SYBASE®
An SAP® Company

管理ガイド 第 1 巻

Replication Server® 15.7.1

ドキュメント ID： DC38789-01-1571-01
改訂： 2012 年 5 月

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

目次

表記の規則

ここでは、Sybase® マニュアルで使用しているスタイルおよび構文の表記規則について説明します。

表記の規則

構文要素	定義
`mono-spaced(fixed-width)`	• SQL およびプログラム・コード • 表示されたとおりに入力する必要のあるコマンド • ファイル名 • ディレクトリ名
`italic mono-spaced`	SQL またはプログラム・コードのスニペット内では、ユーザ指定の値のプレースホルダ (以下の例を参照)
italic	• ファイルおよび変数の名前 • 他のトピックまたはマニュアルとの相互参照 • 本文中では、ユーザ指定の値のプレースホルダ (以下の例を参照) • 用語解説に含まれているテキスト内の用語
bold san serif	• コマンド、関数、ストアド・プロシージャ、ユーティリティ、クラス、メソッドの名前 • 用語解説のエントリ (用語解説内) • メニュー・オプションのパス • 番号付きの作業または手順内では、クリックの対象となるボタン、チェック・ボックス、アイコンなどのユーザ・インタフェース (UI) 要素

必要に応じて、プレースホルダ (システムまたは設定固有の値) の説明が本文中に追加されます。次に例を示します。

次のコマンドを実行します。

```
installation directory¥start.bat
```

installation directory はアプリケーションがインストールされた場所です。

構文の表記規則

構文要素	定義
{ }	中カッコで囲まれたオプションの中から必ず 1 つ以上を選択する。コマンドには中カッコは入力しない。
[ ]	角カッコは、オプションを選択しても省略してもよいことを意味する。コマンドには角カッコは入力しない。
( )	このカッコはコマンドの一部として入力する。
|	縦線はオプションのうち 1 つのみを選択できることを意味する。
,	カンマは、表示されているオプションを必要な数だけ選択でき、選択したものをコマンドの一部として入力するときにカンマで区切ることを意味する。
...	省略記号 (...) は、直前の要素を必要な回数だけ繰り返し指定できることを意味する。省略記号はコマンドには入力しない。

大文字と小文字の区別

- すべてのコマンド構文およびコマンドの例は、小文字で表記しています。ただし、複写コマンド名では、大文字と小文字が区別されません。たとえば、**RA_CONFIG**、**Ra_Config**、**ra_config** は、すべて同じです。
- 設定パラメータの名前では、大文字と小文字が区別されます。たとえば、**Scan_Sleep_Max** は、**scan_sleep_max** とは異なり、パラメータ名としては無効になります。
- データベース・オブジェクト名は、複写コマンド内では、大文字と小文字が区別されません。ただし、複写コマンドで大文字と小文字が混在したオブジェクト名を使用する場合 (プライマリ・データベースの大文字と小文字が混在したオブジェクト名と一致させる場合)、引用符でオブジェクト名を区切ります。次に例を示します。**pdb_get_tables** "*TableName*"
- 識別子および文字データでは、使用しているソート順によっては大文字と小文字が区別されます。
  - “binary” などの大文字と小文字を区別するソート順を使用する場合には、識別子や文字データは、大文字と小文字を正しく入力してください。
  - “nocase” などの大文字と小文字を区別しないソート順を使用する場合には、識別子や文字データは、大文字と小文字をどのような組み合わせでも入力できます。

用語

Replication Agent™ は、Adaptive Server® Enterprise、Oracle、IBM DB2 UDB、Microsoft SQL Server 用の Replication Agent を表現するために使用される一般的な用語です。具体的な名前は、次のとおりです。

- RepAgent － Adaptive Server Enterprise 用の Replication Agent スレッド
- Replication Agent for Oracle
- Replication Agent for Microsoft SQL Server
- Replication Agent for UDB － Linux、Unix、Windows 用の IBM DB2

Replication Server の概要

Replication Server® は、分散データベース・システムにデータを複写します。Replication Server の利点と機能、データを複写するためのさまざまな方法と概念、複写システムを管理するユーザの役割の定義について説明します。

Replication Server について

Replication Server は、複数のデータベースの複写データを管理し、データの整合性と一貫性を維持します。複写システム内のデータベースを使用するクライアントには、ローカル・データへのアクセスが提供されるので、ネットワークと集中管理されたコンピュータ・システムの負荷が軽減されます。

Replication Command Language (複写コマンド言語)
複写コマンド言語 (RCL: Replication Command Language) を使用すると、複写ファンクションのカスタマイズと、複写システムのモニタおよび管理ができます。たとえば、テーブル、データ・ロー、またはカラムのレベルで、データのサブセットを複写するように要求することができます。この機能によって、レプリケート・サイトで必要とされるデータだけを複写できるので、さらにオーバヘッドを削減できます。

異機種データ・サーバ
Replication Server は、異機種データ・サーバをサポートしています。既存のデータベースとアプリケーションを変換しなくても、それらに対して複写システムを構築できます。企業の成長と変化に応じて、必要性に合うようにデータ・サーバを複写システムに追加することができます。

複写モデル
Replication Server は、ネットワーク内でのデータの複写に、基本的なパブリッシュとサブスクライブ・モデルを使用しています。あるユーザがプライマリ・データベース内で利用可能なデータを「パブリッシュ」すると、別のユーザがレプリケート・データベースにそのデータを配信するように「サブスクライブ」します。ユーザは、データに対する変更 (更新、挿入、削除オペレーション) とストアド・プロシージャに対する変更のどちらでも、この方法を使用して複写できます。

データをパブリッシュし、サブスクライブする指示は、各データベースを管理している、つまり「コネクション」を持っている Replication Server に与えられます。ユーザは、プライマリ Replication Server で、「複写定義」を作成します。このプライマリ Replication Server は、パブリッシュ対象のデータを格納しているプライマリ・データベースを管理します。複写定義には、どのカラムを複写するかなど

の情報を指定します。また、「データベース複写定義」の場合には、複写するデータベース・オブジェクトの情報を指定します。ユーザは、レプリケート Replication Server で「サブスクリプション」を作成します。この レプリケート Replication Server は、情報を受け取るレプリケート・データベースを管理します。

Replication Server ルート

Replication Server は、ユーザ定義のルートを介して相互に通信します。通常、プライマリ Replication Server は、1 つ以上のルートを経由してレプリケート Replication Server にデータを送信します。このルートは、プライマリ・データベースからレプリケート・データベースへデータを送信するよう設定されています。また、レプリケートからプライマリにストアド・プロシージャを送信して、プライマリ・データの更新を要求することもできます。この場合、データは 1 つまたは複数のルートを経由して、レプリケート Replication Server からプライマリ Replication Server へ渡されます。

複写システムの構造は、コネクションとルートによって定義されます。これらによって、Replication Server は相互にメッセージを送信し、データベースにコマンドを送信できます。コネクションは、メッセージを Replication Server からデータベースに転送します。ルートは、送信元 Replication Server から送信先 Replication Server に要求を転送します。

非同期トランザクション複写

複写は、非同期に発生します。つまり、プライマリ・データベースでのデータの更新は、更新自体とは別のトランザクションでレプリケート・データベースに転送されます。

非同期複写には重要な利点がありますが、システム設計者は、元の更新と複写された更新との間に遅延が発生することに注意する必要があります。

ローカル・データを複写する利点

ローカル・データ・サーバにテーブルを複写すると、クライアントはエンタープライズ・データにローカルでアクセスできるようになるため、パフォーマンスが向上し、データの可用性が高まります。

パフォーマンスの向上

標準的な Replication Server システムでは、データ要求はローカル・データ・サーバ上で完了します。WAN にはアクセスしません。ローカル・クライアントのパフォーマンスは、次の理由で向上します。

- データ転送速度は、WAN よりも LAN の方が高速である。

- ローカル・アクセスは、WAN を介したネットワーク通信量による影響を受けない。ローカル・データ・サーバのリソースを共有するローカル・クライアントは、集中管理されたリソースを社内全体のユーザと奪い合うことがない。

データの可用性の増大
データは、Replication Server システムのローカル・データベースとリモート・データベースに複写されるため、クライアントは次のようなフォールト・トレラントな環境で動作できます。

- リモート・データベースで障害が発生した場合でも、クライアントは複写データのローカル・コピーを使用できる。
- WAN に障害が発生した場合は、クライアントは複写データのローカル・コピーを使用できる。
- ローカル・データ・サーバに障害が発生した場合は、クライアントは別のサイトの複写データを使用できる。

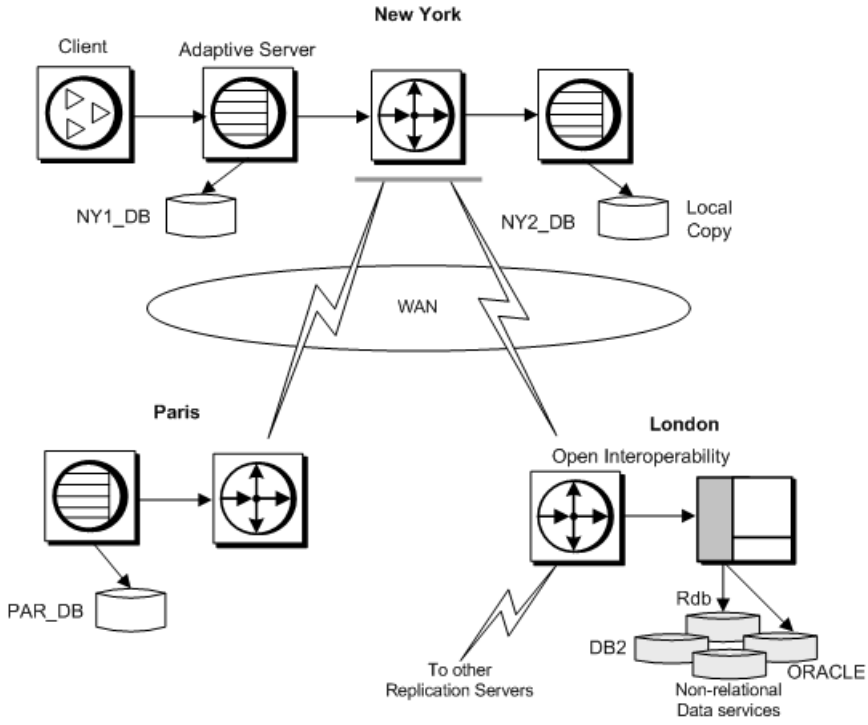
他の場所でネットワークやデータベースに障害が発生しても、ローカル・データベースでの作業が停止させられることはありません。WAN の通信に障害が発生すると、Replication Server は複写テーブルに対するさまざまなオペレーションを「*ステーブル・キュー*」 (ディスク領域) に格納します。障害で使用できなくなったデータベースにある複写テーブルは、通信がレジュームされたときに更新されます。ローカル・データ・サーバに障害が発生しても、クライアントは一時的にデータのレプリケート・コピーにアクセスして作業を続行できます。

Replication Server と分散データベース・システム

分散データベース・システムでは、クライアント・アプリケーションは、(事業拠点が地理的に分散している企業であっても) 企業内の複数のデータベース・サーバ上のデータにアクセスできます。

Replication Server は、レプリケート・データベース上のデータを常に最新の状態に保ち、送信元データベースの負荷を軽減します。以下の図に示すように、事業拠点が地理的に分散している企業は、多数の LAN と 1 つ以上の WAN で構成されていると考えられます。

図 1 : 分散環境における複写システム



Replication Server により、分散システムでのリモート・アクセスに関連するパフォーマンスとデータ可用性の一般的な問題を最小限に抑えることができます。Replication Server はデータのコピーを複数提供するので、クライアントは、リモートの集中型データベースではなくクライアント自体のローカル・データを利用できます。さらに、必要なデータだけを送信先データベースにコピーすることもできます。Replication Server では、複写するテーブルの全部または一部を指定した複写定義を作成できます。その後、必要とするローだけのサブスクリプションを作成できます。複写するデータベース・オブジェクトを指定したデータベース複写定義を作成できます。データベース・オブジェクトとは、テーブル、関数、システム・プロシージャ、トランザクション、データ定義言語 (DDL: Data Definition Language) を指します。また、ストアド・プロシージャの複写定義 (「ファンクション複写定義」と呼びます) を作成して、大量のデータの高速複写を容易にしたり、レプリケート・データベースからプライマリ・データベースに更新を複写したりできます。アプリケーションで必要な場合は、複数のプライマリ・テーブルから 1 つの集中型データベースに複写データを統合、つまり「ロールアップ」できます。

テーブル複写定義とファンクション複写定義の両方の複写定義を 1 つの「パブリケーション」にグループ化して、一度にすべてのサブスクリプションを作成でき

ます。パブリケーションを使用すると、複数のサブスクリプションを編成し、1 つのコマンドでそれらのサブスクリプションをモニタできます。

「Replication Agent」は、Adaptive Server を実行しているサイトの RepAgent です。Replication Agent は、データベースのトランザクション情報を Replication Server に転送し、複数のレプリケート・データベースに分配できるようにします。Sybase では、Microsoft SQL Server、DB2、Oracle 用の Replication Agent も提供しています。RepAgent は Adaptive Server スレッドです。他の Replication Agent はすべて別のプロセスです。

Replication Server には、分散システム内でデータを複写するためのモデルがいくつかあります。使用するアプリケーションに最適なモデルについては、『Replication Server デザイン・ガイド』を参照してください。選択するモデルによって、システムを設定する方法が決まります。

分配モデルに応じた複写システムの設定には、次の作業があります。

- プライマリ・データとレプリケート・データを格納するためのテーブルの作成。
- Replication Server 間のルートとコネクションの設定、およびプライマリ・データに対するアクセスを制御するパーミッションの設定。
- 複写するデータを特定する複写定義の作成。
- レプリケート・データベースからその複写定義へのサブスクリプションの作成。

参照：

- Replication Server の技術的概要 (27 ページ)
- 複写システムの管理 (83 ページ)

Replication Server の基本プライマリ・コピー・モデル

基本プライマリ・コピー・モデルは、Replication Server がデータをコピーするために使用する最も単純な方法であり、これによって、1 つの送信元データベース (プライマリ・データベース) から 1 つ以上の送信先データベース (レプリケート・データベース) に更新内容を分配することが可能になります。

一貫性を保証するために、送信元テーブルがプライマリ・テーブルとして指定されます。テーブルの他のバージョンはすべて、レプリケートです。この方法では、レプリケート・テーブルは読み込み専用で、データを修正しないオペレーションに使用されます。

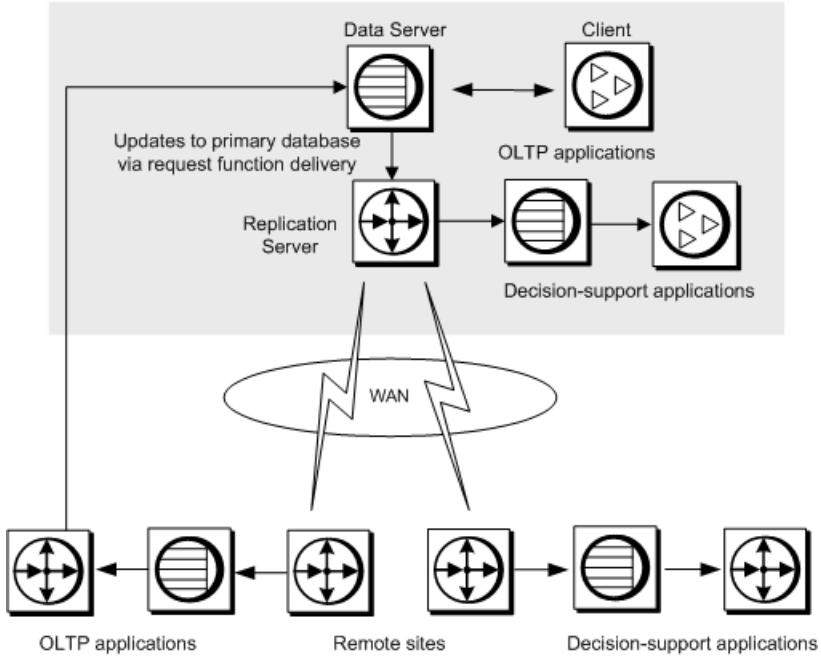
プライマリ・テーブルに更新が発生すると、Replication Server はその更新内容を取得してレプリケート・データ・サーバに送信します。このモデルでは、リモート・サイトのクライアントは、ネットワーク経由でプライマリ・データベースにアクセスしてプライマリ・データを直接更新することも、レプリケート・ストアド・プロシージャを使用して間接的に更新することもできます。

プライマリ・データベースと送信先データベース間の通信が失敗すると、プライマリ・データベースで実行されたオペレーションは、レプリケート・サイトに配布できるようになるまで Replication Server のステーブル・キューに格納されます。同様に、リモートで実行されたオペレーションは、プライマリ・データベースに配信できるようになるまでステーブル・キューに保持されます。

この方法では、リモートのクライアント・アプリケーションは、基本的なプライマリ・コピー・モデルを維持しながら、Replication Server のフォールト・トレランスを利用できます。

この図は、プライマリ・コピー方法を使用してデータを複写する Replication Server の構成を示しています。

図 2 : Replication Server の基本プライマリ・コピー・モデル



参照：

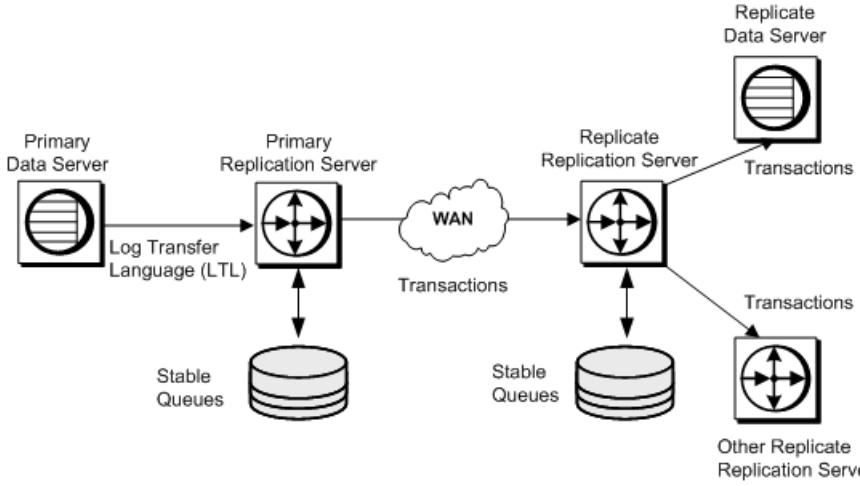
- 複写用データの指定方法 (36 ページ)
- 複写ファンクションの管理 (385 ページ)
- Replication Server でのトランザクション処理 (47 ページ)

複写システムの処理

標準的な複写システムは基本プライマリ・コピー・モデルに基づいています。このモデルでは、プライマリ Replication Server とデータ・サーバは WAN を隔ててレプリケート Replication Server から分離されています。

注意： この例では、プライマリ・データがレプリケート・データベースで更新される状況については取り上げていません。

図 3 : 複写システムの概要



複写システムの概要に関するこの図は、プライマリ・データベースからレプリケート・データベースにデータが複写されるしくみを示しています。次の処理が発生します。

1. RepAgent は、プライマリ・データベース・ログを読み込んで、複写するようマーク付けされたテーブルまたはストアド・プロシージャ用のトランザクションを、Replication Server に送信されるコマンドに変換します。Replication Server は、分散同時制御法を用いてトランザクションをステーブル・キューに格納します。
2. プライマリ Replication Server は、次の処理を実行します。
   a. そのデータに対するサブスクリプションを持つレプリケート・データベースを管理している Replication Server がどれであるかを判断します。
      プライマリ Replication Server は、サブスクリプションを作成した Replication Server への直接ルート、または中間に 1 つまたは複数の中間 Replication Server を介在させた間接ルートを持つことができます。

   b. 適切なレプリケート Replication Server にトランザクションを転送します。そこでトランザクションは、ステーブル・キューに格納されます。
   c. データに対するサブスクリプションがあるすべてのローカル・レプリケート・データベースに、トランザクションを適用します。
3. レプリケート Replication Server は、次の処理のどちらかまたは両方を実行します。
   - トランザクションを別の Replication Server にルート指定します。
   - 管理対象のレプリケート・データベースにトランザクションを適用します。

参照：
- 分散同時制御 (53 ページ)

プライマリ・コピー・モデル・システムの設定

基本プライマリ・コピー・モデルに従って複写システムを設定するには、複写システム・コンポーネントを作成して設定する必要があります。

- Replication Server 間にルートとコネクションを設定する。
- プライマリ・データベースとレプリケート・データベースにテーブルを作成する。このテーブルは各データベース内で同じ構造を持つ必要があります。
- テーブルに対するインデックスを作成して適切なパーミッションを付与する。
- **sp_setreptable** システム・プロシージャを使用して、テーブルの複写を許可する。
- プライマリ・サイトのテーブルの複写定義を作成する。
- 各サイトで、プライマリ・サイトのテーブル複写定義に対するサブスクリプションを作成する。

参照：
- ルートの管理 (161 ページ)
- データベース・コネクションの管理 (189 ページ)
- Replication Server のセキュリティ管理 (233 ページ)
- 複写テーブルの管理 (299 ページ)
- サブスクリプションの管理 (405 ページ)

他の分散データ・モデル

Replication Server では、基本的なプライマリ・コピー・モデルだけでなく、他の分散データ・モデルに基づいたシステムも設計できます。

他の分散データ・モデルには以下が含まれます。

- 分散プライマリ・フラグメント

- コーポレート・ロールアップ
- 再分配コーポレート・ロールアップ

これらの分散データ・モデルの詳細については、『Replication Server デザイン・ガイド』の「実装方式」を参照してください。

ウォーム・スタンバイ・アプリケーションは、別のタイプのアプリケーション・モデルです。『Replication Server 管理ガイド 第 2 巻』の「ウォーム・スタンバイ・アプリケーションの管理」を参照してください。

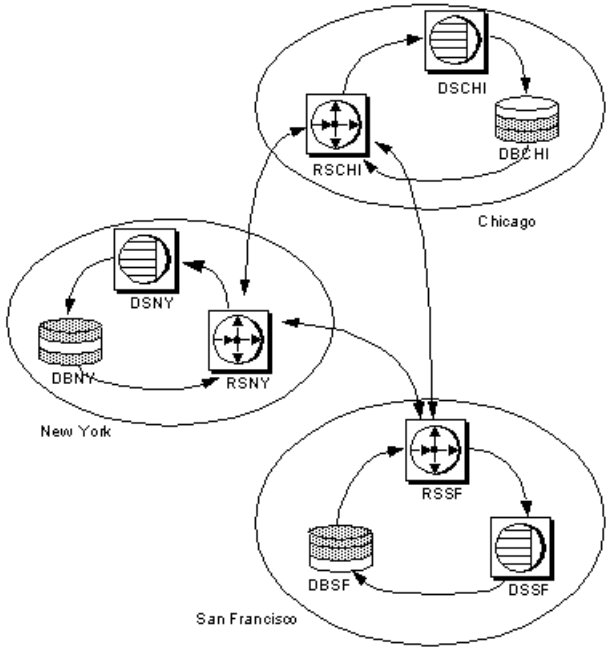
分散プライマリ・フラグメント

分散プライマリ・フラグメント・モデルを使用するアプリケーションには、プライマリ・データと複写データの両方を含む分散テーブルがあります。

各サイトの Replication Server は、ローカル・プライマリ・データに行われた修正を他のサイトに分配して、他のサイトから受信した修正をローカルに複写されるデータに適用します。

この図は、分散プライマリ・フラグメント・モデルでのデータの流れを示しています。

図 4 : 分散プライマリ・フラグメント・モデル



分散プライマリ・フラグメント・システムを設定するために必要な作業は、基本プライマリ・コピー・システムを作成する場合と似ていますが、次のような例外と追加作業があります。

- アプリケーションは、複数のサイトが同じデータを同時に更新するのを避けるか、それに対処する必要があります。各フラグメントが 1 つの「所有者」サイトを持つようにすることをおすすめします。
- データベースは、プライマリ・データベースとレプリケート・データベースのどちらにでもなり得ます。同じ構造を持つテーブルが、プライマリ・サイトとレプリケート・サイトの両方に存在することを確認してください。
- 各プライマリ・サイトから、そのデータのサブスクリプションを作成するすべてのサイトへのルートを作成します。
- 「リモート」サイトも含め、プライマリ・データが存在するすべてのサイトで複写定義を作成します。
- 各サイトで、他のサイトの複写定義に対するサブスクリプションを作成します。サイト数が *n* 個であれば、複写定義ごとに *n*-1 個のサブスクリプションを作成します。

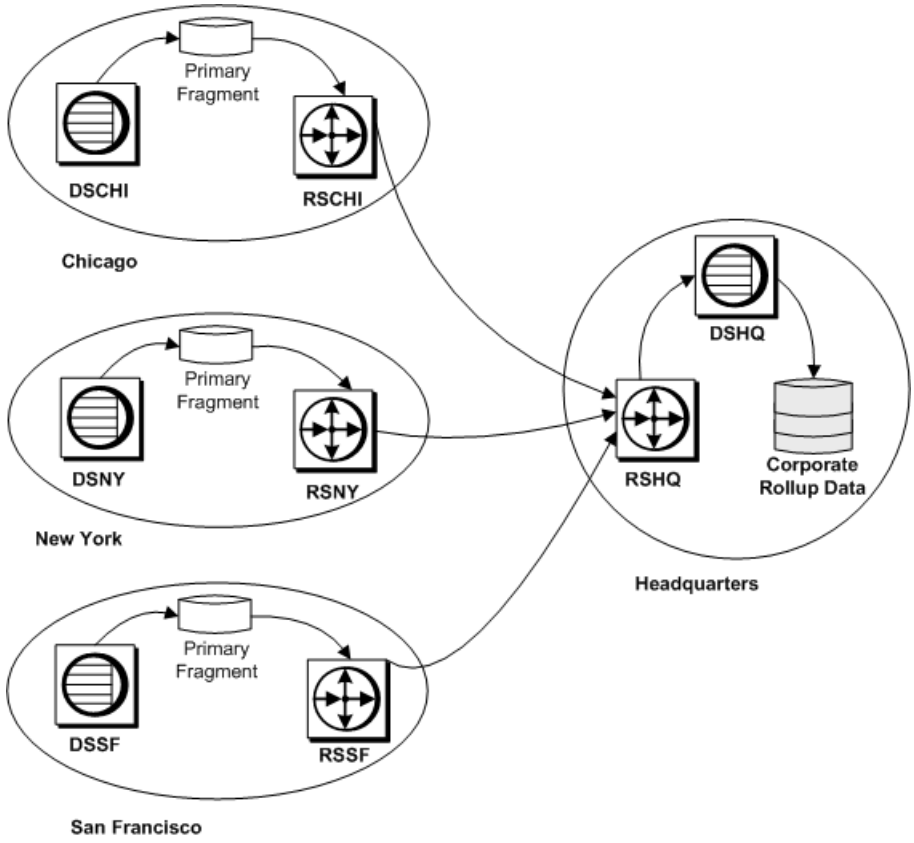
コーポレート・ロールアップ

コーポレート・ロールアップ・モデルには、複数の分散プライマリ・フラグメント・モデルと 1 つの集中化された統合レプリケート・テーブルがあります。

各プライマリ・サイトのテーブルには、そのサイトでプライマリであるデータだけが含まれます。これらのサイトにはデータは複写されません。コーポレート・ロールアップ・テーブルは、プライマリ・サイトのデータの「ロールアップ」です。

この図は、コーポレート・ロールアップ・アプリケーション・モデルでのデータの流れを示しています。

図 5 : コーポレート・ロールアップを持つ分散プライマリ・フラグメント・モデル



コーポレート・ロールアップ・モデルでは、プライマリ・サイトごとにそれぞれ別の複写定義が必要となります。データが統合されているサイトは、各プライマリ・サイトの複写定義に対するサブスクリプションを持ちます。

コーポレート・ロールアップ・アプリケーションの作成

コーポレート・ロールアップ・アプリケーションは分散プライマリ・フラグメントから作成できます。

1. 各プライマリ・サイトで、Replication Agent をアクティブにする。ただし、データは中央サイトからは複写されないため、中央サイトの Replication Agent をアクティブにする必要はない。
2. 各プライマリ・データベースと中央サイトのデータベースにテーブルを作成する。
3. プライマリ・データが格納されている各リモート・データベースで、テーブルを複写できるようにする。

4. プライマリ・データが格納されている各リモート・サイトで、テーブルの複写定義を作成する。
5. データが統合される本社サイトで、リモート・サイトの複写定義に対するサブスクリプションを作成する。

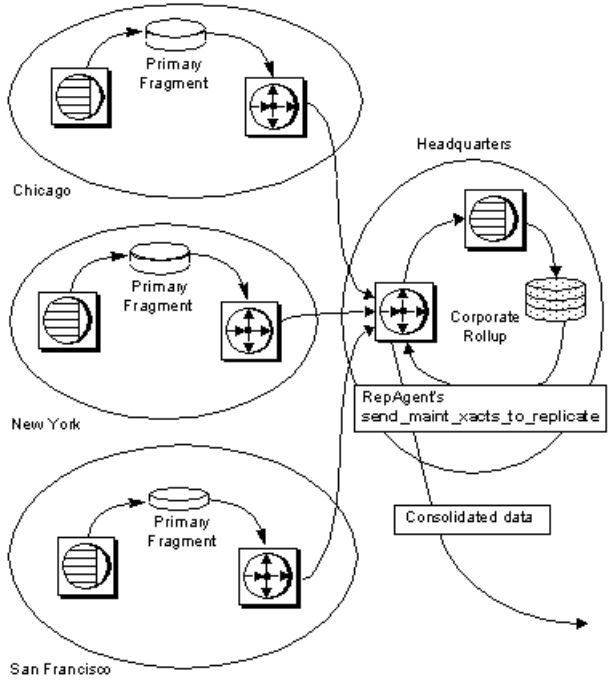
再分配コーポレート・ロールアップ

再分配コーポレート・ロールアップ・モデルは、コーポレート・ロールアップ・モデルに似ています。

リモート・サイトに分配されるプライマリ・フラグメントは、中央サイトの統合テーブルにロールアップされます。ただし、フラグメントが統合されるサイトでは、Replication Agent は統合テーブルをプライマリ・データであるかのように処理します。次にデータは、Replication Server に転送されてサブスクライバに分配されます。

この図は、再分配コーポレート・ロールアップ・モデルに基づくアプリケーションでのデータの流れを示しています。

図 6 : 再分配コーポレート・ロールアップを持つ分散フラグメント・モデル



統合テーブルは、複写定義で記述されます。他のサイトは、この後に、このテーブルに対してサブスクリプションを作成できます。コーポレート・ロールアップ・テーブルをアプリケーションで直接更新できないようにしてください。すべての更新は、プライマリ・サイトから開始するようにします。

再分配コーポレート・ロールアップ複写システムの構築に伴う作業は、コーポレート・ロールアップ・モデルと同じですが、次の点が異なります。

- 本社サイトで、統合データベースの Replication Agent をアクティブにする必要があります。これにより、すべての更新が 1 つのクライアント・アプリケーションによって行われたかのように Replication Server に送信されます。

  RepAgent では、**send_maint_xacts_to_replicate** オプションを **true** に設定する必要があります。そうしないと、Replication Agent のフィルタは、複写データをプライマリ・データとして再分配しなくなります。
- データは本社サイトから再分配されるため、本社の Replication Server には Replication Agent が必要となります。
- 本社サイトでは、複写定義を各テーブルについて作成する必要があります。他のサイトは、この複写定義に対するサブスクリプションを作成できますが、プライマリ・サイトは、それ自身のプライマリ・データに対してサブスクリプションを作成することはできません。
- 本社の Replication Server は、統合レプリケート・テーブルに対するサブスクリプションを作成する、他のサイトへのルートを持つ必要があります。プライマリ・サイトがサブスクリプションを作成する場合は、本社からそれらのサイトへのルートを作成する必要があります。
- ロールアップ・サイトには、そのプライマリ・データに対するサブスクリプションの再作成を許可しないでください。再作成を許可すると、トランザクションがシステムの無限ループを発生させることがあります。

参照：

- 複写システムの管理 (83 ページ)

Replication Server と ASE 以外のデータ・サーバ

Replication Server は、オープン・インタフェースによって ASE 以外のデータ・サーバをサポートしています。必要とされる一連の基本的なデータ・オペレーションとトランザクション処理命令をサポートしていれば、どのようなデータ記憶システムでもデータ・サーバとして使用できます。

Sybase では、Replication Server を Oracle データ・サーバと接続するための ExpressConnect for Oracle、および Replication Server をサポートされる ASE 以外の他のデータ・サーバと接続するための Sybase Enterprise Connect™ Data Access (ECDA) を提供しています。データ・サーバが Sybase Open Client™ と Open Server™ をサポートしていない場合は、Open Server ゲートウェイを作成することによって、Replication Server がデータ・サーバにアクセスできるようになります。

さまざまなベンダのデータベースで Replication Server を使用する方法の詳細については、『Replication Server 異機種間複写ガイド』を参照してください。

その他のオープン・アーキテクチャ・コンポーネントには、次のものがあります。

- Replication Agent

  Replication Agent は、プライマリ・データに加えられた変更を検出し、他のデータベースに分配するために Replication Server に送信します。

  Adaptive Server の RepAgent スレッドは、Adaptive Server データベース用の Replication Agent です。

  Adaptive Server 以外のデータ・サーバを使用する場合は、そのデータ・サーバ用の複写エージェントを用意してください。Microsoft SQL Server、IBM DB2 UDB、Oracle の各データベース用の Replication Agent を Sybase から入手できます。詳細については、『Replication Server デザイン・ガイド』と、Replication Agent に関する Sybase のマニュアルを参照してください。
- エラー・クラスとエラー処理アクション

  エラー・クラスを使用すると、データ・サーバのタイプに合わせてデータベース・エラーを処理するようにシステムを調整できます。データ・サーバが Replication Server に返すエラーに応じて、エラー・アクションを指定できます。Replication Server には、Adaptive Server 用のデフォルト・エラー・クラスが用意されています。Replication Server 15.2 以降には、Oracle、Microsoft SQL Server、IBM DB2 UDB の各データベース用のデフォルト・エラー・クラスが用意されています。詳細については、『Replication Server 管理ガイド 第 2 巻』の「エラーと例外の処理」を参照してください。
- ファンクション、ファンクション文字列、ファンクション文字列クラス

  Replication Server は、ファンクション文字列を使用して、送信先データベースのタイプに合わせて複写オペレーションを適切にフォーマットします。複写システム管理者を支援するために、Replication Server では、特定タイプのデータベース用のすべてのファンクション文字列をファンクション文字列クラスにグループ化しています。

  Replication Server には、Adaptive Server 用のデフォルト・ファンクション文字列クラスと、Oracle、Microsoft SQL Server、IBM DB2 UDB の各データベース用のファンクション文字列クラスをインストールするコネクション・プロファイルが用意されています。使用するデータベースとアプリケーションに適したコマンドを実行するように、ファンクション文字列をカスタマイズできます。詳細については、『Replication Server 管理ガイド 第 2 巻』の「データベース・オペレーションのカスタマイズ」を参照してください。

ウォーム・スタンバイ・アプリケーション

アクティブ・データベースのスタンバイ・コピーとして機能する一連のデータベースを管理するには、ウォーム・スタンバイ・アプリケーションを使用します。

クライアントがアクティブ・データベースを更新すると、Replication Server はトランザクションをスタンバイ・データベースにコピーして、両方のデータベースの一貫性を維持します。何らかの理由でアクティブ・データベースに障害が発生した場合は、スタンバイ・データベースに切り替えてそれをアクティブ・データベースとし、処理をほとんど中断することなくオペレーションをレジュームできます。

Replication Server には、ウォーム・スタンバイ・アプリケーションを設定するための方法が 2 とおりあります。どちらの方法でも、アクティブ・データベースとスタンバイ・データベースは、Adaptive Server データベースまたは Oracle データベースでなければなりません。これらは、システム内の他のデータベースに対して、プライマリ・データベースまたはレプリケート・データベースとしての機能を果たすことができます。

- 1 番目の方法では、MSA (Multi-Site Availability) 機能を使用して、アクティブ・データベースを 1 つ、スタンバイ・データベースを 1 つ以上設定します。
- 2 番目の方法では、アクティブ・データベースとスタンバイ・データベースを 1 つずつ設定できます。このとき、いずれも、同じ Replication Server で管理する必要があります。このウォーム・スタンバイ・アプリケーションは、Replication Server システム内で 1 つの論理単位と見なされます。

  Adaptive Server データベース用のウォーム・スタンバイ・アプリケーションを設定するには、『Replication Server 管理ガイド 第 2 巻』の「ウォーム・スタンバイ・アプリケーションの管理」を参照してください。

  Oracle データベース用のウォーム・スタンバイ・アプリケーションを設定するには、『Replication Server 異機種間複写ガイド』の「Oracle に対する異機種ウォーム・スタンバイ」を参照してください。

参照：

- MSA を使用した複写オブジェクトの管理 (463 ページ)

混合バージョンの複写システム

複写システムには、さまざまなバージョンの Replication Server または Adaptive Server を含めることができます。各システムには、それぞれ異なる問題があります。

- 複写システム・ドメインに Replication Server 15.5 以降がある場合は、複写システム・ドメインのシステム・バージョンとすべてのサイトおよびルート・バージョンが 12.6 以降でなければなりません。

  バージョン 15.5 以降にアップグレードするには、その前に Replication Server をバージョン 12.6 以降にアップグレードし、サイト・バージョンを 12.6 以降に設定して、ルートを 12.6 以降にアップグレードする必要があります。

  詳細については、『Replication Server 設定ガイド』の「Replication Server のアップグレードまたはダウングレード」を参照してください。
- すべての Replication Server がバージョン 12.6 以降で、システム・バージョンが 12.6 に設定されている場合、各 Replication Server はその「*サイト・バージョン*」に応じた機能を使用できます。たとえば、バージョン 15.5 が動作している Replication Server では 15.2 の機能をすべて使用できますが、11.0.2 が動作している Replication Server で使用できるのは 11.0.2 の機能だけです。このようなシステムは、「*混合バージョン・システム*」と呼ばれ、各 Replication Server はそれぞれの機能をすべて使用できます。

混合バージョン・システムの制限

異なるバージョンの Replication Server 間の対話は、最も古いバージョンの機能に制限されます。

新機能に対応する情報は、古いバージョンの Replication Server では利用できないことがあります。ファンクション文字列の継承や複数の複写定義など、新しいバージョンで採用された各機能を、混合バージョン環境で使用する場合の制限事項の詳細については、マニュアルを参照してください。

混合バージョン・システム、およびサイト・バージョンとシステム・バージョンの設定の詳細については、使用しているプラットフォームの『Replication Server インストール・ガイド』、『Replication Server 設定ガイド』、『Replication Server リリース・ノート』を参照してください。

混合バージョンの Adaptive Server

Replication Server バージョン 15.0 以降は、さまざまなバージョンの Adaptive Server とともに使用できます。

Adaptive Server 以外のデータ・サーバをデータの送信元および送信先として使用することはできますが、Replication Server には、ウォーム・スタンバイ・データベース用に Adaptive Server が必要となり、Replication Server システム・データベース (RSSD: Replication Server System Database) 用に Adaptive Server または SQL Anywhere® が必要となります。

注意： Sybase では、Adaptive Server システム・データベース (`tempdb`、`model`、`sybsystemprocs`、`sybsecurity`、および `sybsystemdb` など) の複写はサポートしていません。Adaptive Server システム・データベースの master の複写は、Adaptive Server が master データベースの複写をサポートしている場合にのみサポートされます。

Replication Server バージョン 15.0.1 の一部の機能では、Adaptive Server バージョン 15.0.1 以降を使用する必要があります。

Replication Server バージョン 15.2 の一部の機能では、Adaptive Server バージョン 15.0.3 以降を使用する必要があります。たとえば、Replication Server バージョン 15.2 の SQL 文の複写機能では、Adaptive Server バージョン 15.0.3 以降が必要です。

Replication Server での Adaptive Server の使用方法の詳細については、使用しているプラットフォームの『Replication Server インストール・ガイド』、『Replication Server 設定ガイド』、『Replication Server リリース・ノート』を参照してください。

複写システムのセキュリティ

Replication Server では、システム・セキュリティに不可欠なログイン名、パスワード、パーミッションを綿密に管理できます。さらに、Replication Server では、ネットワークを介したデータ転送を保護する、サード・パーティ製のセキュリティ・メカニズムをサポートしています。

参照：

- Replication Server のセキュリティ管理 (233 ページ)

Replication Server のセキュリティ機能

Replication Server では、複数の機能を使用してセキュリティを実現しています。

Replication Server では、ログイン名、暗号化、パーミッションを使用してセキュリティを実現しています。

- Replication Server ログイン名
  各 Replication Server には、ログイン名の独自のセットがあります。これらのログイン名は、データ・サーバのログイン名とは異なります。ログイン名がこのように区別されていることによって、複写システム管理者は複写データと複写システムの他の側面を制御できます。
- データ・サーバのログイン名
  データ・サーバのログイン名は、データ・サーバに接続するためにクライアント・アプリケーションで使用されます。クライアントは通常、プライマリ・データを更新するパーミッションを付与されます。ただし、レプリケート・テーブルでは通常、データの選択または表示のパーミッションは付与されますが、データを変更することは許可されません。これらのパーミッションは、アプリケーションに従ってデータ・サーバ内で管理されています。
- データ・サーバのメンテナンス・ユーザ・ログイン名
  Replication Server は、レプリケート・テーブルを含む各ローカル・データ・サーバのデータベースに対して、特別なデータ・サーバの「*メンテナンス・ユーザ*」ログイン名を使用します。これにより、Replication Server は、データベース内のレプリケート・テーブルを管理、更新できます。
- パスワードの暗号化
  複写システムの機密領域では、パスワードを暗号化できます。
- パーミッション・システム
  grant コマンドと **revoke** コマンドを使用して、Replication Server のパーミッションを Replication Server ログイン名に割り当てたり、取り消したりします。

参照：

- Replication Server の役割と責任 (23 ページ)

ネットワークベースのセキュリティ機能

Replication Server は、サードパーティのネットワークベースのセキュリティ・メカニズムをサポートします。

サードパーティのネットワークベースのセキュリティ・メカニズムによって、以下の処理が実行されます。

- ネットワーク上のサーバに統一化ログインを設定。
  セキュリティ・メカニズムは、ログイン時にユーザを認証します。認証された各ユーザは、必要に応じてリモート・サーバに提示できるセキュリティ・クレ

デンシャルを付与されます。その結果、ユーザは 1 つのログイン名を使用して、さまざまなサーバに間断なくアクセスできます。
- ネットワーク上での安全なデータ転送
  次のようなさまざまなデータ保護サービスを選択できます。
  - データ転送の暗号化と復号化
  - 転送が不正に変更されていないことの確認
  - 各転送元の確認
  - 転送が通信傍受されて再送信されていないことの確認
  - 転送が送信された順序どおりに受信されていることの確認

参照：
- ネットワークベース・セキュリティの管理 (262 ページ)

Replication Server の役割と責任

複写システムの管理は、主に複写システム管理者の役割です。データベース管理者は、Replication Server の管理作業の一部を支援することによって補助的な役割を果たします。サイトによっては、役割の区別が明確ではなく、一部の責任が重複していることもあります。

Replication System Administrator (複写システム管理者)

複写システム管理者は、複写システムのインストール、設定、管理を担当します。

WAN では、この役割はさまざまな場所にいるさまざまな担当者によって果たされることがあります。このような場合、Replication Server を管理するためのさまざまな作業で、複写システム管理者間の協力が必要になることもあります。

複写システム管理者は、**sa** ユーザ・パーミッションを持ち、複写システム内でほぼすべてのコマンドを実行できます。システム管理において、複写システム管理者は、ローカル・データベースとリモート・データベースの両方についてデータベース管理者と協力する必要があります。

データベース管理者

データベース管理者は、次の 2 つの作業を担当します。

データベース管理者は、次の作業を担当します。

- ローカル・データ・サーバを管理する (ログイン名とパーミッションの管理など)。

- 分散データベース・システム内のデータを管理する。さまざまな作業において、さまざまなデータベースのデータベース管理者と協力する必要があります。

Replication Server の作業と責任

複写システムを管理するための作業と役割を割り当てるには、Replication Server の作業と責任に関するこの表を使用します。

表 1 : Replication Server の作業と責任

作業	リファレンス	役割
Replication Server のインストールと初期設定。	インストール・ガイドおよび設定ガイド	複写システム管理者 (以下 RSA)、データベース管理者 (以下 DBA)
Replication Server の起動と停止。	複写システムの管理 (83 ページ)	RSA
Replication Server のクワイス。	複写システムの管理 (83 ページ)	RSA、DBA
ログイン名とデータベース・ユーザの追加、および適切なパーミッションの管理。	Replication Server のセキュリティ管理 (233 ページ)	RSA、DBA
Replication Server のモニタリング。	複写システムの管理 (83 ページ)	RSA
Replication Server の設定。	複写システムの管理 (83 ページ)	RSA
複写テーブルとストアド・プロシージャの作成、またはテーブル・スキーマの変更。 • 複写テーブルとストアド・プロシージャの作成と修正。 • テーブルとファンクション複写定義の作成と修正。 • レプリケート・サイトでのサブスクリプションの作成とマテリアライゼーション化。	• 複写テーブルの管理 (299 ページ) • 複写ファンクションの管理 (385 ページ) • サブスクリプションの管理 (405 ページ)	RSA、DBA

作業	リファレンス	役割
データ・サーバのファンクション文字列クラスとファンクション文字列の定義。	『Replication Server 管理ガイド 第 2 巻』の「データベース・オペレーションのカスタマイズ」	RSA、DBA
ルートの管理。 • ルートの作成と修正。	ルートの管理 (161 ページ)	RSA
データベース・コネクションの管理とモニタリング。 • コネクションのサスペンドとレジューム。	データベース・コネクションの管理 (189 ページ)	RSA
ウォーム・スタンバイ・アプリケーションの作成。	『Replication Server 管理ガイド 第 2 巻』の「ウォーム・スタンバイ・アプリケーションの管理」	RSA、DBA
Replication Server のローカライズ。	Replication Server デザイン・ガイド	RSA
プライマリ・データベース・コネクションとレプリケート・データベース・コネクション追加。	データベース・コネクションの管理 (189 ページ)	RSA、DBA
Replication Server の追加または削除。	複写システムの管理 (83 ページ)	RSA
拒否されたトランザクションの処理。	『Replication Server 管理ガイド 第 2 巻』の「エラーと例外の処理」	RSA、DBA
ローカル・データ・サーバの管理。 • データ・サーバのサスペンドとレジューム。	Adaptive Server またはローカル・サーバのマニュアル。	DBA
RSSD の管理。	複写システムの管理 (83 ページ)	RSA、DBA
Embedded Replication Server システム・データベース (ERSSD) の管理。	複写システムの管理 (83 ページ)	RSA、DBA
複写用データベースの作成、削除、修正。	Adaptive Server またはローカル・サーバのマニュアル。	DBA
データベース・ユーザのログイン名とパスワードの設定。	Adaptive Server またはローカル・サーバのマニュアル。	DBA
定期的なバックアップの実行。	『Replication Server 管理ガイド 第 2 巻』の「Replication Server の検証とモニタリング」	DBA

作業	リファレンス	役割
データベースのリカバリ手順の適用。	『Replication Server 管理ガイド 第 2 巻』の「複写システム・リカバリ」	RSA、DBA
データベースの矛盾の調整。	サブスクリプションの管理 (405 ページ)	RSA、DBA
Replication Server で実行されたユーザ・アクションの監査。	コマンド監査 (293 ページ)	RSA

Replication Server の技術的概要

ここでは、Replication Server と複写システムの技術面の概要について説明します。

具体的には以下の項目を取り上げます。

- Replication Server に基づく分散データベース・システムのコンポーネント
- プライマリ・データベースからレプリケート・データベースへのトランザクションの移動
- プライマリ・サイトとレプリケート・サイトでのデータの受信と配信の役割を担う Replication Server の動作
- 複写システムに関連した問題の診断とトラブルシューティング

複写システムのコンポーネント

複写環境は、Replication Server を実行する前に用意するかアセンブルしておく必要のあるコンポーネントとリソースで構成されます。

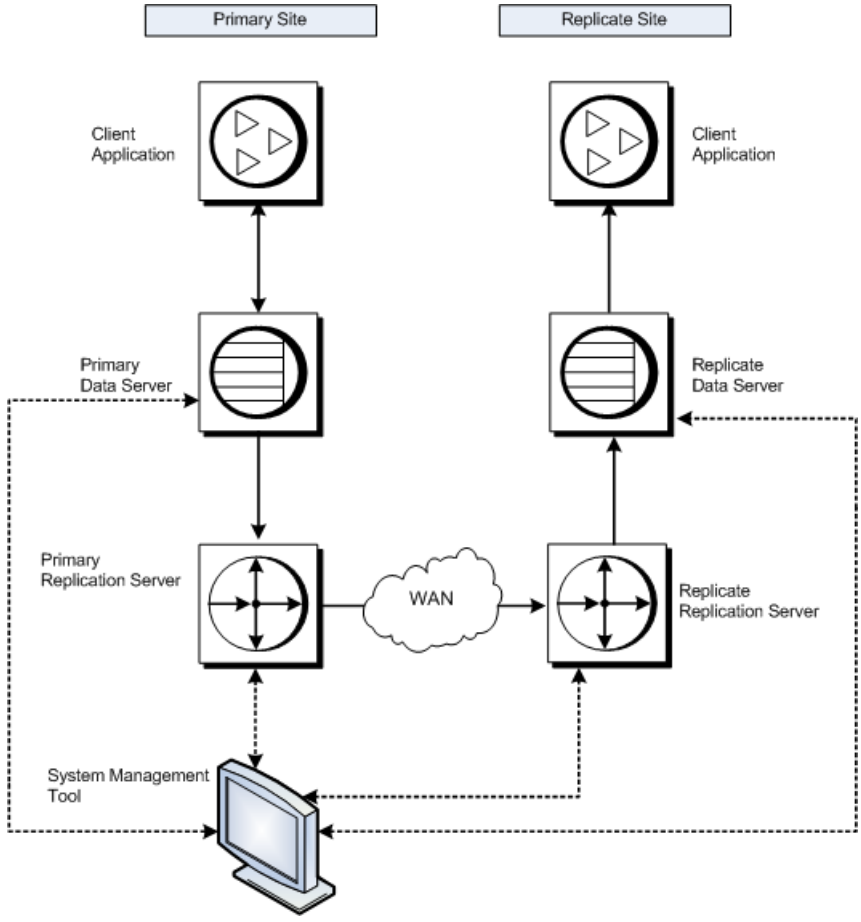
Replication Server 環境のコンポーネントには、次のものがあります。

- Replication Server
- Adaptive Server などのデータ・サーバ
- Replication Server システム・データベース (RSSD)
- Replication Agent、ExpressConnect for Oracle、および ECDA
- クライアント・アプリケーション
- Sybase Control Center や Sybase Central™ などのシステム管理ツール

各コンポーネントは、Open Client/Server™ Interface を使用して他のコンポーネントと通信します。

Replication Server ドメインに関するこの図は、Replication Server をベースとする、WAN ベースの分散データベース・システムの単純な構成を示しています。

図 7 : Replication Server ドメイン



Replication Server

各プライマリ・サイトまたはレプリケート・サイトの Replication Server は、ローカル・データ・サーバのデータ複写アクティビティを調整し、他のサイトの Replication Server とデータを交換します。

Replication Server は、以下の方法で、保証されたトランザクションの配信を各レプリケート・サイトに提供します。

- Replication Agent を介してプライマリ・データベースからトランザクションを受け取り、データのサブスクリプションを持つレプリケート・データベース・サイトに配信する。

- 他の Replication Server からトランザクションを受け取り、ローカルのレプリケート・データベースに適用するか、データに対するサブスクリプションを持つ他の Replication Server に転送する。
- レプリケート・データベースからデータ更新の要求を受信して、プライマリ・データベースに適用する。

これらのタスクの実行に必要な情報は、Replication Server システム・データベースに格納される Replication Server システム・テーブルに格納されています。

Replication Server の内部要素の詳細については、『Replication Server 管理ガイド 第 2 巻』の「パフォーマンス・チューニング」の「Replication Server の内部処理」を参照してください。

参照：

- Replication Server システム・データベース (RSSD) (30 ページ)

ID サーバ

ID サーバは、複写システム内のすべての Replication Server とデータベースを登録している Replication Server です。

ID サーバとして機能する Replication Server は、通常の Replication Server タスクに加え、複写システム内の各 Replication Server とデータベースにユニークな ID 番号を割り当てます。ID サーバはまた、複写システムのバージョン情報を管理します。この点を除けば、ID サーバはその他の Replication Server と変わらない働きをします。

新しい Replication Server や、新しいデータベースを管理する Replication Server がログインして ID 番号を取り出せるようにするには、ID サーバが次の場合に稼働している必要があります。

- Replication Server をインストールする場合
- ルートを作成する場合
- データベース・コネクションを作成または削除する場合

上記の条件があるので、ID サーバは、複写システムをインストールするときに最初にインストールおよび起動する Replication Server になります。Replication Server が 1 つだけの場合や、Replication Server を初めてインストールする場合は、その Replication Server は ID サーバにもなります。既存の複写システムに Replication Server を追加する場合、その複写システムの ID サーバである Replication Server の名前を知る必要があります。

ID サーバには、Replication Server が ID サーバと接続する時に使用する、Replication Server 用のログイン名が必要です。このログイン名は、複写システムを

設定および管理するときに、**rs_init** 設定プログラムによって、複写システム内のすべての Replication Server の設定ファイルに記録されます。

警告！ ID サーバは、複写環境にとって重要なものなので、いったんインストールすると、移動が困難です。いったん ID サーバの名前を決定してしまうと、別の Replication Server へは変更できません。Sybase では、設定ファイルに記録した ID サーバの名前を変更する手順はサポートしていません。

参照：
- 複写システムの管理 (83 ページ)

複写システム・ドメイン

「複写システム・ドメイン」とは、同じ ID サーバを使用するすべての複写システム・コンポーネントを指します。

企業によっては、独立した複数の複写システムを持つところもあります。ID サーバは、複写システム内のメンバ Replication Server およびデータベースを決定するので、複数の複写システムがある場合の各複写システムを ID サーバ・ドメインともいいます。

複数の ID サーバ・ドメインを設定するために特別な作業は必要ありません。どの Replication Server やデータベースも、1 つの複写システム、つまり ID サーバ・ドメインに属し、その ID サーバ・ドメイン内でユニークな ID 番号を持ちます。

次の制限事項のもとで、複数の複写システム・ドメインを設定できます。

- 異なるドメインに属する Replication Server 間では、データを交換できません。各ドメインは、相互に通信できない独立した複写システムとして扱います。異なるドメインに属する Replication Server 間にルートを作成することはできません。
- 1 つのデータベースを管理できるのは、1 つのドメイン内のただ 1 つの Replication Server のみです。どのデータベースも、ただ 1 つの ID サーバのドメイン内に存在します。つまり、異なるドメインから同じデータベースへのコネクションを複数作成することはできません。

参照：
- 複写システム・ドメインの追加 (92 ページ)

Replication Server システム・データベース (RSSD)

RSSD (Replication Server システム・データベース) は、Replication Server システム・テーブルを含むデータベースです。

各 Replication Server には、1 つの Replication Server に対するシステム・テーブルを保持する RSSD または Embedded Replication Server システム・データベース

(ERSSD) が 1 つ必要です。RSSD は Adaptive Server によって管理されます。ERSSD は SQL Anywhere® によって管理されます。

システム・テーブル

Replication Server システム・テーブルには、Replication Server が複写データを送受信するために必要な情報が保持されます。

システム・テーブルには次のような情報が保持されます。

- 複写データと関連情報の記述
- 複写定義やサブスクリプションなどの複写オブジェクトの説明
- Replication Server ユーザ用のセキュリティ・レコード
- 他の Replication Server サイトに対するルート指定情報
- ローカル・データベースのアクセス・メソッド
- その他の管理情報

Replication Server のシステム・テーブルは、Replication Server のインストール時に RSSD にロードされます。

システム・テーブル全体のリストについては、『Replication Server リファレンス・マニュアル』の「Replication Server システム・テーブル」を参照してください。

システム・テーブルの内容は、RCL コマンドや Sybase Central™ プロシージャの実行などの、Replication Server アクティビティのときに変更されます。システム・テーブルを変更できるのは、複写システム管理者と、**rs_systabgroup** グループのメンバだけです。

システム・テーブルに問い合わせてステータス情報を調べるには、次の手順に従います。

- Sybase Central を使用して、複写システムの詳細とプロパティを表示します。
- Replication Server のシステム情報コマンドまたはシステム管理コマンドを使用します。詳細については、『Replication Server リファレンス・マニュアル』の「複写コマンド言語」の「システム情報コマンド」と『Replication Server リファレンス・マニュアル』の「複写コマンド言語」の「システム管理コマンド」を参照してください。
- Adaptive Server ストアド・プロシージャを使用して、複写システムについての情報を表示します。『Replication Server リファレンス・マニュアル』の「Adaptive Server コマンドとシステム・プロシージャ」を参照してください。

警告！ RSSD テーブルは、Replication Server の内部で使用するものです。Sybase 製品の保守契約を結んでいるサポート・センタからの指示がないかぎり、RSSD テーブルを直接変更しないでください。

RSSD と複写エージェントの仕様

Replication Agent は、Replication Server がルートの送信元である場合に、RSSD が必要とします。

ルートの送信元である Replication Server は、その RSSD 内の情報の一部を他の Replication Server に分配します。

RSSD は、RSSD がサポートする Replication Server 専用です。ユーザ・データの格納には使用しないでください。ただし、1 つのデータ・サーバに RSSD とユーザ・データベースを格納することはできます。RSSD 用のデータベース・デバイス領域には、少なくとも 20MB (データ用に 10MB とログ用に 10MB) が必要です。データベースとデータベース・ログは別々のデバイスに置くことをおすすめします。

参照：

- ルートの管理 (161 ページ)

Adaptive Server などのデータ・サーバ

Adaptive Server など、複写システム内のデータ・サーバは、プライマリ・データとレプリケート・データのいずれかが格納されているデータベースを管理します。

クライアント・アプリケーションは、Adaptive Server を使用して、データの格納と取得を行い、トランザクションやクエリを処理します。

各 Replication Server は、その Replication Server システム・データベース (RSSD) 用の Adaptive Server データベースまたはその Embedded Replication Server システム・データベース (ERSSD) 用の SQL Anywhere データベースを必要とし、そのデータベースには Replication Server のシステム・テーブルが含まれています。

Replication Server は、オープン・インタフェースによって ASE 以外のデータ・サーバもサポートしています。必要とされる一連の基本的なデータ・オペレーションとトランザクション処理命令をサポートしていれば、どのようなシステムでもデータの格納に使用できます。プライマリ・データベースを含むデータ・サーバの場合は、互換性のある Replication Agent プログラムと Sybase Enterprise Connect™ Data Access (ECDA) コンポーネントを使用する必要があります。

アクティブにサポートされているデータ・サーバとは、データ・サーバがプライマリまたはレプリケート・データ・サーバとして機能するために必要なすべてのソフトウェア、マニュアル、サポートを Sybase が提供しているデータ・サーバです。

アクティブにサポートされている ASE 以外のデータ・サーバのリストについては、『Replication Agent リリース・ノート』を参照してください。異機種データ・サーバのサポートの詳細については、『Replication Server 異機種間複写ガイド』を参照してください。

Replication Agent

Replication Agent は、他のデータベースにコピーする必要のあるプライマリ・データベース内のアクションを Replication Server に通知します。

Replication Agent は、データベース・トランザクション・ログを読み込んで、複写テーブルとストアド・プロシージャ用のログ・レコードを、そのデータベースを管理している Replication Server に転送します。データベースを管理している Replication Server は、そのデータのサブスクリプションを作成するデータベースに変更内容を配信します。

Replication Agent は、プライマリ・データを含む各データベースと、複写する必要のあるストアド・プロシージャが実行される各データベースに必要です。複写するようにマーク付けされた複写データを含み、ストアド・プロシージャは含まないデータベースの場合は、Replication Agent を必要としません。

Replication Agent は、ログ転送言語 (LTL: Log Transfer Language) のコマンドを実行することによって、Replication Server と通信します。

LTL コマンドの詳細については、『Replication Server デザイン・ガイド』の「Replication Agent の概要」を参照してください。

システムに適した複写エージェント

使用する Replication Agent は、使用している複写システムのデータ・サーバによって異なります。

サポートされる Replication Agent は、次のとおりです。

- RepAgent － Adaptive Server データ・サーバ用の複写エージェントです。Adaptive Server のスレッドである RepAgent は、『Replication Server 管理ガイド』で説明している Replication Agent です。
- ASE 以外のデータ・サーバに使用できる複写エージェントは、次のとおりです。
  - IBM DB2
  - IBM DB2 UDB
  - Microsoft SQL Server
  - Oracle

ExpressConnect for Oracle

ExpressConnect for Oracle は、Replication Server とレプリケート Oracle データ・サーバ間の直接通信を提供します。

Replication Server Options 15.5 以降で使用できる ExpressConnect for Oracle では、ゲートウェイ・サーバを個別にインストールして設定する必要がないため、パフォーマンスが向上し、複写システムを管理する際の複雑性が軽減されます。

詳細については、『Replication Server 異機種間複写ガイド』と『Replication Server Options』のマニュアルを参照してください。

Enterprise Connect Data Access

Sybase Enterprise Connect™ Data Access (ECDA) は、異機種データベース環境にあるデータへのアクセスを可能にするソフトウェア・アプリケーションと接続ツールを統合したセットです。

ECDA を使用すると、LAN ベースの Sybase 以外のデータ・ソースやメインフレームのデータ・ソースにアクセスできるようになります。ECDA は、企業内で透過的なデータ・アクセスを提供するコンポーネントで構成されます。Open Server インタフェースを提供していない、アクティブにサポートされている ASE 以外のデータベースごとに、固有の ECDA コンポーネントが必要になります。これは、ASE 以外のデータベースに複写する場合に必須です。

詳細については、『Replication Server 異機種間複写ガイド』と『Replication Server Options』のマニュアルを参照してください。

クライアント・アプリケーション

「クライアント・アプリケーション」は、データ・サーバにアクセスするプログラムです。

データ・サーバが Adaptive Server である場合、Open Client Client-Library™ か DB-Library™、Embedded SQL™、または PowerBuilder® などの Open Client/Server Interface と互換性のある他のフロントエンド開発ツールで作成されたプログラムが、クライアント・アプリケーションになります。Open Client/Server には、クライアント／サーバ通信用のルーチンとプロトコルが含まれています。

単純な複写システムでは、クライアントがプライマリ・データベースを更新して、Replication Server がレプリケート・データベースを更新します。ストアド・プロシージャを複写することによって、クライアントはどのレプリケート・データベースからでもプライマリ・データを更新できます。

システム管理ツール

複写システムの管理には、Sybase Control Center や Sybase Central™ などのシステム管理ツールを使用できます。

Sybase Control Center for Replication

Sybase® Control Center for Replication は、複写環境内のサーバのステータスと可用性をモニタリングする際に、中間層サーバである Replication Monitoring Services (RMS) に取って代わる、Web ベースのソリューションです。

Sybase Control Center for Replication を使用すると、大規模で複雑、かつ地理的に分散している複写環境のモニタリングと管理を行うことができます。このソリューションにより、サーバとコンポーネント・オブジェクトの検索、並べ替え、フィルタリングを行って、現在の Replication Manager と Replication Monitoring Services では対応しきれない大規模な環境をサポートすることが可能になります。

Sybase Control Center for Replication では、特定のサーバの可用性とステータスを表示するためにサーバ・モニタとヒート・チャートを使用し、ステータス情報を簡単に確認できます。サーバ・モニタには、サーバのバージョンやプラットフォームなど、高レベルの情報が表示されます。また、レプリケーション・パフォーマンスのトラブルシューティングに役立つ重要なパフォーマンス・カウンタも表示されます。

データ・フローの制御とレプリケーション・パラメータの設定を支援してサーバ・パフォーマンスを向上させるため、Sybase Control Center for Replication にはどのレプリケーション・モニタからも簡単にアクセスできるクイック管理ツールが搭載されています。

Sybase Control Center for Replication には、モニタ以外にトポロジ・ビューも用意されており、サーバ、サーバ間コネクション、環境内でのデータ・フロー、およびレプリケーション・パスのソースとターゲットがグラフィカルに表示されます。パフォーマンス・カウンタをモニタリングするためにグラフとチャートも使用できます。

詳細については、Sybase Control Center の「Sybase Control Center for Replication」を参照してください。

Sybase Central

Sybase Central は、Sybase 製品用のグラフィカル管理ツールです。

Sybase のエンタープライズ・マネージメント・ストラテジが実装され、すべてのサーバおよびミドルウェア製品全体にシームレスに統合された 1 つの管理コンソールを呼び出します。Sybase でサポートされるすべてのプラットフォームで実行中の Sybase 製品に接続して管理します。Sybase Central は、Replication Server がサポートしているすべての Windows および UNIX プラットフォームで使用できます。

Sybase Central のプラグインである Replication Manager により、複写環境の開発、管理、モニタリングを行うことができます。

Sybase Central の Replication Manager (RM) プラグイン

Sybase Central の Replication Manager プラグインは、複写環境の開発、管理、モニタリング用の管理ユーティリティです。

従来は RCL コマンドで実行していた次のような管理作業の多くは、Replication Manager では使いやすいインタフェースで実行できます。

- Replication Server オブジェクト (コネクション、ルート、複写定義、サブスクリプションなど) の作成、変更、削除を行う。
- 複写システム・コンポーネントの管理、モニタリング、トラブルシューティングを行う。
- サーバの可用性およびコネクションとルートの状態のモニタリングを行う。
- すべての Replication Server オブジェクトの RCL スクリプトを生成する。ユーザが RCL または SQL をサーバに送信できるスクリプト・エディタ・ウィンドウを提供する。
- Replication Server、Replication Agent、RepAgent スレッド、コネクション、ルートの設定パラメータを含む複写ドメインを管理する。
- コネクションとルートのサスペンドとレジュームによってデータのフローを制御する。
- Replication Server のステーブル・キュー内のトランザクションを表示する。
- Replication Server 例外ログ内のトランザクションを表示し、ユーザがトランザクションを編集して再送信できるようにする。
- ウォーム・スタンバイ環境を管理する。

Replication Monitoring Services (RMS)

Replication Monitoring Services は、10 台以上のサーバを必要とするかなり複雑な複写環境の場合に使用できるモニタリング・ツールです。

RMS を使用すると、環境内のさまざまなサーバとコンポーネントをモニタリングできます。RMS は複写環境で Replication Manager とサーバ間の中間層として動作します。RMS は、データのフローを制御したり、設定パラメータを指定する機能も備えています。

複写用データの指定方法

Replication Server では、リモート・データベースで複写するデータとストアド・プロシージャを定義することも、送信先データベース自体を指定することもできます。

Replication Server は、リレーショナル・データベース・モデルを使用して、固定数のカラムと可変数のローを持つテーブル内のデータを表現します。複写する各

テーブルには、各ローをユニークに識別するために「プライマリ・キー」として使用できる、1 つまたは複数のカラムが必要です。

設計と計画の一環として、複写システム用の送信元データベースと送信先データベースを指定して、複写データが 1 つの Replication Server から別の Replication Server に送信されるルートを作成します。

通常、送信元データベースにはプライマリ・データが含まれ、「プライマリ・データベース」と呼ばれます。一方、送信先データベースにはレプリケート・データが含まれ、「レプリケート・データベース」と呼ばれます。システムの実装方法によっては、1 つのデータベースがプライマリ・データとレプリケート・データの両方を含む場合もあります。トランザクションまたはストアド・プロシージャの実行は、プライマリ・データベースからレプリケート・データベースに複写されます。ストアド・プロシージャの実行は、レプリケート・データベースからプライマリ・データベースに複写されることもあります。

参照：

- Replication Server の基本プライマリ・コピー・モデル (9 ページ)

テーブル用の複写定義とサブスクリプション

各プライマリ・テーブル (送信元テーブル) を記述するために、1 つまたは複数の「複写定義」を作成します。

複写定義には、テーブルのカラムとデータ型、プライマリ・キーを構成するカラム、プライマリ・データのサブスクリプションを作成するために使用できるカラムを示し、テーブルのプライマリ・バージョンのロケーションを指定します。

複写定義には、使用方法をカスタマイズするための追加設定を含めることできます。たとえば、ウォーム・スタンバイ・アプリケーションのスタンバイ・データベースに複写するためだけの複写定義を作成することができます。

次に、複写定義で定義されるデータに対するトランザクション用に、「サブスクリプション」を作成します。サブスクリプションは、すべてのローのトランザクションまたは条件に合うローだけのトランザクションをコピーするように Replication Server に指示します。テーブルのコピーは、必要なローまたはカラムだけに限定できます。

通常は、サブスクリプションを作成すると、Replication Server はプライマリ・データベースからレプリケート・データベースに、要求された最初のデータをコピーします。このプロセスを「サブスクリプション・マテリアライゼーション」と呼びます。サブスクリプションが作成されてマテリアライズされると、Replication Server はプライマリ・データに対するデータベース・オペレーションが発生するたびに、その分配を開始します。

参照：
- 複写テーブルの管理 (299 ページ)
- サブスクリプションの管理 (405 ページ)

データベース・オブジェクト用の複写定義

Multi-Site Availability (MSA) を使用すると、レプリケート・データベースに送信するデータを記述する 1 つのデータベース複写定義を作成できます。

データベース複写定義には、複写するデータベース・オブジェクトを記述します。個々のテーブル、トランザクション、ファンクション、システム・ストアド・プロシージャ、DDL について、複写するかしないかを選択できます。

次に、サブスクリプションを作成する各データベースで、データベース複写定義に記述されているデータ用の「データベース・サブスクリプション」を 1 つ作成します。データベース・サブスクリプションでは、コピー対象のデータを限定できません。

MSA は、プライマリ・データベースの複写定義およびサブスクリプションを作成するデータベースごとのサブスクリプションを 1 つしか必要としない簡単な複写方法を提供します。データの変換が必要な場合、または、ターゲット・コネクションとは異なる **replicate minimal columns** 設定または動的 SQL 設定がテーブルに必要な場合は、テーブル複写定義とファンクション複写定義を追加する必要があります。

参照：
- MSA を使用した複写オブジェクトの管理 (463 ページ)

ストアド・プロシージャ用の複写定義

ストアド・プロシージャの複写定義は、「ファンクション複写定義」と呼びます。

オペレーションによっては、ストアド・プロシージャを複写する方が、テーブルを複写するよりも大幅にパフォーマンスが向上することがあります。また、ストアド・プロシージャを複写すると、レプリケート・データベースからプライマリ・データベースにデータを更新できます。

参照：
- 遅延名前解決 (146 ページ)
- 複写ファンクションの管理 (385 ページ)
- サブスクリプションの管理 (405 ページ)

通常の複写での複写ファンクションの利点

複写ファンクションの利点について説明します。

Adaptive Server は、Transact-SQL® コマンドによって変更される各ローのレコードをログに記録しています。1 つの Transact-SQL コマンドで複数のローが変更される場合、Replication Server は Replication Agent から受け取った各ログ・レコードを、そのトランザクション内で別々のコマンドとして処理します。たとえば、プライマリ・データベース内の 1000 のローを変更する 1 つの **update** コマンドの結果を複写するために、Replication Server は各レプリケート・データベースで、1000 の **update** コマンドを実行します。

多数のローを変更するコマンドを実行すると、レプリケート Adaptive Server と複写システムのパフォーマンスに影響することがあります。複写システムを通じて配信される大量のローにより、ステーブル・キューの空き領域を使い尽くしてしまうことがあります。

アプリケーションが、プライマリ・テーブル内の複数のローを更新する場合、複写ストアド・プロシージャを使用して、送信先データベース内のデータを管理できます。ストアド・プロシージャ内のコマンドは複数のローを変更できるので、ストアド・プロシージャを使用すると、複写システムを介してローのイメージを渡さなくてもレプリケート・データベース内のローを更新できます。ストアド・プロシージャの実行とそのパラメータを反映するただ 1 つのレコードが、複写システムを介して複写されます。

SQL 文の複写を使用して、複数のローを変更する単一の Transact-SQL コマンドのパフォーマンスを向上させることができます。『Replication Server 管理ガイド 第 2 巻』の「パフォーマンス・チューニング」の「SQL 文の複写」を参照してください。

Replication Server では、Adaptive Server の遅延名前解決のサポートを有効にできます。

参照：

- 複写ファンクションの管理 (385 ページ)
- サブスクリプションの管理 (405 ページ)
- 遅延名前解決 (146 ページ)

複写ファンクションの使用

ファンクション複写定義は複写ストアド・プロシージャを記述します。

ファンクション複写定義には、次の項目が含まれます。

- パラメータとデータ型

- ストアド・プロシージャが変更できるプライマリ・データのロケーション
- ストアド・プロシージャ実行のサブスクリプションを作成するのに使用できるパラメータ
- 送信先データベースで実行するストアド・プロシージャの名前

複写ファンクションの配信には、2 つのタイプがあります。

- 適用 ― 最初にプライマリ・データベースで実行されて、プライマリ・データに影響します。Replication Server は、ストアド・プロシージャとそのパラメータを送信して、適用ファンクション複写定義用のサブスクリプションのあるレプリケート・サイトにデータ変更を非同期に適用します。メンテナンス・ユーザは、レプリケート・サイトで適用ファンクションを実行します。
- 要求 ― 最初にプライマリ・データベースで実行されて、プライマリ・データに影響します。Replication Server は、ストアド・プロシージャとそのパラメータを送信して、要求ファンクション複写定義用のサブスクリプションのあるレプリケート・サイトにデータ変更を非同期に適用します。プライマリ・データベースでストアド・プロシージャを実行するユーザーによって、レプリケート・サイトで要求ファンクションが実行されます。

  通常、要求ファンクションの配信は、他のサイトにあるデータベースのリモート・データを非同期に変更するために使用されます。変更内容は、通常のデータ複写または適用ファンクション配信のどちらかによって、要求元のサイトに複写されます。

参照：
- 複写ファンクションの管理 (385 ページ)
- サブスクリプションの管理 (405 ページ)

パブリケーション

「パブリケーション」は、関連するテーブルとストアド・プロシージャに対する複写定義をまとめて、1 つのグループとしてサブスクリプションを作成するために使用できます。

パブリケーションは、プライマリ Replication Server で作成して、送信先 Replication Server でサブスクリプションを作成します。

パブリケーションを使用する場合は、次のオブジェクトを作成して管理します。

アーティクル ― 複写定義、プライマリ・データベース、パブリケーションを特定します。レプリケート・データベースに送信するローまたはパラメータの数を限定することもできます。

パブリケーション ― プライマリ・データベースのアーティクルの集まりです。

パブリケーション・サブスクリプション ― パブリケーションに対するサブスクリプションです。パブリケーション・サブスクリプションを作成すると、

Replication Server は、パブリケーションの各アーティクルのサブスクリプションを作成します。

パブリケーションによって、システムが理解できる形式で複写定義とサブスクリプションをグループ化できます。また、一連のテーブルとプロシージャ用に、ただ 1 つのサブスクリプションのステータスを作成して確認することができます。

テーブル複写の概要

ここでは、プライマリ (複写元) テーブルからレプリケート (複写先) テーブルにトランザクション・データを複写するために複写システム内のデータ・サーバと Replication Server で実行する必要のある作業について説明します。

1. レプリケート・データ・サーバでの作業 ― プライマリ・テーブルからデータが複写されるテーブルのコピーを作成します。コピーには、プライマリ・テーブルのカラムの全部または一部を含むことができます。
2. プライマリ Replication Server での作業 ― 複写するテーブル・データを識別する複写定義を作成します。異なるレプリケート・データベースに複写されるテーブルごとに 1 つまたは複数の複写定義を作成できます。ストアド・プロシージャ用の複写定義も作成できます。
   複写定義の作成が完了すると、トランザクションが、複写定義のサブスクリプション作成の条件を満たしている送信先 Replication Server に複写するために使用できるようになります。
   複写定義を参照する一連のアーティクルを作成して、それらを 1 つのパブリケーションにグループ化できます。レプリケート・データベースに送信されるトランザクションが、特定のローだけに作用するように限定するには、アーティクルに **where** 句を使用します。
3. プライマリ・データ・サーバで、次の操作を行います。**sp_setreptable** システム・プロシージャを使用して、Adaptive Sever がプライマリ・データ・サーバである場合にテーブルを複写するようマーク付けします。Replication Agent で異なるデータ・ソースを使用する場合は、プライマリ・オブジェクトを複写するようにマーク付けする方法について、Replication Agent のマニュアルを参照してください。
   プライマリ・データ・サーバのテーブルを複写するようにマーク付けすると、プライマリ・データベースの Replication Agent は、そのテーブルのトランザクションをプライマリ Replication Server に転送できます。

   `text`、`unitext`、または `image` カラムを複写する場合は、**sp_setrepcol** システム・プロシージャの使用が必要な場合もあります。
4. レプリケート Replication Server での作業 ― プライマリ Replication Server で作成された複写定義用のサブスクリプションを作成します。サブスクリプションによって、レプリケート・テーブルは、「マテリアライゼーション」と呼ばれるプロセスを通じてプライマリ・テーブルから最初のデータを受信し、後続の複写データ更新の受信を開始できます。

各複写定義には複数のサブスクリプションを作成できますが、1 つのレプリケート・テーブルがサブスクリプションを作成できるのは、1 つの複写定義だけです。サブスクリプションを設定すると、レプリケート・テーブルのトランザクションをすべて受信できます。また、**where** 句を使用して、特定のローに作用するトランザクションだけを受信することもできます。
プライマリ Replication Server で作成されたパブリケーション用のパブリケーション・サブスクリプションを作成します。そうすると、Replication Server は、パブリケーション内の各アーティクルに対して 1 つのアーティクル・サブスクリプションを作成します。
サブスクリプションを作成すると、データ複写のプロセスは完了します。

参照：
- 複写対象テーブルへのマーク付け (327 ページ)
- 複写ファンクションの管理 (385 ページ)
- サブスクリプションの管理 (405 ページ)
- 複写システム例でのテーブルの複写 (440 ページ)

複写データの管理用コマンド

Replication Manager プラグインのオンライン・ヘルプには、Sybase Central のテーブル複写定義、ファンクション複写定義、サブスクリプションに関する作業と概念が示されています。Replication Server には、複写データを管理するためのテーブル複写定義、ファンクション複写定義、パブリケーション、サブスクリプション、パブリケーション・サブスクリプション RCL コマンドなども用意されています。

参照：
- テーブル複写定義を管理するためのコマンド (306 ページ)
- ファンクション複写定義を管理するコマンド (389 ページ)
- パブリケーションを作成して管理するためのコマンド (369 ページ)
- サブスクリプション・コマンド (426 ページ)
- パブリケーション・サブスクリプションを作成して管理するためのコマンド (453 ページ)

Replication Server コネクションの確立

Replication Server は、Open Client/Server Interface を使用して、クライアント・アプリケーションとサーバ間で通信します。

Replication Server、Adaptive Server、その他のデータ・サーバ用のゲートウェイ・ソフトウェアなどのサーバ・プログラムは、interfaces ファイルまたは LDAP

(Lightweight Directory Access Protocol) サーバのいずれかのディレクトリ・サービスに登録されているので、クライアント・アプリケーションとその他のサーバ・プログラムは、それらを見つけることができます。

注意： ネットワークベース・セキュリティを使用している場合、ネットワーク・セキュリティ・メカニズムのディレクトリ・サービスを使用して、Replication Server、Adaptive Server、ゲートウェイ・ソフトウェアを登録してください。詳細については、ネットワークベースのセキュリティ・メカニズムに添付されているマニュアルを参照してください。

Interfaces ファイル

interfaces ファイルには、Replication Server やデータ・サーバなどの、複写システムのサーバ用のネットワーク定義が記述されています。

通常は、各サイトの 1 つの interfaces ファイルに、すべてのローカルおよびリモート Replication Server とデータ・サーバ用のエントリが含まれます。各サーバのエントリには、他のサーバとクライアント・プログラムの接続に必要なユニークな名前とネットワーク情報が含まれます。サイトの interfaces ファイルには、次のコンポーネントのエントリが必要です。

- ID サーバ (Replication Server が ID サーバではない場合)
- Replication Server
- この Replication Server の RSSD Adaptive Server または ERSSD SQL Anywhere
- ERSSD Replication Agent (現在のサイトからルートを作成した場合)
- この Replication Server で管理されるデータベースを持つデータ・サーバ
- RSSD など、Adaptive Server データベースをバックアップする Backup Server
- このサイトに複写されるプライマリ・データを含むデータベースを管理する他のサイトにある Replication Server
- このサイトで管理されるプライマリ・データ用のサブスクリプションを持つ他のサイトにある Replication Server
- この Replication Server が、中間 Replication Server のないルートを持つ他の Replication Server

Replication Server を起動するときには、デフォルトの interfaces ファイルを使用するか、コマンド・ラインで別の interfaces ファイルを指定できます。interfaces ファイルは、通常は Sybase のリリース・ディレクトリにあります。interfaces ファイルを修正するには、テキスト・エディタを使用します。詳細については、使用しているプラットフォームの『Replication Server インストール・ガイド』および『Replication Server 設定ガイド』を参照してください。

LDAP サーバ

LDAP サーバは、サーバ名やコネクション・プロパティなどのコンポーネント情報を共有するための、グローバルなディレクトリ・サービスを提供しています。

LDAP ディレクトリ・サービスを使用すると、ネットワークベースのシステム内でコンポーネントによるディレクトリ情報の検索ができます。LDAP サービスまたはゲートウェイは、どのタイプでも LDAP サーバです。LDAP ドライバは、LDAP クライアント・ライブラリを呼び出して LDAP サーバへのコネクションを確立します。LDAP ドライバとクライアント・ライブラリは、クライアントとサーバの間の通信プロトコルと交換されるメッセージの内容を定義します。LDAP は、TCP (Transmission Control Protocol) で直接実行されます。

LDAP ドライバが LDAP サーバに接続すると、LDAP サーバは次の 2 つの認証モデルのいずれかに基づいてコネクションを確立します。

- 匿名アクセス ― 認証情報は不要で、通常は読み込み専用の権限で使用されます。
- ユーザ名とパスワードによるアクセス ― Replication Server へのアクセスに使用するユーザ名とパスワードとは異なります。

Replication Server は、LDAP URL に対する拡張機能としてアクセス情報を使用します。アクセス情報は、次のファイルから取得されます。

- UNIX – `$SYBASE/$SYBASE_OCS/config/libtcl.cfg`
- `Windows – %SYBASE%¥%SYBASE_OCS%¥ini¥libtcl.cfg`

Replication Server は、サーバ名と接続プロパティを認証する目的でのみ LDAP を使用します。LDAP サービスは Replication Server の interfaces ファイルの代わりに使用できます。Replication Server ユーザ認証のために LDAP を使用することはできません。

Replication Server は、Open Client/Server ライブラリを使用して LDAP サーバに接続し、Open Client/Server 設定とプロシージャを使用して LDAP サービスを設定および管理します。LDAP ディレクトリの設定の詳細については、使用しているプラットフォームの『Replication Server 設定ガイド』を参照してください。Open Client/Server の LDAP サポートの詳細については、『Open Client Client-Library/C リファレンス・マニュアル』を参照してください。

Replication Server コネクションの確立

LAN または WAN 上のサイトのデータ・サーバと Replication Server を接続するには、各サイトの複写システム管理者がコネクションとルートを定義します。

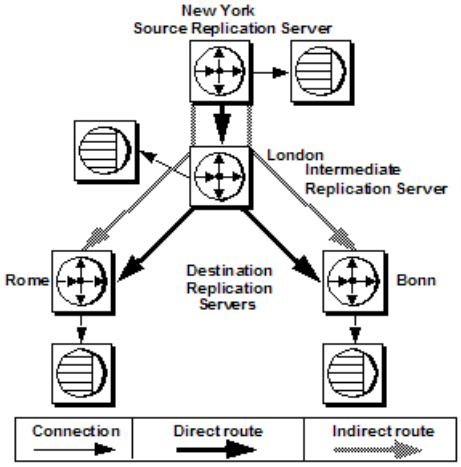
コネクションとルートの編成は、複写の計画における基本事項です。設定するコネクションとルートによって、必要な Replication Server コンポーネントの数が決

まります。さらに、送信元データベースと送信先データベース間の複写のマップ方法は、システムのパフォーマンスとデータの可用性に影響します。

データのコピー先を指定するには、システム内の Replication Server 間、および Replication Server とデータベース間に、次のパスまたはメッセージ・ストリームを作成する必要があります。

- Replication Server からデータベースへのコネクション
  Replication Server はプライマリ・データベースから受け取ったトランザクションを、自ら管理するレプリケート・データベースへのコネクションを通じて分配します。Replication Server は、複数のデータベースへのコネクションを持つことがありますが、各データベースが Replication Server からのコネクションとして持つことができるのは 1 つだけです。
  ウォーム・スタンバイ・アプリケーションも「論理コネクション」を使用します。これは、データベースとそのスタンバイ・データベースの両方を表します。
- Replication Server から別の Replication Server へのルート
  プライマリ・データを含んでいるデータベースを管理する各送信元 Replication Server から、データのサブスクリプションを作成する各送信先 Replication Server へのルートを指定する必要があります。
  送信元 Replication Server から送信先 Replication Server までは、直接のルートを指定できます。また、送信元 Replication Server と送信先 Replication Server との間に中間 Replication Server を持つ間接ルートを指定することもできます。

図 8 : ルートとコネクション



この図は、ヨーロッパに複数の拠点を持つ企業の例を示しています。ニューヨークの Replication Server は、ヨーロッパ向けの情報をすべて、ロンドンの Replication Server を介してルート指定します。この方法によって、ニューヨークの Replication

Server が作成する直接のコネクション数が減り、WAN のトラフィックが減ります。データは、ニューヨークからヨーロッパの各拠点にではなく、ニューヨークからロンドンに一度だけ送信されます。ロンドンの Replication Server は、その複写データを他のヨーロッパの拠点に配信します。

複写システムのルートとコネクションの設計方法の詳細と規則については、『Replication Server デザイン・ガイド』を参照してください。

論理コネクションの詳細については、『Replication Server 管理ガイド 第 2 巻』の「ウォーム・スタンバイ・アプリケーションの管理」を参照してください。

参照：

- ルートの管理 (161 ページ)
- データベース・コネクションの管理 (189 ページ)

データベース・オペレーションの指定

Replication Server は、データベース・オペレーションを、プライマリ・データベースから送信先 Replication Server に、名前と一連のデータ・パラメータで構成される「ファンクション」として配信します。

送信先 Replication Server は次に、「ファンクション文字列」を使用して、送信先データ・サーバによって認識されるコマンドにファンクションをマップします。これらのコマンドは、トランザクション制御命令 (**begin transaction** または **commit transaction**)、またはデータ操作命令 (**insert**、**update**、または **delete**) を表します。ファンクション文字列は、ファンクションをデータ・サーバ固有のコマンドに変換するテンプレートまたはメタコマンドとして機能します。ファンクション文字列を使用すると、プライマリ・サイトはデータを複数の異機種データ・サーバに複写できます。ファンクション文字列は、データ・サーバのタイプによって「ファンクション文字列クラス」に分類されます。

たとえば、プライマリ Replication Server は、**rs_insert** ファンクションを送信先 Replication Server に送信します。送信先 Replication Server は、データベースが Adaptive Server、DB2、またはその他のデータベースのどれであっても、適切なファンクション文字列を使用して、受け取ったファンクションをそのサイトで使用しているデータ・サーバに固有の挿入コマンドに変換します。

ファンクションには次の 2 つのタイプがあります。

- システム・ファンクション ― Replication Server によって提供される、または複写システムに新しいデータベースをインストールすると使用可能になる、ファンクション文字列を持つデータ・サーバ・オペレーションを表します。

- ユーザ定義ファンクション ― Replication Server アプリケーションをカスタマイズして、ストアド・プロシージャを配信できるようにします。

詳細については、『Replication Server 管理ガイド 第 2 巻』の「データベース・オペレーションのカスタマイズ」を参照してください。

ファンクション文字列

ファンクション用のファンクション文字列は、Replication Server に付属しているファンクション文字列クラス用に自動的に生成されます。

ファンクション文字列は、提供されるクラスのどれからもそのファンクション文字列を継承しない、ユーザ作成のファンクション文字列クラス用にカスタマイズする必要があります。ファンクション文字列をカスタマイズするには、データ・サーバ固有のコマンドを使用するか、リモート・プロシージャ・コール (RPC: Remote Procedure Call) を呼び出して、既存のファンクション文字列を修正します。カスタマイズされたファンクション文字列には、カラムの値、プロシージャ・パラメータ、システムで定義されている情報、ユーザ定義の変数を表す「ファンクション文字列変数」を含めることができます。Replication Server は、この変数を実際の値に置き換えてから、ファンクション文字列をデータ・サーバに送信します。

詳細については、『Replication Server 管理ガイド 第 2 巻』の「データベース・オペレーションのカスタマイズ」を参照してください。

ファンクション文字列クラス

ファンクション文字列クラスは、データベースで使用されるすべてのファンクション文字列で構成されています。

Replication Server では、Adaptive Server 用のファンクション文字列クラスを提供しています。また、Replication Server バージョン 15.2 では、Sybase® IQ、IBM DB2 UDB、Microsoft SQL Server、Oracle など、ASE 以外のデータ・サーバ用のファンクション文字列クラスを提供しています。ファンクション文字列は、データ・サーバ固有の命令を含むことがありますが、同じデータ・サーバ・タイプによって管理される複数のデータベースで使用されることもよくあります。すべて新しいファンクション文字列で構成されるクラスを作成することも、既存の「親クラス」からファンクション文字列を継承する「派生クラス」を作成することもできます。

Replication Server でのトランザクション処理

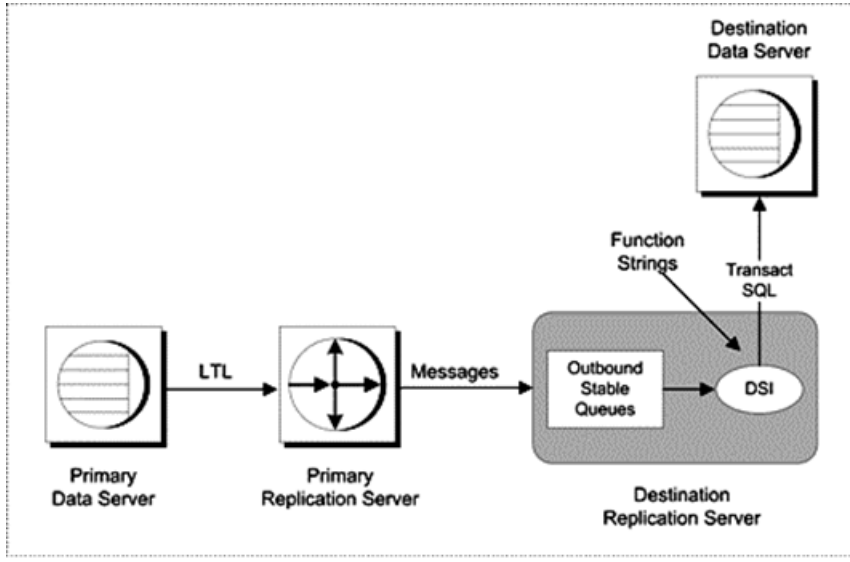
Replication Server では、データ・サーバの格納データを保護するために必要なトランザクション処理サービスの提供は、データ・サーバに依存しています。

配信されるデータの整合性を保証するために、データ・サーバは次のトランザクション処理規則に従う必要があります。

- 1 つのトランザクションが、1 つの作業単位になります。トランザクション内のオペレーションは、すべて実行されるか、まったく実行されないかのいずれかです。
- トランザクションの結果は永続的です。トランザクションは、コミットされたあとに取り消すことはできません。

Replication Server は、コミットされたトランザクションをプライマリ・サイトからレプリケート・サイトにコピーし、コピーされたデータがプライマリ・データと同じ状態で渡されるように、トランザクションをコミットされた順序どおりに配信します。

この図は、Replication Server でのトランザクションの変換方法を示しています。



プライマリ Replication Server が、サブスクリプションを作成するサイトにトランザクションを送信すると、送信先 Replication Server はアウトバウンド・データ・サーバ・インタフェース (DSI: Data Server Interface) ステーブル・キューにトランザクションを格納します。

ステーブル・キュー

Replication Server をインストールするときに、Replication Server が「ステーブル・キュー」を設定するために使用するディスク・パーティションを設定します。複写オペレーションの間に、Replication Server はこれらのキューに更新内容を一時的に格納します。

ステーブル・キューには 3 つのタイプがあり、次のようにそれぞれ異なるタイプのデータが格納されます。

- インバウンド・キュー ― Replication Agent からのメッセージだけを保持します。追加するデータベースがプライマリ・データを含んでいる場合、または要求ストアド・プロシージャが非同期配信用のデータベースで実行される予定の場合は、Replication Server はインバウンド・キューを作成して、そのデータベースに対する Replication Agent からのメッセージを受け入れる準備をします。
- アウトバウンド・キュー ― レプリケート・データベースまたはレプリケート Replication Server 用のメッセージを保持します。次の各送信先に対してアウトバウンド・キューが 1 つずつあります。
  - Replication Server によって管理される各レプリケート・データベースに対して、1 つの DSI (データ・サーバ・インターフェイス) アウトバウンド・キューがあります。
  - Replication Server がルートを持っている各 Replication Server に対して、1 つの Replication Server インタフェース (RSI: Replication Server Interface) アウトバウンド・キューがあります。
- サブスクリプション・マテリアライゼーション・キュー ― 新しく作成されるサブスクリプション、または削除されるサブスクリプションに関連するメッセージを保持します。このキューは、サブスクリプション・マテリアライゼーション中のプライマリ・データベースから、またはマテリアライゼーション解除中のレプリケート・データベースから、有効なトランザクションの「スナップショット」を格納します。

複写システムでステーブル・キューの領域がさらに必要な場合は、後からパーティションを追加できます。

トラブルシューティングのためにキューの内容を調べる方法については、『Replication Server トラブルシューティング・ガイド』を参照してください。

参照：

- ステーブル・キューのパーティション (51 ページ)

キューの管理

Replication Server は、各ステーブル・キューの管理にステーブル・キュー・マネージャ (SQM) スレッドを使用します。

スレッドは、メッセージの受信などの特定のタスクを管理するサブプロセスです。一部のキューには、追加のステーブル・キュー・トランザクション (SQT: Stable Queue Transaction) スレッドを持つものもあります。SQM および SQT スレッドの詳細については、『Replication Server 管理ガイド 第 2 巻』の「パフォーマンス・チューニング」の「Replication Server の内部処理」で「プライマリ Replication Server での処理」を参照してください。

トランザクションをステーブル・キューから出す準備ができると、次のスレッドのいずれかが、キュー内のトランザクションを送信します。

- 「データ・サーバ・インタフェース」 (DSI) スレッド ― データ・サーバとのコネクションを管理します。
- 「Replication Server インタフェース」(RSI) スレッド ― レプリケート Replication Server とのコネクションを管理します。

強化されたキューの **dump** コマンドにより、ステーブル・キューの識別、ダンプするステーブル・キューの内容の制御、追加の出力ファイル・オプションのサポートを柔軟に行うことができます。また、Replication Server には、特定のトランザクションの削除とリストアを SQM から行うための新しいコマンドも用意されています。

ステーブル・キューを管理するコマンド

Replication Server には、ステーブル・キューを管理するための RCL コマンドが用意されています。

- **sysadmin dump_queue**
- **sysadmin sqt_dump_queue**
- **resume connection**
- **sysadmin log_first_tran**
- **sysadmin sqm_zap_tran**
- **sysadmin sqm_unzap_tran**
- **sysadmin dump_tran**

これらのコマンドの詳細については、『Replication Server リファレンス・マニュアル』を参照してください。

DSI スレッド

DSI スレッドは、送信先データベースに割り当てられているファンクション文字列クラス内のファンクション文字列で指定されているように、トランザクションの変更内容を RPC または言語に変換します。

Replication Server は、DSI スレッドを起動して、コネクションを持っているレプリケート・データベースにトランザクションを送信します。

DSI スレッドは、次のタスクを実行します。

- 小さなトランザクションをコミット順にグループ化する。
- データベース・コネクションに割り当てられたファンクション文字列クラスに従って、ファンクションをファンクション文字列にマップする。
- レプリケート・データベースのトランザクションを実行する。
- 割り当てられたエラー・アクションに従って、データ・サーバから返されるエラーに対してアクションを実行し、失敗したトランザクションがある場合は、例外ログに記録する。

Replication Server からレプリケート・データベースにトランザクションを送信するときのパフォーマンスを向上させるには、複数の DSI スレッドを使用してトラン

ザクションが適用されるように、データベース・コネクションを設定します。この機能の詳細については、『Replication Server 管理ガイド 第 2 巻』の「パフォーマンス・チューニング」で「並列 DSI スレッドの使用」を参照してください。

Replication Server バージョン 15.2 では、Adaptive Server® Enterprise 12.0 以降で、大量の **insert** 文を同じテーブルで複写するときのパフォーマンスを向上するバルク・コピー・インのサポートが導入されています。Replication Server は、Open Client™ Open Server™ Bulk-Library を使用して、バルク・コピー・インを実装します。詳細については、『Replication Server 管理ガイド第 2 巻』の「パフォーマンス・チューニング」で「DSI バルク・コピー・イン」を参照してください。

DSI スレッドは、Replication Server によってサポートされるすべてのデータ・ソースのトランザクションの混合を適用する場合があります。これらのトランザクションは、送信先データ・サーバの単一のアウトバウンド・ステーブル・キューで処理されます。

RSI スレッド

RSI スレッドは、1 つの Replication Server から別の Replication Server にメッセージを送信します。各送信先 Replication Server に対して、1 つの RSI スレッドが存在します。

プライマリ Replication Server がトランザクションを処理すると、他の Replication Server 宛てのトランザクションが RSI アウトバウンド・キューに書き込まれます。RSI スレッドは、各送信先 Replication Server にログインし、ステーブル・キューから送信先 Replication Server にメッセージを転送します。

ある Replication Server から別の Replication Server に直接ルートが作成されると、送信元 Replication Server の RSI スレッドは、送信先 Replication Server にログインします。間接ルートが作成される場合、Replication Server は、新しいステーブル・キューと RSI スレッドを作成しません。間接ルートに対するメッセージは、直接ルートに対する RSI スレッドによって処理されます。

参照：

- Replication Server コネクションの確立 (42 ページ)

ステーブル・キューのパーティション

Replication Server は、データ・サーバまたは他のサイトに向けられたメッセージをパーティションに格納します。

Replication Server は、ステーブル・デバイスの領域をパーティションに分けて割り当てて、さらにそのパーティションをセグメントとブロックに分割します。各ステーブル・キューには、別の Replication Server またはデータベースに配信されるメッセージが保持されます。

rs_init プログラムは、初期パーティションを Replication Server に割り当てます。**rs_init** でのパーティション操作の詳細については、『Replication Server インストール・ガイド』と『Replication Server 設定ガイド』を参照してください。

初期パーティションの最小値は、20MB です。Replication Server が管理するデータベースの数と Replication Server がメッセージを配信するリモート・サイトの数によっては、パーティションの追加が必要になる場合があります。サブスクリプションが開始される場合や長時間実行されるトランザクションがある場合は、さらに大きなパーティションが必要になることもあります。

Replication Server は、さまざまなサイズのパーティションをいくつでも持つことができます。パーティション・サイズの合計が、Replication Server がキューに格納できるトランザクションの容量です。

create partition コマンドを使用して追加のパーティションを割り当てるか、**alter partition** を使用してパーティションを拡大します。詳細については、『Replication Server リファレンス・マニュアル』を参照してください。

Replication Server 用にパーティションを選択する場合は、次のガイドラインに従います。

- Replication Server のパーティションには、オペレーティング・システムのロー・パーティションの使用をお勧めします。
- オペレーティング・システムが使用する目的で、パーティションをマウントしないでください。
- ファイル・システムの格納や、スワップ領域用、または Adaptive Server デバイス用など、他の目的にパーティションを使用しないでください。
- パーティション全体を Replication Server に割り付けます。パーティションの一部だけを Replication Server に割り付けた場合でも、残りの部分を他の目的に使用することはできません。
- パーティションに対する読み込み／書き込みのパーミッションは、Replication Server を起動するユーザ以外のどのユーザにも付与しないでください。
- **configure replication server** を使用して設定できる **sqm_async_seg_delete** パラメータはデフォルトで on なので、バージョン 15.7 以降にアップグレードする場合は Replication Server に大規模なパーティションが必要になる可能性があります。次を参照してください。
  - 『Replication Server 設定ガイド』の「Replication Server のインストールと設定の準備」の「複写システムのプラン作成」の「各 Replication Server の最初のディスク・パーティション」
  - 『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**sqm_async_seg_delete**」と「**alter partition**」

Replication Server がパーティションにキュー・セグメントを割り付ける方法を選択するか、またはデフォルトのメカニズムをそのまま使用できます。デフォルトの

メカニズムでは、順番に並べられたリスト内の次のパーティションにキュー・セグメントを割り付けます。別の割り付けメカニズムを選択するには、**alter connection** コマンドまたは **alter route** コマンドを使用します。詳細については、『Replication Server 管理ガイド 第 2 巻』の「パフォーマンス・チューニング」で「Allocate Queue Segments」を参照してください。

ステーブル・キューに対するディスク・ファイルの使い方

パーティション用にディスク・ファイルを使用するには、ファイルを作成してから、**create partition** コマンドを実行します。

空のファイルを作成して、Replication Server がそのファイルに対して読み書きできるようにパーミッションを設定できます。Replication Server は、ユーザが指定するサイズにそのファイルを拡張します。

パーティションは、ロー・ディスク・パーティション (こちらを使用することをおすすめします)、またはオペレーティング・システム・ファイルのどちらでもかまいません。ロー・ディスク・パーティションに対するディスク書き込みは、オペレーティング・システムによってバッファに入れられないので、選択できる場合には、ロー・ディスク・パーティションを使用する方が最適なリカバリ性を得られます。

分散同時制御

プライマリ・データを格納するデータ・サーバは、分散データベース・システムで必要とされるほとんどの同時制御機能を提供します。トランザクションが、プライマリ・バージョンのテーブルの更新に失敗すると、プライマリ Replication Server はその更新内容を他のサイトに分配しません。

トランザクションがプライマリ・データの更新に成功すると、Replication Server はその変更内容を分配します。障害が発生しないかぎり、そのデータのサブスクリプションを持つすべてのサイトで更新は正常に実行されます。

複数のデータベース内のデータを変更するトランザクション

複数のデータ・サーバ内のプライマリ・データを修正するトランザクションは、さらに別の同時制御を必要とすることがあります。

トランザクション処理の必要条件は、トランザクション内のすべてのオペレーションが実行されるか、どれも実行されないかのいずれかです。トランザクションが 1 つのデータ・サーバで失敗した場合、そのトランザクションで更新された他のすべてのデータ・サーバでロールバックされなければなりません。

マルチデータベース・トランザクションを複写する場合は、各データベースに対する更新は、1 つのデータベースに 1 つの Replication Agent があるので、それぞれ独立したトランザクションとしてレプリケート・データベースに送信されます。

レプリケート・テーブルの更新の失敗

プライマリ・データを修正しても、送信先サイトでデータのコピーの更新に失敗することがあります。

プライマリ・バージョンは「公式」なコピーであり、そこで成功した更新は、コピーの送信先サイトでも成功する必要があります。更新に失敗する場合、次のいずれかの理由によると考えられます。

- システム・リカバリのあとにレプリケート・バージョンとプライマリ・バージョンの同期がとれておらず、データの消失が検出された。
  詳細については、『Replication Server 管理ガイド 第 2 巻』の「複写システム・リカバリ」を参照してください。
- テーブルのコピーを格納するデータ・サーバが、プライマリ・バージョンを格納するデータ・サーバによる要求を受け付けないように制限されている。
- テーブルのコピーを格納するデータ・サーバが、データベース内の領域が不足したり、トランザクション・ログが満杯になったことなどのシステム障害が原因で、トランザクションを拒否している。

トランザクションが失敗すると、Replication Server は、アプリケーションに適した処理を行うように例外ログにトランザクションを記録します。Replication Server は、エラー・アクション機能を使用してエラーを柔軟に処理できます。この機能によって、独自に定義した設定内容に基づいてデータ・サーバ・エラーに対応できます。たとえば、トランザクションが失敗したときに、そのサイトでリトライするように指定できます。

適切な解決方法はアプリケーションによって異なるので、例外ログ内のトランザクションは、各サイトのクライアントが解決する必要があります。場合によっては、インテリジェント・アプリケーション・プログラム内で、拒否されたトランザクションを処理するためのロジックをカプセル化することによって、解決方法を自動化できます。

Replication Agent によるトランザクション処理

Replication Agent は、データベース・トランザクション・ログをスキャンして、サブスクリプションを作成したデータベースに分配するように Replication Server にトランザクション情報を送信します。Adaptive Server の RepAgent スレッドによるトランザクション処理について説明します。

注意： 他のデータベースの Replication Agent では、動作が異なることがあります。Replication Server Options のマニュアルを参照。

Adaptive Server のログ・トランケーションの調整

Adaptive Server のデータベース・トランザクション・ログに空き領域がある間は、Adaptive Server はトランザクションの処理を続けます。ログが満杯になるのを避けるには、定期的に空にする (トランケートする) 必要があります。

Adaptive Server の **dump transaction** コマンドを使用するか、Adaptive Server の **trunc log on chkpt** オプションを “on” に設定すると、ログを自動的にトランケートできます。

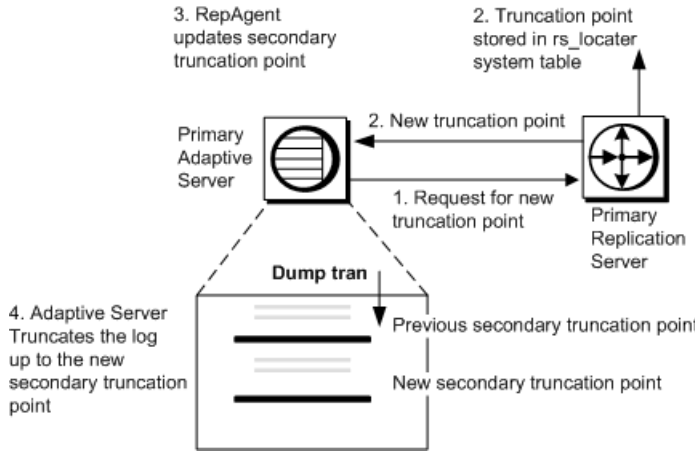
各プライマリ・データベースは、そのデータベース・ログ内にプライマリ・トランケーション・ポイントと「*セカンダリ・トランケーション・ポイント*」を保持しています。プライマリ・トランケーション・ポイントは、Adaptive Server が処理を終了した最後のログ・レコードにマークを付けます。セカンダリ・トランケーション・ポイントは、通常、最も早くからオープンされながらも Replication Server によってまだ完全には適用されていないトランザクションに対する **begin transaction** コマンドを含んでいるログ・レコードにマークを付けます。Replication Server は、最新のセカンダリ・トランケーション・ポイントのコピーを、RSSD の `rs_locater` テーブルに格納します。

RepAgent は、事前に決められた件数 (バッチ) のレコードをスキャンし終えるか、ログの終端に達して新しいアクティビティがなくなると、新しいセカンダリ・トランケーション・ポイントを要求します。Replication Server は、セカンダリ・トランケーション・ポイントを移動できるようにする情報を RepAgent に提供して、トランザクション・レコードのバッチの受信を通知します。

Adaptive Server は、セカンダリ・トランケーション・ポイントを超えてログをトランケートしないことによって、すでに処理されて Replication Server に渡されたトランザクションだけが確実に削除されるようにします。

RepAgent は、「Adaptive Server のログ・トランケーション」の図に示す方法で、セカンダリ・トランケーション・ポイントを更新します。

図 9 : Adaptive Server のログ・トランケーション



1. RepAgent は、プライマリ Replication Server から新しいセカンダリ・トランケーション・ポイントを要求します。
2. プライマリ Replication Server は、最新のセカンダリ・トランケーション・ポイントを RepAgent に返して、それを `rs_locater` システム・テーブルにも書き込みます。
3. RepAgent は、トランザクション・ログ内のセカンダリ・トランケーション・ポイントを更新します。
4. 次のチェックポイントまたは **dump transaction** コマンドで、ログは新しいセカンダリ・トランケーション・ポイントまでトランケートされます。

「スキーマ」情報は、データベースの構造を記述したものです。テーブルの削除、クラスタード・インデックスの作成、カラム名の変更など、データベース・オブジェクトのスキーマを変更するたびに、Adaptive Server はそのオブジェクトに対する現在のスキーマ情報を記録します。したがって、RepAgent がトランザクション・ログをスキャンするときは、元のデータベース・オブジェクトが変更されていたり、すでに存在していない場合であっても、常にテーブルやプロシージャの正しいスキーマを取得できます。プライマリ・サイトでスキーマ変更を実行する前にトランザクション・ログを排出する必要はありません。

Sybase Central での複写環境の管理

Replication Manager (RM) プラグインと Replication Monitoring Services (RMS) を使用して、複写環境を管理することができます。

Sybase では、システム管理ツールを Sybase Central という 1 つのデスクトップ製品に統合しています。Sybase Central から、Replication Server や Adaptive Server などの各サーバ製品を管理できます。

Replication Server をインストールして設定した後、Sybase Central を起動し、RM を使用して、複写に関与するすべてのデータ・サーバ、Replication Server、Replication Agent で構成される新しい複写環境を作成します。

Replication Manager に、環境内で管理する各サーバのアイコンが表示された 2 つのウィンドウ枠が表示されます。このウィンドウを使用して、サーバのステータスをモニタし、サーバと複写システムの他のコンポーネントを診断、管理するメニュー・コマンドを実行します。

参照：
- 複写システムのコンポーネント (27 ページ)

Sybase Central の起動と停止

Sybase Central の起動方法と停止方法について説明します。

Sybase Central の起動

複数の方法を使用して、Sybase Central を起動することができます。

Sybase Central を起動するには、次の手順に従います。

- Windows では、次のいずれかを実行します。
  - [スタート] > [プログラム] > [Sybase] > [Sybase Central v6.0] を選択します。
  - Sybase Central へのショートカットをデスクトップに作成しておきます。
  - `%SYBASE%¥Shared¥Sybase Central 6.0.0/win32` 内の `scjview.exe` をダブルクリックします。
  - Sybase Central を [スタートアップ] プログラム・グループに追加しておきます。

- UNIX では、`$SYBASE/shared/sybcentral600` 内の `scjview.sh` を実行します。

Sybase Central の停止

Sybase Central を停止するには、[ファイル] > [終了] を選択します。

オンライン・ヘルプ

Replication Manager プラグインは、Sybase Central による特定のタスクを実行するためにオンライン・ヘルプを提供します。

Replication Manager プラグインには、次のような異なるタイプのヘルプがあります。

- トピック・ヘルプ
- ヒント
- ステータス・バー・メッセージ

トピック・ヘルプ

トピック・ヘルプは、Sybase Central を使って複写システムを管理する方法について説明します。トピック・レベルのヘルプを表示するには、Sybase Central のメイン・メニューから [ヘルプ] を選択し、[Replication Manager オンライン・ヘルプ] を選択します。

Replication Manager ヘルプ・ブラウザは、2 つのウィンドウ枠で表示されます。左ウィンドウ枠には目次が表示され、右ウィンドウ枠には選択したトピックの内容が表示されます。

[目次] タブをクリックすると、カテゴリ別にトピックを参照できます。

- 「ブック・アイコン」は見出しを表します。ブック・アイコンをダブルクリックすると、その見出しの下にサブエントリが表示されます。サブエントリは、別のブック・アイコンまたはページ・アイコンです。
  トピックの見出しは、参照しやすいように Replication Server の概念別 (「ユーザの管理」、「データベース接続の管理」など) に整理されています。
- 「ページ・アイコン」は、上位の見出しに対応する作業または概念について説明するトピックを表します。通常、各トピックは、該当の見出しの下に手順を実行する順に並べられています。ページ・アイコンをダブルクリックすると、トピックが表示されます。

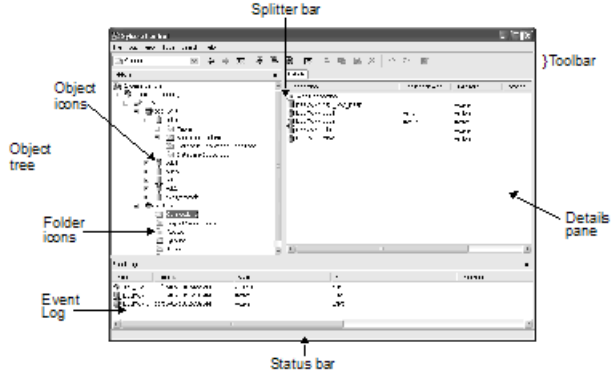
ヒント
ヒントとは、マウス・ポインタをコントロール (ツールバー・ボタンやメニュー・オプション) の上に置いたときに、そのコントロールの説明を表示する小さなポップアップ・ウィンドウのことです。

ステータス・バー
ステータス・バーとは、アプリケーション・ウィンドウの下部にある情報表示バーのことです。Sybase Central のステータス・バーには、カーソルが現在ポイントしているメニュー・コマンドの簡単な説明が表示されます。ヘルプ行はステータス・バーの左側に表示されます。

Replication Manager ユーザ・インタフェース機能

Sybase Central フレームワーク内に表示される Replication Manager ユーザ・インタフェース機能によって、複写環境とサーバ・オブジェクトにアクセスできます。

図 10 : Sybase Central のメイン・ウィンドウ内の Replication Manager



メイン・ウィンドウは左右のウィンドウ枠に分割されています。複写環境に接続すると、次のように表示されます。

- 左のウィンドウ枠には、階層リスト (オブジェクト・ツリー) が表示されます。このリストには、次の項目が表示されます。
  - 複写環境内のフォルダとオブジェクトのアイコン
  - Adaptive Server Enterprise の Sybase Central プラグインなど、他のプラグインのアイコン (プラグインがインストールされている場合)
- 右ウィンドウ枠には、左ウィンドウ枠で選択したフォルダまたはオブジェクトの内容が表示される。

ウィンドウ枠のサイズを調節するには、マウス・ポインタを使用して分割バーを左または右にドラッグします。

Replication Manager の使用方法

フォルダとオブジェクトのアイコン

メイン・ウィンドウには、フォルダ・アイコンとオブジェクト・アイコンが含まれます。

各フォルダ・アイコンには、複写環境内にあるそのフォルダ・タイプのすべてのオブジェクトが含まれます。たとえば、[接続] フォルダにはすべてのコネクションが表示されます。フォルダ・アイコンは、左右どちらのウィンドウ枠にも表示されます。

注意： Replication Server コネクションは、Adaptive Server の 1 つのデータベースに関連付けられています。

各オブジェクト・アイコンは、データベース、テーブル、複写定義、コネクションなど、1 つのサーバ・オブジェクトを表します。オブジェクトによっては、他のオブジェクトを包含しているものもあります。たとえば、データベースにはテーブルが含まれています。

オブジェクトを選択するには、そのアイコンをクリックします。選択したオブジェクトのタイプによって、使用できるコマンドの範囲が決まります。ほとんどのアクティビティでは、オブジェクトを選択してから、そのオブジェクトに対する操作を実行します。

[詳細] タブ

左ウィンドウ枠でオブジェクトを選択すると、右ウィンドウ枠に 1 つ以上のタブが表示されます。ほとんどのオブジェクトでは、そのオブジェクトに関する一般情報のリストを含む [詳細] というタブが 1 つ表示されます。

[詳細] リストに表示される内容は次のとおりです。

- サブコンポーネント ― 別のオブジェクトに含まれる複写オブジェクトまたはデータベース・オブジェクト。
- ファンクション・コンポーネント ― ダブルクリックするとウィザードが起動する。

注意： スレッド・オブジェクトを選択すると、右ウィンドウ枠に複数のタブが表示されます。

Sybase Central オブジェクト・ツリー内での移動

異なる方法を使用すると、オブジェクト・ツリーの別の部分を表示することができます。

現在の表示を上下に移動するには、左または右ウィンドウ枠のスクロール・バーを使用します。

リストを展開するか折りたたんで別の詳細レベルを表示するには、次のいずれかを実行します。

- [+] ボタンまたは [-] ボタンをクリックします。アイコンの横に [+] ボタンが表示されている場合は、そのアイコンに対応するオブジェクトのリストを展開できます。[-] ボタンは、そのアイコンに対応するオブジェクトのリストが完全に展開されていることを示します。
- フォルダ・アイコンをダブルクリックします。右ウィンドウ枠のリストが展開され、ビューがフォルダ内のオブジェクトのリストに変更されます。ほとんどのオブジェクトでは、右ウィンドウ枠でオブジェクト・アイコンをダブルクリックするとプロパティ・シートが開き、情報が表示されます。

表示のカスタマイズ

表示をカスタマイズすることで、[表示] メニューの [フォルダ]、[ツールバー]、[ステータス・バー]、または [イベント・ログ] を選択して、フォルダ、ツールバー、ステータス・バー、またはイベント・ログを非表示または再表示することができます。

右のウィンドウ枠のタブの表示位置を上下左右に移動するには、[ツール] > [オプション] を選択します。

キーボード・ショートカット

マウスを使用するだけでなく、キーボード・ショートカットを使用してメニュー・コマンドを選択し、ダイアログ・ボックス間を移動することもできます。

すべてのメニュー・タイトルとメニュー・コマンドには、*「ニーモニック」*と呼ばれる下線付きの文字が表示されます。メニューを選択するには、[Alt] キーを押しながら*ニーモニック・キー*を押します。メニュー・コマンドを選択するには、ニーモニック・キーを押します。[Ctrl] キーを押しながら別のキーを押すか、ファンクション・キーを押すことによって直接実行できるコマンドもあります。これらのショートカットはメニューに示されています。

ダイアログ・ボックスまたはプロパティ・シートで別のコントロール (フィールド、リスト、ボタンなど) に移動するには、[Tab] キーを使用します。プロパティ・シートで別のタブを選択するには、[Tab] キーを使用して現在のタブを選択してから、左右の矢印キーを使用して他のタブを選択します。

コンテキスト・メニュー

コンテキスト・メニューは選択したオブジェクトに固有であり、選択したオブジェクトに対して実行されるコマンドが含まれます。

複数のオブジェクトに対して複数のコマンドを同時に実行できます。コンテキスト・メニューを表示するには、オブジェクト・アイコンを右クリックします。表示されるメニューから目的のコマンドを選択します。

ツールバー

Sybase Central のツールバーを使用すると、頻繁に使用するメニュー・コマンドをすばやく実行できます。

ツールバーには次のコントロールがあります。

- ドロップダウン・リスト・ボックス ― 現在選択されているオブジェクトの階層を表示する。上位の階層のオブジェクトを選択すると、メイン・ウィンドウのフォーカスを変更できる。
- ボタン ― メニュー・コマンドをすばやく実行できる。

ツールバーを表示または非表示にするには、[表示] メニューの [ツールバー] コマンドのオン／オフを切り替えます。

ステータス・バー

ステータス・バーとは、アプリケーション・ウィンドウの下部にある情報表示バーのことです。Sybase Central のステータス・バーには、カーソルが現在ポイントしているメニュー・コマンドの簡単な説明が表示されます。ヘルプ行はステータス・バーの左側に表示されます。

ステータス・バーを表示または非表示にするには、[表示] メニューの [ステータス・バー] コマンドのオン／オフを切り替えます。

[イベント・ログ] ウィンドウ枠

Replication Manager では、複写環境内で発生したイベントを示す [イベント・ログ] ウィンドウ枠が表示されます。

次のようなイベントがあります。

- コネクション、ルート、キューのコンポーネント・ステータスの変更
- サーバの可用性の変更
- バックグラウンド・スレッドの完了
- バックグラウンド・プロセスのステータス
- RMS イベント・トリガの実行

[バックグラウンド・プロセス]
バックグラウンドによる複写処理をモニタし、これらの実行を停止することができます。

テーブルもマテリアライズするサブスクリプションの作成など、Replication Manager によって実行される一部のタスクは非常に時間がかかることがあります。Replication Manager が他のタスクを実行できるように、このようなタスクはバックグラウンドで実行されます。時間のかかるタスクを開始すると、Replication Manager にプロセスが実行中であることを示すメッセージ・ウィンドウが表示されます。[プロセスの停止] をクリックしてバックグラウンド・プロセスをキャンセルするか、[閉じる] をクリックしてウィンドウを閉じ、バックグラウンドでのプロセスの実行を続行します。

バックグラウンド・タスクが完了すると、Replication Manager によってイベント・ログにイベント・エントリが書き込まれます。

バックグラウンド・プロセスのステータスを後で確認するには、[バックグラウンド・プロセス] ダイアログを開きます。このダイアログには、現在実行中のすべてのプロセスのリストが表示されます。

以下の項目を表示する [バックグラウンド・プロセス] ウィンドウにアクセスするには、[検索] > [バックグラウンド・プロセス] を選択します。

- [プロセス] ― プロセスの名前
- [開始時刻] ― プロセスの開始時刻
- [ステータス] ― プロセスのステータス

スクリプト・エディタ
スクリプト・エディタを使用すると、生成された RCL コマンドを表示できます。RCL コマンドには、コネクション、ルート、複写定義などのオブジェクトを作成する構文が含まれます。

Replication Manager には、複写コマンド言語 (RCL: Replication Command Language) スクリプト・エディタと、SQL (Structured Query Language) スクリプト・エディタの 2 つのスクリプト・エディタが用意されています。これらのエディタの動作は同じですが、RCL スクリプト・エディタは RCL キーワードを強調表示し、SQL スクリプト・エディタは SQL キーワードを強調表示する点が異なります。

スクリプト・エディタへのアクセス
スクリプト・エディタにアクセスする方法について説明します。

1. RCL を生成する Replication Server オブジェクトを選択します。
2. 選択したオブジェクトを右クリックします。

3. コンテキスト・メニューの [RCL 生成] を選択します。選択したスクリプト・エディタ・ウィンドウが開き、選択したオブジェクトを作成するために必要な RCL が表示されます。

ステータスのモニタ

Replication Manager では、環境内のサーバとコンポーネントのステータスがグラフィック表示されます。

Replication Manager が表示するオブジェクト・アイコンは、オブジェクトのステータスによって異なります。[プロパティ] ダイアログには、サーバ、コネクション、ルート、キューのステータスも表示されます。

環境のステータスとは、そのコンポーネントの状態のことです。サーバまたはコンポーネントのステータスには、現在のステータスと、そのステータスについて説明する理由のリストが含まれます。

作業を進めるうちに、開いているダイアログ・ボックスと Sybase Central ウィンドウ内の情報が同期しなくなることがあります。ウィンドウの内容を更新するには、[ビュー] > [フォルダのリフレッシュ] または [ビュー] > [すべてリフレッシュ] を選択するか、[F5] を押します。

コネクション・ステータスの非表示

コネクション・ステータスを個々のコネクション・アイコンに表示したり、Replication Server のロールアップ・ステータスの一部として表示したりしない場合は、コネクション・ステータスを非表示に (またはフィルタ) できます。

コネクション・ステータスを非表示にするオプションは次のとおりです。

- [Replication Agent のステータスを非表示] — [詳細] リスト、[接続プロパティ] ダイアログ・ボックス、Replication Agent スレッドの接続先である Replication Server のロールアップ・ステータスで、Replication Agent スレッドのステータスを非表示にする。
- [DSI スレッドのステータスを非表示] — [詳細] リスト、[接続プロパティ] ダイアログ・ボックス、DSI スレッドが関連付けられている Replication Server のロールアップ・ステータスで、DSI スレッドのステータスを非表示にする。

コネクション・ステータスを非表示にするには、次を実行します。

1. ステータスを非表示にするコネクションを右クリックします。
2. ドロップダウン・メニューから、[コネクション・ステータスの非表示] を選択します。

   コネクション・ステータスを非表示にするオプションが示されたダイアログ・ボックスが表示されます。
3. オプションを選択します。

そのコネクションのステータスが「非表示」になります。[接続プロパティ] ダイアログ・ボックスおよび Replication Server のロールアップ・ステータスのステータスも非表示になります。この変更はイベント・ログに記録されます。

Replication Manager の別のインスタンスによるコネクション・ステータスのフィルタ

コネクション・ステータスのフィルタリングは、Replication Manager によってローカルで格納されます。

したがって、Replication Manager の別のインスタンスはフィルタリング状態を共有することはありません。たとえば、Replication Manager のあるインスタンスを使用してコネクションを作成し、そのコネクションについて複写ステータスを非表示にするように設定した場合、同じ環境をモニタしている別の Sybase Central プラグイン・インスタンスがコネクション・ステータスをフィルタすることはありません。フィルタリング情報を使用できるのは、元の Replication Manager インスタンスだけです。

また、Sybase Central の外部で (**rs_init** またはコマンド・ラインから) 作成されたコネクションは、Replication Manager によって自動的にフィルタされるわけではありません。Sybase Central 内から手動でフィルタリングを設定する必要があります。

ウォーム・スタンバイ環境でのコネクション・ステータスのフィルタ

ウォーム・スタンバイ環境を作成する場合、アクティブなデータ・サーバ・インタフェース (DSI) スレッド・コネクションとスタンバイ RepAgent スレッド・コネクションのフィルタリング状態が Replication Manager によって自動的に設定されます。

コンテキスト・メニューからコネクション・ステータスの非表示オプションのいずれかを選択して、物理コネクションのフィルタリングを手動で設定します。

共通の作業の実行

オブジェクトの作成

Sybase Central では、複写定義、サブスクリプション、コネクション、その他の Replication Server オブジェクトを作成できます。

1. 作成するオブジェクトのタイプに応じたフォルダを選択します。
2. [ファイル] > [新規作成] を選択します。
3. オブジェクト名を選択します。次のいずれかの動作が発生します。
   - 選択したオブジェクトの作成を支援するウィザードがある場合は、ウィザードが開きます。ウィザードの指示に従ってください。

- ウィザードが存在しない場合は、プロパティ・シートが表示されます。新しいオブジェクトの情報を入力してください。

オブジェクトのプロパティ

オブジェクトを作成すると、Sybase Central ウィンドウのすべてのウィンドウ枠にオブジェクトがアイコンとして表示されます。オブジェクトの [プロパティ] ダイアログを開くことによって、オブジェクトを表示または更新できます。

[プロパティ] ダイアログには、オブジェクトに関する情報や、複写環境内の他のオブジェクトとの関連性についての情報が表示されます。また、関連するオブジェクトへの直接のナビゲーション・パスが表示される場合もあります。このダイアログでは、タブ付きの各ページで新しいオブジェクトの情報を入力できます。

[プロパティ] ダイアログ・ボックスには、通常 3 つのタブがあります。

- [一般] タブ ― すべてのステータスが表示される。
- [通信] タブ ― Replication Manager がサーバと通信する方法に関する情報が表示される。
- [パラメータ] タブ ― サーバとコンポーネントの設定パラメータが表示され、パラメータを変更できる。

注意： 別のタブが含まれた [プロパティ] ダイアログもあります。たとえば、コネクションの場合は、[一般]、[セキュリティ]、[パラメータ] の各タブがあります。

オブジェクトのプロパティの表示

Sybase Central のオブジェクトのプロパティを表示するには、オブジェクト・アイコンを選択してから [ファイル - プロパティ] を選択します。

オブジェクトの削除

Sybase Central では、複写定義、サブスクリプション、コネクション、その他の Replication Server オブジェクトを削除できます。

1. オブジェクト・アイコンを選択します。
2. [編集] > [削除] を選択します。
3. 確認のダイアログで削除を確認します。

Replication Server のデータ・サーバの名前

Replication Manager のデータ・サーバにはユニークな名前が必要です。また、Sybase 以外のデータ・サーバの名前は、Replication Agent の **rs_source_ds** 設定パラメータと一致している必要があります。

既存の環境で Replication Agent と設定パラメータに同じ名前を使用している場合は、サーバの追加ウィザードの 3 ページ目でサーバ名、ホスト、ポート番号を手動で追加して、エージェントの名前を変更します。

複写環境の設定

複写環境には、Replication Server、データ・サーバ、Replication Agent などの複写オブジェクトが含まれます。環境を作成し、設定してから、複写アクティビティを実行します。

複写環境の規模と複雑さに応じて、環境に 2 層または 3 層の管理ソリューションを設定できます。

2 層管理ソリューション

2 層管理ソリューションでは、Replication Manager (RM) は管理層経由で通信することなく、環境内のサーバに直接接続します。

このソリューションでは、10 台未満のサーバで構成された小規模で単純な複写環境を管理し、複写環境内のコンポーネントを作成、変更、削除できます。

3 層管理ソリューション

3 層管理ソリューションでは、Replication Manager は Replication Monitoring Services (RMS) を使用して、大規模で複雑な複写環境をモニタできます。RM は、RMS を介して環境内のサーバに接続します。

RMS は、複写環境のモニタリング機能を備えています。このソリューションでは、RMS は複写環境内のサーバと他のコンポーネントのステータスをモニタします。RM には、RMS によって提供される情報を表示するクライアント・インタフェースが用意されています。

2 層ソリューションの準備

2 層管理ソリューションに向けて複写環境を準備するために、いくつかの手順を完了する必要があります。

注意： Replication Manager は Sybase interfaces ファイルを必要としませんが、このファイルを使用することもできます。

1. Replication Server と Sybase Central ソフトウェアをインストールします。詳細については、使用しているプラットフォームの『リリース・ノート』と『インストール・ガイド』を参照してください。
2. 複写システムで使用するデータ・サーバを指定します。データ・サーバがまだインストールされていない場合は、使用するデータ・サーバのインストール・ガイドを参照してインストールしてください。

3. **rs_init** を使用して Replication Server を設定します。使用しているプラットフォームの詳細については、『Replication Server 設定ガイド』を参照してください。
4. Sybase Central を起動する。
5. Sybase Central で複写環境を作成し、データ・サーバと Replication Server を追加します。

環境の作成

複写環境を作成するには、環境名の指定、環境オブジェクトの作成、パーミッションの設定、サーバの追加を行います。

複写環境オブジェクトの作成

Replication Manager を使用して、複写環境オブジェクトを作成します。

1. Sybase Central で、Replication Manager ウィンドウの左ウィンドウ枠にある [Replication Manager] アイコンをクリックします。[詳細] ウィンドウ枠に [複写環境の追加] アイコンが表示されます。
2. 右側のウィンドウ枠で、[複写環境の追加] をダブルクリックします。
3. 環境の名前を入力し、[次へ] をクリックします。
4. 環境にアクセスするときに使用するユーザ名とパスワードを入力します。[次へ] をクリックします。

   注意： Java は roman8 文字セットをサポートしていません。サーバ・コネクションの [文字セット] フィールドで roman8 を選択しないでください。サーバのデフォルトが roman8 である場合、サーバ・コネクションの [文字セット] フィールドで [デフォルト] を選択しないでください。サーバおよび Replication Manager と互換性がある別の文字セットを選択する必要があります。
5. サーバのリストから環境に追加するサーバを選択し、各サーバのユーザ名とパスワードを追加します。Replication Server を追加する場合は、RSSD のユーザ名とパスワードを入力してください。[次へ] をクリックします。

   サーバを追加する場合は、次のように特定のパーミッションが付与されているユーザ名とパスワードを使用してください。
   - Replication Server － **sa** パーミッション。
   - Adaptive Server Enterprise – **sa_role** と **sso_role**。
   - Replication Server RSSD – データベース所有者。

注意： リストからサーバを選択しても、サーバ名、ホスト、ポート番号を入力してもかまいません。このリストは、`$SYBASE` ディレクトリ内の interfaces ファイルから作成されます。

6. 概要ページをチェックして、必要なサーバをすべて追加していることを確認します。その後、[完了]をクリックします。

注意： 環境の作成時にすべてのサーバを追加する必要はありません。サーバの追加ウィザードを使用すると、既存の環境に新しいサーバを追加できます。

左ウィンドウ枠の Replication Manager オブジェクトの下に、指定した名前の新しい環境オブジェクトが表示されます。

注意： Sybase Central の動作中に interfaces ファイルを更新する場合は、ウィザードを再起動するか、実行中のダイアログ・ボックスを開き直します。変更を有効にするために、Sybase Central を再起動する必要はありません。

Sybase Central での複写環境からのサーバの削除

Replication Manager を使用して、複写環境からサーバを削除します。

1. 削除するサーバを選択します。
2. 次のいずれかを実行します。
   - ツールバーの [削除] アイコンをクリックします。
   - 選択したサーバを右クリックし、[削除] を選択します。

注意： Sybase Central は、複写環境のサーバ・リストからサーバを削除し、環境からサーバのアイコンを削除しますが、複写システムからサーバが削除されるわけではありません。削除したサーバと関連するルートまたはデータベース・コネクションがまだ存在している場合は、ダイアログ・ボックスにサーバ名が表示されたままになることがあります。

複写環境への接続と複写環境からの切断

Replication Manager では環境情報が保存されるため、Sybase Central の再起動時に情報を再作成する必要はありません。既存の複写環境を接続または切断する必要があるだけです。

既存の複写環境への接続

Replication Manager を使用して、既存の複写環境に接続します。

1. 接続先の環境を選択します。
2. ログイン・ダイアログ・ボックスで、ユーザ名とパスワードを入力します。
3. [OK] をクリックします。これで環境の管理を開始できます。

複数の環境または RMS ドメインに同時に接続できます。

複写環境からの切断

Replication Manager を使用して、複写環境から切断します。

1. 切断する環境を選択します。
2. [ツール] > [切断] を選択します。[詳細] ビューに、切断する環境のステータスが表示されます。

Replication Manager を使用した複写環境の設定

Replication Manager を使用して、希望する複写環境のタイプを素早く作成するために複写の設定ウィザードを起動します。

複写環境を作成したら、複写の設定ウィザードを使用して、複写タスクと次のような稼動中の複写環境タイプのいずれかに、コネクション、データベース複写定義、サブスクリプションを作成します。

- 単純なウォーム・スタンバイ環境
- 1 つのプライマリと複数のレプリケートで構成される環境
- 双方向複写環境

1. 作成済みの環境オブジェクトを選択します。
2. 右側のウィンドウ枠で、[複写の設定] をダブルクリックします。複写の設定ウィザードが表示されます。
3. 複写の設定ウィザードで、作成する環境のタイプを選択します。タイプは次のとおりです。
   - 標準の Replication Server ウォーム・スタンバイ環境。
   - プライマリ・データベースが複数のレプリケート・サイトに複写される環境。
   - 双方向複写環境。

標準ウォーム・スタンバイ環境の設定

複写の設定ウィザードを使用して、単純なウォーム・スタンバイ環境を設定します。

前提条件

設定するアクティブなサーバがリストに表示されない場合は、[サーバの追加] をクリックしてサーバの追加ウィザードを起動します。手順に従って、複写環境オブジェクトを作成します。

手順

1. 複写の設定ウィザードで作成する環境のタイプを選択したら、[次へ] をクリックします。
2. アクティブ・サーバとアクティブ・データベースを選択します。
3. データベース接続を管理する Replication Server を選択します。
4. スタンバイ・サーバとスタンバイ・データベースを選択します。
5. 論理コネクション名を入力します。

   既存のコネクションを使用してウォーム・スタンバイ論理コネクションを作成するときは、論理コネクション名としてアクティブ・データベースの既存のデータ・サーバ名とデータベース名を使用してください。詳細については、『Replication Server 管理ガイド 第 2 巻』を参照してください。
6. 管理ユーザのユーザ名とパスワードを入力します。メンテナンス・ユーザが存在しない場合は、ウィザードによって作成されます。デフォルト値を受け入れるか、独自の値を入力します。
7. Replication Server に接続するために RepAgent が使用するユーザ名とパスワードを選択します。RepAgent ユーザが存在しない場合は、ウィザードによって作成されます。デフォルト値を受け入れるか、独自の値を入力します。
8. マテリアライゼーション方法を選択します。
9. 複写環境の概要情報を確認します。
10. すべて正しければ [完了] をクリックします。正しくない場合は、[戻る] をクリックして前のウィンドウに戻り、複写環境の情報を変更します。ウィザードの最後のウィンドウに戻り、[完了] をクリックします。

    Replication Manager によって、次の複写オブジェクトが作成されます。
    - 論理コネクション
    - 物理コネクション
    - アクティブおよびスタンバイ Adaptive Server Enterprise サーバのメンテナンス・ユーザ

参照：

- 複写環境オブジェクトの作成 (68 ページ)

1 つのプライマリと複数のレプリケートを含む環境の設定

Multi-Site Availability (MSA) を使用して、1 つのサイトから複数のレプリケート・サイトにデータを複写するウォーム・スタンバイ環境を設定します。

前提条件

設定するアクティブなサーバがリストに表示されない場合は、[サーバの追加] をクリックしてサーバの追加ウィザードを起動し、サーバ・オブジェクトを追加します。手順に従って、複写環境オブジェクトを作成します。

手順

1. プライマリ・サーバとプライマリ・データベースを選択します。[次へ] をクリックします。
2. データベース接続を管理する Replication Server を選択します。[次へ] をクリックします。
3. レプリケート・サーバを選択してから、それに対応するデータベースを選択し、[追加] をクリックします。

   [接続] リストに、該当する *data server.database* 接続が表示されます。

   複写環境に必要なレプリケート・サーバとレプリケート・データベースのペアごとにこの手順を繰返します。[次へ] をクリックします。
4. 管理ユーザのユーザ名とパスワードを入力します。メンテナンス・ユーザが存在しない場合は、ウィザードによって作成されます。デフォルト値を受け入れるか、独自の値を入力します。[次へ] をクリックします。

   選択したすべてのコネクションで、このメンテナンス・ユーザのログインが使用されます。
5. Replication Server に接続するために RepAgent が使用するユーザ名とパスワードを選択します。RepAgent ユーザが存在しない場合、ウィザードによって RepAgent ユーザが作成され、デフォルトのユーザ名とパスワードが割り当てられます。デフォルト値を受け入れるか、独自の値を入力します。[次へ] をクリックします。

   すべてのコネクションでこの RepAgent ログインが使用されます。
6. 複写テーブルをマテリアライズする (実体化する) 方法を指定します。
   - [実体化せずにサブスクリプションを作成] – この方法は、プライマリ・データがレプリケートですでにロードされており、更新処理が実行中でない場合に使用します。

- [サブスクリプションを定義してバルク実体化を使用] – この方法では、ユーザが指定したメカニズムによって複写システムの外部でサブスクリプションを初期化します。

7. [サブスクリプションを定義してバルク実体化を使用] を選択した場合は、[トランザクション・ログにダンプ・マーカを使用] をクリックして、**dump** と **load** の調整を使用します。[次へ] をクリックします。
8. 複写環境の概要情報がすべて正しければ、[完了] をクリックします。正しくない場合は、[戻る] をクリックして前のウィンドウに戻り、複写環境の情報を変更します。ウィザードの最後のウィンドウに戻り、[完了] をクリックします。

   設定が終了すると、Replication Manager によって次の複写オブジェクトが作成されます。
   - 物理コネクション
   - プライマリ・データベースのデータベース複写定義
   - レプリケート・データベースごとに 1 つ以上のデータベース・サブスクリプション
   - Adaptive Server Enterprise サーバのメンテナンス・ユーザ

参照：
- 複写環境オブジェクトの作成 (68 ページ)

双方向複写環境の作成

データを複数のロケーションで更新し、各サイトに複写する環境を定義します。

1. 双方向複写環境に含めるサーバとデータベースを指定します。
2. レプリケート・サーバを選択してから、それに対応するデータベースを選択し、[追加] をクリックします。

   [接続] リストに、該当する *data server.database* 接続が表示されます。

   複写環境に必要なレプリケート・サーバとレプリケート・データベースのペアごとにこの手順を繰返します。[次へ] をクリックします。
3. 管理ユーザのユーザ名とパスワードを入力します。メンテナンス・ユーザが存在しない場合は、ウィザードによって作成されます。デフォルト値を受け入れるか、独自の値を入力します。[次へ] をクリックします。

   選択したすべてのコネクションで、このメンテナンス・ユーザのログインが使用されます。
4. Replication Server に接続するために RepAgent が使用するユーザ名とパスワードを選択します。RepAgent ユーザが存在しない場合、ウィザードによって RepAgent ユーザが作成され、デフォルトのユーザ名とパスワードが割り当てられます。デフォルト値を受け入れるか、独自の値を入力します。[次へ] をクリックします。

すべてのコネクションでこの RepAgent ログインが使用されます。

5. 複写テーブルをマテリアライズする (実体化する) 方法を指定します。
   - [実体化せずにサブスクリプションを作成] – この方法は、プライマリ・データがレプリケートですでにロードされており、更新処理が実行中でない場合に使用します。
   - [サブスクリプションを定義してバルク実体化を使用] – この方法では、ユーザが指定したメカニズムによって複写システムの外部でサブスクリプションを初期化します。
6. [サブスクリプションを定義してバルク実体化を使用] を選択した場合は、[トランザクション・ログにダンプ・マーカを使用] をクリックして、**dump** と **load** の調整を使用します。[次へ] をクリックします。
7. 複写環境の概要情報がすべて正しければ、[完了] をクリックします。正しくない場合は、[戻る] をクリックして前のウィンドウに戻り、複写環境の情報を変更します。ウィザードの最後のウィンドウに戻り、[完了] をクリックします。

   設定が終了すると、Replication Manager によって次の複写オブジェクトが作成されます。
   - 物理コネクション
   - プライマリ・データベースのデータベース複写定義
   - レプリケート・データベースごとに 1 つ以上のデータベース・サブスクリプション
   - Adaptive Server Enterprise サーバのメンテナンス・ユーザ

Replication Server オブジェクトの管理

Replication Manager を使用すると、コネクション、複写定義、サブスクリプション、キューなどの Replication Server オブジェクトを作成、管理できます。

Sybase 以外のデータ・サーバに対応するために、Replication Manager は DirectConnect™ を使用して、データ・サーバと通信し、Replication Manager のインタフェースとしての役割を果たします。DirectConnect のステータスは、Sybase 以外のデータ・サーバのステータスに反映されます。

注意： Replication Manager では、Sybase 以外のデータ・サーバのデータベース複写定義、データベース・サブスクリプション、論理コネクションの作成はサポートしていません。

接続

コネクションでは、データベースから Replication Server、または Replication Server からデータベースに接続します。Replication Server はプライマリ・データベースから受け取ったトランザクションを、自ら管理するレプリケート・データベースへのコネクションを通じて分配します。

コネクションの作成

Replication Manager を使用して、データベースへのコネクションを作成します。

1. Sybase Central オブジェクト・ツリーで [接続] フォルダを選択します。
2. [詳細] ウィンドウ枠で、[接続の追加] アイコンをダブルクリックします。データベース接続の追加ウィザードが開きます。[次へ] をクリックします。
3. ドロップダウン・リストから、アクティブなサーバとデータベースを選択します。[次へ] をクリックします。
4. ユーザ名を入力するか、デフォルト値を受け入れます。
5. パスワードを入力します。[次へ] をクリックします。
6. 表示されたオプションから選択します。[次へ] をクリックします。
7. 概要情報を確認したら、[完了] をクリックします。
8. Sybase 以外のデータ・サーバへの DirectConnect を介したレプリケート・コネクションを作成した場合は、レプリケーションに必要なテーブルとプロシージャを生成するスクリプトを手動で実行します。

複写定義

複写定義では、Replication Server に対する送信元テーブルを記述して、コピー対象のカラムを指定します。

同時に、送信先テーブルの属性も記述できます。指定した特性に一致する送信先テーブルでは、複写定義に対するサブスクリプションを作成できます。

Replication Server では、データベース、テーブル、ストアド・プロシージャの各レベルで複写できます。RM を使用すると、データベース、テーブル、またはストアド・プロシージャの複写定義を作成できます。ストアド・プロシージャの複写定義は、「ファンクション複写定義」と呼ばれます。ファンクション複写定義とファンクション・サブスクリプションを作成、編集、削除できます。

プライマリ・データベースでの複写定義の作成

Replication Manager を使用して、複写定義をプライマリ・データベースに作成します。

1. オブジェクト・ツリーで、複写定義を作成するデータベースをダブルクリックします。[データベース複写定義] フォルダが表示されます。
2. [データベース複写定義] フォルダをダブルクリックします。[新しいデータベース複写定義の追加] ウィンドウが表示されます。
3. [一般] タブで複写定義名を入力します。

注意： 他のタブでは、他の複写定義設定を指定できます。

4. プライマリ・データベースで実行される DDL をレプリケート・データベースに複写する場合は、[すべての DDL を複写] をクリックします。
5. [OK] をクリックします。

サブスクリプション

サブスクリプションは、複写するデータが格納されているプライマリ・データベースを特定します。

特定の複写定義に対するサブスクリプションを作成するためにレプリケート・データベースでサブスクリプションを作成します。データベース、テーブル、ストアド・プロシージャの各タイプの複写定義に対するサブスクリプションを作成できます。

データベース複写定義に対するサブスクリプションの作成

Replication Manager を使用して、プライマリ・データベースへのサブスクリプションを作成します。

1. オブジェクト・ツリーで、サブスクリプションを作成するデータベースをダブルクリックします。
2. [データベース・サブスクリプション] フォルダをダブルクリックします。
3. [詳細] ウィンドウ枠で、[サブスクリプションの追加] アイコンをダブルクリックします。
4. サブスクリプションの名前を入力します。
5. プライマリで、コネクションおよびサブスクリプションを作成するデータベース複写定義を選択します。
6. ドロップダウン・リストからマテリアライゼーション・メソッドを選択します (オプション)。
7. subscribe to truncate table 句を実行するかどうかを指定します (オプション)。
8. [OK] をクリックします。

キュー

サーバ間 (Adaptive Server、Replication Server など) で渡されるデータは、Replication Server 内のステーブル・キューに格納されます。Replication Manager により、キューの使用状況の統計情報とキューの内容が表示されます。

キュー・データの表示

Replication Manager を使用して、キュー・データを表示します。

1. オブジェクト・ツリーで [キュー] フォルダをクリックします。[詳細] ウィンドウ枠にキューが表示されます。
2. [詳細] ウィンドウ枠で、データを表示するキューを右クリックします。
3. コンテキスト・メニューの [データの表示]を選択します。[データの表示] ダイアログ・ボックスが開きます。
4. 表示するデータをフィルタするには、フィルタ・フィールドのいずれかを選択します。
5. データをソートするには、[セグメント]、[トランザクション]、[オリジン]、[サイズ]、[ステータス]、[コミット時間]、または [ユーザ] を選択します。

[キュー・データの表示] ダイアログ・ボックス

[キュー・データの表示] ダイアログ・ボックスでは、キューのトランザクションのトラブルシューティングのために、キューのデータをフィルタしたり、ソートしたりできます。また、特定のコマンドの編集、削除、削除の取り消しを行ったり、キューにある最初のトランザクションをパージすることもできます。

[キュー・データの表示] ダイアログ・ボックスには、次のオプションがあります。

- フィルタ・フィールド ― RM がキューのデータを表示するときに使用するフィルタのタイプを選択できる。次のようなフィルタがある。
  - カラム
  - カラム値
  - セグメント
  - 表示するブロックの数
  - 表示するローの数
  - 最初のセグメントで開始するかどうか
  - セグメントの最後までのすべてのデータを含めるかどうか
  - すべてのローを含めるかどうか
  - 削除したデータを表示するかどうか
  - キューの最後までのすべてのデータを表示するかどうか
- 一般的なボタン ― 次の操作を実行できる。
  - 現在のフィルタを使用してキュー・データを表示する。
  - ダイアログ・ボックスを閉じる。
  - キューから最初のトランザクションをパージする。
  - トランザクションを編集する。
  - トランザクションを削除する。

  - トランザクションの削除を取り消す。
  - トランザクションをグループ化する。[キュー・データ] スクロール・リストの表示はグループ化されたトランザクションに戻ります。

- [キュー・データ] スクロール・リスト ― 現在のキューのデータ・ローが含まれる。各カラムには、各ローに含まれるコマンドとトランザクションに関する固有の情報が含まれる。たとえば、特定のカラムでキュー・データをソートするには、該当のカラム名を選択する。[キュー・データ] スクロール・リストが再表示され、選択したカラムに基づいてデータがソートされる。カラム名の横に矢印が表示され、そのカラムに基づいてデータをソートしたことが示される。次のカラムをソートできます。
  - セグメント
  - トランザクション名
  - コマンド
  - オリジン・サイト
  - オリジンのコミット時刻
  - オリジン・ユーザ
  - トランザクション ID
  - オリジン QID

注意： Replication Server がスタンドアロン・モードの場合、実行できる操作はキュー・トランザクションの削除、削除の取り消し、またはパージだけです。

複写環境のモニタに使用

複雑な複写環境をモニタするために、Replication Manager は Replication Monitoring Services を介して環境内のサーバに接続します。Replication Monitoring Services は、複写環境のモニタリング機能を提供する中間管理層です。

RMS は、複写システムのオプション・コンポーネントです。大規模で複雑な環境をモニタするときに RMS が使用されます。RMS は、データのフローを制御したり、設定パラメータを指定する機能も備えています。

3 層ソリューションでは、複写環境をモニタするために RMS サーバを設定します。このソリューションでは、RMS は複写環境内のサーバと他のコンポーネントの正常性と可用性をモニタします。

RMS を起動、停止、設定するプラットフォームについては、『Replication Server 設定ガイド』を参照してください。

3 層ソリューションの準備

3 層環境を作成するときには、RMS サーバに接続します。

この環境では、`interfaces` ファイルまたは `sql.ini` ファイル (Windows の場合) で、ホスト名、ポート番号、サーバ名を編集する必要があります。これらのファイルは、テキスト・エディタまたは **dsedit** ユーティリティを使用して編集できます。複写環境内の他のサーバが使用する `interfaces` ファイルと同じファイルを使用できます。

RM では、interfaces ファイル内に RMS のエントリを必要としません。RMS のホスト名とポート番号を RM に直接指定できます。RMS によって管理されるサーバは、RMS の interfaces ファイルに含まれている必要があります。

RMS の設定については、使用しているプラットフォームの『Replication Server 設定ガイド』を参照してください。

RMS への接続

RMS が起動したら、RM プラグインを使用して RMS に接続します。RM プラグインは、RMS でモニタする必要のあるサーバを追加する場合にも使用できます。

1. ツールバーの [接続] アイコンをクリックして、[複写ドメインへの接続] ウィンドウを開きます。
2. [RMS サーバ] を選択します。
3. RMS への接続に必要なユーザ名とパスワードを入力します。
4. ドロップダウン・リストのサーバの一覧から [RMS] を選択するか、[オプション] ボタンをクリックして RMS のコネクション情報を指定します。
5. サーバ名、ホスト、ポート番号を入力します。
6. [OK] をクリックします。オブジェクト・ツリーに RMS サーバが追加されます。

『Replication Server リファレンス・マニュアル』の「Replication Monitoring Services API」を参照してください。

RMS を使用したサーバの追加と削除

3 層環境内のサーバは、2 層環境と同様に追加、削除されます。違いは、オブジェクトを選択してプロパティを参照した場合に表示されるプロパティにあります。

RMS はサーバとコンポーネントをモニタするように設計されているため、RMS が使用するプロパティを参照するだけで、複写環境をモニタし、トラブルシューティングできます。

管理対象オブジェクトの表示

RMS フォルダを使用して、RMS によって管理されたオブジェクトを表示します。

オブジェクト・ツリーで [RMS] フォルダをダブルクリックするか展開して、RMS が管理する複写オブジェクトを表示します。RMS の下に、コネクション、ルート、キュー、スレッドが表示されます。[ルート] フォルダなどの複写オブジェクトを選択すると、作成されたルートのリストを表示できます。これらの複写オブジェクトは、Replication Manager を使用して管理できます。

Replication Manager で RMS のオブジェクトを表示する方法は、2 層環境でのオブジェクトの表示方法とまったく同じです。

イベント・トリガの追加

RMS を使用すると、イベント・トリガに基づいてスクリプトを実行し、複写環境をモニタできます。

Replication Monitoring Services は、複写環境をモニタするように設計されています。環境内で何らかの処理が発生すると、サーバとコンポーネントのステータスが変更されます。これらの変更は、イベント・ログに表示されます。RMS を使用してイベント・トリガを作成し、これらの変更をモニタできます。

複写環境でイベントが発生すると、イベント・トリガによってユーザに通知されます。指定したイベントが発生すると、RMS はスクリプトを実行します。たとえば、ユーザは、コネクションがサスペンドしたときに電子メール・メッセージを要求するスクリプトを設定できます。この機能により、イベント発生時の通知方法を指定できます。RMS がモニタするすべてのサーバまたはコンポーネントのイベント・トリガを作成できます。

Replication Server のイベント・トリガの作成

RMS を使用して、イベント・トリガを作成します。

1. オブジェクト・ツリーで Replication Server を選択します。
2. デスクトップの右側にある [イベント・ログ] ウィンドウ枠を選択します。
3. [サーバ・イベント・トリガの追加] アイコンをダブルクリックします。
4. イベントをトリガするステータス変更を選択します。
5. [実行前に待機] の値を入力します (オプション)。この値を指定すると、RMS はイベントが変更されるまで待機してからトリガを実行するようになります。
6. [間隔ごとに実行] を選択します (オプション)。このオプションを選択すると、RMS は 1 回だけでなく、モニタリング間隔ごとにトリガを実行するようになります。

7. イベントの発生時に RMS が実行するスクリプトの名前を入力します。
8. [OK] をクリックします。[イベント・ログ] ウィンドウ枠に新しいイベントが表示されます。

複写システムの管理

Replication Server を起動、停止し、複写システムをモニタ、維持、設定します。

複写システムの設定

複写システムを設定するための基本的な手順を説明します。この手順には、各企業組織での複写のニーズについての計画と慎重な考慮が必要です。

Replication Server を使用するのが初めての場合は、複写システムの計画に役立つ情報として最初に『Replication Server デザイン・ガイド』を参照してください。

これらの手順の一部は、Replication Server の設定ユーティリティである **rs_init** を使用して実行できます。このユーティリティを使用すると、Replication Server の設定やデータベースのシステムへの追加ができます。Sybase Central を使用すると、データベースの追加、複写定義の作成、サブスクリプションの作成など、ここに記載している作業の大半を実行できます。

複写システムを設定する前に、使用しているプラットフォームの『Replication Server インストール・ガイド』に従って Sybase ソフトウェアをインストールしてください。

Replication Server のインストール後に、**rs_init** ユーティリティ・プログラムを使用して、複写システムの起動と設定、データベースの追加を行います。

rs_init の詳細については、『Replication Server 設定ガイド』を参照してください。

コネクションとルートの作成

あるデータベースから別のデータベースにデータを複写するには、Replication Server がその送信元から送信先にデータを転送できるルートとコネクションを最初に設定する必要があります。

- コネクションの作成
  Sybase Central または **rs_init** を使用してデータベースを複写システムに追加すると、コネクションが作成されます。Adaptive Server データベース以外のデータベースに、または Adaptive Server データベースではないデータベースからデータを複写する場合を除き、コマンド・ライン・オプションの **create connection** を使用してコネクションを作成する必要はありません。
  『Replication Server 設定ガイド』を参照して、**rs_init** を使用します。
- ルートの作成

ルートは、Sybase Central を使用するか、送信元 Replication Server で **create route** コマンドを実行して作成します。

参照：

- データベース・コネクションの管理 (189 ページ)
- ルートの管理 (161 ページ)

パーミッションとセキュリティの設定

複写システムに対する Replication Server のセキュリティを設定するため、ログイン名、パスワード、パーミッションを設定します。

Replication Server のログイン名と特定のパーミッションは次のものに必要です。

- 複写データの設定、または Replication Server のモニタと管理を行うユーザ。これらのユーザは、Sybase Central またはコマンド・ラインで作成できます。
- 複写システムのコンポーネント (データ・サーバや Replication Server など)。システム・ユーザは、**rs_init** またはコマンド・ラインで作成できます。

  rs_init の詳細については、使用しているプラットフォームのインストールと設定に関するガイドを参照してください。

ネットワークベース・セキュリティが有効なサイトの場合は、Replication Server の通信に対して Replication Server 用の安全な経路を設定してメッセージ保護オプションを選択できます。

参照：

- Replication Server ユーザのセキュリティ管理 (242 ページ)
- Replication Server システムのセキュリティ管理 (233 ページ)
- ネットワークベース・セキュリティの管理 (262 ページ)

複写システムの確認

複写定義やサブスクリプションの作成またはシステム診断の実行は、複写システム全体が機能していることを確認してから行ってください。

詳細については、『Replication Server 管理ガイド 第 2 巻』の「Replication Server の検証とモニタリング」の「複写システムの検証」を参照してください。

複写定義の作成

複写用にテーブルを設定するには、Adaptive Server で複写するようにマーク付けして、Replication Server にその複写定義を定義します。複写定義は、テーブルを記述し、複写するカラムについての情報を含んでいます。

- ストアド・プロシージャを複写する場合は、プライマリ・データベースとレプリケート・データベースの両方にストアド・プロシージャを作成します。
- プライマリ・データベースからレプリケート・データベースにプロシージャを複写する場合は、プライマリ・データベース内のストアド・プロシージャを複写するようマーク付けします。
- レプリケート・データベースからプライマリ・データベースにプロシージャを複写する場合は、レプリケート・データベース内のストアド・プロシージャを複写するようマーク付けします。

ストアド・プロシージャ用のファンクション複写定義は、ストアド・プロシージャをレプリケート・データベースからプライマリ・データベースに複写する場合であっても、プライマリ Replication Server に作成します。

参照：

- 複写テーブルの管理 (299 ページ)
- 複写ファンクションの管理 (385 ページ)

サブスクリプションの作成

Replication Server にテーブル用の複写定義を作成する場合、そのテーブル複写定義用のサブスクリプションをレプリケート・データベースに作成する必要があります。サブスクリプションは、プライマリ・テーブルから、指定されたレプリケート・データベースにデータをコピーするように Replication Server に指示します。

同じように、ストアド・プロシージャ用のファンクション複写定義を作成する場合は、そのファンクション複写定義用のサブスクリプションをレプリケート・データベースに作成する必要があります。ただし、プライマリ・データベースを更新するテーブルまたはファンクションの複写定義用のサブスクリプションは、作成する必要がありません。

参照：

- サブスクリプションの管理 (405 ページ)

Replication Server 作業の実行

RCL コマンドで **rs_init**、Sybase Central または **isql** を使用して、Replication Server と対話させます。

rs_init を使用すると、新しい Replication Server を設定して、システムに新しいデータベースを追加できます。

Sybase Central の Replication Manager プラグイン・コンポーネントには、Replication Server システムの管理に関連する作業の多くを実行するためのグラフィカル・ユーザ・インタフェースが用意されています。

RCL コマンドは、クライアント・アプリケーションを使用して Replication Server に接続することによって実行します。Sybase Central や **isql** などのユーティリティ・プログラム、または Open Client Client-Library を使用して作成するカスタム・アプリケーション・プログラムを使用できます。

RCL コマンドは Transact-SQL コマンドと似ています。すべての RCL コマンドの完全な構文については、『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」を参照してください。

コマンドの多くは、必要に応じて実行するコマンドなので、**isql** を使用すると便利です。

rs_init の使用

rs_init ユーティリティを使用して、新しい Replication Server を設定し、Adaptive Server データベースを複写システムに追加します。

Replication Server がすでにある場合は、**rs_init** を使用して、新しいバージョンにアップグレードしたり、以前のバージョンにダウングレードすることもできます。**rs_init** は、Sybase ソフトウェアとともにインストールされます。対話型でも、リソース・ファイルでも使用できます。

rs_init を使用する手順については、使用しているプラットフォームの『Replication Server 設定ガイド』を参照してください。

Sybase Central での Replication Server の管理

Sybase Central は、Replication Server のシステム管理ツールです。

Sybase Central は、複写システムのコンポーネントをモニタし、Replication Server の作業を実行できるようにするグラフィカル・ユーザ・インタフェースを提供します。

Sybase Central を使用すると、次の処理を実行できます。

- オブジェクトのグループ化やステータス情報を表示できる、複写システムのトポロジをグラフィカルな表現形式で表示します。Sybase Central には、作業の実行用とオブジェクトのモニタ用のメニューもあります。
- 複数の Replication Server コネクションを表示して、キューの内容を選択的に表示できます。
- Replication Server のコマンド・ラインと **isql** から実行できる作業の多くを、同等の Transact-SQL コマンドや RCL コマンドより高速に実行できます。たとえば、ユーザの管理、ルートとコネクションの作成、複写定義とサブスクリプションの作成、ウォーム・スタンバイ・アプリケーションの管理ができます。

参照：

- Sybase Central での複写環境の管理 (57 ページ)

isql の使用

isql を使用すると、Replication Server と対話することができます。

isql ユーティリティを使用すると、次の処理を実行できます。

- Replication Server の設定ファイルのプライマリ・ユーザ名とパスワードを使用する ERSSD (Embedded Replication Server システム・データベース) コマンドの実行
- RCL コマンドの対話型実行
- テキスト・ファイルに格納されたスクリプトの実行

簡単な操作の場合は、**isql** を対話型で使用するのが最も簡単です。

複雑な操作の場合は、**isql** を使用してスクリプトを実行することをおすすめします。この実行方法によって、Replication Server を設定するために実行した RCL コマンドの記録を残すことができます。スクリプトは、必要なときにいつでも編集して再実行できます。スクリプトは、新しいシステムを確認する場合や障害の原因を調査する場合にも役立ちます。

isql を使用すると、Replication Server または Adaptive Server にログインできます。Replication Server と対話させる対話型およびスクリプトの両方の方法によって、**isql** を使用できます。Adaptive Server で **isql** を使用する方法の詳細については、使用しているオペレーティング・システムの『Adaptive Server ユーティリティ・プログラム』を参照してください。

対話的な isql の使用

簡単な操作の場合は、**isql** を対話的に使用できます。

1. 必要に応じて、Replication Server を起動します。
2. 次のコマンドを使用して Replication Server にログインします。
   ```
   isql -Uuser_name -Ppassword -Sserver_name
   ```
   -S フラグを使用して、Replication Server の名前を指定します。

   ログインが受け入れられると、**isql** から次のプロンプトが表示されます。
   ```
   1>
   ```
3. 実行する RCL コマンドを入力します。

   行末で [Return] キーを押すと、**isql** によって行番号が増えます。コマンドによっては、複数行を必要とするものもあります。
4. コマンドを実行するには、「go」(1 行に単体で、ブランクを入れずに) を入力して、[Return] キーを押します。

   コマンドをキャンセルするには、「reset」を入力して [Return] キーを押します。プロンプトの行番号は 1 にリセットされます。

   RCL コマンドには、すぐに結果が表示されるものがあります。また、非同期に実行されるコマンドもあります。つまり、必要がない場合には要求された処理の完了を待たずにシステムプロンプトを返す場合もあり、単に構文エラーをレポートしてシステムプロンプトを返す場合もあります。
5. **isql** を終了するには、行の始めに「quit」と入力します。

注意： 非同期コマンドのステータスは、ステータスを表示する RCL コマンドを実行するか、対象となるサイトの RSSD システム・テーブルに問い合わせることによって調べることができます。システム・テーブルとそれらに問い合わせるために使用できるストアド・プロシージャの詳細については、『Replication Server リファレンス・マニュアル』の「Replication Server システム・テーブル」を参照してください。

参照：

- Replication Server の起動 (89 ページ)

isql によるスクリプトの実行

RCL コマンドのスクリプトを作成し、そのスクリプトを **isql** を使用して実行することができます。これは、複数のサイトにある Replication Server で同じコマンド・セットを実行する必要がある場合に役立ちます。

1. Replication Server が実行していない場合は、起動します。

2. スクリプト用のテキスト・ファイルを作成して、実行する RCL コマンドをそのファイルに入力します。対話型の場合と同じように、**go** コマンドだけを指定した行で各コマンドを分離します。
3. 次の **isql** 構文を使用してスクリプトを実行します。

```
isql -Uuser_name -Ppassword -Sserver_name
            -iscript_name
```

isql により、標準出力として、画面にスクリプトのコマンドの結果が表示されます。オプションとして、次のように出力をファイルにリダイレクトすることもできます。

```
isql -Uuser_name -Ppassword -Sserver_name
            -iscript_name > output_file
```

参照：

- Replication Server の起動 (89 ページ)

Replication Server の起動

順序通りに、Replication Server と複写システム・コンポーネントを起動します。

Replication Server を再起動する必要があるのは、通常、システム・ファイルを再設定する場合か、Replication Server を停止させるような障害がシステムに発生した場合だけです。最初は、複写システムはインストール・プロセスによって起動されます。

1. Replication Server が管理するデータベースを含むデータ・サーバを起動します。
2. Replication Server が RSSD 用の Adaptive Server Enterprise を使用している場合は、RSSD を起動します。詳細については、使用しているプラットフォームの『Replication Server インストール・ガイド』を参照してください。
3. UNIX システムの場合は **repserver** コマンド、Windows システムの場合は **repsrvr.exe** コマンドを実行するか、Replication Server の実行ファイルを実行することによって、Replication Server を起動します。
4. データ・サーバの起動時に RepAgent が自動的に起動するように設定されていない場合は、データ・サーバ用と RSSD 用に RepAgent を起動します。
5. Replication Server がエラーなく起動したことを確認するには、次のことを行います。
   a) `repserver.log` ファイルにエラー・メッセージ (左端に文字 E が表示される) がないか調べて、『Replication Server 管理ガイド 第 2 巻』の「エラーと

例外の処理」の「エラー・ログ・ファイル」の「Replication Server エラー・ログ」の説明に従います。

b) **isql** を使用して各 Replication Server にログインするか、各サーバにログインするスクリプトを使用してログインします。詳細については、『Replication Server 管理ガイド 第 2 巻』の「Replication Server の検証とモニタリング」の「Replication Server のモニタリング」の「サーバ・ステータスの確認」を参照してください。

参照：

- Replication Server 実行プログラム (90 ページ)

Replication Server 実行プログラム

repserver または **repsrvr.exe** コマンドを使用して、Replication Server プログラムを実行します。

たとえば、**repserver** を実行するには、“sybase” ユーザとしてオペレーティング・システムにログインし、次の構文を使用して **repserver** を実行します。

```
repserver [-C config_file] [-i id_server] [-S rs_name]
    [-I interfaces_file] [-E errorlog_file] [-M] [-v]
    [-K keytab_file]
```

repserver コマンドの各パラメータの詳細については、『Replication Server リファレンス・マニュアル』の「実行プログラム」を参照してください。

rs_init プログラムを実行すると、実行ファイル “RUN_*name*” が作成されます。*name* は Replication Server の名前です。この実行ファイルは、インストールされている Replication Server 用に設定されたパラメータを持つ **repserver** コマンドを指定します。通常は、この実行ファイルを実行することによって Replication Server を起動します。

Replication Server 実行プログラムは `bin` サブディレクトリに、Replication Server `run` ファイルは Sybase リリース・ディレクトリの `install` サブディレクトリにあります。詳細については、使用しているプラットフォームの『Replication Server インストール・ガイド』および『Replication Server 設定ガイド』を参照してください。

Replication Server 設定ファイル

Replication Server は、必要とする起動情報を Replication Server 設定ファイルから見つけます。

このファイルは、**rs_init** プログラムで作成されますが、テキスト・エディタで編集できます。ただし、暗号化されたパスワードを含んでいる場合は、**rs_init** を使用して修正する必要があります。詳細については、使用しているプラットフォームの『Replication Server インストール・ガイド』と『Replication Server 設定ガイド』

を参照してください。Replication Server 設定ファイルのデフォルト名は、Replication Server の名前に “.cfg” を追加したものです。

isql による Replication Server の停止

isql ユーティリティを使用して、Replication Server を停止します。

Replication Server を停止すると、以後のコネクションは拒否され、スレッドが終了して、Replication Server が終了します。

1. **isql** を使用して、システム管理者として Replication Server にログインします。

   ```
   isql -Usa -Psa_password -Sservername
   ```

2. 次のように入力します。

   ```
   shutdown
   go
   ```

Replication Server の追加

Replication Server を複写システムに追加するには、使用しているプラットフォームの『Replication Server インストール・ガイド』と『Replication Server 設定ガイド』の説明に従って **rs_init** プログラムを使用します。

追加する Replication Server が自社のシステムにどのように適合するかについては、常に慎重に検討して分析してください。サーバに必要なその他の処理を調べて、それらの処理に必要な名前とアカウントを指定します。

最初にインストールする Replication Server は、ID サーバでなければなりません。ID サーバは、新しい Replication Server をインストールしたり、複写システムにデータベースを追加したりするときに稼働している必要があります。

Sybase Central を使用すると、複写環境を作成するときに Replication Server を複写システムに追加することができます。

それぞれの Replication Server をインストールするときに、**rs_init** は次のタスクを実行します。

- Replication Server 用の設定ファイルを作成する
- Replication Server を起動するための実行ファイルを作成する
- Adaptive Server で RepAgent パラメータを設定する
- RSSD または ERSSD を作成して初期化する
- 必要に応じて Replication Server と、RSSD 用 RepAgent を起動する

追加する各 Replication Server に対して **rs_init** を実行した後に、次を実行します。

1. Replication Server のルート指定を決定して、新しい Replication Server に合うように既存システムのルートを修正します。
2. 新しいデータベースを追加する場合は、そのデータベースを複写用に準備します。
3. Replication Server コマンド用の適切なパーミッションをユーザに付与します。
4. 必要に応じて、Replication Server 用に複写定義、サブスクリプション、ファンクション文字列クラス、エラー・クラスを追加します。

詳細については、『Replication Server 管理ガイド 第 2 巻』の「データベース・オペレーションのカスタマイズ」および『Replication Server 管理ガイド 第 2 巻』の「エラーと例外の処理」を参照してください。

参照：

- ルートの管理 (161 ページ)
- データベース・コネクションの管理 (189 ページ)
- Replication Server のセキュリティ管理 (233 ページ)
- 複写テーブルの管理 (299 ページ)
- サブスクリプションの管理 (405 ページ)
- 複写環境オブジェクトの作成 (68 ページ)

複写システム・ドメインの追加

複写システム・ドメインには、同じ ID サーバを使用するすべての複写システム・コンポーネントが含まれます。

ほとんどの複写システムは、1 つの ID サーバを持つ 1 つのドメインとして設定されます。ただし、次のような状況では、2 つの別々のデータ環境のレプリケートを必要とすることがあります。

- 企業が、別々のグループ、サイト、または独立の組織によるデータ管理を必要とする場合
- ID サーバが単一障害発生点となることを避け、フォールト・トレラントなシステムを作成する必要がある場合
  ドメイン内の ID サーバに障害が発生すると、システムの性能が低下します。ID サーバが停止している間は、新しい Replication Server やデータベースをドメインに追加することはできません。

複数の複写システム・ドメインを使用する場合は、完全に独立したデータ環境にしてください。たとえば、データ環境記録担当者のグループと記録一覧のグループが存在するとします。これら 2 つのグループ間にデータの共有も関係もない場合には、それぞれのデータ環境に 1 つずつ別々のドメインを作成できます。

複写システム・ドメイン追加のガイドライン

複数の複写システム・ドメイン用に複数の ID サーバを作成する前に、特定のガイドラインに従う必要があります。

- すべての Replication Server 名とデータ・サーバ名は、複数のドメイン全体でユニークな名前にします。
  ユニークな名前を使用することによって、特に、サーバ用のネットワーク・アクセス情報を格納している interfaces ファイルの管理を簡単にして混乱を避けることができます。
- Replication Server またはデータベースが複写システムに追加されるたびに ID 番号を割り当てる必要があります。
- 将来のドメイン間データ転送の可能性 (つまり、ドメインのマージ) に対応できるように、ユニークな名前と重複しない ID 番号を維持します。
  - 各ドメインに対して、異なる範囲のデータベースと Replication Server の ID 番号を使用します。
  - 追加ドメインの ID 番号を、最初のドメインの範囲と重複しないように、十分な大きさの値にします。
- 複写定義の名前を、ID サーバのドメイン内とドメイン間全体でユニークな名前にします。

ID 番号の割り当て例

複写環境のそれぞれの ID サーバに指定された範囲内でのみ ID 番号を使用することができます。

ID 番号は、Replication Server またはデータベースが複写システムに追加されるたびに増やされます。デフォルトでは、データ・サーバの最初の ID 番号は 101 です。Replication Server の場合は、16,777,317 です。データ・サーバで可能な最後の ID 番号は 16,777,316 です。Replication Server で可能な最後の番号は、33,554,431 です。

2 つのドメインを作成している場合は、テーブルに従って ID 番号を割り当てることができます。

表 2 : 複数の ID サーバに対する推奨 ID 番号

コンポーネント	最初の ID 番号	最後の ID 番号
1 番目のドメイン・データ・サーバ	101	99,999
2 番目のドメイン・データ・サーバ	100,000	16,777,316
1 番目のドメイン Replication Server	16,777,317	17,777,316

コンポーネント	最初の ID 番号	最後の ID 番号
2 番目のドメイン Replication Server	17,777,317	33,554,431

rs_init プログラムを使用して ID サーバをインストールする場合は、最初の Replication Server ID と最初のデータベース ID を指定できます。

注意： 使用する範囲がドメイン間で重複しないようにしてください。ID 番号が増えても次の範囲と重複することがないように、十分な大きさの範囲を使用してください。たとえば、範囲が 99,999 まであると、ほぼすべての状況に対応できます。

Replication Server 設定パラメータの設定

複写システム内の Replication Server または特定のオブジェクトは、RSSD または ERSSD 内にある `rs_config` システム・テーブルの設定パラメータを更新するいくつかの方法を使用することで設定できます。

また、このテーブルの設定ステータス情報を確認することもできます。

注意： Replication Server の起動情報は、**rs_init** で作成される設定ファイルに格納されます。Replication Server 設定ファイルのデフォルト名は、Replication Server の名前に "`.cfg`" を追加したものです。このファイルに格納される設定パラメータの詳細については、『Replication Server リファレンス・マニュアル』の「実行プログラム」を参照してください。

Replication Server 設定パラメータ

Replication Server は、RSSD または ERSSD 内の `rs_config` システム・テーブルから設定パラメータを読み込みます。

設定パラメータはすべてデフォルト値を持ち、デフォルト値は、**rs_init** ユーティリティを使用して RSSD または ERSSD を作成するとテーブルに挿入されます。デフォルト値は、ほとんどの複写システムにとって十分な値です。通常、デフォルト値を変更する必要があるのは、独特の環境や特別な状況の場合だけです。たとえば、システムに多数の複写定義やサブスクリプションがある場合、パフォーマンスをチューニングするためにパラメータを調整しなければならない場合があります。デフォルト値は、**configure replication server** を使用して変更できます。

Replication Server とそのコンポーネントの名前とバージョン番号を制御する基本的なパラメータは、**rs_init** でも設定できます。

rs_init は、パスワード暗号化設定パラメータも設定します。

多くの設定パラメータは、特定のオブジェクトに対する値も持っています。これらの値は、インストール後に、これらのパラメータが許容する範囲内でシステム

を微調整する必要がある場合に設定します。たとえば、デフォルトのルート・パラメータは、現在の Replication Server を起点とするすべてのルートに影響します。必要に応じて、**configure replication server** を使用してこれらのパラメータのデフォルト設定を変更します。また、**alter route** コマンドを使用して、各ルートのパラメータ値を個別に設定することもできます。

設定パラメータによっては、複写システムについて技術的に理解していなければならないものもあります。

RSSD は定期的にバックアップすることが重要です。また、その状態に何か変更を加えるときも常にバックアップします。ERSSD は、毎日自動バックアップするように設定されています。

『Replication Server 管理ガイド 第 2 巻』の「パフォーマンス・チューニング」を参照してください。『Replication Server 管理ガイド 第 2 巻』の「パフォーマンス・チューニング」の「パフォーマンスに影響する設定パラメータ」を参照してください。

参照：

- Replication Server の技術的概要 (27 ページ)
- RSSD の管理 (99 ページ)
- Embedded Replication Server システム・データベースの管理 (101 ページ)

基本的な設定パラメータ

基本的なパラメータは、Replication Server とそのコンポーネントの名前とバージョン番号を制御します。

警告！ 基本的な設定パラメータの値は、変更しないでください。これらの値は、**rs_init** を実行すると設定されるので、変更してよいのは、Replication Server をアップグレードまたはダウングレードするときに **rs_init** プログラムによって修正される場合だけです。

表 3 : 基本的な設定パラメータ

設定パラメータ	説明
current_rssd_version	この RSSD でサポートされる Replication Server のバージョン。Replication Server は、起動時にこの値をチェックする。
id_server	この Replication Server の ID サーバの名前。
minimum_rssd_version	この RSSD を使用できる Replication Server の最小バージョン。**current_rssd_version** が Replication Server のバージョンよりも大きい場合は、Replication Server の起動時にこの値がチェックされる。

設定パラメータ	説明
oserver	現在の Replication Server の名前。
prev_min_rssd_version	**rs_init** インストール・アップグレードに従って、この値には **minimum_rssd_version** の前の値が入っている。
prev_rssd_version	**rs_init** インストール・アップグレードに従って、この値には **current_rssd_version** の前の値が入っている。
rssd_error_class	RSSD のエラー・クラス。デフォルト値は **rs_sqlserver_error_class**
send_enc_pw	Replication Server が暗号化されたパスワードを使用して RSSD へのクライアント・コネクションを作成できるようにする。値は “on” または “off” (デフォルト)。

参照：

• Replication Server クライアント・コネクションへの暗号化パスワードの送信 (239 ページ)

さまざまなタイプの設定パラメータ

Replication Server とさまざまなデータベース・オブジェクトに影響を与える異なるタイプの設定パラメータが `rs_config` システム・テーブルにあります。

パラメータを変更する方法も、そのパラメータが影響するオブジェクトによって異なります。さまざまなタイプの設定パラメータを、次に示します。

- ローカル Replication Server ― 現在の Replication Server だけに影響するパラメータです。これらには、基本的な設定パラメータと『Replication Server 管理ガイド 第 2 巻』の「パフォーマンス・チューニング」の「パフォーマンスに影響する設定パラメータ」の「パフォーマンスに影響する Replication Server パラメータ」でリストされているパラメータが含まれます。
- ルート ― 現在の Replication Server から他の Replication Server へのルートに影響するパラメータです。
- データベース・コネクション ― Replication Server を起点とするデータベース・コネクションに影響するパラメータです。詳細については、『Replication Server 管理ガイド 第 2 巻』の「パフォーマンス・チューニング」の「並列 DSI スレッドの使用」の「並列 DSI パラメータ」を参照してください。
- 論理データベース・コネクション ― ウォーム・スタンバイ・アプリケーションに対する論理データベース・コネクションに適用される Replication Server パラメータです。論理コネクション・パラメータに対するデフォルト値とターゲットごとの値の設定方法の詳細については、『Replication Server 管理ガイド 第 2 巻』の「ウォーム・スタンバイ・アプリケーションの管理」の「Alter

Warm Standby Database Connections」の「Alter Logical Connections」の「論理コネクションに影響を与えるパラメータの変更」を参照してください。
- ネットワークベース・セキュリティ・サービス ― ネットワーク・セキュリティに影響するパラメータです。
- パフォーマンス ― Replication Server のパフォーマンスに影響するパラメータです。詳細については、『Replication Server 管理ガイド 第 2 巻』の「パフォーマンス・チューニング」の「パフォーマンスに影響する設定パラメータ」および『Replication Server 管理ガイド 第 2 巻』の「パフォーマンス・チューニング」の「並列 DSI スレッドの使用」の「並列 DSI のパラメータ」を参照してください。

参照：
- Replication Server パラメータの変更 (97 ページ)
- ルートの変更 (175 ページ)
- 物理コネクションに影響するパラメータの設定と変更 (199 ページ)
- ネットワークベース・セキュリティの管理 (262 ページ)
- 基本的な設定パラメータ (95 ページ)

Replication Server パラメータの変更

現在の Replication Server に影響する設定パラメータは、Replication Server で **configure replication server** コマンドを使用することによって変更できます。

configure replication server を使用してデフォルト設定パラメータ値を変更するには、Replication Server にログインして、**isql** プロンプトで **configure replication server** を実行します。

次の構文を使用します。ここで、*config_param* は設定パラメータの名前を示す文字列であり、*value* はパラメータに設定する値を表す文字列です。

```
configure replication server
        set config_param to 'value'
```

config_param 文字列は、パラメータ名全体と一致していなければなりません。新しいパラメータを有効にするために、Replication Server を再起動する必要がある場合があります。

例 1

たとえば、Open Server メッセージ・キューに入れられるメッセージの最大数を 5 に変更するには、送信元 Replication Server にログインしてから、次の手順に従います。

1. **configure replication server** コマンドを実行します。
   ```
   configure replication server set num_msgs to '5'
   ```

2. Replication Server を再起動します。

例 2

この例では、**configure Replication Server** を使用して、**ha_failover** パラメータを変更し、Replication Server から Adaptive Server への RSSD 以外のすべてのコネクションに対するフェールオーバのサポートを有効にします。

1. **configure replication server** を実行します。フェールオーバ・サポートを有効にする Replication Server にログインし、次のように入力します。

   ```
   configure replication server
   set ha_failover to 'on'
   ```

   詳細については、『Replication Server 管理ガイド 第 2 巻』の「複写システム・リカバリ」の「Sybase フェールオーバをサポートするための複写システムの設定」を参照してください。
2. Replication Server を再起動します。

設定の変更は、Replication Server を再起動したあとに有効になります。

『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**configure replication server**」を参照してください。

セキュリティに影響するパラメータがあります。パフォーマンスに影響を与えるパラメータについては、『Replication Server 管理ガイド 第 2 巻』の「パフォーマンス・チューニング」を参照してください。

参照：

- Replication Server のセキュリティ管理 (233 ページ)
- Replication Server の起動 (89 ページ)

動的パラメータの設定

動的設定パラメータでは、新しい値を有効にするために、Replication Server を再起動する必要がなくなりました。パラメータ値を変更するには、**configure replication server** を使用します。

これらのパラメータの値を取得するには、**admin config** コマンドを使用します。

admin config の構文は次のとおりです。

```
admin config [,"connection" |,"logical_connection" |,"route" ]
[,server
[,database]] [,configuration_name]
```

このコマンドの使用方法の詳細については、『Replication Server リファレンス・マニュアル』』を照してください。

動的設定パラメータ
動的設定パラメータの **configure replication server** を使用して、Replication Server を停止および再起動せずに、Replication Server の動作に影響を与えます。

表 4 : 動的設定パラメータ

init_sqm_write_delay	**init_sqm_write_max_delay**
memory_limit	**num_concurrent_subs**
queue_dump_buffer_size	**sqm_recover_segs**
sqm_warning_thr_ind	**sqm_warning_thr1**
sqm_warning_thr2	**sqt_max_cache_size**
sqt_init_read_delay	**sqt_max_read_delay**
sts_cachesize	**sts_full_cache**_system_table_name_

RSSD の管理

各 Replication Server の RSSD 内のデータは、複写システムを稼働させ続けるために必須のものです。

複写システム管理者または Adaptive Server のシステム管理者は、データベースの状態をモニタして定期的にダンプを実行することによって、RSSD を管理します。災害リカバリでは、システムを完全にリカバリするために、RSSD の最新のバックアップを使用する必要があります。したがって、複写システムを定期的にバックアップすることが非常に重要です。

また、ルート、複写定義、サブスクリプションの追加、または接続先データベースのファンクション文字列を変更するなどの状態を変更する作業を実行したあとに、RSSD をバックアップすることも重要です。

システム・テーブルは、Replication Server のインストール時に RSSD にロードされます。システム・テーブルに問い合わせてシステムのステータスを調べることはできますが、通常は、そのテーブルに直接変更を加えないでください。システム・テーブルの詳細については、『Replication Server リファレンス・マニュアル』を参照してください。

RSSD コネクションに対するフェールオーバ・サポートの有効化

Sybase フェールオーバを使用して、バージョン 12.0 以降の 2 つの Adaptive Server をコンパニオンとして設定できます。プライマリ・コンパニオンに障害が発生した場合、そのプライマリ・コンパニオンのデバイス、データベース、およびコネクションがセカンダリ・コンパニオンに引き継がれます。

Adaptive Server における Sybase フェールオーバの動作の詳細については、『高可用性システムにおける Sybase フェールオーバの使用』を参照してください。これは、Adaptive Server Enterprise のマニュアル・セットの一部です。

RSSD コネクションに対するフェールオーバ・サポートを有効にするには、以下のいずれかの方法に従ってください。

- Replication Server を新規にインストールするときに、**rs_init** を使用します。手順については、使用しているプラットフォームの『Replication Server 設定ガイド』の「rs_init による Replication Server の設定とデータベースの追加」を参照してください。
- Replication Server をインストールした後に Replication Server の設定ファイルを編集します。
  1. テキスト・エディタを使用して、Replication Server の設定ファイルを開きます。デフォルトのファイル名は、Replication Server の名前に拡張子 "`.cfg`" が付いたものです。
     設定ファイルは 1 行 1 エントリになっています。
  2. "`RSSD_ha_failover=no`" という行を検索して、次のように変更します。

     ```
     RSSD_ha_failover=yes
     ```
  3. RSSD コネクションに対するフェールオーバ・サポートを無効にするには、"`RSSD_ha_failover=yes`" という行を次のように変更します。

     ```
     RSSD_ha_failover=no
     ```

     変更内容はすぐに反映されます。つまり、フェールオーバ・サポートを有効にするために Replication Server を再起動する必要はありません。

RSSD を最新の状態にリカバリできない場合は、『Replication Server 管理ガイド 第 2 巻』の「複写システム・リカバリ」の「RSSD 障害からのリカバリ」を参照してください。

注意： プラットフォーム間の **dump** と **load** または **bcp** などのコマンドを使用した RSSD データベースのマイグレーションはできません。移行するには、新しいプラットフォームでの複写システムの再構築が必要です。

RSSD 以外の Replication Server から Adaptive Server へのコネクションに対してフェールオーバのサポートを有効にする方法については、『Replication Server 管理ガイド 第 2 巻』の「複写システム・リカバリ」の「Sybase フェールオーバをサポートするための複写システムの設定」を参照してください。

Embedded Replication Server システム・データベースの管理

Replication Server は、Adaptive Server Enterprise Replication Server システム・データベース (RSSD) 上でも、Embedded (埋め込み) RSSD (ERSSD) 上でも動作します。ERSSD は、Replication Server RSSD の管理に Adaptive Server Enterprise を使用することを望まないユーザ向けに設計されています。

ERSSD は、Replication Server のインストールおよび管理を簡単に行えます。ERSSD を使用するように指定すると、バックグラウンドで自動的にインストール、設定、起動が行われます。バックアップ手順は自動化され、事前に設定されます。

留意： ERSSD から RSSD へはマイグレートできません。ERSSD を使用するには、Replication Server のインストール時に ERSSD を選択してください。『Replication Server インストール・ガイド』を参照してください。

ERSSD は、Replication Server のオプションとして SQL Anywhere に実装されます。Sybase は、Adaptive Server Enterprise に実装されている従来の RSSD も引き続きサポートします。ERSSD の機能はすべて、ERSSD だけに該当し、Adaptive Server Enterprise に実装されている従来の RSSD の動作には影響しません。

ERSSD は、次の 3 つのオペレーティング・システムで実行されます。

- データベース・ルート・ファイル
- トランザクション・ログ・ファイル
- トランザクション・ログのミラー・ファイル

rs_init の起動時に、これらのファイルのディレクトリを指定し、使用している ERSSD の名前が interfaces ファイルにあることを確認してください。

ヒント： パフォーマンスを向上させ、ディスク障害から保護するため、これらのファイルはそれぞれ別の物理デバイスに配置してください。

ERSSD 設定に関する情報の取得

設定の取得方法と ERSSD に関するその他の情報について説明します。

ERSSD には、事前設定されたバックアップ・タイム、バックアップ間隔、バックアップ・ディレクトリがあります。これらのデフォルト値を変更しないのであれば、ERSSD を設定する必要はありません。バックアップ・ディレクトリには、ファイルが 4 つあります。このディレクトリは、ERSSD とともに Replication Server をインストールするときに指定します。現在のデフォルト値を確認するには、次を入力します。

```
sysadmin erssd
```

Replication Server 設定ファイルには、次のようなパスがあります。

- ERSSD データベース・ファイルのパス
- ERSSD トランザクション・ログ・ファイルのパス
- ERSSD トランザクション・ログ・ミラー・ファイルのパス
- バックアップ・ディレクトリのパス

ERSSD 設定パラメータとコマンド

ERSSD 設定パラメータを使用して、バックアップ・タイムとバックアップ・ディレクトリ、および ERSSD ルーティングを設定します。

構文

```
configure replication server
set
     {erssd_backup_start_time |
     erssd_backup_start_date |
     erssd_backup_dir |
     erssd_backup_interval | erssd_ra}
to 'value'
```

警告！ これらの値を **rs_config** テーブルに対して直接更新しないでください。

表 5 : ERSSD 設定パラメータ

パラメータ	値	デフォルト
erssd_backup_start_time	バックアップ開始時刻。 指定形式は、12 時間制の “hh:mm AM” か “hh:mm PM”、または 24 時間制の “hh:mm”。	01:00 AM
erssd_backup_start_date	バックアップ開始日。 指定形式は “MM/DD/YYYY”。	本日
erssd_backup_interval	データベースとログのバックアップ間隔。 指定形式は “nn hours”、“nn minutes”、“nn seconds” のいずれか。	24 hours

パラメータ	値	デフォルト
erssd_backup_dir	バックアップ・ファイルを格納するロケーション。 ディレクトリのフル・パスを指定する。このパスを設定すると、予定していなかったバックアップをただちに実行できる。	トランザクション・ログ・ミラーと同じディレクトリで、初期値は **rs_init** で指定する。
erssd_ra	値としてサーバ名を指定する。ユーザが現在のサイトからルートを作成した場合のみ使用される。	*erssd_name*_**ra**。ここで、*erssd_name* はユーザの複写システムの ERSSD 名。

Backup ERSSD

自動フル・バックアップは、データベース・ファイルとトランザクション・ログ・ファイルの両方を対象としており、デフォルト時刻またはユーザが設定した時刻に実行されます。ERSSD のスケジュールされていないバックアップを実行することができます。

バックアップ・ディレクトリには、ファイルが 4 つあります。このディレクトリは、ERSSD とともに Replication Server をインストールするときに指定します。

トランザクション・ログはミラーリングされていて、重要なデータは特別に保護されるので、トランザクション・ログ・ファイルは完全にリカバリできます。

ERSSD のスケジュールされていないバックアップを実行するには、次のように入力します。

```
sysadmin erssd, backup
```

ERSSD バックアップ・ディレクトリのファイル

Replication Server バックアップ・ディレクトリには、ERSSD データベースとトランザクション・ログのバックアップ・ファイルが含まれます。バックアップ・ディレクトリは、ERSSD とともに Replication Server をインストールするときに指定します。

表 6 : ERSSD のバックアップ・ディレクトリのファイル

ファイル名	ファイル定義
erssd_name.**db**	バックアップ・データベース・ファイル
erssd_name.**log**	バックアップ・トランザクション・ログ
erssd_name.**db.pre**	以前のバックアップ・データベース・ファイル
erssd_name.**log.pre**	以前のバックアップ・トランザクション・ログ

ERSSD のルート指定

create route コマンドを使用して、ERSSD を使用して Replication Server からルートを作成します。

送信元サーバおよび送信先サーバのバージョンがどちらも 15.0 以降であれば、ERSSD を使用して Replication Server からルートを作成できます。Replication Agent 名が Replication Server の interfaces ファイルに記載されていること、また、**create route** の実行時に ERSSD Replication Agent が Open Server として起動されていることを確認してください。名前が interfaces ファイルに記載されていない場合はコマンドが失敗します。

デフォルトの ERSSD Replication Agent 名は *erssd_name*_**ra** です。デフォルト名を、使用している Replication Agent サーバの名前で置き換えるには、次のように入力します。

```
configure replication server
set erssd_ra to 'value'
```

注意： Sybase では、ERSSD を SQL Anywhere (SA) のオプションとして提供しています。また、Adaptive Server Enterprise では従来の RSSD も引き続きサポートしています。

ERSSD ファイルの移動

sysadmin erssd コマンドを使用して、ERSSD データベース・ファイル、トランザクション・ログ、またはトランザクション・ログ・ミラーを移動します。

設定ファイル自体は編集しないでください。データベース・ファイル、トランザクション・ログ、トランザクション・ログ・ミラーの移動は、負荷のかかるオペレーションです。本当に必要な場合だけ実行してください。**sysadmin erssd** の詳細については、『Replication Server リファレンス・マニュアル』を参照してください。

ERSSD ユーザ管理

ERSSD ユーザ・パスワードを変更し、ユーザを追加し、パーミッションを割り当てることができます。

ERSSD には、システム管理者の役割も果たすプライマリ・ユーザと、メンテナンス・ユーザの 2 種類のユーザしか存在しません。名前とパスワードは、設定ファイルに記述されています。次の操作によってユーザ・パスワードを変更できます。

- **alter user** を使用して、プライマリ・ユーザのパスワードを変更します。
- **alter connection** を使用して、メンテナンス・ユーザのパスワードを変更します。

どちらのコマンドも、Replication Server と ERSSD とでパスワードを変更し、Replication Server 設定ファイルを更新します。

これらのコマンドの詳細については、『Replication Server リファレンス・マニュアル』を参照してください。

ERSSD でユーザを追加するには、プライマリ・ユーザとして **isql** を使用して ERSSD にアクセスし、**grant connect to** *username* **identified by** *password* コマンドを実行します。

Replication Server システム・テーブルを読み込むパーミッションをユーザに与えるには、**grant membership in group rs_systabgroup to** *username* コマンドを実行します。

ユーザに **sa** 権限を付与するには、**grant dba to** *username* コマンドを実行します。

ERSSD ファイル・サイズの縮小

sysadmin erssd の **defrag** パラメータを使用して、未使用スペースをリカバリし、ERSSD サイズを縮小して、ERSSD データベース・ファイルを再構築します。

Replication Server が ERSSD からローを削除 (たとえば、Replication Server が例外ログから例外を削除) しても、ERSSD データベースは ERSSD ファイルシステムにスペースを解放しません。したがって、ディスク領域をリカバリするために ERSSD データベースをデフラグする必要があります。

defrag パラメータは **dbunload** SQL Anywhere コマンドを使用して、ERSSD を再構築します。

ERSSD をデフラグする前に、次のことを実行します。

- デフラグには、元の ERSSD の作業領域にできるだけ多くのスペースが必要なように、ERSSD に十分なディスク領域があることを確認します。
- サイト・バージョンを 15.0 以降に設定して、**defrag** パラメータを使用できるようにします。
- 複写システムがビジー状態ではないことを確認します。

sysadmin erssd を実行すると、**defrag** は次を実行します。

1. eRSSD を停止します。
2. **dbltm** が実行中の場合、ERSSD Replication Agent の **dbltm** を停止します。
3. **dbunload** を呼び出して、ERSSD データベース・ファイルを再構築し、デフラグします。**dbunload** は古いトランザクション・ログを、トランザクション・ログ・ディレクトリに `erssdName.olg` として保存します。
4. ERSSD を起動します。

dREC リカバリ・デーモンは、ERSSD の **dbltm** を再起動します。

注意： デフラグの処理中、トランザクション・ログ・ディレクトリ内にファイルがさらに生成される場合があります。これらのファイルを削除しないでください。ファイルが必要ではない場合は、Replication Server および **dbltm** はそのファイルを削除します。

ERSSD リカバリ手順

ERSSD は、オペレーティング・システムのクラッシュ、データベース・サーバのクラッシュ、異常停止によるクラッシュからのリカバリを自動管理します。メディア障害によって損害を被ったデータベースをリカバリし、メディアに障害が発生した後にクリーン・リカバリを実行するように設計されている、いくつかの手順を使用することができます。

詳細については、『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**sysadmin erssd**」を参照してください。

Replication Server のリカバリの前提条件

リカバリ・コマンドを使用する前に、関連する環境変数を設定します。

UNIX プラットフォームの場合

- 環境変数 PATH を設定して、`$SYBASE/$SYBASE_REP/ASA11/bin` を含めます。

```
setenv PATH $SYBASE/$SYBASE_REP/ASA11/bin:$PATH
```

- 環境変数 LD_LIBRARY_PATH (HP の場合は SHLIB_PATH、AIX の場合は LIB_PATH) を設定して、`$SYBASE/$SYBASE_REP/ASA11/lib` を含めます。

```
setenv LD_LIBRARY_PATH $SYBASE/$SYBASE_REP/ASA11/lib:
$LD_LIBRARY_PATH
```

Windows の場合

- 環境変数 PATH を設定して、`%SYBASE%¥%SYBASE_REP%¥ASA11¥win32` を含めます。

```
set PATH=%SYBASE%¥%SYBASE_REP%¥ASA11¥win32;%PATH%
```

データベース・ファイルのメディア障害からのリカバリ

ERSSD データベース・ファイルをリカバリすることができます。

1. 現在のトランザクション・ログのバックアップ・コピーをもう 1 つ余分に作成します。データベース・ファイルが破損してしまった場合、最後のバックアップ以降に加えられた変更内容の記録は、トランザクション・ログのみに存在します。
2. リカバリ処理中に使用するファイルを入れておくためのリカバリ・ディレクトリを作成します。

3. 最新のフル・バックアップからデータベース・ファイルをリカバリ・ディレクトリにコピーします。データベース・ファイルは、バックアップ・ディレクトリにあります。名前は、*erssd_name*.db です。
4. バックアップ・トランザクション・ログをリカバリ・ディレクトリにコピーします。バックアップ・トランザクション・ログは、*erssd_name*.log という名前で、バックアップ・ディレクトリ内にあります。
5. バックアップ・トランザクション・ログからリカバリ・データベースに、トランザクションを適用します。

```
dbsrv11 erssd_name.db -a erssd_name.log
```

6. オンライン・トランザクション・ログをリカバリ・ディレクトリにコピーします。オンライン・トランザクション・ログは、*erssd_name*.log という名前で、ログ・ディレクトリ内にあります。
7. オンライン・トランザクション・ログからリカバリ・データベースに、トランザクションを適用します。

```
dbsrv11 erssd_name.db -a erssd_name.log
```

8. データベース・ファイルのコピーをもう 1 つ作成して、リカバリ後バックアップを作成します。
9. データベース・ファイルを運用ディレクトリに移して、データベースを再起動します。Replication Server エラー・ログから **dbspawn** コマンドを使用します。
10. リカバリ・データベースに対して妥当性検査を実施します。

```
dbvalid -c
"uid=primary_user_name;
pwd=primary_user_password;eng=erssd_name
LINKS=tcpip
(DOBROAD=NONE;HOST=localhost;PORT=port)"
```

11. Replication Server を再起動します。

データベース・トランザクション・ログに対するメディア障害からのリカバリ

ERSSD データベース・トランザクション・ログをリカバリすることができます。

1. 破損したファイルを特定します。特定するには、トランザクション・ログとそのミラー・ログの両方に対して Log Translation ユーティリティを実行して、どちらのログがエラー・メッセージを生成しているのかを確認します。次の例では、Log Translation ユーティリティ **dbtran** によって、トランザクション・ログ *erssd_name*.log が変換され、変換後の出力が *db_name*.sql に保存されます。

   ```
   dbtran erssd_name.log
   ```

   Log Translation ユーティリティでは、破損していないファイルはエラーなしで変換されますが、破損しているファイルの場合は変換時にエラーが発生します。

2. 破損しているファイルを正常なファイルで上書きコピーして、2 つのファイルを同一の内容にします。
3. Replication Server エラー・ログの **command** を使用して、データベースを再起動します。
4. Replication Server を再起動します。

ERSSD のデータベース・トランザクション・ログの再起動

Replication Agent が実行中で、既存のルートが存在する場合、ログに対するメディア障害からリカバリした後に ERSSD のデータベース・トランザクション・ログを再起動することができます。

ERSSD は SQL Anywhere データベースです。この手順のシステム・プロンプトで、**dblog**、**dbeng11**、および **dbstop** SQL Anywhere コマンドを実行する必要があります。SQL Anywhere のマニュアルを参照してください。

1. **shut down** を使用して、Replication Server を停止します。
2. 次のように、システム・プロンプトで ERSSD データベース・ログを再起動します。

   ```
   dblog -il erssd_name.db
   ```
3. 次のように、システム・プロンプトで ERSSD データベース・サーバを起動します。

   ```
   dbeng11 erssd_name.db
   ```
4. 次のように、ERSSD データベースのロケータ値をゼロ (0) にリセットします。
   a) **isql** を起動します。
   b) 次のように、ERSSD データベースで rs_zeroltm を実行します。

      ```
      rs_zeroltm erssd_name, erssd_name
      ```
   c) **isql** を終了します。
5. 次のように、システム・プロンプトで ERSSD データベースを停止します。

   ```
   dbstop -c
   "eng=erssd_name;uid=primary_user_name;
   pwd=primary_user_password"
   ```
6. Replication Server を起動します。

参照：

- Replication Server の起動 (89 ページ)
- isql による Replication Server の停止 (91 ページ)

Replication Server のクワイス

複写システムを「クワイス」(静止)するということは、送受信するメッセージを保持している Replication Server がシステムに存在しない状態にすることを意味します。

データベースのリカバリ、ルートの変更、システムのトラブルシューティングを行うために、システム内のすべての Replication Server をクワイスしなければならない場合があります。Replication Server は、次のような場合にクワイス状態になります。

- サブスクリプション・マテリアライゼーション・キューが存在しない場合
- Replication Server がすべてのキュー内のすべてのメッセージの読み取りを完了した場合
- インバウンド・キューのトランザクション・キャッシュに完了したトランザクションがない場合
- RSI キュー内のメッセージが送信されて、受信が確認された場合
- DSI キュー内のメッセージが適用されて、受信が確認された場合

複写システムのクワイス

一連のコマンドまたは Sybase Central を使用すると、複数の Replication Server で構成されるシステムをクワイスできます。

1. 各 Replication Server で **suspend log transfer from all** コマンドを実行します。これによって、RepAgent は Replication Server に接続できなくなります。
2. 各 Replication Server で **admin quiesce_force_rsi** を実行します。

   このコマンドは、Replication Server にキュー内のすべてのメッセージを他の Replication Server に配信させてから、システムのクワイスに成功したかどうかをレポートします。

   クワイス状態にするには、データのフローに従うのが最も効率的です。たとえば、データが TOKYO_RS から MANILA_RS を経由して SYDNEY_RS に移動する場合には、Replication Server をこの順序でクワイス状態にします。
3. **admin quiesce_check** を使用して、Replication Server がクワイス状態であることを確認します。必要に応じて、すべての Replication Server がクワイス状態になるまで手順 2 と 3 を繰り返します。
4. すべての Replication Server がクワイス状態になったら、各 Replication Server でもう一度 **admin quiesce_force_rsi** を実行します。**admin quiesce_check** を使用して、各 Replication Server がクワイス状態であるかどうかを調べます。必要に応

じて、すべての Replication Server がクワイス状態になるまでこの手順を繰り返します。

この手順が必要なのは、1 つの Replication Server がクワイス状態になった場合でも、その直前に別の Replication Server にメッセージを送信していることがあるためです。これらのメッセージによって、これら 2 つの Replication Server 間または複写システム内の複数の Replication Server 間で、さらに通信が始まることがあります。手順 2 と 3 を繰り返すことによって、複写システム全体を確実にクワイスできます。

Replication Server の削除

複写システムから Replication Server を削除する方法は、Replication Server がアクティブ (動作中) かどうかによって異なります。アクティブな Replication Server を削除するのが最も簡単な方法です。Replication Server を削除する場合、ルートとサブスクリプションを削除する必要があります。

参照：
- サブスクリプションの管理 (405 ページ)
- ルートの管理 (161 ページ)

アクティブな Replication Server の削除

サブスクリプション、複写定義、コネクション、ルートを削除し、アクティブな Replication Server をサービスから安全に削除するために、適切な順序でその他のタスクを実行する必要があります。

1. プライマリ Replication Server (サービスから削除するサーバ) で定義されている複写定義を調べるために、RSSD に問い合わせます。これは、**rs_helprep** ストアド・プロシージャを使用して実行できます。詳細については、『Replication Server リファレンス・マニュアル』の「RSSD ストアド・プロシージャ」の「**rs_helprep**」および『Replication Server リファレンス・マニュアル』の「Replication Server システム・テーブル」を参照してくだい。
2. サブスクリプションと複写定義を削除します。
   a) プライマリ Replication Server で定義されている各複写定義について、データのサブスクリプション作成を管理するすべての Replication Server の各サブスクリプションに対して、**drop subscription** コマンドを実行します。

      レプリケートにあるデータを保持するには、**drop subscription** コマンドに **without purge** を指定して実行します。

レプリケートにあるデータを削除するには、**drop subscription** コマンドに **with purge** を指定して実行します。

b) 対象の Replication Server で管理するプライマリ・データ用のすべての複写定義 (手順 1 で調べたもの) を削除します。

Replication Server がルート指定をしている Replication Server の RSSD から複写定義が削除されるまで待ちます。

c) 削除している Replication Server で、他の Replication Server に関する複写定義に対するすべてのサブスクリプションを削除します。

レプリケートにあるデータを保持するには、**drop subscription** コマンドに **without purge** を指定して実行します。

レプリケートにあるデータをパージするには、**drop subscription** コマンドに **with purge** を指定して実行します。

3. 対象の Replication Server がファンクション文字列クラス用またはエラー・クラス用のプライマリ Replication Server である場合は、別の Replication Server で **move primary** コマンドを実行して、それぞれのクラス用のプライマリ Replication Server を変更します。

move primary オペレーションの実行中は、以前のプライマリ・サイトから新しいプライマリ・サイトへ、そして新しいプライマリ・サイトから以前のプライマリ・サイトへのルートが存在している必要があります。プライマリ・サイトの役割を果たす予定の Replication Server にも、以前のプライマリ・サイトと同じ Replication Server すべてに対するルートが必要です。

4. データベース・コネクションを削除します。

a) Adaptive Server で **sp_stop_rep_agent** システム・プロシージャを使用して、Replication Server に接続されているすべての RepAgent を停止します。

b) **drop connection** コマンドを使用して、この Replication Server で管理されているすべてのデータベースに対するコネクションを削除します。

注意： サービスから削除された Replication Server によってこれまで管理されていたデータベース内のレプリケート・データを管理し続ける場合は、どれか他の Replication Server からそれらのデータベースへのコネクションを作成して、新しいサブスクリプションを作成する必要があります。

5. 次のルート指定作業を実行します。

a) 対象の Replication Server が 1 つのルート上の中間サイトである場合は、**alter route** コマンドを使用して、中間サイトではなくなるようにします。

b) 対象の Replication Server *から*始まるルートをすべて削除します。

このためには、Replication Server から別の Replication Server へのルートごとに、**drop route** コマンドを実行します。

c) 対象の Replication Server へのすべてのルートを削除します。

このためには、削除対象の Replication Server までのルートを持っている各 Replication Server に対して **drop route** コマンドを実行します。

6. 対象の Replication Server に対するサブスクリプションとルートがすべて削除されたあとに、ID サーバによって管理されているリストから対象の Replication Server を削除します。これを行うには、ID サーバで **sysadmin droprs** コマンドを実行します。

```
sysadmin droprs, replication_server
```

『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**sysadmin droprs**」を参照してください。

7. ID サーバによって管理されるデータベース・リストから、対象の Replication Server によって管理されるすべてのデータベースを削除します。これには、RSSD も含めます。データベースを削除するには、各データベースについて ID サーバで次のように **sysadmin dropdb** コマンドを実行します。

```
sysadmin dropdb, data_server, database
```

『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**sysadmin dropdb**」を参照してください。

参照：

- drop subscription コマンド (437 ページ)
- ルートの管理 (161 ページ)

アクティブではない Replication Server の削除

アクティブではない Replication Server とは、動作していない Replication Server のことです。アクティブでない Replication Server をサービスから除外するには、ルートを削除およびパージし、サービスから Replication Server を安全に削除するためのその他のタスクを実行する必要があります。

1. 対象の Replication Server へのすべてのルートを削除します。

このためには、この Replication Server へのルートを持っている各 Replication Server に対して、**drop route** コマンドに **with nowait** オプションを指定して実行します。次に例を示します。

```
drop route to OLD_RS with nowait
```

このコマンドは、この Replication Server で管理されるデータ用の OLD_RS で作成されたサブスクリプションについての情報も削除します。

2. 削除する Replication Server が、システム・デフォルトの **rs_default_function_class** と **rs_sqlserver_error_class** 以外のいずれかのファンク

ション文字列クラスまたはエラー・クラスに対するプライマリである場合は、新しいプライマリで各クラス用の代わりを作成します。その方法は、次のとおりです。

a) このクラスを使用する他のすべての Replication Server へのルートを持っている Replication Server を選択します。
b) もとのクラスと同じファンクション文字列またはエラー・アクションを含んでいる Replication Server に、新しいクラスを作成します。『Replication Server 管理ガイド 第 2 巻』の「データベース・オペレーションのカスタマイズ」および『Replication Server 管理ガイド 第 2 巻』の「エラーと例外の処理」を参照してください。
c) もとのクラスを使用している各データベース・コネクションを、新しいクラスを使用するように変更します。

3. この Replication Server からのルートを持っている各 Replication Server について、Replication Server のルートをパージします。

ルートをパージするには、削除対象の Replication Server がルートを持っていた各 Replication Server に対して **sysadmin purge_route_at_replicate** コマンドを実行します。次に例を示します。

```
sysadmin purge_route_at_replicate, OLD_RS
```

このコマンドは、次のものも削除します。

- サービスから削除する Replication Server から発信されているデータのサブスクリプション情報。
- サービスから削除する Replication Server で定義されているファンクション文字列クラスとエラー・クラス。削除対象の Replication Server が、**rs_default_function_class**、**rs_sqlserver_function_class**、または **rs_sqlserver_error_class** 用のプライマリ・サイトである場合は、これらのクラスは削除されずに、プライマリ Replication Server のない状態にリセットされます。

4. ID サーバによって管理されているリストから対象の Replication Server を削除します。これを行うには、ID サーバで **sysadmin droprs** コマンドを実行します。

```
sysadmin droprs, replication_server
```

『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**sysadmin droprs**」を参照してください。

5. ID サーバによって管理されるデータベース・リストから、対象の Replication Server によって管理されるすべてのデータベースを削除します。これには、RSSD も含めます。データベースを削除するには、各データベースについて ID サーバで次のように **sysadmin dropdb** コマンドを実行します。

```
sysadmin dropdb, data_server, database
```

『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**sysadmin dropdb**」を参照してください。

これによって、複写システムからのアクティブではない Replication Server の削除が完了します。

次のステップ

次の 3 つの補足事項に留意してください。

- この Replication Server によって以前に管理されていたデータベース内のデータを引き続き複写する場合は、それらのデータベースをどれか別の Replication Server に再度割り当てる必要があります。
- この Replication Server のデータに対するサブスクリプションは、通常のサブスクリプション・マテリアライゼーション解除が行われていないので、レプリケート・データはレプリケート Replication Server からは削除されていません。
- たとえば、この Replication Server が間接ルート上の中間サイトにある場合は、複写システムを管理するための追加ルートを作成する必要があります。

参照：

- データベース・コネクションの管理 (189 ページ)

RepAgent の管理と Adaptive Server のサポート

Adaptive Server の機能に対する Replication Server のサポート、および Adaptive Server の Replication Agent である RepAgent のセットアップ、設定、管理の方法について説明します。

RepAgent は Adaptive Server のスレッドです。データベース・トランザクション・ログをスキャンして、サブスクリプションを作成したデータベースに分配するように Replication Server にトランザクション情報を送信します。

『Replication Server 管理ガイド 第 2 巻』の「パフォーマンス・チューニング」の「Replication Server の内部処理」の「プライマリ Replication Server での処理」で、「Replication Agent ユーザ・スレッド」を参照してください。

参照：

- Replication Server の技術的概要 (27 ページ)

RepAgent の準備

Replication Server と Adaptive Server をシステムにインストールしたあと、Replication Server が管理する各データベース用に RepAgent を有効にする必要があります。

データベースが次のどちらかの条件を満たす場合、Replication Server が管理する各データベース用に RepAgent を有効にする必要があります。

- プライマリ・データを含んでいる場合
- 複写するようマーク付けされたストアド・プロシージャを含んでいる場合

さらに、Replication Server がいずれかのルートの送信元サイトである場合、Replication Server RSSD 用の RepAgent を有効にする必要があります。

RepAgent の準備には、**rs_init** を使用する場合と、コマンド・ライン・オプションを使用する必要がある場合があります。

- 新しい Replication Server をインストールするか新しいデータベースを追加する場合は、**rs_init** を使用して RepAgent を設定します。このプロセスにより、RepAgent が有効になり、デフォルト・パラメータが設定され、RepAgent が起

動します。**rs_init** の詳細については、使用しているプラットフォームの『Replication Server 設定ガイド』を参照してください。
- 既存のレプリケート・データベースをプライマリ・データベースに変更するには、コマンド・ライン・オプションを使用する必要があります。

マルチスレッド RepAgent
デフォルトで、Adaptive Server RepAgent は、プライマリ・データベース・ログをスキャンして、LTL を生成し、生成した LTL を Replication Server に送信する単一のスレッドで構成されます。マルチスレッド RepAgent では、スキャンおよび送信アクティビティは別々のスレッドで実行されます。その後、プライマリ・データベースから追加パスを使用してマルチパス複写をサポートするように、マルチスレッド RepAgent を設定します。

『Replication Server 管理ガイド 第 2 巻』の「パフォーマンス・チューニング」の「Multi-Path Replication」を参照してください。

コマンド・ライン・オプションを使用した RepAgent の設定

コマンド・ラインから RepAgent を設定するための基本的な手順は、次のとおりです。

1. **sp_addserver** を使用して、ローカル Adaptive Server を定義します。
2. **sp_configure** を使用して、Adaptive Server 上で RepAgent 機能を有効にします。
3. **sp_start_rep_agent** を使用して、各データベースに対して RepAgent 機能を有効にします。
4. **alter connection** を使用して、Replication Server 上でのログ転送を有効にします。
5. **sp_start_rep_agent** を使用して、Adaptive Server 上の RepAgent を起動します。

ローカル Adaptive Server の定義

Adaptive Server を初めて起動する場合、Adaptive Server システム・プロシージャ **sp_addserver** を実行して、ローカル・サーバのエントリを Adaptive Server の `sysservers` テーブルに追加する必要があります。

sp_addserver の使用方法の詳細については、『Adaptive Server Enterprise リファレンス・マニュアル』を参照してください。

Adaptive Server での RepAgent の有効化

sp_configure を使用して、Adaptive Server 上で RepAgent 機能を有効にします。

この作業は、各 Adaptive Server 上で 1 度だけ実行する必要があります。

sp_configure 'enable rep agent threads' は、動的オプションです。このため、このオプションはすぐに有効になります。ただし、RepAgent を有効にしたあとで Adaptive Server を再起動しなければならない場合があります。これは、Adaptive

Server によって、RepAgent スレッドに固定数の静的な専用プロセス構造を割り当てるためです。そうしないと、RepAgent はユーザ・コネクション専用のプールからプロセス構造を借用することになります。

Adaptive Server にログインし、**isql** プロンプトで次のコマンドを入力します。

```
sp_configure 'enable rep agent threads', 1
```

プライマリ・データベースでの RepAgent の有効化

sp_config_rep_agent を実行して、各プライマリ・データベースに対する RepAgent を有効にし、RepAgent の設定パラメータにデフォルト値を設定します。

このデフォルト値は、あとで再設定できます。この例では、*dbname* は RepAgent を有効にするデータベースの名前、*repserver_name* は RepAgent が接続する Replication Server、*repserver_username* と *repserver_password* は、Replication Server にログインするために RepAgent が使用する名前とパスワードです。

注意： *repserver_username* が Replication Server で有効なユーザであり、Replication Server の **connect source** パーミッションを持っていることを確認してください。Replication Server でユーザ名とパスワードを確認してから、**sp_config_rep_agent** を使用してください。

Adaptive Server にログインします。**isql** プロンプトで、次のように入力します。

```
use dbname
go
sp_config_rep_agent dbname, enable, 'repserver_name',
        'repserver_username', 'repserver_password'
```

参照：

- RepAgent の設定 (118 ページ)

プライマリ・データベースのログ転送の有効化

set log transfer on を使用して、Replication Server とプライマリ・データベース間の各コネクションに対してログ転送を有効にします。

注意： ログ転送をオンにするには、**rs_init** または **create connection** を使用して、Replication Server とデータ・サーバとの間にデータベース・コネクションを事前に作成しておく必要があります。**rs_init** を使用したコネクションの作成については、使用しているプラットフォームの『Replication Server 設定ガイド』を参照してください。

alter connection を使用して、プライマリ・データベースへのデータベース・コネクションのログ転送をオンにします。たとえば、Replication Server で次のように入力します。

```
alter connection to TOKYO_DS.pubs2
    set log transfer on
```

参照：
- データベース・コネクションの管理 (189 ページ)

RepAgent の設定

sp_config_rep_agent を使用して、デフォルトの RepAgent 設定パラメータを変更します。

デフォルトの RepAgent 設定パラメータは、**rs_init** または **sp_config_rep_agent** を使用して RepAgent を有効にしてから設定します。RepAgent に影響を及ぼす設定パラメータは、RepAgent が有効になっているデータベースの `sysattributes` テーブルに格納されます。新しいパラメータを有効にするには、RepAgent を再起動する必要があります。システムがネットワーク・ベースのセキュリティをサポートしている場合、RepAgent のネットワーク・セキュリティ設定パラメータも使用されます。

RepAgent を設定するには、次の手順に従います。

1. Adaptive Server にログインし、データベースを指定します。
   次に例を示します。
   ```
   use dbname
   go
   ```
2. 設定するパラメータごとに 1 回ずつ、**sp_config_rep_agent** を実行します。
   たとえば、Replication Server にバッチで送信されるログ・レコードの最大件数を 2,000 に変更するには、次の手順に従います。
   ```
   sp_config_rep_agent dbname, 'scan batch size', '2000'
   go
   ```
3. RepAgent を再起動して、新しいパラメータを有効にします。
   ```
   sp_start_rep_agent dbname
   go
   ```

『Replication Server リファレンス・マニュアル』の「Adaptive Server コマンドとシステム・プロシージャ」を参照してください。

参照：
- ネットワークベース・セキュリティの管理 (262 ページ)
- RepAgent の起動 (125 ページ)

RepAgent に影響する設定パラメータ

設定パラメータとともに **sp_config_rep_agent** を使用すると、RepAgent がデータベースに接続する方法および RepAgent のパフォーマンスの調整方法に影響を与えます。

表 7 : RepAgent に影響する設定パラメータ

設定パラメータ	説明
auto start	Adaptive Server が再起動しデータベースをリカバリしたときに RepAgent を自動的に起動するかどうかを指定する。Adaptive Server の再起動時に RepAgent を自動的に起動するには、true に設定する。 デフォルト値は false
batch ltl	true に設定すると、LTL コマンドを Replication Server にバッチで送信する。それ以外の場合は、LTL コマンドを 1 回 に 1 つずつ Replication Server に送信する。 デフォルト値は true
connect database	RepAgent が設定されるデータベースの名前。または、RepAgent がリカバリ・モードで Replication Server に接続するときに使用するテンポラリ・データベースの名前。
connect dataserver	RepAgent データ・サーバの名前。または、RepAgent がリカバリ・モードで Replication Server に接続するときに使用するテンポラリ・データ・サーバの名前。
data limits filter mode	ログ・レコード内に 250 個を超えるカラムや、長さ 255 バイトを超えるカラムまたはパラメータが含まれる場合に、RepAgent がそのログ・レコードをどのように処理してから Replication Server に送信するかを指定する。値は、次のとおり。 • **off** — RepAgent はすべてのレコードを渡す。Replication Server 12.1 以前のバージョンでは、この設定が望ましくない結果をもたらす場合がある。 • **stop** — ログ・レコードに Replication Server 12.1 以前のバージョンの制限を超えるデータが含まれる場合、RepAgent は停止する。 • **skip** — RepAgent は Replication Server 12.1 以前のバージョンの制限を超えるデータを含むログ・レコードをスキップし、エラー・ログにメッセージを通知する。 • **truncate** — RepAgent は、カラムごとに 255 バイトを超えるデータ、テーブルごとに 250 カラムを超えるデータをトランケートする。 デフォルト値は **off** (Replication Server 12.5 以降)、**stop** (Replication Server 12.1 以前)

設定パラメータ	説明
ha failover	Sybase フェールオーバ機能がインストール済みの場合、サーバがフェールオーバしたあとに RepAgent を自動的に起動するかどうかを指定する。 デフォルト値は true
ltl batch size	RepAgent が Replication Server に送信する End-of-Message (EOM) パケットの LTL バッチ・サイズを指定する。RepAgent は、データ・パケットが設定した LTL バッチ・サイズに達すると EOM パケットを送信する。LTL バッチ・サイズに大きな値を設定することによって、EOM がトリガするスレッショルドを高く設定した場合、大きなデータ・パケットに対する小さなデータ・パケットの比率は低くなる。 デフォルト値は true
ltl metadata reduction	true に設定すると Sybase RepAgent テーブルのメタデータ低減が有効になる。RepAgent のテーブル・メタデータの低減を有効にした場合、Replication Server ではエグゼキュータ・コマンド・キャッシュを使用したキャッシュが有効になる。 『Replication Server 管理ガイド 第 2 巻』の「パフォーマンス・チューニング」の「チューニング・パラメータの使用についての注意事項」の「エグゼキュータ・コマンド・キャッシュ」を参照。 デフォルト値は false
max number replication paths	RepAgent が複数のレプリケーション・パスを介してプライマリ・データベースのデータを複写するために使用できるパスの最大数を設定する。RepAgent は、各 RepAgent パスに対して 1 つの RepAgent 送信者スレッドを生成する。 デフォルト値は 1 有効な値の範囲は、1 から MAXINT (2,147,483,647 パス) の間。 **max number replication paths** が 1 より大きい場合で **multipath distribution model** が **connection** に設定されていない場合、RepAgent はパス専用にバインドしないすべての複写オブジェクトのデフォルトのパスを引き続き使用する。レプリケーション・パスの最大数が複写オブジェクトにバインドしたパスの数より少ない場合、RepAgent はエラーを報告し終了する。 『Replication Server 管理ガイド 第 2 巻』の「パフォーマンス・チューニング」の「Multi-Path Replication」で「複数のプライマリ・レプリケーション・パス」の「マルチスレッド RepAgent および RepAgent の複数のパスを有効にする」の「RepAgent 用のレプリケーション・パスの最大数の設定」を参照。

設定パラメータ	説明
multipath distribution model	RepAgent のマルチパス分散モードは次のように設定する。 • **connection** – コネクション別分散。 • **object** – オブジェクト・バインド別分散。 デフォルト値は **object** 分散モデルをオブジェクト・バインド別から分散コネクション別分散に変更すると、RepAgent はすべてのオブジェクトのバインドを無視し、警告を表示する。オブジェクト・バインド別分散に戻して RepAgent を再起動すると、オブジェクトのバインドが保持される。
multithread rep agent	true に設定すると、RepAgent スキャナと送信者アクティビティに対して別々のスレッドを使用するマルチスレッド RepAgent が有効になる。マルチスレッド RepAgent が有効になると、複数のプライマリ・レプリケーション・パスが構築される。 『Replication Server 管理ガイド 第 2 巻』の「パフォーマンス・チューニング」の「Multi-Path Replication」で「複数のプライマリ・レプリケーション・パス」の「マルチスレッド RepAgent および RepAgent の複数のパスを有効にする」の「マルチスレッド RepAgent の有効化」を参照。 デフォルト値は false
net password encryption	Adaptive Server 15.0.2 では、このパラメータが true に設定されている場合、RepAgent は、CS_SEC_ENCRYPTION コネクション・プロパティと CS_SEC_EXTENDED_ENCRYPTION コネクション・プロパティの両方を設定する。それ以外の場合、これらのプロパティは設定されない。 デフォルト値は true **注意：** 統一化ログインまたは相互認証のセキュリティ・プロパティが設定されている場合、net password encryption パラメータは無視されます。これは、セキュリティ・プロパティが認証にクレデンシャルを使用しているためです。

設定パラメータ	説明
number of send buffers	マルチスレッド RepAgent のスキャナと送信者タスクが使用できる送信バッファの最大数を設定する。 複数のプライマリ・レプリケーション・パスを作成するには、**multithread rep agent** RepAgent パラメータを使用し、マルチスレッド RepAgent を有効にする。 デフォルト値は 50 バッファ 範囲：1 から MAXINT (2,147,483,647 バッファ) の間。 変更内容を有効にするには、RepAgent を再起動する必要がある。 『Replication Server 管理ガイド 第 2 巻』の「パフォーマンス・チューニング」の「Multi-Path Replication」で「複数のプライマリ・レプリケーション・パス」の「マルチスレッド RepAgent および RepAgent の複数のパスを有効にする」の「送信バッファの数の設定」を参照。
priority	個々の RepAgent に相対的な優先値を設定する。推奨値は **4**、**5**、および **6** で、**6** は低優先度、**5** は中程度の優先度、および **4** は高優先度をそれぞれ示す。 デフォルト値は **5**
rs name	RepAgent が接続してトランザクション・ログからトランザクションを転送する先の Replication Server の名前。Replication Server の名前を変更した場合は、**rs name** を使用する。
rs password	RepAgent が Replication Server へのログインに使用するパスワード。RepAgent パスワードを変更する場合は、**rs password** を使用する。
rs username	RepAgent が Replication Server へのログインに使用するユーザ名。RepAgent ユーザ名を変更する場合は、**rs_username** を使用する。
retry timeout	リカバリ可能なエラーのあと、または Replication Server がダウンしたときに、Replication Server に再接続しようとするまでに RepAgent が非アクティブである秒数。 デフォルト値は 60 秒
scan batch size	バッチごとに Replication Server に送信するログ・レコードの最大件数。この件数のレコードが送信されると、RepAgent は Replication Server に新しいセカンダリ・トランケーション・ポイントを要求する。 デフォルト値は 1000 レコード

設定パラメータ	説明
scan timeout	Replication Server にバッチを送信してから Replication Server に新しいセカンダリ・トランケーション・ポイントを問い合わせるまでに、RepAgent が非アクティブである時間。ログにレコードがさらにある場合、RepAgent はスキャンを再開する。ログに処理するレコードがなくなり、Replication Server がまだセカンダリ・トランケーション・ポイントを送信することによって受信を通知していない場合、RepAgent はこのパラメータで設定されている時間だけタイムアウトを再度延長する。 デフォルト値は 15 秒
schema cache growth factor	テーブルまたはストアド・プロシージャ・スキーマを RepAgent スキーマ・キャッシュに置くことができる有効期間の長さを制御する。値が大きいほど、必要なメモリ量が増加する。範囲は 1 ～ 10。 デフォルト値は 1
send buffer_size	RepAgent が Replication Server との通信に使用する送信バッファのサイズ (キロバイト) を制御する。値は、2K、4K、8K、および 16K。 デフォルト値は 2K **注意：** 送信バッファのサイズは、データベース・ページのサイズとは関係がありません。
send maint xacts to replicate	true に設定すると、RepAgent は、メンテナンス・ユーザによって生成されたレコードを、サブスクリプションを作成するサイトに分配するため Replication Server に送信する。それ以外の場合、RepAgent はメンテナンス・ユーザからのレコードを Replication Server に送信しない。 デフォルト値は false
send structured oqids	RepAgent がオリジン・キュー ID (OQID) を構造化トークンとバイナリ文字列のどちらとして送信するかを指定する。true に設定すると、RepAgent は OQID を構造化トークンとして送信するため、LTL で必要な領域が減少し、スループットが向上する。 デフォルト値は false
send warm standby xacts	通常、スキーマ・トランザクションとシステム・トランザクションは、ウォーム・スタンバイ・データベースには送信されない。true に設定すると、RepAgent は、スキーマ・トランザクション、システム・トランザクション、メンテナンス・ユーザ・トランザクションを送信する。それ以外の場合、RepAgent は、トランザクションをウォーム・スタンバイ・データベースに送信しない。 デフォルト値は false

設定パラメータ	説明
short ltl keywords	RepAgent が Replication Server に送信する LTL を、省略形にするかどうかを指定する。true に設定すると、RepAgent は LTL の省略形を使用するため、必要な領域と、Replication Server に送信されるデータ量が減少する。 デフォルト値は true
skip ltl errors	true に設定すると、RepAgent は Replication Server が返す LTL エラーを無視する。このオプションは、通常、リカバリ中はオンになる。 デフォルト値は false
skip unsupported features	Replication Server でサポートしていない機能のログ・レコードをスキップするよう RepAgent に指示する。このオプションは、通常、Replication Server のバージョンが Adaptive Server のバージョンより古い場合に使用する。 デフォルト値は false
startup delay	Adaptive Server の起動時のどの段階で、特定の RepAgent を起動するかを制御する。RepAgent の起動を指定した時間だけ遅らせて、RepAgent が Replication Server に接続しようとする前に、Replication Server が稼働できるようにする。デフォルトでは、RepAgent は自動起動時に遅延なしで起動する。値を秒単位で設定すると、RepAgent の起動が、指定された秒数だけ遅れる。 デフォルト値は 0 秒です。

マスタ・キーと rs パスワード

レプリケーションを続行するには、マスタ・キー・パスワードと **rs password** 属性を設定します。

Adaptive Server では、マスタ・キーとともに syb_extpasswdkey サービス・キーを作成しても、マスタ・キー・パスワードをメモリに手動でも自動的にも設定していない場合、Adaptive Server RepAgent が起動時にブロックされ、マスタ・キー・パスワードを設定するまでは **sp_who** によって "MASTER KEY SLEEP" と表示されます。レプリケーション・パスごとに、RepAgent が Replication Server へのログインに使う **rs password** 属性が 1 つ存在します。syb_extpasswdkey サービス・キーが削除されると、Adaptive Server は既存の RepAgent **rs password** 属性をすべてリセットします。「**sp_encryption helpextpasswd**」と入力すると、"必要とするリセット" と表示されます。レプリケーションを続行するには、**rs password** 属性をすべてリセットします。

Adaptive Server Enterprise の『暗号化カラム・ユーザーズ・ガイド』の「外部パスワードと隠しテキストのセキュリティ保護」の「サービス・キー」を参照してください。

RepAgent の起動

sp_start_rep_agent を使用して、Adaptive Server 上の RepAgent を起動します。

前提条件

Adaptive Server の interfaces ファイルに、Replication Server のエントリがある必要があります。

手順

通常、RepAgent スレッドを起動する必要があるのは、次の場合のみです。

- RepAgent パラメータを再設定した場合
- RepAgent を明示的に停止した場合

Adaptive Server の再起動時に RepAgent が自動的に起動するのは、次の場合です。

- RepAgent を **sp_start_rep_agent** で最低でも一度起動し、**sp_stop_rep_agent** で停止しなかった場合
- **sp_config_rep_agent** を使用して **auto start** を true に設定した場合

注意： RepAgent は、関連付けられたデータベースが完全にリカバリされてオンラインになっており、ログ転送がプライマリ・データベースへのコネクションでオンになっている場合のみ再起動できます。

sp_start_rep_agent および **sp_config_rep_agent** の各オプションの詳細については、『Replication Server リファレンス・マニュアル』の「Adaptive Server コマンドとシステム・プロシージャ」を参照してください。

1. Adaptive Server にログインします。
2. **isql** プロンプトで、**sp_start_rep_agent** と入力します。
   たとえば、`pubs2` データベースの RepAgent を有効にするには、次のように入力します。

   ```
   sp_start_rep_agent pubs2
   ```

   (オプション) Adaptive Server が再起動するたびに RepAgent を自動的に起動するには、次のように入力します。

   ```
   sp_config_rep_agent 'auto start', 'true'
   ```

RepAgent の停止

RepAgent を停止するには、Adaptive Server にログインして、**sp_stop_rep_agent** を実行します。

RepAgent は、再起動すると、最も古いトランザクションからレコードのスキャンを開始しますが、最後に処理されたトランザクションよりあとのレコードしか送信しません。その結果、Replication Server は重複したレコードを受信しません。

RepAgent が **sp_stop_rep_agent** で停止すると、**auto start** が **sp_config_rep_agent** を使用して事前に true に設定されていない限り、データ・サーバの起動中にデータベースがオンラインになっても RepAgent は自動的に起動されません。それ以外の場合には、**sp_start_rep_agent** を実行して、RepAgent を起動し自動起動を再開する必要があります。

たとえば、RepAgent を停止するには、次のように入力します。

```
sp_stop_rep_agent pubs2
```

この方法で RepAgent を停止すると、Adaptive Server は、現在のトランザクションのバッチの終了時に、RepAgent を適切に停止します。

nowait オプションを指定すると、RepAgent をただちに停止できます。次に例を示します。

```
sp_stop_rep_agent pubs2, nowait
```

nowait オプションを指定して RepAgent を停止すると、Adaptive Server は現在実行中のオペレーションが終了するのを待たずに RepAgent を終了します。

RepAgent の無効化

RepAgent を無効にするには、**sp_config_rep_agent dbname** を使用します。

注意： RepAgent は、レプリケート・データベースをプライマリ・データベースに変更する場合、または Replication Server を以前のバージョンにダウングレードする場合にかぎって無効にしてください。

まず **sp_stop_rep_agent** を使用して RepAgent を停止してから、**sp_config_rep_agent** を使用して RepAgent を無効にします。

通常は、RepAgent を無効にすると、その処理によってセカンダリ・トランケーション・ポイントも無効になります。次に例を示します。

```
sp_config_rep_agent pubs2, 'disable'
```

いったんセカンダリ・トランケーション・ポイントが無効になると、このポイント以降のログがトランケートされる場合があります。

セカンダリ・トランケーション・ポイントを維持しながら RepAgent を無効にするには、**preserve secondary truncpt** オプションを使用します。

```
sp_config_rep_agent pubs2, 'disable', 'preserve
        secondary truncpt'
```

この方法は、RepAgent を無効にして、RepAgent を一時的に無効にする場合に使用します。

プライマリ・データベースをレプリケート・データベースに変更している場合、ログ転送もオフにします。RepAgent を無効にした後、**alter connection** を使用してログ転送をオフにします。

たとえば、Replication Server にログインして、次のように入力します。

```
alter connection to TOKYO_DS.pubs2
    set log transfer off
```

RepAgent ネットワーク・セキュリティの設定

ネットワークベースのセキュリティ機能を使用すると、RepAgent と Replication Server との間の安全な経路を確立できます。

sp_config_rep_agent を使用して、次の設定を変更できます。

- アクティブなセキュリティ・メカニズム
- 統一化ログイン
- 相互認証
- メッセージの機密保持
- メッセージの整合性
- メッセージ・リプレイの検出
- メッセージ・オリジンのチェック
- メッセージ順序不整合のチェック

参照：
- ネットワークベース・セキュリティの管理 (262 ページ)

ログ転送アクティビティの管理

リカバリ、トラブルシューティング、診断作業の実行をする場合に、ログ転送コマンドを使用するとログ転送をサスペンドおよび再開できます。

ログ転送コマンドは次のとおりです。

- **resume log transfer** と **suspend log transfer**
- **alter connection ... set log transfer on/off**

注意： ログ転送が **alter connection** を使用してあらかじめオンにされていないと、RepAgent は Replication Server に接続できません。

RepAgent スレッドがデータベースとトランザクション・ダンプをリプレイできるように、RepAgent スレッドをリカバリ・モードで起動する方法の詳細については、『Replication Server 管理ガイド 第 2 巻』の「複写システム・リカバリ」を参照してください。

ログ転送のサスペンド

1 つまたはすべての RepAgent を切断して、Replication Server に接続できないようにするには、**suspend log transfer** コマンドを実行します。

Replication Server へのログ転送は、**resume log transfer** コマンドを使用してレジュームするまでサスペンド状態のままです。

suspend log transfer コマンドは RSSD に情報を記録するので、Replication Server を停止して再起動すると、その Replication Server に対するログ転送はサスペンド状態のままになります。

注意： ログ転送をサスペンドするのは、複写システムをクワイス状態にするときの最初の手順です。

ログ転送をサスペンドするには、Replication Server にログインし、**isql** プロンプトで次のように入力して **suspend log transfer** を実行します。

```
suspend log transfer from {data_server.database | all}
```

構文の説明は次のとおりです。

- *data_server* – ログ転送をサスペンドするデータベースが存在するデータ・サーバです。
- *database* — ログ転送をサスペンドするデータベースです。
- **all** – すべての RepAgent からのログ転送をサスペンドして、すべての RepAgent に対する以後のコネクションを許可しないように Replication Server に指示します。

次に、**suspend log transfer** コマンドの使用例を示します。

- 次のコマンドは、TOKYO_DS データ・サーバによって管理されている `pubs2` というデータベースに対するログ転送をサスペンドします。
  ```
  suspend log transfer from TOKYO_DS.pubs2
  ```
- すべての RepAgents から現在の Replication Server へのログ転送をサスペンドするには、次のように入力します。

```
suspend log transfer from all
```

どちらの例でも、コマンドの実行後に、対象の RepAgent が停止されないで Replication Server にいくつかのメッセージを送信し続けることがあります。RepAgent をすぐに停止させるには、Adaptive Server にログインし、RepAgent が有効になっているデータベースの名前と **nowait** オプションを指定して、**sp_stop_rep_agent** を実行します。

参照：

- Replication Server のクワイス (109 ページ)

ログ転送の再開

Replication Server に RepAgent を再接続するには、Replication Server にログインして、**resume log transfer** コマンドを **isql** プロンプトに入力します。

```
resume log transfer from {data_server.database | all}
```

構文の説明は次のとおりです。

- *data_server* — ログ転送をレジュームするデータベースが存在するデータ・サーバです。
- *database* — ログ転送をレジュームして RepAgent のコネクションを許可するデータベースです。
- **all** — この Replication Server への接続をすべての RepAgent に許可します。

次に、**resume log transfer** コマンドの使用例を示します。

- 次のコマンドは、TOKYO_DS データ・サーバによって管理されている pubs2 というデータベースに対するログ転送をレジュームします。
  ```
  resume log transfer from TOKYO_DS.pubs2
  ```
- すべての RepAgents からこの Replication Server へのログ転送をレジュームするには、次のように入力します。
  ```
  resume log transfer from all
  ```

alter connection と set log transfer オプションの使用

ログ転送の停止には、**alter connection** で **set log transfer** オプションを使用します。

ログ転送を停止するには、**set log transfer** オプションをオフにします。次に例を示します。

```
alter connection to TOKYO_DS.pubs2
   set log transfer off
```

ログ転送をオフにすると、Replication Server によって DIST スレッドが削除され、RepAgent は Replication Server にログインできなくなります。

Replication Server がプライマリ・データベースを認識しなくなっている場合には、**rs_init** または **create connection** を使用してこのコネクションを再確立してから、**alter connection** を使用してログ転送をオンにする必要があります。

ログ転送をオンにするには、**set log transfer** オプションをオンにします。次に例を示します。

```
alter connection to TOKYO_DS.pubs2
set log transfer on
```

RepAgent ステータスと設定情報の確認

RepAgent をモニタして、コマンドおよびシステム・プロシージャを使用して設定パラメータの値を確認できます。Adaptive Server プラグインを Sybase Central で使用して、RepAgent をモニタすることもできます。

RepAgent 情報の表示

Adaptive Server で **sp_help_rep_agent** を使用して、RepAgent をモニタします。

sp_help_rep_agent には以下の情報が表示されます。

- Recovery ― データベースのリストア中のステータスとその他の情報
- Configuration parameters – 単一および複数のレプリケーション・パスの RepAgent 設定パラメータの現在の設定
- Process – ステータス、スリープ・ステータス、成功しなかったコネクションに対するリトライ回数 (リトライが行われた場合)、最後のエラー・メッセージの番号など、単一および複数のレプリケーション・パスの RepAgent プロセスについての情報
- Send buffers - RepAgent に割り当てた送信バッファの数
- Scanned transactions ― ログ・トランザクションの現在のバッチについての情報 (起動、終了、現在のマーカ、バッチ内のレコード件数、最も古いトランザクションについての情報)
- Security ―ネットワークベース・セキュリティ・メカニズムの現在の設定内容
- All ― 上記の情報すべて

Adaptive Server にログインして、**sp_help_rep_agent** を **isql** プロンプトで実行します。

```
sp_help_rep_agent [dbname[, 'recovery' | 'config' | 'process' |
'send' | 'scan' | 'security' | 'all']]
```

dbname は、RepAgent が有効化されているデータベースの名前です。

1 つまたはすべてのオプションについて現在のステータス情報を表示できます。次に例を示します。

- RepAgent プロセスについての情報を表示するには、Adaptive Server にログインして次のように入力します。
  ```
  sp_help_rep_agent pubs2, 'process'
  ```
- RepAgent のログ・スキャンについての情報を表示するには、次のように入力します。
  ```
  sp_help_rep_agent pubs2, 'scan'
  ```

詳細な構文、使用方法の情報、**sp_help_rep_agent** の出力の例については、『Replication Server リファレンス・マニュアル』の「Adaptive Server コマンドとシステム・プロシージャ」を参照してください。

RepAgent 設定パラメータ値の表示

特定の RepAgent の各設定パラメータのデフォルト値、現在値、ランタイム値のリストを表示するには、Adaptive Server にログインして、**sp_config_rep_agent** をオプションなしで実行します。

次に例を示します。

```
sp_config_rep_agent pubs2
```

データベース名を指定しない場合、**sp_config_rep_agent** は、RepAgent が有効になっているすべてのデータベースの設定値を表示します。

特定のパラメータの値を表示するには、パラメータ名を指定します。次に例を示します。

```
sp_config_rep_agent pubs2, 'scan batch size'
```

『Replication Server リファレンス・マニュアル』の「Adaptive Server コマンドとシステム・プロシージャ」の「**sp_config_rep_agent**」を参照してください。

RepAgent スレッド情報の表示

Adaptive Server での RepAgent スレッドのステータスを表示するには、**sp_who** を実行します。

画面出力には、“cmd” カラムの “REP AGENT” ローに RepAgent の情報が表示されます。

たとえば、**sp_who** では、RepAgent の次のローが表示されます。

```
fid spid status loginame orignam hostname blk_spid dbname  cmd
block_xloid
------------------------------------------------------------------
----------
...
0   23 background NULL    NULL      0        pubs2     REP AGENT   0
...
```

sp_who の構文と使用法の詳細については、『Adaptive Server Enterprise リファレンス・マニュアル』を参照してください。

Replication Server での RepAgent スレッドのユーザ・ステータスを表示するには、**admin who** を実行します。Replication Server で、“name” カラムの “REP AGENT” ローに RepAgent スレッドのユーザ情報が表示されます。

admin who コマンドの詳細とさまざまな出力例については、『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」を参照してください。

RepAgent 情報メッセージとエラー・メッセージのログ・ファイルの確認

Adaptive Server のエラー・ログ・ファイルに記録されている RepAgent のエラー・メッセージと情報メッセージを取得します。

Adaptive Server のエラー・ログの詳細については、『Adaptive Server Enterprise システム管理ガイド』を参照してください。

たとえば、RepAgent を起動すると、次のメッセージが Adaptive Server のエラー・ログに生成されます。

```
00:00000:00022:2003/09/18 12:16:39.15 server Started
RepAgent on database, ‘pubs2’ (dbid = 4).
```

RepAgent を停止すると、次のメッセージが生成されます。

```
00:00000:00022:2003/09/18 12:17:17.07 server Shutting
down RepAgent for database, ‘pubs2’ (dbid=4).
```

RepAgent のパフォーマンスをモニタするためのカウンタの使用

Adaptive Server には、RepAgent のパフォーマンスをモニタするためのカウンタがいくつか用意されています。**sp_sysmon** を使用すると、RepAgent のパフォーマンス・データをモニタできます。

sp_sysmon を開始すると、サンプル間隔の間に使用される一連のカウンタに累積されていたデータはすべてクリアされます。サンプル間隔の最後に、カウンタ値が読み込まれ、レポートが出力されて、実行が停止されます。

sp_sysmon で指定すると、出力する情報を RepAgent のカウンタだけ、または Adaptive Server のすべてのカウンタのいずれかにすることができます。**sp_sysmon** は、各データベースに対する RepAgent カウンタ情報を表示します。

sp_sysmon の使用法と構文の詳細については、『Adaptive Server Enterprise 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。

カウンタを使用して Replication Server のアクティビティをモニタするには、『Replication Server 管理ガイド 第 2 巻』の「カウンタを使ったパフォーマンスのモニタリング」を参照してください。

sp_sysmon の開始

sp_sysmon を開始するには、一定の時間間隔を使用するか、**begin_sample** パラメータと **end_sample** パラメータを使用します。

一定の時間間隔

実行時間を分単位で指定してサンプルを得るには、一定の時間間隔を使用します。

たとえば、**sp_sysmon** を 10 分間実行して、すべてのカウンタの情報を出力するには、次のコマンドを実行します。

```
sp_sysmon "00:10:00"
```

レポートから RepAgent のセクションだけ出力するには、次のように入力します。

```
sp_sysmon "00:10:00", repagent
```

begin_sample と end_sample

begin_sample と **end_sample** を使用すると、**sp_sysmon** を呼び出して、サンプリングの開始と停止、クエリの発行、任意の時点での結果の出力を実行できます。

たとえば、RepAgent のカウンタ・グループのサンプリングを開始して停止するには、次のように入力します。

```
sp_sysmon begin_sample
go
execute proc1
go
sp_sysmon end_sample,repagent
```

RepAgent アクティビティの sp_sysmon からのサンプル出力

sp_sysmon から RepAgent カウンタのアクティビティのサンプル出力を表示します。

sp_sysmon のサンプル出力

```
Replication Agent
-----------------
Replication Agent: pubs2
Replication Server: NY_RS
```

```
                        per sec     per xact    count    % of total
                        -------     --------    -----    ----------
```

```
Log Scan Summary
 Number of Log Scans       n/a        n/a           1        n/a
 Amount of Time
   for Log Scan (ms)       n/a        n/a        3822        n/a
   Longest Time
   for Log Scan (ms)       n/a        n/a        3822        n/a
   Average Time
   per Log Scan (ms)       n/a        n/a        3822        n/a

Log Scan Activity
   Updates                 n/a        n/a           5        n/a
   Inserts                 n/a        n/a           5        n/a
   Deletes                 n/a        n/a           5        n/a
   Store Procedures        n/a        n/a           0        n/a
   DDL Log Records         n/a        n/a           0        n/a
   Writetext Log Records   n/a        n/a           0        n/a
   Text/Image Log Records  n/a        n/a          10        n/a
   CLRs                    n/a        n/a           0        n/a
   Checkpoints Processed   n/a        n/a           0        n/a

Transaction Activity
    Opened                 n/a        n/a           7        n/a
    Commited               n/a        n/a           7        n/a
    Aborted                n/a        n/a           0        n/a
    Delayed Commit         n/a        n/a           0        n/a
    Maintenance User       n/a        n/a           0        n/a

Log Extension Wait
    Count                  n/a        n/a           3        n/a
    Amount of time (ms)    n/a        n/a        7822        n/a
    Longest Wait (ms)      n/a        n/a        5110        n/a
    Average Time (ms)      n/a        n/a      2607.3        n/a

Schema Cache

  Usage
    Max Ever Used          n/a        n/a           0        n/a
    Schemas reused         n/a        n/a           0        n/a

  Forward Schema Lookups
    Count                  n/a        n/a           0        n/a
    Total Wait (ms)        n/a        n/a           0        n/a
    Longest Wait (ms)      n/a        n/a           0        n/a
    Average Time (ms)      n/a        n/a         0.0        n/a

  Backward Schema Lookups
    Count                  n/a        n/a           0        n/a
    Total Wait (ms)        n/a        n/a           0        n/a
    Longest Wait (ms)      n/a        n/a           0        n/a
    Average Time (ms)      n/a        n/a         0.0        n/a

Truncation Point Movement
    Moved                  n/a        n/a           0        n/a
    Gotten from RS         n/a        n/a           0        n/a
```

```
Connections to Replication Server
    Success                 n/a         n/a            0         n/a
    Failed                  n/a         n/a            0         n/a

Network Packet Information
    Packets Sent            n/a         n/a            6         n/a
    Full Packets Sent       n/a         n/a            2         n/a
    Largest Packet          n/a         n/a         2048         n/a
    Amount of Bytes Sent    n/a         n/a         7695         n/a
    Average Packet          n/a         n/a       1282.5         n/a

I/O Wait from RS
    Count                   n/a         n/a            6         n/a
    Amount of Time (ms)     n/a         n/a          766         n/a
    Longest Wait (ms)       n/a         n/a          206         n/a
    Average Wait (ms)       n/a         n/a        127.7         n/a


-------------------------------------------------------------------------
```

RepAgent カウンタのアクティビティのサンプル出力の説明

sp_sysmon の RepAgent カウンタのアクティビティのサンプル出力にある各セクションでは、異なるアクティビティが示されています。

Log Scan Summary

RepAgent は、トランザクション・ログのレコードをすべてスキャンしますが、スキャンしたレコードをすべて処理して Replication Server に送信することは必要ではありません。たとえば、RepAgent は、複写するようマーク付けされていないテーブルに対してデータ操作言語 (DML: Data Manipulation Language) が生成したレコードは送信しません。

Log Scan Summary には、次のログ・レコード件数がレポートされます。

- RepAgent がスキャンしたレコード数
- RepAgent が処理し Replication Server に送信したレコード数

Log Scan Activity

Log Scan Activity には、RepAgent によって処理され Replication Server に送信される、さまざまな種類のログ・レコードの情報があります。

Log Scan Activity には、次の件数がレポートされます。

- **update** 文の影響を受けたロー
- **insert** 文の影響を受けたロー
- **delete** 文の影響を受けたロー
- ストアド・プロシージャの実行
- 複写予定の DDL を含んでいるログ・レコード

- **WriteText** コマンドによって生成された処理済みのログ・レコード
- `text`、`unitext`、または `image` データのあるテーブル用に処理された、DML ログ・レコード
- トランザクションの一部または全部がロールバックされたときに生成される、CLR (Compensation Log Record)
- このログ・レコードが書き込まれた時点でアクティブなトランザクションがあったことを示すチェックポイント・ログ・レコード

Transaction Activity

Transaction Activity には、次のトランザクションの件数がレポートされます。

- プライマリ・データベースで開かれたトランザクション
- コミットされたトランザクション
- アボートされたトランザクション
- 準備ステータスのままのトランザクション
- メンテナンス・ユーザによって開かれたトランザクション

Log Extension Wait

通常の処理中に、RepAgent はトランザクション・ログの末尾に達します。末尾に達すると、プライマリ・データベースでさらにアクティビティがレジュームされるまで待機します。

Log Extension Wait には、次のものがレポートされます。

- RepAgent がトランザクション・ログの拡張を待っていた回数
- RepAgent がログの拡張を待っていたミリ秒 (ms) 単位の合計時間
- RepAgent がログの拡張を待っていた最長時間 (ms)
- RepAgent がログの拡張を待っていた平均時間 (ms)

Schema Cache

複写するようマーク付けされたオブジェクトの構造が、**alter table** などによって変更された場合、Adaptive Server はトランザクション・ログに特殊なレコードを記録する必要があります。このレコードはあとで、RepAgent がそのオブジェクトの正しいスキーマを特定するのに役立ちます。

Schema Cache には、スキーマ・アクティビティ、およびトランザクション・ログ内を前方および後方にスキャンしてオブジェクト・スキーマの変更点を検索する RepAgent のアクティビティがレポートされます。

- 使用法

  Usage には、次のものがレポートされます。

  - Replication Agent が最後に再起動された時点以降のスキーマ・キャッシュ内のアクティブなスキーマの最大数

    - 新しいスキーマ用のスペースを空けるために、スキーマ・キャッシュからスキーマを削除する必要がある回数
- Forward Schema Lookups

  Forward Schema Lookups には、次のものがレポートされます。
    - RepAgent が実行した前方スキャンの回数
    - RepAgent が前方スキャンの実行に費やした合計時間 (ms)
    - RepAgent が前方スキャンの実行に費やした最長時間 (ms)
    - RepAgent が前方スキャンの実行に費やした平均時間 (ms)
- Backward Schema Lookups

  RepAgent は、トランザクション内で DDL が実行されると、後方スキャンを実行します。Backward Schema Lookups には、次のものがレポートされます。
    - RepAgent が実行した後方スキャンの回数
    - RepAgent が後方スキャンの実行に費やした合計時間 (ms)
    - RepAgent が後方スキャンの実行に費やした最長時間 (ms)
    - RepAgent が後方スキャンの実行に費やした平均時間 (ms)

Truncation Point Movement

Truncation Point Movement には、次のものがレポートされます。

- RepAgent がセカンダリ・トランケーション・ポイントを移動させた回数
- RepAgent が Replication Server に新しいトランケーション・ポイントを要求した回数

Connections to Replication Server

Connections to Replication Server には、次のものがレポートされます。

- Replication Server に正常に接続できたコネクション数
- Replication Server に正常に接続できなかったコネクション数

Network Packet Information

Network Packet Information には、次のものがレポートされます。

- Replication Server に送信されたパケット数
- Replication Server に送信された全パケット数
- Replication Server に送信された最長パケット
- Replication Server に送信されたバイト数
- 平均パケット・サイズ

I/O wait from Replication Server

RepAgent は、LTL を生成後、Replication Server に送信します。これは Open Client 機能を使用して実行されます。

I/O Wait from Replication Server には、次のものがレポートされます。

- RepAgent がバッチを Replication Server に送信した回数
- RepAgent が Replication Server から受け取った結果を処理するのに費やした合計時間 (ms)
- RepAgent が Replication Server から受け取った結果を処理するのに費やした最長経過時間 (ms)
- RepAgent が Replication Server から受け取った結果を処理するのに費やした平均経過時間 (ms)

拡張された制限値のサポート

Replication Server バージョン 12.5 以降では、複写定義に対して拡張された制限値がサポートされています。

拡張された制限値のサポートは、次のとおりです。

- カラム数の増加 (最大 1024)
- カラムとパラメータの拡張 (最大 32768 バイト)
- データ・ローの拡張 (データ・サーバのデータ・ページの幅まで)
- 16K より長いワイド・メッセージ

Replication Server のサイト・バージョンが 12.5 以降である場合、LTL バージョンは、Replication Server によって自動的に 400 以上に設定されます。RepAgent が Adaptive Server 12.5 以降で動作している場合、Replication Server が connect source で LTL バージョン 400 以降を指定したときにかぎり、RepAgent は拡張された制限値でデータを送信します。

Replication Server のサイト・バージョンが 12.1 以前である場合、LTL バージョンは 400 より前になります。RepAgent が Adaptive Server 12.5 以降で動作している場合、12.1 以前のバージョンの Replication Server に対しては、制限値が拡張されたデータを送信しないことをおすすめします。**data limits filter mode** パラメータを **sp_config_rep_agent** と使用することによって、RepAgent による拡張された制限値のデータの処理方法を指定できます。

参照：

- RepAgent の設定 (118 ページ)

長い識別子のサポート

Replication Server バージョン 15.0 以降では、複写オブジェクトの識別子の最大長が拡大され、255 バイトになりました。

影響を受ける複写オブジェクト識別子は、次のとおりです。

- テーブル名とカラム名
- ストアド・プロシージャ名とパラメータ名
- ファンクションとパラメータ – ファンクション複写定義および内部使用のみ
- ファンクション文字列名
- 複写定義 – テーブル複写定義、ファンクション複写定義、データベース複写定義を含む。
- アーティクル名
- パブリケーション名

Replication Server のサイト・バージョンが 15.0 の場合、LTL バージョンは自動的に 700 に設定されます。RepAgent が Adaptive Server 15.0 以降で動作している場合、Replication Server が connect source で LTL バージョン 700 以降を指定したときにかぎり、RepAgent は拡張されたサイズでデータを送信します。

Replication Server のサイト・バージョンが 12.6 以前である場合、LTL バージョンは 700 より前になります。RepAgent が Adaptive Server 15.0 以降で動作している場合、12.6 以前のバージョンの Replication Server に対しては、長い識別子を持つデータを送信しないことをおすすめします。

data limit filter mode パラメータを **config_rep_agent** と使用することによって、RepAgent による長い識別子を持つデータの処理方法を指定できます。

注意： **create function**、**alter function**、**drop function** の各コマンドは、長い識別子をサポートしていません。ファンクション名およびこれらのコマンドのパラメータは、最大で 30 バイトです。

参照：

- RepAgent の設定 (118 ページ)

bigdatetime および bigtime データ型のサポート

Replication Server では、Adaptive Server に含まれている bigdatetime データ型および bigtime データ型の複写をサポートしています。

これらのデータ型をレプリケート・データベースとウォーム・スタンバイ・データベースに複写するには、そのデータ型を複写定義、ファンクション複写定義、サブスクリプション内で指定します。

bigdatetime と bigtime を使用すると、Adaptive Server はデータおよび時間データをマイクロ秒の精度で格納できます。bigdatetime は TIMESTAMP データ型に対応し、bigtime は Sybase IQ の TIME データ型に対応します。

データ型の説明については、『Replication Server リファレンス・マニュアル』の「トピック」の「データ型」で、「日時および日付と時刻のデータ型」を参照してください。

rs_helprep を使用すると、bigdatetime および bigtime に関する情報を表示できます。

rs_subcmp は、bigdatetime および bigtime をサポートしています。

コマンドの説明については、『Replication Server リファレンス・マニュアル』を参照してください。

以下の例では、複写定義、ファンクション複写定義、サブスクリプションで bigdatetime と bigtime を使用する方法を示します。

この例では、以下の単語は次のことを示します。

- PDS – プライマリ・データ・サーバ
- pdb1 – プライマリ・データベース
- RDS – レプリケート・データ・サーバ
- rdb1 – レプリケート・データベース
- tb1 – テーブル
- col1、col2、col3 – カラム
- **rep1** – 複写定義
- **func1** – ファンクション複写定義
- **sub1** – サブスクリプション

複写定義

```
create replication definition rep1
with primary at PDS.pdb1
with all tables named tb1
```

```
(col1 int, col2 bigdatetime, col3 bigtime)
primary key (col1)
```

ファンクション複写定義

```
create function replication definition func1
with primary at PDS.pdb1
(@par1 int, @par2 bigdatetime, @par3 bigtime)
searchable parameters (@par1)
```

サブスクリプション

```
create subscription sub1 for rep1
with replicate at RDS.rdb1
where col3 = '14:20:00.010101'
without materialization
```

bigdatetime および bigtime データ型のシステム・テーブル・サポート

rs_columns、rs_datatypes、および rs_objects の各システム・テーブルは、bigdatetime データ型および bigtime データ型の複写をサポートするように拡張されています。

rs_columns

カラム	データ型	説明
coltype	tinyint	カラムまたはパラメータのデータ型。 • 35 – bigdatetime • 36 – bigtime

rs_datatype

カラム	データ型	説明
base_coltype	tinyint	データ型の基本データ型の ID。値は次のとおり。 • 35 – bigdatetime • 36 – bigtime

rs_objects

カラム	データ型	説明
attributes	int	マスク。次のうち 1 つ以上。 • 0x02 – 複写定義に bigdatetime または bigtime カラムが含まれ、Replication Server 15.5 以降にのみ送信できる。

bigdatetime と bigtime の混合バージョン情報

bigdatetime と bigtime は Adaptive Server バージョン 15.5 以降でのみサポートされています。

少なくともプライマリ・データ・サーバが Adaptive Server 15.5 以降であれば、次のように対処できます。

- プライマリおよびレプリケート Replication Server がバージョン 15.5 以降で、レプリケート Adaptive Server がこれらのデータ型をサポートしていない場合は、その 2 つのデータ型をそれぞれ varchar データ型にマッピングする定義を複写定義に含めます。または、複写定義でその 2 つのデータ型の代わりに varchar データ型を使用します。
- プライマリ Replication Server がバージョン 15.5 以降であり、レプリケート Replication Server と Adaptive Server がこれらのデータ型をサポートしない場合、複写定義でその 2 つのデータ型の代わりに varchar データ型を使用します。
- プライマリ Replication Server、レプリケート Replication Server、レプリケート Adaptive Server がこれらのデータ型をサポートしていない場合は、RepAgent が自動的に varchar データ型を Replication Server に送信します。

Adaptive Server 共有ディスク・クラスタのサポート

Replication Server と RepAgent スレッドは、どちらも Adaptive Server 共有ディスク・クラスタ環境をサポートしています。

Sybase 共有ディスク・クラスタでは、複写の送信元または複写の送信先としてデータベースを使用できます。RepAgent の設定や複写対象テーブルへのマーク付けなど、すべてのタスクをクラスタ内の任意のインスタンスから実行できます。複写ステータスは、クラスタ全体で一貫しています。

Adaptive Server クラスタ環境から新しいコネクションを追加したり、Adaptive Server クラスタ環境に新しいコネクションを追加したりする場合、コネクション

構文の *servername* を *instancename* ではなく *clustername* にする必要があります。select @@servername を使用すると、*clustername* を取得できます。

デフォルトでは、RepAgent はクラスタ・コーディネータで起動しますが、クラスタ内の任意のインスタンスで起動するように設定できます。たとえば、プライマリ・データベース `pdb` の RepAgent を、常に “ase2” インスタンスで起動するように設定するには、次のように入力します。

```
sp_config_rep_agent pdb, "cluster instance name", "ase2"
```

新しい設定を有効にするには、**sp_start_rep_agent** を使用して RepAgent を再起動する必要があります。RepAgent がクラスタ・コーディネータで起動するデフォルトの動作に戻すには、次のように入力します。

```
sp_config_rep_agent pdb, "cluster instance name",
"coordinator"
```

インスタンスは、起動すると、そのノードで起動するように設定された RepAgent があるかどうかを確認します。そのように設定された RepAgent がある場合、データベースが自動的に起動するようマーク付けされているときは、RepAgent が起動します。

クラスタ・コーディネータが起動すると、特定のインスタンスで起動するように設定されていないすべての RepAgent も起動させます。コーディネータ・ノードで障害が発生するか、適切に停止された場合、RepAgent は新しいコーディネータ・ノードで起動します。

RepAgent が、コーディネータ・ノード以外のインスタンスで起動するように設定されており、このインスタンスで障害が発生した場合、RepAgent はコーディネータで起動します。

cluster instance name 設定パラメータについては、『Replication Server リファレンス・マニュアル』の「Adaptive Server コマンドとシステム・プロシージャ」の「**sp_config_rep_agent**」を参照してください。

Adaptive Server データ圧縮

Replication Server では、Adaptive Server データ圧縮機能がサポートされています。

Adaptive Server のデータ圧縮機能を使用すると、同じ容量のデータをより小さい記憶領域に格納して、キャッシュ・メモリの消費量を削減し、I/O 要求の緩和によってパフォーマンスを向上させることができます。Adaptive Server は、`text`、`image`、`unitext` などのラージ・オブジェクト (LOB) データ型と LOB 以外のデータ型を圧縮できます。『Adaptive ServerR Enterprise 圧縮ユーザーズ・ガイド』を参照してください。

Adaptive Server では、データをロー内またはロー外に格納します。ローのデータカラムのうち、そのローの他のすべてのカラムと連続した場所にあるカラムは、ロー内に格納されます。LOB データは、データ・サイズが大きいためロー外の他の場所に格納されます。ロー外データの実際の場所を示すポインタがロー内にあります。

システムの稼働条件

- Adaptive Server – プライマリ・データベースとレプリケート・データベースの両方について、バージョン 15.7 ESD #1 以降。
- Replication Server – プライマリおよびレプリケート Replication Server について、バージョン 15.7.1 以降。

Replication Server における圧縮データのサポート

Replication Server は、圧縮解除を行わず、プライマリ Adaptive Server データベースの圧縮された LOB カラムを圧縮されたままの形式でテキスト値の圧縮解除を行わずにレプリケートします。

Adaptive Server データベース間における圧縮データの複写サポートは、データがロー内とロー外のいずれであるかと、データ型が LOB であるかどうかによって異なります。

- Replication Server は、Adaptive Server データベース間における LOB 以外のロー内圧縮データの複写をサポートしています。RepAgent は、プライマリ・データベースからのローを圧縮解除してから Replication Server に送信します。
- Replication Server が Adaptive Server データベース間におけるロー外 LOB 圧縮カラムの複写をサポートするのは、プライマリ・データベースとレプリケート・データベースの LOB スキーマ、文字セット、エンディアン、バージョン、およびページ・サイズが同一の場合のみです。
- 圧縮 LOB データを正しく複写するには、レプリケート Replication Server の文字セットとエンディアン・タイプがプライマリおよびレプリケートの Adaptive Server データ・サーバと同じであることが必要です。
- Replication Server は、ロー内 LOB 圧縮データの複写をサポートしません。

圧縮 LOB カラムは、複写定義で `image` データ型として指定する必要があります。次に例を示します。

```
create replication definition pubs_copy_rep
with primary at TOKYO_DS.pubs2
with primary table named 'publishers'
with replicate table named joe.'pubs_copy'
(pub_id, pub_name as pub_name_set, au_photo image)
primary key (pub_id)
```

既存のテーブル内の LOB カラムに対して LOB 圧縮をオンに設定している場合は、**sp_configure "enable compression"** が 0 に設定されているときに High-Volume Adaptive Replication (HVAR) を有効にしないでください。

テーブルのラージ・オブジェクト圧縮を変更して、以下を行った場合、更新は正しく複写されません。

1. 最初に Adaptive Server でカラムを圧縮カラムとしてマーク付けする。
2. **sp_setrepcol** を **always_replicate** パラメータと共に使用して、テーブルの複写定義で圧縮 LOB カラムの複写を有効にする。
3. 挿入を実行する。
4. Adaptive Server でカラムを非圧縮カラムとしてマーク付けする。

更新が確実に正しく複写されるようにするには、**sp_setrepcol** を **replicate_if_changed** パラメータと共に使用して、テーブルの複写定義で LOB カラムの複写を有効にします。

ステーブル・キュー内のデータ破損を防ぐため、プライマリおよびレプリケート Adaptive Server データベースの LOB スキーマ、文字セット、エンディアン、アーキテクチャ、バージョン、およびページ・サイズが異なる場合、または、ルート内のすべての Replication Server のサイト・バージョンが 15.7.1 より前のバージョンである場合は、**sp_setrepcol** を使用して、ロー外の圧縮 LOB カラムの複写を無効にしてください。

```
sp_setrepcol table_name, lob_column_name, 'do_not_replicate'
```

これを行わない場合は、破損データをキューから削除する必要があります。

破損データの削除

ロー外の圧縮 LOB カラムの複写を無効にしていない場合にキューから破損データを削除するには、カスタム・ファンクション文字列を使用します。

1. 圧縮 LOB カラムの複写を無効にします。

   ```
   sp_setrepcol table_name, lob_column_name, 'do_not_replicate'
   ```

2. レプリケート Replication Server で High-Volume Adaptive Replication (HVAR) を無効にします。
   たとえば、データベース・コネクションの HVAR を無効にするには、次のように入力します。

   ```
   alter connection to data_server.database
   set dsi_compile_enable to 'off'
   go
   ```

詳細については、『Replication Server リファレンス・マニュアル』の「**alter connection**」と『Replication Server 管理ガイド：第 2 巻』の「Enable HVAR」を参照してください。

3. DSI スレッドがシャットダウンする場合に、キューから破損データを削除するには、カスタム・ファンクション文字列を使用します。
   たとえば、`mytab5` テーブルのテーブル・スキーマと **reptab5** という名前の複写定義を使用して、**reptab5.rs_insert** という名前のファンクション文字列を **rs_insert** ファンクションに基づいて作成します。
   - テーブル・スキーマ：
     ```
     create table mytab5
     (c1 int primary key, c2 text null compressed = 5)
     ```
   - 複写定義：
     ```
     create replication definition reptab5
     with primary at repl3_18540.pdb
     with all tables named mytab5
     (c1 int, c2 image null)
     primary key (c1)
     ```
   - カスタム・ファンクション文字列：
     ```
     alter function string reptab5.rs_insert
     for rs_sqlserver_function_class
     output language
     'insert mytab5(c1, c2) values(?c1!new? , '''')'
     ```

遅延名前解決

遅延名前解決は、ストアド・プロシージャを作成するときに、それによって内部で使用されるオブジェクトを解決しないままでストアド・プロシージャを作成できるようにするための Adaptive Server の機能です。オブジェクト解決のフェーズはそのストアド・プロシージャが Adaptive Server で初めて実行されるときまで延期されます。

ストアド・プロシージャは初回以降の実行では通常に実行されます。

15.5 より古いバージョンの Replication Server では、サポートされているデータ定義言語 (DDL) コマンドをスタンバイ・データベースに複写できるように、ウォーム・スタンバイ・アプリケーションをセットアップして、アクティブ・データベースで **sp_reptostandby** を有効にできます。『Replication Server 管理ガイド 第 2 巻』の「ウォーム・スタンバイ・アプリケーションの管理」を参照してください。

ただし、ウォーム・スタンバイではない環境にあるスタンバイ・データベースまたはレプリケート・データベースでは、Replication Server がテンポラリ・テーブルを複写しないので、テンポラリ・テーブルを参照するストアド・プロシージャを作成できません。ストアド・プロシージャを作成するプロセスは、そのストアド・プロシージャが内部で使用するオブジェクトを解決する必要があります。た

だし、レプリケート・データベースにもスタンバイ・データベースにもテンポラリ・テーブルがないので、Replication Server はレプリケート・データベースにもスタンバイ・データベースにもストアド・プロシージャを作成しません。

Replication Server では遅延名前解決がサポートされているので、テンポラリ・テーブル (存在しないテーブルと存在しないプロシージャ) を参照するストアド・プロシージャをレプリケート・データベースにもスタンバイ・データベースにも複写できます。

『Adaptive Server Enterprise Transact-SQL ユーザーズ・ガイド』、『Adaptive Server Enterprise システム管理ガイド：第 1 巻』、および『Adaptive Server Enterprise リファレンス・マニュアル：コマンド』を参照してください。

遅延名前解決をサポートするための Replication Server の設定

Replication Server で遅延名前解決を有効または無効にするには、ウォーム・スタンバイ・アプリケーション内で **deferred_name_resolution** パラメータを、物理コネクションの場合は **alter connection** で設定し、論理コネクションの場合は **alter logical connection** で設定します。

たとえば、SYDNEY_DS データ・サーバの *pubs2* ウォーム・スタンバイ・データベース上で遅延名前解決を有効するには、次のように入力します。

```
alter logical connection to SYDNEY_DS.pubs2
set deferred_name_resolution to 'on'
```

alter connection または **alter logical connection** を使用して **deferred_name_resolution** を実行した後、コネクションをサスペンドして再開してください。デフォルトでは、**deferred_name_resolution** はオフです。

注意： Replication Server で遅延名前解決のサポートを設定する前に、Adaptive Server で遅延名前解決のサポートを有効にします。

増分データ転送のサポート

Adaptive Server 15.5 では、Adaptive Server から別の Adaptive Server にテーブル全体を転送する代わりに、テーブルのデータを小刻みに転送できます。Replication Server では、Adaptive Server の増分データ転送機能に関連するデータ定義言語がサポートされています。

増分転送としてマーク付けされたレプリケート・テーブル上で実行されるデータ変更操作に対して正常に複写が実行されます。

transfer table コマンドを使ってレプリケート・テーブルにデータをロードするときにテーブルにユニーク・インデックス・コマンドがあり、増分転送のデータがそ

のテーブルに既に存在する場合、Adaptive Server は内部で **insert** コマンドを **update** コマンドに変換します。

transfer table コマンドは初回に転送を開始したデータ・サーバとデータベースにのみ適用されます。

ウォーム・スタンバイまたは MSA 環境内のアクティブ・データベースでテーブルを増分転送用にマークしてから、アクティブ・データベースが終了してスタンバイ・データベースに切り替わると、増分データ転送がスタンバイ・データベースで正しくレジュームされない可能性があります。これは、アクティブ・データベースとは異なり、スタンバイ・データベースには増分データ転送アクティビティの記録がないことが原因です。したがって、スタンバイ・データベースでも増分データ転送を初期化する必要があります。

『Adaptive Server Enterprise Transact-SQL ユーザーズ・ガイド』の「データの追加、変更、転送、削除」の「増分データ転送」を参照してください。

Adaptive Server レプリケーション機能のパフォーマンスの強化

Replication Server には、Adaptive Server レプリケーション機能のパフォーマンスを強化するための機能がいくつか用意されています。

- SQL 文の複写 – Replication Server では、ログベースの複写を補完し、バッチ・ジョブによるパフォーマンスの低下に対処するための Adaptive Server 内での SQL 文の複写がサポートされています。SQL 文の複写は、次の場合に使用することをおすすめします。
  - データ操作言語 (DML: Data Manipulation Language) 文が複写されたテーブル上の多数のローに影響を及ぼす場合
  - 基本のアプリケーションを変更してストアド・プロシージャの複写を有効にすることが困難な場合
- High Volume Adaptive Replication – Replication Server には、Adaptive Server への複写のパフォーマンスを強化するための High Volume Adaptive Replication (HVAR) が含まれています。
  HVAR は、大量のトランザクションを 1 つのグループにまとめ、それによってバルク・オペレーション処理が向上します。したがって、複写のスループットとパフォーマンスも向上します。
  HVAR は、プライマリ・データベースと同じスキーマを持つレプリケート・データベースのあるシステムのアーカイブとレポートを行うオンライン・トランザクション処理 (OLTP: creating online transaction processing) の作成に特に役立ちます。

『Replication Server 管理ガイド 第 2 巻』の「パフォーマンス・チューニング」の「SQL 文の複写」、および『Replication Server 管理ガイド 第 2 巻』の「パフォーマ

ンス・チューニング」の「Advanced Services Option」で「High Volume Adaptive Replication」を参照してください。

インメモリ・データベースおよびリラックス持続性データベース

Replication Server では、インメモリ・データベースおよびリラックス持続性データベースの Adaptive Server 機能がサポートされています。

インメモリ・データベース (IMDB: In-memory database) はすべてがキャッシュ内に存在し、データにもログにもディスク・ストレージは使用されません。したがって、ディスク I/O の必要もありません。この結果、従来のディスク常駐型データベース (DRDB: disk-resident database) に比べてより良いパフォーマンスを期待できるとともに、その他の利点も享受できます。インメモリ・データベースはキャッシュ内にのみ存在するので、それをサポートするホストが停止したりデータベース障害が発生したりすると、データベースのリカバリができません。

リラックス持続性によって、Adaptive Server はインメモリ・データベースから得られるパフォーマンスのメリットをディスク常駐型データベースに拡張します。ディスク常駐型データベースでは、ディスクへの書き込みを実行することで、ACID プロパティとも呼ばれる原子性、一貫性、整合性、持続性のトランザクション・プロパティの維持を確実に行います。従来のディスク常駐型データベースは、サーバの障害からトランザクションを確実にリカバリできるよう、完全な持続性で動作します。リラックス持続性データベースは、コミットされたトランザクションの完全な持続性と引き換えに、トランザクションの負荷に対する実行時のパフォーマンスを向上させます。**no_recovery** レベルで作成されたリラックス持続性データベースは、サーバが終了したり停止されたりすると、データもログもリカバリできないという点で、インメモリ・データベースに似ています。リラックス持続性データベースは **at_shutdown** レベルで作成することもできます。この場合、データベースが適切に停止されると、トランザクションがディスクに書き込まれます。

詳細については、『Adaptive Server Enterprise インメモリ・データベース・ユーザーズ・ガイド』を参照してください。

Replication Server のサポート

Replication Server は、プライマリおよびレプリケート・データベースとして、次をサポートしています。

- インメモリ・データベース
- 持続性が **no_recovery** に設定されているリラックス持続性データベース

便宜上、このドキュメントでは、持続性を **non_recovery** に設定したリラックス持続性データベースを「リラックス持続性データベース」と呼んでいます。

データ、オブジェクト・スキーマ、設定情報を次のいずれかから取得して、インメモリ・データベースおよびリラックス持続性データベースを新しいプライマリ・データベースまたはレプリケート・データベースとして初期化できます。

- 基本情報の入ったテンプレート・データベース。
- 別のデータベースからのデータベース・ダンプ。ダンプをターゲットのインメモリ・データベースまたはリラックス持続性データベースにロードします。

ダンプのソース・データベースには、別のインメモリ・データベース、リラックス持続性データベース、従来のディスク常駐型データベースのいずれも使用できます。

注意： インメモリ・データベースからの複写またはインメモリ・データベースへの複写は、リラックス持続性データベースからの複写またはリラックス持続性データベースへの複写に比べて高速ではない可能性があります。インメモリ・データベースの DML は、さまざまな要因によって変化します。

さらに、インメモリ・データベースやリラックス持続性データベースへの複写時またはこれらのデータベースからの複写時のパフォーマンスは、持続性が full の従来のディスク常駐型データベースであるプライマリ・データベースおよびレプリケート・データベースのパフォーマンスと比較した場合、違いはありません。

複写用プライマリ・データベースとしてのインメモリ・データベース

インメモリ・データベースとリラックス持続性データベースは、複写システムのプライマリ・データベースとして設定できます。

テンプレート・データベースを使用したインメモリ・プライマリ・データベース用の複写の設定

同じディスク常駐型データベースをテンプレートとして使用して、複数のインメモリ・データベースまたはリラックス持続性データベースをプライマリ・データベースとして初期化できます。インメモリ・プライマリ・データベースまたはリラックス持続性プライマリ・データベースは、テンプレート・データベースの設定を継承します。

注意： Adaptive Server を停止または終了してから再起動すると、Adaptive Server は、テンプレートからインメモリ・データベースまたはリラックス持続性データベースを自動的に再作成します。

1. テンプレート・データベースを作成します。テンプレート・データベースは、Replication Server からのインバウンド・コネクションを使用したデータベースの名前を使用します。通常の場合、これは、プライマリ・データベースの名前です。
   たとえば、NY_DS データ・サーバに `ny_db` という名前のテンプレート・データベースを作成するには、次のようにします。

```
create database ny_db on publicdev=10 log on publicdevlog=10
go
```

2. **rs_init** を使用して、プライマリ・データベースとレプリケート・データベースを複写システムに追加します。
3. 次のコマンドを使用して、テンプレート・データベース上の RepAgent を停止します。

```
sp_stop_rep_agent ny_db
go
```

4. Replication Server からデータベースへのコネクションをサスペンドします。
5. 次のように、テンプレート・データベースの名前を `template1` に変更します。

```
use master
go
sp_dboption ny_db, single, true
go
sp_renamedb ny_db, template1
go
sp_dboption template1, single, false
go
```

6. 手順 1 で作成したテンプレートを使用して、持続性を **no_recovery** に設定したインメモリ・データベースまたはリラックス持続性データベースを作成します。

```
create inmemory database ny_db
use template1 as template
on imdb_cache_dev = '50' log on imdb_cache_dev_log='50'
with DURABILITY=NO_RECOVERY
go
```

7. 次のように、Adaptive Server の停止および再起動後に RepAgent が自動的に起動するように設定します。

```
use template1
go
sp_config_rep_agent template1, 'auto start', true
```

8. Replication Server からデータベースへのコネクションをレジュームします。

データベース・ダンプを使用したインメモリ・プライマリ・データベース用の複写の設定

別のデータベースからのダンプを使用する場合、インメモリ・データベースまたはリラックス持続性データベースをプライマリ・データベースとして初期化するように複写を設定できます。インメモリ・プライマリ・データベースまたはリラックス持続性プライマリ・データベースは、ダンプを取得したデータベースの設定を継承します。

1. インメモリ・データベースまたはリラックス持続性データベースを次のように作成します。

```
create inmemory database ny_db
on imdb_cache_dev2 = '50' log on imdb_cache_dev_log2='50'
with DURABILITY=NO_RECOVERY
go
```

2. **rs_init** を使用して、インメモリ・プライマリ・データベースを複写システムに追加します。
3. 作成した新しいインメモリ・データベースをロードするときに使用するデータベースからダンプを取得します。
4. データベース・ダンプからインメモリ・データベースをロードします。
   次に例を示します。

```
use master
go
sp_dboption ny_db, single, true
go
load database ny_db from '/remote/Based_on_loaddb/IMDB.dump'
go
online database ny_db
go
sp_dboption ny_db, single, false
go
```

5. データベース上で RepAgent を起動します。

```
sp_start_rep_agent ny_db
go
```

6. 必要に応じて、Replication Server からプライマリ・データベースへの DSI コネクションをレジュームします。

インメモリ・データベースの停止または終了

Adaptive Server を停止または終了してから再起動すると、Adaptive Server は、テンプレートからインメモリ・データベースまたはリラックス持続性データベースを自動的に再作成します。

作成したインメモリ・データベースまたはリラックス持続性データベースへのコネクションをレジュームすると、Replication Server がコマンドを再適用する場合があります。これは、Adaptive Server の再起動時に、最後に適用されたコマンドを検出するために Replication Server が使用する情報が失われるためです。

複写用レプリケート・データベースとしてのインメモリ・データベース

インメモリ・データベースとリラックス持続性データベースは、複写システムのレプリケート・データベースとして設定できます。

テンプレート・データベースを使用したインメモリ・レプリケート・データベース用の複写の設定

同じディスク常駐型データベースをテンプレートとして使用して、複数のインメモリ・データベースまたはリラックス持続性データベースを初期化できます。

注意： Adaptive Server を停止または終了してから再起動すると、Adaptive Server は、テンプレートからインメモリ・データベースまたはリラックス持続性データベースを自動的に再作成します。作成したインメモリ・レプリケート・データベースまたはリラックス持続性レプリケート・データベースへのコネクションをレジュームすると、Replication Server がコマンドを再適用する場合があります。これは、Adaptive Server の再起動時に、最後に適用されたコマンドを検出するために Replication Server が使用する情報が失われるためです。

1. テンプレート・データベースを作成します。テンプレート・データベースは、Replication Server へのアウトバウンド・コネクションを使用したデータベースの名前を使用します。通常、これはレプリケート・データベースの名前です。

   ```
   create database tokyo_db on publicdev=10 log on publicdevlog=10
   go
   ```

2. **rs_init** を使用して、複写システムにレプリケート・データベースを追加します。
3. テンプレート・データベースで RepAgent を停止することによって、テンプレート・データベースへの DSI スレッドをサスペンドします。次に例を示します。

   ```
   suspend connection to TOKYO_DS.tokyo_db
   ```

4. 次のように、テンプレート・データベースの名前を **template1** に変更します。

   ```
   use master
   go
   sp_dboption tokyo_db, single, true
   go
   sp_renamedb tokyo_db, template1
   go
   sp_dboption template1, single, false
   go
   ```

5. 手順 1 で作成したテンプレートを使用して、持続性を **no_recovery** に設定したインメモリ・データベースまたはリラックス持続性データベースを作成します。

   ```
   create inmemory database tokyo_db
   use template1 as template
   on imdb_cache_dev = '50' log on imdb_cache_dev_log='50'
   with DURABILITY=NO_RECOVERY
   go
   ```

6. Replication Server に接続し、レプリケート・データベースへのコネクションをレジュームします。

```
resume connection to TOKYO_DS.tokyo_db
```

データベース・ダンプを使用したインメモリ・レプリケート・データベース用の複写の設定

別のデータベースからのダンプを使用する場合、インメモリ・データベースまたはリラックス持続性データベースをレプリケート・データベースとして初期化するように複写を設定できます。インメモリ・レプリケート・データベースまたはリラックス持続性レプリケート・データベースは、ダンプを取得したデータベースの設定を継承します。

1. インメモリ・データベースまたはリラックス持続性データベースを次のように作成します。

   ```
   create inmemory database tokyo_db
   on imdb_cache_dev2 = '50' log on imdb_cache_dev_log2='50'
   with DURABILITY=NO_RECOVERY
   go
   ```

2. テーブルとストアド・プロシージャ、ユーザ、レプリケート・データを受信するために必要となるパーミッションなどのオブジェクトを作成するか、データベース・ダンプをロードします。
3. **rs_init** を使用して、インメモリ・データベースまたはリラックス持続性データベースへの Replication Server のコネクションを作成します。
4. ダンプを次のように実行して、インメモリ・データベースまたはリラックス持続性データベースの現在の状態を保存します。
   a) インメモリ・データベースまたはリラックス持続性データベースへのコネクションを次のようにサスペンドします。

      ```
      suspend connection to RDS.imdb1
      go
      ```

   b) インメモリ・データベースまたはリラックス持続性データベースのデータベース・ダンプを次のように取得します。

      ```
      dump database imdb1 to '/databases/dump/tokyo_db.dump'
      go
      ```

   c) インメモリ・データベースまたはリラックス持続性データベースへのコネクションを次のようにレジュームします。

      ```
      resume connection to RDS.imdb1
      go
      ```

インメモリ・データベースおよびリラックス持続性データベースのリストア

ホスト・データ・サーバがシャット・ダウンまたは再起動すると、インメモリ・データベースおよびリラックス持続性データベースはキャッシュ内にのみ存在するため、オブジェクト定義、データ、RepAgent 設定は失われます。インメモリ・データベースまたはリラックス持続性データベースは、ホスト・データ・サーバの再起動後、再初期化またはリストアする必要があります。

次の方法のいずれかを使用して、テンプレートでインメモリ・データベースまたはリラックス持続性データベースを再初期化するか、データベース・ダンプでインメモリ・データベースまたはリラックス持続性データベースをリストアします。

- ダンプからの再作成
- データベース再同期化
- バルク・マテリアライゼーション

データベースの dump と load を実行するためのディスク領域と時間が十分あることを確認し、Replication Server がトランザクションをスキップする期間が許容範囲内にあることを確認します。許容できる時間の長さは、アウトバウンド・キュー内のセグメントを **admin who, sqm** でモニタすることによって見積もることができます。『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**admin who**」を参照してください。

Adaptive Server 停止後のインメモリ・データベースおよびリラックス持続性データベースのダンプからの再作成

インメモリ・データベースおよびリラックス持続性データベースは Adaptive Server が再起動されたときに再作成されるため、ダンプからのデータはリストアできます。

1. 再作成されたインメモリ・レプリケート・データベースまたはリラックス持続性レプリケート・データベースを、新しいダンプまたはレプリケート・データベースから取得されたアーカイブ済みの任意のダンプで再移植します。

   注意： ダンプがソースのダンプからロードされたものでない場合、レプリケート・テーブルに欠落ローが発生します。

   たとえば、ホストの Adaptive Server が再起動するときに `tokyo_db` データベースを元の `tokyo_db.source` ダンプでロードするには、次のように入力します。

   ```
   use master
   go
   sp_dboption tokyo_db, single, true
   go
   load database tokyo_db from '/databases/dump/tokyo_db.dump'
   go
   online database tokyo_db
   go
   sp_dboption tokyo_db, single, false
   go
   ```

2. 再作成したインメモリ・レプリケート・データベースまたはリラックス持続性レプリケート・データベースへのコネクションを再作成します。

プライマリ・データベースから直接再同期する

インメモリ・データベースをプライマリ・データベースから再同期します。

1. RepAgent による複写プロセスを停止します。Adaptive Server で次のコマンドを実行します。
   ```
   sp_stop_rep_agent database
   ```
2. レプリケート・データベースとの Replication Server DSI コネクションをサスペンドします。
   ```
   suspend connection to dataserver.database
   ```
3. レプリケート・データベースのアウトバウンド・キューからデータを削除し、プライマリ・データベースの RepAgent からの再同期マーカを待機するように Replication Server に指示します。
   ```
   resume connection to data_server.database skip to
   resync marker
   ```
4. RepAgent に再同期モードで起動するよう指示し、再同期マーカを Replication Server に送信します。
   - トランケーション・ポイントが元の位置から移動していない場合は、Adaptive Server で次のコマンドを実行します。
     ```
     sp_start_rep_agent database, 'resync'
     ```
   - トランケーション・ポイントが元の位置から移動している場合は、Adaptive Server で次のコマンドを実行します。
     ```
     sp_start_rep_agent database, 'resync purge'
     ```
5. Replication Server システム・ログで次のメッセージを検索して、DSI が RepAgent から再同期マーカを受信して受け入れていることを確認します。
   ```
   DSI for data_server.database received and processed
   Resync Database Marker. Waiting for Dump Marker.
   ```

注意： 複数のデータベースを再同期する場合は、再同期する各データベースの DSI コネクションが再同期マーカを受け入れていることを確認します。

6. プライマリ・データベース・コンテンツのダンプを取得します。詳細については、『Adaptive Server Enterprise リファレンス・マニュアル：コマンド』の「コマンド」で「**dump database**」を参照してください。Adaptive Server は自動的にダンプ・データベース・マーカを生成します。
7. Replication Server システム・ログで次のメッセージを検索して、Replication Server がダンプ・データベース・マーカを処理していることを確認します。
   ```
   DSI for data_server.database received and processed
   Dump Marker. DSI is now suspended. Resume after database has been
   reloaded.
   ```

Replication Server がダンプ・マーカを受け取ると、DSI コネクションが自動的にサスペンドされます。

8. プライマリ・データベースのダンプをレプリケート・データベースに適用します。詳細については、『Adaptive Server Enterprise リファレンス・マニュアル：コマンド』の「コマンド」で「**load database**」を参照してください。
9. レプリケート・データベースにダンプを適用したら、次のコマンドを使用して DSI をレジュームします。

```
resume connection to data_server.database
```

バルク・マテリアライゼーションを使用してインメモリ・レプリケート・データベースまたはリラックス持続性データベースを再同期する

2 つのバルク・マテリアライゼーション・メソッドのどちらかを使用して、インメモリ・データベースまたはリラックス持続性データベースをリストアできます。

前提条件

バルク・マテリアライゼーションを開始する前に、複写定義とサブスクリプションが存在することを確認します。

手順

1. インバウンド・キューとアウトバウンド・キューをすばやく空にするには、インメモリ・データベースまたはリラックス持続性データベースがレプリケート・データベースになっているサブスクリプションのアクティブ化を解除します。

```
deactivate subscription subscription_name
for {table_repdef_name | func_repdef_name |{publication pub_name |
database replication definition db_repdef_name}
with primary at dataserver.database}
with replicate at dataserver.database
go
```

   サブスクリプションのアクティブ化を解除した後は、Replication Server はインバウンド・キュー内のすべてのトランザクションをインメモリ・データベースまたはリラックス持続性データベースのアウトバウンド・キューに送信しません。

   それに対して、サブスクリプションを削除すると、インバウンド・キューに書き込まれていたコミットされたすべてのトランザクションが Replication Server のダウンストリームに分配されます。アクティブ化解除はプライマリ・サイトでのみ行われるので、サブスクリプションのアクティブ化解除は DSI が実行されていなくても行うことができます。アクティブ化解除のマーカがアウトバウンド・キューに到達すると、Replication Server ログにそのエントリが表示されます。

```
The deactivate marker for subscription subscription_name
arrives at outbound queue: data_server_name.database_name.
```

アクティブ化解除のマーカがアウトバウンド・キューに到達した後、**sysadmin sqm_purge_queue** を使用してレプリケート・サイトのアウトバウンド・キューをパージし、アウトバウンド・キューを空にします。『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**sysadmin sqm_purge_queue**」を参照してください。

2. プライマリ Replication Server とレプリケート Replication Server の両方で **check subscription** を実行して、サブスクリプションのステータスがプライマリ Replication Server では DEFINED に、レプリケート Replication Server では VALID になっていることを確認します。
3. インメモリ・データベースまたはリラックス持続性データベースを作成する場合は、『Replication Server 管理ガイド 第 1 巻』の「サブスクリプションの管理」の「サブスクリプション・マテリアライゼーション・メソッド」の「バルク・マテリアライゼーション」に記載されている "アトミック・マテリアライゼーションのシミュレート" または "ノンアトミック・マテリアライゼーションのシミュレート" バルク・マテリアライゼーション・メソッドを使用します。次のいずれかの手順を実行します。
   - アトミック・マテリアライゼーションをシミュレートする ― 手順 4 ～ 9 を実行する
   - 非アトミック・マテリアライゼーションをシミュレートする ― 手順 4 ～ 13 を実行する

オートコレクションの有効化

インメモリ・レプリケート・データベースまたはリラックス持続性レプリケート・データベースへのコネクションをレジュームする前に、サブスクリプションに使用した複写定義のオートコレクションを有効にして、更新や挿入を削除および挿入のペアに変換する必要があります。

オートコレクションの有効化は、テンプレートまたはダンプを使用して作成されたレプリケート・データベースに適用されます。オートコレクションを使用すると、インメモリ・レプリケート・データベースまたはリラックス持続性レプリケート・データベースをホストしている Adaptive Server が停止または終了した場合でも、Replication Server は Replication Server のキューにあるメッセージの複写を継続できます。

最低限の DML ロギングと複写

Replication Server は、レプリケート・データベースとして動作するインメモリ・データベースまたはリラックス持続性データベースへの最低限の DML ロギングをサポートしています。

Adaptive Server では、ディスクのトランザクション・ログに書き込まれるログ・レコードを最適化するために、**insert**、**update**、**delete**、slow **bcp** などの一部のデータ操作言語 (DML) のコマンドをすべての種類の低持続性データベース (持続性が **at_shutdown** または **no_recovery** に設定されたインメモリ・データベースやリラックス持続性データベースなど) で実行する場合、最低限のロギングを行うか、ロギングを実行しないで済ますことができます。DML の最低限のロギングは、データベース単位、テーブル単位、セッション単位で実行できます。『Adaptive Server Enterprise インメモリ・データベース・ユーザーズ・ガイド』の「最低限のログを取る DML」を参照してください。

注意： 最低限の DML ロギングのセッションレベルの設定は、データベース・レベルの設定およびテーブル・レベルの設定よりも優先されます。

複写では完全なロギングを使用するため、Adaptive Server 15.5 の最低限のデータ操作言語 (DML) ロギングを行う機能はデータベース・レベルまたはテーブル・レベルなど、同じレベルでの互換性はありません。ただし、最低限の DML ロギングと複写は異なるレベルに共存できるため、あるテーブルで最低限の DML ロギングを実行する一方、他のテーブルで複写を実行することによりパフォーマンスを向上させることができます。

たとえば、テーブル・レベルなど、同じレベルで複写と最低限の DML ロギングを設定する場合は、複写ステータスの設定に失敗し、次のシナリオで説明されているエラー・メッセージが表示されます。

- 最低限の DML ロギングを使用するようにデータベースを作成した場合
  - **sp_reptostandby** を使用してデータベースを複写するようマーク付けすると、マーク付けに失敗し次のエラーが表示されます。

    ```
    Cannot set replication for database database_name
    as it is minimally logged. Use ALTER DATABASE to
    set full DML logging and try again.
    ```

  - テーブルのサブセットを複写するためにテーブルとストアド・プロシージャを複写するようマーク付けし、最低限の DML ロギング機能を使用してデータベース内のテーブルにマーク付けすると、次の警告が表示されます。

    ```
    Warning: database_name is using minimal logging.
    Replicated objects will continue to use full DML
    logging.
    ```

- データベースで完全なロギングを使用している場合に、**sp_reptostandby** を使用してそのデータベースを複写するようマーク付けし、データベースに最低限の

DML ロギングを設定するためにデータベースを変更しようとすると、変更に失敗し次のメッセージが表示されます。

```
Cannot alter database database_name to set minimal
logging because this database is marked for
replication. Remove replication and try again.
```

- 完全なロギングを使用しているデータベースに複写するようマーク付けされているオブジェクトがある場合は、データベース・レベルで最低限の DML ロギングを設定できます。ただし、複写するようマーク付けされたオブジェクトが完全なロギングを使用することを警告する次のメッセージが表示されます。

  ```
  Warning: Database database_name has objects marked
  for replication. Replicated objects will continue to
  use full logging.
  ```

- 完全な DML ロギングを使用しているデータベースに最低限のロギングを使用するように定義されたテーブルがある場合に、そのデータベースを複写するようマーク付けすると、次の警告が表示されます。

  ```
  Warning: Database database_name has tables that use
  minimal DML logging. These tables will not be
  replicated.
  ```

- 完全なロギングを使用するテーブルを作成して複写するようマーク付けし、そのテーブルに対して最低限の DML ロギングを設定すると、設定に失敗し次のメッセージが表示されます。

  ```
  Cannot alter the table table_name to set minimal DML
  logging because this table is marked for
  replication. Remove replication and try again.
  ```

- 最低限の DML ロギングを使用するテーブルを作成し、そのテーブルを複写するようマーク付けすると、マーク付けに失敗し次のエラーが表示されます。

  ```
  Cannot set replication for table table_name because
  it is using minimal logging. Use ALTER TABLE to set
  full logging and try again.
  ```

ルートの管理

ルートとは、送信元 Replication Server から送信先 Replication Server への一方通行のメッセージ・ストリームです。

送信元 Replication Server または送信先 Replication Server によって管理されるデータベースの数がいくつであっても、1 つの送信元 Replication Server から 1 つの送信先 Replication Server に対して作成するルートは 1 つです。

ルートは次の内容を転送します。

- 送信元 Replication Server で管理されるプライマリ・データベースから送信先 Replication Server で管理されるレプリケート・データベースへのデータ修正コマンドと、適用済みファンクションまたは適用済みストアド・プロシージャ。
- 送信元 Replication Server RSSD から送信先 Replication Server RSSD へのシステム・テーブル修正コマンド。
- レプリケート・データベースからプライマリ・データベースへの要求ファンクションまたは要求ストアド・プロシージャ (この場合は、レプリケート Replication Server が送信元で、プライマリ Replication Server が送信先)。

ルートを作成すると、送信元 Replication Server は次の処理を行います。

- 送信先サイトへのメッセージを保持するための、RSI アウトバウンド・キューの作成。
- 送信先 Replication Server にログインする RSI スレッドの起動と、RSI アウトバウンド・キューから送信先 Replication Server へのトランザクションの転送。

ルート指定の準備

ルートの作成または変更を行う前に、複写システムの関係するコンポーネントが実行されていること、および使用するルートが設計されていることを確認してください。

具体的には、次の準備が必要です。

- ルートがシステム内のどの場所で必要なのかについて事前に入念な判断をする。設計作業の一部として、各送信元 Replication Server とその送信先 Replication Server がどこに存在するかを知っておいてください。
- どのルートを直接ルートにして、どれを間接ルートにするかを特定します。間接ルートは、送信先 Replication Server までに 1 つ以上の中間 Replication Server を経てメッセージを転送します。直接ルートを使用するのと間接ルートを使用

するのとでは、システムのパフォーマンスに大きな違いが生じることがあります。

- 直接ルートを作成する場合は、送信元 Replication Server サイトの interfaces ファイルに、送信先 Replication Server を定義する。
  interfaces ファイルに送信先 Replication Server の RSSD のエントリがあることを確認する。
- 送信元、送信先、およびルート内にあるすべての中間 Replication Server が動作していることを確認する。
- 送信元 Replication Server RSSD の RepAgent スレッドが動作中であることを確認する。

『Replication Server デザイン・ガイド』を参照してください。

参照：

- ルート指定スキーム (163 ページ)

ルート指定ルール

ルート指定スキームを決定したら、ルート指定ルールに基づいて必要なルートを設定できます。

ルート指定ルールは以下のとおりです。

- プライマリ・データを格納しているデータベースを管理する Replication Server には、そのデータのサブスクリプションを持つデータベースを管理する Replication Server への直接ルートまたは間接ルートが必要です。
- 要求ファンクションを送出するレプリケート・データベースを管理する Replication Server には、プライマリ・データベースを管理する Replication Server への直接ルートまたは間接ルートが必要です。送出される複写ファンクションがレプリケート・データベースにない場合は、レプリケート Replication Server からプライマリ Replication Server へのルートは不要です。
- 間接ルートを構成する各中間ルートは、直接ルートであることが必要です。
- クラス・スコープを持つシステム・ファンクションのファンクション文字列は、ファンクション文字列クラスのプライマリ Replication Server でカスタマイズします。この場合、プライマリ Replication Server から、ファンクション文字列を使用するデータベースを管理する Replication Server へのルートを作成します。
  『Replication Server 管理ガイド 第 2 巻』の「データベース・オペレーションのカスタマイズ」で、「ファンクション、ファンクション文字列、クラスの処理」の「システム・ファンクションの概要」の「ファンクション文字列クラス・スコープを持つシステム・ファンクション」を参照してください。

- エラー・クラスは、プライマリ Replication Server でカスタマイズします。この場合、プライマリ Replication Server から、エラー・マッピング機能を使用するデータベースを管理する Replication Server へのルートを作成する必要があります。
- **move primary** コマンドを使用して、ファンクション文字列クラスまたはエラー・クラスの新しいプライマリ・サイトとして割り当てる Replication Server には、次の要件があります。
  - クラスの現在のプライマリ・サイトである Replication Server との間にルートが存在すること
  - クラスに対する現在のプライマリ・サイトである Replication Server とまったく同じ他の Replication Server へのルートが存在する。

  『Replication Server 管理ガイド 第 2 巻』の「エラーと例外の処理」で、「データ・サーバのエラー処理」の「エラー・クラスのプライマリ Replication Server の変更」を参照してください。また、『Replication Server 管理ガイド 第 2 巻』の「データベース・オペレーションのカスタマイズ」で、「Create a Function-string Class」の「ファンクション文字列クラスのプライマリ・サイト」も参照してください。

ルート指定スキーム

Replication Server では、直接ルート、間接ルート、および専用ルートがサポートされています。

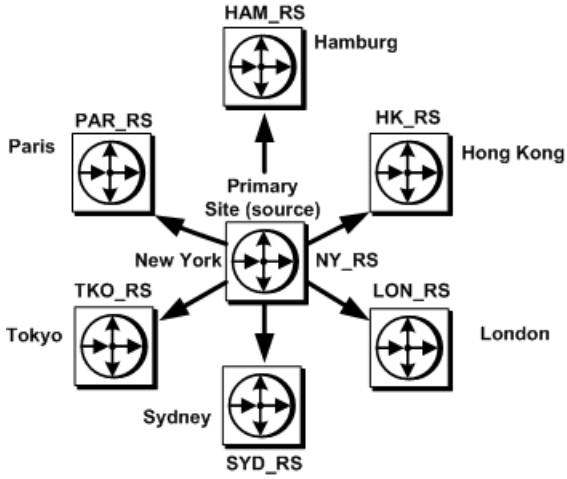
直接ルート

中間サイトのないルートを直接ルートと呼びます。直接ルートのシステムは、送信元 Replication Server と送信先 Replication Server 間のネットワーク・コネクションになります。

たとえば、次の図は、1 つのプライマリ・サイトと 6 つのレプリケート・サイトからなる 7 つのサイトを持つ企業の放射状構成を示しています。それぞれのレプリケート・サイトには、プライマリ・サイトを始点とするルートが 1 つずつあります。プライマリ・サイトからの 6 つのルートはすべて直接ルートです。したがって、プライマリ Replication Server には、ネットワークを介して 6 つのレプリケート・サイトに接続されている 6 つのステーブル・キューと 6 つの RSI スレッドがあります。

レプリケート・サイト TKO_RS がプライマリ・サイト NY_RS に要求ファンクションを送信する場合、システムには NY_RS から TKO_RS への直接ルートだけでなく、TKO_RS から NY_RS への直接ルートも必要となります。

図 11 : 直接ルート構成を使用して接続されたサイト



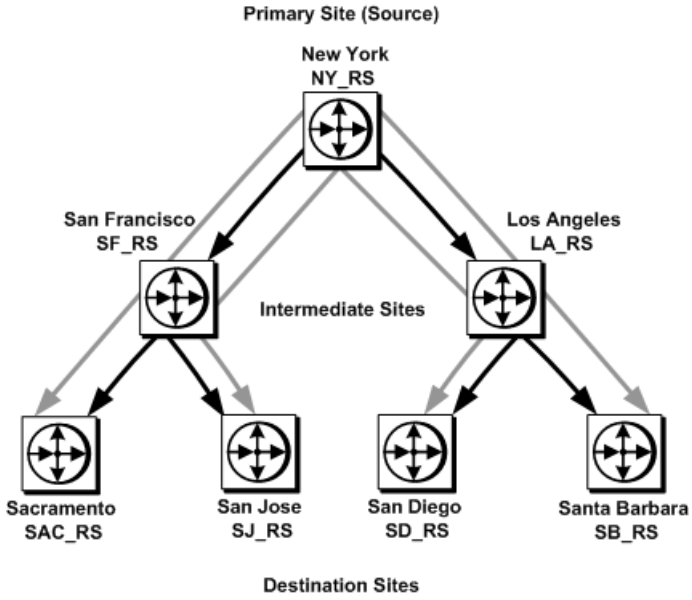
間接ルート

中間サイトのあるルートを間接ルートと呼びます。

次の例では、1 つのプライマリ・サイトと 6 つのレプリケート・サイトからなる 7 つのサイトを持つ企業の間接ルートを使用した構成を示しています。それぞれのレプリケート・サイトには、プライマリ・サイトを始点とするルートが 1 つずつあります。2 つのルートのみがプライマリ・サイトからの直接ルートであり、4 つのルートは間接ルートです。2 つの中間サイトには、それぞれ 2 つの直接ルートがあります。

この例では、NY_RS → SAC_RS のルートは、直接ルート NY_RS → SF_RS と SF_RS → SAC_RS で構成された間接ルートです。間接ルートでは、送信元 Replication Server は送信先 Replication Server へのメッセージを中間 Replication Server に送信し、中間 Replication Server は送信先 Replication Server へのルート (直接ルートまたは間接ルート) を利用します。

図 12 : 階層状構成の間接ルートで接続されたサイト



間接ルートを作成するには、目的の間接ルートに沿って連続する各 Replication Server 間に直接ルートを作成します。すべての直接ルートを作成してから、間接ルート自体を作成します。

たとえば、NY_RS→SAC_RS の間接ルートを作成するには、まず直接ルート NY_RS→SF_RS と SF_RS→SAC_RS を作成します。次に、作成済みの直接ルートをもとに間接ルートを作成します。

間接ルートを設定することによって、プライマリ・サイトでの処理量を減らし、中間 Replication Server に負荷を分散します。

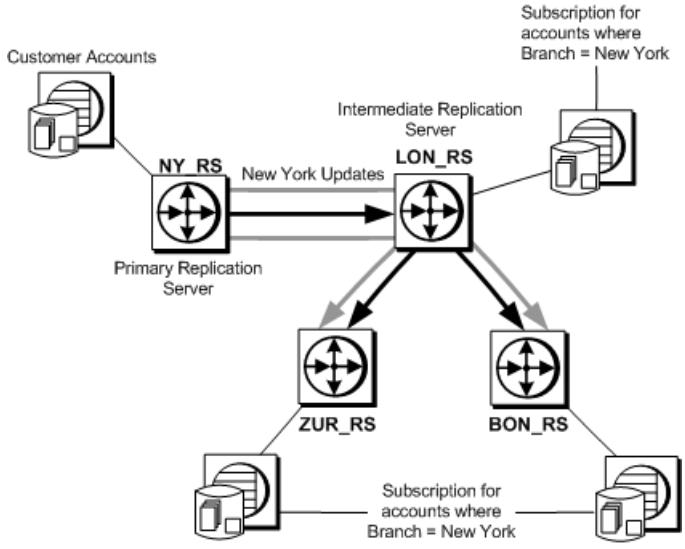
表 8 : 例に示されたサイト間の直接ルートと間接ルート

直接ルート	間接ルート
NY_RS → SF_RS	NY_RS → SAC_RS
NY_RS→LA_RS	NY_RS→SJ_RS
SF_RS→SAC_RS	NY_RS→SD_RS
SF_RS→SJ_RS	NY_RS→SB_RS
LA_RS→SD_RS	
LA_RS→SB_RS	

間接ルートを使用する場合、プライマリ Replication Server は、同じ中間サイトを経由する送信先サイトに共通しているサブスクリプションの部分をルート指定できます。サブスクリプションが重複している場合、プライマリ Replication Server は送信先サイトに共通している中間 Replication Server に対して、ローの変更ごとにメッセージを 1 つだけ送信すればよいことになります。

この重複するサブスクリプションを持つサイトの例では、LON_RS にある中間 Replication Server は、ニューヨークにある銀行本店で変更が発生するたびに、顧客口座のローに加えられた変更内容を受け取ります。ニューヨークでの変更は、チューリヒ支店とボン支店のレプリケート・サイトでも必要になります。LON_RS は ZUR_RS と BON_RS に変更内容を配信するように設定されているので、NY_RS プライマリ Replication Server は、各変更のコピーを 1 つだけ LON_RS に送信します。NY_RS → ZUR_RS と NY_RS → BON_RS の 2 つの間接ルートを使用することによって、直接ルートの数も減少します。

図 13 : 重複するサブスクリプションを持つサイト



間接ルートは、ネットワーク上の複数のサイトにコンピューティング・リソースを分配する場合に便利ですが、複数の Replication Server によってメッセージがキューに入れられるため、データの伝達が全体的に少し遅くなります。レプリケート・サイトが少ない場合は、直接ルートを使用することをおすすめします。間接ルートを使用する場合は、最適な伝達時間を実現するために中間サイトの数を最小限にしてください。

専用ルート

専用ルートは、特定のプライマリ・コネクションのトランザクションのみを分散します。レプリケート Replication Server への専用ルートを作成して、優先度が高いトランザクションを複写したり、特定のプライマリ・コネクション用に輻輳が少ないパスを維持できます。

注意： 2 つの Replication Server 間に直接ルートがある場合、作成できるのはこれらのサーバ間の 1 本の専用ルートだけです。Replication Server 間に間接ルートしかない場合、専用ルートは作成できません。

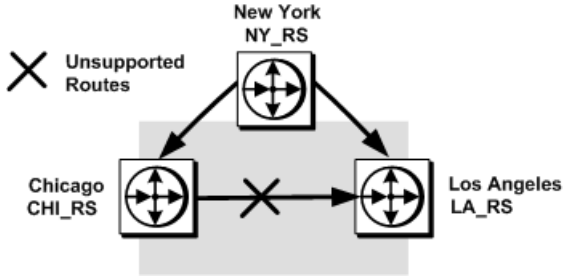
『Replication Server 管理ガイド 第 2 巻』の「パフォーマンス・チューニング」の「Multi-Path Replication」の「専用ルート」を参照してください。

サポートされていないルート指定スキーム

中間 Replication Server は、 1 つ以上の Replication Server からトランザクションを受け入れることができます。ただし、Replication Server は、ルートが同じ送信元 Replication Server から分岐し、同じ中間 Replication Server または送信先 Replication Server に収束するルート指定スキームはサポートしていません。

この例では、サポートできる NY_RS → LA_RS のルートは 1 つだけです。NY_RS から LA_RS へのルートがサポートされている場合、CHI_RS から LA_RS へのルートはサポートされません。

図 14 : サポートされるルートとサポートされないルートの例



ルートの作成

ルートは、送信元 Replication Server で作成します。

送信元 Replication Server と送信先 Replication Server 間に直接ルートを作成すると、送信元 Replication Server はすぐに次の処理を実行します。

- 送信先サイトへのメッセージを保持するために、RSI アウトバウンド・ステーブル・キューを作成する。
- 送信先 Replication Server またはルート上の次の Replication Server にログインする、RSI スレッドを開始する。

注意： バージョン 15.0 の Replication Server から、旧バージョンの Replication Server (バージョン 11.03 以降) へのルートを作成することもできます。

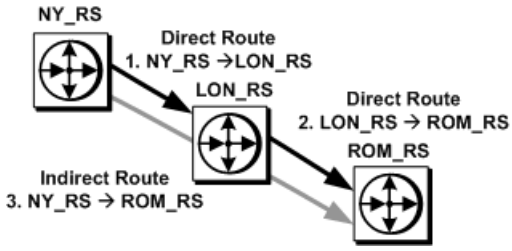
直接ルートまたは間接ルートのどちらかを作成すると、送信先 Replication Server は、複写 RSSD システム・テーブル用に送信先サイトでサブスクリプションを作成して、マテリアライズします。この処理によって、送信先 Replication Server は、複写定義とファンクション・クラスを受信できるようになります。

間接ルートを作成する場合、Replication Server は RSI キューを作成しません。間接ルートでは、間接ルートを構成している直接ルート・セグメントの RSI アウトバウンド・キューを使用します。

Replication Server が送信先 Replication Server にシステム情報の転送を開始できるようになるには、直接ルートを作成してから間接ルートを作成する必要があります。

たとえば、2 つの直接ルート (1 → 2 と 2 → 3) を事前に作成しておかないと、間接ルート (1 → 3) は作成できません。また、正しい順序でルートを設定する必要があります。

図 15 : 直接ルートと間接ルートの作成順序



参照：

- Replication Server の技術的概要 (27 ページ)

create route コマンド

create route には、いくつかの前提条件、ガイドライン、および設定パラメータがあります。

注意： Sybase Central でもルートを作成できます。

create route コマンドの構文は次のとおりです。

```
create route to dest_replication_server{
 with primary at dataserver.database|
    set next site [to] thru_replication_server|
    [set username [to] user]
    [set password [to] passwd]
    [set route_param to 'value']
    [set security_param to 'value']}
```

ルートを作成するには、次の手順に従います。

- 直接ルート専用のログイン名、パスワード、その他のパラメータを指定します。
- 直接ルートのログイン名とパスワードを送信先 Replication Server に作成してから、直接ルートを作成します。オプションとして、**rs_init** ユーティリティでこのユーザを作成できます。
  - ネットワークベース・セキュリティと統合ログインを有効にしている場合は、ユーザ名とパスワードは省略可能です。デフォルトのユーザ名はプリンシパル・ユーザ名であり、Replication Server にログインするとき、または Replication Server を起動するときに **-S** フラグで指定します。
  - 送信先 Replication Server に存在しない *user* と *passwd* を使用してルートを作成する場合、その送信先に *user* と *passwd* を追加または変更します。
  - 現在の Replication Server から送信先 Replication Server への直接ルートを設定する場合は、**next site** 句は使用しないでください。
- 専用ルートを作成するには **with primary at** 句を使用します。

  注意： 2 つの Replication Server 間に直接ルートがある場合、作成できるのはこれらのサーバ間の 1 本の専用ルートだけです。Replication Server 間に間接ルートしかない場合、専用ルートは作成できません。
- 間違いを防ぐために、**create route** コマンドは一度に 1 つずつ入力してください。ルートが有効になるまで待ってから、次のルートを作成します。
  間違えた場合は、最後の手段としてそのルートだけを削除し、再作成します。**drop route** コマンドには、**with nowait** オプションを指定します。ルートは作成されていないので、現在の状態で削除するには **with nowait** オプションを使用する必要があります。

ルートの作成時には、メモリ・サイズ、ルートを介して一度に送信できるデータ量のサイズ、タイムアウト、同期インターバルを管理する設定パラメータに対して、デフォルト値を受け入れることができます。また、ルートの作成時または変更時に、独自の値を設定することもできます。

サイトでネットワークベース・セキュリティが有効になっている場合は、ルートのセキュリティ・パラメータも設定できます。パフォーマンスに影響を及ぼすルート・パラメータのリストおよび説明については、『Replication Server 管理ガイド 第 2 巻』の「パフォーマンス・チューニング」の「パフォーマンスに影響する設定パラメータ」を参照してください。

参照：
- プリンシパル・ユーザの設定 (268 ページ)
- ルートの削除 (183 ページ)
- ネットワークベース・セキュリティの管理 (262 ページ)
- 間接ルートを直接ルートに変更する方法 (177 ページ)

ルートに影響を与える設定パラメータ

ルートを構成するために使用できるいくつかのパラメータがあります。

表 9 : ルートに影響を与える設定パラメータ

route_param	*value*
disk_affinity	次のパーティションを割り当てるための割り付けヒントを指定する。現在のパーティションが満杯になった場合に、次のセグメントの割り付け先となるパーティションの論理名を入力する。指定できる値は “*partition_name*” と “off”。 デフォルト値は off
rsi_batch_size	トランケーション・ポイントが要求される前に、別の Replication Server に送信されるバイト数。 デフォルト値は 256KB 最小値：1K 最大値：128MB
rsi_fadeout_time	Replication Server が送信先 Replication Server とのコネクションをクローズするまでのアイドル時間 (秒単位)。 デフォルト値は-1 (Replication Server はコネクションをクローズしない)
rsi_packet_size	他の Replication Server との通信に使用するパケット・サイズ (バイト単位)。値の範囲は、1,024 ～ 16,384。 デフォルト値は 4096 バイト
rsi_sync_interval	RSI 同期確認メッセージ間の秒数。Replication Server は、RSI アウトバウンド・キューと送信先 Replication Server の同期をとるために、これらのメッセージを使用する。この値は 0 よりも大きくすること。 デフォルト値は 60 秒

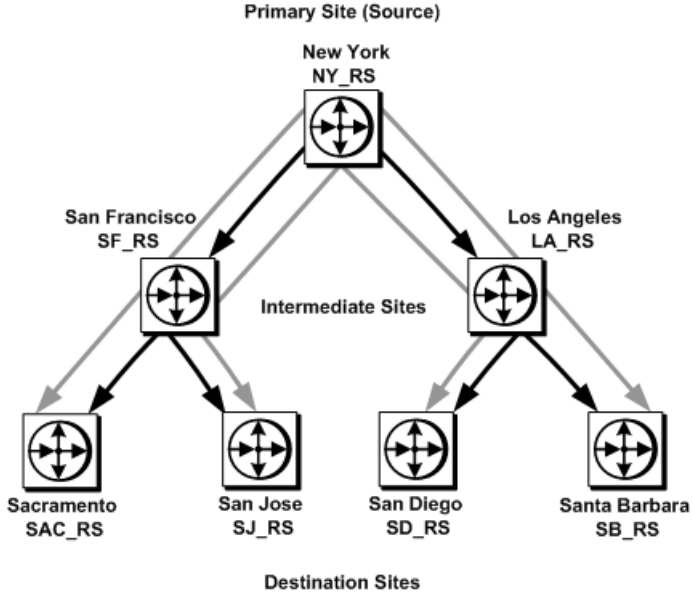
route_param	*value*
rsi_xact_with_large_msg	サイズの大きいメッセージが検出された場合のルートの動作を指定する。このパラメータは、レプリケート・サイトのサイト・バージョンが 12.1 以前で、直接ルートの場合にのみ適用される。指定できる値は “skip” または “shutdown”。 デフォルト値は shutdown
save_interval	メッセージが送信先 Replication Server に正常に渡された後に、Replication Server がそのメッセージを保存しておく時間 (分単位)。 デフォルト値は 0 分

直接ルートと間接ルートの作成例

例を使用して、直接および間接ルートを作成し、設定パラメータを設定する方法について説明します。

プライマリ Replication Server から中間 Replication Server への直接ルートと、中間 Replication Server から送信先 Replication Server への直接ルートを作成してから、間接ルートを作成する必要があります。

図 16 : 階層状構成の間接ルートで接続されたサイト



例 1
プライマリ Replication Server である NY_RS から SF_RS への直接ルートを作成するには、NY_RS で次のコマンドを入力します。

```
create route to SF_RS
set username SF_rsi_user
set password SF_rsi_ps
```

例 2
直接ルート SF_RS → SAC_RS と SF_RS → SJ_RS を作成するには、中間 Replication Server である SF_RS で次のコマンドを入力します。

```
create route to SAC_RS
set username SAC_rsi_user
set password SAC_rsi_ps
```

```
create route to SJ_RS
set username SJ_rsi_user
set password SJ_rsi_ps
```

例 3
直接ルートを作成したら、それらを経由する間接ルートを作成できます。

次の例では、中間サイト SF_RS を経由して、プライマリ・サイト NY_RS からサイト SAC_RS と SJ_RS への間接ルートを作成します。プライマリ Replication Server である NY_RS で次のコマンドを入力します。

```
create route to SAC_RS
set next site SF_RS
```

```
create route to SJ_RS
set next site SF_RS
```

例 4
ルートの作成とパラメータの設定は同時に実行できます。

SF_RS へのルートに対して **rsi_packet_size** を 4,096 バイトに設定するには、次のように入力します。

```
create route to SF_RS
set username SF_rsi_user
set password SF_rsi_ps
set rsi_packet_size to '4096'
```

参照：
- 間接ルート (164 ページ)

プライマリ・テーブルを管理するための Replication Server の設定

レプリケート専用 Replication Server として設定されていた Replication Server からのルートを追加できます。

まず、Replication Server RSSD 用に RepAgent を設定する必要があります。プライマリ・データベースとして機能するデータベースにも RepAgent が必要です。Replication Server およびプライマリ Adaptive Server データベースで RepAgent を設定する必要があります。

1. Replication Server で RepAgent を次のように設定します。
   a) RepAgent が Replication Server にログインできるように、RepAgent ユーザを作成します。**create user** コマンドを使用します。ここで、*ra_user_name* は RepAgent ユーザの名前、*ra_password* は RepAgent のパスワードです。
   ```
   create user ra_user_name
   set password {ra_password | null}
   ```
   b) **grant** コマンドを使用して、このユーザに **connect source** パーミッションを付与します。
   ```
   grant connect source to ra_user_name
   ```
   Replication Server がすでにプライマリ・データベースを管理している場合は、既存の「RepAgent ユーザ」を新しいプライマリ・データベース用に使用できます。
   c) **log transfer on** オプションを付けて、**alter connection** コマンドを実行します。
   ```
   alter connection to data_server.database
   set log transfer to 'on'
   ```
2. Adaptive Server データベースで次のように RepAgent を設定します。
   a) Adaptive Server の名前がまだ定義されていない場合は、次のコマンドを使用して名前を定義します。ここで、*lname* は RSSD の名前です。
   ```
   sp_addserver lname, local
   ```
   b) Adaptive Server の RepAgent スレッドが有効になっていない場合は、次のようにして有効にします。
   ```
   sp_configure 'enable rep agent threads'
   ```
   c) **sp_config_rep_agent** システム・プロシージャを使用して、RSSD の RepAgent を次のように設定します。
   ```
   sp_config_rep_agent dbname, 'enable', 'rs_name',
           'rs_user_name', 'rs_password'
   ```

注意： Adaptive Server で設定する *rs_user_name* と *rs_password* は、手順 1 において Replication Server で作成した *ra_user_name* および *ra_password* と同じ値にする必要があります。

d) 次のコマンドを入力して RepAgent を起動します。

```
sp_start_rep_agent dbname
```

参照：

- RepAgent の設定 (118 ページ)

ルートのサスペンドおよびレジューム

直接ルートを変更する場合、そのトポロジを変更する場合、またはリモート・サイトに対してその他のメンテナンスを実施する場合は、メッセージが送信先 Replication Server に送信されないようにルートをサスペンドします。ルートのメンテナンスが完了したら、ルートを再度アクティブにしてアクティビティをレジュームできます。

ルートのサスペンドとレジュームは、Sybase Central で実行することも、**suspend route** および **resume route** RCL コマンドを使用して実行することもできます。

専用ルートのサスペンドとレジュームは、**suspend route** および **resume route** を使用して実行できます。『Replication Server 管理ガイド 第 2 巻』の「パフォーマンス・チューニング」の「マルチパス・アプリケーション」の「専用ルート」を参照してください。

suspend route

suspend route ― 別の Replication Server へのルートをサスペンドします。

ルートがサスペンドされている間、送信先 Replication Server にメッセージは送信されず、Replication Server へのメッセージはステーブル・キューに保持されます。**suspend route** コマンドの構文は次のとおりです。

```
suspend route to dest_replication_server
[with primary at dataserver.database]
```

コマンドに **with primary at** *dataserver.database* 句を含めると、専用ルートを指定できます。ここで、*dataserver.database* はプライマリ・コネクションの名前です。

たとえば、CHI_RS という名前の Replication Server へのルートをサスペンドするには、次のように入力します。

```
suspend route to CHI_RS
```

resume route

resume route ― サスペンドされているルートをレジュームします。

ルートをレジュームすると、送信元 Replication Server はキュー内のメッセージを送信先 Replication Server に送信できるようになります。また、このコマンドを使用して、エラーの結果として自動的にサスペンドされたルートをレジュームすることもできます。**resume route** コマンドの構文は次のとおりです。

```
resume route to dest_replication_server
[with primary at dataserver.database |
skip transaction with large message]
```

コマンドに **with primary at** *dataserver.database* 句を含めると、専用ルートを指定できます。ここで、*dataserver.database* はプライマリ・コネクションの名前です。

たとえば、CHI_RS という名前の Replication Server へのルートをレジュームするには、次のように入力します。

```
resume route to CHI_RS
```

ルートの変更

直接ルートのトポロジ、ユーザ名、パスワード、一部の設定パラメータは、**alter route** コマンドまたは Sybase Central を使用して変更できます。

注意： **alter route** を使用して、間接ルートのパラメータを変更することはできません。

alter route の構文は次のとおりです。

```
alter route to dest_replication_server{
    set next site [to] thru_replication_server |
    set username [to] 'user' set password [to] 'passwd' |
    set password [to] 'passwd' |
    set route_param [to] 'value' |
    set security_param [to] 'value' |
    set security_services [to] 'default'}
```

ルートを変更するには、次の手順に従います。

1. ルートをサスペンドします。
2. **alter route** コマンドを該当する設定パラメータを使用して実行します。
3. 変更を有効にするために、ルートをレジュームします。

参照：

- ネットワークベース・セキュリティの管理 (262 ページ)

ルートのトポロジの変更

ルートのトポロジを変更するには、直接ルートを間接ルートに変更するか、間接ルートを直接ルートに変更するか、または既存ルート内の次の中間サイトを変更します。

直接ルートを間接ルートに変更する方法

既存の直接ルートを間接ルートに変更します。

1. 直接ルートの始点である送信元 Replication Server で、次のように入力します。
   ```
   suspend route to dest_replication_server
   ```
2. RepAgent のあるデータベースを管理する各 Replication Server で、次のように入力します。
   ```
   suspend log transfer from all
   ```
   複写システムをクワイス状態にすることで、メッセージがエラーを発生しないで新しいルート設定に変更されるようにします。
3. 新しい間接ルートが使用する追加ルートを作成します。
   - 現在の Replication Server に、新しい間接ルートの中間サイトとして指定する Replication Server に対する直接ルートがまだない場合は、そのルートを作成します。
   - 新しい間接ルートの中間サイトとして指定する Replication Server に、送信先サイトへの直接ルートまたは間接ルートがまだない場合は、そのルートを作成します。
4. 間接ルートに変更する直接ルートに対して、送信元 Replication Server で次のコマンドを入力します。ここで、*dest_replication_server* は変更するルートの送信先 Replication Server であり、*thru_replication_server* はルートの中間 Replication Server です。
   ```
   alter route to dest_replication_server
   set next site [to] thru_replication_server
   ```
5. 事前にログ転送をサスペンドしておいた各 Replication Server で、次のコマンドを入力してログ転送コネクションをレジュームします。
   ```
   resume log transfer from all
   ```
6. 送信元 Replication Server で次のコマンドを入力して、サスペンドされているルートをレジュームします。
   ```
   resume route to dest_replication_server
   ```

参照：

- 複写システムのクワイス (109 ページ)

間接ルートの次の中間サイトを変更する方法

既存の間接ルート上にある次の中間サイトを変更します。

1. 直接ルートの始点である送信元 Replication Server で、次のコマンドを入力します。
   ```
   suspend route to dest_replication_server
   ```
2. RepAgent のあるデータベースを管理する各 Replication Server で、次のように入力します。
   ```
   suspend log transfer from all
   ```
   複写システムをクワイス状態にすることで、メッセージがエラーを発生しないで新しいルート設定に変更されるようにします。
3. 間接ルートが使用する追加ルートを作成します。
   - 現在の Replication Server に、間接ルートの新しい中間サイトとして指定する Replication Server に対する直接ルートがまだない場合は、そのルートを作成します。
   - 間接ルートの新しい中間サイトとして指定する Replication Server に、送信先サイトへの直接ルートまたは間接ルートがまだない場合は、そのルートを作成します。
4. 新しい中間 Replication Server を指定する間接ルートに対して、送信元 Replication Server で次のコマンドを入力します。ここで、*dest_replication_server* は変更するルートの送信先 Replication Server であり、*thru_replication_server* はルートの新しい中間 Replication Server です。
   ```
   alter route to dest_replication_server
   set next site thru_replication_server
   ```
5. 事前にログ転送をサスペンドしておいた各 Replication Server で、次のコマンドを入力してログ転送コネクションをレジュームします。
   ```
   resume log transfer from all
   ```
6. 送信元 Replication Server で次のコマンドを入力して、サスペンドされているルートをレジュームします。
   ```
   resume route to dest_replication_server
   ```

参照：

- 複写システムのクワイス (109 ページ)

間接ルートを直接ルートに変更する方法

既存の間接ルートを直接ルートに変更します。

1. 間接ルートの始点である送信元 Replication Server で、次のコマンドを入力します。

```
suspend route to dest_replication_server
```

2. RepAgent のあるデータベースを管理する各 Replication Server で、次のように入力します。

```
suspend log transfer from all
```

   複写システムをクワイス状態にすることで、メッセージがエラーを発生しないで新しいルート設定に変更されるようにします。

3. 直接ルートに変更する間接ルートに対して、送信元 Replication Server で次のコマンドを入力します。ここで、*dest_replication_server* は変更するルートの送信先 Replication Server であり、*user* と *passwd* は直接ルートに使用する RSI ユーザのログイン名とパスワードです。

```
alter route to dest_replication_server
set username user set password passwd
```

4. 事前にログ転送をサスペンドしておいた各 Replication Server で、次のコマンドを入力してログ転送コネクションをレジュームします。

```
resume log transfer from all
```

5. 送信元 Replication Server で次のコマンドを入力して、サスペンドされているルートをレジュームします。

```
resume route to dest_replication_server
```

参照：

- 複写システムのクワイス (109 ページ)

直接ルートの RSI ユーザのパスワードを変更する方法

alter route に **set password** を指定して、既存の直接ルートの RSI ユーザ・パスワードを変更します。

1. 送信元 Replication Server で、次のコマンドを入力して各直接ルートをサスペンドします。

```
suspend route to dest_replication_server
```

2. 送信元 Replication Server で、次のコマンドを入力します。ここで、*dest_replication_server* は変更するルートの送信先 Replication であり、*passwd* は RSI ユーザ・ログイン名に使用するパスワードです。

```
alter route to dest_replication_server
set password passwd
```

3. 送信元で、次のコマンドを入力してサスペンドされている各ルートをレジュームします。

```
resume route to dest_replication_server
```

直接ルートに影響するパラメータの変更

特定の直接ルートのパラメータをルートの作成後に変更するには、**alter route** または Sybase Central を使用します。

alter route を使用してそれぞれのルートに設定される設定パラメータは、**configure replication server** で設定されるデフォルト・パラメータを上書きします。したがって、**configure replication server** を使用してデフォルト・パラメータを設定してから、**alter route** でそれぞれのルートに対する設定をカスタマイズすることができます。

たとえば、**alter route** を使用して **rsi_sync_interval** パラメータを 120 秒に変更するには、送信元の Replication Server にログインしてから、次の手順に従います。

1. ルートを次のようにサスペンドします。
   ```
   suspend route to dest_replication_server
   ```
2. **alter route** コマンドを実行します。
   ```
   alter route to dest_replication_server
   set rsi_sync_interval to '120'
   ```
3. サスペンドされているルートを次のようにレジュームします。
   ```
   resume route to dest_replication_server
   ```

設定の変更は、ルートをレジュームしたあとに有効になります。

参照：

- ルートに影響を与える設定パラメータ (170 ページ)
- すべてのルートの設定パラメータの変更 (179 ページ)

すべてのルートの設定パラメータの変更

送信元 Replication Server を始点とするすべてのルートのデフォルト設定パラメータを変更するには、**configure replication server** コマンドを使用します。

alter route を使用してそれぞれのルートに設定される設定パラメータは、**configure replication server** で設定されるデフォルト・パラメータを上書きします。したがって、**configure replication server** を使用してデフォルト・パラメータを設定してから、**alter route** でそれぞれのルートに対する設定をカスタマイズすることができます。

configure replication server でルート・パラメータを変更する構文は、次のとおりです。

```
configure replication server
 set route_param to 'value'
```

次に、**configure replication server** を使用して **rsi_save_interval** パラメータを 2 分に変更する例を示します。このコマンドを実行するには、送信元 Replication Server にログインして次の手順に従います。

1. 送信元 Replication Server からのルートをすべてサスペンドします。各ルートについて、次のように入力します。
   ```
   suspend route to dest_replication_server
   ```
2. **configure replication server** コマンドを実行します。
   ```
   configure replication server
    set rsi_save_interval to '2'
   ```
3. 送信元 Replication Server からサスペンドされているルートをレジュームします。各ルートについて、次のように入力します。
   ```
   resume route to dest_replication_server
   ```

設定の変更は、ルートをレジュームした後に有効になります。

参照：

- ルートに影響を与える設定パラメータ (170 ページ)

ルート指定の変更例

ルート指定スキームを変更する方法について説明します。

この例では、「階層状構成の間接ルートで接続されたサイト」の図のルート指定スキームを「変更された間接ルート」の図のスキームに変更し、LA_RS が NY_RS と SF_RS の間の中間サイトになると同時に SB_RS への直接および間接ルートが削除されるようにします。

図 17 : 階層状構成の間接ルートで接続されたサイト

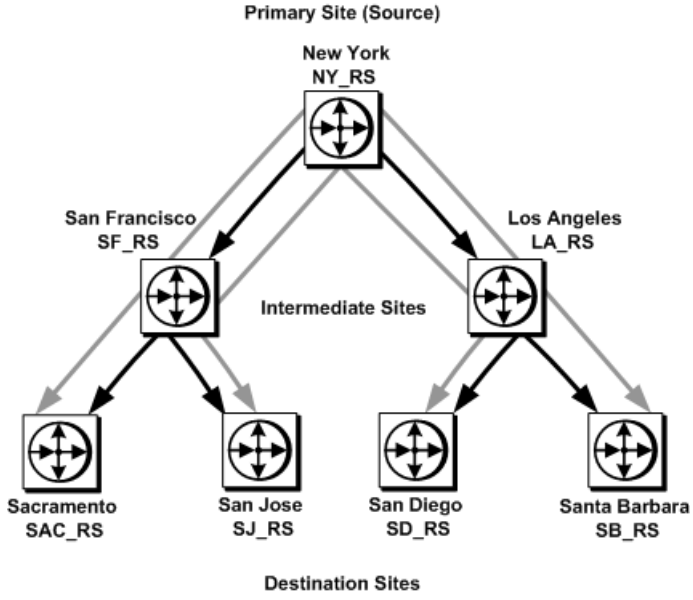
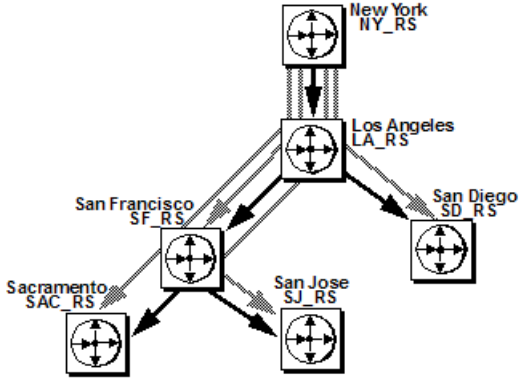


図 18 : 変更された間接ルート



1. RepAgent のあるデータベースを管理する各 Replication Server で、次のように入力します。

   ```
   suspend log transfer from all
   ```

   複写システムをクワイス状態にすることで、メッセージがエラーを発生しないで新しいルート設定に変更されるようにします。
2. LA_RS には SF_RS への直接ルートが必要なので、Replication Server LA_RS で次のコマンドを入力して、直接ルートを作成します。

```
create route to SF_RS
set username SF_rsi_user
set password SF_rsi_ps
```

3. LA_RS には、SF_RS を経由して SAC_RS と SJ_RS に至る間接ルートが必要です。

   これらのルートを作成すると、SAC_RS と SJ_RS 宛てのメッセージを SF_RS に送信するように LA_RS が指示されます。SF_RS には、SAC_RS と SJ_RS への直接ルートがすでにあります。Replication Server LA_RS で次のコマンドを入力します。

   ```
   create route to SAC_RS
   set next site SF_RS
   ```

   ```
   create route to SJ_RS
   set next site SF_RS
   ```

4. プライマリ Replication Server である NY_RS は、以前は SF_RS を経由して SAC_RS と SJ_RS に至る間接ルートを持つように設定されていました。Replication Server LA_RS が次の Replication Server になるように、これらのルートを変更します。Replication Server NY_RS で次のコマンドを入力します。

   ```
   alter route to SAC_RS
   set next site LA_RS
   ```

   ```
   alter route to SJ_RS
   set next site LA_RS
   ```

5. プライマリ Replication Server である NY_RS から SF_RS への直接ルートを、LA_RS を中間 Replication Server とする間接ルートに変更する必要があります。Replication Server NY_RS で次のコマンドを入力します。

   ```
   alter route to SF_RS
   set next site LA_RS
   ```

6. 事前にログ転送をサスペンドしておいた各 Replication Server で、次のコマンドを入力して各 Replication Server へのログ転送コネクションをレジュームします。

   ```
   resume log transfer from all
   ```

7. NY_RS から SB_RS への間接ルートを削除します。NY_RS で次のコマンドを入力します。

   ```
   drop route to SB_RS
   ```

8. LA_RS から SB_RS への直接ルートを削除します。LA_RS で次のコマンドを入力します。

   ```
   drop route to SB_RS
   ```

   LA_RS を経由して NY_RS から SD_RS に至る間接ルートは、そのままです。

参照：

- ログ転送の再開 (129 ページ)

- 複写システムのクワイス (109 ページ)

ルートの削除

ルートを削除すると、コマンドを実行した Replication Server から指定したリモート Replication Server へのルートがクローズされます。

ルートを削除すると、関係している Replication Server で次の処理が実行されます。

- システム・テーブル・サブスクリプションが削除される。
- 直接ルートの場合は、アウトバウンド・ステーブル・キューが削除され、RSI スレッドが停止される。
- ルートについての情報が削除される。

次の場合は、ルートを削除できません。

- ルートが、別の送信先 Replication Server への間接ルートによって使用されている直接ルートである場合。
- 送信元 Replication Server が、送信先 Replication Server によってサブスクリプションを作成される複写定義を持っている場合。
- 送信元 Replication Server が、ファンクション文字列クラスまたはエラー・クラスのプライマリ・サイトとして指定されている場合。派生ファンション文字列クラスのプライマリ・サイトは、その親クラスと同じです。

削除中のルートのステータスは、次のいずれかの方法でモニタすることができます。

- Sybase Central では、Sybase Central メイン・ウィンドウの右ウィンドウ枠にステータス情報が表示されます。
- コマンド・ラインから、**rs_helproute** ストアド・プロシージャを実行します。

drop route コマンド

drop route コマンドを使用してルートを削除します。

Sybase Central からルートを削除することもできます。**drop route** コマンドの構文は次のとおりです。

```
drop route to dest_replication_server
 [with primary at dataserver.database]
 [with nowait]
```

コマンドに **with primary at** *dataserver.database* 句を含めると、専用ルートを指定できます。ここで、*dataserver.database* はプライマリ・コネクションの名前です。専用ルートを削除してから、共有ルートを削除してください。専用ルートが削除されると、指定プライマリ・コネクションから送信先 Replication Server へのトランザクションは、共有ルートを通るようになります。『Replication Server 管理ガイド

第 2 巻』の「パフォーマンス・チューニング」の「Multi-Path Replication」の「専用ルート」を参照してください。

with nowait オプションは、送信先 Replication Server と通信できない場合であっても、ルートをクローズするように Replication Server に指示します。

警告！ with nowait 句は、最後の手段としてのみ使用してください。**with nowait** 句は、送信先 Replication Server を今後使用する予定がない場合、または送信先 Replication Server がダウンしているときに送信元 Replication Server からのルートを削除しなければならない場合、または直接ルートのログイン名とパスワードを追加または変更しようとする場合にだけ使用してください。これ以外の場合は、できる限り **with nowait** 句を使用しないでください。

この句を使用すると、ルートのアウトバウンド・キューにトランザクションが含まれている場合でも、ルートが強制的に削除されます。その結果、Replication Server はプライマリ・コネクションから一部のトランザクションを破棄する可能性があります。この句は、専用ルートが送信先 Replication Server と通信できない場合でも、専用ルートを削除するように、Replication Server に指示します。

with nowait 句を使用した後、**sysadmin purge_route_at_replicate** コマンドを使用して、サブスクリプションやルート情報などのプライマリ Replication Server に対する参照をレプリケート Replication Server のシステム・テーブルからすべて削除します。

---

sysadmin purge_route_at_replicate コマンド

sysadmin purge_route_at_replicate は、指定したプライマリ Replication Server からルートが削除された後に、その Replication Server を始点とするすべてのサブスクリプションとルート情報を削除します。

レプリケート Replication Server からプライマリ Replication Server へのルートが存在する場合は、そのルートを削除してから、このコマンドを実行します。Replication Server は、確定化チェックを実行してからこのコマンドを処理します。

レプリケート Replication Server で、次の構文を使用して **sysadmin purge_route_at_replicate** を実行します。ここで、*replication_server* はプライマリ Replication Server です。

```
sysadmin purge_route_at_replicate, replication_server
```

ルートのアップグレード

ルートをアップグレードすると、Replication Server はソフトウェアの新機能に関する情報を交換できるようになります。

ルートのバージョンとは、送信元 Replication Server と送信先 Replication Server の最も古いサイト・バージョンのことです。ルートの送信元 Replication Server と送信先 Replication Server をルートの一方の端でアップグレードし、そのサイト・バージョンも新しい Replication Server バージョンに設定した後で、ルートをアップグレードする必要があります。

ルートをアップグレードすると、システム・テーブル内のデータが再マテリアライズされるため、新しくアップグレードした Replication Server が新機能に関する情報を利用できるようになります。アップグレード後は、以前は利用できなかった新しいタイプの情報を交換できます。

ある Replication Server を始点または終点とするルートの現在のバージョン番号を表示するには、**admin show_route_versions** コマンドを使用します。

構文の詳細と使用方法については、『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**admin show_route_versions**」を参照してください。

ルートをアップグレードする場合、次の 2 つのケースが考えられます。

- 送信元 Replication Server で新機能が使用されていない場合は、**sysadmin fast_route_upgrade** を使用してルートをアップグレードする。
- 上記以外のケースでは、**route_upgrade**、**route_upgrade_recovery**、**route_upgrade_status** の各コマンドを使用してルートをアップグレードする。

ルートは、一度アップグレードすると、ダウングレードできません。また、混合バージョンの複写システムを使用している場合は制限があります。

ルートのアップグレードと Replication Server のサイト・バージョンの設定の詳細については、使用しているプラットフォームの『Replication Server インストール・ガイド』と『Replication Server 設定ガイド』を参照してください。

参照：

- 混合バージョンの複写システム (20 ページ)

ルートのモニタリング

ルートは、時間の経過とともにステータスがさまざまに変化することがあります。ルートのステータスをモニタするために使用できるユーティリティの概要を取得します。

ルートを作成すると、送信先 Replication Server は送信元 Replication Server のシステム・テーブルに対してサブスクリプションを作成します。処理するデータの量によっては、サブスクリプションのマテリアライゼーションに数分かかることがあります。ルートの削除も、ある程度時間がかかることがあります。

- **admin who** コマンドを使用すると、スレッド・ステータス情報を表示できます。
- 作成中のルートの現在のステータスなど、包括的なステータス情報を確認するには、**rs_helproute** を使用します。

admin who を使用した RSI スレッド・ステータスの表示

admin who を使用すると、RSI スレッド情報が表示できます。

- **admin who** は、RSI スレッドなど、システム内のスレッドをすべて表示します。
- **admin who, rsi** は、RSI スレッドのステータスを表示します。RSI スレッドは、Replication Server が他の Replication Server に情報を送信するために開始するスレッドです。

スレッド・ステータス情報の詳細については、『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**admin who**」を参照してください。

rs_helproute ストアド・プロシージャの使用

作成中のルートの現在のステータスなど、包括的なルート・ステータス情報を取得するには、**rs_helproute** を使用します。

ルートの送信元 Replication Server または送信先 Replication Server の RSSD で、**rs_helproute** ストアド・プロシージャを実行します。**rs_helproute** ストアド・プロシージャの構文は次のとおりです。

```
rs_helproute [replication_server]
```

Replication Server の名前を指定すると、**rs_helproute** は、指定した Replication Server が送信元または送信先になっているルートの情報だけを返します。名前を指定しない場合は、現在の Replication Server が送信元または送信先になっているルートすべてについての情報を返します。

rs_helproute は、次の 2 タイプの情報を返します。

- ルート・ステータス。これは、**rs_helproute** を実行するサイトでのルートの状態を表します。ルートは、送信元と送信先の両方の **rs_helproute** が “Active” を返す場合、有効です。
  他のルート・ステータス値は、次のとおりです。
  - Being created (作成中)
  - Being dropped (削除中)
  - Being dropped **with nowait** (with nowait で削除中)
- システム・テーブル・サブスクリプションのリスト。これは、作成中のシステム・テーブル・サブスクリプションを示します。ルートを削除中の場合は、どのサブスクリプションが削除中であるかを示します。
  システム・テーブル・サブスクリプションがリストに表示されていない場合、そのルートは作成済みで動作中であることを示します。

『Replication Server リファレンス・マニュアル』の「RSSD ストアド・プロシージャ」の「**rs_helproute**」を参照してください。

ルートの作成に関する問題を解決する方法については、『Replication Server トラブルシューティング・ガイド』を参照してください。

データベース・コネクションの管理

データベース・コネクションの管理には、データベースの複写準備、データベースへのコネクションの作成、最適な複写パフォーマンスのためのコネクションの設定などの作業があります。

データベースの複写準備

複写システムにデータベースを追加するには、Replication Server がプライマリ・データを分配して、格納された複写データを管理できるように、事前にデータベースを準備する必要があります。

- Sybase Adaptive Server が管理するデータベースの場合

  Adaptive Server データベースを Replication Server で使用できるように準備するには、Sybase Central または **rs_init** を使用します。

  rs_init の詳細については、『Replication Server インストール・ガイド』と『Replication Server 設定ガイド』を参照してください。
- ASE 以外のデータ・サーバが管理するデータベースの場合

  必要な準備作業については、『Replication Server デザイン・ガイド』を参照してください。また、異機種データ型サポート (HDS: Heterogeneous Datatype Support) 機能を使用できるようにデータベースを準備する方法については、使用しているプラットフォームの『Replication Server 設定ガイド』を参照してください。HDS を使用すると、プライマリ・データベースの一定のデータ型のカラム値を、レプリケート・データベースで使用可能な別のデータ型に変換できます。

新しいデータベースを既存のシステムに接続する場合は、そのデータベースが各システムに適合するかどうかについて常に十分な検討と解析を行ってください。そのデータベースに必要な他の処理について調べて、その処理に必要な名前とログイン名を指定します。

既存の「複写専用」データベースが将来、複写ファンクションの配信元になるかプライマリ・データを格納するようになる可能性がある場合には、プライマリ・テーブルを管理できるようにそのデータベースを設定できます。このような設定を行っておけば、将来このような変更が発生した場合に複写専用データベースをアップグレードする必要がなくなります。

参照：

- HDS を使用したデータ型の変換 (377 ページ)

Adaptive Server データベースの複写準備

Adaptive Server データベースを複写できるように準備するには、Sybase Central または **rs_init** を使用します。

1. `rs_lastcommit` システム・テーブルを作成します。
2. **rs_update_lastcommit** ストアド・プロシージャと **rs_get_lastcommit** ストアド・プロシージャをプライマリ・データベースとレプリケート・データベース用にロードし、**rs_marker** ストアド・プロシージャをプライマリ・データベース専用にロードします。
3. `rs_threads` システム・テーブルを作成します。
4. **rs_initialize_threads** ストアド・プロシージャと **rs_update_threads** ストアド・プロシージャをデータベースにロードします。
5. メンテナンス・ユーザ・ログイン名を作成し、メンテナンス・ユーザがデータベースにログインできることを確認します。
6. Replication Server からデータベースへのコネクションを作成して、Replication Server がデータベースを管理できるようにします。

データベースにプライマリ・データがある場合、Sybase Central または **rs_init** は次の処理を行います。

- Adaptive Server で RepAgent を有効にする。
- データベースで RepAgent を有効にして設定する。
- Adaptive Server データベースのセカンダリ・トランケーション・ポイントを “valid” に設定して、RepAgent が読み込みを完了するまで Adaptive Server がデータベース・ログ・レコードをトランケートできないようにする。
- 必要に応じて、Replication Server に RepAgent のユーザ名とパスワードを作成する。
- RepAgent を起動する。

各手順の詳細については、使用しているプラットフォームの『Replication Server インストール・ガイド』と『Replication Server 設定ガイド』を参照してください。

参照：

- メンテナンス・ユーザの管理 (192 ページ)

ASE 以外のサーバの複写準備

Replication Server 15.2 では、コネクション・プロファイルを使用すると ASE 以外のサーバに単純化されたインストールと設定で接続できます。

ASE 以外のデータベースの複写準備については、『Replication Server 異機種間複写ガイド』を参照してください。

Replication Server 15.2 では、異機種 (ASE 以外) のデータ型がサポートされているため、データ型定義、ファンクション文字列、クラス・レベル変換をインストールするためのスクリプトを編集および実行する必要はありません。スクリプトによって提供されるファンクションは、Replication Server のインストールの一部として、または Replication Server とともにインストールされるコネクション・プロファイルに含まれています。これらの機能強化により、異機種環境および ASE 以外の環境のインストールと設定が簡略化されます。詳細については、『Replication Server 設定ガイド』の「ASE 以外のサポート機能のインストールと実装」を参照してください。

また、Replication Server 15.2 以降では、ASE 以外のレプリケート・データベース用のエラー・クラスがサポートされます。『Replication Server 管理ガイド 第 2 巻』の「エラーと例外の処理」の「データ・サーバのエラー処理」を参照してください。

接続プロファイル

接続プロファイルには、アクティブにサポートされている ASE 以外の各種データ・サーバに関連する、コネクションの設定とレプリケート・データベース・オブジェクトの定義が含まれます。

これらの接続プロファイルと単純な構文とを一緒に使用すると、Adaptive Server Enterprise、IBM DB2、Microsoft SQL Server、Oracle など、アクティブにサポートされているデータ・サーバ間のコネクションを作成できます。Replication Server は、接続プロファイルを使用してコネクションを設定し、レプリケート・データベース・オブジェクトを作成します。

接続プロファイルにより、インストールするファンクション文字列クラス、エラー・クラス、クラス・レベル変換を指定します。また、接続プロファイルのオプションを使用して、コマンドをバッチ処理するかどうかなどのその他のアクションや、使用するコマンド・セパレータを指定できます。

注意： 接続プロファイルを使用してコネクションを作成するときに、システム・テーブル・サービス (STS: System Table Services) キャッシュがリフレッシュされるため、Replication Server を再起動する必要はありません。

『Replication Server 管理ガイド 第 2 巻』の「データベース・オペレーションのカスタマイズ」の「Manage Function Strings」の「ASE 以外のサーバのコマンドのバッチ処理」を参照してください。

参照：
- データベース・コネクションの作成 (194 ページ)

既存の Adaptive Server データベースのアップグレード

新機能を使用するには、Replication Server の最新バージョンで動作するようにデータベースをアップグレードすることが必要な場合があります。データベースをアップグレードするには、**rs_init** を使用します。

データベースをアップグレードすると、複写ロールと、データベースで必要なパーミッション (**update**、**insert**、**delete**) が、データベース・メンテナンス・ユーザに与えられます。複写ロールは、複写に関して必要な Adaptive Server コマンドを実行するための権限をメンテナンス・ユーザに付与します。

データベースに付与されている権限を確認するには、そのデータベースで **sp_displaylogin** システム・プロシージャを使用します。

メンテナンス・ユーザへの複写ロールの付与

メンテナンス・ユーザに複写ロールを付与するには、**sp_role** を使用します。
データベースで以下を実行します。

```
sp_role "grant", replication_role, maintenance_user
```

テーブルに対するパーミッションの付与

データベース内のテーブルにパーミッションを付与するには、**grant all** を使用します。
各テーブルに対して、データベース内で以下を実行します。

```
grant all on table_name to maintenance_user
```

メンテナンス・ユーザの管理

複写データを更新する場合、Replication Server はデータ・サーバにメンテナンス・ユーザとしてログインします。データベース所有者 (またはシステム管理者) は、複写テーブルに対してローを挿入、削除、更新したり、複写ストアド・プロシージャを実行したりするために必要なパーミッションをメンテナンス・ユーザに付与する必要があります。

Sybase Central または **rs_init** は、はじめにメンテナンス・ユーザ用のログイン名を作成して、そのユーザをレプリケート・データベースに追加します。詳細については、使用しているプラットフォームの『Replication Server インストール・ガイド』と『Replication Server 設定ガイド』を参照してください。

メンテナンス・ユーザのログイン名とパスワードは、データベースに対して **create connection** コマンドを実行すると Replication Server に与えられます。なお、

Sybase Central または **rs_init** プログラムは、このコマンドを自動的に実行します。データ・サーバのログイン名のパスワードを変更する場合は、Sybase Central または **alter connection** コマンドを使用して、Replication Server コネクションのパスワードを変更します。

rs_init ユーティリティは、ASE 以外のデータベースには使用できません。ASE 以外のデータベース・サーバでは、メンテナンス・ユーザ・ログイン名を作成し、管理する必要があります。ASE 以外のサーバに関する情報については、『Replication Server 異機種間複写ガイド』を参照してください。

現在のメンテナンス・ユーザの調査

データベースに対するメンテナンス・ユーザとして現在割り当てられているログイン名を調べるには、**rs_helpuser** を使用します。

RSSD で Adaptive Server ストアド・プロシージャ **rs_helpuser** を入力します。ここで *user* は、情報を表示するログイン名です。

```
rs_helpuser [user]
```

データベース内でのパーミッションの付与

`rs_lastcommit` システム・テーブルと、それを使用するストアド・プロシージャにアクセスするためのパーミッションをメンテナンス・ユーザに付与するには、Sybase Central または **rs_init** を使用します。複写に関係するテーブルとストアド・プロシージャに対するパーミッションを付与するには、**grant all** を使用します。

ユーザ・テーブルとストアド・プロシージャについては、Sybase Central と **rs_init** はどちらもパーミッションをメンテナンス・ユーザに付与しません。複写テーブルに対するトランザクションの複写または複写ストアド・プロシージャの実行の複写を行うには、事前に複写テーブルとストアド・プロシージャにパーミッションを付与しておく必要があります。

データベース内で複写される各テーブルと、複写の結果実行される各ストアド・プロシージャに対して、次の **grant** コマンドを実行します。

```
grant all on table_name to maint_user
```

注意： メンテナンス・ユーザに付与されるパーミッションの 1 つに **replication_role** があります。このパーミッションを取り消すと、**truncate table** を複写することはできません。ただし、メンテナンス・ユーザが **sa_role** を付与されているか、そのテーブルを所有しているか、またはデータベース所有者としてエイリアスが定義されている場合を除きます。

ASE 以外のレプリケート・データベースのメンテナンス・ユーザに付与されるパーミッションについては、『Replication Server 異機種間複写ガイド』を参照してください。

プライマリ・データベースへのパーミッションの付与

レプリケート・データベースがプライマリ・データを保持している場合、そのデータベースは同時にプライマリ・データベースでもあります。プライマリ・データベースでは、サブスクリプションと要求ファンクションという 2 つの複写オブジェクトについて、特別なパーミッションが必要です。

サブスクリプションが作成されると、**rs_marker** ストアド・プロシージャがプライマリ・データベースで実行されます。サブスクリプションを作成できるすべてのデータベース・ユーザは、**rs_marker** を実行するためのパーミッションを持っている必要があります。

プライマリ・データベースは、レプリケート・サイトにあるクライアントから要求ファンクションの配信によってトランザクションを受信することもあります。これらのトランザクションは、要求ファンクションを実行したユーザによって実行されたかのように、プライマリ・サイトで実行されます。要求ファンクションを実行するパーミッションを持つユーザ・ログイン名はどれも、**rs_update_lastcommit** を実行するパーミッションも持つ必要があります。このコマンドは、各 DSI トランザクションで実行されます。

パーミッション要件は、要求ファンクションおよび要求ストアド・プロシージャの場合と同じです。

次の **grant** コマンドによって、データベース内のどのユーザでも **rs_marker** と **rs_update_lastcommit** を実行できるようになります。

```
grant execute on rs_marker to public
grant execute on rs_update_lastcommit to public
```

上記のストアド・プロシージャは、ユーザの代理である Replication Server だけが実行するようにしてください。Sybase Central または **rs_init** は、これらのパーミッションを “public” に付与します。サブスクリプションの作成、要求ファンクションの実行、またはストアド・プロシージャの要求を許可されているデータベース・ユーザだけに、パーミッションを制限することもできます。

参照：

- 複写ファンクションの管理 (385 ページ)

データベース・コネクションの作成

コネクションは、Replication Server に対してデータベースを定義します。Replication Server は、データベースを管理するように指定されています。それがレ

プリケート・データベースである場合は、データベースにトランザクションを分配するように指定されています。

データベース・コネクションは Replication Server に次の情報を提供します。

- コネクションの対象となるデータ・サーバとデータベースの名前
- データ・サーバから返されるエラーを処理するために使用するエラー・クラス
- データベースで使用するファンクション文字列クラス
- メンテナンス・ユーザのログイン名とパスワード
- データベース・コネクションに対する RepAgent スレッドがあるかどうかを示す情報
- ウォーム・スタンバイ・アプリケーション用にアクティブ・データベースとスタンバイ・データベースを作成するためのオプション
- コネクションに影響する設定パラメータ

使用方法：

- Adaptive Server データベースとの標準のコネクションを作成する場合は、Sybase Central、**rs_init**、または **create connection** を使用します。
- Adaptive Server データベースとの代替コネクションを作成するには、**create alternate connection** を使用します。たとえば、代替コネクションを作成して、複数の複写パスを構築できます。
  - 『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**create alternate connection**」を参照してください。
  - 『Replication Server 管理ガイド 第 2 巻』の「パフォーマンス・チューニング」の「Multi-Path Replication」を参照してください。

  を参照してください。
- Sybase 以外のデータベースに対するコネクションを作成するには、**using profile** 句を用いた **create connection** を使用します。『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**create connection using profile**」を参照してください。

データベース・コネクションを追加するための必要情報

データベース・コネクションを追加するときには、いくつかの項目を指定する必要があります。

rs_init を使用してデータベースを追加する方法については、使用しているプラットフォームの『Replication Server インストール・ガイド』と『Replication Server 設定ガイド』を参照してください。

データベースを追加するときは、次の情報を指定します。

- Replication Server 名

- Replication Server システム管理者のユーザ名とパスワード
- Adaptive Server 名
- Adaptive Server システム管理者のユーザ名とパスワード
- データベース名
- データベースに RepAgent が必要かどうか
- メンテナンス・ユーザのユーザ名とパスワード
- データベース所有者のユーザ名とパスワード
- 物理コネクションが既存の論理コネクション用かどうか

rs_init ユーティリティは、ASE 以外のデータベースには使用できません。ASE 以外のデータベース・サーバでは、メンテナンス・ユーザ・ログイン名を作成し、管理する必要があります。接続プロファイルを使用して、ASE 以外のデータベースへのコネクションを作成できます。

ASE 以外のデータベースへのアクセスの設定については、『Replication Server 異機種間複写ガイド』と Replication Server Options のマニュアルを参照してください。

参照：
- 接続プロファイル (191 ページ)

論理コネクションへのデータベースの追加

Sybase Central または **create logical connection** コマンドで作成した既存の論理コネクションに、物理コネクションを追加する場合、追加の情報を指定する必要があります。

指定内容

- アクティブ・コネクションまたはスタンバイ・コネクション
- 論理データ・サーバの名前
- 論理データベースの名前

さらに、スタンバイ・コネクションを追加する場合は、Sybase Central または **rs_init** で次の情報を指定します。

- アクティブ・データ・サーバの名前
- アクティブ・データベースの名前
- アクティブ・データベースのシステム管理者のユーザ名とパスワード
- ダンプ・メソッドとロード・メソッドを使用して、スタンバイ・データベースを初期化するかどうか
- 複写を開始するためにダンプ・マーカを使用するかどうか

『Replication Server 管理ガイド 第 2 巻』の「ウォーム・スタンバイ・アプリケーションの管理」を参照してください。

RepAgent スレッドを必要とするデータベースの追加

RepAgent を必要とする Adaptive Server プライマリ・データベースを追加する場合、Replication Server のユーザ名とパスワードを指定する必要があります。

create connection コマンドの使用方法

ASE 以外のデータ・サーバ用にデータベースを追加するには、**create connection** コマンドを、使用する ASE 以外のデータ・サーバに固有の接続プロファイルとともに使用します。Adaptive Server データ・サーバ用にデータベースを追加するには、Sybase Central または **rs_init** を使用します。これらはどちらも、データベースを複写用に準備します。

create connection を Adaptive Server データ・サーバに対して使用することもできます。

create connection を使用する場合は、手動でデータベースを複写用に準備します。

データベースを管理する Replication Server で **create connection** を入力します。構文は次のとおりです。

```
create connection to data_server.database
    set error class [to] error_class
    set function string class [to] function_class
    set username [to] user
    [set password [to] passwd]
    using profile connection_profile;version
    [set database_param [to] 'value']
    [set security_param [to] {'required' | 'not_required'}]
    [with {log transfer on, dsi_suspended}]
    [as active for logical_ds.logical_db |
    as standby for logical_ds.logical_db
    [use dump marker]
    [display_only]
```

レプリケート・データベースになる予定のないデータベースへのコネクションを作成する場合は、**with dsi_suspended** 句を使用します。この句は、DSI がサスペンドされている状態でコネクションを起動します。

as active、**as standby**、**use dump marker** 句は、ウォーム・スタンバイ・データベースの論理コネクションに対して物理コネクションを作成するときにだけ使用します。ウォーム・スタンバイ・アプリケーションで使用できるのは、Adaptive Server と Oracle のデータベースだけです。

使用しているシステムがネットワークベース・セキュリティをサポートしている場合は、**set** *security_param* コマンドを使用します。

参照：

- データベースの複写準備 (189 ページ)

- ネットワークベース・セキュリティの管理 (262 ページ)

using profile 句

ASE 以外のデータベース用の Replication Server とともにインストールされている接続プロファイルのいずれかを指定するには、**using profile** 句を指定して **create connection** を使用します。

using profile 句では、指定した接続プロファイルであらかじめ定義された情報を使用して、Replication Server および Adaptive Server 以外のデータベース間のコネクションを設定し、必要に応じて RSSD および指定した *data_server.database* を修正します。

接続プロファイルによって、ファンクション文字列クラス、エラー・クラス、クラス・レベル変換が指定されるため、ASE 以外のデータベースに対する **create connection** コマンドで対応する句を指定する必要はありません。特定のバージョンの接続プロファイルを使用するように指定するには、*version* を使用します。

たとえば、Oracle のレプリケート・データベースへのコネクションを、**rs_ase_to_oracle** 接続プロファイルを使用して作成するには、次のようにします。

```
create connection to oracle_db.ORACLE_DS
using profile rs_ase_to_oracle;eco
set username to ora_maint
set password to ora_maint_pwd
```

Replication Server で定義されている各プロファイルの接続プロファイル名、バージョン、コメントをリストするには **admin show_connection_profiles** コマンドを使用し、オプションで指定する文字列が名前に含まれる接続プロファイルのみを表示するには *match_string* オプションを使用します。

```
admin show_connection_profiles[, "match_string"]
```

『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**create connection using profile**」を参照してください。

『Replication Server 異機種間複写ガイド』の「Heterogeneous Warm Standby for Oracle」の「ウォーム・スタンバイ・データベースの設定」の「Creating Connection to the Standby Database」を参照してください。

データベース・コネクションの変更

接続が作成されている Replication Server でのデータベース・コネクションの属性を変更するには、Sybase Central または **alter connection** を使用します。

alter connection を使用する場合、変更を反映させるには、コネクションをサスペンドして再開する必要があります。

1. **suspend connection** を使用して、コネクションに関するアクティビティをサスペンドします。
2. **alter connection** コマンドは、関連するパラメータを付けて実行します。

   注意： **alter connection** コマンドに **set log transfer off** 句を指定すると、プライマリ・サイトから RepAgent コネクションが削除されます。この句を使用する前に、データベース内のデータに対して定義されている複写定義が存在しないことを確認してください。
3. **resume connection** を使用して、コネクションに関するアクティビティを再開します。

参照：
- データベース・コネクションのサスペンド (199 ページ)
- 物理コネクションに影響するパラメータの設定と変更 (199 ページ)
- データベース・コネクションの再開 (225 ページ)

データベース・コネクションのサスペンド

データベース・コネクションをサスペンドする必要があるのは、データベース・コネクションを変更するか、またはメンテナンスのためにデータ・サーバをサービスから削除する場合です。

sa パーミッションを持っている場合は、データ・サーバへのアクセスを一時的にサスペンドできます。

データ・サーバのアクセスがサスペンドされている間は、Replication Server はそのデータ・サーバに対するトランザクションをキューに入れます。キュー内のトランザクションは、コネクションがレジュームされると適用されます。

Sybase Central または次のコマンドを使用すると、データ・サーバへのアクセスを一時的にサスペンドできます。

```
suspend connection to data_server.database
    [with nowait]
```

デフォルトでは、**suspend connection** は、現在のトランザクションを完了してからサスペンドします。トランザクションの途中でコネクションをサスペンドするには、**with nowait** 句を使用します。これは、ラージ・トランザクションがレプリケート・データベース内での障害の原因となる場合に適しています。

物理コネクションに影響するパラメータの設定と変更

コネクションの設定パラメータは、コネクションの作成時に設定します。これらの設定パラメータは、Sybase Central または **alter connection** コマンドを使用してあとから更新できます。

設定は、1 つのデータベース・コネクションの設定を変更することも、1 つの Replication Server を始点とするデータベース・コネクションすべての設定を変更す

ることもできます。Replication Server に多数のデータベース・コネクションを追加する場合は、サーバのパフォーマンスを微調整するために、コネクションすべてに影響する設定パラメータを変更することもできます。

現在の Replication Server を始点とするコネクションすべての設定パラメータを変更するには、**configure replication server** コマンドを使用します。

alter connection を使用してそれぞれのコネクションに設定される設定パラメータは、**configure replication server** で設定されるパラメータを上書きします。したがって、**configure replication server** でデフォルト・パラメータを設定してから、**alter connection** で特定のコネクションの設定をカスタマイズできます。

Replication Server には、データベース・コネクションに影響する、さまざまなタイプの設定パラメータが用意されています。以下に対する設定パラメータがあります。

- 物理データベース・コネクション。
- 論理データベース・コネクション。『Replication Server 管理ガイド 第 2 巻』の「ウォーム・スタンバイ・アプリケーションの管理」の「Alter Warm Standby Database Connections」の「Alter Logical Connections」の「Change Parameters Affecting Logical Connections」を参照してください。
- ネットワークベース・セキュリティ。
- 並列 DSI 接続の設定とチューニング。『Replication Server 管理ガイド 第 2 巻』の「パフォーマンス・チューニング」の「Use Parallel DSI Threads」を参照してください。
- Replication Server のパフォーマンスのチューニング。『Replication Server 管理ガイド 第 2 巻』の「パフォーマンス・チューニング」の「パフォーマンスに影響する設定パラメータ」の「Connection Parameters that Affect Performance」を参照してください。

参照：
- すべてのコネクションに影響するパラメータの変更 (202 ページ)
- ネットワークベース・セキュリティの管理 (262 ページ)

1 つのコネクションに影響するパラメータの変更

コネクションを作成したあとは、**alter connection** コマンドを使用して、その設定パラメータを変更できます。

alter connection を使用すると、データベース・コネクションの属性を変更できます。たとえば、Sybase Central または **rs_init** を使用して Adaptive Server データベース・コネクションを追加したあとで、システム提供クラスではなく派生ファンクション文字列クラスをデータベース・コネクションで使用することにした場合に、このコマンドを使用します。**alter connection** の構文は次のとおりです。

```
alter connection to data_server.database {
    set function string class [to] function_class |
    set error class [to] error_class |
    set password [to] passwd |
    set log transfer [to] {on | off} |
    set database_param [to] 'value'} |
    set security_param to {'required' | 'not_required'} |
    set security_services [to] "default'
}
```

Replication Server に接続されているデータ・サーバとデータベースを指定して、変更する属性を 1 つまたは複数指定します。属性は次のとおりです。

function_class ― データベース・コネクションで使用するファンクション文字列クラスです。

error_class ― データベース・エラーの処理に使用するエラー・クラスです。

passwd ― データベース・コネクションのログイン名に使用する新しいパスワードです。

log transfer on ― このコネクションを使用して、トランザクションが Replication Server に送信されるようにします。

log transfer off ― このコネクションを使用した、プライマリ・データベースから Replication Server へのトランザクションの送信を停止します。

database_param ― コネクションに影響する設定パラメータを更新します。

security_param ― コネクションに影響するネットワーク・セキュリティ設定パラメータを更新します。

set security_services [to] 'default' ― このコネクション用のネットワークベース・セキュリティ機能をすべて "not required" にリセットします。

alter connection の使用例

SYDNEY_DS データ・サーバ内の `pubs2` データベースのファンクション文字列クラスを **sqlserver_derived_class** に変更するには、SYDNEY_RS Replication Server で次のコマンドを入力します。

```
suspend connection to SYDNEY_DS.pubs2
alter connection to SYDNEY_DS.pubs2
    set function string to class
      sqlserver_derived_class
resume connection to SYDNEY_DS.pubs2
```

alter connection コマンドのキーワードとオプションの詳細については、『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」を参照してください。

参照：
- ネットワークベース・セキュリティの管理 (262 ページ)

すべてのコネクションに影響するパラメータの変更

送信元 Replication Server を始点とするすべてのコネクションに対してデフォルト設定パラメータを設定するには、**configure replication server** コマンドを使用します。

configure replication server の構文は次のとおりです。

```
configure replication server
 set database_param to 'value'
```

設定の変更は、コネクションをレジュームしたあとに有効になります。

dsi_fadeout_time 値の変更

configure replication server を使用して、**dsi_fadeout_time** パラメータを DSI コネクションがクローズしないように変更する例を示します。送信元 Replication Server にログインして、以下のコマンドを実行します。

1. 送信元 Replication Server からのコネクションをすべてサスペンドします。各コネクションについて、次のように入力します。
   ```
   suspend connection to data_server.database
   ```
2. **configure replication server** を実行します。次のように入力します。
   ```
   configure replication server
   set dsi_fadeout_time to '-1'
   ```
3. サスペンドされている送信元 Replication Server からのコネクションをすべてレジュームします。各コネクションについて、次のように入力します。
   ```
   resume connection to data_server.database
   ```

Sybase フェールオーバのサポートの有効化

次の例では、**configure Replication Server** を使用して、**ha_failover** パラメータを変更し、Replication Server から Adaptive Server への RSSD 以外のすべてのコネクションに対するフェールオーバのサポートを有効にします。

1. **configure replication server** を実行します。フェールオーバ・サポートを有効にする Replication Server にログインし、次のように入力します。
   ```
   configure replication server
   set ha_failover to 'on'
   ```

   『Replication Server 管理ガイド 第 2 巻』の「複写システム・リカバリ」の「Sybase フェールオーバをサポートするための複写システムの設定」を参照してください。

物理データベース・コネクションに影響する設定パラメータ

1 つのコネクションの属性を変更するには、パラメータを付けて **alter connection** を使用します。Replication Server へのすべてのコネクションのパラメータ属性を変更するには **configure replication server** を使用します。

表 10 : データベース・コネクションに影響する設定パラメータ

パラメータ (database_param)	*値 (value)*
batch	デフォルトの “on” の場合は、レプリケート・データベースに対してコマンド・バッチを使用できる。 デフォルト値は ASE の場合は “on”、ASE 以外のデータベースの場合は “off”。 『Replication Server 管理ガイド 第 2 巻』の「データベース・オペレーションのカスタマイズ」の「Manage Function Strings」の「ASE 以外のサーバのコマンドのバッチ処理」を参照。
batch_begin	**begin transaction** を他のコマンド (**insert** や **delete** など) と同じバッチで送信できるかどうかを示す。 デフォルト値は on 『Replication Server 管理ガイド 第 2 巻』の「データベース・オペレーションのカスタマイズ」の「Manage Function Strings」の「ASE 以外のサーバのコマンドのバッチ処理」を参照。
command_retry	失敗したトランザクションをリトライする回数。0 以上の値を指定する。 デフォルト値は 3
db_packet_size	ネットワーク・パケットの最大サイズ。データベースと通信する場合、ネットワーク・パケットの値はそのデータベースで受け入れられる範囲内にする必要がある。Adaptive Server を再設定して使用する場合は、この値を変更できる。 最大値：16,384 バイト デフォルト値はすべての Adaptive Server データベースに対して、512 バイトのネットワーク・パケット

パラメータ *(database_param)*	値 *(value)*
deferred_name_resolution	Replication Server で遅延名前解決を有効にし、Adaptive Server での遅延名前解決をサポートする。 Replication Server で遅延名前解決のサポートを有効にする前に、レプリケート Adaptive Server で遅延名前解決がサポートされていることを確認する。 デフォルト値は off
disk_affinity	次のパーティションを割り当てるための割り付けヒントを指定する。現在のパーティションが満杯になった場合に、次のセグメントの割り付け先となるパーティションの論理名を入力する。指定できる値は “*partition_name*” と “off”。 デフォルト値は off
dsi_alt_writetext	レプリケート・データベースにラージ・オブジェクトの更新を送信する方法を制御する。値は、次のとおり。 • **dcany** – プライマリ・キー・カラムを含む **writetext** コマンドを生成する。この設定により、インタフェースとして DirectConnect Anywhere™ を使用して ASE 以外のレプリケート・データベースにデータを読み込む際に、フル・テーブル・スキャンが行われなくなる。 • **off** (デフォルト) – テキスト・ポインタを含む Adaptive Server の **writetext** コマンドを生成する。
dsi_bulk_copy	コネクションのバルク・コピー・イン機能を on または off にする。**dynamic_sql** と **dsi_bulk_copy** の両方が on の場合、Replication Server は必要に応じてバルク・コピー・インを適用し、Replication Server がバルク・コピー・インを使用できない場合は、動的 SQL を使用する。 デフォルト値は off

パラメータ *(database_param)*	値 *(value)*
dsi_bulk_threshold	トランザクション内の連続する **insert** コマンドの数。この数に到達すると、バルク・コピー・インを使用するように Replication Server をトリガする。ステーブル・キュー・トランザクション (SQT) は、大量の **insert** コマンドのバッチを検出すると、バルク・コピー・インを適用するかどうかを決定するために、指定された数の **insert** コマンドをメモリに保持する。これらのコマンドはメモリに保持されるため、この値を **dsi_large_xact_size** の設定値よりも大きい値に設定しないことをおすすめする。 Replication Server は、Sybase IQ への Real-Time Loading (RTL) Replication と Adaptive Server への High Volume Adaptive Replication (HVAR) に、**dsi_bulk_threshold** を使用する。1 つのテーブルに対する insert、delete、または update の各オペレーションに対するコマンドの数が、コンパイル後に指定する数よりも小さい場合、RTL と HVAR はバルク・インタフェースを使用せず、代わりに言語を使用する。 最小値：1 デフォルト値は 20 **注意：** **dsi_bulk_threshold** を使用するには、**dsi_compile_enable** を 'on' に設定する必要がある。
dsi_charset_convert	文字セット変換方法の指定。このパラメータは、該当の DSI で適用されるすべてのデータと識別子に適用される。値は、次のとおり。 • "on" (デフォルト) – プライマリ Replication Server の文字セットからレプリケート Replication Server 文字セットに変換する。文字セットに互換性がない場合は、DSI をエラーで停止する。 • "allow" – 文字セットに互換性がある場合に変換する。変換されていない更新も、すべてデータベースに適用する。 • "off" – 変換を行わない。このオプションは、異なるが互換性のある文字セットがあるときに、変換を一切実行しない場合に役立つ。サブスクリプション・マテリアライゼーションでは、"off" に設定しても、"allow" を設定した場合と同様に動作する。

パラメータ *(database_param)*	値 *(value)*
dsi_check_lock_wait	DSI エグゼキュータ・スレッドが **rs_thread_check_lock** ファンクション文字列を実行するまでの時間 (ミリ秒単位)。このファンクションは、ロック・ステータスについてレプリケート・データベースに問い合わせる。 デフォルト値は 3,000 ミリ秒 (3 秒)
dsi_cmd_batch_size	Replication Server が、コマンド・バッチ内に配置する最大バイト数。 デフォルト値は 8,192 バイト
dsi_cmd_prefetch	データ・サーバからの応答の待機中に、DSI がコマンドの次のバッチを事前構築することを許す。このため、DSI の効率が向上する。データ・サーバのパフォーマンスを高めるように調整する場合、この機能を使用するとパフォーマンスがさらに向上する可能性がある。 デフォルト値は off **dsi_compile_enable** を 'on' に設定すると、**dsi_cmd_prefetch** に設定した値は無視される。

dsi_cmd_separator	コマンド・バッチ内でコマンドを区切るために使用する文字。たとえば、異なるセパレータ文字を指定したあとでデフォルトの文字に戻すときは、次のように入力する。
	`alter connection to data_server.database`
	`set dsi_cmd_separator to '<Return>'`
	このとき、2 つの一重引用符で囲まれた部分では [Return] キーを押し、文字は入力しない。
	デフォルト値は改行文字 (¥n)
	注意： [Return] キーは対話型の更新においてのみ有効であり、DDL 生成スクリプトなどのスクリプトの実行では適用されません。このパラメータは、対話型モードで更新する必要があります。スクリプト内から再設定することはできません。

パラメータ *(database_param)*	値 *(value)*
dsi_command_convert	replicate コマンドの変換方法を指定する。 変換の種類は次のオペレーションの組み合わせによって指定される。 • **d** – delete • **i** – insert • **u** – update • **t** - truncate • **none** – オペレーションなし **dsi_command_convert** に対するオペレーションの組み合わせには、**i2none**、**u2none**、**d2none**、**i2di**、**t2none**、**u2di** がある。 数字の “2” を入力する必要がある。変換前のオペレーションは "2" の前に、変換後のオペレーションは "2" の後ろにある。次に例を示す。 • **d2none** – delete コマンドを複写しない。 • **i2di**,**u2di** – insert と update の両方を、delete とその後の insert に変換する。これはオートコレクションと同等のオペレーション。 • **t2none** – **truncate table** コマンドを複写しない。 デフォルト値はなし。 このパラメータはテーブルレベルで設定することもできる。 設定では、データベースレベルでは **alter connection** を使用し、テーブルレベルの設定では **for replicate table named** 句を指定して **alter connection** を使用する。
dsi_commit_check_locks_intrvl	DSI エグゼキュータ・スレッドが **rs_dsi_check_thread_lock** ファンクション文字列の実行と実行の間に待機するミリ秒 (ms) 数。並列 DSI とともに使用される。『Replication Server 管理ガイド第 2 巻』の「パフォーマンス・チューニング」の「Use Parallel DSI Threads」を参照。 デフォルト値は 1,000 ミリ秒 (1 秒) 最小値：0 最大値：86,400,000 ミリ秒 (24 時間)

パラメータ *(database_param)*	値 *(value)*
dsi_commit_check_ locks_max	トランザクションをロールバックしてリトライする前に、DSI エグゼキュータ・スレッドが複写データベース内の他のトランザクションをブロックしているどうかをチェックする最大回数。並列 DSI とともに使用される。『Replication Server 管理ガイド 第 2 巻』の「パフォーマンス・チューニング」の「Use Parallel DSI Threads」を参照。 デフォルト値は 400 最小値：1 最大値：1,000,000
dsi_commit_control	コミット制御について、Replication Server が内部テーブルを使用して内部的に処理するか (on)、`rs_threads` システム・テーブルを使用して外部的に処理するか (off) を指定する。並列 DSI とともに使用される。『Replication Server 管理ガイド 第 2 巻』の「パフォーマンス・チューニング」の「Use Parallel DSI Threads」を参照。 デフォルト値は on

パラメータ *(database_param)*	値 *(value)*
dsi_compile _enable	サーバレベル、データベースレベル、またはテーブルレベルで RTL または HVAR を有効または無効にする。 設定では、サーバレベルでは **configure replication server** を、データベースレベルでは **alter connection** を、テーブルレベルでは **for replicate table named** 句を指定した **alter connection** を使用する。 デフォルト値は • off – サーバレベルとデータベースレベル。Replication Server はログ順、ローごとの連続変更複写を使用する。 • on – テーブルレベル テーブル上のすべてのオペレーションをログ順に複写する必要があるトリガがテーブルにあるためコンパイルを使用できない場合のように、新しいロー変更を複写すると問題が発生する場合、問題のテーブルで **dsi_compile_enable** を 'off' に設定する。 **注意：** テーブルレベルで **dsi_compile_enable** を 'on' に設定する前に、サーバレベルまたはデータベースレベルで、**dsi_compile _enable** を 'on' に設定する。 **dsi_compile_enable** を 'on' に設定すると、**replicate_minimal_columns** と **dsi_cmd_prefetch** に設定した値は無視される。 サーバレベル、データベースレベル、またはテーブルレベルで **dsi_compile_enable** を実行した後、コネクションをサスペンドして再開する。 HVAR については、『Replication Server 管理ガイド 第 2 巻』の「パフォーマンス・チューニング」の「Advanced Services Option」の「High Volume Adaptive Replication to Adaptive Server」を参照。 RTL については、『Replication Server 異機種間複写ガイド』の「Sybase IQ as Replicate Data Server」を参照。 ライセンス：Advanced Services オプションで個別にライセンス供与される。『Replication Server 管理ガイド 第 2 巻』の「パフォーマンス・チューニング」の「Advanced Services Option」を参照。

パラメータ *(database_param)*	値 *(value)*
dsi_compile_max_cmds	トランザクションのグループの最大サイズを、コマンド数で指定する。HVAR または RTL がコンパイルしている現在のグループで最大グループ・サイズに達すると、HVAR または RTL は新しいグループを開始する。 読み込むデータがなくなると、グループが最大コマンド数に達していなくても、HVAR または RTL は現在のトランザクションのセットを現在のグループにグループ化する処理を終了する。 設定では、サーバレベルでは **configure replication server** を、データベースレベルでは **alter connection** を使用する。 最小値：100 デフォルト値は 10,000 **注意：** **dsi_compile_max_cmds** を使用するには、**dsi_compile_enable** を 'on' に設定する必要がある。
dsi_dataserver_make	接続するレプリケート・データベースを含むデータ・サーバ・タイプを指定する。 有効な値は次のとおりです。ASE、IQ、および ORA。 そのコネクションに関連付けるコネクタを識別するには、**dsi_dataserver_make** と **dsi_connector_type** を使用する。 Sybase IQ に複写するには IQ に設定する。Adaptive Server に複写するには ASE に設定し、Oracle に複写するには ORA に設定する。 **dsi_dataserver_make** をデータベース・レベルで設定できる。 このパラメータを指定しない場合、Replication Server は、データベース・コネクションのファンクション文字列のクラス名からデータ・サーバ・タイプを推測する。 ファンクション文字列のクラスがカスタマイズされている場合、Replication Server はデータ・サーバ・タイプを推測できないため、デフォルトを 'ASE' に設定する。
dsi_exec_request_sproc	プライマリ Replication Server の DSI で、要求ストアド・プロシージャをオンまたはオフにする。 デフォルト値は on
dsi_fadeout_time	DSI コネクションがクローズされるまでのアイドル時間 (秒単位)。-1 を指定すると、コネクションは永久にクローズしない。 デフォルト値は 600 秒

パラメータ *(database_param)*	値 *(value)*
dsi_ignore_underscore_name	トランザクション・パーティショニング・ルールが **name** に設定されている場合に、Replication Server がアンダースコアで始まるトランザクション名を無視するかどうかを指定する。値は "on" と "off"。 デフォルト値は on
dsi_isolation_level	トランザクションの独立性レベルを指定する。ANSI 標準および Adaptive Server でサポートされている値は、次のとおり。 • 0 ― 個々のトランザクションで書き込まれたデータが実際のデータであることを保証する。 • 1 ― ダーティ・リードを防止し、個々のトランザクションで書き込まれたデータが実際のデータであることを保証する。 • 2 – 繰り返し不可能読み出しとダーティ・リードを防止し、あるトランザクションで書き込まれたデータが実際のデータであることを保証する。 • 3 – 幻ロー、繰り返し不可能読み出し、ダーティ・リードを防止し、あるトランザクションで書き込まれたデータが実際のデータであることを保証する。 カスタム・ファンクション文字列を使用することで、Replication Server は、レプリケート・データ・サーバが使用できる任意の独立性レベルをサポートできる。サポートは、ANSI 標準のみに制限されない。 デフォルト値は、ターゲット・データ・サーバの現在のトランザクションの独立性レベル。
dsi_keep_triggers	データベース内の複写トランザクションに対してトリガを起動するかどうかを指定する。 “off” – Adaptive Sever データベースで Replication Server が **set triggers off** になるため、コネクション上でトランザクションが実行されても、トリガは起動しない。 この設定はスタンバイ・データベースに対して使用する。 “on” – スタンバイ・データベースを除くすべてのデータベースで指定する。 デフォルト値は on (スタンバイ・データベースを除く)

パラメータ *(database_param)*	値 *(value)*
dsi_large_xact_size	ラージ・トランザクションと見なされるまでに 1 つのトランザクション内で許可されるコマンドの数。 デフォルト値は 100 最小値：4 最大値：2,147,483,647 **dsi_compile_enable** が on の場合、このパラメータは無視される。
dsi_max_cmds_in_batch	出力コマンドのバッチ処理の対象にできるソース・コマンドの最大数を定義する。 パラメータの変更を反映させるには、コネクションをサスペンドして再開する必要がある。 範囲：1 – 1000 デフォルト値は 100
dsi_max_cmds_to_log	1 つのトランザクションで例外ログに書き込むコマンドの数。 デフォルト値は-1 (すべてのコマンド) 有効な値 0 から 2147483647
dsi_max_xacts_in_group	グループ化できるトランザクションの最大数を指定する。大きい値を指定するほど、レプリケート・データベースでのデータ遅延時間が短縮される。値の範囲：1 – 1000. デフォルト値は 20 **dsi_compile_enable** が on の場合、このパラメータは無視される。
dsi_max_text_to_log	失敗したトランザクション内の各 **rs_writetext** ファンクションの例外ログに書き込むバイト数。このパラメータを変更して、ラージ `text`、ラージ `unitext`、またはラージ `image` カラムの操作を伴うトランザクションによって RSSD またはそのログが満杯にならないようにする。 デフォルト値は-1 (すべての `text`、`unitext`、または `image` カラム)

パラメータ *(database_param)*	値 *(value)*
dsi_non_blocking_commit	コミット後に Replication Server がメッセージを保存している期間を延長する長さ (分単位) を指定する。値の範囲：0 ～ 60 分 デフォルト値は 0 – 非ブロッキング・コミットは無効。 Adaptive Server 15.0 以降で遅延コミット機能を使用できる場合、または Oracle 10g v2 で同等の機能を使用できる場合には、このパラメータを有効にすることで複写パフォーマンスが向上する。
dsi_num_large_xact_threads	ラージ・トランザクションで使用するために予約されている並列 DSI スレッドの数。最大値は、**dsi_num_threads** の値から 1 を引いた値。 デフォルト値は 0
dsi_num_threads	使用する並列 DSI スレッドの数。最大値は 255。 デフォルト値は 1
dsi_partitioning_rule	利用可能な並列 DSI スレッド間でトランザクションを分割するために DSI が使用する、1 つ以上のパーティショニング・ルールを指定する。指定できる値は **origin**、**origin_sessid**、**time**、**user**、**name**、**none** のいずれか。 詳細については、『Replication Server 管理ガイド 第 2 巻』の「パフォーマンス・チューニング」の「並列 DSI スレッドの使用」の「パーティショニング・ルール: 競合を減らして並列処理を増やす」を参照。 デフォルト値はなし **dsi_compile_enable** が on の場合、このパラメータは無視される。

パラメータ *(database_param)*	値 *(value)*
dsi_proc_as_rpc	Replication Server がストアド・プロシージャの複写を適用する方法を指定する。 • リモート・プロシージャ・コール (RPC) の呼び出しを使用するには、on に設定する。 • 言語呼び出しを使用するには、off に設定する。 デフォルト値は off Adaptive Server がレプリケート・データベースである場合は、**dsi_proc_as_rpc** を on または off にできる。 レプリケート・データベースが Oracle である場合は、次のように設定する。 • ExpressConnect for Oracle (ECO) を使用している場合は on に設定する。ECO では、RPC を使用したストアド・プロシージャの複写のみがサポートされている。Replication Server から Oracle データベースへのコネクションを作成するときに Oracle ECO 接続プロファイルのいずれかを使用する場合、Replication Server はデフォルトで **dsi_proc_as_rpc** を on に設定する。Replication Server Options 15.5 の『Installation and Configuration Guide ExpressConnect for Oracle 15.5』の「Configuring ExpressConnect for Oracle」を参照。 • ECDA Option for Oracle を使用している場合は off に設定する。ECDA では、RPC のストアド・プロシージャの複写はサポートされていない。

パラメータ *(database_param)*	値 *(value)*
dsi_quoted_identifiers	DSI の引用符付き識別子のサポートを有効または無効にする。値は **on** (有効) または **off** (無効)。 デフォルト値は off 引用符付き識別子とは、スペースや非英数字などの特殊文字が含まれる、英字以外の文字で始まる、または予約語に相当するオブジェクト名であり、正しく解析されるように二重引用符文字で囲む必要がある。このパラメータは、次の場合に有効にする。 **1.** データ・サーバに引用符付き識別子を転送できるコネクションを作成または修正する。 **create connection** コマンドまたは **alter connection** コマンドを使用して、**dsi_quoted_identifier** を “on” または “off” にする。 **dsi_quoted_identifier** の値を確認するには、**admin config** コマンドを使用する。 **2.** 複写定義で識別子を引用符付きとしてマーク付けする。 引用符付き識別子機能は、混合バージョン環境ではサポートされない。引用符付き識別子の複写を成功させるには、プライマリ Replication Server とレプリケート・データ・サーバに接続する Replication Server のバージョンを 15.2 にする。ただし、ルート上の中間 Replication Server は、古いバージョンでもかまわない。
dsi_replication	DSI によって適用されたトランザクションを、トランザクション・ログ内で複写対象としてマーク付けするかどうかを指定する。**dsi_replication** を “off” に設定すると、DSI は Adaptive Serve データベース内で **set replication off** を実行し、DSI が実行するトランザクションのログ・レコードに、Adaptive Server が複写情報を追加するのを防ぐ。これらのトランザクションは、メンテナンス・ユーザによって実行されるため、(スタンバイ・データベースがある場合を除いて) 通常は、これ以上複写されることはない。そのため、このパラメータを “off” に設定すると、不必要な情報がトランザクション・ログに書き込まれるのを防ぐことができる。 レプリケート・データベース用のウォーム・スタンバイ・アプリケーション内のアクティブなデータベースや複写統合レプリケート・アプリケーション・モデルを使用するアプリケーションの場合は、**dsi_replication** を “on” に設定する必要がある。 デフォルト値は on (ウォーム・スタンバイ・アプリケーション内のスタンバイ・データベースの場合は “off”)

パラメータ (database_param)	***値 (value)***
dsi_row_count_validation	同期されていないテーブル・ローがあり、デフォルトのエラー・アクションとメッセージをバイパスする場合、**dsi_row_count_validation** を **off** に設定してロー・カウントの検証を無効にできる。 デフォルト値は **on** で、ロー・カウントの検証を有効にする。 特定のコネクションに対して **dsi_row_count_validation** を設定した場合は、データベース・コネクションをサスペンドして再開する必要はない。パラメータはただちに有効になる。ただし、新しい設定は、このコマンドを実行した後で Replication Server が処理する複写オブジェクトのバッチに影響する。設定の変更は Replication Server が現在処理している複写オブジェクトのバッチには影響しない。

パラメータ *(database_param)*	値 *(value)*
dsi_serialization_method	一貫性を保ちながら、トランザクションを開始できるタイミングを決定するために使用するメソッドを指定する。どの場合でもコミット順は保持される。 これらのメソッドは並列処理量の多い順になる。並列処理が多いほど、レプリケート・データベースに適用されるときに並列トランザクション間の競合が増える可能性がある。競合を減らすには、**dsi_partition_rule** オプションを使用する。 • **no_wait** ― 他のトランザクションの状態に関係なく、トランザクションが準備でき次第すぐに開始できることを指定する。 • **wait_for_start** ― 開始直前にコミットするようにスケジュールされているトランザクションが開始した直後に、トランザクションを開始できることを指定する。 • **wait_for_start** ― 直前にコミットするようスケジュールされているトランザクションの準備が終了するまでは、トランザクションを開始できないよう指定する。 • **wait_after_commit** ― 直前にコミットするようにスケジュールされているトランザクションのコミットが完了するまで、トランザクションが待機することを指定する。 **注意：** **dsi_commit_control** を “on” に設定している場合は、 **dsi_serialization_method** は **no_wait** にのみ設定できる。 以下のオプションは、Replication Server の旧バージョンとの下位互換性のためにのみ維持されている。 • **none** ― **wait_for_start** と同じ。 • **single_transaction_per_origin** ― **dsi_partitioning_rule** が **origin** に設定された **wait_for_start** と同じ。 **注意：** **isolation_level_3** 値は逐次化メソッドとしてサポートされなくなったが、**dsi_serialization_method** を **wait_for_start** に設定し、**dsi_isolation_level** を 3 に設定した場合と同じである。 デフォルト値は **wait_for_commit**

パラメータ (database_param)	***値 (value)***
dsi_sqt_max_cache_size	データベース・コネクションの SQT (ステーブル・キュー・トランザクション・インタフェース) キャッシュ・メモリの最大量 (バイト単位)。 デフォルトの “0” は、コネクションの最大キャッシュ・サイズとして **sqt_max_cache_size** の現在の設定を使用することを示す。 デフォルト値は 0 32 ビット版 Replication Server の場合： • 最小値 – 0 • 最大値 – 2147483647 64 ビット版 Replication Server の場合： • 最小値 – 0 • 最大値 – 2251799813685247
dsi_stage_all_ops	Replication Server と Sybase IQ InfoPrimer の統合を設定する場合に、指定したテーブルがコンパイルされないようにする。 緩やかに変化する次元 (SCD) のテーブルなどのように、テーブル履歴を保存する必要がある場合、**dsi_stage_all_ops** を on に設定する。 『Replication Server 異機種間複写ガイド』の「Sybase IQ as Replicate Data Server」の「Replication Server と Sybase IQ InfoPrimer の統合」の「パラメータ」の **dsi_stage_all_ops** を参照。

パラメータ *(database_param)*	値 *(value)*
dsi_text_convert_multiplier	レプリケート・サイトでの `text` または `unitext` データ型カラムの長さを変更する。文字セット変換によって `text` または `unitext` データ型カラムが拡張または縮小される場合、**dsi_text_convert_multiplier** を使用する。Replication Server は、プライマリ・サイトの `text` または `unitext` データの長さに **dsi_text_convert_multiplier** の値を乗算することによって、レプリケート・サイトでの `text` または `unitext` データの長さを判別する。この値のデータ型は `float`。 • 文字セット変換によって `text` または `unitext` データ型カラムのサイズが拡張される場合は、**dsi_text_convert_multiplier** に 1.0 以上の値を指定する。 • 文字セット変換によって `text` または `unitext` データ型カラムのサイズが縮小される場合は、**dsi_text_convert_multiplier** に 1.0 以下の値を指定する。 デフォルト値は 1
dsi_timer	**dsi_timer** 設定パラメータを使用して、プライマリ・データベースでトランザクションがコミットされる時刻と、スタンバイ・データベースまたはレプリケート・データベースでトランザクションがコミットされる時刻との遅延を指定する。Replication Server は、この遅延期間が終了した後、アウトバウンド・キューのトランザクションをコミット順に処理する。 **dsi_timer** を **alter connection** または **alter logical connection** とともに実行した後、コネクションをサスペンドして再開する。 遅延を hh:mm のフォーマットで指定する。 • 最大値：24 hours • デフォルト値は 00:00 で、遅延がないことを意味する。 **注意：** Replication Server は、プライマリ・データベースでの Replication Agent と、**dsi_timer** を実行する DSI コネクションの Replication Server との時差をサポートしていない。

パラメータ *(database_param)*	値 *(value)*
dsi_xact_group_size	1 つにグループ化されたトランザクションに配置する最大バイト数 (ステーブル・キュー・オーバヘッドを含む)。グループ化されたトランザクションとは、DSI が単一のトランザクションとして適用するトランザクション・セットのことである。–1 はグループ化が行われないことを意味する。 **dsi_xact_group_size** を最大値に設定し、**dsi_max_xacts_in_group** でグループ内のトランザクション数を制御することを推奨する。 **注意：** このパラメータは、Replication Server バージョン 15.0 以降では使用されなくなっているが、旧バージョンの Replication Server との互換性を保つために保持されている。 最大値：2,147,483,647 デフォルト値は 65,536 バイト **dsi_compile_enable** が on の場合、このパラメータは無視される。
dump_load	コーディネート・ダンプを有効にする目的でのみ、レプリケート・サイトで “on” に設定する。『Replication Server 管理ガイド第 2 巻』の「複写システム・リカバリ」の「Recover from Primary Database Failures」の「コーディネート・ダンプからのロード」を参照。 デフォルト値は off
exec_cmds_per_time-slice	LTI または RepAgent エグゼキュータ・スレッドが、他のスレッドに CPU を解放しなければならなくなる前に処理できる LTL コマンドの数を指定する。値の範囲は 1 ～ 2,147,483,647。 デフォルト値は 2,147,483,647
dynamic_sql	動的 SQL 機能を有効または無効にする。このパラメータを “on” に設定した場合にのみ、動的 SQL 関連の他の設定パラメータが有効になる。 デフォルト値は off
dynamic_sql_cache_size	コネクションの動的 SQL 文を使用できるデータベース・オブジェクトの数についてのヒントを Replication Server に提供する。 最小値：1 最大値：65536 デフォルト値は 100

パラメータ *(database_param)*	値 *(value)*
dynamic_sql_cache_management	DSI/E スレッドの動的 SQL キャッシュを管理する。値：**mru** - **dynamic_sql_cache_size** に達すると、最後に使用された文を保持し、それ以外の文の割り付けを解除して新しい動的文を割り付ける。**fixed** (デフォルト) - **dynamic_sql_cache_size** に達すると、Replication Server は新しい動的文の割り付けを停止する。
exec_nrm_request_limit	正規化待機中のプライマリ・データベースからのメッセージ用に使用可能なメモリ量を指定する。 **configure replication server** で **nrm_thread** を 'on' に設定してから、**exec_nrm_request_limit** を使用する。 最小値：16,384 バイト 最大値：2,147,483,647 バイト デフォルト値： • 32 ビット版 – 1,048,576 バイト (1MB) • 64 ビット版 – 8,388,608 バイト (8MB) **exec_nrm_request_limit** の設定を変更した後、Replication Agent をサスペンドして再開する。 ライセンス：Advanced Services オプションで個別にライセンス供与される。『Replication Server 管理ガイド 第 2 巻』の「パフォーマンス・チューニング」の「Advanced Services Option」を参照。
exec_sqm_write_request_limit	インバウンド・キューへの書き込み待ちメッセージ用に使用可能なメモリ量を指定する。 デフォルト値は 1MB 最小値：16KB 最大値：2GB

パラメータ *(database_param)*	値 *(value)*
md_sqm_write_request_limit	アウトバウンド・キューへの書き込み待ちメッセージ用にディストリビュータが使用可能なメモリ量を指定する。 **注意：** Replication Server 12.1 の場合、**md_source_memory_pool** は **md_sqm_write_request_limit** で置換される。**md_source_memory_pool** は旧式の Replication Server との互換性のために保持される。 デフォルト値は 1MB 最小値：16K 最大値：2GB
rep_as_standby	**rep_as_standby** が **on** の場合、テーブル・サブスクリプションが **sp_reptostandby** によってマーク付けされたテーブルを複写する。 **rep_as_standby** を **on** にし、この機能を有効にするには、RepAgent パラメータの **send maint xacts to replicate** を **false** に、**send warm standby xacts** を **true** に設定する必要がある。 デフォルト値は off

パラメータ *(database_param)*	値 *(value)*
replicate_minimal_columns	Replication Server がすべてのトランザクションのすべての複写定義カラムを送信するか、レプリケート・データベースで更新オペレーションまたは削除オペレーションを実行する必要があるカラムだけを送信するかを指定する。 値は "on" と "off"。 デフォルト値は On 複写定義に **replicate minimal columns** 句が含まれない場合、または、複写定義がまったくない場合、Replication Server はこのコネクションレベルのパラメータを使用する。 それ以外の場合、Replication Server はこのパラメータの値を無視する。 **admin config** を使用して、**replicate_minimal_columns** 設定情報を表示できる。 **dsi_compile_enable** を on に設定すると、**replicate_minimal_columns** に設定した値は無視される。 『Replication Server 管理ガイド 第 2 巻』の「パフォーマンス・チューニング」の「動的 SQL で強化された Replication Server のパフォーマンス」の「Use Replicate Minimal Columns with Dynamic SQL」を参照。
save_interval	メッセージが送信先データ・サーバに正常に渡された後に、Replication Server がそのメッセージを保存しておく時間 (分単位)。 デフォルト値は 0 分
stage_operations	on に設定すると、Replication Server と Sybase IQ InfoPrimer の統合を設定する際に、Replication Server は指定したコネクションのステージング・テーブルにオペレーションを書き込む。 『Replication Server 異機種間複写ガイド』の「レプリケート・データ・サーバとしての Sybase IQ」の「Replication Server と Sybase IQ InfoPrimer の統合」の「パラメータ」の **stage_operations** を参照。

パラメータ *(database_param)*	値 *(value)*
sub_sqm_write_request_limit	アウトバウンド・キューへの書き込み待ちメッセージ用にサブスクリプション・マテリアライゼーション／マテリアライゼーション解除スレッドが使用可能なメモリ量を指定する。 デフォルト値は 1MB 最小値：16K 最大値：2GB
unicode_format	U&″ フォーマットの Unicode データの送信をサポートする。 **unicode_format** を次の値のいずれかに設定する。 • `string` – Unicode 文字を文字列形式に変換する。たとえば、文字列 "hello" は "hello" として送信される。 • `ase` – Unicode 文字を `U&''` フォーマットで送信する。たとえば、文字列 "hello" は "U&'¥0068¥0065¥006c¥006c¥006f' " として送信される。2 バイト Unicode 値は、Adaptive Server Enterprise が要求するネットワーク順序で送信される。 デフォルト値は `string`
use_batch_markers	**use_batch_markers** が **on** に設定されている場合、ファンクション文字列 **rs_batch_start** と **rs_batch_end** が実行される。 **注意：** **rs_begin** および **rs_commit** ファンクション文字列に含まれていないコマンドのバッチの開始時および終了時に、追加の SQL 変換を送信する必要があるレプリケート・データ・サーバの場合にのみ、このパラメータを **on** に設定する必要がある。 デフォルト値は off 『Replication Server 管理ガイド 第 2 巻』の「データベース・オペレーションのカスタマイズ」の「ファンクション文字列の管理」の「ASE 以外のサーバのコマンドのバッチ処理」を参照。

パフォーマンス向上のための Replication Server コネクション・パラメータの変更

設定パラメータを使用して、レプリケーション・パフォーマンスを向上させることができます。

設定パラメータのデフォルト値は、平均的な環境と使用状況に合わせて設定されています。使用しているシステム構成や各サイトでの Replication Server の使用状況によっては、一部のデフォルト値を慎重に変更することによって、パフォーマンスが向上する場合があります。パフォーマンスと設定パラメータの概要につい

ては、『Replication Server 管理ガイド 第 2 巻』の「パフォーマンス・チューニング」の「パフォーマンスに影響する設定パラメータ」を参照してください。『Replication Server 管理ガイド 第 2 巻』の「パフォーマンス・チューニング」の「Use Parallel DSI Threads」も参照してください。

多数の新しいコネクションを追加する場合は、パフォーマンスを向上させるために Replication Server パラメータの **memory_limit** または **num_threads** を変更することができます。

memory_limit パラメータと **num_threads** パラメータの詳細については、『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**configure replication server**」を参照してください。

memory_limit の値を大きくする

Replication Server に指定されているメモリ量を増やすには、Replication Server で **configure replication server** を使用して、**memory_limit** パラメータに指定された値を大きくします。

たとえば、**memory_limit** を 25MB に増やすには、**configure replication server** を次のように実行します。

```
configure replication server
    set memory_limit to '25'
```

num_threads の値を大きくする

Replication Server が使用できる Open Server スレッドの数を増やさなければならない場合があります。そのためには、Replication Server で **configure replication server** を使用して、**num_threads** パラメータに指定された値を大きくします。

たとえば、**num_threads** を 70 に増やすには、**configure replication server** を次のように実行します。

```
configure replication server
        set num_threads to '70'
```

データベース・コネクションの再開

データベース・コネクションの属性を変更した後は、Sybase Central または **resume connection** コマンドを使用して、コネクションに対するアクティビティを再開できます。

コマンド・ラインからデータベース・コネクションを再開するには、次のように入力します。

```
resume connection to data_server.database
[skip [n] transaction | execute transaction]
```

コネクションがレジュームされると、Replication Server は `rs_lastcommit` システム・テーブルからローを検索して、トランザクション・ストリーム内のどこからトランザクションの送信を開始すればよいかを判断します。

オプションの **skip [n] transaction** 句は、コネクション・キュー内の指定した数のトランザクションを省略してからコネクションをレジュームするように Replication Server に指示します。最初の *n* 個のトランザクションは例外ログに書き込まれます。

skip [n] transaction 句は、最初の *n* 個のトランザクションによって Replication Server がコネクションをサスペンドし、障害の原因を排除できない場合に必要になります。たとえば、トランザクションが **retry_stop** または **stop_replication** エラー・アクションを割り当てられているデータ・サーバ・エラーを発生させた可能性がある場合です。または、トランザクションを手動で中断するために **suspend connection** と **with nowait** 句を使用する必要があったにもかかわらず、それを行わなかった場合などです。

警告！ resume connection に **skip transaction** 句を指定して実行する場合は、トランザクションの消失が原因で発生する矛盾をすべて訂正してください。**skip transaction** 句は、トランザクション失敗の状況を訂正できない場合にだけ使用します。

オプションの **execute transaction** 句は、コネクションのキューに並んでいる最初のトランザクションを実行するように Replication Server に指示します。この句は、システム・トランザクションが実行に失敗した場合にだけ使用します。システム・トランザクションのエラー処理については、『Replication Server 管理ガイド第 2 巻』の「エラーと例外の処理」の「システム・トランザクションの重複検出」を参照してください。

レプリケート・データベースからプライマリ・データベースへの変更

各プライマリ・データベースには、データベース・ログをスキャンし、レプリケート・データベースへの分配のためにデータを Replication Server に転送する Replication Agent が必要です。複写専用として指定されている Adaptive Server データベースを、複写ファンクションの送信元またはプライマリ・データを格納するように変更する場合は、そのデータベース用に RepAgent スレッドを有効にする必要があります。

1. Replication Server での RepAgent の設定

   a) RepAgent が Replication Server にログインできるように、RepAgent ユーザを作成します。**create user** コマンドを使用します。ここで、*ra_user_name* は RepAgent ユーザの名前、*ra_password* は RepAgent のパスワードです。

```
create user ra_user_name
set password {ra_password | null}
```

b) **grant** コマンドを使用して、このユーザに **connect source** パーミッションを付与します。

```
grant connect source to ra_user_name
```

Replication Server がすでにプライマリ・データベースを管理している場合は、既存の「RepAgent ユーザ」を新しいプライマリ・データベース用に使用できます。

c) **log transfer on** オプションを付けて、**alter connection** コマンドを実行します。

```
alter connection to data_server.database
set log transfer to 'on'
```

2. Adaptive Server データベースでの RepAgent の設定

a) Adaptive Server の名前がまだ定義されていない場合は、次のコマンドを使用して名前を定義します。ここで、*lname* は Adaptive Server の名前です。

```
sp_addserver lname, local
```

b) Adaptive Server の RepAgent スレッドが有効になっていない場合は、次のようにして有効にします。

```
sp_configure 'enable rep agent threads'
```

c) **sp_config_rep_agent** を使用して、データベースの RepAgent を設定します。

```
sp_config_rep_agent dbname, 'enable', 'rs_name',
        'rs_user_name', 'rs_password'
```

注意： Adaptive Server で設定する *rs_user_name* と *rs_password* は、手順 1 において Replication Server で作成した *ra_user_name* および *ra_password* と同じ値にする必要があります。

d) **sp_setreplicate** システム・プロシージャを使用して **rs_marker** ストアド・プロシージャを作成し、そのレプリケート・ステータスを “true” に設定します。

rs_marker ストアド・プロシージャは、Sybase リリース・ディレクトリの `scripts` ディレクトリにあるファイル `rs_install_primary.sql` または `rsinssys.sql` にあります。

e) 次のコマンドを入力して RepAgent を起動します。

```
sp_start_rep_agent dbname
```

参照：

- RepAgent の設定 (118 ページ)

rs_marker ストアド・プロシージャの作成

rs_marker ストアド・プロシージャを作成して、複写ステータスをチェックします。

Replication Server は、サブスクリプション・マテリアライゼーション中に、プライマリ・データベース内で **rs_marker** システム・ファンクションを実行します。このファンクションは、同じ **rs_marker** という名前の複写ストアド・プロシージャを実行することによって動作します。このプロシージャは、複写するようマーク付けされているかどうか確認して、マーク付けされていない場合は警告を発行します。**rs_marker** ストアド・プロシージャは複写されるので、Adaptive Server はその実行を RepAgent が読み込むことのできるデータベースのトランザクション・ログに記録します。

データベースがプライマリ・データを持つように指定すると、Sybase Central と **rs_init** は **rs_marker** を作成します。プライマリ・データを持たないデータベースには必要ありません。ストアド・プロシージャのテキスト自体は、Sybase リリース・ディレクトリの `scripts` ディレクトリにある `rs_install_primary.sql` または `rsinssys.sql` にあります。

次にサンプル・テキストを示します。

```
create procedure rs_marker
    @rs_api varchar(255)
as
    declare @rep_constant smallint
    select @rep_constant = -32768
    if not exists (select sysstat from sysobjects
      where name = 'rs_marker'
        and type = 'P'
        and sysstat & @rep_constant != 0)
    begin
      print "Have your DBO execute
      ''sp_setreplicate'' on the procedure
      ''rs_marker''"
 return(1)
 end
```

rs_marker は、データベース内のデータを変更しません。その目的は、実行して、トランザクション・ログに記録されるようにすることです。

rs_marker に複写のマーク付けがされていない場合は、**sp_setreplicate** を使用してマーク付けできます。

```
sp_setreplicate rs_marker, 'true'
```

プライマリ・データベースからレプリケート・データベースへの変更

プライマリ・データベースからレプリケート・データベースへ変更できます。

サーバでの操作内容：

1. 現在のレプリケート Replication Server での作業。
   a) データベースにある複写定義に対するサブスクリプションとパブリケーション・サブスクリプションをすべて削除します。
2. 現在のプライマリ Replication Server での作業。
   a) データベース用に定義されているすべての複写定義を削除します。
3. Adaptive Server での作業。
   a) RepAgent を停止します。

   ```
   sp_stop_rep_agent dbname
   ```

   b) RepAgent を無効にします。

   ```
   sp_config_rep_agent dbname, disable
   ```

4. Replication Server
   a) データベースを管理している Replication Server にログインして、**log transfer off** オプションを使用して **alter connection** を実行します。

   ```
   alter connection to data_server.database
   set log transfer off
   ```

5. Adaptive Server
   a) **rs_marker** のステータスを “false” に設定します。

   ```
   sp_setreplicate rs_marker, 'false'
   ```

   b) すべての複写オブジェクトのレプリケート・ステータスを “false” に設定します。
      1. 引数を指定しないで **sp_setreptable** を実行し、データベースの複写テーブルとストアド・プロシージャのすべてのリストを生成します。
      2. **sp_setreptable** と **sp_setrepproc** を使用して、各テーブルとストアド・プロシージャのレプリケート・ステータスを 1 つずつ “false” に設定します。

データベース・コネクションの削除

複写システムからデータベースを削除するには、Sybase Central を使用するか、または **drop connection** を実行します。

コマンドを実行する前に、データベース内のデータに対する複写定義用のサブスクリプションをすべて削除します。プライマリ・データベースに対するコネク

ションを削除する場合は、まずデータベース内のテーブルに対する複写定義をすべて削除します。

注意： **drop connection** は、Replication Server システム・テーブルからデータベース・コネクション情報を削除します。システム内のどのデータベースからもレプリケート・データを削除することはありません。レプリケート・データを削除するには、**drop subscription** に **with purge** オプションを指定して実行します。

コネクションを削除するには、コネクションを削除するデータベースがあるデータ・サーバを指定します。構文を以下に示します。

```
drop connection to data_server.database
```

たとえば、SYDNEY_DS データ・サーバにある pubs2 データベースに対するコネクションを削除するには、次のように入力します。

```
drop connection to SYDNEY_DS.pubs2
```

注意： ログ転送用に RepAgent を使用している場合は、必要に応じてプライマリ・データベースで RepAgent を停止して無効にしてください。

論理コネクションの削除についての詳細については、『Replication Server 管理ガイド 第 2 巻』の「ウォーム・スタンバイ・アプリケーションの管理」の「ウォーム・スタンバイ・データベース・コネクションの変更」の「論理データベース・コネクションの削除」を参照してください。

参照：

- RepAgent の無効化 (126 ページ)

ID サーバからのデータベースの削除

sysadmin dropdb は、ID サーバからデータベースを削除するときに使用します。

複写システム・データベース、データ・サーバ、Replication Server は、ID サーバの RSSD にある rs_idnames システム・テーブルにリストされています。場合によっては、このシステム・テーブルからデータベースのエントリを削除する必要があります。

たとえば、**drop connection** コマンドが失敗した後で、そのコネクション名を再使用したいとします。その場合、そのデータベースに対応するローを rs_idnames システム・テーブルから ID サーバに削除させる必要があります (このシステム・テーブルでは、物理データベース・コネクションの ltype カラムは “P” です)。

ID サーバにログインして **sysadmin dropdb** コマンドを実行し、指定したデータベースのエントリを削除します。**sysadmin dropdb** の構文は次のとおりです。

```
sysadmin dropdb, data_server, database
```

sysadmin コマンドを実行するには、**sa** パーミッションが必要です。

データベース・コネクションのモニタリング

データベース・コネクションをモニタするには、Sybase Central、またはストアド・プロシージャと RCL コマンドを使用します。

トラブルシューティングのためにコネクションをモニタする必要がある場合は、『Replication Server トラブルシューティング・ガイド』を参照してください。

参照：

- 複写システムの管理 (83 ページ)

現在のデータベース・コネクションの表示

現在のデータベース・コネクションのステータスを確認するには、Sybase Central または **admin show connections** を使用します。

admin show_connections は、Replication Server からのデータベース・コネクションすべてについての情報を表示します。**admin show_connections** は、Replication Server からのすべてのルートについての情報も表示します。

『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**admin show_connections**」を参照してください。

Replication Server によって管理されるデータベースのリスト表示

Replication Server によって管理されるデータベースをリスト表示するには、Sybase Central または Replication Server RSSD の **rs_helpdb** ストアド・プロシージャを使用します。

`rs_databases` システム・テーブルは、その Replication Server へのルートを持っている他の Replication Server によって管理されるデータベースも含めて、Replication Server によって管理されるすべてのデータベースのエントリを含んでいます。

rs_helpdb の構文は次のとおりです。

```
rs_helpdb [data_server, database]
```

詳細な使用方法と構文の内容については、『Replication Server リファレンス・マニュアル』の「RSSD ストアド・プロシージャ」の「**rs_helpdb**」を参照してください。

DSI スレッド・ステータスの表示

DSI スレッド・ステータスを表示するには、Sybase Central または **admin who** コマンドを使用して、スレッド・ステータス情報を表示します。

- **admin who** は、DSI スレッドなど、システム内のスレッドをすべて表示します。
- **admin who, dsi** は、DSI スレッドのステータスを表示します。このスレッドは、データ・サーバにトランザクションを送信するために Replication Server が開始します。

スレッド・ステータス全体のリストについては、『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**admin who**」を参照してください。

参照：

- 複写システムの管理 (83 ページ)

Replication Server のセキュリティ管理

ログイン名、パスワード、パーミッションの慎重な管理は、複写システムのセキュリティに不可欠です。

Replication Server のログイン名と特定のパーミッションは次のものに必要です。

- データ・サーバや Replication Server など、複写システムの各コンポーネント Replication Server 15.2 では、単一のユーザ名とパスワードを使用する複写システムでコンポーネントにアクセスできるようにする Replication Server ゲートウェイを使用できます。
- 複写データの設定、または Replication Server のモニタと管理を行う各ユーザ

複写システム全体にわたって、暗号化パスワードを設定し、暗号化されたパスワードを変更できます。パスワード暗号化の詳細については、使用しているプラットフォームの『Replication Server インストール・ガイド』および『Replication Server 設定ガイド』を参照してください。

ログイン名、パスワード、パーミッションの作成や修正などの Replication Server のセキュリティ、および、これらの修正に関連する依存性を管理するために RCL コマンドを使用できます。さらに Replication Server では、ネットワーク経由の安全なメッセージ転送を保証し、複写システム内の複数の Replication Server へのシームレスなログインに対するユーザ認証を可能にする、サード・パーティのセキュリティ・サービスをサポートしています。

参照：

- ネットワークベース・セキュリティの管理 (262 ページ)
- Sybase Central の起動 (57 ページ)

Replication Server システムのセキュリティ管理

RSSD、RepAgent、ID サーバ、Replication Server など、Replication Server システムのさまざまなコンポーネントに対してログイン名とパスワードを設定する必要があります。

あるプロセスから別のリモート・プロセスへのログインが必要な場合がよくあります。そのような場合は、ログイン中のプロセスに割り当てられたログイン名とパスワードが、リモート・プロセスにも存在する必要があります。リモート・プロセスにログインするために使用されるパスワードを、現在ログイン中のプロセスについてだけ変更しても、ログインは失敗します。

一般的な規則として、システム・ログイン名を指定または変更する場合は、その名前をユニークなものにする必要があります。異なるロールに対して同じログイ

ン名を使用すると、パスワードを変更するたびに、この項で説明する依存性の多くに影響が出ます。

複写システムのログイン名

複写システムのコンポーネントには、いくつかのログイン名が必要です。

表 11 : 複写システムのログイン名の概要

送信元サーバ	送信先サーバとデータベース	ログイン名の説明
Primary Replication Server (プライマリ Replication Server)	プライマリ Adaptive Server と RSSD	RSSD プライマリ・ユーザ
Replicate Replication Server (レプリケート Replication Server)	レプリケート Adaptive Server と RSSD	RSSD メンテナンス・ユーザ
Replicate Replication Server (レプリケート Replication Server)	レプリケート Adaptive Server とレプリケート・データベース	データベース・メンテナンス・ユーザ
RSSD 用 RepAgent	Replication Server	RSSD 用 RepAgent ユーザ
プライマリ・データベース用 RepAgent	Replication Server	プライマリ・データベース用 RepAgent ユーザ
Replication Server	ID サーバ (Replication Server)	ID サーバ・ユーザ
Replication Server	他の Replication Server	Replication Server ユーザ (RSI ユーザ)

RSSD のログイン名とパスワード

Replication Server はプライマリ・ユーザ・ログイン名とメンテナンス・ユーザ・ログイン名を使用して、セキュリティを管理します。

Replication Server をインストールすると、**rs_init** プログラムは RSSD を管理するために Adaptive Server のプライマリ・ログイン名とメンテナンス・ログイン名を作成します。

Replication Server は「プライマリ・ユーザ」のログイン名を使用して、プライマリ Replication Server 用の RSSD 内のシステム・テーブルを修正します。この修正には、他の Replication Server 用の RSSD に複写する、ルート、複写定義、ファンクション文字列情報の変更が含まれることがあります。プライマリ・ユーザの設定は、**rs_init** を使用してプライマリ RSSD を作成するときに行います。

Replication Server は、「メンテナンス・ユーザ」のログイン名を使用してレプリケート RSSD に修正を適用します。RepAgent は、メンテナンス・ユーザによって行われた RSSD の修正をフィルタして、他の RSSD に複写されないようにします。メンテナンス・ユーザの設定は、**rs_init** を使用してレプリケート RSSD を作成するときに行います。

プライマリ・ユーザまたはメンテナンス・ユーザのログイン名またはパスワードが変更された場合は、Replication Server の設定ファイルを編集してこれらの変更と一致させ、Replication Server を再起動します。

RSSD プライマリ・ユーザのログイン名とパスワードの変更のガイドライン

RSSD プライマリ・ユーザのログイン名とパスワードを変更する場合は、次のガイドラインに従います。

- ルートの作成中には、RSSD プライマリ・ユーザのログイン名やパスワードを変更しないでください。
  ルートの作成中、送信先 Replication Server はプライマリ・ユーザのログイン名とパスワードを使用して、複写される RSSD システム・テーブル用の送信先サイトで、サブスクリプションの作成とマテリアライゼーションを行います。
- RSSD プライマリ・ユーザのログイン名やパスワードと同じ変更を Replication Server にも適用してください。
  - 暗号化されているパスワードでもクリア・テキストのパスワードでも、変更する場合は **alter user** に **set password** 句を指定します。
  - ログイン名とパスワード (暗号化されている場合でもクリア・テキストの場合でも) の両方を変更するには、**drop user** を使用して古いログイン名を削除してから、**create user** を使用して新しいログイン名とパスワードを作成します。次に、そのユーザに **primary subscribe** パーミッションを付与します。
  - Replication Server の設定ファイルを新しいログイン名とパスワードに更新します。パスワードが暗号化されている場合は、**rs_init** を使用します。
  - 更新を有効にするために、Replication Server を再起動します。

  コマンド構文の詳細については、『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**alter user**」を参照してください。

参照：

- Replication Server パーミッションの管理 (251 ページ)
- Replication Server のパーミッション (251 ページ)

RepAgent 用の Replication Server ログイン名とパスワード

RepAgent 用の Replication Server ログイン名とパスワードを変更するための、ガイドライン、コマンド、プロシージャについて説明します。

RepAgent は、RSSD の複写システム・テーブルまたはプライマリ・データベースに対する変更についての情報を、データベース・トランザクション・ログから取り出して、分配用として Replication Server に送信します。

Replication Server には、RepAgent 用のログイン名が必要です。**rs_init** プログラムは、**create user** コマンドを使用してこの Replication Server ユーザを追加します。

RepAgent 用の Replication Server ログイン名やパスワードを変更する場合は、次のガイドラインに従います。Replication Server に作成するログイン名とパスワードは、Adaptive Server に RepAgent を設定するときに使用するものと同じでなければなりません。

Replication Server での作業：

- パスワードを変更するには、**alter user** コマンドに **set password** 句を指定して実行します。
- ログイン名とパスワードの両方を変更する場合は、**drop user** コマンドを使用して古いユーザ・ログイン名を削除してから、**create user** コマンドを使用して新しいログイン名とパスワードを作成します。次に、そのユーザに **connect source** パーミッションを付与します。

Adaptive Server での作業：

- ログイン名とパスワードを変更するには、**sp_config_rep_agent** システム・プロシージャに *dbname*、*rs_servername*、*rs_username*、*rs_password* の各オプションを指定して実行します。
  これによって、データベース `sysattributes` テーブル内のログイン名とパスワードが更新されます。パスワードは常に暗号化されます。
- 更新を有効にするために、RepAgent を再起動します。

構文の詳細については、『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の **alter user** と、『Replication Server リファレンス・マニュアル』の「Adaptive Server コマンドとシステム・プロシージャ」の **sp_config_rep_agent** を参照してください。

ID サーバのログイン名とパスワード

ID サーバのログイン名とパスワードを変更するためのガイドラインについて説明します。

ID サーバは、複写ドメイン内の Replication Server とデータベースを登録します。Replication Server は、Replication Server 設定ファイル内の *ID_user* 設定パラメータ

を使用して ID サーバに接続します。各 Replication Server について、ID サーバのログイン名とパスワードが ID サーバのエントリと一致する必要があります。

ID サーバは、最初にインストールした Replication Server でなければなりません。ID サーバのログイン名とパスワードは、**rs_init** を使用して設定します。

ID サーバのログイン名やパスワードを変更した場合は、ID サーバ自体の Replication Server 設定ファイルのほか、ID サーバに定義されている各 Replication Server の Replication Server 設定ファイルでも、*ID_user* を変更してください。パスワードの変更は **rs_init** を使用して行うことができます。

Replication Server 内の ID サーバのログイン名とパスワードも変更する必要があります。

参照：
- Replication Server ログイン名とパスワードの管理 (243 ページ)

他の Replication Server 用の Replication Server ログイン名とパスワード

オペレーションを送信するために、Replication Server は他の Replication Server にログインします。

ログイン名は、**rs_init** を使用して作成します。このログイン名は、ある Replication Server から別の Replication Server への直接ルートを作成するときに使用します。

直接ルートで使用されるログイン名のパスワードを変更するには、**alter route** コマンドを実行します。

参照：
- ルートの管理 (161 ページ)

メンテナンス・ユーザの Adaptive Server ログイン名とパスワード

Replication Server は、メンテナンス・ユーザのログイン名を使用して RSSD データベースまたはユーザ・データベースの Adaptive Server にログインします。プライマリのオペレーション (**insert**、**delete**、または **update**) をレプリケート・データベースに適用するときに、Replication Server はメンテナンス・ユーザのログイン名とパスワードを使用します。

注意： メンテナンス・ユーザに付与されるパーミッションの 1 つに **replication_role** があります。メンテナンス・ユーザの **replication_role** を取り消すと、メンテナンス・ユーザが **sa_role** を付与されているか、テーブルを所有しているか、データベース所有者としてエイリアス指定されている場合を除いて、Replication Server は **truncate table** を複写しません。

メンテナンス・ユーザのパスワードを変更するには、**alter connection** コマンドを使用します。

パスワードの暗号化

Replication Server はすべてのパスワードを暗号化し、暗号化されたフォーマットでパスワードを保管および転送します。

Replication Server では、新しい Replication Server のインストールに対してすべてのパスワードを保存するときに、クリア・テキストではなくパスワードの暗号化を使用します。

rs_init、**create user**、**alter user**、**create connection**、**alter connection**、**create route**、および **alter route** を使用してパスワードを指定または変更すると、Replication Server では、アルゴリズムを使用して、`rs_users` と `rs_maintusers` RSSD システム・テーブルおよび Replication Server 設定ファイルのパスワードをすべて暗号化します。パスワードは復号化することはできません。

Replication Server では、`rs_encryptionkeys` RSSD システム・テーブルの `rs_password_key` ローおよび設定ファイルの **RS_random** 属性を使用して、パスワードの暗号化と復号化をサポートします。Replication Server を起動する場合、および Replication Server によってシステム・テーブルまたは設定ファイル内にインストールに固有のランダム値が見つからない場合、Replication Server はシステム・テーブルの `rs_password_key` ローおよび **RS_random** 属性のために、それらの値を自動的に生成します。

alter encryption key *password_key_row_name* **regenerate** コマンドで、システム・テーブルと設定ファイルにパスワード暗号化キーのランダム値を再生成できます。`rs_encryptionkeys` の `rs_password_key` ローにあるパスワードの暗号化キーを再生成するには、以下を入力します。

```
alter encryption key rs_password_key regenerate
```

警告！ 設定ファイルの **RS_random** 属性を手動で変更または削除しないでください。Replication Server が起動しなくなります。

Replication Server は、以下の場合、暗号化されたパスワードを取得および起動できません。

- 設定ファイルをバックアップおよびリストアするときに、対応する RSSD または ERSSD を使用しなかった。
- **RS_random** 属性が消失するか破損している。

有効な **RS_random** 属性が存在しないため、すべてのユーザがログインできない場合、**RS_random** 属性が存在するのであれば、設定ファイルから削除し、sa ユーザ・パスワードをリセットします。その後、Replication Server にログインし、すべてのユーザとメンテナンス・ユーザのパスワードを手動で設定できます。

アップグレードおよびダウングレードの注意事項については、『Replication Server 設定ガイド』の「パスワードの暗号化」を参照してください。

参照：

- sa ユーザ・パスワードを紛失した場合または忘れた場合のリセット (250 ページ)

Replication Server クライアント・コネクションへの暗号化パスワードの送信

Replication Server は、**isql** の **-X** オプションをサポートしており、クライアント・コネクションの確立時にネットワークを介して暗号化パスワードを送信できます。

RSSD との最初のコネクションを除く、すべての Replication Server クライアント・コネクションで暗号化パスワードが送信されるようにするには、Replication Server の **send_enc_password** 設定パラメータを “on” に設定します。たとえば、次のように入力します。

```
configure replication server
 set send_enc_password to 'on'
```

RSSD との最初のコネクションを含む、すべての Replication Server クライアント・コネクションで暗号化パスワードが送信されるようにするには、テキスト・エディタを使用して *rs_name*.cfg ファイルの **RS_send_enc_pw** 設定パラメータを “on” に設定します。

RS_send_enc_pw を “on” にした場合、**send_enc_password** が “off” に設定されていたとしても、RSSD とのすべての Replication Server コネクションで暗号化パスワードが送信されます。

既存の暗号化パスワードのマイグレーション

新しく作成されたパスワードは、連邦情報処理標準 (FIPS: Federal Information Processing Standards) 認定の 140-2 暗号化アルゴリズムを使用します。

Replication Server 設定ファイル内や、rs_users テーブルと rs_maintusers テーブル内の既存の暗号化パスワードをマイグレートするには、次の表のコマンドを使用します。

表 12 : 新しいアルゴリズムでパスワードを暗号化するためのコマンド

作業	コマンド／手順
既存のユーザ・パスワードを新しいアルゴリズムに暗号化する	`alter user user set password password` 構文の説明は次のとおり。 • *user* は、既存のユーザのログイン名。 • *passwd* は、新しいアルゴリズムを使用して暗号化する既存のパスワード。
既存のデータベース・メンテナンス・ユーザ・パスワードを新しいアルゴリズムに暗号化する	`alter connection to data_server.database set password to password` ここで、*password* は、新しいアルゴリズムを使用して暗号化する既存のパスワード。
既存のルート・ユーザ・パスワードを新しいアルゴリズムに暗号化する	`alter route to dest_replication_server set password to passwd` 構文の説明は次のとおり。 • *dest_replication_server* は、送信先 Replication Server の名前。 • *passwd* は、新しいアルゴリズムを使用して暗号化する既存のパスワード。
設定ファイル内の既存のユーザ・パスワードを新しいアルゴリズムに暗号化する	• 新しいアルゴリズムを使用してパスワードを暗号化するには、**rs_init** を使用する。

拡張パスワード暗号化のサポート

バージョン 15.1 以降の Replication Server では、サーバ認証またはクライアント認証、拡張パスワード暗号化をサポートするためのキー・ペアの生成、ネットワーク上の Replication Server 間および Replication Server とプライマリ／レプリケート・データ・サーバ間で送信されるパスワードの暗号化と復号化のための暗号法を提供するために Sybase 共通セキュリティ・インフラストラクチャ (CSI: Common Security Infrastructure) が使用されます。

拡張パスワード暗号化では、接続プロパティ **CS_SEC_EXTENDED_ENCRYPTION** が有効になっている Open Client アプリケーションが Replication Server に接続できるようにする非対称キー暗号化が使用されます。これにより、Replication Server が他のサーバに接続するときに **CS_SEC_EXTENDED_ENCRYPTION** を有効にすることもできます。

非対称キー暗号化では、パスワードの暗号化にはパブリック・キー、パスワードの復号化にはプライベート・キーが使用されます。プライベート・キーはネットワークを介して共有されないため、安全です。

注意： 拡張パスワード暗号化機能を使用するには、拡張パスワード暗号化をサポートしているサーバ (ASE 15.0.2 ESD #2 以降など) が必要です。

Sybase Central の依存性

Replication Manager は、サーバを Replication Manager に追加したときに指定したログイン名とパスワードを使用して Replication Server と RSSD にログインします。

Replication Manager を使用している場合は、ログイン情報を更新してください。この情報は、[Replication Server のプロパティ] ダイアログ・ボックスで確認できます。

Replication Server オブジェクト作成の依存性

ログイン名とパスワードの依存性は、Replication Server オブジェクトを作成するとき、特にプライマリ Replication Server またはレプリケート Replication Server で実行されるサブスクリプションと複写ファンクション (適用ファンクションと要求ファンクション) を作成するときにも生じます。

サブスクリプション
サブスクリプションを作成する場合、レプリケート Replication Server にログインするために使用したログイン名が、プライマリ Replication Server とプライマリ Adaptive Server の両方に存在している必要があります。このログイン名には、3 つのサーバすべてで同じパスワードが設定されていなければなりません。

サブスクリプションを削除する場合、レプリケート Replication Server は、レプリケート Replication Server にログインするときに使用したログイン名とパスワードで、プライマリ Replication Server にログインします。プライマリ Replication Server のこのログイン名のパスワードは、**drop subscription** 処理が完了するまで変更しないでください。

Replication Server で自動的に作成される RSSD の「プライマリ」ユーザのログイン名は、ルートの作成時に「サブスクリプションを作成するユーザ」として使用されます。RSSD プライマリ・ユーザには、サブスクリプションを作成するユーザに対する規則が適用されます。

サブスクリプションに関する注意事項：
- サブスクリプションを、**sa** ユーザとして作成しないでください。
- サブスクリプションを作成するときにプライマリ Replication Server で発行される **select** コマンドは、複写定義に所有者名が指定されていない場合、テーブル所有者名を含みません。所有者名が指定されていない場合は、ユーザがテーブ

ルを所有しているのか、“dbo” ユーザがテーブルを所有しているのかを確認してください。
- サブスクリプションのマテリアライゼーションまたはマテリアライゼーション解除が行われている間は、パスワードを変更しないでください。

複写ファンクションとストアド・プロシージャ
プライマリ Replication Server は、レプリケート Replication Server から要求ファンクションまたは要求ストアド・プロシージャを受信すると、レプリケート・サイトで要求ファンクションまたは要求ストアド・プロシージャを開始したユーザのログイン名とパスワードで、プライマリ・データ・サーバにログインします。

したがって、レプリケート・データ・サーバで要求ファンクションまたは要求ストアド・プロシージャを実行するには、そのユーザがプライマリ・データ・サーバで同じログイン名とパスワードを持っており、ストアド・プロシージャに対する **execute** パーミッションを持っている必要があります。

レプリケート Replication Server は適用ファンクションまたは適用ストアド・プロシージャをプライマリ・サイトから受信すると、メンテナンス・ユーザのログイン名とパスワードを使用してレプリケート・データベース内のストアド・プロシージャを実行します。

Replication Server ユーザのセキュリティ管理

Replication Server には専用のログイン名があります。このログイン名はデータ・サーバのログイン名とは別のものです。Replication Server ログイン名は、システム管理者が Replication Server コマンドを実行するために必要なものです。

ユーザは、Replication Server によって複写されたデータにアクセスするとき、Replication Server ログイン・アカウントを必要としません。複写データは、特定のデータベースにアクセスするためのパーミッションを持っているユーザが利用できます。データベース管理者は、データベースを作成し、そのデータベースへのアクセスを認可する責任があります。

ユーザのパスワード暗号化は、有効／無効を切り替えることができます。

参照：
- ユーザ、パスワード、パーミッションの検査 (258 ページ)

Replication Server ログイン名とパスワードの管理

複写システム管理者または **sa** パーミッションを持つユーザは、ログイン名を管理し、パスワードのセキュリティを実装して管理できます。

表 13 : ログイン名の管理用コマンド

コマンド	作業
create user	新しいログイン名の作成
alter user	ログイン名のパスワードの変更
drop user	ユーザ・ログイン名の削除

パスワード入力の非表示

ユーザを作成または変更するときに設定したパスワードを非表示にするには、**isql --conceal** オプションを ":?" ワイルドカード文字と共に使用します。

1. ユーザを作成したり、ユーザのパスワードを変更したりする前に、**--conceal** オプションを指定して **isql** を開始します。
   たとえば、test_2 というユーザの ny_rs という Replication Server で次のように入力します。

   ```
   isql -Sny_rs -Utest_2 --conceal
   Password:
   ```

2. **alter user** または **create user** コマンドを入力したら、クリア・テキストでパスワードを入力する代わりに、パスワード入力用コマンド・ラインの先頭 :? ワイルドカード文字のペアを入力します。
   たとえば、test_2 というユーザのパスワードを変更するには、次のように入力します。

   ```
   alter user test_2
   set password
   :?
   verify password
   :?
   go
   ```

   新しいパスワードの確認入力を求めるには、":?:?" のように、同じライン内でワイルドカード文字のペアを 2 回続けて使用します。次に例を示します。

   ```
   alter user test_2
   set password
   :?:?
   verify password
   :?
   go
   ```

isql は、ワイルドカード文字のペアを置き換えるための情報の入力を求めます。ワイルドカード文字のペア :? を入力するたびに、別の :? プロンプトが生成されます。:? プロンプトで入力する文字は一切表示されません。

3. **set password** 句に対応する最初の :? プロンプトで新しいパスワードを入力し、**verify password** 句に対応する 2 番目の :? プロンプトで古いパスワードを入力します。

```
:? new_pasword
:? old_pasword
```

ワイルドカード文字を ":?:?" のように 2 回続けてコマンドで使用した場合は、次のように表示されます。

```
:?
Confirm :?
:?
```

set password 句に相当する最初の :? プロンプトで新しいパスワードを入力し、2 番目の Confirm :? プロンプトで新しいパスワードを再入力して、**verify password** 句に相当する 3 番目の :? プロンプトで古いパスワードを入力します。
入力したパスワードがパスワード・セキュリティ要件に適合している場合は、次のように表示されます。

```
User 'test_2' is altered.
```

Replication Server ログイン名の作成

新しいログイン名を Replication Server に追加するには、**create user** コマンドを使用します。

ログイン名の作成時に、そのユーザのパスワードを指定します。そのユーザがパスワードを持たない場合は、パスワードを “null” に設定して空の文字列を指定します。

create user コマンドには **sa** パーミッションが必要です。**create user** の構文は次のとおりです。

```
create  user user
set password {new_password | null}
[set password_parameter to 'parameter_value']
```

ユーザのパスワードは最長 30 文字までで、英字、数字、記号を含むことができます。大文字、小文字は区別されます。パスワードがスペースを含む場合は、パスワード全体を一重引用符で囲んでください。

ユーザは、**alter user** コマンドを使用して各自のパスワードを変更できます。

ユーザ名が “thomk” でパスワードが “vacUUm” のログイン名を作成するには、次のように入力します。

```
create user thomk
 set password vacUUm
```

システム管理者として、ユーザのパスワードの有効期間を設定できます。たとえば、初期パスワードが 1Buiopr89、パスワード有効期間が 90 日の、jsmith という名前のユーザを作成するには、次のように入力します。

```
create user jsmith
set password to 1Buiopr89
set password_expiration to '90'
```

『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の **create user** を参照してください。

参照：

- パーミッションのまとめ (254 ページ)

Replication Server パスワードの変更

複写システム管理者は、**alter user** コマンドを使用して、任意のユーザのパスワードを変更できます。各ユーザは自分のパスワードを変更できます。

alter user の構文は次のとおりです。

```
alter user user
set password {new_password | null}
[verify password old_password]
[set password_parameter to 'parameter_value']
```

create user を使用してパスワードを指定するときと同じ規則が、**alter user** にも適用されます。

verify password 句は、あるユーザが他のユーザのパスワードを変更できないようにするものであり、**sa** パーミッションを持たないユーザに必要です。

たとえば、“louise” というログイン名を持つユーザが自分のパスワードを “EnnuI” から “somNIfic” に変更するには、次のように入力します。

```
alter user louise
 set password somNIfic
 verify password EnnuI
```

システム管理者として、ユーザのパスワードの有効期間を設定できます。たとえば、ユーザ jsmith のパスワードとパスワード有効期間を 60 日に変更するには、次のように入力します。

```
alter user jsmith
set password to newpass
set password_expiration to '60'
```

『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の **alter user** を参照してください。

全ユーザを対象としたパスワード設定オプション

全ユーザを対象とした Replication Server のパスワード・セキュリティは、**configure replication server** コマンドのオプションを使用して実装および管理できます。Replication Server では、新しい Replication Server のインストールに対してすべてのパスワードを保存するときに、クリア・テキストではなくパスワードの暗号化を使用します。

構文

```
configure replication server
set password_param to 'parameter_value'
```

パスワード設定オプション

password_parameter	**説明**
min_password_len	最小文字数。 • 0 - 最小長なし。 • 範囲 – 6 〜 16 (デフォルトは 6)。
max_password_len	最大文字数。**max_password_len** は常に **min_password_len** より大きい値に設定する。 範囲 – 13 〜 30 (デフォルトは 30)。
password_lowercase_required	小文字が必須であるかどうか。 • True – 必須。 • False – 不要 (デフォルト)。
password_uppercase_required	大文字が必須であるかどうか。 • True – 必須。 • False – 不要 (デフォルト)。
password_numeric_required	数値が必須であるかどうか。 • True – 必須。 • False – 不要 (デフォルト)。

password_parameter	説明
password_special_char_required	特殊文字が必須であるかどうか。 • True – 必須。 • False – 不要 (デフォルト)。

simple_passwords_allowed	このオプション (または "simple_passwords_allowed") を false に設定した場合、Replication Server ではパスワードにユーザ名またはユーザ・パスワード辞書の値を含めることが許可されない。
	• True – 許可 (デフォルト)。
	• False – 不許可。
	パスワード辞書は RSSD の `rs_dictionary` システム・テーブルに作成できる。テーブルにはデフォルト値は格納されない。独自のスクリプトを作成して値をテーブルに挿入する必要がある。次に例を示す。
	`insert into rs_dictionary (words) values ("abcd");`
	`insert into rs_dictionary (words) values ("1234");`

disallowed_prev_passwords	ユーザが自分のパスワードを変更するときに再使用できない以前のパスワードの数。 • 0 – 許可される以前のパスワード。 • 範囲 – 0 ～ 32,767 (デフォルトは 0)。 管理者がパスワードをリセットする場合、パラメータ値はユーザ・パスワードに適用されない。

password_parameter	説明
password_expiration	パスワードの有効期限が切れてから経過した日数。 • 0 - パスワードの有効期限は切れない (デフォルト)。 • 範囲 - 0 ～ 32,767。 **password_expiration** は **alter user** および **create user** とともに使用できる。 パスワードの有効期限が切れると、Replication Server はアカウントをロックし、パスワードの有効期限が切れたことをユーザに通知する。ユーザはこのパスワードをリセットしないと、管理者がパスワードをリセットしないかぎり、いったん接続を解除された以降はログインできなくなる。新しいパスワードは、パスワード要件をすべて満たす必要がある。 **rs_init** が **connect source** パーミッションまたは ID ユーザで作成するユーザのパスワードには、有効期限はない。そのようなパスワードは、Replication Server でユーザ全員に対して設定された **password_expiration** のどのような設定でもオーバーライドする。データベース、他の Replication Server、および Replication Agent では、**connect source** パーミッションがあるユーザ ID が使用される。 管理者は、Replication Agent または RSI 向けに作成されたユーザのパスワードの有効期限が切れないようにパスワードを設定する。
initial_password_expiration	最初のパスワードの有効期限が切れてから経過した日数。 • 0 - 最初のパスワードの有効期限は切れない。 • 範囲 – 0 ～ 32,767 (デフォルトは 0)。 ユーザの最初のパスワードは、管理者がユーザを作成するとき、またはユーザのパスワードをリセットするときに設定するパスワード。
max_failed_logins	アカウントをロックする前に Replication Server が許可するログイン要求の最大失敗回数。 • 0 – アカウントは決してロックされない。 • 範囲 – 0 ～ 32,767 (デフォルトは 0)。 Replication Server は、**password_lock_interval** に設定されている時間間隔に従ってアカウントをロックする。

password_parameter	説明
password_lock_interval	ユーザが **max_failed_logins** に設定されているログイン最大試行回数に達した場合にアカウントがロックされる分数。 • 0 – アカウントは管理者がパスワードをリセットするまでロックされる。 • 範囲 – 0 ～ 32,767 (デフォルトは 0)。
unused_login_expiration	ユーザ・アカウントが期限切れになる前の未使用の日数。 • 0 – 未使用のアカウントは期限切れにならない。 • 範囲 – 0 ～ 32,767 (デフォルト)。 Replication Server は、**unused_login_expiration** より長く未使用になっているアカウントをロックする。管理者はパスワードをリセットすることでこのアカウントを再度アクティブにできる。

例

- すべてのユーザのパスワードの最小の長さを 8 文字に設定します。

```
configure replication server
set min_password_len to '8'
```

- すべてのユーザのパスワードの有効期間を 90 日に設定します。

```
configure replication server
set password_expiration to '90'
```

使用法

- **password_expiration** は、**alter user** および **create user** とともに使用できる唯一のパラメータです。詳細については、『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」で「**alter user**」および「**create user**」を参照してください。
- **create user** または **alter user** を使用して個々のユーザに対して指定されるパスワード設定は、**configure replication server** を使用して設定されるすべての値をオーバーライドします。
- 現在のユーザのステータスと設定を表示するには、**rs_helpuser** ストアド・プロシージャを使用します。
- パスワードの最小の長さ
  - 管理者またはユーザがユーザ・パスワードを変更する場合、新しいパスワードは、**configure replication server** を使用してシステムに対して設定されている最小の長さを満たしたものにしなければなりません。

    - パスワードの最小の長さを設定しても、次回ユーザがパスワードを変更するときまで、現在のユーザ・パスワードには影響がありません。
- パスワード有効期間
    - ユーザのパスワードを管理者またはそのユーザ本人が変更すると、Replication Server はパスワード有効期限の日付を記録します。ユーザがログインすると、Replication Server はログイン日付とパスワード有効期限設定を比較します。パスワード有効期限設定が、そのユーザに対してまたはシステム・レベルで設定されており、かつ Replication Server がパスワードの有効期限が切れたと判断した場合、Replication Server はユーザにパスワードを変更するよう通知し、アカウントをロックします。Replication Server がアカウントのロックを解除するのは、要件を満たす新しいパスワードがユーザによって入力された場合だけです。ユーザが接続を解除してからパスワードを変更した場合は、管理者がパスワードをリセットしてください。
    - "connect source" 権限を持つユーザ (Replication Agent ユーザなど) については、**password_expiration** を 0 に設定することをおすすめします。これは、パスワードの有効期限が切れて、データのレプリケーションが妨げられることを防ぐためです。
- **password_encryption** パラメータは廃止されています。

パーミッション
パスワード・パラメータを設定するには sa パーミッションが必要です。

Replication Server ログイン名の削除
既存のログイン名を Replication Server から削除するには、**drop user** コマンドを使用します。

drop user には、**sa** パーミッションが必要です。**drop user** の構文は次のとおりです。

```
drop user user
```

たとえば、次のコマンドはログイン名 “thomk” を削除します。

```
drop user thomk
```

sa ユーザ・パスワードを紛失した場合または忘れた場合のリセット
sa ユーザのパスワードを紛失したか忘れた場合にパスワードをリセットするには、**reset_password=sa** パラメータを使用します。このパラメータを使用して、他のアカウントのパスワードをリセットすることはできません。

1. **reset_password=sa** パラメータを Replication Server 設定ファイルに追加します。

   注意： **isql** を使用して **reset_password=sa** を実行することはできません。

2. Replication Server を再起動します。
   Replication Server を起動すると、sa ユーザが 1 回だけ使用できるランダム・パスワードが生成されます。その後で **reset_password=sa** パラメータが設定ファイルから削除され、次回以降の再起動時にパスワードが生成されるのを防ぎます。
3. 生成されたワンタイム・パスワードを使用して、sa ユーザとしてログインします。
   パスワードの有効期限がただちに切れ、Replication Server によって sa ユーザ・アカウントがロックされます。これは、同じパスワードを使用した別のユーザによってログインが試みられるのを防ぐためです。パスワードの期限が切れたことが Replication Server によって sa ユーザに通知されます。
4. 要件をすべて満たしている新しいパスワードを入力します。

Replication Server パーミッションの管理

複写システム管理者は、**grant** コマンドと **revoke** コマンドを使用して Replication Server のパーミッションを管理します。パーミッションによって、各ユーザが実行できる RCL コマンドが決まります。

Replication Server ログイン名を持っているユーザであれば、すべての **admin** コマンドと **check subscription** コマンドを実行できます。これ以外のコマンドは、必要なパーミッションを付与されているユーザだけが実行できます。

Replication Server のパーミッション

Replication Server のユーザには、次の 4 つのパーミッションのいずれかを付与できます。

表 14 : Replication Server のパーミッション

パーミッション	説明
sa	**sa** パーミッションを持つユーザは複写システム管理者である。このユーザはすべての Replication Server コマンドを実行でき、他のユーザに対して、**sa** やその他のパーミッションを付与することも取り消すこともできる。
create object	**create object** パーミッションを持つユーザは、複写定義、サブスクリプション、ファンクション文字列などのオブジェクトを作成できる。**create object** パーミッションを持つユーザは、自動的に **primary subscribe** パーミッションを持つ。

パーミッション	説明
primary subscribe	**primary subscribe** パーミッションを持つユーザは、Replication Server によって管理されるデータベース内のプライマリ・データに対してサブスクリプションを作成するのに必要なコマンドを実行できる。プライマリ・サイトに **primary subscribe** パーミッションを持ち、レプリケート・サイトに **create object** パーミッションを持つユーザは、プライマリ・サイトのデータに対するサブスクリプションを作成することはできるが、プライマリ・サイトに複写定義やファンクション文字列を作成することはできない。
connect source	**connect source** パーミッションは次のものに対して必要。 • RepAgent が Replication Server にログインするために使用し、RepAgent がログ転送言語 (LTL) という RCL コマンドのサブセットを実行できるようにするログイン名。 • 複写データまたは複写ファンクションを送信するために送信元 Replication Server が送信先 Replication Server に接続するときに使用するログイン名。このログイン名は、**create route** コマンドを使用して指定する。

サブスクリプション作成の必要条件

サブスクリプションを作成する前に必要なパーミッションについて説明します。

サブスクリプションの作成者は、プライマリおよびレプリケート Replication Server の両方にアカウントを持ち、それらのアカウントは同じログイン名とパスワードを持つ必要があります。サブスクリプションの作成者が、レプリケート Replication Server で 1 つまたは複数のコマンドを入力すると、レプリケート Replication Server からプライマリ Replication Server に要求が渡されます。

オプション句 **use dump marker** と **subscribe to truncate table** を使用する場合、レプリケート Replication Server のログイン名とパスワードを、プライマリ・データベースとレプリケート・データベースに対してだけでなく、プライマリ Replication Server に対しても同じにする必要があります。

サブスクリプション作成者は、サブスクリプションをマテリアライズするために、レプリケート Replication Server (サブスクリプション・データの送信先) に少なくとも **create object** パーミッションを持つ必要があります。

プライマリ Replication Server (サブスクリプション・データの送信元) では、サブスクリプション作成者は、サブスクリプションの作成に関連するすべてのコマンドをレプリケート・サイトに入力するために、少なくとも **primary subscribe** パーミッションを持つ必要があります。

- **create subscription** (アトミックおよびノンアトミック・マテリアライゼーションの場合)

- **define subscription** (バルク・マテリアライゼーションの場合)
- **activate subscription** (バルク・マテリアライゼーションの場合)
- **validate subscription** (バルク・マテリアライゼーションの場合)
- **drop subscription**

primary subscribe パーミッションは **create object** パーミッションのサブセットであり、プライマリ Replication Server で提供されます。このパーミッションによって、レプリケート・サイトの各ユーザは、プライマリ・サイトに格納されているデータに対してサブスクリプションを作成できます。ただし、レプリケート・サイトから、サブスクリプション以外のオブジェクトをプライマリ・サイトに作成することはできません。

注意： **create object** パーミッションと **sa** パーミッションを持つユーザも、レプリケート Replication Server からサブスクリプションを作成できます。レプリケート・サイトのユーザがサブスクリプションを作成するためにプライマリ Replication Server で必要な最小限のパーミッションは、**primary subscribe** です。

サブスクリプションを作成するユーザには、次の Adaptive Server パーミッションが必要です。

- プライマリ・データベース内のテーブルに対する **select** パーミッション
- レプリケート・テーブルに対する **insert**、**update**、**delete** パーミッション
- プライマリ・データベース内の **rs_marker** ストアド・プロシージャの **execute** パーミッション

複写システム管理者は、プライマリ・サイトの **primary subscribe** と **create object** パーミッションを、サブスクリプション作成のためにこれらを必要とするユーザだけに付与するようにしてください。

primary subscribe または **create object** パーミッションを持っているユーザは、テーブルに対する **select** パーミッションを持っていなくてもサブスクリプションの作成を開始できます。このような場合、Replication Server は次のように対応します。

- サブスクリプションがアトミック・マテリアライゼーションで作成される場合、**select with holdlock** オペレーションは、マテリアライゼーション時にプライマリ・データベースで失敗します。dSUB (サブスクリプション・リトライ・デーモン) は、サブスクリプションが削除されるか、そのユーザにプライマリ・データベースのテーブルに対する **select** パーミッションが付与されるまで、**select with holdlock** をリトライします。
- サブスクリプションがノンアトミック・マテリアライゼーションで作成される場合、**select** オペレーションは、マテリアライゼーション時にプライマリ・データベースで失敗します。dSUB は、サブスクリプションが削除されるか、**select** パーミッションが付与されるまで、**select** をリトライします。

- サブスクリプションがバルク・マテリアライゼーションを使用して作成される場合は、**select** トランザクションは存在しないので、エラーは発生せずに、サブスクリプションが作成されます。

パーミッションのまとめ

ユーザは、各 RCL コマンドを実行するために最小限のパーミッションが必要です。

create object パーミッションを持つユーザには、自動的に **primary subscribe** パーミッションが付与されます。**sa** パーミッションを持つユーザは、すべてのコマンドを実行できます。

表 15 : RCL コマンドの実行に最小限必要なパーミッション

実行対象	最小限必要なパーミッション
abort switch	**sa**
activate subscription	レプリケートで **create object**、プライマリで **primary subscribe**
create partition	**sa**
admin コマンド	どのユーザでも実行可能
allow connections	**sa**
alter connection	**sa**
alter database replication definition	**create object**
alter function	**create object**
alter function replication definition	**create object**
alter applied function replication definition	**create object**
alter request function replication definition	**create object**
alter function string	**create object**
alter function string class	**sa**
alter logical connection	**sa**
alter partition	**sa**
alter queue	**sa**
alter replication definition	**create object**
alter route	**sa**

実行対象	最小限必要なパーミッション
alter user	**sa** — ユーザは **verify** 句を指定することによって各自のパスワードを変更できる
assign action	**sa**
check publication	どのユーザでも実行可能
check subscription	どのユーザでも実行可能
configure connection	**sa**
configure logical connection	**sa**
configure replication server	**sa**
configure route	**sa**
create article	**create object**
create connection	**sa**
create database replication definition	**create object**
create error class	**sa**
create function	**create object**
create function replication definition	**create object**
create applied function replication definition	**create object**
create request function replication definition	**create object**
create function string	**create object**
create function string class	**sa**
create logical connection	**sa**
create partition	**sa**
create publication	**create object**
create replication definition	**create object**
create route	**sa**
create subscription	レプリケートで **create object**、プライマリで **primary subscribe**
create user	**sa**
define subscription	レプリケートで **create object**、プライマリで **primary subscribe**

実行対象	最小限必要なパーミッション
drop article	**create object**
drop connection	**sa**
drop database replication definition	**create object**
drop error class	**sa**
drop function	**create object**
drop function replication definition	**create object**
drop function string	**create object**
drop function string class	**sa**
drop logical connection	**sa**
drop partition	**sa**
drop publication	**create object**
drop replication definition	**create object**
drop route	**sa**
drop subscription	レプリケートで **create object**、プライマリで **primary subscribe**
drop user	**sa**
grant	**sa**
ignore loss	**sa**
move primary	**sa**
rebuild queues	**sa**
resume connection	**sa**
resume distributor	**sa**
resume log transfer	**sa**
resume queue	**sa**
resume route	**sa**
revoke	**sa**
set proxy	**sa**
set autocorrection	**create object**
set log recovery	**sa**

実行対象	最小限必要なパーミッション
shutdown	**sa**
suspend connection	**sa**
suspend distributor	**sa**
suspend log transfer	**sa**
suspend route	**sa**
switch active	**sa**
sysadmin コマンド	**sa**
validate subscription	レプリケートで **create object**、プライマリで **primary subscribe**
wait for create standby	**sa**
wait for switch	**sa**

パーミッションの付与

ユーザにパーミッションを付与するには、**grant** コマンドを使用します。

パーミッションの付与と取り消しの権限は複写システム管理者だけが保有します。**sa** パーミッションを付与されているユーザであれば、複写システム管理者の役割を果たすことができ、**sa** パーミッションを付与することによって他のユーザに **grant** と **revoke** の権限を与えることができます。

grant コマンドの構文は次のとおりです。

```
grant
 {sa | create object | primary subscribe | connect source}
 to user
```

user は、パーミッションを付与されるユーザのログイン名です。一度に付与できるパーミッションは 1 つだけです。

パーミッションが割り当てられるのは、Replication Server ユーザに対してであり、データベース・ユーザに対してではありません。**create object** パーミッションを持っている Replication Server ユーザは、Replication Server によって管理されるデータベースに関する Replication Server オブジェクトを作成できます。

次の例では、複写システム管理者が **create object** パーミッションをログイン名 “thomk” に付与します。

```
grant create object to thomk
```

パーミッションの取り消し

以前にユーザに付与したパーミッションを削除するには、Sybase Central または **revoke** コマンドを使用します。

revoke コマンドの構文は次のとおりです。

```
revoke {sa | create object | primary subscribe |
 connect source}
 from user
```

注意： **sa** ログイン名の **sa** パーミッションを取り消すことや、ログイン名自体を削除することはできません。これによって、Replication Server には複写システム管理者が確実に存在することになります。

4 つのパーミッションはそれぞれ独立して管理されます。これらのパーミッションは任意の順序で付与と取り消しができ、結果に変わりはありません。

次の **revoke** コマンドは、**sa** パーミッションを持たないユーザ “louise” が複写定義を作成できないようにします。

```
revoke create object from louise
```

ユーザ、パスワード、パーミッションの検査

Replication Server ユーザのログイン名、パスワード、パーミッション、およびスレッドは、**rs_helpuser** ストアド・プロシージャを使用したり、RSSD 内の `rs_maintusers` および `rs_users` システム・テーブルに問い合わせたりすることで表示できます。Sybase Central を使用して Replication Server ログイン名に関する情報を表示することもできます。

rs_helpuser ストアド・プロシージャの使い方

Replication Server に認識されているユーザのログイン名についての情報を表示するには、**rs_helpuser** ストアド・プロシージャを使用します。

rs_helpuser の構文は次のとおりです。

```
rs_helpuser [user]
```

パラメータの指定を省略すると、**rs_helpuser** は現在の Replication Server に認識されているすべてのユーザ・ログイン名についての情報を表示します。パーミッションは、各プライマリ・ユーザまたはメンテナンス・ユーザのログイン名に対して表示されます。

ログイン名パラメータを指定すると、**rs_helpuser** はそのログイン名についてだけ情報を表示します。

rs_maintusers システム・テーブルに対する問い合わせ

RSSD 内の rs_maintusers システム・テーブルには、メンテナンス・ユーザのログイン名とパスワードの情報が含まれています。

rs_maintusers には、パスワードが暗号化されているかクリア・テキストかを示すカラムと、暗号化パスワードを保持するカラムがあります。

たとえば、次のクエリを RSSD で実行すると、メンテナンス・ユーザについての入手可能なすべての情報 (ログイン名など) が表示されます。

```
select * from rs_maintusers
```

rs_users システム・テーブルに対する問い合わせ

RSSD 内の rs_users システム・テーブルには、Replication Server ユーザのログイン名とパスワードの情報が含まれています。

rs_users には、パスワードが暗号化されているかクリア・テキストかを示すカラムと、暗号化パスワードを保持するカラムもあります。

また、rs_users システム・テーブルには、各ログイン名に対するパーミッションを格納する permissions カラムもあります。permissions カラムは、ユーザに付与されているパーミッションのビットマスクです。

たとえば、次のクエリを RSSD で実行すると、**sa** パーミッションを持つユーザが表示されます。

```
select username, uid from rs_users
 where permissions & 0x0001 != 0
```

rs_users システム・テーブル内のパーミッション・ビットマスク値

パーミッションの各タイプに、ビットマスク値があります。

表 16 : rs_users システム・テーブル内のパーミッション・ビットマスク値

パーミッション	マスク値
sa	0x0001
connect source	0x0002
create object	0x0004
primary subscribe	0x0008

Replication Server ゲートウェイ

バージョン 15.2 では、Replication Server に Replication Server ゲートウェイが導入されています。これにより、さまざまなサーバへの明示的なログインが最小限に抑えられ、複写システムの管理が容易になります。

複写システムの管理では、複写システム管理者 (RSA: Replication System Administrator) が、複数の Replication Server、ID サーバ、対応する Replication Server システム・データベース (RSSD: Replication Server System Database) にログインします。また、RSA は、Replication Server と RSSD の間でログインを頻繁に切り替えます。

Replication Server ゲートウェイは、RSSD のプライマリ・ユーザ名とパスワードを使用して RSSD に、ID サーバのユーザ名とパスワードを使用して ID サーバに、リモート・サーバ ID (RSI: Remote Server Identification) を使用してリモート Replication Server に、メンテナンス・ユーザ ID を使用してリモート Adaptive Server にログインします。Replication Server 自体にアクセスするとき、この情報を複数回提供する必要はありません。

カスケード・コネクション

Replication Server ゲートウェイでは、Replication Server と、Replication Server に直接接続されていないサーバとの通信を可能にするカスケード・コネクションもサポートされます。

また、1 つのクライアント・コネクションを使用して複写ドメインを管理することもできます。たとえば、ID サーバに接続し、その後、ID サーバの RSSD に接続できます。この場合、プライマリの制御 Replication Server と ID サーバの両方がゲートウェイであり、コマンドが ID サーバの RSSD に渡され、結果セットが返されます。

Replication Server ゲートウェイの有効化

Replication Server をその RSSD、ID サーバ、またはリモート Replication Server へのゲートウェイにするには、**connect** コマンドを使用します。

注意： **connect** コマンドを発行するには、Replication Server への初回のログイン用に、**sa** ロールが必要です。

構文：

```
connect [to] [rssd | idserver | srv_name | ds_name.db_name]
```

パラメータ：

- **rssd** – Replication Server をその RSSD のゲートウェイにする。設定ファイルの *RSSD_primary_user* エントリと *RSSD_primary_pw* エントリをゲートウェイが使用できるようにする。
- **idserver** – Replication Server をその ID サーバのゲートウェイにする (Replication Server 自体が ID サーバでない場合)。設定ファイルの *ID_user* エントリと *ID_pw* エントリをゲートウェイが使用できるようにします。
- *srv_name* – ゲートウェイを接続するリモート Replication Server の名前。Replication Server ゲートウェイは、RSI を使用してリモート・サーバにログインするため、リモート・サーバへの直接ルートが必要。

  注意： Replication Server は、それ自体に直接接続できません。ただし、カスケード・コネクションを使用することで、この問題に対処できます。
- *ds_name.db_name* – ゲートウェイを接続するリモート・データ・サーバとデータベースの名前。Replication Server ゲートウェイは、メンテナンス・ユーザを通じてリモート・データ・サーバにログインします。これにより、指定されたデータベースのメンテナンス・ユーザに許可されているタスクを実行できるようになる。ただし、接続先のデータ・サーバで定義された他のデータベースにはアクセスできない。

  Replication Server ゲートウェイは、Adaptive Server と、Enterprise Connect Data Access (ECDA) を必要としない Sybase® IQ データ・サーバに直接接続できるようにします。その他のデータ・サーバの場合、Replication Server ゲートウェイは、ECDA を使用して Replication Server とリモート・データ・サーバに接続する必要があります。

コネクションの追跡

カスケード・コネクションを管理するには、**show connection** コマンドと **show server** コマンドを使用します。

ゲートウェイで作成されたカスケード・コネクションは、コネクション・スタックで保持され、最初の **connect** コマンドを発行した Replication Server がスタックの一番下に置かれます。

- **show connection** – コネクション・スタックの内容を一覧表示する。
- **show server** – 現在稼働中のサーバを表示する。

コネクションの削除

サーバへのコネクションを終了するには、**disconnect** コマンドを使用します。

disconnect では、コネクション・スタックを一度に 1 つずつ終了します。すべてのコネクションを終了するには、**disconnect all** を使用します。

```
{disconnect | disc} [all]
```

Replication Server 15.1 以前では、**disconnect** コマンドの動作が異なります。

Replication Server 15.1 以前では、**disconnect** コマンドは、ゲートウェイ・モードを終了し、最初の **connect** コマンドを発行した Replication Server に稼働中のサーバのステータスを返します。コネクション・スタックに Replication Server バージョン 15.2 と 15.1 以前が含まれる場合に **disconnect** コマンドを発行すると、**show connection** コマンドや **show server** コマンドを実行したときに、想定した出力が表示されない可能性があります。

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

Replication Server ゲートウェイ制限事項

Replication Server ゲートウェイを使用する場合、Replication Server は文字セットの変換を実行できないため、クライアントとサーバで同じロケール・セットを使用してください。

ネットワークベース・セキュリティの管理

サード・パーティのネットワークベースのセキュリティ・メカニズムに対する Replication Server のサポートについて説明します。

クライアント／サーバ環境では、データ転送の機密が保たれるように安全なデータ経路を提供することが重要です。Replication Server は、次の機能に重点を置いたサード・パーティのネットワークベースのセキュリティ・メカニズムをサポートしています。

- 認証と統一化ログイン
- 安全なメッセージ転送

ネットワークベース・セキュリティを使用すると、ユーザはログイン時にセキュリティ・システムによって認証 (ユーザが本人であることを確認する処理) されます。各ユーザは、パスワードの代わりに、リモート・サーバに提示することのできる、クレデンシャルを受け取ります。その結果、ユーザは複写システムのコンポーネントに、1 つのログイン名で連続的にアクセスできます。

Replication Server バージョン 12 以降では、MIT Kerberos バージョン 5 以降、CyberSafe Kerberos バージョン 5 セキュリティ・サーバ、Transarc DCE バージョン 1.1 セキュリティ・サーバをサポートしています。選択するセキュリティ・メカニズムに応じて、データ転送の安全性を保つために、次の中から機能を 1 つ以上選択できます。

- 統一化ログイン ― ユーザは、セキュリティ・メカニズムによって発行される単一のクレデンシャルで複写システムの複数のコンポーネントにログインできる。

- 機密保持 ― 暗号化されたデータの送信と受信を有効にする。
- 整合性 ― データが改ざんされていないことを保証する。
- リプレイの検出 ― データが傍受されていないことを確認する。
- オリジン検査 ― 各データ・パケットの送信元を確認する。
- 順序不整合の検出 ― データ・パケットが送信された順序どおりに受信されたかどうかを検査する。

セキュリティ・メカニズムを使用すると、Replication Server は、他の Replication Server、Adaptive Server、Kerberos または DCE セキュリティ・メカニズムと特定の Replication Server 稼働条件をサポートしている他のデータ・サーバとの間に、安全なコネクションを確立できます。Replication Server 間で安全なデータ転送を行うための方法は、自分で選ぶことができます。

セキュリティ・サービスの機能

セキュリティ・サービスがどのように機能するかは、Replication Server (または Adaptive Server やその他のデータ・サーバ) がクライアントとして動作するか、サーバとして動作するかで異なります。

クライアントは、セキュリティ・メカニズムを使用して、リモート・サーバへの安全な経路を確保します。Replication Server は、(クライアントとして) リモート・サーバにログインし、さらに (サーバとして) 受信ログインを受け入れます。

クライアントとして動作する場合、Replication Server はセキュリティ・メカニズムを使用して、リモート Replication Server または Adaptive Server への安全な経路を確保します。セキュリティ・メカニズムは、安全な経路が確立されたあとに、メッセージ保護機能を提供できます。サーバとして動作する場合、Replication Server は、そのデフォルトのセキュリティ設定に基づいてログインを受け入れたり、拒否したりします。

ログインの認証

セキュリティ・サービスでは、ログイン認証を使用します。

クライアントが認証サービスを要求すると、次のような処理が行われます。

1. クライアントは、セキュリティ・メカニズムを使用してログイン名を検証し、関連するセキュリティ情報を含んだクレデンシャルを受け取ります。
2. クライアントは、そのクレデンシャルをサーバに送信して、安全なコネクションの確立を要求することをサーバに知らせます。
3. サーバは、セキュリティ・メカニズムを使用してクライアントのクレデンシャルを認証します。クレデンシャルが正当なものでない場合、安全なコネクションは拒否されます。
4. サーバは、メッセージ保護のプロパティを検査し、プロパティに互換性があれば、コネクションを確立します。

メッセージ保護

セキュリティ・サービスでは、メッセージ保護を使用します。

現在の Replication Server (クライアント) がデータ保護サービスを要求すると、次の処理が行われます。

1. クライアントは、セキュリティ・メカニズムを使用して、サーバに送信するデータ・パケットを準備します。
   たとえば、クライアントがメッセージの機密保持を要求すると、セキュリティ・メカニズムはリモート・サーバに送信されるコマンドを暗号化します。クライアントが順序不整合の検査を要求すると、セキュリティ・メカニズムは各データ・パケットのタイムスタンプを挿入します。
2. クライアントは、送信先サーバにデータを送信します。
3. サーバは、データを受信すると、セキュリティ・メカニズムを使用して適切な復号化や検証を実行します。
4. サーバは、要求されたセキュリティ処理を実行するためにセキュリティ・メカニズムを使用して、結果をクライアントに返します。たとえば、サーバは結果を暗号化して返したり、クライアントに返すそれぞれのデータ・パケットにタイムスタンプを挿入したりします。

稼働条件と制限事項

ネットワークベース・セキュリティの稼働条件と制限事項について説明します。

ネットワークベース・セキュリティを有効にするには、次が必要になります。

- ネットワーク・セキュリティを有効にするすべてのマシンに、ネットワークベースのセキュリティ・メカニズムがインストールされていること。そのセキュリティ・メカニズムは、Replication Server によってサポートされている必要があります。

  注意： 使用するセキュリティ・メカニズムが、MIT Kerberos、CyberSafe Kerberos、または Transarc DCE であることを確認してください。Sybase のネットワークベース・セキュリティは、他の Kerberos セキュリティ・メカニズムまたは DCE セキュリティ・メカニズムでは機能しません。
- すべてのクライアント Replication Server と送信先 Replication Server が、バージョン 11.5 以降の Replication Server であること。
- すべてのクライアント・データ・サーバと送信先データ・サーバが、Adaptive Server バージョン 12 以降か、互換性のある異機種データ・サーバであること。互換性のある異機種データ・サーバは、Replication Server にインストールされているセキュリティ・メカニズムと set proxy の概念をサポートしていなければなりません。『Replication Server 異機種間複写ガイド』を参照してください。

適用される制限は、次のとおりです。

- 安全な経路の両端 (クライアントとサーバ) は、同じセキュリティ・メカニズムをサポートしていて、セキュリティ・パラメータが同じ機能設定になっている必要があります。
- ユーザ名は、複写システム全体でユニークなものにする必要があります。複数のセキュリティ・システムをサポートしている複写システムで、ユーザ名がユニークであることを保証できない場合は、セキュリティ違反の発生を防ぐために、要求ストアド・プロシージャを無効にする必要があります。

参照：

- ネットワーク・セキュリティの管理 (286 ページ)
- 発生する可能性のあるセキュリティ問題 (289 ページ)

ネットワークベース・セキュリティの設定

ネットワークベース・セキュリティを設定するには、いくつかの手順を実行する必要があります。

1. 必要に応じて設定パラメータと環境変数を変更する。
2. Replication Server のプリンシパル・ユーザを確認する。
3. ネットワークベース・セキュリティ・メカニズムをアクティブ化します。
4. サーバとクライアントを起動します。
5. コネクション、ルート、その他の Replication Server の経路に対してセキュリティ・サービスを設定する。

設定パラメータと環境変数の変更

ネットワーク・セキュリティの設定に必要な設定ファイルについて説明します。

設定ファイルは、インストール時に Sybase ディレクトリ構造内のデフォルトの場所に作成されます。ネットワーク・セキュリティのために必要な設定ファイルは、次のとおりです。

- `libtcl.cfg`
- `objectid.dat`
- `interfaces` ファイル

Kerberos セキュリティ・サービスを使用している場合は、KRB5_KTNAME 環境変数の変更も必要なことがあります。

libtcl.cfg の設定

ドライバとは、外部サービス・プロバイダに対するインタフェースを提供するライブラリです。`libtcl.cfg` ファイルをテンプレートとして使用します。ここ

に、マシンにインストールされているセキュリティ・ドライバについての設定情報をすべて入力してください。

`libtcl.cfg` ファイルの場所は、`$SYBASE/SYBASE_REP/config` ディレクトリ (UNIX) または `%SYBASE%¥ini` ディレクトリ (Windows) です。Sybase ドライバの詳細については、『Open Client/Server 設定ガイド』を参照してください。

セキュリティ・ドライバ・エントリの構文は次のとおりです。

```
provider=driver init-string
```

構文の説明は次のとおりです。

- *provider* は、セキュリティ・メカニズムのローカル名 (dce など) です。グローバル・オブジェクト識別子に対するローカル名のマッピングは、`objectid.dat` に定義されます。
  - DCE セキュリティ・メカニズムのデフォルト・ローカル名は “dce” です。
  - Kerberos セキュリティ・メカニズムのデフォルト・ローカル名は “csfkrb5” です。

  デフォルトではないローカル・メカニズム名を使用する場合は、`objectid.dat` ファイルのローカル名を変更する必要があります。
- *driver* は、セキュリティ・ドライバの名前 (次に例を示します。`libsdce.so`) です。
- *init-string* は、ドライバの初期設定文字列です。
  - DCE ドライバの場合、*init-string* を使用します。ここで、*cell_name* は DCE セルの名前です。

    ```
    secbase=/.:/cell_name
    ```
  - *Kerberos ドライバの場合*、init-string *には次の構文を使用します。ここで*、domaine_name

    ```
    secbase=@domaine_name
    ```

    は Kerberos ドメインの名前です。

サイトの `libtcl.cfg` をカスタマイズするには、テキスト・エディタを使用します。必要のない行の前には “;” (セミコロン) 記号を付けます。パラメータを 1 つ変更するごとに、Replication Server を再起動して変更内容を有効にしてください。

- DCE ドライバのエントリ例を次に示します。

  ```
  [SECURITY]
  dce=libsdce.so secbase=/.:/cell_name
  ```
- Kerberos ドライバのエントリ例を次に示します。

  ```
  [SECURITY]
  csfkrb5=libsybskrb.so
  secbase=@ASE libgss=/krb5/lib/libgss.so
  ```

objectid.dat の設定

`objectid.dat` ファイルは、グローバル・オブジェクト識別子 (OID: object identifier) をローカル名にマップします。

ファイルの場所は、`$SYBASE/config` ディレクトリ (UNIX) または `%SYBASE%/ini` ディレクトリ (Windows) です。`objectid.dat` ファイルを編集する必要が生じるのは、`libtcl.cfg` ファイルでセキュリティ・サービスのローカル名を変更した場合のみです。

- DCE の `objectid.dat` ファイルでのエントリ例を次に示します。

```
[secmech]
1.3.6.1.4.1.897.4.6.1 = dce, dcesecmech
```

- Kerberos の `objectid.dat` ファイルでのエントリ例を次に示します。

```
[secmech]
1.3.6.1.4.1.897.4.6.6 = csfkrb5, kerberos
```

interfaces ファイルの設定

`interfaces` ファイルには、サーバのネットワーク情報とセキュリティ情報が入っています。

`interfaces` ファイルの場所は、`$SYBASE/SYBASE_REP/interfaces` (UNIX) または `%SYBASE%¥ini¥sql.ini` (Windows) です。ネットワーク・セキュリティを使用する場合は、`secmech` 行を含める必要があります。これにより、サポートされるセキュリティ・サービスのグローバル識別子が指定されます。サポートされるセキュリティ・メカニズムは、その OID によってリストされます。セキュリティ・メカニズムが複数ある場合は、カンマで区切ります。

DCE または Kerberos の `interfaces` ファイルのエントリ例を次に示します。ここで、*server_principal_user_name* は Replication Server のプリンシパル・ユーザ名です。

```
#
server_principal_user_name
        query tcp ether plum 1050
        master tcp ether plum 1050
        secmech 1.3.6.1.4.1.897.4.6.1
```

参照：

- プリンシパル・ユーザの識別 (269 ページ)

環境変数の設定 (Kerberos)

Replication Server がクライアントとして機能する場合は、KRB5CCANME を設定してクレデンシャル・キャッシュのロケーションを示します。

クレデンシャルは、TGT (Ticket Granted Ticket) とチケットのセッション・キーの組み合わせです。Replication Server は、リモート・サーバへのログイン時にこのクレデンシャルを使用して自らを証明します。

Kerberos ネットワーク・セキュリティを使用している場合は、共有ライブラリ・パスと KRB5_KTNAME 環境変数の再設定が必要になることがあります。

- 共有ライブラリ・パスに指定したディレクトリに、共有ライブラリ・ファイルがあることを確認してください。これは、実行時にクライアントが共有ライブラリ・ファイルを見つけられるようにするためです。共有ライブラリ・ファイルには次のものがあります。
  - `libgss.so` (Sun Solaris の場合)
  - `libgss.sl` (HP-UX の場合)
- サーバの keytab ファイルが Kerberos のシステム・デフォルト以外の場所にある場合は、KRB5_KTNAME 環境変数を keytab ファイルのフル・パス名に設定します。
- LD-LIBRARY_PATH 環境変数に、Adaptive Server、Open Client/Server、Replication Server の `lib` ディレクトリのパスと、CyberSafe の `lib` ディレクトリのパスが入っていることを確認します。
- 同様に、PATH 環境変数に、CyberSafe `bin` ディレクトリのパスと Adaptive Server、Open Client/Server、Replication Server の `bin` ディレクトリのパスが入っていることを確認します。

プリンシパル・ユーザの設定

ネットワークベース・セキュリティで統一化ログインが有効になっている場合は、Replication Server はプリンシパル・ユーザとしてリモート・サーバにログインする必要があります。

ネットワーク・セキュリティが有効でない場合、Replication Server は利用可能なユーザ名のいずれかを、実行するタスクに応じて使用してリモート・サーバにログインします。プリンシパル・ユーザのクレデンシャルは、ネットワーク・セキュリティがアクティブのときに、Replication Server が他のプロセスにログインするために持つ、唯一のクレデンシャルです。

Replication Server が別の Replication Server またはデータ・サーバにログインすると、クレデンシャルに含まれているプリンシパル・ユーザ名がサーバのネーム・スペースにマップされて、安全なコネクションが確立されます。

注意： プリンシパル・ユーザ名は、ユニークなものにしてください。Replication Server は、同じ名前の別のサーバにはログインできません。

Replication Server は、リモート・サーバで (プリンシパル・ユーザとして) **set proxy** コマンドを実行し、現在のタスクに適したユーザに切り替わります。

プリンシパル・ユーザの識別

Replication Server にログインするときや、Replication Server を起動するときに、**-S** フラグを使用してプリンシパル・ユーザ名を指定できます。

各 Replication Server に対してプリンシパル・ユーザを設定するのは、複写システム管理者の仕事です。プリンシパル・ユーザ名には Replication Server 名を使用することをおすすめします。

-S フラグでプリンシパル・ユーザ名を指定しなかった場合、Replication Server は Replication Server 名を使用します。

セキュリティ・メカニズムに対するプリンシパル・ユーザの識別

セキュリティ・メカニズムのセキュリティ管理者は、セキュリティ・メカニズムに対して Replication Server プリンシパル名を定義しなければなりません。

DCE の場合：

- DCE の **dcecp** ツールの **user create** コマンドを使用してプリンシパル・ユーザを作成します。
  DCE にサーバを定義する場合は、新しいプリンシパル・ユーザがサーバとして動作できるように指定するオプションを使用します。
- **keytab create** コマンド (**dcecp** ユーティリティ) を使用して DCE keytab ファイルを作成します。このファイルには、プリンシパル・ユーザのパスワードが暗号化された形式で格納されます。

注意： DCE は UNIX ではサポートされていません。

CyberSafe Kerberos の場合：

- Kerberos の **csfadml** ツールを使用して、プリンシパル・ユーザを作成します。
- **csfadml** を使用してキー・テーブル・ファイルを抽出します。

MIT Kerberos の場合：

- 管理コマンド **addprinc** を使用して、プリンシパル・ユーザを作成します。
- 管理コマンド **ktadd** を使用してキー・テーブル・ファイルを抽出します。

セキュリティ・メカニズムにサーバとユーザを識別させる方法の詳細については、セキュリティ・メカニズム・プロバイダのマニュアルを参照してください。

Replication Server に対するプリンシパル・ユーザの識別

システム・セキュリティと統一化ログインを使用して Replication Server に接続する他のプロセス (RepAgent、データ・サーバ、他の Replication Server など) のプリ

ンシパル・ユーザを、現在の Replication Server の `rs_users` テーブルで識別する必要があります。**create user** コマンドを使用すると、プリンシパル・ユーザ名を `rs_users` に追加できます。

複写システムへの Replication Server プリンシパル・ユーザの識別
Replication Server が統一化ログインを使用して接続する ID サーバと RSSD を含む送信先プロセス (Replication Server とデータ・サーバ) に、Replication Server のプリンシパル・ユーザ名を追加する必要があります。

Adaptive Server にログイン名を追加する方法については、『Adaptive Server Enterprise システム管理ガイド』を参照してください。

ネットワークベース・セキュリティのアクティブ化

セキュリティ・サービスを設定する前に、**configure replication server** コマンドを使用して、Replication Server のネットワークベース・セキュリティをアクティブ化します。

1. Replication Server にログインして次のように入力します。

   ```
   configure replication server
     set use_security_services to 'on'
   ```

2. Replication Server を停止します。
3. **repserver** コマンドまたは Replication Server 実行ファイルを実行して Replication Server を再起動します。
   - DCE セキュリティ・メカニズムを使用している場合は、**-K** フラグで keytab ファイルのロケーションを指定してください。
   - Kerberos セキュリティ・メカニズムを使用している場合は、KRB5_KTNAME 環境変数 (UNIX) またはキー・テーブル・レジストリ・キー・エントリ (Windows 2000、2003) で、keytab ファイルのロケーションを指定してください。

   repserver コマンドの構文とその他の情報については、『Replication Server リファレンス・マニュアル』を参照してください。

参照：
- ネットワークベース・セキュリティの無効化 (286 ページ)

サーバとクライアントの起動

ネットワーク・セキュリティ環境を正常に機能させるには、サーバとクライアントを起動する前に、必ず正当なクレデンシャルを付与してください。

Kerberos システムの場合は次のようにします。

- UNIX システムでは、**kinit** の実行後にサーバとクライアントを起動します。
- Windows NT システムでは、サーバとクライアントをシングル・サインオン機能を使用して自動的に起動するか、Kerberos のクレデンシャル管理機能を使用して手動で起動します。

詳細については、Kerberos のマニュアルを参照してください。

Transarc DCE システムの場合も同様です。適切な環境の設定については、Transarc のマニュアルを参照してください。

Replication Server のセキュリティ・サービスの設定

ネットワークベース・セキュリティ機能を設定するための Replication Server パラメータについて説明します。

設定パラメータで、次のことを指定できます。

- 統一化ログイン
- 相互認証
- サポートされているセキュリティ・メカニズムの選択
- 暗号化によるメッセージの機密保持
- 他の安全なメッセージ転送機能。メッセージの整合性、オリジン検査、リプレイの検出、順序不整合の検出など

注意： 選択したセキュリティ・メカニズムによっては、これらのセキュリティ機能の一部がサイトで利用できない場合があります。

システム設定時には、**rs_init** プログラムでデフォルトのパラメータを設定します。**rs_init** については、使用しているプラットフォームの『Replication Server 設定ガイド』を参照してください。この項では、これらのパラメータをコマンド・ラインで設定する方法について説明します。

Replication Server の経路の識別

Replication Server は、ローカル・データ・サーバのデータ複写アクティビティを調整し、他のサイトの Replication Server やデータ・サーバとデータ交換を行います。ネットワークベース・セキュリティを設定できる Replication Server の経路について説明します。

- Replication Server がクライアントとして動作している場合、次に対してセキュリティを設定できます。
  - Replication Server が他のサーバにログインするときに確立されるすべての経路。これらは、デフォルトのグローバル設定です。
  - RSSD へのコネクション
  - 個々のコネクション
  - 個々のルート

  - Replication Server から ID サーバへの経路
  - ルートの作成、サブスクリプションの作成、サブスクリプションの削除に使用される経路
- Replication Server がサーバとして動作している場合、次に対してセキュリティを設定できます。
  - すべての受信ログイン。これらは、デフォルトのグローバル設定です。
  - Replication Server へのユーザ・コネクション (ログオン時に設定)

さまざまな種類のネットワーク経路の保護
送信ログインのためのさまざまな種類の経路にセキュリティを設定します。

表 17 : ネットワーク経路

経路	セキュリティ設定方法	特殊なパラメータと例外
現在の Replication Server (クライアントとして動作) を始点とするすべての経路	**configure replication server** を使用して、グローバル・セキュリティ・パラメータを設定する。個々の経路について特に別の値で上書きしないかぎり、これがすべての送信ログインに対するデフォルト設定となる。	**use_security_services** を使用すると、すべてのネットワーク・セキュリティを 1 つのコマンドでオフにできる。
RSSD へのコネクション	テキスト・エディタを使用して `rs_config` ファイルを設定する。	セキュリティ・パラメータは、“RSSD_” プレフィクスで始まる。例：**RSSD_unified_login**
個々のコネクション	次のいずれかのコマンドで、リモート・データベースへのコネクションにセキュリティ・パラメータを設定する。 • **create connection** • **alter connection** これらのコマンドの詳細については、『Replication Server リファレンス・マニュアル』を参照。	要求ストアド・プロシージャをサスペンドするには、**dsi_exec_request_sproc** を使用する。

経路	セキュリティ設定方法	特殊なパラメータと例外
create route コマンドを使用して定義された個々のルート	次のいずれかのコマンドで、セキュリティ・パラメータを設定する。 • **create route** • **alter route** これらのコマンドの詳細については、『Replication Server リファレンス・マニュアル』を参照。	
Replication Server から ID サーバ	**configure replication server** を使用して、セキュリティ・パラメータを設定する。 このコマンドの詳細については、『Replication Server リファレンス・マニュアル』を参照。	セキュリティ・パラメータは “id-” プレフィクスで始まる。例：**id_msg_confidentiality**
次の目的での Replication Server からプライマリ Replication Server およびプライマリ・データベースへの経路 • ルートの作成 • サブスクリプションの作成または削除	ルートの作成、サブスクリプションの作成または削除を行うユーザが Replication Server にログインするときに使用したセキュリティ設定が、Replication Server によって複製される。	
すべての受信ログイン (Replication Server はサーバとして動作)	**configure replication server** を使用して、受信ログインのパラメータを設定する。送信ログインのデフォルト・パラメータと受信ログインのパラメータが、同時に同一に設定される。	

経路	セキュリティ設定方法	特殊なパラメータと例外
ユーザが Replication Server にログインするときに確立される経路	**isql** ユーティリティを使用してセキュリティ・パラメータを設定する。	この経路に設定されるセキュリティ・パラメータは、Replication Server ですべての受信ログイン用に設定されたセキュリティ・パラメータとの互換性が必要である。 この経路のセキュリティは、**rs_init** ユーティリティでは設定できない。

参照：

- データベース・コネクションに対するセキュリティ設定 (278 ページ)
- 他の経路の安全を確保するための Replication Server でのセキュリティ設定の流用 (285 ページ)
- ネットワークベース・セキュリティの無効化 (286 ページ)

セキュリティ設定パラメータ

Replicaton Server には、通常、すべての経路で使用できるセキュリティ設定パラメータがあります。

表 18 : Replication Server に影響を及ぼすセキュリティ・パラメータ

configuration_parameter	説明
msg_confidentiality	Replication Server が暗号化データを送受信するかどうかを示す。“required” に設定すると、送信データと受信データの暗号化が必要になる。“not_required” に設定すると、Replication Server は、送信されてくる暗号化データも非暗号化データも受け入れる。指定できる値は “required” または “not_required”。 デフォルト値は not_required
msg_integrity	データの改ざんに対するチェックを行うかどうかを示す。指定できる値は “required” または “not_required”。 デフォルト値は not_required
msg_origin_check	データの送信元を確認するかどうかを示す。指定できる値は “required” または “not_required”。 デフォルト値は not_required

configuration_parameter	説明
msg_replay_detection	データが傍受および再送されていないことを確認するために、データをチェックするかどうかを示す。指定できる値は “required” または “not_required”。 デフォルト値は not_required
msg_sequence_check	データ・パッケージが送信された順序どおりに受信されたかを検査するかどうかを示す。指定できる値は “required” または “not_required”。 デフォルト値は not_required
mutual_auth	コネクションを確立する前に認証を行うように、リモート・サーバに指示する。指定できる値は “required” または “not_required”。 デフォルト値は not_required
security_mechanism	ネットワークベース・セキュリティ・メカニズムの名前を指定する。 デフォルト値は `libtcl.cfg` にリストされる最初のセキュリティ・メカニズム
unified_login	Replication Server が送信ログインを探し、受信ログインを受け入れる方法を示す。値は、次のとおり。 • “required” ― 常にクレデンシャルを使用してリモート・サーバにログインしようとする。クレデンシャルによる受信ログインだけを受け入れる。 • “not_required” ― 常にパスワードを使用してリモート・サーバにログインしようとする。受信ログインは、クレデンシャルとパスワードのどちらを使用していても受け入れる。 **注意：** 先に **unified_login** を “required” にしないと、他のセキュリティ・パラメータは有効にならない。 デフォルト値は not_required

互換性のある設定の計画

ネットワークベース・セキュリティを設定するときは、安全な経路の両端で設定されているセキュリティ機能間の対話について計画する必要があります。各経路の両端のセキュリティ設定には、互換性が必要です。

Replication Server は、受信ログインを受け入れ、他のサーバへのログインを開始します。すべての受信ログイン用のセキュリティ・パラメータは (Replication Server がサーバとして動作しているときに)、**configure replication server** コマンドを使用して設定します。ネットワーク経路に対して設定し、(Replication Server がクライ

アントとして動作する場合は) 送信ログインに対して設定するセキュリティ・パラメータもあります。

注意： 各サーバについて、セキュリティ機能の選択と設定を行うのは複写システム管理者の仕事です。Replication Server は、リモート・サーバへの経路を確立する前に、そのサーバのセキュリティ機能の問い合わせを行いません。経路の両端のセキュリティ機能に互換性がないと、ログインは失敗します。

configure replication server コマンドに **use_security_services** パラメータを使用して実行すると、クライアント／サーバ間の対話で互換性があるすべてのセキュリティ設定のアクティブ化／非アクティブ化が可能です。セキュリティ・サービスのパラメータに互換性がない場合、サーバはクライアントのログインを許可しません。たとえば、パラメータがクライアント側では “not_required”、サーバ側では “required” に設定されている場合などです。

参照：
- さまざまな種類のネットワーク経路の保護 (272 ページ)

互換性のあるクライアント／サーバ設定
クライアントの設定と互換性があるサーバの設定を使用します。

表 19 : 互換性のあるクライアント／サーバ設定

クライアント	サーバ
use_security_services が “off”：セキュリティ・サービスを使用しない	互換性のある設定は次のとおり。 • **use_security_services** が “off”、または • **use_security_services** が “on”、セキュリティ機能が “not required”
use_security_services が “on”、セキュリティ機能が “not required”	互換性のある設定は次のとおり。 • **use_security_services** が “on”、セキュリティ機能が “not required”、または • **use_security_services** が “off”
use_security_services が “on”、セキュリティ機能が “required”	互換性のある設定は次のとおり。 • **use_security_services** が “on”、セキュリティ機能が “required”

デフォルト値の設定
すべての送信ログイン (Replication Server がクライアントとして動作しているとき) と受信ログイン (Replication Server がサーバとして動作しているとき) に、セキュリティのデフォルトを設定するには、**configure replication server** を使用します。

次の送信経路については、デフォルトのセキュリティ設定を無効にすることができます。

- 個々のコネクション
- 個々のルート
- Replication Server から ID サーバへの経路

注意： 受信ログインのセキュリティを制御するデフォルトのセキュリティ設定は、無効にはできません。

Replication Server は、他のサーバへの経路をオープンしようとするとき、その経路用にセキュリティ・パラメータが設定されているかどうかを検査します。設定されていない場合、Replication Server は、**configure replication server** を使って指定されたデフォルトのセキュリティ設定を使用します。

グローバル・セキュリティ・パラメータを設定するには、Replication Server にログインし、**configure replication server** を **isql** プロンプトで実行します。構文は次のとおりです。

```
configure replication server {
  set security_mechanism to 'mechanism_name' |
  set security_parameter to { 'required' |
    'not_required' }}
```

Replication Server に影響を及ぼす、すべてのセキュリティ設定パラメータを設定できます。これらのパラメータは、RSSD の `rs_config` テーブルに格納されます。これらのパラメータを設定するには **sa** パーミッションが必要です。

統一化ログインの要求

Replication Server に接続するすべてのサーバとユーザに対して、セキュリティ・メカニズムによる認証を要求するには、**unified_login** を 'required' に設定します。

Replication Server にログインし、**isql** プロンプトで次のコマンドを実行します。

```
configure replication server
  set unified_login to 'required'
```

unified_login を 'not_required' にすると、サーバとユーザはクレデンシャルかパスワードのどちらかで Replication Server に接続できるようになります。

注意： 他のセキュリティ・サービスを有効にするには、**unified_login** を 'required' にする必要があります。

データ暗号化の要求

Replication Server が送信または受信するすべてのデータを暗号化するには、**msg_encryption** を 'required' に設定します。

Replication Server にログインし、**isql** プロンプトで次のコマンドを実行します。

```
configure replication server
  set msg_encryption to 'required'
```

参照：
- データベース・コネクションに対するセキュリティ設定 (278 ページ)
- ルートのセキュリティ設定 (280 ページ)
- ID サーバに対するセキュリティの設定 (282 ページ)

RSSD へのコネクションに対するセキュリティ設定
起動時に、Replication Server は RSSD に接続して設定情報を確認します。ネットワークベース・セキュリティを使用して、Replicatin Server から RSSD への経路を保護します。

Replication Server の設定時に、**rs_init** が RSSD コネクションを作成し、Replication Server 設定ファイルの *Rep_Server_name.cfg* にデフォルトのセキュリティ情報を格納します。デフォルトでは、**rs_init** はすべてのネットワーク・セキュリティ・パラメータを “not required” に設定します。この経路を保護するには、テキスト・エディタを使用して目的のデフォルト値を “required” に設定する必要があります。

RSSD の設定値は、“RSSD_” プレフィクスで始まります。次に例を示します。

- **RSSD_mutual_auth**
- **RSSD_msg_origin_check**

参照：
- セキュリティ設定パラメータ (274 ページ)

データベース・コネクションに対するセキュリティ設定
個々のコネクションに対してセキュリティを設定するには、**create connection** または **alter connection** を使用します。

これらのコマンドで設定されたセキュリティ・パラメータは、データ・サーバへの送信コネクションのセキュリティに影響を与えます。これらのコマンドは、**configure replication server** で設定されたパラメータを上書きします。

安全なコネクションの作成
create connection でコネクションを作成するときに、セキュリティ・パラメータを設定できます。

通常、**create connection** は、コネクションを Sybase 以外のデータベースに追加する場合に使用します。

次に、**create connection** コマンドでセキュリティ機能を指定する構文を示します。『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の

「**create connection**」には、**create connection** の使用方法の詳細が記載されています。

```
create connection to data_server.database...
  set username [to] user
  [set password [to] passwd]
  [set security_mechanism [to] 'mechanism_name' |
  set dsi_exec_request_sproc [to] { 'on' | 'off' } |
  set security_mechanism [to] 'mechanism_name' |
  set security_parameter [to] { 'required' |
    'not_required' } ]
```

create connection で設定できるセキュリティ設定パラメータに加えて、**dsi_exec_request_sproc** という特別なセキュリティ・パラメータも設定できます。

コネクション・パラメータは、RSSD 内の `rs_config` テーブルに格納されます。これらのパラメータを設定するには **sa** パーミッションが必要です。

コネクションの両端に設定されるセキュリティ・パラメータには、互換性が必要です。

参照：
- セキュリティ設定パラメータ (274 ページ)

コネクション用の特別なセキュリティ・パラメータ
特別なパラメータを使用してコネクションを保護できます。

表 20 : コネクション用の特別なセキュリティ・パラメータ

security_parameter	**説明**
dsi_exec_request_sproc	プライマリ Replication Server で要求ストアド・プロシージャが “off” か “on” かを示す。複数のセキュリティ・システムを持つ環境で使用する。 デフォルト値は off

参照：
- 互換性のある設定の計画 (275 ページ)
- 複数のセキュリティ・メカニズムの使用 (289 ページ)

コネクションのセキュリティ変更
データベース・コネクションのセキュリティ設定を変更するには、**alter connection** を使用します。

構文：

```
alter connection to data_server.database {
...
```

```
set password to passwd |
set security_mechanism to 'mechanism_name' |
set dsi_exec_request_sproc to { 'on' | 'off' } |
set security_parameter to { 'required' |
    'not_required' }}
```

データベース・コネクションのセキュリティ・パラメータを変更するには、Replication Server で次のことを行います。

1. **suspend connection** を実行して、コネクションに関するアクティビティをサスペンドします。
2. **alter connection** を実行して、セキュリティ・パラメータを変更します。

   注意： パラメータは一度に 1 つずつ設定します。
3. **resume connection** を実行して、コネクションに関するアクティビティを再開します。

alter connection を使用したセキュリティ設定の例

- Replication Server がターゲット・データベース (`TOKYO_DS.pubs2`) にクレデンシャルを使って接続するようにするには、次のコマンドを実行します。

  ```
  alter connection to TOKYO_DS.pubs2
     set unified_login to 'required'
  ```

  注意： 他のセキュリティ・サービスを有効にするには、**unified_login** を “required” にする必要があります。
- 複数のセキュリティ・システムを持つ環境の TOKYO データ・サーバで、要求ストアド・プロシージャを “off” に切り替えるには、次のコマンドを実行します。

  ```
  alter connection to TOKYO_DS.pubs2
     set dsi_exec_request_sproc to 'off'
  ```

参照：
- セキュリティ設定パラメータ (274 ページ)
- コネクション用の特別なセキュリティ・パラメータ (279 ページ)

ルートのセキュリティ設定

create route または **alter route** を使用して、個々のルートに対してセキュリティを設定します。

このコマンドで設定されたセキュリティ・パラメータは、送信先 Replication Server への送信ログインに対するセキュリティに影響します。このコマンドを実行すると、**configure replication server** で設定されたデフォルトのパラメータが無効になります。

安全なルートの作成
create route でルートを作成するときに、セキュリティ・パラメータを設定できます。

次に、**create route** コマンドでセキュリティ機能を指定する構文を示します。

```
create route to dest_replication_server {
...
[set username to 'user' ]
[set password to 'passwd' ]
[set security_mechanism to 'mechanism_name' |
set security_parameter to { 'required' |
    'not_required' } ]
```

これらのパラメータは、RSSD の `rs_config` テーブルに格納されます。これらのパラメータを設定するには **sa** パーミッションが必要です。

ルートの両端に設定されるセキュリティ・パラメータには、互換性が必要です。

参照：
- 互換性のある設定の計画 (275 ページ)
- セキュリティ設定パラメータ (274 ページ)

ルートのセキュリティ変更
ルートのセキュリティ設定を変更するには、**alter route** を使用します。

次に、セキュリティを変更するための構文を示します。

```
alter route to dest_replication_server {
...
set password to 'passwd' |
set security_mechanism to 'mechanism_name' |
set security_parameter to { 'required' |
        'not_required' }}
```

ルートのセキュリティ・パラメータを変更するには、Replication Server にログインし、次のことを行います。

1. **suspend route** を実行して、ルートに関するアクティビティをサスペンドします。
2. **alter route** を実行して、セキュリティ・パラメータを変更します。

   注意： パラメータは一度に 1 つずつ設定します。
3. **resume route** を実行して、ルートに関するアクティビティをレジュームします。

alter route を使用したセキュリティ設定の例

- Replication Server がターゲット Replication Server (TOKYO_RS) にパスワードを使って接続するようにするには、次のコマンドを実行します。

```
alter route to TOKYO_RS
   set username 'TOKYO_rsi_user'
```

```
alter route to TOKYO_RS
   set password 'TOKYO_rsi_pw'
```

```
alter route to TOKYO_RS
   set unified_login to 'not_required'
```

注意： **unified_login** が “not_required” の場合、RSI ユーザとパスワードを指定する必要があります。

- ターゲット Replication Server (TOKYO_RS) と交換するすべてのメッセージについて、不正変更の有無をチェックするかどうかを指定するには、次のコマンドを実行します。

```
alter route to TOKYO_RS
  set msg_integrity to 'required'
```

参照：

- セキュリティ設定パラメータ (274 ページ)

ID サーバに対するセキュリティの設定

Replication Server から ID サーバへのネットワーク・コネクションに対して、ネットワークベース・セキュリティを設定するには、**configure replication server** コマンドを使用します。

構文を以下に示します。

```
configure replication server
  set id_security_param to { 'required' |
    'not_required' }
```

『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**configure replication server**」を参照してください。セキュリティ・パラメータは、RSSD の `rs_config` テーブルに格納されます。これらのパラメータを設定するには、**sa** パーミッションが必要です。この経路に対する設定であることを識別できるように、ID サーバのパラメータには、“id_” プレフィクスが付いています。次に例を示します。

- **id_msg_confidentiality**
- **id_security_mechanism**

configure replication server で設定される ID サーバのセキュリティ・パラメータは動的です。これらのパラメータは設定後すぐに有効になり、Replication Server を再起動する必要はありません。

configure replication server を使用した ID サーバに対するセキュリティの設定の例

- すべてのメッセージの送信元を確認するように設定するには、送信元 Replication Server にログインして次のように入力します。

```
configure replication server
   set id_msg_origin_check 'required'
```

- Replication Server がクレデンシャルを使用して ID サーバにログインするように設定するには、次のように入力します。

```
configure replication server
  set id_unified_login to 'required'
```

参照：

- セキュリティ設定パラメータ (274 ページ)

Replication Server へのログイン

Replication Server に接続するには、**isql** などのクライアント・アプリケーションや、Open Client Client-Library で作成したカスタム・アプリケーション・プログラムを使用します。

セキュリティを有効にする isql コマンド・ライン・オプション

isql とともに使用して Replication Server とのコネクションに対するネットワークベース・セキュリティ・サービスを有効にするコマンド・ライン・オプションがいくつかあります。

表 21 : セキュリティを有効にする isql コマンド・ライン・オプション

オプション名	意味
-K *keytab_file*	DCE セキュリティだけで使用する。サーバにログインしているユーザのセキュリティ・キーを含んでいる DCE keytab ファイルを指定する。keytab ファイルは、DCE **dcecp** ユーティリティを使用して作成できる。詳細については、DCE のマニュアルを参照。Replication Server は、このファイルに対する読み込みパーミッションを持っている必要がある。 **注意：** Kerberos ユーザの場合：キー・テーブルのレジストリ・キー・エントリ (Windows 2000、2003) を使用して keytab ファイルのロケーションを指定する。
-S *server_name*	サーバのネットワーク名を指定する。統一化ログインが有効な場合は、このオプションはプリンシパル・ユーザも指定する。

オプション名	意味
-V *security_options*	統一化ログインを指定する。このオプションを使用する場合は、ネットワークのセキュリティ・システムにログインしてから **isql** ユーティリティを実行する必要がある。**-U** オプションを指定する場合は、セキュリティ・メカニズムに認識されるネットワーク・ユーザ名を指定する必要がある。**-P** オプションで指定されたパスワードは無視される。 **-V** のあとには、追加のセキュリティ・サービスを有効にするオプションの文字列を指定する。オプションと、それによって有効にされるサービスは次のとおり。 • **c** – データの機密保持 • **i** – データの整合性 • **m** – 相互認証 • **o** – データ・オリジン・スタンピング・サービス • **r** – データのリプレイ検出 • **q** – 順序不整合の検出
-X	コネクションを暗号化パスワード付きで指定する。
-Z *security_mechanism*	Replication Server へのコネクションに使用するセキュリティ・メカニズムの名前を指定する。 サポートされるセキュリティ・メカニズム名は `libtcl.cfg` ファイルにリストされている。セキュリティ・メカニズムの指定がない場合は、デフォルトが使用される。デフォルトは、`libtcl.cfg` の SECURITY の下で最初にリストされているセキュリティ・メカニズムである。

Replication Server への接続例

Replication Server へは、セキュリティ・メカニズムにログインしてから Replication Server にログインするか、Replication Server に直接ログインすることによって接続できます。

プリンシパル・ユーザを識別するには **-S** フラグを指定する必要があります。

セキュリティ・メカニズムから Replication Server への接続

最初に DCE セキュリティ・メカニズムにログインしてから Replication Server にログイン方法の例を次に示します。

注意： DCE を使用している場合、他のユーザとしてログインするには、**-U** オプションと **-K** オプションを指定する必要があります。

1. DCE セキュリティ・メカニズムにログインしてクレデンシャルを受け取ります。
   - DCE の場合、次のように入力します。
     ```
     dce_login user_name password
     ```
   - Kerberos の場合、次のように入力します。
     ```
     kinit user_name password
     ```
2. **isql** を使用して Replication Server にログインします。
   - DCE の場合、次のように入力します。
     ```
     isql -Srs_server_name -Vsecurity_option
     ```
   - Kerberos の場合、次のように入力します。
     ```
     isql -Srs_server_name -Vsecurity_option
     ```

セキュリティの外部からの Replication Server への接続
セキュリティ・メカニズムの外部から Replication Server に接続する例を次に示します。

次のように入力します。

- DCE の場合：
  ```
  isql -Srs_server_name -Uuser_name
       -Kkeytab_file
  ```
- Kerberos の場合：
  ```
  isql -Srs-server_name -Yuser_name
  ```

他の経路の安全を確保するための Replication Server でのセキュリティ設定の流用
コマンド・ラインで Replication Server にログインするときに使用するセキュリティ・サービスは、クライアント／サーバ間の経路を安全に保つだけではありません。このセキュリティ・サービスは、後で Replication Server が他の経路をオープンするときに、セッション内で複製することもできます。

次のコマンドを実行すると、Replication Server は内部的な経路によってプライマリ Replication Server とプライマリ・データベースにログインします。

- **create subscription**
- **drop subscription**
- **create route**

これらの経路を安全なものにするために、Replication Server は、**create subscription**、**drop subscription**、または **create route** を実行するユーザが Replication Server へのログイン時に入力したセキュリティ設定を流用します。

ネットワーク・セキュリティの管理

ネットワーク・セキュリティの管理と維持について説明します。

set proxy を使用したログインの切り替え

現在のユーザがターゲット・データ・サーバで Replication Server コマンドを実行できるようにするには、**set proxy** を使用します。

unified_login が有効になっている場合、Replication Server は常にプリンシパル・ユーザとしてリモート・プロセスにログインします。しかし、Replication Server コマンドは、そのオペレーションの権限を持つユーザが、ターゲット・データ・サーバで実行する必要があります。たとえば、レプリケート・データベースに変更を適用するときは、メンテナンス・ユーザのログイン名を使用する必要があります。Replication Server は、Adaptive Server の **set proxy** コマンドを使用して、ログイン・ユーザから必要とされるユーザに自動的に切り替わります。

set proxy コマンドは、**rs_setproxy** ファンクション文字列を使用してカスタマイズできます。**rs_setproxy** は、データ・サーバでログイン名を変更します。**rs_setproxy** には、ファンクション文字列クラス・スコープがあります。

『Replication Server リファレンス・マニュアル』の「Replication Server システム・ファンクション」の「**rs_setproxy**」と、『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**set proxy**」を参照してください。

ネットワークベース・セキュリティの無効化

すべてのセキュリティ・サービスを無効にするには、**use_security_services to 'off'** パラメータを使用します。

ネットワーク・セキュリティを無効にすると、Replication Server はセキュリティ・クレデンシャルを使用した受信ログインを受け入れなくなり、他のプロセスにログインするときにセキュリティ・クレデンシャルを使用しなくなります。すべてのセキュリティ・サービスはアクティブでなくなります。

1. Replication Server にログインし、**isql** プロンプトで次のコマンドを入力します。

   ```
   configure replication server
       set use_security_services to 'off'
   ```

2. **repserver** コマンドまたは Replication Server 実行ファイルを実行して Replication Server を再起動します。

参照：

- Replication Server 実行プログラム (90 ページ)
- ネットワークベース・セキュリティの設定 (265 ページ)

セキュリティ・メカニズムの変更

別のセキュリティ・メカニズムに変更するには、**set security_mechanism** パラメータを使用します。

Replication Server にログインし、**isql** プロンプトで次のコマンドを実行します。

```
configure replication server
   set security_mechanism to 'mechanism_name'
```

切り替えて新たに使用するセキュリティ・メカニズムは、インストール済みで、`libtcl.cfg` ファイルの SECURITY セクションにリストされていなければなりません。

ターゲットごとの値のデフォルト値へのリセット

set security_services to 'default' オプションを使用すると、コネクションやルートに対するターゲットごとのセキュリティ値を一度に変更できます。

コネクションのターゲットごとの値のリセット

ターゲットごとのセキュリティ設定を **configure replication server** で設定されるグローバル値に変更するには、**alter connection** を使用します。

変更は、コネクションをレジュームしたあとに有効になります。この手順は、**use_security_services** の設定には影響を及ぼしません。

1. Replication Server にログインし、**suspend connection** を **isql** プロンプトで実行します。次のように入力します。

   ```
   suspend connection to dataserver.database
   ```

2. **alter connection** でセキュリティ設定を変更します。次のように入力します。

   ```
   alter connection to dataserver.database
       set security_services to 'default'
   ```

3. **resume connection** を使用して、ルートまたはコネクションを再開し、変更内容を有効にします。次のように入力します。

   ```
   resume connection to dataserver.database
   ```

ルートのターゲットごとの値のリセット

set security_services to 'default' パラメータを **alter route** とともに使用して、ターゲットごとのセキュリティ設定を **configure replication server** で設定されるグローバル値に変更します。

変更は、ルートをレジュームしたあとに有効になります。この手順は、**use_security_services** の設定には影響を及ぼしません。

1. ルートをサスペンドします。次のように入力します。

```
suspend route to dest_replication_server
```

2. ルートを変更します。次のように入力します。

```
alter route to dest_replication_server
  set security_services to 'default'
```

3. ルートをレジュームします。次のように入力します。

```
resume route to dest_replication_server
```

セキュリティ・サービスについての情報の表示

Replication Server のネットワークベース・セキュリティについての情報を表示するには、**admin security_property** コマンドと **admin security_setting** コマンドを使用します。

使用できるセキュリティ・メカニズムとセキュリティ・サービスを調べる

Replication Server でどのセキュリティ・メカニズムとセキュリティ・サービスがサポートされているのかを調べるには、**admin security_property** を使用します。

```
admin security_property[, security_mechanism]
```

Replication Server は、サポートされているセキュリティ・メカニズムの名前、そのメカニズムで使用可能なセキュリティ・サービス、それらのサービスが各サイトでサポートされているかどうかを表示します。

現在のセキュリティ設定を調べる

サポートされているセキュリティ・サービスのステータスを調べるには、**admin security_setting** を使用します。

ルートや ID サーバに設定されているセキュリティ・パラメータのステータス情報を表示できます。次の構文を使用します。ここで、*rs_idserver* は ID サーバの名前であり、*rep_server* は送信先 Replication Server の名前です。

```
admin security_setting[, rs_idserver |, rep_server ]
```

Replication Server での Replication Server ログインへのセキュリティ・システム・ログインのマップ

Replication Server で、セキュリティ・システム・ログインを Replication Server ログインにマップし、無効文字に変換する方法について説明します。

ネットワークベース・セキュリティ・メカニズムは、Replication Server で有効ではないログイン名を使用できます。たとえば、Replication Server のログイン名は 30 文字を超えてはならず、*、(、% などの特殊文字を含むことはできません。Replication Server におけるログイン名は、有効な識別子でなければなりません。これについては、『Replication Server リファレンス・マニュアル』の「トピック」の「識別子」を参照してください。

セキュリティ・ログイン名が有効な識別子でない場合、Replication Server は無効文字を自動的に有効な文字にマップし、ログイン名を 30 文字にトランケートします。

Replication Server による無効文字の変換

Replication Server が無効文字を変換する際の、無効文字の有効文字へのマップについて説明します。

表 22 : 無効文字の有効文字への変換

不正な文字	変換後
¥ % & , := > ‘ ' ~	アンダースコア_
!^ ( ) .< ?{ }	ドル記号$
“ - ; * + / [ ] |	ポンド記号#

複数のセキュリティ・メカニズムの使用

複数のセキュリティ・メカニズムの設定方法について説明します。

複写システムが複数のセキュリティ・メカニズムをサポートしている場合は、Replication Server に複数のセキュリティ・メカニズムをインストールして各経路の両端で同じメカニズムをサポートできるようにする必要があります。この場合、次のように対処できます。

1. すべてのルート、コネクション、その他の経路に対して、**configure replication server** を使用して Replication Server を設定します。デフォルトのセキュリティ・メカニズム名が、`libtcl.cfg` ファイルの SECURITY の下に最初にリストされていることを確認します。
2. 異なるセキュリティ・メカニズムを使用する個々の経路に対して、セキュリティを設定します。そのセキュリティ・メカニズムが `libtcl.cfg` にリストされていることを確認します。

Replication Server のセキュリティ・メカニズムとサポートされているセキュリティ・パラメータを調べるには、**admin security_property** コマンドを使用します。特定の経路のセキュリティ・メカニズムと現在の設定を調べるには、**admin security_settings** コマンドを使用します。

参照：

- セキュリティ・サービスについての情報の表示 (288 ページ)
- さまざまな種類のネットワーク経路の保護 (272 ページ)

発生する可能性のあるセキュリティ問題

プライマリ・データベースとレプリケート・データベースで、異なるセキュリティ・メカニズムが使用されていて、Adaptive Server ユーザ名がこれらのサイトで

ユニークであることが保証されない場合は、要求ストアド・プロシージャに対してセキュリティ違反が発生することがあります。

システムがこの状態の場合、プライマリ・データベースとのコネクションについて、**dsi_exec_proc** パラメータを “off” にすることによって、セキュリティを維持できます。**alter connection** を実行して **dsi_exec_proc** を “off” にすると、Replication Server の要求ストアド・プロシージャ機能が無効になります。

構文は次のとおりです。

```
alter connection to  data_server.database
  set dsi_exec_request_sproc 'off'
```

SSL セキュリティの管理

Replication Server の SSL (Secure Sockets Layer) Advanced Security オプションは、セッションベースのセキュリティを提供します。SSL は、クレジット・カード番号や株取引などの機密情報をインターネット経由で転送する際のセキュリティを保護するための標準です。

SSL の概要

SSL は TLS (Transport Layer Security) とも呼ばれ、複数の暗号化アルゴリズムを備えた、管理の容易なライトウェイト・セキュリティ・メカニズムです。これは、強力なセキュリティ機能を必要とするデータベース・コネクションやルートで使用されることを目的としています。

SSL は、認証局 (CA) によって発行された証明書を使用して、身元の確認と検証を行います。証明書は電子的なパスポートのようなもので、認証されたエンティティのパブリック・キーや発行元 CA の署名など、エンティティを識別するために必要な情報がすべて含まれています。

サード・パーティの SSL セキュリティ・メカニズムを使用する方法の詳細については、そのソフトウェアのマニュアルを参照してください。追加情報については、IETF (Internet Engineering Task Force) の Web サイトも参照してください。

SSL のインストールには、次のものが必要です。

- 認証局 ― 証明書の検証と署名を行う有効なエンティティです。各 CA には、デジタル署名を発行するための独自の検証方式があります。
- 証明書 ― サーバ、ユーザ、組織、その他のエンティティを識別する電子ドキュメントです。証明書には、証明されたエンティティのパブリック・キーと発行した CA の署名が含まれます。
- フィルタ ― ポートに送信される情報、またはポートから送信される情報をフィルタする特別なネットワーク・ドライバです。

- ID ファイル ― 証明書と証明書のプライベート・キーを連結します。
- trusted ルート・ファイル ― 証明書のリストが含まれます。Open Client/Server は、trusted ルート・ファイルにリストされた CA のみを受け入れます。
- CipherSuites ― クライアントとサーバの認証、証明書の転送、データの暗号化、セキュリティ・セッション・キーの確立を行うための一連の暗号化アルゴリズムです。

SSL プロトコルは、TCP/IP の上や、HTTP または TDS などのアプリケーション・プロトコルの下で実行されます。SSL 接続が確立される前に、サーバとクライアントは一連のメッセージを交換し、安全な暗号化セッションをネゴシエートして承認します。この処理を SSL ハンドシェークといいます。

SSL ハンドシェーク

標準の SSL ハンドシェークは、3 つの手順で構成されます。

標準の SSL ハンドシェークの手順は、以下のとおりです。

1. クライアントは、クライアントがサポートする SSL オプションを含む接続要求をサーバに送信します。
2. サーバは、証明書と、サポートされている暗号化アルゴリズム (CipherSuites と呼ばれる)、キー交換アルゴリズム、デジタル署名のリストを返します。
3. クライアントとサーバの両方が CipherSuite を承認すると、安全な暗号化セッションが確立されます。

Replication Server 上の SSL

Replication Server による SSL のサポートについて説明します。

Replication Server は SSL API を直接呼び出しません。Replication Server の SSL サポートは、Sybase Open Client/Server が提供する機能に基づいています。Sybase は、Open Client/Server アプリケーションで SSL をサポートするために、Certicom の SSL Plus™ ライブラリ API を使用します。Open Client/Server での SSL のサポートの詳細については、『Open Server Server-Library/C リファレンス・マニュアル』を参照してください。

Replication Server の Advanced Security オプションは、サーバ認証とデータ暗号化をサポートしていますが、クライアント認証はサポートしていません。受信コネクションの場合、Replication Server は SSL ポートおよび SSL 以外のポートの両方をサポートします。送信コネクションの場合は、接続先サーバ上で SSL ポートと SSL 以外のポートの両方をサポートします。クライアントは、ユーザ名とパスワードを使用してサーバにログインする必要があります。ユーザ名とパスワードは Replication Server によって検証されます。このコネクションが作成されると、安全な暗号化セッションが確立されます。

SSL によって保護されたリンクを使用すると、Replication Server のパフォーマンスが低下することがあります。機密データを送信するコネクションまたはルートのみで SSL を使用することをおすすめします。

SSL の要件

SSL の要件は 3 つあります。

- RS 15.0 以降は、TLS バージョン 1.0 または 2.0 をサポートします。SSL バージョン 3.0 はサポートしません。
- SSL には、Replication Server バージョン 12.5 以降が必要です。それより前のバージョンの Replication Server は、SSL をサポートしていません。
- Replication Server 管理者は、サーバ証明書と trusted ルート CA 証明書を、Replication Server の外部ファイルとして生成し、セキュリティを保護する必要があります。

Replication Server での SSL セキュリティの設定

Replication Server での SSL サービスの設定方法について説明します。

前提条件

Replication Server で SSL サービスを設定する前に、SSL Plus のユーザ・マニュアルと、使用しているサード・パーティ製 SSL セキュリティ・ソフトウェアのマニュアルを参照してください。

手順

1. SSL ドライバの場所を含むように `$SYBASE/$SYBASE_OCS/config/libtcl.cfg` を編集します。
2. 信頼された CA 証明書を含むように `$SYBASE/config/trusted.txt` を編集します。
3. SSL コネクションを受け入れる Replication Server ごとに、信頼された CA から証明書を取得します。
4. 証明書と証明書のプライベート・キーを連結する ID ファイルを作成します。
5. **rs_init** を使用して、Replication Server 上で SSL を有効にし、暗号化された SSL パスワードを Replication Server 設定ファイルに追加します。

   注意： configure replication server と **use_ssl** オプションを使用しても、Replication Server 上で SSL を有効または無効にできます。
6. Replication Server の interfaces ファイルまたはディレクトリ・サービスに SSL エントリを作成します。

7. Replication Server を再起動します。

各手順の詳細については、『Replication Server 設定ガイド』の「Secure Sockets Layer」を参照してください。

Replication Server での SSL セキュリティの有効化

rs_init を使用して SSL セキュリティ機能を有効または無効にできます。また、**configure replication server** コマンドを **use_ssl** オプションとともに使用しても、SSL を有効または無効にできます。

use_ssl を使用して SSL を有効化するには、以下を入力します。

```
configure replication server
    set use_ssl to 'on'
```

SSL を無効にするには、**use_ssl** を “off” に設定します。デフォルトでは、SSL は Replication Server で有効になっていません。**use_ssl** を “off” に設定すると、Replication Server は SSL コネクションを受け入れません。

use_ssl は静的オプションです。値を変更したあとに、Replication Server を再起動する必要があります。

コマンド監査

Replication Server のコマンド監査機能を有効にして、ユーザの操作と実行したコマンドに関する情報を記録します。

コマンド監査によって記録されるアクションの種類

コマンド監査機能を有効にすると、Replication Server は、エントリを生成する以下の各ユーザ・アクションの情報を監査ログに記録します。

- 記録されるコマンドのクラスに属するコマンドの実行
- 必要なパーミッションがユーザに付与されていないことによるコマンドの実行エラー。たとえば、**create user** には sa パーミッションが必要です。
- 構文エラーなどのその他の理由によるコマンドの実行エラー
- Replication Server を起動したユーザ
- ログイン要求の失敗回数

注意： コマンド監査では、パスワードは、ログ・ファイルに記録されるとき、アスタリスク (*****) で表示されます。

コマンド監査では、記録された情報の種類を含むか、または記録されたコマンド・クラスに属するコマンドが 1 つしか存在しない場合でも、コマンドのバッチ全体が記録されます。このとき、バッチの前にログのエラー記述が挿入されます。

コマンド監査によって記録されるコマンドのクラス

Replication Server は、情報を表示する場合にのみ使用し、Replication Server を設定する場合には使用しない **admin** コマンドなどのシステム情報コマンドを除き、コマンド監査が有効な場合にすべての複写コマンド言語のデータ定義言語 (DDL: Data Definition Language) コマンドを記録します。

たとえば、Replication Server は、**alter route**、**configure replication server**、**connect**、**create connection**、**create partition**、**create user**、**grant**、**ignore loss**、**sysadmin sqm_purge_queue**、および **wait for create standby** を記録しますが、**admin config** と **admin sqm_readers** は記録しません。

コマンド監査の設定

コマンド監査は、デフォルトの記録先である Replication Server ログか他のユーザ指定の記録先ファイルにコマンドを記録するように設定できます。監査をオンまたはオフに設定するには、**configure replication server** を使用するときに **audit_enable** パラメータを指定します。

```
configure replication server
set audit_enable to {on|off}
```

コマンド監査を有効にするには **audit_enable** を on にします。デフォルトは off です。

コマンド・ログの保存先ファイルを指定する場合は、**audit_dest** をオプションで設定します。

```
configure replication server
set audit_destination to ['log'|'filename']
```

audit_enable が on の場合、**audit_dest** を設定できます。**audit_dest** のデフォルト値は **log** で、コマンドは Replication Server ログに記録されます。他の保存先のファイル名とパスを指定します。**audit_dest** で指定したファイルの読み込み／書き込みパーミッションが所有者に付与されていることを確認します。UNIX では、ファイルが存在しない場合、0600 のパーミッションを持つログ・ファイルが Replication Server によって作成されます。0666 などの別のパーミッションを持つ独自のログ・ファイルを UNIX で作成した場合は、そのパーミッションが Replication Server によって保持されます。たとえば、`/tmp/RSaudit.log` にコマンドを記録するとします。

```
configure replication server
set audit_dest to '/tmp/RSaudit.log'
```

コマンド監査ログのメッセージの例

ログ・ファイルでは、対応するコマンドまたはユーザ・アクションの前に "AUDIT" が配置されます。

- 正常に実行されたコマンドは次のようになります。

```
I. 2012/03/29 02:30:23. AUDIT: incoming command (issued by sa):
sysadmin site_version
```

- 必要なパーミッションがユーザに付与されていないため実行に失敗したコマンドは、次のようになります。

```
I. 2012/03/29 02:31:46. AUDIT Command failed: SA permission
required for:
I. 2012/03/29 02:31:46. AUDIT: incoming command (issued by user1):
shutdown
```

- その他の理由で実行に失敗したコマンドは、次のようになります。

```
I. 2012/03/29 03:18:15. AUDIT The following command batch has one
or more failures:
I. 2012/03/29 03:18:15. AUDIT: incoming command (issued by sa):
sysadmin badcommand
```

- Replication Server を UNIX プラットフォームで起動したユーザの場合 –

```
I. 2012/03/29 03:18:03. AUDIT: incoming command (issued by none):
Repserver started by username: ny_admin1
```

- Replication Server へのログイン試行が失敗した場合 –

```
I. 2012/03/22 02:12:52. AUDIT: Failed login attempt for user sa
```

- 正常に実行されたコマンドのバッチは、次のようになります。

```
I. 2012/03/29 03:22:19. AUDIT: incoming command (issued by sa):
create user user3 set password *********
I. 2012/03/29 03:22:19. AUDIT: incoming command (issued by sa):
sysadmin site_version
```

- 必要なパーミッションがユーザに付与されていないためコマンドの 1 つが実行に失敗したコマンドのバッチは、次のようになります。

```
I. 2012/03/29 03:23:14. AUDIT Command failed: SA permission
required for:
I. 2012/03/29 03:23:14. AUDIT: incoming command (issued by user1):
admin who
I. 2012/03/29 03:23:14. AUDIT: incoming command (issued by user1):
shutdown
```

- その他の理由でコマンドの少なくとも 1 つが実行に失敗したコマンドのバッチは、次のようになります。

```
I. 2012/03/29 03:24:08. AUDIT The following command batch has one
or more failures:
I. 2012/03/29 03:24:08. AUDIT: incoming command (issued by sa):
sysadmin site_version
I. 2012/03/29 03:24:08. AUDIT: incoming command (issued by sa):
sysadmin badcommand
```

セキュリティの推奨事項

管理タスクの実行、SSL、暗号化、パーミッションとロール、設定ファイルなど、Replication Server のセキュリティ問題に関する推奨事項です。

- ベスト・プラクティスとして、管理タスクは Replication Server のローカル・ホストに対してだけ実行してください。
  デフォルトでは、Replication Server のホスト名とポート番号を知っている管理者はだれでも、Replication Server にリモートでアクセスして管理できます。
- master データベース・トランザクションに依存するユーザ・データベース・トランザクション (テーブルの作成など) を実行する場合は、master データベース・トランザクション (ユーザの新規作成やパスワードの変更など) がすべてのレプリケート Adaptive Servers に正常にレプリケートされるまで待ってから行ってください。
  Replication Server では、単一の Adaptive Server データベース内で実行されるトランザクションのトランザクション・コミット順が維持されます。しかし、複数の Adaptive Server データベースにまたがって実行されるトランザクションのコミット順は維持されません。たとえば、プライマリ Adaptive Server で次のようにします。
  - master データベース・トランザクション (mylogin ユーザの作成など) を作成するには、sa ユーザを使用して次のコマンドを入力します。

    ```
    sp_addlogin 'mylogin', 'password'
    go
    use mydb
    go
    sp_adduser
    'mylogin'
    go
    ```

  - ユーザ・データベース・トランザクション (mylogin ユーザ ID を使用した `mytab` テーブルの作成など) を作成するには、次のコマンドを入力します。

    ```
    use mydb
    go
    create table mytab (mycol int)
    go
    ```

  Replication Server が、**sp_addlogin** プロシージャの前に **create table** コマンドをレプリケートすることは可能ですが、この場合、レプリケート Adaptive Server での **create table** は失敗します。これは、mylogin ユーザがレプリケート・データベースにまだ存在していないからです。
- Replication Server は、Secure Sockets Layer (SSL) を使用してセッションベースのセキュリティを提供できます。SSL は、認証局 (CA) によって発行された証明書を使用して、身元の確認と検証を行います。

SSL 証明書の機密性が損なわれた場合、新しい Replication Server 名と証明書番号で CA に新しい証明書を要求する必要があります。

- 管理者は、Replication Server ログでパーミッションを管理して、監査人に読み込み専用アクセス権を付与します。デフォルトでは、Replication Server で作成されたユーザは、ロールを付与されていない状態で、サポート・ロールに十分な RSSD テーブルへの読み込み専用アクセス権を持っています。
- ステーブル・キューでの機密データについては、ディスクレベルの暗号化を検討してください。
  プライマリ・データベースおよびレプリケート・データベースと Replication Server との間の SSL ベースの接続を使用した場合でも、Replication Server はデータをステーブル・キューに一時的に保持する必要があるため、この保持データは暗号化されません。
- 機密データを送信するコネクションまたはルートには、SSL を使用することをお勧めします。Replication Server の SSL (Secure Sockets Layer) Advanced Security オプションは、セッションベースのセキュリティを提供します。
- Replication Server は、ホスト名、ポート、ユーザ名、パスワードなどの初期設定プロパティを、**rs_init** ユーティリティが使用する `.res` サフィックスのファイルに格納します。`.res` ファイルについて、UNIX では umask パーミッション、Windows ではディレクトリ・パーミッションを適切に設定します。このファイルが必要ない場合は、削除してください。
  初期設定以降は、**rs_init** は `.res` ファイルを必要としませんが、Replication Server はこのファイルをオペレーティング・システム・パーミッションによってのみ保護されるオペレーティング・システムのファイル・システムに格納します。

参照：

- SSL セキュリティの管理 (290 ページ)

複写テーブルの管理

レプリケート・テーブル、テーブル複写定義、パブリケーションを管理するための準備、手順、具体的なコマンドについて説明します。

あるデータベースから別のデータベースにデータをコピーする場合、次の方法のうちサイトに最も適した方法を選択できます。

- 単一のデータベース複写定義を使用する方法。データベース複写定義では、個々のテーブル、トランザクション、ファンクション、システム・ストアド・プロシージャ、データ定義言語 (DDL) を複写するかどうかを選択できます。データベース複写では、Muliti-Site Availability (MSA) を使用できます。
- ファンクション複写定義を使用する方法。ファンクション複写定義ごとに、複写対象のシステム・ストアド・プロシージャを指定します。
- テーブル複写定義を使用する方法。テーブル複写定義では、それぞれ複写する特定のテーブルを識別するか、複写対象カラムのサブセットを指定します。

参照：
- MSA を使用した複写オブジェクトの管理 (463 ページ)
- 複写ファンクションの管理 (385 ページ)

複写テーブルの管理の概要

Replication Server を使用すると、あるデータベース (プライマリ) のテーブルから別のデータベース (レプリケート) のテーブルにデータをコピーして更新することができます。

注意： プライマリ・データベースは「送信元」ともいいます。また、レプリケート・データベースは「送信先」ともいいます。

テーブルを送信元として設定するには「複写定義」を作成します。複写定義は、コピーするデータのロケーションを指定し、そのデータが存在するテーブルの構造を記述するものです。

送信元テーブルからデータをコピーするには、まず送信先データ・サーバにテーブルの複製を作成する必要があります。次に、送信先テーブルを管理する Replication Server で、複写定義への「サブスクリプション」を作成します。サブスクリプションは SQL の **select** 文に似ています。

テーブルのデータすべてを複製したくない場合は、カラムのサブセットをコピーするように複写定義に指定したり、サブスクリプションに **where** 句を使用してローのサブセットを受信するように指定したりできます。

関連するテーブルの複写定義およびストアド・プロシージャを「パブリケーション」に含めて、それらすべてに対するサブスクリプションを 1 つのグループとして作成できます。パブリケーションを使用すると、サブスクリプションを構成して、グループ内のすべてのサブスクリプションのステータス情報を単一のコマンドでモニタできます。

異機種データ型サポート (HDS) 機能を使用すると、複写された値のデータ型を変更できます。HDS では、複写されたカラム値のデータ型を、レプリケート・データ・サーバで使用可能なデータ型に変換できます。HDS は、Sybase 環境、Sybase 以外の環境、Sybase と Sybase 以外のデータ・サーバが混在する環境で使用できます。

参照：

- サブスクリプションの管理 (405 ページ)
- サブスクリプションの例 (439 ページ)

複写システムのプラン作成

複写システムのプランを作成する場合は、設計を考慮し、データ複写の制限に注意し、複写システムの準備を整えます。

設計上の考慮事項

複写システムを設定するとき、いくつかの設計上の考慮事項があります。

複写システムを設定する場合は、次の点について考慮してください。

- セキュリティ (ユーザ・ログイン名とパスワード、コマンドの実行に必要なパーミッション、サード・パーティ・セキュリティ・システムなど)。
- 同時制御。具体的には、あるクライアントがデータを使用しているときに、そのデータを別のクライアントが変更することによって複写システムに発生する矛盾を防ぎます。『Replication Server デザイン・ガイド』の「概要」の「トランザクション管理」を参照してください。
- CPU、メモリ、ディスク、ネットワーク・リソース。『Replication Server デザイン・ガイド』の「容量の計画」を参照してください。
- 複写データ・モデルとルート指定スキーム。
- データ送信元またはデータ送信先として異機種データ・サーバを使用する場合の稼働条件。『Replication Server デザイン・ガイド』の「ASE 以外のデータ・サーバのサポート」を参照してください。
- バージョンが異なる Adaptive Server と Replication Server の互換性およびデータ複写の制限。

Sybase の互換性の問題についての詳細は、使用しているプラットフォームの『Replication Server リリース・ノート』を参照してください。

参照：

- Replication Server のセキュリティ管理 (233 ページ)
- Replication Server と分散データベース・システム (7 ページ)
- ルート指定スキーム (163 ページ)
- 混合バージョンの複写システム (20 ページ)

データ複写の制限

複写システムを設計するとき、いくつかの制限があります。

複写システムを設計するときは、次の制限事項も考慮してください。

- Adaptive Server と Replication Server のシステム・テーブルは、通常の複写ではコピーできません。ただし、ウォーム・スタンバイ・アプリケーションでは、一部のシステム・テーブルにおけるサポート対象のコマンドとシステム・プロシージャの実行をコピーできます。詳細については、『Replication Server 管理ガイド 第 2 巻』の「ウォーム・スタンバイの複写情報」を参照してください。また、一部のデータは、複写システムの RSSD 間で自動的にコピーされます。
- コピーするテーブルには、ユニークなプライマリ・キーが必要です。
- クライアント・アプリケーションは、キーが別のローのキーと重複するような方法で、ユニーク・インデックスまたはプライマリ・キー・カラムを更新することはできません。このような更新を行うと、Replication Server がトランザクションをコピーするときに、ローが重複したり、レプリケート・データベースでエラーが発生したりする可能性があります。
  たとえば、`pk_col` が `table1` のプライマリ・キー・カラムである場合、次のコマンドによって、レプリケート・データベースでエラーや無効なデータが発生する可能性があります。

  ```
  update table1
  set pk_col = pk_col + 1
  ```

  複写テーブルにプライマリ・キーまたはユニーク・インデックスの制約がある場合、更新は失敗し、レプリケート・データベースの DSI スレッドはサスペンドします。
- 複写テーブルにトリガが関連付けられている場合は、トリガ内にコミット文を含めないでください。レプリケート・ サイトにコミット文を含むトリガがあると、重複キーの原因となり、Replication Server のリカバリが失敗することがあります。
- 1 つの複写システムでバージョンの異なる複数の Replication Server を動作させることはできますが、一部の機能が制限される場合があります。

- 仮想計算カラムは複写できません。Replication Agent は Replication Server に仮想カラムを転送できず、Replication Server は仮想カラムを挿入または更新できないためです。
- 以下を含む暗号化カラムを複写する場合：
  - カラムで定義されたドメイン・ルールまたは検査制約。Replication Server は insert 文と update 文のドメイン・ルールまたは検査制約をバイパスします。
  - カラムで定義されたアクセス・ルール。Replication Server は、**update**、**delete**、または **select** 文を処理すると、エラー番号 2929「アクセス・ルールを付加できません」を返します。

参照：
- 混合バージョンの複写システム (20 ページ)

複写システムの準備

データ複写を完了するまでには、いくつかの作業があります。

1. 複写システムを設定します。
   a) Replication Server をインストールします。詳細については、使用しているプラットフォームの『Replication Server インストール・ガイド』と『Replication Server 設定ガイド』を参照してください。
   b) プライマリとレプリケートになる予定のデータベースを作成します。『Adaptive Server Enterprise リファレンス・マニュアル』または使用している Sybase 以外のデータベース・ソフトウェアのマニュアルを参照してください。
   c) Replication Server からプライマリ・データベースおよびレプリケート・データベースへのコネクションを確立します。使用しているプラットフォームの詳細については、『Replication Server 設定ガイド』を参照してください。
   d) Replication Server 間に必要なすべてのルートを確立します。
   e) 送信元データベースに Replication Agent を設定して起動します。
2. すべての複写システム・コンポーネントが動作していることを確認します。『Replication Server 管理ガイド 第 2 巻』の「Replication Server の検証とモニタリング」を参照してください。

参照：
- データベース・コネクションの作成 (194 ページ)
- ルートの管理 (161 ページ)
- 複写システムの管理 (83 ページ)

テーブルの複写手順の概要

複写手順は、テーブル複写定義とサブスクリプションを使用してテーブル間のデータを複写するのに必要ないくつかの手順で構成されています。

1. 複写システムのプランを作成し、複写システムの準備を適切に整えていることを確認します。
2. テーブルがない場合は、データベース所有者としてプライマリ・データベースにテーブルを作成します。データベース所有者以外がテーブル所有者の場合は、複写定義を作成するときにテーブル所有者名を指定します。
   - Adaptive Server で、**create table** を使用してテーブルを作成するか、または **sp_help** を使用してテーブルが存在することを確認します。
   - Adaptive Server 以外の送信元からデータを複写する場合は、使用しているデータベース・ソフトウェアの指示に従ってテーブルを作成します。異機種間の複写では、この手順のデータ・サーバ処理が異なる場合があります。
3. お使いのデータベースでストアド・プロシージャがサポートされている場合は、プライマリ・データベースで **rs_send_repserver_cmd** を実行すると、データのコピー元であるテーブルの複写定義を複数作成できます。それ以外の場合は、プライマリ Replication Server で 1 つまたは複数の複写定義を作成します。異なるテーブル・ビューを使用するサイトごとに、各複写定義に対するサブスクリプションを作成できます。

   複写定義を作成する場合は、手順 8 に説明されている、サブスクリプションを作成するテーブルの条件が満たされていなければなりません。複写定義には、送信元テーブルのカラムのすべてまたは一部を含めることができます。送信元テーブルおよび送信先テーブルには、同じまたは別のテーブル名、所有者名、カラム名、データ型を指定できます。また、複写された値のデータ型を変更できます。
4. パブリケーションを使用する場合は、プライマリ Replication Server で次の手順を実行します。
   a) **create publication** を使用して、複写するテーブルに対して 1 つ以上のパブリケーションを作成します。
   b) パブリケーションに含める各複写定義に対して、複写定義を拡張するための「*アーティクル*」を 1 つまたは複数作成します。**create article** を使用します。アーティクルには、送信先データベースに送信するローのサブセットを指定する **where** 句を含めることができます。
   c) **validate publication** を使用してパブリケーションを確定化し、パブリケーションに対してサブスクリプションを作成できるようにします。
5. プライマリ・テーブルを複写するようマーク付けします。

プライマリ Adaptive Server で **sp_setreptable** を使用して、テーブル複写を有効にします。この手順によって、RepAgent スレッドはテーブルに対するトランザクションをプライマリ Replication Server に転送できるようになります。

注意： プライマリが Adaptive Server 以外の場合に、テーブルとカラムにマークを付ける手順については、使用している Replication Agent のマニュアルを参照してください。

---

6. プライマリ・テーブルに `text`、`unitext`、`image`、または `rawobject` カラムがある場合は、プライマリ Adaptive Server で **sp_setrepcol** を使用して、これらのカラムの複写ステータスを調整する必要があります。

注意： プライマリが Adaptive Server 以外の場合は、使用している Replication Agent のマニュアルを参照してください。

---

7. サブスクリプションを作成するユーザのログイン名を準備します。レプリケート Replication Server でサブスクリプションを作成するログイン名は、プライマリ Replication Server にも存在する必要があります。

8. レプリケート・データベースで、複写定義によってパブリッシュされたスキーマに一致するテーブルを作成します。データベース所有者、または複写定義に指定されたものと同じテーブル所有者として、レプリケート・テーブルを作成します。

   Adaptive Server で、**create table** を使用してテーブルを作成するか、または **sp_help** を使用してテーブルが存在することを確認します。

   レプリケート・テーブルは、プライマリ・テーブルとテーブル名や所有者が異なっていてもかまいません。また、プライマリ・テーブルの一部のカラムだけを含むことも、異なるカラム名やデータ型を使用することもできます。複写定義には、このようなプライマリ・テーブルとレプリケート・テーブルの違いをすべて指定する必要があります。

注意： カラムが null 値を受け入れる場合、カラムに定義されたデフォルト値がある場合、または、カスタム・ファンクション文字列を使用してカラムに値を適用する場合には、複写定義にないカラムをレプリケート・テーブルに含めることができます。

---

9. レプリケート・データベースのメンテナンス・ユーザのログイン名に対して、レプリケート・テーブル用の **select**、**insert**、**delete**、**update** の各パーミッションを付与します。メンテナンス・ユーザは、複写トランザクションに対するコマンドを実行します。

10. 必要に応じて、ファンクション、ファンクション文字列、ファンクション文字列クラスを使用して、データベース・オペレーションをカスタマイズします。Replication Server のファンクション文字列は、データ・サーバ・オペレーションを実行します。

詳細については、『Replication Server 管理ガイド 第 2 巻』の「データベース・オペレーションのカスタマイズ」を参照してください。

11. レプリケート Replication Server でサブスクリプションを作成します。パブリケーションを使用する場合は、手順 12 に進んでください。

    レプリケート Replication Server にログインし、コピーするデータのテーブル複写定義に 1 つ以上のサブスクリプションを作成します。複写定義のカラムのローすべてにサブスクリプションを作成することも、**where** 句を使用して特定のローだけをコピーすることもできます。

    レプリケート・データベースは、プライマリ・テーブルの複数の複写定義にサブスクリプションを作成できます。これに対して、レプリケート・テーブルは、プライマリ・テーブルの 1 つの複写定義にしかサブスクリプションを作成できません。

    サブスクリプションを作成する場合、「マテリアライゼーション」と呼ばれるプロセスで、レプリケート・テーブルに初期テーブル・データが入ります。ほとんどの場合、データは Replication Server によってレプリケート・テーブルに自動的にコピーされます。また、手動でデータのマテリアライゼーションを行うこともできます。

12. パブリケーションを使用する場合は、手順 4 で作成したパブリケーションに対してパブリケーション・サブスクリプションを作成します。レプリケート Replication Server で **create subscription** を実行します。

    パブリケーション・サブスクリプションを作成すると、Replication Server は、パブリケーション内の各アーティクルに対してサブスクリプションを作成します。アーティクル・サブスクリプションには、**where** 句がありません。

13. サブスクリプション・ステータスをチェックします。

    サブスクリプション・データがレプリケート・データベースで完全にマテリアライズされ、トランザクションが正常に複写していることを確認します。

参照：

- 複写用データの指定方法 (36 ページ)
- 複写システムのプラン作成 (300 ページ)
- create replication definition コマンドの使用 (311 ページ)
- テーブルごとに複数の複写定義を作成する方法 (324 ページ)
- パブリケーションの使用 (367 ページ)
- 複写対象テーブルへのマーク付け (327 ページ)
- text、unitext、image、および rawobject カラムの複写 (335 ページ)
- サブスクリプションの管理 (405 ページ)
- Replication Server のセキュリティ管理 (233 ページ)

- パブリケーション・サブスクリプション (452 ページ)
- ファンクション複写定義を管理するコマンド (389 ページ)
- サブスクリプション・コマンド (426 ページ)
- サブスクリプションの例 (439 ページ)
- 複写システムの準備 (302 ページ)

テーブル複写定義を管理するためのコマンド

Replication Server には、テーブル複写定義を管理するためのコマンドおよびストアド・プロシージャがあります。

表 23 : テーブル複写定義を管理するためのコマンド

コマンド	作業
create replication definition	コピー対象のカラム、テーブルのロケーションなどの情報を記述した、プライマリ・テーブルの複写定義を作成する。
alter replication definition	次のような方法で既存の複写定義を変更する。 • カラムの追加または削除 • プライマリ・キーを追加または削除する • 複写定義にサーチャブル・カラムを追加または削除する • 異なるレプリケート・テーブル、テーブル所有者、カラム名、データ型を指定する • 最少カラム複写の変更 • カラム・レベルのデータ型変換を追加または変更する • `text`、`unitext`、`image`、または `rawobject` カラムの複写ステータスを変更する • スタンバイ・データベースに対する複写での複写定義の使用方法を変更する
drop replication definition	複写システムから複写定義を削除する。複写定義は、その複写定義用のすべてのサブスクリプションを削除すると、削除できるようになる。
rs_send_repserver_cmd	プライマリ・データベースで複写定義の変更要求を直接実行する。 **rs_send_repserver_cmd** ストアド・プロシージャを使用し、プライマリ・データベースで **create replication definition**、**alter replication definition**、**drop replication definition** の各コマンドを実行する。
admin verify_repserver_cmd	Replication Server が複写定義の変更要求を実行できることを確認する。

参照：

- 複写定義の作成 (307 ページ)
- 複写定義の変更 (351 ページ)

- 複写定義の削除 (365 ページ)
- プライマリ・データベースでの複写定義変更の直接実行 (352 ページ)
- 複写定義 RCL コマンドの確認 (354 ページ)
- 複写定義の変更要求プロセスの使用 (356 ページ)

複写定義の作成

複写定義では、Replication Server に対するプライマリ・テーブルを記述して、コピー対象のカラムを指定します。

同時に、レプリケート・テーブルの属性も記述できます。指定された特性に一致するレプリケート・テーブルは、複写定義にサブスクリプションを作成することができます。1 つのプライマリ・テーブルの複数の複写定義を作成し、特定の用途に合わせて各複写定義をカスタマイズできます。

複写定義は、次のいずれかの方法で作成できます。

- 送信元テーブルを管理する Replication Server で **create replication definition** を実行する。
- プライマリ・データベースで複写定義変更要求を直接実行する際に、**rs_send_repserver_cmd** に **create replication definition** 句を指定して実行する。

各複写定義に関する情報は、プライマリ Replication Server からのルートに沿って、条件を満たす各 Replication Server に送信されます。混合バージョン・システムには制限があります。

複写定義は、RSSD にある `rs_objects` および `rs_columns` の各システム・テーブルに格納されます。複写定義のプライマリ・バージョンは、プライマリ Replication Server にあります。

参照：

- テーブルごとに複数の複写定義を作成する方法 (324 ページ)
- create replication definition コマンドの使用 (311 ページ)
- プライマリ・データベースでの複写定義変更の直接実行 (352 ページ)
- 混合バージョン・システムでの複写定義の制限 (327 ページ)

複写定義の設定

次に、複写定義を作成するために必要な情報を示します。

各複写定義には、次の情報が含まれている必要があります。

- 複写定義の名前です。

- プライマリ・テーブルとレプリケート・テーブルの名前。
  Replication Server は、別の名前を指定しないかぎり、複写定義名は送信先と送信元の両方のテーブルと同じ名前であると想定します。
  スペースや特殊文字を含むテーブル名を使用するには、引用符付き識別子のサポートを有効にします。
- プライマリ・テーブルが存在するデータ・サーバとデータベースの名前。
- コピー対象のカラム名とデータ型。プライマリ・テーブルのカラムは、すべてまたは一部をコピーできます。特に指定しないかぎり、レプリケート・カラムの名前およびデータ型は、プライマリ・カラムの名前およびデータ型と同じです。
  スペースや特殊文字を含むカラム名を使用するには、引用符付き識別子のサポートを有効にします。
- プライマリ・キー。送信元テーブル内の各ローをユニークに識別する 1 つまたは複数のカラムです。

オプションで、複写定義に次のものを含めることもできます。

- プライマリ・テーブルとレプリケート・テーブルの所有者の名前。デフォルトのテーブル所有者は、データベース所有者 (dbo) です。
- サーチャブル・カラムの名前。サーチャブル・カラムは、レプリケート・テーブルにコピーするプライマリ・テーブルのローを示すために、サブスクリプションの **where** 句に指定できるカラムです。
- ウォーム・スタンバイ・アプリケーションの場合は、複写定義を使用してスタンバイ・データベースにデータをコピーするかどうかを指定できます。また、すべてのテーブル・カラムをコピーするか、複写定義のカラム・リストにあるカラムだけをコピーするかを指定することもできます。
- 更新オペレーションと削除オペレーションに必要な最少数のカラムだけをコピーするかどうかを指定できます。このオプションにより、システム全体のパフォーマンスを向上させることができます。
- `text`、`unitext`、`image`、`rawobject` の各カラムの複写オプション。
- カラム・レベルのデータ型変換。

引用符付き識別子

Replication Server 15.2 以降では、引用符付き識別子のサポートを有効にできます。

テーブル名やカラム名などのオブジェクト名に対して、DSI の引用符付き識別子のサポートを有効にできます。このようなオブジェクト名は、次の理由により、正しく解析されるように二重引用符文字で囲む必要があります。

- スペースや英数字以外の文字などの特殊文字を含む。
- アルファベット以外の文字で始まる。
- 予約語に相当する。

引用符付き識別子のサポートの有効化

引用符付き識別子のサポートを有効にするには、**create connection** または **alter connection** コマンドを使用して、**dsi_quoted_identifier** パラメータを “on” に設定します。

『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」には、**create connection** と **alter connection** のコマンド構文の詳細が記載されています。

データ・サーバは、引用符付き識別子を異なる方法で受け取ります。Adaptive Server と Microsoft SQL Server は引用符付き識別子を想定しないため、引用符付き識別子に関する特別なコネクション設定が必要になります。Oracle と UDB では、引用符付き識別子を受け入れるための特別なコネクションは必要ありません。

参照：

- 物理データベース・コネクションに影響する設定パラメータ (203 ページ)

引用符付き識別子をサポートするためのシステム・ファンクション

rs_set_quoted_identifier ファンクション文字列は、DSI コネクションを Adaptive Server と Microsoft SQL Server に設定するために追加されています。

dsi_quoted_identifier を “on” に設定すると、コネクションが確立されたときに、**rs_set_quoted_identifier** ファンクション文字列がコネクション設定の一部としてレプリケート・データベースに送信されます。Replication Server は、このファンクション文字列を使用して、引用符付き識別子をコネクション経由で送信できるようにします。

『Replication Server リファレンス・マニュアル』の「Replication Server システム・ファンクション」の「**rs_set_quoted_identifier**」と『Replication Server 管理ガイド第 2 巻』の「データベース・オペレーションのカスタマイズ」の「ファンクション、ファンクション文字列、クラスの処理」の「システム・ファンクションの概要」を参照してください。

引用符付きとしての識別子のマーク付け

識別子を引用符付き識別子としてマークを付けるには、**create replication definition** または **alter replication definition** コマンドに **quoted** パラメータを指定して使用します。

レプリケート・サーバ・コネクションに対する **dsi_quoted_identifier** が "on" に設定され、引用符付き識別子としてマークを付けられている識別子を持つ複写定義に **dsi_quoted_identifier** のサブスクリプションが作成されている場合、マークを付けられている識別子は二重引用符で囲まれます。

dsi_quoted_identifier を “on” に設定し、識別子にマークが付けられている場合、複写定義にサブスクリプションを作成するレプリケート・サーバは、マーク付けされた識別子を引用符付き識別子として受け取ります。**dsi_quoted_identifiers** が “off” の場合、マーク付けは無視され、レプリケート・サーバに送信された識別子は二重引用符で囲まれません。

引用符付き識別子としてカラム `foo_col1` を持つテーブル `foo` を作成するには、次のように入力します。

```
create replication definition repdef
   with primary at primaryDS.primaryDB
   with all tables named “foo”
   (“foo_col1” int quoted, “foo_col2” int)
   primary key (“foo_col1”)
```

`foo` という名前のテーブルを引用符付きとしてマーク付けするには、次のように入力します。

```
alter replication definition repdef
   alter replicate table name “foo” quoted
```

`foo_col1` のマーク付けを解除するには、次のように入力します。

```
alter replication definition repdef
   with replicate table name “foo”
   alter columns “foo_col1” not quoted
```

create replication definition および **alter replication definition** の全コマンド構文については、『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」を参照してください。

注意： ウォーム・スタンバイ・データベースおよび複写定義のサブスクライバへの複写時に、プライマリ・テーブル名は引用符付きとしてマーク付けされているが、レプリケート・テーブル名はマーク付けされていない場合 (またはその逆)、Replication Server は、プライマリ・テーブル名とレプリケート・テーブル名の両方を引用符付きとして送信します。

混合バージョンの制限

引用符付き識別子機能は、混合バージョン環境ではサポートされません。引用符付き識別子の複写を成功させるには、プライマリ Replication Server とレプリケート・データ・サーバに接続する Replication Server のバージョンを 15.2 にします。ただし、ルート上の中間 Replication Server は、古いバージョンでもかまいません。

create replication definition コマンドの使用

複写するテーブルの特性を Replication Server に記述するには、**create replication definition** を使用します。

プライマリ・データベースで複写定義の変更要求を直接実行するには、送信元テーブル・データベースを管理する Replication Server で **create replication definition** を実行するか、**rs_send_repserver_cmd** ストアド・プロシージャを **create replication definition** 句を指定して実行します。

複写定義には、送信元データ・サーバとデータベースの名前を含める必要があります。

次の例では、**publishers** という名前の基本的な複写定義を、同名の送信元テーブルと送信先テーブルに対して作成する方法を示します。プライマリ・データベースは、TOKYO_DS データ・サーバによって管理される `pubs2` です。テーブルのすべてのカラムを含め、`pub_id` カラムをプライマリ・キーとして指定しています。

```
create replication definition publishers
with primary at TOKYO_DS.pubs2
(pub_id char(4), pub_name varchar(40),
city varchar(20), state char(2))
primary key (pub_id)
```

コマンド構文の詳細と使用法のガイドラインについては、『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**create replication definition**」を参照してください。

参照：

- プライマリ・データベースでの複写定義変更の直接実行 (352 ページ)

複写定義名とテーブル名の指定

複写定義には、1 つのグローバルな「ネーム・スペース」があります。つまり、プライマリ Replication Server からのルートを持つすべての Replication Server において、名前は同一の複写定義を参照します。

Replication Server は、**create replication definition** を入力したときに、複写定義名がユニークでなければならないという要件を常に適用できるわけではありません。新しい複写定義を作成する場合は、同じ名前を持つ既存の複写定義 (テーブルまたはファンクション) が存在しないことを確認してください。

デフォルトでは、複写定義名は、送信元と送信先の*両方*のテーブル名になります。

場合によっては、送信元テーブルと送信先テーブルに別々の名前を使用したり、テーブルと複写定義の名前を別にしたりする必要があります。複写定義名とは異なるテーブル名を指定するには、**with all tables named**、**with primary table named**、**with replicate table named** のいずれかのオプション句を含めてください。

送信元テーブルと送信先テーブルが同じ名前の場合
送信元テーブルとすべての送信先テーブルが同じ名前を共有するが、複写定義には別の名前を指定したい場合は、**with all tables named** を使用してテーブル名を指定します。

たとえば、**publishers_rep** という複写定義を publishers という送信元テーブルおよび送信先テーブルに対して作成するには、次のコマンドを入力します。

```
create replication definition publishers_rep
with primary at TOKYO_DS.pubs2
with all tables named publishers
...
```

送信元テーブルと送信先テーブルの名前が異なる場合
送信元テーブルと送信先テーブルの名前が異なる場合は、**with primary table named** を使用して送信元テーブルの名前を指定するか、または **with replicate table named** を使用して送信先テーブルの名前を指定します。

いずれか一方の句を使用することも、両方の句を同時に使用することもできます。

別のテーブル名を指定していない場合、複写定義名は送信元テーブルと送信先テーブルの*両方*の名前であるとみなされます。

たとえば、複写定義 **publishers_rep** を送信元テーブル publishers1 と送信先テーブル publishers2 に対して作成するには、次のコマンドを入力します。

```
create replication definition publishers_rep
with primary at TOKYO_DS.pubs2
with primary table named publishers1
with replicate table named publishers2
...
```

複写定義と送信元テーブルの名前が publishers で、送信先テーブル名が publishers2 の場合は、次のように入力します。

```
create replication definition publishers
with primary at TOKYO_DS.pubs2
with replicate table named publishers2
...
```

この例では、**publishers** という複写定義名が送信元テーブルの名前にもなります。

送信元テーブルまたは送信先テーブルの所有者名の指定
オプションの修飾子として、送信元または送信先テーブルの名前とともにテーブル所有者の名前を指定できます。テーブルの所有者が複写定義に指定されたテーブル所有者と一致しない場合、データ・サーバのオペレーションは失敗します。

たとえば、送信元テーブル publishers とユーザ “ravi” が所有する送信先テーブル publishers2 の複写定義を作成するには、次のように入力します。

```
create replication definition publishers
with primary at TOKYO_DS.pubs2
with replicate table named ravi.publishers2
...
```

カラム名とデータ型の指定

複写定義を作成するときは、コピー対象となるテーブルのカラムの名前とデータ型をリストします。

別のレプリケート (パブリッシュ) カラム名またはデータ型を指定しないかぎり、レプリケート・テーブルのカラム名とデータ型はプライマリ・テーブルと同じになります。

すべてのカラムの名前とそのデータ型はカッコで囲みます。カラムが複数ある場合は、カンマを使用して、各カラムとそのデータ型を次のカラムから区切ります。

たとえば、次のコマンドは、複写定義 **publishers_rep1** を `publishers` という送信元テーブルおよび送信先テーブルに対して作成します。これには、すべてのカラムとそのデータ型が含まれます。

```
create replication definition publishers_rep1
with primary at TOKYO_DS.pubs2
with all tables named publishers
(pub_id char(4),
pub_name varchar(40),
city varchar(20),
state char(2))
primary key (pub_id)
```

次のコマンドは、複写定義 **publishers_rep2** を `city` カラムを省略して作成します。このカラムを必要としない送信先サイトは、この複写定義にサブスクリプションを作成できます。

```
create replication definition publishers_rep2
with primary at TOKYO_DS.pubs2
with all tables named publishers
(pub_id char(4),
pub_name varchar(40),
state char(2))
primary key (pub_id)
```

パフォーマンスは、カラムをテーブル内と同じ順序で複写定義にリストすると最適になります。

使用できるデータ型は、Replication Server でサポートされているネイティブ・データ型とユーザ定義データ型のみです。プライマリ・テーブルにユーザ定義データ型のカラムがある場合、複写定義ではサポートされている互換データ型を使用する必要があります。また、インストール・プロセスの一環として、Replication Server に用意されているユーザ定義データ型を使用することもできます。

Replication Server でサポートされているデータ型の詳細については、『Replication Server リファレンス・マニュアル』の「トピック」の「データ型」を参照してください。

送信元カラムと送信先カラムの名前が異なる場合
送信元テーブルに 1 つだけ複写定義が必要であり、送信元カラム名が送信先テーブルの対応するカラム名と異なる場合は、複写定義で *column_name* **as** *replicate_column_name* 句を使用します。

たとえば、送信元テーブル `publishers1` と送信先テーブル `publishers2` で、送信元カラム `pub1_name` が送信先カラム `pub2_name` に対応する場合は、次のように入力します。

```
create replication definition publishers_rep
with primary at TOKYO_DS.pubs2
with primary table named publishers1
with replicate table named publishers2
(pub_id char(4),
pub1_name as pub2_name varchar(40),
city varchar(20),
state char(2))
primary key (pub_id)
```

複数のプライマリ・テーブル複写定義のデータ型
1 つの送信元テーブルに複数の複写定義を作成するには、宣言されたカラムのデータ型 (プライマリ・テーブル内にあるカラムのデータ型) が同じでなければなりません。ただし、カラムのデータ型が `rawobject` または `rawobject in row` (それぞれ `image`、`varbinary` に対応) の場合を除きます。

具体的には、次のことができます。

- 1 つの複写定義でカラムのデータ型を `rawobject` として宣言し、同じテーブルに対する他の複写定義で、同じカラムのデータ型を `image` として宣言する。
- 1 つの複写定義でカラムのデータ型を `rawobject in row` として宣言し、同じテーブルの別の複写定義で同じカラムのデータ型を `varbinary` として宣言する。

レプリケート (パブリッシュ) カラムのデータ型は、同じテーブルでも複写定義ごとに異なってもよく、制限はありません。

カラムがプライマリ・テーブルの既存の複写定義にリストされている場合、同じプライマリ・テーブルの次の複写定義でそのカラムのデータ型をオプションであると指定すると、最初の定義 (データ型を指定した定義) が削除されても、データ型は前の複写定義から継承されて、次の複写定義で保持されます。

カラムのデータ型を変更するには、**alter replication definition** コマンドを使用します。

参照：
- カラム・データ型の変更 (362 ページ)

レプリケート・テーブルの追加カラム
カラムのデフォルト値が定義されている場合や、カスタム・ファンクション文字列を使用してカラムに値を適用する場合には、複写定義にないカラムをレプリケート・テーブルに含めることができます。

それには、カラムを **create table** に指定して、null 値を受け入れるようにします。送信元ローが送信先テーブルにコピーされると、余分なカラムは null 値で埋められます。また、ローカル・データ・サーバによって別個に更新される場合もあります。

text、unitext、image、Java カラムの含有
`text`、`unitext`、`image`、または Java データ型の `rawobject` カラムと `rawobject in row` カラムのデータを送信先サイトにコピーするには、これらのカラムを複写定義に含めます。

`text`、`unitext`、`image`、または Java カラムを複写する場合は、追加の特別な手順と考慮事項があります。

参照：
- text、unitext、image、および rawobject カラムの複写 (335 ページ)
- Replication Server における Java データ型 (332 ページ)

特殊データ型の使用
特定のサイトに更新を分配するには、`rs_address` 特殊データ型をビットマップ・サブスクリプションとともに使用します。

コピー対象のテーブルに `identity` カラムがある場合は、`identity` 特殊データ型を使用できます。

コピー対象のテーブルに `timestamp` カラムがある場合は、`timestamp` 特殊データ型も使用できます。

参照：
- ビットマップ・サブスクリプション (445 ページ)
- identity カラムの複写 (348 ページ)
- timestamp カラムの複写 (349 ページ)
- rs_address データ型の使用 (347 ページ)

ユーザ定義データ型の使用
プライマリ・データベースで複写された値のデータ型をレプリケート・データベースで受け入れ可能なデータ型に変更するには、**create replication definition** コマンドの **map to** 句を使用します。

参照：
- HDS を使用したデータ型の変換 (377 ページ)

プライマリ・キーの指定

「プライマリ・キー」とは、各ローをユニークに識別するカラム、またはカラムの組み合わせです。

Adaptive Server を含む多数のデータ・サーバでは、テーブル内でのローの重複が認められますが、Replication Server では、送信元テーブルと送信先テーブルで各ローのプライマリ・キー・カラムの値がユニークでなければなりません。

送信元テーブルのプライマリ・キー・カラムを識別するには、**primary key** 句を **create replication definition** に含める必要があります。プライマリ・キー・カラムは、カラム・リストにも含まれている必要があります。

Replication Server は、送信先サイトでデフォルトの **rs_update** または **rs_delete** ファンクション文字列を適用するときに、update 文または delete 文の **where** 句にプライマリ・キーの値を指定します。

プライマリ・キー・カラムの名前はカッコで囲んでください。次に例を示します。

```
create replication definition publishers
with primary at TOKYO_DS.pubs2
(pub_id char(4), pub_name varchar(40),
city varchar(20), state char(2))
primary key (pub_id)
```

プライマリ・キー・カラムが複数ある場合は、カラムとカラムの間をカンマで区切ります。

注意： `text`、`unitext`、`image`、`rawobject`、`rawobject in row`、または `rs_address` データ型のカラムは、プライマリ・キーの一部として含めることはできません。

サーチャブル・カラムの指定

送信先サイトにコピーするローを制限するには、**create replication definition** で **searchable columns** を使用して、**create subscription** または **define subscription** (またはパブリケーションの **create article**) の **where** 句で使用するカラムを指定します。

複写定義に **searchable columns** 句を含めていない場合、その複写定義を参照するサブスクリプションまたはアーティクルで **where** 句を使用することはできません。

サーチャブル・カラムの名前はカッコで囲みます。サーチャブル・カラムが複数ある場合は、カラムとカラムの間をカンマで区切ります。

次の例では、3 つのカラム pub_id、pub_name、state が、サーチャブル・カラムとして指定されています。これらのカラムはすべて、サブスクリプションの **where** 句に含めることができます。

```
create replication definition publishers
with primary at TOKYO_DS.pubs2
(pub_id char(4), pub_name varchar(40),
city varchar(20), state char(2))
primary key (pub_id)
searchable columns (pub_id, pub_name, state)
```

参照：

- where 句の使用 (428 ページ)

サーチャブル・カラムの制限

サーチャブル・カラムには、いくつかの制限があります。

サーチャブル・カラムには、次の制限があります。

- text、unitext、image カラム、または Java の rawobject カラムや rawobject in row カラムをサーチャブル・カラムとして指定することはできません。
- **searchable columns** 句に含まれるカラムには、null 値を指定できません。
- サブスクリプションで **where** 句を使用してビットマップ比較を実行するには、rs_address データ型を使用するカラムすべてを複写定義の **searchable columns** 句に含める必要があります。
- 複写定義の **searchable columns** リストのサーチャブル・カラムが増えるほど、Replication Server によるサブスクリプションの処理速度は低下します。つまり、サーチャブル・カラムが少ないほど、Replication Server によるテーブルのサブスクリプションに対するローの評価は効率的になります。

参照：

- rs_address データ型の使用 (347 ページ)

カラムの最少セットの複写

複写システムのパフォーマンスを強化するには、**create replication definition** に **replicate minimal columns** を指定します。この句を使用すると、削除オペレーションと更新オペレーションに必要なカラムだけを送信してデータベースを複写できます。

通常、Replication Server は、挿入の場合の他に、更新や削除を適用する場合にも、各レプリケート・データベースに各ローのすべてのカラムを送信します。複写定

義がテーブルに使用されていない場合、または複写定義がスタンバイ・コネクションに使用されていない場合、通常、Replication Server は、最大限のカラムをスタンバイ・データベースに送信します。

注意： SQL Remote データベースを複写するときは、すべてのカラムを送信する必要があります。最少カラムを送信すると、複写は失敗します。

replicate minimal columns を設定すると、次のようになります。

- **delete** オペレーションの場合、送信元 Replication Server は、プライマリ・キー・カラムだけを送信先 Replication Server またはスタンバイ・データベースに送信します。
- **update** オペレーションの場合、送信元 Replication Server は、更新オペレーションによって変更されるカラムとプライマリ・キー・カラムだけを送信先 Replication Server またはスタンバイ・データベースに送信します。

注意： **replicate minimal columns** は、すべてのカラムがコピーされる挿入オペレーションには適用されません。

送信先 Replication Server は、レプリケート・データベースまたはスタンバイ・データベースに適用するデータ・サーバ・コマンドを作成するときに、プライマリ・キー・カラムを使用します。

次の複写定義には、**replicate minimal columns** が含まれています。

```
create replication definition publishers
with primary at TOKYO_DS.pubs2
(pub_id char(4), pub_name varchar(40),
city varchar(20), state char(2))
primary key (pub_id)
replicate minimal columns
```

最少カラム設定の変更

既存の複写定義を変更して、カラムの最少セットだけを複写するか、またはすべてのカラムを複写するには、**alter replication definition** を使用します。

最少カラムと rs_update および rs_delete ファンクション文字列

replicate minimal columns を指定し、デフォルト以外の **rs_update** ファンクション文字列と **rs_delete** ファンクション文字列を作成する必要がある場合は、*rs_default_fs* ファンクション文字列変数を使用して、ファンクション文字列のデフォルトの動作を示します。

詳細については、『Replication Server 管理ガイド 第 2 巻』の「データベース・オペレーションのカスタマイズ」の「デフォルトのシステム変数の使用」を参照してください。

最少カラムとオートコレクション

replicate minimal columns を指定した場合、オートコレクションを同時に指定することはできません。オートコレクションは、各 **update** オペレーションまたは **insert** オペレーションを **delete** と **insert** の連続オペレーションに変換することによって、マテリアライゼーション中に発生する可能性のある矛盾を自動的に修正する機能です。

オートコレクションがオンに設定されている状態で最少カラムを設定した場合 (たとえば、**alter replication definition** を使用)、オートコレクションは実行されなくなります。Replication Server は、更新オペレーションに対して情報メッセージのログを記録します。

ノンアトミック・マテリアライゼーションを使用してサブスクリプションを作成する場合は、オートコレクションをオンに設定する必要があります。最少カラム複写が複写定義に設定されていて、ノンアトミック・マテリアライゼーション、またはノンアトミック・マテリアライゼーションをシミュレートするバルク・マテリアライゼーション・メソッドを使用する新しいサブスクリプションを作成する場合、オートコレクションによって矛盾を解決することはできません。

参照：

- ノンアトミック・マテリアライゼーションのオートコレクションの使用 (410 ページ)
- サブスクリプション・マテリアライゼーション・メソッド (406 ページ)

ウォーム・スタンバイ・アプリケーションでの複写定義の使用

ウォーム・スタンバイ・アプリケーションでは複写定義を使用する必要はありません。ただし、複写定義を使用すると、サブスクリプションを使わずに、スタンバイ・データベースへの情報の流れを制御できます。

スタンバイ・データベースへの複写のためだけに複写定義を作成することも、既存の複写定義をこの目的に使用することもできます。

send standby を **create replication definition** で次のように使用します。

- トランザクション・データをスタンバイ・データベースに複写するには、この複写定義で **send standby** を任意の形式で使用します。Replication Server は、複写定義のプライマリ・キーと最少カラム設定を使用するようになります。
- テーブル内のすべてのカラムをスタンバイ・データベースに送信するには、**send standby** または **send standby all columns** を使用します。
- 複写定義に指定されたカラムだけをスタンバイ・データベースに送信するには、**send standby replication definition columns** を使用します。

send standby を省略すると、このテーブルのデータをスタンバイ・データベースに複写するときに別の複写定義が使用されるか、複写定義が使用されません。

次の例の複写定義は、スタンバイ・データベースにトランザクションを複写するときに使用されます。スタンバイ複写では、プライマリ・キーとカラムの最少セットの設定が使用されます。複写定義に指定されたカラムだけが、スタンバイ・データベースに複写されます。この複写定義では `city` カラムが省略されています。

```
create replication definition publishers_ws
with primary at LDS.pubs2
with all tables named 'publishers'
(pub_id char(4),
pub_name varchar(40),
state char(2))
primary key (pub_id)
send standby replication definition columns
replicate minimal columns
```

同じプライマリ・テーブルの複写定義がすでに存在し、スタンバイで使用するためのマークが付けられている場合、**send standby** を使用して新しい複写定義を作成すると (または別の複写定義を変更すると)、スタンバイで使用することを示す前の複写定義のマークが解除されます。

ウォーム・スタンバイ・アプリケーションでの複写定義の使用方法の詳細については、『Replication Server 管理ガイド 第 2 巻』の「ウォーム・スタンバイ・アプリケーションの管理」の「複写定義とサブスクリプションの使用」を参照してください。

参照：

- プライマリ・キーの指定 (316 ページ)
- カラムの最少セットの複写 (317 ページ)

text、unitext、image カラムの複写の指定

`text`、`unitext`、または `image` データ型のカラムを含んでいるテーブルの複写定義を作成する場合、いくつかの制限があります。

次のことを行ってください。

- 複写する `text`、`unitext`、または `image` の各カラムをカラム・リストに含めます。
- 各カラムをオプションの **replicate_if_changed** 句または **always_replicate** 句に含めます。
  各句で、`text` および `image` カラムの名前をカッコで囲みます。カラムが複数ある場合は、カラムとカラムの間をカンマで区切ります。
- `text`、`unitext`、または `image` の各カラムの対応するステータスが Adaptive Server にあることを確認します。

『Replication Server 管理ガイド 第 2 巻』の「ウォーム・スタンバイ・アプリケーションの管理」の「ASE ウォーム・スタンバイ・アプリケーションで複写される

情報」の「text、unitext、image、および rawobject データの複写」を参照してください。

参照：

- text、unitext、image、および rawobject カラムの複写 (335 ページ)

計算カラムの複写の指定

計算カラムの複写定義を作成するには、マテリアライズされたカラムの複写定義でベース・カラム・データ型を使用します。複写定義に仮想カラムを含めないでください。

rawobject カラムと rawobject in row カラムの複写の指定

複写定義に Java カラムを含めることができます。Replication Server は、Java カラムを `rawobject` データ型または `rawobject in row` データ型として複写します。

Java データ型を持つテーブルの複写定義を作成するには、次の作業を行います。

- 複写する `terawobjectxt` カラムまたは `rawobject in row` カラムをそれぞれカラム・リストに含めます。
- `rawobject` カラムそれぞれを、オプションの句である **replicate_if_changed** または **always_replicate** に含めます。

  各句で、`rawobject` カラムの名前をカッコで囲みます。カラムが複数ある場合は、カラムとカラムの間をカンマで区切ります。

  注意： `rawobject in row` カラムには複写ステータスはありません。
- 各 `rawobject` カラムに対応するステータスが Adaptive Server にあることを確認します。

参照：

- text、unitext、image、および rawobject カラムの複写 (335 ページ)

カラム・レベル・データ型変換の指定

複写定義には、カラム・レベルのデータ型変換を指定できます。

Sybase では、一連のデータ型定義を用意しています。このデータ型定義は、使用しているプラットフォームの『Replication Server 設定ガイド』に記載されている手順に従ってインストールしてください。

- *declared_datatype* は、Replication Agent から Replication Server に配布される値のデータ型を定義します。
- *published_datatype* は、カラム・レベル変換後の値のデータ型を定義します。複写定義の **map to** 句を使用して定義されたカラム・レベル変換が存在しない場

合、パブリッシュ・データ型はデフォルトで宣言したデータ型に設定されます。

- *delivered datatype* は、レプリケート・データベースに配信される値のデータ型です。レプリケート・データ・サーバのコネクションに使用されるファンクション文字列クラスに対して定義されたクラス・レベル変換が存在する場合、これはパブリッシュ・データ型と異なる場合があります。定義されたクラス・レベル変換が存在しない場合、配信されるデータ型はデフォルトでパブリッシュ・データ型に設定されます。

参照：

- HDS を使用したデータ型の変換 (377 ページ)

拡張された制限値に基づく複写定義の作成

Replication Server バージョン 12.5 以降では、複写可能なカラム・サイズ、パラメータの長さ、カラム数の制限値が前のバージョンよりも拡張されました。また、以前よりも大きいデータ・ローやメッセージを処理できます。

Replication Server は、Adaptive Server バージョン 12.5 以降で導入された制限値の拡張機能をサポートしています。詳細については、Adaptive Server のマニュアルを参照してください。Sybase 以外のデータ・サーバで Replication Server の制限値拡張機能を使用する方法については、使用している Sybase Replication Agent のマニュアルと、『Replication Server 異機種間複写ガイド』を参照してください。

拡張された制限値を適用する前に

拡張された制限値を使用するには、サイト、ルート、および Adaptive Server のバージョンがサポート対象のバージョン・レベルにアップグレードされていることを確認します。

拡張された制限値を適用するには、プライマリ Replication Server およびレプリケート Replication Server の両方のサイト・バージョンが 12.5 以降にアップグレードされていることを確認してください。12.5 以降にアップグレードされていれば、LTL バージョンが自動的に 400 に設定されます。さらに、拡張された制限値を適用するすべてのルートが 12.5 以降に設定されていることを確認してください。Adaptive Server を使用している場合は、プライマリ・データベースとレプリケート・データベースが両方ともバージョン 12.5 以降に設定されていることを確認してください。プライマリ・データベースとレプリケート・データベースの両方が同じページ・サイズに設定されている必要があります。

Replication Server のホワイト・ペーパー「Using Adaptive Server Enterprise version 12.5 with Replication Server version 12.1 and earlier:Schema-length and compatibility issues.」を参照してください。

参照：
- 混合バージョン・システムでの複写定義の制限 (327 ページ)

拡張された制限値の使用
レプリケート・データベースとスタンバイ・データベースの両方に、拡張された制限値を使用する複写定義を作成できます。

拡張された制限値は、次のように定義されます。

- ワイド・カラム ― 256 ～ 32,768 バイトのデータ・ロー。
- カラム数の増加 ― カラムが 251 以上ある複写定義。
- ワイド・データ ― 最大でデータ・サーバのデータ・ページ・サイズまでのデータ・ロー。Adaptive Server バージョン 12.5 以降では、2K、4K、8K、16K のページ・サイズをサポートしている。
- ワイド・メッセージ ― 16K より長いメッセージ。

ワイド・カラム
Replication Server では、`char`、`varchar`、`binary`、`univarchar`、`unichar`、`unitext`、または Java `inrow` データを含む、最大 32,768 バイトのワイド・カラムを複写できます。カラムの最大幅はシステムによって異なり、データ・サーバのページ・サイズとカラムの総数で決まります。

ワイド・カラムは、プライマリ・キーおよびサーチャブル・カラムとして使用でき、複写定義の **where** 句でも使用できます。

注意： サブスクリプションやアーティクルの **where** 句の最大バイト数は 255 バイトです。このため、サブスクリプションやアーティクルの **where** 句では、ワイド・カラムを使用できません。

カラム数の増加
複写定義内のカラム数について明示的な制限はありません。Replication Server には、プライマリ・キー・カラム数またはサーチャブル・カラム数の制限はありません。

Replication Server では、プライマリ・キー・カラムを使用してデータ・サーバに対する SQL 文の **where** 句を作成します。したがって、複写定義内でプライマリ・キーとして実際に使用するカラム数を決定するときは、データ・サーバ側の制限値を考慮する必要があります。

同様に、Replication Server では複写定義内のサーチャブル・カラム数に制限はありませんが、サブスクリプションまたはアーティクルの **where** 句内のカラム数は、データ・サーバによる制限を受けることがあります。

ワイド・データ
データ・ローは、データ・サーバ上のデータ・ページと同じサイズにすることができます。Adaptive Server バージョン 12.5 以降では、2K、4K、8K、16K のページ・サイズをサポートしている。

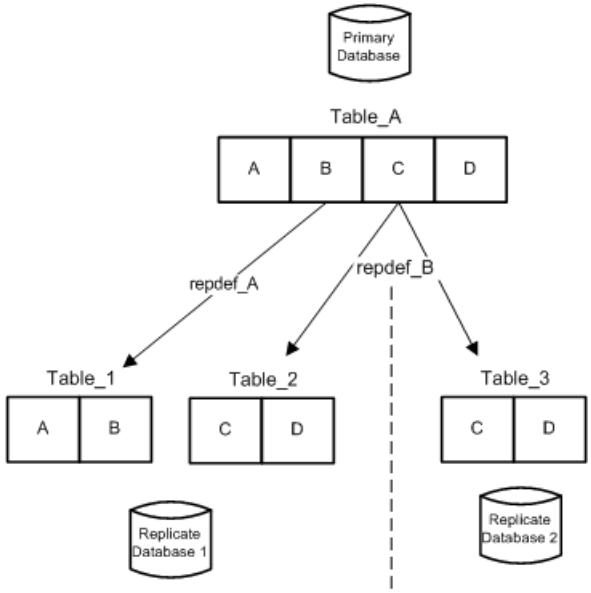
ワイド・メッセージ
Replication Server は、SQM が管理するステーブル・キューに、データ・ローをメッセージとしてコピーします。これらのメッセージには、データ・ローの複写前後のイメージとその他の情報が含まれます。そのため、もとになるデータ・ローよりもかなり多くの領域を必要とします。制限値の拡張によって、メッセージは複数ブロックにまたがることができるようになり、16K という制限がなくなりました。

テーブルごとに複数の複写定義を作成する方法

プライマリ・テーブルまたは他の複写テーブルと特性が異なる複写テーブルによってサブスクリプションを作成できるように、1 つのプライマリ・テーブルの複数の複写定義を作成し、各複写定義をカスタマイズできます。

たとえば、1 つのプライマリ・テーブルに対して 2 種類の複写定義を作成できます。一方はカラム A と B を複写し、もう一方はカラム C と D を複写します。「1 つのプライマリ・テーブルの複数の複写定義の使用」の図に示されているように、サブスクリプションを作成する各サイトは、必要なカラムだけを受け取ります。

図 19 : 1 つのプライマリ・テーブルの複数の複写定義の使用



各複写定義では、プライマリ・テーブルを記述するだけでなく、カラム数を制限したり、レプリケート・テーブルに対して別のカラム名、別のパブリッシュ・データ型、または別のテーブル名を指定することができます。指定された特性に一致するレプリケート・テーブルは、複写定義にサブスクリプションを作成することができます。

1 つのプライマリ・テーブルに対して作成した各複写定義では、同じ宣言がされたカラム・データ型を使用する必要があります (データ型が `rawobject` または `rawobject in row` の場合を除きます)。また、同じ **null** ステータスと **not null** ステータスを、`text`、`unitext`、`image` の各カラムに対して使用する必要があります。カラムのデータ型または null ステータスを変更するには、**alter replication definition** を使用します。

複写ステータスは、**alter replication definition** を使用して変更できます。たとえば、`text`、`unitext`、`image` カラムの複写ステータスを **replicate_if_changed** から **always_replicate** に変更できます。カラムの複写ステータスは、同じプライマリ・テーブルの他の複写定義でも変更されます。

参照：

- カラム・データ型の変更 (362 ページ)

複数の複写定義の制限

1 つのプライマリ・テーブルに複数の複写定義があるときに適用される制限があります。

- レプリケート・データベースは、複数の複写定義にサブスクリプションを作成できます。ただし、1 つのレプリケート・テーブルは、特定のプライマリ・テーブルの 1 つの複写定義に対してのみサブスクリプションを作成できます。
- バージョン 12.0 以前の Replication Server は、ユーザ定義のデータ型を使ったカラムを宣言する複写定義や、カラム・レベル変換を使用する複写定義に、サブスクリプションを作成できません。
- 1 つのプライマリ・テーブルに対して作成した各複写定義では、同じカラム・データ型を使用する必要があります (データ型が `rawobject` または `rawobject in row` の場合を除きます)。また、`text`、`unitext`、`image`、`rawobject` の各カラムに対して、同じ **null** ステータスまたは **not null** ステータスと、同じ複写ステータスを使用する必要があります。
- 単一のプライマリ・ストアド・プロシージャに複数の複写定義を作成することはできません。
- 1 つのプライマリ・テーブルに対する複数の複写定義は、Replication Server バージョン 11.5 以降でのみサポートされています。ただし、互換性がある場合 (同じプライマリ・テーブル名とレプリケート・テーブル名、同じプライマリ・カラム名とレプリケート・カラム名を持ち、テーブル所有者名を含んでいない場合) は、1 つの複写定義にマークを付けて、前のバージョンの Replication Server に送信できます。

参照：

- カラム名とデータ型の指定 (313 ページ)
- 混合バージョン・システムでの複写定義の制限 (327 ページ)

複写定義とファンクション文字列

ファンクション文字列は、データベースで実行するために、Replication Server ファンクションをデータ・サーバ・コマンドにマップします。

プライマリ Replication Server は、複写定義ごとに、複写定義スコープのシステム・ファンクションのデフォルト・ファンクション文字列を作成します (**rs_insert**、**rs_update**、**rs_delete**、**rs_select** など)。これらのファンクション文字列は、プライマリ Replication Server からのルートを持つ他の Replication Server に対して複写定義とともに分配されます。

システム・ファンクションのファンクション文字列を作成することが必要になる場合があります (つまり、Replication Server がこれらのファンクション文字列を自動的に作成しない場合があります)。

ファンクション文字列およびファンクション文字列クラスの詳細については、『Replication Server 管理ガイド 第 2 巻』の「データベース・オペレーションのカスタマイズ」を参照してください。

参照：
- データベース・オペレーションの指定 (46 ページ)

混合バージョン・システムでの複写定義の制限

混合バージョン・システムの複写定義には、いくつかの制限があります。

- 30 文字を超える識別子を含む複写定義を作成または変更する場合、このような複写定義のサブスクリプションを作成できるのは、15.0 以降の Replication Server だけです。
- Replication Server をバージョン 15.5 移行へアップグレードすると、サイト・バージョンが 1550 以降である場合に、複写定義の作成、変更、または削除のみ実行できます。
- **alter replication definition** を実行して複写定義からいくつかのカラムを削除し、サイト・バージョンが 1550 よりも古いレプリケート・サイトから複写定義へのサブスクリプションがある場合、プライマリ Replication Server は **alter replication definition** コマンドを拒否します。
- **alter replication definition** 要求を **with DSI_suspended** パラメータを指定して発行しても、サイトバージョンが 1550 より古いレプリケート DSI はサスペンドされません。

参照：
- 混合バージョンの複写システム (20 ページ)

複写対象テーブルへのマーク付け

テーブルの複写定義を作成したら、**sp_setreptable** を使用して、複写対象のテーブルにマークを付けます。

テーブルに複写のマークを付けると、RepAgent はテーブルのログ・レコードを Replication Server に転送し始めます。一度テーブルに複写のマークを付ければ、別の複写定義のために、再度そのテーブルに複写のマークを付ける必要はありません。

注意： Sybase 以外のデータ・サーバで複写対象のテーブルにマークを付ける方法については、使用している Replication Agent のマニュアルを参照してください。

参照：
- サブスクリプションの例 (439 ページ)

sp_setreptable システム・プロシージャの使用

sp_setreptable を使用するには、データベース所有者またはデータ・サーバのシステム管理者であることが必要です。

sp_setreptable コマンドの詳細については、『Replication Server リファレンス・マニュアル』の「Adaptive Server コマンドとシステム・プロシージャ」を参照してください。

複写の有効化

複写対象のプライマリ Adaptive Server テーブルを指定するには、**sp_setreptable** を使用します。

テーブルのデータベースを管理している Adaptive Server にログインし、次のように入力します。

```
sp_setreptable table_name, 'true'
```

このようにテーブルにマークを付けることによって、テーブル名がユニークでなければならないことが示されます。

注意： 該当のデータベースを管理する Replication Server でテーブルの複写定義も作成する場合を除き、複写対象のテーブルにマークを付けないでください。RepAgent は、Replication Server に対して、マークを付けられているテーブルに対するトランザクションのデータの転送を開始します。複写定義が存在しない場合、Replication Server はメッセージ 32032 をレポートし、そのエラー・ログ・ファイルが満杯になることがあります。また、Replication Server のパフォーマンスが大幅に低下する場合があります。ウォーム・スタンバイ・アプリケーションは複写定義を必要としないため、この問題は生じません。

複写ステータスの確認

sp_setreptable を使用すると、データベース内の特定のテーブルまたはすべてのテーブルの複写ステータスを確認できます。

テーブルの複写ステータスを確認するには、次のように入力します。

```
sp_setreptable table_name
```

データベース内のすべてのテーブルの複写ステータスを確認するには、次のように入力します。

```
sp_setreptable
```

SQL 文の複写の有効化

SQL 文の複写を有効化して設定するには、**sp_setrepdefmode** を使用します。

sp_setrepdefmode には、以下のためのオプションがあります。

- 特定の DML オペレーションについての SQL 文の複写の有効化または無効化。
- スレッショルドの設定。SQL 文の複写をアクティブにするには、このスレッショルドに達する必要がある。

SQL 文の複写に該当する DML オペレーションには、次のものがあります。

- **U** – update
- **D** – delete
- **I** – insert select

テーブルの複写モードを **UDI** の任意の組み合わせに設定すると、RepAgent は、指定された DML オペレーションでの SQL 文の複写を有効にするための追加の情報を送信します。

たとえば、**update**、**delete**、および **insert select** オペレーションでの SQL 文の複写をテーブル t で有効にするには、次を入力します。

```
sp_setrepdefmode t, 'UDI', 'on'
go
```

『Replication Server 管理ガイド 第 2 巻』の「パフォーマンス・チューニング」の「SQL 文の複写」を参照してください。

owner_on ステータスでの複写の有効化

ユーザ・テーブルは、同じ名前でも所有者が異なる場合があります。Adaptive Server を使用すると、テーブルに複写のマークを付けて、テーブルを識別するときに、テーブル所有者情報を考慮するように指定できます。

“owner on” ステータスでテーブルに複写のマークを付けるには、Adaptive Server にログインして、次のように入力します。

```
sp_setreptable table_name, 'true', owner_on
```

Replication Server では、テーブルの複写定義は、テーブル所有者を識別しなければなりません。たとえば、**sp_setreptable** でテーブルの所有者ステータスを “owner on” に設定した場合は、複写定義を作成するときに所有者名を含める必要があります。そうしないと、Replication Server はレプリケート・データベースで正しいテーブルを検出できません。

送信元テーブルの所有者と送信先テーブルの所有者は異なっていてもかまいません。

注意： テーブルに “owner off” ステータスを指定すると、Replication Server はテーブル所有者情報をレプリケート・サイトに送信しません。ただし、「スタンバ

イ・データベース」に複写する場合は、Replication Server はテーブル所有者を “dbo” として送信します。

Sybase 以外のデータ・サーバで、名前が同じで所有者が異なるユーザ・テーブルを使用できるかどうかは、使用している Replication Agent のマニュアルを参照してください。

テーブルの所有者ステータスの変更

sp_setrepdefmode システム・プロシージャを使用すると、以前に複写するようマーク付けされたテーブルの所有者ステータスを変更できます。

すでに複写するようマーク付けされているテーブルのステータスを “owner on” に変更するには、Adaptive Server にログインして、次のように入力します。

```
sp_setrepdefmode table_name, owner_on
```

すでに複写するようマーク付けされているテーブルのステータスを "owner off" に変更するには、Adaptive Server にログインして、次のように入力します。

```
sp_setrepdefmode table_name, owner_off
```

複写定義に所有者情報を含めることによって、所有者ステータスの変更を反映する必要があります。テーブル所有者を含む新しい複写定義を作成するには、**create replication definition** を Replication Server で使用してください。

テーブルの所有者ステータスのチェック

sp_setreptable システム・プロシージャを使用すると、テーブルの所有者ステータスをチェックできます。

次のように入力します。

```
sp_setreptable table_name
```

複写の無効化

sp_setreptable を **false** パラメータとともに使用してテーブルの複写をオフにします。

```
sp_setreptable table_name, 'false'
```

Sybase 以外のデータ・サーバで複写を無効にする方法については、使用している Replication Agent のマニュアルを参照してください。

Java カラムの複写

プライマリ・データベースに格納された Java カラムは、スタンバイ・データベースやレプリケート・データベースに複写できます。

Replication Server では、Java オブジェクトをまったく変更せずにそのまま直列化されたフォーマットで複写システムに渡します。Adaptive Server データベースの Java クラスについては、Adaptive Server Enterprise のマニュアル『Adaptive Server Enterprise における Java』を参照してください。

Java カラムの複写の制限

Java カラムの複写定義とサブスクリプションは通常の手順で作成できますが、適用される制限に注意してください。

- プライマリ・データベースとレプリケート・データベースは、両方とも Sybase Adaptive Server バージョン 12.0 以降である必要があります。
- Replication Server は、Java オブジェクトをパラメータとして持つストアド・プロシージャを複写しません。しかし、このようなストアド・プロシージャの効果は通常のテーブル複写によって複製できます。
- Java カラムは、プライマリ・キーの一部には使用できません。
- Java カラムはサーチャブルではないため、サブスクリプションの式で Java カラムを評価できません。

アップグレードの考慮事項

現在の Replication Server をアップグレードし、サイト・バージョンを現在のリリースに設定すると、Replication Manager のルート・アップグレード機能によって、Java カラムを含む複写定義がアップストリーム Replication Server から現在の Replication Server にコピーされます。

Replication Server は、Java カラムを持つ複写定義をバージョン 12.0 より前の Replication Server には送信しません。しかし、ファンクション文字列を操作することで、古いバージョンの Replication Server に Java カラムを複写できます。

参照：

- ファンクション文字列を使用して Java カラムを古い Replication Server に複写する (333 ページ)

Replication Server における Java データ型

Java カラムは、次の 2 種類の Replication Server データ型のいずれかとして、複写システムに受け渡されます。

- `rawobject` ― `image` データと同様に、データベース内の独立したロケーションに情報が格納されます。`rawobject` の基本データ型は `image` です。これが Replication Server における Java カラムのデフォルト・データ型です。Replication Server は、`image` データと同じ方法で `rawobject` データを扱います。
- `rawobject in row` ― `char` データなどと同様に、データベースの、テーブルに割り付けられている連続したデータ・ページ上に情報が格納されます。`rawobject in row` の基本データ型は `varbinary(255)` です。Replication Server による `rawobject in row` データの取り扱い方法は、`varbinary(255)` データの取り扱い方法と同じです。

`rawobject` と `rawobject in row` は、それぞれの基本データ型とのみ互換性があります。相互の互換性はありません。つまり、`rawobject` を `rawobject in row` に複写したり、その逆を行ったりすることはできません。

Replication Server の調整ユーティリティ **rs_subcmp** は、Java データ型を基本データ型として扱います。『Replication Server リファレンス・マニュアル』の「実行プログラム」の「**rs_subcmp**」を参照してください。

参照：

- text、unitext、image、および rawobject カラムの複写 (335 ページ)

Java カラムの複写定義の作成

create replication definition コマンドと、`rawobject` データ型および `rawobject in row` データ型を使用すると、Java カラムの複写定義を作成できます。

複写定義を作成するときには、次の点を考慮してください。

- `rawobject` 値には、複写ステータスがあります。常に複写するか、変更された場合にのみ複写するかを選択できます。これらの値には、null ステータスもあります。
- `rawobject in row` 値には、複写ステータスも null ステータスもありません。

`rawobject` 値と `rawobject in row` 値には、次の制限があります。

- プライマリ・キーの一部になれません。

- サブスクリプションの式では評価できません。Java カラムはサーチャブルではないので、サブスクリプションやアーティクルの **where** 句で使用できません。

次に、Java カラムを含むテーブル `t` のサンプルの複写定義 **p1** を作成する例を示します。

```
create replication definition p1
    with primary at ds.db
    with all tables name t
      (c1 int,
      c2 rawobject null,
      c3 rawobject not null,
      c4 rawobject in row)
    primary key (c1)
    replicate_if_changed (c2)
    always_replicate (c3)
```

カラム `c2` と `c3` は、`rawobject` カラムです。これらのカラムには、複写ステータスと null ステータスがあります。カラム `c4` は `rawobject in row` カラムです。このカラムには、複写ステータスも null ステータスもありません。カラム `c2`、`c3`、`c4` は、プライマリ・キーの一部でもサーチャブルでもありません。

参照：

- text、unitext、image、および rawobject カラムの複写 (335 ページ)

Java カラムのファンクション文字列

Replication Server は、**rs_raw_object_serialization** ファンクション文字列を使用して、Java カラムを直列化されたフォーマットでレプリケート・データベースに渡します。これにより、Replication Server は Java カラムを直接更新できるようになります。

rs_raw_object_serialization は、**rs_sqlserver_function_class** と **rs_default_function_class** に含まれています。

複写定義が `rawobject` データ型を参照している場合、Replication Server は `image` データの場合と同様に、`rawobject` カラムごとに **rs_get_textptr**、**rs_textptr_init**、**rs_datarow_for_writetext**、**rs_writetext** の各ファンクション文字列を作成します。

ファンクション文字列を使用して Java カラムを古い Replication Server に複写する

Replication Server バージョン 12.0 は、Java データ型を持つ複写定義をバージョン 12.0 より前の Replication Server に送信しません。ただし、対応する基本データ型 (`image` と `varbinary(255)`) を使用し、**rs_usedb** および **rs_insert** ファンクション文字列を操作すると、古いバージョンの Replication Server に Java カラムを複写できます。

次の例で、その方法を示します。

1. プライマリ・データベースとレプリケート・データベースに、Java カラムを持つテーブルを作成します。

```
create table tInfo
    (c1 integer,
    c2 Name rawobject in row,
    c3 Address rawobject null,
    c4 AccountInfo rawobject not null)
```

   `Name`、`Address`、`AccountInfo` は Java クラスであり、`c2`、`c3`、`c4` は Java カラムです。

2. テーブル `tInfo` の複写定義を作成します。

   12.0 より前のバージョンの Replication Server が 1 つでもある場合は、次のように、`rawobject in row` の基本データ型 (`varbinary(255)`) と `rawobject` の基本データ型 (`image`) を使用して複写定義を作成します。

```
create replication definition tInfo1
with primary at DS-1.dbase
with all tables name tInfo
(c1 integer,
c2 varbinary(255),
c3 image null,
c4 image not null,
primary key (c1)
...
```

   プライマリ・データベースとレプリケート・データベースが Replication Server バージョン 12.0 以降で管理されている場合、複写定義は次のようになります。

```
create replication definition tInfo
with primary at DS-1.dbase
with all tables named tInfo
    (c1 integer,
    c2 rawobject in row,
    c3 rawobject null,
    c4 rawobject not null)
    primary key (c1)
    ...
```

3. プライマリ・データベース・コネクションとレプリケート・データベース・コネクションの **rs_usedb** と **rs_insert** の各ファンクション文字列を変更します。『Replication Server 管理ガイド 第 2 巻』の「データベース・オペレーションのカスタマイズ」の「ファンクション文字列の管理」の「alter function string」を参照してください。

   - **rs_usedb** の場合

```
alter function string rs_usedb
for function_string_class_name
output language
```

```
'use ?rs_destination_db!sys_raw? set
raw_object_serialization on'
```

この変更は、Adaptive Server に対して、サブスクリプション・マテリアライゼーションで Java カラム・データを連続したバイナリ値として返すよう通知します。また、Replication Server は、Java カラムを連続したバイナリ値で挿入または更新できるようになります。

- **rs_insert** の場合

```
alter function string tInfo1.rs_insert
for function_string_class_name
output language
'insert tInfo(c1, c2, c4)
values (?c1!new?, ?c2!new?, 0xaced000574000130)'
```

これにより、特別なバイナリ値 0xaced000574000130 をカラム **c4** に挿入するように、`tInfo1` の **rs_insert** が変更されます。**rs_insert** を変更しないでデフォルト値のままにしておくと、Adaptive Server は直列化エラーを返します。

このように、同じテーブルに対して、カラムのプライマリ (宣言した) データ型が異なる 2 つの複写定義を作成できます。プライマリ Replication Server がバージョン 12.0 以降である場合は、テーブル **tInfo** に対して **tInfo** と `tInfo1` の両方の複写定義を作成できます。この場合、レプリケート Replication Server がバージョン 12.0 以降であれば、**tInfo** にサブスクリプションを作成できます。バージョン 12.0 より前ならば、**tInfo1** にサブスクリプションを作成できます。

注意： この方法で、Java カラムをスタンバイ・データベースに複写することはできません。スタンバイ・コネクションでは、**rs_default_function_class** ファンクション文字列クラスを使用しますが、これを変更することはできません。

text、unitext、image、および rawobject カラムの複写

Replication Server を使用すると、`text`、`unitext`、`image`、`rawobject` の各 Adaptive Server データ型を使用するカラムを複写できます。

- `text`、`unitext`、`image`、`rawobject` カラムを複写するときは、複写定義と Adaptive Server の `text`、`unitext`、`image`、`rawobject` の各カラムに、互換性のある複写ステータスを指定します。
- プライマリ・キーの一部またはサーチャブル・カラムとして、`text`、`unitext`、`image`、または `rawobject` カラムを含めることはできません。
- `text`、`unitext`、`image`、または `rawobject` カラムの複写がターゲット・データベースの 1 つのローだけに影響するように、ユニークなカラム・セット

を指定します。複写定義を使用している場合は、このカラムまたはカラム・セットをプライマリ・キーに含めます。

- text、unitext、image、rawobject カラムを複写するには、次の手順に従います。
  - **create replication definition** を使用して、text、unitext、image、または rawobject カラムを含むテーブルの複写定義を作成します。

    『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**create replication definition**」を参照してください。
  - **sp_setreptable** を使用して、複写対象のテーブルにマークを付けます。

    『Replication Server リファレンス・マニュアル』の「Adaptive Server コマンドとシステム・プロシージャ」の「**sp_setreptable**」を参照してください。
- **sp_setreptable** は、text、unitext、image、または rawobject カラムの複写ステータスを **always_replicate** に設定します。
- text、unitext、image、または rawobject カラムの一部を複写しない場合は、**sp_setrepcol** を使用してそれらのカラムの複写ステータスを変更します。
- **create subscription** を使用して複写定義のサブスクリプションを作成し、text、unitext、image、または rawobject データの複写を開始します。

注意： プライマリ・データベースで更新を実行するときは、text、unitext、image、または rawobject カラムとそれ以外のカラム (char カラムなど) を 1 つのコマンドで更新できます。ただし、これらの更新をレプリケート・データベースにコピーするときには、レプリケート・サーバは text、unitext、image、rawobject の更新用コマンドと、その他のデータ型の更新用コマンドの 2 つのコマンドを実行します。DSI が特定の複写エラーを無視するように設定していると、ローの一部しか複写されないことがあります。この場合、作成されるレプリケート・テーブルの整合性が失われることになります。

ASE 以外のデータ・サーバについては、『Replication Server 異機種間複写ガイド』と Replication Server Options のマニュアルを参照してください。

参照：

- text、unitext、image、または rawobject カラムのステータスの変更 (339 ページ)
- create subscription コマンド (431 ページ)

DirectConnect Anywhere を使用した ASE 以外のサーバへのラージ・オブジェクトの複写

Replication Server は、`writetext` コマンドを ECDA に渡し、`update` 文に変換することによって、**text** や **image** などのラージ・オブジェクトを ASE 以外のサーバに複写します。

writetext コマンドには、レプリケート・データベースを検索して送信するために **update** 文が使用するラージ・オブジェクト・ポインタが含まれます。ほとんどのデータ・サーバは、ラージ・オブジェクトを更新するユニークな実装を独自に備えています。そのため、これらのサーバにラージ・オブジェクトを複写すると、更新ごとにレプリケート・データベースの完全なテーブル・スキャンが必要になることが頻繁にあるため、時間がかかり、効率的ではありません。

Replication Server には、ECDA に送信される **writetext** コマンドにプライマリ・キーを含めるためのオプションが用意されています。プライマリ・キーにより、ECDA は、レプリケート・データベースの検索と複写を効率的に実行できる **update** 文を作成できます。

Replication Server には、データ・サーバ・インタフェース (DSI: Data Server Interface) の設定パラメータ **dsi_alt_writetext** が導入されています。**dsi_alt_writetext** を使用すると、テキスト・ポインタまたは一連のプライマリ・キーを **writetext** コマンドに含めるように Replication Server に指示できます。

注意： この機能を使用するには、ECDA 15.0 ESD #2 が必要です。

詳細については、『Replication Server リファレンス・マニュアル』を参照してください。

text、unitext、image、または rawobject の複写定義作成のガイドライン

`text`、`unitext`、`image`、または `rawobject` カラムのテーブル複写定義を作成するときは、いくつかのガイドラインに従う必要があります。

- 複写する `text`、`unitext`、`image`、または `rawobject` の各カラムをカラム・リストに含めます。
- `text`、`unitext`、`image`、または `rawobject` の各カラムのデータ型を指定します。
- 送信先テーブルのカラムに **null** を使用できるかどうかを指定します。この設定は、送信元テーブルと送信先テーブルの定義方法と一貫している必要があります。
- 各カラムをオプションの **replicate_if_changed** 句または **always_replicate** 句に含めます。

text、unitext、image、rawobject カラムへの null 値の指定

レプリケート・テーブルで、text、unitext、image、または rawobject の各カラムに null 値を使用できるかどうかを指定するには、複写定義でカラムのデータ型の後に **null** または **not null** を使用します。

この設定は、プライマリ・テーブルとレプリケート・テーブルの定義方法と一貫している必要があります。text、unitext、image、rawobject カラムのデフォルト値は **not null** です。これは、レプリケート・テーブルが null 値を受け入れないことを示します。

複数の複写定義を使用する場合、null 値の設定は、プライマリ・テーブルのすべての複写定義で同じでなければなりません。

text、unitext、image、または rawobject 以外のデータ型を使用するカラムには、**null** または **not null** を指定しないでください。null 値を含むカラムは、サーチャブル・カラムにはできません。

次に示すテーブル au_pix の複写定義の例には、image データ型のカラム pic があります。このカラムには、レプリケート・テーブルで null 値を使用できます。pic カラムは、**replicate_if_changed** 句に含まれています。

```
create replication definition au_pix
with primary at TOKYO_DS.pubs2
(au_id char(11),
pic image null)
primary key (au_id)
replicate_if_changed (pic)
```

text、unitext、image、または rawobject カラムを持つテーブルのマーク付け

sp_setreptable を使用すると、テーブルに複写のマークを付けるときに、Adaptive Server の text、unitext、image、rawobject カラムに初期複写ステータスを設定できます。

sp_setreptable は、text、unitext、image、または rawobject カラムの複写ステータスを **always_replicate** に設定します。

注意： text、unitext、image、rawobject カラムを複写しない場合は、**sp_setreplicate** を使用して複写対象のテーブルにマークを付けます。これにより、text、unitext、image、rawobject カラムの複写ステータスが **do_not_replicate** に設定されます。

sp_setreptable を使用して複写対象のテーブルにマークを付け、そのテーブルに text、unitext、image、および rawobject カラムが含まれている場合は、

テーブルのすべてのデータ・ローの text、unitext、image、または rawobject のすべてのカラムに対して、内部オペレーションを完了する必要があります。この内部変更は、1 つのトランザクションで実行されます。大きなテーブルの場合、このオペレーションに時間がかかり、大量のデータ処理を伴うことがあります。

text、unitext、image、または rawobject カラムを持つ大きなテーブルで **sp_setreptable** を使用する前に、このオペレーションに必要なログ・スペースが十分にあることを確認してください。また、クライアント・アプリケーションや複写システム管理への影響が最も少ない時間を選ぶようにします。

オプションの **use_index** を使用すると、text、unitext、image、または rawobject を持つテーブルのマーク付け処理を高速化できます。このオプションを使用すると、Adaptive Server は、テーブル内の各 text、unitext、image、または rawobject の内部ノンクラスタード・インデックスを作成します。次に例を示します。

```
sp_setreptable aux_pix, true, null, use_index
```

『Replication Server リファレンス・マニュアル』の「Adaptive Server コマンドとシステム・プロシージャ」の「**sp_setreptable**」を参照してください。

注意： Sybase 以外のデータ・サーバで複写対象のテーブルにマークを付ける方法については、使用している Replication Agent のマニュアルを参照してください。

text、unitext、image、または rawobject カラムのステータスの変更

sp_setreptable を使用して、text、unitext、image、または rawobject カラムの複写ステータスを調整します。

text、unitext、image、または rawobject カラムを持つ複写対象のテーブルにマークを付けると、**sp_setreptable** は text、unitext、image、または rawobject カラムの複写ステータスを **always_replicate** に設定します。

複写ステータスは、1 つのプライマリ・テーブルのすべての複写定義で同じです。**alter replication definition** を使用して、1 つの複写定義の複写ステータスを変更した場合、同じプライマリ・テーブルのすべての複写定義の複写ステータスを変更したことになります。

マーク付けされたテーブルの text、unitext、image、または rawobject カラムの一部を複写しない場合 (または **sp_setreplicate** を使用してテーブルにマークを付け、text、unitext、image、rawobject カラムの複写ステータスを **do_not_replicate** に設定した場合) は、**sp_setrepcol** を使用して複写ステータスを調整します。1 つまたはすべてのカラムの複写ステータスを **always_replicate**、**do_not_replicate**、または **replicate_if_changed** に設定できます。

sp_setrepcol を使用するには、データベース所有者またはデータ・サーバのシステム管理者であることが必要です。

注意： **sp_reptostandby** を使用してスタンバイ・データベースへの複写対象のデータベースにマークを付け、**sp_setreptable** を使用してレプリケート・データベースへの複写対象のデータベース・テーブルにマークを付けた場合、Replication Server は、text、unitext、image、rawobject カラムを **always_replicate** として、スタンバイ・データベースとレプリケート・データベースにコピーします。text、unitext、image、rawobject カラムを **replicate_if_changed** としてコピーする場合は、**sp_setrepcol** を使用して複写ステータスを調整します。ウォーム・スタンバイ・アプリケーションでの複写の詳細については、『Replication Server 管理ガイド 第 2 巻』の「ウォーム・スタンバイ・アプリケーションの管理」の「ウォーム・スタンバイの複写情報」の「ウォーム・スタンバイ・アプリケーションでの text、unitext、image、rawobject データの複写」を参照してください。

『Replication Server リファレンス・マニュアル』の「Adaptive Server コマンドとシステム・プロシージャ」の「**sp_setrepcol**」を参照してください。

text、unitext、image、rawobject カラムの複写ステータス

text、unitext、image、rawobject カラムの複写ステータスを設定するには、3 つのステータス句があります。

表 24 : text、unitext、image、rawobject カラムの複写ステータス

ステータス句	説明
always_replicate	Adaptive Server は、ローのカラムが変更されるたびに、text、unitext、image、または rawobject カラムの複写情報をログに記録する。
replicate_if_changed	Adaptive Server は、カラム・データが変更された場合にのみ、text、unitext、image、または rawobject カラムの複写情報をログに記録する。
do_not_replicate	Adaptive Server は、text、unitext、image、または rawobject カラムの複写情報をログに記録しない。

カラム複写の有効化

sp_setrepcol を使用して、image データ型または rawobject データ型のカラムに複写のマークを付けます。

構文：

```
sp_setrepcol table, column, status
```

たとえば、テーブル au_pix で image データ型の pic カラムに複写のマークを付けるには、次のいずれかを入力します。

- sp_setrepcol au_pix, pic, always_replicate
- sp_setrepcol au_pix, pic, replicate_if_changed
- sp_setrepcol au_pix, pic, replicate_if_changed, use_index

カラム複写の無効化

sp_setrepcol を **do_not_replicate** パラメータとともに使用して、image または rawobject カラムの複写をオフにします。

構文：

```
sp_setrepcol table, column, do_not_replicate
```

たとえば、テーブル au_pix で image データ型の pic カラムの複写を無効にするには、次のように入力します。

```
sp_setrepcol au_pix, pic, do_not_replicate
```

すべてのカラムの複写の有効化／無効化

テーブル内の同じ複写ステータスを持つ text、unitext、image、rawobject のすべてのカラムにマークを付けるには、カラム名の代わりに “null” を入力します。

たとえば、**replicate_if_changed** ステータスを持つ text、unitext、image、rawobject のすべてのカラムにマークを付けるには、次のように入力します。

```
sp_setrepcol table_name, null, replicate_if_changed
```

指定したカラムの複写ステータスを表示するには、**sp_setrepcol** にテーブル名と text、unitext、image、および rawobject カラム名を指定して実行します。次に例を示します。

```
sp_setrepcol table_name, column_name
```

テーブル内のすべての text、unitext、image、rawobject カラムの複写ステータスを表示するには、テーブル名を指定して **sp_setrepcol** を実行します。次に例を示します。

```
sp_setrepcol table_name
```

text、unitext、image、rawobject カラムの複写ステータスの変更

text、unitext、image、rawobject カラムを複写するときは、複写定義と Adaptive Server の各カラムに互換性のある複写ステータスを指定します。

1 つのテーブル複写定義で text、unitext、image、rawobject カラムの複写ステータスを変更すると、これらの text、unitext、image、rawobject カラ

ムを含む同じテーブルの他のすべての複写定義で複写ステータスが自動的に変更されます。

プライマリ・データベースが **rs_send_repserver_cmd** ストアド・プロシージャをサポートしている場合、以下によりプライマリ・データベースの text、unitext、image、rawobject カラムの複写ステータスを変更できます。

1. **sp_setrepcol** を使用して、カラムの複写ステータスを変更します。次に例を示します。
   ```
   sp_setrepcol authors, au_pix, replicate_if_changed
   ```
2. **rs_send_repserver_cmd** を使用して複写ステータスを変更するための複写定義の変更要求を実行します。次に例を示します。
   ```
   exec rs_send_repserver_cmd 'alter replication definition authors
   replicate_if_changed au_pix'
   ```

プライマリ・データベースが **rs_send_repserver_cmd** をサポートしていない場合、またはデータベース・ログにテーブルのデータ・ローがない場合は、以下のようにします。

1. データベースのカラムのステータスを新しい複写ステータスに変更するには、**sp_setrepcol** を使用します。
2. カラムのステータスを新しい複写ステータスに変更するには、**alter replication definition** を使用します。

replicate_if_changed ステータスのサブスクリプションの問題

replicate_if_changed ステータスの text、unitext、image、または rawobject カラムを持つ複写定義にサブスクリプションを作成する場合は、text、unitext、image、または rawobject データのバルク・マテリアライゼーションまたはマテリアライゼーションを使用します。

参照：

- バルク・マテリアライゼーション (411 ページ)
- text、unitext、image、rawobject データのマテリアライズ (442 ページ)

text、unitext、image データを複写するためのファンクション文字列

text、unitext、または image データ型のカラムを Adaptive Server 以外のデータベースに複写する場合は、text、unitext、または image カラムごとに、**rsdatarow_for_writetext**、**rs_get_textptr**、**rs_textptr_init**、**rs_writetext** の各ファンクション文字列を作成します。

ファンクション文字列名には、複写定義の text、unitext、または image カラム名を使用します。

注意： rawobject カラムまたは rawobject in row カラムは、それぞれの基本データ型として複写しないかぎり、Sybase 以外のデータベースには複写できません。rawobject の基本データ型は image であり、rawobject in row の基本データ型は varbinary です。

『Replication Server リファレンス・マニュアル』の「Replication Server システム・ファンクション」を参照してください。

ラージ・オブジェクト (LOB) データ型の複写

Replication Server は、Microsoft SQL Server 2005 の varchar(max)、nvarchar(max)、varbinary(max) の各データ型の複写をサポートしています。これらのデータ型は、それぞれ 2,147,483,647 バイトまでのデータを格納できます。

Replication Server では、テーブル・レベルの複写環境でのユーザ定義データ型 (UDD: User-Defined Datatype) として LOB データ型が導入されています。また、Replication Server は、新しい LOB データ型に対してデータベース・レベルの複写をサポートしています。これらの LOB データ型は、text、unitext、image の各データ型に直接マップされます。

表 25 : 基本データ型への LOB データ型のマッピング

新しい LOB データ型	基本となる型
varchar(max)	text
nvarchar(max)	unitext
varbinary(max)	image

LOB データ型の制限事項

LOB データ型には、いくつかの制限事項があります。

- LOB カラムは Microsoft SQL Server のみに複写できます。
- 複写定義で LOB カラムをプライマリ・キーとして定義できない。
- 複写定義で LOB カラムをサーチャブルとして定義できない。
- 新しい LOB データ型のいずれかをパラメータとして含むストアド・プロシージャを複写できない。
- テキスト・ポインタを使用して、新しい LOB データ型のデータを操作できない。

混合バージョン環境では、プライマリ Replication Server とレプリケート Replication Server のサイト・バージョンが 15.2 以降、LTL バージョンが 710 以降であることが必要です。

ASE 以外のデータ・サーバに関する LOB データ型の問題については、『Replication Server 異機種間複写ガイド』を参照してください。一般的な LOB データ型の詳細については、『Replication Server リファレンス・マニュアル』を参照してください。

LOB データ型の部分更新

部分更新トランザクションでは、完全更新のように **delete** コマンドと **replace** コマンドを発行しないで、テーブル・カラムのユーザ定義位置に文字列を直接書き込みます。

部分更新を実装するには、**rs_updatetext** LTL コマンドを次のように使用します。

```
{distribute|_ds} command_tags {applied|_ap} 'table'.rs_updatetext
{partialupd|_pu} [{first|_fi}] [last] [{changed|_ch}] [with log]
[{withouttp|_wo}] [{offset|_os}=offset {deletelen|
_dln}=deletelength]
[{textlen|_tl}=length] text_image_column
```

部分更新では、複数の文字セットの変換はサポートしていません。この変換は、Microsoft SQL Server 2005 でのみサポートされます。

計算カラムの複写

計算カラムを使用すると、式を作成したり、式の結果をテーブル・カラムに挿入したりできます。

計算カラムには、次の 2 種類があります。

- マテリアライズ ― 挿入または更新ごとに値が計算される。マテリアライズされた計算カラムは、通常カラムと同様に格納される。
- 非マテリアライズ ― クエリで参照された場合にのみ、値が計算される。仮想計算カラムは、テーブルまたはインデックス・ページに格納される。

計算カラムの式には、次のものがあります。

- 決定性 (deterministic) ― いつ評価されても値は同じになる。
- 非決定性 (nondeterministic) ― 評価されるたびに値は異なる (日付スタンプなど)。

計算カラムの作成と管理の詳細については、『Adaptive Server Enterprise システム管理ガイド』を参照してください。

Replication Server は、他のカラムの複写と同様に、マテリアライズされた計算カラムを DML 文で複写します。非マテリアライズ計算カラムは複写しません。

計算カラムの複写は、ファンクション文字列によってサポートされています。Replication Server バージョン 15.0 以降では、コネクションの確立時にレプリケート・データベース DSI で、クラス・レベルのファンクション文字列 **rs_set_dml_on_computed** が適用されます。これは、**set dml_on_computed "on"** を **use database** 文の後に発行します。レプリケート Adaptive Server のバージョンが 12.5.x 以前である場合、このコマンドは無視されます。

以下を含むテーブルの複写定義を作成または変更する場合は、次のようになります。

- deterministic カラム - 複写定義にこれらのカラムを含めるかどうかを選択できる。deterministic カラムは、常に同じ値となるため、これらを含めなくても複写定義を作成できる。また、複写された各挿入および各更新は、レプリケート・データベースで値を計算できる。
- 非決定性計算カラム ― プライマリ・データベースとレプリケート・データベースの同期を維持するには、複写定義に非決定性計算カラムを含める必要がある。

暗号化カラムの複写

Replication Server バージョン 15.0 では、Adaptive Server での暗号化カラムの複写をサポートしています。

Replication Server は、クリア・テキスト値ではなく暗号化テキスト値として、バイナリ・フォーマットでプライマリ Adaptive Server データベースの暗号化カラムを複写します。

プライマリ・データベースとレプリケート・データベースの暗号化キーは、同じであることが必要です。レプリケート・データベースで複写を使用して暗号化キーを作成するか、**dump** および **load** コマンドを使用して暗号化キーが同じであることを確認します。

ウォーム・スタンバイ環境および MSA 環境の Replication Server は、暗号化キーの **create**、**alter**、**drop** の各コマンドを複写します。また、カラムを暗号化または復号化するために、**alter table** も複写します。**create**、**alter**、**drop encryption key** の各 DDL コマンドを複写するには、プライマリ・データベースとレプリケート・データベースの **system_encr_passwd** が同じである必要があります。

暗号化キーが別のデータベースに格納されている場合は、それらの暗号化キーを使用して暗号化されたカラムを含むデータベースと同時に同期されていることを確認します。

古い暗号化キー、または初期化ベクトルとパディング間の相違が原因で、プライマリ・データベースとレプリケート・データベース間でデータが異なる場合は、データを手動で同期して **update** 文と **delete** 文が失敗しないようにします。

制限
暗号化カラムの複写には、制限があります。

- `text` カラムと `image` カラムは暗号化できません。
- Replication Server は暗号化テキストで値を受け取り、その値をクリア・テキスト値と比較することができないため、暗号化カラムをサブスクリプションまたはアーティクルの **where** 句で使用することはできません。暗号化カラムをサーチャブル・カラムとして使用することはできません。
- 暗号化カラムをプライマリ・キーで使用する場合、INIT_VECTOR NULL と PAD NULL を使用して暗号化キーを定義する必要があります。
  初期化ベクトルとパディングは、同じキーで暗号化された 2 つの類似する値が 2 つの異なる暗号化テキストになるように、暗号化テキストをランダム化することを目的としています。プライマリ・サイトとレプリケート・サイトで暗号化されたデータの暗号化テキストが異なる場合、Replication Server がプライマリ・サイトの更新前イメージをレプリケート・サイトのデータと一致させようとしても、この試行は失敗します。
  初期化ベクトルが使用されていない場合は、送信元データベースと送信先データベースの暗号化テキストが完全に一致しています。Replication Server は、暗号化カラムの暗号化テキストを使用して、**update/delete** 文の **where** 句を発行するため、この一致が必須となります。
- ウォーム・スタンバイ環境または MSA 環境でテーブルのデータを複写するときに、テーブル複写定義を使用しない場合は、INIT_VECTOR NULL および PAD NULL として定義されたキーで、そのテーブルに含まれるすべての暗号化カラムを暗号化する必要があります。
- 複写定義で宣言するすべての暗号化カラムは、`varbinary` フォーマットにする必要があります。カラムの長さを判別するには、Adaptive Server Enterprise の『暗号化カラム・ユーザーズ・ガイド』の「データの暗号化」の「暗号化カラムの長さ」を参照してください。

注意： **rs_subcmp** は、Adaptive Server での暗号化カラムの複写をサポートしています。

暗号化データの複写

Replication Server は、暗号化データの複写をサポートしています。

サイトでスキーマの変更が複写される場合、次の DDL 文が複写されます。

- **alter encryption key**
- 暗号化の拡張指定を含む **create table** と **alter table**
- **create encryption key**
- **grant create encryption key** と **revoke create encryption key**
- キーの **grant select** と **revoke select**
- カラムの **grant decrypt** と **revoke decrypt**
- **sp_encryption system_encr_passwd**
- **drop encryption key**

キーは、暗号化された形式で複写されます。

システムで DDL が複写されない場合、レプリケート・サイトで、暗号化キーを手動で同期化します。Adaptive Server の **ddlgen** ユーティリティは、キー値の複写のための特殊な形式の **create encryption key** をサポートしています。**ddlgen** ユーティリティの詳細については、Adaptive Server のマニュアルを参照してください。

DML の複写の場合、**insert** コマンドと **update** コマンドによって、暗号化された形式のまま暗号化カラムを複写します。この方法では、複写されたデータが、Replication Server によってディスク上のステーブル・キュー内で処理される間も保護されます。

特殊データ型の使用

複写定義で `rs_address`、`identity`、`timestamp` 特殊データ型のカラムを使用する方法について説明します。

rs_address データ型の使用

`rs_address` 特殊データ型を使用すると、ユニークなサブスクリプション解析手法が可能になります。`rs_address` データ型のビットマップ (基本となる `int` データ型に基づく) を、サブスクリプションの **where** 句のビットマスクと比較して、ローをレプリケートすべきかどうかを判断できます。

前提条件

where 句を使用してビットマップ比較を実行するには、`rs_address` データ型を使用するすべてのカラムを複写定義の **searchable columns** 句に含める必要があります。

手順

1. このサブスクリプション解析メソッドを使用するには、int データ型のカラムを使用するテーブルを作成してください。
2. これらのカラムをカラム・リストに含める複写定義を作成します。ただし、データ型を int ではなく rs_address として宣言します。

参照：
- ビットマップ・サブスクリプション (445 ページ)

identity カラムの複写

identity カラムには、自動的に生成される連続番号 (請求書番号、従業員番号、レコード番号など) が格納されます。identity カラムの値によって、テーブルの各ローがユニークに識別されます。

identity カラムでは、1 ～ 10^{38} -1 の位取り 0 の numeric 基本データ型を使用します。

identity カラムは、**update** コマンドによって更新されることはありません。レプリケート・サイトから (要求ファンクションを使用して) プライマリ・データに適用される **update** は、identity データを含む identity カラムを更新できません。

複写定義への identity カラムの指定

identity カラムを含むテーブルの複写定義を作成するには、カラムの宣言したデータ型として identity を指定するか、カラム・レベル変換を使用して、カラムのパブリッシュ・データ型として identity を指定します。

1 つのテーブルに対する 1 つまたは複数の複写定義は、identity データ型のカラムを 1 つしかパブリッシュできません。

1 つの複写定義でカラムを identity としてパブリッシュした場合、別の複写定義では同じカラムを numeric としてパブリッシュし、この 2 番目の複写定義へのサブスクライバの挿入によって余分なコマンドが送信されないようにします。

Replication Server による identity カラムの複写方法

identity カラムを複写するときに Replication Server が適用するコマンドについて説明します。

Replication Server は、レプリケート・テーブルに次のコマンドを適用してから、**insert** を実行します。

```
set identity_insert table_name on
```

insert の実行後、Replication Server はレプリケート・テーブルに次のコマンドを適用します。

```
set identity_insert table_name off
```

identity カラムを含むテーブルすべてに対して、Transact-SQL の **identity_insert** オプションを使用するには、メンテナンス・ユーザはレプリケート・データベースのそのテーブルの所有者でなければなりません (または、“dbo” ユーザであるか、“dbo” ログイン名のエイリアスでなければなりません)。

timestamp カラムの複写

Replication Server は、Replication Server のデータ型として timestamp をサポートしています。

timestamp は varbinary(8) として定義され、ステータス・ビット・インジケータによって varbinary と区別されます。このデータ型を使用すると、レプリケート・データベース、スタンバイ・データベース、MSA データベースに timestamp カラムを複写できます。

また、複写定義で timestamp をプライマリ・キーとして定義したり、複写定義やファンクション複写定義でサーチャブル・カラムとして定義したりできます。

timestamp の複写をサポートするために、**send_timestamp_to_standby** 設定パラメータも追加されています。**send_timestamp_to_standby** が有効であり、複写定義が存在しない場合は、timestamp カラムがレプリケート・データベースに送信されます。

テーブルに timestamp カラムが含まれている場合、メンテナンス・ユーザは、レプリケート・データベースの該当のテーブルの所有者である必要があります。または、レプリケート・データベースの “dbo” ユーザであるか、“dbo” ログイン名のエイリアスである必要があります。

複写定義での timestamp カラムの指定

timestamp カラムを含むテーブルの複写定義を作成するには、カラムの宣言したデータ型として timestamp を指定するか、カラム・レベル変換を使用して、カラムのパブリッシュ・データ型として timestamp を指定します。

1 つのテーブルの 1 つまたは複数の複写定義で、timestamp データ型の複数のカラムをパブリッシュすることはできません。

注意： 複写定義で timestamp をサポートするには、レプリケート Adaptive Server がバージョン 15.0.2 以降であることが必要です。

『Replication Server リファレンス・マニュアル』の「トピック」の「データ型」の「日付と時間のデータ型」を参照してください。

複写定義の修正

複写定義を表示、変更、および削除する方法と、テーブルの変更を Replication Server がどのようにサポートするかについて説明します。

また、次のような場合に、**alter table** コマンドによるテーブルの変更を Replication Server がどのようにサポートするかについても説明します。

- テーブルにレプリケート・サイトからのサブスクリプションがある場合
- **send standby replication definition columns** 句を指定した複写定義を使用して、テーブルをスタンバイ・データベースに複写した場合

注意： Adaptive Server Enterprise バージョン 12.0 以降では、既存テーブルの変更 (null 入力不可能なカラムの追加、カラムの削除、カラム・データ型の変更) が可能です。

サブスクリプションのないウォーム・スタンバイ・テーブルに影響を及ぼす **alter table** 変更については、『Replication Server 管理ガイド 第 2 巻』の「ウォーム・スタンバイ・アプリケーションの管理」の「ウォーム・スタンバイ・データベースの複写定義とサブスクリプション」の「ウォーム・スタンバイにおける alter table のサポート」を参照してください。

参照：

- 複写データの修正 (365 ページ)
- 混合バージョン・システムでの複写定義の制限 (327 ページ)

テーブル・スキーマの管理

対応する複写定義の変更に合わせてプライマリ・スキーマを調整したり、ストアド・プロシージャの変更を保存したりするには、**rs_send_repserver_cmd** ストアド・プロシージャを使用してプライマリ・スキーマまたはストアド・プロシージャを変更しながら、プライマリ・データベースで複写定義要求を実行します。

Replication Server は、テーブルの複写定義にテーブル・スキーマに関する情報を格納します。

Replication Server は、複写定義を変更するときに新しい複写定義を作成する場合があります。Replication Server は、新しいバージョンの複写定義を使用して Replication Server システムに入力される新しいデータ・ローを処理しながら、古いバージョンの複写定義を使用して複写システム内に既にある古いデータ・ローを処理します。

rs_send_repserver_cmd をサポートしないデータベースでは、プライマリ Replication Server で複写定義の変更要求を発行する前に、データベース・ログに古い複写定義を使用するデータ・ローがないことを確認します。

参照：

- 複写定義の変更要求プロセスの使用 (356 ページ)

複写定義と複写定義バージョンの表示

既存の複写定義の情報を表示するには、**rs_helprep** ストアド・プロシージャおよび **rs_helpreptable** ストアド・プロシージャを使用します。複写定義バージョンの情報を表示するには、**rs_helprepversion** ストアド・プロシージャを使用します。

ストアド・プロシージャの使用方法については、『Replication Server リファレンス・マニュアル』の「RSSD ストアド・プロシージャ」を参照してください。

複写定義の変更

カラムがプライマリ・テーブルに追加された後、または送信先データベースが元の複写定義に指定されていないカラムを必要とする場合は、複写定義の変更が必要となることがあります。

ほとんどの場合、複写定義を変更するときには、送信元または送信先テーブルのデータベース・スキーマも変更します。送信元サイトと送信先サイト間でスキーマの変更を必ず調整します。

参照：

- 複写データの修正 (365 ページ)

複写定義の変更要求プロセス

複写定義への変更を要求すると、Replication Server によって複写定義の変更とデータ複写の伝達が自動的に調整されます。

複写定義の変更は、データベース・スキーマの変更中に、**alter replication definition**、**alter applied replication definition**、または **alter request function replication definition** コマンドを使用してプライマリ・データベースで直接要求できます。

Replication Server 内では、以下のことができます。

- プライマリ・データベースから直接複写定義コマンドを発行できる。
- ターゲット・データベースの古いバージョンの複写定義で、Replication Server がすべてのデータを適用してから、Replication Server がターゲット DSI をサスペンドするように指示する **alter replication definition** コマンドを使用できる。これにより、新しいバージョンの複写定義が到着する前にターゲット・スキーマ

の変更とカスタム・ファンクション文字列の変更を行うためのウィンドウが表示される。
- Replication Server が複写定義の変更要求を実行できることを確認するために、データの変更なしで変更要求を実行できる。
- Replication Agent から送信された複写定義要求で失敗したものを Replication Server に省略させることができる。プライマリ Replication Server で複写定義コマンドが失敗すると、Replication Agent が停止する。Replication Server がそのコマンドを省略しない限り、Replication Agent を再起動すると、失敗したコマンドが再び実行される。

注意： Adaptive Server の他に、Replication Server では、**rs_send_repserver_cmd** のサポートが、これらの ASE 以外のデータベースのサポートされているバージョン (Microsoft SQL Server および Oracle) まで拡張されています。したがって、IBM DB2 に複写定義変更プロセスを使用することはできません。サポートされているデータベース・バージョンについては、Replication Agent の『リリース・ノート』を参照してください。

複写定義の変更要求を発行すると、Replication Server は要求された変更の種類に基づいて、新しいバージョンの複写定義を作成する必要があるかどうかを決定します。Replication Server が新しいバージョンの複写定義を作成する場合、複写定義が変更される前に自動的に古いバージョンの複写定義を使ったプライマリの更新が行われ、複写定義の変更後には新しいバージョンの複写定義を使ったプライマリの更新が行われます。

注意： 複写定義の変更要求プロセスのこれらの手順は、**alter function replication definition** コマンドに該当しません。**alter function replication definition** はファンクション複写定義を直接変更するため、Replication Server 環境をクワイスする必要があります。

参照：
- プライマリ・データベースでの複写定義変更の直接実行 (352 ページ)
- DSI のサスペンド (353 ページ)
- 複写定義 RCL コマンドの確認 (354 ページ)
- 問題のあるコマンドの省略とエラー処理 (355 ページ)
- 複写定義の変更要求プロセスの使用 (356 ページ)

プライマリ・データベースでの複写定義変更の直接実行

プライマリ・データベースで複写定義の変更要求を直接実行するには、**rs_send_repserver_cmd** ストアド・プロシージャを使用します。

Replication Server では、次の複写定義 RCL コマンドの **rs_send_repserver_cmd** がサポートされています。

- **alter replication definition**
- **create replication definition**
- **drop replication definition**
- **alter applied function replication definition**
- **create applied function replication definition**
- **alter request function replication definition**
- **create request function replication definition**

構文は次のとおりです。

exec rs_send_repserver_cmd '*rs_api*'

rs_api には、変更する複写定義を含めます。

たとえば、プライマリ・データベースで `address`、`city`、`state`、`zip` カラムを **authors** 複写定義から削除するには、次のように入力します。

```
exec rs_send_repserver_cmd 'alter replication
definition authors drop address, city, state, zip'
```

プライマリ・データベースで **rs_send_repserver_cmd** を実行する場合、Replication Agent は *rs_api* に格納された RCL コマンドを Replication Server に送信し、RCL コマンドを実行します。これにより、Replication Server は適切な複写定義バージョンを使用してプライマリ・データを複写します。つまり、**rs_send_repserver_cmd** より前のプライマリ・データは古い複写定義バージョンで複写され、**rs_send_repserver_cmd** より後のプライマリ・データは新しい複写定義バージョンで複写されます。

必ずしも、プライマリ・データ・サーバから直接、複写定義の変更要求を発行する必要はありません。たとえば以下のような状況では、プライマリ Replication Server から直接、複写定義の変更要求を実行できます。

- 複写定義へのサブスクリプションがない
- 複写定義へのサブスクリプションはあるが、プライマリ・データベース・ログにテーブルまたはストアド・プロシージャのデータがない
- テーブル複写定義との間でサーチャブル・カラムを追加または削除する
- ファンクション複写定義との間でサーチャブル・パラメータを追加または削除する
- 動的 SQL をオンまたはオフにするために複写定義を変更する

警告！ Replication Server は、Replication Agent によって Replication Server に送信されるすべてのコマンドを受け入れるため、プライマリ・データベースで **rs_send_repserver_cmd** へのアクセスを制御する必要があります。

DSI のサスペンド

スタンバイ DSI (存在する場合) と、各サブスクリプション複写 DSI スレッドをサスペンドするには、**alter replication definition**、**alter applied function replication**

definition、**alter request function replication definition** を **with DSI_suspended** オプションとともに使用します。

Replication Server は、古いバージョンの複写定義のデータをすべてスタンバイ・データベースまたはレプリケート・データベースに適用した後に、スタンバイ・データベースまたはレプリケート・データベースの DSI スレッドをサスペンドします。

Replication Server が DSI スレッドをサスペンドした後、ターゲット・スキーマまたはターゲット・ストアド・プロシージャおよび任意のカスタム・ファンクション文字列を変更できます。

DSI スレッドを再開すると、Replication Server は変更された複写定義を使用してプライマリの更新を複写します。

次の例は、**alter replication definition** を実行する前に存在したプライマリ・データがターゲット・データベースに複写された後に、Replication Server がターゲット DSI をサスペンドするように指示する方法を示します。

```
alter replication definition pubs_rep
alter columns with pub_name as pub_name_set
with DSI_suspended
```

次の場合、**with DSI_suspended** を使用する必要はありません。

- 複写定義へのサブスクリプションがない。
- カスタム・ファンクション文字列を変更する必要がない。
- レプリケート・データベースまたはスタンバイ・データベースのスキーマを変更する必要がない。

注意： サイト・バージョンが 1550 より古いレプリケート Replication Server からのサブスクリプションがある場合、その Replication Server のレプリケート DSI スレッドはサスペンドされません。

複写定義 RCL コマンドの確認

複写定義 RCL コマンドを確認するには、**admin verify_repserver_cmd** を使用します。

Replication Agent が複写定義 RCL を実行するために Replication Server に送信したのに、複写定義 RCL を実行できないと、Replication Agent は停止します。この状況を回避するには、プライマリ・データベースから直接 RCL を実行する前に、プライマリ・データベースで **admin verify_repserver_cmd** を使用して、Replication Server が正常に複写定義要求を実行できることを確認するようにおすすめします。要求を正常に実行できない場合、Replication Server はエラーを返します。

Replication Server でサポートされる **admin verify_repserver_cmd** の複写定義コマンドは、**rs_send_repserver_cmd** と同じです。

- **alter replication definition**
- **create replication definition**
- **drop replication definition**
- **alter applied function replication definition**
- **create applied function replication definition**
- **alter request function replication definition**
- **create request function replication definition**

次の例は、**admin verify_repserver_cmd** がコマンド・ラインで “columns” キーワードを使用するなどの構文エラーを検出できることを示します。

```
admin verify_repserver_cmd, 'alter replication
definition authors drop columns address, city, state, zip
with DSI_suspended'
```

Replication Server では次のようなメッセージが返されます。

```
Line 1, character 71: Incorrect syntax with the keyword
'columns'.
```

問題のあるコマンドの省略とエラー処理

次回 Replication Agent が起動したときに、複写定義要求で失敗したものを省略するよう Replication Server に指示するには、**sysadmin skip_bad_repserver_cmd** を使用します。

警告！ sysadmin skip_bad_repserver_cmd は注意して使用してください。コマンドを実行した後に、プライマリ Replication Server で正しい複写定義コマンドを実行せずに Replication Agent を再起動すると、正しくない複写定義バージョンでプライマリ・データが複写される場合があります。

この例では、**sysadmin skip_bad_repserver_cmd** が、SYDNEY_DS データ・サーバの `pubs2` データベースで最後に失敗した複写定義コマンドを省略するよう、Replication Server と Replication Agent に指示します。

```
sysadmin skip_bad_repserver_cmd, SYDNEY_DS, pubs2
```

rs_send_repserver_cmd を使用してプライマリ・データベースで実行した複写定義要求がプライマリ Replication Server で失敗すると、Replication Agent が停止します。Replication Server がそのコマンドを省略しない限り、Replication Agent を再起動すると、失敗したコマンドが再び実行される。次の場合に複写定義要求が失敗することがあります。

- 要求の実行時にサブスクリプションが実行されている。
- Replication Server が複写定義またはカラムを検出できない。
- 構文エラーがある。

サブスクリプションの完了を待ったり、複写定義を修正したりして Replication Agent が停止する原因になった問題を解決できる場合は、Replication Agent を再起

動することができます。これにより Replication Agent から複写定義要求が再発行されます。

複写定義要求の失敗が、構文エラーなどの簡単に解決できないエラーにより引き起こされている場合は、次の手順に従います。

1. プライマリ Replication Server で直接正しい複写定義コマンドを実行します。
2. Replication Agent および Replication Server が Replication Agent から送信された最後の失敗要求を省略することを確認するために、プライマリ Replication Server で **sysadmin skip_bad_repserver_cmd** を実行します。
3. Replication Agent を再起動します。

複写定義の変更要求プロセスの使用

複写定義の変更要求プロセスの使用方法について、すべての手順を含む例を挙げて説明します。

前提条件

下のような状況では、プライマリ Replication Server から直接、複写定義の変更要求を実行できます。

- 複写定義へのサブスクリプションがない。
- 複写定義へのサブスクリプションはあるが、プライマリ・データベース・ログにプライマリ・テーブルまたはストアド・プロシージャのデータがない。
- サーチャブル・カラム・パラーメータを追加または削除する。
- 動的 SQL を有効または無効にする。

手順

1. プライマリ・スキーマとストアド・プロシージャへの変更をグループ化します。
2. プライマリ Replication Server にログインします。
3. 複写定義のすべてのサブスクリプションとアーティクルが有効になるまで、つまり実行中のマテリアライゼーションまたはマテリアライゼーション解除がなくなるまで待機します。
4. プライマリ・スキーマとストアド・プロシージャへの変更により影響を受ける各複写定義に対して、プライマリ Replication Server で **admin verify_repserver_cmd** を使用して複写定義要求をテストします。

```
admin verify_repserver_cmd, 'alter replication definition authors
drop address, city, state, zip
with DSI_suspended'
```

5. 複写システムに多くの複写定義が存在し、複写定義の変更要求も数多くある場合、パフォーマンスを向上するために、次のシステム・テーブルの **sts_full_cache** を無効にすることができます。

- rs_objects:

```
configure replication server
set sts_full_cache_rs_objects to 'off'
go
```

- rs_columns:

```
configure replication server
set sts_full_cache_rs_columns to 'off'
go
```

- rs_objfunctions:

```
configure replication server
set sts_full_cache_rs_objfunctions to 'off'
go
```

ヒント： RSSD の変更が多い場合は、定期的に RSSD テーブルで Adaptive Server の **update statistics** コマンドを実行します。複写定義の作成、変更、または削除などの複写定義の変更要求で影響を受けるテーブルは rs_objects、rs_columns、および rs_objfunctions です。ファンクション文字列の作成、変更、または削除などのファンクション文字列の変更要求で変更を受けるテーブルは rs_funcstrings および rs_systext です。

6. プライマリ・データベースで、次の手順に従います。
   a) 変更するプライマリ・テーブルおよびストアド・プロシージャへのデータ変更をすべてサスペンドします。
   b) テーブル・スキーマとストアド・プロシージャを変更します。
   c) 手順 4 で確認した複写定義要求に、**rs_send_repserver_cmd** ストアド・プロシージャを実行します。次に例を示します。

```
exec rs_send_repserver_cmd 'alter replication definition
authors
drop address, city, state, zip
with DSI_suspended'
```

   d) プライマリ・テーブルおよびストアド・プロシージャへのデータ変更を再開します。
7. **with DSI_suspended** を使用して複写定義要求を発行する場合、サブスクリプションを作成している各レプリケート・サイトで Replication Server により DSI がサスペンドされるまで待ってから、次の手順に従います。
   a) 必要に応じてレプリケート・テーブル・スキーマまたはストアド・プロシージャを変更します。
   b) 必要に応じてカスタム・ファンクション文字列を変更し、カスタム・ファンクション文字列がレプリケート・サイトに複写されるのを待ちます。
   c) レプリケート DSI を再開します。
8. 手順 5 でテーブルの **sts_full_cache** 設定を変更した場合は、ここで設定を元に戻すことができます。

手順 4 に従い、**admin verify_repserver_cmd** を使用して複写定義要求をテストすることを強くおすすめします。**admin verify_repserver_cmd** を使用すると、プライマ

リ Replication Server で **rs_send_repserver_cmd** を実行したときに、複写定義要求コマンドが失敗して Replication Agent が停止することを回避できる可能性が高くなります。

スキーマ変更により、レプリケート・サイトで複写スキーマやカスタム・ファンクション文字列の変更などのアクションの実施が必要になる場合は、複写定義に影響を与える手順 6c の一番最後の複写定義要求のみに **with DSI_suspended** を指定して発行する必要があります。場合によっては、レプリケート DSI を複数回再開することが必要になります。

複写スキーマ、ストアド・プロシージャ、またはカスタム・ファンクション文字列に変更が必要ない場合は、手順 7 を省略できます。

複写定義コマンドが失敗したために Replication Agent が停止した場合は、問題のあるコマンドを省略して Replication Agent をリカバリすることができます。

データベースが **rs_send_repserver_cmd** をサポートしていない場合は、変更中のスキーマのデータ・ローがプライマリ・データベース・ログに含まれなくなるまで待機してからプライマリ Replication Server で複写定義の変更要求を実行する必要があります。

参照：
- 問題のあるコマンドの省略とエラー処理 (355 ページ)

複写定義に可能な変更
複写定義で変更できるものについて説明します。

複写定義は、次のいずれかの方法で変更できます。

- 別の複写テーブル名の指定
- カラム・リストにカラムを追加する
- 別のレプリケート・カラム名を指定する
- `text`、`unitext`、`image`、または `rawobject` カラムの複写の指定内容を変更する
- プライマリ・キー・カラム・リストにカラムを追加する
- プライマリ・キー・カラム・リストからカラムを削除する
- サーチャブル・カラム・リストにカラムを追加する
- 複写定義からカラムを削除する
- サーチャブル・カラム・リストからカラムを削除する
- 宣言したデータ型またはパブリッシュ・データ型を変更する
- 最少カラムの複写指定を変更する
- スタンバイ・データベースに対する複写での複写定義の使用方法を変更する

注意： 複写定義スコープを持つファンクション文字列は、テーブルまたは複写定義にカラムを追加したときに自動的に変更されるわけではありません。場合に

よっては、ファンクション文字列を手動で変更する必要があります。
『Replication Server 管理ガイド 第 2 巻』の「データベース・オペレーションのカスタマイズ」の「ファンクション文字列の管理」を参照してください。

複写定義の変更要求を送信するには、複写定義を作成したプライマリ Replication Server で **alter replication definition** を使用するか、プライマリ・データベースで **rs_send_repserver_cmd** を使用します。複写定義で変更できる情報は、複写要求の作成時に指定する情報と同じです。

- プライマリ・テーブルまたはレプリケート・テーブルの名前を変更するには、複写定義を削除して再作成します。
- プライマリ・カラムを削除するか名前を変更したり、カラム・データ型を変更したりするには、プライマリ・カラムを持つすべての複写定義を削除してから再作成し、複写定義の変更要求プロセスに従います。

『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**alter replication definition**」を参照してください。

参照：
- 複写定義の作成 (307 ページ)
- 複写テーブルの名前変更 (366 ページ)
- 複写定義の変更要求プロセス (351 ページ)

別の複写テーブル名の指定
送信元テーブルから別の名前の送信先テーブルにデータを複写するには、複写定義を変更できます。

次に例を示します。

```
alter replication definition publishers
with replicate table named publishers2
```

テーブルの所有者の変更
Replication Server では、テーブルの所有者が異なる場合、これらのテーブルが別々のテーブルとして扱われます。**alter...modify owner** を使用して Adaptive Server でレプリケートされたテーブルの所有者を変更するには、該当するテーブル複写定義も変更する必要があります。

1. プライマリ Replication Server で、新しい所有者を持つ新規のテーブル複写定義を作成します。

   注意： 1 つのプライマリ・テーブルに対して複数の複写定義を作成できます。既存の複写定義を置換するには、新しい所有者を持つ新規の複写定義を作成する必要があります。

2. 現在の所有者の複写定義に対するサブスクリプションがある場合は、レプリケート Replication Server で新しい複写定義に対するサブスクリプションを作成します。
3. 現在の所有者の複写定義に対するカスタム・ファンクション文字列がある場合は、プライマリ Replication Server で新しい複写定義に対するカスタム・ファンクション文字列を作成します。
4. プライマリ Adaptive Server でテーブル所有者を変更します。
5. (オプション) 現在のサブスクリプションと複写定義を削除します。
   a) 現在の所有者を持つテーブルのデータがすべて処理されるまで待ちます。
   b) レプリケート Replication Server で現在のテーブル所有者に対して作成された複写定義のサブスクリプションを削除します。
   c) プライマリ Replication Server で現在の所有者の複写定義を削除します。

指定カラムの変更
カラムとプライマリ・キーの追加または削除、カラム・データ型の変更、プライマリ・カラム名とは異なるレプリケート・カラム名の指定など、複写定義のカラムを追加または変更する方法について説明します。

複写定義へのカラムの追加
複写定義にカラムを追加する方法について説明します。

`zip` (郵便番号情報) という `char` 型カラムを複写定義 **publishers** に追加するには、次の手順に従います。
複写定義を変更して、`char` 型カラムを追加します。

```
alter replication definition publishers_rep
add zip char(10)
```

`rep_zip_char` のように、送信先テーブルに追加したカラムの名前が送信元カラムと異なる場合は、次のコマンドを入力します。

```
alter replication definition publishers_rep
add zip as rep_zip char(10)
```

『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**alter replication definition**」を参照してください。

参照：
- 送信元テーブルおよび送信先テーブルのカラムの追加および削除 (366 ページ)

複写定義からのカラムの削除
複写定義からカラムを削除する方法について説明します。

この例では、**alter replication definition** によって、`address`、`city`、`state`、`zip` カラムを **authors** 複写定義から削除します。

```
alter replication definition authors
drop address, city, state, zip
```

- サイト・バージョンが 1550 より古いレプリケート Replication Server からのサブスクリプションがある場合、プライマリ Replication Server はカラムを削除するための複写定義の変更要求を拒否します。

  注意： 複写定義を変更してカラムを削除する場合、サイト・バージョンが 1550 より古いレプリケート Replication Server ではオートコレクションまたは動的 SQL 設定のリセットが必要になることがあります。
- プライマリ・テーブルに複数の複写定義がある場合、**alter replication definition** はコマンド・ラインの *repdef_name* で指定した複写定義のカラムのみを削除します。
- **drop** パラメータは、テーブル複写定義のカラムを削除します。カラムがプライマリ・キーまたはサーチャブル・カラムの一部である場合、**drop** はプライマリ・キー・リストまたはサーチャブル・カラム・リストのカラムを削除します。次のようなカラムの場合、Replication Server は、カラムを削除するための複写定義の変更要求を拒否します。
  - 唯一のカラム
  - 複写定義の唯一のプライマリ・キー・カラム
  - サブスクリプションまたはアーティクルの **where** 句内
  - アーティクルまたはサブスクリプションの **where** 句に指定されているサーチャブル・カラムの前

サーチャブル・カラムの削除
複写定義からサーチャブル・カラムを削除する方法について説明します。

前提条件
サーチャブル・カラムがサブスクリプションまたはアーティクルの **where** 句で使用されていない場合にのみ、複写定義からサーチャブル・カラムを削除できます。

手順

1. **drop subscription** を使用して、**where** 句から削除するサーチャブル・カラムを除外したいサブスクリプションをすべて削除します。
2. **alter replication definition** を使用して、サーチャブル・カラムを削除します。次に例を示します。

   ```
   alter replication definition publishers_rep
   drop searchable columns zip
   ```

   この例では、サーチャブル・カラム `zip` を複写定義 **publishers_rep** から削除しています。

『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**alter replication definition**」を参照してください。

3. **create subscription** を使用して、変更された複写定義にサブスクリプションを再作成します。

参照：

- drop subscription コマンド (437 ページ)
- create subscription コマンド (431 ページ)

プライマリ・キーの追加または削除

プライマリ・キーを追加または削除する方法について説明します。Replication Server は、レプリケート・テーブルまたはスタンバイ・テーブルでの正しいローの検出をプライマリ・キーに依存しています。

カラム `zip` をプライマリ・キーとして複写定義に追加するには、次のように入力します。

```
alter replication definition publishers_rep
add primary key zip
```

プライマリ・キーを削除するには、次のように入力します。

```
alter replication definition publishers_rep
drop primary key zip
```

すべてのプライマリ・キー・カラムを置き換えるには、まず、対応する複写定義を変更して新しいプライマリ・キーを追加してから、テーブル内の古いプライマリ・キー・カラムを削除します。

警告！ すべてのプライマリ・キー・カラムがプライマリ・テーブルから消失していると、DSI は停止します。

カラム・データ型の変更

カラム・データ型の変更には、いくつかの規則と制限があります。

- 宣言したカラム・データ型 (プライマリ・テーブル内のデータ型) は、サブスクリプションの **where** 句またはアーティクルの **where** 句で使用されている場合は変更できません。
- `rs_address` データ型は変更できません。
- カラム・データ型を `text`、`unitext`、`image`、`rawobject`、または `rawobject in row` に変更できるのは、カラムがプライマリ・キーまたはサーチャブル・カラムでない場合だけです。

- パブリッシュ (レプリケート) データ型を変更するには、カラムの宣言した (プライマリ) データ型 (変更の有無に関わらない) と **[map to]** 句を含める必要があります。
- あるカラムのデータ型と null 入力可能性を変更すると、テーブルのすべての複写定義の同じカラムが影響を受けます。ただし、`rawobject` または `rawobject in row` とその基本データ型の間での変更は、現在の複写定義だけに影響します。
- カラムの null 入力可能性の変更は、`text`、`unitext`、`image`、`rawobject` の各カラムに対してのみ行います。

参照：
- HDS を使用したデータ型の変換 (377 ページ)

別のレプリケート・カラム名の指定
プライマリ・カラム名と異なるレプリケート・カラム名を指定する方法について説明します。

送信元カラム `zip` のデータを `rep_zip2` という送信先カラムに複写するには、次のように入力します。

```
alter replication definition publishers_rep
alter columns with zip as rep_zip
```

このコマンドは次の場合に入力します。

- 既存の送信先テーブルを変更して、カラム `rep_zip` を追加する場合
- 送信先テーブルを削除して再作成し、元のカラム `zip` の代わりにカラム `rep_zip` を含める場合

text、unitext、image、rawobject の複写ステータスの変更
alter replication definition を使用して、複写定義の `text`、`unitext`、`image`、`rawobject` カラムの複写ステータスを変更します。

参照：
- text、unitext、image、rawobject カラムの複写ステータスの変更 (341 ページ)

サーチャブル・カラムの追加
サーチャブル・カラムを使用すると、カラムの値に基づいてサブスクリプションを作成できます。

サーチャブル・カラムを追加して利用するには、次の手順に従います。

1. **drop subscription** を使用して、**where** 句に追加されたサーチャブル・カラムを含めたいサブスクリプションすべてを削除します。
2. **alter replication definition** を使用して、サーチャブル・カラムを追加します。次に例を示します。

   ```
   alter replication definition publishers_rep
   add searchable columns zip
   ```

   (この例では、`zip` カラムをサーチャブルにしています。)

   『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**alter replication definition**」を参照してください。
3. **create subscription** を使用して、変更された複写定義にサブスクリプションを再作成します。

参照：

- drop subscription コマンド (437 ページ)
- create subscription コマンド (431 ページ)

最少カラム複写の変更

Replication Server が更新オペレーションおよび削除オペレーションをコピーするときに、各ロー内のすべてのカラムではなく、最少数のカラムを使用するように指定できます。

次のコマンドを入力します。

```
alter replication definition publishers_rep
replicate minimal columns
```

ウォーム・スタンバイ複写での複写定義の変更

ウォーム・スタンバイ・アプリケーションでスタンバイ・データベースにデータを複写するために使用する複写定義を指定するには、**alter replication definition** を使用します。

また、テーブル内のすべてのカラムを複写するか、または複写定義のカラムだけを複写するかも指定できます。

『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**alter replication definition**」を参照してください。

参照：

- ウォーム・スタンバイ・アプリケーションでの複写定義の使用 (319 ページ)

複写定義の削除

複写定義を削除する方法について説明します。

複写定義を削除するには、まず、その複写定義を参照するサブスクリプションとアーティクルをすべて削除する必要があります。

指定した複写定義に対する既存のサブスクリプションのリストを表示するには、**rs_helpsub** を使用します。『Replication Server リファレンス・マニュアル』の「RSSD ストアド・プロシージャ」の「**rs_helpsub**」を参照してください。

すべての複写定義に対する既存のサブスクリプションのリストを表示するには、**rs_helprep** を使用します。『Replication Server リファレンス・マニュアル』の「RSSD ストアド・プロシージャ」の「**rs_helprep**」を参照してください。

プライマリ Replication Server で **drop replication definition** を入力します。たとえば、**publishers_rep** という複写定義を削除するには、次のコマンドを入力します。

```
drop replication definition publishers_rep
```

『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**drop replication definition**」を参照してください。

参照：

- drop subscription コマンド (437 ページ)
- アーティクルの削除 (376 ページ)

複写データの修正

複写データを修正し、複写を管理するための関連オペレーションを実行する方法について説明します。

複写データを修正する前に、複写システムを計画したときの問題を慎重に再検討ください。複写データを修正する場合は、存在する可能性がある依存性と作業手順について確認してください。

参照：

- サブスクリプションの管理 (405 ページ)
- 複写システムのプラン作成 (300 ページ)
- 複写定義の変更要求プロセスの使用 (356 ページ)

新しいテーブルの追加

新しい送信元テーブルを追加したり、既存の送信元テーブルの新しい送信先のコピーを追加するには、複写手順の概要の手順に従ってください。

参照：

- テーブルの複写手順の概要 (303 ページ)

複写テーブルの名前変更

複写テーブルの名前を変更するには、複写定義名とテーブル名を指定し、複写定義の変更要求プロセスに従う必要があります。

参照：

- 複写定義名とテーブル名の指定 (311 ページ)
- 複写定義の変更要求プロセスの使用 (356 ページ)

送信元テーブルおよび送信先テーブルのカラムの追加および削除

送信元テーブルおよび送信先テーブルのカラムを追加または削除するには、複写定義の変更要求プロセスを使用します。

参照：

- 複写定義の変更要求プロセスの使用 (356 ページ)

サーチャブル・カラムの変更

サーチャブル・カラムは、複写定義に追加できます。

参照：

- サーチャブル・カラムの追加 (363 ページ)

サーチャブル・カラムの削除

サーチャブル・カラムがサブスクリプションまたはアーティクルの **where** 句で使用されていない場合にのみ、複写定義からサーチャブル・カラムを削除できます。

1. **drop subscription** を使用して、**where** 句から削除するサーチャブル・カラムを除外したいサブスクリプションをすべて削除します。
2. **alter replication definition** を使用して、サーチャブル・カラムを削除します。次に例を示します。

```
alter replication definition publishers_rep
drop searchable columns zip
```

この例では、サーチャブル・カラム zip を複写定義 **publishers_rep** から削除しています。

『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**alter replication definition**」を参照してください。

3. **create subscription** を使用して、変更された複写定義にサブスクリプションを再作成します。

参照：
- drop subscription コマンド (437 ページ)
- create subscription コマンド (431 ページ)

送信元テーブルまたは送信先テーブルのカラム・データ型の変更

ウォーム・スタンバイのみの設定で、プライマリ・テーブルとレプリケート・テーブルのカラム・データ型を変更するには、複写定義の変更要求プロセスを使用します。

参照：
- 複写定義の変更要求プロセスの使用 (356 ページ)

パブリケーションの使用

パブリケーションを使用すると、同じまたは関連するテーブルとストアド・プロシージャの複写定義をまとめて、1 つのグループとしてサブスクリプションを作成できます。

送信元 Replication Server で「パブリケーション」に複写定義をまとめ、送信先 Replication Server の「パブリケーション・サブスクリプション」を使い、それらにサブスクリプションを作成します。

パブリケーションを使用して、一連のテーブルとプロシージャの 1 つのパブリケーション・サブスクリプションのステータスをモニタします。

次の手順は、パブリケーションを使用してデータを複写するときの手順をまとめたものです。

1. 複写定義を作成、または選択してパブリケーションに含める。
2. パブリケーションを作成する。
3. 選択された複写定義を参照するアーティクルを作成する。
4. パブリケーションを確定化する。
5. パブリケーションのサブスクリプションを作成する。

注意： レプリケート・データベースは、同じプライマリ・テーブルの異なる複写定義に、直接、またはパブリケーションを介して、サブスクリプションを作成で

きます。ただし、それぞれの複写定義が、レプリケート・データベース内の異なるテーブルを参照する場合にかぎります。

パブリケーションを使用するには、プライマリ Replication Server のバージョンが 11.5 以降であることが必要です。パブリケーション・サブスクリプションを使用するには、レプリケート Replication Server と、プライマリ Replication Server とレプリケート Replication Server からのルートが、バージョン 11.5 以降であることが必要です。

パブリケーションを使用する場合は、次のオブジェクトを作成して管理します。

- アーティクル ― テーブルやファンクション複写定義をパブリケーションに置くことのできるストアド・プロシージャやテーブルのための複写定義を拡張したものです。アーティクルには、レプリケート・データベースが受信するローのサブセットを指定した **where** 句が含まれている場合もあれば、含まれていない場合もあります。
- パブリケーション ― 同じプライマリ・データベースからのアーティクルの集合です。
- パブリケーション・サブスクリプション ― パブリケーションに対するサブスクリプションです。パブリケーション・サブスクリプションを作成すると、Replication Server は、パブリケーションの各アーティクルのサブスクリプションを作成します。パブリケーション・サブスクリプションには、**where** 句がありません。

一般に、パブリケーションとパブリケーション・サブスクリプションは、複写定義やサブスクリプションの場合と同じように管理します。ただし、パブリケーションを作成する場合は、レプリケート・テーブルが受信するローのサブセットを指定できます。これには、サブスクリプション内ではなくアーティクル内に **where** 句を含めます。

コマンド・ラインを使用して、パブリケーションを作成し、管理できます。

参照：

- 複写ファンクションの管理 (385 ページ)
- サブスクリプションの管理 (405 ページ)

コマンド・ラインでのパブリケーションを使用したデータの複写

コマンド・ラインでデータを複写するためのパブリケーションを作成し、サブスクリプションのためにパブリケーションを準備する方法について説明します。また、パブリケーションおよびその関連アーティクルと複写定義を修正および削除する方法についても説明します。

パブリケーションを作成して管理するためのコマンド

パブリケーションを管理するには、いくつかのコマンドを使用できます。

パブリケーションを管理するためのすべてのコマンドは、**check publication** を除き、すべて送信元 Replication Server で実行されます。送信元 Replication Server では、**create object** パーミッションが必要です。**check publication** は、送信元 Replication Server ではすべてのユーザが実行でき、送信先 Replication Server では、両方のサーバで同じログイン名とパスワードを持つユーザであれば実行できます。

表 26 : パブリケーションを管理するためのコマンド

コマンド	作業
create publication	サブスクリプションを作成する 1 つ以上のデータベースに複写されるテーブルまたはストアド・プロシージャのグループのパブリケーションを作成する。
create article	パブリケーションのアーティクルを作成する。これにより、1 つ以上の **where** 句を追加して、送信先データベースに送信するローのサブセットを指定できる。 アーティクルを作成するには、アーティクルの基になるパブリケーションと複写定義が存在する必要がある。
validate publication	パブリケーションに少なくとも 1 つのアーティクルが含まれているかどうかをチェックし、パブリケーションに VALID およびサブスクリプションの準備ができていることを示すマークを付ける。
check publication	パブリケーションのステータスと、パブリケーションに含まれているアーティクルの数を表示する。
drop publication	*rs_publications* システム・テーブルからパブリケーションを削除する。 パブリケーションに関連する複写定義は、他のパブリケーションまたはサブスクリプションに含まれていなければ削除できる。
drop article	パブリケーションと *rs_articles* システム・テーブルからアーティクルを削除する。 アーティクルに関連する複写定義は、他のアーティクルまたはサブスクリプションに含まれていなければ削除できる。
rs_helppubs	パブリケーションとアーティクルの情報を表示する。

パブリケーション・サブスクリプションを処理するには、RCL コマンドも使用できます。

参照：

- パブリケーション・サブスクリプションを作成して管理するためのコマンド (453 ページ)

コマンド・ラインでのパブリケーションとアーティクルの作成

RCL コマンドを使用してサブスクリプションのパブリケーションを準備し、パブリケーションのサブスクリプションを作成する方法について説明します。

1. 送信元 Replication Server での手順
   a) データのコピー元にするテーブルまたはストアド・プロシージャの複写定義を作成または選択します。

      複写定義は、送信元および送信先のテーブルまたはストアド・プロシージャと、サブスクリプションを作成するデータベースに送信するカラムまたはパラメータを指定します。
   b) **create publication** を使用して、複写定義をグループ化するパブリケーションを作成します。

      パブリケーション情報は、送信元 Replication Server RSSD の `rs_publications` システム・テーブルに格納されます。この情報には、パブリケーション、データ・サーバ、およびデータベースの名前が含まれます。パブリケーション名は、送信元 Replication Server とデータベースでユニークでなければなりません。

      次の例では、**pubs2_pub** というパブリケーションを作成しています。プライマリ・データベースは、TOKYO_DS データ・サーバによって管理される `pubs2` です。

      ```
      create publication pubs2_pub
            with primary at TOKYO_DS.pubs2
      ```

      パブリケーション情報は、送信先 Replication Server でパブリケーションに対してサブスクリプションを作成するまで、送信先 Replication Server にはコピーされません。

      構文の詳細と使用方法のガイドラインについては、『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」を参照してください。
   c) **create article** を使用して、パブリケーションのアーティクルを作成します。

      各アーティクルは、それぞれが属するパブリケーションとそれが識別するテーブルまたはファンクションの複写定義を指定します。パブリケーションには、同じまたは異なる複写定義に基づいてアーティクルを含めることができます。複写定義とパブリケーションは、アーティクルを作成するときに存在していなければなりません。

アーティクルには、パブリケーション、複写定義、送信元データ・サーバ、送信元データベースの名前が含まれます。アーティクル情報は、rs_articles および rs_whereclauses システム・テーブルに保存されます。各アーティクル名は、パブリケーション内でユニークでなければなりません。

次の例では、パブリケーション **pubs2_pub** の複写定義 **titles_rep** に基づいて、**titles_art** というアーティクルを作成しています。

```
create article titles_art
    for pubs2_pub with primary at TOKYO_DS.pubs2
    with replication definition titles_rep
```

アーティクルには、サブスクリプションを作成するデータベースに送信するローまたはパラメータを指定した **where** 句を含めることができます。

アーティクルを作成すると、パブリケーションは不確定化されて、サブスクリプションに適さなくなります。アーティクルを作成したら、**validate publication** を使用してパブリケーションのステータスを VALID に変更してから、パブリケーションに対してサブスクリプションを作成します。

構文の詳細と使用方法のガイドラインについては、『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」を参照してください。

d) **validate publication** を使用して、パブリケーションのステータスを VALID に変更します。

パブリケーションを確定化すると、Replication Server は、パブリケーションに少なくとも 1 つのアーティクルが含まれているかどうかをチェックし、パブリケーションにサブスクリプションの準備ができていることを示すマークを付けます。

パブリケーションのアーティクルを追加または削除するたびに、Replication Server はパブリケーションを不確定化します。パブリケーションに VALID およびサブスクリプションの準備ができていることを示すマークを付けるには、**validate publication** を実行する必要があります。

パブリケーションを確定化したら、パブリケーションに対してパブリケーション・サブスクリプションを作成できます。

次の例では、**pubs2_pub** パブリケーションを確定化しています。送信元データベースは、データ・サーバ TOKYO_DS によって管理されている pubs2 です。

```
validate publication pubs2_pub
            with primary at TOKYO_DS.pubs2
```

構文の詳細と使用方法のガイドラインについては、『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」を参照してください。

2. 送信先 Replication Server での手順
   a) **create subscription** を使用して、パブリケーション・サブスクリプションを作成します。

      パブリケーション・サブスクリプションを作成すると、Replication Server は、パブリケーションの各アーティクルのサブスクリプションを作成します。

参照：
- 複写定義の作成 (307 ページ)
- create article コマンドでの where 句の指定 (372 ページ)
- パブリケーション・サブスクリプション (452 ページ)

create article コマンドでの where 句の指定

アーティクルには、1 つ以上の **where** 句を含めることができます。**where** 句は、複写するカラムまたはパラメータの値の基準を設定します。

where 句を省略すると、Replication Server は、テーブル複写定義で指定されたカラムのすべてのロー、またはファンクション複写定義で指定されたすべてのパラメータをコピーします。

アーティクルの **where** 句の構文は次のとおりです。

```
[where (column_name | @param_name)
      {< | > >= | <= | = | &} value
    [and {column_name | @param_name}
      {< | > >= | <= | = | &} value]...]
    [or where (column_name | @param_name)
      {< | > >= | <= | = | &} value
    [and {column_name | @param_name}
      {< | > >= | <= | = | &} value]...]
...
```

where 句の各カラム名は、テーブル複写定義のサーチャブル・カラム・リストにリストされている必要があります。各カラムの値は、そのカラムの比較対象のカラムと同じデータ型であることが必要です。

注意： アーティクル内の各 **where** 句は、**or** 演算子によってジョインされます。ただし、**!=**、**!<**、**!>**、**or** の各演算子は、**where** 句内では使用できません。**&** 演算子は、`rs_address` カラムに対してのみ使用できます。

次の例では、`type` カラムの値が 'popular_comp' であるローに複写を制限する **where** 句を使用して、**pubs2_pub** というパブリケーションに対して **titles_art** というアーティクルを作成しています。

```
create article titles_art
    for pubs2_pub with primary at TOKYO_DS.pubs2
    with replication definition titles_rep
    where type = 'popular_comp'
```

構文の詳細と使用方法のガイドラインについては、『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」を参照してください。

参照：
- ビットマップ・サブスクリプション (445 ページ)
- rs_address データ型の使用 (347 ページ)

パブリケーション情報の表示

パブリケーションとアーティクルに関する情報を表示するには、**check publication** コマンドと **rs_helppub** ストアド・プロシージャを使用します。

パブリケーション・ステータスとアーティクルの数の表示

パブリケーションに含まれるアーティクルの数と現在のステータスを表示するには、**check publication** を使用します。

すべてのユーザが、プライマリ Replication Server またはレプリケート Replication Server で **check publication** を実行できます。ただし、レプリケート Replication Server で **check publication** を実行する場合は、プライマリ・サーバとレプリケート・サーバで同じログインとパスワードが必要です。

次の例では、**pubs2_pub** パブリケーションのアーティクルのステータスと数を表示します。

```
check publication pubs2_pub
    with primary at TOKYO_DS.pubs2
```

『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**check publication**」を参照してください。

パブリケーションとアーティクルの情報の表示

パブリケーションとアーティクルの情報を表示するには、プライマリ Replication Server またはレプリケート Replication Server の RSSD で、**rs_helppub** ストアド・プロシージャを使用します。

注意： **rs_helppub** はプライマリ・サイトでもレプリケート・サイトでも実行できますが、**rs_helppub** が実行されたサイトで格納されているパブリケーション情報だけが表示されます。たとえば、**rs_helppub** をプライマリ・サイトで実行した場合、**rs_helppub** はプライマリ・サイトで作成されたすべてのパブリケーションについての情報を表示します。それに対して、**rs_helppub** をレプリケート・サイトで実行した場合、**rs_helppub** が表示するのは、レプリケート・サイトでサブスクリプションが作成されたパブリケーションの情報だけです。

次に **rs_helppub** の使用例をいくつか示します。

- サイトのすべてのパブリケーションをリストするには、次のように入力します。
  ```
  rs_helppub
  ```
  画面出力には、パブリケーション名、ステータス、プライマリ Replication Server とデータベースの名前、アーティクルの数、パブリケーションに対する最終変更日付が表示されます。
- 特定のパブリケーションについて詳細情報を表示するには、次のように入力します。
  ```
  rs_helppub publication_name, primary_dataserver,
      primary_db
  ```
  画面出力には、上記の情報に加え、関連するアーティクルの名前、複写定義、プライマリ・テーブルとレプリケート・テーブルが表示されます。パブリケーションに対してサブスクリプションが作成されている場合は、サブスクリプションの名前、レプリケート・データベース、所有者、サブスクリプションに対する最終変更日付が表示されます。
- 特定のアーティクルについての情報を表示するには、次のように入力します。
  ```
  rs_helppub publication_name, primary_dataserver,
      primary_db, article_name
  ```
  画面出力には、アーティクルが属しているパブリケーションの名前、関連する複写定義、ステータス情報、**where** 句とサブスクリプション (存在する場合) が表示されます。

『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**rs_helppub**」を参照してください。

パブリケーション情報の変更

アーティクルまたはパブリケーションを変更する場合は、アーティクルまたはパブリケーションを削除して再作成します。アーティクルの **where** 句をより選択的にすることもできます。

アーティクルの **where** 句をより選択的にするには、次のいずれかを行います。

- アーティクルを削除し、**where** 句を変更したアーティクルを再作成する。
- (同じ複写定義に対して) 別のアーティクルを作成し、新しいローまたはパラメータに合わせて **where** 句を調整する。

参照：

- パブリケーションの削除 (375 ページ)
- アーティクルの削除 (376 ページ)

パブリケーションへのアーティクルの追加

アーティクルを既存のパブリケーションに追加する方法について説明します。

1. 送信元 Replication Server での手順
   a) アーティクルの基本となる複写定義を作成するか、または選択します。
   b) **create article** を使用して、新しいアーティクルを作成します。
   c) **validate publication** を使用してパブリケーションを確定化し、新しいアーティクルにサブスクリプションを作成できるようにします。
2. 送信先 Replication Server での手順
   a) 新しいアーティクルのサブスクリプションを作成するには、**create subscription** または **define subscription** を入力し、**for new articles** 句を含めます。

参照：

- パブリケーション・サブスクリプション (452 ページ)

パブリケーションの削除

パブリケーションとそのすべてのアーティクルをシステム・テーブルから削除するには、**drop publication** を使用します。

レプリケート Replication Server で、パブリケーションに対して作成されたすべてのサブスクリプションを削除してから、そのパブリケーションを削除します。

送信元データベースを管理する Replication Server で **drop publication** を実行します。**create object** パーミッションが必要です。

次の例では、**pubs2_pub** パブリケーションと、このパブリケーションに含まれるアーティクルを削除しています。

```
drop publication pubs2_pub
        with primary at TOKYO_DS.pubs2
```

パブリケーション情報は、プライマリ Replication Server からすぐに削除されますが、次の作業を行うまでレプリケート Replication Server からは削除されません。

- 削除されたパブリケーションに対してサブスクリプションを作成する。
- レプリケート Replication Server で **check publication** を入力する。

『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**drop publication**」を参照してください。

参照：

- パブリケーションとアーティクルのサブスクリプションの削除 (459 ページ)

関連する複写定義の削除
パブリケーションに関連する複写定義を削除するには、**drop_repdef** 句を指定して **drop publication** を実行します。

Replication Server は、他のパブリケーションまたはサブスクリプションから参照されていないパブリケーションに関連するすべての複写定義を削除します。

たとえば、**pubs2_pub** に関連するすべての複写定義を削除するには、次のように入力します。

```
drop publication pubs2_pub
    with primary at TOKYO_DS.pubs2
    drop_repdef
```

アーティクルの削除
パブリケーションからアーティクルを削除するには、**drop article** を使用します。

レプリケート Replication Server でアーティクルに対して作成されたサブスクリプションを削除してから、そのアーティクルを削除します。

送信元データベースを管理する Replication Server で **drop article** を実行します。**create object** パーミッションが必要です。

次の例では、**pubs2_pub** パブリケーションの **titles_art** アーティクルを削除しています。

```
drop article titles_art
    for pubs2_pub with primary at TOKYO_DS.pubs2
```

『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**drop article**」を参照してください。

参照：
- パブリケーションとアーティクルのサブスクリプションの削除 (459 ページ)

関連する複写定義の削除
アーティクルに関連する複写定義を削除するには、**drop_repdef** 句を指定して **drop article** を実行します。

Replication Server は、複写定義が他のパブリケーションまたはサブスクリプションで参照されていない場合に複写定義を削除します。

たとえば、**pubs2_pub** アーティクルと、このアーティクルが参照する複写定義を削除するには、次のように入力します。

```
drop article titles_art
    for pubs2_pub with primary at TOKYO_DS.pubs2
    drop_repdef
```

HDS を使用したデータ型の変換

異機種間複写システムでは、データ・サーバから別のデータ・サーバに情報を複写するときに、多くの場合、プライマリ・データ・サーバに格納されている値を変更して、レプリケート・データ・サーバの別のデータ型に正常にコピーできるようにする必要があります。

ユーザが作成したファンクション文字列でもこのようなデータ型変換を行うことができますが、ユーザ入力がかなり必要になるだけでなく、レプリケート・データ・サーバの機能による制限も受けます。

異なるデータ・サーバ間でのデータ型変換をより簡単に行えるように、Replication Server には異機種データ型サポート (HDS) が用意されています。これは、Replication Server でデータ型変換を簡単に行うためのしくみです。HDS は、次のデータ・サーバ間での特定のデータ型の変換をサポートしています。

- Adaptive Server Enterprise
- DB2
- Oracle
- Microsoft SQL Server
- UDB

HDS を使用するときに、プライマリ・データベースの変換対象のカラムとデータ型、および変換したデータを受信するレプリケート・データ・サーバを選択できます。

情報ソースは次のとおりです。

- 『Replication Server 異機種間複写ガイド』を参照してください。
- HDS を有効にするオブジェクトのインストールと設定については、使用しているプラットフォームの『Replication Server 設定ガイド』を参照してください。
- ファンクション文字列の詳細については、『Replication Server リファレンス・マニュアル』を参照してください。

異機種データ型サポートの概要

Adaptive Server 間で情報を複写する場合、データ型変換は通常不要です。ただし、プライマリ・データベースとレプリケート・データベースでデータ型が異なる場合には、HDS を使用してデータ型変換を実行できます。

次のデータベース間で複写を実行するときに、HDS 機能を使用できます。

- Adaptive Server データベース間 — 異なる Adaptive Server データ型間
- Sybase 以外の同種のデータベース間 - DB2 `TIMESTAMP` から DB2 `DATE` など

- Sybase 以外の異機種のデータベース間 ― Oracle から DB2 など
- Adaptive Server と Sybase 以外のデータベース間 ― Adaptive Server から Oracle など

HDS は、プライマリ・データ・サーバとレプリケート・データ・サーバのデータ型に互換性がない場合にも対処できます。一般に、このような非互換性には、次の 4 タイプがあります。

- 範囲の不一致 ― 例えば、`datetime` の日付の許容範囲は、Sybase では 1753 年 1 月 1 日～ 9999 年 12 月 31 日だが、別のデータ・サーバでは 0001 年 1 月 1 日～ 9999 年 12 月 31 日のみが許容される場合がある。
- フォーマットの不一致 ― プライマリ・データ・サーバの日付フォーマットが “CCYY-MM-DD” で、レプリケート・データ・サーバの日付フォーマットが “MM/DD/CCYY” の場合など。
- 長さの不一致 ― 例えば、プライマリ・テーブルのカラムの文字長は 10 であるが、レプリケート・テーブルのカラムの文字長が 15 である場合がある。
- デリミタの不一致 ― 例えば、Sybase ではバイナリ・データをプレフィクス “0x” で区切るが、別のデータ・サーバではバイナリ・データを一重引用符で囲む場合がある。

各データ・サーバの Replication Agent は、レプリケート値を Replication Server が理解できるデータ型フォーマットで Replication Server に配布します。これには、リテラル値、デリミタ情報、その他のデータ型属性が含まれます。Replication Server は、値をその基本データ型として処理します。基本データ型は、『Replication Server リファレンス・マニュアル』に記載されているネイティブ Replication Server データ型のいずれかです。

データ型変換は、次の 2 とおりの方法で実装できます。

- クラス・レベル変換 ― 特定のコネクションに対して、データ型のインスタンスをすべて変換する。
  ASE 以外のデータベースの接続プロファイルを指定するコネクションを作成します。接続プロファイルには、クラス・レベル変換が含まれます。
  『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**create connection using profile**」を参照してください。
- カラム・レベル変換 ― テーブル複写定義に記述されているカラムのインスタンスをすべて変換する。
  カラム・レベル変換用の **map to** 句を使用します。

データ型変換の使用開始にあたって

異機種間複写システムでデータ型変換を設定する方法について説明します。

1. 使用しているプライマリおよびレプリケートのデータ・サーバで実行可能なデータ型変換を確認します。必要な変換とその配布方法を決定します。
   - クラス・レベル変換
     サポートされているクラス・レベル・データ型変換のリストについては、『Replication Server 異機種間複写ガイド』を参照してください。
   - カラム・レベル変換
2. 複写定義を作成または変更して、カラム・レベル変換を設定します。

参照：

- サブスクリプションの管理 (405 ページ)
- 複写ファンクションの管理 (385 ページ)

クラス・レベル変換の作成

ASE 以外のデータ・サーバのクラス・レベル変換は、接続プロファイルで定義され、接続プロファイルによってインストールされます。

詳細については、『Replication Server 異機種間複写ガイド』を参照してください。

システム定義変数

クラス・レベル変換では、カラム値だけでなく、システム定義変数のデータ型も変更します。

たとえば、クラス・レベル変換で `datetime` を `rs_db2_timestamp` に変換する場合、システム定義変数 *rs_origin_begin_time* (`datetime` データ型) は、そのコネクションで *rs_db2_timestamp* に変換されます。

参照：

- 接続プロファイル (191 ページ)

カラム・レベル変換の作成

カラム・レベル変換は、特定のカラム (データ型) とテーブルの複写された各インスタンスに影響します。カラム・レベル変換は、**create replication definition** コマンドまたは **alter replication definition** コマンドを使用して定義します。

カラム・レベル変換を設定するには、複写定義を作成または変更します。このとき、**map to** オプションを使用して、変換対象のカラムおよび変換前と変換後のデータ型を指定します。

- 新しい複写定義を作成する場合は、**create replication definition** を使用します。サポートされているデータ型変換のリストについては、『Replication Server 異機種間複写ガイド』を参照してください。
- 既存テーブルのカラムを追加または変更する場合は、**alter replication definition** を使用します。

Sybase では、複写カラムのデータ型の変更に使用できる、データ型定義のセットとデータ型クラスを用意しています。各データ型クラスには、次のように特定のデータ・サーバのデータ型定義が含まれています。

- Adaptive Server — **rs_sqlserver_dt_class**
- DB2 — **rs_db2_dt_class**
- Microsoft SQL Server — **rs_msss_dt_class**
- Oracle — **rs_oracle_dt_class**
- Sybase IQ – **rs_iq_dt_class**
- UDB – **rs_udb_dt_class**

データ型クラスは複写されず、変更することもできません。カラム・レベル変換は、サブスクリプション解析後、クラス・レベル変換の前に実行されます。

テーブル複写定義を作成または変更するときに、特定のカラムに対してカラム・レベル変換をアクティブにすることができます。HDS 用にカラム変数とデータ型変数を指定した **create replication definition** の構文は次のとおりです。

```
create replication definition replication_definition
with primary at data_server.database
...
(column_name [as replicate_column_name] declared_datatype [null |
not null]
[map to published_datatype])
...
```

構文の説明は次のとおりです。

- 宣言したデータ型は、Replication Agent から Replication Server に配布される値のデータ型によって次のように異なります。
  - Replication Agent が `datetime` などのネイティブ Replication Server データ型を Replication Server に配布する場合、宣言したデータ型はネイティブ・データ型です。
  - それ以外の場合は、宣言したデータ型はプライマリ・データベースの元のデータ型のデータ型定義である必要があります。
    たとえば、Replication Agent は DB2 の `TIMESTAMP` データ型の値を、デリミタを含む文字列として Replication Server に配信します。この場合、宣言したデータ型はデータ型定義 `rs_db2_timestamp` になります。
- パブリッシュ・データ型は、カラム・レベル変換後 (クラス・レベル変換を実行する場合は、その前) のカラムのデータ型です。パブリッシュ・データ型は、

通常、Replication Server のネイティブ・データ型またはレプリケート・データベースのデータ型定義です。パブリッシュ・データ型が複写定義から省略された場合、デフォルトで宣言したデータ型になります。

宣言したデータ型とパブリッシュ・データ型のどちらにも基本データ型があります。たとえば、データ型 rs_db2_timestamp の基本データ型は char(26) です。ネイティブ・データ型 char(26) の基本データ型も char(26) です。データ型定義は、Sybase 以外のデータ型を、Replication Server のネイティブ・データ型に置き換えて記述します。基本データ型は、データ型定義に対応するように最大長と最小長を固定し、他のデータ型属性に対するデフォルト値を提供します。基本データ型は、ログ転送言語 (LTL) または Replication Server 管理者によって実行されるコマンド (**create subscription** など) でデータ型定義のデータ型の値が Replication Server に配信されるときの、値の区切りを定義します。

注意： ネイティブ・データ型には、Replication Server がサポートするすべてのデータ型が含まれます。ただし、text、unitext、image、rawobject、rawobject in row の各データ型をデータ型定義に使用することはできません。また、これらのデータ型を変換元または変換先として使用することもできません。

たとえば、birthdate カラムの値を datetime (birthdate のプライマリ・テーブル・データ型) からレプリケート・データベースの DB2 の DATE データ型に変換するテーブル複写定義 **ase_employee_repdef_for_db2** を作成するには、プライマリ Replication Server にログインし、次のように入力します。

```
create replication definition
  ase_employee_repdef_for_db2
    with primary at ase_server.ase_database
    with all tables named 'employee'
      (empid int,
      first_name char(20),
      last_name char(20),
      ...
      birthdate datetime map to rs_db2_date,
      salary money,
      ...
```

この例では、birthdate はカラム名であり、datetime は宣言したデータ型、rs_db2_date はパブリッシュ・データ型です。宣言したデータ型はネイティブ・データ型であるため、ネイティブ・データ型と基本データ型は同じです。つまり、datetime の基本データ型は datetime です。パブリッシュ・データ型 rs_db2_date は DB2 のデータ型定義であり、基本データ型は char(10) です。

参照：

- クラス・レベル変換とカラム・レベル変換を同時に使用する (382 ページ)

データ型定義の働き

データ型定義を使用すると、重要な情報を失うことなくデータ型を別のデータ型に変換できます。

宣言したデータ型として使用した場合、データ型定義には、リテラル値とそのデータ型属性 (デリミタ、範囲情報、精度、位取り、長さ、最小値、最大値など) を取得して、それらを Replication Server が処理できるネイティブ・データ型のフォーマットに変換するメカニズムが備わっています。

パブリッシュ・データ型として使用した場合、データ型定義は、Replication Server のネイティブ・データ型フォーマットの値 (属性情報を含む) を取り出して、できるだけ多くの情報をパブリッシュ・データ型で保持しながら、他のデータベースで使用できるデータ型フォーマットに変換します。

宣言したデータ型とパブリッシュ・データ型の両方にデータ型定義が使用された場合、両方の変換が実行されます。

注意： Microsoft SQL Server は、15.0 の新しい符号なし integer 型を直接サポートしていないため、複写定義で **map to** 句を使用する必要があります。

カラム・レベル変換と複数の複写定義

通常、複数の複写定義で宣言されたカラムは、各複写定義で同じ「宣言したデータ型」を使用する必要があります。「パブリッシュ・データ型」は異なってもかまいません。

ただし、複数の複写定義で宣言された `rawobject` カラムと `rawobject in row` (Java) カラムでは、`rawobject` (または `rawobject in row`) データ型を使用することも、宣言したデータ型の基本データ型を使用することもできます。たとえば、`rawobject` と `image`、または `rawobject in row` と `varbinary` を、同じ Java カラムの複数の複写定義で使用できます。Adaptive Server での Java カラムについては、Adaptive Server Enterprise のマニュアル『Adaptive Server Enterprise における Java』を参照してください。

クラス・レベル変換とカラム・レベル変換を同時に使用する

クラス・レベルとカラム・レベルのデータ型変換を同じカラムに対してアクティブにすると、両方の変換が適用されます。

カラム・レベル変換は、サブスクリプション解析のあとに実行され、カラム・レベル変換のあと、クラス・レベル変換が実行されます。そして、その直後にレプリケート・データベースに配信されます。

この実行順序によって、必ず、カラム・レベル変換の方がクラス・レベル変換よりも優先されます。つまり、特定のコネクションの変換 (クラス・レベル変換) は、

特定のテーブルやカラムに対して定義された変換 (カラム・レベル変換) には影響しません。

変換の確認

カラム・レベル変換またはクラス・レベル変換を設定する前に、これらの変換によって値がどのように変更されるかを確認できます。

特定の変換の結果を参照するには、**admin translate** コマンドを使用します。**admin translate** は、値と、変換前および変換後のデータ型を受け取り、変換後の値を返します。このコマンドは、Replication Server の診断バージョンでの使用が最も効果的です。変換が失敗した場合、失敗の原因を追跡できるためです。

構文を以下に示します。

```
admin translate, value, source_datatype, target_datatype
```

構文の説明は次のとおりです。

- *value* — 変換する値のリテラル表現であり、変換元データ型の基本データ型が必要とするデリミタを含みます。
- *source_datatype* — 変換前の値のデータ型定義またはデータ型です。
- *target_datatype* — 変換後の値のデータ型定義またはデータ型です。

`char(26)` などのように、変換前または変換後のデータ型の基本データ型が長さの指定を必要とする場合は、データ型名を引用符で囲みます。

たとえば、`db2_date` から `datetime` への日付の変換を確認するには、Replication Server にログインし、次のように入力します。

```
admin translate, '04/29/1989', db2_date, datetime
```

この例では、*value* は文字列 “04/29/1989” で、これを一重引用符で囲む必要があります。詳細と使用例については、『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**admin translate**」を参照してください。

複写ファンクションの管理

複写ファンクションを使用して、送信元データベースから送信先データベースにストアド・プロシージャの実行を複写します。

ファンクション複写を使用すると、Replication Server はストアド・プロシージャの実行を送信元データベースから送信先データベースに複写します。つまり、送信元データベースでストアド・プロシージャを実行すると、Replication Server は送信先データベースで別のストアド・プロシージャの実行を開始します。これらの 2 つのストアド・プロシージャを同じ名前にする必要はなく、同じタスクを実行させる必要もありません。

複写ストアド・プロシージャに関する複写システム設計の問題の詳細については、『Replication Server デザイン・ガイド』を参照してください。

テーブル複写定義に関連するストアド・プロシージャの分配については、『Replication Server 管理ガイド 第 2 巻』の「非同期プロシージャ」を参照してください。サブスクリプションが存在しない 15.1 より前のバージョンの要求ファンクションの分配については、『Replication Server 15.6 管理ガイド 第 2 巻』の「バージョン 15.1 より前の要求ファンクションの複写」を参照してください。

送信元のストアド・プロシージャと送信先に渡される情報は、次の項目を指定するファンクション複写定義を作成することによって特定します。

- 送信元データベースと送信先データベースでのストアド・プロシージャの名前 (名前が異なる場合)
- 送信先のストアド・プロシージャに渡されるデータ型とパラメータ

分散アプリケーションの要件を満たすために、Replication Server では、複写ファンクションを実装する 2 つの方法が用意されています。次のいずれかの方法を使用してください。

- メンテナンス・ユーザがレプリケート・データベースにトランザクションを配信する「適用ファンクション」。
- プライマリ・データベースでストアド・プロシージャを呼び出したユーザがレプリケート・データベースにトランザクションを配信する「要求ファンクション」。

ファンクションが**適用ファンクション複写定義**によって複写されている場合は、**maint_user** がレプリケート・データベースでトランザクションを実行します。ファンクションが**要求ファンクション複写定義**によってレプリケート・データベースで複写されている場合は、**origin_user** がトランザクションを実行します。

参照：
- 適用ファンクション (391 ページ)
- 要求ファンクション (395 ページ)

複写ファンクションの前提条件および制限

適用ファンクションまたは要求ファンクションを複写システムに実装するには、前提条件を満たしていることと、複写ストアド・プロシージャの使用に関する制限について理解していることを確認してください。

複写ファンクションの前提条件

適用ファンクションまたは要求ファンクションを実装するため、準備することがいくつかあります。

- 使用しているアプリケーションの必要に応じて、適用ファンクションまたは要求ファンクションをどのように使用するのかを理解します。詳細については、『Replication Server デザイン・ガイド』を参照してください。
- プライマリ Replication Server で RepAgent を設定するには、レプリケーション・システムの管理に関係する手順および『Replication Server 設定ガイド』の設定手順に従います。
- プライマリ Replication Server からレプリケート Replication Server までのルートを設定します。
- 複写ファンクションは、フラグメント化されたプライマリ・データに関与するアプリケーションで使用できます。これを使用するには、それぞれのプライマリ・フラグメントについて、ファンクション複写定義とストアド・プロシージャを作成します。フラグメント化されたプライマリ・データの操作方法の詳細については、『Replication Server デザイン・ガイド』を参照してください。

  ここで説明した複写ファンクションの内容は、そのほとんどが、Replication Server の基本的なプライマリ・コピー・モデルを使用して、単一の送信元データベースが 1 つ以上の送信先データベースにデータを分配することを前提としています。

参照：
- 複写システムの管理 (83 ページ)
- ルートの管理 (161 ページ)
- Replication Server の基本プライマリ・コピー・モデル (9 ページ)

複写ファンクションの制限

複写ストアド・プロシージャを使用するとき、いくつかの制限があります。

- ファンクション複写定義を含むすべての複写定義の名前は、複写システム内でユニークにする必要があります。
- 複写システムのプライマリ・ファンクションに対して適用ファンクション複写定義を作成する場合は、そのファンクションに、次の条件をどちらも満たす既存のファンクション複写定義がないことを確認します。
  - ファンクション複写定義が、**create function replication definition** コマンドを使用して作成されている。
  - ファンクション複写定義が、Replication Server 15.0.1 以前のバージョンでサブスクリプションのない要求ファンクション複写に使用されている。

  それ以外の場合、既存の要求ファンクション複写は無効になります。詳細については、『Replication Server 15.6 管理ガイド 第 2 巻』の「バージョン 15.1 より前の要求ファンクションの複写」を参照してください。
- Replication Server は、複写ストアド・プロシージャ内の、**begin** 文または **commit** 文を含むネストされたトランザクションをサポートしません。

  ネストされたストアド・プロシージャを持つストアド・プロシージャが複写するようにマーク付けされた場合、次の現象が発生します。
  - RepAgent が外部ストアド・プロシージャ呼び出しだけを Replication Server に転送する。
  - RepAgent が停止する。
  - Adaptive Server のエラー・ログにエラー・メッセージが表示される。

  maint_user またはレプリケート・データベースが、**sp_setrepproc** または **sp_setreplicate** を使用してストアド・プロシージャを複写する場合、Adaptive Server は常にトランザクション内でストアド・プロシージャを実行します。プライマリ・データベースのトランザクション内で複写ストアド・プロシージャを明示的に実行しなかった場合でも、レプリケート・データベースの **maint_user** によって適用されるときは、Adaptive Server は、プロシージャの開始時に暗黙の **begin transaction** を設定します。

  詳細については、『Replication Server 管理ガイド第 2 巻』の「パフォーマンス・チューニング」の「パフォーマンスに影響する設定パラメータ」の「パフォーマンスに影響するコネクション・パラメータ」にある「**dsi_max_xacts_in_group**」を参照してください。複写ストアド・プロシージャに、**begin transaction**、**commit transaction**、**rollback transaction** などのコマンドがあると、そのストアド・プロシージャの実行時にエラーが発生します。たとえば、**rollback transaction** コマンドは、意図したロールバック・ポイントであるネストされた **begin transaction** ではなく、トランザクション・グループの最初までロールバックする可能性があります。

- Adaptive Server ストアド・プロシージャのような複写ファンクションには、`text` データ型や `image` データ型のパラメータを含めることはできません。詳細については、『Adaptive Server Enterprise リファレンス・マニュアル』を参照してください。
- Adaptive Server は、複写ストアド・プロシージャを囲むトランザクションが開始されたデータベースに複写ストアド・プロシージャの呼び出しのログを記録します。
  - ユーザが明示的にトランザクションを開始しない場合、Adaptive Server は、ストアド・プロシージャが実行される前にユーザの現在のデータベース内でトランザクションを開始します。
  - ユーザがあるデータベース内でトランザクションを開始し、複写ストアド・プロシージャの実行は別のデータベース内で行った場合でも、その実行ログはトランザクションが開始されたデータベースに記録されます。
- 単一のトランザクションが 1 つ以上の要求ファンクションを呼び出して適用ファンクションを実行する場合、またはデータ更新言語 (「混合モード」トランザクション) を含む場合、Replication Server は他のすべてのオペレーションが終了したあとに、別のトランザクションでまとめて要求ファンクションを処理します。
- 複写ファンクションと異機種データ型変換を使用する場合は、以下のことに注意してください。
  - **create applied/request function replication definition** または **alter applied/request function replication definition** によるパラメータ値のデータ型の変更はできません。ただし、データ型定義を使用して適用ファンクション複写定義のパラメータを宣言することはできます。この定義は、そのあとクラス・レベル変換を行うものです。
  - Replication Server は、パラメータ値を要求ファンクション用に変換しません。ただし、ファンクション文字列のマップ時に、宣言したデータ型のパラメータ値用に定義されたデリミタを使用して SQL が生成されます。

警告！ 複写ファンクション内にコミット文を入れないでください。コミット文を入れると、重複キーの原因となり、Replication Server のリカバリが失敗します。

ファンクション複写定義を管理するコマンド

ファンクション複写定義に関連するコマンドについて説明します。

表 27 : ファンクション複写定義を管理するコマンド

コマンド	作業
drop function replication definition	複写システムからファンクション複写定義を削除する。ファンクション複写定義は、その複写定義用のすべてのサブスクリプションを削除すると、削除できるようになる。
create applied replication definition	プライマリ・データベースとレプリケート・データベース両方について、ストアド・プロシージャとそのパラメータを記述する適用ファンクション複写定義を作成する。プライマリ・データのロケーションも記述する。**maint_user** が、レプリケート・サイトで適用ファンクションを適用する。
create request replication definition	プライマリ・データベースとレプリケート・データベース両方について、ストアド・プロシージャとそのパラメータを記述する要求ファンクション複写定義を作成する。プライマリ・データのロケーションも記述する。プライマリ・サイトでストアド・プロシージャを実行しているユーザが、レプリケート・サイトで要求ファンクションを適用する。
alter applied replication definition	**create applied function replication definition** コマンドで作成された適用ファンクション複写定義を修正する。たとえば以下を実行する。 • 送信元データベースで呼び出されたプライマリ・ストアド・プロシージャに別の名前を指定する。 • 送信先データベースで呼び出されたストアド・プロシージャに別の名前を指定する。 • パラメータまたはサーチャブル・パラメータを追加する。 • スタンバイ・データベースに対する複写での複写定義の使用方法を変更する。 パラメータが追加されると、このプライマリ・ファンクションに対して作成されたすべての適用ファンクション複製定義に変更が適用される。

コマンド	作業
alter request replication definition	**create request function replication definition** コマンドで作成された要求ファンクション複写定義を修正する。たとえば以下を実行する。 • 送信元データベースで呼び出されたプライマリ・ストアド・プロシージャに別の名前を指定する。 • 送信先データベースで呼び出されたストアド・プロシージャに別の名前を指定する。 • パラメータまたはサーチャブル・パラメータを追加する。 • スタンバイ・データベースに対する複写での複写定義の使用方法を変更する。 パラメータが追加されると、このプライマリ・ファンクションに対して作成されたすべての要求ファンクション複製定義に変更が適用される。
create function replication definition	複写するストアド・プロシージャとそのパラメータを記述するファンクション複写定義を作成する。プライマリ・データのロケーションも記述する。このコマンドは今後廃止され、**create applied function replication definition** コマンドと **create request function replication definition** コマンドで置き換えられる。
alter function replication definition	ファンクション複写定義を修正する。たとえば以下を実行する。 • 送信先データベースで呼び出されたストアド・プロシージャに別の名前を指定する • パラメータまたはサーチャブル・パラメータを追加する • スタンバイ・データベースに対する複写での複写定義の使用方法を変更する このコマンドは、**create function replication definition** コマンドで作成されたファンクション複写定義を修正する場合のみ使用できる。
rs_send_repserver_cmd	プライマリ・データベースで複写定義の変更要求を直接実行する。
admin verify_repserver_cmd	Replication Server が複写定義の変更要求を実行できることを確認する。

参照：

- 複写ファンクションの修正または削除 (401 ページ)
- プライマリ・データベースでの複写定義変更の直接実行 (352 ページ)
- 複写定義 RCL コマンドの確認 (354 ページ)
- テーブル複写定義を管理するためのコマンド (306 ページ)
- サブスクリプション・コマンド (426 ページ)
- 適用ファンクションの実装 (392 ページ)
- 要求ファンクションの実装 (396 ページ)

複写ファンクションの使用

複写ストアド・プロシージャとは、**sp_setrepproc** または **sp_setreplicate** のいずれかを使用して複写するようマーク付けされた Adaptive Server ストアド・プロシージャのことです。「ファンクション複写定義」は、プライマリ・ストアド・プロシージャと複写ストアド・プロシージャ、およびそのパラメータとロケーションを記述します。

次の 3 つのコマンドを使用すると、ファンクション複写定義を作成できます。

- **create applied function replication definition**
- **create request function replication definition**
- **create function replication definition** (今後廃止)

ユーザがファンクション複写定義を作成すると、Replication Server では、そのファンクション複写定義内の情報を含むファンクションが作成されます。

独自のファンクション複写定義を持つ複写ストアド・プロシージャが呼び出されると、そのファンクションが送信元 Replication Server から送信先 Replication Server に転送されます。通常、複写ストアド・プロシージャはプライマリ・データベースで呼び出され、レプリケート・データベースに配信されます。唯一の例外は、サブスクリプションを持たず、このような複写定義を使用した、15.1 より前のバージョンでの要求ファンクション複写です。この場合、ストアド・プロシージャはレプリケート・データベースで呼び出され、プライマリ・データベースに配信されます。どの状況でも、プライマリ Replication Server は常に、複写定義が作成された Replication Server です。この Replication Server はプライマリ・データベースを制御します。詳細については、『Replication Server 15.6 管理ガイド 第 2 巻』の「バージョン 15.1 より前の要求ファンクションの複写」を参照してください。

転送されたファンクションは、対応するストアド・プロシージャ、つまり送信先データベースで呼び出されるストアド・プロシージャにパラメータを渡します。「ファンクション文字列」は、サブスクリプションを作成するデータベースで解釈できる構文にファンクションを変換します。ファンクション複写を適切に使用すると、複数のオペレーションを単一のファンクションにカプセル化できるので、パフォーマンスを大幅に向上させることができます。複写ストアド・プロシージャは、複写のためにデータを更新する必要はありません。

適用ファンクション

プライマリ・データベースで実行されたオペレーションをレプリケート・データベースに分配するには、「適用ファンクション」を使用します。

適用ファンクションを使用すると、パフォーマンスが大幅に向上します。たとえば、クライアント・アプリケーションで非常に多くのローを更新する場合、ロー

を 1 つずつ複写するのではなく、複数のローを更新する適用ファンクションを作成できます。

適用ファンクションを使用するには、まずプライマリ・データベースでストアド・プロシージャを作成してから、それに対応するストアド・プロシージャをレプリケート・データベースに作成します。**sp_setrepproc** コマンドを使用して、ストアド・プロシージャに複写のマーク付けをします。プライマリ Replication Server で、そのストアド・プロシージャの適用ファンクション複写定義を作成します。レプリケート Replication Server は、そのファンクション複写定義のサブスクリプションを作成できます。プライマリ・データベースでストアド・プロシージャが呼び出されると、今度はレプリケート Replication Server が、サブスクリプションを作成するレプリケート データベースでストアド・プロシージャを実行します。

Replication Server は、ストアド・プロシージャの実行がサブスクリプションの対象となるまで、ストアド・プロシージャがレプリケート・データベースで必要とするデータを判別できないので、ファンクション複写定義のサブスクリプションを作成するときに、バルク・マテリアライゼーションまたは非マテリアライゼーション・メソッドを使用する必要があります。

Replication Server は、メンテナンス・ユーザとしてレプリケート・データベースのストアド・プロシージャを実行します。これは通常のデータ複写と同じです。

適用ファンクションの実装

適用ファンクションを実装するために複写オブジェクト作成し、コマンドを実行する方法について説明します。

適用ファンクションと要求ファンクションは非常に似ています。違いは、メンテナンス・ユーザがレプリケート・サイトで適用ファンクションを実行するのに対し、プライマリ・データベースでストアド・プロシージャを実行するユーザがレプリケート・サイトで要求ファンクションを実行することです。

1. 適用ファンクションの前提条件および制限事項を確認します。
2. ストアド・プロシージャが修正するレプリケート・テーブルを含むレプリケート・データベースを設定します。
3. プライマリ・データベースでストアド・プロシージャを作成します。ストアド・プロシージャは、プライマリ・データを修正する場合も、しない場合もあります。たとえば、次のストアド・プロシージャは、*@pub_name* パラメータを使用して `pub_name` カラム (`publishers` テーブル内) を更新します。

```
create proc update_pubs
@pub_id char(4), @pub_name varchar(40),
as
update publishers
set pub_name = @pub_name
where pub_id = @pub_id
```

4. プライマリ・データベースで、**sp_setrepproc** システム・プロシージャを使用して、複写ファンクションを配信するストアド・プロシージャにマークを付けます。次に例を示します。

```
sp_setrepproc update_pubs, 'function'
```

5. レプリケート・データベースで、プライマリ・データベースのストアド・プロシージャとパラメータおよびデータ型が同じストアド・プロシージャを作成します。通常、これらの 2 つのストアド・プロシージャは同じオペレーションを実行します。次に例を示します。

```
create proc update_pubs
pub_id char(4), @pub_name varchar(40),
as
update publishers
set pub_name = @pub_name
where pub_id = @pub_id
```

注意： レプリケート・データベースに作成されたストアド・プロシージャの名前は同じである必要はありませんが、パラメータ名とデータ型は同じである必要があります。

警告！ 適用ファンクションの配信によってレプリケート・データベースで呼び出されるストアド・プロシージャは、ユーザ定義トランザクションの内部で呼び出されます。ユーザ定義トランザクション内で許可されていないオペレーション (**dump transaction** コマンドや **dump database** コマンドなど) については、『Adaptive Server Enterprise Transact-SQL ユーザーズ・ガイド』を参照してください。

このストアド・プロシージャは、複写対象としてマーク付けしないでください。適用ファンクションの配信では、プライマリ・データベースのストアド・プロシージャだけが複写するようマーク付けされます。

ただし、レプリケート・データベースがスタンバイ・データベースを変更しているときに、スタンバイ・データベースに対してストアド・プロシージャの複写を使用する場合は、アクティブ・レプリケート・データベースとスタンバイ・レプリケート・データベースのストアド・プロシージャに複写対象のマークを付けます。

6. レプリケート・データベースで、ストアド・プロシージャに対する **execute** パーミッションをメンテナンス・ユーザに付与します。次に例を示します。

```
grant execute on update_pubs to maint_user
```

7. プライマリ Replication Server で、そのストアド・プロシージャの適用ファンクション複写定義を作成します。次に例を示します。

```
create applied function replication definition
update_pubs_rep
with primary at TOKYO_DS.pubs2
with all functions named update_pubs
(@pub_id char(4), @pub_name varchar(40),
```

```
 @state char (2))
searchable parameters (@pub_name, @state)
```

ファンクション複写定義には、プライマリ・データベースのストアド・プロシージャと同じパラメータ名とデータ型を使用します。複写したいパラメータだけを含めることができます。ファンクション複写定義にパラメータがない場合でも、この句にカッコを含める必要があります。

サーチャブル・パラメータを指定すると、ファンクションのパラメータ値に基づいて、ファンクション呼び出しにサブスクリプションを作成できます。前述の例では、*@pub_name* および *@state* がサーチャブル・パラメータです。たとえば、"CA" の更新にのみサブクスクリプションを作成できる場合などがあります。

Adaptive Server の `timestamp` データ型を複写する必要がある場合は、ファンクション複写定義にデータ型 `binary(8)` を宣言します。

『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**create applied function replication definition**」を参照してください。

8. ファンクション複写定義を作成すると、Replication Server は、対応するファンクションをデフォルトのファンクション文字列クラスで自動的に作成します。『Replication Server 管理ガイド 第 2 巻』の「データベース・オペレーションのカスタマイズ」で、「ファンクション、ファンクション文字列、クラスの処理」の「ファンクション」の「ユーザ定義ファンクション」を参照してください。

   デフォルトのファンクション文字列クラスまたはデフォルトから継承したクラスを使用しない場合や、ファンクションの呼び出しをカスタマイズする場合は、ユーザ定義ファンクションにファンクション文字列を作成する必要があります。
9. レプリケート Replication Server で、**create subscription** と非マテリアライゼーション・メソッド、または **define subscription** と他のバルク・マテリアライゼーション・コマンドを使用して、ファンクション複写定義にサブスクリプションを作成します。

   注意： アトミック・マテリアライゼーションまたはノンアトミック・マテリアライゼーションではなく、非マテリアライゼーション・メソッドまたはバルク・マテリアライゼーションを使用する必要があります。これは、Replication Server が、レプリケート・サイトでストアド・プロシージャに必要なデータを事前に判別できないためです。

   次に例を示します。

```
create subscription pubs_sub
for update_pubs_rep
with replicate at SYDNEY_DS.pubs2
where @state = 'CA'
without materialization
```

ファンクション複写定義にサーチャブル・パラメータを指定すると、ファンクションのパラメータ値に基づいて、ファンクション呼び出しにサブスクリプションを作成できます。この例では、*@state* パラメータの値が CA である場合にのみ、サブスクリプションはローを受信します。

『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**create subscription**」を参照してください。

10. 手順 1 から 9 で設定した Replication Server とデータベース・オブジェクトがすべて正しいロケーションにあることを確認します。これで、適用ファンクションを実行できます。

『Replication Server リファレンス・マニュアル』の「RSSD ストアド・プロシージャ」で、RSSD に複写システムに関する情報を問い合わせるために使用できる **rs_helpfunc** などのストアド・プロシージャの詳細を参照してください。

参照：

- 複写ファンクションの前提条件および制限 (386 ページ)
- ストアド・プロシージャへの複写のマーク付け (399 ページ)
- 複写ファンクションの修正または削除 (401 ページ)
- 複写ファンクションのファンクション文字列の作成または修正 (403 ページ)
- 非マテリアライゼーションの場合の create subscription コマンドの使用 (433 ページ)

要求ファンクション

プライマリ・データベースでストアド・プロシージャを呼び出した元のユーザがプライマリ・データベースからレプリケート・データベースに複写ストアド・プロシージャを配信する場合は、「要求ファンクション」を使用します。

通常、このタイプのファンクション複写を使用して、リモート・サイトから権限のあるユーザが中央データを変更できるようにします。たとえば、リモートのクライアント・アプリケーションが、中央データを変更する必要があるとします。クライアント・アプリケーションは最初にリモート・サイトでストアド・プロシージャを実行します。このプロシージャは、リモート・データベースで変更を行っても行わなくてもかまいません。ストアド・プロシージャを実行するときは、レプリケート Replication Server は要求ファンクションを中央サイトに渡し、そこで対応するストアド・プロシージャが呼び出されて中央データを更新します。この例では、リモート・データベースがプライマリ・データベースであり、中央データベースがこの要求ファンクションのレプリケート・データベースとなります。

プライマリ・コピー・モデルでは、単一の中央データベースにすべての最新の更新が含まれています。リモート・サイトのクライアント・アプリケーションは、要求ファンクションを使用して中央データを更新できます。中央テーブルに更新

が発生すると、Replication Server はその更新内容を取得し、適用ファンクションによってレプリケート・データ・サーバに送信します。ストアド・プロシージャの実行は、適切なデータベースに配信できるようになるまで Replication Server のステーブル・キューに格納されます。

要求ファンクションを使用するには、リモート・データベースでストアド・プロシージャを作成して、それに対応するストアド・プロシージャを中央データベースに作成します。次に、Replication Server で、リモート・データベースを制御する要求ファンクション複写定義を作成します。中央データベースを制御する Replication Server は、この要求ファンクション複製定義のサブスクリプションを作成できます。リモート・データベースのストアド・プロシージャが呼び出されると、中央データベースのストアド・プロシージャが呼び出されます。

中央データベースを管理する Replication Server は、リモート・データベースのストアド・プロシージャを実行したユーザとして、中央データベースのストアド・プロシージャを実行します。これにより、認可されたユーザだけが中央データを変更できることが保証されます。

アプリケーション内では、Replication Server は中央データベース内で変更されたデータの一部または全部を複写します。変更は、テーブル複写定義のサブスクリプションを持つ Replication Server によって管理されているリモート・データベースに分配されるか、または別の適用ファンクションとして分配されます。いずれの場合も、トランザクションの影響は、中央データベース、リモート・データベースの順に受信されます。

要求ファンクションを使用する場合、すべての更新は中央データベースで行われます。これによって、Replication Server のプライマリ・コピー・データ・モデルが維持され、複写システムにおけるネットワーク障害やトラフィック超過が防止されます。

要求ファンクションの実装

要求ファンクションを実装するために複写オブジェクトを作成し、コマンドを実行する方法について説明します。

適用ファンクションと要求ファンクションは非常に似ています。違いは、メンテナンス・ユーザがレプリケート・サイトで適用ファンクションを実行するのに対し、プライマリ・データベースでストアド・プロシージャを実行するユーザがレプリケート・サイトで要求ファンクションを実行することです。

1. 要求ファンクションの前提条件および制限事項を確認します。
2. レプリケート Adaptive Server でセキュリティを管理するために、ストアド・プロシージャを実行するユーザのログイン名とパスワードをレプリケート Adaptive Server で作成します。
3. レプリケート・データベースで、実際のデータを更新する複写ストアド・プロシージャを作成します。次に例を示します。

```
create proc update_pubs
@pub_id char(4), @pub_name varchar(40)
as
update publishers
set pub_name = @pub_name
where pub_id = @pub_id
```

警告！ 要求ファンクションの配信で呼び出されるストアド・プロシージャは、ユーザ定義トランザクションの内部で呼び出されます。ユーザ定義トランザクション内で許可されていないオペレーション (**dump transaction** コマンドや **dump database** コマンドなど) については、『Adaptive Server Enterprise Transact-SQL ユーザーズ・ガイド』を参照してください。

このストアド・プロシージャは複写対象としてマーク付けしないでください。ただし、このデータベースがウォーム・スタンバイ・アプリケーションの一部でもある場合、スタンバイ・データベースに対してストアド・プロシージャを複写したいときは、アクティブ・データベースのストアド・プロシージャに複写対象のマークを付けます。

4. レプリケート・データベースで、手順 2 でログイン名とパスワードを作成したユーザに、ストアド・プロシージャに対する **execute** パーミッションを与えます。要求ファンクションをレプリケート・データベースで複写するときは、このユーザが実行します。次に例を示します。

```
grant execute on update_pubs to pubs_user
```

5. プライマリ・データベースで、レプリケート・データベースのストアド・プロシージャとは別の名前を持つが、パラメータとデータ型が同じ要求プライマリ・ストアド・プロシージャを作成します。新しいストアド・プロシージャは何もしないか、または保留中の更新があることを示すメッセージを表示します。通常、このストアド・プロシージャの目的は、自分のデータベースのデータを変更する代わりに、他のデータベースに要求を送信することです。次に例を示します。

```
create proc update_pubs_request
@pub_id char(4), @pub_name varchar(40)
as
print "Transaction accepted."
```

注意： レプリケート・データベースとプライマリ・データベースで作成するストアド・プロシージャに異なる名前を使用します。通常のアプリケーションでは、ファンクションは適用ファンクションとして、後でプライマリ・データベースに複写し直されます。手順 8 で要求ファンクション複写定義を作成する場合は、プライマリ・データベースとレプリケート・データベースのストアド・プロシージャの名前を指定する必要があります。

6. プライマリ・データベースで、**sp_setrepproc** システム・プロシージャを使用して、複写ファンクションを配信するストアド・プロシージャにマークを付けます。次に例を示します。

```
sp_setrepproc update_pubs_request, 'function'
```

7. プライマリ・データベースで、ストアド・プロシージャを呼び出すプライマリ Replication Server のユーザに、そのストアド・プロシージャに対する **execute** パーミッションを付与します。次に例を示します。

   ```
   grant execute on update_pubs_request to pubs_user
   ```

8. 要求プライマリ・ストアド・プロシージャを管理するプライマリ Replication Server で、このストアド・プロシージャの要求ファンクション複写定義を作成します。次に例を示します。

   ```
   create request function replication definition
       update_pubs_request_rep
   with primary at TOKYO_DS.pubs2
   with primary function named update_pubs_request
   with replicate function named update_pubs
   (@pub_id char(4), @pub_name varchar(40)),
    @state char (2))
   searchable parameters ( @state)
   ```

   要求ファンクション複写定義には、レプリケート・データベースのストアド・プロシージャと同じパラメータ名とデータ型を使用します。複写したいパラメータだけを含めることができます。

   『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**create request function replication definition**」を参照してください。

9. ファンクション複写定義を作成すると、Replication Server は、対応するユーザ定義ファンクションを自動的に作成します。

   デフォルトのファンクション文字列を使用しない場合、またはファンクションの呼び出しをカスタマイズする場合は、ユーザ定義ファンクションにファンクション文字列を作成する必要があります。

10. レプリケート Replication Server で、**create subscription** と非マテリアライゼーション・メソッド、または **define subscription** と他のバルク・マテリアライゼーション・コマンドを使用して、要求ファンクション複写定義にサブスクリプションを作成します。次に例を示します。

    ```
    create subscription pubs_sub
    for update_pubs_request_rep
    with replicate at SYDNEY_DS.pubs2
    where @state = 'CA'
    without materialization
    ```

    ファンクション複写定義にサーチャブル・パラメータを指定すると、ファンクションのパラメータ値に基づいて、ファンクション呼び出しにサブスクリプションを作成できます。この例では、@state パラメータの値が "CA" である場合にのみ、サブスクリプションはローを受信します。

    『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**create subscription**」を参照してください。

注意： アトミック・マテリアライゼーションまたはノンアトミック・マテリアライゼーションではなく、非マテリアライゼーション・メソッドまたはバルク・マテリアライゼーションを使用する必要があります。これは、Replication Server が、レプリケート・サイトでストアド・プロシージャに必要なデータを事前に判別できないためです。

11. 手順 1 から 10 で設定した Replication Server とデータベース・オブジェクトがすべて正しいロケーションにあることを確認します。これで、プライマリ・データベースで要求ファンクションを実行できます。

    『Replication Server リファレンス・マニュアル』の「RSSD ストアド・プロシージャ」で、RSSD に複写システムに関する情報を問い合わせるために使用できる **rs_helpfunc** などのストアド・プロシージャの詳細を参照してください。

参照：

- 複写ファンクションの前提条件および制限 (386 ページ)
- Replication Server のセキュリティ管理 (233 ページ)
- ストアド・プロシージャへの複写のマーク付け (399 ページ)
- 複写ファンクションのファンクション文字列の作成または修正 (403 ページ)
- 非マテリアライゼーションの場合の create subscription コマンドの使用 (433 ページ)

ストアド・プロシージャへの複写のマーク付け

ストアド・プロシージャに複写のマークを付けるには、**sp_setrepproc** システム・プロシージャを使用します。

構文は次のとおりです。

```
sp_setrepproc [proc_name [, {'false' | 'table' | {'function' [,
{'log_current' | 'log_sproc'} ] } } ] ]
```

パラメータ：

proc_name — 現在のデータベース内のストアド・プロシージャ名です。

'function' — ファンクション複写定義に関連するストアド・プロシージャの複写を有効にします。

'table' — テーブル複写定義に関連するストアド・プロシージャの複写を有効にします。テーブル複写定義に関連するストアド・プロシージャの複写については、『Replication Server 管理ガイド 第 2 巻』の「非同期プロシージャ」を参照してください。

'false' — ストアド・プロシージャの複写を無効にします。

'log_current' ― 複写しているストアド・プロシージャの実行ログを、ストアド・プロシージャが存在するデータベースではなく、現在のデータベースに記録します。

'log_sproc' – 複写しているストアド・プロシージャの実行ログを、現在のデータベースではなく、ストアド・プロシージャが存在するデータベースに記録します。**'log_sproc'** がデフォルト・パラメータです。

sp_setrepproc は次のガイドラインに従って使用してください。

- データベース内のすべての複写オブジェクトをリストするには、パラメータを指定せずに **sp_setrepproc** を入力します。
- ストアド・プロシージャの複写ステータスを確認するには、ストアド・プロシージャの名前だけを指定して **sp_setrepproc** を入力します。
- 各タイプの複写を有効にしたり、ストアド・プロシージャの複写を無効にしたりするには、ストアド・プロシージャ名と **'function'**、**'table'**、または **'false'** を指定して、**sp_setrepproc** を入力します。**sp_setrepproc** を使用してストアド・プロシージャの複写ステータスを変更するには、システム管理者またはデータベース所有者である必要があります。
- 選択したデータベースでの複写ストアド・プロシージャの実行をログに記録する場合、現在のデータベースでの実行をログに記録するには **sp_setrepproc** に **'log_current'** を指定して入力し、ストアド・プロシージャが存在するデータベースでの実行をログに記録するには **'log_sproc'** を指定します。

適用ファンクション複写と要求ファンクション複写のいずれの場合も、**'function'** を指定して、ストアド・プロシージャに関連する複写定義のタイプを示します。

『Replication Server リファレンス・マニュアル』の「Adaptive Server コマンドとシステム・プロシージャ」の「**sp_setrepproc**」を参照してください。

複写ファンクションへのサブスクリプションの作成

create subscription と非マテリアライゼーション・メソッド、または **define subscription** とバルク・マテリアライゼーション用の他のコマンドを使用して、適用ファンクションまたは要求ファンクションのファンクション複写定義にサブスクリプションを作成する必要があります。バルク・マテリアライゼーション用コマンドには、**activate subscription**、**validate subscription**、**check subscription** があります。

唯一の例外は、15.1 以前のバージョンでサブスクリプションを作成せずに要求ファンクションのために使用されているファンクション複写定義です。詳細については、『Replication Server 15.6 管理ガイド 第 2 巻』の「バージョン 15.1 より前の要求ファンクションの複写」を参照してください。

ファンクション複写定義にサーチャブル・パラメータを指定すると、そのパラメータの値に基づいて、ファンクションにサブスクリプションを作成できます。

ファンクション複写定義のサブスクリプションは、**drop subscription** を使用して削除します。これらのサブスクリプションは、ファンクションに関連するレプリケート・データをパージすることなく削除されます。**without purge** オプションを指定する必要はありません。

バルク・マテリアライゼーションの詳しいコマンド構文については、『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」を参照してください。

参照：
- バルク・マテリアライゼーション (411 ページ)

複写ファンクションの修正または削除

複写ファンクションを修正または削除する方法について説明します。

ファンクション複写定義の変更

テーブル複写定義と同様に、ファンクション複写定義を変更すると、ファンクション複写定義の新しいバージョンが作成される場合があります。

Replication Server は、新しいバージョンのファンクション複写定義を使用して Replication Server システムに入力される新しいデータ・ローを処理しながら、古いバージョンの複写ファンクション定義を使用して複写システム内に既にある古いデータ・ローを処理します。

参照：
- 複写定義の変更要求プロセス (351 ページ)

ファンクション複写定義の修正

新しいパラメータまたは新しいサーチャブル・パラメータを追加したり、送信先ストアド・プロシージャの名前を変更したりするには、**alter applied function replication definition** コマンドと **alter request function replication definition** コマンドを使用してファンクション複写定義を変更します。

コマンドの構文は次のとおりです。

- **alter applied function replication definition**

```
alter applied function replication definition
function_applied_rep_def
    {with replicate function named `proc_name` |
    add @param_name datatype[, @param_name datatype]... |
    add searchable parameters @param_name[, @param_name]... |
```

```
    send standby {all | replication definition} parameters ...|
    }[with DSI_suspended]
```

- **alter request function replication definition**

```
alter request function replication definition
function_request_rep_def
    {with replicate function named 'proc_name' |
    add @param_name datatype[, @param_name datatype]... |
    add searchable parameters @param_name[, @param_name]... |
    send standby {all | replication definition} parameters ...|
    }[with DSI_suspended]
```

この 2 つのコマンドは、それぞれ **create applied function replication definition** コマンドと **create request function replication definition** コマンドによって作成されたファンクション複写定義を変更するために使用します。『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」で、**alter applied function replication definition** と **alter request function replication definition** の詳細を参照してください。

15.1 より前のバージョンの Replication Server からのファンクション複写定義を修正するには、『Replication Server 15.6 管理ガイド 第 2 巻』の「バージョン 15.1 より前の要求ファンクションの複写」を参照してください。

where 句 (**define subscription** コマンド内) に新しいサーチャブル・パラメータを追加するには、適用ファンクションおよび要求ファンクションのファンクション複写定義のサブスクリプションを削除して再作成します。

参照：

- 複写ファンクションのファンクション文字列の作成または修正 (403 ページ)
- 適用ファンクションの実装 (392 ページ)
- 要求ファンクションの実装 (396 ページ)

ファンクション複写定義の削除

パラメータを変更または削除する、あるいはファンクション複写定義の名前を変更するには、**drop function replication definition** コマンドを使用してファンクション複写定義を一度削除してから、再作成します。

このコマンドの構文は次のとおりです。

```
drop function replication definition function_rep_def
```

その後、ファンクション複写定義を削除して、再度作成してください。ファンクション複写定義を削除すると、対応するユーザ定義ファンクションとファンクション文字列も削除されます。ファンクション複写定義のサブスクリプションは、先に削除する必要があります。サブスクリプションは、ファンクション複写定義を再作成したあとに再作成できます。

複写ファンクションのファンクション文字列の作成または修正

ファンクション複写定義を作成または変更する場合、Replication Server は、対応するユーザ定義ファンクションを自動的に作成または変更します。ただし、**rs_default_function_class** のファンクション文字列を継承するクラスを使用しない場合は、ユーザ定義ファンクションのファンクション文字列を直接または間接的に作成する必要があります。

『Replication Server 管理ガイド 第 2 巻』の「データベース・オペレーションのカスタマイズ」で、「ファンクション、ファンクション文字列、クラスの処理」の「ファンクション」の「ユーザ定義ファンクション」を参照してください。

複写ファンクションの送信先データベースに割り当てられたファンクション文字列クラスのユーザ定義ファンクション用に、ファンクション文字列を作成します。ユーザ定義ファンクションのファンクション文字列を作成するには、プライマリ Replication Server で **create function string** を使用します。

『Replication Server 管理ガイド 第 2 巻』の「データベース・オペレーションのカスタマイズ」で、「ファンクション文字列の管理」の「ファンクション文字列とファンクション文字列クラス」を参照してください。

ファンクション複写定義を削除すると、Replication Server は、ユーザ定義ファンクションとファンクション文字列も必ず削除します。

ファンクション文字列は、それを許容するファンクション文字列クラスでカスタマイズできます。通常のアプリケーションでは、複写ユーザ定義ファンクションはストアド・プロシージャのパラメータ値を送信先 Replication Server に渡し、ファンクション文字列は送信先データベース内のこれらの値によってストアド・プロシージャを実行します。

デフォルトのファンクション文字列を変更して、監査ログへのデータ挿入などの別のアクションを実行するには、複写ファンクションのプライマリ Replication Server で **alter function string** コマンドを使用します。複写ファンクションの送信先データベースに割り当てられたファンクション文字列クラスを使用すると、ファンクション文字列をカスタマイズできます。

ファンクション文字列の作成および変更については、『Replication Server 管理ガイド 第 2 巻』の「データベース・オペレーションのカスタマイズ」を参照してください。『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**create function string**」を参照してください。

ストアド・プロシージャでのパブリケーションの使用

パブリケーションを使用すると、ストアド・プロシージャかテーブルまたはその両方をその複写定義とともに選択し、それらすべてを 1 つのグループとしてサブスクリプションを作成することができます。パブリケーションを使用すると、複写定義とサブスクリプションを構成して、単一のコマンドでそれらのステータスをモニタできます。

参照：

- パブリケーションの使用 (367 ページ)
- パブリケーション・サブスクリプション (452 ページ)

サブスクリプションの管理

サブスクリプションは、SQL の **select** 文に似ています。サブスクリプションは、サブスクリプションを作成している複写定義やパブリケーション、送信元および送信先データベースとデータ・サーバ、最初の情報をコピーするための「マテリアライゼーション」のメソッドを識別します。

where 句を使用すると、送信先データベースが送信元データベースから受け取るローまたはパラメータのサブセットを指定できます。

マテリアライゼーションとは、サブスクリプションで指定したデータをプライマリ・データベースからレプリケート・データベースへコピーして、レプリケート・テーブルを初期化する処理のことです。レプリケート・データはネットワークを介して転送するか、またはサブスクリプションが大量のデータを扱う場合は、メディアからロードできます。メディアからの初期化をバルク・マテリアライゼーションと呼びます。マテリアライゼーションによる複写システムへの影響の与え方によって、4 つのマテリアライゼーション・メソッドのどれかを使用します。

データベース複写定義のサブスクリプションでは、プライマリ・データベースのデータベース・オブジェクトをレプリケート・データベースに複写するよう Replication Server に指示します。個々のテーブル、トランザクション、ファンクション、システム・ストアド・プロシージャ、データ定義言語 (DDL) を複写するかしないかを個別に選択できます。Multi-Site Availability (MSA) と呼ばれるこのメソッドでは、プライマリ・データベースごとに 1 つのデータベース複写定義、サブスクリプションを作成するデータベースごとに 1 つのサブスクリプションのみが必要となります。

テーブル複写定義のサブスクリプションでは、プライマリ・テーブルのデータを指定のレプリケート・テーブルに複写するよう Replication Server に指示します。プライマリ・テーブルの複写定義を作成したら、レプリケート・サイトは、プライマリ・データベースの複写定義にサブスクリプションを作成して、更新を受信する必要があります。

ファンクション複写定義のサブスクリプションでは、非マテリアライゼーション・メソッドかバルク・マテリアライゼーション・メソッドを使用する必要があります。

パブリケーション・サブスクリプションを作成することによって、複写定義アーティクル・グループのサブスクリプションを作成できます。パブリケーション・サブスクリプションに **where** 句を含めることはできません。アーティクル内のローのサブセットにサブスクリプションを作成するには、アーティクルを作成するときに **where** 句を含める必要があります。

レプリケート・データが保持されるデータベースを管理する Replication Server で、サブスクリプションを作成します。事前に作成した複写定義によって、プライマリ・データのロケーションおよびプライマリ・テーブルの構造が定義されています。また、レプリケート・テーブルの構造がプライマリ・テーブルと構造が異なる場合は、オプションとしてレプリケート・テーブルの構造も定義されています。

プライマリ Replication Server とレプリケート Replication Server の両方の `rs_subscriptions` システム・テーブルに、サブスクリプションが追加されます。選択するマテリアライゼーション・メソッドによって、サブスクリプションの作成方法が変わります。

サブスクリプションでは大量のローを複写できるため、マテリアライゼーションがネットワークの負担になったり、プライマリ・データまたはレプリケート・データを使用するアプリケーションの処理を妨げたりする可能性があります。

参照：

- MSA を使用した複写オブジェクトの管理 (463 ページ)
- 複写ファンクションの管理 (385 ページ)

サブスクリプション・マテリアライゼーション・メソッド

Replication Server にはサブスクリプションを作成するためのメソッドが 4 つあるので、複写システムに対するマテリアライゼーションの影響を調整することができます。

表 28 : サブスクリプション・マテリアライゼーション・メソッド

メソッド	説明
アトミック・マテリアライゼーション (デフォルト)	このメソッドは、デフォルト形式の **create subscription** コマンドによって呼び出され、単一のアトミック・オペレーションでネットワークを介してサブスクリプション・データをコピーする。Replication Server は **rs_select_with_lock** ファンクションを実行してプライマリ・データを取り出す。このメソッドを使用すると、マテリアライゼーション処理全体で完全な一貫性が保たれるが、プライマリ・データまたはレプリケート・データを使用するトランザクションを一時的に妨げることがある。プライマリ・データベースで長時間実行されるトランザクションが許されない場合は、このメソッドをラージ・サブスクリプションに使用しないこと。

メソッド	説明
ノンアトミック・マテリアライゼーション	このメソッドは、**without holdlock** 句を指定した **create subscription** コマンドを使用して呼び出され、アトミック・メソッドに似ている。ただし、プライマリ・データベースのクライアントがマテリアライゼーション中にトランザクションを処理できるようにするため、マテリアライゼーション中の一貫性制約が緩和される点が異なる。Replication Server は **rs_select** ファンクションを実行してプライマリ・データを取り出す。サブスクリプション・データは、一連のトランザクションにコピーされる。ユーザはプライマリ・データを更新できるので、このメソッドでは、マテリアライゼーション中にトランザクションに矛盾が生じたり、データが不完全になったりする場合がある。ただし、マテリアライゼーションが終了するとすべての矛盾が完全に訂正される。矛盾を解決するために、レプリケート・テーブルのオートコレクションを有効にする必要がある。
非マテリアライゼーション	このメソッドは、**without materialization** 句を指定した **create subscription** コマンドによって呼び出される。このメソッドを使用すると、サブスクリプション・データがすでにレプリケート・データベースに存在する場合に、サブスクリプションを作成できる。また、テーブル複写定義、ファンクション複写定義、データベース複写定義のサブスクリプションも作成できる。
バルク・マテリアライゼーション	このメソッドは、ネットワークを介してコピーするにはデータが多すぎる場合に適している。これは、「手動」のマテリアライゼーション・メソッドであり、磁気テープなどのメディアからサブスクリプション・データをロードできる。 レプリケート・データベースでデータを初期化する必要がある場合に、このメソッドをデータベース複写定義およびファンクション複写定義のサブスクリプションに使用する。 バルク・マテリアライゼーションに使用するコマンドは、**define subscription**、**activate subscription**、および **validate subscription** である。

アトミック・マテリアライゼーション

アトミック・マテリアライゼーションは、デフォルトのマテリアライゼーション・メソッドです。これは、マテリアライゼーション処理全体で完全なデータの一貫性を確保して管理する、最も簡単な方法です。

アトミック・マテリアライゼーションにおいて、Replication Server は、サブスクリプションを作成したユーザ名とレプリケート Replication Server に定義されたパスワードを使って、プライマリ・データ・サーバにログインします。このため、サブスクリプションを作成するユーザ・アカウントは、レプリケート Replication Server とプライマリ・データベースの両方に同じパスワードで定義する必要があります。また、同じログイン名とパスワードのユーザ・アカウントが、プライマリ Replication Server にも必要です。

Replication Server は、プライマリ・データ・サーバにログインすると、**rs_select_with_lock** ファンクションに指定された **select with holdlock** オペレーションによって、サブスクリプション・ローを選択します。**holdlock** は繰り返し可能読み出しを実行し、**select** トランザクションが完了するまでプライマリ・サイトの他のトランザクションがデータを更新するのを防ぎます。ローは、レプリケート・サイトのマテリアライゼーション・キューに転送されて、レプリケート・データベースに適用されます。ステーブル・キューには、オペレーションを実行するのに十分なパーティション領域を確保しておく必要があります。

アトミック・マテリアライゼーションは、**select with holdlock** オペレーションが、プライマリ・データベースを使用するクライアント・アプリケーションを妨害するほど長時間継続しない、比較的小さなサブスクリプションに最適です。サブスクリプションで多数のローを選択する場合は、ノンアトミック・マテリアライゼーションまたはバルク・マテリアライゼーションを選択して、プライマリ・データベースのクライアントが影響を受けないようにすることができます。

データがすでにレプリケート・データベースに存在する場合は、非マテリアライゼーション・メソッドを使用できます。

アトミック・マテリアライゼーションでは、プライマリ・テーブルに対する変更が可能ですが、マテリアライゼーションのアクティブ化段階が完了するまで、データ・サーバの変更は有効になりません。

インクリメンタル・アトミック・マテリアライゼーション

incrementally オプションを使用すると、レプリケート・データベースで長時間実行されるトランザクションを回避できます。

incrementally オプションは、マテリアライゼーション・データを 1 つの大きなトランザクションではなく一連のトランザクションでレプリケート・データベースに送信します。それ以外の点では、インクリメンタル・アトミック・マテリアライゼーションと非インクリメンタル・アトミック・マテリアライゼーションは同じです。マテリアライゼーションが完了して、サブスクリプションが確定化するまでは、サブスクリプション・データは使用可能ですが、不完全です。

ローは、正常に挿入されるとステーブル・キューから削除されます。そのため、必要なパーティション領域が少なくてすみます。また、データベース・トランザクション・ログも、必要に応じてマテリアライゼーション中にトランケートできます。

マテリアライゼーション中、レプリケート・サイトのユーザにはサブスクリプション・データの一部しか見えないため、クエリの結果が不確定化されることがあります。ただし、挿入されたローにただちにアクセスできるメリットがあります。

複写テーブルの管理方法に関するトピックで示された **publishers_rep** 複写定義が、次の例でサブスクリプションを作成するために使用されます。この例の **create**

subscription コマンドには **where** 句がないため、Replication Server はサブスクリプションによって複写定義内のすべてのローを複写します。**incrementally** キーワードによって、レプリケート・データベースのトランザクション・ログが満杯になることを防ぎます。レプリケート・サイトのクライアントは、処理が終了するまで、サスペンドされるか、または `publishers` テーブルがマテリアライズ中で不完全なデータが含まれると警告される可能性があります。

```
create subscription publishers_sub
for publishers_rep
with replicate at SYDNEY_DS.pubs2
incrementally
```

参照：

- 複写定義名とテーブル名の指定 (311 ページ)

ノンアトミック・マテリアライゼーション

without holdlock オプション (**create subscription** コマンド内) を使用するノンアトミック・マテリアライゼーションは、いくつかの違いを除けばアトミック・マテリアライゼーションと同じです。

ノンアトミック・マテリアライゼーションがアトミック・マテリアライゼーションと異なる理由は、次のとおりです。

- データが、holdlock なしでプライマリ・データベースから選択される。プライマリ・サイトのクライアントは、**select** オペレーションの処理中にデータを更新できる。
- トランザクションは、レプリケート・データベースで常にインクリメンタルに適用される。

注意： 複写定義に **replicate minimal columns** 機能が設定されている場合は、ノンアトミック・マテリアライゼーションを使用して新しいサブスクリプションを作成することはできません。

バージョン 15.2 では、ノンアトミック・マテリアライゼーションにおいて、Replication Server は 1,000 ローのトランザクションとして、レプリケート・データベースへインクリメンタルにローを挿入します。テーブルを使用するレプリケート・サイトのクライアントには、マテリアライゼーション中、サブスクリプション・データの一部しか見えません。このため、クエリの結果が不確定化されることがあります。サブスクリプションは、データがレプリケート・データベースにコピーされる前にアクティブ化されるので、プライマリ・テーブルの変更が、場合によってはレプリケート・テーブルに 2 回適用される可能性があります。そのため、ノンアトミック・マテリアライゼーションを使用するときは、オートコレクションを有効にする必要があります。オートコレクションは、データの 2 回目の適用がエラーにならないようにします。

ノンアトミック・マテリアライゼーションのオートコレクションの使用

オートコレクションを有効にするには、ノンアトミック・マテリアライゼーションを使用してサブスクリプションを作成する予定の各複写定義に対して、**on** オプションを指定した **set autocorrection** コマンドを発行します。

オートコレクションを使用すると、Replication Server がプライマリ・テーブルでローの更新または挿入を行う場合、更新または挿入を削除と挿入に変換して、既存のローが原因で更新または挿入オペレーションが失敗しないようにします。

ノンアトミック・サブスクリプション・マテリアライゼーションにおいて、Replication Server は holdlock なしでデータを選択します。Replication Server は、選択したデータをレプリケート・データベースに追加したあとで、複写コマンドを適用します。オートコレクションを有効にすると、**without holdlock** オプションを使用してデータを選択することによって発生する一時的な矛盾が、Replication Server によって訂正されます。

ただし、マテリアライゼーション中にサブスクリプション・データを変更する複写ストアド・プロシージャを実行する場合は、オートコレクションによってもレプリケート・データベースが訂正されないことがあります。ファンクションを呼び出す間、オートコレクションは矛盾を防ぐことはできません。

ノンアトミック・マテリアライゼーションを使用するサブスクリプションをマテリアライズしたあとは、パフォーマンスを向上させるためにオートコレクションを無効にできます。オートコレクションを無効にすると、最少カラムの複写も指定できます。

『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**set autocorrection**」を参照してください。

参照：

- カラムの最少セットの複写 (317 ページ)

非マテリアライゼーション

create subscription に **without materialization** 句を指定して使用すると、マテリアライゼーションがすでに完了している場合にサブスクリプションをアクティブ化できます。

このメソッドを使用するには、次の条件が必要です。

- サブスクリプション・データがすでにレプリケート・データベースに存在していること。
- プライマリ・テーブルとレプリケート・テーブルが同期していること。
- Replication Server ステーブル・キューをそれ以上更新しないように、プライマリ・テーブルのアクティビティが停止されていること。

without materialization 句を使用してサブスクリプションを作成する場合、Replication Server はプライマリ Replication Server にサブスクリプションを作成するユーザとしてログインします。**create subscription** を実行するユーザは、プライマリ Replication Server とレプリケート Replication Server で同じログイン名とパスワードを持っている必要があります。

create subscription に **without materialization** 句を指定して使用すると、ファンクション複写定義にサブスクリプションを作成することもできます。

バルク・マテリアライゼーション

バルク・マテリアライゼーションでは、手動でデータベース間のサブスクリプション・データを転送します。

バルク・マテリアライゼーションは、サブスクリプションが大きすぎてネットワークを介してコピーできない場合に使用します。バルク・マテリアライゼーションは、プライマリ・データベース・クライアントやネットワークにほとんど影響を与えません。

バルク・マテリアライゼーションは、複写ファンクションのファンクション複写定義のサブスクリプションを作成するために使用できます。

バルク・マテリアライゼーションでは、マテリアライゼーション処理の異なる時点で実行されるコマンド **define subscription**、**activate subscription**、**validate subscription** を使用します。サブスクリプション・ステータスをチェックするには、**check subscription** コマンドを使用します。

バルク・マテリアライゼーションを使う場合、次の調整を行ってください。

- メディアに対するプライマリ・サイトのサブスクリプション・データのダンプ。
- メディアからレプリケート・サイトのテーブルへのロード。
- メディア・ダンプ後にプライマリ・サイトで行われた更新の適用。

注意： バルク・マテリアライゼーションでは、プライマリ・データベースとレプリケート・データベースでテーブル名やカラム名などが異なる場合、特別な処理を必要とすることがあります。

プライマリ・サイトとレプリケート・サイト間でデータの一貫性を保つバルク・マテリアライゼーション・メソッドは 3 つあります。使用するメソッドは、主にプライマリ・データを使用するアプリケーションが、割り込みを許容するかどうかによって決まります。

テーブル複写定義またはファンクション複写定義のどちらかのサブスクリプションに対して、これらのうち任意のメソッドを使用できます。ファンクション複写定義のサブスクリプションでは、レプリケート・データベースで実行される複写ストアド・プロシージャの影響を受けるのはどのレプリケート・テーブルか、明白ではない場合があります。

バルク・マテリアライゼーションを開始するには、まずこれらの問題をレプリケート・データベース内の既存データと関連付けて考える必要があります。

参照：
- 複写ファンクションの管理 (385 ページ)

バルク・マテリアライゼーション・メソッドの概要

3 つのバルク・マテリアライゼーション・メソッドがあります。

表 29 : バルク・マテリアライゼーション・メソッドの概要

メソッド	処理の概要
プライマリ・テーブルへの更新を停止して、データのスナップショットをとる	すべてのアプリケーションでプライマリ・データの更新を停止して、**select** 文またはデータベース・ダンプによって、プライマリ・データベースからサブスクリプション・データを取り出す。サブスクリプションを定義し、レプリケート・データベースの DSI をサスペンドした状態にするオプションによって、サブスクリプションをアクティブにする。このときクライアントは、プライマリ・データへの更新をレジュームできる。サブスクリプション・データをレプリケート・データベースへロードしたら、DSI をレジュームしてサブスクリプションを確定化できる。
アトミック・マテリアライゼーションをシミュレートする	サブスクリプション・データの取り出し中に、クライアント・アプリケーションでプライマリ・データに対するトランザクションを続行できる。サブスクリプションを定義したら、プライマリ・データをロックしてサブスクリプション・データを取り出し、サブスクリプションをアクティブにする。**activate subscription** コマンドで、レプリケート・データベースの DSI をサスペンドした状態にする。サブスクリプション・データをレプリケート・データベースへロードしたら、DSI をレジュームしてサブスクリプションを確定化できる。
ノンアトミック・マテリアライゼーションをシミュレートする	このメソッドは、アトミック・マテリアライゼーションのシミュレートと同じであるが、まずサブスクリプションをアクティブにしてから、プライマリ・データベースのデータをロックせずに取り出す点が異なる。このため、レプリケート・データベースのデータは、サブスクリプションが確定化されるまでプライマリ・データベースのデータと矛盾する場合があるので、レプリケート・データのオートコレクションを有効にする必要がある。

プライマリ・データベースで更新を停止してスナップショットをとる

プライマリ・データベースへの更新を停止してスナップショットをとるには、次のいずれかのバルク・マテリアライゼーション・メソッドを使用できます。

- Adaptive Server の **mount** コマンドを使用する

- Adaptive Server の **dump** コマンドと、**load**、**select**、または **bcp** コマンドを使用する

次の 2 つのメソッドがあります。プライマリ・データへの更新をサスペンドできる場合は、これらの方法でプライマリ・データベースからデータを取得できます。一貫性を保つために、マテリアライゼーション中はプライマリ・データベースへの更新がすべてサスペンドされます。

Adaptive Server の mount コマンドを使用してプライマリ・データベースからデータを取得する

プライマリ・データベースからデータを取得するには、**mount** コマンドを使用します。

前提条件

このメソッドを使用できるのは、Adaptive Server バージョン 12.5.1 以降を使用していて、プライマリ・データベースとレプリケート・データベースがまったく同じである場合にかぎられます。

手順

1. 複写システム全体が動作していることを確認します。

   『Replication Server 管理ガイド 第 2 巻』の「Replication Server の検証とモニタリング」を参照してください。
2. プライマリ・データに対するトランザクションを直接または Replication Server を通して間接的に生成するクライアント・アプリケーションを停止することによって、プライマリ・データベース内のデータへの更新をサスペンドします。
3. プライマリ Replication Server からレプリケート Replication Server へのデータの複写に関連する、複写システムのコンポーネントをクワイスします。

   admin quiesce_for_rsi コマンドをプライマリ Replication Server、レプリケート Replication Server、すべての中間 Replication Server で使用します。
4. Adaptive Server コマンド **quiesce database** *tag_name* **hold** *db_name list* **[for external dump] to** *manifest_file* **[with override]]** を実行して、マニフェスト・ファイルを生成します。

   『Adaptive Server Enterprise リファレンス・マニュアル』を参照してください。
5. データベースとログの両方のデバイスのデータ・ダンプを作成して、プライマリ・データベースのサブスクリプション・データのスナップショットをとります。

   データ・ダンプの作成には、**tar**、**zip** などのユーティリティか、UNIX の **dd** コマンドを使用できます。

6. **mount database** を使用して、レプリケート・データベースへのスナップショット・データのロードを開始します。
7. master データベースとロードされたユーザ・データベース間のユーザ情報の不一致を解消します。
8. レプリケート・データベースがまだない場合は、**rs_init** を使用して複写システムにレプリケート・データベースを追加します。
9. **define subscription** をレプリケート Replication Server で実行します。
10. **check subscription** をプライマリ Replication Server とレプリケート Replication Server で使用して、サブスクリプションが定義されていることを確認します。サブスクリプション・ステータスが両方のサーバで DEFINED である場合は、手順 11 に進みます。
11. **activate subscription** をレプリケート Replication Server で実行します。
12. **check subscription** をプライマリ Replication Server とレプリケート Replication Server で使用して、サブスクリプションがアクティブになっていることを確認します。サブスクリプション・ステータスが両方のサーバで ACTIVE である場合は、手順 13 に進みます。
13. **quiesce release** を実行して、プライマリ・データの更新をレジュームします。
14. **validate subscription** をレプリケート Replication Server で実行します。
15. **check subscription** をプライマリ Replication Server とレプリケート Replication Server で使用して、両方のサーバでサブスクリプションが VALID になっていることを確認します。

この手順を完了すると、サブスクリプションが作成され、レプリケート・データはプライマリ・データと一致し、複写はアクティブになります。

Adaptive Server の dump と、load、select、または bcp の各コマンドを使用してプライマリ・データベースからデータを取得する

Adaptive Server の **dump** と、**load**、**select**、または **bcp** の各コマンドおよびユーティリティを使用して、プライマリ・データベースからデータを取得します。

1. 複写システム全体が動作していることを確認します。

   『Replication Server 管理ガイド 第 2 巻』の「Replication Server の検証とモニタリング」を参照してください。
2. プライマリ・データに対するトランザクションを生成するクライアント・アプリケーションを停止することによって、プライマリ・データベース内のデータへの更新をサスペンドします。
3. プライマリ Replication Server からレプリケート Replication Server へのデータの複写に関連する、複写システムのコンポーネントをクワイスします。

admin quiesce_force_rsi コマンドをプライマリ Replication Server、レプリケート Replication Server、すべての中間 Replication Server で使用します。

4. プライマリ・データベースに対して **suspend log transfer** を実行します。
5. **select** 文またはデータベース・ダンプを使用して、プライマリ・データベースからサブスクリプション・データのスナップショットをとります。
6. **define subscription** をレプリケート Replication Server で実行します。
7. **check subscription** をプライマリ Replication Server とレプリケート Replication Server で使用して、サブスクリプションが定義されていることを確認します。サブスクリプション・ステータスが両方のサーバで DEFINED である場合は、手順 9 に進みます。
8. レプリケート Replication Server で **activate subscription** コマンドに **with suspension** 句を指定して実行します。
9. **check subscription** をプライマリ Replication Server とレプリケート Replication Server で使用して、サブスクリプションがアクティブになったことを確認します。レプリケート Replication Server でサブスクリプションがアクティブになると、レプリケート Replication Server への DSI コネクションがサスペンドします。

   サブスクリプション・ステータスが両方のサーバで ACTIVE である場合は、手順 11 に進みます。
10. プライマリ Replication Server でプライマリ・データベースから **resume log transfer** を実行します。
11. レプリケート・データベースへのスナップショット・データのロードを開始します。

    注意： レプリケート・データベースへのデータのロードが終了するまでの間に、次の手順に進むことができます。
12. **validate subscription** をレプリケート Replication Server で実行して、サブスクリプションを確定化します。
13. **check subscription** をプライマリ Replication Server とレプリケート Replication Server で使用して、両方のサーバのサブスクリプション・ステータスが VALID になっていることを確認します。
14. レプリケート・データベースへのスナップショット・データのロードが終了したら、**resume connection** コマンドを実行してレプリケート・データベースへのコネクションをレジュームします。

この手順を完了すると、サブスクリプションが作成され、レプリケート・データはプライマリ・データと一致し、複写はアクティブになります。

アトミック・マテリアライゼーションのシミュレート

バルク・マテリアライゼーション・メソッドとしてアトミック・マテリアライゼーションをシミュレートするメソッドは、プライマリ・データベースへの更新をサスペンドできない場合に使用します。

このメソッドは、サブスクリプション・データの取り出し、サブスクリプションのアクティブ化、レプリケート・データベースへの DSI コネクションのサスペンドを、プライマリ・データ・サーバでの 1 つのトランザクションですべて実行することによって、複写データの一貫性を保ちます。

1. 複写システム全体が動作していることを確認します。
   『Replication Server 管理ガイド 第 2 巻』の「Replication Server の検証とモニタリング」を参照してください。
2. **define subscription** コマンドをレプリケート Replication Server で実行します。
3. サブスクリプションがプライマリ Replication Server およびレプリケート Replication Server の両方で定義されるまで待ちます。プライマリ Replication Server とレプリケート Replication Server の両方で、**check subscription** コマンドを実行して、サブスクリプションのステータスが DEFINED であることを確認します。
4. このトランザクションの例に示すように、**select with holdlock** と **rs_marker** ストアド・プロシージャを含む単一のトランザクションを実行して、サブスクリプションをアクティブ化します。

   ```
   begin transaction
   select from table with holdlock
   where search_conditions
   execute rs_marker
   'activate subscription subid
   with suspension'
   commit transaction
   ```

   subid は、サブスクリプションを識別する整数です。サブスクリプションの *subid* は、RSSD の `subid` フィールド (`rs_subscriptions` システム・テーブル内) にあります。サブスクリプションが定義されたら、プライマリ Replication Server またはレプリケート Replication Server の RSSD で次のクエリを実行することによって、サブスクリプションの *subid* を検索できます。

   ```
   select subid from rs_subscriptions
   where subname = 'subscription'
   and dbid in (select connid from rs_databases
   where dbname = 'rep_connection_dbname'
   and dsname = 'rep_connection_dsname')
   ```

   rep_connection_dbname と *rep_connection_dsname* は、デフォルト・コネクションまたは代替コネクションについて指定する場合があります。

5. プライマリ Replication Server とレプリケート Replication Server で、サブスクリプションがアクティブになるまで待機します。レプリケート Replication Server で、**check subscription** コマンドを実行して、サブスクリプションのステータスが ACTIVE になったことを確認します。レプリケート Replication Server で、サブスクリプションのステータスが ACTIVE になると、レプリケート・データベースへのデータベース・コネクションはサスペンドします。
6. プライマリ Replication Server でサブスクリプションがアクティブになったらすぐに、**select** またはデータベース・ダンプを使用して、プライマリ・データベースからデータを取り出します。
7. サブスクリプションの ID 番号 (*subid*) を、`rs_subscriptions` システム・テーブルに問い合わせることによって検索します。

   ```
   select subid from rs_subscriptions
   where subname = 'subscription'
   and dbid in (select connid from rs_databases
   where dbname = 'rep_connection_dbname'
   and dsname = 'rep_connection_dsname')
   ```

8. **rs_marker** ストアド・プロシージャを、プライマリ・データベースで次のように実行します。

   ```
   rs_marker 'validate subscription subid'
   ```

   警告！ rs_marker ストアド・プロシージャには、必ずサブスクリプションの正しい *subid* 番号を付けて実行してください。`rs_subscriptions` システム・テーブルの `subid` カラムには、各サブスクリプションのユニークな ID 番号が格納されています。それ以外の番号または文字列を入力すると、重大な問題が発生します。

   rs_marker の詳細については、『Replication Server リファレンス・マニュアル』を参照してください。
9. レプリケート・データベースにサブスクリプション・データをロードします。
10. レプリケート・データベースで複写定義のオートコレクションを有効にします。
11. **resume connection** コマンドを使用して、レプリケート・データベースに対するデータベース・コネクションをレジュームします。
12. プライマリ Replication Server とレプリケート Replication Server の両方で、サブスクリプションが確定化されるまで待ちます。レプリケート Replication Server で、**check subscription** コマンドを実行して、サブスクリプション・ステータスが VALID になったことを確認します。サブスクリプション・ステータスが VALID になれば、レプリケート・データとプライマリ・データの一貫性が保たれます。
13. レプリケート・データベースのオートコレクションを無効にします。

以上の手順により、サブスクリプションが作成されて複写がアクティブになります。

参照：

- ノンアトミック・マテリアライゼーションのオートコレクションの使用 (410 ページ)
- 複写システム例でのテーブルの複写 (440 ページ)

ノンアトミック・マテリアライゼーションのシミュレート

バルク・マテリアライゼーションとしてノンアトミック・マテリアライゼーションをシミュレートするメソッドは、プライマリ・データベースへの更新をサスペンドできない場合か、サブスクリプション・データを取り出す **select** オペレーションまたは **dump** オペレーション中にプライマリ・データをロックできない場合に使用します。

このメソッドでは、レプリケート・データがプライマリ・データと矛盾する可能性があるレプリケート・サイトでの不安定な時間が許容されます。ただし、サブスクリプションが VALID になるまでに、データは一貫しなければなりません。マテリアライゼーション中はオートコレクションをオンにして、プライマリ・データベースで継続される更新によって発生する矛盾を、エラーが発生することなく解決できるようにする必要があります。

警告！ replicate minimal columns 機能が複写定義に設定されている場合や、適用ファンクションや適用ストアド・プロシージャをプライマリ・データベースから実行してレプリケート・データベースのデータを修正する場合には、このメソッドを使用しないでください。いずれの場合も、オートコレクションでは矛盾を解決できません。

1. 複写システム全体が動作していることを確認します。
   『Replication Server 管理ガイド 第 2 巻』の「Replication Server の検証とモニタリング」を参照してください。
2. **define subscription** コマンドをレプリケート Replication Server で実行します。
3. サブスクリプションがプライマリ Replication Server およびレプリケート Replication Server の両方で定義されるまで待ちます。プライマリ Replication Server とレプリケート Replication Server の両方で、**check subscription** コマンドを実行して、サブスクリプションのステータスが DEFINED であることを確認します。
4. レプリケート Replication Server で **activate subscription** コマンドに **with suspension** 句を指定して実行します。
5. プライマリ Replication Server とレプリケート Replication Server で、サブスクリプションがアクティブになるまで待機します。レプリケート Replication Server で、**check subscription** コマンドを実行して、サブスクリプションのステータス

が ACTIVE になったことを確認します。レプリケート Replication Server で、サブスクリプションのステータスが ACTIVE になると、レプリケート・データベースへのデータベース・コネクションはサスペンドします。

6. プライマリ Replication Server でサブスクリプションがアクティブになったらすぐに、**select** またはデータベース・ダンプを使用して、プライマリ・データベースからデータを取り出します。
7. サブスクリプションの ID 番号 (*subid*) を、`rs_subscriptions` システム・テーブルに問い合わせることによって検索します。

```
select subid from rs_subscriptions
where subname = 'subscription'
and dbid in (select connid from rs_databases
where dbname = 'rep_connection_dbname'
and dsname = 'rep_connection_dsname')
```

rep_connection_dbname と *rep_connection_dsname* は、デフォルト・コネクションまたは代替コネクションについて指定する場合があります。

8. **rs_marker** ストアド・プロシージャを、プライマリ・データベースで次のように実行します。

```
rs_marker 'validate subscription subid'
```

警告！ rs_marker ストアド・プロシージャには、必ずサブスクリプションの正しい *subid* 番号を付けて実行してください。`rs_subscriptions` システム・テーブルの *subid* カラムには、各サブスクリプションのユニークな ID 番号が格納されています。それ以外の番号または文字列を入力すると、重大な問題が発生します。

rs_marker の詳細については、『Replication Server リファレンス・マニュアル』を参照してください。

9. レプリケート・データベースにサブスクリプション・データをロードします。
10. レプリケート・データベースで複写定義のオートコレクションを有効にします。
11. **resume connection** コマンドを使用して、レプリケート・データベースに対するデータベース・コネクションをレジュームします。
12. プライマリ Replication Server とレプリケート Replication Server の両方で、サブスクリプションが確定化されるまで待ちます。レプリケート Replication Server で、**check subscription** コマンドを実行して、サブスクリプション・ステータスが VALID になったことを確認します。サブスクリプション・ステータスが VALID になれば、レプリケート・データとプライマリ・データの一貫性が保たれます。
13. レプリケート・データベースのオートコレクションを無効にします。

以上の手順により、サブスクリプションが作成されて複写がアクティブになります。

参照：
- ノンアトミック・マテリアライゼーションのオートコレクションの使用 (410 ページ)
- 複写システム例でのテーブルの複写 (440 ページ)

バルク・コピー・インとサブスクリプション・マテリアライゼーション

バルク・コピー・インを使用すると、サブスクリプション・マテリアライゼーションのパフォーマンスが向上します。

dsi_bulk_copy を on にすると、各トランザクションの **insert** コマンドの数が **dsi_bulk_threshold** を超えた場合に、Replication Server は、バルク・コピー・インを使用してサブスクリプションをマテリアライズします。

注意： 通常の複写で、**autocorrection** が on の場合、テーブルではバルク・オペレーションが無効になります。ただし、マテリアライゼーションでは、**dsi_bulk_threshold** に達していて、マテリアライゼーションが障害からリカバリするノンアトミック・サブスクリプションでない場合は、**autocorrection** が有効になっていても、バルク・オペレーションが適用されます。

参照：
- 物理データベース・コネクションに影響する設定パラメータ (203 ページ)

マテリアライゼーション解除処理

マテリアライゼーション解除では、サブスクリプションを削除し、オプションとしてレプリケート・データベースからデータを削除します。また、プライマリ・サイトとレプリケート・サイトの RSSD からサブスクリプション情報も削除します。

サブスクリプションを削除すると、Replication Server はプライマリ・データベースからレプリケート・データベースへの変更の送信を停止します。**drop subscription** コマンドを使用すると、テーブル複写定義とファンクション複写定義の両方のサブスクリプションを削除できます。

drop subscription は、プライマリ Replication Server とレプリケート Replication Server の RSSD からサブスクリプションを削除します。

テーブル複写定義のサブスクリプションを削除する場合、Replication Server がレプリケート・データベースからサブスクリプションのローを削除するように指定できます。または、ローを手動で削除することもできます。

ファンクション複写定義のサブスクリプションを削除しても、そのファンクションに関連するレプリケート・データはレプリケート・データベースから削除されません。

マテリアライゼーション解除には、次の 2 つのメソッドがあります。

- **with purge** マテリアライゼーション解除 ― 他のサブスクリプションで使用されていないローを選択して削除する。
- **without purge** マテリアライゼーション解除 ― レプリケート・テーブルのローを手動で削除できる。

どちらの場合も、削除されたサブスクリプションのデータが同じレプリケート・サイトの別のサブスクリプションに含まれていなければ、プライマリ Replication Server はそのデータの送信を停止します。

注意： ユーザ定義データ型の場合、クラス・レベル変換またはカラム・レベル変換の対象となるカラムを **where** 句で指定するサブスクリプションは、自動的にマテリアライゼーション解除することはできません。バルク・マテリアライゼーション・メソッドまたは非マテリアライゼーション・メソッドを使用する必要があります。

マテリアライゼーション解除とローのパージ

削除しようとしているサブスクリプションによって複写されたローを削除する場合は、**with purge** 句を使用します。

1000 ローずつインクリメンタルに削除するには、**incrementally** オプションを使用します。レプリケート・データベースのメンテナンス・ユーザがこのオプションを使用するには、テーブルに対して **select** パーミッションを持っている必要があります。

サブスクリプションのマテリアライゼーション解除とレプリケート・テーブルからのローのパージでは、**rs_select** または **rs_select_with_lock** システム・ファンクションのファンクション文字列を使用します。これらのシステム・ファンクション用にファンクション文字列を作成する必要がある場合もあります。

- レプリケート・データベース用のコネクションが、デフォルトで生成されるファンクション文字列を持つファンクション文字列クラス、またはそのようなクラスから継承するファンクション文字列クラスを使用する場合、Replication Server は **rs_select_with_lock** または **rs_select** ファンクションに対応するデフォルトのファンクション文字列を生成します。
- コネクションが他のファンクション文字列クラスを使用する場合は、サブスクリプションの **where** 句と一致する入力テンプレートで、ファンクション文字列を作成する必要があります。作成には **create function string** コマンドを使用します。

詳細については、『Replication Server 管理ガイド 第 2 巻』の「データベース・オペレーションのカスタマイズ」の「ファンクション文字列クラス」を参照してください。

ファンクション文字列をカスタマイズできるファンクション文字列クラスを使用している場合は、**alter function string** コマンドによって、既存のデフォルト・ファンクション文字列またはカスタム・ファンクション文字列を、アプリケーションが必要とする select オペレーションを実行するファンクション文字列に置き換えることができます。

rs_select ファンクション文字列と **rs_select_with_lock** ファンクション文字列の作成または変更の詳細については、『Replication Server 管理ガイド 第 2 巻』の「データベース・オペレーションのカスタマイズ」の「ファンクション文字列の管理」を参照してください。

ローをパージしないマテリアライゼーション解除

without purge オプションを使用してサブスクリプションを削除すると、サブスクリプションによって複写されたローがレプリケート・テーブルに残ります。

ファンクション複写定義のサブスクリプションは、**without purge** オプションを使用すると自動的に削除されます。このオプションを指定する必要はありません。ただし、レプリケート・テーブル内のローをそのまま保持したい場合は、このオプションを指定する必要があります。ローを手動で削除する場合は、**with suspension** オプションも使用する必要があります。

マテリアライゼーションとマテリアライゼーション解除のモニタリング

サブスクリプションは、完全に設定されるか複写システムから削除される前にいくつかの段階を経過します。サブスクリプションの設定またはサブスクリプション・データの削除を行う段階について説明します。

サブスクリプションを設定する段階は次のとおりです。

- 定義 – **create subscription** または **define subscription** で、プライマリ Replication Server とレプリケート Replication Server の RSSD にサブスクリプションを追加します。
- アクティブ化 – サブスクリプション解析後に実行されます。プライマリ Replication Server は、サブスクリプション・レゾルーション・エンジン (SRE: Subscription Resolution Engine) にサブスクリプションを追加します。SRE は、ログ・レコードを現在のサブスクリプションと比較して、複写テーブルに対する変更をどこに分配する必要があるかを判別します。

- マテリアライゼーション ― アトミック・サブスクリプションおよびノンアトミック・サブスクリプションの場合は、プライマリ Replication Server はプライマリ・データベースからサブスクリプション・データを取り出して、レプリケート・データベースに適用されるようにレプリケート Replication Server にそのデータをコピーします。
- 確定化 ― プライマリ Replication Server とレプリケート Replication Server はどちらも、サブスクリプションを完全にマテリアライズしてプライマリ・データと矛盾しないことを確認します。

drop subscription コマンドを使用してサブスクリプション・データを削除する段階は、次のとおりです。

- マテリアライゼーション解除 ― サブスクリプションに対する更新のレプリケート・データベースへの送信を停止します。**with purge** 句が指定された場合は、サブスクリプション・データをレプリケート・データベースから削除します (データが他のサブスクリプションに含まれていない場合)。**without purge** 句が指定された場合、Replication Server はデータをレプリケート・データベースから削除しません。
- 削除 ― プライマリ Replication Server およびレプリケート Replication Server の両方の RSSD からサブスクリプションを削除します。

マテリアライゼーションまたはマテリアライゼーション解除は、これらのどの段階でも失敗する可能性があります。そのため、**check subscription** コマンドを使用してサブスクリプションの進行状況をモニタする必要があります。

check subscription コマンドの他にも、**admin who** コマンドを使用して、サブスクリプションを処理している Replication Server スレッドのステータスをチェックすることもできます。アトミック・マテリアライゼーションおよびノンアトミック・マテリアライゼーションの場合、Replication Server はレプリケート・テーブルに追加されるローを含むマテリアライゼーション・キューを構築します。**admin who, sqm** コマンドはキュー・アクティビティをモニタすることができ、**admin who, dsi** コマンドは DSI スレッドが実行されているかどうかを確認できます。

『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」で、**admin who** の実行とその結果の解釈の詳細を参照してください。

サブスクリプション・ステータスと対処方法を詳しく説明した全般的なトラブルシューティング情報については、『Replication Server トラブルシューティング・ガイド』を参照してください。

参照：

- check subscription コマンド (436 ページ)

サブスクリプションを作成する前に複写システムの準備ができているかどうかを確認する

サブスクリプションを作成するには、まず複写システムの準備ができているかどうかを確認します。

1. 複写システム内のすべてのコンポーネントが動作していることを確認します。詳細については、『Replication Server 管理ガイド 第 2 巻』の「Replication Server の検証とモニタリング」の「複写システムの検証」を参照してください。
2. 次のデータベース・オブジェクトとパーミッションが存在していることを確認します。
   - プライマリ・テーブルに 1 つ以上の複写定義が存在すること。
   - ウォーム・スタンバイ・アプリケーションで、**sp_setreptable** または **sp_reptostandby** によって複写するようマーク付けされたプライマリ・テーブルが存在すること。
   - 複写定義に対応するテーブルがレプリケート・データベースに存在すること。そのカラムは、複写定義でレプリケート・データベースに指定されたカラムと一致しなければなりません。そのデータ型は、対応するプライマリ・カラムと一致しなければなりません。
     また、このテーブルは、サブスクリプションを作成するユーザとそれを保守するユーザが参照できなければなりません。複写定義に所有者名が含まれる場合は、このテーブルをすべてのデータベース・ユーザが参照できなければなりません。複写定義に所有者名が含まれていない場合は、データベース所有者にテーブルを作成させるのが、そのテーブルをアクセス可能にする最も簡単な方法です。
   - レプリケート・データベースのメンテナンス・ユーザには、次のものが必要です。
     レプリケート・テーブルに対する **select**、**insert**、**update**、**delete** パーミッションと、複写で使用されるファンクションの **execute** パーミッション。
     テーブルのサブスクリプションに **subscribe to truncate table** 句がある場合は、メンテナンス・ユーザに **replication_role**、**sa_role**、またはデータベース所有者のエイリアスが必要です。
3. 複写システム全体で使用される文字セットとソート順が、推奨ガイドラインに沿っていることを確認してください。これらは、サブスクリプションの処理において重要な役割を果たします。またサブスクリプションが有効であるためには、この文字セットとソート順がすべての場所で一貫性を持っていることが必要です。ガイドラインについては、『Replication Server デザイン・ガイド』を参照してください。

4. サブスクリプション・マテリアライゼーション・メソッドの 1 つを選択して、その選択したメソッドの次の要件を確認します。
   - ノンアトミック・マテリアライゼーションでは、レプリケート・テーブルのオートコレクションを有効にします。『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」で、**set autocorrection** の詳細を参照してください。
     複写定義に **replicate minimal columns** 機能が設定されている場合は、ノンアトミック・マテリアライゼーションを使用して新しいサブスクリプションを作成することはできません。
   - アトミックおよびノンアトミック・マテリアライゼーションの場合
     デフォルトのファンクション文字列クラス、またはデフォルトのファンクション文字列クラスから継承するファンクション文字列クラスは、**rs_select_with_lock** または **rs_select** ファンクション用のデフォルトのファンクション文字列を生成します。他のファンクション文字列クラスを使用する場合は、**rs_select_with_lock** または **rs_select** ファンクション用のファンクション文字列を、サブスクリプションの **where** 句と一致する入力テンプレートで作成する必要があります。
     rs_select または **rs_select_with_lock** を修正するには、レプリケート・データベース・コネクション内のファンクションではなく、プライマリ・データベース・コネクションと関連付けられているファンクション文字列クラスのファンクションを使用します。
     詳細については、『Replication Server 管理ガイド 第 2 巻』の「データベース・オペレーションのカスタマイズ」の「ファンクション文字列クラス」を参照してください。また、『Replication Server 管理ガイド 第 2 巻』の「データベース・オペレーションのカスタマイズ」の「ファンクション文字列の管理」で「入力テンプレート」も参照してください。
   - アトミックまたはノンアトミック・マテリアライゼーション・メソッドのいずれかとともにサブスクリプションを作成しており、かつ複写定義に引用符付き識別子が必要なカラムがある場合、引用符付き識別子を使用するようプライマリ・コネクションを設定してください。
5. サブスクリプションを作成する場合は、通常ユーザのログイン名を使用します。メンテナンス・ユーザとして、サブスクリプションを作成しないでください。

   サブスクリプションを作成するユーザが、次のログイン名とパーミッションを持っていることを確認します。
   - レプリケート Replication Server、プライマリ Replication Server、プライマリ・データ・サーバで同じログイン名とパスワード。バルク・マテリアライゼーション・メソッドまたは非マテリアライゼーション・メソッドを使用する場合は、プライマリ・データ・サーバのログイン名は必要ありません。

- プライマリ・テーブルに対する **select** パーミッション。これは、バルク・マテリアライゼーションまたは非マテリアライゼーションを使用する場合にはあてはまりません。
- **execute** パーミッション。これは、プライマリ・データベースまたは非マテリアライゼーションの **rs_marker** ストアド・プロシージャに対するパーミッションです。
- レプリケート Replication Server での **create object** または **sa** パーミッション。
- プライマリ Replication Server での **primary subscribe**、**create object**、または **sa** パーミッション。

参照：

- ノンアトミック・マテリアライゼーションのオートコレクションの使用 (410 ページ)
- サブスクリプション・マテリアライゼーション・メソッド (406 ページ)

サブスクリプション・コマンド

RCL コマンドによるサブスクリプションの管理方法について説明します。

RCL コマンドまたは Sybase Central を使用すると、次のことができます。

- アトミック・マテリアライゼーション、ノンアトミック・マテリアライゼーション、および非マテリアライゼーション・メソッドのサブスクリプションを作成する。
- バルク・マテリアライゼーションのサブスクリプションを定義、アクティブ化、および確定化する。
- マテリアライゼーション処理中のサブスクリプション・ステータスをチェックする。
- サブスクリプションを削除して、マテリアライゼーション解除処理を開始する。
- サブスクリプションを作成または定義するときに、**truncate table** コマンドの複写を有効にする。

where 句を使用すると、どのテーブル・ローまたはファンクション呼び出しを複写するかを制御できます。**where** 句には、テーブル複写定義またはファンクション複写定義に指定されているサーチャブル・カラムまたはサーチャブル・パラメータだけを指定できます。**where** 句を指定しない場合は、複写定義のカラムのすべてのロー、またはすべてのファンクション呼び出しが複写されます。

Adaptive Server Enterprise バージョン 11.5 以降を使用している場合は、**subscribe to truncate table** キーワードを含めると、送信先データベースで **truncate table** コマンドを再び生成できます。

表 30 : サブスクリプションを管理するためのコマンド

コマンド	作業
create subscription	次のいずれかを使用して、初期バージョンの複写データを転送するサブスクリプションを作成する。 • アトミック・マテリアライゼーション ― サブスクリプションの初期バージョンのデータを、単一のトランザクションとしてコピーする。 • ノンアトミック・マテリアライゼーション ― 一連のトランザクションでデータをコピーする。レプリケート・サイトのユーザは、すべてのデータを受信する前にその一部を見ることができる。Replication Server は、サブスクリプション・データのセット全体にマテリアライゼーション・キューを作成しない。 **create subscription** に **without materialization** 句を指定して使用し、レプリケート・データベースに初期バージョンの複写データがすでに存在するサブスクリプションをアクティブ化する。**create subscription** コマンドでは、テーブル複写定義のサブスクリプションも作成できる。ファンクション複写定義には、**create subscription** に **without materialization** 句を指定して使用する。
define subscription	バルク・マテリアライゼーションの最初の手順で、サブスクリプションを定義する。複写ファンクションを作成するときは、**define subscription** などのバルク・マテリアライゼーション・コマンドを使用して、テーブル複写定義またはファンクション複写定義のいずれかにサブスクリプションを作成できる。必要に応じて手動でデータを転送する。マテリアライゼーションが終了し、サブスクリプションがアクティブ化して確定化されると、データの複写が開始される。**check subscription** を使用してサブスクリプション・ステータスを確認する。
activate subscription	バルク・マテリアライゼーションの 2 番目の手順。プライマリ Replication Server およびレプリケート Replication Server の両方で、サブスクリプションをアクティブ化する。これによって、プライマリ Replication Server からレプリケート Replication Server へ、サブスクリプションのデータに対する変更の送信が開始される。
validate subscription	バルク・マテリアライゼーションの 3 番目の手順。サブスクリプション・ステータスをプライマリ・サイトとレプリケート・サイトの両方で VALID に変更する。
check subscription	プライマリ・サイトとレプリケート・サイトの両方でサブスクリプション・ステータスを確認する。このコマンドは、すべてのタイプのサブスクリプション・マテリアライゼーションで使用する。
drop subscription	複写システムからサブスクリプションを削除する。テーブル複写定義のサブスクリプションの場合は、マテリアライゼーション解除というオプションの処理でレプリケート・テーブルからサブスクリプション・ローを削除する。

参照：

- 複写ファンクションの管理 (385 ページ)
- truncate table の複写の有効化 (429 ページ)
- テーブル複写定義を管理するためのコマンド (306 ページ)
- ファンクション複写定義を管理するコマンド (389 ページ)

where 句の使用

サブスクリプションに 1 つの **where** 句を含めると、選択的なサブスクリプションを作成できます。

where 句の構文は、Transact-SQL の **where** 句のサブセットです。**where** 句は、複写定義に対するサブスクリプションの **create subscription** コマンドと **define subscription** コマンドでサポートされます。サポートされる構文はどちらのコマンドでも同じであり、非常に限定的なサブスクリプションを作成できます。これは、Replication Server のサブスクリプション・レゾルーション・エンジンによって効率的な処理が行われるように設計されています。

注意： サブスクリプション式で Java カラムは評価できません。このため、`rawobject` 型または `rawobject in row` 型のカラムを、サブスクリプションの **where** 句に挿入することはできません。

テーブル複写定義のサブスクリプションにおける、**where** 句の構文は次のとおりです。

```
where column_name{< | > | <= | >= | = | &} value
    [and column_name{< | > | <= | >= | = | &}
     value]...
```

ファンクション複写定義のサブスクリプションにおける、**where** 句の構文は次のとおりです。

```
where @param_name
        {< | > | <= | >= | = | &} value
    [and @param_name
        {< | > | <= | >= | = | &} value]...
```

他のデータ型の値の入力形式については、『Replication Server リファレンス・マニュアル』の「トピック」の「データ型」を参照してください。

注意： **!=**、**!<**、**!>**、**or** 演算子はサポートされていません。**or** 演算子を使用する代わりに、複数のサブスクリプションを作成できます。**&** 演算子は、`rs_address` カラムに対してのみ使用できます。

where 句の各カラム名は、テーブル複写定義またはファンクション複写定義の「**サーチャブル・カラム**」リストにリストされていることが必要です。各カラムの *value* は、それが比較されるカラムと同じデータ型でなければなりません。

たとえば、*state* = CA (テーブル複写定義 **publishers_rep**) であるデータにサブスクリプションを作成するように指定するには、次のように入力します。

```
create subscription publishers_sub1
for publishers_rep
with replicate at SYDNEY_DS.pubs2
where state = 'CA'
```

注意： **create subscription** 文の **where** 句の最大サイズは、255 文字です。

publishers の *state* = CA または *state* = MA であるデータにサブスクリプションを作成する場合、2 つのサブスクリプションを作成する必要があります。前述のコマンドに加えて、次のように入力します。

```
create subscription publishers_sub2
for publishers_rep
with replicate at SYDNEY_DS.pubs2
where state = 'MA'
```

注意： クラス・レベル変換またはカラム・レベル変換の対象となる異機種データ型カラムのサブスクリプションに **where** 句を使用する場合は、比較するデータ型が正しいことを確認してください。

参照：

- ビットマップ・サブスクリプション (445 ページ)
- 異機種データ型カラムのサブスクリプション (444 ページ)
- rs_address データ型の使用 (347 ページ)

truncate table の複写の有効化

Adaptive Server Enterprise バージョン 11.5 以降を使用している場合は、サブスクリプションを作成または定義するときに、特定の送信先データベース・テーブルへの **truncate table** コマンドの複写を有効にできます。

truncate table コマンドにより、1 つ以上のパーティションをトランケートできます。Replication Server は、プライマリ・データベースで実行されたコマンドと同じコマンドを再作成します。このためには、レプリケート・サイトのパーティション名が同じである必要があります。そうでない場合、DSI は停止します。

truncate table を省略してレプリケート・サイトで適切なアクションを適用するか、**rs_truncate** ファンクション文字列を使用してレプリケート・サイトのアクションをカスタマイズすることを選択できます。LTL バージョンが 700 に設定されると、Replication Agent はこのコマンドを送信します。

truncate table の複写を有効にするサブスクリプションを作成または定義するには、Replication Server にログインして次のように入力します。

```
create subscription subscription
    for table_rep_def
```

```
    with replicate at data_server.database
     ...
    subscribe to truncate table
```

truncate table が送信先データベースで実行されると、Adaptive Server はデータ・ページ全体の割り付けを解除します。ローを 1 つずつ削除するのではありません。

注意： Replication Server は、メンテナンス・ユーザとしてレプリケート・データベースで **truncate table** を実行します。メンテナンス・ユーザに付与されるパーミッションの中に **replication_role** があります。メンテナンス・ユーザの **replication_role** を取り消すと、**truncate table** を複写できません。ただし、そのメンテナンス・ユーザに **sa_role** が付与されているか、メンテンナス・ユーザがテーブルを所有しているか、またはメンテナンス・ユーザがデータベース所有者としてエイリアス指定されている場合を除きます。

ウォーム・スタンバイ・アプリケーションは、サブスクリプションを作成せずに、**truncate table** の実行をスタンバイ・データベースにコピーできます。『Replication Server 管理ガイド 第 2 巻』の「ウォーム・スタンバイ・アプリケーションの管理」の「ウォーム・スタンバイ・データベース・コネクションの変更」で、「論理コネクションの変更」の「スタンバイ・データベースへのトランケート・テーブルの複写」を参照してください。

『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」で、**define subscription** と **create subscription** の詳しいコマンド構文と使用法のガイドラインを参照してください。

subscribe to truncate table のステータスを変更する

レプリケート・テーブル上のすべてのサブスクリプションに対する "subscribe to truncate table" のステータスを変更するには、**sysadmin apply_truncate_table** コマンドを使用します。

特定のデータベース上のレプリケート・テーブルに対するサブスクリプションはすべて、**truncate table** の複写をサポートしているか、またはサポートしていないかのどちらかでなければなりません。テーブルに対する既存のサブスクリプションすべてが **truncate table** の複写をサポートしていない場合、**truncate table** の複写を有効にするサブスクリプションは作成できません。

たとえば、レプリケート・テーブルのすべてのサブスクリプションに対して **truncate table** の複写をオンにするには、レプリケート Replication Server にログインして **isql** プロンプトで次のコマンドを実行します。

```
sysadmin apply_truncate_table, data_server,
 database, {table_owner|''|""}, table_name'on'
```

data_server はレプリケート・データ・サーバの名前、*database* はデータ・サーバによって管理されるレプリケート・データベースの名前、*table_owner* はレプリケート・テーブルの所有者、そして *table_name* はレプリケート・テーブルの名前です。

複写定義にレプリケート・テーブル所有者を指定した場合は、**sysadmin apply_truncate_table** コマンドによってテーブル所有者も指定する必要があります。複写定義でレプリケート・テーブルの所有者を指定しなかった場合は、テーブル所有者名として '' (2 つの一重引用符文字) または "" (2 つの二重引用符文字) を入力します。

『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**sysadmin apply_truncate_table**」を参照してください。

create subscription コマンド

複写定義のサブスクリプションを作成してデータを複写するには、**create subscription** コマンドを使用します。

サブスクリプションを作成するには、次の 3 つのメソッドがあります。

- アトミック
- ノンアトミック
- 非マテリアライゼーション

where 句を使用すると、テーブル複写定義に指定されたサーチャブル・カラムの値に基づいて、プライマリ・テーブルから特定のローだけを複写できます。**where** 句を指定しないと、すべてのローが複写されます。

Adaptive Server Enterprise バージョン 11.5 以降を使用している場合は、**subscribe to truncate table** キーワードを含めると、送信先データベースで **truncate table** コマンドを再び生成できます。

注意： **create subscription** は、32K より大きい `text` データ、`unitext` データ、`image` データを自動的にトランケートします。

『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**create subscription**」を参照してください。

参照：

- where 句の使用 (428 ページ)
- truncate table の複写の有効化 (429 ページ)
- 複写定義の作成 (307 ページ)

アトミック・マテリアライゼーションの場合の create subscription コマンドの使用

create subscription を使用してアトミック・マテリアライゼーションを有効にします。

アトミック・マテリアライゼーションでサブスクリプションを作成するには、データが複写されるデータベースを管理する Replication Server で、**create subscription** コマンドを実行します。次に構文を示します。

```
create subscription subscription
    for table_rep_def
    with replicate at data_server.database
    [where search_conditions]
     [incrementally]
     [subscribe to truncate table]
```

subscription はアクティブにするサブスクリプションの名前、*table_rep_def* はサブスクリプションを作成するテーブル複写定義の名前、*data_server.database* はレプリケート・データベースを表します。

subscription 名は、複写定義とレプリケート・データベースでユニークでなければなりません。

ファンクション複写定義のサブスクリプションを作成するには、**define subscription** を使用するか (バルク・マテリアライゼーション・メソッドの場合)、または **create subscription** に **without materialization** 句を指定して使用する (非マテリアライゼーション・メソッドの場合) 必要があります。

オプションのキーワード **incrementally** を使用すると、Replication Server によって 1000 ローずつ挿入が送信されて、サブスクリプションが初期化されます。

キーワード **incrementally** を使用しない場合は、レプリケート・データベースのサブスクリプション・ローすべてが単一のトランザクションによって挿入されます。すべてのローが、一度にレプリケート Replication Server のステーブル・キューに保持されます。そのため、これらのローを収める十分なパーティション領域が必要です。さらに、レプリケート・データベースのトランザクション・ログには、トランザクションのログを記録するための十分な領域が必要です。

ノンアトミック・マテリアライゼーションの場合の create subscription コマンドの使用

ノンアトミック・マテリアライゼーションでサブスクリプションを作成するには、**create subscription** コマンドに **without holdlock** 句を指定して使用します。

構文は次のとおりです。

```
create subscription subscription
```

```
    for table_rep_def
    with replicate at data_server.database
    [where search_conditions]
    without holdlock
    [subscribe to truncate table]
```

subscription はアクティブにするサブスクリプションの名前、*table_rep_def* はサブスクリプションを作成するテーブル複写定義の名前、*data_server.database* はレプリケート・データベースを表します。

ノンアトミック・マテリアライゼーションは常にインクリメンタルです。

レプリケート・サイトのクライアントは、すべてのサブスクリプション・データがマテリアライズされるまで、サスペンドされるかまたはレプリケート・テーブルのデータが不完全で矛盾する可能性があることを警告されるため、マテリアライゼーションとマテリアライゼーション解除をモニタしてください。

参照：

• マテリアライゼーションとマテリアライゼーション解除のモニタリング (422 ページ)

非マテリアライゼーションの場合の create subscription コマンドの使用

サブスクリプション・データを初期化しないサブスクリプションを作成するには、レプリケート・データベースを管理する Replication Server で、**create subscription** コマンドに **without materialization** 句を指定して実行します。

非マテリアライゼーションでの **create subscription** コマンドの構文は次のとおりです。

```
create subscription subscription
for {table_rep_def | function_rep_def | publication pub |
    database replication definition db_repdef
    with primary at server_name.db }
with replicate at server_name.db
[where search_conditions]
without materialization
[subscribe to truncate table]
```

subscription は作成するサブスクリプションの名前、*table_rep_def* はサブスクリプションのテーブル複写定義の名前、*function_rep_def* はサブスクリプションのファンクション複写定義の名前、*pub* はサブスクリプションのパブリケーションの名前、*db_repdef* はサブスクリプションのデータベース複写定義の名前、*server_name.db* はプライマリ・データベースまたはレプリケート・データベースを表します。

without materialization 句は、サブスクリプション・データを最初に初期化せずにサブスクリプションをアクティブ化します。プライマリ・データベースになにもアクティビティがなく、レプリケート・データベースにデータがすでに存在する場合は、 **create subscription** に **without materialization** 句を指定して使用します。

define subscription コマンド

define subscription は、バルク・マテリアライゼーションを使用してサブスクリプションを作成するために使用します。

データが複写されるデータベースを管理する Replication Server で **define subscription** コマンドを実行します。**define subscription** は、サブスクリプション・ステータスを DEFINED に設定します。**define subscription** の構文は次のとおりです。

```
define subscription subscription
for {table_rep_def | function_rep_def
    publication pub_name | database replication definition db_repdef
    with primary at data_server.db
with replicate at data_server.db
[where search_conditions]
[subscribe to truncate table]
```

subscription は定義するサブスクリプションの名前、*table_rep_def* はサブスクリプションのテーブル複写定義の名前、*function_rep_def* はサブスクリプションのファンクション複写定義の名前、*pub_name* はサブスクリプションのパブリケーションの名前、*db_repdef* はサブスクリプションのデータベース複写定義の名前、*data_server.db* はプライマリ・データベースまたはレプリケート・データベースを表します。

subscription 名は、複写定義とレプリケート・データベースでユニークでなければなりません。

『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**define subscription**」を参照してください。

参照：

- 複写テーブルの管理 (299 ページ)
- 複写ファンクションの管理 (385 ページ)

activate subscription コマンド

サブスクリプションのプライマリ・データベースからレプリケート・データベースへの更新の分配を開始するには、**activate subscription** コマンドをバルク・マテリアライゼーション処理中に使用します。

activate subscription は、サブスクリプション・ステータスを ACTIVE に設定します。

active subscription は、**define subscription** コマンドを使用してサブスクリプションを作成した Replication Server で実行します。**activate subscription** の構文は次のとおりです。

```
activate subscription subscription
    for { table_rep_def | function_rep_def | publication pub_name |
    with primary at data_server.db }
with replicate at data_server.db
[with suspension [at active replicate only]]
```

subscription はアクティブにするサブスクリプションの名前、*table_rep_def* はサブスクリプションのテーブル複写定義の名前、*function_rep_def* はサブスクリプションのファンクション複写定義の名前、*pub_name* はサブスクリプションのパブリケーションの名前、*data_server.db* はプライマリ・データベースまたはレプリケート・データベースを表します。

with suspension 句を使用すると、サブスクリプション・ステータスを ACTIVE に変更してから DSI をサスペンドできます。これによって、サブスクリプション・データがロードされる前に、レプリケート Replication Server が複写テーブルへの更新を送信するのを防ぎます。レプリケート・サイトでデータをロードしてから、**resume connection** を実行して更新を適用します。

with suspension を使用しない場合は、サブスクリプションがマテリアライズされるまで、プライマリ・テーブルに対する更新を禁止する必要があります。

データベースがウォーム・スタンバイ・アプリケーションの一部である場合、**with suspension** 句は、アクティブ・データベースとスタンバイ・データベースの DSI をサスペンドします。このため、データを両方のデータベースにロードしてから、アクティブ・データベースへの更新を許可できます。ログを記録しながらアクティブ・データベースにデータをロードする場合は、**with suspension at active replicate only** 句を使用して、スタンバイ DSI をアクティブな状態に維持します。この場合、サブスクリプション・データはアクティブ・データベースから複写されます。ウォーム・スタンバイ・アプリケーションのアクティブ・データベースの DSI はサスペンドされます。この句は、スタンバイ・データベースの DSI はサスペンドしません。

『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**activate subscription**」を参照してください。

validate subscription コマンド

validate subscription を使用すると、バルク・マテリアライゼーション処理を終了して、サブスクリプション・ステータスを VALID に設定できます。

validate subscription は、サブスクリプションを作成した Replication Server で実行します。構文は次のとおりです。

```
validate subscription subscription
for { table_ref_def | function_rep_def | publication pub_name
    with primary at data_server.db }
with replicate at data_server.db
```

subscription は確定化するサブスクリプションの名前、*table_rep_def* はサブスクリプションのテーブル複写定義の名前、*function_rep_def* はサブスクリプションのファンクション複写定義の名前、*pub_name* はサブスクリプションのパブリケーションの名前、*data_server.db* はプライマリ・データベースまたはレプリケート・データベースを表します。

check subscription コマンド

check subscription を使用して、このコマンドが入力された Replication Server でのサブスクリプション・ステータスをレポートします。

サブスクリプションの作成中、サブスクリプション・ステータスがプライマリ Replication Server とレプリケート Replication Server で異なることがよくあります。このため、両方のサイトで **check subscription** を入力する必要があります。プライマリ・データベースとレプリケート・データベースが単一の Replication Server によって管理されている場合、**check subscription** は、プライマリ・データベースとレプリケート・データベースの両方のサブスクリプションのステータスを表示します。

構文は次のとおりです。

```
check subscription subscription
for { table_rep_def | function_rep_def | publication pub_name |
    database replication definition db_repdef
    with primary at data_server.db }
with replicate at data_server.db
```

subscription はチェックするサブスクリプションの名前、*table_rep_def* はサブスクリプションのテーブル複写定義の名前、*function_rep_def* はサブスクリプションのファンクション複写定義の名前、*pub_name* はサブスクリプションのパブリケーションの名前、*db_repdef* はサブスクリプションのデータベース複写定義の名前、*data_server.db* はプライマリ・データベースまたはレプリケート・データベースを表します。

このコマンドが返すメッセージには、サブスクリプション・ステータス情報が含まれています。サブスクリプションにエラーがある場合、メッセージは特定のエラー・メッセージが含まれているログを示します。

『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**check subscription**」で、**check subscription** が返すメッセージのリストを参照してください。

drop subscription コマンド

drop subscription を使用して、テーブル複写定義とファンクション複写定義の両方のサブスクリプションを削除します。

サブスクリプションを削除すると、Replication Server はプライマリ・データベースからレプリケート・データベースへの変更の送信を停止します。

drop subscription をレプリケート Replication Server で実行します。レプリケート Replication Server では **create object** パーミッションが、プライマリ Replication Server では **create object** パーミッションまたは **primary subscribe** パーミッションが必要です。

構文は次のとおりです。

```
drop subscription subscription
for {table_rep_def | function_rep_def | article article_name in
pub_name |
    publication pub_name | database replication definition db_repdef
    with primary at data_server.db }
with replicate at data_server.database
[without purge
[with suspension [at active replicate only ]] |
[incrementally] with purge]
```

without purge マテリアライゼーション解除メソッドを選択すると、Replication Server は、レプリケート・データベースからサブスクリプション・データを削除しません。

with purge マテリアライゼーション解除メソッドを選択すると、Replication Server はレプリケート・データベースにログインして、そこからデータを選択します。このデータが他のどのサブスクリプションにも属していない場合、サブスクリプション・データはレプリケート・データベースから削除されます。

テーブル複写定義のサブスクリプションを削除する場合は、サブスクリプションの作成時に使用したマテリアライゼーション・メソッドに関係なく、サブスクリプション・ローをパージできます。ローは、別のサブスクリプションに一致しない場合にのみ削除されます。

check subscription を使用すると、**drop subscription** の進行状況を表示できます。サブスクリプション・ステータスがプライマリ Replication Server およびレプリケート Replication Server に存在しなくなると、コマンドは終了です。

ファンクション複写定義のサブスクリプションは、常にそのファンクションに関連するレプリケート・データをパージすることなく削除されます。**without purge** オプションを指定する必要はありません。

テーブル複写定義のサブスクリプションを削除する場合は、2 つの基本的なメソッドから選択できます。各メソッドにはそれぞれ重要な意味があるため、

Replication Server では、これら 2 つのメソッドのうちどちらかを明示的に選択する必要があります。

- **with purge** ― Replication Server は、サブスクリプションのローが別のサブスクリプションに属していない場合、それらのローをレプリケート・データベースから削除、あるいはマテリアライゼーション解除します。Replication Server は、メンテナンス・ユーザとしてログインして、**select** オペレーションを実行します。トランザクションごとに 1000 ローずつ削除してマテリアライゼーション解除するように指定するには、**incrementally** オプションを使用します。
- **without purge** ― サブスクリプションのローは、レプリケート・データベースに残ります。**with suspension** オプションを指定すると、ローを手動で削除できるように、**drop subscription** が終了したとき、レプリケート・データベースへのコネクションがサスペンドしたままの状態になります。

ウォーム・スタンバイ・アプリケーションでは、オプション **with suspension at active replicate only** を指定すると、アクティブ・レプリケート・データベースはサスペンドしますが、スタンバイ・レプリケート・データベースはサスペンドしません。

警告！ ローを手動で削除する場合は、これらのローを必要とする重複するサブスクリプションのローを削除しないでください。

次に例を示します。

- with purge を使用してサブスクリプションを削除するには、次のように入力します。

```
drop subscription publishers_sub
for publishers_rep
with replicate at SYDNEY_DS.pubs2
with purge
```

- without purge を使用してサブスクリプションを削除するには、次のように入力します。

```
drop subscription publishers_sub
for publishers_rep
    with replicate at SYDNEY_DS.pubs2
without purge
```

- without purge を使用してサブスクリプションを削除し、レプリケート・データベースの DSI をサスペンドして、サブスクリプションのローを手動で削除できるようにするには、次のように入力します。

```
drop subscription publishers_sub
for publishers_rep
with replicate at SYDNEY_DS.pubs2
without purge
with suspension
```

- レプリケート・データベースに対してウォーム・スタンバイ・アプリケーションが存在する場合は、アクティブ・データベースへのコネクションだけをサス

ペンドして、スタンバイ DSI はアクティブにしておくことができます。すると、Replication Server はアクティブ・レプリケート・データベースからスタンバイ・データベースへ、ロー削除トランザクションを複写します。この場合は、次のように入力します。

```
drop subscription publishers_sub
for publishers_rep
with replicate at SYDNEY_DS.pubs2
without purge
with suspension at active replicate only
```

『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**drop subscription**」を参照してください。

サブスクリプションの例

このサブスクリプションの例では、RCL コマンドを使用して、テーブル複写定義にアトミック・サブスクリプションを作成することによって、プライマリ Adaptive Server データベースからレプリケート Adaptive Server データベースへ `publishers` テーブルを複写する方法を示します。これは、Sybase Central で実行することもできます。

以下の場所での複写システムの例：

- プライマリ・サイト：
  - Replication Server の名前は TOKYO_RS です。
  - プライマリ・バージョンの `publishers` テーブルは、TOKYO_DS という Adaptive Server の `pubs2` データベースにあります。TOKYO_RS から `pubs2` データベースへのコネクションが、Sybase Central または **rs_init** を使用して追加されており、データベースに RepAgent が設定されています。
  - TOKYO_RS のシステム・データベースの名前は `TOKYO_RSSD` で、Adaptive Server TOKYO_DS によって管理されています。
  - TOKYO_RS から SYDNEY_RS へのルートがあります。
- レプリケート・サイト：
  - Replication Server の名前は SYDNEY_RS です。
  - `publishers` テーブルのレプリケート・コピーは、SYDNEY_DS という Adaptive Server の `pubs2` データベースにあります。SYDNEY_RS から `pubs2` データベースへのコネクションは、Sybase Central または **rs_init** を使用して追加されています。
  - SYDNEY_RS のシステム・データベースの名前は `SYDNEY_RSSD` で、Adaptive Server SYDNEY_DS によって管理されています。

複写システム例でのテーブルの複写

2 つの Adaptive Server 間でテーブルを複写する手順について説明します。

1. レプリケート・テーブルを準備するには、Sybase Central または **isql** を使用してプライマリ・サイトおよびレプリケート・サイトの各サーバにログインし、複写システムのコンポーネントをチェックします。
2. TOKYO_DS プライマリ・データ・サーバでプライマリ・テーブルを準備するには、TOKYO_DS で `pubs2` データベースにログインし、`publishers` テーブルが存在することを確認します。

   ```
   isql -Usa -P -STOKYO_DS
   use pubs2
   go
   sp_help publishers
   go
   ```

3. サブスクリプションを作成するには、ログイン名を準備し、TOKYO_DS Adaptive Server でサブスクリプションを作成するユーザ "pubs2_user" に適切なパーミッションを付与します。このユーザが両方の Replication Server に必要です。
   a) TOKYO_DS でログイン名 "pubs2_user" を作成します。

      ```
      isql -Usa -P -STOKYO_DS
      sp_addlogin pubs2_user, pubs2_pw, pubs2
      go
      ```

   b) TOKYO_DS で、ログイン名 "pubs2_user" を `pubs2` データベースに追加し、pubs2_user に **select** パーミッション (`publishers` テーブルに対するパーミッション) を付与します。

      ```
      use pubs2
      go
      sp_adduser pubs2_user
      go
      grant select on publishers to pubs2_user
      go
      ```

   c) TOKYO_RS プライマリ Replication Server でログイン名 "pubs2_user" を作成し、pubs2_user に **primary subscribe** パーミッションを付与します。

      ```
      isql -Usa -P -STOKYO_RS
      create user pubs2_user
      set password pubs2_pw
      go
      grant primary subscribe to pubs2_user
      go
      ```

   d) SYDNEY_RS レプリケート Replication Server でログイン名 "pubs2_user" を作成し、SYDNEY_RS レプリケート Replication Server の pubs2_user に **create object** パーミッションを付与します。

```
isql -Usa -P -SSYDNEY_RS
create user pubs2_user
set password pubs2_pw
go
grant create object to pubs2_user
go
```

4. TOKYO_RS で、複写定義 **publishers_rep** (publishers テーブルの複写定義) を作成します。

```
isql -Ujohn -P -STOKYO_RS
create replication definition publishers_rep
with primary at TOKYO_DS.pubs2
with all tables named 'publishers'
(pub_id char(4), pub_name varchar(40),
city varchar(20), state char(2))
primary key (pub_id)
searchable columns (pub_id, pub_name)
replicate minimal columns
go
```

この例では、ユーザ "john" が複写定義を作成します。このユーザには、TOKYO_RS の **create object** パーミッションが必要です。

5. TOKYO_DS で、publishers プライマリ・テーブルを複写するようマーク付けします。**sp_setreptable** システム・プロシージャを使用してテーブルに複写するようマーク付けするには、データベース所有者かデータ・サーバのシステム管理者でなければなりません。次のように入力します。

```
sp_setreptable publishers, 'true'
go
```

6. SYDNEY_DS レプリケート・データ・サーバで pubs2 データベースにログインし、publishers テーブルが存在することを確認します。

```
isql -Usa -P -SSYDNEY_DS
use pubs2
go
sp_help publishers
go
```

Sybase Central または **rs_init** を使用してレプリケート・データベース pubs2 を追加すると、メンテナンス・ユーザが作成されて **replication_role** が与えられます。メンテナンス・ユーザに **replication_role**、**sa_role**、またはデータベース所有者のエイリアスがあれば、**truncate table** を複写できます。

SYDNEY_DS で、メンテナンス・ユーザに **select**、**insert**、**delete**、**update** パーミッション (publishers テーブルに対するパーミッション) があることを確認します。

```
grant all on publishers to SYDNEY_DS_maint
go
```

7. pubs2_user として SYDNEY_RS Replication Server にログインし、サブスクリプション **publishers_sub** (複写定義 **publishers_rep** に対するサブスクリプション) を作成します。

```
isql -Upubs2_user -Ppubs2_pw -SSYDNEY_RS
create subscription publishers_sub
for publishers_rep
with replicate at SYDNEY_DS.pubs2
subscribe to truncate table
go
```

   このサブスクリプションは、デフォルトのアトミック・マテリアライゼーションを使用します。**where** 句は含まれないため、すべてのローが複写されます。送信先データベースで、**truncate table** コマンドが再び実行されます。

8. まだ SYDNEY_RS にログインしている間に、**check subscription** コマンドを使用して、サブスクリプション・マテリアライゼーションのステータスをモニタします。

```
check subscription publishers_sub
for publishers_rep
with replicate at SYDNEY_DS.pubs2
go
```

9. 挿入したローがレプリケート・テーブルにコピーされるかどうか確認することで、複写が正常に行われるかどうか確認できます。
   a) TOKYO_DS で、`publishers` テーブルにローを挿入します。

```
isql -Usa -P -STOKYO_DS
use pubs2
go
insert publishers
values ('9950', 'Who Donut', 'Butler', 'CA')
go
```

   b) SYDNEY_DS で、挿入したローが `publishers` テーブルのレプリケート・コピーに複写されたことを確認します。

```
isql -Usa -P -SSYDNEY_DS
use pubs2
go
select * from publishers
go
```

text、unitext、image、rawobject データのマテリアライズ

通常、`text`、`unitext`、`image`、または `rawobject` データ型を使用するカラムを持つテーブルのサブスクリプションには、どのマテリアライゼーション・メソッドでも使用できます。

アトミックまたはノンアトミック・マテリアライゼーションを使用する場合、レプリケート・データベースを管理する Replication Server は、すべてのサブスクリ

プション・データをサブスクリプション・マテリアライゼーション・キューに選択します。

text、unitext、image、または rawobject 型のデータをマテリアライズするときは、データ・ローのサイズが 32K よりも小さい場合にかぎり、自動マテリアライゼーションを使用できます。それ以外の場合は、バルク・マテリアライゼーションを使用する必要があります。

サイズの大きなデータ・ローを大量にマテリアライズする場合は、Replication Server にデータのキュー領域が十分あることを確認してから、サブスクリプションを作成してください。大量の text、unitext、image、rawobject データを持つテーブルでは、マテリアライゼーションを完了するために、テンポラリ・パーティションを Replication Server に追加しなければならない場合があります。

ノンアトミック・マテリアライゼーション

ノンアトミック・サブスクリプション・マテリアライゼーションを使用しており、**replicate_if_changed** 複写ステータスをいずれかの text、unitext、image、または rawobject カラムに設定した場合、Replication Server はエラー・ログ・ファイルに警告メッセージを表示します。

サブスクリプション・マテリアライゼーション中にアプリケーションがプライマリ・テーブルを修正すると、データが矛盾する可能性があると警告されます。この場合は、**rs_subcmp** プログラムを実行してレプリケート・テーブルとプライマリ・テーブルのデータを一致させます。

ロー・マイグレーション

特定の条件のもとでは、text、unitext、image、rawobject カラムのデータが、ロー・マイグレーションの結果としてレプリケート・テーブルで失われる可能性があります。

ロー・マイグレーションは、**where** 句を含むサブスクリプションで起こります。**where** 句で指定されたカラムを更新すると、ローがサブスクリプションで確定化されたり、サブスクリプションにマイグレートしたりする場合があります。この状態になると、Replication Server はレプリケート・テーブルで **insert** を実行します。完全なローを挿入するには、各 **insert** にすべてのカラムの値が必要です。これには、プライマリ・テーブルで変更されなかった text、unitext、image、rawobject カラムも含まれます。

アプリケーションでローをサブスクリプションにマイグレートできる場合に、text、unitext、image、または rawobject カラムに **replicate_if_changed** 複写ステータスを設定していると、Replication Server はエラー・ログに警告メッセージを表示します。そのメッセージの内容は、ローがサブスクリプションにマイグ

レートされたが、その text、unitext、image、または rawobject データは失われているというものです。

ステータスが **replicate_if_changed** である text、unitext、image、または rawobject カラムが、プライマリ・テーブルの **update** オペレーションで変更されず、**update** によってローがサブスクリプションにマイグレートされると、レプリケート・テーブルで挿入されたローの text、unitext、image、または rawobject データが失われます。この場合は、**rs_subcmp** プログラムを実行してレプリケート・テーブルとプライマリ・テーブルのデータを一致させます。

異機種データ型カラムのサブスクリプション

クラス・レベル変換またはカラム・レベル変換が定義済みでアクティブな場合は、テーブル複写定義のサブスクリプションを通常の方法で作成します。ただし、**where** 句の使用に関して、次のようないくつかの制限があります。

- クラス・レベル変換またはカラム・レベル変換の対象となるカラムを **where** 句で指定するサブスクリプションは、自動的にマテリアライゼーション解除することはできません。バルク・マテリアライゼーション・メソッドまたは非マテリアライゼーション・メソッドを使用する必要があります。
- クラス・レベル変換またはカラム・レベル変換を **where** 句で指定するサブスクリプションの作成と定義には、注意が必要です。**where** 句の比較値は、宣言したデータ型のフォーマットでなければなりません。HDS 変換は、サブスクリプションを作成してから実行します。

  たとえば、サーチャブル・カラム starttime を datetime として宣言したものの、rs_db2_time としてパブリッシュする場合、**where** 句の比較値は datetime フォーマットで記述する必要があります。

```
create subscription db2_time_sub
for table_rep_def XXXXX
            with primary at AAAAA
            with replicate at BBBBB
where starttime > '19000101 23:14:02'
```

  "where starttime > '23:14:02,'"のように rs_db2_time フォーマットで記述しないでください。

複写テーブルを設定するときは、異機種データ型変換の詳細を参照してください。また、『Replication Server 異機種間複写ガイド』も参照してください。

参照：

- 複写テーブルの管理 (299 ページ)

ビットマップ・サブスクリプション

ビットマップ・サブスクリプションを使用すると、ビットマップ比較に基づいて複写するサブスクリプションを作成できます。

テーブルの複写定義を作成する場合は、ビットマップ・カラムのデータ型を `rs_address` と指定します。この特殊データ型は、Replication Server に対して、これらの `int` カラムをビットマップとして扱うように指示します。

create subscription コマンドと **define subscription** コマンドは、`rs_address` カラムまたはパラメータの **where** 句で、ビットマップ比較演算子 (&) をサポートします。

Adaptive Server テーブルでは、`int` カラムを使用してビットマップを保持します。これは、Adaptive Server では、ビット処理演算子を整数値に対して使用できるためです。`int` カラムの長さは 32 ビットです。アプリケーションに 33 ビット以上必要な場合は、複写定義に複数の `rs_address` カラムを用意できます。

サブスクリプションを作成する場合は、& 演算子を使用して各 `rs_address` カラムをビットマスクと比較することによって、ビットマップ比較を指定します。各サブスクリプションは、`rs_address` カラムごとに 1 つの比較を使用できます。

ビットマップ・サブスクリプションの例
一例として、`book_type` という名前の `rs_address` カラムを使用して、顧客が興味を持つ図書のカテゴリを記録するアプリケーションについて考えます。

表 31 : ビットマップ比較の例

ビット番号	図書カテゴリ
0	サイエンス・フィクション (SF)
1	ミステリ
2	ビジネス
3	料理
4	一般コンピュータ
5	コンピュータ・サイエンス
6	心理学
7	参考書

ビットが設定されている場合、顧客は対応するカテゴリの図書に対する興味を表明しています。各ビットは、最下位から最上位の順に番号が付けられています。

たとえば、顧客がミステリ、料理、コンピュータ・サイエンス、心理学関係の図書に興味を持っている場合、最下位の 8 ビットは 01101010 で、32 ビット整数値では 106 です。したがって、この顧客のローの `book_type` カラムには、値 106 が入っています。

特定の図書カテゴリに興味を持つ顧客のサブスクリプションを作成するには、目的のカテゴリのビットマスクを作成し、& 演算子を使用して、**create subscription** コマンドまたは **define subscription** コマンドの **where** 句の `book_type` カラムと作成したビットマスクを比較します。& 演算子は、ビット単位の AND 演算を実行します。結果がゼロ以外の場合、ローはサブスクリプションに一致します。

`rs_address` カラムの場合にのみ、次のように **where** 句でビットマップ比較演算子の & がサポートされます。

```
where rs_address_column1 & bitmask
[and rs_address_column2 & bitmask]
[and other_search_conditions]
```

たとえば、ミステリまたはビジネス書に興味を持つすべての顧客のサブスクリプションを作成する場合、マスクの下位 8 ビットは 00000110 です。ビットマスクを 32 ビットの整数値に変換すると、6 になります。アトミック・マテリアライゼーションまたはノンアトミック・マテリアライゼーションでは、次のようにサブスクリプションを作成できます。

```
create subscription mystery_or_business
for customers
with replicate at BRANCH_22.BOOK_DB
where book_type & 6
```

バルク・マテリアライゼーションで使用する **define subscription** コマンドでも、同じような方法を使用できます。非マテリアライゼーション・メッソドまたはバルク・マテリアライゼーションが必要なファンクション複写定義のサブスクリプションには、カラム名ではなくパラメータ名を指定します。

32 ビット整数値だけでなく、`rs_address` カラムと **where** 句の 32 ビット 16 進数を比較することもできます。16 進数を使用する場合は、必要に応じて各数字に 0 を埋め込み、8 桁の 16 進数値を作成します。

警告！ 16 進数値は、Adaptive Server と Replication Server の両方でバイナリ文字列として扱われます。バイナリ文字列は、バイトをコピーすることによって整数に変換されます。その結果のビット・パターンは、異なるプラットフォームでは、異なる整数値として示されることがあります。たとえば、0x0000100 は、バイト 0 を最上位バイトとするプラットフォーム上では 65,536 と見なされ、バイト 0 を最下位バイトとするプラットフォーム上では 256 と見なされます。これらのバイト順の違いが原因で、16 進数を含むビットマップ・サブスクリプションは、複写システムに異なるプラットフォームが存在する場合は機能しないことがあります。

rs_address カラムを、サブスクリプションの **where** 句内の 16 進数と比較する場合には、細心の注意が必要です。

変更されたカラムが rs_address カラムだけの場合は、変更されたビットがレプリケート・データベースでのローの挿入または削除を指示していないかぎり、Replication Server はローを複写しません。このフィルタリングによって、レプリケート・データベース内の rs_address カラムは、プライマリ・データベースの対応するカラムと同一にはならない場合があります。これは、rs_address カラムを使用して送信先レプリケート・データベースを指定するアプリケーションの最適化です。

create subscription コマンドと **create replication definition** コマンドを使用したビットマップ・サブスクリプションの作成の詳細については、『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」を参照してください。

データ型の変換の詳細については、『Adaptive Server Enterprise リファレンス・マニュアル』と『Open Client/Server Common Libraries リファレンス・マニュアル』を参照してください。

参照：

- where 句の使用 (428 ページ)

サブスクリプション情報の取得

データを複写した後で、サブスクリプションについての情報を取得したり、データが矛盾なく複写されているかどうか確認することが必要になる場合があります。Replication Server には、情報を取得するためのストアド・プロシージャと、一貫性を確認するためのスタンドアロン・ユーティリティが用意されています。

サブスクリプション情報の表示

Replication Server RSSD で **rs_helpsub** ストアド・プロシージャおよび **rs_helprepdb** ストアド・プロシージャを使用して、Replication Server でサブスクリプションに関する情報を表示します。

rs_helpsub を使用して、Replication Server でサブスクリプションに関する情報を表示します。構文は次のとおりです。

```
rs_helpsub [subscription_name
    [, replication_definition
    [, data_server, database]]]
```

rs_helprepdb ストアド・プロシージャを使用して、現在の Replication Server で複写定義のサブスクリプションを持つデータベースに関する情報を表示します。構文は次のとおりです。

```
rs_helprepdb [, data_server, database]
```

パラメータの説明については、『Replication Server リファレンス・マニュアル』の「RSSD ストアド・プロシージャ」を参照してください。

サブスクリプションの一貫性の確認

プライマリ・テーブルとレプリケート・テーブルの間に発生する可能性がある矛盾の種類について説明します。

サブスクリプションを作成すると、Replication Server は、プライマリ・データベースからレプリケート・データベースへトランザクションを送信します。複写システムは、テーブルのレプリケート・コピーとプライマリ・コピーの一貫性を保ちます。

レプリケート・データは、プライマリ・バージョンと矛盾する場合があります。たとえば、レプリケート・テーブルに対する更新パーミッションをデータベースのメンテナンス・ユーザに制限しなかった場合、クライアントがレプリケート・データを直接更新して、矛盾が発生する可能性があります。

Replication Server では、プライマリ・テーブルで発生した更新をレプリケート・テーブルに転送するのに若干の時間を要するため、プライマリ・テーブルとレプリケート・テーブルが一時的に矛盾した状態になる場合があります。しかし、Replication Server がレプリケート・データベースで更新を適用すると同時に、遅延時間による矛盾はなくなります。

プライマリ・テーブルとレプリケート・テーブルの間に発生する可能性がある矛盾には、次の 3 つの種類があります。

- プライマリ・テーブルで欠落したローが、レプリケート・テーブルから失われる。
- プライマリ・テーブルで矛盾するローが、レプリケート・テーブルの対応するローと異なる。
- レプリケート・テーブルの孤立したローが、プライマリ・テーブルに存在しないか、またはレプリケート・テーブルのサブスクリプションと一致しない。

遅延によって発生する一時的な矛盾と、システムの不適切な使用またはシステム障害が原因で発生する本来の矛盾を区別する必要があります。次の項で説明する **rs_subcmp** プログラムが、これを区別するのに役立ちます。サブスクリプションを削除して再作成するか、または **rs_subcmp** を使用すると、矛盾を訂正できます。

rs_subcmp を使用した矛盾の検索と訂正

rs_subcmp スタンドアロン実行プログラムを使用して、レプリケート・テーブルとテーブルのプライマリ・バーションを比較して、Sybase データベースのローの欠落、孤立、矛盾を検索し、必要に応じてそれらを訂正できます。

UNIX システムでは、このプログラムは **rs_subcmp** という名前です。PC システムでは、この実行プログラムは **subcmp** という名前です。**rs_subcmp** プログラムは、Sybase リリース・ディレクトリの `bin` サブディレクトリにあります。詳細については、使用しているプラットフォームの『Replication Server インストール・ガイド』および『Replication Server 設定ガイド』を参照してください。

このプログラムは、プライマリ・データ・サーバとレプリケート・データ・サーバにログインし、両方のテーブルのローを選択し、比較することによって動作します。

プライマリ・データとレプリケート・データ間の差異には遅延時間によると考えられるものもあるため、**rs_subcmp** は矛盾を識別してから、指定された回数の反復を実行します。**rs_subcmp** は、更新が複写されるまで待って、訂正されたローをそのリストから削除します。

最も良い使用方法は、遅延時間の少ないときに **rs_subcmp** を使用して、データ処理の繰り返しを避けることです。

rs_subcmp には、標準出力にローの矛盾を表示するか、矛盾を訂正するか、または矛盾を表示して訂正するかを指示できます。

設定ファイルを作成すると、エラーが発生しやすい複雑なコマンド・ラインでの入力を避けることができます。次に、データ・サーバ TOKYO_DS と SYDNEY_DS の `pubs2` データベースの `sales` テーブルを比較する、**rs_subcmp** 設定ファイルを示します。

```
PDS=TOKYO_DS
RDS=SYDNEY_DS
PDB=pubs2
RDB=pubs2
RTABLE=sales
RSELECT=select * from sales ¥
        order by stor_id, ord_num
RUSER=sa
KEY=stor_id
KEY=ord_num
RECONCILE=Y
RECONCILE_CHECK=Y
WAIT=15
NUM_TRIES=5
VISUAL=Y
```

パラメータ **PTABLE**、**PSELECT**、**PUSER** はプライマリ・データベースに使用されますが、この例には示されていません。これらの値はレプリケート・データベース内の対応するパラメータの値と同じであるため、設定ファイルに含める必要はありません。

RSELECT 行と **PSELECT** 行は、(使用する場合は) 1 行で入力する必要があります。ある行を次の行に続けて書く場合は、次のように各改行文字の前に円記号を付けます。

```
RSELECT=select * from sales ¥
        order by stor_id, ord_num
```

注意： 更新のフィルタ処理が原因で、rs_address データ型のカラムが、プライマリ・データベースとレプリケート・データベースで一致しない場合があります。**RSELECT** パラメータまたは **PSELECT** パラメータを使用して、rs_address カラムを選択しないでください。

rs_subcmp を実行する場合、コマンド・ライン・オプションによって設定ファイル内の値を上書きできます。たとえば、次の例で示すように、設定ファイルでデータ・サーバ TOKYO_DS の名前を TOKYO_DS2 に変更する代わりに、コマンド・ラインで **-S** フラグを使用してその名前を指定することができます。

```
rs_subcmp -f sales_cmp -S TOKYO_DS2 > sales_badrows
```

この例では、**-f** オプションで設定ファイル名 sales_cmp を指定しています。設定ファイルで **VISUAL** パラメータが "Y" に設定されている (**-V** コマンド・ライン・オプションに相当) 場合は、矛盾するローのリストが生成されます。この例では、出力がファイルにリダイレクトされています。

スキーマ比較

スキーマ比較は、格納されたデータは同じでも、スキーマが異なる可能性のある 2 つのデータベースのスキーマを比較するときに役立ちます。

たとえば、config.cfg ファイルを使用して 2 つのデータベース間のすべてのスキーマを比較するには、次のように指定します。

```
rs_subcmp -f config.cfg
```

スキーマ比較を実行するたびに、2 つのテーブルまたはデータベース間の比較結果の詳細を示すレポート・ファイルが作成されます。レポート・ファイルには、reportPROCID.txt という名前が付けられます。矛盾がある場合、**rs_subcmp** は reconcilePROCID.sql という名前の調整スクリプトを作成します。レポート・ファイルと調整スクリプトは、**rs_subcmp** を発行した同じディレクトリに保存されます。

注意： スキーマ比較のために **rs_subcmp** を実行する前に、使用する環境で **ddlgen** が機能していることを確認してください。

『Replication Server リファレンス・マニュアル』の「実行プログラム」の「**rs_subcmp**」を参照してください。

データの手動調整

実行前に文の調整を確認するため、**rs_subcmp** を使用して調整ファイルを作成できます。

コマンド・ライン・オプション **-g** を **rs_subcmp** で使用するか、設定ファイル・パラメータ RECONCILE_FILE を “Y” に設定して調整ファイルの作成を指示できます。

rs_subcmp のパフォーマンスの強化

ハッシュ・アルゴリズムにより、**rs_subcmp** のパフォーマンスが強化され、プライマリ・テーブルと複写テーブルのデータが圧縮されます。

圧縮されたデータは、**rs_subcmp** によってフェッチされます。

rs_subcmp は、プライマリ・テーブルと複写テーブルを比較するときにデータのロー全体を取得するのではなく、プライマリ・テーブルまたは複写テーブルから各データ・ローの圧縮されたデータだけを転送し、テーブル間の矛盾を検証または調整します。

rs_subcmp のパフォーマンスを向上させるためには、コマンド・ライン・パラメータ **-h**、**-H**、または、これらに対応する設定ファイル・パラメータの FASTCMP か HASH_OPTION を使用します。

注意： ハッシュ・アルゴリズムをサポートするには、**rs_subcmp** では、ASE 15.0.2 以降が必要です。また、大文字と小文字を区別した比較を処理することはできません。`text`、`unitext`、または `image` の各データ型を処理することもできません。さらに、ユーザが `float` データ型の精度を指定することもできません (最大精度が使用されます)。また、比較のパフォーマンスを向上させるために、ASE パラメータ **default data cache** を 128M 以上に設定することをおすすめします。

rs_subcmp プログラムには多数のオプションがあり、コマンド・ラインまたは設定ファイルで指定できます。これらの設定ファイル・パラメータとコマンド・ライン・オプションのリストについては、『Replication Server リファレンス・マニュアル』の「実行プログラム」の「**rs_subcmp**」を参照してください。

パブリケーション・サブスクリプション

パブリケーション・サブスクリプションでは、単一のコマンドを使用して複写定義のグループにサブスクリプションを作成します。

ユーザは、複写定義とそのアーティクルを、プライマリ Replication Server のパブリケーションにまとめます。レプリケート Replication Server で、そのパブリケーションに対するパブリケーション・サブスクリプションを作成します。

パブリケーション・サブスクリプションを作成すると、Replication Server は、パブリケーションの各アーティクルのサブスクリプションを作成します。

パブリケーション・サブスクリプションとアーティクル・サブスクリプションは、単一のサブスクリプションの規則と必要条件に従っていますが、例外が 1 つあります。これらのサブスクリプションには、**where** 句を含めることはできません。レプリケート Replication Server が受け取るローのサブセットを指定するには、アーティクルに **where** 句を含めます。

パブリケーションを使用するには、プライマリ Replication Server のバージョンが 11.5 以降であることが必要です。パブリケーション・サブスクリプションを使用するには、レプリケート Replication Server と、プライマリ Replication Server とレプリケート Replication Server からのルートが、バージョン 11.5 以降であることが必要です。

次の制限が適用されます。

- パブリケーションが確定化されて初めて、それに対してパブリケーション・サブスクリプションを作成できる。
- パブリケーション・サブスクリプションの名前が、パブリケーション、送信先データ・サーバ、および送信先データベースでユニークでなければならない。
- 1 つのプライマリ・テーブルに対する異なる複写定義を参照する 1 つ以上のパブリケーションに、アーティクルを含めることができる。ただし、各レプリケート・テーブルのプライマリ・テーブルごとに、複数の複写定義のサブスクリプションを作成することはできない。
  パブリケーション・サブスクリプションを作成して管理するには、コマンド・ラインを使用する。

参照：

- create article コマンドでの where 句の指定 (372 ページ)
- コマンド・ラインでのパブリケーションを使用したデータの複写 (368 ページ)
- パブリケーションの使用 (367 ページ)

パブリケーション・サブスクリプションを作成して管理するためのコマンド

パブリケーション・サブスクリプションを作成して管理するために使用できるコマンドについて説明します。

コマンドは、**check subscription** を除いてすべて、送信元 Replication Server では **primary subscribe** パーミッションまたは **create object** パーミッションを、送信先 Replication Server では **create object** パーミッションを必要とします。**check subscription** は誰でも実行できます。

表 32 : パブリケーション・サブスクリプションを管理するためのコマンド

コマンド	作業
create subscription *sub_name* **for publication** *pub_name*	パブリケーション・サブスクリプションと、パブリケーション内の各アーティクル・サブスクリプションを作成する。**create subscription** を使用すると、次のことができる。 • アトミック・マテリアライゼーション、ノンアトミック・マテリアライゼーション、または非マテリアライゼーションの各メソッドを使用して、テーブル複写定義のサブスクリプションを作成する。 • 非マテリアライゼーション・メソッドを使用して、ファンクション複写定義のサブスクリプションを作成する。
define subscription *sub_name* **for publication** *pub_name*	パブリケーション・サブスクリプションと、パブリケーション内の各アーティクル・サブスクリプションを定義する。**activate subscription** および **validate subscription** とともに使用する。 **define subscription** を使用すると、バルク・マテリアライゼーション・メソッドを使用して、テーブル複写定義またはファンクション複写定義のアーティクルにサブスクリプションを作成できる。
activate subscription *sub_name* **for publication** *pub_name*	パブリケーション・サブスクリプションと、パブリケーション内の各アーティクル・サブスクリプションをアクティブ化する。バルク・マテリアライゼーションで、**define subscription** および **validate subscription** とともに使用する。
validate subscription *sub_name* **for publication** *pub_name*	パブリケーション・サブスクリプションと、パブリケーション内の各アーティクル・サブスクリプションを確定化する。バルク・マテリアライゼーションで、**define subscription** および **activate subscription** とともに使用する。
check subscription *sub_name* **for publication** *pub_name*	パブリケーション・サブスクリプションと、そのアーティクル・サブスクリプションすべてのステータスを表示する。

コマンド	作業
check subscription *sub_name* **for article** *article_name* **in** *pub_name*	アーティクル・サブスクリプションのマテリアライゼーション・ステータスを表示する。
rs_helppubsub	パブリケーション・サブスクリプションについての情報を表示する。
drop subscription *sub_name* **for publication** *pub_name*	パブリケーション・サブスクリプションと、そのアーティクル・サブスクリプションすべてを、プライマリ・サイトおよびレプリケート・サイトの `rs_subscriptions` システム・テーブルから削除する。
drop subscription *sub_name* **for article** *article_name* **in** *pub_name*	パブリケーション・サブスクリプションとプライマリ・サイトおよびレプリケート・サイトの `rs_subscriptions` システム・テーブルから、アーティクル・サブスクリプションを削除する。

参照：

- create subscription コマンドを使用して、パブリケーション・サブスクリプションを作成する。 (455 ページ)
- バルク・マテリアライゼーションを使用したパブリケーション・サブスクリプションの作成 (457 ページ)
- サブスクリプション情報の表示 (447 ページ)
- パブリケーションとアーティクルのサブスクリプションの削除 (459 ページ)
- パブリケーションを作成して管理するためのコマンド (369 ページ)

truncate table コマンドの複写を有効にする

パブリケーション・サブスクリプションを作成、リフレッシュ、または定義するときに、レプリケート・テーブルに対する **truncate table** の複写を有効にすることができます。

truncate table の複写をレプリケート・テーブルで有効にしない場合は、レプリケート・データベースで **truncate table** を自分で実行しなければなりません。

たとえば、パブリケーション・サブスクリプション **pubs2_sub** を作成して **truncate table** の複写を有効にするには、送信先 Replication Server で次のコマンドを入力します。

```
create subscription pubs2_sub
        for publication pubs2_sub
        with primary at TOKYO_DS.pubs2
        with replicate at SYDNEY_DS.pubs2
        subscribe to truncate table
```

同じレプリケート・テーブルに対するサブスクリプションはすべて、**truncate table** を一貫して使用しなければなりません。**truncate table** の複写を有効にしてい

ないサブスクリプションをレプリケート・テーブルが持っている場合に、**truncate table** の複写を有効にしている別のサブスクリプションを追加すると、そのパブリケーション・サブスクリプションは失敗します。

パブリケーション・サブスクリプションをアクティブ化して確定化するときに、**subscribe to truncate table** を含める必要はありません。

参照：

- truncate table の複写の有効化 (429 ページ)

パブリケーション・サブスクリプションの作成

パブリケーションが確定化すれば、それに対してサブスクリプションを作成できます。

パブリケーション・サブスクリプションを作成すると、Replication Server は、パブリケーションの各アーティクルのサブスクリプションを作成します。

パブリケーション・サブスクリプションとアーティクル・サブスクリプションは、パブリケーション、プライマリ・データベースとレプリケート・データベース、およびマテリアライゼーション・メソッドを指定します。**where** 句は含まれません。複写するローのサブセットを指定するには、アーティクルの記述に **where** 句を含めます。

参照：

- create article コマンドでの where 句の指定 (372 ページ)

create subscription コマンドを使用して、パブリケーション・サブスクリプションを作成する。

create subscription を使用すると、パブリケーション・サブスクリプションとパブリケーション内の各アーティクル・サブスクリプションを作成できます。

create subscription を使用すると、アトミック、ノンアトミック、または非マテリアライゼーションの各メソッドを使用して、送信先データベースの送信元データをマテリアライズできます。

create subscription を、送信先データベースを管理する Replication Server で実行します。サブスクリプション情報は、プライマリ・サイトとレプリケート・サイトの `rs_subscriptions` システム・テーブルに保存されます。

この例では、パブリケーション **pubs2_pub** の **pubs2_sub** というサブスクリプションを作成します。また、**pubs2_pub** の各アーティクルにも、サブスクリプション **pubs2_sub** を作成しています。送信元データベースは、データ・サーバ TOKYO_DS によって管理されている `pubs2` です。送信先データベースも `pubs2` という名前で、データ・サーバ SYDNEY_DS によって管理されています。

```
create subscription pubs2_sub
        for publication
        with primary at TOKYO_DS.pubs2
        with replicate at SYDNEY_DS.pubs2
```

注意： **create subscription** 文の **where** 句の最大サイズは、255 文字です。

構文の詳細と使用方法のガイドラインについては、『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」を参照してください。

パブリケーション・サブスクリプションのマテリアライゼーション・メソッドの指定

パブリケーション・サブスクリプションのマテリアライゼーション・メソッドは、通常のサブスクリプションにマテリアライゼーション・メソッドを指定する場合と同じように指定します。

create subscription を使用する場合は、アトミック・マテリアライゼーション、ノンアトミック・マテリアライゼーション、または非マテリアライゼーションの各メソッドを指定できます。デフォルトのメソッドは、**select with holdlock** オペレーションを使用するアトミック・マテリアライゼーションです。

アーティクル・サブスクリプションは、親サブスクリプションの名前と、一般にそのマテリアライゼーション・メソッドの名前を共有します。ただし、ファンクション複写定義は、バルク・マテリアライゼーション・メソッドまたは非マテリアライゼーション・メソッドを必要とします。**create subscription** を使用する場合で、パブリケーションのアーティクルがファンクション複写定義を参照するときは、Replication Server は、パブリケーション・サブスクリプションに指定されたマテリアライゼーション・メソッドに関係なく、非マテリアライゼーション・メソッドをこれらのアーティクル・サブスクリプションに使用します。

参照：

- サブスクリプション・マテリアライゼーション・メソッド (406 ページ)

パブリケーション・サブスクリプションのリフレッシュ

既存のパブリケーションにアーティクルを追加する場合は、アーティクル・サブスクリプションを既存のパブリケーション・サブスクリプションに追加して、新しいアーティクルにサブスクリプションを作成する必要があります。この場合、**for new articles** 句を使用してサブスクリプションをリフレッシュします。

この句は、サブスクリプションをパブリケーションと照合し、サブスクリプションが作成されていないアーティクルすべてにサブスクリプションを作成するように、Replication Server に指示します。

たとえば、パブリケーション・サブスクリプション **pubs2_sub** をリフレッシュするには、送信先 Replication Server で次のコマンドを入力します。

```
create subscription sub for publication pub
        with primary at TOKYO_DS.pubs2
```

```
        with replicate at SYDNEY_DS.pubs2
        for new articles
```

check subscription を使用して、サブスクリプションがパブリケーション内の各アーティクルに存在するかどうか調べます。

参照：

- パブリケーション・サブスクリプション情報の表示 (460 ページ)

バルク・マテリアライゼーションを使用したパブリケーション・サブスクリプションの作成

バルク・マテリアライゼーションを使用すると、磁気テープなどのメディアからサブスクリプション・データをロードできます。転送するデータ量が多すぎてネットワーク経由でコピーできない場合に、このメソッドを使用します。

このメソッドを使用して、ファンクション複写定義のサブスクリプションを作成することもできます。

パブリケーション・サブスクリプションをバルク・マテリアライゼーションで作成する場合は、**define subscription**、**activate subscription**、**validate subscription** を使用する必要があります。これらのバルク・マテリアライゼーション・コマンドを使用すると、単一のサブスクリプションを作成するのと同じようにパブリケーション・サブスクリプションを作成できます。**where** 句をパブリケーション・サブスクリプションに含めることはできません。

参照：

- create article コマンドでの where 句の指定 (372 ページ)

パブリケーション・サブスクリプションに対する define subscription コマンドの使用

define subscription を使用すると、パブリケーション・サブスクリプションとパブリケーション内の各アーティクル・サブスクリプションを作成できます。

define subscription は、常に、バルク・マテリアライゼーションを使用してサブスクリプションを作成します。**define subscription** を、送信先データベースを管理する Replication Server で実行します。サブスクリプション情報は、送信元サイトと送信先サイトの `rs_subscriptions` システム・テーブルに保存されます。

パブリケーション・サブスクリプション内のすべてのサブスクリプションは、同時に作成されます。

この例では、パブリケーション **pubs2_pub** の **pubs2_sub** というサブスクリプションを作成します。

```
define subscription pubs2_sub
    for publication pubs2_pub
```

```
    with primary at TOKYO_DS.pubs2
    with replicate at SYDNEY_DS.pubs2
```

バルク・マテリアライゼーションでパブリケーション・サブスクリプションを定義する場合は、送信先テーブルに対する **truncate table** の複写を有効にできます。

構文の詳細と使用方法のガイドラインについては、『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」を参照してください。

参照：

- truncate table コマンドの複写を有効にする (454 ページ)

パブリケーション・サブスクリプションに対する activate subscription コマンドの使用

activate subscription を使用して、パブリケーション・サブスクリプションとそのサブスクリプション・サブセットをアクティブ化します。

activate subscription を、送信先データベースを管理する Replication Server で実行します。

activate subscription を実行する前に、まず **define subscription** を実行する必要があります。また、パブリケーション・サブスクリプションのステータスが DEFINED でなければなりません。

パブリケーション・サブスクリプション内のすべてのサブスクリプションは、同時にアクティブ化されます。

たとえば、**pubs2_sub** パブリケーション・サブスクリプションで各サブスクリプションをアクティブ化するには、以下を入力します。

```
activate subscription sub for publication pub
    with primary at TOKYO_DS.pubs2
    with replicate at SYDNEY_DS.pubs2
```

構文の詳細と使用方法のガイドラインについては、『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」を参照してください。

パブリケーション・サブスクリプションの validate subscription コマンドの使用

validate subscription を使用して、パブリケーション・サブスクリプションとそのサブスクリプション・サブセットのサブスクリプション・ステータスを VALID に設定します。

validate subscription を、レプリケート・データベースを管理する Replication Server で実行します。

validate subscription を実行する前に、まず **activate subscription** を実行する必要があります。また、パブリケーション・サブスクリプションのステータスが ACTIVE でなければなりません。

パブリケーション・サブスクリプション内のすべてのサブスクリプションは、同時に確定化されます。

次の例では、パブリケーション・サブスクリプション **pubs2_sub** 内のすべてのサブスクリプションを確定化しています。

```
validate subscription sub for publication pub
    with primary at TOKYO_DS.pubs2
    with replicate at SYDNEY_DS.pubs2
```

構文の詳細と使用方法のガイドラインについては、『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」を参照してください。

バルク・マテリアライゼーションを使用したパブリケーション・サブスクリプションのリフレッシュ

バルク・マテリアライゼーションを使用してパブリケーション・サブスクリプションをリフレッシュする場合は、パブリケーション・サブスクリプションを定義するときに **for new articles** 句を使用します。

サブスクリプションをアクティブ化して確定化する場合、この句を再度指定する必要はありません。次の例では、パブリケーション・サブスクリプション **pubs2_sub** をリフレッシュしています。

```
define subscription pubs2_sub
    for publication pubs2_pub
    with primary at TOKYO_DS.pubs2
    with replicate at SYDNEY_DS.pubs2
    for new articles
```

構文と使用方法のガイドラインについては、『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**define subscription**」を参照してください。

パブリケーション内の各アーティクルにサブスクリプションがあるかどうかチェックするには、**check subscription** をプライマリ Replication Server またはレプリケート Replication Server で実行します。

参照：

- パブリケーションおよびアーティクル・サブスクリプションのステータスの確認 (460 ページ)

パブリケーションとアーティクルのサブスクリプションの削除

drop subscription を使用して、パブリケーション・サブスクリプションとそのアーティクル・サブスクリプションすべてを削除したり、単一のアーティクル・サブスクリプションを削除します。

drop subscription は、パブリケーション・サブスクリプションとそのアーティクル・サブスクリプションに関する情報を、送信元サーバおよび送信先サーバのシ

ステム・テーブルから削除します。これによって、送信先サーバからパブリケーション情報が削除されることはありません。このため、パブリケーションに対して別のサブスクリプションを作成することができます。Replication Server は、プライマリ・サイト情報が変更されている場合に、それを再ロードするだけです。

サブスクリプションによって送信先データベースに複写された既存のローを保持するには、**without purge** 句を使用します。サブスクリプションは一度にすべて削除されます。

次の例では、**without purge** を使用して、パブリケーション **pubs2_pub** のサブスクリプション **pubs2_sub** を削除しています。

```
drop subscription pubs2_sub
    for publication pubs2_pub
    with primary at TOKYO_DS.pubs2
    with replicate at SYDNEY_DS.pubs2
    without purge
```

サブスクリプションによって送信先データベースに複写された既存のローを削除するには、**with purge** 句を使用します。サブスクリプションは一度に 1 つずつ削除されます。

次の例では、**with purge** を使用しています。

```
drop subscription pubs2_sub
    for publication pubs2_pub
    with primary at TOKYO_DS.pubs2
    with replicate at SYDNEY_DS.pubs2
    with purge
```

次の例では、サブスクリプションで複写されたローは削除せず、**pubs2_art** アーティクルを削除するには、次を入力します。

```
drop subscription sub for article pubs2_art
    with primary at TOKYO_DS.pubs2
    with replicate at SYDNEY_DS.pubs2
    without purge
```

構文の詳細と使用方法のガイドラインについては、『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」を参照してください。

パブリケーション・サブスクリプション情報の表示

パブリケーションとアーティクル・サブスクリプションについての情報は、**check subscription** コマンドまたは **rs_helppubsub** ストアド・プロシージャを使用して表示できます。

パブリケーションおよびアーティクル・サブスクリプションのステータスの確認

check subscription をプライマリ Replication Server またはレプリケート Replication Server で使用すると、パブリケーション・サブスクリプションとそのアーティク

ル・サブスクリプションのステータス、またはアーティクル・サブスクリプションのステータスをチェックできます。

check subscription は、VALID、MATERIALIZING、ACTIVE などのステータスを説明メッセージとともに返します。完全な構文と使用法のガイドライン、およびステータス・メッセージのリストについては、『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**check subscription**」を参照してください。

- 次の例は、パブリケーション・サブスクリプション **pubs2_sub** のサブスクリプション・ステータスを表示します。

```
check subscription pubs2_sub
            for publication pubs2_pub
    with primary at TOKYO_DS.pubs2
    with replicate at SYDNEY_DS.pubs2
```

  パブリケーション・サブスクリプションのステータスが VALID の場合、Replication Server は、サブスクリプションが現在のものであるかどうかもチェックします。サブスクリプションの作成後にパブリケーションを変更すると、パブリケーション・サブスクリプションはパブリケーションと同期しなくなります。新しいアーティクルのサブスクリプションを作成して、それを現在のサブスクリプションにするには、**create subscription** または **define subscription** を使用して、サブスクリプションをリフレッシュします。
- 次の例は、サブスクリプション **pubs2_sub** 内のアーティクル **pubs2_art** のサブスクリプション・ステータスを表示します。

```
check subscription sub for article pubs2_art
    in pubs2_pub
    with primary at TOKYO_DS.pubs2
    with replicate at SYDNEY_DS.pubs2
```

パブリケーションとアーティクル・サブスクリプションの情報の表示

パブリケーション・サブスクリプションとアーティクル・サブスクリプションについての情報を表示するには、プライマリ Replication Server またはレプリケート Replication Server の RSSD のどちらかで **rs_helppubsub** ストアド・プロシージャを実行します。

次に例を示します。

- サイトのすべてのパブリケーション・サブスクリプションをリストするには、次のように入力します。

```
rs_helppubsub
```

  サイトに認識されている各パブリケーション・サブスクリプションについて、サブスクリプションとそれに関連するパブリケーションの名前、プライマリ・データベースとレプリケート・データベースおよびデータ・サーバの名前、ス

テータス情報、パブリケーション・サブスクリプションの最終変更日付が表示されます。

- 特定のパブリケーション・サブスクリプションについての情報を表示するには、次のように入力します。

```
rs_helppubsub subscription_name
```

subscription_name という名前のすべてのパブリケーション・サブスクリプションについて、前述の例で説明した情報が表示されます。

- 特定のパブリケーション・サブスクリプションとそのアーティクル・サブスクリプションについての情報を表示するには、次のように入力します。

```
rs_helppub subscription_name, publication_name,
    primary_dataserver, primary_db,
    replicate_dataserver, replicate_db
```

subscription_name という名前のすべてのパブリケーション・サブスクリプションについて、前述の例で説明した情報が表示されます。各アーティクル・サブスクリプションについて、サブスクリプションとアーティクルの名前、プライマリ Replication Server およびレプリケート Replication Server のステータス情報、複写定義名、オートコレクション・ステータス、アーティクル・サブスクリプションの最終変更日付が表示されます。

完全な構文と使用法のガイドライン、および出力例については、『Replication Server リファレンス・マニュアル』の「RSSD ストアド・プロシージャ」の「**rs_helppubsub**」を参照してください。

MSA を使用した複写オブジェクトの管理

Multi-Site Availability (MSA) を使用すると、複写システムをすばやく簡単に設定できます。

MSA には次のような機能があります。

- プライマリ・データベースのための 1 つの複写定義と、サブスクリプションを必要とするデータベースのための 1 つのサブスクリプションしか必要としない簡単な複写方法。
- 個々のテーブル、トランザクション、ファンクション、システム・ストアド・プロシージャ、データ定義言語 (DDL) を複写するかどうかを選択できる、複写フィルタリング方式。
- すべてのレプリケート・データベース (非ウォーム・スタンバイ・データベースも含む) への DDL の複写。
- 複数のレプリケート・サイト (ウォーム・スタンバイ・データベースと非ウォーム・スタンバイ・データベースの両方) への複写。

既存の複写構造の上に MSA を導入することもできます。MSA を実装する手順は、ウォーム・スタンバイまたはレプリケート・データベースへの複写に使用している手順と似ています。

データベースの複写
テーブル複写およびファンクション複写を使用する場合は、個々のテーブル複写定義、ファンクション複写定義、およびサブスクリプションを使用して複写する個々のデータを記述します。

この方法によって、データを変換し、レプリケート・データベースに格納される情報を詳細に制御できます。ただし、複写するそれぞれのテーブルまたはファンクションにマークを付け、それぞれに対する複写定義を作成し、各レプリケート・データベースで各複写定義へのサブスクリプションを作成する必要があります。

MSA を使用すると、1 つの複写定義内でテーブル、ファンクション、トランザクション、DDL、システム・ストアド・プロシージャなどの特定のデータベース・オブジェクトを識別できます。また、データベース全体を複写することを選択したり、データベース内の特定のテーブル、ファンクション、トランザクション、DDL、システム・ストアド・プロシージャを複写するかどうかを選択できます。部分テーブルの複写が必要ない場合は、MSA は簡単な設定および管理で複写を行えます。

複写対象がウォーム・スタンバイ・データベースの場合
非 MSA ウォーム・スタンバイの複写環境では、プライマリ・データベースの変更がウォーム・スタンバイ・データベースにそのまま直接コピーされます。

この方法によって、DDL が複写できます。送信するデータを変更する、またはデータに条件を設定するには、テーブル複写定義およびファンクション複写定義を追加する必要があります。各プライマリ・データベースは、スタンバイ・データベースを 1 つだけ持つことができます。このウォーム・スタンバイ・アプリケーションの詳細説明については、『Replication Server 管理ガイド 第 2 巻』の「ウォーム・スタンバイ・アプリケーションの管理」を参照してください。

MSA は、『Replication Server 管理ガイド 第 2 巻』の「ウォーム・スタンバイ・アプリケーションの管理」に記載されているウォーム・スタンバイ・アプリケーションのすべての機能を備えています。さらに、MSA は以下の機能を提供します。

- 複数のスタンバイ・データベースへの複写
- 特定のデータベース・オブジェクトを複写するかどうかの選択

MSA での DDL の双方向複写のサポート
MSA (Multi-Site Availability) を設定することによって、2 つの Adaptive Server データベース間に DDL トランザクションの双方向複写を設定できます。

Replication Server 15.0 以降では、**dsi_replication_ddl** という Replication Server 設定パラメータを使用して、この双方向複写をサポートします。**dsi_replication_ddl** が on に設定されている場合、DSI はレプリケート・データベースに set replication off を送信し、システム・ログの後続の DDL トランザクションが複写されないようにマーク付けするようレプリケート・データベースに指示します。この結果、これらの DDL トランザクションが元のデータベースに複写されなくなるため、双方向 MSA 複写環境での DDL トランザクションの複写が可能になります。

MSA 混合バージョン環境
MSA 混合環境では、プライマリ Replication Server は上位のバージョンのデータ機能をフィルタします。

互換性のないコマンドは、スタンバイ Replication Server に送信されません。互換性のないパラメータがある場合、設定パラメータ **dist_stop_unsupported_cmd** によって DIST はサスペンドされます。このパラメータは、次の構文のいずれかを使用して設定できます。

- ```
  configure replication server
    set 'dist_stop_unsupported_cmd' to ['on' | 'off']
  ```
- ```
  alter connection srv.db
    set 'dist_stop_unsupported_cmd' to ['on' | 'off']
  ```

- ```
  alter logical connection lsrv.ldb
    set 'dist_stop_unsupported_cmd' to ['on' | 'off']
  ```

デフォルトでは、**dist_stop_unsupported_cmd** は off になっています。このパラメータが on の場合、コマンドを送信先に送信できないと、DIST は自動的にサスペンドします。トランザクション全体を省略して DIST をレジュームするか、パラメータを off に再設定してください。

注意： バージョン 15.0 のプライマリ Replication Server からデータベース・サブスクリプションを作成するには、レプリケート Replication Server のバージョンが 15.0 以降である必要があります。

参照：
- 複写テーブルの管理 (299 ページ)
- 複写ファンクションの管理 (385 ページ)

MSA での DDL の双方向複写のサポートの設定

MSA での双方向複写のサポートの設定について説明します。

1. 双方向 MSA 複写環境を作成します。
2. 次の例に示すように、送信先データベースのメンテナンス・ユーザに「set session authorization」権限を付与します。

   ```
   grant set session authorization to maint_user
   ```
3. 次の例に示すように、**dsi_replication_ddl** 設定パラメータを “on” に設定するように送信先データベースのコネクションを変更し、双方向 DDL 複写を有効にします。

   ```
   alter connection to dataserver.database set dsi_replication_ddl on
   ```
4. DDL トランザクションを複写します。

参照：
- 双方向複写環境の作成 (73 ページ)

MSA システムの設定

MSA 複写は、さまざまな方法で設定できます。

代表的な MSA 複写アーキテクチャの設定方法は次のとおりです。
   ```

- デフォルトのシングルパス・レプリケーションまたはマルチパス・レプリケーションを使用するシンプルなデータベース全体の複写。
- 指定したテーブルとファンクションの複写
- 複数のレプリケート・データベースへの複写

DDL やシステム・ストアド・プロシージャを複写する場合、アーキテクチャを設定するときに簡単に構文を追加できます。

参照：
- DDL とシステム・プロシージャの複写 (492 ページ)

簡単なシナリオでのデータベースの複写

この単純なシナリオでは、データベース複写定義とサブスクリプションを使用して、プライマリ・データベース全体を 1 つ以上のレプリケート・データベースに複写します。

マルチパス・レプリケーションを設定するには、『Replication Server 管理ガイド第 2 巻』の「パフォーマンス・チューニング」の「Multi-Path Replication」にある「MSA 環境での複数のレプリケーション・パスの作成」を参照してください。

この例に構文を追加することで、簡単に DDL やシステム・ストアド・プロシージャを複写できます。

1. **sp_reptostandby** を使用して、プライマリ・データベースを複写するようマーク付けします。次に例を示します。

   ```
   sp_reptostandby primary_db, 'all'
   ```

   注意： sp_reptostandby は、ユーザ・ストアド・プロシージャには複写対象のマーク付けをしません。各ユーザ・ストアド・プロシージャには、個別に **sp_setrepproc** を使用して、マーク付けする必要があります。

   ASE 以外のデータ・サーバについては、『Replication Server 異機種間複写ガイド』を参照してください。
2. RepAgent パラメータ **send warm standby xacts** を true に設定し、RepAgent がシステム・トランザクションと DDL をスタンバイ・データベースとレプリケート・データベースの両方に送るようにします。たとえば、プライマリ・データ・サーバで次のように入力します。

   ```
   sp_config_rep_agent primary_db,
           'send warm standby xacts', 'true'
   ```

   ASE 以外のデータ・サーバについては、『Replication Server 異機種間複写ガイド』を参照してください。
3. プライマリ Replication Server で **create database replication definition** を使用してデータベース複写定義を作成します。次に例を示します。

```
create database replication definition repdef_1
  with primary at PDS.primary_db
```

完全な構文と使用方法の詳細については、『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**create database replication definition**」を参照してください。

4. サブスクリプションを作成するデータベースごとにデーベース・サブスクリプションを作成します。この例では、**create subscription** と非マテリアライゼーション・メソッドを使用して、データベース・サブスクリプションを作成しています。プライマリ・データベースとレプリケート・データベースはサブスクリプションより前に同期されています。**create subscription** は、プライマリ・データベースのアクティビティをサスペンドできる場合にも使用できます。

   たとえば、レプリケート Replication Server で次のように入力します。

```
create subscription sub_1
  for database replication definition repdef_1
    with primary at PDS.primary_db
    with replicate at RDS.rdb
  without materialization
  subscribe to truncate table
```

   データベース・サブスクリプションの作成時には、非マテリアライゼーション・メソッド (手順 4 を参照) を使用することも、バルク・マテリアライゼーション・メソッドを使用してデータベースを同期させることもできます。使用する手順は、どちらのマテリアライゼーション・メソッドを選ぶか、プライマリ・テーブルのアクティビティがサスペンド可能かどうかによって異なります。

参照：

- マテリアライゼーション (486 ページ)
- DDL とシステム・プロシージャの複写 (492 ページ)

テーブルとファンクションの複写

MSA の機能を使用して、特定のテーブルまたはファンクションを複写します。

この例に構文を追加することで、簡単に DDL やシステム・ストアド・プロシージャを複写できます。

1. 複写するテーブル、ストアド・プロシージャ、データベースにマーク付けし、データベース複写定義を作成します。

   この例では、`table1` と `table2` のみを複写します。次の 2 つのうち、いずれかの方法でテーブルを指定できます。

   - **sp_reptostandby** を使用して、データベースを複写するようマーク付けします。次に、データベース複写定義を作成し、複写するテーブルを **create**

replication definition を使用して指定します。さらに、レプリケート・データをレプリケート・データベースとスタンバイ・データベースに送るように、RepAgent に指示する必要があります。

プライマリ・データ・サービスで、次を入力します。

```
sp_reptostandby primary_db, 'all'
sp_config_rep_agent primary_db,
  'send warm standby xacts', 'true'
```

プライマリ Replication Server で、次を入力します。

```
create database replication definition rep_1B
  with primary at PDS.pdb
    replicate tables in (table1, table2)
```

- または **sp_setreptable** および **sp_setrepproc** を使用して、特定のテーブルとストアド・プロシージャに複写するようマーク付けします。次に、データベース複写定義を作成します。次に例を示します。

```
sp_setreptable table1, 'true'
```

```
sp_setrptable table2, 'true'
```

```
create database replication definition rep_1A
  with primary at PDS.pdb
```

注意： DDL の変更を複写できるのは、**sp_reptostandby** を使用してそのデータベースを複写するようマーク付けした場合に限られます。

ASE 以外のデータ・サーバについては、『Replication Server 異機種間複写ガイド』を参照してください。

2. データベース・サブスクリプションを作成します。マテリアライゼーションなしでサブスクリプションを作成するには、単純なシナリオのプロシージャに従って、データベースを複写します。バルク・マテリアライゼーションを使用してサブスクリプションを作成することもできます。

注意： **sp_reptostandby** を使用してデータベースにマーク付けしてから、データベース複写定義を作成せずにテーブル複写定義とサブスクリプションを作成することもできます。この方法では個々のテーブルにマーク付けする必要がなく、しかも一部のテーブルを選択して複写できます。この場合、データベース・コネクション・パラメータ **rep_as_standby** を必ず **on** に設定してください。

暗号化カラムを処理するときの考慮事項に注意してください。

参照：

- 暗号化カラムの複写 (345 ページ)
- マテリアライゼーション (486 ページ)
- 簡単なシナリオでのデータベースの複写 (466 ページ)
- DDL とシステム・プロシージャの複写 (492 ページ)

レプリケート・データベースをウォーム・スタンバイ・データベースとして使用

MSA を使用して、DDL などのデータベース・オブジェクトを複数のレプリケート・データベースまたはウォーム・スタンバイ・データベースに複写します。

論理コネクションに対してデータベース複写定義およびデータベース・サブスクリプションを作成できます。論理コネクションの設定の詳細については、『Replication Server 管理ガイド 第 2 巻』の「ウォーム・スタンバイ・アプリケーションの管理」を参照してください。

複数のウォーム・スタンバイ・アーキテクチャの基本設定の例では、1 つのプライマリ・データベース (`dsA.db`) から 2 つのレプリケート・データベース (`dsB.db` および `dsC.db`) に複写します。単一の Replication Server が複写を制御しており、プライマリ・データベースとの間でスタンバイ複写のみが実行されます。`dsA` のみが、DDL およびシステム・ストアド・プロシージャを複写できます。ユーザが `dsB.db` または `dsC.db` に切り替わると、DDL やシステム・ストアド・プロシージャは複写されません。

この例に構文を追加することで、簡単に DDL やシステム・ストアド・プロシージャを複写できます。

注意： この例では、送信先サイトごとに異なるデータベース複写定義を使用します。両方のスタンバイ・データベースに共通する複写テーブルと複写ファンクションのセットを処理するデータベース複写定義を 1 つ作成してから、共通しないテーブルとファンクション用のテーブル・サブスクリプションとファンクション・サブスクリプションを作成することもできます。

1. データベースのアクティビティをすべてサスペンドします。
2. **sp_reptostandby** を使用して、`dsA.db`、`dsB.db`、および `dsC.db` を複写するようマーク付けします。
3. 各データ・サーバで、それぞれの RepAgent の **send warm standby xacts** を true に設定します。次に例を示します。

```
sp_config_rep_agent dbname,
    'send warm standby xacts', 'true'
```

4. Replication Server で各コネクションの **dsi_replication** を off に設定します。次に例を示します。

```
alter connection to dsB.db
        set dsi_replication 'off'
```

注意： ウォーム・スタンバイ・コネクションでは、切り替えが発生した場合にトランザクション・ログ内の複写データが再度複写されないように、

dsi_replication を off に設定することをおすすめします。通常の複写では、**dsi_replication** をデフォルトの on に設定しておいてください。

5. データベースごとにデータベース複写定義を作成します。それぞれのデータベースをプライマリとして定義します。次に例を示します。

```
create database replication definition rep_2
  with primary at dsA.db
  replicate DDL

  replicate system procedures

create database replication definition rep_2
  with primary at dsB.db

create database replication definition rep_2
  with primary at dsC.db
```

6. 各データベースがプライマリ・データベースの場合もスタンバイ・データベースの場合もあるので、各データベースが他のすべてのデータベースのサブスクリプションを作成するように、サブスクリプションを作成または定義します。マテリアライゼーション・メソッドは、サブスクリプションごとに異なるものを使用できます。次に例を示します。

```
create subscription sub_2B
  for database replication definition rep_2
    with primary at dsB.db
  with replicate at dsA.db
  without materialization
  subscribe to truncate table

create subscription sub_2C
  for database replication definition rep_2
    with primary at dsC.db
  with replicate at dsA.db
  without materialization
  subscribe to truncate table

define subscription sub_2A
  for database replication definition rep_2
    with primary at dsA.db
  with replicate at dsB.db
  subscribe to truncate table
  use dump marker

create subscription sub_2C
  for database replication definition rep_2
    with primary at dsC.db
  with replicate at dsB.db
  without materialization
  subscribe to truncate table

define subscription sub_2A
  for database replication definition rep_2
    with primary at dsA.db
  with replicate at dsC.db
```

```
  subscribe to truncate table
  use dump marker
```

```
create subscription sub_2B
  for database replication definition rep_2
    with primary at dsB.db
  with replicate at dsC.db
  without materialization
  subscribe to truncate table
```

7. dsA.db をダンプします。
8. dsB.db の DSI をサスペンドした状態で、データベースを dsB.db にロードします。
9. dsB.db とのコネクションをレジュームします。
10. dsC.db の DSI をサスペンドした状態で、データベースを dsC.db にロードします。
11. dsC.db とのコネクションをレジュームします。
12. データベースのアクティビティをレジュームします。

参照：

- DDL とシステム・プロシージャの複写 (492 ページ)

切り替え

どのスタンバイ状態でも、切り替えには、アクティブ・データベースからのユーザの切断と別のアクティブ・データベースへの再接続が伴います。この場合、切り替えを行うときには、トランザクションが失われないように、キューが空になるまで待つ必要があります。

論理コネクションと切り替えの詳細については、『Replication Server 管理ガイド第 2 巻』の「ウォーム・スタンバイ・アプリケーションの管理」を参照してください。

複写対象データへのマーク付け

sp_reptostandby、**sp_setreptable**、および **sp_setrepproc** を使用して、複写対象のデータベース、テーブル、ファンクションにマーク付けできます。

注意： ASE 以外のデータ・サーバについては、『Replication Server 異機種間複写ガイド』を参照してください。

sp_reptostandby を使用してデータベースにマーク付けする場合、次の点に注意してください。

- RepAgent 設定パラメータ **send warm standby xacts** を true に設定する必要があります。
- ユーザ定義のストアド・プロシージャは、**sp_setrepproc** を使用して個別にマーク付けしないかぎり複写されません。
- RepAgent は Replication Server に DDL、システム・プロシージャ、トランザクションを送信します。これらを Replication Server のデータベース複写定義でフィルタできます。
- テーブル複写定義とテーブル・サブスクリプションを使用し、データベース・コネクション・パラメータ **rep_as_standby** を on に設定すると、テーブル・データをレプリケート・データベースとウォーム・スタンバイ・データベースの両方に送信できます。

データベースが **sp_reptostandby** でマーク付けされていないと、マーク付けしたテーブルとファンクションの DDL は複写されません。複写対象のデータへのマーク付け用の異なるシステム・プロシージャを使用してデータを複写する方法の概要については、データ複写テーブルを参照してください。

テーブルは、データがどのように複写されるかをまとめたものです。

表 33 : データの複写

データのマーク付け用システム・プロシージャ	テーブルとファンクションのサブスクリプションのみ	データベース・サブスクリプションのみ	テーブル、ファンクション、データベースのサブスクリプションが共存
sp_setreptable と **sp_setrepproc**	• マーク付けされたデータを複写する • DLL を複写しない	• マーク付けされたデータを複写する • DLL を複写しない	• マーク付けされたデータを複写する • DLL を複写しない
sp_reptostandby	• **rep_as_standby** をチェックする • DLL を複写しない	• 全データを複写する • DLL を複写する (オプション)	• **rep_as_standby** をチェックする • DLL を複写する (オプション)
sp_setreptable、**sp_setrepproc**、**sp_reptostandby**	• **rep_as_standby** をチェックする • DLL を複写しない	• 全データを複写する • DLL を複写する (オプション)	• **rep_as_standby** をチェックする • DLL を複写する (オプション)

データベース複写定義の管理

テーブル、ファンクション、トランザクション、DDL、システム・ストアド・プロシージャは、複写オブジェクトになることができます。データベース複写定義では、複写オブジェクトにフィルタを指定して、同じ複写オブジェクトまたは複写オブジェクト全体を複写に含めたり、複写から除外したりできます。

たとえば、データベース複写定義を作成するには、次のようにします。

```
create database replication definition rep_1C
  with primary at PDS.pdb
    replicate tables in (table1, table2)
    not replicate functions in (fc_a)
    not replicate system procedures
    replicate transactions
    replicate DDL
```

この例では、次のオブジェクトを複写しています。

- `table1` と `table2`
- **fc_a** を除くすべてのファンクション
- すべてのトランザクション
- サポートされている DDL コマンド

また、この例では、次のオブジェクトは複写しません。

- `table1` と `table2` を除くすべてのデータベース・テーブル
- ファンクション **fc_a**
- すべてのシステム・プロシージャ

完全な構文と使用法の詳細については、『Replication Server リファレンス・マニュアル』の「**create database replication definition**」を参照してください。

注意： データベース複写定義では、**send-standby-all-columns**、**send-standby-all-parameters**、および **send_standby_repdef_cols** の各オプションをサポートしていません。データベース複写定義が存在する場合、Replication Server はすべてのカラムまたはパラメータを送信します。

データベース複写定義の変更

alter database replication definition を使用して、データベース複写定義を変更します。

alter database replication definition では、1 度に 1 つのフィルタを置換できます。次に例を示します。

```
alter database replication definition rep_1C
  with primary at PDS.pdb
```

```
not replicate tables in (table2)
with dsi_suspended
```

構文と使用法の詳細については、『Replication Server リファレンス・マニュアル』を参照してください。

alter database replication definition を実行すると、Replication Server はインバウンド・キューに **rs_marker** を 1 つ書き込みます。このコマンドは、マーカがディストリビュータ (DIST: Distributor) に到達してはじめて有効になります。その時点までに、ディストリビュータはデータベース・サブスクリプション・レゾルーション・エンジン (DSRE: Database Subscription Resolution Engine) を再構築して変更内容の組み込みを完了しています。

サブスクリプションが関連付けられているデータベース複写定義を変更すると、レプリケート・テーブルが同期しなくなる可能性があります。再度同期させるには、次のいずれかを実行します。

- Replication Server をクワイス状態にし、トランザクション・ログを排出して手動で変更内容を適用する。
- **with_dsi_suspended** オプションを使用する。このオプションを指定すると、レプリケート Replication Server は、“alter database replication definition” マーカを読み取るとレプリケート DSI をサスペンドする。

データベース複写定義を変更し、レプリケート・テーブルを再同期させるには、次を実行します。

1. **alter database replication definition** を実行し、**with dsi_suspended** 句を追加します。
2. レプリケート DSI がサスペンドするまで待ちます。
3. バルク・マテリアライゼーションを使用してレプリケート・テーブルを再同期させます。
4. コネクションをレジュームします。

データベース複写定義の削除

データベース複写定義は削除できますが、その前に、関連付けられているサブスクリプションをすべて削除しておく必要があります。

構文と使用法については、『Replication Server リファレンス・マニュアル』の「**drop database replication definition**」を参照してください。

データベース複写フィルタ

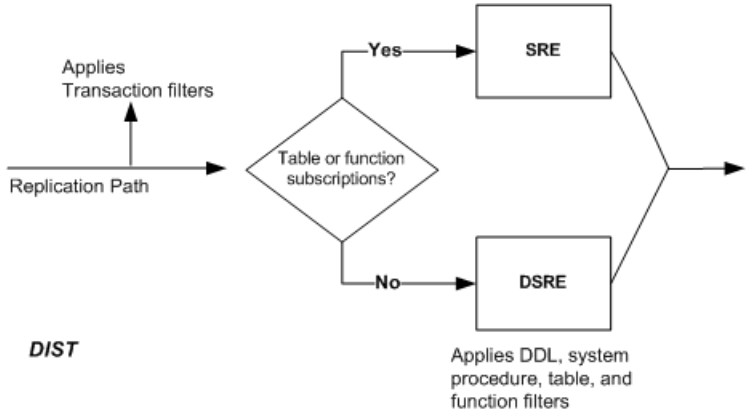
複写フィルタについて説明します。

サブスクリプション・レゾルーション・エンジン (SRE) は、テーブル・サブスクリプションおよびファンクション・サブスクリプションのローを評価します。

データベース・サブスクリプション・レゾルーション・エンジン (DSRE) は、トランザクション以外のデータベース・オブジェクトを評価します。データベース複写定義によってトランザクションがレプリケート Replication Server に送信されると、DSRE が他のデータベース・オブジェクトを評価したり、SRE がトランザクションのローを評価したりする前に、DIST がそのトランザクションを評価します。このため、Replication Server はデータベース・サブスクリプション、テーブル・サブスクリプション、またはファンクション・サブスクリプションに適合するデータが含まれている場合でもトランザクションをフィルタします。

同一のテーブルまたはファンクションに対してデータベース・サブスクリプションとテーブル・サブスクリプションまたはファンクション・サブスクリプションが共存する場合、テーブル・サブスクリプションまたはファンクション・サブスクリプションの方が優先されます。この場合、DIST は評価する複写テーブルまたは複写ファンクションを DSRE には渡さず、SRE に渡します。

図 20 : データベース複写フィルタの評価



データベース複写定義に関する情報の表示

rs_helpdbrep を使用して、データベースまたはデータ・サーバの特定のデータベース複写定義またはすべてのデータベース複写定義に関する情報を表示します。

たとえば、**rep_1B** データベース複写定義に関する情報を表示するには、次のように入力します。

```
rs_helpdbrep rep_1B, PDS, pdb
```

構文や使用方法については、『Replication Server リファレンス・マニュアル』の「RSSD ストアド・プロシージャ」の「**rs_helpdbrep**」を参照してください。

データベース複写定義、テーブル複写定義、ファンクション複写定義の併用

データベース複写定義を使用する場合は、テーブル複写定義やファンクション複写定義を追加する必要はありません。ただし、データの転送に、テーブルにターゲット・コネクションとは異なる **replicate minimal columns** または動的 SQL 設定が必要な場合は、複写定義を追加する必要があります。

send standby 句を含むテーブル複写定義またはファンクション複写定義を作成して使用すると、次のことができます。

- 複写テーブルまたは複写ファンクションの名前の変更
- 複写カラムの名前の変更
- 異なるカラムのデータ型のパブリッシュ
- より少ない数のカラムまたはパラメータの複写
- 最少カラムのみの複写
- 動的 SQL の複写なし

send standby 句がオプションであるテーブル複写定義またはファンクション複写定義を作成して使用すると、次のことができます。

- 異なるテーブル・カラムのデータ型の宣言
- マテリアライゼーションまたはマテリアライゼーション解除のためのオートコレクションの設定
- データベース複写より優先させるためのテーブル・サブスクリプションまたはファンクション・サブスクリプションの使用
- **dynamic_sql** 設定で結合したスタンバイ・データベースでの動的 SQL の除外

注意： データベース複写定義とデータベース・サブスクリプションがあるかぎり、テーブル・サブスクリプションやファンクション・サブスクリプションなしでテーブル複写定義またはファンクション複写定義を使用できます。テーブル・サブスクリプションまたはファンクション・サブスクリプションを使用する必要があるのは、それらのサブスクリプションの機能を利用する場合のみです。

プライマリ側にはデータベース複写定義とテーブル複写定義があり、レプリケート側にはデータベース・サブスクリプションはあるもののテーブル・サブスクリプションがない場合、次のように、テーブルまたはファンクションの複写定義に **send standby replication definition columns/parameters** 句があるどうかによっても複写の動作が左右されます。

- **send standby** 句がある場合、データベース・サブスクリプションはテーブル複写定義またはファンクション複写定義に従います。したがって、テーブル複写定義のプライマリ・キー・カラムと最少のレプリケート・カラムの設定を使用

してレプリケート・データベースに複写します。データベース・サブスクリプションは、常に **send standby all columns** を **send standby replication definition columns** として扱います。

- テーブル複写定義に **send standby** 句がなく、特定のテーブルに対して別の複写定義がある場合、データベース・サブスクリプションは内部テーブル複写定義 (それらすべての複写定義を結合したもの) を使用してデータを複写します。すべてのカラムが複写され、宣言したカラムまたはデータ型にデータが変換されます。
- 所有者が *dbo* ではないテーブルのテーブル複写定義を作成する場合、所有者情報を複写するようにテーブルを個別にマーク付けし、所有者情報を複写定義に含める必要があります。

参照：

• データベース・サブスクリプション、テーブル・サブスクリプション、ファンクション・サブスクリプションの併用 (489 ページ)

データベース複写定義の変更

データベース複写定義を追加または削除しても、テーブル・サブスクリプションやファンクション・サブスクリプションには影響しません。

データベース複写定義を変更する場合、既存のデータベース複写フィルタを置き換えます。または、これが新しいカテゴリである場合は、フィルタ・カテゴリをデータベース複写フィルタに追加します。

Replication Server はインバウンド・キューにマーカを追加し、DIST がコマンドを処理できるようにします。新しいフィルタは、マーカより後にコミットされたトランザクションに適用されます。

- テーブル・サブスクリプションがある場合は、何もする必要がありません。
- テーブル・サブスクリプションがない場合は、**dsi_suspended** 句を **alter database replication definition** コマンドに追加するか、手動でテーブルのマテリアライゼーションまたはマテリアライゼーション解除を行います。

テーブルおよびファンクション複写定義を変更するには、Replication Server によって複写定義の変更とデータ複写の伝達が自動的に調整される複写定義の変更要求プロセスを使用します。

参照：

• 複写定義の変更要求プロセス (351 ページ)

複写定義およびサブスクリプションの使用の削減

Adaptive Server データベースのみを含んでいる複写システムでは、ウォーム・スタンバイ環境または Multi-Site Availability (MSA) 環境におけるテーブルやストアド・プロシージャの複写定義およびサブスクリプションの必要性を削減できます。

複写定義の目的が以下の一部またはすべてを指定することのみである場合は、プライマリ・テーブルまたはストアド・プロシージャの複写定義を作成する必要はありません。

- プライマリ・キー・カラム
- 引用符が付く可能性があるテーブル名またはカラム名。
- レプリケート・テーブルまたはストアド・プロシージャのカスタム・ファンクション文字列。

参照：
- 引用符付き識別子 (308 ページ)

プライマリ・キー・カラムと引用符付きのテーブル名またはカラム名

Adaptive Server の RepAgent は、ログ転送言語 (LTL) を使用して、引用符が付いている可能性があるテーブル名またはカラム名の指定とテーブル・カラムがテーブル・プライマリ・キーの一部であるかどうかの指定を行います。RepAgent は、プライマリ・キーと引用符付き識別子情報を Replication Server に送信します。そのため、Adaptive Server データベースしか含まれていない複写システムにおいて、プライマリ・キーと引用符付き識別子情報を指定することが複写定義の唯一の目的である場合、複写定義は必要となりません。

複写定義の必要性が低下することにより、多くのテーブルがあるデータベース、多くのカラムがあるテーブル、またはスキーマを頻繁に変更するテーブルが関係する複写環境の管理が容易になります。現在、複写定義がないテーブルのレプリケーション・パフォーマンスは向上します。これは、RepAgent が Replication Server にテーブルのプライマリ・キー情報を直接提供し、結果として Replication Server が **update**、**delete**、および **select** コマンドの **where** 句にプライマリ・キー・カラムだけをパックするからです。

- 複写定義が MSA データベースによってサブスクライブされている場合、複写定義のプライマリ・キー・カラムは、そのデータベースの **where** 句をパックする目的で使用されます。
- 複写定義に **send standby replication definition columns** または **send standby all columns** のマークが付いている場合、この複写定義のプライマリ・キー・カラムは、同じテーブルのいずれの複写定義もサブスクライブしていないスタンバ

イ・データベースおよび MSA データベースの **where** 句をパックする目的で使用されます。
- あるテーブルの複写定義が 1 つまたは複数あり、そのいずれにも **send standby** マークが付いていない場合は、そのテーブルのいずれの複写定義もサブスクライブしていないスタンバイ・データベースおよび MSA データベースの **where** 句をパックする目的で、複写定義のプライマリ・キー・カラムを統合したものが使用されます。
- テーブルの複写定義がなく、プライマリ・キーに関する情報が LTL によって送信される場合は、LTL で示されるプライマリ・キー・カラムが **where** 句をパックする目的で使用されます。
- テーブルの複写定義がなく、どのプライマリ・カラムも LTL コマンドによって示されない場合は、`text`、`image`、`unitext` の各カラムを除くすべてのカラムが **where** 句をパックする目的で使用されます。

『Replication Server 管理ガイド 第 2 巻』の「ウォーム・スタンバイ・アプリケーションの管理」を参照してください。

引用符付きテーブル名とカラム名および複写定義
Replication Server がターゲット・データベースの DML コマンドをパックするときに、テーブルの複写定義が 1 つまたは複数あり、次の条件に当てはまる場合は、テーブル名またはカラム名に引用符を付けるべきかどうかが複写定義に基づいて決定されます。

- MSA データベースによってサブスクライブされている。
- 複写定義をサブスクライブしていない MSA データベースに対して **send standby [replication definition | all] columns** マークが付けられている。
- スタンバイ・データベースに対して **send standby replication definition columns** マークが付けられている。

それ以外の場合、Replication Server は LTL に基づいてテーブル名またはカラム名をパックします。

システムの稼働条件
RepAgent は、LTL バージョン 740 以降でのみプライマリ・キー情報および引用符付き識別子情報を送信します。これは、Adaptive Server 15.7 以降および Replication Server 15.7 以降でサポートされています。

ターゲットスコープ・カスタム・ファンクション文字列

Adaptive Server データベースだけがあるレプリケーション・システムでは、複写定義の目的がレプリケート・テーブルやスタンバイ・テーブルまたはストアド・プロシージャのカスタム・ファンクション文字列を指定することだけである場合、ウォーム・スタンバイ環境または Multi-Site Availability (MSA) 環境におけるプライ

マリ・テーブルまたはストアド・プロシージャの複写定義を作成する必要はありません。

ターゲットスコープ・ファンクション文字列と呼ばれるカスタム・ファンクション文字列は、レプリケート・テーブルやスタンバイ・テーブルまたはストアド・プロシージャに対して直接作成できます。この際、複写定義を定義する必要はありません。これにより、ウォーム・スタンバイ環境または MSA 環境における複写定義要件がさらに緩和されます。ターゲットスコープ・ファンクション文字列の作成には **create function string** を使用し、ファンクション文字列の管理には **alter function string** と **drop function string** を使用します。

ストアド・プロシージャとシステム・テーブルのサポート
ターゲットスコープ・ファンクション文字列に関する情報を表示するには、**rs_helpobjfstring** を使用します。詳細については、『Replication Server リファレンス・マニュアル』の「RSSD ストアド・プロシージャ」で「**rs_helpobjfstring**」を参照してください。

`rs_targetobjs` システム・テーブルには、ターゲットのテーブルまたはストアド・プロシージャに関する情報が格納されます。『Replication Server リファレンス・マニュアル』の「Replication Server システム・テーブル」の「`rs_targetobjs`」を参照してください。

Replication Server では、`rs_targetobjs` の値を他の Replictation Server の RSSD にレプリケートしません。`rs_targetobjs` はシステム・テーブル・サービス (STS) プライマリ・キャッシュ・キーとして STS キャッシュ (`objname`、`objowner`、`dbid`、`objtype`) 内にあります。**sts_full_cache_rs_targetobjs** を使用して、テーブルのフル・キャッシュの有効または無効を切り替えます。

```
configure replication server set sts_full_cache_rs_targetobjs to
{on|off}
```

sts_full_cache_rs_targetobjs のデフォルト値は off です。

ウォーム・スタンバイ環境と Muliti-Site Availability 環境
ウォーム・スタンバイ・システムまたは Muliti-Site Availability (MSA) システムがターゲットスコープ・ファンクション文字列を使用できるようになるには、いくつかの条件が満たされていなければなりません。

- MSA システムがテーブルまたはストアド・プロシージャの複写定義をサブスクライブしている場合は、この複写定義のファンクション文字列が Replication Server によって使用されます。このファンクション文字列は、デフォルト・ファンクション文字列とカスタム・ファンクション文字列のいずれの場合もあります。
- MSA システムがテーブルまたはストアド・プロシージャのいずれの複写定義もサブスクライブしていない場合は、ターゲット・オブジェクトの複写定義が

send standby all 句または **send standby replication definition columns** 句でマーク付けされていれば、この複写定義のファンクション文字列が Replication Server によって使用されます。
- ウォーム・スタンバイ・システムでは、複写定義に **send standby replication definition columns** 句でマーク付けした場合に、この複写定義のファンクション文字列が使用されます。

これらの条件がいずれも満たされていない場合で、レプリケート・テーブルやスタンバイ・テーブルまたはストアド・プロシージャのターゲットスコープ・ファンクション文字列があるときは、この文字列がウォーム・スタンバイ・システムまたは MSA システムで使用されます。

ターゲットスコープおよび複写定義スコープのファンクション文字列の比較
ターゲットスコープおよび複写定義スコープのファンクション文字列の間にはいくつか異なる点があります。

複写定義スコープ	ターゲットスコープ
複写定義スコープ・ファンクション文字列は、ファンクション文字列クラスに関連付けられている。	ターゲットスコープ・ファンクション文字列は、ターゲット (スタンバイまたはレプリケート) データベースに関連付けられている。
複写定義スコープのファンクション文字列を、複写定義を管理する Replication Server のファンクション文字列クラスの複写定義に対して作成する。	ターゲット・テーブルまたはストアド・プロシージャについて、スタンバイ・データベースまたはレプリケート・データベースを管理する Replication Servers のターゲットスコープ・ファンクション文字列を作成する。
Replication Server は、ファンクション文字列を複写定義に対して作成または変更したときに、ファンクション文字列のカラムまたはパラメータが有効かどうかを確認する。	Replication Server は、ファンクション文字列テンプレートを使用して DML コマンドを変換し、コマンドをスタンバイ・データベースまたはレプリケート・データベースに適用するまで、ファンクション文字列のカラムまたはパラメータが有効かどうかを確認しない。カラムまたはパラメータに関する情報が不正な場合は、DSI コネクションはシャットダウンする。ファンクション文字列を訂正してから、DSI コネクションを再開する。

複写定義スコープ	ターゲットスコープ
複写定義スコープのファンクション文字列を作成する場合、Replication Server は一連のデフォルトのファンクション文字列を複写定義に対して作成する。	ターゲットスコープ・ファンクション文字列を作成する場合、Replication Server は指定するファンクション文字列のみを作成する。 ただし、**rs_get_textptr** と **rs_writetext** ファンクション文字列は共存するため例外である。**rs_writetext** のカスタム・ファンクション文字列がある場合は、**rs_get_textptr** のファンクション文字列も存在する。共に、カスタム・ファンクション文字列またはシステム作成のデフォルトのファンクション文字列として存在できる。

ターゲットスコープ・ファンクション文字列のコマンド

レプリケート・テーブルとスタンバイ・テーブルおよびストアド・プロシージャのターゲットスコープ・ファンクション文字列を作成および管理するには、**create function string**、**alter function string**、および **drop function string** を使用します。

ターゲットスコープ・ファンクション文字列を作成および管理するには、*[owner.]table*、*stored procedure*、*data_server.database* の各パラメータを使用します。

```
{create | alter | drop} function string
        {replication_definition |
        [owner.] table |
         stored_procedure}.function[; function_string]

        for {[function class]function_class |
        [database] data_server.database}...
```

例

NY_DS データ・サーバ内の rdb1 ターゲット・データベースに dbo.authors テーブルのカスタム・ファンクション文字列 **rs_insert** を作成するには、次のように入力します。

```
create function string dbo.authors.rs_insert
     for database NY_DS.rdb1
     output language
     'insert authors values (
           ?au_id!new? ,
           ?au_lname!new? ,
           ?au_fname!new? ,
           ?phone!new? ,
           ?address!new? ,
           ?city!new? ,
           ?state!new? ,
           "00000" ,
```

```
            ?contract!new?)
      update fn_monitor set insert_count = insert_count + 1'
```

完全な構文、パラメータの説明、その他の例、および使用方法については、『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」で「**create function string**」、「**alter function string**、および「**drop function string**」を参照してください。

ターゲットスコープのファンクション文字列のリスト

Adaptive Server や Oracle など、ストアド・プロシージャをサポートしているデータベースのターゲットスコープ・ファンクション文字列をリストするには、**rs_helpobjfstring** ストアド・プロシージャを使用します。

レプリケート・テーブルやスタンバイ・テーブルまたはストアド・プロシージャのカスタム・ファンクション文字列をリストするには、次のように入力します。

```
rs_helpobjfstring data_server, database, [owner.]object_name[,
function_name]
```

upd_datetime ストアド・プロシージャのターゲットスコープ・ファンクション文字列を作成するとします。

```
create function string upd_datetime.upd_datetime
     for database NY_DS.rdb1
     with overwrite
          output language
          'update datetime set
          row_num = ?row_num!param?,
          datecol = ?datecol!param?,
          timecol = ?timecol!param?,
          ndatecol = ?ndatecol!param?,
          ntimecol = ?ntimecol!param?,
          comment = ?comment!param?
          where
          row_num = ?row_num!param?'
```

次のように入力します。

- `rs_helpobjfstring NY_DS,rdb1,upd_datetime`

  または

- `rs_helpobjfstring NY_DS,rdb1,upd_datetime,upd_datetime`

次のようなメッセージが表示されます。

```
Function String information for Target Object: 'upd_datetime'.

Object Name     Object Type         Function Name
-----------     -----------         --------------
upd_datetime    stored procedure    upd_datetime

Function String Name    Output Type Option        System Generated
--------------------    -------------------       -----------------
upd_datetime            language not applicable   no
```

```
        --- Beginning of Function String Text ---

FString Text
-------------------------------------------------------------------
 update datetime set
        row_num = ?row_num!param?,
        datecol = ?datecol!param?,
        timecol = ?timecol!param?,
        ndatecol = ?ndatecol!param?,
        ntimecol = ?ntimecol!param?,
        comment = ?comment!param?
        where
        row_num = ?row_num!param?

        --- End of Function String Text ---


(return status = 0)
```

その他の例と使用方法については、『Replication Server リファレンス・マニュアル』の「RSSD ストアド・プロシージャ」で「**rs_helpobjfstring**」を参照してください。

複写定義を削減するための複写システムの設定

Adaptive Server ウォーム・スタンバイ環境または Multi-Site Availability (MSA) 環境から複写定義を削減するように複写システムを設定します。

コマンドとシステム・プロシージャの完全な構文については、『Replication Server リファレンス・マニュアル』を参照してください。

1. 次のように、プライマリ Replication Server でログ転送をサスペンドします。
   ```
   suspend log transfer from all
   ```
2. 次のように、プライマリ・データ・サーバ上で RepAgent を停止します。
   ```
   sp_stop_rep_agent dbname
   ```
3. プライマリ Replication Server およびレプリケート Replication Server で、必要に応じて、サイト・バージョンをアップグレード済みの Replication Server バージョンに設定します。新しいサイト・バージョンは 1571 以降にする必要があります。
   ```
   sysadmin site_version, new site version
   ```
4. プライマリー・キー、引用符付きの識別子、およびカスタマイズされたファンクション文字列を定義するためにのみ存在するすべての複写定義を削除します。
   a) Replication Server の RSSD または ERSSD では、**rs_helpcheckrepdef** を使用して、このような複写定義を識別します。

b) プライマリ Replication Server で、**drop replication definition** を使用して、**rs_helpcheckrepdef** によってリストされた複写定義を削除します。

5. プライマリ・データベースで、プライマリ・キーまたはユニーク・インデックスを含み、複写定義を含まないテーブル用のユニーク・インデックスを作成します。

   RepAgent は、以下をプライマリ・キーとして選択します。

   - テーブル上で定義されたプライマリ・キー
   - プライマリ・インデックスが存在しない場合、最下位インデックス識別子を持つユニークな識別子

   注意： テーブルで定義されたプライマリ・キーまたはユニーク・インデックスが存在しない場合、RepAgent はプライマリ・キー情報を LTL に送信しません。テーブルに複写定義が含まれていない場合、Replication Server は、`text`、`image`、または `unitext` カラム以外のすべてのカラムをプライマリ・キー・カラムとして処理します。この結果、テーブルのレプリケーション・パフォーマンスが低下する場合があります。

   a) パラメータを指定せずに **sp_setreptable** を実行し、**sp_setreptable** で複写するようにマーク付けしたすべてのテーブルのプライマリ・キーとインデックスの情報を確認します。

   ```
   Name       Repdef Mode   Index Mode   Primary Key
   ---------  ------------  -----------  -------------------
   hola       owner_off     no index     hola_index
   inds       owner_off     no index     inds_1335724833
   t_applied  owner_off     no index     t_applied_855723122
   t5         owner_off     no index     con_pk_t5
   ```

   単一のテーブルを指定することもできます。次に例を示します。

   ```
   sp_setreptable inds
   ```

   b) ユニーク・インデックスまたはプライマリ・キーを指定しないでテーブル用のユニーク・インデックスを作成します。

   『Adaptive Server Enterprise Transact-SQL ユーザーズ・ガイド』の「テーブルのインデックスの作成」を参照してください。

6. 手順 4 で削除した複写定義でファンクション文字列がカスタマイズされている場合、スタンバイ・データベースまたは MSA データベースを管理する Replication Server でカスタマイズされたターゲットスコープ・ファンクションとして再作成します。複数のターゲット・データベースが存在する場合、データベースごとにファンクション文字列を再作成します。
7. ログ転送をレジュームして、RepAgent を起動します。

   a) 次のように、プライマリ・データ・サーバで RepAgent を起動します。

   ```
   sp_start_rep_agent dbname
   ```

   b) 次のように、プライマリ Replication Server でログ転送をレジュームします。

```
resume log transfer from all
```

注意： テーブル・スキーマのプライマリ・キーを変更する場合、テーブルにサブスクリプションが作成され、**dynamic_sql** が on であるデータベースからのすべてのコネクションをサスペンドしてレジュームする必要があります。

データベース・サブスクリプションの管理

プライマリ・データベースでデータベース複写定義を作成する場合は、サブスクリプションを作成する各データベースでデータベース・サブスクリプションも作成する必要があります。

非マテリアライゼーション・メソッドまたはバルク・マテリアライゼーション・メソッドが使用できます。データベース・サブスクリプションを作成する場合は、**where** 句を使用して、サブスクリプションを作成するデータの条件を設定することはできません。すべてのデータのサブスクリプションが作成されます。

特定のテーブルまたはファンクションの条件を設定する必要がある場合は、テーブル・サブスクリプションまたはファンクション・サブスクリプションを同時に追加できます。

データベース・サブスクリプションがある場合、そのコネクションの DSI は常に通常の複写の DSI と同様に扱われます。つまり、**dsi_replication** パラメータは off、**dsi_keep_triggers** パラメータは on です。

データベース・サブスクリプションとテーブル複写定義、ファンクション複写定義があり、テーブル・サブスクリプションまたはファンクション・サブスクリプションがない場合、次のようになります。

- テーブル複写定義とファンクション複写定義に **send standby** 句が含まれていれば、データベース・サブスクリプションはそのテーブル複写定義またはファンクション複写定義に従います。
- テーブル複写定義とファンクション複写定義に **send standby** 句が含まれていなければ、すべてのカラムとパラメータが複写され、宣言したカラムおよびパラメータのデータ型にデータが変換されます。

参照：

- データベース・サブスクリプション、テーブル・サブスクリプション、ファンクション・サブスクリプションの併用 (489 ページ)

マテリアライゼーション

データベース・サブスクリプションには、非マテリアライゼーション・メソッドまたはバルク・マテリアライゼーション・メソッドのいずれかが必要です。

マテリアライゼーションなしのサブスクリプション
プライマリ・データベースとレプリケート・データベースがサブスクリプションより前に同期されている場合、またはプライマリ・データベースのアクティビティをサスペンドできる場合、データベース・サブスクリプションを作成するには **create subscription** を **without materialization** 句とともに使用します。

非マテリアライゼーション・メソッドを採用する場合は、データベース複写の簡単なシナリオに説明されているように、**bcp**、**dump** と **load**、**mount** と **unmount** などのメソッドを使用してレプリケート・データベースをマテリアライズできます。Replication Server は初期データベース同期処理を調整しないので、多くの場合、データベース・アプリケーションをサスペンドする必要があります。この方法は、プラットフォーム間の **dump** と **load** (XPDL) によって複写データベースをマテリアライズする場合に使用します。手順については、『Replication Server 管理ガイド第 2 巻』の「ウォーム・スタンバイ・アプリケーションの管理」にある「ASE ウォーム・スタンバイ・データベースの設定」の「作業 4：スタンバイ・データベースの追加」の「スタンバイ・データベースを初期化する方法の決定」にある「プラットフォーム間でのダンプとロード」を参照してください。

参照：
- 簡単なシナリオでのデータベースの複写 (466 ページ)

バルク・マテリアライゼーションを使用したサブスクリプション
dump と **load** または手動の調整方法を使用して、データベースを同期させることができます。

ダンプとロードの調整を使用してデータベース・サブスクリプションを作成するには、**define subscription** を **use dump marker** 句とともに使用します。プライマリ・データベースとレプリケート・データベースの両方と Replication Server に同じサーバ・ユーザ ID、パスワード、ロールが設定されていることが必要です。

```
define subscription sub_2
  for database replication definition repdef_1
    with primary at PDS.primary_db
```

```
  with replicate at RDS.rdb
  subscribe to truncate table
  use dump marker
```

サブスクリプションの定義後のデータベースの同期
サブスクリプションを定義した後にデータベースを同期します。

1. *PDS.pdb* をダンプします。ダンプ・マーカがレプリケート Replication Server に到達すると、レプリケート・データベースとの DSI コネクションがサスペンドされます。手順 2 が完了するまでデータが複写されないようにサスペンドされ

ます。ダンプ・マーカが複写されると、Replication Server は自動的にサブスクリプションをアクティブ化し、確定化します。

警告！ activate subscription を実行しないでください。実行すると、Replication Server でダンプ・マーカの待機が無効になります。

2. *PDS.primary_pdb* を *RDS.rdb* にロードします。
3. *RDS.rdb* をオンラインにします。
   オンライン・データベース *RDS.rdb*
4. *RDS.rdb* を複写として準備します。
   a) 必要に応じて、*RDS.rdb* でセカンダリ・トランケーション・ポイントを無効にします。
   ```
   dbcc settrunc('ltm', 'ignore')
   ```
   b) 必要に応じて、RepAgent を無効にします。
   ```
   sp_config_rep_agent primary_db, 'disable'
   ```
   c) トランザクション・ログをダンプします。
5. *RDS.rdb* との DSI コネクションをレジュームします。

データベース・サブスクリプションの変更

データベース・サブスクリプションを直接変更することはできません。データベース・サブスクリプションを変更するには、**drop subscription** を使用して既存のデータベース・サブスクリプションを削除してから、新しいデータベース・サブスクリプションを作成します。

データベース・サブスクリプションの削除

drop subscription を使用してデータベース・サブスクリプションを削除します。

必ず **without purge** オプションを指定してください。このオプションを指定すると、Replication Server はそのデータベース・サブスクリプションによってレプリケートに追加されたローを削除しません。次に例を示します。

```
drop subscription sub_1a
  for database replication definition rep_1
    with primary at PDS.pdb
  with replicate at RDS.rdb

  without purge ...
```

データベース・サブスクリプションを削除しても、既存のテーブル・サブスクリプションやファンクション・サブスクリプションには影響しません。同様に、テーブル・サブスクリプションまたはファンクション・サブスクリプションを削除しても、既存のデータベース・サブスクリプションには影響しません。ただし、この場合は、レプリケート・テーブルの再マテリアライゼーションが必要になることがあります。

構文と使用法の詳細については、『Replication Server リファレンス・マニュアル』を参照してください。

データベース・サブスクリプションに関する情報の表示

rs_helpdbsub を使用すると、データベースまたはデータ・サーバの特定のデータベース・サブスクリプションまたはすべてのデータベース・サブスクリプションに関する情報を表示できます。

たとえば、**sub_2B** データベース・サブスクリプションに関する情報を表示するには、次のように入力します。

```
rs_helpdbsub sub_2B, dsA, db
```

構文や使用方法については、『Replication Server リファレンス・マニュアル』の「RSSD ストアド・プロシージャ」の「**rs_helpdbsub**」を参照してください。

データベース・サブスクリプション、テーブル・サブスクリプション、ファンクション・サブスクリプションの併用

データベース・サブスクリプションとテーブル・サブスクリプションまたはファンクション・サブスクリプションが共存している場合、データベース・サブスクリプションよりもテーブル・サブスクリプションまたはファンクション・サブスクリプションの方が優先されます。

したがって、Replication Server はデータベース・サブスクリプションではなく、テーブル・サブスクリプションまたはファンクション・サブスクリプションに従ってテーブルまたはファンクションを複写します。

データベース・サブスクリプションは、**where** 句も **for new articles** 句もサポートしていません。データベース・サブスクリプションを使用している場合は、次の目的でのみテーブル・サブスクリプションを作成できます。

- テーブル・サブスクリプションの **where** 句を使用します。
- データベース複写定義によってフィルタされたテーブルを複写します。

  注意： データベース・サブスクリプションは **subscribe to truncate table** 句をサポートしていますが、テーブル・サブスクリプションがあるテーブルは対象外です。
- テーブルのオートコレクションを実装します。

参照：

- データベース複写定義、テーブル複写定義、ファンクション複写定義の併用 (476 ページ)

サブスクリプションの作成および削除

データベース・サブスクリプションとテーブル・サブスクリプションまたはファンクション・サブスクリプションを併用している場合、サブスクリプションを作成または削除するための手順は複数あります。

- 既存のテーブル・サブスクリプションが特定のテーブルを参照するデータベース・サブスクリプションを作成する場合は、データベースを同期するときに複写テーブルを上書きしないように、次の操作を実行してください。
  1. 複写テーブルをバックアップする。
  2. **dump** と **load** を使用してレプリケート・データベースを同期させる。
  3. 複写テーブルをレプリケート・データベースにコピーし戻す。
- テーブル・サブスクリプションまたはファンクション・サブスクリプションを削除するときは、with suspension 句を使用してください。レプリケート DSI のサスペンド後に、レプリケート・テーブルまたはレプリケート・ファンクションのマテリアライゼーション解除や再同期ができます。
- テーブルを含むデータベース・サブスクリプションがある場合にテーブル・サブスクリプションを追加するには、必ずバルク・マテリアライゼーションを使用してテーブル・サブスクリプションを定義してください。テーブル・サブスクリプションを定義しても、テーブルのデータベース複写は停止しません。テーブルのデータベース複写が停止するのは、そのテーブル・サブスクリプションがアクティブになったときです。
- データベース・サブスクリプションを削除する場合は、テーブル・サブスクリプションがない複写テーブルを手動ですべて消去してください。Replication Server は複写テーブルのマテリアライゼーションを解除しません。

MSA 環境での Master データベースの複写

Adaptive Server ログインは master データベース間で複写できます。master データベースの複写は、ログインとロールの管理に使用する DDL コマンドとシステム・コマンドに制限されています。

master データベースを複写しても、システム・テーブルのデータ、master データベース内の他のユーザ・テーブルのデータやプロシージャは複写されません。

送信元 Adaptive Server と送信先 Adaptive Server では、同じハードウェア・アーキテクチャのタイプ (32 ビット・バージョンと 64 ビット・バージョンは互換性があります)、また同じオペレーティング・システム (バージョンは異なっていてもかまいません) を使用している必要があります。

データベースに適用される制約と制限、およびサポートされる DDL とシステム・プロシージャについては、『Replication Server 管理ガイド 第 2 巻』の「ウォーム・スタンバイ・アプリケーションの管理」の「ウォーム・スタンバイの複写情報」

の「**sp_reptostandby** を使用した複写の有効化」の「**sp_reptostandby** を使用する場合の制限と条件」 で、以下を参照します。

- サポートされている DDL コマンドとシステム・プロシージャ
- master データベースの複写：制限事項

Replication Server バージョン 12.0 以降で master データベースを複写する場合、ウォーム・スタンバイ、および Replication Server 12.6 以降の MSA を使用できます。プライマリ Adaptive Server またはアクティブ Adaptive Server は、バージョン 15.0 ESD #2 以降である必要があります。

ウォーム・スタンバイ環境での master データベースの複写の詳細については、『Replication Server 管理ガイド 第 2 巻』の「ウォーム・スタンバイ・アプリケーションの管理」の「ASE のウォーム・スタンバイ環境での master データベースの複写」を参照してください。

1. **rs_init** を使用して、プライマリ master データベースとレプリケート master データベースを設定します。
2. **bcp** を使用するか、各 master データベースで **syslogins** と **suids** を手動で同期させます。**dump** と **load** を使用して、レプリケート master データベースをマテリアライズしないでください。
3. 次のように、プライマリ master データベースにマークを付けます。

   ```
   sp_reptostandby master, 'all'
   ```

4. 次のように、プライマリ master データベース上で RepAgent を停止します。

   ```
   sp_stop_rep_agent master
   ```

5. ウォーム・スタンバイ・トランザクションを送信するように、レプリケート・プライマリ master データベースを設定します。

   ```
   sp_config_rep_agent master,'send warm standby
   xacts','true'
   ```

6. 次のように、プライマリ master データベース上で RepAgent を再起動します。

   ```
   sp_start_rep_agent master
   ```

7. 次のように、データベース複写定義を作成して、システム・プロシージャを複写します。

   ```
   create database replication definition master_dbrep
   with primary at PDS.master
   replicate system procedures
   ```

8. 次のように、サブスクリプションを作成する master データベースごとにデータベース・サブスクリプションを作成します。データをマテリアライズしないでください。

   ```
   create subscription master_dbsub1
   for database replication definition master_dbrep
   ```

```
with primary at PDS.master
with replicate at RDS.master
without materialization
```

DDL とシステム・プロシージャの複写

MSA を使用して DDL を非スタンバイ・データベースに複写する方法について説明します。

複写でサポートされる DDL コマンドのリストについては、『Replication Server 管理ガイド 第 2 巻』の「ウォーム・スタンバイ・アプリケーションの管理」の「ウォーム・スタンバイの複写情報」の「sp_reptostandby を使用した複写の有効化」の「サポートされている DDL コマンドとシステム・プロシージャ」を参照してください。

適用される制限は、次のとおりです。

- システム・プロシージャを複写する場合で、プライマリ・データベースとレプリケート・データベースの名前が異なっているときは、データベース複写定義で **sp_config_rep_agent** および **sp_add_user** システム・プロシージャをフィルタしてください。これらのシステム・プロシージャはデータベース名をパラメータとして使用するからです。次に例を示します。

  ```
  create database replication definition myrepdef
    with primary at PDS.pdb
    not replicate system procedures in
    (sp_config_rep_agent, sp_add_user)
  ```

- DDL を複写する場合は、プライマリ・データベースとレプリケート・データベースに同じログイン名とパスワードが必要です。DSI は、オリジナルのサーバ・ログイン名とパスワードを使用してレプリケート・データベースにログインします。
- ユーザ定義トランザクションに含まれている DDL を複写する場合は、必ず Adaptive Server のデータベース・オプション **ddl in tran** を **true** に設定してください。そうしないと、DSI が DLL の複写時に停止します。
- **set proxy** のプライマリ Adaptive Server での実行後、Replication Server は DDL コマンドの複写をサポートしません。プライマリ Adaptive Server で **set proxy** を実行すると、Replication Server はエラー 5517 を返します。

  ```
  A REQUEST transaction to database '...' failed because the
  transaction    owner's password is missing. This prevents the
  preservation of transaction ownership.
  ```

注意： 異機種環境では、Replication Agent が Transact-SQL または ANSI SQL (推奨) 内の DDL コマンドを取得して送信できる場合、Sybase 以外のデータ・サーバで DDL を複写できます。

DDL とシステム・プロシージャを複写するには、次の手順に従います。

1. **sp_reptostandby** を使用してプライマリ・データベースにマーク付けする。
2. RepAgent パラメータ **send warm standby xacts to true** を設定する (スタンバイ・データベースがない場合も含む)。
3. データベース・サブスクリプションを作成する。
4. プライマリ・データ・サーバとレプリケート・データ・サーバが両方とも同じバージョンの Adaptive Server であることを確認する。

ユーザ・ストアド・プロシージャの複写

ユーザ・ストアド・プロシージャを複写するには、**sp_setrepproc** を使用して各プロシージャに個別にマーク付けします。データベース複写定義はユーザ・ストアド・プロシージャの所有者情報をチェックしません。

カスタム・ファンクション文字列

rs_default_function_class ファンクション文字列クラスに含まれていないファンクション文字列のみカスタマイズできます。

複写定義スコープを持つファンクションの場合、次のようになります。

- DSI は、**send standby** 句があるテーブル複写定義またはファンクション複写定義を使用しないファンクションに対しては **rs_default_function_class** を使用する。
- それ以外のファンクションに対しては、コネクションと関連付けられたファンクション文字列クラスを使用する。

ファンクション文字列クラス・スコープを持つファンクションの場合、DSI はコネクションと関連付けられたファンクション文字列クラスを常に使用します。

複写スケジュールの管理

Replication Server では複写タスクのスケジュールを設定できます。Replication Server では複写を一定時間遅らせることもできます。

複写タスクのスケジュール

複写スケジュールを作成および管理する方法について説明します。

スケジュールの作成

create schedule を使用して、シェル・コマンドを指定時刻に実行するスケジュールを作成します。

たとえば、レプリケート・データベースがフリーズしてプライマリ・データベースからデータを受信していないときに、レプリケート・データベースの特定のステータスに関するレポートを作成できます。この例では、レプリケーションが夜中の指定された期間にのみ行われるようにスケジュールを設定することによって、次の日の処理でレプリケート・データベースが変更されないようにし、レプリケート・データベースでフリーズされた前の日のデータに対してレポートが行われるようにすることができます。これは、レプリケート・データベースへのコネクションが 1 日の特定の時刻にサスペンドおよびレジュームされるようにスケジュールを設定することによって行うことができます。

構文：

```
create schedule sched_name as 'sched_time' [set {on | off}] for exec
'command'
```

パラメータ：

- *sched_name* – 指定されたスケジュールの名前。以下の条件を満たす必要があります。
  - 名前は識別子の規則に従う必要がある。
  - ユニークでなければならない。
  - 長さは 1 ～ 30 バイト。
- '*sched_time*' – '*command*' を実行する時刻と日付を指定します。時刻と日付のパラメータ間にスペースを 1 つ挿入して区切る、制限された UNIX **cron** スタイルの日付と時刻を示します。

  ```
  [mm] [HH] [DOM] [MON] [DOW]
  ```

  時刻と日付の各パラメータは、以下のとおりです。

パラメータ	説明	値
mm	正時から経過した分数。	0 ～ 59。有効値をすべて含めるには、“*” を使用する。
HH	24 時間表記での時間。	0 ～ 23。有効値をすべて含めるには、“*” を使用する。
DOM	日	1 ～ 31。日をすべて含めるには、“*” を使用する。
MON	月	1 ～ 12。月をすべて含めるには、“*” を使用する。
DOW	曜日	0 ～ 6 で、0 = 日曜日。曜日をすべて含めるには、“*” を使用する。

  - 有効値をすべて指定するには、アスタリスク “*” を使用します。たとえば、“17 20 * * *” は毎日のスケジュールで午後 8:17 を意味します。
  - 範囲外の値を区切るには、カンマ “,” を使用します。たとえば、“17 20 1,15 * *” は毎月 1 日と 15 日の午後 8:17 を表します。ただし、1 と 15 は *DOM* パラメータの値です。
  - 値の範囲 (両端を含む) を指定するには、ハイフン “-” を使用します。たとえば、“17 20 * * 1-5” は月曜日から金曜日の午後 8:17 を表します。ただし、"1-5" は *DOW* パラメータの値の範囲です。
  - *DOM*、*MON*、または *DOW* パラメータの場合、*DOM* と *DOW* の両方のパラメータを使用して日を指定できます。Replication Server は、文字列で指定されたスケジュールすべてに従います。たとえば、“0 12 16 * 1” は毎週月曜日の午後 12:00 および毎月 16 日の午後 12:00 を表します。
- **set {on | off}** – スケジュール作成時に、有効と無効を指定します。デフォルトでは、スケジュールは on です。
- ‘*command*’ – スクリプト、実行可能ファイル、指定スケジュールで実行されるバッチ・ファイルなどのシェル・コマンドを指定します。
  シェル・コマンドには、以下の特徴があります。
  - UNIX では `$SYBASE/$SYBASE_REP/sched`、Windows では `%SYBASE%¥%SYBASE_REP%¥sched` に存在しなければならない。
  - シェル・コマンド内にスペースで区切ったパラメータを複数含めることができる。
  - Windows では、**create schedule** はシェル・コマンドまたはバッチ・ファイル内の最後のパラメータで指定されたコマンドを実行します。さらに、**stdout** を **create schedule** コマンド・ラインのファイルに含めてください。

  シェル・コマンド名には、以下の特徴があります。
  - ASCII 英数字文字だけを使用できる (A ～ Z、a ～ z、および 0 ～ 9)

- “.”、“-”、および “_” 文字を使用できる。
- “¥” と “/” 文字を使用できない。
- Windows 上で実行する場合、.bat サフィックスを含める必要がある。たとえば、名前は Windows では suspend_conn.bat、UNIX では suspend_conn.sh としてください。

例 1

Windows で、SYDNEY_DS データサーバの pubs2 データベースへのコネクションを毎週月曜日の午後 12:00 と毎月 16 日の午後 12:00 にサスペンドする “schedule1” を作成します。

1. テキスト・ファイル sql.txt を作成し、これにスケジュールする実際の Replication Server コマンド・ラインを含めます。たとえば、sql.txt に以下を含めることができます。

   ```
   suspend connection to SYDNEY_DS.pubs2
   go
   ```

2. Windows で、バッチ・ファイル suspend_conn.bat を作成し、これに **isql** および sql.txt のコマンド・ラインを実行するための該当するパラメータを含めます。たとえば、suspend_conn.bat に以下を含めることができます。

   ```
   %SYBASE¥OCS-15_0¥bin¥isql.exe -Usa -P -S SYDNEY_DS -I %SYBASE%
   ¥sql.ini -i %SYBASE%¥REP-15_5¥sched¥sql.txt
   ```

3. スケジュール “schedule1” を作成します。

   ```
   create schedule schedule1 as ‘0 12 16 * 1’ for exec
   ‘test.bat > c:¥temp¥test.out’
   go
   ```

例 2

UNIX で、suspend_conn.sh スクリプトを実行するための “schedule2” を作成します。このスケジュールは、SYDNEY_DS データサーバの pubs2 データベースへのコネクションを毎日午後 8:17 にサスペンドします。

```
create schedule schedule2 as ‘17 20 * * *’ for exec
‘suspend_conn.sh’
```

例 3

resume_conn.sh スクリプトを実行するための “schedule3” を作成します。このスケジュールは、SYDNEY_DS データサーバの pubs2 データベースへのコネクションを毎日午前 7:15 にレジュームします。

```
create schedule schedule2 as '15 7 * * *' for exec
‘resume_conn.sh’
```

スケジュールのシステム・テーブル・サポート

Replication Server では、作成したスケジュールを格納するための rs_schedule システム・テーブルと rs_scheduletxt システム・テーブルを用意しています。

『Replication Server リファレンス・マニュアル』の「Replication Server システム・テーブル」を参照してください。

スケジュールの変更

alter schedule を使用して、スケジュールを有効または無効にします。

構文：

alter schedule *sched_name* **set [on|off]**

たとえば、schedule1 を無効にするには、次のように入力します。

```
alter schedule schedule1 set off
```

スケジュールの削除

drop schedule を使用して、Replication Server からスケジュールを削除します。

たとえば、schedule1 を削除するには、次のように入力します。

```
drop schedule schedule1
```

スケジュールの表示

admin “schedule” を使用して、スケジュールを表示します。

構文：

admin "schedule"、' *sched_name*'

たとえば、schedule1 というスケジュールを表示するには、次のように入力します。

```
admin “schedule”, ‘schedule1’
```

注意： admin “schedule” では、**“schedule”** 句を二重引用符で囲む必要があります。

出力は次のようになります。

```
Schedule  Schedule    Status  Type  Owner  Sequence  Command
Name      Time
--------  ----------  ------  ----  -----  --------  -------

s1        27 * * * *  1       0     sa     1         suspend_conn.sh
```

スケジュール名を指定しない場合、**admin “schedule”** は、Replication Server のすべてのスケジュールを表示します。

複写の遅延

Replication Server を設定して、複写を一定時間遅らせることができます。

誤って削除されてしまったテーブルやレコードなど、プライマリ・データベースにコミットされた人為的なエラーから回復できるように、レプリケート・データベースをプライマリ・データベースから一定時間遅らせることにより、フェールバック・システムとしてレプリケート・データベースを使用できます。

dsi_timer パラメータを **alter connection** または **alter logical connection** と併用して、プライマリ・データベースでトランザクションがコミットされる時刻と、スタンバイ・データベースまたはレプリケート・データベースでトランザクションがコミットされる時刻との遅延を指定します。Replication Server は、この遅延期間が終了した後、アウトバウンド・キューのトランザクションをコミット順に処理します。

最大 24 時間までの遅延を指定できます。デフォルトの遅延値は 00:00 です。これは、遅延がないことを意味します。

dsi_timer を **alter connection** または **alter logical connection** とともに実行した後、コネクションをサスペンドして再開します。

注意： Replication Server は、プライマリ・データベースでの RepAgent と、**dsi_timer** を実行する DSI コネクションのある Replication Server との時差をサポートしていません。

SYDNEY_DS データ・サーバの `pubs2` レプリケート・データベースでトランザクションがコミットされる時刻を 2 時間遅らせるには、次を入力します。

```
alter connection to SYDNEY_DS.pubs2
set dsi_timer to '02:00'
```

遅延レプリケーション・ステータスの表示

admin who を使用し、**dsi_timer** を使用して設定した複写での遅延のステータスを表示します。

表 34 : 遅延複写ステータス

状態	説明
Active, DSI timer	コマンドの処理を実行中 (アクティブ状態)。**dsi_timer** はオン。
Awaiting Command, DSI timer	クライアントからのコマンドを待機中。**dsi_timer** はオン。

状態	説明
Awaiting Message, DSI timer	Open Server™ メッセージ・キューからのメッセージを待機中。**dsi_timer** はオン。

追加の説明や情報の入手

Sybase Getting Started CD、製品マニュアル Web サイト、オンライン・ヘルプを利用すると、この製品リリースについて詳しく知ることができます。

- Getting Started CD (またはダウンロード) – PDF フォーマットのリリース・ノートとインストール・ガイド、その他のマニュアルや更新情報が収録されています。
- Sybase 製品マニュアル Web サイト (http://sybooks.sybase.com/) にある製品マニュアルは、Sybase マニュアルのオンライン版であり、標準の Web ブラウザを使用してアクセスできます。マニュアルはオンラインで参照することも PDF としてダウンロードすることもできます。この Web サイトには、製品マニュアルの他に、EBFs/Maintenance、Technical Documents、Case Management、Solved Cases、Community Forums/Newsgroups、その他のリソースへのリンクも用意されています。
- 製品のオンライン・ヘルプ (利用可能な場合)

PDF 形式のドキュメントを表示または印刷するには、Adobe の Web サイトから無償でダウンロードできる Adobe Acrobat Reader が必要です。

注意： 製品リリース後に追加された製品またはマニュアルについての重要な情報を記載したさらに新しいリリース・ノートを製品マニュアル Web サイトから入手できることがあります。

サポート・センタ

Sybase 製品に関するサポートを得ることができます。

組織でこの製品の保守契約を購入している場合は、サポート・センタとの連絡担当者が指定されています。マニュアルだけでは解決できない問題があった場合には、担当の方を通して Sybase 製品のサポート・センタまでご連絡ください。

Sybase EBF と Maintenance レポートのダウンロード

EBF と Maintenance レポートは、Sybase Web サイトからダウンロードしてください。

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

2. メニュー・バーまたはスライド式メニューの [Support (サポート)] で [EBFs/Maintenance (EBF/メンテナンス)] を選択します。
3. ユーザ名とパスワードの入力が 求められたら、MySybase のユーザ名とパスワードを入力します。
4. (オプション) [Display (表示)] ドロップダウン・リストからフィルタを指定し、期間を指定して、[Go (実行)] をクリックします。
5. 製品を選択します。

   鍵のアイコンは、「Authorized Support Contact」として登録されていないため、一部の EBF/Maintenance リリースを ダウンロードする権限がないことを示しています。未登録ではあるが、Sybase 担当者またはサポート・センタから有効な情報を得ている場合は、[My Account (マイ・アカウント)] をクリックして、「Technical Support Contact」役割を MySybase プロファイルに追加します。
6. EBF/Maintenance レポートを表示するには [Info] アイコンをクリックします。ソフトウェアをダウンロードするには製品の説明をクリックします。

Sybase 製品およびコンポーネントの動作確認

動作確認レポートは、特定のプラットフォームでの Sybase 製品のパフォーマンスを検証します。

動作確認に関する最新情報は次のページにあります。

- パートナー製品の動作確認については、http://www.sybase.com/detail_list?id=9784 にアクセスします。
- プラットフォームの動作確認については、http://certification.sybase.com/ucr/search.do にアクセスします。

MySybase プロファイルの作成

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

1. http://www.sybase.com/mysybase を開きます。
2. [Register Now (今すぐ登録)] をクリックします。

アクセシビリティ機能

アクセシビリティ機能を使用すると、身体障害者を含むすべてのユーザーが電子情報に確実にアクセスできます。

Sybase 製品のマニュアルには、アクセシビリティを重視した HTML 版もあります。

オンライン・マニュアルは、スクリーン・リーダーで読み上げる、または画面を拡大表示するなどの方法により、視覚障害を持つユーザがその内容を理解できるよう配慮されています。

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

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

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

製品マニュアルには、アクセシビリティ機能に関する追加情報も記載されています。

索引

数字

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

あ

い

う

こ

さ

せ

そ

た

ち

つ

て

と

な

に

ね

の

は

ひ

ふ

へ

ほ

ま

む

め

れ

ろ

わ