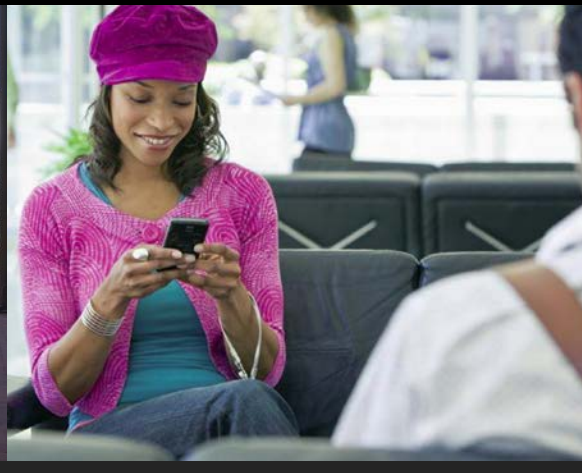
SYBASE
An SAP Company

SQL Anywhere から ASE へのマイグレーション

A TECHNICAL WHITE PAPER





目次

目次

著者
Glenn Paulley paulley@sybase.com

執筆協力者
Mark Culp mculp@sybase.com
Karim Khamis kkhamis@sybase.com

改訂履歴
第 3 版 – 2011 年 7 月
第 2 版 – 2011 年 6 月
第 1 版 – 1998 年、著者 Tom Slee

はじめに

Sybase は、ASE (Adaptive Server Enterprise)、SQL Anywhere、Ultra Light、Sybase IQ など、それぞれに特色のあるデータベースサーバー製品を提供しています。このテクニカルホワイトペーパーでは、アプリケーションを SQL Anywhere から ASE に移行する場合について説明します。

SQL Anywhere と ASE の両製品は、それぞれまったく異なる 2 つの市場をターゲットとするリレーショナルデータベース管理システムとして、同時期に開発されました。SQL Anywhere の主なターゲットとしては、ランタイムリソースが重視される埋め込み環境やモバイルコンピューティング環境に加えて、低メンテナンス性と Web 統合が重視されるワークグループサーバーアプリケーションが挙げられます。これに対して、ASE は、中・大規模のエンタープライズ環境向けのミッションクリティカルな OLTP アプリケーションを主なターゲットとするデータベースサーバーです。実際の環境で求められる高レベルなパフォーマンスや機能性を実現すると同時に、高度な構成、チューニング、および管理機能を提供します。

Sybase のお客様の中には、サーバー環境の統合や ASE 製品固有の機能の活用といった目的で、単独または複数の SQL Anywhere アプリケーションを ASE サーバーに移行することが可能かどうかを検討する必要に迫られている方もいらっしゃることでしょう。このホワイトペーパーでは、SQL Anywhere 12.0.1 と ASE 15.5 の両製品の特徴を比較しながら、システム間の主な差異について概要を説明します。

開発の経緯やターゲットの市場が異なるということからもわかるように、SQL Anywhere と ASE は「プラグコンパチブル (完全互換)」ではありません。つまり、両製品では機能セットが異なるので、SQL Anywhere を使用するアプリケーションを作成した場合、そのままでは ASE 環境で動作しない可能性があります。逆に、ASE 固有の機能を使用するアプリケーションを作成した場合、SQL Anywhere 環境で動作しない可能性もあります。したがって、両データベース管理システム間でアプリケーションを移行する際には、必要に応じてカスタマイズすることになります。

このホワイトペーパーは、アプリケーションを SQL Anywhere から ASE に移行する際に最小限の労力で対応できるよう、必要な知識と情報をお客様に提供することを目的としています。具体的には、両製品の差異を説明し、アプリケーションのスムーズな移行を可能にする移行プロセスを提案します。さらに、データベースシステム間で、アプリケーションの移植性を最大限に高めることを希望している開発者向けにチェックリストを紹介します。

このホワイトペーパーの利用法

このホワイトペーパーの内容に興味がある読者の方は、次のどちらかのケースに該当すると考えられます。「既存の SQL Anywhere アプリケーションを ASE に移行したい」というケースと、「SQL Anywhere アプリケーションを開発する際に、ASE への移植性を最大限に高めたい」というケースです。

前者のケースでは、既存のアプリケーションが存在していることが前提になります。詳細については、このホワイトペーパーの第 3 項と第 4 項（「製品の差異の概要」と「移行プロセス」）を参照してください。各項には、アプリケーションを SQL Anywhere から ASE に移行する場合に役立つ情報が記載されています。

後者のケースでは、アプリケーションの移植性を最大限に高めることを目的としています。詳細については、このホワイトペーパーの第 3 項（「製品の差異の概要」）を参照してから、「SQL Anywhere から ASE への移植性チェックリスト」の項を参照してください。各項には、アプリケーションの移植性を最大限に高めると同時に、後の移行に伴う問題を最小限に抑える場合に役立つ情報が記載されています。

このホワイトペーパーでは、SQL Anywhere 12.0.1 と ASE 15.5 の両製品の特徴を比較しています。このホワイトペーパーに記載されている内容のほとんどは、両製品の過去および今後のバージョンにも当てはまると考えられます。ただし、バージョンアップの繰り返しにより大幅な変更となっている可能性もあるため、動作や構文規則の詳細な説明については、各製品の特定のバージョンに対応するマニュアルを参照してください。両サーバーは固有のさまざまな機能で構成される複雑なソフトウェア製品であり、その互換性の分析には限界があります。

両製品の SQL 言語機能については、執筆時点で最新の SQL 言語の国際標準である ISO 9075 SQL/2008 標準と比較しています。

アプリケーションの要件、作業負荷、およびシステムのコンピューティングインフラストラクチャーは、それぞれ固定されるものではありません。このため、このホワイトペーパーでは、特定の用途に対して SQL Anywhere または ASE のどちらが適しているのかという点については言及しません。さらに、このホワイトペーパーに記載されている情報によって、アプリケーションやデータベースシステムのパフォーマンス、ベンチマーク、パフォーマンスの測定およびチューニングに伴う課題が解決するわけではありません。説明を簡潔にするため、このホワイトペーパーでは、単一のデータベース、数ギガバイト程度のデータ量、およびユーザー数が（数千人ではなく）数百人のユーザーコミュニティという、比較的シンプルな構成の SQL Anywhere アプリケーションを想定しています。高可用性、読み取り専用スケールアウト、複製または同期化、プロキシテーブル（SQL Anywhere のマルチデータベース機能）など、さらに高度な SQL Anywhere の機能については、このホワイトペーパーでは取り上げません。SQL Anywhere 環境が想定よりも複雑な場合でも（たとえば、データベース間に相互関係が設定されている場合など）、このホワイトペーパーに記載されているアドバイスは有効です。ただし、実際の移行プロセスは、このホワイトペーパーに記載されているプロセスから、さらに増える可能性があります。

製品の差異の概要

冒頭で簡単に触れたように、SQL Anywhere と ASE は、開発の経緯やターゲットの市場がまったく異なる製品です。このため、両製品にはさまざまな差異があります。この項では、構造、管理、言語、インターフェイスといった観点から、SQL Anywhere と ASE の差異について説明します。

構造

SQL Anywhere データベースと ASE データベースの構造には、それぞれ異なる部分があります。また、SQL Anywhere サーバーと ASE サーバーでは、動作が大きく異なります。

ASE サーバーでは、ASE サーバーによって管理される他の全データベースの情報を含む、マスターと呼ばれるデータベースが常に存在します。さらに、新しいデータベースのテンプレートである、モデルと呼ばれるデータベースに加えて、*sybsystemprocs* データベース、*sybsystemdb* データベース、テンポラリテーブルが作成される *tempdb* データベースなども存在します。これに対して、SQL Anywhere では、マスターやモデルのようなサーバー全体の構成データベースは存在しません。SQL Anywhere サーバーでは、それぞれ専用の構成関連の情報を含む、ユーザーデータベースを管理します。SQL Anywhere では、データベースごとに一意なテンポラリテーブルが (必要に応じて) テンポラリファイルに書き込まれ、データベースが起動するたびに自動的に再作成されます。

ASE サーバー内では、ASE サーバーによって使用されるディスクの所有者はデータベースではなく ASE サーバーです。ストレージデバイスの領域は、仮想ディスクデバイスとしてサーバーで使用可能になります。特定のデータベースでは、いずれかの仮想ディスクデバイスのストレージが必要に応じて使用されます。各デバイスは、raw ディスクパーティション (UNIX システムで一般的に使用される) またはオペレーティングシステムファイルに該当します。通常、マスターデータベースは、リカバリに備えて、ユーザーデータベースとは別のメディア上に保持されます。ASE では、任意のデータベースデバイスを別のディスクにミラーリングし、ディスクを分割することで、メディアに障害が発生した場合でもリカバリをスムーズに実行できます。この機能は、SQL Anywhere には用意されていません。

SQL Anywhere では、最大 16 個の dbspace と呼ばれるファイルでデータベースが構成されています。各 dbspace は、通常のオペレーティングシステムファイルです。テーブルやインデックスなど、データベースオブジェクトを特定の dbspace 内に作成できるという点で、dbspace は ASE のセグメントと似ている部分もあります。各 SQL Anywhere データベースの最小構成には、プライマリデータベース dbspace、トランザクションログ dbspace、およびテンポラリ dbspace が含まれます。オプションとして、DBA はミラートランザクションログ dbspace を作成できます。さらに、管理者は最大 12 の dbspace を追加し、永続テーブル、マテリアライズドビュー、および／またはそのインデックスを格納することで、データベースオブジェクトを複数のディスクに分散させることができます。各 dbspace は通常のオペレーティングシステムファイルなので、通常のファイルシステ

ムユーティリティを使用して、SQL Anywhere データベースをコンピューター間でコピーできます。SQL Anywhere データベースは特定のオペレーティングシステムに依存せず、コンピューター同士のアーキテクチャー、チップセット、ファイルシステムなどが異なる場合でも、そのままコンピューター間で移動できます。

サーバーの動作中に、SQL Anywhere では、論理操作の内容（INSERT、UPDATE、DELETE、CREATE TABLE、ALTER TABLE など、個別の SQL 文と同義）がデータベースのトランザクションログに記録されます。SQL Anywhere の dbtran ユーティリティを使用すると、トランザクションログに記録された操作の内容を SQL 文に変換して、別のデータベースに対して再現できます。これに対して、ASE では、データベースページに対する物理操作がトランザクションログに記録されます。このため、データベースのリカバリを除いて、トランザクションログに記録された操作の内容を再現する機能はありません。

ASE サーバーのメモリ管理は、SQL Anywhere の場合よりも柔軟な構成が可能です。SQL Anywhere の場合、サーバーで単一のバッファープールが保持され、メモリ内オブジェクトとクエリ実行用に、データベースファイルとデータベースヒープの両方のページが格納されます。ASE では、特定の用途（ソートなど）や特定のテーブル、パーティション、および／またはインデックスに応じて、DBA は複数の独立したバッファープールを構成できます。バッファープール自体をパーティション化することで、それぞれ異なる容量を使用して、サイズの異なる I/O 操作をサポートできます。

ASE は VSA (Virtual Server Architecture) を採用することで、大規模な対称型のマルチプロセッサーコンピューター上でも効率的に動作できます。内部的には、ASE サーバーは複数のオペレーティングシステムプロセスで構成され、各プロセス内では、ASE サーバーは専用の内部スレッドを使用することで、ユーザーの作業負荷を効率的に管理できます。プロセスとスレッドそれぞれの構成は、DBA が管理します。SMP マシン上で実行する場合、ASE では、クエリ間およびクエリ内並列処理により、作業負荷をプロセッサー間で均等に分散させることが可能になります。SQL Anywhere でも、クエリ間およびクエリ内並列処理はサポートされていますが、SQL Anywhere のスレッドモデルはまったく異なるものです。SQL Anywhere では、接続要求はワーカースレッド（ワーカー）の共有プールによって処理されます。プールのサイズは、サーバーのマルチプログラミングレベルに対応しています。ワーカープールのサイズは、システムの負荷に応じて SQL Anywhere によって自動的に変更されます。

このような構造上の差異の多くは通常 1 回かぎりの構成の問題なので、SQL Anywhere から ASE への移行時に大きな問題になることはありません。ただし、重要なポイントとして、ASE データベースオブジェクト (ユーザー、テーブル、トランザクションログなど) はデータベースサーバー内の単一のデータベースに存在する一方で、複数のデータベースを対象とする操作では、データは常に他のデータベースからアクセスできる状態になります。

ASE は、SQL Anywhere よりもリソースの消費量が多くなるということも重要なポイントです。たとえば、メモリ占有容量は ASE の方が多くなります。したがって、ソフトウェアの移行と並行してハードウェアプラットフォームをアップグレードする場合には、注意が必要になります。

管理

データベース管理者の視点では、SQL Anywhere と ASE には大きな差異があります。この差異は、それぞれのサーバーで想定される動作環境の違いに起因するものです。SQL Anywhere には、さまざまなセルフ管理テクノロジが採用されており、ゼロアドミニストレーション環境での動作が可能になっています。これに対して、ASE には、サーバーのランタイム環境を詳細にカスタマイズする目的で、さまざまな管理および診断用の設定が用意されています。このようなカスタマイズの自由度によって、ASE データベースサーバーには SQL Anywhere サーバーよりも必要な管理作業が多くなります。したがって、この両製品の管理作業の差異について把握しておくことは重要なポイントになります。

DBA コマンド

データベースおよび／またはそのユーザーの管理について、SQL Anywhere では、Sybase Central から実行する方法と、特定の SQL DDL 文 (ALTER DATABASE など) を使用する方法があります。これに対して、ASE では、SQL 文として実装されている DBA コマンドの種類は限られています (CREATE DATABASE、CREATE TABLE、CREATE PROCEDURE、CREATE FUNCTION、CREATE INDEX、および GRANT 文)。これ以外の管理コマンドについては、システムストアドプロシージャーとして実装され、DBA に提供されています。

SQL Anywhere で使用可能な ASE の管理ストアドプロシージャーの種類は限られています。付録「SQL Anywhere サポート対象 ASE システムプロシージャー」には、SQL Anywhere でサポートされている ASE システムストアドプロシージャーのリストが記載されています。このストアドプロシージャーの種類によっては、ASE 機能のサブセットのみが実装されていることがあり、実行時に「T-SQL feature not supported (サポート対象外の T-SQL 機能)」というメッセージが表示される可能性があります。

SQL Anywhere システムを ASE に移行する場合、その管理スクリプトの内容を確認した上で、ASE の管理スクリプトとして再作成する必要があります。たとえば、Sybase Central や Interactive SQL のプロシージャーは、ASE の同等のプロシージャーで置き換える必要があります。また、SQL スクリプトは、ASE の同等の SQL コマンドやストアドプロシージャー呼び出しとして再作成する必要があります。

重要なポイントとして、Sybase Central は、ASE と SQL Anywhere の両データベースサーバーを含む、Sybase のさまざまなデータベースサーバー製品や同期製品の管理用途に特化して設計されています。このため、Sybase Central を使用すると、DBA が SQL Anywhere と ASE の両製品を管理する必要がある場合や、SQL Anywhere と ASE の間で移行を実施する必要がある場合に、一定の方法で対応できます。

セキュリティ

ASE によって実現されるセキュリティの大部分は、SQL Anywhere の場合と共通しています。したがって、移行時に問題が生じることはほとんどありません。ただし、両製品には、セキュリティ面でいくつかの相違があります。

SQL Anywhere と ASE で採用されているセキュリティモデルは、いくつかの点で異なります。その大きな理由としては、次の 2 つが挙げられます。

1. ASE 環境には、マスターデータベースが存在します。
2. ASE ではロールベースのセキュリティモデルが提供されるのに対して、SQL Anywhere では権限ベースのセキュリティモデルが提供されます。

ASE 環境では、管理者がユーザーによるサーバーへのアクセスを許可する場合、そのユーザーに対してログイン権限を割り当ててから、アクセス対象の各データベースのユーザー権限を割り当てる必要があります。さらに、管理者は、ユーザーがログインしたときに接続するデータベース (ユーザーのデフォルトデータベース) を指定できます。これに対して、SQL Anywhere 環境の場合は若干シンプルです。ユーザーはアクセス対象の各データベースのログイン権限を必要としますが、SQL Anywhere サーバー全体にアクセスするためのサーバー全体のログイン権限は必要としません。パスワードの管理用に、SQL Anywhere では、CREATE LOGIN POLICY 文によって管理される「ログインポリシー」というメカニズムが採用されています。ASE では、パスワードの管理は sp_passwordpolicy システムプロシージャーによって行われます。

セキュリティ面のその他の重要な差異としては、グループおよびロールの権限モデルが両製品で若干異なることが挙げられます。通常、SQL Anywhere では、グループは一連の SQL Anywhere ユーザーで構成されます。ただし、グループを他のグループのメンバーに設定することも、グループにパスワードを設定してグループとしてのログインを許可することも可能です。

これに対して、ASE では、グループはデータベース内のユーザーのシンプルな集合です。パスワードは設定されず、他のグループから属性を継承できません。ただし、グループに代わるものとして、ロールメカニズムが追加されています。ロールはサーバー全体で有効な構造体であり、ユーザーをメンバーに設定できます。ロールには、パスワードを設定できます。また、ロールを階層状に配置して、特定のロールに既存のロールの属性を継承させることもできます。さらに、ロールにデータベースのセキュリティ権限を割り当てることで、SQL Anywhere のグループ機能と同様の機能を実現できます。ただし、ASE ロールとしてのログインは許可されませんが、パスワードが設定されている場合、SQL Anywhere グループとしてのログインは許可されることに注意してください。

ユーザーパーミッション (DBA ではなく、通常のユーザーに付与されるアクセス権) の観点では、範囲をデータベースに限定すると、ほとんどの権限が共通しています。SQL Anywhere と ASE の両製品では、GRANT 文を使用することで、"alter"、"delete"、"references"、"select"、"update"、および "all" 権限をデータベースユーザーに付与し、テーブルやビューに対する操作の実行を許可できます。さらに、両データベースシステムには、ユーザーがストアドプロシージャーオブジェクトを実行する場合に必要な "execute" 権限が用意されています。SQL Anywhere では、ストアドプロシージャーの実行時に、SQL SECURITY 句を使用する "invoker" と "definer" セキュリティの両方がサポートされています (ASE ではサポートされていません)。

管理パーミッション（ユーザーに割り当てられる権限で、管理タスクの実行を許可するもの）の観点では、SQL Anywhere と ASE にはいくつかの差異があります。ASE には、企業情報システムをターゲットとする製品に求められる、構成可能で高度な管理権限のモデルが用意されています。

SQL Anywhere では、DBA および RESOURCE 権限を使用して、それぞれのユーザーによるデータベースの管理とオブジェクトの作成を許可します。オブジェクトの所有者は、所有するオブジェクトに対する全権限（権限を他のユーザーに付与する権限も含む）を自動的に付与されます。ASE では、サーバーの管理は、次の 1 つまたは複数の ASE システムロールが割り当てられているユーザーによって実行されます。

- sa_role – システム管理者用
- oper_role – データベースのバックアップおよびリストアを実行できるオペレーター用
- keycustodian_role – 暗号キーを管理する管理者用
- sso_role – セキュリティ業務を実行できるシステムセキュリティ責任者用

ASE ログインの sa は上記 3 つのロールに定義された全権限を保持しているので、あらゆるセキュリティ操作を実行できます。

通常、ASE の予約済みデータベースユーザー dbo（database owner ：データベース所有者）は ASE データベース内の全オブジェクトを所有し、権限を必要とするデータベースユーザーに権限を付与します。これに対して、SQL Anywhere の dbo は、さまざまな診断用のシステムストアドプロシージャー、SQL ユーザー定義関数、グローバルテンポラリテーブルとカタログビューに加えて、Mobile Link 同期に使用されるテーブルを所有するグループです。SQL Anywhere では、dbo グループは SYS グループのメンバー、PUBLIC グループは dbo グループのメンバーに相当します。

上記のようなセキュリティ面の差異によって移行時に問題が生じる可能性はほとんどありません。ただし、セキュリティの影響を受けるアプリケーションを使用する場合は、移行の前にセキュリティ権限を確認しておく必要があります。

DBA プロシージャー

データベースサーバー用のデータベース管理プロシージャーは、SQL Anywhere と ASE の両製品で大きく異なります。DBA が両システムで実行する必要があるタスクには共通するものもありますが、多くの場合、タスクを実行するときに使用する文やシステムプロシージャーは、それぞれのシステムで固有のものです。ただし、Sybase Central は SQL Anywhere と ASE のどちらのデータベースサーバーの管理にも対応できるので、DBA は最小限のサーバー固有の知識を習得するだけで済みます。以降の項では、SQL 文やシステムストアドプロシージャーを使用して手動で DBA プロシージャーを実行する方法を紹介しています。各プロシージャーは、Sybase Central でも実行できます。

両システムで異なる主な DBA プロシージャーは、次のとおりです。

スペース管理

SQL Anywhere では、dbinit ユーティリティを実行してデータベースを作成します。

```
dbinit <database file>
```

または、CREATE DATABASE 文を実行します。さらに、別のデータベースファイル内のデータベースにスペースを追加するには、次の文を実行します。

```
CREATE DBSPACE <name> AS <file>
```

ただし、SQL Anywhere では必要に応じて自動的にデータベースファイルを拡張できるので、CREATE DBSPACE 文は通常、単にデータベースのサイズを増やすのではなく、複数の物理ディスクドライブにまたがるデータベースをパーティション化してデータベースのパフォーマンスを向上させる場合に使用します。SQL Anywhere で dbspace を明示的に拡張するには、ALTER DBSPACE 文を使用します。ASE では、データベースのスペースは、最初に DISK INIT 文を使用して割り当てる必要があります。

```
DISK INIT name="<name>", physname="<filename>",vdevno=<nn>, size=<size>
```

さらに、次の文を使用して、1 つまたは複数のデータベースデバイス (スペースの領域) 上に ASE データベースを作成します。

```
CREATE DATABASE <name> ON <device> = <size> LOG ON <device> = <size>
```

(この例は、リカバリのためにデータベースログがデータから切り離されているケースを示しています。このプロシージャーはオプションですが、おすすめします。)
さらに、ASE データベースにスペースを追加するには、ALTER DATABASE 文を実行します。

```
ALTER DATABASE <name> ON <device> = <size>
```

または

```
ALTER DATABASE <name> LOG ON <device> = <size>
```

データまたはログスペースのどちらが必要であるかに応じて、上記のどちらかの文を実行します。ASE では、データベースデバイスのサイズは自動的に増加しないことに注意してください。ただし、ASE は「しきい値」をサポートしており、スペースの調整をよりセルフ管理に委ねることができます。

メモリ管理

デフォルトでは、データベースの作業負荷と他のアプリケーションのメモリ要求に応じて、SQL Anywhere

データベースサーバーは使用するバッファープールの容量を動的に調整します。DBA はサーバーコマンドラインオプションを使用して、サーバーの初期バッファープールサイズと、拡張可能なバッファープールの最小および最大サイズを設定できます。SQL Anywhere は、メモリを大量に消費する操作 (ソートなど) を含む、サーバー操作全般に対して単一のバッファープールを使用します。SQL Anywhere サーバーは、ヒープやその他の内部データ構造に対しても、バッファープールメモリを利用します。したがって、SQL Anywhere インスタンスのメモリ占有容量は、主にバッファープールの最大サイズによって決定されます。Windows プラットフォームでは、SQL Anywhere のデフォルトのバッファープールサイズはデータベースの最大サイズ (バイト単位) とマシンの物理メモリ容量の 4 分の 1 です。

これに対して、ASE サーバーは、ストアドプロシージャー専用のキャッシュなど、特定の用途に応じて個別に構成できる、複数の独立したバッファープールを利用します。さらに、ASE サーバーでは「ネームドキャッシュ」がサポートされているので、DBA は特定のデータベースオブジェクトに対してバッファープールのパラメーターを明示的に指定できます。ASE では、メモリ構成パラメーターを変更するには、DBA が sp_configure システムストアドプロシージャーを使用します。

ASE を構成してアプリケーションに十分なキャッシュ容量を確保するには、『ASE システム管理ガイド』を参照してください。

ユーザーとグループの管理

前述のように、ユーザーが使用する SQL Anywhere データベースごとに、そのユーザーを追加する必要があります。具体的には、GRANT 文を実行します。

```
GRANT CONNECT TO <user> IDENTIFIED BY <password>
```

または、sp_addlogin システムストアドプロシージャーを使用します。

```
sp_addlogin <user>, <password>
```

ASE では、2 段階の手順でユーザーを追加します。最初の手順では、sp_addlogin システムストアドプロシージャーを使用して、サーバーへのアクセス権 (ログイン) をユーザーに付与します。

```
sp_addlogin <login>, <password>
```

2 番目の手順では、管理者が特定のデータベースへのアクセスをユーザーに許可します。具体的には、最初に USE 文で特定のデータベースに切り替えてから、sp_adduser システムプロシージャーを実行します。

```
USE <database name>
go
sp_adduser <login>
```

ログイン名を別のデータベースユーザー名に (または、複数のログインを単一のデータベースユーザー名

に) マッピングすることも可能です。

同様に、次の文を使用して、SQL Anywhere ユーザーグループを作成します。

```
GRANT CONNECT TO <group> IDENTIFIED BY <password>
GRANT GROUP TO <group>
```

さらに、次の文を使用して、ユーザーをグループに追加します。

```
GRANT MEMBERSHIP IN GROUP <group> TO <user>
```

ストアドプロシージャー呼び出しにより、SQL Anywhere グループを追加することも可能です。

```
sp_addgroup <group>
```

さらに、次の文を使用して、ユーザーを追加します。

```
sp_changegroup <group>, <user>
```

ASE では、グループ管理に必要な唯一のインターフェイスは、ストアドプロシージャーインターフェイスです。次のプロシージャーを使用して、ユーザーを新しいグループに追加します。

```
sp_addgroup <group>
sp_changegroup <group>, <user>
```

ユーザーとグループの管理は、SQL Anywhere と ASE でほぼ共通しています。移行時に、ユーザーとグループの管理スプリクトを更新するだけで、ASE ストアドプロシージャーを使用することができます。ユーザー用の特定のメカニズム (パスワードの期限切れなど) は、両製品で異なります。

データベースの作成

前述のように、データベースの作成プロセスは、SQL Anywhere と ASE で異なります。

SQL Anywhere では、それぞれのデータベース自体に、スキーマ、インスタンスデータ、ログインおよび認証情報が格納されています。SQL Anywhere データベースは独立したファイル (dbspace) に分割できますが、特にバックアップやリカバリを考慮すると、ファイル全体を一括管理する必要があります。SQL Anywhere データベースは、プライマリデータベースの dbspace 名をコマンドラインのパラメーターに指定することで、SQL Anywhere サーバープロセスから起動されます。SQL Anywhere の独自機能として、共有メモリ接続を経由したアプリケーションプログラムから接続を開始することで、データベース (およびサーバープロセス) を起動できます。この機能は、ASE ではサポートされていません。

ASE では、サーバーを最初に作成する必要があります (Windows で SyConfig グラフィカルユーティリティを使用)。(マスターデータベースを含む) サーバーを作成してから、次の文を使用して、ユーザーデータベースをサーバーに作成します。

```
CREATE DATABASE <name> ON <device> LOG ON <device>
```

データベースの作成方法の詳細については、『ASE システム管理ガイド』を参照してください。

両製品のプロセスは異なりますが、ともに 1 回実行するだけで有効なタスクです。ASE への移行時には、新しい環境に合わせてデータベース作成スクリプトが更新されていることを確認するだけで済みます。

バックアップとリカバリ

SQL Anywhere と ASE の両製品では、フル機能によるバックアップおよびリカバリを同時に実行できます。ただし、これまでの説明からもわかるように、前提となる両システムのターゲット環境に起因して、いくつかの差異があります。

SQL Anywhere 環境では、データベースサーバーのバックアッププロセスとして、次の手順が挙げられます。

- dbvalid ユーティリティを使用して、データベースが破損している可能性をチェックします。

```
dbvalid -c “uid=<DBAuser>;pwd=<password>;dbf=<dbfilepath>”
```

データベースへの接続が他にも同時に発生している場合、dbvalid ユーティリティで誤検出される可能性があります。

- データベース全体（データベースとトランザクションログ）をバックアップするため、dbbackup ユーティリティを使用します。

```
dbbackup -c “uid=<DBAuser>;pwd=<password>;dbf=<database file>” <backup location>
```

- 別の方法として、トランザクションログのみをバックアップ（増分バックアップ）することも可能です。

```
dbbackup -c “uid=<DBAuser>;pwd=<password>;dbf=<database file>” -t <backup location>
```

SQL Anywhere では、SQL 文（VALIDATE および BACKUP）を使用しても、同じ手順を実行できます。

ASE での検証／バックアッププロセスについては、手順の順序は同じですが、次のように、使用する文が異なります。

☐ データベースの検証には、dbcc ユーティリティを使用します。

```
DBCC CHECKDB ( <database> )
```

☐ データベースのバックアップには、DUMP DATABASE 文を使用します。

```
DUMP DATABASE <database> TO <outputdevice>
```

☐ トランザクションログのバックアップには、DUMP TRANSACTION 文を使用します。

```
DUMP TRANSACTION <database> TO <outputdevice>
```

バックアップのターゲットについても、若干の差異があります。SQL Anywhere ではファイルシステムのディレクトリをバックアップするのに対して、ASE では（ファイルシステムのファイルを含む）出力デバイスをバックアップします。

その他の差異としては、サイトによっては、SQL Anywhere データベースサーバーをシャットダウンしてから、オペレーティングシステムのユーティリティ（ファイルのコピーなど）を使用して、データベースファイルとトランザクションログファイルをバックアップするという点が挙げられます。ただし、このアプローチは ASE では採用しないでください。ASE データベースサーバーをバックアップする場合は、DUMP DATABASE および／または DUMP TRANSACTION 文を必ず使用する必要があります。

SQL Anywhere と ASE のデータベースはともに、Sybase Central からバックアップおよびリストアできるという点も重要なポイントです。

パフォーマンスとチューニング

パフォーマンス分析は、DBA の一般的なタスクです。具体的には、データベースクエリを調査して、実行の高速化や、実行時に必要なリソースの最小化を実現します。クエリのパフォーマンスを向上させるプロセスとしては、元の SQL 要求の変更、データベースオブジェクトの根本的な見直し、システム構成の変更などが挙げられます。

データベースのパフォーマンスチューニングの原則は、リレーショナルデータベースシステム全般で共通しています。また、SQL Anywhere と ASE では、クエリのチューニングプロセスはほぼ同じです。ただし、両データベースシステムの内部的な差異と、診断の実行に使用できるツールについては、差異のポイントとして、DBA が十分に把握しておく必要があります。

SQL Anywhere と ASE の両製品では、コストベースのクエリ最適化を採用することで、列挙された一連の候補から最適なクエリプランを選択します。クエリのコストを評価するため、両システムでは、カラムヒストグラムなどのデータベース統計を使用しています。ただし、両製品のクエリオプティマイザーは、最適化プロセスの制御に使用するアルゴリズムおよびヒューリスティックが根本的に異なります。

統計。SQL Anywhere 製品では、SQL 文の実行に伴い、カラム統計とインデックス統計が自動的に作成および保持されます。カラム統計は、セルフ管理ヒストグラム形式で、各データベースのシステムカタログに格納されます。精度を確保するため、カラム統計は常に監視されており、推定誤差が大きくなった場合には、誤差を補正する定期メンテナンス動作が自動的に開始されます。ただし、SQL Anywhere ヒストグラムは、DBA が CREATE STATISTICS および DROP STATISTICS 文を使用して明示的に管理できます。

これに対して、ASE では、UPDATE STATISTICS、UPDATE INDEX STATISTICS、および UPDATE TABLE STATISTICS 文を使用して、統計を明示的に作成します。また、DELETE STATISTICS 文を使用して特定の統計を削除する方法がサポートされています。両システムでは、SQL 要求を最適化する上で適切な統計が存在しない場合、同様のヒューリスティックが使用されます。

イベント。SQL Anywhere では、イベントハンドラーとともに、システム定義イベントとユーザー定義イベントがサポートされています。各イベントは SQL ストアドプロシージャーと同様に、特定のトリガー動作が発生したときに自動的に呼び出されます。イベントは、ディスクスペースの監視、インデックスの再編成、トランザクションデッドロックの通知など、さまざまな管理を自動化する上で有用なメカニズムとして機能します。TRIGGER EVENT 文を使用して、イベントを手動でトリガーすることもできます。イベントの代わりに、ASE では固有の「しきい値」がサポートされています。たとえば、セグメントの空きディスク領域の容量にしきい値を設定して、ストアドプロシージャーの呼び出しをトリガーするような場合が考えられます。ただし、ASE のしきい値には、SQL Anywhere のイベントのような汎用性はありません。

スレッディング。SQL Anywhere では、システムの負荷に応じて、サーバーのマルチプログラミングレベル（同時要求の最大数）が自動的に調整されます。SQL Anywhere サーバーの SQL 要求はワーカーの共通プールによって実行され、システムのスループットはサーバーによる調整で自動的に最大化されます。ASE では、DBA が特定の作業負荷に応じてサーバーのスレッディング構成を明示的に構成する必要があります。

クエリのヒント。SQL Anywhere と ASE の両製品には、さまざまな方法でクエリオプティマイザーを強制実行する機能が用意されています。ASE に用意されている「抽象プラン」機能を使用すると、特定の SQL 文に対して実行される固有のアクセスプランをかなり正確に作成できます。抽象プランは、SQL Anywhere ではサポートされていません。この他にも、両システムには、精度は劣るものの、クエリオプティマイザーによるアクセスプランの選択に影響を及ぼす方法が用意されています。たとえば、両システムでは、特定のロッキング動作を指定する SQL クエリの FROM 句のヒントがサポートされています。SQL Anywhere では、実行プラン選択への入力として拡張 SQL 構文が用意されており、選択検索条件に対応するロー数の推測（「ユーザー推定」と呼ばれる）が可能になります。ASE では、この構文は使用できません。

プロファイリングとアクセスプラン分析。SQL Anywhere のアプリケーションプロファイリング機能を使用すると、サマリーレベルと詳細レベルの両方で SQL 要求および／または例外の取得が可能になりま

す。さらに、プロファイリングデータは同じサーバー上の異なるデータベースから取得してそのデータベースに保存することも、TCP/IP 接続を介して別のサーバーに伝送することも可能なので、運用データベースに影響を及ぼしません。SQL Anywhere のアクセスプランは、テキスト形式またはグラフィカル形式で表示できます。グラフィカルプランは、Interactive SQL または Sybase Central を使用して表示できます。このプランには、オプティマイザーの推計だけが含まれている場合と、実際のランタイム統計も含まれている場合があるので、推計および実際の述部選択と中間結果のカーディナリティの比較を実行できます。

ASE では、SET SHOWPLAN ON 文を使用して、クエリプランを表示できます。ASE には、(MDA テーブルを含む) クエリプランとクエリパフォーマンスの診断用に、いくつかのメカニズムが用意されています。ASE クエリプランの詳細については、『ASE 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。

インデックス化。パフォーマンスとチューニングの重要なポイントとして、インデックス化も挙げられます。SQL Anywhere では、平衡 B+ ツリーを採用することで、long 値 (LONG VARCHAR、LONG BINARY、および同等の型) を含む、全データ型のインデックス化に対応できます。SQL Anywhere では、CLUSTERED インデックスがサポートされていますが、ベーステーブル内のローの基本ストレージ構造は未変更のままです。CLUSTERED 属性に基づいて、同じページまたは隣接ページ上のローのクラスター化がサーバーで実行されます。ただし、CLUSTERED インデックスに対応するベーステーブルページで、ローの辞書式順序は保証されません。これに対して、ASE では、APL と DOL の両テーブルを対象に、クラスタードインデックス編成と非クラスタードインデックス編成の両方を実行できます。APL テーブルでは、CLUSTERED インデックスによって、辞書式順序はディスク上のローの物理的な順序に一致することが保証されます。ただし、DOL テーブルでは、CLUSTERED インデックスによって実現するのはベストエフォート型のロー配置であり、クエリの ORDER BY 句をサポートするには、ソートの実行が必要になります。

要約すると、両製品のデータベースに対してパフォーマンスとチューニングに関連する作業を実行する場合、プロセスはほぼ同じですが、ASE の方が調整の自由度が高く、クエリ実行環境として適切な構成を実現できます。ASE のパフォーマンスの詳細については、『ASE 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。

システムモニタリング

SQL Anywhere には、サーバー動作のモニタリングに対応するメカニズムがいくつか用意されています。その具体例としては、dbconsole ユーティリティ、Sybase Central、Windows パフォーマンスモニター、および SQL Anywhere サーバーと Mobile Link サーバーの両方のリモートモニタリングに対応する SQL Anywhere Monitor 製品が挙げられます。SQL Anywhere でのトリガーされたイベントのサポートにより、高度な自動制御機能の開発が可能になると同時に、検出した問題の解決を依頼するため、サーバーからシステム管理者への電子メールの自動送信も可能になりました。

Sybase Central では、ASE との連携によるシステムモニタリングも実行できます。その他のモニタリング機能としては、ASE の SCC モニタリングコンソール、ASE サーバーエラーログ、Windows イベントログ、システムストアドプロシージャー（sp_who、sp_lock、sp_monitor、sp_sysmon）などが挙げられます。ASE は Windows パフォーマンスモニターとも統合されているので、インタラクティブなダイアログを使用してモニタリングを実行できます。この他にも、さまざまなサードパーティから ASE のモニタリングツールが提供されています。

両サーバーに用意されているモニタリング機能は多くの部分で共通していますが、それぞれの配備環境に応じて改良が加えられています。SQL Anywhere アプリケーションの移行時には、使用可能な ASE のモニタリング機能を確認しておくことをおすすめします。詳細については、『ASE システム管理ガイド』を参照してください。

データのインポートとエクスポート

データのインポートとエクスポートは、1 回の操作でデータベースサーバーに対して大量のデータをロードまたはアンロードするプロセスです。

SQL Anywhere には、データのインポートとエクスポートに対応するさまざまなメカニズムが用意されています。SQL Anywhere では、次の操作を実行できます。

- ☐ UNLOAD TABLE 文を使用して、単一のテーブルをアンロードする
- ☐ UNLOAD SELECT 文を使用して、クエリ結果をファイルに出力する
- ☐ SQL Anywhere の OPENSTRING 句を SQL 文で使用して、サーバーまたはクライアントのホストコンピューター上にあるファイルの読み取りアクセスを許可する
- ☐ SQL Anywhere のリモートデータアクセス機能（プロキシテーブル）を使用して、別のサーバーやサーバーのホストコンピューター上の基本ファイルシステムに対するデータの読み書きを行う

この他にも、SQL Anywhere に用意されている DBUNLOAD 機能を使用すると、一連のコマンドファイル（データベースオブジェクトの再作成用）および一連のデータファイル（データベーステーブル内のデータの ASCII 表現を含む）として、データベース全体をエクスポートできます。

ASE には、BCP (Bulk Copy ：バルクコピー) として知られる、処理の効率に優れたインポート／エクスポートメカニズムが用意されています。BCP を使用すると、ASCII 形式や独自のバイナリ形式のファイルをデータベースに対してロードおよびアンロードできます。BCP は、C 言語の API や bcp コマンドラインユーティリティからも使用できます。

SQL Anywhere から ASCII 形式でアンロードしたデータを ASE にインポートできるようにするため、各 SQL Anywhere テーブルは個別のファイルにアンロードする必要があります。具体的には、SQL Anywhere の UNLOAD TABLE 文を使用します。

```
UNLOAD TABLE <tablename>
TO <tablename>.dat
DELIMITED BY '¥x09'
QUOTES OFF
```

このアンロードファイルは、次の bcp コマンドを使用して、ASE テーブルにロードできます。

```
bcp <tablename> in <tablename>.dat -c -r "¥r¥n" -U <user> -P <password>
-S<servername>
```

データ移行時に BCP を使用する場合の詳細については、このホワイトペーパーの付録を参照してください。

ASE との互換性を確保するため、SQL Anywhere では ASCII (テキスト) とバイナリの両モードの BCP もサポートされています。BCP の入出力形式は ASE と互換性があるので、BCP を使用すると、SQL Anywhere データベースをアンロードしてそのまま ASE に移行できます。ただし、文字列データやバイナリデータを SQL Anywhere から ASE に移行する場合、DBA は、照合、空の文字列、NULL 値、空白／ゼロパディングなど、両サーバーのセマンティックの差異について注意する必要があります。

別の方法としては、プロキシテーブルを使用する方法があります。プロキシテーブルは SQL Anywhere のリモートデータアクセス機能であり、「プッシュ」モード (INSERT 文を ASE サーバーに発行) で SQL Anywhere データを ASE に移行します。また、ASE の OMNI Connect 機能を使用して、ASE サーバー内の SQL Anywhere データに対してプロキシテーブルを作成する方法もあります。SQL Anywhere に対するプロキシテーブルにより、ASE は SQL Anywhere からデータを取得し、INSERT 文を再実行して ASE データベースに挿入することが可能になります。後者の方法では文字列データに関連する ASE のセマンティックを確実に処理できるので、上記 2 つの方法では、後者をおすすめします。

インポート操作を実行する場合に重要なポイントとして、最初にターゲットテーブルからインデックスを削除し、データのロード後にインデックスを再構築した場合、BCP による ASE データのインポートを高速化できるということが挙げられます。BCP を使用した場合、トリガーは起動されません。

構成

SQL Anywhere と ASE の両データベースサーバー製品は、高度な構成が可能です。ただし、両製品のアーキテクチャー、機能、対象の動作環境の違いにより、一連の構成オプションは両製品で異なります。ASE はエンタープライズシステム環境での動作を想定していることから、SQL Anywhere よりも多くの構成オプションが用意されています。

SQL Anywhere の構成は、次のいずれかの方法で実行します。

- □ 起動時にデータベースサーバーに対して実行するコマンドラインオプション（初期バッファープールサイズ、TCP/IP の無効化など）
- □ sa_server_option システムストアドプロシージャー
- □ SET OPTION 文（接続単位やユーザー単位で、さまざまなオプションのカスタマイズが可能）

ほとんどの場合（ごくわずかな例外を除いて）、SQL Anywhere のオプションは通常のサーバー動作に応じて明示的に設定する必要があります。SQL Anywhere は、さまざまなアプリケーションに対応するデフォルト設定のままで動作するよう設計されています。

サーバーオプション。SQL Anywhere のサーバー構成を変更するには、sa_server_option システムプロシージャーを使用します。付録「SQL Anywhere サーバーオプション」に、sa_server_option による設定が可能な 40 個のサーバーオプションのリストを記載しています。各オプションの大半は、動作中のサーバーの診断（RequestLogging など）や特定のセルフ管理機能（自動統計更新など）の無効化を実行する場合に使用します。一般に、配備時や通常のサーバー運用時に、サーバーオプションの変更が必要になる機会は少ないと考えられます。

データベースオプション。SQL Anywhere には、サーバー動作をカスタマイズする約 120 個のオプション（SET OPTION 文を使用して設定）が用意されています。付録「SQL Anywhere オプション」に、SQL Anywhere オプションと、対応する同等の ASE オプションのリストを記載しています。SQL Anywhere では、Transact-SQL との互換性を確保するため、セマンティックを変更可能な一部の ASE オプションもサポートされています。詳細については、付録「SQL Anywhere による ASE オプションのサポート」を参照してください。

ASE の構成は、ほとんどの場合、システムストアドプロシージャー sp_dboption および sp_configure を使用して実行します。後者のプロシージャー sp_configure は、さまざまな ASE 構成オプションの設定に対応しています（ASE 15.5 で約 390 個のオプション）。sp_configure を使用して設定できる構成オプションは、次のカテゴリに分類されます。

ASE 構成オプション（カテゴリ別）– sp_configure による設定	オプション数
バックアップ／リカバリ	7
キャッシュマネージャー	7
コンポーネント統合サービス	12
診断	7
ディスク I/O	6
DTM 管理	8
エラーログ	4

ASE 構成オプション (カテゴリ別) – sp_configure による設定	オプション数
拡張ストアドプロシージャー	6
一般情報	2
Java サービス	5
言語	5
ロックマネージャー	16
メモリ使用	75
メタデータキャッシュ	9
モニタリング	20
ネットワーク通信	13
OS リソース	5
物理メモリ	11
プロセッサー	4
クエリチューニング	42
REP エージェントスレッド管理	1
セキュリティ関連	25
共有ディスククラスター	13
SQL サーバー管理	69
Unicode	6
ユーザー環境	10

さらに、ASE では、T-SQL SET 文を使用して値を変更するオプションの使用もサポートされています。ASE では、アプリケーション内部からの設定または DBA による設定が可能な合計 81 個のオプションがサポートされています。

ASE 構成の詳細については、『ASE システム管理ガイド』を参照してください。Windows 環境での実行時には、『ASE 設定ガイド Windows版』を参照してください。

重要なポイントとして、両製品は Sybase Central から構成できるということに注意してください。

言語

SQL Anywhere と ASE では、SQL 言語の主要な各種ダイアレクトがサポートされています。SQL Anywhere のネイティブダイアレクトは Watcom SQL であり、SQL/2008 標準 (具体的には、第 3 部「永続的ストアドモジュール」) にほぼ準拠しています。ASE のネイティブダイアレクトは Transact-SQL であり、同様に SQL/2008 標準にほぼ準拠していますが、いくつかの点で標準とは異なります。最も顕著な相違

点としては、ASE Transact-SQL ストアドプロシージャー内の文はセミコロンでは区切られないことが挙げられます。

両製品のネイティブ SQL ダイアレクトは異なりますが、Transact-SQL ダイアレクトの共通サブセットによるプログラミングは可能であり、アプリケーションの移植性は確保されています。便宜上、基本 TDS ワイヤープロトコル（ASE はネイティブ対応）を使用する Open Client および jConnect アプリケーションでは、ASE 互換の動作を可能にするため、固有のオプション設定が自動的に継承されます。具体的には、アプリケーションからデータベースの接続を開始するときに、sp_tsql_environment システムプロシージャーを実行します。sp_tsql_environment システムプロシージャーの内容は、次のとおりです。

```
CREATE PROCEDURE dbo.sp_tsql_environment()
    BEGIN
        IF DB_PROPERTY( 'IQStore' ) = 'Off' THEN
            -- SQL Anywhere datastore
            SET TEMPORARY OPTION close_on_endtrans='OFF';
        END IF;
        SET TEMPORARY OPTION ansinull='OFF';
        SET TEMPORARY OPTION tsql_variables='ON';
        SET TEMPORARY OPTION ansi_blanks='ON';
        SET TEMPORARY OPTION chained='OFF';
        SET TEMPORARY OPTION quoted_identifier='OFF';
        SET TEMPORARY OPTION allow_nulls_by_default='OFF';
        SET TEMPORARY OPTION on_tsql_error='CONTINUE';
        SET TEMPORARY OPTION isolation_level='1';
        SET TEMPORARY OPTION date_format='YYYY-MM-DD';
        SET TEMPORARY OPTION timestamp_format='YYYY-MM-DD HH:NN:SS.SSS';
        SET TEMPORARY OPTION time_format='HH:NN:SS.SSS';
        SET TEMPORARY OPTION date_order='MDY';
        SET TEMPORARY OPTION escape_character='OFF';
    END;
```

共通する *SQL* 言語の構造

ASE と SQL Anywhere でサポートされている SQL 言語には、重複する部分もかなりあります。その一方で、両データベース製品では、さまざまな拡張機能がコアに追加されています。このような拡張機能は、製品間で移植できません。

ASE と SQL Anywhere の両サーバーで共通する SQL の要素は、次のとおりです。

- SQL/2008 標準の基本構文（SELECT、UNION、UPDATE、DELETE、INSERT、GRANT、FETCH、REVOKE、COMMIT、ROLLBACK、CREATE TABLE、CREATE INDEX、CREATE VIEW など）。
- SQL Anywhere では、Watcom SQL と Transact-SQL の両ダイアレクトでの SET 文による変数割り当てがサポートされています。ASE では、SET 文による変数割り当てに加えて、SELECT

文による変数割り当てもサポートされています。SELECT 文による変数割り当ては、SQL Anywhere の Transact-SQL ダイアレクトでサポートされています。

- SQL Anywhere には、システムカタログの ASE 互換ビュー (ASE と同様に、ユーザー dbo が所有者) が用意されています。SQL Anywhere のカタログテーブルとカタログビューの所有者は、ユーザー SYS です。両システムのカタログテーブルの構造には、大きな差異があります。SQL Anywhere のカタログテーブルにアクセスする場合は、カタログベーステーブル自体にアクセスするのではなく、ビューを介してアクセスします。
- SQL Anywhere と ASE の両製品では、quoted_identifier オプションの値に応じて、一重引用符または二重引用符を使用してリテラル文字列定数を囲むことができます。また、角かっこ ([および]) を使用して、識別子を表記することもできます。
- SQL Anywhere と ASE の両製品では、サーバーサイドの双方向カーソルがサポートされています (ローを取得するには、FETCH 文を使用します)。
- SQL Anywhere と ASE の両製品で、CREATE INDEX 文については同じ基本構文がサポートされていますが、若干の差異があります。ASE では、一意なインデックス (または UNIQUE 制約条件) で最大 1 つの NULL 値がサポートされており、SELECT DISTINCT の動作に対応しています。これは、SQL Anywhere のデフォルト設定には当てはまりません。SQL Anywhere では、UNIQUE 制約条件は NOT NULL カラムに対して定義する必要があります。さらに、SQL Anywhere では、一意なインデックスに NULL 値の複数のエントリが格納されることがあります。ただし、NULL 値のインデックスキーは、重複分析で無視されます。SQL Anywhere で ASE と同じ動作を再現するには、CREATE INDEX 文で WITH NULLS NOT DISTINCT 句を使用する必要があります。物理ストレージレイヤー (FILLFACTOR など) に固有の句は、両製品で異なります。
- カタログテーブル内のオブジェクトの難読化は、SQL Anywhere と ASE の両製品でサポートされています。SQL Anywhere では、イベント、マテリアライズドビュー、プロシージャー、関数、トリガー、およびビューを定義し、ALTER 文の SET HIDDEN 構文を使用して、それぞれの定義を難読化できます。同様に、ASE では、システムストアドプロシージャー sp_hidetext を使用して、カタログオブジェクトの定義を難読化できます。
- SQL Anywhere と ASE の両製品では、CLUSTERED インデックスがサポートされています。ただし、ASE では、クラスタードインデックスによって辞書式順序で格納されるよう、テーブルのローの物理的な格納位置が変更されます。SQL Anywhere では、CLUSTERED として宣言されたインデックスはサーバーに対するヒントとして機能します。このヒントに基づいて、INSERT および UPDATE の実行時に、同じディスクページまたは物理的に隣接するディスクページ上に、類似するキー値のローを配置するよう処理されます。したがって、SQL Anywhere では、テーブルのローを辞書式順序で返す際に、ソートの実行が必要になることがあります。
- SQL Anywhere と ASE の両製品では、ログインプロシージャーがサポートされています。ASE では、「ログイントリガー」と呼ばれています。
- SQL Anywhere と ASE の両製品では、AS キーワードによる SELECT リスト式のエイリアス表記がサポートされています。
- SQL Anywhere では、ASE オプション (ansi_permissions、dup_in_subquery、

fipsflagger、flushmessage、quoted_identifier、string_rtruncation、chained、close、nocount、noexec、parseonly、procid、self_recursion、showplan、char_convert、prefetch、rowcount、textsize、datefirst、dateformat、language、role、cursor_rows、ansinull、arithabort、arithignore、identity_insert、offsets、statistics) に対する Transact-SQL の SET 文の構文がサポートされています。Transact-SQL の SET 文を使用して、上記以外の ASE オプションを設定しようとすると、エラーが発生します。詳細については、付録「SQL Anywhere による ASE オプションのサポート」を参照してください。

- ☐ SQL Anywhere では、Transact-SQL のテンポラリテーブル (# で表記)、ローカルテンポラリテーブル (LOCAL TEMPORARY TABLE)、または永続ベーステーブルに対する SELECT INTO を実行できます。ASE では、LOCAL TEMPORARY TABLE 構文はサポートされていません。ただし、ASE では、EXISTING キーワードがサポートされており、ベーステーブルを作成するか、ベーステーブルが既に存在しているかを明示的に制御できます。
- ☐ SQL Anywhere と ASE の両製品では、CREATE TRIGGER 文による INSTEAD OF トリガーがサポートされています。ただし、SQL Anywhere では、Watcom SQL ダイアレクトが使用されている場合にのみ INSTEAD OF トリガーがサポートされています。このため、SQL Anywhere と ASE の両製品で INSTEAD OF トリガーがサポートされているにもかかわらず、トリガーの定義自体は両システム間で移植できません。
- ☐ SQL Anywhere と ASE の両製品では、CREATE TABLE 文の COMPUTE 句 による計算カラムの定義と、計算カラムに対するインデックスの作成がサポートされています。SQL Anywhere では、計算カラムをアプリケーションから直接更新する処理が許可されているのに対して、ASE では許可されていません。ASE では、COMPUTE 式に対する関数インデックスがサポートされています。関数インデックスを使用する場合、計算式はベース (またはテンポラリ) テーブル内のカラムに該当しません。SQL Anywhere では、計算カラムは必ずテーブルのスキーマの構成要素になり、SELECT 文で取得できます。
- ☐ SQL Anywhere と ASE の両製品では、データ型変換関数 CAST()、CONVERT()、INTTOHEX()、および HEXTOINT() がサポートされています。SQL Anywhere では、ASE の関数 HEXTOBIGINT() および BIGINTTOHEX() はサポートされていません。
- ☐ Transact-SQL の各数値関数 (COS()、SIN()、SQRT() など) は、SQL Anywhere でサポートされています。RAND() 関数の動作は、両サーバーで異なります。SQL Anywhere の ATAN2() 関数は、ASE の ATN2() 関数に対応する関数です。ASE では、SQL Anywhere の関数 MOD()、REMAINDER()、および TRUNCNUM() はサポートされていません。
- ☐ 文字列関数 ASCII()、CHAR()、CHARINDEX()、CHAR_LENGTH()、DIFFERENCE()、LEN()、LOWER()、LTRIM()、PATINDEX()、REPLICATE()、REVERSE()、RIGHT()、RTRIM()、SOUNDEX()、SPACE()、STR()、STUFF()、SUBSTRING()、および UPPER() は、ASE と SQL Anywhere の両製品でサポートされており、空の文字列については ASE の動作の影響を受け、文字列関数によって返される値には 16KB の制限があります (下記を参照)。次の点に注意してください。

    - o SQL Anywhere の STR() 関数では、目的の長さに合わせて値を丸める処理は実行

されない代わりに、アスタリスクの生成により、オーバーフローを通知します。

- ASE の STR_REPLACE() 関数は、SQL Anywhere の REPLACE() 関数に対応する関数です。
- SQL Anywhere では、SUBSTRING() 関数のセマンティックは ANSI_SUBSTRING オプションによって制御されます。

□ Transact-SQL の日時関数 GETDATE()、DATENAME()、DATEPART()、DATEDIFF()、および DATEADD() は、両製品でサポートされています。

□ COALESCE() および NULLIF() 関数は、SQL Anywhere と ASE の両製品でサポートされています。

□ SQL Anywhere では、DELETE 文に対する ASE Transact-SQL の拡張機能がサポートされており、複数の FROM 句が許容されます。

□ ASE と同様に、SQL Anywhere では、複数のジョインに対する UPDATE 文がサポートされており、カーソルの SELECT 文でジョインやその他の複合テーブル式が使用されている場合は、UPDATE WHERE CURRENT OF が許容されます。ただし、各文を正確に使用するには、ASE と SQL Anywhere で制限事項に差異があることに注意する必要があります。たとえば、ASE では、複数のスクロール可能なカーソルに対する UPDATE および DELETE WHERE CURRENT OF はサポートされていません。

□ SQL Anywhere では、特殊な比較演算子 *= および =* を使用する Transact-SQL の外部ジョイン構文はサポートされていません。ただし、SQL Anywhere では、オプションを使用してこのサポートを有効にする必要があります。さらに、Transact-SQL の外部ジョインの一部のクラスは、ASE ではサポートされている一方、SQL Anywhere ではサポートされていません。具体例としては、ネストされたテーブル式を含む、Transact-SQL の外部ジョインが挙げられます。Transact-SQL の外部ジョインは、SQL Anywhere ではおすすめしません（下記を参照）。

□ ASE と SQL Anywhere の両製品では、システムプロシージャー xp_startmail()、xp_stopmail()、および xp_sendmail() を使用してサーバーから電子メールを送信する機能がサポートされています。さらに、SQL Anywhere では、xp_startsmtp() および xp_stopsmtp() システムプロシージャーを使用して SMTP 経由で Google メールアカウント宛にメッセージを送信する機能がサポートされています。上記のプロシージャーは、ASE ではサポートされていません。ただし、ASE では、xp_readmail()、xp_findnextmsg()、および xp_deletemail() プロシージャーを使用して電子メールメッセージを受信する機能がサポートされています。

□ SQL Anywhere と ASE の両製品では、xp_cmdshell() 拡張ストアドプロシージャーによるバッチコマンドの実行がサポートされています。ASE では、拡張ストアドプロシージャーを使用するには、Open Server の XP Server が動作している必要があります。SQL Anywhere では、拡張ストアドプロシージャーのサポートは組み込まれています。

□ SQL Anywhere と ASE の両製品では、SQL/2008 標準で定義されている 4 種類のアイソレーションレベル（READ_UNCOMMITTED、READ_COMMITTED、REPEATABLE_READ、および SERIALIZABLE）がサポートされています。ただし、SQL 標準の同時実行制御セマンティッ

クをサポートするため、ロッキングメカニズムを的確に実装する方法は両製品で異なり、その差異は微妙でわかりにくいことがあります。具体例を次に示します。

- SQL Anywhere では、デフォルトのアイソレーションレベルは READ_UNCOMMITTED です。ASE では、デフォルトのアイソレーションレベルは READ_COMMITTED です。
- ASE では、文が READ_UNCOMMITTED アイソレーションレベルで実行された場合、SQL クエリと、各クエリに対して生成されたアクセスプランに制限が適用されます。
- SQL Anywhere のデフォルトでは、削除後にコミットされていないローはブロックされません。
- ローレベルのロッキングに加えて、ASE では、疑似カラムレベルのロッキングと、さまざまなバリエーションのページレベルのロッキングがサポートされています。SQL Anywhere では、疑似カラムレベルのロッキングもページレベルのロッキングもサポートされていません。SQL Anywhere では、LOCK TABLE 文によるテーブルレベルのロッキングがサポートされています。

開発者向けのアドバイスとして、ASE アプリケーションのロッキングと同時実行制御の詳細について、『ASE パフォーマンス & チューニングガイド』の「ロック」を参照することをおすすめします。

- ☐ SQL Anywhere と ASE の両製品では、ロック時の接続待機時間を制限する機能がサポートされています。具体的には、SQL Anywhere では blocking_timeout オプションを使用し、ASE では lock オプション（ASE の SET 文で設定可能）を使用します。ASE の lock オプションは、SQL Anywhere ではサポートされていません。
- ☐ SQL Anywhere と ASE の両製品では、READPAST テーブルヒントと、同時実行制御に影響を及ぼすその他のテーブルヒント（HOLDLOCK など）がサポートされています。ただし、厳密には、ヒント構文は両製品で若干異なります。
- ☐ SQL Anywhere と ASE の両製品では、CREATE SCHEMA 文がサポートされています。
- ☐ SQL Anywhere では、Transact-SQL 文のサブセットが広範にサポートされており、ストアドプロシージャーまたは関数定義の要素に使用できます。
- ☐ Transact-SQL のフロー制御文（WHILE、GOTO、CONTINUE）は、SQL Anywhere によってサポートされている Transact-SQL ダイアレクトで使用可能です。
- ☐ 両製品では、2 つのトランザクションモードがサポートされています。ほとんどの SQL Anywhere アプリケーションは、"chained" トランザクションモードに基づいて作成されています。このトランザクションモードでは、COMMIT を明示的に実行して、データベースに対する変更内容をコミットする必要があります。これに対して、一般的な ASE アプリケーションの多くは、"unchained" トランザクションモードに基づいて作成されています。このトランザクションモードでは、文は独立したトランザクションとして扱われ、正常に実行されるとデータベースに自動的にコミットされます。移行時にはトランザクションモードについても考慮する必要があることに注意してください。chained と unchained のどちらのモードも、両サーバーで使用できます。
- ☐ 組み込みデータ型 INTEGER、SMALLINT、TINYINT、BIGINT、NUMERIC(p,s)、

DECIMAL(p,s)、REAL、DOUBLE、FLOAT、CHAR(n)、NCHAR(n)、VARCHAR(n)、NVARCHAR(n) は、SQL Anywhere と ASE の両製品で共通してサポートされています（下記の注意点に準拠)。互換性を確保するため、その他の ASE のデータ型は、次の表に示すように、ユーザー定義ドメインとして SQL Anywhere でサポートされています。

ASE のデータ型	SQL Anywhere のドメイン名	SQL Anywhere の基本データ型	注
MONEY	MONEY	NUMERIC(19,4)	
SMALLMONEY	SMALLMONEY	NUMERIC(10.4)	
DATETIME	DATETIME	TIMESTAMP	下記を参照
BIGDATETIME	サポート対象外		TIMESTAMP を使用
BIGTIME	サポート対象外		TIME を使用
SMALLDATETIME	SMALLDATETIME	TIMESTAMP	下記を参照
UNICHAR	サポート対象外		NCHAR を使用
UNIVARCHAR	サポート対象外		NVARCHAR を使用
TEXT	TEXT	LONG VARCHAR	下記を参照
IMAGE	IMAGE	LONG BINARY	下記を参照

ASE ではサポートされていない SQL 言語の構成要素・構文

SQL 言語の構成要素・構文のうち、SQL Anywhere の SQL ではサポートされている一方で、ASE ではサポートされていないものは、次のとおりです。

一般

- ☐ ASE では、// はコメント区切り記号としてサポートされていません。SQL Anywhere と同様に、ASE では、先頭に 2 つのマイナス記号 - -、末尾に改行文字を使用する ISO コメント構文に加えて、/* および */ がコメント区切り記号としてサポートされています。
- ☐ ASE では、大文字小文字を区別しない照合を使用するようデータベースが構成されていないかぎり、識別子の大文字小文字は区別されます。ただし、SQL Anywhere では、大文字小文字が区別されるデータベースで定義された場合でも、識別子の大文字小文字は区別されません。大文字小文字が区別されない識別子は、SQL/2008 標準のコア要件に含まれています。
- ☐ ASE では、SQL 言語のキーワードと競合する識別子を表記する場合に、逆引用符（「バッククォート」）の使用はサポートされていません。
- ☐ ASE のデフォルトでは、特に指定がないかぎり、カラムに NULL 値は使用できないという想定です。SQL Anywhere のデフォルトでは、この逆の想定です。SQL Anywhere では、この動作は

allow_nulls_by_default オプションを使用して制御できます。ASE にも同様のオプションがあり、sp_dboption システムプロシージャーを使用して設定します（下記を参照）。

- ISO SQL/2008 標準のアイソレーションレベル（READ_UNCOMMITTED、READ_COMMITTED、REPEATABLE_READ、および SERIALIZABLE）に加えて、SQL Anywhere では、SNAPSHOT、STATEMENT_SNAPSHOT、および READ_ONLY_STATEMENT_SNAPSHOT という 3 種類のスナップショットアイソレーションのモードと、BEGIN SNAPSHOT 文がサポートされています。スナップショットアイソレーションは、ASE ではサポートされていません。SQL Anywhere と ASE の両製品で SQL/2008 標準のアイソレーションレベルがサポートされている一方、その実装方法は大きく異なるので、同時トランザクションのセマンティックに影響を及ぼす可能性があります。

データ型

SQL Anywhere と ASE の両製品では、広範なデータ型がサポートされています。データ型によっては SQL/2008 標準に準拠する一方で、さまざまな旧来の動作を示すものもあり、クエリのセマンティックに影響を及ぼす可能性があります。データ型の互換性というトピックは複雑なため、このホワイトペーパーでは、詳細に説明できません。ただし、データ型に伴う一般的な問題については、以降にまとめています。

暗黙的なデータ型の変換。SQL Anywhere と ASE の両製品では、さまざまな型階層がサポートされています。その結果、さまざまな暗黙的な型変換の規則もサポートされています。たとえば、DATE カラム Y を含むテーブル T に対して、次のクエリを実行します。

```
SELECT Y+5 FROM T
```

SQL Anywhere では、カラム Y の各日付（5 日単位の増分）の TIMESTAMP 値が返されます。ASE では、上記のクエリを実行すると、構文エラーが返されます。これは、ASE の型階層では、INTEGER の DATE 値の加算がサポートされていないことに起因します。構文エラーを回避するには、クエリで ASE の DATEADD() 関数を代わりに使用する必要があります。混合ドメイン式に対する SQL Anywhere のサポートの詳細については、ホワイトペーパー『Mixed-type Comparisons in Adaptive Server Anywhere 9』を参照してください。

ASE でサポートされていないデータ型。SQL Anywhere では、ASE でサポートされていないデータ型の一部がサポートされています。具体的には、次のとおりです。

- BIT_AND()、BIT_LENGTH()、BIT_OR()、BIT_SUBSTR()、および BIT_XOR() 関数とビット単位演算子 &、|、^、および ~ (それぞれ AND、OR、XOR、および NOT に相当) を含む、BIT VARYING (または VARBIT)。BIT VARYING 型は SQL/1999 の言語機能 F511 に相当し、現行の SQL/2008 標

準には含まれていません。ASE では、INTEGER、BINARY、および VARBINARY データ型に対するビット単位演算子 &、|、^、および ~ の使用はサポートされていません。

- ☐ TIMESTAMP WITH TIME ZONE。SQL/2008 の言語機能 F411 に相当します。SQL Anywhere では、DATETIMEOFFSET を TIMESTAMP WITH TIME ZONE のシノニム (同義語) に使用できます。
- ☐ ST_GEOMETRY ベース型とそのサブクラスを含む、あらゆる空間オブジェクトデータ型。さらに、空間述部 ST_CONTAINS、ST_COVERS、ST_COVERSBY、ST_CROSSES、ST_DISJOINT、ST_ISEMPTY、ST_EQUALS、ST_INTERSECTS、ST_OVERLAPS、ST_RELATE、ST_TOUCHES、および ST_WITHIN と、特殊なキャスト関数 TREAT()。ASE では、空間データ型や空間データ関数はサポートされていません。空間データ型は、ISO SQL/MM 標準に含まれています。
- ☐ UNIQUEIDENTIFIER。UUID (Universally Unique Identifier ：ユニバーサルユニーク識別子) に使用され、別名は GUID です。ASE では、UUID に対してユーザー定義のデータ型を作成できますが、VARCHAR または VARBINARY 型で保存されます。ASE では NEWID() 関数がサポートされていますが、SQL Anywhere の UUIDTOSTR() および STRTOUUID() 関数はサポートされていません。

真数型。ASE と SQL Anywhere の両製品では、INTEGER、SMALLINT、および BIGINT 型の符号付きと符号なしのバージョンがサポートされています (各データ型で、同じ値の範囲に対応)。両製品では、TINYINT および UNSIGNED TINYINT データ型もサポートされています。DECIMAL (または NUMERIC) 型の動作については、両製品で若干の差異があります。これは、SQL Anywhere に設定されている DECIMAL 型のデフォルトの桁数と精度が ASE の場合と異なることに起因します。デフォルトの精度と桁数は、SQL Anywhere では (30,6) であるのに対して、ASE では (18,0) です。デフォルトの精度と桁数が異なるので、算術演算の結果に影響を及ぼす可能性があります。

概数型。SQL Anywhere と ASE の両製品では、FLOAT(n)、FLOAT、REAL、および DOUBLE PRECISION 型がサポートされています。

通貨型。SQL Anywhere では、ASE の MONEY および SMALLMONEY データ型は、ユーザー定義のデータ型 (ドメイン) として実装されます。具体的には、それぞれの基本表現に、NUMERIC(19,4) および NUMERIC(10,4) を使用します。SQL Anywhere で実装された MONEY および SMALLMONEY データ型は、ASE の場合よりも値の有効範囲が若干大きくなっています。

文字型とバイナリ文字列型。SQL Anywhere と ASE の両製品のデータ型のサポートで最も大きな差異は、文字とバイナリ文字列のセマンティックに関連するものです。具体例を次に示します。

- ASE では、空の文字列は常に単一の空白文字であると評価され、空の文字列の INSERT を実行すると、カラム値に単一の空白文字が設定されます。これに対して、SQL Anywhere では、空の文字列と単一の空白文字はまったく異なる値として扱われます。
- ASE では、固定長の値の末尾に挿入された空白文字は常に切り捨てられるので意味はありません。さらに、CHAR 値が VARCHAR 値に CAST された場合、末尾の空白文字は一括して自動的に削除されます。末尾の空白文字の切り捨ては、ASE での文字列比較と LIKE 述部の両方のセマンティックに影響を及ぼします。SQL Anywhere では、末尾の空白文字は保持され、有意の文字として扱われます。たとえば、固定長または可変長の文字列値に LENGTH() 関数を使用したときに、末尾の空白文字はカウントの対象になります。
- ASE では、固定長のカラムで NULL 値を許容すると宣言した場合、そのカラムは、セマンティックで対応する差異に基づいて、自動的かつ暗黙的に可変長のカラムに変換されます。SQL Anywhere では、このケースは当てはまりません。ASE での自動型変換の詳細については、『ASE Enterprise 15.5 リファレンス・マニュアル』の「ビルディング・ブロック」を参照するか、『ASE Enterprise 15.5 リファレンス・マニュアル』の「コマンド」に記載された CREATE TABLE 文の説明を参照してください。
- ASE では、デフォルトの文字列値は大文字小文字が区別されるので、SQL/2008 標準に準拠していることになります。SQL Anywhere のデフォルトでは、文字列比較で大文字小文字は区別されません。ただし、大文字小文字の区別は、SQL Anywhere データベースの作成時に指定できます。
- ASE では、固定長の文字およびバイナリ文字列は、宣言された長さまで所定の値が埋められます (パディング)。文字列の場合は文字列値に空白文字が埋められ、バイナリ文字列の場合はゼロが埋められます。このパディングの動作について、文字列の場合は、SQL/2008 標準に準拠しています。バイナリ文字列の場合は、末尾のゼロの扱いは実装で定義されたものです。ただし、SQL Anywhere では、文字列値のパディングは発生しません。これは、空白パディングが有効なデータベースの場合も同様です。ただし、SQL Anywhere では、空白パディングが有効であるという条件で 2 つの文字列を比較する場合、サーバー側で空白パディングの動作を再現できます。SQL Anywhere では、空白パディングの動作は、バイナリ文字列データには影響を及ぼしません。
- ASE では、BINARY、VARBINARY、CHAR、VARCHAR、NCHAR、または NVARCHAR 型のカラムや変数に対して宣言できる最大長は、データベースのページサイズによって制限されます。これは、SQL Anywhere には当てはまりません。SQL Anywhere では、BINARY、VARBINARY、CHAR、VARCHAR、NCHAR、または NVARCHAR 型のカラムや変数の最大長は 32KB に制限されます。ASE では、ページサイズの値が大きい場合は、IMAGE または TEXT 型のカラムを宣言する必要があります。

- ☐ SQL Anywhere と ASE の両製品では、CHAR、VARCHAR、NCHAR、および NVARCHAR 型がサポートされています。漢字型 (National character) のサポートは、SQL/2008 標準の SQL 言語機能 F421 に相当します。
- ☐ SQL Anywhere では、CHAR、VARCHAR、NCHAR、または NVARCHAR 型のカラムや変数の宣言による CHARACTER-length セマンティックがサポートされています。CHARACTER-length セマンティックは SQL/2008 の言語機能 T061 に相当し、ASE ではサポートされていません。
- ☐ LONG BINARY、LONG VARCHAR、および LONG NVARCHAR 型については、SQL Anywhere ではサポートされていますが、ASE では直接サポートされておらず、それぞれ IMAGE、TEXT、および UNITEXT が対応する型になります。UNICODE 文字データは CHAR または NCHAR 型のカラムに格納できるので、UNITEXT 型は SQL Anywhere ではサポートされていません。SQL Anywhere では、ドメインがいずれかの LONG 型に該当する値は、使用に際して特に制約はなく、インデックスの作成も可能です。これに対して、ASE では、TEXT または IMAGE 型の値は、使用に際して厳密な制約があります。インデックスは作成できず、プロシージャーのパラメーターとしての使用、ORDER BY、GROUP BY、または UNION 句での使用、COMPUTE 式での参照、(LIKE 述部を除く) 述部での使用といった用途でも許容されません。
- ☐ ASE では、文字列関数 (SUBSTRING()、RTRIM()、LTRIM()、REVERSE()、UPPER()、および LOWER()) の結果は 16KB に制限されます。これは、SQL Anywhere には当てはまりません。SQL Anywhere では、文字列関数から返される値の最大長は 2GB です。
- ☐ SQL Anywhere では、ASE の場合と異なる CHAR および NCHAR 値の照合がサポートされています。さらに、SQL Anywhere では、UNICODE 文字セット用にオープンソースの ICU ライブラリが使用されています。これに対して、ASE では、専用の UNILIB ライブラリが使用されています。両ライブラリのセマンティックには若干の差異があります。

DATE、TIME、および TIMESTAMP データ型。次の表に、ASE の DATE、TIME、DATETIME、SMALLDATETIME、BIGTIME、および BIGDATETIME ドメインでサポートされている値の範囲と、SQL Anywhere の対応する日付／時刻データ型 (DATE、TIME、および TIMESTAMP) を示します。次の点に注意してください。

- ☐ BIGTIME および BIGDATETIME 型は、SQL Anywhere ではサポートされていません。
- ☐ 日付／時刻データ型 (DATE、TIME、および TIMESTAMP) は、SQL/2008 標準のコア機能に相当します。

ASE のデータ型	範囲	対応する SQL Anywhere のデータ型	範囲	注
TIME	12:00:00AM ～ 11:59:59:999PM	TIME	12:00:00.000000 AM ～ 11:59:59.999999 PM	3
DATE	0001 年 1 月 1 日 ～ 9999 年 12 月 31 日	DATE	0001 年 1 月 1 日 ～ 9999 年 12 月 31 日	5
DATETIME	1753 年 1 月 1 日 ～ 9999 年 12 月 31 日	TIMESTAMP	0001 年 1 月 1 日 ～ 9999 年 12 月 31 日および 12:00.000000AM ～ 11:59:59.999999 PM	1、3、4
BIGDATETIME	0001 年 1 月 1 日 ～ 9999 年 12 月 31 日および 12:00.000000AM ～ 11:59:59.999999 PM	TIMESTAMP	0001 年 1 月 1 日 ～ 9999 年 12 月 31 日および 12:00.000000AM ～ 11:59:59.999999 PM	1、4
BIGTIME	12:00:00.000000 AM ～ 11:59:59.999999 PM	TIME	12:00:00.000000 AM ～ 11:59:59.999999 PM	
SMALLDATETIME	1900 年 1 月 1 日 ～ 2079 年 6 月 6 日	SMALLDATETIME	0001 年 1 月 1 日 ～ 9999 年 12 月 31 日および 12:00.000000AM ～ 11:59:59.999999 PM	2

注：

1. SQL Anywhere の TIMESTAMP 日付型の有効範囲は、1600 年 2 月 28 日 23:59:59 (1600-02-28 23:59:59) ～ 7911 年 1 月 1 日 00:00:00 (7911-01-01 00:00:00) です。この範囲を逸脱すると、TIMESTAMP の時刻部分は不完全になり、分数や秒数に関係する組み込み関数から無効な結果が返される可能性があります。
2. SQL Anywhere では、TIMESTAMP を使用して実装されるドメインという扱いで、ASE の SMALLDATETIME 型がサポートされています。このため、ASE で発生していた SMALLDATETIME 値による日付の制約は解消されています。
3. ASE でサポートされている DATETIME および TIME 型の精度は、マイクロ秒 (約 300 分の 1 秒) に留まっています。BIGTIME および BIGDATETIME 型では、この精度不足が解消されています。

4. SQL Anywhere では、default_timestamp_increment オプションがサポートされています。このオプションでは、値が一意になるように、TIMESTAMP 型のカラムに付加する時間 (マイクロ秒単位) を指定します。ASE には、この機能は用意されていません。
5. SQL Anywhere には、nearest_century オプションが用意されています。このオプションは、年数 2 桁の日付値を挿入するときの ASE の動作を再現する場合に使用します。

その他のデータ型。その他のデータ型のうち、構文やセマンティックで固有の差異を示すものは、次のとおりです。

- ☐ BIT – ASE と SQL Anywhere の両製品では、BIT データ型がサポートされています。ASE では、BIT 型に NULL 値を使用できません。SQL Anywhere のデフォルトでは、BIT 型のカラムで NULL 値は許容されません。ただし、CREATE TABLE および ALTER TABLE 文で、明示的に NULL 値を許容するよう指定することは可能です。SQL Anywhere では、BIT 型のカラムにインデックスを作成できます。これに対して、ASE では、BIT 型のカラムにインデックスを作成できません。
- ☐ TIMESTAMP – TIMESTAMP データ型は、ASE と SQL Anywhere の両製品で異なります。また、DATETIME および BIGDATETIME データ型も同様に異なります。ASE では、TIMESTAMP は古い型であり、TSEQUAL 関数で使用した場合、ローが修正されたかどうかを示すことができます。SQL Anywhere では、CREATE TABLE 文のカラム定義に DEFAULT TIMESTAMP を使用すると、Transact-SQL の TIMESTAMP 型は TIMESTAMP 型のカラムにも格納されます。

データ定義文

SQL Anywhere と ASE の両製品は、データベースの管理用に、異なるメカニズムを採用してきた経緯があります。SQL/2008 標準に含まれている基本 DDL 文 (CREATE TABLE など) 以外に、ASE サーバーには、サーバーの運用管理用の組み込みストアドプロシージャーが用意されています。これに対して、SQL Anywhere の管理機能の大部分は、特定の SQL 文を使用することで実現します。たとえば、データベースにスペースを追加する場合に、ASE では、sp_dbextend システムプロシージャーを使用するのに対して、SQL Anywhere では、ALTER DBSPACE ADD 文を発行します。

このテクニカルホワイトペーパーでは、さまざまな機能のうち、両製品で異なるものに限定して説明しています。したがって、あらゆる内容を網羅しているわけではありません。データベース管理に関連する両製品の詳細については、各サーバーの製品マニュアルを参照してください。

両サーバーで異なる固有のポイントは、次のとおりです。

- ☐ Transact-SQL および Watcom SQL ダイアレクトの SQL ストアドプロシージャーやユーザー定義関数に加えて、SQL Anywhere では、6 種類の外部ランタイム環境がサポートされています。具体例としては、C/C++ で作成された埋め込み SQL および ODBC アプリケーションと、Java、Perl、PHP、さらに Microsoft .NET Framework の CLR (Common Language Runtime ：共通言語ランタイム) に基づく C# や Visual Basic などの言語で作成されたアプリケーションが挙げられます。SQL Anywhere では、従来のインプロセス C/C++ API もサポートされています。ASE では、外部ランタイム環境としてサポートされているのは Java だけです。SQL

Anywhere でサポートされている外部環境に関係する文としては、INSTALL EXTERNAL ENVIRONMENT、REMOVE EXTERNAL ENVIRONMENT、START EXTERNAL ENVIRONMENT、STOP EXTERNAL ENVIRONMENT、ALTER EXTERNAL ENVIRONMENT、START JAVA、および STOP JAVA 文が挙げられます。

- □ マテリアライズドビュー (CREATE MATERIALIZED VIEW、ALTER MATERIALIZED VIEW、REFRESH MATERIALIZED VIEW、および DROP MATERIALIZED VIEW 文) は、ASE ではサポートされていません。
- □ 暗号化されたテーブル、カラム、およびデータベースの管理は、両システムで異なります。SQL Anywhere では、暗号化されたテーブルと暗号化されたデータベースのどちらもサポートされています。これに対して、ASE では、暗号化されたカラムがサポートされています。SQL Anywhere の CREATE [ENCRYPTED | DECRYPTED] DATABASE および CREATE [ENCRYPTED | DECRYPTED] FILE 文は、ASE ではサポートされていません。
- □ イベント (システムイベント、時限イベント、および TRIGGER EVENT 文によるイベントの明示的なトリガー) は、ASE ではサポートされていません。
- □ CREATE TEXT INDEX、ALTER TEXT INDEX、および DROP TEXT INDEX 文は、SQL Anywhere の全文検索機能に含まれています。ASE では、全文検索をサポートするには、サードパーティ製のソフトウェアを使用します。また、全文インデックスの検索結果を含めるクエリ構文は、SQL Anywhere の場合と異なります。
- □ ASE では、SQL Anywhere の CREATE TABLE 文の NOT TRANSACTIONAL 句はサポートされていません。ASE では、ローのストレージレイアウトに関係する SQL Anywhere 固有の句 (PCTFREE、COMPRESSED、NO INDEX、INLINE、および PREFIX 句) はサポートされていません。
- □ ASE では、CREATE TEMPORARY PROCEDURE および CREATE TEMPORARY FUNCTION 文による、テンポラリストアドプロシージャーと SQL ユーザー定義関数の作成はサポートされていません。
- □ DECLARE TEMPORARY TABLE および CREATE LOCAL TEMPORARY TABLE 文は、ASE ではサポートされていません。ASE では、# で始まる名前が指定されたときに、テンポラリテーブルが自動的に作成されます。この構文は、SQL Anywhere でもサポートされています。DECLARE LOCAL TEMPORARY TABLE および CREATE LOCAL TEMPORARY TABLE は、SQL/2008 標準の言語機能 F531 に含まれています。
- □ グローバルテンポラリテーブル (汎用のスキーマが用意されているが、各接続では専用の独立したテーブルのインスタンスが使用される) は、ASE ではサポートされていません。SQL Anywhere では、CREATE GLOBAL TEMPORARY TABLE 文を使用して、グローバルテンポラリテーブルが作成されます。SQL Anywhere の共有グローバルテンポラリテーブル用の構文 (SHARE BY ALL) は、ASE ではサポートされていません。ただし、ASE では、複数の接続で共有できるテンポラリテーブルがサポートされています。共有テンポラリテーブルを宣言するには、テーブル名の先頭に # 記号を使用する代わりに、tempdb. プレフィックスを使用してテンポラリテーブルを宣言します。CREATE GLOBAL TEMPORARY TABLE は、SQL/2008 標準の言語機能 F531 に含まれています。

- ☐ 連鎖参照整合性制約 ON DELETE CASCADE、ON UPDATE CASCADE、ON DELETE RESTRICT、および ON UPDATE RESTRICT の使用と、CHECK ON COMMIT 句の指定による各制約の遅延チェックは、ASE ではサポートされていません。ASE では、トリガーの使用により、連鎖制約が実装されています。
- ☐ SQL Anywhere では、BEFORE および AFTER row トリガーと statement トリガーがサポートされています。FOR EACH ROW トリガーは、SQL/2008 の言語機能 T212 に相当します。ASE では、after statement トリガーのみがサポートされています。
- ☐ SQL Anywhere では、パラメタライズドビューがサポートされており、ビューをクエリで参照するときにインスタンス化する必要がある SQL 変数への参照をビュー定義に含めることができます。この機能は、ASE ではサポートされていません。
- ☐ SQL Anywhere では、外部キー定義での MATCH SIMPLE の使用がサポートされています。ASE では、MATCH FULL のみがサポートされています。MATCH SIMPLE は、SQL/2008 の言語機能 F741 に相当します。
- ☐ SQL Anywhere では、外部キー定義での UNIQUE および CLUSTERED の使用がサポートされており、一意なインデックスやクラスタードインデックスを作成して、外部キー制約を適用できます。ASE では、個別の UNIQUE 制約を宣言するか、一意なインデックスおよび／またはクラスタードインデックスを関連カラムに作成する必要があります。
- ☐ SQL Anywhere では、システム生成されたサロゲートキーの指定用に、DEFAULT AUTOINCREMENT および DEFAULT GLOBAL AUTOINCREMENT 句がサポートされています。ASE では、SQL Anywhere と ASE の両製品でサポートされている IDENTITY 句を使用します。ただし、ASE では、IDENTITY を使用しているカラムに対して複数の制約が課せられます。これは、SQL Anywhere には当てはまらず、自動増分値のギャップの処理は、両製品で異なります。
- ☐ ASE では、SEQUENCE 識別子ジェネレーターはサポートされていません。SEQUENCE 識別子ジェネレーターは、SQL/2008 の言語機能 T176 に含まれています。
- ☐ ASE では、COMPRESSED カラムはサポートされていません。ASE では、DUMP 処理時にデータベースを圧縮できます。ただし、ASE データベースの操作中には、ASE データベースのカラム値は圧縮されません。SQL Anywhere では、カラム単位の圧縮がサポートされています。
- ☐ CREATE TRIGGER 文の ORDER 句は、ASE ではサポートされていません。
- ☐ さまざまなデータ定義文により、SQL Anywhere では、OR REPLACE 構文や IF [NOT] EXISTS 構文 (たとえば、CREATE OR REPLACE PROCEDURE) がサポートされています。このため、バッチ処理やプロージャーのロジックを簡素化できます。ASE では、OR REPLACE と IF [NOT] EXISTS のどちらもサポートされていません。
- ☐ VALIDATE、REORGANIZE、COMMENT ON、BACKUP、RESTORE、UNLOAD、および UNLOAD TABLE 文は、ASE ではサポートされていません。
- ☐ SQL Anywhere でのビューの依存性のトラッキングと、ALTER VIEW 文の実装は、ビューに従属ビューが設定されている場合や、基本となるベーステーブルが変更された場合、ASE でビューを管理する方法と異なります。ASE でビューを使用する場合の詳細については、『ASE リファレンス・マニュアル』の「コマンド」に記載された CREATE VIEW 文の説明を参照してください。

☐ SQL Anywhere では、ベーステーブル、テンポラリテーブル、およびストアドプロシージャーのヒストグラムやその他の統計はセルフ管理されます。ただし、SQL Anywhere では、CREATE STATISTICS、ALTER STATISTICS、DROP STATISTICS、および ALTER DATABASE CALIBRATION 文がサポートされており、各統計や、SQL Anywhere のクエリオプティマイザーによって使用されるコストモデルの明示的な管理に対応しています。これに対して、ASE では、UPDATE [TABLE | INDEX | ALL] STATISTICS 文がサポートされています。

データ操作文

全般的に、SQL Anywhere では、Watcom SQL と Transact-SQL の両ダイアレクトに対応する SQL クエリが個別に実装されています。ただし、はっきりとした例外もあり、たとえば、共通テーブル式は、Transact-SQL のプロシージャー、関数、およびトリガーに含まれる SQL クエリ式ではサポートされていません。

したがって、SQL クエリや DML 文で使用されている構文が Transact-SQL の ASE 実装でサポートされていない場合を判別するのは難しいこともあります。以降では、SQL Anywhere でサポートされているのに対して、ASE でサポートされていない機能を広く分類しています (ただし、あらゆる機能を網羅するものではありません)。SQL Anywhere と ASE の両製品の互換性を正確に把握するには、各製品のマニュアルを参照してください。

DML 文の言語機能のうち、SQL Anywhere ではサポートされている一方で、ASE ではサポートされていないものは、次のとおりです。

☐ SIMILAR TO または REGEXP 述部による正規表現文字列の照合と、REGEXP_SUBSTR() および SIMILAR() 関数は、ASE ではサポートされていません。SIMILAR TO 述部は、SQL/2008 の言語機能 T141 に相当します。SQL Anywhere の REGEXP 述部は、ISO SQL/2008 標準の SQL 言語機能 T841 の LIKE_REGEX () と同等のものです。
☐ MERGE および INSERT ON EXISTING 文。MERGE 文は SQL/2008 標準の F312 および F313 に含まれているのに対して、INSERT ON EXISTING は SQL Anywhere 独自の拡張機能です。どちらの文も、ASE ではサポートされていません。
☐ DML 文内の全文検索に使用する CONTAINS 述部と対応する CONTAINS テーブル式。CONTAINS 述部とテーブル式は、ASE ではサポートされていません。
☐ 集合演算子 INTERSECT [ALL] および EXCEPT [ALL]。EXCEPT は、SQL/2008 標準のコア機能に相当します。INTERSECT [ALL] および EXCEPT [ALL] はそれぞれ、SQL/2008 の言語機能 F302 および F304 に相当します。上記の言語機能は、ASE ではサポートされていません。ただし、INTERSECT [ALL] および EXCEPT [ALL] は、SQL Anywhere の Transact-SQL プロシージャーでは許容されています。
☐ SQL Anywhere では、SELECT リスト内で定義されたエイリアスは SELECT ブロックのネームスペースを構成する要素になり、そのブロック (またはネストされたブロック) 内の任意の句から参照でき

るようになります。これは SQL 標準に対する SQL Anywhere 独自の拡張機能であり、ASE ではサポートされていません。さらに、SQL Anywhere では、エイリアスをテーブルカラムと同じ名前で定義した場合、そのエイリアス定義が支障になり、未修飾のカラムを同じ名前で参照することはできなくなります。たとえば、INTEGER カラム X、Y、および Z を含むテーブル T があるとします。SQL Anywhere では、次の SQL 文

```
SELECT X AS Y FROM T WHERE Y > 5
```

は、次の SQL 文と同義になります。

```
SELECT X FROM T WHERE X > 5.
```

- ☐ OPENXML システムプロシージャーは、ASE ではサポートされていません。
- ☐ LOAD TABLE 文は、ASE ではサポートされていません。ASE では、BCP ユーティリティを使用するか、TRANSFER TABLE 文を使用する必要があります。
- ☐ SQL Anywhere では、WITH 句を使用してクエリ式内でビューをインライン宣言する、共通テーブル式がサポートされています。ASE では、代わりに、明示的に永続ビューを作成する必要があります。共通テーブル式は、SQL/2008 の言語機能 T121 に相当します。
- ☐ SQL Anywhere では、再起共通テーブル式 (WITH RECURSIVE) を使用する再起 SQL クエリ (別名、BOM (Bill-of-Material ：部品表) クエリ) がサポートされています。再起共通テーブル式は、SQL/2008 の言語機能 131 に相当します。再起クエリは、ASE ではサポートされていません。
- ☐ SQL Anywhere では、tsql_outer_joins 接続オプションを ON に設定することで、Transact-SQL の外部ジョイン (*= および =* 二項比較演算子を使用) をサポートできます。ただし、Transact-SQL の外部ジョインのセマンティックは不明瞭であり、SQL Anywhere と ASE の両製品の以前のリリースでは、Transact-SQL の外部ジョインを含むクエリの結果は、アクセスプランによって左右されました。ASE の各リリースで Transact-SQL の外部ジョインのセマンティックにどのような相違があるのか、具体例については、『ASE Transact-SQL ユーザーズ・ガイド』の「Correlated subqueries containing Transact-SQL outer joins」を参照してください。開発者向けのアドバイスとして、Transact-SQL の外部ジョインを実際のアプリケーションに使用することは、例外なく避けることをおすすめします。詳細については、Sybase の Web サイトで公開されているホワイトペーパー『Semantics and Compatibility of Transact-SQL outer joins』を参照してください。
- ☐ SQL Anywhere と ASE の両製品では、CASE 式がサポートされています。ただし、SQL Anywhere では、IF 文に加えて、IF 式がサポートされています。ASE では IF 式がサポートされていないので、ASE を使用する場合は、IF 式を CASE 式に変換する必要があります。
- ☐ 単一の INSERT 文で VALUES 句を使用して複数のローを挿入する機能は、ASE ではサポートされていません。
- ☐ INSERT、UPDATE、DELETE、および MERGE 文の実行によって生成されたテーブル式に対して SELECT を実行し、修正されたローの修正前と修正後のイメージを取得する機能は、ASE ではサポートされていません。次に示すのは、一例です。

```
SELECT old_products.ID, old_products.name, old_products.UnitPrice AS OldPrice,
   final_products.UnitPrice AS NewPrice, SOI.id AS OrderID, SOI.Quantity
FROM
( UPDATE Products SET UnitPrice = UnitPrice * 1.07 )
   REFERENCING ( OLD AS old_products FINAL AS final_products )
   JOIN SalesOrderItems AS SOI ON SOI.ProductID = old_products.ID
WHERE SOI.ShipDate BETWEEN '2000-04-10' AND '2000-05-21' AND SOI.quantity > 36
ORDER BY old_prodcuts.ID;
```

ASE では、この機能を実装するには、UPDATE トリガーを使用して、ローイメージを別のベーステーブルにコピーする必要があります。コピーしたテーブルは、以降の SQL クエリで、別のテーブルとジョインできます。

- テーブル関数。結果セットを返すストアドプロシージャーの呼び出しを SELECT、UPDATE、DELETE、および MERGE 文の FROM 句に追加する機能は、ASE ではサポートされていません。
- FROM 句内のテーブル式（CROSS APPLY、OUTER APPLY、LATERAL、CROSS JOIN、KEY JOIN、NATURAL JOIN、FULL OUTER JOIN、および OPENSTRING 句）は、ASE ではサポートされていません。LATERAL は SQL/2008 の言語機能 T491 に相当し、CROSS JOIN および FULL OUTER JOIN は SQL/2008 の言語機能 F401 に含まれています。
- ASE では、2 つのテーブル間に設定されている参照整合性制約に基づいてジョイン条件を自動的に生成する、SQL Anywhere の KEY JOIN 言語拡張機能はサポートされていません。
- IS [NOT] OF <type> および IS [NOT] DISTINCT FROM 述部に加えて、IS [NOT] TRUE | FALSE | UNKNOWN 論理述部も、ASE ではサポートされていません。
- WINDOW 句とウィンドウ関数（RANK()、DENSE_RANK()、CUME_DIST()、PERCENT_RANK()、ROW_NUMBER()、FIRST_VALUE()、および LAST_VALUE() 関数)。このような OLAP 構造は、SQL/2008 の言語機能 T611 および T612 に含まれています。どちらの SQL 機能も、ASE ではサポートされていません。
- GROUPING() 関数に付随する GROUP BY ROLLUP、GROUP BY CUBE、および GROUP BY GROUPING SETS 句は、SQL/2008 の言語機能 T431 に相当します。ASE では、「ベクトル集合」と GROUP BY ALL 句を使用すると、ほぼ同等の機能を実現できます。
- COVAR_SAMP()、COVAR_POP()、CORR()、REGR_AVGX()、REGR_AVGY()、REGR_SLOPE()、REGR_INTERCEPT()、REGR_R2()、REGR_COUNT()、REGR_SXX()、REGR_SXY() など、線形回帰分析に対応する関数を含む、各種の統計分析関数は、ASE ではサポートされていません。このような関数は、SQL/2008 の言語機能 T611、T612、および T621 に含まれています。
- 結果セットのページネーション（SELECT ブロック内の TOP [START AT] および LIMIT [OFFSET] 句）は、ASE ではサポートされていません。ASE では、SELECT TOP のみがサポートされています。
- SQL Anywhere の日時関数 NOW()、TODAY()、HOUR()、MINUTE()、および QUARTER() は、ASE ではサポートされていません。DAY() 関数および GETDATE() 関数は、

両製品でサポートされています。ASE の GETUTCDATE() 関数は、SQL Anywhere ではサポートされていません。

- 特殊なレジスター（CURRENT USER、CURRENT REMOTE USER、CURRENT TIME、CURRENT TIMESTAMP、CURRENT UTC TIMESTAMP、CURRENT DATE、CURRENT DATABASE、および CURRENT PUBLISHER）は、ASE ではサポートされていません。ただし、特殊なレジスターの一部については、ASE の組み込み関数として同等の機能が用意されています。たとえば、特殊なレジスター CURRENT TIMESTAMP と ASE の組み込み関数 GETDATE() は、同等の機能を提供します。
- SQL Anywhere のデータ型変換関数の一部（STRING()、CASE()、DATETIME() など）は、ASE ではサポートされていません。
- その他の SQL Anywhere の関数（ROWID()、LIST()、ARGN()、IFNULL()、NUMBER()）と文字列連結演算子 || は、ASE ではサポートされていません。
- SQL Anywhere の関数 TRUNCATE()、INSERTSTR()、LCASE()、LOCATE()、SIMILAR()、STRING()、TRACEBACK()、TRIM()、UCASE()、BYTE_LENGTH()、および SORTKEY() は、ASE ではサポートされていません。
- SQL Anywhere のシステムプロシージャー sa_rowgenerator、sa_split_list、および sa_conn_info は、ASE ではサポートされていません。ASE のマスターデータベースに含まれている spt_values テーブルは、sa_rowgenerator プロシージャーや SQL Anywhere の dbo.row_generator システムテーブルの場合と同様の方法で、整数値の SELECT を実行できます。
- SQL Anywhere では、SQL SELECT 文を使用してサーバーやデータベースの設定に対するクエリを実行する PROPERTY()、DB_PROPERTY()、および DB_EXTENDED_PROPERTY() 組み込み関数がサポートされています。ASE では、この用途にはグローバル変数（名前の先頭に @@ が付加された変数）が使用されます。
- DECLARE CURSOR および OPEN CURSOR 文に対する各種の修飾子は、SQL Anywhere と ASE の両製品で異なります。SQL Anywhere では、ASE と同様に、ASENSITIVE および INSENSITIVE カーソルがサポートされています。ただし、SQL Anywhere では、SENSITIVE および KEYSET-DRIVEN カーソルもサポートされています。SQL Anywhere では、コミットされた複数のトランザクションに対してオープンの状態で保持される WITH HOLD カーソルがサポートされています。WITH HOLD 構文は ASE ではサポートされていない一方で、WITH HOLD カーソルが ASE のデフォルトになっています。ただし、ASE で CLOSE ON ENDTRAN オプションを使用すると、この動作を変更できます。
- SQL Anywhere の SELECT、INSERT、UPDATE、DELETE、および MERGE 文で使用される OPTION 句は、ASE ではサポートされていません。
- SQL Anywhere では、FOR UPDATE BY [LOCK | TIMESTAMP] 構文によるカーソルの結果セットに対するペシミスティック（悲観的）およびオプティミスティック（楽観的）な同時実行制御がサポートされています。FOR UPDATE BY LOCK の場合、カーソルから取得されたローに対して意図的ロックが設定されます。ASE では、FROM 句に HOLDLOCK テーブルヒントを使用することで、SELECT 文に対するロックによるペシミスティックな同時実行制御が実現します。ASE では、特殊な

TIMESTAMP データ型と TSEQUAL() 組み込み関数の使用により、オプティミスティックな同時実行制御がサポートされています。

SQL のプロシージャー、トリガー、およびバッチ

前述のように、SQL Anywhere では、Watcom SQL と Transact-SQL の両ダイアレクトがサポートされています。Watcom SQL は SQL Anywhere のネイティブダイアレクトであり、SQL/2008 標準で定義されているストアドプロシージャー言語に対して、Transact-SQL よりも忠実に準拠しています。

SQL Anywhere でサポートされている Transact-SQL ダイアレクトは、SQL Anywhere の各機能を網羅しています。ASE の Transact-SQL と Microsoft SQL Server の Transact-SQL の広範なサブセットのサポートに加えて、特に SQL Anywhere による複合 SQL クエリ式のサポートにも対応しています。この結果、SQL Anywhere で定義された Transact-SQL のプロシージャーは、基本的にそのままでも ASE サーバーで動作できます。

バッチ、プロシージャー、関数、およびトリガーで使用されるダイアレクトの種類については、ダイアレクトに応じて適切なエラー処理を実施するため、サーバー側で判別する必要があります。これは、Transact-SQL のプロシージャーやバッチによるエラー処理は、Watcom SQL の場合と大きく異なることに起因します（下記を参照）。

Watcom SQL と Transact-SQL の SQL プロシージャーや SQL 関数の互換性について、いくつかの重要なポイントがあります。具体例を次に示します。

- **文の区切り記号。**Watcom SQL では、BEGIN-END ブロック内の文を区切る場合、セミコロンを使用します。SQL Anywhere の Transact-SQL ダイアレクトでは、文の区切り記号はサポートされていません。同様に、ASE でサポートされている Transact-SQL でも、文の区切り記号はサポートされていません。
- **名前のスコーピング。**ASE の Transact-SQL ダイアレクトでは、テンポラリテーブルや変数といったオブジェクトを宣言すると、プロシージャーや関数が終了するまで、そのオブジェクトを保持できます。SQL Anywhere では、オブジェクトのスコープは、宣言を含んでいる BEGIN ブロックに制限されます。
- **エラー処理。**デフォルトでは、Watcom SQL ダイアレクトのプロシージャーはエラーが発生した時点で終了し、呼び出し元の環境に SQLSTATE および SQLCODE 値を返します。明示的なエラー処理を Watcom SQL のストアドプロシージャーに組み込むには、EXCEPTION および RESIGNAL 文を使用するか、エラーが発生した場合にも続けて次の文を実行するよう、ON EXCEPTION RESUME 文を使用してプロシージャーで指定します。

これに対して、Transact-SQL ダイアレクトのプロシージャーでは、エラーが発生した場合でも、デフォルトの ON_TSQL_ERROR オプションに従って、以降の文は続行されます。グローバル変数 @@error に、直前に実行された文のエラーステータスが保持されます。SQL Anywhere では、Transact-SQL プロシージャーに使用される ASE の RAISERROR 文がサポートされています。

- **フロー制御文。**SQL Anywhere でサポートされている Watcom SQL および Transact-SQL ダイアレクトには、CREATE PROCEDURE など、一般的な文の異なるバリエーションが用意されています。また、両ダイアレクトに共通する文もサポートされています。Watcom SQL では、専用の LOOP および FOR 制御文、Watcom SQL バージョンの IF および WHILE 文、CALL、SIGNAL、および EXECUTE IMMEDIATE 文、Watcom SQL ダイアレクトバージョンの CREATE PROCEDURE、CREATE FUNCTION、および CREATE TRIGGER 文がサポートされています。SQL Anywhere の Transact-SQL ダイアレクトでは、Transact-SQL バージョンの IF、CASE、WHILE、CREATE PROCEDURE、および CREATE TRIGGER 文と、Transact-SQL の BEGIN TRANSACTION および EXECUTE 文がサポートされています。
- **オプションの動作。**SQL Anywhere では、Transact-SQL の互換性に関連して、さまざまな固有のオプションがサポートされています。具体例としては、ansinull、chained、continue_after_raiserror、on_tsql_error、および tsql_variables 接続オプションが挙げられます。

SQL Anywhere と ASE の両製品での Transact-SQL の互換性に関連する固有の問題と、Watcom SQL と SQL Anywhere の Transact-SQL ダイアレクトの問題は、次のとおりです。

- Watcom SQL と Transact-SQL の両ダイアレクトでは、サポートされている CREATE PROCEDURE、CREATE FUNCTION、および CREATE TRIGGER 文のバージョンが異なります。Watcom SQL ダイアレクトでは、プロシージャー、関数、およびトリガーの本体は、BEGIN/END を使用した複合文であることが前提です。
- ストアドプロシージャーと関数では、パラメーターのデフォルト設定は、Transact-SQL の場合は INPUT、Watcom SQL の場合は INOUT です。
- FOR 文は、ASE ではサポートされていません。FOR 文は、SQL/2008 の言語機能 P002 に含まれています。
- CASE 文は、ASE ではサポートされていません。CASE 文は、SQL/2008 の言語機能 P002 に含まれています。CASE 文は、SQL Anywhere の Watcom SQL と Transact-SQL の両ダイアレクトでサポートされています。
- SQL Anywhere では、Watcom SQL のストアドプロシージャー内と Watcom SQL および Transact-SQL のユーザー定義 SQL 関数内の SQL SECURITY 句（INVOKER および

DEFINER) がサポートされています。SQL SECURITY 句は、SQL/2008 の言語機能 T324 に相当します。SQL SECURITY 句は、ASE ではサポートされていません。

- ☐ トリガー動作条件 INSERTING、DELETING、UPDATING、および UPDATE(<column list>) は、ASE ではサポートされていません。
- ☐ SQL Anywhere では、EXECUTE IMMEDIATE 文がサポートされており、SQL Anywhere の場合、結果セットを返すことも可能です。EXECUTE IMMEDIATE は SQL/2008 の言語機能 B031 に相当しますが、EXECUTE IMMEDIATE 文から結果セットを返す機能は製品独自の拡張機能です。Transact-SQL で対応する文は、EXECUTE 文です。
- ☐ Watcom SQL ダイアレクトのストアドプロシージャーを呼び出す場合、SQL Anywhere では、CALL 文を使用します。これに対して、Transact-SQL では、EXEC 文を使用します。
- ☐ Watcom SQL の BEGIN ATOMIC 文は、ASE ではサポートされていません。代わりに、ASE のトランザクションサポート機能を使用する必要があります。
- ☐ ROLLBACK TO SAVEPOINT および RELEASE SAVEPOINT 文は SQL/2008 の言語機能 T271 に含まれていますが、ASE ではサポートされていません。ASE では、操作を取り消して特定のセーブポイントの状態に戻すには、ROLLBACK 文を使用します。
- ☐ CREATE VARIABLE および DROP VARIABLE 文と、名前が @ 文字で始まらない SQL 変数の使用は、ASE ではサポートされていません。

SQL Anywhere ではサポートされていない *SQL* 言語の構成要素・構文

ASE には、さまざまな SQL 言語機能が用意されています。ただし、SQL Anywhere では使用できないものや、動作に若干の差異があるものも含まれています。文のデフォルト設定や複合的なセマンティックには、両製品で若干異なる部分があるということも重要なポイントです。ASE の SQL 機能のうち、SQL Anywhere でサポートされていない機能は、次のとおりです。

- ☐ SQL Anywhere では、パーティション化されたテーブルはサポートされていません。ASE では、ハッシュベースまたは範囲ベースのパーティション化を実行して、パーティション化されたベーステーブルを作成できます。どちらのオプションも、SQL Anywhere では使用できません。
- ☐ ロールベースのセキュリティは、SQL Anywhere ではサポートされていません。前述の「管理」の項の「セキュリティ」を参照してください。
- ☐ ASE では、特定のマスターデータベースに関連付けられた複数のデータベースへのアクセスがサポートされています。これに伴い、ASE では、*database.creator.table* という形式の 3 つの要素で構成されるテーブル識別子と、*database.creator.table.column* という形式の 4 つの要素で構成されるカラム識別子がサポートされています。SQL Anywhere では、4 つの要素で構成される識別子を解析できますが、各 SQL Anywhere データベースは自己完結型 (他のデータベースを参照する必要がない) なので、識別子の名前の *database* 部分は無視されます。SQL Anywhere クエリ内でのカラム参照の修飾は、*creator.table.column* という形式か、テーブルの相関名を基準にします。

- ☐ ASE の Unicode 文字列関数 TO_UNICHAR()、UHIGHSURR()、ULOWSURR()、および USCALAR() は、SQL Anywhere ではサポートされていません。
- ☐ ASE には、包括的な監査機能が用意されており、ASE データベース内のベーステーブルに監査レコードが保存されます。SQL Anywhere では、監査機能によって追加レコードがトランザクションログに記録されます。ただし、SQL 文からは直接、ログレコードを照会できません。監査証跡を確認するには、SQL Anywhere の dbtran ユーティリティを使用して、トランザクションログを SQL に変換する必要があります。
- ☐ SQL Anywhere では、BIGDATETIME、BIGTIME、UNICHAR、および UNIVARCHAR データ型は直接サポートされていません。SQL Anywhere では、BIGDATETIME データ型は TIMESTAMP にほぼ相当し (ただし、日付の許容範囲の上限値と下限値の精度は劣る)、BIGTIME は SQL Anywhere の TIME 型に相当します。UNICHAR および UNIVARCHAR 型は、SQL Anywhere では、データベースの照合に応じて、CHAR (NCHAR) または VARCHAR (NVARCHAR) 型に宣言する必要があります。TIME、DATE、および TIMESTAMP データ型の互換性の詳細については、前述の「データ型」の項を参照してください。
- ☐ SQL Anywhere では、ASE の SYSNAME および LONGSYSNAME データ型はサポートされていません。
- ☐ ASE では、各カラムの暗号化がサポートされています。SQL Anywhere では、暗号化されたテーブルのみがサポートされています (データベース全体を暗号化することも可能)。
- ☐ SQL Anywhere では、ストアドプロシージャーを含む Transact-SQL のグループはサポートされていません。
- ☐ SQL Anywhere では、単一のブロック内で同じ名前の複数のカーソルはサポートされていません。
- ☐ ASE では、識別子 (テーブル名、カラム名、ビュー名など。最長 255 バイト (区切り識別子の場合は 253 バイト)) がサポートされています。SQL Anywhere では、最長 128 バイトの識別子がサポートされています。
- ☐ SQL Anywhere では、ASE の抽象クエリプランはサポートされていません。
- ☐ SQL Anywhere では、ASE の DBCC 機能はサポートされていません。
- ☐ SQL Anywhere では、トレースフラグはサポートされていません。
- ☐ SQL Anywhere では、SQL プロシージャー、SQL ユーザー定義関数、およびトリガー定義では、Transact-SQL のオプションである WITH RECOMPILE 句の使用がサポートされていますが、指定した場合、無視されます。SQL Anywhere では、プロシージャーはデータベースの起動後、最初に実行された時点で常にリコンパイルされます。コンパイルされたプロシージャーは、スキーマの変更によってプロシージャーが最新の状態でなくなるか、データベースが停止するまで保持されます。
- ☐ SQL Anywhere では、INSERT 文の処理を制御する ASE オプション ignore_dup_key、ignore_dup_row、および allow_dup_row はサポートされていません。テーブル内の重複データ処理をカスタマイズするには、MERGE 文または INSERT ON EXISTING 文を使用します。
- ☐ SQL Anywhere では、ASE の DEALLOCATE CURSOR 文はサポートされていません (解析後、無視されます)。Adaptive Server Enterprise では、Transact-SQL のプロシージャーと関数には、同じカーソル名を使用して、複数の DECLARE CURSOR 文を含めることができます。ASE では、DEALLOCATE CURSOR 文は、現在のスコープからカーソルを消去する場合に使用

します。これにより、以降の OPEN 文では、事前に宣言された正しいカーソルを参照できるようになります。この機能は、SQL Anywhere ではサポートされていません。SQL Anywhere では、特定のスコープ内の各カーソルには、それぞれ一意な名前を設定する必要があります。Transact-SQL ダイアレクトのプロシージャーに同じ名前で複数のカーソル宣言が含まれている場合、解析時には、そのプロシージャーはエラーなしとして処理されます。ただし、実行時には、同じカーソル名で 2 番目の DECLARE CURSOR 文が実行されると、エラーが発生します。SQL Anywhere では、DEALLOCATE CURSOR の代わりに DROP STATEMENT を使用して、前処理 (prepared) 文でシステムリソースを解放します。

- ASE の TRANSFER TABLE 文は、SQL Anywhere ではサポートされていません。代わりに、LOAD TABLE 文 (OPENSTRING および／または READ_CLIENT_FILE との組み合わせ)、UNLOAD または UNLOAD TABLE 文、および INSERT 文を使用することで、SQL Anywhere のリモートデータアクセス機能を使用してデータを転送できます。
- SQL Anywhere では、ドメイン整合性を適用する目的でデータベースオブジェクトとして表現される、ASE の再使用可能な規則やデフォルト設定はサポートされていません。
- SQL Anywhere では、GROUP BY 句に対する ASE の拡張機能 (ベクトル集合や GROUP BY ALL など) はサポートされていません。
- SQL Anywhere では、ASE で使用されるシステム関数やシステムストアドプロシージャーの多くはサポートされていません。詳細については、『ASE リファレンス・マニュアル』の「Procedures」と『ASE システム管理ガイド』を参照してください。
- SQL Anywhere では、ASE のシステム構成に関係する特定の SQL 文の一部はサポートされていません。具体例としては、MOUNT、UNMOUNT、USE、および SHUTDOWN 文が挙げられます。SQL Anywhere では、USE 文は解析後、無視されます。
- SQL Anywhere では、ASE 固有のグローバル変数の多くはサポートされていません。「SQL Anywhere による ASE グローバル変数のサポート」を参照してください。

要約すると、ASE と SQL Anywhere では、両システムでサポートされている SQL の広範なサブセットが共有されていることになります。ただし、SQL Anywhere と ASE の両製品には、それぞれ使用が想定される環境に特化した独自の有用な拡張機能も用意されています。アプリケーションで SQL Anywhere の SQL サポートを使用している場合、アプリケーションを ASE に移行するときにどのような修正が必要になるか、アプリケーション開発者側で判別する必要があります。

インターフェイス

各製品には、さまざまなプログラミングインターフェイスが用意されています。

SQL Anywhere には、次のインターフェイスが用意されています。

- ☐ ODBC インターフェイス
- ☐ OLE DB、ADO、および ADO.NET インターフェイス
- ☐ タイプ 2 JDBC インターフェイス
- ☐ jConnect (ASE のタイプ 4 JDBC インターフェイス。ASE の基本 TDS ワイヤープロトコルを使用) のサポート
- ☐ SQL Anywhere の組み込み Web サービス機能による直接 HTTP および HTTPS 接続
- ☐ PHP、Perl、Python、Django、および Ruby のサポート
- ☐ C プログラミング用の ESQL (Embedded SQL ：埋め込み SQL) インターフェイス
- ☐ TDS を使用する Sybase Open Client アプリケーションのサポート

ASE では、C アプリケーションプログラム用の OpenClient インターフェイスと、次のインターフェイスが用意されています。

- ☐ ODBC インターフェイス
- ☐ タイプ 4 JDBC インターフェイス (jConnect)
- ☐ ADO.NET のサポート
- ☐ 埋め込み SQL のプレコンパイラー (C および COBOL 用)

SQL Anywhere とは異なり、ASE では、共有メモリ接続はサポートされていません。ASE の TDS ワイヤープロトコルでは、TCP/IP が必要になります。動作していないデータサーバーの自動起動は、埋め込み SQL Anywhere アプリケーションで一般的に使用される機能です。ただし、ASE アプリケーションには上記の制約があるので、この機能には対応していません。

Web サービス

ASE と SQL Anywhere の両製品では、Web サービスがサポートされていますが、サポートの範囲や方法は異なります。ASE で採用されているモデルでは、Web サービスは独立した Web サービスエンジンで処理されます。このプロセスは、SOAP Web サービスを提供する場合にも、SOAP Web サービスを利用する場合にも対応できます。

ASE の Web サービスエンジンはサーバーとして (HTTP トランスポートプロトコル上で) SOAP 要求を受け取り、必要な処理を実行して、関連する SOAP 応答を生成します。構成ファイルに基づいて、Web サービスエンジンは SOAP 要求をストアドプロシージャー呼び出しにマッピングするか、ASE データベーステーブルのクエリに直接マッピングします。Web サービスエンジンと ASE サーバーの通信は、jConnect を使用した JDBC 経由で行われます。その際に、ASE の TDS ネットワークトランスポートプロトコルが利用されます。

ASE では、Web サービスを利用する際に、Web サービスエンジンが ASE の CIS (Component Integration Services ：コンポーネント統合サービス) の構成要素として使用されます。外部 Web サー

ビスを利用するため、プロキシテーブルが作成され、SOAP 要求からの応答はプロキシテーブルの関連するロー／カラム構造体にマッピングされます。

SQL Anywhere でも、SOAP Web サービスの提供および利用が可能です。ただし、データベースサーバー内に直接、SOAP のサポートを組み込む必要があります。SOAP Web サービスに対応するため、SQL Anywhere では、組み込み HTTP サーバーを使用して SOAP 要求を受け取ります。SOAP 要求はストアドプロシージャーの呼び出しにマッピングされ、ストアドプロシージャーによって生成された結果セットは組み込み HTTP サーバーによって SOAP 応答に変換されます。

SOAP Web サービスを利用するため、SQL Anywhere では ASE と異なるアプローチが採用されています。CIS およびプロキシテーブルの代わりに、SQL Anywhere では、RPC (Remote Procedure Call ：リモートプロシージャーコール) パラダイムが使用されます。SOAP Web サービスを利用するため、SQL Anywhere データベース内にプロシージャー (または関数) が作成され、外部 Web サービスにマッピングされます。プロシージャーが呼び出されると、SQL Anywhere では、適切な HTTP SOAP 要求が外部 Web サービスに発行され、SOAP エンベロープが呼び出し元に返されます。呼び出し元では、SOAP 応答をリッピングし、SQL Anywhere の OPENXML システムプロシージャーを使用して情報を抽出できます。

SQL Anywhere では、SOAP Web サービスの提供や利用以外の機能も用意されています。組み込み HTTP サーバーと RPC クライアントのサポートにより、HTTP の HEAD、GET、POST、PUT、および DELETE メソッドの処理にも対応できます。その結果、開発者は他のメディア形式を簡単に提供および利用できるようになります。組み込み HTTP サーバーでは、結果セットの HTML、XML、および JSON への自動変換だけでなく、ストアドプロシージャーによる RAW 状態での文字データやバイナリデータの出力も可能になります。RAW データの送受信のサポートにより、専用の Web サーバーやアプリケーションサーバーを用意せずに、データベースサーバーとブラウザーで情報を直接やり取りすることも可能になります。同様に、SQL Anywhere の RPC クライアントプロシージャーのサポートにより、開発者は SOAP 単独のサポートを考慮するだけで、任意の外部 HTTP サーバーに対する HTTP 要求を作成できるようになります。たとえば、外部 HTTP Web サーバーに対して HTTP GET 要求を発行し、そのサイトからリソースをフェッチできます。HTTP メソッド全般のサポートにより、開発者は REST 準拠のアプリケーションを作成および配備できるようになります。

移行プロセス

この項では、ASE データベースサーバー上で動作するよう、7 つのステップで SQL Anywhere アプリケーションを移行するプロセスについて説明します。このプロセス全体で、ASE サーバーのセットアップ、データベースとデータの移行、アプリケーションプログラムの移行、および管理手順に対応します。

次の 7 つのステップで構成されるプロセスを順番に実行することで、SQL Anywhere から ASE への移行を無理なく確実に実行できます。実際の移行プロセスはこの項の説明よりも複雑であり、ステップを実行する順番もそれほど重視されません。ただし、この項で説明している各ステップは理想的なものであり、移行作業を計画する上で参考になる情報も含まれています。

このプロセスの各ステップについては、概要のみを説明しています。各ステップを実行する場合の詳細については、『ASE システム管理ガイド』、『ASE リファレンス・マニュアル』の「プロシージャー」、その他の関連マニュアル (『ASE Transact-SQL ユーザーズ・ガイド』など) も参照してください。

ステップ *1*：サーバー構造の移行

このステップの目的は、ASE サーバー構造を作成して、既存の SQL Anywhere アプリケーションを ASE 環境でサポートできるようにすることです。

ステップ 1.1 ASE サーバーの作成

移行の最初のステップでは、ASE サーバーを作成します。これは手順をそのまま実行するだけのプロセスであり、標準の ASE インストールユーティリティを使用します。インストール先のプラットフォームに対応する ASE インストールガイドの指示全般に従ってください。

ステップ 1.2 ASE サーバーの構成

ASE サーバーでは、アプリケーションをサポートする上で所定の構成が必要になることがあります。具体的には、sp_configure システムストアドプロシージャーまたは Sybase Central を使用して、次の構成パラメーターを設定する必要があります。

- ☐ number of user connections (ユーザー接続数)
- ☐ total memory (合計メモリ容量)
- ☐ number of engines (エンジン数)

number of user connections パラメーターは、アプリケーションを使用する全ユーザーがログインできるように、余裕のある大きな値を設定してください。この値の設定と同時に、total memory パラメーターを設定します。具体的には、サーバーのキャッシュサイズ (サーバー起動時のエラーログに通知) が少なくとも SQL Anywhere で使用されたキャッシュと同じ大きさになるように設定してください。

ステップ 1.3 データベースのスペースの割り当て

ASE サーバーデータデバイスを作成して、アプリケーションで必要になるデータとトランザクションログを保持する必要があります。

データとログを配置する場所を計画する際には、ASE のプラットフォーム固有のマニュアルに記載された指示に従ってください。ただし、ほとんどの場合、データとトランザクションログは別の物理ディスクに配置する必要があります。

前述の「管理」の項で説明したように、ASE デバイスは DISK INIT 文を使用して作成します。

少なくとも、現在、SQL Anywhere で使用されているデータ量やトランザクションログのサイズを保持できるよう、十分なデバイス領域を作成する必要があります。通常、両製品のストレージ構成は異なるので、アプリケーションの移行時には、さらに追加の領域が必要になります。

ステップ 2 : SQL Anywhere サーバーデータベースの移行

このステップの目的は、SQL Anywhere データベースを ASE サーバーに移行することです。

このステップの結果、アプリケーションをサポートできる状態で、全体が空のデータベース構造が作成されます。

ステップ 2.1 ASE データベースの作成

ステップ 1.3 で作成したデータデバイス上に、単一の ASE データベースを作成します。具体的には、sa_role 権限を持つユーザーとして ASE サーバーにログインした状態で、CREATE DATABASE 文を使用します。

ステップ 2.2 ASE データベースオプションの設定

sp_dboption システムプロシージャーを使用して設定された特定の ASE データベースオプションは、場合によっては、移行対象のアプリケーションをサポートするよう設定する必要があります。具体的には、次のとおりです。

- ☐ abort tran on log full は、データベースのログが容量の限界に達したときに（デフォルトの動作は、全トランザクションを保留)、ユーザートランザクションを中止する場合に設定する必要があります。
- ☐ allow nulls by default は、デフォルトでカラムの NULL 値を許容する（SQL Anywhere の場合のデフォルト設定）と DDL で想定している場合に設定する必要があります。
- ☐ BCP (Bulk Copy ：バルクコピー）機能によるデータのインポートを許可するには、select into/bulkcopy を設定する必要があります。移行時には、ほぼ確実に、このオプションを設定する必要があります。
- ☐ 対象が運用データベースではなく、開発データベースである場合、トランザクションログの（増分）バックアップは通常、実行されないので、trunc log on checkpoint を設定する必要があります。このオプションを設定せず、トランザクションログのバックアップが実行されない場合、ログは容量の限界に達します。

各オプションは、必要に応じて、sa_role 権限を持つユーザーが sp_dboption システムストアドプロシージャーまたは Sybase Central を使用して設定します。

ステップ 2.3 ASE データベースの所有権の設定

通常、ASE サーバーでは、アプリケーションによって使用されるデータベースを所有する「所有者」ログインが設定されています。これは、SQL Anywhere で、パスワードの設定がないグループをアプリケーションのテーブルの所有者に設定する方法に相当します。

所有者ログインを設定するには、sp_addlogin システムストアドプロシージャーを使用して、ログインを作成します。次に、sp_changedbowner ストアドプロシージャーを使用して、新しいアプリケーションデータベースの所有者を、作成したログインに変更します。使いやすさを考慮すると、ほとんどの事例で、所有者ログインには、アプリケーションデータベースをデフォルトのデータベースに設定することをおすすめします（作成時に設定されていない場合、sp_modifylogin を使用して設定)。

この設定で、所有者ログインがアプリケーションデータベースを使用すると、ユーザー dbo (database owner ：データベース所有者）であると認識されます。この状態を確認するには、所有者ログインとしてログインしているときに、アプリケーションデータベース内で、クエリ select user_name() を実行します。

ステップ 2.4 ASE サーバーのログインとデータベースユーザーの設定

アプリケーションを使用する各ユーザーには、ASE サーバーに対する専用のユーザーログインを設定する必要があります。さらに、アプリケーションデータベースへのアクセス権も必要になります。この場合、ログインにデータベースへのアクセス権を付与するには、データベース内に個別のユーザーとして作成するか、ログインに単一または複数のデータベースユーザーをマッピングできるようにします。別の選択肢として、Windows 統合ログインを使用する方法もあります。統合ログインを構成する場合の詳細については、『ASE 設定ガイド Windows 版』を参照してください。

ユーザーログインを作成するには、ユーザーごとに sp_addlogin システムストアドプロシージャーを実行し、デフォルトのデータベースに指定されているのがアプリケーションデータベースであることを確認します。

各ログインをユーザーとしてデータベースに追加するには、データベースの各ログインに対して sp_adduser システムストアドプロシージャーを実行します。この操作は、ステップ 2.3 で作成した dbo ユーザーが実行する必要があります。別の選択肢として、全ユーザーを単一のデータベースユーザー名にマッピングする方法もあります。具体的には、sp_addlogin および sp_adduser システムストアドプロシージャーを実行して、単一のユーザー名（およびそのログイン）を作成します。さらにログインごとに sp_addalias を実行して、作成済みの単一のユーザー名にマッピングします。

ログインごとに個別のデータベースユーザーを作成した場合、データベースグループを作成して各ユーザーをメンバーに設定することをおすすめします。具体的には、sp_addgroup および sp_changegroup システムストアドプロシージャーを実行します。作成したグループには、アプリケーションに応じて適切なアクセス権を割り当てることができます。

ステップ 3：データベース DDL（構造）の移行

このステップの目的は、アプリケーションの SQL Anywhere スキーマを ASE データベースに移行して、アプリケーションデータを格納できる状態にすることです。

ステップ 3.1 SQL Anywhere 固有の機能の確認と移行

データベース DDL を移行する最初のステップでは、SQL Anywhere 固有の機能を切り分けて、ASE または両製品で共通してサポートされている同等の機能で置き換えます。このステップにより、データベースを ASE 環境で作成する準備が整います。

このステップで有効なアプローチとしては、Sybase PowerDesigner などの CASE ツールを使用して、現在の SQL Anywhere のデータベーススキーマを物理データモデルとして抽出する方法が考えられます。このデータモデルに基づいて、ASE データベースを生成できます。

PowerDesigner などの CASE ツールによるアプローチを採用しないという想定では、別のアプローチとして、元の SQL Anywhere データベースの作成に使用した DDL ソースコードを使用する方法が

考えられます。このソースコードが存在しない場合は、SQL Anywhere の dbunload ユーティリティまたは Sybase Central を使用して、一連の SQL ファイルとして再作成します。

DDL を ASE で使用する場合に支障となる機能について、SQL Anywhere の DDL 全体を詳しく調べる必要があります。新しいバージョンの DDL を作成して、ASE データベースサーバーで許容される DDL で SQL Anywhere の機能を置き換える必要があります。特に重要なポイントとして、両データベースサーバーの同じデータセットがビューに含まれていることを確認しておく必要があります。

問題を引き起こす可能性がある SQL Anywhere サーバーの SQL 言語機能のリストについては、前述の「言語」の項を参照してください。

その他の情報については、後述の「SQL Anywhere から ASE への移植性チェックリスト」の項を参照してください。

ステップ 3.2 ASE データベースに対する DDL の実行

前のステップで作成した DDL を ASE サーバーに対して実行し、データベース構造を作成します。この操作は、アプリケーションデータベースの所有者であるサーバーログインを使用して実行する必要があります。

通常の方法としては、Sybase 提供の Interactive SQL クライアントで Sybase ASE プラグインを使用します。別の選択肢として、元の ASE isql クライアントを使用する方法もあります。isql クライアントは、個別の DDL スクリプトで実行する必要があります。

```
isql -S <servername> -U <dbo login> -P <password> -I <DDL file> -o <output file>
```

この例では、アプリケーションデータベースに対して DDL スクリプトを実行し、実行結果（正常処理を通知するメッセージ）をファイルに出力します。

重要なポイントは、テーブル、ビュー、およびインデックスを作成する場合、dbo ログインとして実行し、dbo データベースユーザーが各データベースオブジェクトの所有者になるようにする必要があるということです。

ステップ 4：管理手順とセキュリティ手順の確認

移行プロセスの次のステップでは、アプリケーションのサポートに対応できるよう、新しい ASE サーバーが適切に管理され、アプリケーションのニーズに応じて、適切なセキュリティ対策が講じられていることを確認します。

このステップの結果、管理用のツールや手順が有効に機能し、データベースの一連のセキュリティ権限によって、アプリケーションのセキュリティニーズを適切にサポートします。

ステップ 4.1 セキュリティモデルの移行

セキュリティモデルの移行に伴う作業として、移行した ASE アプリケーションに実装されているデータベースセキュリティが元の SQL Anywhere アプリケーションと同様に機能していることを確認します。

このステップには、さまざまな面で考慮すべき点があります。

- ☐ **RESOURCE 権限。** SQL Anywhere の RESOURCE 権限を持つユーザーについては、ASE アプリケーションデータベースの dbo に対するエイリアスを設定するか、dbo から適宜、CREATE TABLE/CREATE VIEW/CREATE RULE/CREATE DEFAULT/CREATE PROCEDURE 権限を付与する必要があります。
- ☐ **DBA 権限。** SQL Anywhere データベースの DBA 権限を持つ各ユーザーについては、アプリケーションデータベースの dbo にマッピングするか、実行できる必要がある操作の種類に応じて、sa_role および／または oper_role システムロールをユーザーのログインに付与する必要があります。
- ☐ **グループ。** 最もシンプルなケースとしては、SQL Anywhere データベースのグループにパスワードが関連付けられておらず、そのグループが階層化されていない場合 (グループが他のグループのメンバーになっていない状態) が挙げられます。この場合、各 SQL Anywhere グループをそのままアプリケーションデータベースの ASE グループとして実装するだけで済みます。(階層に関係なく) SQL Anywhere データベースのグループにパスワードが設定されている場合、各 SQL Anywhere グループを ASE ユーザーロールとして実装する必要があります。(パスワードの設定の有無に関係なく) SQL Anywhere データベースのグループが階層化されている場合、各 SQL Anywhere グループを ASE ユーザーロールとして実装する必要があります。ASE グループ／ロールの実装により、必要に応じて、SQL Anywhere ユーザーをグループのメンバーに設定するか、SQL Anywhere ユーザーにロールを付与します。グループ／ロールに適切なデータベース権限を付与します。各権限は、SQL Anywhere の権限と同じ権限または類似の権限です。

最後に、SQL Anywhere アプリケーションの任意のユーザーに対して個別に権限が割り当てられている場合、その権限は ASE アプリケーションデータベースの対応するデータベースユーザーにそのまま割り当てる必要があります。

ステップ 4.2 セキュリティスクリプトの移行

前のステップが完了した時点で、アプリケーションに用意されているセキュリティの設定および無効化のスクリプトは、前のステップで SQL Anywhere セキュリティモデルの移行に伴って作成した ASE セキュリティモデルを反映するため、SQL Anywhere の管理者が更新する必要があります。

ステップ 4.3 管理手順の確認

既存の SQL Anywhere アプリケーションには、多くの場合、さまざまな管理手順が定義されています。各手順については、ASE 環境に移行したアプリケーションでも適切かつ十分に対応できることを確認する必要があります。

具体的には、次の手順について考慮する必要があります。

- ☐ ベーステーブルのカラムおよびインデックス統計を最新の状態に維持する。
- ☐ ASE サーバー全体に対するバックアップおよびリカバリの戦略。
- ☐ データベースの破損が生じていないことを定期的に検証する。

ASE サーバーとそのアプリケーションが正常に機能するには、上記の手順が不可欠です。SQL Anywhere の管理者は、検討した手順の内容が実際の ASE 環境に適合していることを確認する必要があります。

ステップ 4.4 管理スクリプトの移行

前のステップの結果、既存の管理スクリプトに対する変更内容がほぼ確定します。既存の管理スクリプトが存在しない場合は、この段階で、サーバーの管理プロセスを部分的に自動化するスクリプトを作成しておくことをおすすめします。

変更が必要な内容としては、DUMP DATABASE および DUMP TRANSACTION 文によるバックアップ、DBCC によるデータベースの整合性チェック、UPDATE STATISTICS および UPDATE INDEX STATISTICS 文による統計の更新が挙げられます。

このステップの結果、新しい一連の管理スクリプトにより、アプリケーションの ASE サーバーに必要とされる定期的な管理タスクを自動化できます。

ステップ 5：アプリケーションデータの移行

プロセスのデータ移行ステップに伴い、SQL Anywhere からデータを抽出し、新しく作成した ASE データベースにインポートします。

ステップ 5.1 SQL Anywhere データのエクスポート

SQL Anywhere データベースから、アプリケーションデータ全体をエクスポートする必要があります。前述のように、SQL Anywhere データベースからデータをアンロードする方法として、次のように複数の選択肢が用意されています。

- ☐ UNLOAD TABLE または UNLOAD SELECT 文を使用して、区切り形式のローを含む ASCII ファイルを作成します。
- ☐ ASE の BCP ユーティリティを使用して、SQL Anywhere データを ASCII またはバイナリ BCP 形式でアンロードします。
- ☐ SQL Anywhere の dbunload ユーティリティを使用して、ASCII ファイルをテーブルごとに 1 つ、サーバーマシンのディレクトリに作成します。
- ☐ Sybase Central で、適切な SQL Anywhere データベースを選択してから、[File] > [Unload Database] メニューオプションを使用します。
- ☐ SQL Anywhere でプロキシテーブルを使用して、データの処理を ASE サーバーに移行します（INSERT...FROM SELECT 文を使用）。
- ☐ ASE から SQL Anywhere へのプロキシテーブルを使用して、INSERT...FROM SELECT 文により、データを挿入します。

UNLOAD を使用する方法に問題があるとすれば、特殊文字（特に文字列カラムとバイナリカラムに含まれる引用符や改行文字）が正常に解釈されない可能性があります。このため、フィールド間に適切な区切り文字を使用して、SQL Anywhere テーブルを ASCII テキストファイルにアンロードする必要があります。区切り文字としては一般に、タブ文字やコンマが使用されますが、他にも使用できる文字があります。

この項では、UNLOAD TABLE 文を使用すると想定します。たとえば、SQL Anywhere データベースのアプリケーションテーブルごとに、次の文を使用します。

```
UNLOAD TABLE <tablename>
TO D:¥MIGRATION¥DATADIR¥<tablename>.dat
DELIMITED BY ','
QUOTES OFF
```

このステップの結果、一連の .dat ファイルがディレクトリに作成されます。このファイルには、各 SQL Anywhere アプリケーションデータベーステーブルから取得されたデータが含まれています。

ステップ 5.2 ASE へのデータのインポート

事前にエクスポートした SQL Anywhere ファイルを ASE にインポートするプロセスでは、BCP コマンドラインユーティリティを使用します。BCP コマンドは、テーブル、データファイル、およびユーザー ID 情報を指定して、各データベーステーブルに対して実行する必要があります。ユーザー情報として指定するのは、ASE アプリケーションデータベースの dbo ユーザーの情報です。

たとえば、データをテキストモードでロードするには、次の文を実行します。

```
bcp [<database name>.][<owner name>.]<tablename> in <tablename>.dat
-c
-t','
-r '¥r¥n'
-v
-S <servername> -U <DBOuser> -P<password>
```

このプロセスは、オペレーティングシステムのコマンドスクリプトを使用すると簡単に自動化できます。具体的には、ディレクトリ内のファイルのリストに基づいて、データベースの全テーブルをロードします。

重要なポイントとして、ロード対象のテーブルからインデックスとトリガーが削除されている状態で、ASE の bcp コマンドを使用してデータをインポートすると、処理が何倍も高速になるということが挙げられます。トリガーとインデックスは、ロードが完了した後で再作成できます。参照整合性制約を一時的に削除しても、ロードプロセスの処理効率は向上します。

データ移行時に BCP を使用する場合の詳細については、付録「BCP によるデータ移行」を参照してください。

前述のように（「データのインポートとエクスポート」の項を参照）、バイナリデータ値はゼロパディングによって問題を引き起こす可能性があります。アンロード／BCP に起因する問題が検出された場合、別のアプローチとして、ASE サーバーに対する SQL Anywhere のプロキシテーブルを使用する方法が考えられます。具体的には、SQL INSERT...FROM SELECT 文を使用して、サーバー間でデータを直接コピーする操作を可能にします。

ステップ 6：アプリケーションプログラム（クエリとインターフェイス）の移行

このステップまで、移行プロセスでは、データベースサーバーの構成、データベーススキーマ、およびアプリケーションのデータを重点的に扱ってきました。このステップでは、アプリケーション自体の移行を扱います。具体的には、(1)「ASE プログラミングインターフェイスの構成とテスト」、(2)「新しい ASE データベースに対してアプリケーションの SQL 文が正常に機能することの確認」を実行します。

ステップ 6.1 SQL Anywhere 固有の機能に対応する SQL 文の確認

アプリケーションプログラムの移行プロセスで、最初に最も重要なステップを実行します。具体的には、SQL Anywhere データベースに対して使用していた SQL 文が ASE データベースに対しても有効に機能するように設定します。このステップには、2 つのポイントがあります。最初のポイントとして、SQL 文をコンパイルして実行したときに、確実にエラーが発生しないようにする必要があります。次に重要なポイントとして、SQL 文を検証して、両データベースに対して同じセマンティックが保持されていることを確認します。

このステップでは通常、アプリケーションに含まれている各 SQL 文とともに、SQL ストアドプロシージャー、関数、およびトリガーの定義を最初に手動で確認し、移植時に発生する可能性がある問題を明らかにして、回避策を講じます。問題を引き起こす可能性がある SQL Anywhere 機能のリストについては、前述の「言語」の項を参照してください。ただし、このチェックリストはあらゆるケースを網羅したものではなく、プロセスを進める上での参考情報として活用してください。

ストアドプロシージャーなど、サーバー上に配置された SQL オブジェクトについては、Sybase Central の [Open as Watcom SQL] および [Open as Transact-SQL] 機能を使用すると、Watcom ダイアレクトの SQL を Transact-SQL に変換する操作をスムーズに進めることができます。

SQL Anywhere 固有の機能を使用する SQL 文については、ASE データベースサーバーでも動作するように文を修正する必要があります。

ステップ 6.2 ASE に対する SQL 文のテスト

このステップでは、アプリケーションの SQL 文をそれぞれ ASE サーバーに対してテストし、ASE 環境でも SQL Anywhere 環境の対応するクエリとまったく同様に機能することを確認します。このステップを実行する場合に必要な作業量は未確定ですが、少なく見積もることは避けてください。テスト対象のプログラムやデータベースインスタンスが限定される場合、ロッキングや同時実行制御といったセマンティックの差異は判別が難しいことがあります。

ステップ 6.3 クライアントプログラムの再構築

次のステップでは、クライアントプログラムをそれぞれ再構築して、ASE データベースサーバーに接続できる状態にします。埋め込み SQL プログラムの場合は、ASE プレコンパイラーからの実行や、コンパイル後に、ASE サーバーに接続できることの検証といった作業を伴います。

全般的に、ODBC および JDBC プログラムは、SQL Anywhere 固有の機能に依存していなければ、再構築する必要はありません。多くの場合、再構築が不要なアプリケーションについては、対応する ASE ドライバー（ASE ODBC ドライバーや jConnect）をそのまま使用して、修正せずに実行できます。ただし、jConnect と SQL Anywhere JDBC ドライバーでは、ワイヤーレベルプロトコルに若干の差異があります。この結果、クライアント間で転送されるデータ量に影響が生じて、さらにロッキングや同時実行制御にも影響が生じる可能性があります。

文字列データ型、バイナリ文字列データ型、および日付／時刻データ型のサポートは、ASE と SQL Anywhere の両製品で異なります。したがって、このようなデータ型を使用するアプリケーションコードを分析する際には、注意が必要です。

ステップ *6.4* クライアントプログラムのテスト

このステップでは、ASE 準拠のクエリと ASE 接続が設定されているクライアントプログラムをテストして、ASE サーバーに対する実行時に問題が発生しないことを確認します。最後のステップの結果、ASE サーバーでホスティングされているアプリケーションの機能にまったく問題がないことが確認されます。

ステップ *7*：移行の検証

移行プロセスの最後のステップでは、精密なシステム整合性テストを実行して、移行を検証します。理想としては、一連の統合テスト、システムテスト、およびユーザー受け入れテストを自動化する手段があると、移行の検証が容易になります。このような手段がない場合は、別のテストプロセスを実行して、移行全般に問題がないことを確認する必要があります。

テストには、次の検証が含まれます。

- ☐ ユーザーインターフェイスのトランザクション
- ☐ バッチ処理
- ☐ 管理手順（バックアップなど）
- ☐ 障害復旧
- ☐ アプリケーションパフォーマンス

上記の検証が完了した時点で、アプリケーションは SQL Anywhere から ASE に問題なく移行したと判断されます。

まとめ

このホワイトペーパーの目的は、SQL Anywhere アプリケーションを ASE に移行する場合に陥る可能性がある落とし穴を明らかにして、移行プロセスのスムーズな実施を支援することでした。これまで説明してきたように、SQL Anywhere と ASE の両製品には多くの共通点があります。ただし、前述のように、開発の経緯やターゲット市場の違いに起因して、両製品には若干の差異があります。さらに、SQL の互換性についても、構文とセマンティックの両面で、さまざまな問題があり、アプリケーションの移行プロセスは複雑になっています。

両製品の移行プロセスとして、次の手順が挙げられます。

- ☐ 適切な ASE サーバーの作成および構成
- ☐ アプリケーションのデータベーススキーマの移行
- ☐ アプリケーションのデータの移行
- ☐ 管理手順とセキュリティ手順が新しい環境に適合することの確認
- ☐ アプリケーションの SQL 文が ASE 環境で機能することの確認
- ☐ プログラミングインターフェイスの移行に問題がないことの確認
- ☐ 移行したシステムに対するテストで、元の SQL Anywhere システムとの互換性に問題がないことの確認

SQL Anywhere から ASE への移植性チェックリスト

この付録では、SQL Anywhere から ASE へ移植可能なアプリケーションを計画しなければならない場合、あるいはマイグレーションの際に移植性を査定する必要がある場合に役に立つチェックリストを用意しました。このチェックリストでは、ASE に容易に移植できるアプリケーションにするために評価すべきポイントに言及しています。

□ SQL Anywhere と ASE の間で共通のデータタイプを可能な限り、信用してください。BIGDATETIME のようなデータタイプのいくつかでは、ASE のタイプ宣言は SQL Anywhere ではサポートされていません。しかしながら、セマンティクスは、SQL Anywhere の他のタイプ相当です。その他 BIT VARYING のようなタイプでは、ASE では異なるデータタイプを使用し、それに応じてアプリケーションを修正する必要があります。

SQL Anywhere のデータタイプ	ASE の相当のデータタイプ	備考
TIME	BIGTIME	1
DATE	DATE	
DATETIME	DATETIME	2
TIMESTAMP	DATETIME	2
TIMESTAMP	BIGDATETIME	2
SMALLDATETIME	SMALLDATETIME	
MONEY	MONEY	
SMALLMONEY	SMALLMONEY	
INTEGER	INTEGER	
UNSIGNED INTEGER	UNSIGNED INTEGER	
SMALLINT	SMALLINT	
UNSIGNED SMALLINT	UNSIGNED SMALLINT	
TINYINT	TINYINT	
UNSIGNED TINYINT	UNSIGNED TINYINT	
BIGINT	BIGINT	
UNSIGNED BIGINT	UNSIGNED BIGINT	
UNIQUEIDENTIFIER	VARCHAR(32) or VARBINARY(16)	
DECIMAL(p,s)	DECIMAL(p,s)	
NUMERIC(p,s)	NUMERIC(p,s)	
FLOAT	FLOAT(p)	
REAL	FLOAT(p)	
DOUBLE	DOUBLE	

TEXT	TEXT	
CHAR	CHAR	
VARCHAR	VARCHAR	
LONG VARCHAR	TEXT	
NCHAR	NCHAR	
NVARCHAR	NVARCHAR	
LONG NVARCHAR	UNITEXT	
BIT	BIT	
BIT VARYING	INTEGER or VARBINARY	3
VARBINARY	VARBINARY	
LONG BINARY	IMAGE	
BINARY	BINARY	
IMAGE	IMAGE	

注：

1. 精度が問題にならない場合は、SQL Anywhere で TIME データ型を使用して、TIME カラムを直接 ASE に移行します。精度が問題になる場合は、移行時に、ASE で BIGTIME データ型を使用します。
2. 精度が問題にならない場合は、SQL Anywhere で DATETIME データ型を使用して、DATETIME カラムを直接 ASE に移行します。精度が問題になる場合は、移行時に、ASE で BIGDATETIME データ型を使用します。
3. BIT VARYING は、ASE ではサポートされていません。ASE では別のデータ型を使用して、必要に応じて、表現やアプリケーションプログラムのロジックを修正します。

- ☐ ASE では、識別子の大文字小文字が区別されるので、可能なかぎり、データベースオブジェクトに対する標準的な命名規則を設定して、実際のデータベースオブジェクトに適用します。有効なアプローチとしては、SQL 文に含まれるキーワードと識別子には必ず大文字を使用するという方法が考えられます。ただし、サードパーティ製の各種ツールは、必ずしも特定の命名規則に準拠しているわけではないため、各自のデータベースに適した命名規則を十分に検証した上で、設定してください。
- ☐ ASE への移植性を最大限に高めるには、大文字小文字を区別する照合と空白パディングを有効にして SQL Anywhere データベースを作成します。
- ☐ NUMERIC および DECIMAL 型の桁数と精度については、両製品のデフォルト値は異なるので、デフォルト値をそのまま使用しないでください。移植を容易にするには、SQL Anywhere で NUMERIC (DECIMAL) 型の PRECISION および SCALE を明示的に指定します。
- ☐ SQL Anywhere では、CREATE DOMAIN を使用せずに、対応するシステムプロシージャー sp_addtype を使用してください。
- ☐ SQL Anywhere では、ASE DATEFORMAT オプションの ASE のデフォルト設定に合わせて、date_order オプションを MDY に設定します。
- ☐ SQL Anywhere では、日付・文字列変換の ASE のデフォルト設定に合わせて、date_format option オプションを設定します (Mmm dd yyyy など)。
- ☐ SQL Anywhere では、タイムスタンプ・文字列変換の ASE のデフォルト設定に合わせて、timestamp_format オプションを設定します (yyyy-mm-dd hh:mm:ss.zzzzzz)。ASE データベースで DATETIME 型を使用する予定がある場合は、デフォルト設定 (yyyy-mm-dd hh:mm:ss.zzz) を使用します。
- ☐ SQL Anywhere では、時刻・文字列変換の ASE のデフォルト設定に合わせて、time_format オプションを設定します (hh:mm:ss.zzzzzzAA)。
- ☐ DEFAULT AUTOINCREMENT ではなく、IDENTITY カラムプロパティを使用します。
- ☐ 連鎖 RI (Referential Integrity ：参照整合性) 制約の使用は避けてください。ASE では、トリガーを使用して、この機能を実現します。専用の RI 実装を作成する場合は、Sybase PowerDesigner など、CASE ツールの使用を検討してください。
- ☐ テンポラリテーブルを作成するには、DECLARE LOCAL TEMPORARY TABLE 文ではなく、暗黙的な #tablename 機能を使用します。ASE では、CREATE GLOBAL TEMPORARY TABLE 文はサポートされていないことに注意してください。

- ☐ DECLARE CURSOR 文で FOR UPDATE BY LOCK を使用しないでください。
- ☐ OPEN CURSOR 文で WITH HOLD および ISOLATION LEVEL 句を使用しないでください。
- ☐ DECLARE カーソル処理文で UNIQUE、SCROLL、NOSCROLL、DYNAMIC SCROLL、および INSENSITIVE 修飾子を使用しないでください。アプリケーションで SQL Anywhere SCROLL カーソル（KEYSET-DRIVEN カーソル）を使用している場合、テーブル定義を修正して、Transact-SQL TIMESTAMP データ型を含める必要があります。この修正により、Transact-SQL の TSEQUAL 関数を使用して、基本となるベーステーブルローが変更されたかどうかを判別できるようになります。
- ☐ ASE ESQL/C プログラムでアレイフェッチを実行する場合、ARRAY キーワードは不要です（ASE の ESQL プレコンパイラーで暗黙的に処理されます)。
- ☐ EXECUTE IMMEDIATE 文の使用は控えてください。ASE では、EXECUTE 文を使用します。
- ☐ CALL 文の使用は控えてください。ASE では、EXECUTE 文を使用します。
- ☐ BEGIN ATOMIC 文を使用しないでください。代わりに、ASE のトランザクション文を使用します。
- ☐ RELEASE SAVEPOINT 文の使用は控えてください。ASE では、トランザクションの完了まで、セーブポイントはそのまま保持されます。
- ☐ CREATE VARIABLE 文の代わりに、DECLARE 文を使用して、ローカル SQL 変数を作成します。
- ☐ SQL プロシージャー、関数、およびトリガーに含まれている SQL 変数の名前が必ず @ 文字で始まっていることを確認してください。
- ☐ SQL Anywhere との互換性を最大限に高めるため、Transact-SQL の SELECT 文ではなく、SET 文を使用して、SQL 変数に値を割り当てます。
- ☐ トリガーとインデックスの名前がデータベース全体で一意であることを確認します。
- ☐ Watcom SQL ダイアレクトバージョンの CREATE PROCEDURE、CREATE FUNCTION、および CREATE TRIGGER を使用しないでください。代わりに、各文の Transact-SQL 版を使用します。INSTEAD OF トリガーは、SQL Anywhere の Transact-SQL ダイアレクトではサポートされていないことに注意してください。
- ☐ row トリガーまたは before 文トリガーを使用しないでください。ASE では、after statement トリガーのみがサポートされています。
- ☐ Watcom SQL のフロー制御文および例外処理文は ASE ではサポートされていないので、使用は控えてください。具体的には、FOR 文の使用は避けてください（ASE ではサポートされていません)。
- ☐ トランザクションモード（chained または unchained）を決定します。SQL Anywhere ではデフォルトモードが chained に設定されているのに対して、ASE ではデフォルトモードが unchained に設定されています。このため、トランザクションモードは明示的に設定する必要があります。
- ☐ 移植性を最大限に高めるため、全データベースオブジェクトの所有者である dbo グループで、SQL Anywhere のセキュリティモデルを使用します。
- ☐ データベース管理スクリプトで、両データベースに用意されているシステムストアドプロシージャー（sp_addgroup、sp_addlogin、sp_addmessage、sp_addtype、sp_adduser、sp_changegroup、sp_dboption、sp_dropgroup、sp_droplogin、sp_dropmessage、

sp_droptype、sp_dropuser、sp_getmessage、sp_helptext、および sp_password) を使用します。この対応により、上記のストアドプロシージャーが管理スクリプトで使用されている場合、両環境でスクリプトを移植しやすくなります。

- □ デフォルトの日付については、SQL Anywhere では現在の日付が使用されるのに対して、ASE ではデータ型に応じて 1970 年 1 月 1 日または 0001 年 1 月 1 日が使用されます。このため、デフォルトの日付をそのまま使用しないでください。
- □ 問題を引き起こす可能性があるキーワードの使用は避けてください。下記のキーワードの表を参照してください。上位互換性を確保するため、SQL/2008 標準で規定されているキーワードについては、現在、SQL Anywhere および ASE でサポートされていない場合でも、使用は控えてください。
- □ バッチ処理で SQL Anywhere の文の終止符に ; を使用しないでください。これは Watcom SQL ダイアレクトの表記です。代わりに、Transact-SQL ダイアレクトの BEGIN/END を使用してください。
- □ SQL Anywhere の コメントインジケーター // を使用しないでください。代わりに、-- および /* ... */ を使用します。
- □ SQL Anywhere には、SELECT column INTO @variable という構文が用意されています。ASE への移植のしやすさを考慮して、代わりに Transact-SQL 版の SELECT @variable = column を使用するか、両製品でサポートされている SET 文を使用します。
- □ CROSS JOIN、JOIN、KEY JOIN、LATERAL、NATURAL JOIN、APPLY、OUTER APPLY、および FULL OUTER JOIN 句は使用しないでください。Transact-SQL の外部ジョインの使用は控えてください。ASE 15.5 では、ANSI の外部ジョイン構文がサポートされています。
- □ SQL Anywhere 独自の日時関数 (TODAY()、NOW()、HOUR()、MINUTE()、QUARTER() など) の使用は避けてください。代わりに、ASE 互換のシステム関数を使用します。
- □ SQL Anywhere 独自のデータ型変換関数 (CAST()、STRING()、CASE()、DATETIME() など) の使用は避けてください。代わりに、両製品でサポートされている CONVERT() 関数を使用します。
- □ SQL Anywhere の関数 ARGN()、COALESCE()、IFNULL()、ISNULL()、ROWID()、NUMBER() と演算子 || の使用は避けてください (文字列の連結には、代わりに + を使用)。
- □ SQL Anywhere の関数 REMAINDER()、MOD()、TRUNCATE()、INSERTSTR()、LCASE()、LOCATE()、PLANS()、SIMILAR()、LOCATE()、STRING()、TRACEBACK()、TRIM()、および UCASE() の使用は避けてください。
- □ WHERE 句での推定ヒントの使用は控えてください (ASE ではサポートされていません)。
- □ 比較述部で暗黙的な型変換を使用する場合、注意が必要です。型階層と暗黙的な型変換の規則は、両製品で異なります。
- □ REGEXP や IS [NOT] DISTINCT FROM など、ASE でサポートされていない述部の使用は避けてください。
- □ ASE でサポートされていない SQL Anywhere の言語機能 (テーブル関数、MERGE 文、マテリアライズドビューなど) に依存することは避けてください。

重要なポイントとして、文のデフォルト設定や複合的なセマンティックなど、下位レベルの詳細には、両製品で若干異なる部分があります。

SQL Anywhere のサーバーオプション

SQL Anywhere では、サーバーの動作を設定するためにコマンドラインレベルのスイッチと、sa_server_option システムプロシージャーの両方をサーバーレベルで用意しています。設定可能なサーバーオプションのリストは、以下の表のとおりです。これらのうち一部は、ASE のサーバーオプションに対応しています。

SQL Anywhere のサーバーオプション	使用可能な値	SQL Anywhere のデフォルト値	備考
AutoMultiProgrammingLevel	YES, NO	YES	ASE のスレッドは設定オプション (ワーカープロセス数)。ASE パフォーマンス & チューニング シリーズ: 基本 と ASE システム管理ガイドを参照
AutoMultiProgrammingLevelStatistics	YES, NO	NO	ASE では非適用.
CacheSizingStatistics	YES, NO	NO	ASE のキャッシュサイズデータは、sp_cacheconfig システムプロシージャーを使用して表示。また、起動時に ASE のサーバーエラーログ内に表示
CollectStatistics	YES, NO	YES	ASE では非適用
ConnsDisabled	YES, NO	NO	ASE の sp_locklogin プロシージャーを使用
ConnsDisabledForDB	YES, NO	NO	ASE の sp_locklogin プロシージャーを使用
ConsoleLogFile	ファイル名		サーバー起動時に ASE のエラーログを特定する必要がある
ConsoleLogMaxSize	ファイルサイズ、byte		ASE では、エラーログのサイズは、マニュアルで管理する必要がある
CurrentMultiProgrammingLevel	Integer	20	ASE のスレッドは設定オプション (ワーカープロセス数)。ASE パフォーマンス & チューニング シリーズ: 基本 と ASE システム管理ガイドを参照
DatabaseCleaner	ON, OFF	ON	ASE では非適用
DeadlockLogging	ON, OFF, RESET, CLEAR	OFF	ASE の設定オプション「print deadlock information」を使用して、デッドロックメタデータをサーバーエラーログにアウトプット。sp_sysmon プロシージャーを使用して、デッドロックの発生を監視することも可能
DebuggingInformation	YES, NO	NO	複数の sp_configure オプション設定に相当。ASE システム管理のマニュアルを参照
DropBadStatistics	YES, NO	YES	ASE の統計管理は、CREATE STATISTICS と CREATE INDEX STATISTICS 文経由で行われる

DropUnusedStatistics	YES, NO	YES	ASE の統計管理は、CREATE STATISTICS と CREATE INDEX STATISTICS 文経由で行われる
IdleTimeout	Integer、分	240	ASE の統計管理は、CREATE STATISTICS と CREATE INDEX STATISTICS 文経由で行われる
IPAddressMonitorPeriod	Integer、秒	ポータブルデバイスは 120、その他は 0	ASE では非適用
LivenessTimeout	Integer、秒	120	ASE リファレンス・マニュアル: プロシージャー内の、sp_serveroption システムプロシージャーの ”timeout” パラメーター参照
MaxMultiProgrammingLevel	Integer	CurrentMultiProgrammingLevel 値の 4 倍	ASE のスレッドは設定オプション (ワーカープロセス数)。ASE パフォーマンス & チューニング シリーズ: Basics and the ASE システム管理ガイドを参照
MessageCategoryLimit	Integer	400	ASE では非適用
MinMultiProgrammingLevel	Integer	-gtc サーバーオプションの値の最小と、コンピューターの論理 CPU の数	ASE のスレッドは設定オプション (ワーカープロセス数)。ASE パフォーマンス & チューニング シリーズ: 基本 と ASE システム管理ガイドを参照
OptionWatchAction	MESSAGE, ERROR	MESSAGE	ASE では非適用
OptionWatchList	データベースオプションのカンマで区切られたリスト		ASE では非適用
ProcedureProfiling	YES, NO, RESET, CLEAR	NO	ASE では非適用
ProfileFilterConn	connection-id		ASE では非適用
ProfileFilterUser	user-id		ASE では非適用
QuittingTime	valid date and time		ASE では非適用
RememberLastPlan	YES, NO	NO	ASE のアプリケーショントレーシング機能を参照。sp_showplan() を使用して、ある接続の現在実行している文のプランを見ることが可能
RememberLastStatement	YES, NO	NO	ASE のアプリケーショントレーシング機能を参照。接続のために オプション “max SQL text monitored” の設定と sp_configure()を使用して SQL テキストを保存可能
RequestFilterConn	connection-id, -1		ASE のアプリケーショントレーシング機能参照

RequestFilterDB	database-id, -1		ASE のアプリケーショントレーシング機能参照
RequestLogFile	filename		ASE のアプリケーショントレーシング機能参照
RequestLogging	SQL, HOSTVARS, PLAN, PROCEDURES, TRIGGERS, OTHER, BLOCKS, REPLACE, ALL, YES, NONE, NO	NONE	ASE のアプリケーショントレーシング機能参照
RequestLogMaxSize	ファイルサイズ、bytes		ASE のアプリケーショントレーシング機能参照
RequestLogNumFiles	Integer		ASE のアプリケーショントレーシング機能参照
RequestTiming	YES, NO	NO	ASE のアプリケーショントレーシング機能参照
SecureFeatures	feature-list		ASE では非適用。ASE のセキュリティーモデルは SQL Anywhere のものと実質大きく異なる。
StatisticsCleaner	ON, OFF	ON	ASE では非適用
WebClientLogFile	ファイル名		ASE では非適用
WebClientLogging	ON, OFF	OFF	ASE では非適用

SQL Anywhere のオプション

SQL Anywhere の動作の代替方法としては、主に接続、ユーザー、または PUBLIC レベルでオプションをセットすることのできる SET OPTION 文経由になります。SQL Anywhere 12.0.1 では、 129 のオプションをサポートしています。そのうちの 16 が ASE の Transact-SQL 互換性に関連しています。

SQL Anywhere のオプション	SQL Anywhere のデフォルト値	ASE の相当オプションまたは設定	備考
allow_nulls_by_default	ON	“allow_nulls_by_default” は、sp_dboption 経由で設定	ASE のデフォルト設定は OFF
allow_read_client_file	OFF		ASE ではサポート対象外
allow_snapshot_isolation	OFF		ASE ではサポート対象外
allow_write_client_file	OFF		ASE ではサポート対象外
ansi_blanks	OFF		ASE trailing blank を格納せず、全ての fixed-length カラムを blankpadded として扱う
ansi_close_cursors_on_rollback	OFF		
ansi_permissions	ON	ansi_permissions オプション	SQL Anywhere のデフォルト設定は、ON。一方 ASE のデフォルト設定は OFF
ansi_substring	ON		ASE では、SQL Anywhere の SUBSTRING 関数の拡張はサポート対象外
ansi_update_constraints	CURSORS		ASE では非適用.
ansinull	ON	ansinull オプション	ASE のデフォルトは OFF
auditing	OFF		ASE システム管理ガイド内の監査機能参照
auditing_options	4294967295		ASE システム管理ガイド内の監査機能参照
background_priority	OFF		特定のセッションまたはアプリケーションの実行プライオリティーを設定するは、sp_setpsexe, sp_addexeclass, sp_bindexeclass を使用
blocking	ON	lock オプション	
blocking_others_timeout	0		ASE では非適用.
blocking_timeout	0	lock オプション	ASE では、ロックオプションは、最大待機時間を設定可能

chained	ON	chained オプション	
checkpoint_time	60	sp_configure「recovery interval in minutes」オプション	
cis_option	0		ASE では非適用
cis_rowset_size	50		ASE では非適用
close_on_endtrans	ON	close on endtran オプション	
collect_statistics_on_dml_updates	ON		ASE では非適用
conn_auditing	ON		ASE システム管理ガイド内の監査機能参照
connection_authentication	'' (empty string)		ASE では非適用
continue_after_raiserror	ON		デフォルトの ON が ASE の動作に相当
conversion_error	ON	arithignore オプション	
database_authentication	'' (empty string)		ASE では非適用
date_format	YYYY-MM-DD		ASE の CONVERT() 関数のデフォルトのスタイルは、100
date_order	YMD	dateformat オプション	ASE のデフォルトは、US English データベースのための MDY
debug_messages	OFF		ASE では非適用
dedicated_task	OFF		ASE では非適用
default_dbspace	'' (empty string)		ASE では非適用
default_timestamp_increment	1		ASE ではサポート対象外
delayed_commit_timeout	500		ASE では非適用
delayed_commits	OFF	delayed_commit オプション	セッションオプションは、sp_dboption 経由で設定されるデータベースレベルの “delayed commit” オプションをオーバーライド
divide_by_zero_error	ON	ASE の arithabort arith_overflow オプションを使用	
escape_character	ON		このオプションは留保
exclude_operators	'' (empty string)		ASE では特定のオペレーターのための特定のオプションとともに trace flags を使用
extended_join_syntax	ON		ASE では非適用
fire_triggers	ON		ASE では非適用
first_day_of_week	7	datefirst オプション	

for_xml_null_treatment	Omit		ASE では非適用
force_view_creation	OFF		このオプションは留保
global_database_id	2147483647		ASE では非適用
http_connection_pool_basesize	10		「ASE Web Services ユーザガイド」の設定のセクションを参照
http_connection_pool_timeout	60		「ASE Web Services ユーザガイド」の設定のセクションを参照
http_session_timeout	30		「ASE Web Services ユーザガイド」の設定のセクションを参照
integrated_server_name	'' (empty string)		ASE リファレンス・マニュアル: プロシージャの ASE 設定ガイド Windows NT 版 と sp_loginconfig システムプロシージャーを参照
isolation_level	0	トランザクション分離レベルオプション	
java_class_path	'' (empty string)		ASE では、CLASSPATH 環境変数または installjava ユーティリティを使用
java_location	'' (empty string)		この機能は廃止
java_main_userid	'' (empty string)		この機能は廃止
java_vm_options	'' (empty string)		ASE リファレンス・マニュアル: プロシージャーの sp_pciconfig プロシージャーを参照
lock_rejected_rows	OFF		このオプションは、ASE のロックの実装方法のために並行のダイアレクトが ASE にはない
log_deadlocks	OFF	sp_configure システムプロシージャーの “print deadlock information” オプションを使用	
login_mode	STANDARD		For Windows 統合ログインについては、ASE 設定ガイド Windows 版を参照。Kerberos については、ASE システム管理ガイドを参照
login_procedure	NULL		login_procedure をパスワード管理に使用している場合は、ASE は sp_configure と sp_passwordpolicy システムプロシージャー経由でビルトインのパスワード設定をサポート
materialized_view_optimization	STALE		ASE では非適用.

max_client_statements_cached	10		ASE では非適用.
max_cursor_count	50		ASE では非適用
max_hash_size	10		SQL Anywhere でのこのオプションのサポートは終了
max_plans_cached	20		ASE は、FORCE RECOMPILE 文が使用されなければ全てのプロシージャーのアクセスプランをデフォルトでキャッシュ。サーバーにわたるステートメントキャッシュがクライアントに使用される。ASE システム管理ガイドと sp_configure システムプロシージャーのステートメントキャッシュサイズ設定を参照
max_priority	NORMAL		ASE の優先順位は、「execution classes」と呼ばれる。ASE パフォーマンス & チューニングガイド と、sp_bindexeclass と sp_showexeclass システムプロシージャーを参照
max_query_tasks	0	parallel_degree オプション	ASE では、最大マルチプログラミングレベルは、sp_configure システムプロシージャーの「ワーカープロセス数」オプション経由で設定
max_recursive_iterations	100		ASE では非適用
max_statement_count	50		ASE では非適用
max_temp_space	0	sp_add_resource_limit() を使用	ASE では、フリースペースの管理は、スレッドホールドの使用経由
min_password_length	0		sp_passwordpolicy システムプロシージャー参照
nearest_century	50		ASE では、その世紀の日付を決定するため Year threshold は 50 に固定
non_keywords	'' (empty string)		ASE では非適用
odbc_describe_binary_as_varbinary	OFF		ASE の ODBC ドライバーは、固定と可変長のバイナリータイプを区別
odbc_distinguish_char_and_varchar	OFF		ASE の ODBC ドライバーは、固定と可変長の文字列のタイプを区別
oem_string	'' (empty string)		ASE では非適用
on_charset_conversion_failure	IGNORE		ASE ではサポート対象外

on_tsql_error	CONDITIONAL (or CONTINUE for jConnect connections)		ASE の Transact-SQL プロシージャーには、CONTINUE が最も近い
optimization_goal	ALL-ROWS	plan optgoal オプション	
optimization_level	9	plan opttimeoutlimit オプション	
optmization_workload	MIXED	plan optgoal オプション	
pinned_cursor_percent_of_cache	10		ASE システム管理ガイド と sp_configure システムプロシージャーのステートメントキャッシュサイズの設定参照
post_login_procedure	'' (empty string)		ASE では非適用
precision	30		ASE のデフォルトは 18
prefetch	ON		ASE では非適用
preserve_source_format	ON		ASE では非適用
prevent_article_pkey_update	ON		ASE では非適用
priority	NORMAL		ASE の優先順位は、「execution classes」と呼ばれる。ASE パフォーマンス & チューニングガイド、および sp_bindexeclass と sp_showexeclass system プロシージャーを参照
progress_messages	OFF		ASE では非適用
query_mem_timeout	-1		ASE では非適用
quoted_identifier	ON	quoted_identifier オプション	ASE では、デフォルトの設定は OFF
read_past_deleted	ON		ASE ではサポート対象外
recovery_time	2	"recovery interval in minutes" option of sp_configure	
remote_idle_timeout	15		ASE では非適用
replicate_all	OFF		SQL Anywhere におけるこのオプションのサポートは終了
request_timeout	0	sp_add_resource_limit() を使用	ASE システム管理ガイドのリソース制限の章参照
reserved_keywords	'' (empty string)		ASE では非適用
return_date_time_as_string	OFF		このオプションは、TIMESTAMP 値のミリ秒単位の精度をサポートする古い TDS クライアントのみ必要

rollback_on_deadlock	ON		ASE ではサポート対象外
row_counts	OFF		ASE ではサポート対象外
scale	6		ASE のデフォルトは、0 (zero)
secure_feature_key	'' (empty string)		ASE では非適用
sort_collation	NULL		ASE では非適用
sql_flagger_error_level	OFF	fipsflagger	FIPSFLAGGER は、ASE のエントリーレベル SQL/1992 の TSQL 拡張に対して警告を発行
sql_flagger_warning_level	OFF	fipsflagger	FIPSFLAGGER は、ASE のエントリーレベル SQL/1992 の TSQL 拡張に対して警告を発行
st_geometry_asbinary_format	WKB		ASE では非適用
st_geometry_astext_format	WKT		ASE では非適用
st_geometry_asxml_format	GML		ASE では非適用
st_geometry_describe_type	CHAR		ASE では非適用
st_geometry_interpolation	INTERPOLATE		ASE では非適用
st_geometry_on_invalid	ERROR		ASE では非適用
string_rtruncation	ON	string_rtruncation オプション	
subsume_row_locks	ON		ASE では非適用
suppress_tds_debugging	OFF		ASE では非適用
synchronize_mirror_on_commit	OFF		ASE では非適用
tds_empty_string_is_null	OFF		ASE では非適用 – ASE は empty strings を格納しない
temp_space_limit_check	ON	sp_add_resource_limit() を使用	ASE では、フリースペースの管理は、スレッドホールド経由で実行
time_format	HH:NN:SS.SSS		ASE の CONVERT() 関数のデフォルトスタイルは 100
time_zone_adjustment	0		ASE では非適用
timestamp_format	YYYY-MM-DD HH:NN:SS.SSS		ASE の CONVERT() 関数のデフォルトスタイルは 100
timestamp_with_time_zone_format	YYYY-MM-DD HH:NN:SS.SSS+HH:NN		ASE では TIMESTAMP_WITH_TIME_ZONE は非サポート
truncate_timestamp_values	OFF		BIGDATETIME と BIGTIME タイプは少数第 6 桁の精度をサポート可能
tsql_outer_joins	OFF		ASE の TSQL 外部ジョインは常に有効

tsql_variables	OFF		ASE SQL プロシージャー、トリガー、関数、バッチの全ての変数は、'@'で始まる必要がある
updatable_statement_isolation	0		ASE では非適用
update_statistics	ON		ASE では非適用
upgrade_database_capability	'' (empty string)		ASE では非適用
user_estimates	OVERRIDE-MAGIC		ASE では非適用
uuid_has_hyphens	ON		ASE は NEWID() 関数のオプションをサポートしており GUID がダッシュを含むかどうか指示。もし特定されていない場合には、デフォルトはダッシュなし。Set uuid_has_hyphens を to OFF に設定すると ASE のデフォルトと互換
verify_password_function	'' (empty string)		sp_passwordpolicy システムプロシージャー参照
wait_for_commit	OFF		ASE では非適用
webservice_namespace_host	'' (empty string)		ASE では非適用。ASE リファレンス・マニュアル: Web Services 参照

SQL Anywhere でサポートされている ASE のオプション

SQL Anywhere でサポートされている Transact-SQL ダイアレクトでは、アプリケーションは、Transact-SQL SET 文を発行して SQL Anywhere でサポートされている ASE のオプションの設定をすることができます。Transact-SQL SET 文は、SET <option> <value> のフォーマットであるのに対し、SQL Anywhere SET OPTION 文は、SET [TEMPORARY] OPTION [<user>].<option> = <value> の形式になります。以下のオプションは、SQL Anywhere の Transact-SQL SET 文の中で特定することができます。この表に掲載されていない An attempt to set an ASE Transact-SQL オプションを設定しようとすると、シンタックスエラーになります。

ASE のオプション	SQL Anywhere で許される値	SQL Anywhere で設定されている相当オプション	備考
ansi_permissions	ON または OFF	ansi_permissions	SQL Anywhere のデフォルトは ON; ASE のデフォルトは OFF
ansinull	ON または OFF	ansinull	SQL Anywhere のデフォルトは ON; ASE のデフォルトは OFF 設定は、sp_tsql_environment プロシージャーで自動的に変換
arithabort	any	none	SET 文は受けつけても無視
arithignore	ON または OFF	divide_by_zero_error	ASE と SQL Anywhere のデフォルトは異なる
chained	ON または OFF	chained	ASE と SQL Anywhere のデフォルトは異なる。 設定は sp_tsql_environment プロシージャーで自動的に変換
char_convert	any	none	SET 文は受けつけても無視
close on endtran	ON または OFF	close_on_endtrans	二つのオプション間のスペルの違いに注意
cursor_rows	any	none	SET 文は受けつけても無視。カーソルを行数に制限するには、SELECT TOP を使用
datefirst	Integer between 1 and 7	first_day_of_week	ASE リファレンス・マニュアル: コマンド参照
dateformat	Date format strings, eg. “YMD”	date_order	SQL Anywhere の date_format オプションと混乱しないように。 date_order は sp_tsql_environment プ

			ロシージャーで自動的に設定
dup_in_subquery	any	none	SET 文は解析されるが、実行すると「not supported」エラーが発生.
fipsflagger	ON または OFF	sql_flagger_warning_level	SQL Anywhere の warning flagging レベルは、「SQL/92 Entry level」に設定
flushmessage	any	none	SET 文は受けつけても無視
identity_insert	ON	none	SQL Anywhere では、アプリケーションは特定の値を常に IDENTITY カラムに挿入することが可能
language	any	none	SET 文は解析されるが、実行すると「not supported」エラーが発生.
nocount	any	none	SET 文は受けつけても無視
noexec	OFF	none	SET 文は受けつけても無視
offsets	any	none	SET 文は受けつけても無視
parseonly	OFF	none	SET 文は受けつけても無視
prefetch	any	none	SET 文は受けつけても無視
procid	OFF	none	SET 文は受けつけても無視
quoted_identifier	ON または OFF	quoted_identifier	ASE と SQL Anywhere のデフォルトは異なる
role	any	none	SET 文は解析されるが、実行すると「not supported」エラーが発生.
rowcount	integer	none	サポート対象。SQL Anywhere のマニュアル SET statement [TSQL] 参照
self_recursion	ON または OFF	none	サポート対象。SQL Anywhere のマニュアル SET statement [TSQL] 参照
showplan	any	none	SET 文は受けつけても無視
statistics	any	none	SET 文は受けつけても無視
string_rtruncation	ON または OFF	string_rtruncation	SQL Anywhere のデフォルトは ON ; ASE のデフォルトは OFF

transaction isolation level	0,1,2,3, または 相当の string 値; さらに SNAPSHOT isolation levels	isolation_level	ASE は分離レベル 0 から 3 のサポートのみ。分離レベルは sp_tsql_environment プロシージャーで自動的に 1 に設定
textsize	byte 単位のサイズ	none	サポート対象

SQL Anywhere でサポートされている ASE のシステムプロシージャー

SQL Anywhere では、管理タスクの実行にシステムストアドプロシージャーよりも、主 SQL 文を使用しますが、全てではないものの、ASE でサポートされているシステムストアドプロシージャーの多くを SQL Anywhere でもサポートしています。

プロシージャー	説明	備考
sp_addgroup	相当する GRANT CONNECT, GROUP TO <group name> s 文を発行することでデータベースに対してグループを追加	
sp_addlogin	相当する GRANT CONNECT 文を発行することで新しいログイン ID をデータベースに追加	passwdexp、minpwdlen、maxfailedlogins、auth_mech パラメーターは SQL Anywhere のサポート対象外
sp_addmessage	相当する CREATE MESSAGE 文を発行	withlog と replace パラメーターは SQL Anywhere ではサポート対象外
sp_addtype	相当する CREATE DOMAIN 文を発行	
sp_adduser	相当する GRANT CONNECT 文を発行。grpname パラメーターが特定されている場合には、 GRANT MEMBERSHIP IN GROUP 文	
sp_changegroup	相当する GRANT MEMBERSHIP IN GROUP 文を発行	
sp_dboption	データベースオプションを変更	allow_nulls_by_default オプションのみが SQL Anywhere でもサポート対象。プロシージャーが SET OPTION 文を発行し、オプション値を設定
sp_dropgroup	あるユーザー ID に相当する REVOKE GROUP 文を発行	
sp_droplogin	相当する REVOKE CONNECT 文を発行し、データベースからユーザーを除外	sp_dropuser と同様
sp_dropmessage	相当する DROP MESSAGE 文を発行	特定された場合には、言語パラメーターは無視される
sp_droptype	相当する DROP DOMAIN 文を発行	
sp_dropuser	相当する REVOKE CONNECT 文を発行し、データベースからユーザーを除外	ASE では、何等かのオブジェクトを持つユーザーをドロップできない。SQL Anywhere では、これらのオブジェクトは自動的にドロップされる。SQL Anywhere マニュアルの REVOKE CONNECT 文参照

sp_getmessage	SYSUSERMESSAGE カタログテーブルからエラー文を返す	サポート対象
sp_helptext	データベースオブジェクトのソース文を返す(例えば、ストアドプロシージャー)	objname パラメーターのみが SQL Anywhere でサポート対象
sp_password	GRANT CONNECT 文を発行し、任意のユーザーのパスワードを追加または変更	ASE バージョンの即時のパラメーターは、SQL Anywhere のサポート対象外

SQL Anywhere でサポートされている ASE のグローバル変数

ASE のグローバル変数は、ASE リファレンス・マニュアル: ビルディング・ブロック, 第 3 章で解説されています。この表は、ASE 15.5 でサポートされているグローバル変数と、それらの SQL Anywhere でのサポートについて簡単にまとめたものです。

ASE のグローバル変数	ASE 説明	SQL Anywhere のサポート
@@active_instances	クラスターないのアクティブインスタンスの数を返す	サポート対象外
@@authmech	ユーザーを認証するために使用されているメカニズムを示す read-only 変数	サポート対象外
@@bootcount	ASE インストレーションがブートされた回数	サポート対象外
@@boottime	ASE が最後にブートされた日と時間を返す	サポート対象外
@@bulkarraysize	バルクコピーインターフェースを使用して移行される前にローカルサーバーメモリーにバッファーする行数を返す。SELECT INTO を使用してリモートサーバーに行を移行するためにコンポーネント統合サービスとのみ使用	サポート対象外
@@bulkbatchsize	バルクインターフェースを使って、select into proxy_table 経由でリモートサーバーに移行された行の数を返す。select into を使用して行をリモートサーバーに移行するためのコンポーネント統合サービスとのみ使用	サポート対象外
@@char_convert	文字セットコンバージョンが有効でない場合、0 を返す。文字セットコンバージョンが有効な場合には、1 を返す	0 (Transact-SQL との互換性の目的で実装)
@@cis_rpc_handling	CIS RPC ハンドリングがオフの場合には、0 を返す。CIS RPC ハンドリングがオンの場合には、1 を返す。詳細については、「コンポーネント統合サービス・ユーザーズ・ガイド」を参照	サポート対象外
@@cis_version	コンポーネント統合サービスの日付とバージョンを返す	サポート対象外
@@client_csexpansion	サーバー文字セットからクライアント文字セットにコンバージョンする時に使用される拡張係数を返す。例えば、2 という値を含む場合、サーバー文字セットの文字は、クライアント文字セットに翻訳後、byte 数の 2 倍までとる	サポート対象外
@@client_csid	クライアントの文字セットが初期化されたことがなければ-1 を返す。クライアントの文字セットが初期化されていれば、接続の syscharsets からクライアントの文字セット ID を返す	-1 (Transact-SQL との互換性の目的で実装)
@@client_csname	クライアントの文字セットが初期化されたことがなければ NULL を返す。クライアントの文字セットが初期化されていれば接続の文字列の名前を返す	NULL (Transact-SQL との互換性の目的で実装)

@@clusterboottime	最初にクラスターの起動を起動させたインスタンスがシャットダウンしていても、クラスターが最初に起動した日と時間を返す	サポート対象外
@@clustercoordid	現在のクラスターのコーディネーターのインスタンス ID を返す	サポート対象外
@@clustermode	“共有ディスククラスター”という文を返す	サポート対象外
@@clustername	クラスターの名前を返す	サポート対象外
@@cmpstate	HA 環境における ASE の現在のモードを返す。HA 環境ではない場合には使用されていない	サポート対象外
@@connections	ユーザーが試みたログインの回数を返す	サーバーが最後に起動してからのログイン回数
@@cpu_busy	最後に ASE が起動されてからの ASE の動作に CPU が使用した時間を ticks で返す	0 (Transact-SQL との互換性の目的で実装)。‘ApproximateCPUTime’ の接続プロパティーを使用して、特定の接続に使用された CPU 時間を取得
@@cursor_rows	スクロール可能なカーソルのために設計されたグローバル変数。 カーソル結果セットないの行の総数を表示。以下の値を返す: • -1 – カーソルは: 1. 動的 – 動的カーソルは全ての変更を反映するので、カーソルを qualify する行の数は、継続的に変化。全ての qualified 行が返されると確実には言えない 2. semi_sensitive で、scrollable。しかし, but the scrolling ワークテーブルはまだ完全に populated されていない – カーソルを qualify する行数は、この値が返される時点ではわからない • 0 – オープンのカーソルがない、最後にオープンしたカーソルを qualify する行がない、あるいは、最後にオープンしたカーソルは closed または deallocated のいずれか • n – 最後にオープンした、またはフェッチしたカーソルの結果セットは、フルに populated。返された値は、カーソル結果セット内の行の総数	サポート対象外
@@curloid	オープンのカーソルがない、最後にオープンしたカーソルを qualify する行がない、あるいは、最後にオープンしたカーソルは closed または deallocated のいずれか	サポート対象外
@@datefirst	n の値が 1 と 7 の間の SET DATEFIRST n を使用して設定。Tinyint としてあらわされる、各週の特定の最初の日を示し、現在の値の @@datefirst を返す。 ASE のデフォルト値は、SET DATEFIRST 7 を特定して設定する Sunday (us_language デフォルトでは)。設定と値の詳細は、SET コマンドの DATEFIRST オプションを参照	サポート対象外。SQL Anywhere の first_day_of_week オプション参照

@@dbts	現在のデータベースのタイムスタンプを返す	TIMESTAMP タイプの値は、DEFAULT TIMESTAMP で定義された全てのカラムに対して使用された最新の生成値を表す
@@error	システムによって生成された、最新のエラー番号を返す	最後に実行された文の成功または失敗をチェックする Transact-SQL エラーコード。前のトランザクションが成功の場合、0 が返される。前のトランザクションが失敗の場合、システムで生成された最後のエラー番号が返される。 if @@error != 0 return のような文はエラーが発生した場合に exit を発生させる。全ての文が PRINT statements or IF tests を含み、@@error をリセット。そのため、状態チェックは、すぐに確認したい成功の文に下側なければならない。
@@errorlog	$SYBASE ディレクトリに相対して(Windows プラットフォームでは%SYBASE%) ASE エラーログが保存されているディレクトリへのフルパスを返す	サポート対象外
@@failedoverconn	プライマリコンパニオンへの接続がフェイルオーバーして、セカンダリコンパニオンサーバーで実行している場合、0 よりも大きい値を返す。高可用性環境でのみ使用され、session-specific	サポート対象外
@@fetch_status	Returns: • 0 – フェッチオペレーション成功 • -1 – フェッチオペレーション失敗 • -2 – 将来の使用用に値を確保	最後のフェッチ文の結果の状態情報を含む。この機能は、Microsoft との互換性上は異なる値を返すという点を除いて @@sqlstatus と同じ。@@fetch_status は以下の値を含む可能性がある。 0 – フェッチ文は成功 • -1 - フェッチ文はエラー • -2 – 結果セット内にデータがない
@@guestuserid	ゲストユーザーの ID を返す	サポート対象外
@@hacmpservername	高可用性セットアップの対のサーバーの名前を返す	サポート対象外
@@haconnection	その接続のフェールオーバープロパティが有効の場合、0 よりも大きい値を返す。これは、セッション特定のプロパティ	サポート対象外
@@heapmemsize	ヒープメモリープールのサイズを byte で返す。ヒープメモリーの詳細は、システム管理ガイドを参照	サポート対象外

@@identity	最新の IDENTITY カラム値を返す	INSERT または SELECT INTO 文によって IDENTITY または DEFAULT AUTOINCREMENT カラムに対して挿入された最後の値
@@idle	ASE が最後に起動されてからアイドル状態だった時間を ticks で返す	0 (Transact-SQL との互換性の目的で実装)
@@instanceid	それが実行されるインスタンスの ID を返す	サポート対象外
@@instancename	それが実行されるインスタンスの名前を返す	サポート対象外
@@invaliduserid	無効なユーザーID に対して-1 の値を返す	サポート対象外
@@io_busy	ASE が input, output オペレーションの実行に費やした時間を tick で返す	0 (Transact-SQL との互換性の目的で実装)
@@isolation	現在の Transact-SQL プログラムのセッション特定の分離レベル(0, 1, または 3)を返す	接続の同時分離レベル。@@isolation はアクティブレベルの値をとる。Note that both ASE も SQL Anywhere も、SQL リクエストの特定のテーブルの効果的な分離レベルを変更することができるテーブルヒントをサポート
@@jsinstanceid	Job Scheuler が実行されている、あるいは有効になると実行されるインスタンスの ID	サポート対象外
@@kernel_addr	Kernel リージョンを含む最初の共有メモリのリージョンのスタートアドレスを返す。結果は、0xaddress ポインター値のフォーム	サポート対象外
@@kernel_size	最初の共有メモリリージョンの一部である kernel region のサイズを返す	サポート対象外
@@langid	syslanguages.langid で特定されている、使用されあてている言語の server-wide 言語 ID を返す	現在の接続に使用されている言語のユニークな言語 ID
@@language	syslanguages.name で特定される使用されている言語の名前を返す	接続に使用されている言語の名前
@@lastlogindate	各ユーザーログインに利用可能で、@@lastlogindate は DATETIME データタイプを含み、その値は、現在のセッションが確立する前のログインアカウントの lastlogindate カラム。この変数は、各ログインセッションに特定で、そのアカウントへの以前のログインを決定するためにそのセッションで使用することが可能。 そのアカウントが以前使用されていない、または “sp_passwordpolicy 'set', enable last login updates” が 0 の場合、@@lastlogindate の値は NULL.	サポート対象外。sa_get_user_status システムストアドプロシージャー参照
@@lock_timeout	SET LOCK WAIT n を使用して設定。ミリセカンドで current lock_timeout setting を返す。 @@lock_timeout は n の値を返す。デフォルトの値は、no timeout。セッションの最初に set lock wait n が実行されない場	サポート対象外。SQL Anywhere の blocking_timeout オプション参照

	合、@@lock_timeout は -1 を返す。	
@@maxcharlen	ASE のデフォルトの文字セットの最長を byte で返す	CHAR 文字セットの bytes 単位の最長
@@max_connections	現在のコンピューター環境で ASE で可能な最大同時接続数を返す。ユーザー接続設定パラメーターで、@@max_connections の値よりも少ない、あるいは同じ値の接続数ならいくつでも ASE を設定可能。	パーソナルサーバーは、サーバーに対して実行できる同時接続の最大数は 10。ネットワークサーバーの場合は、アクティブクライアントの最大数（それぞれのクライアントが複数接続をサポートできるので、データベース接続ではない）
@@maxgroupid	最高のグループユーザーID を返す。最高の値は 1048576	サポート対象外
@@maxpagesize	サーバーの論理ページサイズを返す	サポート対象外。SQL Anywhere の SELECT PROPERTY('PageSize') 参照
@@max_precision	サーバーで設定される decimal と numeric データタイプで使用される precision level を返す。この値は、38 に固定	サポート対象外。PRECISION と SCALE 接続オプション参照
@@maxspid	spid の最大有効値を返す	サポート対象外
@@maxsuid	サーバーユーザー ID の最大値を返す。デフォルトの値は、2147483647	サポート対象外
@@maxuserid	最高のユーザーID を返す。最高値は 2147483647	サポート対象外
@@mempool_addr	global memory pool table address を返す。結果は、 0xaddress ポインター値の形で出される。この変数は internal 使用用。	サポート対象外
@@min_poolsize	named cache pool の最少サイズを kilobytes で返す。DEFAULT_POOL_SIZE である 256 と、現在の最大データベースページサイズの値をベースに計算される	サポート対象外
@@mingroupid	最低のグループユーザーID を返す。最低値は 16384	サポート対象外
@@minspid	Spid の最低値である 1 を返す	サポート対象外
@@minsuid	最大ユーザーID を返す。最低値は 32768	サポート対象外
@@minuserid	最低のユーザーID を返す。最低の値は-32768	サポート対象外
@@monitors_active	sp_sysmon で表示されるメッセージの数を減らす	サポート対象外
@@ncharsize	現在のサーバーのデフォルトの国語文字セットの最長のサイズを byte で返す	NCHAR 文字セット内の文字の Byte 単位の最長サイズ
@@nestlevel	現在の nesting のレベルを返す	-1 (Transact-SQL との互換性の目的で実装)

@@nodeid	現在のインストレーションの 48-bit ノード識別子を返す。ASE は、マスターデバイスが最初に使用された時に最初に nodeid を生成し、ASE インストレーションを独自に識別	サポート対象外
@@optgoal	クエリ最適化のための源氏アの最適化ゴール設定を返す	サポート対象外。OPTIMIZATION_GOAL 接続オプションを使用
@@options	セッションの設定オプションの 16 進表現を返す	サポート対象外
@@opttimeoutlimit	クエリ最適化のための現在の最適化タイムアウトリミットの設定を返す	サポート対象外
@@pack_received	ASE によって読み込まれたインプットパケット数を返す	0 (Transact-SQL との互換性の目的で実装)
@@pack_sent	ASE によって書き込まれたアウトプットパケット数を返す	0 (Transact-SQL との互換性の目的で実装)
@@packet_errors	読み込み、書き込みパケット時に検出された ASE のエラー数を返す	0 (Transact-SQL との互換性の目的で実装)
@@pagesize	サーバーの仮想ページサイズを返す	サポート対象外
@@parallel_degree	現在の最大 parallel degree の設定を返す	サポート対象外
@@probesuid	プローブユーザーID のために、値 2 を返す	サポート対象外
@@procid	現在実行中のプロシージャーのストアドプロシージャーID を返す	現在実行中のプロシージャーのストアドプロシージャーID
@@quorum_physname	クォーラムデバイスの物理パスを返す	サポート対象外
@@recovery_state	これら返される値をもとに ASE がリカバリ状態にあるかどうか示す。 • NOT_IN_RECOVERY – ASE はスタートアップリカバリーまたはフェイルオーバーリカバリーの状態ではない。リカバリは完了し、オンライン可能なデータベースは全てオンラインに戻っている • RECOVERY_TUNING – ASE はリカバリの状態で (スタートアップまたはフェイルオーバー) 、リカバリータスクの最適数をチューニング中 • BOOTIME_RECOVERY – ASE はスタートアップリカバリーの状態で、タスクの最適数のチューニングを完了している。全てのデータベースがリカバリしたわけではない。 • FAILOVER_RECOVER – ASE は HA フェイルオーバー間にリカバリの状態で、リカバリータスクの最適数のチューニングを完了している。全てのデータベースがオンラインに戻っているわけではない。	サポート対象外
@@repartition_degree	現在の動的再パーティショニングの程度の設定を返す	サポート対象外
@@resource_granularity	クエリの最適化のために、最大リソース使用ヒント設定を返す	サポート対象外

@@rowcount	最後のクエリで影響のある行の数を返す。@@rowcount の値は、特定のカーソルが forward-only か、scrollable かで影響がある。カーソルがデフォルト、つまり non-scrollable カーソル、の場合、@@rowcount の値は、forward 方向にのみ結果セットの行数がフェッチされるまで一つずつ増加。これらの行は、基礎となるテーブルからクライアントにフェッチされる。@@ rowcount の最大値は、結果セット内の行の数 デフォルトカーソルでは、@@rowcount は 0 に設定。by any command that does not return or affect rows, such as an if or set command, or an update or delete statement that does not affect any rows. カーソルが scrollable な場合、 @@rowcount に最大値はない。 方向に関係なく、各フェッチで値は増加し続け、最大値はない。Scrollable カーソルの@@rowcount 値は、基礎となるテーブルからではなく、結果セットから列数をクライアントに反映	最後の文で影響された行の数。@@rowcount の値は、文の後すぐにチェクする必要がある。挿入、変更、削除は@@rowcount を影響のあった行の数に設定 カーソルでは、@@rowcount は、カーソル結果セットからクライアントに返された行の累積数を最後のフェッチリクエストまで、表す @@rowcount は、IF 文のような、どの行にも影響を与えない文ではゼロにはリセットされない。
@@scan_parallel_degree	nonclustered インデックススキャンの現在の最大並列度の設定を返す	サポート対象外。max_query_tasks 接続オプション参照
@@servername	ASE サーバーの名前を返す	現在のデータベースサーバーの名前
@@setrowcount	現在の set rowcount 値を返す	サポート対象外
@@shmem_flags	共有メモリ領域プロパティを返す。この変数は internal 使用目的。Integer の 13 bits に相当する合計 13 の異なるプロパティがある。低 bit から高 bit までを表す有効な値は: MR_SHARED, MR_SPECIAL, MR_PRIVATE, MR_READABLE, MR_WRITABLE, MR_EXECUTABLE, MR_HWCOHERENCY, MR_SWCOHERENC, MR_EXACT, MR_BEST, MR_NAIL, MR_PSUEDO, MR_ZERO.	サポート対象外
@@spid	現在のプロセスのサーバープロセス ID を返す	この接続は、現在の接続をハンドリング。sa_conn_info システムプロシージャーによって表示されるものと同じ値
@@sqlstatus	フェッチ文を実行した結果の状況情報（例外の警告）を返す	最後のフェッチ文の結果の状況情報を含む。 @@sqlstatus は以下の値を含む可能性がある • 0 – 成功したフェッチ文 1 – エラーになったフェッチ文 □ 2 – 結果セットにはデータがない •
@@ssl_ciphersuite	現在の接続に SSL が使用されていない場合には NULL を返す。さもなくば、現在の接続の SSL handshake で選択した chipher スイートの名前を返す	サポート対象外

@@stringsize	toString()方法で返される文字データの総数を返す。デフォルトは 50。最大値はおそらく 2 GB。ゼロの値は、デフォルト値を特定。詳細はコンポーネント統合サービス・ユーザーズ・ガイド参照	サポート対象外
@@system_busy	ASE がシステムタスクを実行している最中の tick 数	サポート対象外
@@sys_tempdbid	実行中のインスタンスの効果的なローカルシステムテンポラリーデータベースのデータベース ID を返す	サポート対象外
@@system_view	「instance」または「cluster」のセッション特定のシステムビュー設定を返す	サポート対象外
@@tempdbid	そのセッションのアサインされたテンポラリーデータベースの有効なテンポラリーデータベース ID (dbid) を返す	サポート対象外
@@textcolid	@@textptr で参照されるカラムのカラム ID を返す	サポート対象外
@@textdataptnid	@@textptr で参照されるテキストパーティションのパーティション ID を返す	サポート対象外
@@textdbid	@@textptr で参照されるカラムとオブジェクトを含むデータベースのデータベース ID を返す	サポート対象外
@@textobjid	@@textptr で参照されるカラムを含むオブジェクトのオブジェクト ID を返す	サポート対象外
@@textptnid	@@textptr で参照されるカラムを含むデータパーティションのパーティション ID を返す	サポート対象外
@@textptr	プロセスで挿入または更新された最新のテキスト、ユニテキスト、またはイメージカラムのテキストポインターを返す(textptr 関数と同じではない)	サポート対象外
@@textptr_parameters	TEXTPTR_PARAMETERS 設定パラメーターがオフの場合に 0 を返す。TEXTPTR_PARAMETERSの現在の状況ががオンの場合には1を返す。詳細はコンポーネント統合サービス・ユーザーズ・ガイド参照	サポート対象外
@@textsize	SELECT が返すテキスト、ユニテキスト、イメージデータの byte 数の制限を返す。デフォルトの制限は isql は 32KB byte。デフォルトはクライアントソフトウェアに依存。SET textsize でセッション単位で変更可能	テキストまたはイメージデータの最大サイズを byte 単位で指定し、select 文で返す textsize オプションの現在の値。デフォルトの設定は、32765 で、これが READTEXT を使用して返せる最大の byte 文字列。この値は、SET 文を使用して設定可能
@@textts	@@textptr で参照されるカラムのテキストタイムスタンプを返す	サポート対象外

@@thresh_hysteresis	しきい値を有効にするために必要なフリースペースの減少を返す。ヒステリシス値としても知られるこの値は、2KB データベースページで測られる。これは、しきい値をデータベースセグメント上にどれだけ近くおくことができるのかを決定	0 (Transact-SQL との互換性の目的で実装)
@@timeticks	tick ごとのマイクロセカンド数を返す。tick ごとの時間はマシンに依存	0 (Transact-SQL との互換性の目的で実装)
@@total_errors	読み込み、書き込み中に ASE が検出するエラーの数を返す	0 (Transact-SQL との互換性の目的で実装)
@@total_read	ASE が読み取るディスクの数を返す	0 (Transact-SQL との互換性の目的で実装) 。データベースレベルで DiskRead プロパティを使用
@@total_write	ASE が書き込むディスクの数を返す	0 (Transact-SQL との互換性の目的で実装) 。データベースレベルで DiskWrite プロパティを使用
@@tranchained	現在の Transact-SQL プログラムのトランザクションモードが unchained の場合には、0 を返す。現在の Transact-SQL プログラムのトランザクションモードが chained の場合は 1 を返す	SQL Anywhere でもサポート
@@trancount	現在のユーザーセッションのトランザクションの nesting レベルを返す	SQL Anywhere でもサポート。バッチの各 BEGIN TRANSACTION はトランザクション数を増やす
@@transactional_rpc	リモートサーバーへの RPC がトランザクショナルの場合は、0 を返す。リモートサーバーへの RPC がトランザクショナルでない場合は、1 を返す。詳細は、リファレンス・マニュアルの ENABLE XACT コーディネーションと SET OPTION TRANSACTION_RPC を参照。またコンポーネント統合サービス・ユーザーズ・ガイド参照	サポート対象外
@@transtate	現在のユーザーセッションで文が実行された後の現在のトランザクションの状況を返す	-1 (Transact-SQL との互換性の目的で実装)
@@unicharsize	nichar の文字サイズである 2 を返す	サポート対象外。SQL Anywhere の UNICODE は、1 to から 4 byte まで
@@user_busy	ASE がユーザータスクを実行していた間の tick 数	サポート対象外
@@version	現在の ASE の日、バージョン、その他を返す	SQL Anywhere でもサポート。 @@version はバージョン番号とビルド情報を返す
@@version_number	integer として、現在の ASE のリリースの全バージョンを返す	サポート対象外
@@version_as_integer	integer として、Adaptive Server の現在のリリースの最新のアップグレードバージョンの数を返す。例えば、ASE version 12.5, 12.5.0.3, または 12.5.1 を実行中であれば、12500 を返す	サポート対象外

SQL Anywhere の予約語

SQL Anywhere では、SQL キーワードは大文字小文字を区別しません。そのため、以下のキーワードは、大文字、小文字、あるいはその組み合わせであるかもしれません。以下のリストのワードで、大文字か小文字かの違いだけの文字列は、全て予約語になります。NON_KEYWORDS オプションを使用すると、特定のキーワードを、無効にすることができます。

RESERVED_KEYWORDS オプションを使用すると、デフォルトでは無効になっている個々のキーワードを有効に変更できます。sa_reserved_words システムプロシージャーを使用すると、予約語のリストを入手することができます。

SQL Anywhere version 12.0.1 における予約は以下のとおりです。

SQL Anywhere のキーワード			
add	all	alter	and
any	as	asc	attach
backup	begin	between	bigint
binary	bit	bottom	break
by	call	capability	cascade
case	cast	char	char_convert
character	check	checkpoint	close
comment	commit	compressed	conflict
connect	constraint	contains	continue
convert	create	cross	cube
current	current_timestamp	current_user	cursor
date	datetimeoffset	dbspace	deallocate
dec	decimal	declare	default
delete	deleting	desc	detach
distinct	do	double	drop
dynamic	else	elseif	encrypted
end	endif	escape	except
exception	exec	execute	existing
exists	externlogin	fetch	first
float	for	force	foreign
forward	from	full	goto
grant	group	having	holdlock

identified	if	in	index
inner	inout	insensitive	insert
inserting	install	instead	int
integer	integrated	intersect	into
is	isolation	join	kerberos
key	lateral	left	like
limit	lock	login	long
match	membership	merge	message
mode	modify	natural	nchar
new	no	noholdlock	not
notify	null	numeric	nvarchar
of	off	on	open
openstring	openxml	option	options
or	order	others	out
outer	over	passthrough	precision
prepare	primary	print	privileges
proc	procedure	publication	raiserror
readtext	real	reference	references
refresh	release	remote	remove
rename	reorganize	resource	restore
restrict	return	revoke	right
rollback	rollup	save	savepoint
scroll	select	sensitive	session
set	setuser	share	smallint
some	spatial	sqlcode	sqlstate
start	stop	subtrans	subtransaction
synchronize	table	temporary	then
time	timestamp	tinyint	to
top	tran	treat	trigger
truncate	tsequal	unbounded	union
unique	uniqueidentifier	unknown	unsigned
update	updating	user	using
validate	values	varbinary	varbit
varchar	variable	varying	view
wait	waitfor	when	where
while	window	with	within

work	writetext	xml	

SQL Anywhere の将来のリリースでは、追加のキーワードがこのリストに追加されると予測されます。思われます。SQL Anywhere 12.0.1 のユーザーは、SQL Anywhere の新しいキーワードと競合するリスクを避けるため、アプリケーションに SQL/2008 標準のキーワードは使用しないようにしてください。

ASE の予約語

ASE では、SQL キーワードは大文字小文字を区別しません。そのため以下のキーワードは、大文字、小文字、またはその組み合わせであるかもしれません。しかしながら、識別子は大文字小文字を区別します。

ASE 15.5 の予約語は以下のとおりです:

ASE のキーワード			
add	all	alter	and
any	arith_overflow	as	asc
at	authorization	avg	begin
between	break	browse	bulk
by	casade	case	char_convert
check	checkpoint	close	clustered
coalesce	compute	confirm	connect
constraint	continue	controlrow	convert
count	count_big	create	current
cursor	database	dbcc	deallocate
declare	decrypt	delete	desc
deterministic	disk	distinct	drop
dummy	dump	else	encrypt
end	endtran	errlvl	errordata
errorexit	escape	except	exclusive
exec	execute	exists	exit
exp_row_size	external	fetch	fillfactor
for	foreign	from	goto
grant	group	having	holdlock
Identity	identity_gap	identity_start	if
in	index	inout	insensitive
insert	install	intersect	into
is	isolation	jar	join
key	kill	level	like
lineno	load	lock	materialized

max	max_rows_per_page	min	mirror
mirrorexit	modify	national	new
noholdlock	nonclustered	nonscrollable	non_sensitive
not	null	nullif	numeric_truncation
of	off	offsets	on
once	online	only	open
option	or	order	out
output	over	partition	perm
permanent	plan	prepare	primary
print	privileges	proc	procedure
processexit	proxy_table	public	quiesce
raiserror	read	readpast	reconfigure
references	remove	reorg	replace
replication	reservepagegap	return	returns
revoke	role	rollback	rowcount
rows	rule	save	schema
scroll	scrollable	select	semi_sensitive
set	setuser	shared	shutdown
some	statistics	stringsize	stripe
sum	syb_identity	syb_restree	syb_terminate
table	temp	temporary	textsize
to	tracefile	tran	transaction
trigger	truncate	tsequal	union
unique	unpartition	update	use
user	user_option	using	values
varying	view	waitfor	when
where	while	with	work
writetext	xmlextract	xmlparse	xmltest
xmlvalidate			

BCP によるデータ移行

この付録では、BCP を使用してデータを SQL Anywhere から ASE に移行する場合の詳細について記載しています。

前述のように、BCP でデータ移行を実行する場合には、2 種類のアプローチがあります。

- SQL Anywhere の dbunload ユーティリティを使用して、SQL Anywhere テーブルからデータを抽出します。抽出したデータを ASCII ファイルに保存してから、BCP を使用して、データを ASE データベースのテーブルにロードします。
- BCP を使用して、SQL Anywhere テーブルからデータを抽出します。次に、BCP を使用して、データを ASE データベースのテーブルにロードします。SQL Anywhere では、テキスト (ASCII) とバイナリ BCP の両形式がサポートされています。

BCP ユーティリティを実行して、データをロードおよびアンロードする場合のコマンドラインについては、下記を参照してください。BCP が TDS ワイヤープロトコルを介して SQL Anywhere に接続している場合、共有メモリを転送手段として使用できないことに注意してください。BCP では、TCP/IP 接続を使用する必要があります。

"demo" サーバーで実行されている SQL Anywhere のサンプル "Demo" データベースから "Products" テーブルをテキストモードでアンロードする場合、BCP コマンドは、次のようになります。

```
bcp GroupO.Products out products.dat -c -t, -r '¥r¥n' -U dba -P sql -S demo
```

最初の 3 つの引数はそれぞれ、テーブル名、転送方向、および関連付けたファイル名を指定しています。-c オプションは、ネイティブバイナリ形式ではなく、テキスト ("character") 形式で BCP ファイルをアンロードする必要があることを示しています。-t 引数は、出力ファイルのカラム区切り記号 ("tab") にコンマを使用することを示しています (データ自体にコンマが含まれている場合は、この指定を変更する必要があります)。-r は、各ローの終端文字として '¥r¥n' (キャリッジリターン (復帰)／ラインフィード (改行) の組み合わせ) を指定しています。最後の 3 つの引数はそれぞれ、ユーザー名、パスワード、およびサーバー名を指定しています。この場合のサーバー名は実際のサーバー名ではなく、ASE sql.ini (ASE のインストール先の ¥ini サブディレクトリ内に配置) ファイルに含まれているエントリの名前に相当します。この例では、sql.ini ファイルの内容は、次のとおりです。

```
[demo]
master=TCP,localhost,2638
query=TCP,localhost,2638
```

SQL Anywhere サーバーは BCP ユーティリティと同じコンピューター上で動作しているので、SQL Anywhere のデフォルトポート 2638 を使用しています。

同じ名前の ASE データベース (demo) 内に存在する同じ名前のテーブル (所有者は dbo) に、このデータファイルを再ロードする場合、BCP コマンドは、次のようになります。

```
bcp demo.dbo.Products in products.dat -c -t, , -r '¥r¥n' –S<servername> -U<username>
-P<password>
```

コマンドライン引数の意味は、前の例と同じです。ただし、2 番目のパラメーター in は、products.dat ファイルのアンロードではなく、ロードであることを示しています。

BCP ユーティリティのコマンドライン構文の詳細については、ご使用のプラットフォームに対応する ASE ユーティリティのマニュアルを参照してください。

BCP による移行に伴い、固有の問題が発生する可能性があります。具体的には、データ型と、データ型の変換について注意が必要です。

- ☐ SQL Anywhere の DECIMAL(19,4) および DECIMAL(20,4) 型は、ASE の MONEY 型にマッピングされます。ただし、ASE の MONEY 型の方が値の有効範囲は小さくなっています (約 1015 または 1016 に対して、約 1014)。
- ☐ SQL Anywhere の DECIMAL(10,4) は、ASE の SMALLMONEY 型に対応しています。ただし、ASE の SMALLMONEY 型の方が値の有効範囲は小さくなっています (約 1,000,000 に対して、約 215,000)。
- ☐ BCP バージョン 15.5 では、SQL Anywhere の TIME および TIMESTAMP 型は ASE の BIGTIME および BIGDATETIME 型にマッピングされ、SQL Anywhere の値の精度は保持されます。BIGTIME および BIGDATETIME をサポートしていない古いバージョンの BCP では、SQL Anywhere の型はそれぞれ、ASE の TIME および DATETIME 型にマッピングされます。前述のように、DATETIME でサポートされている日付の有効範囲は小さく、精度が劣ります。具体的には、DATETIME が 1753 年から始まるのに対して、SQL Anywhere の型は 0001 年から始まります。ASE の DATETIME カラムでは、1753 年よりも前の日付は許容範囲外になります。ただし、この仕様は、ほとんどのアプリケーションで問題にはならないと考えられます。

dbunload/BCP によるアプローチを採用した場合に発生する可能性がある固有の問題は、次のとおりです。

- ☐ ASE へのデータのロード時に使用する BCP コマンドの呼び出しに対して、ローの区切り記号を明示的に設定する必要があります。具体的には、-r '¥r¥n' という追加のコマンドライン引数を BCP に対して指定します。この指定は、ローの終端文字は単独の改行文字 (BCP のデフォルト) ではなく、復帰文字に改行文字が続く組み合わせであることを意味します。

□ 文字変換は、BCP クライアントによるアンロード処理と再ロード処理の両方で発生することがあります。運用データの移行を進める前に、SQL Anywhere と ASE の両製品で使用される呼び出しの互換性をそれぞれ検証する必要があります。

まとめ

SQL Anywhere 12.0.1 と Sybase ASE 15.5 の両製品には多くの共通点がある一方で、さまざまな差異もあります。アプリケーションの移行に伴い、使用するデータベースを SQL Anywhere データベースから ASE に変更すると、両製品の差異による影響を受けると考えられます。要約すると、次のような差異があります。

- ☐ SQL Anywhere でサポートされている SQL/2008 標準の言語機能のうち、ASE でサポートされていないものとして、テーブル関数、LATERAL および FULL OUTER JOIN を使用するテーブル式、検索述部 (REGEXP など)、WINDOW 関数、RECURSIVE UNION クエリ、ローレベルトリガー、MERGE 文などが挙げられます。
- ☐ 上記の SQL 標準の機能に加えて、SQL Anywhere でサポートされている一般的な機能のうち、マテリアライズドビュー、スナップショットアイソレーション、CLR (Common Language Runtime : 共通言語ランタイム) プロシージャー、および組み込みの全文検索機能などは、ASE ではサポートされていません。
- ☐ サーバーの構成と管理は、両製品では基本的に異なります。
- ☐ 両サーバー製品には、さまざまなセマンティックの差異があり、結果として、SQL Anywhere から ASE にアプリケーションを移植するときに問題となっています。その具体例を次に示します。

    - o カタログ識別子や文字列データ値に対する大文字小文字の区別
    - o ロッキングや同時実行制御の詳細なセマンティック
    - o 暗黙的な型変換の規則
    - o 製品でサポートされている照合設定の文字の辞書式順序
    - o エラー処理と固有のエラーコード

このような移植／アプリケーションの変換に伴う問題を軽減するには、ASE でサポートされていない SQL 言語の構成要素・構文の使用を控えるとともに、実際に移行したアプリケーションを ASE サーバーでテストする際に、前述のセマンティックの問題点について注意する必要があります。