part_04,
      p5 values <= (9, 1055000) on part_05,
      p6 values <= (9, 1100000) on part_06,
      p7 values <= (12, 1055000) on part_07,
      p8 values <= (12, 1100000) on part_08)
```

しかし、120 万行のローが均等に (パーティションごとに 150,000 ローずつ) 分散される代わりに、奇数のパーティションには 250,000 ローが格納され、偶数のパーティションには 50,000 ローしか格納されません。1 番目と 2 番目の月 (1 月と 2 月) に関しては、Adaptive Server がデータ値を最初のパーティションの最初のキーと比較した時点で、値がキーより小さいため (1 < 3 と 2 < 3)、customer_key の値にかかわらず、1 月と 2 月のデータはすべて最初のパーティションに格納されます。3 月のデータになって初めて、値が key1 と等しくなるため (3=3)、customer_key の値が key2 と比較されます。その結果、偶数のパーティションには、月が分割キーと等しく、customer_key の値がその前の分割キーの customer_key の値より大きいデータしか格納されません。

セマンティック分割とデータの偏り

15.0 より前のバージョンの Adaptive Server では、セグメントベースのラウンドロビン分割方式で並列クエリがサポートされていました。しかし、分割の偏りが特定の割合を超えている場合、オプティマイザは分割のバランスがとれていないために効果的な並列クエリのサポートを提供できないと判断し、クエリを逐次方式で処理していました。並列クエリの使用時にこれを防ぐ 1 つの方法は、分割の偏りをモニタし、クラスタード・インデックスを削除および再作成して分割のバランスをとり直すことです。

Adaptive Server 15.0 のセマンティック分割では、データの偏りを考慮する必要がなくなりました。並列クエリ最適化のためにパーティションの深さを評価する代わりに、Adaptive Server オプティマイザはパーティションのタイプ、クエリの探索引数などを考慮します。セマンティック分割には未知のデータ分散の性質があるため、データの偏りが生じる可能性がありますが、ハッシュ、リスト、または範囲パーティションはデータの場所を知らせるので、偏りは重要ではありません。

パーティションの削除

Adaptive Server 15.0 ではパーティションを削除できません。特定のパーティションを取り除くには、テーブルの分割を解除した後、必要のないパーティションを DDL 文から除外し、前と同じ分割方式を使用してテーブルを再分割します。パーティション内のすべてのデータをトランケートすることもできます。

パーティションの削除は複雑になる場合があります。多くの場合、パーティションを削除する主な理由は、分割されたデータをアーカイブしたためです。パーティションを追加または削除することで、テーブル全体を再分割せずに、分割方式の小さな問題を修正できます。しかし、それによって次のような問題が起きる可能性があります。

たとえば、10、20、30、40、50 などの分割キーで範囲分割されたテーブルを作成し、最初のパーティション (10 以下の値) のデータをアーカイブした後、そのパーティションを削除し、20、30、40、50 などのパーティション範囲を残したとします。その後で、ユーザが分割キー値 5 を持つ有効なデータを入力したとします。範囲分割の仕組みにより、新しく挿入されたデータは最初のパーティション (20 以下の値) に正常に追加されます。しかし、データの一部をアーカイブする (またはデータがさらに追加されて分割のバランスをとり直す) 必要がある場合に、元の分割キーと同じ分割キー (10 以下の値) を使用して新しいパーティションを追加すると、問題が発生します。ローカル・インデックスはすべて「新しい」最初のパーティションを参照するため、以前に挿入されたデータ (値 5) が取り残される可能性があります。この問題の最善の解決策は、新しいパーティションを追加するときに 5 のローを移動することです。つまり、データを移動するときにパーティションを追加する代わりに、テーブルを再分割することになります。

パーティションを削除すると、パーティションのすべてのデータが削除されますが、それによって問題が起きることはありません。しかし、前の例で示したように、新しいデータを挿入すると問題が起きる場合があります。そして、データの分散に関する多くの疑問が生じます。

たとえば、10 個のパーティションを使用する整数カラム上のハッシュ・パーティションがあるとします。パーティションの 1 つを削除すると、使用可能なデータ値の 10 分の 1 のハッシュ・バケットが削除されます。また、削除した特定のハッシュ・バケットに整数 (5、32、41 など) のハッシュ・キーが保持されていたとします。ユーザが値 32 を挿入した場合、残りの 9 個のパーティションにまたがるフル・ドメインを反映するようにハッシュ・アルゴリズムが変化するかどうかを考えてください。その場合、パーティションを削除する目的がデータを再分配するためか削除するためか、つまり、実際には再分割なのかどうかも考えてください。おそらく、ハッシュ・バケットが削除されているので、代わりに値を拒否する必要があるでしょう。これはリストされていない値をリスト・パーティションに挿入するのに似ています。

パーティションの削除は、単なるパーティションとそのすべてのデータの削除よりも複雑になる場合があります。Sybase では、Adaptive Server 15.0 の最初のリリースにこの機能を含めないことにしましたが、今後のリリースでこの機能が導入される可能性があります。しかし、テーブルを再分割し、パーティションをトランケートすることによって、再分割を行わない場合の考慮事項に有効に対処することができます。

パーティションへのデータの入出力

bcp を使用すると、パーティションからデータを抽出できます。また、そのパーティションのキー値のみを含む where 句を指定すると、パーティションからデータを選択できます。bcp を使用して、データをパーティションに直接ロードすることができます。

select を使用してパーティション名を指定し、1 つのパーティションからデータを取得することはできません。

分割キーの値を変更することにより、パーティション間でデータを移動できます。分割キーで使用されているカラムを別のパーティションにある値に更新すると、ローが物理的に別のパーティションに移動します。移動は遅延更新を使用して行われます。遅延更新では、既存のローが現在のパーティションから「削除」され、新しいパーティションに「挿入」されます。たとえば、テーブルを state カラムで分割している場合、カラムを NY から CA に更新するとパーティションが変わります。遅延更新により、NY ローが削除され、削除されたローの挿入がトランザクション・ログに記録されます。これは他の遅延更新オペレーションと同じです。

計算カラムと関数ベース・インデックスの変更

この項では、計算カラムと関数ベース・インデックスに関連するマイグレーションの問題について説明します。

計算カラムの評価

Adaptive Server 15.0 にマイグレートする場合で、計算カラムを考慮している場合は、計算カラムがいつ評価されるかを理解する必要があります。特に、nondeterministic なカラムの見込み出力を事前に決めようとしている場合はなおさらです。次に評価のルールを示します。

- マテリアライズされていない (仮想) 計算カラムの式は、クエリ処理中に評価されるので、現在のユーザのセッションのステータスが反映されます。
- マテリアライズされた (物理) 計算カラムの式は、参照先カラムが変更されたときだけ評価されます。

  たとえば、次のテーブルには 3 組の計算カラムがあり、それぞれが別々に評価されます。

```
create table test_table (
  rownum     int      not null,
  status     char(1)  not null,
  -- virtual columns
  sel_user   as suser_name(),
  sel_date   as getdate(),
  -- materialized columns
  cr_user    as suser_name() materialized,
  cr_date    as getdate() materialized,
  upd_user   as (case when status is not null
                 then suser_name() else 'dbo' end)
                        materialized,
```

```
  upd_date   as (case when status is not null
                  then getdate() else 'jan 1 1970' end)
                        materialized
)
```

- sel_user と sel_date － ユーザがテーブルにクエリを実行したときに評価される仮想カラムです。
- cr_user と cr_date － 他のカラムを参照しない物理的なマテリアライズされたカラムです。これらの式は、ローが挿入されたときだけ評価されます。更新による影響は受けません。
- upd_user と upd_date － これらのカラムは status カラムを参照します。ただし、status カラムは値を決定しません。これらのカラムは、status カラムを任意の値に設定する挿入と更新によって status カラムが変更された場合だけ変更されます。

  評価の結果として、後ろの 2 組の計算カラム、つまり cr_user と cr_date、および upd_user と upd_date はクエリによる影響を受けません。これらは nondeterministic 関数に基づいていますが、値はすべてのクエリに対して一定です。

マテリアライズされていない計算カラムと無効な値

マテリアライズされていない計算カラムの式は、クエリ時にのみ評価され、DML オペレーション時には評価されません。このため、計算カラムが作成される前に、式の作成に使用された計算式が検証されていないと、クエリの問題が発生することがあります。

次の例では、計算カラム b がクエリ時まで評価されないため、select 文が実行されるまではドメイン・エラーが発生しません (select 文がトリガに埋め込まれている場合は、エラーが特に予測できません)。

```
create table t (a int, b compute sqrt(a))
go
insert t values (2)
insert t values (-1)
insert t values (3)
go
select * from t
go
1> select * from t
2> go
a b
----------- --------------------
2   1.414214
Domain error occurred.
```

長い識別子の変更

Adaptive Server 15.0 は、テーブル名、カラム名、インデックス名で長い識別子をサポートしています。通常の識別子の場合は 255 バイト、区切り識別子の場合は 253 バイトです。制限が拡張されたため、一部のシステム・テーブルと関数で長い識別子を使用できるようになりました。

値をバインドするために、対応するアプリケーションの変更のある識別子名を変更します。既存のアプリケーションで、識別子の名前を 30 バイト (以前の制限) のみでバインドしていないことを確認します。バインドしていると、予期しない動作、アクセス違反、バス・エラー、または一般保護違反が発生する場合があります。

エラー・メッセージにおける変更

既存のアプリケーションが特定のエラー番号やテキストを予期していることがあります。Adaptive Server バージョン 15.0 では、エラー・メッセージが次のように変更されており、これがアプリケーションに影響を与える可能性があります。

- 多くのエラー・メッセージで “ASE” が明示されるようになりました。
- すでに存在するテンポラリ・テーブルを作成すると、メッセージ 2714 の代わりに 12822 が表示されます。
- エラー・メッセージ 4916 の代わりに、エラー・メッセージ 587 が識別カラムのオーバフローを示します。
- カタログに存在しない Java 関数を作成すると、構文エラー・メッセージ 195 の代わりに、エラー・メッセージ 14216 が表示されます。
- 非所有者が sp_procxmode を実行し、ストアド・プロシージャに関連するトランザクション・モードを変更すると、エラー・メッセージ 10354 が表示されます。
- 算術オーバフロー・エラーでは、エラー・メッセージ 3606、重大度 16 が表示されるようになりました。
- dbcc checktable 出力で、メッセージ 2579 はメッセージ 12907 に置き換えられました。

Open Client/SDK と Adaptive Server の互換性

この項では、Open Client のさまざまなリリースを Adaptive Server リリース 15.0 とともに使用する場合の問題について説明します。

Open Client の新機能

Adaptive Server と Open Client の互換は、Open Client の機能の設定に基づいて行われます。この「機能」とは、具体的には、アプリケーションが特定の接続で送信する要求のタイプと、サーバが特定の接続で返すサーバ応答のタイプのことです。Adaptive Server 15.0 で拡張されたサイズを Open Client で使用するには、Open Client の機能を新しいサイズに合わせて有効にし、コードを変更します。ワイド・テーブル機能を有効にする方法については、『Open Client Client-Library C リファレンス・マニュアル』を参照してください。

注意　DB-Library™ では、Adaptive Server 12.5 以降で拡張されたサイズは使用できません。カラムの拡張は Adaptive Server と Open Client 12.5 で導入されました。Open Client 12.5.1 を使用して作成したアプリケーションは、Adaptive Server 15.0 に接続できます。ただし、Adaptive Server 15.0 で追加された機能を利用するには、Open Client 15.0 での再コンパイルが必要になることがあります。

Open Client 15.0 の新機能には、次のようなものがあります。

- 長い識別子 － テーブル名、カラム名などのオブジェクトで、最大 255 バイトがサポートされます。
- スクロール可能カーソル。
- 新しいデータ型 － 8 バイト integer、unicode、text。

Adaptive Server と Open Client の互換性

使用できる Open Client の機能は、実行しているソフトウェアのバージョンによって異なります。ここでは、その組み合わせについて説明します。

- Adaptive Server 15.0 と Open Client 15.0 は完全に互換性があります。ただし、Open Client 15.0 を Adaptive Server 15.0 とともに使用する前に、次の操作を実行してください。
  a　Open Client アプリケーションを 15.0 ライブラリに再リンクします。
  b　CS_VERSION を使用して新しいバージョン番号を設定します。
  c　jConnect を使用する場合は、ドライバを再接続します。

- 15.0 より前の Open Client アプリケーションと Adaptive Server を一緒に実行する場合で、アプリケーションを 15.0 Open Client に再リンクしていない場合、Open Client リリース 15.0 より前の機能には問題ありません。ただし、リリース 15.0 のサイズは有効になりません。Adaptive Server はリリース 15.0 より前の制限に従ってデータを送信し、制限を超えるデータがあればトランケートします。
- Open Client 15.0 が Adaptive Server に接続し、その Adaptive Server がリリース 12.5 より前のリモート Adaptive Server に対してクエリを実行する場合、リモート・サーバはワイド・データをトランケートしてから結果を返します。

  拡張された新しいサイズを使用するには、トランザクションに関係するすべてのサーバとクライアントがリリース 12.5 以降であることが必要です。
- Adaptive Server リリース 15.0 からは、DataDirect ではなく Adaptive Server 15 SDK に附属する Sybase ODBC および OLEDB リリース 15.0 ドライバを使用します。
- 使用中のアプリケーションが jConnect 5.5 に依存している場合は、そのアプリケーションを jConnect 6.05 にマイグレートするか、既存の jConnect 5.5 リリース領域を使用することをおすすめします。
  - Open Client 15.0 では、ライブラリの命名規則が変更されています。“syb” という文字が Open Client ライブラリに埋め込まれるようになりました。たとえば、*libct.a* の代わりに *libsybct.a* が使用されます。
  - 15.0 より前のバージョンの Open Client 接続は、クライアントが (たとえば isql -A や CS_PACKETSIZE を使用して) パケット・サイズを要求したときに確立されていました。

    Adaptive Server 15.0 では、クライアント接続にこの接続方法を引き続き使用できますが、パケット・サイズは Adaptive Server への提案と見なされ、Adaptive Server はサーバ設定のパケット・サイズに基づいてパケット・サイズをネゴシエートします。

    コードの変更や再コンパイルは必要ありません。ただし、クライアント・アプリケーションで特定のパケット・サイズを使用する必要がある場合は、次の方法でこの機能を制限できます。
    - Open client － ct_capability(CS_N)_SRVPKTSIZE) を使用します。
    - jConnect － Packetsize | Restricted の最大値を指定します。
    - ADO.NET/ODBC/OLEDB － 接続の正規化された計算カラムのプロパティ RestrictedMaximumPacketSize を指定します。

第 5 章

データベース管理の変更

この章では、あらかじめ認識していないと問題が発生することがある Adaptive Server のシステム管理の変更点について説明します。変更点と新機能の全リストについては、『Adaptive Server Enterprise 新機能ガイド』を参照してください。

注意 アプリケーションとシステム管理の変更は、マイグレーション準備の中で最も時間がかかる作業ですが、このマニュアルでは、これらの変更を行う方法については説明しません。各自のニーズやリソースに合った変更方法を選択してください。たとえば、独自のプランやスクリプトを開発したり、Sybase コンサルティング・サービスに相談することもできます。Sybase 製品の保守契約を結んでいるサポート・センタでは、ASE Migration Resources Web ページ (http://sybase.com/support/techdocs/migration) でマイグレーションについての情報を提供しており、バグやエラー状態などの技術的な問題の解決に役立ちます。

注意 マイグレーションを行わないで (システムを再構築しないで) アップグレードする場合は、Adaptive Server のバージョン間で dbid が変更されることはありません。管理スクリプトにこれらの ID を使用し、新しいシステムを構築する場合にかぎり、新しい ID 変換を使用するため、これが問題になります。

バージョン 11.5 以前からのアップグレード

Adaptive Server バージョン 15.0 へアップグレードできるのは、Adaptive Server バージョン 11.9.x、12.0.x、または 12.5.x だけです。

Adaptive Server バージョン 11.5 を現在使用している場合は、最初にバージョン 12.0 にアップグレードしてからバージョン 15.0 にアップグレードすることをおすすめします。

バージョン 11.9.2 以降からのアップグレード

ここでは、Adaptive Server バージョン 15.0 へのアップグレードに影響することがある、Adaptive Server バージョン 11.9.2 以降の機能について説明します。

オプティマイザの変更点

Adaptive Server バージョン 15.0 のクエリ・プロセッサはセルフチューニング機能を備えているので、以前のバージョンほど人手の介入を必要としません。また、エンジンがステップ間のデータ・フローをサポートするので、ステップ間の実体化のために使用されるワーク・テーブルの数は従来より少なくなります。ただし、ハッシュ操作とマージ操作が有効であると Adaptive Server が判断した場合は、使用されるワーク・テーブルが多くなります。

次の Adaptive Server 15.0 の構成パラメータが、最適化に影響します。

- max repartition degree
- max resource granularity
- optimization timeout limit
- optimization goal
- prod-consumer overlap factor (Adaptive Server 15.0.1 以降)
- min pages for parallel scan (Adaptive Server 15.0.1 以降)

Adaptive Server 15.0 では、group by アルゴリズムが変更されています。並列プランに set statistics io on を使用することはできません。

抽象プランの機能強化

Adaptive Server 15.0 では、抽象プランの構文に新しいアルゴリズムを使用できます。これにより、最適化目標のクエリ・レベルの設定、タイムアウト、すべての set <QP algorithm> on/off/default 動作 (バージョン 15.0.1 以降) を扱えるようになりました。

詳細については、「抽象クエリ・プランによるクエリの修正」(117 ページ) と『パフォーマンス＆チューニング・ガイド：オプティマイザと抽象プラン』の抽象プランに関する章を参照してください。

クエリ測定基準の取得

Adaptive Server 15.0 では、クエリ処理測定基準を取得できます。これは、sysqueryplans の set statistics time/io に表示されるデータをクエリごとに集計したものです。sysquerymetrics を使用して、次の統計データにアクセスできます。

- 特定のクエリについて集計された時間測定基準の数
- 経過時間、論理 I/O コスト (LIO)、物理 I/O コスト (PIO) の最大値、最小値、平均値

詳細については、「sysquerymetrics および sp_metrics の使用」(105 ページ) と『クエリ・プロセッサ』ガイドの「第 5 章 クエリ処理 (QP) 測定基準」を参照してください。

アップグレード後の統計の更新

sysstatistics に格納された統計データは、アップグレードのデータ・コピー・セクション中にアップグレードされます。その結果、ローがサーバのページ・サイズを超える場合があります。ただし、このようなローがデータ・コピー中に挿入される前に、大きな統計ローは 2 つに分割されます。分割されていない以前のテーブルの統計が、アップグレード後に分割されたテーブルの統計として使用され、オブジェクトの ID が sysstatistics..partitionid カラムのパーティション ID として使用されます。

アップグレード後に (update statistics を複数のテーブルに最近実行していない場合は特に) update statistics を実行することを強くおすすめします。Adaptive Server 15.0 にはソート、グループ化、union、ジョイン、その他の処理を行うためのアルゴリズムが複数あるので、最新の統計が生成されている必要があります。以前のバージョンの Adaptive Server にはアルゴリズムが 1 つしかなく、このアルゴリズムではレポート生成システムで使われるデータのような比較的変化の少ないデータに update statistics を実行する必要はありませんでした。しかし、Adaptive Server 15.0 では、統計が最新の状態でなければ、実際のデータ量が古い統計に基づいて予測される量をはるかに超えるため、低速なアルゴリズムが選択される可能性があります。

警告！ 必要のない変更を統計に加えることは、パフォーマンスに悪影響を与える場合があります。統計を更新するときは (大きなテーブルの histogram tuning factor を変更する場合は特に)、デフォルトより大きなステップ数を使用してください。そうすることにより、重複する値が大量に含まれるカラムのインデックス統計は、より正確にデータの偏りを表すようになります。

誤ったパラメータを指定して update statistics コマンドを実行すると、ヒストグラムのステップ数が変わったり、パフォーマンスに影響することがあります。まず、テスト環境で変更を適用してください。

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』を参照してください。

自動 update statistics

Adaptive Server 12.5.1 では、Job Scheduler ユーティリティ・プログラムを使用して update statistics の実行を自動化する機能が追加されました。Adaptive Server 15.0 では、この機能が強化され、Job Scheduler によって特定のテーブルまたはパーティションに update statistics が実行される前に、データベース管理者がスレッショルドを設定できるようになりました。このスレッショルドの制御によって、必要なときにだけ update statistics が実行されるようになり、メンテナンス操作に要する時間が短縮されます。

Job Scheduler エンジンの設定と Sybase Central インタフェースの詳細については、『Job Scheduler ユーザーズ・ガイド』の第 4 章「ジョブ・スケジューリングへのテンプレートの使用」を参照してください。

「統計の自動更新」といっても、インデックス統計が Adpative Server で自動的に管理され、update statistics が不要になるわけではありません。ユーザが各 DML 文に対する統計の変更をサーバで追跡するように頻繁に要求すれば、update statistics を実行する必要性は完全になくなります。しかし、これは、たとえトランザクションのスコープ外で行ったとしても、OLTP 操作の実行を遅くする原因になります。これは複数のユーザが systabstats の同じローを変更しようとしている状況に近く、それによって起こる競合のため、システムは事実上シングルスレッドで動作することになります。

また、大きなテーブルにローを少しずつ追加すると、新しく計算された密度が精度の損失のため以前と同じ値になり、統計が正確に更新されない場合があります。たとえば、現在 1,000,000 ローのテーブルがあり、各ローの date カラムにはそのローをテーブルに追加した日付が格納される (つまりカラムのデフォルト値が getdate() に設定されている) ときに、このテーブルに 100,000 ローを追加するとします。カラムを 1 つずつ追加すると、新しいローはテーブルの 1/1,000,000 (つまり .000001) の増加にしかならないため、範囲セル密度は変わりません。しかし、100,000 ローの追加はテーブル全体のサイズを 10% 増やします。

datachange 関数

自動 update statistics は、datachange 関数に基づいて実行されます。この関数は、テーブル内で変更されたデータのパーセンテージを返します。この関数を既存の update statistics スクリプトに追加して、この追跡機能を利用することができます。例：

```
select @datachange = datachange("authors", null, null)
if @datachange > 50
begin
    update statistics authors
end
```

datachange 関数を使用する場合は、次のことを考慮する必要があります。

- datachange から返されるパーセンテージは、DML オペレーションの回数とテーブル・サイズに基づく値です。ただし、遅延オペレーションは 2 つのオペレーション (1 つの delete と 1 つの再 insert) としてカウントされます。したがって、複数のレコードを同じ文で更新すると、datachange からは、実際に変更されたローの最大で 2 倍のパーセンテージが返されることがあります。
- datachange 関数のパラメータは *table_name*、*partition_name*、*column_name* で、この順番で渡します。パラメータを使用すると、特定の変わりやすいフィールドまたはパーティションの変更を検出することや、すべてのインデックスでの変更を検出する代わりに特定のインデックスのインデックス統計を更新することができます。
- datachange から返される値は、変更されたローの数ではなく変更のパーセンテージです。ロー数それ自体は役に立つ情報ではないからです。ロー数が役に立つのは、テーブルのサイズに関する比較を行う場合だけです。たとえば、`datachange=5,000` の値は、テーブルにあるローの数が 5,100 であれば大きな数ですが、ロー数が 500 万のテーブルでは小さな数です。パーセンテージを使えば、管理スクリプトで相対的なスレッショルドを簡単に指定できます。

パーティションでの *update statistics*

Adaptive Server 15.0 へマイグレートした後で特定のパーティションに update statistics を実行すると、通常の update statistics の実行に必要な時間が短縮されます。update statistics の実行に時間がかかるほとんどの大きなテーブルには履歴データが大量に含まれていますが、このような履歴ローが変更されることはほとんどないにもかかわらず、update statistics ではすべてのローがスキャンされます。また、テーブル全体を見た場合、たとえ 100 万ローの追加であったとしても 5 億ローのテーブルであれば 0.2 % の変更にしかならず、統計を更新する必要はないと考えられます。ただし、このようなローは頻繁に使用され、その分散のヒューリスティックが範囲セル密度に含まれない可能性が高いため、クエリの最適化に負担をかける可能性があります。

データを分割した場合、最初の update statistics を実行した後は (日付の範囲を分割した場合は特に) update statistics の実行時に古い静的なデータをスキップできます。この最初の実行の後は、datachange を使用して現在のパーティション内の変更の量をチェックして、必要な場合にだけ update statistics を実行します。すべてのパーティションには名前が付けられます。名前を指定しなかった (つまりハッシュ分割構文を使用した) 場合、tempdb と同じように、デフォルト名が自動的に与えられます。システムから提供された名前を確認するには、sp_help を使用します。

次の例では、システム提供の名前に基づき、p_partkey カラムのデータを取得します。

```
select datachange("mytable","part_1360004845", "p_partkey")
go
---------------------------
100.000000
```

これは、update statistics を part_1360004845 パーティションに対して実行するのが適切と考えられる状況です。

update statistics を特定のパーティション (ここではカラム) に実行するには、次のように指定します。

```
update statistics mytable partition part_1360004845 (p_partkey)
go
```

update statistics の実行時間を短縮できるので、これまで時間の制約により避けていた統計の作成やカラムのヒューリスティックを実行できるようになります。たとえば、次の例では、col1 のヒストグラムと col1,col2 ペアの組み合わせの密度を作成します。:

```
    update statistics mytable (col1, col2)
```

統計をパーティションごとに更新し、col2 のヒストグラムを作成するには、次のように指定します。

```
update statistics mytable partition part_1360004845 (col1, col2)
go
update statistics mytable partition part_1360004845 (col2)
go
```

関数の変更

Adaptive Server 15.0 では、以前のバージョンで領域の使用量の確認に使用されたシステム関数が非推奨となり、パーティション対応バージョンの関数に置き換えられました。表 5-1 に、これらの関数と構文の変更点を示します。

表 5-1: Adaptive Server 15.0 で変更された関数のリスト

12.5 の関数と構文	15.0 の関数
data_pgs(object_id, {doampg | ioampg})	data_pages(dbid, object_id [, indid [, ptn_id]])
used_pgs(object_id, doampg, ioampg)	used_pages(dbid, object_id [, indid [, ptn_id]])

12.5 の関数と構文	15.0 の関数
reserved_pgs(object_id,{doampg | ioampg})	reserved_pages(dbid, object_id [, indid [, ptn_id]])
rowcnt(sysindexes.doampg)	row_count(dbid, object_id [, ptn_id])
ptn_data_pgs(object_id, partition_id)	(data_pages())

Adaptive Server 15.0 では、OAM ページ・パラメータの doampg および ioampg が、indid (インデックス ID) および ptn_id (パーティション ID) に置き換えられ、使いやすくなりました。また、これらの関数の使用方法も変更されました。

Adaptive Server 12.5 では、これらの関数は sysindexes テーブルに関するクエリに使用されました。これは領域割り付けの追跡に sysindexes が使用されるからであり、そのため、sysindexes doampg カラムと ioampg カラムは OAM ページ・パラメータを提供しました。たとえば、Adaptive Server 12.5 では、特定のテーブルの各ノンクラスタード・インデックスに使用される領域の合計を計算する場合に、次のようなクエリを使用しました。

```
-- ASE 12.5 logic to report the spaced used by nonclustered indices
select name, indid, used_pgs(id, doampg, ioampg)
   from sysindexes
   where id=object_id('salesdetail')
     and indid > 1
```

Adaptive Server 15.0 では、領域は sysindexes ではなく syspartitions にリンクされるため、前記のクエリを Adaptive Server 15.0 で使用するには、次のように書き直します。

```
-- ASE 15.0 logic to report the spaced used by nonclustered indices
select i.name, p.indid, sum(used_pages(db_id(), p.id ,p.indid))
   from sysindexes i, syspartitions p
   where i.id=object_id('salesdetail')
     and p.id=object_id('salesdetail')
     and i.indid > 1
     and p.indid > 1
     and p.id=i.id
     and p.indid=i.indid
   group by i.name, p.indid
   order by p.indid
```

非推奨の Adaptive Server 12.x 関数は現在も存在しますが、返される値は 0 です。これらの関数で使用される sysindexes.doampg および sysindexes.ioampg は Adaptive Server で管理されなくなりました。syspartitions は、似たような構造をカラム datoampage および indoampage に持ちますが、これらのカラムの値はパーティション・ベースであり、分割されたテーブルの場合はインデックス領域の使用量を集計する必要があります。

領域をレポートするシステム関数

表 5-1 に、変更された関数と新旧の構文を示します。これらの変更はエンドユーザ・アプリケーションには影響しませんが、スクリプトやユーティリティには影響する場合があります。

スクリプトに最大の影響を与える可能性がある変更は、sysindexes.doampg が sysindexes.indid に置き換えられ、ioampg が partition_id に置き換えられたことです。以前のバージョンでは sysindexes のスキャンが使用されましたが、Adaptive Server 15.0 では、syspartitions (多くの場合に sysindexes とジョインされる) のスキャンが使用されます。たとえば、Adaptive Server 12.5 では、ノンクラスタード・インデックスで使用される領域をレポートするためにこれが使用されます。

```
select name, indid, used_pgs(id, doampg, ioampg)
from sysindexes
where id=object_id('authors')
and indid > 1
```

Adaptive Server 15.0 でノンクラスタード・インデックスで使用される領域をレポートするには、次の構文を使用します。

```
select i.name, p.indid, used_pages(dbid(), p.id ,p.indid)
from sysindexes I, syspartitions p
where i.id=object_id('authors')
and i.indid > 1
and p.indid > 1
and p.id=i.id
and p.id=object_id('authors')
and p.indid=i.indid
order by indid
```

スクリプトが異なるのは、Adaptive Server 15.0 では記憶領域が sysindexes ではなく syspartitions にリンクされているからです。

次の例は、Adaptive Server 15.0 でノンクラスタード・インデックスで使用される領域をパーティションごとにレポートします。

```
select p.name, i.name, p.indid, used_pages(dbid(), p.id,
p.indid, p.partitionid)
from sysindexes I, syspartitions p
where i.id=object_id('authors')
and i.indid > 1
and p.indid > 1
and p.id=i.id
and p.id=object_id('authors')
and p.indid=i.indid
order by p.partitionid, p.indid
```

システム・テーブルの変更

Sybase では、Adaptive Server 15.0 でシステム・テーブルを更新して、セマンティック・パーティションと暗号化カラムの変更を反映し、新しいシステム・テーブルを追加し、その他のテーブルにパーティション情報を加え、新しいオブジェクトをオブジェクト・クラスに追加しました。これらの変更は、サードパーティ・ツールとカスタム・スクリプトに影響する場合があります。

詳細については、『Adaptive Server Enterprise 新機能ガイド バージョン 15.0』を参照してください。

ディスク領域割り付けの変更

以前のバージョンの Adaptive Server では、ディスク領域割り付けは sysindexes でレポートされましたが、Adaptive Server 15.0 では syspartitions でレポートされます。表 5-2 に、以前のバージョンの Adaptive Server での領域ポインタと、Adaptive Server 15.0 の syspartitions でそれらに対応するものを示します。

表 5-2: 異なるバージョンの Adaptive Server での領域ポインタ

領域関連付けの種類	以前のバージョンでの sysindexes のカラム名	バージョン 15.0 の syspartitions で対応するもの
ユニーク・ロー	id + indid	id+indid+partionid
最初のページ	first	firstpage
ルート・ページ	root	rootpage
データ OAM ページ	doampg	datoampage
インデックス OAM ページ	ioampg	indoampage

これらのロケーションを使用するカスタム DBA スクリプトまたは従来の dbcc コマンドを変更して、現在の実装を反映する必要があります。Sybase は、パブリッシュされた dbcc コマンドをすでに変更しました。従来ドキュメント化されていない dbcc コマンド (dbcc pglinkage() など) を使用している場合、これらのコマンドは非推奨または管理対象外になったため、正常に動作しない可能性があります。データ破損の問題が起きた場合は、Sybase 製品の保守契約を結んでいるサポート・センタまで Adaptive Server 15.0 での適切な使用方法をお問い合わせください。以前のバージョンで使用していた、ドキュメント化されていない dbcc コマンドを使い続けると、データが破損する可能性があります。

サードパーティ・ツールへの変更

Adaptive Server 15.0 へアップグレードしても、通常の DML やクエリの演算のためにデータベースにアクセスするサードパーティ・アプリケーションまたは社内アプリケーションは影響を受けません。ただし、システム・テーブルとシステム関数の変更は、DDL を実行したり、システム・カタログにアクセスしたりするサードパーティ・ツールやカスタム・アプリケーションにとって問題となる可能性があります。

たとえば、Embarcadero の以前のバージョンの DBArtisan は、Adaptive Server 12.5 と互換性があり、テーブル名とテーブル内のロー数を表示するオブジェクト・ブラウザなどの機能は広く利用されています。このロー・カウントは、rowcnt 関数を使用して取得されましたが、現在この関数は非推奨です。その結果、これらのバージョンの DBArtisan を使用すると、大量の警告に加えて、テーブルのロー数として 0 が返されるため、15.0 サーバで「すべてのデータが失われる」ように見えて驚かされる場合があります。Sybase では非推奨の関数を互換性のために残してありますが、警告メッセージは返されます。

サードパーティ・ユーティリティのベンダに問い合わせて、Adaptive Server 15.0 と互換性のあるバージョンのリリース時期を確認してください。

データベース ID の変更点

Adaptive Server バージョン 12.0 以降では、次のデータベースの dbid は 31513 から始まります。

- dbccalt
- dbccdb
- sybsecurity
- sybsystemdb
- sybsystemprocs

アップグレード後にこれらのデータベースを削除して再作成する場合は、これらの新しい dbid が適用されます。その他すべてのデータベースの dbid は、前のバージョンと同じ方法で決定されます。

ASE plug-in for Sybase Central

この項では、Sybase Central バージョン 4.3 に関連する問題について説明します。

注意 Sybase Central を実行するには、Java がインストールされている必要があります。

Adaptive Server 15.0 には、Sybase Central リリース 4.3 が組み込まれています。ただし、このバージョンの Sybase Central は、他の Adaptive Server 実装からインストールされる Sybase Central 4.3 と同じビルドではない場合があります。Adaptive Server 15.0 に付属する Sybase Central ビルドは、SySAM および Unified Agent Framework (UAF) のプラグインをサポートし、以前のリリースにはなかった新機能 (リモート・サーバの起動、停止、ping、リモート・エラー・ログの表示、サーバの自動検出、サーバ・グループなど) を備えています。

以前のリリースでは、Sybase Central のロケーションは *$SYBASE* でした。Adaptive Serer 15.0 に組み込まれたバージョンのロケーションは、*$SYBASE/shared/sybcentral43* です (Windows では *%SBASE%¥Shared¥Sybase Central43*)。このようにロケーションが変更されたため、プログラムを起動するアイコンを以前のバージョンのままにしていたり、*path* 設定や CLASSPATH が以前のロケーションを指していると、問題が起こります。

古い Sybase Central ディレクトリの名前を変更して、新しいバージョンとの関連付けを解除してください。新しいバージョンをしばらく使用し、すべての製品プラグインと新しいバージョンとの互換性が確認できた後で、古いバージョンは削除できます。製品プラグインは、www.sybase.com で個別に入手できます。

sql.ini にリストされているサーバの表示

Adaptive Server の Sybase Central プラグインでは、*sql.ini* ファイルにリストされているすべてのサーバが表示されるわけではなくなりました。かわりに、Sybase Central では、以前に接続したことのあるサーバ、または Windows NT サービスとして起動されているサーバだけをリストします。新しいサーバに初めてアクセスするには、Adaptive Server プラグインの [接続] メニュー・オプションを使用して *sql.ini* ファイルにリストされているサーバを選択します。

トラブルシューティング

Sybase Central を使用するときは、次のことに注意してください。

- 最新のビルド (ASE 15.0 ESD1 では 4.3.0.2419) を使用していることを確認します。Adaptive Server 15.0 プラグインと互換性のない古いバージョン (バージョン 4.1 など) の Sybase Central またはバージョン 4.3 の古いビルドを起動している場合があります。最新のビルドの Sybase Central を使用していることを確認し、さらに *%SYBASE%¥Shared¥Sybase Central 4.3* からそのバージョンを起動していることを確認します。このバージョンを起動していることを確認するには、DOS ウィンドウを開いてこのディレクトリに移動し、*scjview.bat* を実行します。
- UAF プラグインを使用する Sybase Central では、Java セキュリティ・ポリシーが適用されます。ただし、VPN ソフトウェアでは、InfoExpress の VSClient などの TCP リダイレクトが使用されます。これらに互換性がないことがわかった場合は、VPN クライアント アプリケーションを終了してから、Sybase Central を再度実行してみてください。

Sybase Central がクラッシュすると、スタック・トレースが *scj-errors.txt* という名前のファイルに作成されます (複数のクラッシュがあった場合は、ファイル名に通し番号が与えられるので、2 番目のファイルは *scj-errors-2.txt* になります)。このファイルは、Sybase Central ホーム・ディレクトリに格納されます。Sybase 製品の保守契約を結んでいるサポート・センタに問題を報告するときは、このファイルを添付してください。プラグインのバージョンと、Sybase Central およびプラグインで使用されている Jar ファイルのバージョンを確認するために、このファイルが必要です。

Interactive SQL

Adaptive Server プラグインには、Interactive SQL が組み込まれています。Interactive SQL は、すべての Sybase サーバの共通のクライアント・ユーティリティであり、Replication Server などの Open Server ベースのアプリケーションに接続できます。Adaptive Server プラグインは、Java ベース・クライアントとして Linux およびその他の UNIX プラットフォームで動作します。以前のリリースの Adaptive Server に含まれていた Jisql ユーティリティよりも高機能の製品です。

Interactive SQL を使用すると、サーバに対して SQL 文の実行、スクリプトの作成、データの表示を行うことができます。Interactive SQL は、次の目的で使用できます。

- データベース内の情報を一覧表示する
- クエリを図示する
- 結果セットを編集し、変更点をデータベースに反映する
- MicroSoft Excel およびその他の形式でエクスポートする
- アプリケーションで使用する予定の SQL 文をテストする
- データをデータベースにロードし、管理作業を実行する

Interactive SQL では、コマンド・ファイルまたはスクリプト・ファイルを実行できます。たとえば、データベースに対して実行する繰り返し可能なスクリプトを作成し、Interactive SQL を使用してそのスクリプトをバッチとして実行できます。

バッチ・スクリプトにあるエラーの訂正

Adaptive Server 15.0 および ESD #1 のバッチ・スクリプトには、Interactive SQL クラス・ファイルを呼び出す部分に誤りがあります。-Dpath オプションには *%path%* 値が存在しますが、一部のスクリプトではこのオプションは不適切です。

この問題を修正するには、*%SYBASE%¥dbisql¥bin* に移動し、*dbisql.bat* ファイルを編集して実行行を以下のように変更します。

注意 ここでは見やすくするために改行していますが、実際にはこのコマンドは連続した 1 行です。この行をコピーし、スクリプトに貼り付けるときは、特にスイッチ文字のダッシュ (“-”) に注意してください。

```
"%JRE_DIR%¥bin¥java" -Disql.helpFolder="%DBISQL_DIR%¥help" -
Dsybase.jsyblib.dll.location="%SYBROOT%¥Shared¥win32¥¥" -
Djava.security.policy="%DBISQL_DIR%¥lib¥java.policy" -Dpath="%path%" -classpath
"%DBISQL_DIR%¥lib;%isql_jar%;%jlogon_jar%;%jodbc_jar%;%xml4j_jar%;%jconn_jar%;%dspa
rser_jar%;%helpmanager_jar%;%jcomponents_jar%;%jh_jar%;%jsyblib_jar%;%planviewer_ja
r%;%sceditor_jar%;%uafclient_jar%;%jinicore_jar%;%jiniext_jar%;%jmxremote_jar%;%jmx
ri_jar%;%commonslogging_jar%;%log4j_jar%" sybase.isql.ISQLLoader -ase %*
```

また、*%path%* 環境変数から不要な部分を省くこともできます。例：

```
set PATH="c:¥sybase¥ASE-15_0¥bin;c:¥sybase¥OCS-15_0¥bin;.;"
```

SQL Advantage のサポート廃止

Sybase は、SQL Advantage™ のサポートを終了しました。SQL Advantage を使用して Adaptive Server 15.0 に接続できますが、Open Client 15.0 の新しい API 機能がないため、機能が制限されます。

Adaptive Server 15.0 PC クライアントがあるマシンで SQL Advantage が正常に実行されない場合、Open Client 12.5 の前のパスにある Open Client 15.0 が SQL Advantage で使用されている可能性があります。Open Client 15.0 では、一部の *dll* ライブラリ名が変更されたため、このようなライブラリを SQL Advantage で使用するとエラーになります。これを回避するには、Open Client 12.5 のライブラリ・ディレクトリのみをパス環境変数に含む (Open Client 15.0 のライブラリ・ディレクトリを含まない) シェル・スクリプトを作成します。このシェル・スクリプトを使用して、SQL Advantage を起動します。

sybsyntax

オンライン・ヘルプと Transact-SQL の構文を isql コマンド・ラインに表示する sybsyntax ユーティリティは、Adaptive Server バージョン 12.0 で削除されましたが、Adaptive Server version 12.5 で追加されました。最新の構文を表示するには、最新バージョンの sybsyntax をインストールしてください。

マニュアルの変更

Adaptive Server のマニュアル・セットが次のように変更されました。

- 『Managing and Monitoring Adaptive Server Enterprise』は廃止になりました。
- 『ユーティリティ・ガイド』が、プラットフォームに関係なくすべてのユーティリティで構成される汎用的なマニュアルになりました。

sybsystemdb

master データベースは、spt_values テーブルと 2 フェーズ・コミットに使用されます。設定ユーティリティによって、マスタ・デバイス上にこのデータベースが自動的に作成されます。

このデータベースでは多くのアクティビティが行われ、ログ領域を大量に使用するため、sybsystemdb を別のデバイスに作成してからアップグレードするか、またはアップグレード後にこのデータベースを別のデバイスに移動することを強くおすすめします。これによって、sybsystemdb のログがいっぱいになってもマスタ・デバイスは保護されます。

sybsystemdb を必要とする機能を使用する予定がない場合は、最小サイズのままでも構いません。

syslogins での bcp

前のバージョンの Adaptive Server からデータを syslogins にコピーするために bcp を使用すると、Adaptive Server 12.5 で追加されたカラムが原因でエラーになります。

bcp を使用してデータをバージョン 12.5 の syslogins にコピーする場合は、データを保持するダミー・テーブルを作成し、このテーブルに 2 つのカラムを追加してから、そのデータを 12.5 の syslogins にコピーします。

注意 この問題は、追加カラムのある他のシステム・テーブルでも発生することがあります。アップグレードの一部としてこのデータをコピーする必要がある場合は、最初に syslogins でこの問題が発生します。

Adaptive Server リリースでのシステム・テーブルの変更点については、『Adaptive Server Enterprise 新機能ガイド』を参照してください。

ユーザとログインの最大数

Adaptive Server バージョン 12.5 では、サーバへのログインの最大数とデータベースへのユーザの最大数が増えました。

- ログイン数 − 20 億プラス 32000
- データベースのユーザ数 − 20 億マイナス 1032193
- データベースのグループ数 − 1032193

これらの変更による影響を以下に示します。

- ユーザ ID (uid) に負の値を使用できる
- sysusers でグループや役割に割り当てられているサーバ・ユーザ ID (suid) の値は、suid の符号を逆にした値ではなくなりました。Adaptive Server version 12.0 では、sysusers でグループや役割に割り当てられている suid はすべて、-2 (INVALID_SUID) に設定されます。

ID の範囲については、『Adaptive Server Enterprise 新機能ガイド』を参照してください。

新しい予約語

予約語の詳細については「予約語」(38 ページ) を参照してください。

設定パラメータ

アップグレードでは、設定した設定パラメータの値が保持されます。サーバ設定ファイルで “DEFAULT” に設定されているパラメータだけが、新しいバージョンのデフォルト値に変更されます。設定したパラメータの値が、Adaptive Server 15.0 の新しいデフォルト値より小さくないことをチェックしてください。アップグレード中またはアップグレード後にデフォルト値より値が小さいと問題になるのは、以下のパラメータです。

- stack size
- cpu grace time
- enable housekeeper GC

スタックのサイズ

最近の数回のリリースで、stack size パラメータのデフォルト値が大きくなっています。スタック・サイズはプラットフォームごとに異なりますが、Solaris の一連のバージョンでは、次の表のようにスタック・サイズの要件が高くなっています。

サーバのバージョン	最小スタック・サイズ
11.0.x	24576 (24K)
11.5	34816 (32K)
11.9.2	34816 (32K)
12.0	46090 (45K) − 32 ビット 86016 (84K) − 64 ビット
12.5	46090 (45K) − 32 ビット 86016 (84K) − 64 ビット
15.0	46090 (45K) − 32 ビット 86016 (84K) − 64 ビット

設定値が小さすぎるスタック・サイズを使用すると、アップグレード中またはアップグレード後の処理中にサーバでスタック・オーバフローが発生する可能性があります。

stack size パラメータを確認して、必要に応じてデフォルト値にリセットします。stack size 設定パラメータを設定するには、設定ファイルで既存の値を “DEFAULT” という用語に置き換えます。

設定パラメータの一般的な情報については、『システム管理ガイド 第 1 巻』を参照してください。新しい設定パラメータと変更点の全リストについては、『Adaptive Server Enterprise 新機能ガイド』を参照してください。

メモリの増加

Adaptive Server 15.0 では以前のリリースより多くのメモリが使用され、データ・サーバ、プロシージャ・キャッシュ、名前付きキャッシュに割り付けられるメモリが増えました。このリリースでは、多くのシステム・プロシージャが他のプロシージャを呼び出すように変更されました。Adaptive Server 15.0 のクエリの最適化は、以前のリリースよりはるかに複雑なため、プロシージャ・キャッシュのサイズを最小の値に設定したサイトでは、領域が不足する可能性があります。

たとえば、Adaptive Server 15.0 では、ソートとグループ化をメモリ内で行うアルゴリズムが使用されるため、ソート・バッファとソート用の追加メモリを追跡するために使用される補助スキャン・バッファには、従来より大きなプロシージャ・キャッシュ領域が必要です。Adaptive Server 15.0 では、ワーク・テーブルではなくメモリ内でソートが行われるため、tempdb がバインドされたデータ・キャッシュのメモリを増やす必要があります。

必要な追加メモリの量を予測することは不可能ですが、現在の割り付けサイズを少しのパーセンテージ (5 ～ 25 パーセント程度) 増やすだけで十分と考えられます。アップグレード後に、メモリの使用量を注意して監視してください。sp_helpcache を使用して、アップグレードの前後のキャッシュ・サイズをチェックし、アップグレード完了後に必要に応じてメモリを調整してデフォルト・データ・キャッシュの元の値をリストアします。

キャッシュとメモリ・プールの設定方法の詳細については、『システム管理ガイド 第 2 巻』を参照し、メモリの問題の詳細については、『パフォーマンス & チューニング・ガイド：基本』を参照してください。

アップグレードとサーバ機能に影響する変更点

以下の Adaptive Server バージョン 15.0 の変更点は、アップグレードとサーバ機能に影響します。

- sysanchors という名前の擬似カタログにデータベース単位で保存されるロー「アンカー」を使用するシステム・カタログ。sysgams と同様、sysanchors は通常のデータベース・テーブルではなく、クエリの対象となりません。
- ほとんどのシステム・カタログはロー・ロックである。
- オブジェクトのパーティション ID は、そのオブジェクトに固有である。

Adaptive Server 15.0 以降へマイグレートするときに注意する必要があるその他の問題について、以下に説明します。

- Adaptive Server の以前のリリースで「パーティション」と呼ばれていたものは、「スライス」という呼び名に変更され、「パーティション」という用語は使用されなくなりました。バージョン 12.5.x 以前から Adaptive Server 15.0 以降にアップグレードすると、バージョン 12.5.x でサポートされているスライスで分割されたテーブルを含むすべてのデータベース・テーブルが、分割されていないテーブルに変更されます。インデックスは変更されず、分割されていないグローバル・インデックスとして保持されます。分割されているテーブルが必要な場合は、アップグレード後に手動で再分割する必要があります。
- マイグレート中に syspartitions は、sysslices へ名前が変更されます。このデータが使用されるのはアップグレード中だけで、アップグレードが完了すると削除されます。アップグレードの処理が完了した後は、sysslices は空のテーブルとして残されます。
- このリリースでは、前のリリースより多くのプロシージャ・キャッシュを使用するようになりました。プロシージャ・キャッシュの量はアプリケーションによって異なりますが、一般に Adaptive Server 15.0 では以前のリリースより約 10 パーセント多く使用されます。
- Adaptive Server では、以前のリリースの Adaptive Server から sort の redo を行う必要がある create index を再発行する load tran コマンドの実行が制限されます。Adaptive Server では、create index が最初に実行され、後続の load tran コマンドが無効になった時点まで、この操作が含まれるログがリカバリされます。load database コマンドを実行するまで、load tran コマンドを実行できません。

デバイス・サイズ

バージョン 15.0 では、20 億以上のデータベース・デバイスがサポートされ、各デバイスには 2,147,483,648 の 2K ブロック (最大 4 テラバイト) を格納できます。メモリでサポートできる数だけデバイスを設定できます。

bigint のサポート

Adaptive Server 15.0 には、bigint 真数値データ型が組み込まれています。

表 5-3 に、bigint データ型で使用できる値の範囲を示します。

表 5-3: bigint データ型の範囲

データ型	符号付きデータ型の範囲
bigint	-2^{63} ～ 2^{63} - 1 (-9,223,372,036,854,775,808 ～ +9,223,372,036,854,775,807 の間の整数)

bigint データ型の一部として、hextobigint、biginttohex、count_big 関数も組み込まれています。詳細については、『リファレンス・マニュアル：ビルディング・ブロック』を参照してください。

符号なし整数 (*unsigned int*) のサポート

符号なし整数のデータ型 (たとえば unsigned int や unsigned smallint) を使用すると、必要な記憶域のサイズを増やすことなく、既存の整数データ型の正の値の範囲を拡大できます。符号なしデータ型には、負の数を格納できません。

整数 identity

Adaptive Server バージョン 15.0 では、次のデータ型を identity 値として使用できます。

- bigint
- int
- numeric
- smallint
- tinyint
- unsigned bigint
- unsigned int
- unsigned smallint

独立したデバイス ID カラム

以前のリリースでは、デバイス番号 (*vdevno*) は、32 ビット・ページ実装の上位バイトでした。多くの場合、*vdevno* を取得するには、2^{24} のような値を使用する複雑な計算を実行して上位バイトを分離する必要がありました。また、この方法では、デバイスの数が 255 に暗黙に制限されます。master..sysusages のデバイス・フラグメントを master..sysdevices に関連付けようとする場合、high および low の仮想ページ番号に基づいて between 句を使用して 2 つのテーブルをジョインする必要がありました。

Adaptive Server 15.0 では、仮想ページ番号が 2 つの 32 ビットの整数になりました。1 つの整数がデバイス番号 (*vdevno*) を表し、もう 1 つがページ ID を表します。*vdevno* は、sysusages と sysdevices に含まれています。この変更の結果、領域の消費量を計算するスクリプトは修正が必要です。たとえば、次のスクリプトは Adaptive Server 12.5 で使用するために書かれました。

```
select d.name, u.size
from sysuages u, sysdevices d
where u.vstart >= d.low
and u.vstart <= d.high
and u.dbid = <database id>
```

これを Adaptive Server 15.0 で使用するには、次のように書き直します。

```
select d.name, u.size
from sysusages u, sysdevices d
where u.vdevno = d.vdevno
and u.dbid = <database id>
```

ファイル・システムか、それともロー・パーティションか

ファイル・システムとロー・パーティション・デバイスのどちらを使用するか決めるときは、オペレーティング システムのファイル・システム実装、アプリケーションの I/O プロファイル、使用可能なオペレーティング・システムのリソース (メモリや CPU など)、ファイル・システム・キャッシュの制限に従ったオペレーティング・システムのチューニング、スワッピングの優先度など、多くの要素を考慮する必要があります。

以前のリリースの Adaptive Server では、大量の読み込み処理でファイル・システムの先読みが Adaptive Server の非同期プリフェッチより速く実行されるため、このようなケースではファイル・システムを使用するのが適切です。しかし、書き込み処理では (同時実行性の高い環境では特に) ロー・パーティションを使う方が有利です。

Adaptive Server 12.0 には、デバイス dsync パラメータが組み込まれていました。このパラメータには、ファイル・システム・デバイス用の dsync I/O が実装され、デバイスへの更新を記憶メディア上で直接実行したり、更新を UNIX ファイル・システムにバッファしたりできました。dsync ではリカバリ性を保証するためにファイル・システムのバッファが迂回されると思われるかもしれませんが、実際はファイル・システム・バッファが使用され、ファイル・システムへの書き込みのたびに強制的にバッファがフラッシュされていました。このように、Adaptive Server とファイル・システム・キャッシュの両方でバッファリングが行われ、さらにフラッシュが要求されることは、ファイル・システム・デバイスへの書き込みがロー・パーティションへの書き込みより応答時間が長くなる原因でした。

Adaptive Server 15.0 では、disk init、disk reinit、sp_deviceattr に directio パラメータが組み込まれました。このパラメータにはダイレクト I/O が実装されているので、オペレーティング・システムのバッファ・キャッシュを迂回してデータをディスクに直接転送するように Adaptive Server を設定して、書き込み時間を短縮できます。directio パラメータは、以下のプラットフォームでサポートされています。

- Sun Solaris
- IBM AIX
- Microsoft Windows

directio を使用する前の考慮事項

directio を使用する前に、次のことを考慮する必要があります。

- オペレーティング・システム・カーネルのチューニング、特殊なオプション (たとえば Solaris の forcedirectio など) を指定したファイル・システムのマウント、大量のダイレクト I/O 処理の実行に必要なオペレーティング・システム・パッチ・レベルかどうかの確認が必要な場合があります。
- directio パラメータと dsync パラメータは互いに排他的です。現在 dsync パラメータを使用しているが、今後は directio を使用する場合は、最初に dsync パラメータを無効にしてから directio を有効にする必要があります。ただし、デバイス属性を変更すると、Adaptive Server の再起動が必要です。
- ファイル・システム・キャッシュのメモリ要件と、オペレーティング・システムでの I/O 要求処理の変更に伴う CPU 要件が、Adaptive Server リソースに影響しないことを確認してください。
- デバイスでロー、dsync、またはダイレクト I/O を使用するように切り替えながら、それぞれの状況でユーザの負荷が 25%、50%、75%、100% のときのアプリケーションのスケーリングをテストします。このようなシナリオを個別にテストすると、スケーリングが直線的か、使用時の負荷が 100% に近づくにつれて低下するかがわかります。ある地点でパフォーマンスは横ばいになるが、ユーザ数が増えるためユーザの負荷をさらに増やす必要がある場合は、リソースを Adaptive Server に追加する必要があります。
- SMP ハードウェア上のファイル・システムで tempdb を使用して Adaptive Server を実行している場合、dsyncio を無効にするより、directio を有効にする方がメリットは大きくなります。ただし、tempdb デバイス設定を変えてさまざまな負荷レベルでアプリケーションをテストしてください。ユーザの同時実行性が低い tempdb または数は少ないがサイズが大きいテンポラリ・テーブルがある小規模な SMP システムでは dsyncio を無効にすると大きな利点がある一方で、大規模な SMP システム、またはユーザの同時実行性が高い tempdb かサイズは小さいが書き込み処理が多い tempdb があるシステムでは、directio を有効にすると大きな利点があることがわかります。毎晩のバッチ・レポート作成には dsyncio を無効にして tempdb をファイル・システムのデバイスで実行し、日中の OLTP 処理テンポラリ・データベースには directio を有効にしてロー・パーティションまたはファイル・システムのデバイスを使用すると、メリットがあります。

dsyncio を有効にすると、ソート処理のクエリ・パフォーマンスが極端に低下することがあります (たとえば order by 句をクエリに追加したり、tempdb に dsyncio を有効にした場合)。

一般にソート処理を使用するときは、dsyncio を無効にするか、directio を有効にします。

次の例では、dsyncio を無効にした新しいデバイスに tempdb を作成します。

```
USE master
Go
DISK INIT  name = 'tempdbdev01', physname = '/tempdb_data' , size = '4G', dsync
= 'false'
Go
DISK INIT name = 'tempdblogdev01', physname = '/tempdb_log', size = '4G', dsync
= 'false'
Go
ALTER DATABASE tempdb ON tempdbdev01 = '4G'  LOG ON tempdblogdev01 = '4G'
Go
USE tempdb
Go
EXEC sp_dropsegment 'logsegment', 'tempdb', 'master'
go
EXEC sp_dropsegment 'system', 'tempdb', 'master'
go
EXEC sp_dropsegment 'default', 'tempdb', 'master'
go
```

tempdb 用にすでにデバイスを設定してある場合、必要なのは dsyncio を無効にすることだけですが、Adaptive Server の再起動が必要です。

```
EXEC sp_deviceattr 'tempdbdev01', 'dsync', 'false'
Go
EXEC sp_deviceattr 'tempdblogdev01', 'dsync', 'false'
```

ロー・ロックのシステム・カタログ

Adaptive Server 15.0 では、メッセージ・テーブル、偽のテーブル (ロー指向ではないテーブル)、ログを除いて、システム・カタログはロー・ロックです。現在、これらのテーブルにはクラスタード・インデックスはありませんが、代わりに “placement” インデックスとして新しいインデックス ID が与えられます。Adaptive Server バージョン 15.0 のデータ・レベルでは、ページは互いに連鎖していません。テーブルの開始位置は設定されなくなり、現在はランダムに生成されます。

より大きなデータベースへの対応

Adaptive Server 15.0 では、ロー・ロックのシステム・カタログが使用されるようになり、いくつかの新しいシステム・カタログが追加された結果、より大きなデータベースが使用されます。

最小サイズは、次のように変更されました。

- データベースの最小サイズ − 6 アロケーション・ユニット (1536 ページ、各 2K ページは 3MB)
- マスタ・データベースの最小サイズ − 26 アロケーション・ユニット (6656 ページ、各 2K ページは 13MB)
- tempdb の最小サイズ − 4MB と 6 AU のいずれか大きい方
- sybsystemprocs の最小サイズ − 124MB (132MB 推奨)

default database size 設定パラメータのデフォルト値は、次の値に変更されました。

- 2K ページ − 3MB
- 4K − 6MB
- 8K − 12MB
- 16K − 24MB

アップグレード対象のデータベースに空き領域が十分にあることを確認してください。カタログの変更に必要な領域のサイズは preupgrade ユーティリティで計算されますが、アップグレードを実行するときは、使用可能な領域がデータベースに豊富にある必要があります。10GB 未満のデータベースでは、最低でも 25% の空き領域を作ることをおすすめします。また、トランザクション・ログをできるだけクリアしてください。カタログの変更ではログ領域も必要とされるので、完全なデータベース・ダンプを実行した後は、トランザクション・ログをトランケートします。

アップグレードの前に、使用可能なディスク領域の約 50% だけが master データベースで使用され、新しいストアド・プロシージャを格納できるディスク領域が sybsystemprocs にあることを確認します。master データベースのダンプを作成し、トランザクション・ログをダンプしてトランケートします。トランザクション・ログは、master データベースの領域が消費される大きな要因です。

DBExpert のようなユーティリティでは、抽象クエリ・プランが使用されます。このプランは、system セグメントにキャプチャされます。カタログの拡張に必要な領域サイズより大きな空き領域が system セグメントにあることを確認してください。Adaptive Server 15.0 では、system セグメント領域を使用する sysquerymetrics 機能が追加されました。アップグレードの前に、既存の system セグメントが小さなディスクまたは空き領域が少ししかないディスクに制限されている場合は、system セグメントを他のデバイスにまで拡張して、sysquerymetrics に備えてください。

#temp テーブルの変更点

15.0 より前のバージョンでは、#temp テーブルの名前は 30 文字までに制限されました。具体的には、tempdb 内のユーザ・テンポラリ・テーブルの名前は、ハッシュ記号 (#)、固有の 12 文字、170 バイトのハッシュで構成されました。名前が 12 文字に満たない場合、12 文字に足りない部分にアンダースコアが埋め込まれました。

次の例では、#temp_t1_____0000021008240896 という名前のテンポラリ・テーブルを作成します。

```
select *
into #temp_t1
from mytable
where …
```

Adaptive Server 15.0 では、テンポラリ・テーブル名が次のように変更されました。

- アンダースコアを埋め込まない
- 固有の文字に制限がない
- オブジェクト名には 30 文字の制限がない

この変更は、アプリケーションに影響しません。ただし、Adaptive Server 15.0 で構築されたアプリケーションは、Adaptive Server 12.5 への下位互換性を提供しません。

次の例では、Adaptive Server バージョン 12.5 およびバージョン 15.0 でテンポラリ・テーブルを作成する正しい手順と誤った手順を示します。

次の例は、Adaptive Server 12.5 では失敗しますが、Adaptive Server 15.0 では成功します。バージョン 12.5 で失敗するのは、アンダースコアが自動的に埋め込まれるので、2 つのテーブルが同じ名前になるからです。

```
create table #mytemp (…)
create table #mytemp___ (…)
```

次の例は、Adaptive Server 12.5 では失敗します。テンポラリ・テーブルの名前が 12 文字にトランケートされるからです。Adaptive Server 15.0 では成功します。このバージョンでは、テーブル名がトランケートされません。

```
create table #t12345678901 (…)
create table #t1234567890123 (…)
```

次の例は、Adaptive Server 12.5 では同じテーブルを参照しますが、Adaptive Server 15.0 では別のテーブルを参照します。原因は、前記の例と同じ (名前のトランケート) です。

```
select * from #t12345678901
select * from #t1234567890123456
```

通常の区切り識別子の新しい制限

Adaptive Server バージョン 15.0 では、区切り識別子を二重引用符 (“) または角カッコ ([ ]) で囲みます。また、変数のサイズは最大 254 バイト (@ が 1 バイトにカウントされるため) です。

拡張された識別子は、次の要素に適用されます。

- テーブル名
- カラム名
- インデックス名
- ビュー名
- ユーザ定義のデータ型
- トリガ名
- デフォルト名
- ルール名
- 制約名
- プロシージャ名
- 変数名
- JAR 名
- 関数名
- 時間範囲の指定
- ライトウェイト・プロセス (LWP) または動的文の名前
- アプリケーション・コンテキスト名

次のシステム・テーブルは、長い識別子を理由に変更されました。

- sysattributes
- sysaudits01-08
- syscacheconfig
- syscolumns
- sysconfigures
- sysindexes
- sysjars
- sysobjects
- sysprocesses

- systimeranges
- systypes

長い識別子が原因で起こる可能性がある問題を以下に示します。

- スタック使用量の増加
- ディスク領域の使用量の増加
- 長い識別子に対応するために、ユーザ定義のストアド・プロシージャまたはアプリケーションを変更する必要がある場合があります。

SySAM ライセンス・マネージャ

SySAM ライセンス・マネージャは、Sybase ライセンシングを実施します。基本機能と購入したオプションの両方について、そのプラットフォームの有効なライセンスを要求します。また、SySAM は EBF を受け取る資格を顧客に提供します。各ライセンスは、特定の Adaptive Server ホストまたはライセンス・サーバ・ホストに関連付けられています。

SySAM 2.0 を使用する Adaptive Server 15.0 は、古い SySAM テクノロジを使用する Sybase 製品と共存できます。

SySAM の詳細については、使用しているプラットフォームの『インストール・ガイド』と『設定ガイド』、または SySAM Web サイト (http://www.sybase.com/sysam) を参照してください。

buildmaster の廃止

12.5 より後の Adaptive Server リリースでは、buildmaster ユーティリティは使用されません。この機能は、dataserver -b オプション (Windows では sqlserver.exe) に組み込まれています。このオプションは、ビルド・モードで動作します。dataserver コマンドを使用すれば、論理ページのサイズが 2K、4K、8K、および 16K のマスタ・デバイスと master データベースを作成できます。

新しい Adaptive Server を作成するには、-b オプションと -z オプションを使用して dataserver を発行します。たとえば、デフォルトの論理ページ・サイズ (2K) を使用して 100MB のマスタ・デバイスを構築し、サーバを起動するには、次のコマンドを発行します。

```
dataserver -d /var/sybase/masterdb.dat -b100M -sMASTER2K
```

論理ページ・サイズ 4K を使用して 100MB のマスタ・デバイスを構築するには、次のコマンドを発行します。

```
dataserver -d /var/sybase/masterdb.dat -b100M -z4K -sMASTER4K
```

dataserver ユーティリティの詳細については、『ユーティリティ・ガイド』を参照してください。

警告！ マスタ・デバイスと master データベースへの変更と buildmaster コマンドの廃止によって、アップグレード後にマスタ・デバイスまたは master データベースが失われた場合のリカバリがより複雑になりました。アップグレード前に、master データベースの物理ダンプと bcp コピーの両方を行ってください。

第 6 章

安定性とパフォーマンスの確認

この章は、テスト方法の評価とテスト・プランの作成に役立ちます。

概要

このテストの主な目的は、マイグレーション後に次の点を確認することです。

- アプリケーションの動作が予測可能であること。
- アプリケーションと運用サービスのレベルが保持されているか、それ以上のレベルであること。
- テスト・システムと運用システムが安定しており、データが安全であること。
- アップグレードが成功し、運用システムに悪影響を与えていないこと。

テスト環境の設定

専用のハードウェア設定 (サブネットを含む) を設定し、運用システムと正確に同じ Adaptive Server を再現することが理想です。同一のシステムを作成すると、比較を有効に行い、マイグレーション作業の一部として実際の調整を行うことができます。また、必要に応じて、テスト・システムを運用システムに切り替えることができます。次の各項の説明に従って、テスト・システムを作成します。

- バックアップの作成
- スクリプトによるテスト・システムの作成
- バックアップのロードによるデータベースの作成
- モニタリング・テーブルのインストール
- テスト環境が正確な複写ではない場合

バックアップの作成

運用システムのバックアップを作成します。これらのバックアップを使用して、テスト・システムに移植したり、運用システムをリストアしたりできます。

スクリプトによるテスト・システムの作成

「第 2 章 環境の文書化」で収集、記述、またはリバース・エンジニアリングしたオブジェクト作成スクリプトを使用して、運用システムと同じテスト環境を構築します。

バックアップまたは bcp スクリプトを使用して、テスト・データベースに移植します。

注意 新しいデータベースを作成してから bcp ファイルをロードする場合、運用システムに存在する可能性のある断片化を減らしたり、データベースのパフォーマンス特性を変更したりできます。運用環境を再構築しない場合は、「バックアップのロードによるデータベースの作成」(87 ページ) の説明に従って、データベースを作成します。

テスト・データベースの作成直後にテスト・データベースで dbcc コマンドを実行して、問題がないかどうかを確認します。

モニタリング・テーブルのインストール

Adaptive Server バージョン 12.5.x 以降では、モニタリング・テーブル (または MDA テーブル) を使用できます。

Adaptive Server でモニタリング・テーブルを設定すると、Adaptive Server はモニタリング・テーブルと sysquerymetrics を使用して、(特にアップグレード後に) 影響を受けたクエリと、システム・リソースの使用量の変化を検出します。

モニタリング・テーブルを設定するには、loopback サーバ・エントリを作成し、mon_role 権限を追加し、最新の *installmontable* スクリプトからモニタリング・テーブルをインストールします。以前のリリースのモニタリング・テーブルをインストールし、EBF を適用している場合は、テーブルを再インストールする必要がある可能性があります。

モニタリング・テーブルに加えて、sysquerymetrics システム・テーブルと sp_metrics は、パフォーマンスが低いクエリを識別するのに役立ちます。

モニタリング・テーブルの詳細については、『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』を参照してください。

バックアップのロードによるデータベースの作成

運用環境を再構築しない場合は、create database コマンドの for load オプションを使用してテスト・データベースを作成します。これによって、現在の運用環境の断片化と密度をより忠実に再現するテスト・データベースを作成できます。

1　create database コマンドの for load オプションを使用して、データベースを作成します。

2　運用データベースから作成したバックアップをロードします。

3　バージョン 15.0 でないデータベースを自動的にアップグレードする online database コマンドを発行します。

コマンドの構文については、『リファレンス・マニュアル：コマンド』を参照してください。

テスト環境が正確な複写ではない場合

テストで運用システムより小規模なシステムを使用する必要がある場合、使用するオペレーティング・システム・レベル、ドライバ、ディスクの種類などのコンポーネントを運用システムと同じにしてください。

テスト・システムのディスク領域やメモリが少ない場合は、オプティマイザの決定が変わらないように同じデータ配分を保った状態で、データベースの比率を維持してスケール・ダウンします。一貫した I/O レートを確保するようにメモリをスケール・ダウンします。

使用可能なデバイス間のデータ・レイアウトをできるかぎり再現します。

テスト・システムの方が、使用する CPU の数が少なくなる場合は、トランザクション到着率 (負荷) と同時ユーザを、比率を維持したまま調整します。

テスト対象のアプリケーションの優先順位設定

すべてのアプリケーション関数をテストすることは困難であるため、ユーザ入力を収集して、最も重要なトランザクションを特定します。次の各項で説明する方法で、それらの関数のテストを記述します。このときにテストしない関数は、ユーザ受け入れテストで検証できます。

パフォーマンス基準の確立

マイグレーション・プランによっては、現在のシステムで得られるパフォーマンスと同等またはそれ以上のパフォーマンスが必要になることがあります。たとえば、次のようなガイドラインの適用を決定することがあります。

- 複写使用の並列マイグレーションの場合：
  - 複写メカニズムのオーバヘッドを測定します。たとえば、複写のコストが 10% の場合は、リリース 15.0 のパフォーマンスが 10% 上がるように調整してから並列オペレーションを開始します。
  - 24 時間運用の場合、最初は現在と同等のパフォーマンスを目標とすることが妥当な判断です。
- 複写なしのカットオーバの場合、新旧システム間で同等のパフォーマンスが得られることを目指します。最初の週は、現在と同等のパフォーマンスが妥当な目標です。
- 段階的カットオーバの場合、予測される最も高いパフォーマンスが必要です。運用サーバのカットオーバ後に運用負荷のパフォーマンス調整を行うのが最適です。パフォーマンスが十分に向上するように調整し、テストが成功したらすぐに、運用サーバのカットオーバを実行できます。

マイグレーション方法については、「第 3 章 マイグレーション・プランの作成」を参照してください。

フォールバック手順の開発

テストを実行する前に、問題が発生したときにテスト・システムを既知の状態に戻すことができるようにしておいてください。運用システムをマイグレートするときも同じフォールバック・プランを使用します。

単純なテスト・サイクルは、不要な変更を取り消した方が早く実行される場合があります。ただし、この方法は、ベンチマーク・テスト時の「定期的な実行」にはおすすめしません。定期的な実行のたびに、システムを既知の状態に戻すために後でバックアップからリストアします。

運用システムのアップグレードと同様に、テスト・システムのアップグレードの前後にすべてのデータベースをバックアップします。バックアップは、ディスクのデータ・レイアウトを保存して、断片化やページ分割による混乱を避けるのに役立ちます。

ソース・コード・コントロールを確保するには、スクリプトを使用して、オブジェクトに対するすべての変更を行います。これによって、必要に応じて、より簡単に環境を再作成できます。

テスト方法のまとめ

テスト・プランでは、さまざまなテスト方法を使用できます。表 6-1 は、各種テスト方法とツールのメリットとデメリットを示します。

表 6-1: テスト方法のメリットとデメリット

方法	説明	メリット	デメリット
アドホック・テスト	重要なアプリケーション・プロセス、画面、レポートを 1 つずつ手動でテストする	• 実装が簡単 • フロントエンド・アプリケーションとバックエンド・サーバをテストする	• 複雑なアプリケーションの場合、コードの対象範囲が非常に狭い • 応答時間が決定要素である場合、フロントエンドとバックエンドのボトルネックの区別が困難 • 運用のマルチユーザ負荷を実現できないため、同時実行性と容量の問題が残る
手動のパフォーマンス・スクリプトとその事例	入力を指定して、既知の出力と比較する	• 実装が簡単 • リグレッション・テスト・スイートの共通基盤 • バックエンドに集中することにより、問題の原因の特定に役立つ	• バックエンド・サーバのみをテストする • 運用のマルチユーザ負荷を実現できないため、同時実行性と容量の問題が残る • アドホック・クエリ・テストを実行しない • プロセスまたはトランザクション・プロファイルの確かな分析に依存する

方法	説明	メリット	デメリット
キーストローク取得	アプリケーションでのキーストロークとマウス・クリックを記録、再生する	• フロントエンド・アプリケーションとバックエンド・サーバをテストする • 強力な言語機能とループ機能を備えており、マルチユーザの同時実行性と容量のテストで入力を処理できる	• 処理要件が高く、追加のハードウェアが必要 • マルチユーザ・テスト・シミュレーションを作成するために開発時間が増えたり、テスト・ハーネスのデバッグ時間が増える • プロセスまたはトランザクション・プロファイルの確かな分析に依存する
同時実行性と容量のテスト	サード・パーティの負荷テスト・ツールを使用する	• フロントエンドとバックエンドの両方をテストする • 言語構成体とループ構成体でこのツールが形成される	• テスト・ツールに対する処理要件が高い－追加のハードウェアが必要であり、追加しないと結果に偏りが生じる可能性がある • マルチユーザ・テスト・シミュレーションを作成するために、習得と開発に時間がかかる • テスト・ハーネスのバグのリスクによって、結果に偏りが生じる可能性がある • プロセスまたはトランザクション・プロファイルの確かな分析に依存する
トランザクション生成	シンクライアントでトランザクションのユーザ実行をシミュレートする	• 強力なマルチユーザ負荷テスト • バックエンド・サーバの問題に重点が置かれる	• 通常、習得と開発にかかる時間はキーストローク取得ツールより少ないが、マルチユーザ・テスト・シミュレーションを作成するために開発時間が増える • 結果に偏りが生じるのを避けるためにテスト・ハーネスのデバッグ時間が増える • プロセスまたはトランザクション・プロファイルの確かな分析に依存する
運用負荷取得	ツールを使用して、パフォーマンス特性やセマンティック特性など運用環境での実際のトランザクションを取得して、分析のためにテスト環境で再実行する	• アドホック・クエリなど、実際の運用負荷をテストする • トランザクション・プロファイルの分析がほとんどまたはまったくない場合に特に有用	• 運用環境に新しいソフトウェアを導入する • 有効なパフォーマンス分析を行うために、運用システムとテスト・システムの設定が同一であることが必要

パフォーマンス・スクリプトの記述

この項では、パフォーマンス・スクリプト記述の基本事項について説明します。

ベンチマーク・スクリプトの記述

通常、アプリケーションをベンチマークとして書き直すのではなく、ベンチマーク専用のスクリプトを記述する必要があります。

ベンチマーク・スクリプトを記述するには、次のようにします。

- 各トランザクションの funcs.c に関数を追加する。
- 必要な実行時データ (選択するプライマリ・キーや挿入するデータなど) を生成する。
- SQL またはストアド・プロシージャを Adaptive Server に送信するためのコードを記述する (dbsqlexec() 呼び出しなどを使用)。
- 結果セットを処理するためのコードを記述する。
- システム・プロシージャやシステム・テーブル内でより簡単に特定できるように、各トランザクションを明示的に指定する (“begin tran cust_update” など)。
- ストアド・プロシージャ・ベースのシステムの場合は、ストアド・プロシージャが機能するようにパラメータが設定されていることを確認する。有用なテストを実施するためにパラメータを変更する必要がある場合は、必要な論理を追加する。

パフォーマンス・シミュレーションでは、ボリュームが重要です。通常、機能はアプリケーションと大体同じで、アプリケーションの通常の運用ボリュームで実行されるスクリプトの方が、機能はアプリケーションとまったく同じで、実行されるボリュームが半分のスクリプトより優れています。

負荷が Transact-SQL を発行するクライアント PC に基づいている場合、サード・パーティのパフォーマンス・モニタ・ツールを使用して、データ・ストリームを取得します。

ドライバ

次のドライバが必要です。

- 一般エラー処理
- デッドロック処理
- 結果処理
- 時間測定
- 実行時データ生成

一般エラー処理

エラーが発生した場合、テストの要件に応じて、トランザクションを破棄してカウントしないか、またはトランザクションを再開して全体の応答時間をカウントします。

デッドロック処理

デッドロックされたトランザクションを再実行して、現実的な時間測定を行う必要があります。デッドロックがある場合とない場合の平均応答時間をカウントできます。

結果処理

結果セット全体をフェッチしてクライアントに戻し、それをローカル変数にバインドすることによって、クエリ結果を取得できます。ファイルに結果を出力すると、追加のファイル I/O とオペレーティング・システム・バッファが必要になり、かかる時間が長くなることがあります。

時間測定

時間測定は、どのレベルでもボトルネックが発生する可能性のある多層アプリケーションで特に重要です。通常、ビジネス関数ではなくデータベース・トランザクションの時間を測定します。トランザクションの送信から、結果セットの最後のローが処理されるまでの時間を測定します。後から、論理ビジネス・オペレーションを集計できます。

問題の特定と解決には、細分性が重要です。重要な測定値は、次のとおりです。

- スループット (秒／分／時間あたりのトランザクション数)
- トランザクションあたりの平均応答時間と最大応答時間
- トランザクションごとの応答時間範囲のヒストグラム。次に例を示します。
  - 1 秒未満
  - 1 ～ 2 秒
  - 2 ～ 3 秒
  - 3 秒以上

実行時データ生成

実行時に生成されたデータを使用する場合、キーの偏りが問題になることがあります。select が非現実的な高キャッシュ・ヒット率を示したり、insert、update、delete で同時実行性の問題が発生する可能性があります。これらの問題を避けるには、特定のビジネス状況を再現するのでないかぎり、テストでキー値の範囲全体を使用します。

トランザクションごとに個別のファイルを使用すると、偏りを回避できますが、ユーザを同期させるためのコードを追加する必要があります。ユーザごとに個別のファイルを使用すると、ファイルを作成する作業が増えますが、偏りは防ぐことができます。

メモリ・ジェネレータを使用すると、偏りを回避し、テストを簡単に管理できます。ただし、その場合、ベンチマークは 100% 繰り返し可能ではありません。

テストのまとめ

この項では、新旧の機能、マルチユーザ負荷でのパフォーマンス、統合、ユーザ受け入れなどの特定の問題をターゲットとするテストの完全なテスト・サイクルをまとめます。

表 6-2: テストの説明

段階	目的	最適な方法
機能テスト	各アプリケーションまたはプロセスについて、次の点を処理する。 • 明らかなバグが存在するか。 • トランザクションが同じ結果を返すか。 • アプリケーションがどこかでブレークするか。 新しい機能を使用する場合は、次の点も調べる。 • そのリリースが期待どおりの機能を提供しているか。 • 新機能の制限は何か。	シングルユーザ： • アドホック・テスト • 手動のテスト・スクリプトとその事例 • 既存のアプリケーション・テスト・スイート
ストレス・テスト (ベンチマーク)	非常に大きい負荷を使用して、次の点を処理する。 • マルチユーザ負荷に関連するコード・エラーが存在するか。 • 重要なトランザクションのパフォーマンスが同等または向上しているか。 • 新しいリリースは負荷のある状態で安定しているか。	マルチユーザ： • キーストローク取得 • トランザクション・ジェネレータ • 運用負荷取得
統合テスト	次のようなすべてのシステム・コンポーネントが問題なく連携することを確認する。 • バッチ処理 • オンライン・トランザクション処理 (OLTP) • 意思決定支援システム (DSS) とアドホック・クエリ • バックアップ、リカバリ、dbcc コマンドなどのオペレーション • Adaptive Server 以外の Sybase 製品 • サード・パーティ製品	テスト・スイートですべてのシステム・コンポーネントをモデル化する

段階	目的	最適な方法
エンド・ユーザ受け入れテスト	環境に固有の受け入れテストを実行する。以前の段階で優先されなかった関数もテストする。 **注意** 問題なく進めば、他の段階でほとんどの問題が検出されます。	標準受け入れテスト
マイグレーション・プラン最終テスト	• アップグレード／マイグレーション・プラン全体をひと通り調べて、十分な準備が行われているかどうかを確認する。 • フォールバック手順が機能することを確認する。 • 不測の事態を特定して、テストする。	フォールバック方式を含むすべてのアップグレード手順を 1 つずつ順に実行する。

クエリ処理の変更の決定

この項では、マイグレーションの際に役立つ、Adaptive Server 15.0 のクエリ処理の変更と機能について特に詳しく説明します。

詳細については、Adaptive Server の『クエリ・プロセッサ』を参照してください。

テスト開始前の作業

クエリ問題の診断を開始する前に、次のことを確認します。

- モニタリング (MDA) テーブルがインストールされており、これにアクセスできること。
- Adaptive Server 12.5 および 15.0 の両方のシステムでクエリによるモニタリングが有効にされていること。
- 必要に応じて sp_configure を実行できるパーミッションがあること。
- 診断出力を取得するためにクエリ・プロセッサでさまざまな set コマンド・オプションをオンにするパーミッションがあること。
- トレース・フラグ 3604 および 3605 をオンにできること。
- 膨大な出力を生成し、現在の領域設定では対応できない可能性があるため、十分なファイル領域があること。
- 抽象クエリ・プランを使用してクエリ・プランを取得できること。また、モニタリング・テーブルを実行してテーブルの動作方法を理解し、文、SQL テキスト、プラン・テキストのパイプ設定に必要な大きさを理解できること。

- 練習用領域として使用するテスト・データベースが作成されていること。旧サーバおよび新サーバの出力をこのデータベースにバルク・コピーして SQL 分析を行うことがあるため、データのコピーが容易になるようテスト・データベースは 15.0 サーバに作成します。
- システム・セグメント用に十分な空き領域 (2GB 以上) があること。

マイグレーション時に影響を受けるクエリの判定

この項では、Adaptive Server バージョン 12.5 以降からバージョン 15.0 へのマイグレーションの際に、影響を受けるクエリを判定する方法について説明します。

マイグレーション時のクエリ処理エンジンの変更によって影響を受けるクエリを識別するには、いくつかの方法があります。以下の各項では、さまざまなシナリオと解決方法について説明します。

この処理は、DBExpert 15.0 の Migration Analyzer で自動化できます。Migration Analyzer は、Adaptive Server 12.5 および 15.0 の間でクエリ・プランと実行統計情報を比較して、変更されたクエリと影響の内容を特定します。

影響を受けるクエリの検索

この項では、Adaptive Server 15.0 の新しいクエリ最適化方法の影響を受けるクエリを見つける 2 つの方法について説明します。

抽象クエリ・プランの取得

抽象クエリ・プランは、クエリを実行するためのクエリ・プランを取得、ロード、再利用します。デフォルトでは、取得されたクエリ・プランは sysqueryplans テーブルに格納されます。Adaptive Server 12.5.2 および 15.0 の両方ともクエリ・ハッシュ・キーを使用するため、同一クエリのクエリ・プランは一致します。

クエリ・プランを取得する手順の概要は、次のとおりです。

1. 12.5 サーバ上でクエリ・プランの取得を有効にします。この指定は、セッション・レベルでは set plan dump on で行い、サーバ・レベルでは sp_configure "abstract plan dump", 1 で行います。
2. 12.5 サーバ上で、テストするアプリケーションの 1 つのモジュールを実行します。
3. abstract query plan dump を無効にし、bcp out sysqueryplans を実行します。
4. 15.0 サーバ上でクエリ・プランの取得を有効にします (手順 1 と同じ操作を行います)。
5. 15.0 サーバ上で、手順 2 で実行したモジュールと同じモジュールを実行します。
6. 15.0 サーバ上で abstract query plan dump を無効にします。
7. 15.0 サーバで、queryplans_125 という名前のテーブルを作成します。

8 12.5 サーバのデータを bcp in を使用して queryplans_125 にコピーします。

9 12.5 サーバ上で、15.0 データを 12.5 サーバのテスト・データベースにコピーします。select into を使用して queryplans_150 という名前のテーブルを作成します。

10 両方のテーブルについて、hashkey、type、sequence のインデックスを作成します。

11 両方のサーバで以下のサンプル・クエリを実行して、プランの相違を識別します。クエリを複雑にして、重複を除外することもできます。

次の例では、変更されたクエリのリストを生成します。

```
select t.hashkey
   into #qpchgs
   from queryplans_125 t, queryplans_150 f
   where t.hashkey = f.hashkey
      and t.sequence = f.sequence
      and t.type = 100-- aqp text vs. sql
      and f.type = 100-- aqp text vs. sql
      and t.text != f.text
union all
select f.hashkey
   from queryplans_150 f
   where f.sequence not in (select t.sequence
            from queryplans_125 t
            where f.hashkey = t.hashkey)
union all
select t.hashkey
from queryplans_125 t
where t.sequence not in (select f.sequence
                  from queryplans_150 f
                    where f.hashkey = t.hashkey)
go
```

次の例では、重複を除外します。

```
select distinct hashkey
   into #qpchanges
   from #qpchgs
go
drop table #qpchgs
go
```

次の例では、識別されたクエリの SQL テキストを取得します。

```
select t.hashkey, t.sequence, t.text
from queryplans_125 t, #qpchanges q
where q.hashkey=t.hashkey
and t.type = 10-- sql text vs. aqp
```

抽象クエリ・プランのテキストを比較するには、最初に、どの 15.0 クエリ・プランが長いかを判断します。

```
select t.hashkey, t.sequence, t.text, f.sequence, f.text
   from ueryplans_125 t, queryplans_150 f
   where t.hashkey = f.hashkey
      and t.sequence=*f.sequence
      and t.hashkey in (select hashkey from #qpchanges)
      and t.type = 100-- aqp text vs. sql
      and f.type = 100-- aqp text vs. sql
union all
```

12.5 サーバでクエリ・プランが長い方のクエリを見つけます (12.5 サーバと 15.0 サーバのクエリ・プランが等しい上記の例の場合、結果セットは重複する可能性があります)。

```
select t.hashkey, t.sequence, t.text, f.sequence, f.text
   from ueryplans_125 t, queryplans_150 f
   where t.hashkey = f.hashkey
      and t.sequence*=f.sequence
      and t.type = 100-- aqp text vs. sql
      and f.type = 100-- aqp text vs. sql
order by t.hashkey, t.sequence, f.sequence
```

アプリケーションが同じクエリを何度も実行することはよくあります。この場合、クエリ・プランが 12.5 サーバと 15.0 サーバで異なると、最後の例の結果セットで重複が生じます。

注意　どのクエリも実行測定基準を取得しないため、クエリ・プランによってパフォーマンス特性が変わるかどうかを出力から判断することはできません。

モニタリング・テーブルの使用

新しいクエリ最適化方法の影響を受けるクエリを見つけるこの方法では、hashkey を使用してモニタリング・テーブル内でクエリをユニークに識別できません。この方法は monSysStatement モニタリング・テーブルと関連パイプに基づいており、monSysStatement、monSysSQLText、monSysPlanText モニタリング・テーブルのみを使用し、monProcessStatement、monProcessSQLText、monProcessPlanText モニタリング・テーブルは使用しません。monSys... で始まるモニタリング・テーブルは「ステートフル」であり、これまでに実行された文の履歴が記録されますが、monProcess で始まるテーブルは、現在実行中の文だけが記録されます。

接続のパイプ・ステータスのコンテキストは現在の接続に対して保持され、他の接続とは関係ありません (モニタリング・テーブルの履歴データへのマルチユーザ・アクセスが可能です)。たとえば、monSysStatement テーブルに対してクエリを実行し、100 ローの結果を取得した場合、次にこのテーブルに対してクエリを実行すると、前回のクエリが終了した後に追加された新しいローだけが表示されます。しかし、別の DBA がこの時点で接続し、monSysStatement に対してクエリを実行すると、結果セットにはすべてのローが含まれています。

ポーリング処理を使用してモニタリング・テーブルを繰り返しサンプリングする場合は、前回のクエリ以降のテーブル内の変更のみが表示されます。再接続する場合は、セッションは新しいセッションとして認識され、元の状態が失われるため、結果セットには、前回のセッションで既に提供されたローが含まれる可能性があります。

パイプのサイズは、設定に合わせて正しく指定します。パイプが小さすぎると文やクエリが文字列バッファから削除されてしまうため、使用可能なパイプの数を増やすか、サンプル・クエリを頻繁に実行する必要があります。たとえば、アプリケーションのモジュールが 100,000 個の文を送信する場合、メモリの制約があるため、パイプを 100,000 に設定することは不可能です。しかし、これらの文が 1 時間にわたって送信されるのであれば、1 分あたり平均して 1,667 個のクエリが送信されることになります。ピーク時にその値が倍になることが予想される場合は、1 分あたり 3,333 個のクエリが送信されます。この状況では、パイプを 5,000 に設定し、サンプリングを毎分行うことで、文が失われることを回避できます。

新しいクエリ最適化方法の影響を受けるクエリを、モニタリング・テーブルを使用して見つける一般的な手順を以下に示します。

1 12.5 サーバのパイプ設定を、statement pipe active、sql text pipe active、statement plan text pipe として指定します。

2 次に示すようなクエリを発行して、monSysStatment テーブル、monSysSQLtext テーブル、monSysPlanText テーブルの tempdb にテンポラリ・レポジトリを作成します。

```
select * into tempdb..monSysStatement
from master..monSysStatement where 1=2
```

3 monSysStatment テーブル、monSysSQLtext テーブル、monSysPlanText テーブルに対してクエリを毎分実行し、結果セットを手順 2 で作成したテンポラリ・テーブルに挿入するモニタリング・プロセスを作成します。次に例を示します。

```
insert into tempdb..monSysStatement (select * from
master..monSysStatement)
```

これを毎分実行するには、`waitfor delay "00:01:00"` と共にクエリをループに配置します。

4 テストするアプリケーションの 1 つのモジュールを実行します。

5 アプリケーションを停止し、モニタリングを停止します。

6 手順 3 で収集したモニタリング・テーブルを bcp out でコピーします。

7 Adaptive Server 15.0 で手順 1 ～ 5 を繰り返します。

8 モニタリング・テーブルと同様の名前にバージョン情報が付加された名前を持つ一連のテーブルを、テスト・データベース内に作成します。たとえば、monSysStmt125、monSysStmt150 などの名前を指定します。

9　手順 8 で作成したテーブルを、手順 3 で収集した情報を使用してロードします (bcp 文または insert...select 文を使用)。

10　monSysStmt125 テーブルと monSysStmt15 テーブルの、SPID、KPID、DBID、ProcedureID、BatchID、ContextID、LineNumber カラムにインデックスを作成します。

これで、monSysStmt125 テーブルと monSysStmt15 テーブルを使用して、15.0 サーバの最適化方法の影響を受けるクエリを検索できます。

クエリを調べる場合は、以下の点について考慮してください。

- SQL クエリの ProcedureID 値は 0 (トリガやプロシージャとは異なる)。
- ユーザのクエリは、BatchID で順番にカウントされる SQL バッチでユニークに識別できる。SPID が送信する最初の SQL バッチには BatchID 1 が設定され、2 番目の BatchID には 2、以降同様に設定されます。
- 各 SQL バッチの中で、トリガ、プロシージャ、またはサブプロシージャの実行に伴いコンテキスト (ネスト・レベル) が変化することがあります。ネストが増えるとコンテキストも増加し、ネスト・プロシージャから実行が削除されるとコンテキストは減少します。
- 行番号は現在のコンテキスト内の文を参照します。
- この方法は Adaptive Server CPU のクロック・チックの長さに基づいており、そのデフォルト値は 100 ミリ秒であるため、この方法の精度は 100 ミリ秒にすぎません。100 ミリ秒未満で実行される文は 0 として表示されます。そのため、たとえば、10 ミリ秒かかっていた挿入が 20 ミリ秒かかるようになった場合、この挿入が 1 日に 100 万回行われるのであれば、レポート・エラーが発生する可能性があります。

同じシーケンスのテスト文を発行した場合、monSysStatements は、先頭の BatchID を除いて両方のサーバで同一です。両方のサーバで SPID と KPID の組み合わせを使用して、クエリを識別できます (アプリケーションが複数の接続を使用していることが前提です)。

たとえば、次のクエリは、15.0 サーバ上で 12.5 サーバより低速に実行される SQL 文のリストを選択します。12.5 サーバでは、文の SPID は 123、KPID は 4567890、BatchID は 101 で始まります。15.0 サーバでは、文の SPID は 24、KPID は 1234567、BatchID は 12 で始まります。

```
select f.BatchID, f.ContextID, f.LineNumber, CPU_15=f.CPUTime,
CPU_125=t.CPUTime,
     Wait_15=f.WaitTime, Wait_125=t.WaitTime,
     Mem_15=f.MemUsageKB, Mem_125=t.MemUsageKB,
     PhysIO_15=f.PhysicalReads, PhysIO_125=t.PhysicalReads,

     LogicalIO_15=f.LogicalReads, LogicalIO_125=t.LogicalReads,
     Writes_15=f.PagesModified, Writes_125=t.PagesModified,
     ExecTime_15=datediff(ms,f.StartTime,f.EndTime)/1000.00,
     ExecTime_125=datediff(ms,t.StartTime,t.EndTime)/1000.00,
     DiffInMS= datediff(ms,f.StartTime,f.EndTime)-
```

```
    datediff(ms,t.StartTime,t.EndTime)
    into #slow_qrys
    from monSysStmt150 f, monSysStmt125
    where f.SPID=24 and f.KPID=1234567
    and t.SPID=123 and t.KPID=4567890
    and t.BatchID=f.BatchID+(101-12)-- calculate offset for Batches.
    and t.ContextID=f.ContextID
    and t.LineNumber=f.LineNumber
    and (datediff(ms,f.StartTime,f.EndTime) >
    datediff(ms,t.StartTime,t.EndTime))
    order by 18 desc, f.BatchID, f.ContextID, f.LineNumber
```

この同じクエリを編集して、12.5 サーバより 15.0 サーバ上で高速に実行するクエリを返すことができます。

```
select f.BatchID, f.ContextID, f.LineNumber, CPU_15=f.CPUTime,
CPU_125=t.CPUTime,
     Wait_15=f.WaitTime, Wait_125=t.WaitTime,
     Mem_15=f.MemUsageKB, Mem_125=t.MemUsageKB,
     PhysIO_15=f.PhysicalReads, PhysIO_125=t.PhysicalReads,
     LogicalIO_15=f.LogicalReads, LogicalIO_125=t.LogicalReads,
     Writes_15=f.PagesModified, Writes_125=t.PagesModified,
     ExecTime_15=datediff(ms,f.StartTime,f.EndTime)/1000.00,
     ExecTime_125=datediff(ms,t.StartTime,t.EndTime)/1000.00,
     DiffInMS= datediff(ms,f.StartTime,f.EndTime)-
datediff(ms,t.StartTime,t.EndTime)
   into #slow_qrys
   from monSysStmt150 f, monSysStmt125
   where f.SPID=24 and f.KPID=1234567
     and t.SPID=123 and t.KPID=4567890
     and t.BatchID=f.BatchID+(101-12)-- calculate offset for Batches.
     and t.ContextID=f.ContextID
     and t.LineNumber=f.LineNumber
     and (datediff(ms,f.StartTime,f.EndTime) <
datediff(ms,t.StartTime,t.EndTime))
     order by 18 desc, f.BatchID, f.ContextID, f.LineNumber
```

sysquerymetrics の使用

この方法では、sysqueyrmetrics (論理 IO、CPU 時間、経過時間などのパフォーマンス測定基準カラムを含みます) を使用して、15.0 クエリ・プロセッサの影響を受けるクエリを見つけます。この方法では、ハッシュ・キーに基づくが実行統計情報も生じる厳密なクエリ一致を使用します。

この方法の場合、クエリ・プランではなくクエリ・テキストが表示されるため、最初に 15.0 サーバで低速なクエリを識別し、そのプランを以前のサーバのプランと比較します。この方法の最初の部分は抽象クエリ・プランを取得する方法と似ていますが、sysquerymetrics を使用して詳細な情報を取得します。

sysquerymetrics テーブルの詳細については、『リファレンス・マニュアル：テーブル』を参照してください。

手順は次のとおりです

1　12.5 サーバ上でクエリ・プランの取得を有効にします。この指定は、セッション・レベルでは set plan dump on で行い、サーバ・レベルでは sp_configure "abstract plan dump", 1 で行います。

2　12.5 サーバ上で、テストするアプリケーションの 1 つのモジュールを実行します。

3　12.5 サーバ上で、抽象クエリ・プランのダンプを無効にし (set plan dump off)、bcp out sysqueryplans を実行します。

4　set plan dump *group_name* on を使用して、15.0 サーバ上でクエリ・プランの取得を有効にします。

5　15.0 サーバ上で測定基準の取得を有効にします。この指定は、サーバ・レベルでは sp_configure "enable metrics capture", 1 で行い、セッション・レベルでは set metrics_capture on で行います。

6　15.0 サーバ上で、手順 2 で実行したモジュールと同じモジュールを実行します。

7　15.0 サーバ上で抽象クエリ・プランのダンプを無効にします (set plan dump off)。

8　15.0 サーバ上で、queryplans_125 という名前のテーブルをテスト・データベース内に作成し、12.5 データをバルク・コピーします。

9　15.0 サーバ上で、queryplans_150 という名前のテーブルをテスト・データベース内に作成し、15.0 サーバの抽象クエリ・プラン・データをこのテーブルにコピーします。

10　両方の抽象クエリ・プラン・テーブルについて、hashkey、type、sequence カラムのインデックスを作成します。

11　sp_metrics を使用して sysquerymetrics データのバックアップを作成するか、テスト・データベース内のテーブルにコピーします。

12　以下のクエリを実行して、プランの相違を識別します。

次のクエリは、15.0 サーバで変更されたクエリのリストを作成します。

```
select t.hashkey
into #qpchgs
from queryplans_125 t, queryplans_150 f
where t.hashkey = f.hashkey
and t.sequence = f.sequence
and t.type = 100-- aqp text vs. sql
and f.type = 100-- aqp text vs. sql
and t.text != f.text
union all
select f.hashkey
from queryplans_150 f
where f.sequence not in (select t.sequence
```

```
                    from queryplans_125 t
                    where f.hashkey = t.hashkey)
    union all
    select t.hashkey
    from queryplans_125 t
    where t.sequence not in (select f.sequence
                    from queryplans_150 f
                    where f.hashkey = t.hashkey)
    select distinct hashkey
    into #qpchanges
    from #qpchgs
    go
    drop table #qpchgs
```

次のクエリは、15.0 で変更され、低速で実行するクエリのリストを選択します。このクエリによって、15.0 サーバでより高速またはより低速で実行するクエリが分かるわけではありません。指定の制限を越えており、そのクエリ・プランが 12.5 サーバ・リリースとは異なる 15.0 サーバ内のクエリが識別されます。

```
select hashkey, sequence, exec_min, exec_max, exec_avg,
     elap_min, elap_max, elap_avg, lio_min,
     lio_max, lio_avg, pio_min, pio_max, pio_avg,
     cnt, weight=cnt*exec_avg
     qtext
from <db>..sysquerymetrics-- database under test
where gid = <gid>-- group id sysquerymetrics backed up to
and elap_avg > 2000-- slow query is defined as avg elapsed time > 2000
and hashkey in (select hashkey from #qpchanges)
```

長時間実行されているストアド・プロシージャの検出

この項では、sysquerymetrics を使用して、プロシージャ (サブプロシージャを含む) 内の各行および文の、文レベルでの統計情報を追跡することによって、長時間実行されているストアド・プロシージャを識別する方法について説明します。これは、プロシージャ内で最適化を実現するパラメータを取得する上で役立ちます。この項では、sysquerymetrics を使用して長時間実行されているストアド・プロシージャを識別する方法について説明します。

この機能は Adaptive Server 15.0 の新機能であるため、以下の手順を使用して以前のリリースのクエリを比較することはできません。マイグレーションの際にクエリ・プランが変更されるため、クエリはしばしば低速で実行されます。

この方法を使用する前に、以下の点について考慮してください。

- 結果セットでは、LineNumber が省略されたり、繰り返されたりすることがある。これは、クエリ内のループや if ...else 文が原因です。このため、プロシージャ内にあるループの中のサブプロシージャは、ContextID は同じでも、StartTimes が異なることがあります。

- dump および load または mount database を使用してアップグレードを適切に実行すると、データベース内のユーザ・オブジェクトの objectID は同一で、DBID は異なる。
- プロシージャ・レベルの統計情報を取得するには、値を集計する必要がある。

注意 以下に示すのは単純なスクリプトですが、個々の手順を説明するために分割されています。

a　Adaptive Server 12.5 の個々のプロシージャ・レベルの集計を求めるには、次のクエリを発行します。

```
select DBID, ProcedureID, StartTime, EndTime=max(EndTime),
   ElapsedTime=datediff(ms,StartTime,max(EndTime))/1000.00,
   CPUTime=sum(CPUTime), WaitTime=sum(WaitTime),
   LogicalReads=sum(LogicalReads),PhysicalReads=sum(PhysicalReads),
   PagesModified=sum(PagesModified)
into #procExecs125
from monSysStmt125
group by DBID, ProcedureID, Starttime
```

b　Adaptive Server 15.0 の個々のプロシージャ・レベルの集計を求めるには、次のクエリを発行します。

```
select DBID, ProcedureID, StartTime, EndTime=max(EndTime),
   ElapsedTime=datediff(ms,StartTime,max(EndTime))/1000.00,
   CPUTime=sum(CPUTime), WaitTime=sum(WaitTime),
   LogicalReads=sum(LogicalReads), PhysicalReads=sum(PhysicalReads),
   PagesModified=sum(PagesModified)
into #procExecs150
from monSysStmt150
group by DBID, ProcedureID, Starttime
```

c　両方のクエリとも個々のプロシージャ実行レベルで発生するため、このクエリを 2 回以上実行する場合は、次のクエリを発行して結果セットの平均を求めます。

```
select DBID, ProcedureID, ExecCnt=count(*), ElapsedTime=avg(ElapsedTime),
   CPUTime=avg(CPUTime), WaitTime=avg(WaitTime),
   LogicalReads=avg(LogicalReads), PhysicalReads=avg(PhysicalReads),
   PagesModified=avg(PagesModified)
into #procExecAvgs125
from procExecs125
group by DBID, ProcedureID
```

d　15.0 サーバでより低速に実行されたストアド・プロシージャを見つけるには、次のクエリを発行します。

```
select o.name as 'ProcName', o.type, ExecCnt=t.ExecCnt,
CPU_125=t.CPUTime, CPU_150=f.CPUTime,
   WaitTime_125=t.WaitTime, WaitTime_150=f.WaitTime,
```

```
   LogicalIO_125=t.LogicalReads, LogicalIO_150=f.LogicalReads,
   PhysReads_125=t.PhysicalReads, PhysReads_150=f.PhysicalReads,
   Writes_125=t.PagesModified, Writes_150=f.PagesModified,
   Elapsed_125=t.ElapsedTime, Elapsed_150=f.ElapsedTime
from #procExecAvgs125 t, #procExecAvgs150, <db>..sysobjects o
where t.ProcedureID=o.id
and t.ProcedureID=f.ProcedureID
and t.ElapsedTime < f.ElapsedTime
```

このクエリの結果セットと monSysSQLText をジョインすることによって、低速に実行されたストアド・プロシージャの SQL テキストを表示できます。

パラメータ・セットによって実行の平均に偏りが生じることがあるため、先頭の集計テーブル (#procExecs) と monSysSQLText をジョインして、各バージョンの実行測定基準をそれぞれ表示します。

e 15.0 サーバでより高速に実行されたストアド・プロシージャを見つけるには、次のクエリを発行します。

```
select o.name as 'ProcName', o.type, ExecCnt=t.ExecCnt,
   CPU_125=t.CPUTime, CPU_150=f.CPUTime,
   WaitTime_125=t.WaitTime, WaitTime_150=f.WaitTime,
   LogicalIO_125=t.LogicalReads, LogicalIO_150=f.LogicalReads,
   PhysReads_125=t.PhysicalReads, PhysReads_150=f.PhysicalReads,
   Writes_125=t.PagesModified, Writes_150=f.PagesModified,
   Elapsed_125=t.ElapsedTime, Elapsed_150=f.ElapsedTime
from #procExecs125 t, #procExecs150, <db>..sysobjects o
where t.ProcedureID=o.id
  and t.ProcedureID=f.ProcedureID
  and t.ElapsedTime > f.ElapsedTime
```

長時間実行されているクエリを sysquerymetrics を使用して見つけるには、次の手順に従います。

1 12.5 サーバで、statement pipe active、sql text pipe active、statement plan text pipe を設定します。

2 次に示すようなクエリを発行して、monSysStatment テーブル、monSysSQLtext テーブル、monSysPlanText テーブルのテンポラリ・テーブルを tempdb に作成します。

```
select * into tempdb..monSysStatement from
master..monSysStatement where 1=2
```

3 monSysStatment テーブル、monSysSQLtext テーブル、monSysPlanText テーブルに対してクエリを毎分実行し、結果セットを手順 2 で作成したテンポラリ・テーブルに挿入するモニタリング・プロセスを作成します。次に例を示します。

```
insert into tempdb..monSysStatement (select * from
master..monSysStatement)
```

これを毎分実行するには、`waitfor delay "00:01:00"` と共にクエリをループに配置します。

4　12.5 サーバ上で、テストするアプリケーションの 1 つのモジュールを実行します。

5　アプリケーションを停止し、モニタリングを停止します。

6　tempdb からデータを収集したモニタリング・テーブル情報をバルク・コピーします。

7　15.0 サーバで手順 1 ～ 5 を繰り返します。

8　モニタリング・テーブルと同様の名前にバージョン情報が付加された名前を持つ一連のテーブルを、テスト・データベース内に作成します。たとえば、monSysStmt125、monSysStmt150 などの名前を指定します。

9　手順 8 で作成したテーブルを、手順 3 で収集した情報を使用してロードします (bcp 文または insert...select 文を使用)。

10　monSysStThree3Ducksmt125 テーブルと monSysStmt15 テーブルの、SPID、KPID、DBID、ProcedureID、BatchID、ContextID、LineNumber カラムにインデックスを作成します。

15.0 サーバのクエリ処理に関する問題の診断と修正

この項では、Adaptive Server 15.0 に関する問題を診断する上でのヒントについて説明します。

sysquerymetrics および sp_metrics の使用

Adaptive Server は、sp_metrics を使用して sysquerymetrics データを管理します。収集された最新データは、グローバル ID (GID) が 1 として sysquerymetrics に格納されます。sp_metrics...backup パラメータを使用して以前のデータを保存する場合は、まだ使用されていない 1 より大きい GID を使用する必要があります。次に有効な GID を見つけるには、sysquerymetrics から `max(gid)` を選択し、その値に 1 を加えます。

sp_metrics の例

測定基準の取得を有効にするには、次のように入力します。

```
set metrics_capture on
```

測定基準をフラッシュするには、次のように入力します。

```
sp_metrics 'flush'
```

GID を選択するには (NULL 値または 1 が返される場合は、2 以上の値を使用します)、次のように入力します。

```
select max(gid)+1 from sysquerymetrics
```

データのバックアップを作成するには、次のように入力します。

```
sp_metrics 'backup', '3'
```

測定基準の取得をオフにするには、次のように入力します。

```
set metrics_capture off
```

データを分析するには、次のように入力します。

```
select * from sysquerymetrics where gid = 3
```

データを削除するには、次のように入力します。

```
sp_metrics 'drop', '2', '5'
```

sp_metrics の詳細については、『リファレンス・マニュアル：コマンド』を参照してください。

sysquerymetrics 内のデータのフィルタ

事前に定義された値より小さい値のデータを削除することによって、sysquerymetrics 内のデータをフィルタすることができます。データを削除する前に、allow updates を有効にする必要があります。

測定基準を削除する場合は、開始と終了の範囲が存在する必要があります。たとえば、次のクエリで、3 から始まるグループの測定基準の削除を試みます。

```
sp_metrics 'drop', '2', '5'
```

グループ 2 が存在しないため、削除は失敗します。

sysquerymetrics は、システム・セグメント内で大量の領域を使用します。ただし、領域使用量を削減するには、以下の手順に従います。

1 sp_metrics capture を実行します。

2 取得時間の最後に、次のクエリを発行します。

```
Select max(gid) from sysquerymetrics
sp_metrics 'backup', 'gid'
```

3 次の取得時間を開始します。

4 前の結果をフィルタします。たとえば、looming が 10,000 未満のローを削除します。

5 このシーケンスを 10 ～ 15 回繰り返します。

sysquerymetrics によるリグレッション・テストのサポート

最適化目標、並列リソース、またはテーブルの分割スタイルを変更したり、関数ベース・インデックスなど、他の Adaptive Server 15.0 機能を適用した後で、sysquerymetrics を使用してリグレッション・テストをサポートできます。手順は次のとおりです

- 最初の実行に対して sp_metrics capture を有効にします。この測定基準が参照用に使用されます。
- アプリケーションのモジュールを実行します。

- GID を 2 に設定することによって、参照用の測定基準のバックアップを作成します。
- 測定基準が向上するように、設定を変更します。
- 手順 1 ～ 3 を繰り返し、そのたびに次に高い GID にバックアップを作成します。

sysquerymetrics とハッシュ・キーをジョインすることによって影響を受けたクエリを識別します。次に例を示します。

```
select r.hashkey, r.exec_avg, m.exec_avg, r.elap_avg, m.elap_avg,
     r.lio_avg, m.lio_avg, r.pio_avg, m.pio_avg, r.qtext
from sysquerymetrics r, sysquerymetrics m
where r.gid=2 and m.gid=<#>-- substitute # of current gid
and r.hashkey=m.hashkey
and ((r.exec_avg + (r.exec_avg * 0.1) < m.exec_avg)
     or (r.elap_avg + (r.elap_avg * 0.1) < m.elap_avg)
     or (r.lio_avg + (r.lio_avg * 0.1) < m.lio_avg)
     or (r.pio_avg + (r.pio _avg * 0.1) < m.pio_avg)
```

5 というトレランス要素を追加することを含めて、リグレッションを考慮する値を反映するように、クエリを変更できます。上記の例では、チェックポイント・プロセスの影響を避けるために、参照時間に 10% のトレランス要素が含まれています。

上記の例のユーザは 1 人だけなので、`r.uid = m.uid` は追加されていません。

showplan オプションの使用

Adaptive Server 15.0 の showplan は、使用されている多くの診断トレース・フラグを set オプションに置き換えます。構文は次のとおりです。

```
set option show_option (normal/brief/long/on/off)
```

set option 構文の詳細については、『リファレンス・マニュアル：コマンド』を参照してください。

使用に関する問題

set option を使用する前に、以下の点について考慮してください。

- オプションによっては、クライアント側またはエラー・ログに出力を表示するために、dbcc traceon (3604) または 3605 を設定する必要があります。
- set option show on を実行してから、他のより制限的なオプションを設定します。これによって、show コマンドの一般的なレベルの詳細情報を上書きする brief オプションまたは long オプションを指定できます。
- query metrics capture を有効にすると、set option の出力が生成されないことがあります。

例

次の例では、クエリ・プランの表示、統計情報のレポート、抽象プランの表示を行います (これによって、後で create plan を発行して問題を解決できます)。

```
set showplan on
set option show_missing_stats on
set option show_abstract_plan on
```

不足している統計情報を表示するには、次のクエリを発行します。

```
set option show_missing_stats long
dbcc traceon(3604)
go
DBCC execution completed. If DBCC printed error messages,
contact a user with System Administrator (SA) role.
select * from part, partsupp
where p_partkey = ps_partkey and p_itemtype = ps_itemtype
go
NO STATS on column part.p_partkey
NO STATS on column part.p_itemtype
NO STATS on column partsupp.ps_itemtype
NO STATS on density set for E={p_partkey, p_itemtype}
NO STATS on density set for F={ps_partkey, ps_itemtype}
```

次の例では、インデックス選択や IO コスト計算などのデバッグを行います (以前のリリースではトレース・フラグ 302、310、315 などでこの処理が行われていました)。

```
dbcc traceon(3604)
set showplan on
set option show long
```

set statistics plancost オプション

set statistics plancost オプションによって、クエリ分析を簡略化できます。このオプションを指定すると、各演算子で評価された実際の値を基準にした論理 I/O、物理 I/O、ロー・カウントの予測値が表示され、CPU とソート・バッファ・コストがレポートされます (以下の例では `cpu: 0 bufct: 16`)。この表示を有効にするには、次のように入力します。

```
dbcc traceon(3604)
go
set statistics plancost on
go
```

たとえば、次のクエリを例に考えます。

```
select S.service_key, M.year, M.fiscal_period,count(*)
from telco_facts T,month M, service S
where T.month_key=M.month_key
  and T.service_key = S.service_key
  and S.call_waiting_flag='Y'
  and S.caller_id_flag='Y'
```

```
    and S.voice_mail_flag='Y'
  group by M.year, M.fiscal_period, S.service_key
  order by M.year, M.fiscal_period, S.service_key
```

このクエリは、次のクエリ・ツリーを生成します。

```
                                    Emit
                                     (VA = 7)
                                     12 rows est: 1200
                                      cpu: 500

                                /
                               GroupSorted
                               (VA = 6)
                                12 rows est: 1200

                           /
                          NestLoopJoin
                          Inner Join
                          (VA = 5)
                          242704 rows est: 244857
                    /                     ¥
                   Sort                   IndexScan
                  (VA = 3)                month_svc_idx (T)
                  72 rows est: 24         (VA = 4)
                  lio: 6 est: 6           242704 rows est: 244857
                  pio: 0 est: 0           lio: 1116 est: 0
                  cpu: 0 bufct: 16        pio: 0 est: 0
          /
         NestLoopJoin
         Inner Join
         (VA = 2)

72 rows est: 24
/                     ¥
TableScan             TableScan
month (M)             service (S)
(VA = 0)              (VA = 1)
24 rows est: 24       72 rows est: 24
lio: 1 est: 0         lio: 24 est: 0
pio: 0 est: 0          pio: 0 est: 0
```

set statistics planview が間違ったロー・カウントを予測すると、オプティマイザの予測もオフになります。これは、統計情報の不足や古い統計情報が原因で起こります。これについては、次のクエリで説明します (このクエリも、show_missing_stats が有効な状態で実行されます)。

```
dbcc traceon(3604)
go
set option show_missing_stats on
go
set statistics plancost on
go
```

```
select
            l_returnflag,
            l_linestatus,
            sum(l_quantity) as sum_qty,
            sum(l_extendedprice) as sum_base_price,
            sum(l_extendedprice * (1 - l_discount)) as sum_disc_price,
            sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) as sum_charge,
            avg(l_quantity) as avg_qty,
            avg(l_extendedprice) as avg_price,
            avg(l_discount) as avg_disc,
            count(*) as count_order
from
           lineitem
where
           l_shipdate <= dateadd(day, 79, '1998-12-01')
group by
           l_returnflag,
           l_linestatus
order by
           l_returnflag,
           l_linestatus
go
==================== Lava Operator Tree ====================


                                                    Emit
                                                    (VA = 4)
                                                    4 rows est: 100
                                                    cpu: 800

                                                /
                                                Restrict
                                                (0)(13)(0)(0)
                                                (VA = 3)
                                                4 rows est: 100

                                          /
                                          GroupSorted
                                          (VA = 2)

                                          4 rows est: 100
                                    /
                                    Sort
                                    (VA = 1)
                                    60175 rows est: 19858
                                    lio: 2470 est: 284
                                    pio: 2355 est: 558
                                    cpu: 1900 bufct: 21
                              /
                              TableScan
                              lineitem
```

```
                        (VA = 0)
                        60175 rows est: 19858
                        lio: 4157 est: 4157
                        pio: 1205 est: 4157

============================================================

NO STATS on column lineitem.l_shipdate

(4 rows affected)
```

ロー数の予測値がスキャン・レベルで正しくありません。このクエリには次の述部があります。

```
l_shipdate <= dateadd(day, 79, '1998-12-01')
```

l_shipdate に関する統計情報がない場合、クエリ・プロセッサは任意の値を使用します。この例の場合、クエリ・プロセッサが使用する値によって、60175 という実際のロー・カウントからかけ離れた 19858 という予測ロー・カウントが作成されたため、クエリ・プロセッサはソートの実行を決定しました。ソートされるロー数が実際の数の 3 分の 1 だとクエリ・プロセッサによって予測された場合、ソートのコストはそれほどかからないと判断されます。GroupSorted 演算子のローも 60175 ローから 4 ローへと大幅に減少するため、GroupSorted アルゴリズムのメリットが、多くの場合すべてのデータをメモリにキャッシュするハッシュベースのグループ化アルゴリズムより優先されます。

クエリ・プロセッサは、show_missing_stats オプションからの値に基づいて、l_shipdate カラムに対して update statistics を実行することを決定します。

```
update statistics lineitem(l_shipdate)
```

クエリを再実行すると、次のクエリ・ツリーが生成されます。

```
==================== Lava Operator Tree ====================

                                                    Emit
                                                    (VA = 4)
                                                    4 rows est: 100
                                                    cpu: 0
                                            /
                                            Restrict
                                            (0)(13)(0)(0)
                                            (VA = 3)
                                            4 rows est: 100
                                    /
                                    Sort
                                    (VA = 2)
                                    4 rows est: 100
                                    lio: 6 est: 6
                                    pio: 0 est: 0
                                    cpu: 800 bufct: 16
                            /
                            HashVectAgg
```

```
        Count
        (VA = 1)
        4 rows est: 100
        lio: 5 est: 5
        pio: 0 est: 0
        bufct: 16
/
TableScan
lineitem
(VA = 0)
60175 rows est: 60175
lio: 4157 est: 4157
pio: 1039 est: 4157
```

update statistics を実行すると、TableScan 演算子の予測ロー・カウントは実際のロー・カウントと同じであるため、クエリ・プランは、前の例の Sort と GroupSorted の組み合わせではなく、HashVectAgg (ハッシュベースのベクトル集計) を使用するように変更されます。クエリも高速で実行されます。

クエリを改善するための操作をさらに行うことができます。HashVectAgg 演算子の出力は、予測される rowcount として 100 を示しますが、実際の rowcount は 4 です。カラムのグループ化は l_returnflag と l_linestatus で行われているため、このカラムのペアで密度を作成できます。

```
use tpcd
go
update statistics lineitem(l_returnflag, l_linestatus)
go
set showplan on
go
set statistics plancost on
go
```

クエリを再実行すると、次のクエリ・プランが得られます。

```
QUERY PLAN FOR STATEMENT 1 (at line 2).

4 operator(s) under root

The type of query is SELECT.

ROOT:EMIT Operator

|RESTRICT Operator
|
|   |SORT Operator
|   | Using Worktable2 for internal storage.
|   |
|   |   |HASH VECTOR AGGREGATE Operator
|   |   |  GROUP BY
|   |   |  Evaluate Grouped COUNT AGGREGATE. |   |   |  Evaluate Grouped SUM OR
          AVERAGE AGGREGATE.
```

```
|   |   |  Evaluate Grouped COUNT AGGREGATE.
|   |   |  Evaluate Grouped SUM OR AVERAGE AGGREGATE.
|   |   |  Evaluate Grouped SUM OR AVERAGE AGGREGATE.
|   |   |  Evaluate Grouped SUM OR AVERAGE AGGREGATE.
|   |   |  Evaluate Grouped SUM OR AVERAGE AGGREGATE.
|   |   |  Using Worktable1 for internal storage.
|   |   |
|   |   |   |  SCAN Operator
|   |   |   |  FROM TABLE
|   |   |   |  lineitem
|   |   |   |  Table Scan.
|   |   |   |  Forward Scan.
|   |   |   |  Positioning at start of table.
|   |   |   |  Using I/O Size 2 Kbytes for data pages.
|   |   |   |  With MRU Buffer Replacement Strategy for data pages.

==================== Lava Operator Tree ====================
                                                     Emit
                                                    (VA = 4)
                                                    4 rows est: 4
                                                    cpu: 0

                                               /
                                               Restrict
                                               (0)(13)(0)(0)
                                               (VA = 3)
                                               4 rows est: 4

                                          /
                                          Sort
                                          (VA = 2)
                                          4 rows est: 4
                                          lio: 6 est: 6
                                          pio: 0 est: 0
                                          cpu: 700 bufct: 16
                                     /
                                     HashVectAgg
                                     Count (VA = 1)
                                     4 rows est: 4
                                     lio: 5 est: 5
                                     pio: 0 est: 0
                                     bufct: 16
                                /
                                TableScan
                                lineitem
                                (VA = 0)
                                60175 rows est: 60175
                                lio: 4157 est: 4157
                                pio: 1264 est: 4157
```

HashVectAgg の予測ロー・カウントは、実際のロー・カウントと同じです。

XML としてのクエリ・プラン

Adaptive Server 15.0 では、クエリ・プランを XML で表示できます。これを使用することで、クエリ・プランをグラフィカルに表示する Adaptive Server プラグイン内のプラン・ビューアのような自動化ツールを構築できます。XML クエリ・プランを使用すると、特定のテーブルの統計情報が更新された最後の日時も検索できます。Adaptive Server の以前のリリースでは、optdiag ユーティリティを使用するか、systabstats または sysstatistics に対してクエリを実行することによって、この処理を行っていました。

XML でのクエリ・プランの表示は、1 つの手順で実行できます。showplan は、クエリのテキスト表現として統計情報を提供しません。このため、optdiag の使用、またはシステム・テーブルに対するクエリの実行によって統計情報が最後に更新された日時を判断するには、追加の手順を実行する必要があります。すべての統計情報は XML 出力で 1 つの場所に格納されており、XML の解析はテキストの解析より単純であるため、ツールの開発者は、以前のリリースより簡単に Adaptive Server 15.0 の強化機能を作成できます。

Adaptive Server 15.0 におけるクエリレベルのデバッグ

この項では、Adaptive Server リリース 15.0 でのデバッグ方法について説明します。

統計情報に対する optdiag の使用

optdiag を使用すると、現在の統計情報が古くて新しい統計情報を生成する必要があるかどうかを判断できます。次の例では、optdiag を使用して、le_01 データベースの part テーブルの統計情報を取得します。

```
$SYBASE/ASE-15_0/bin/optdiag statistics le_01.dbo.part -Usa -P

Server name: "tpcd"

Specified database: "le_01"
Specified table owner: "dbo"
Specified table: "part"
Specified column: not specified

Table owner: "dbo"
Table name: "part"

................................................
Statistics for column: "p_partkey"
Last update of column statistics: Sep 13 2005 7:51:39:440PM

Range cell density: 0.0010010010010010
Total density: 0.0010010010010010
Range selectivity: default used (0.33)
In between selectivity: default used (0.25)

Histogram for column: "p_partkey"
Column datatype: integer
```

```
Requested step count: 20
Actual step count: 20
Sampling Percent: 0

Step Weight Value

1 0.00000000 <= 0
2 0.05205205 <= 52

.....................................................
Statistics for column: "p_brand"
Last update of column statistics: Sep 13 2005 7:51:39:440PM

Range cell density: 0.0010010010010010
Total density: 0.0010010010010010
Range selectivity: default used (0.33)
In between selectivity: default used (0.25)
```

統計情報に対する show_final_plan_xml の使用

Adaptive Server 15.0 以降では、XML を使用して、統計情報が最後に更新された日時を判断できます。統計を実行する対象についてクエリを指定すると、そのカラムの統計情報が最後に更新された日時を含めて、クエリにとって有効な統計情報のみが表示されます。optdiag では、使用されているかどうかにかかわらずすべてのインデックスの統計情報が表示され、複数のテーブルが関係する場合は複数回の実行が必要です。次の例では、前の例から part テーブルの統計情報を収集します。

```
1> set plan for show_final_plan_xml to message on
2> go
1> select count(*) from part where p_partkey > 20
2> go

-----------

979
1> select showplan_in_xml(-1)
2> go


-----------
979
<?xml version="1.0" encoding="UTF-8"?>
<query>
             <planVersion> 1.0 </planVersion>
             <statementNum>1</statementNum>
             <lineNum>1</lineNum>
             <text>
                   <![CDATA[
                      SQL Text: select count(*) from part where p_partkey > 20
                   ]]>
                </text>
                   <objName>part</objName>
                   <columnStats>
```

```
                    <column>p_partkey</column>
                    <updateTime>Sep 13 2005 7:51:39:440PM</updateTime>
                </columnStats>
```

XML データのクライアントへの直接送信

次の例では、トレース・フラグ 3604 と client パラメータを使用して、show_final_plan_xml からクライアントへ情報を送信します。

```
1> dbcc traceon(3604)
DBCC execution completed. If DBCC printed error messages, contact a user with
System Administrator (SA) role.
set plan for show_final_plan_xml to client on
go
select * from part, partsupp
where p_partkey = ps_partkey and p_itemtype = ps_itemtype
go
<?xml version="1.0" encoding="UTF-8"?>
<query>
<planVersion> 1.0 </planVersion>
<optimizerStatistics>
<statInfo>
                    <objName>part</objName>
                    <missingHistogram>
                            <column>p_partkey</column>
                            <column>p_itemtype</column>
                    </missingHistogram>
                    <missingDensity>
                            <column>p_partkey</column>
                            <column>p_itemtype</column>
                    </missingDensity>}
              </statInfo>
              <statInfo>
                    <objName>partsupp</objName>
                    <missingHistogram>
                            <column>ps_partkey</column>
                            <column>ps_itemtype</column>
                    </missingHistogram>
                    <missingDensity>
                            <column>ps_partkey</column>
                            <column>ps_itemtype</column>
                    </missingDensity>
              </statInfo>
         </optimizerStatistics>
```

早期のタイムアウト検出と tablecount

15.0 クエリ・プロセッサは、タイムアウト・メカニズムを自動的にアクティブにして、検索エンジンでかかる時間を減らすことができます。また、より積極的なタイムアウトを設定して、クエリ・プロセッサが使用するプロシージャ・キャッシュの量を減らすこともできます。有効な値は 0 から 1000 までの範囲です。

Adaptive Server をタイムアウト時間なしに設定することはできません。

タイムアウトは、多数のテーブル (4 個以上) を処理するクエリを使用する場合に一般的です。タイムアウトのデメリットは、オプティマイザを早期に終了するため、最適なプランを失う可能性があることです。しかし、クエリの最適化が実行時間を越える場合に、クエリの最適化にかかる時間を減らせるというメリットがあります。たとえば、64 ビットの 11.9.3 Adaptive Server を使用するテストで、tablecount が 12 に設定された 12 通りのジョインの最適化には 10 分かかりますが、実行時間は 30 秒だけです。オプティマイザは 1 分以内に最適化プランを見つけますが、大半の時間を 12 の階乗 (479,000,000) 個の組み合わせの検索に費やすため、事態はさらに深刻です。

タイムアウトの発生を検出するには、set option show on パラメータを使用します。タイムアウトが発生すると、次のメッセージが診断出力に表示されます。

```
!! Optimizer has timed out in this opt block !!
```

サーバまたはセッションのレベルでタイムアウト時間の値を大きくすると他のクエリに悪影響を及ぼすことがあり、プロシージャ・キャッシュなどのリソースを多く使用したり、コンパイル時間が増加する可能性があります。一般には、問題を引き起こしているストアド・プロシージャまたはクエリが特定されている場合は、高い値の最適化タイムアウトを指定してその再コンパイルを実行します。

サーバ・レベルでは、次の構文でタイムアウト時間を設定します。

```
sp_configure "optimization timeout limit", time_out_period
```

セッション・レベルでは、次の構文でタイムアウト時間を設定します。

```
set plan opttimeoutlimit time_out_period
```

クエリ・レベルでは、次のように入力してタイムアウトを強制します。

```
select * from table_name plan "(use opttimeoutlimit
time_out_period)"
```

抽象クエリ・プランによるクエリの修正

抽象クエリ・プランを使用すると、クエリ・コードを変更することなく特定のプランを使用するようにクエリを修正できます。抽象クエリ・プランを使用する以下の手順は、アプリケーションに対して透過的に実行されます。

1　15.0 サーバ内で問題のあるクエリを識別します。

2　抽象クエリ・プランを有効にして、以前のバージョンのサーバのクエリ・プランを取得し、15.0 サーバで問題を生じるクエリを実行します。

3　ap_stdout グループから抽象クエリ・プランを抽出します。

4　抽象クエリを確認および修正して、クエリのパフォーマンスに影響している不完全なプランの使用や他の問題を調整します。

5 15.0 サーバで、create plan を使用して、15.0 サーバ内のデフォルトの ap_stdin グループにクエリ・プランをロードします。

6 15.0 サーバ上で abstract plan load を有効にします。

7 15.0 サーバ上で抽象プラン・キャッシュを無効にして、最適化中の IO を回避します。

8 15.0 サーバでクエリを再実行し、抽象クエリ・プランが使用されるかどうかを判断します。

9 Adaptive Server にクエリのパフォーマンスを向上する新機能が導入されるごとに、抽象クエリ・プランを調整する必要がある可能性があります。

10 Sybase サポート・センタと協力して、元の最適化の問題を解決します。

抽象クエリ・プランの詳細については、『パフォーマンス＆チューニング・ガイド：オプティマイザと抽象プラン』を参照してください。

これらの手順は、クエリの問題を診断し、クエリを再作成し、発生する問題を解決するという一連の処理の枠を越えて、アップグレードが遅れるという問題を解決するのに役立ちます。抽象プランを使用すると、forceplan が実装するジョイン順の処理より制御性に優れた set forceplan on に代わることができます。以下の例でこれを示します。

インデックスの強制

次の例では、インデックスを使用せずに lineitem テーブルをスキャンします。

```
select count(*) from orders, lineitem where o_orderkey = l_orderkey
QUERY PLAN FOR STATEMENT 1 (at line 1).
3 operator(s) under root
The type of query is SELECT.

ROOT:EMIT Operator

|NESTED LOOP JOIN Operator (Join Type: Inner Join)
|
|  |SCAN Operator
|  |  FROM TABLE
|  |  orders
|  |  Table Scan.
|  |  Forward Scan.
|  |  Positioning at start of table.
|  |SCAN Operator
|  |  FROM TABLE
|  |  lineitem
|  |  Table Scan.
|  |  Forward Scan.
|  |  Positioning at start of table.
```

この方法では最適なクエリ・プランが作成されない可能性があり、lineitem に対して l_idx1 インデックスを使用した方が高速に実行される可能性があります。インデックスを使用するようにクエリを書き直します。

```
select count(*) from orders, lineitem (index l_idx1) where o_orderkey = l_orderkey
QUERY PLAN FOR STATEMENT 1 (at line 1).
3 operator(s) under root
The type of query is SELECT.

ROOT:EMIT Operator

|NESTED LOOP JOIN Operator (Join Type: Inner Join)
|
|  |SCAN Operator
|  |  FROM TABLE
|  |  orders
|  |  Table Scan.
|  |  Forward Scan.
|  |  Positioning at start of table.
|
|  |SCAN Operator
|  |  FROM TABLE
|  |  lineitems
|  |  Index : l_idx1
|  |  Forward Scan.
|  |  Positioning by key.
|  |  Keys are:
|  |  l_orderkey ASC
```

インデックスと抽象プランの強制

force パラメータを使用することでほとんどのクエリ・プランの問題は解決できますが、アプリケーション・コードの変更が必要です。コードの変更に問題がない場合でも、抽象クエリ・プランを使用するよりは時間がかかります。

次の例では上記と同じクエリを実行していますが、抽象プランを使用することで改善されています。

最初に、抽象プランを有効にします。

```
set option show_abstract_plan on
go
dbcc traceon(3604)
go
```

Adaptive Server によって抽象クエリ・プランが生成されます。これを編集し、インデックスの使用を強制します。

```
select count(*) from orders, lineitem where o_orderkey = l_orderkey
go
The Abstract Plan (AP) of the final query execution plan:
( nl_join ( t_scan orders ) ( t_scan lineitem ) ) ( prop orders ( parallel 1 ) (
prefetch 2 ) (lru ) ) ( prop lineitem ( parallel 1 ) ( prefetch 2 ) ( lru ) )
```

抽象プランを修正してクエリ・プロセッサの動作を変更し、PLAN 句を使用してクエリ・プロセッサにクエリ・プランを渡すことができます。構文は次のとおりです。

```
SELECT/INSERT/DELETE/UPDATE ...PLAN '( ... )
```

プランを改善するには、テーブル・スキャン (t_scan) をインデックス・アクセスに置換し、(prop ***table_name***) を使用してテーブルを指定します (例は見やすいようにインデントされています)。

```
select count(*) from orders, lineitem where o_orderkey = l_orderkey
plan
"( nl_join
            ( t_scan orders )
            ( t_scan lineitem )
          )
     ( prop orders ( parallel 1 ) ( prefetch 2 ) (lru ) )
     ( prop lineitem ( parallel 1 ) ( prefetch 2 ) ( lru ) )
```

インデックス・スキャンを強制的に実行するには、(i_scan *index_name table_name*) パラメータを使用します。

```
1> select count(*) from orders, lineitem where o_orderkey = l_orderkey
   plan
   "( nl_join
         ( t_scan orders )
         ( i_scan l_idx1 lineitem )
     )
     ( prop orders ( parallel 1 ) ( prefetch 2 ) (lru ) )
     ( prop lineitem ( parallel 1 ) ( prefetch 2 ) ( lru ) )
```

ジョイン順の強制

以前のリリースでは、set forceplan on を使用してジョイン順を設定していました。

次の例では、lineitem テーブルが orders テーブルの外部ジョインになるようにジョイン順を強制的に設定しており、そのジョイン順はクエリの from 句の中で切り替わります。

```
set forceplan on
2> go
1> select count(*) from lineitem, orders where o_orderkey = l_orderkey
2> go
```

抽象プランでジョイン順を強制的に設定することもできます。抽象プランを有効にしてから、次のようなクエリを指定します。

```
select count(*) from orders, lineitem where o_orderkey = l_orderkey
   plan
   "( nl_join
         ( t_scan orders )
         ( t_scan lineitem )
    )
    ( prop orders ( parallel 1 ) ( prefetch 2 ) (lru ) )
    ( prop lineitem ( parallel 1 ) ( prefetch 2 ) ( lru ) )
```

ジョイン順を切り替えることによって、抽象プランでジョイン順を強制的に設定できます。

```
select count(*) from orders, lineitem where o_orderkey = l_orderkey
   plan
   "( nl_join
         ( t_scan lineitem)
         ( t_scan orders )
     )
     ( prop orders ( parallel 1 ) ( prefetch 2 ) (lru ) )
     ( prop lineitem ( parallel 1 ) ( prefetch 2 ) ( lru ) )
```

異なるサブクエリ付加の強制

フラット化できない相関サブクエリに限って、サブクエリ付加を変更できます。サブクエリ付加を変更すると、サブクエリが評価される回数が減少します。

たとえば、以下の 3 テーブルのジョインの場合、骨組みのプラン出力では、3 つの外部テーブルのジョインが実行されてからサブクエリが付加されることが示されるだけです。showplan の出力にこれが表されます。

```
1> select count(*)
2> from lineitem, part PO, customer
3> where l_partkey = p_partkey and l_custkey = c_custkey
4> and p_cost = (select min(PI.p_cost) from part PI where PO.p_partkey = PI.p_partkey)
5> go

The Abstract Plan (AP) of the final query execution plan:
( scalar_agg ( nested ( m_join ( sort ( m_join ( sort ( t_scan customer ) ) ( sort (t_scan
lineitem ) ) ) ) ( i_scan part_indx (table (PO part ) ) ) ( subq ( scalar_agg ( t_scan
(table (PI part ) ) ) ) ) ) ) ) (prop customer ( parallel 1 ) ( prefetch 2 ) ( lru ) ) (
prop (table (PO part)) ( parallel 1 ) (prefetch 2 ) ( lru ) ) (prop lineitem ( parallel
1 ) ( prefetch 2 ) ( lru ) ) ( prop (table(PI part)) ( parallel 1 ) (prefetch 2 ) ( lru ) )
```

抽象プランを修正してクエリ・プロセッサの動作を変更し、PLAN 句を使用してクエリ・プロセッサにクエリ・プランを渡すことができます。構文は次のとおりです。

```
SELECT/INSERT/DELETE/UPDATE ...PLAN '( ... )
```

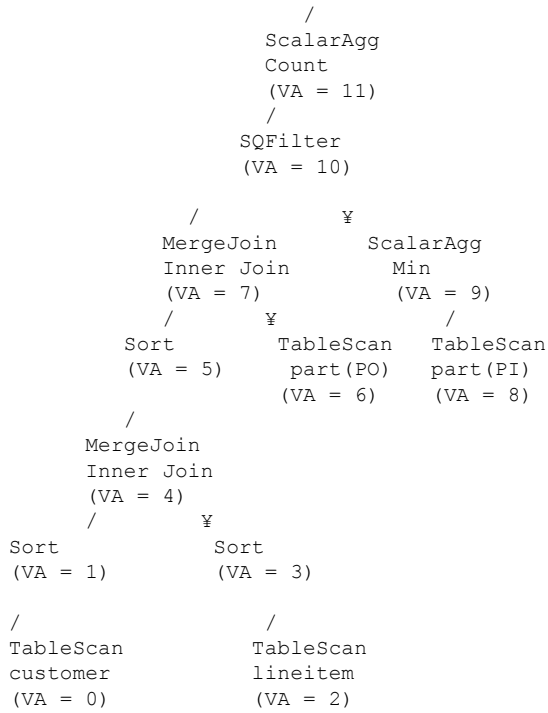
新しいプランは次のようになります。

```
QUERY PLAN FOR STATEMENT 1 (at line 1).

12 operator(s) under root

The type of query is SELECT.

ROOT:EMIT Operator

|SCALAR AGGREGATE Operator
| Evaluate Ungrouped COUNT AGGREGATE.
```

```
|
|   |SQFILTER Operator has 2 children.
|   |   |MERGE JOIN Operator (Join Type: Inner Join)
|   |   |   |
|   |   |   |   |SORT Operator
|   |   |   |   | Using Worktable4 for internal storage.
|   |   |   |   |
|   |   |   |   |   |MERGE JOIN Operator (Join Type: Inner Join)
|   |   |   |   |   |
|   |   |   |   |   |   |SORT Operator
|   |   |   |   |   |   | Using Worktable1 for internal storage.
|   |   |   |   |   |   |
|   |   |   |   |   |   |   |SCAN Operator
|   |   |   |   |   |   |   |   FROM TABLE
|   |   |   |   |   |   |   |   customer
|   |   |   |   |   |   |   |   Table Scan.
|   |   |   |   |   |   |   SORT Operator
|   |   |   |   |   |   | Using Worktable2 for internal storage.
|   |   |   |   |   |   |
|   |   |   |   |   |   |   |SCAN Operator
|   |   |   |   |   |   |   |   FROM TABLE
|   |   |   |   |   |   |   |   lineitem
|   |   |   |   |   |   |   |   Table Scan.
|   |   |   |   |SCAN Operator
|   |   |   |   |  FROM TABLE
|   |   |   |   |  part
|   |   |
|   |   |   Run subquery 1 (at nesting level 1).
|   |   |
|   |   |   QUERY PLAN FOR SUBQUERY 1 (at nesting level 1 and at line 4).
|   |   |
|   |   |   Correlated Subquery.
|   |   |   Subquery under an EXPRESSION predicate.
|   |   |
|   |   |   |SCALAR AGGREGATE Operator
|   |   |   |   Evaluate Ungrouped MINIMUM AGGREGATE.
| | | |
|   |   |   |   |SCAN Operator
|   |   |   |   |   FROM TABLE
|   |   |   |   |   part
|   |   |
|   |   |   END OF QUERY PLAN FOR SUBQUERY 1.
```

トレース・フラグ 526 または set statistics plancost を使用して、演算子ツリーを表示できます。トレース・フラグ 526 では、コストなしで演算子ツリーが表示されます (次の例を参照)。しかし、使用する抽象プランが効率的であるかどうかを判断する上でコストを知ることは有用であるため、set statistics plancost on の使用をおすすめします。次の例では、PI の集計の位置に注意してください。

```
==================== Lava Operator Tree ====================
                      Emit
                      (VA = 12)
```



このクエリ・プランは、最適でない可能性があります。サブクエリは外部テーブル part (PO) に依存しており、このテーブルがスキャンされた後で任意の場所に付加できます。

この例では、最も外側のテーブルが part (PO) で、それに lineitem が続き、最も内側のテーブルは customer という順序が正しいジョイン順であると想定しています。テーブル PO のスキャンにサブクエリを付加する必要がある場合は、前の例で作成された抽象プランから開始して、必要に応じてクエリを修正してください。

```
select count(*)
from lineitem, part PO, customer
where l_partkey = p_partkey and l_custkey = c_custkey
and p_cost = (select min(PI.p_cost) from part PI where PO.p_partkey = PI.p_partkey)
plan
"(scalar_agg
        (m_join
          (sort
             (m_join
                   (nested
                      (scan (table (PO part)))
                      (subq (scalar_agg (scan (table (PI part)))))
                    )
```

```
                    (sort
                       (scan lineitem)
                    )
                )
            )
            (sort
               (scan customer)
            )
        )
    )
(prop customer ( parallel 1 ) ( prefetch 2 ) ( lru ) )
(prop (table (PO part)) ( parallel 1 ) (prefetch 2 ) ( lru ) )
(prop lineitem ( parallel 1 ) ( prefetch 2 ) ( lru ) )
(prop (table(PI part)) ( parallel 1 ) (prefetch 2 ) ( lru ) )
go
==================== Lava Operator Tree ====================

                            Emit
                            (VA = 12)
                            /
                            ScalarAgg
                            Count
                            (VA = 11)
                           /
                         MergeJoin
                         Inner Join
                         (VA = 10)
                         /        ¥
                      Sort        Sort
                      (VA = 7)    (VA = 9)

                     /                 /
                MergeJoin             TableScan
                Inner Join            customer
                (VA = 6)              (VA = 8)
                /         ¥
            SQFilter       Sort
            (VA = 3)       (VA = 5)


            /      ¥           /
        TableScan   ScalarAgg  TableScan
        part(PO)    Min        lineitem
        (VA = 0)    (VA = 2)   (VA = 4)
                      /
                   TableScan
                   part (PI)
                   (VA = 1)

============================================================
The type of query is SELECT.
```

```
ROOT:EMIT Operator

|SCALAR AGGREGATE Operator
| Evaluate Ungrouped COUNT AGGREGATE.

|  |MERGE JOIN Operator (Join Type: Inner Join)
|  |  Using Worktable5 for internal storage.
|  |  Key Count: 1
|  |  Key Ordering: ASC
|  |
|  |  |SORT Operator
|  |  |  Using Worktable3 for internal storage.
|  |  |
|  |  |  |MERGE JOIN Operator (Join Type: Inner Join)
|  |  |  |  Using Worktable2 for internal storage.
|  |  |  |  Key Count: 1
|  |  |  |  Key Ordering: ASC
|  |  |  |
|  |  |  |  |SQFILTER Operator has 2 children.
|  |  |  |  |
|  |  |  |  |  |SCAN Operator
|  |  |  |  |  |  FROM TABLE
|  |  |  |  |  |  part
|  |  |  |  |  |  PO
|  |  |  |  |  |  Table Scan.
|  |  |  |  |  |  Forward Scan.
|  |  |  |  |  |
|  |  |  |  | Run subquery 1 (at nesting level 1).
|  |  |  |  |
|  |  |  |  | QUERY PLAN FOR SUBQUERY 1 (at nesting level 1 and at line 4).
|  |  |  |  |
|  |  |  |  | Correlated Subquery.
|  |  |  |  | Subquery under an EXPRESSION predicate.
|  |  |  |  |


|  |  |  |  |  |SCALAR AGGREGATE Operator
|  |  |  |  |  |  Evaluate Ungrouped MINIMUM AGGREGATE.
|  |  |  |  |  |
|  |  |  |  |  |SCAN Operator
|  |  |  |  |  |  FROM TABLE
|  |  |  |  |  |  part
|  |  |  |  |  |  PI
|  |  |  |  |  |  Table Scan.
|  |  |  |  |  |  Forward Scan.
|  |  |  |  |  END OF QUERY PLAN FOR SUBQUERY 1.
|  |  |  |  |
|  |  |  |  |SORT Operator
|  |  |  |  |  Using Worktable1 for internal storage.
|  |  |  |  |
```

```
|   |   |   |   |  |SCAN Operator
|   |   |   |   |  |  FROM TABLE
|   |   |   |   |  |  lineitem
|   |   |   |   |  |  Table Scan.
|   |   |   |   |  |  Forward Scan.
|   |   |  |SORT Operator
|   |   |   |  Using Worktable4 for internal storage.
|   |   |   |
|   |   |   |  |SCAN Operator
|   |   |   |   |  FROM TABLE
|   |   |   |   |  customer
|   |   |   |   |  Table Scan.
|   |   |   |   |  Forward Scan.
```

この例では、前の例のサブクエリ内の位置と比較すると、ScalarAgg と TableScan がシフトされています。

クエリ・プロセッサとオプティマイザに関する問題のレポート

クエリ処理の問題にはさまざまな種類がありますが、通常、エラー・ログにスタック・トレースが生成されるか (クライアントの切断を伴う場合があります)、パフォーマンスが低下します (原因は不明か、正しいクエリ・プランを使用できない場合のいずれかです)。問題が発生した場合は、問題のクエリを切り離す必要があります。この後、以下に示す出力を収集して、Sybase サポート・センタに連絡してください。Sybase はすべてのお客様と機密保持契約を交わしており、業務上の機密データのご提供について、ご懸念いただく必要はありません。

- 完全なデータベース・ダンプ – 推奨される出力。ただし、完全なデータベース・ダンプが入手できない場合は、関係するテーブルの完全なスキーマ、ストアド・プロシージャのソース・コード、bcp で抽出したデータを用意してください。この情報がなくても一部の問題は解決できますが、提供する情報は近似値から成り、解決を保証するものではありません。データ、正確なデータ・カーディナリティ、有効なデータ・ボリュームのコピーを持つことによって、問題の検出だけでなく解決策のテストも行うコスト計算アルゴリズム用のデータを選択するために、クエリ・プロセッサが使用する手段も持つことができます。
- ddlgen の出力 – 関係するすべてのテーブル、インデックス、トリガ、プロシージャの完全なスキーマを用意するようにしてください。
- optdiag の出力 (使用している場合はシミュレート・モードを含む) – Sybase は、データ・ボリューム、カーディナリティ、クエリ・プロセッサがデータを選択する方法について理解する必要があります。シミュレートされた統計情報を使用することでクエリ・プロセッサに影響を与えている場合は、それも含めます。

- 強制プラン情報 – 最適に実行するプランを強制的に使用して、以下に関する情報を収集します。適切に実行されないプランに関しても、同じ情報を収集してください。
  - set statistics plancost on。
  - set statistics time on。
  - set option show long。大量の出力を発生する可能性があります。場合によっては、これを実行するために、トレース・フラグ 3604 を有効にする必要があります。
  - set showplan on (トレース・フラグ 526 が有効)。

クエリの結果が正しくない場合、Sybase に提供する情報は、並列処理が有効または無効の状態で実行しているかどうかによって異なります。

- 並列処理が無効の場合 – データベース・ダンプまたは bcp で抽出したデータに関するものであれば、すべて非常に役立ちます。ただし、それに加えて、次のオプションを有効にしてクエリを実行した後の出力を収集してください。
  - set option show_code_gen on
  - dbcc traceon(201)
- 並列処理が有効な場合 – 並列クエリでのみ問題が発生し、サーバが逐次モードで実行される場合は問題が生じない場合は、上記の情報に加えて、次の情報も含めます。
  - set option show long
  - set option show_parallel long
  - set option show_elimination long (適切なパーティション排除を使用していない場合)

パフォーマンスのテスト

この項では、アップグレード前後のパフォーマンスのベンチマーク・テストの内容を、以下のように分類してさらに詳しく説明します。

- アップグレード前のシングルユーザ・テスト
- アップグレード前のマルチユーザ・テスト
- テスト・システムのアップグレード
- アップグレード後のシングルユーザ・テスト
- アップグレード後のマルチユーザ・テスト

注意 ベンチマークで、サーバの処理速度をテストします。そのため、負荷の軽いクライアントからベンチマーク・テストを実行して、すべての処理がバックエンドで行われるようにしてください。

アップグレード前のシングルユーザ・テスト

運用システムの Adaptive Server と同じレベルのテスト・システムを作成した後、新しいサーバとの比較を有効にするために、ベンチマーク・テストを実行する前にテスト・システムを運用システムと同期させます。

オプティマイザ

12.x Adaptive Server における 300 シリーズのトレース・フラグに基づくインデックスの選択性と IO 射影を、15.x Adaptive Server における show オプション出力と比較します。オプティマイザの決定が異なる場合、考えられる原因の一部は次のとおりです。

- データとデータ配分の問題。データ・レイアウトをチェックして、これらが運用システムと同じであることを確認します。
- インデックスの定義。インデックス再作成スクリプトが機能したかどうかを確認します。sp_helpindex を使用して、運用システムとテスト・システムのインデックスの定義を比較します。

すべての問題を解決してから、次に進んでください。

以前のリリースの Adaptive Server の抽象プランを取得した場合は、クエリ測定基準機能を使用して、新バージョンのプランとパフォーマンスを比較します。詳細については、http://sybooks.sybase.com/nav/detail.do?docset=900 にある『クエリ・プロセッサ』マニュアルの「クエリ処理 (QP) 測定基準」の章を参照してください。

I/O

テスト・サーバが、同じ量の I/O を実行していることを確認します。物理 I/O と論理 I/O は、運用システムと同じ割合であることが必要です。statistics io の出力を使用して比較します。

アップグレード前のマルチユーザ・テスト

オプティマイザが運用システムと同じように動作したら、次のテストを実行できます。

不定期なベンチマーク・テスト

マルチユーザ・モードでは、不定期なベンチマーク・テストを必要な回数だけ実行して、応答時間とスループットの測定基準を取得します。これで、次の点が解決されます。

- 運用システムとのキャッシュ・ヒット率などの違い
- 飽和状態のデバイスが原因で発生するボトルネック
- その他の問題または誤り

1 つ完了したら、次の実行の前にバックアップからデータベースをリストアするか、クイック・リセットを実行してください。

定期的なベンチマーク・テスト

定期的なベンチマーク・テストを実行するときは、バックアップからデータベースを必ず既知の状態にリストアします。応答時間とスループットの測定基準を収集します。必要に応じた頻度でテストを実行して問題を修正し、最終結果を再生成します。

テスト・システムのアップグレード

インストール・ガイドの指示に従ってテスト・システムでのアップグレードを実行します。テスト・アップグレードが実際のアップグレードのリハーサルとなるように、すべての手順を実行します。「第 5 章 データベース管理の変更」で必要であると判断したメモリとディスク容量を変更します。ただし、ここでは新しい機能を使用するために Adaptive Server の設定を変更しないでください。

注意　アップグレード後の最初の目的は、Adaptive Server の既定のパフォーマンスをテストすることです。新しいパフォーマンス調整機能は後で使用します。

すべての問題を解決し、参考のために記録します。インストール・ガイドに、詳細なトラブルシューティング情報が記載されています。

アップグレード後、次の操作を行います。

- すべてのデータベース上で dbcc を実行する。
- master や sybsystemprocs など、すべてのデータベースをバックアップする。定期的なベンチマーク・テスト間にテスト・システムを既知の状態にリストアするために、バックアップが必要になります。

アップグレード後のシングルユーザ・テスト

シングルユーザ・テストでは、次の内容を確認します。

- オプティマイザが、同等または改良されたクエリ・プランを作成している。実行していた Adaptive Server のバージョンによっては、オプティマイザが変更されているためにクエリ・プランが異なる場合があります。アプリケーション・コードを変更してその違いを補正する必要があるかどうかを判断してください。発生する可能性のあるオプティマイザの問題については、「第 6 章 安定性とパフォーマンスの確認」を参照してください。
- 物理 I/O と論理 I/O のカウントが、アップグレード前と同じである。

アップグレード後のマルチユーザ・テスト

バックアップを再ロードして、テスト・システムを既知の状態に戻してから、ベンチマーク・テストを再実行します。

不定期なベンチマーク・テスト

マルチユーザ・モードでは、不定期なベンチマーク・テストを必要な回数だけ実行して、応答時間とスループットの測定基準を取得します。これで、次の点が解決されます。

- アップグレード前の状態とのキャッシュ・ヒット率などの違い
- ボトルネック
- その他の問題または誤り

アップグレード後に、十分なデフォルトのデータ・キャッシュがあることを確認してください。必要に応じて調整し、問題を修正します。1 つ完了したら、次の実行の前にバックアップからデータベースをリストアするか、クイック・リセットを実行してください。

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』を参照してください。パフォーマンスに影響する可能性のある問題については、最新の TechNotes のテクニカル・インフォメーション・ライブラリと最新のホワイトペーパーを参照してください。

定期的なベンチマーク・テスト

定期的なベンチマーク・テストを実行するときは、バックアップからデータベースを必ず既知の状態にリストアします。応答時間とスループットの測定基準を収集します。

パフォーマンス基準を満たすように、必要に応じて調整します。

必要に応じた頻度でテストを実行して問題を修正し、最終結果を再生成します。

付 録　A

現在の環境のワークシート

この付録では、マイグレーションの計画に役立つ情報を収集するためのガイドラインとサンプル・ワークシートを示します。

マイグレーションの計画の一部として、ASE Migration Resources Web ページ (http://sybase.com/support/techdocs/migration) にある最新の資料を参照してください。SySAM の詳細については、『User's Guide Sybase Software Asset Management』と http://www.sybase.com/sysam を参照してください。

Adaptive Server 運用ワークシート

ビジネス要件収集用のワークシートを次に示します。詳細については、「第 1 章 ビジネス要件の文書化」を参照してください。

このワークシートの構成は、次のとおりです。

- 運用に関するビジネス要件
- バックアップ手順とリストア手順
- データベース・ダンプの詳細
- 管理手順の詳細

運用に関するビジネス要件

これは、運用に関するビジネス要件を記録するワークシートです。

データベース名	運用時間	最大ダウン時間	コメント

バックアップ手順とリストア手順

このワークシートは、バックアップ手順とリストア手順の一般的な調査に役立ちます。

タスク	はい／いいえ	コメント
バックアップ手順とリカバリ手順は文書化されているか。		
自動化されたダンプ手順は適切に機能しているか。		
ダンプの生成数。		
ダンプは別の場所に保存されているか。		
管理作業は文書化されているか。		
Adaptive Server のエラー・ログ・スキャンの実行頻度。		
データベース領域の使用率はモニタされているか。		
データベースの容量計画は最近実行されているか。		

データベース・ダンプの詳細

次のバックアップとリストアの詳細情報は、成功のための基準を定義する上で役立ちます。

データベース名	データベース・ダンプの頻度	使用するダンプ・デバイス	トランザクション・ログ・ダンプの頻度	使用するダンプ・デバイス	コメント

管理手順の詳細

次のワークシートを使用して、詳細な管理情報を記録します。

データベース／オブジェクト名	*dbcc checkdb/checktable* の頻度	*dbcc checkalloc/tablealloc* の頻度	*update statistics* の頻度	領域使用率をモニタする頻度

データ・アーキテクチャ・ワークシート

情報の流れを文書化する方法については、「第 1 章 ビジネス要件の文書化」で説明しています。データ・アーキテクチャ・ワークシートの構成は、次のとおりです。

- クライアント・アプリケーション・コンポーネント
- 運用パフォーマンス測定基準
- トランザクション・プロファイル

クライアント・アプリケーション・コンポーネント

次のワークシートを使用して、アプリケーション・プロファイルとパフォーマンス要件を記録します。

アプリケーション名	アプリケーションの種類	アプリケーションの記述に使用するツール	クライアントの実行場所	同時ユーザ数	アクセス対象のデータベース	最大ダウン時間 (1 日あたり)	優先度 1 の平均応答

運用パフォーマンス測定基準

表 A-1 は、オペレーティング・システムのモニタを使用して測定する、現在の運用パフォーマンスの測定基準です。

表 A-1: 運用システムの測定基準

測定対象	測定する内容
CPU	各サーバの「時間枠」(オンライン時やバッチ処理時など) あたりの CPU の平均使用率と最大使用率 (SMP サーバの全 CPU と CPU ごと)
ディスク I/O	• ディスク別とコントローラ別の 1 秒あたりの I/O の回数と、サーバごとの「時間枠」あたりの I/O キューの長さ • 各サーバの Sybase デバイス別に「時間枠」あたりの毎秒の I/O、読み込み、書き込みの各総数
同時実行性	平均ロック競合を調べます。sp_who を使用すると、現在実行しているプロセスを特定でき、sp_lock を使用すると、ロックを保持している現在のプロセスを調べることができます。
ネットワーク I/O	• 各サーバの NIC 別の「時間枠」あたりの毎秒のパケット数 • サーバごとの「時間枠」あたりの TDS パケット数 (「送信」パケットと「受信」パケット)
メモリ	• サーバごとに「時間枠」あたりの毎秒のページング／スワッピング速度を調べます。 • また、サーバごとに「時間枠」あたりのデータとプロシージャ・キャッシュのヒット率を調べます。

トランザクション・プロファイル

サーバ上の各データベースのトランザクション・プロファイル情報を記録します。

プロセス名 (正確な名前)	プロセスの種類 (OLTP、バッチ)	正確な優先度	ユーザあたりの頻度 (1 時間あたり)	ソース・コードの種類 (ストアド・プロシージャ、ESQL など)	平均応答時間の要件	最大応答時間の要件	現在の平均応答時間と最大応答時間

プロセス名 (正確な名前)	プロセスの種類 (OLTP、バッチ)	正確な優先度	ユーザあたりの頻度 (1 時間あたり)	ソース・コードの種類 (ストアド・プロシージャ、ESQL など)	平均応答時間の要件	最大応答時間の要件	現在の平均応答時間と最大応答時間

注意 応答時間の問題を迅速に特定するには、すべての重要なトランザクションの showplan 出力を保存します。

Adaptive Server インフラストラクチャ・ワークシート

使用している環境を文書化する方法については、「第 3 章 マイグレーション・プランの作成」で説明しています。インフラストラクチャ・ワークシートの構成は、次のとおりです。

- ホスト設定
- Adaptive Server の設定
- データベース・デバイス
- データベースとセグメント
- ダンプ・デバイス

運用環境と開発環境の両方の情報を記録します。

ホスト設定

この項では、ホスト設定のワークシートを示します。

ハードウェア

ハードウェア製造元のサポート・センタ情報を記録します。

Adaptive Server マシン
製造元：
機種：
ハードウェア・ベンダに登録されている顧客 ID：
サポート・センタの電話番号：
サポート・センタの受付時間：
テクニカル・アカウント・マネージャ： • 名前： • 電話番号： • ポケットベルの番号：

CPU リソースを記録します。

CPU
物理プロセッサ数：
チップ速度：
Adaptive Server が使用できるプロセッサ数：
その他の CPU 集中利用プロセス／スレッド：
特定の CPU にバインドされたプロセス／スレッド：

CPU
実行優先度が高いプロセス／スレッド：

物理メモリの使用率

各サーバ上で実行する主なプロセスとメモリのすべての要件をリストします。

アプリケーション	実行時メモリの使用率
オペレーティング・システム	
Adaptive Server のメモリ合計 + 追加のネットメモリ + エクステント I/O バッファ	
Open Server 次のコンポーネントで構成される。 • Backup Server • sybmultbuf • Replication Server • Monitor Server • ゲートウェイ (リストする)	
その他のアプリケーション	
必要メモリの合計	
取り付けられているメモリの合計	

ディスク I/O の設定

一般のディスク情報は、互換性がない場合や容量計画に役立ちます。

コントローラの番号	製造元と機種	ハードウェアのリビジョン	サービス期間 (月)	転送率 (KB/ 秒)

次のディスク・レイアウト情報は、ハードウェアの互換性がない場合や平均故障間隔 (MTBF) が限界に近づいている場合、およびロード・バランスの設定や容量計画に役立ちます。

物理デバイス名	製造元と機種	ハードウェアのリビジョン	サービス期間 (月)	コントローラの番号	容量 (MB)	スループット (1 秒あたりの I/O)	転送率 (KB/ 秒)

次のディスク・レイアウト情報は、再分配が必要な場合や、ロード・バランスの設定、容量計画に役立ちます。

物理デバイス名	バーティション番号	使用箇所 (Sybase、UNIX ファイル・システムなど)	デバイス名	OS ミラー・デバイス名	容量 (MB)	シリンダの範囲

次の論理ボリューム情報は、再分配が必要な場合や、ロード・バランスの設定、容量計画に役立ちます。

論理ボリューム・デバイス	メンバ・ディスク・パーティション	使用箇所 (Sybase、UNIX ファイル・システムなど)	Sybase デバイス	ミラー論理デバイス	容量 (MB)	ストライプ幅 (MB)

ネットワークの設定

ネットワーク・レイアウト情報は、ファームウェアの互換性がない場合や、MTBF、容量計画に役立ちます。

物理デバイス名	製造元と機種	ファームウェアのリビジョン	サービス期間 (月)	サポートするプロトコル	ネットワーク・アドレス	転送率 (KB/ 秒)

テープの設定

テープ・レイアウト情報は、ファームウェアの互換性がない場合や、MTBF、容量計画に役立ちます。

物理デバイス名	製造元と機種	ファームウェアのリビジョン	サービス期間 (月)	容量 (MB)	コントローラの番号	転送率 (KB/ 秒)

オペレーティング・システムの設定

オペレーティング・システムの詳細について記録します。

サーバ・ハードウェア
名前：
リリース・レベル：

サーバ・ハードウェア
パッチの履歴：
カーネル設定パラメータ：
スワップ領域サイズ：
サポート・センタの電話番号：
サポート・センタの受付時間：
テクニカル・アカウント・マネージャ： • 名前： • 電話番号： • ポケットベルの番号：

Adaptive Server の設定

Adaptive Server の設定に関する一般情報を記録します。

一般設定
ホーム・ディレクトリ：
ページ・サイズ：
コンポーネント、リリース・レベル、フィックス・レベル：
データベース環境を再構築するためのスクリプトのロケーションと名前：

一般設定
sp_configure 設定値：
buildmaster 設定値：
License_key 情報

オフピーク時またはシングルユーザ・モードのときに Adaptive Server で dbcc memusage を実行します。

使用法	MB	ページ・サイズ　バイト
設定されているメモリ		
コードのサイズ		
カーネル構造体		
サーバ構造体		
ページ・キャッシュ		
プロシージャ・バッファ		
プロシージャ・ヘッダ		
ページ・バッファの数		
プロシージャ・バッファの数		

データベース・デバイス

データベース・デバイス情報は、再分配が必要な場合や、ロード・バランスの設定、容量計画に役立ちます。

データベース・デバイス名	物理デバイス名	仮想デバイス番号	容量 (ページ)	ページ・サイズ	ミラー・デバイス名

データベースとセグメント

データベースとセグメントの情報は、ロード・バランスの設定や容量計画に役立ちます。

データベース名	データベース・オプション・セット	フラグメント (ページ範囲)	サイズ (MB)	セグメント名	デバイス名

データベース名	データベース・オプション・セット	フラグメント (ページ範囲)	サイズ (MB)	セグメント名	デバイス名

ダンプ・デバイス

ダンプ・デバイス情報は、ロード・バランスの設定や容量計画に役立ちます。

データベース・デバイス名	物理デバイス名	メディアの種類	容量 (MB)

付 録　B

サンプル・マイグレーション作業リスト

この付録では、次のサンプルを提供します。

注意　これらの例は、説明を目的としています。これは、すべてのサイトにそのまま適用できる詳細手順ではありません。マイグレーションはすべてユニークです。使用しているサイトに適した固有のプランを作成してください。

サンプル作業リスト・テンプレート

次のテーブルは、マイグレーションの一部として実行する必要のある作業を詳しく記録するための推奨フォーマットです。必要に応じて変更してください。

作業	日付	開始時刻	終了時刻	持続時間	所有者	ステータス

一般的なマイグレーション作業リストの例

次の一般的な作業リストは、典型的なマイグレーション作業を示します。各自のマイグレーション作業リストは、詳細と順序が異なる場合があります。次の一般的な段階が含まれます。

- マイグレーションの分析
- マイグレーションの準備
- マイグレーションの実装 (インストール／ロード方法の使用)
- マイグレーションの実装 (アップグレード方法の使用)
- マイグレーションの品質保証

マイグレーションの計画の詳細については、「第 3 章 マイグレーション・プランの作成」を参照してください。

マイグレーションの分析

現在の設定の文書化

1 環境のカットオフ・ポイントを確立します。
2 現在のサーバ・インストール情報を文書化します。
3 現在のサーバ設定値を文書化します。
4 ハードウェア設定を文書化します。
5 サーバごとのアプリケーションを文書化します。
6 アプリケーションのサーバ要件を文書化します。
7 アプリケーションのクライアント要件を文書化します。
8 関連するソフトウェアとミドルウェアを文書化します。
9 ソース・データベース作成スクリプトを検索します。
10 ソース・オブジェクト作成スクリプトを検索します。
11 すべてのサーバとデータベース・オブジェクトのカウントを取得します。
12 設定の文書を確認します。
13 設定の文書を更新します。

ビジネス要件の収集

1　ビジネス要件を定義します。
2　制約を定義します。
3　アプリケーションの依存性を定義します。
4　データ・サーバの依存性を定義します。
5　アプリケーションの優先順位を決定します。
6　ベンダの問題を特定します。
7　要件の文書を確認します。
8　要件の文書を更新します。

互換性分析の実行

1　ハードウェアの互換性を分析します。
2　オペレーティング・システムの互換性を分析します。
3　その他の Sybase ソフトウェアの互換性を分析します。
4　Sybase 以外のソフトウェアの互換性を分析します。
5　ミドルウェア／ API の互換性を分析します。
6　通信の互換性を分析します。
7　クライアント・プラットフォームの互換性を分析します。
8　分析結果を文書化します。
9　互換性の分析を確認します。
10　互換性の分析を更新します。

マイグレーション方法の策定

1　マイグレーション方法のドラフトを作成します。
2　方法を確認します (チーム)。
3　方法を更新します (チーム)。
4　方法を確認します (ユーザ／スポンサ)。
5　マイグレーションによるダウン時間の影響を定義します。
6　影響のある部署に通知します。
7　実装プランを見直します。
8　マイグレーションと実装について、ユーザの承認を得ます。

マイグレーションの準備

テスト・プランの作成とテスト・スクリプトの記述

1 システム機能テストを記述します。
2 統合テストを記述します。
3 ストレス・テストを記述します。
4 ユーザ受け入れテストを記述します。
5 テスト・プランを確認します。
6 テスト・プランを更新します。
7 テストの各段階のテスト・スクリプトを作成します。
8 ソース・システム上でスクリプトを実行して、ベースラインを確立します。
9 テスト・プランとベースラインの結果について、ユーザの承認を得ます。

アプリケーションのマイグレーション準備

1 新しい予約語を検索します。
2 新しいデータベース変換と計算を確認します。
3 変更された SQL 構文を含むクエリを検索します。
4 コードの変更を設計します。
5 アプリケーションの変更についてユーザの承認を得ます。

サーバ・マイグレーション・スクリプトの設計と開発

1 サーバ・マイグレーション・スクリプトを設計、開発します。
2 デバイスのファイル・システム設定を設計、開発します。
3 データベース・デバイス・スクリプトを作成します。
4 セキュリティ、ログイン、パスワードの修正を準備します。
5 セキュリティ・スクリプトを作成します。

データベース・マイグレーション・スクリプトの設計と開発

1 データベース・マイグレーション・スクリプトを作成します。
2 データベース・オブジェクト作成スクリプトを作成します。

3　データベース・セキュリティ・スクリプトを作成します。

4　システム管理スクリプトを修正または作成します。

データ・マイグレーション・スクリプトの設計と開発

1　データ抽出スクリプト (bcp など) を設計、開発します。

2　最適なバルク・コピー・オプションを特定します。

3　データ・ロード・スクリプトを設計、開発します。

マイグレーション前のその他の作業の実行

1　フォールバック作業を設計します。

2　フォールバック作業についてユーザの承認を得ます。

3　バックアップを実行します。

4　ソース・コード・コントロール環境を設定します。

5　新しいユーザ環境を設定します。

6　その他のマイグレーション補助プログラムを開発します。

マイグレーションの実装 (インストール／ロード方法の使用)

ターゲット環境の作成

1　ターゲット・システムの準備が完了していることを確認します。

2　ターゲット・システムにマイグレーション・スクリプトを移動します。

3　ファイル・システムを設定します。

4　インストール・ガイドのインストール／アップグレード準備を完了します。

5　Adaptive Server、Backup Server、Open Client をインストールします。

サーバ・マイグレーションの実行

1　システムとサーバへのアクセスを制限します。

2　データベース・デバイスを作成します。

3　サーバ・セキュリティ・スクリプトやその他のスクリプトを実行します。

データベース・マイグレーションの実行

1 データベース作成スクリプトを実行します。
2 データベースのパーティションとセグメントを作成します。
3 データベース・オブジェクト (デフォルト値、制約、ルール、ビューなど) を作成します。
4 データベース・テーブルとストアド・プロシージャを作成します。
5 データベース・セキュリティ・スクリプトを実行します。

データ・マイグレーションの実行

1 ソース・データ抽出スクリプトを実行します。
2 ソース・プラットフォームからターゲット・プラットフォームにデータを移動します。
3 データ・ロード・スクリプトを実行します。

サーバとデータのマイグレーションの完了

1 インデックスとトリガを作成します。
2 dbcc コマンドを実行します。
3 データベースをダンプします。

アプリケーション・マイグレーションの実行

1 アプリケーション・コードの変更を設計します。
2 アプリケーションの変更をそれぞれテストします。
3 アプリケーションの変更を分析して修正します。
4 新しいサーバにアクセスするようにアプリケーションを設定します。
5 事前のチューニングを行います。
6 ユーザにシステムとサーバへのアクセスを許可します。
7 旧バージョンのファイルにアクセスしないようにします。
8 使用可能になったことをユーザに通知します。

マイグレーションの実装 (アップグレード方法の使用)

Adaptive Server のアップグレード

1　ターゲット・システムの準備が完了していることを確認します。
2　ファイル・システムを設定します。
3　ソフトウェアをインストールします。
4　sqlupgrade を使用してアップグレードを実行します。

マイグレーションの完了

1　dbcc コマンドを実行します。
2　データベースをダンプします。

アプリケーション・マイグレーションの実行

1　アプリケーション・コードの変更を設計します。
2　アプリケーションの変更をそれぞれテストします。
3　アプリケーションの変更を分析して修正します。
4　新しいサーバにアクセスするようにアプリケーションを設定します。
5　事前のチューニングを行います。
6　ユーザにシステムとサーバへのアクセスを許可します。
7　旧バージョンのファイルにアクセスしないようにします。
8　使用可能になったことをユーザに通知します。

マイグレーションの品質保証

システム・テストの実行

1　機能テストを実行します。
2　機能テストの結果をベースラインと比較します。
3　機能テストの結果を分析します。
4　修正処理を行います。
5　必要に応じて再テストします。
6　機能テストの結果を文書化します。
7　機能テストの結果についてユーザの承認を得ます。

統合テストの実行

1 統合テストを実行します。
2 統合テストの結果をベースラインと比較します。
3 統合テストの結果を分析します。
4 修正処理を行います。
5 必要に応じて再テストします。
6 統合テストの結果を文書化します。
7 統合テストの結果についてユーザの承認を得ます。

ストレス・テストの実行

1 パフォーマンス／容量テストを実行します。
2 容量テストの結果をベースラインと比較します。
3 容量テストの結果を分析します。
4 修正処理を行います。
5 必要に応じて再テストします。
6 容量テストの結果を文書化します。
7 容量テストの結果についてユーザの承認を得ます。

ユーザ受け入れテストの実行

1 受け入れテストを実行します。
2 受け入れテストの結果をベースラインと比較します。
3 受け入れテストの結果を分析します。
4 修正処理を行います。
5 必要に応じて再テストします。
6 受け入れテストの結果を文書化します。
7 受け入れテストの結果についてユーザの承認を得ます。

運用データのリフレッシュの実行

1 運用データのカットオーバをスケジュールします。
2 運用データを抽出します。
3 ターゲットにデータを移動します。

4　Adaptive Server 運用環境にデータをロードします。

5　新しい環境をバックアップします。

6　データがリフレッシュされたことをユーザに通知します。

並列マイグレーション作業リストの例

このサンプル作業リストは、複写使用の並列マイグレーション方法による Adaptive Server Enterprise 15.0 へのアップグレードを示します。

この作業リストのシナリオは、次のとおりです。

- ある大規模な電気通信会社で、アップグレードのためにシステム・ダウン時間をとることができません。
- この会社では、障害発生時に、15 分以上システムをダウンさせることはできず、1 時間を超えるデータの消失は許されません。運用バックアウトが必要な場合、このような不測の事態に対応する要件をマイグレーションでサポートする必要があります。

この例では、前の例で説明した準備手順は省略します。この例の作業リストの構成は、次のとおりです。

- テスト／受け入れ基準の定義 － リグレッション・テスト・スイート
- ターゲット運用環境の設定
- Replication Server の設定
- リグレッション・テスト・スイートの実行
- サーバ B (シャドウ) の更新
- Adaptive Server 15.0 (サーバ B) でのアップグレード後のリグレッション・テスト・スイートの実行
- Adaptive Server 15.0 (サーバ B) でのユーザ受け入れテストの実行
- Adaptive Server 15.0 (サーバ B) への運用ユーザの移行
- 最終手順の実行

テスト／受け入れ基準の定義 – リグレッション・テスト・スイート

バックエンド・リグレッション・テスト・スイート – 運用負荷

1 バックエンド・クエリを特定します。

2 バンクエンド・クエリをカプセル化します。

3 バックエンド・テスト・スイート (showplan、stat io、stat time の各ラッパ) を作成します。

フロントエンド・シミュレーション・リグレッション・テスト・スイート

1 ターゲット・ユーザ関数を特定します。

2 SQL コードを取得し、ターゲット・ユーザ関数にマップします。

3 ユーザ関数の SQL コードをカプセル化します。

4 フロントエンド・シミュレーション・テスト・スイート (showplan、stat io、stat time の各ラッパ) を作成します。

フェーズ 1 と 2 のフロントエンド・リグレッション・テスト・スイート

1 フロントエンド・テスト・シナリオを特定します。

2 フロントエンド･アプリケーションと受け入れ／テスト手順を確認します。

3 機能テストの方法を文書化します。

4 フロントエンド混合マトリックスを構成します。

ターゲット運用環境の設定

1 サーバ A (運用) と サーバ B (シャドウ) の物理ドライブ設定を特定します。

2 物理ドライブを設定します。

3 運用環境のダンプを実行します。

4 サーバ B に古いシステムをインストールします。

5 サーバ B の古いシステムを設定して、サーバ A を複写します。

6 インタフェースを更新します。

7 データベースを作成します。

8 サーバ B に サーバ A のダンプをロードします。

9　dbcc コマンド (checktable、checkalloc、checkcatalog) を実行します。

10　ユーザ ID を同期させます。

11　checkpoint を実行します。

Replication Server の設定

1　サーバ B に Replication Server をインストールします。

2　サーバ B の Replication Server を設定します。

3　サーバ A に Replication Server をインストールします。

4　サーバ A (セカンダリ) の Replication Server を設定します。

5　2 つのサーバ間の複写機能を確認します。

6　ターゲット・オブジェクト上で複写をテストします。

7　複写の準備が整った環境を確認してチェックポイントを実行します。

8　サーバを再起動します。

リグレッション・テスト・スイートの実行

バックエンド・リグレッション・テスト・スイート – 運用負荷

1　バックエンド・スクリプトを更新します。

2　バックエンド・リグレッション・テストを繰り返し実行します。

3　システムの動的コマンド (sp_who、sp_lock、statistics io など) をモニタし、取得します。

4　以前のシステムのリグレッション・テストを確認し、文書化します。

フロントエンド・シミュレーション・リグレッション・テスト・スイート

1　フロントエンド・シミュレーション・リグレッション・テストを繰り返し実行します。

2　システムの動的コマンド (sp_who、sp_lock など) をモニタし、取得します。

3　以前のシステムのリグレッション・テストを確認し、文書化します。

フェーズ 1 と 2 のフロントエンド・リグレッション・テスト・スイート

1 ローカル・チームにフェーズ 1 とフェーズ 2 のリグレッション・テストを実行します。

2 ユーザにフェーズ 1 とフェーズ 2 のリグレッション・テストを実行します。

3 動的コマンド (sp_who、sp_lock など) をモニタし、取得します。

4 シャドウ・サーバをアップグレードする準備ができているかどうかを判断します。

5 以前のシステムのリグレッション・テストを確認し、文書化します。

6 サーバ B でパフォーマンスと機能を確認します。

サーバ B (シャドウ) の更新

1 sybsystemprocs を変更します。

2 アップグレード前の確認を実行します。

3 サーバ B を Adaptive Server バージョン 15.0 にアップグレードします。

4 メディア・ダンプから Adaptive Server 15.0 環境をロードします。

5 dbcc コマンド (checktable、checkalloc、checkcatalog) を実行します。

6 15.0 の設定パラメータ値をベースラインに設定します。

7 サーバ A の古いデータベースのダンプを実行します。

8 15.0 システムにダンプをロードします。

9 Adaptive Server 15.0 のデータベースで dbcc コマンド (checktable、checkalloc、checkcatalog) を実行します。dbcc のログとエラー・ログを確認します。

10 Adaptive Server を再起動します。

Adaptive Server 15.0 (サーバ B) でのアップグレード後のリグレッション・テスト・スイートの実行

バックエンド・リグレッション・テスト・スイート – 運用負荷

1 バックエンド・リグレッション・テストを実行します。

2 システムの動的コマンド (sp_who、sp_lock、statistics io など) をモニタし、取得します。

3 リグレッション・テストを確認し、文書化します。

フロントエンド・シミュレーション・リグレッション・テスト・スイート

1　フロントエンド・シミュレーション・リグレッション・テストを実行します。

2　システムの動的コマンド (sp_who、sp_lock など) をモニタし、取得します。

3　リグレッション・テストを確認し、文書化します。

フェーズ 1 と 2 のフロントエンド・リグレッション・テスト・スイート

1　フェーズ 1 とフェーズ 2 のリグレッション・テストを実行します。

2　システムの動的コマンド (sp_who、sp_lock など) をモニタし、取得します。

3　リグレッション・テストを確認し、文書化します。

その他のテスト

15.0 のパフォーマンスと機能を確認します。

Adaptive Server 15.0 (サーバ B) でのユーザ受け入れテストの実行

1　運用テスタに、バージョン 15.0 のユーザ・リグレッション・テストを実行してもらいます。

2　システムの動的コマンド (sp_who、sp_lock など) をモニタし、取得します。

3　リグレッション・テストを確認し、文書化します。

4　バージョン 15.0 が成功かどうかを確認します。

Adaptive Server 15.0 (サーバ B) への運用ユーザの移行

1　運用アクティビティがないことを確認します。

2　サーバ A (運用) のダンプを実行します。

3　サーバ B (シャドウ) にダンプをロードします。

4　サーバ B にロードしたばかりの前のリリースのデータベースで dbcc コマンド (checktable、checkalloc、checkcatalog) を実行します。dbcc のログとエラー・ログを確認します。

5　IP アドレスを切り替え、マシンとサーバの名前を変更します。

6　ユーザのテストと確認を実行します。

最終手順の実行

1 複写を有効にします (サーバ B からサーバ A への複写)。

2 バージョン 15.0 (サーバ B) で運用ユーザを開始します。

注意 置き換えアップグレードを実行するため古いバージョンのデータベースがない場合には、古いサーバ用の Adaptive Server インストール環境を新たに作成する必要があります。

新たなインストール環境を作成し、dump と load、または同様のコマンドを使用してデータをマイグレートした場合は、古いインストール環境で古いバージョンの Adaptive Server を起動します。場合によっては、bcp out を使用して、変更されたデータを 15.0 のサーバから古いサーバにコピーすることが必要です。

カットオーバ・マイグレーション作業リストの例

このサンプル作業リストは、カットオーバ・マイグレーション方法を示します。この作業リストのシナリオは、次のとおりです。

- 中規模の企業で、ある程度フォールト・トラレントで簡単なアップグレードを必要としています。
- この会社では、障害発生時に、毎晩取っているバックアップに依存します。1 時間以上システムをダウンさせることはできず、8 時間を超えるデータの消失は許されません。
- 開発システム、テスト・システム、運用システムで環境が構成されています。

この例では、前の例で説明した準備手順は省略し、次の作業リストを示します。

- 開発システムでの Adaptive Server 15.0 環境の設定
- テスト／受け入れ基準の定義 ― リグレッション・テスト・スイート
- テスト・システムでのフォールバック手順の定義
- テスト・システムでの古い環境のベースライン化
- 古いリリースのテスト・システムでのリグレッション・テスト・スイートの実行
- テスト・システムのバージョン 15.0 へのアップグレード
- バージョン 15.0 のテスト・システムでのリグレッション・テスト・スイートの実行

- バージョン 15.0 のテスト・システムでのユーザ／受け入れテストの実行
- テスト・システムでのフォールバック手順の実行
- Adaptive Server バージョン 15.0 への運用サーバのアップグレード
- 最終手順の実行

開発システムでの Adaptive Server 15.0 環境の設定

1. 物理ドライブとローカル配列を設定して、現在の開発システムの環境を複写します。
2. 現在の開発システムのダンプを実行します。
3. Adaptive Server 15.0 をインストールします。
4. 現在の開発環境を複製して、開発システムの Adaptive Server 15.0 を設定します。
5. interfaces ファイルを更新します。
6. Adaptive Server 15.0 にデータベースを作成します。
7. 現在の開発システムのダンプをロードします。
8. Adaptive Server 15.0 のデータベースで dbcc コマンド (checktable、checkalloc、checkcatalog) を実行します。dbcc のログとエラー・ログをチェックします。
9. 古い開発システムと新しい開発システムの間でユーザ ID を同期させます。
10. 15.0 環境で checkpoint を実行します。
11. Adaptive Server 15.0 に開発を移行します。開発者は検証と新機能の開発を開始します。

テスト／受け入れ基準の定義 – リグレッション・テスト・スイート

フロントエンド・シミュレーション・リグレッション・テスト・スイート

1. ターゲット・ユーザ関数を特定します。
2. SQL コードを取得し、ターゲット・ユーザ関数にマップします。
3. ユーザ関数の SQL コードをカプセル化します。
4. フロントエンド・シミュレーション・テスト・スイート (showplan、stat io、stat time の各ラッパ) を作成します。

フロントエンド・リグレッション・テスト・スイート

1 フロントエンド・テスト・シナリオを特定します。

2 フロントエンド・アプリケーションと受け入れ／テスト手順を理解します。

3 機能テストの方法を文書化します。

4 フロントエンド混合マトリックスを構成します。

テスト・システムでのフォールバック手順の定義

1 フォールバック・スクリプトを作成して、古い環境を再作成します。

2 フォールバック手順を文書化します。

テスト・システムでの古い環境のベースライン化

Adaptive Server を再起動します。

古いリリースのテスト・システムでのリグレッション・テスト・スイートの実行

フロントエンド・シミュレーション・リグレッション・テスト・スイート

1 フロントエンド・シミュレーション・リグレッション・テストを繰り返し実行します。

2 システムの動的コマンド (sp_who、sp_lock など) をモニタし、取得します。

3 リグレッション・テストを確認し、文書化します。

フロントエンド・リグレッション・テスト・スイート

1 ローカル・チームにリグレッション・テストを実行します。

2 システムの動的コマンド (sp_who、sp_lock など) をモニタし、取得します。

3 テスト・システムをアップグレードする準備ができているかどうかを判断します。

4 リグレッション・テストを確認し、文書化します。

5 古いリリースのテスト・サーバでパフォーマンスと機能を確認します。

テスト・システムのバージョン 15.0 へのアップグレード

1　現在のシステムのデータベースのダンプを実行します。

2　アップグレード前に、dbcc コマンド (checktable、checkalloc、checkcatalog) を実行します。

3　sybsystemprocs データベースを変更します。

4　アップグレード前の確認を実行します。

5　テスト・システムをバージョン 15.0 にアップグレードします。

6　バージョン 15.0 のデータベースで dbcc コマンド (checktable、checkalloc、checkcatalog) を実行します。dbcc のログとエラー・ログを確認します。

7　15.0 の設定のベースラインを作成します。

バージョン 15.0 のテスト・システムでのリグレッション・テスト・スイートの実行

サーバを再起動してから、テストを開始します。

バックエンド・リグレッション・テスト・スイート − 運用負荷

1　バックエンド・リグレッション・テストを繰り返し実行します。

2　システムの動的コマンド (sp_who、sp_lock、statistics io など) をモニタし、取得します。

3　リグレッション・テストを確認し、文書化します。

フロントエンド・シミュレーション・リグレッション・テスト・スイート

1　フロントエンド・シミュレーション・リグレッション・テストを実行します。

2　システムの動的コマンド (sp_who、sp_lock など) をモニタし、取得します。

3　15.0 システムをアップグレードする準備ができているかどうかを判断します。

4　リグレッション・テストを確認し、文書化します。

フロントエンド・リグレッション・テスト・スイート

1　ユーザ・リグレッション・テストを実行します。

2　システムの動的コマンド (sp_who、sp_lock など) をモニタし、取得します。

3 リグレッション・テストを確認し、文書化します。

その他のテスト

テスト・システムで 15.0 のパフォーマンスと機能を確認します。

バージョン 15.0 のテスト・システムでのユーザ／受け入れテストの実行

1 テスタに、バージョン 15.0 のユーザ・リグレッション・テストを実行してもらいます。

2 システムの動的コマンド (sp_who、sp_lock など) をモニタし、取得します。

3 リグレッション・テストを確認し、文書化します。

4 15.0 の運用システムをアップグレードする準備ができているかどうかを判断します。

テスト・システムでのフォールバック手順の実行

1 バージョン 15.0 のテスト・システムを停止します。

2 古い (現在の運用) リリースのテスト・システムを再作成します。

3 再作成のスクリプトとプロシージャを使用して、古い環境を再作成します。

4 古い運用システムのダンプをロードします。

5 古いデータベースで dbcc コマンド (checktable、checkalloc、checkcatalog) を実行します。dbcc のログとエラー・ログを確認します。

6 古いリリースのテスト・システムで checkpoint を実行します。

Adaptive Server バージョン 15.0 への運用サーバのアップグレード

1 現在の運用システムでデータベースをダンプします。

2 アップグレード前に、dbcc コマンド (checktable、checkalloc、checkcatalog) を実行します。

3 sybsystemprocs データベースを変更します。

4 アップグレード前の確認を実行します。

5 運用システムをバージョン 15.0 にアップグレードします。

6　バージョン 15.0 のデータベースで dbcc コマンド (checktable、checkalloc、checkcatalog) を実行します。dbcc のログとエラー・ログを確認します。

7　運用システムでバージョン 15.0 の設定のベースラインを作成します。

8　ユーザのテストと確認を実行します。

最終手順の実行

1　バージョン 15.0 の運用システムで運用ユーザを開始します。

2　テスト・システムをバージョン 15.0 にアップグレードします。

段階的カットオーバの作業概要

このサンプル作業概要は、Adaptive Server を再構築するための段階的カットオーバ・マイグレーション方法を示します。この作業リストのシナリオは、次のとおりです。

- 大規模の企業に、1 台のサーバで複数のアプリケーションを提供する完成したクライアント／サーバ環境があります。
- この会社では、障害発生時に、毎晩取っているバックアップとトランザクション・ダンプに依存します。1 時間以上システムをダウンさせることはできず、2 時間を超えるデータの消失は許されません。
- 開発プラットフォーム、テスト・プラットフォーム、運用プラットフォームで環境が構成されています。各プラットフォームに複製サーバを配置するため、十分な領域が必要です。
- オブジェクト・レベルでの再構築方式では、データを移動する必要があるため、ターゲット・アプリケーションにシステム・ダウン時間が必要になります。

作業

このシナリオでの作業の概要は、次のとおりです。

1　前のリリースの開発設定を複製して、開発システムの Adaptive Server バージョン 15.0 を設定します。

2　15.0 の開発システムに開発オブジェクトをマイグレートします。

3　リグレッション・テスト・スイートを構成します。

4　オブジェクト・デルタを移動するための bcp スクリプトを構成します。

5 古いテスト環境のベースラインを作成します。

6 テスト・プラットフォームの 15.0 の複製サーバを設定します。

7 15.0 のテスト・システムにテスト／受け入れオブジェクトをマイグレートします。

8 テスト・システムでリグレッション・テスト・スイートを実行します。

9 新旧リリースのテスト・システム間でのオブジェクトの同期を確認します。

10 15.0 のテスト・システムでユーザ受け入れテストを実行します。

11 運用プラットフォームの 15.0 の複製サーバを設定します。

12 15.0 の運用システムに運用オブジェクトをマイグレートします。

13 15.0 の運用システムに運用ユーザを移動します。

14 bcp スクリプトを使用して、新旧リリースの運用システム間でオブジェクトを再び同期させます。

付 録　C

マイグレーションの注意事項チェックリスト

以下のチェックリストは、マイグレーションの計画で考慮する必要がある注意事項を示します。ここに示されているすべての項目が、使用しているサイトに該当するとは限りません。また、これらのリストが、すべての問題を網羅しているわけではありません。マイグレーション・プランの作成の詳細については、「第 3 章 マイグレーション・プランの作成」を参照してください。

論理データ・アーキテクチャ

論理データ・アーキテクチャが次の項目を備えているかどうかを確認します。

- 論理アーキテクチャのデータ・モデルのグラフ表示
- 組織のデータ使用状況を示す有効なマップ
- 組織がデータを使用するためのアプローチ
- プラットフォーム間およびロケーション間でデータを分散するためのアプローチ
- 複写データを管理するためのアプローチ
- レガシー・データを同期するためのアプローチ
- データ・ストアにアクセス、更新、消去するためのアプローチ

論理アプリケーション・アーキテクチャ

論理アプリケーション･アーキテクチャが次の項目を備えているかどうかを確認します。

- 新しい IT アーキテクチャをサポートするために必要な RPC とストアド・プロシージャのリストと簡単な説明
- 共有できる可能性がある関数のリスト
- 初期化、終了、グローバル編集、エラー処理、ロギング、モニタリングなどの共有サービスのリスト
- サーバ、クライアント、ミドルウェア間でのアプリケーションの機能分割のためのアプローチとグラフ表示
- 新しいアーキテクチャに導入する必要のある機能コントロールのリスト (監査プロシージャなど)
- 新しい IT アーキテクチャ･アプリケーションとレガシー･アプリケーションの統合方法の説明
- 新しい IT アーキテクチャ・アプリケーションが準拠する必要のあるグラフィカル・ユーザ・インタフェース標準
- 新しい IT アーキテクチャ・アプリケーションがインタフェースの役割を果たす必要のあるサービスのリスト (イメージ、ボイス･メール、電子メール、ワード・プロセッサ、印刷、ファックス、ファイル転送などのメカニズム)
- 新しい IT アーキテクチャ・アプリケーションに期待されるパフォーマンスを実現するためのアプローチ
- 新しい IT アーキテクチャ・アプリケーションに必要な可用性を実現するためのアプローチ

論理技術アーキテクチャ

論理技術アーキテクチャが次の項目を備えているかどうかを確認します。

- 新しいシステム・インフラストラクチャのハードウェア・コンポーネントとソフトウェア・コンポーネントの、ベンダに依存しない説明 (関数と機能) とグラフ表示
- 通常処理時とピーク処理時のネットワークの予測負荷の情報
- 既存のネットワーク・インフラストラクチャに必要なアップグレードに関する情報 (特定のキャリア・タイプの情報など)
- すべての接続ノード (ワークステーション、データベース・サーバ、ゲートウェイなど) に関する次のような情報
  - 予測数量
  - 新しい IT アーキテクチャの役割 (クライアント、サービス・プロバイダなど)
  - プロトコル処理、格納領域の容量、パフォーマンス、フォールト・トレランス、セキュリティに関するプラットフォーム特性
- システムのインフラストラクチャ・コンポーネントの地理的なロケーションの情報とグラフ表示
- ハードウェアやソフトウェアの機能を使用して可用性要件を満たすためのアプローチ (フォールト・トレランス、ホット・スタンバイなど)

論理サポート・アーキテクチャ

論理サポート・アーキテクチャが次の項目を備えているかどうかを確認します。

- アップグレードまたは導入する次のようなシステム管理プロシージャの情報
  - ソフトウェアの配布
  - パフォーマンス・モニタリングとフォールト・モニタリング
  - フォールト管理
  - 障害時のリカバリ
  - 運用の承認とアクセス制御
- 役割と責任を含む新しいサポート組織
- スタッフの配属プランとトレーニング・プラン

- サポート対象範囲のニーズに対応するための方式 (ロケーションとシフト数)
- 問題解決に必要な応答時間のニーズに対応するための方式

マイグレーション方法の設計

マイグレーション方法が次の項目を備えているかどうかを確認します。

- 新しい IT アーキテクチャの進化を示す、候補アプリケーションの実装シーケンスまたは移行プラン
- 主なマイグレーションの制約に関する情報 (レガシー・システムとのデータ変換や重要なインタフェースなど)
- 新しい IT アーキテクチャが進化してもレガシー・システムの同期を維持するためのブリッジング・ルーチンに関する情報
- アプリケーション開発環境で使用される手法、方法、ツールの情報
- 新しい IT アーキテクチャ・アプリケーションを運用に移す最初の方法
- 新しい IT アーキテクチャ・アプリケーションを構築、テストするためのスタッフの配置、技能、トレーニングの要件
- マイグレーションの準備プロジェクト・プラン

付 録　D

アップグレード前のチェックリスト

このチェックリストは、インストール・ガイドのチェックリストと併用できます。アップグレードを準備するための手順を、最初に行う作業からアップグレード直前に行う作業まで、順を追って説明します。

アップグレード前のチェックリスト

1　Adaptive Server の動作が確認されている最新バージョンのオペレーティング・システムをインストールします。

2　十分なディスク領域があることを確認します。要件については、インストール・ガイドを参照してください。

3　各データベースに少なくとも 10% の空き領域があることを確認します。この領域がアップグレードに必要です。

4　アプリケーションを確認して、SQL の変更による影響を評価します。「第 4 章 必要なアプリケーションの変更」を参照してください。

5　スクリプトやアプリケーションなどに新しい予約語があるかどうかを確認します。このマニュアルの「新しい予約語」を参照してください。

6　アプリケーションとスクリプト内の廃止された環境変数を置き換えます。

7　すべてのデータベースのサイズとデバイス・フラグメント情報を記録します。この情報については、sysdevices テーブルと sysusages テーブルに問い合わせることができます。

8　デフォルトの文字セットとソート順を書き留めます。文字セットとソート順については、『System Administration Guide』(http://manuals.sybase.com:80/onlinebooks/group-as/asg1250e/sag/@Generic__BookView) を参照してください。

9　sybsystemprocs が十分なサイズであり、フラグメントが複数ある場合は、すべてのデバイス・フラグメントにログ・セグメントとデータ・セグメントの両方があることを確認します。詳細については、「第 5 章 データベース管理の変更」の「sybsystemdb」を参照してください。

アップグレードの直前に、以下の作業を行います。

1 “sa” のデフォルト・データベースとして master を設定します。

2 エラー・ログと設定ファイルのロケーションがデフォルトのロケーションでない場合は、デフォルトのロケーションに変更します。

3 次のバックアップを取ります。

- すべてのデータベース
- デバイス
- Sybase ディレクトリ
- (Windows の場合) レジストリの Sybase エントリ

4 次の重要なシステム・テーブルに対して bcp out を実行します。

- syslogins
- sysloginroles
- sysdatabases
- sysusages
- sysdevices

5 Windows の場合、Adaptive Server のサービスの種類を「手動」に変更します。

6 ミラーリングを無効にします。

7 sp_auditoption “enable auditing”、“off” に設定して監査を無効にします。

8 サーバ設定 (*.cfg*) ファイルを保存します。

索引

記号

数字

A

B

C

D

E

G

I

L

O

R

S

U

X

あ

い

え

お

か

き

く

け

こ

さ

し

す

せ

そ

た

ち

て

と

な

ね

は

ひ

ふ

へ

ま

め

も

ゆ

よ

ら

り

ろ

わ