SYBASE®
An SAP® Company

異機種間複写ガイド

Replication Server® 15.7.1

ドキュメント ID： DC20175-01-1571-01
改訂： 2012 年 4 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

複写システムの概要

Sybase は、Adaptive Server Enterprise (ASE) サーバから別の ASE サーバへの基本的な複写システム、および 1 つ以上のサーバが ASE 以外の場合の異機種間複写システムをサポートします。

基本的な複写システム

基本的な Sybase 複写システムはプライマリ Adaptive Server Enterprise (ASE) データベース、Replication Server®、レプリケート ASE データベースから構成されます。

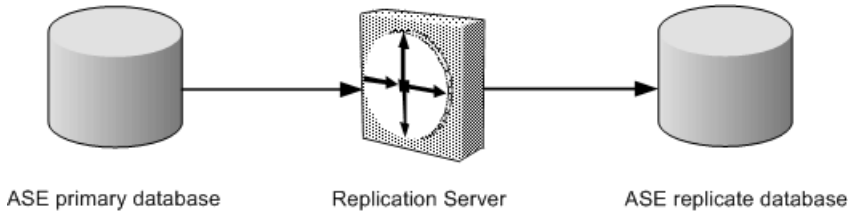
ASE には、Sybase 複写システムをサポートするために必要な機能すべてが備わっており、Replication Server 以外の追加コンポーネントは必要ありません。

次の 3 つのコンポーネントが含まれています。

- *プライマリ・データベース* – 元になるデータ変更オペレーション (トランザクション) が実行されるデータベース。完了したトランザクションだけが複写用に取り出される。
- *Replication Server* – 複写されるトランザクションをプライマリ・データベースから受け取り、レプリケート・データベースに配信する Sybase Open Client™ と Sybase Open Server™ 製品。
- *レプリケート・データベース* - 複写トランザクションを Replication Server から受け取り、プライマリ・データの独自の「コピー」にそのトランザクションを適用するデータベース。

基本的な Sybase 複写システムの図は、2 台の Adaptive Server と 1 台の Replication Server 間のデータの流れを示す基本的な Sybase 複写システムを示しています。

図 1 : 基本的な Sybase 複写システム



データは、プライマリ・データベースから Replication Server へ流れてから、レプリケート・データベースへと流れます。

Sybase 複写システムの基本概念や Replication Server の機能の詳細については、『Replication Server 管理ガイド』を参照してください。

異機種間複写システム

異機種 Sybase 複写システムは、同一または異なるベンダ (ASE から ASE 以外) の 2 つのデータベース間のデータ変更オペレーションから構成されます。

ASE から ASE レプリケーションの詳細については、『Replication Server 管理ガイド 第 1 巻』および『Replication Server 管理ガイド 第 2 巻』を参照してください。

異機種間複写には次が含まれます。

- Adaptive Server Enterprise (ASE) がプライマリ・データ・サーバまたはレプリケート・データ・サーバであり、ASE 以外のデータ・サーバ (IBM DB2 UDB など) がもう一方のデータ・サーバである複写システム
- プライマリ・データ・サーバとレプリケート・データ・サーバの両方が ASE 以外のデータ・サーバである複写システム (Oracle がプライマリ・データ・サーバで IBM DB2 UDB がレプリケート・データ・サーバである場合や、Microsoft SQL Server がプライマリ・サーバで Microsoft SQL Server がレプリケート・サーバである場合など)

ASE は、Replication Server をサポートするために強化されました。Replication Server のサポートに必要なデータ・サーバの要素すべて (プライマリ・データベースのデータ変更の取得メカニズムと、レプリケート・データベースのシステム・テーブルおよびストアド・プロシージャ) が、Adaptive Server Enterprise に組み込まれているか、Replication Server または Adaptive Server ソフトウェアに付属しているユーティリティによって使用可能になります。

ASE 以外のデータ・サーバを含む Sybase 複写システムを実装するには、以下の追加コンポーネントが必要です。

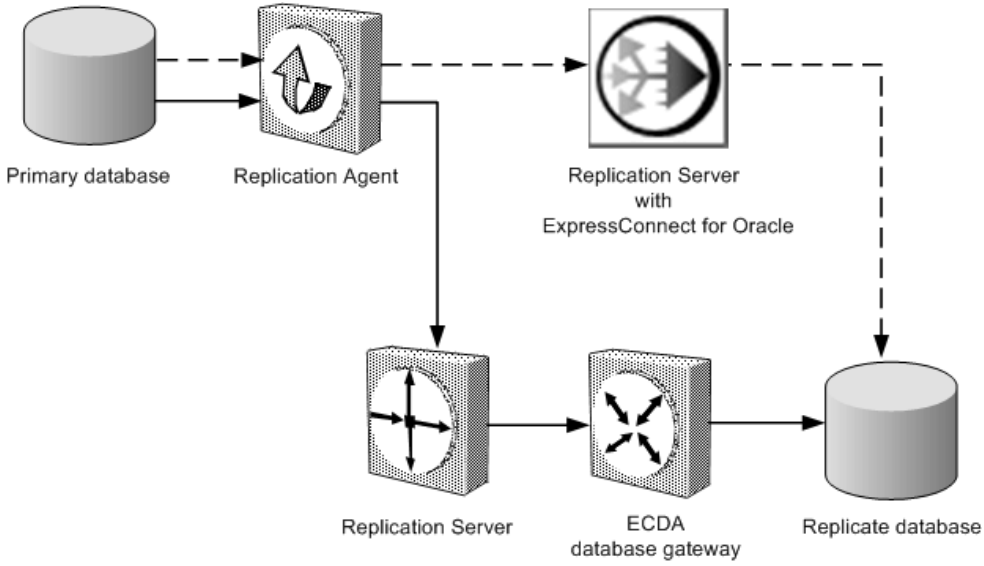
- Replication Agent
- Enterprise Connect™ Data Access (ECDA)、またはコネクティビティの必要条件に関して Replication Server や ExpressConnect for Oracle と互換性があるデータ・サーバ

ASE 以外のデータ・サーバを含む Sybase 複写システムは、データ・サーバ間のデータの流れを示す ASE 以外のデータ・サーバを含む通常の Sybase 複写システムを示しています。

- Replication Agent、Replication Server、Enterprise Connect Data Access データベース・ゲートウェイと、

- Replication Agent、Replication Server、ExpressConnect for Oracle です。

図 2 : ASE 以外のデータ・サーバを含む Sybase 複写システム



ECDA データベース・ゲートウェイを使用している場合、データは、プライマリ・データベースから Replication Agent へ、Replication Agent から Replication Server へ、Replication Server から ECDA データベース・ゲートウェイ、そしてデータベース・ゲートウェイからレプリケート・データベースへと流れます。

ECDA データベース・ゲートウェイは、Sybase Open Client および Sybase Open Server と ODBC またはレプリケート・データ・サーバのネイティブ・プロトコル間のコネクティビティを実現し、SQL 変換などのサービスを提供することによって IBM DB2 UDB、Microsoft SQL Server、Oracle の各データ・サーバをサポートします。また、Replication Server は、ASE 以外のデータ・サーバのデータ型サポートも提供します。

ExpressConnect for Oracle を使用している場合、データは、プライマリ・データベースから Replication Agent へ、Replication Agent から Replication Server へ、Replication Server から直接レプリケート・データベースへと流れます。

Replication Agent は、プライマリ・データベースの完了済みトランザクションを読み取り、配信のために Replication Server に送信することによって、ASE 以外のプライマリ・データ・サーバをサポートします。

Sybase 複写システムのコンポーネント

複写システムのコンポーネントの Sybase 複写システムにおける機能と役割を説明します。

複写システムのコンポーネントには、次のものが含まれます。

- プライマリ・データ・サーバ
- Replication Agent
- Replication Server
- データベース・ゲートウェイ
- ExpressConnect for Oracle
- レプリケート・データ・サーバ

参照：
- Sybase の複写製品 (29 ページ)

プライマリ・データ・サーバ

*プライマリ・データ・サーバ*は、複写システムにおけるデータ変更オペレーションまたはトランザクションの送信元となる 1 つ以上のプライマリ・データベースを管理します。プライマリ・データ・サーバは、複写に必要な情報を取得するように設定されます。

すべてのプライマリ・データ・サーバは、Replication Agent によってサポートされます。ASE の内部には、Replication Agent が含まれています。ASE 以外のサーバには、外部の Replication Agent が必要です。

参照：
- プライマリ・データベース (14 ページ)

サポートされているプライマリ・データベース・サーバ

Sybase の複写テクノロジでは、Adaptive Server Enterprise に加えて、別のリレーショナル・データベース・サーバからのトランザクションの複写もアクティブにサポートしています。

サポートされているリレーショナル・データベース・サーバには、次のものがあります。

- z/OS 上の IBM DB2 UDB
- UNIX/Windows 上の IBM DB2 UDB
- Microsoft SQL Server

- Oracle

これらのデータ・サーバのサポートされている最新バージョンについては、ASE 以外の特定のデータ・サーバをサポートする Replication Agent のマニュアルを参照してください。

Replication Agent

Replication Agent は、他の (レプリケート) データベースに配信するために、データ・スキーマに加えられた変更とストアド・プロシージャの実行を表すトランザクション情報をプライマリ・データ・サーバから Replication Server に転送します。

Replication Agent は、プライマリ・データを含むデータベースごと、または複写ストアド・プロシージャが実行されるデータベースごとに必要です。

Adaptive Server Enterprise では、埋め込み Replication Agent がデータベース管理システム・ソフトウェアに付属しています。ASE 用の Replication Agent は RepAgent と呼ばれ、Adaptive Server スレッドの 1 つです。

ASE 以外のデータ・サーバについては、Sybase では以下の Replication Agent 製品を提供しています。

- Replication Agent for DB2 UDB – IBM z/OS プラットフォームで実行されている IBM DB2 UDB サーバに、プライマリ・データ・サーバのサポートを提供します。
- Replication Agent – Linux、UNIX、または Microsoft Windows プラットフォームで実行されている DB2 UDB、Microsoft SQL Server、Oracle データ・サーバに、プライマリ・データ・サーバのサポートを提供します。

Replication Agent はプライマリ・データベースのトランザクション・ログを読み取ります。プライマリ Replication Server はトランザクションを再構築し、データのサブスクリプションを持つレプリケート・サイトに転送します。

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。

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。

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。

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。

データベース・ゲートウェイ

*データベース・ゲートウェイ*によって、クライアントは特定の通信プロトコルを使用して、異なるプロトコルを使用するデータ・サーバに接続できるようになります。

Sybase Enterprise Connect Data Access 製品群は、クライアントが (Replication Server と同様に) Sybase Open Client および Sybase Open Server プロトコルを使用して、データ・サーバのネイティブな通信プロトコルや標準の ODBC プロトコルを使用する Sybase 以外のデータ・サーバと接続できるようにする、データベース・ゲートウェイ・サーバで構成されます。

また、Sybase Enterprise Connect Data Access 製品を使用して、ASE 以外のレプリケート・データ・サーバからメタデータを取得することもできます。

参照：

- Enterprise Connect Data Access (45 ページ)

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。

参照：

- ExpressConnect for Oracle (49 ページ)

レプリケート・データ・サーバ

*レプリケート・データ・サーバ*は、レプリケート・データ、つまりプライマリ・データベース内のデータの「コピー」であるデータを格納するデータベースを管理します。

Replication Server は、データベース・ユーザとしてログインすることによって、レプリケート・データ・サーバ内のデータを管理します。ASE 以外のデータ・サーバの場合は、データベース・ゲートウェイ・サーバを介して、または直接レプリケート・データ・サーバにログインします。

Replication Server は、必要なデータ・オペレーションとトランザクション処理命令が直接的 (Adaptive Server Enterprise など) または間接的 (Enterprise Connect Data Access データベース・ゲートウェイ・サーバなど) にサポートされている場合、あらゆるサーバをデータ・サーバとして扱うことができます。

参照：
- レプリケート・データベース (15 ページ)

サポートされているレプリケート・データベース・サーバ

Sybase 複写テクノロジは、別のリレーショナル・データベース・サーバでのトランザクションの複写をサポートしています。

リレーショナル・データベース・サーバには、次のものがあります。

- z/OS 上の IBM DB2 UDB
- UNIX/Windows 上の IBM DB2 UDB
- Microsoft SQL Server
- Oracle
- Sybase IQ

Oracle、Microsoft SQL Server、DB2 UDB データ・サーバのサポートされている最新バージョンの詳細については、ASE 以外のこれらのデータ・サーバに関連付けられた、個別の ECDA データベース・ゲートウェイのマニュアルを参照してください。サポートされている ExpressConnect の Oracle データ・サーバ・バージョンの詳細については、インストールおよび設定ガイドを参照してください。

ASE 以外の複写

ASE 以外のサーバを使用して複写する場合は、データ・サーバの種類またはメーカーに関係なく、複写システムにおけるデータ・サーバの役割に関する問題を考慮する必要があります。

異機種間複写システムを適切に実装する上での最大の課題は、さまざまなベンダによって提供されているデータ・サーバ固有の特性に対処することです。1 つのデータ・サーバがプライマリ・データ・サーバとレプリケート・データ・サーバの両方の役割を果たす場合は (双方向複写)、さらに多くの問題を考慮しなければなりません。

プライマリ・データベース

異機種間複写システムを適切に実装するには、プライマリ・データベースの問題を処理する必要があります。

ASE 以外のプライマリ・データベースを使用する場合は、次の事項を検討してください。

- Replication Agent の稼働条件とデータ・サーバへの Replication Agent の干渉と影響。たとえば、一部の Replication Agent は、複写をサポートするためにプライマリ・データベースにデータベース・オブジェクトを作成して使用します。

- 他の複写システム・コンポーネントに必要な、データ・サーバのアクセスとパーミッション。データベースのプライマリ Replication Server と Replication Agent の両方に、データベースで定義された、プライマリ・データベース・オブジェクトにアクセスするための適切なパーミッションを持つユーザ ID とパスワードが必要です。
- データ・サーバと他の複写システム・コンポーネント間の通信をサポートするために必要なコネクティビティ。Replication Agent は、データ・サーバのネイティブ通信プロトコル、ODBC プロトコル、または JDBC プロトコルを使用して、プライマリ・データベースと通信します。Replication Server には、データ・サーバと通信するためにデータベース・ゲートウェイが必要な場合があります。
- 特定のデータ・サーバからの複写に関する固有の制限。たとえば、Replication Agent は、一部のデータ・サーバの設定オプションを制限します。Replication Server には、データベースによって、一部のネイティブなデータ型にサイズの制限がある場合があります。
- RSSD に格納されている複写定義が、特定のデータ・サーバの Replication Agent によってどのように使用されるか。たとえば、データベース・オブジェクト名を識別するときに、Replication Server と Replication Agent の両方が大文字と小文字を区別しますが、大文字と小文字を区別しないデータベースもあります。
- 特定のデータ・サーバから別の種類のデータ・サーバにトランザクションを複写するときに必要となる可能性があるデータ型変換。たとえば、ほとんどすべての種類のデータ・サーバが、それぞれに固有の方法で時間データを表します。あるデータベースの `TIMESTAMP` データ型を、別のデータベースでは `datetime` データ型に「変換」して格納する必要がある場合があります。
- 特定のデータ・サーバに固有の複写システム管理に関する問題。たとえば、データ・サーバに応じて、異なるシステム管理オプションを使用することができます。

特定のデータベースに固有のプライマリ・データベースの問題の詳細については、使用しているデータベースに該当するトピックを参照してください。

レプリケート・データベース

異機種間複写システムを適切に実装するには、レプリケート・データベースの問題を処理する必要があります。

ASE 以外のレプリケート・データベースを使用する場合は、次の事項を検討してください。

- 特定のデータベース・サーバに関する ECDA データベース・ゲートウェイの稼働条件。レプリケート・データベース・サーバおよび Replication Server とともに動作するように DirectConnect™ のアクセス・サービスを設定します。
- トランザクションをレプリケート・データベースに適用するために、複写システムのデータ・サーバで必要とされるアクセスとパーミッション。データベー

スのレプリケート Replication Server と ECDA ゲートウェイの両方に、データベースで定義された、レプリケート・データベース・オブジェクトにアクセスするための適切なパーミッションを持つユーザ ID とパスワードが必要です。

- レプリケート・データ・サーバと他の複写システム・コンポーネント間の通信をサポートするために必要なコネクティビティ。ECDA ゲートウェイは、データ・サーバのネイティブ通信プロトコルか、標準の ODBC または JDBC プロトコルを使用して、レプリケート・データベースと通信します。ASE 以外のデータ・サーバと通信する場合、Replication Server には通常、データベース・ゲートウェイが必要です。
- 特定のデータ・サーバへの複写に関する制限。たとえば、Replication Server には、データベースによって、一部のネイティブなデータ型に制限がある場合があります。
- Replication Server の動作をサポートするために必要なデータベース・オブジェクトの介入と影響。Replication Server には 2 つのテーブルが必要です。また、レプリケート・データベースを管理するためにストアド・プロシージャが必要な場合もあります。
- 特定のデータ・サーバに固有の複写システム管理に関する問題。たとえば、データ・サーバに応じて、異なるシステム管理オプションを使用することができます。

特定のデータベースに固有のレプリケート・データベースの問題の詳細については、使用しているデータベースに該当するトピックを参照してください。

文字セット

文字セットを設定すると、プライマリ・データベースとレプリケート・データベース間のデータの不整合を引き起こす問題を回避します。

プライマリ・データ・サーバとレプリケート・データ・サーバの種類が異なる異機種間複写システムでは、データ・サーバが同じ文字セットをすべてサポートしていない場合があります。そのような場合、複写システム・コンポーネントは、(プライマリ・データ・サーバの文字セットからレプリケート・データ・サーバの文字セットへの) 文字セットの変換を、少なくとも 1 回は実行する必要があります。

プライマリ・データ・サーバとレプリケート・データ・サーバの種類が同じである同機種間複写システムでも、複写システム・コンポーネントが複数の種類のプラットフォームに存在している場合は、文字セットの変換が必要になることがあります。

文字セットの問題を回避するには、次のいずれかを実行する必要があります。

- 複写システムのすべてのサーバおよびプラットフォームで同じ文字セットを使用する。

- 複写システムのすべてのサーバおよびプラットフォームで、互換性のある文字セットを使用し、適切な文字セットの変換を実行するように、複写システム・コンポーネントを設定する。

デフォルト文字セットの設定と上書きの詳細については、該当する Replication Agent のマニュアルを参照してください。

異機種間複写の制限事項

特定の関連データベースによっては、Sybase の複写テクノロジに基づき、異機種間複写システムにはいくつかの制限事項があります。

ストアド・プロシージャの複写

ストアド・プロシージャの複写によって、プライマリ・ストアド・プロシージャ呼び出しに引数として渡されるパラメータ値を含めた、複写するストアド・プロシージャの実行呼び出しが可能になります。

ストアド・プロシージャの複写を使用できるかどうかは、プライマリ・データベースおよびレプリケート・データベースの機能と、関連する Replication Agent および ECDA データベース・ゲートウェイによるサポートによって決まります。ストアド・プロシージャの複写がデータベースで使用可能かどうかを確認するには、個別の Replication Agent および ECDA コンポーネントのマニュアルを参照してください。

所有者指定のオブジェクト名

ASE 以外のデータベース内のレプリケート・テーブルおよびストアド・プロシージャにアクセスする場合、レプリケート・テーブルまたはストアド・プロシージャへの参照に所有者を指定しなければならないことがよくあります。

たとえば、Oracle 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するために割り当てられている Replication Server のメンテナンス・ユーザが *orauser* であるとします。**insert** コマンド (`table1` へのレプリケート) を実行すると、「テーブルが見つからない」というエラーが発生することがあります (テーブル `table1` の所有者が *bob* である場合)。`table1` を検索する場合、Oracle では `bob.table1` ではなく `orauser.table1` を探します。更新するレプリケート・テーブルを適切に指定するには、次の作業を行います。

- 正しいレプリケート・テーブルを参照するエイリアスを、Oracle のレプリケート・データベースで作成します。たとえば、“bob.table1” という完全修飾名を参照する `table1` という同義オブジェクトを Oracle で作成します。

- 複写定義を作成するときに、**with replicate table named [table_owner.['table_name']]** 句を使用します。同じ例を引き続き使用すると、次のような句になります。

```
with replicate table named bob.table1
```

複数のレプリケート・データベースがある場合の所有者の指定
table1 が複数のレプリケート・データベース (たとえば、Oracle レプリケート・テーブル bob.table1) に複写される場合は、問題が多少複雑になります。複写定義で **with replicate table named** 句を使用する方法では、1つのレプリケート・テーブル名しかサポートされません。

この問題に対処するには、必要とされるユニークなレプリケート・テーブル名ごとに、1 つずつ複写定義を作成します。各サブスクリプションが正しい複写定義を参照し、各複写定義が **with replicate table named** 句を使用していることを確認してください。

ラージ・オブジェクトの複写

ラージ・オブジェクト (LOB) データ型 (BLOB、CLOB、IMAGE、TEXT など) は、1つのカラム内における最長の文字ストリームおよびバイナリ・データ・ストリームをサポートします。プライマリ・データとレプリケート・データのどちらの場合においても、LOB データ型のサイズによって固有の問題が生じます。

プライマリ・データベースの LOB 複写の問題
プライマリ・データベースでは、LOB データ型はトランザクション・ログ機能に影響します。

Replication Agent の場合、LOB データの変更の保持をサポートする十分なログ・リソースが必要です。ログに記録されるのは、LOB データの更新後のイメージのみです。LOB 複写の機能は、Replication Agent の機能によって決まります。

レプリケート・データベースの LOB 複写の問題
Sybase 以外のデータベースがレプリケート・データベースである場合、レプリケート・データベースとの通信に使用されるデータベース・ゲートウェイは、Adaptive Server のテキスト・ポインタ処理をエミュレートできなければなりません。

ECDA Option for ODBC、ECDA Option for Oracle、ExpressConnect for Oracle、Mainframe Connect™ DirectConnect for z/OS Option のゲートウェイは、この機能を提供しています。

Adaptive Server Enterprise では、テキスト・ポインタを使用して text および image カラム・データの場所を特定します。このテキスト・ポインタは、これらのラージ・カラム内のデータに対して実際の更新を実行するシステム・ファンク

ションに渡されます。Replication Server でも、内部で同じ方法を使用して LOB データ型に適用します。Replication Server がテキスト・ポインタを取得すると、レプリケート・データベースにデータを適用するデータ・サーバ・ファンクションが呼び出されます。

ECDA Option for ODBC は、Microsoft SQL Server データベースへの LOB の複写をサポートしています。

参照：

- レプリケート・データ・サーバとしての Microsoft SQL Server (103 ページ)

レプリケート・データベースの設定

Replication Server は、Adaptive Server データベースを設定する **rs_init** というユーティリティを提供しています。

rs_init は、Adaptive Server データベースをプライマリまたはレプリケート・データベースとして、次のように設定します。

- Replication Server データベース・コネクションを作成する
- 必要なテーブルとストアド・プロシージャをレプリケート・データベースに作成する
- Replication Server のメンテナンス・ユーザ ID を定義する

異機種間複写のサポートには、**rs_init** に相当するユーティリティは含まれていません。代わりに、コネクションを作成するための Replication Server コマンドと、複写をサポートするオブジェクト (メンテナンス・ユーザなど) を作成するためのプライマリ・データ・サーバ・コマンドとレプリケート・データ・サーバ・コマンドを使用できます。これらのタスクの多くは、Replication Server 15.2 で導入された **create connection** コマンドの “using profile” 句を使用して実現できます。

Replication Server の暗号化カラムのサポート

Replication Server は、Adaptive Server データベース間の暗号化カラム・データの複写をサポートしています。ただし、ASE 以外のレプリケート・データベースへの暗号化カラム・データの複写はサポートしていません。

暗号化カラムを含む ASE データベースに暗号化されていないデータを複写するには、Adaptive Server コネクションに対する **rs_set_ciphertext** ファンクション文字列を無効にします。**rs_set_ciphertext** ファンクション文字列は、デフォルトで、すべての ASE コネクションに対して実行されます。これは、複写するデータが既に暗号化されていて、プライマリ・データベースも同じ暗号を使用した ASE であると仮定されるレプリケート ASE データベースを示します。**rs_set_ciphertext** ファンクション文字列を無効にすることにより、レプリケート ASE は入力レプリケート・データに対して暗号化を実行できます。ASE が入力データを暗号化できるように

することは、プライマリ・データベースが ASE 以外の場合またはプライマリ ASE データベースが暗号化されたカラムを使用しない場合に適しています。

rs_set_ciphertext ファンクション文字列

rs_set_ciphertext は、Adaptive Server テーブルへの暗号化カラムの複写を制御します。

ファンクション文字列 **rs_set_ciphertext** を変更して、ASE 固有のコマンド "**set ciphertext on**" の実行を無効にします。

```
alter function string rs_set_ciphertext
for some_function_string_class
output language
''
```

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とは、サブスクリプションを作成してアクティブ化し、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をコピーすることによって、レプリケート・データベースを初期化することです。

プライマリ・データベースからデータを複写するには、プライマリ・データベースの状態と一致した状態になるように、各レプリケート・データベースを設定し、データを読み込んでおく必要があります。Replication Server では、次の 2 種類のサブスクリプション・マテリアライゼーションをサポートしています。

- バルク・マテリアライゼーション ― サブスクリプションを手動で作成してアクティブ化し、複写システムのコントロール外のデータ・アンロード・ユーティリティとデータ・ロード・ユーティリティを使用してレプリケート・データベースにデータを読み込みます。
- 自動マテリアライゼーション ― Replication Server コマンドを使用してサブスクリプションを作成し、レプリケート・データベースにデータを読み込みます。

異機種間複写ではバルク・マテリアライゼーション・メソッドがサポートされていますが、個別の Replication Agent 機能に基づいて複雑さが異なります。

サブスクリプション・マテリアライゼーション全般については、『Replication Server 管理ガイド』を参照してください。また、個別の Replication Agent とそのマテリアライゼーションのサポートの詳細については、該当する Replication Agent のマニュアルを参照してください。

Replication Server の rs_dump コマンド

rs_dump コマンドは通常、複写システム全体におけるデータベースのダンプ・アクティビティを調整するために使用されます。

レプリケート・コネクションが **rs_dump** トランザクションを受け取ると、Replication Server はそのコネクションに対して **rs_dump** ファンクション文字列を実

行します。コマンドが、必要なあらゆる操作を実行するように、**rs_dump** ファンクション文字列をカスタマイズできます。

ASE 以外のプライマリ・データベースの複写の場合、一部の Replication Agent には Sybase 以外のプライマリ・データベースから **rs_dump** コマンドを呼び出す方法が用意されています。プライマリ・データベースからの **rs_dump** の実行がサポートされているかどうかを確認するには、該当する Replication Agent のマニュアルを参照してください。

レプリケート・データベースについては、**rs_dump** のデフォルト・ファンクション文字列は用意されていません。

rs_dump コマンド、その使用方法、およびファンクション文字列の変更の詳細については、『Replication Server リファレンス・マニュアル』を参照してください。

Replication Server rs_marker コマンド

rs_marker コマンドは、マテリアライゼーション・プロセスを支援するプライマリ・データベース・トランザクション・ログのマーカ・メカニズムです。

rs_marker を実行すると、**activate subscription** コマンドと **validate subscription** コマンドがプライマリ Replication Server に渡されます。ほとんどの Replication Agent は、マテリアライゼーションを支援するために **rs_marker** の呼び出しをサポートしています。

rs_marker 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』を参照してください。特定のデータベースでの **rs_marker** の使用方法と可用性の詳細については、該当する Replication Agent のマニュアルを参照してください。

Replication Server の rs_dumptran コマンド

rs_dumptran コマンドは通常、複写システム全体におけるデータベース・トランザクションのダンプ・アクティビティを調整するために使用されます。

レプリケート・コネクションが **rs_dumptran** トランザクションを受け取ると、Replication Server はそのコネクションに対して **rs_dumptran** ファンクション文字列を実行します。コマンドが必要なあらゆる操作を実行するように、**rs_dumptran** ファンクション文字列をカスタマイズできます。

異機種間複写では、Sybase 以外のプライマリ・データベースに対して **rs_dumptran** はサポートされません。

レプリケート・データベースについては、**rs_dumptran** のデフォルト・ファンクション文字列は用意されていません。

rs_dumptran コマンド、その使用方法、およびファンクション文字列の変更の詳細については、『Replication Server リファレンス・マニュアル』を参照してください。

Replication Server の rs_subcmp ユーティリティ

プライマリ・テーブルとレプリケート・テーブルを比較し、確認された差異を必要に応じて調整するために使用できる **rs_subcmp** 実行プログラムが用意されています。

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ティビティが確保されている場合には、Sybase 以外のデータベースをサポートするために、**rs_subcmp** を使用できます。プライマリ・データベースおよびレプリケート・データベース用に、両者の比較可能な出力を生成するカスタム **SELECT** コマンドを開発する必要もあります。その他に、そのような機能を備えたサード・パーティのツールを購入するか、または独自のアプリケーションを作成する方法があります。

参照：

- 異機種データベースの調整 (269 ページ)

動的 SQL

動的 SQL により、Replication Server データ・サーバ・インタフェース (DSI) を使用して、ターゲット・ユーザ・データベースで動的 SQL 文を作成し、繰り返し実行できます。

動的 SQL は、UNIX、Windows、Linux 上の Oracle、DB2 UDB z/OS、DB2 UDB で使用できます。Microsoft SQL と Sybase IQ では使用できません。

バルク・コピー

バルク・コピーにより、Replication Server データ・サーバ・インタフェース (DSI) は、Open Client Open Server Bulk-Library インターフェイスを使用して同じテーブルで大量の insert 文を複写する場合にパフォーマンスが向上します。

ExpressConnect for Oracle を使用する場合、Sybase IQ と Oracle を例外として、バルク・コピーは ASE 以外のデータ・サーバのいずれにも使用できません。

Replication Server rs_ticket ストアド・プロシージャ

rs_ticket は、Replication Server のパフォーマンス、モジュールのハートビート、複写の正常性、テーブルレベルのクワイスをモニタするのに役立つ、プライマリ・データベースのストアド・プロシージャです。

rs_ticket は、Oracle、DB2 UDB on UNIX、Windows、Linux、Microsoft SQL で使用できます。DB2 UDB z/OS では使用できません。『Replication Agent リファレンス・マニュアル』を参照してください。

複写システムの ASE 以外の設定

Replication system の問題は、異機種または ASE 以外のデータ・サーバと構成が異なるために発生します。

ASE 以外のプライマリから Adaptive Server レプリケートへの複写

最も簡単な異機種間複写シナリオは、ASE 以外のプライマリ・データベースから Adaptive Server レプリケート・データベースへの単方向の複写です。

固有の要件は、ASE 以外のプライマリ・データベースからトランザクション・データを抽出するために設計された Replication Agent が存在すること、およびプライマリ・データベースのネイティブ・データ型を Adaptive Server データ型に変換するための Replication Server の異機種データ型サポート (HDS) 機能を適用することだけです。

『Replication Server 管理ガイド 第 1 巻』の「複写テーブルの管理」の「HDS を使用したデータ型の変換」を参照してください。

複写システムのコンポーネント
ASE 以外のプライマリから Adaptive Server レプリケートへの複写の設定には、次のコンポーネントが必要です。

- ASE 以外のプライマリ・データ・サーバ。Oracle などです。
- プライマリ・データ・サーバ用に設計された Replication Agent
- Replication Server
- Adaptive Server レプリケート・データ・サーバ

複写システムに関する問題
ASE 以外のプライマリから Adaptive Server レプリケートへの複写の設定では、ユーザ ID がプライマリ・データベースにトランザクションを適用しない場合でも、プライマリ・データベースの Replication Server データベース・コネクション

に、(Replication Agent に対してのみ検証される) プライマリ・データベースに有効なユーザ ID とパスワードが必要になる場合があります。

ASE サーバのプライマリから ASE 以外のサーバのレプリケートへの複写

簡単な異機種間複写シナリオは、Adaptive Server プライマリ・データベースから ASE 以外のレプリケート・データベースへの単方向の複写です。

固有の要件は、レプリケート・データベースにトランザクション・データを適用するためのコンポーネント、および Adaptive Server データ型をレプリケート・データベースのネイティブ・データ型に変換するための Replication Server の HDS 機能を適用することだけです。

HDS の詳細については、『Replication Server 管理ガイド 第 1 巻』の「複写テーブルの管理」で「HDS を使用したデータ型の変換」を参照してください。

複写システムのコンポーネント
Adaptive Server プライマリから ASE 以外のレプリケートへの複写の設定には、次のコンポーネントが必要です。

- Adaptive Server プライマリ・データベース
- Replication Server
- Microsoft SQL Server の ECDA Option for ODBC など、レプリケート・データ・サーバ向けに設計された Oracle または関連する ECDA データベース・ゲートウェイを複写するための ExpressConnect for Oracle
- ASE 以外のレプリケート・データ・サーバ。Microsoft SQL Server など

複写システムに関する問題
Adaptive Server プライマリから ASE 以外のレプリケートへの複写の設定では、以下の問題を考慮してください。

- レプリケート・データベースの Replication Server データベース・コネクションに、レプリケート・データベースに有効なユーザ ID とパスワード (メンテナンス・ユーザ) が指定されている必要があります。このユーザ ID には、レプリケート・データベースにレプリケート・トランザクションを適用する権限が必要です。
- レプリケート・データベース用の正しいプロファイルを使用して、Replication Server レプリケート・データベース・コネクションを作成します。接続プロファイルは、レプリケート・データベースの正しいファンクション文字列クラスとエラー・クラスを指定します。また、クラス・レベル変換の定義とレプリケート・データベース・オブジェクトの作成の複写サポートを含める場合があります。

ASE 以外のプライマリから ASE 以外のレプリケートへの複写

ASE プライマリから ASE 以外のレプリケートへのシナリオは、ASE 以外のデータ・サーバの組み合わせに応じて複雑さが異なります。

複写システムのコンポーネント
ASE 以外のプライマリから ASE 以外のレプリケートへの複写の設定には、次のコンポーネントが必要です。

- ASE 以外のプライマリ・データ・サーバ。Oracle などです。
- プライマリ・データ・サーバ用に設計された Replication Agent。Replication Agent for Oracle などです。
- Replication Server
- レプリケート・データ・サーバ用に設計されたゲートウェイ (ECDA Option for ODBC、ExpressConnect for Oracle など)。
- ASE 以外のレプリケート・データ・サーバ。Microsoft SQL Server など

複写システムに関する問題
ASE 以外のプライマリから ASE 以外のレプリケートへの複写の設定では、以下の問題を考慮してください。

- Replication Server プライマリ・データベース・コネクションに、プライマリ・データベースに有効なユーザ ID とパスワードが必要になる場合があります。このユーザ ID には、レプリケート・トランザクションを適用する権限が必要です (プライマリ・データベースにトランザクションが複写されない場合でも必要です)。
- レプリケート・データベース用の正しいプロファイルを使用して、Replication Server レプリケート・データベース・コネクションが作成されている必要があります。接続プロファイルは、レプリケート・データベースの正しいファンクション文字列クラスとエラー・クラスを指定します。また、クラス・レベル変換の定義とレプリケート・データベース・オブジェクトの作成の複写サポートを含める場合があります。

ASE 以外から ASE 以外への双方向複写

ASE 以外から ASE 以外への双方向複写シナリオでは、複写は各データベース間で発生します。

ASE 以外の各データベースに、Replication Agent と ECDA データベース・ゲートウェイの両方が必要です。

複写システムのコンポーネント
ASE 以外のプライマリから ASE 以外のレプリケートへの双方向複写の設定には、次のコンポーネントが必要です。

- ASE 以外のプライマリ・データ・サーバ。UNIX、Windows、Linux 上の DB2 UDB などです。
- プライマリ・データ・サーバ用に設計された Replication Agent。Replication Agent for Oracle、Microsoft SQL Server、DB2 UDB などです。
- ECDA データベース・ゲートウェイは、レプリケート・データベースとして動作する「プライマリ」データ・サーバ用に設計されています。ECDA Option for ODBC (for DB2 UDB) などです。
- Replication Server
- レプリケート・データ・サーバの ECDA データベース・ゲートウェイ。ECDA Option for ODBC (for Microsoft SQL Server) などです。
- プライマリ・データ・サーバとして動作する「レプリケート」データ・サーバ用に設計された Replication Agent。Replication Agent for Linux、Microsoft Windows、UNIX などです。
- ASE 以外のレプリケート・データ・サーバ。Microsoft SQL Server など

複写システムに関する問題
技術的には、Replication Server データベース・コネクションを 2 つ (各データベースに「プライマリおよびレプリケート」コネクションを 1 つずつ) 使用するだけで、双方向複写シナリオを設定できます。

注意： 双方向複写の問題に関する以下の説明では、2 つのデータベースを Database #1 と Database #2 と呼びます。これは、どちらのデータベースも複写システムで「プライマリ」と「レプリケート」の両方の役割を果たすためです。

ASE 以外のプライマリから ASE 以外のレプリケートへの双方向複写の設定では、以下の問題を考慮してください。

- Database #1 用の Replication Server プライマリ・データベース・コネクションに、プライマリ・データベースに有効なユーザ ID とパスワードが指定されている必要があります。このユーザ ID は、Database #2 用の Replication Server レプリケート・データベース・コネクションで指定されているユーザ ID (メンテナンス・ユーザ) と同じでなければなりません。このユーザ ID には、Database #1 のレプリケート・テーブルにトランザクション・オペレーションを適用する権限が必要です。
- Database #1 の Replication Agent は、Database #2 内のレプリケート・テーブルからトランザクションが返されないようにするために、メンテナンス・ユーザのトランザクションを無視するように設定されている必要があります。メンテナンス・ユーザのトランザクションを無視するように Replication Agent を設定す

る方法の詳細については、該当する Replication Agent のマニュアルを参照してください。

- Database #2 用の Replication Server プライマリ・データベース・コネクションに、プライマリ・データベースに有効なユーザ ID とパスワードが指定されている必要があります。このユーザ ID は、Database #1 用の Replication Server レプリケート・データベース・コネクションで指定されているユーザ ID (メンテナンス・ユーザ) と同じでなければなりません。このユーザ ID には、Database #2 のレプリケート・テーブルにトランザクション・オペレーションを適用する権限が必要です。
- Database #2 の Replication Agent は、Database #1 内のレプリケート・テーブルからトランザクションが返されないようにするために、メンテナンス・ユーザのトランザクションを無視するように設定されている必要があります。メンテナンス・ユーザのトランザクションを無視するように Replication Agent を設定する方法の詳細については、該当する Replication Agent のマニュアルを参照してください。
- レプリケート・データベース用の正しいプロファイルを使用して、Database #1 と Database #2 への Replication Server レプリケート・データベース・コネクションが作成されている必要があります。接続プロファイルは、レプリケート・データベースの正しいファンクション文字列クラスとエラー・クラスを指定します。また、クラス・レベル変換の定義とレプリケート・データベース・オブジェクトの作成の複写サポートを含める場合があります。

Sybase の複写製品

Sybase では、Sybase の複写テクノロジに基づいた、異機種データ・サーバや ASE 以外のデータ・サーバを含む複写システムを特にサポートする製品群を提供しています。

Sybase の複写製品には次のものがあります。

- Replication Server。Sybase の高度な複写テクノロジの中心部であり、特に Sybase 複写システム内の ASE 以外のデータ・サーバをサポートするための機能が組み込まれています。
- Replication Agent、および Enterprise Connect Data Access (ECDA) または ExpressConnect for Oracle のいずれかで構成されている Replication Server Options。
  - Replication Agent は、ASE 以外のプライマリ・データベースから複写データを取得する方法を提供することで Replication Server をサポートします。このサポートは、DB2 UDB、Microsoft SQL Server、Oracle データ・サーバに対して提供されます。
  - ECDA データベース・ゲートウェイは、ASE 以外のさまざまなデータベースへのアクセスを提供し、これらのデータベースが Sybase 複写システムでレプリケート・データベースとして機能できるようにすることで Replication Server をサポートします。
  - ExpressConnect for Oracle は、別のゲートウェイ・サーバを必要とせずに、Replication Server と Oracle データベースとの間の直接通信を提供し、Replication Server をサポートします。Express Connect for Oracle は Replication Server Options 15.5 以降でのみ使用できます。
- Replication Agent for IBM DB2 UDB。メインフレーム上の IBM DB2 UDB からデータを複写します。

Replication Server

Replication Server は、リモートの集中型データベースからではなく、ローカルのデータにアクセスできます。集中型データ・システムと比べると、複写システムではシステムのパフォーマンスとデータ可用性が向上し、通信のオーバヘッドが軽減されます。

Replication Server は、データを複写するための、費用効果が高く障害に強い (フォールト・トレラントな) システムを提供します。Replication Server は、トランザクション (データのコピーではなく追加された変更作業) とストアド・プロシージャの呼び出し (ストアド・プロシージャの実行によって生じるオペレーションで

はありません) を複写するので、パフォーマンスが高い分散データ環境が実現すると同時に、システム全体で複写データのトランザクションの整合性が確保されます。

Replication Server の動作

Replication Server は、ネットワーク上でトランザクションの整合性を維持しながら、レプリケート・トランザクションを管理することにより、ネットワークを介してデータを配信します。

アプリケーション開発者とシステム管理者に、データとストアド・プロシージャを複写するための、柔軟性のあるパブリッシュとサブスクライブ・モデルを提供します。

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、次の処理を実行します。

- 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信し、それらをデータに対するサブスクリプションを持っているサイトに分配する。
- 他の Replication Server からトランザクションを受け取り、ローカル・データベースに適用する。

Replication Server のシステム・テーブルには、これらのタスクの実行に必要な情報が格納されます。システム・テーブルには、複写データと次に示す複写オブジェクトの説明が格納されます。

- 複写定義およびサブスクリプション
- Replication Server ユーザ用のセキュリティ・レコード
- 他の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方法
- その他の管理情報

Replication Server のシステム・テーブルは、Replication Server システム・データベース (RSSD) と呼ばれるデータベースに格納されます。

Replication Server で複写情報を管理するには、複写コマンド言語 (RCL) を使用します。RCL コマンドは、SQL コマンドに似ており、Sybase の対話型 SQL ユーティリティである **isql** を使用して、Replication Server で実行できます。RCL 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。

パブリッシュとサブスクライブ・モデル

ユーザは、他 (レプリケート・サイト) の Replication Server がサブスクリプションを作成する対象のプライマリ・サイトにデータがパブリッシュされます。

プライマリ・データベースで実行されたトランザクションは Replication Agent によって検出され、ローカルの Replication Server に転送されます。次に、この Replication Server が、ネットワークを介して送信先サイトの Replication Server に情報を分配します。その後、これらの Replication Server により、リモート・クライアントの要求に従って、レプリケート・データベースが更新されます。

プライマリ・データは、Replication Server が他のデータベースに複写するデータのソースです。データをパブリッシュおよびサブスクライブするには、まず、複写定義を作成してプライマリ・データのスコープとロケーションを指定します。複写定義には、テーブルの構造を記述します。データベース複写定義を使用すると、個々のテーブル、ファンクション、DDL を複写できます。テーブル複写定義には、テーブルの構造を記述し、テーブルに対して更新と削除を問い合わせるために使用するキーを宣言します。

複写定義を作成しただけでは、Replication Server がデータを複写するようにはなりません。さらに、複写定義に対するサブスクリプションを作成して、別のデータベースにデータを複写するように Replication Server に指示する必要があります。サブスクリプションは、次のような SQL の **select** 文に似ています。**where** 句を追加して、ローカル・データベースに複写するテーブルのローを指定できます。

さまざまなオブジェクトをフィルタするために、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。レプリケート・テーブルでは、異なる複写定義のサブスクリプションを作成し、データの異なるビューを取得することができます。

複写定義またはパブリケーションのサブスクリプションを作成後に、Replication Server は、データのサブスクリプションを持つデータベースにトランザクションを複写します。

複写ファンクション

この場合、1 つの複写ファンクションに多数の変更をカプセル化することによって、通常のデータ複写よりもパフォーマンスを向上させることができます。

複写ファンクションはテーブル複写定義と関連付けられていないので、データを直接修正するかどうかに関係なくストアド・プロシージャを実行できます。

データ・サーバによっては、Replication Server を使用して、ストアド・プロシージャ呼び出しをデータベース間で非同期に複写できます。

注意： Replication Server は、すべての種類のデータ・サーバでストアド・プロシージャの複写をサポートしているわけではありません。特定のデータ・サーバにおけるストアド・プロシージャの複写の詳細については、該当する Replication Agent のマニュアルを参照してください。

複写ファンクションを使用すると、別のデータベースでストアド・プロシージャを実行できます。複写ファンクションを使用すると、次のことが実行できます。

- サブスクリプションを作成するサイトに、ストアド・プロシージャの実行を複写する。
- データベースの実際の変更ではなく、ストアド・プロシージャの名前とパラメータだけを複写することによって、パフォーマンスを向上させる。

Replication Server は、適用ファンクションと要求ファンクションの両方をサポートします。

- 「適用ファンクション」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。レプリケート・サイト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し、プライマリ・データベースで複写対象としてマークを付けます。
- 要求ファンクションは、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ます。要求ファンクションのサブスクリプションはありません。レプリケート・データベースで、複写対象としてストアド・プロシージャにマークを付けます。

トランザクション管理

Replication Server では、トランザクション処理サービスの提供をデータ・サーバに依存しています。分散データの一貫性を保証するために、データ・サーバは、原子性や一貫性といったトランザクション処理規則に従う必要があります。

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・データを持つテーブルの更新に失敗すると、Replicatin Server は、そのトランザクションを他のサイトには分配しません。トランザクションによってプライマリ・データが更新されると、Replication Server は変更を分配し、障害が発生しないかぎり、データのサブスクリプションを作成したすべてのサイトで更新が正常に行われます。

他のシステム・コンポーネントとの関係

Replication Server は、サーバまたはクライアントとして、複写システムの他のコンポーネントと対話します。

サーバとしての Replication Server は、以下からのコネクションをサポートしています。

- プライマリ・データベースからのデータベース・コマンドの送信に使用される Replication Agent
- 複写システム内で、メッセージ配信処理を分散させ、複写システムにスケーラビリティを提供するための Replication Server
- 管理、データ・サーバの識別、メッセージのパブリケーションとサブスクリプションなどに使用されるユーザ・ツールまたは管理ツール

クライアントとしての Replication Server は、以下に接続します。

- 外部 Adaptive Server Enterprise データベース上にあるかまたは内部 Embedded RSSD (ERSSD) である Replication Server システム・データベース (RSSD)
- ASE 以外のレプリケート・データベースに接続するためのデータベース・ゲートウェイ
- ExpressConnect for Oracle を使用すると、Oracle はデータベースを直接複写します。

Replication Server の通信プロトコル

Replication Server は、基本となる通信プロトコルとして Sybase Tabular Data Stream™ (TDS) を使用する Open Client および Open Server アプリケーションです。

Replication Server にサービスを要求するすべてのクライアントに、Open Client インタフェースが実装されている必要があります。これらのクライアントには、Replication Agent、システム管理ツール、**isql** などのユーザ・インタフェース・ツールが含まれます。

他の Replication Server やレプリケート・データ・サーバにメッセージを配信するクライアントとして、Replication Server は Open Client インタフェースを使用します。そのため、Replication Server がデータ・サーバにメッセージを送信する必要がある場合、そのデータ・サーバが TDS で実行されている Open Server インタフェースをサポートしているか、Replication Server とそのレプリケート・データ・サーバ間に Open Server/TDS ブリッジまたは「ゲートウェイ」アプリケーションがあることが必要です。

Sybase IQ は、Replication Server にとって Open Server に見えるため、複写用に追加のゲートウェイ・ソフトウェアは必要ありません。ExpressConnect を使用して直接レプリケート・データベースへ接続できる Oracle を除き、DB2 UDB へ複写するためのゲートウェイ・ソフトウェア、Microsoft SQL Server、および Oracle は、Sybase ECDA データベース・ゲートウェイの形式です。ECDA ゲートウェイには、Open Server/TDS からレプリケート・データ・サーバのネイティブ・インタフェースに接続するもの (ECDA Option for Oracle など) もあれば、Open Server/TDS からデータ・サーバの ODBC または JDBC ドライバに接続するものもあります。Replication Server の設定は、使用するゲートウェイによって異なります。

ExpressConnect を使用した Oracle への複写では、追加のゲートウェイを必要としません。ExpressConnect はネイティブの Oracle 接続を使用するため、ExpressConnect を使用した Replication Server は Oracle に直接接続できます。

Replication Server のユーザ ID とパーミッション

Replication Server には、いくつかの異なるユーザ ID が必要です。他のコンポーネント (またはユーザ) が Replication Server にアクセスするために必要なユーザ ID もあれば、Replication Server が複写システム内の他のコンポーネントにアクセスするために必要なものもあります。

ユーザ ID は、Replication Server の **create connection** コマンドを使用して、Replication Server で定義します。

注意： 複写システムの設定によっては、下記のリストにある一部のユーザ ID が不要な場合もあります。たとえば、プライマリ・データベースとレプリケート・データベースに別の Replication Server を使用している場合、プライマリ Replication Server には、レプリケート・データベースにアクセスするためのユーザ ID は必要ありません。

Replication Server で定義されるユーザ ID は、次のとおりです。

- Replication Agent ユーザ ― プライマリ Replication Server にログインするために Replication Agent が使用します。このユーザ ID には、LTL インタフェースを介してデータベース・コマンドを配信するために、**connect source** パーミッションが必要です。
- Replication Server ユーザ ― Replication Server にログインし、メッセージを転送するために、他の Replication Server が使用します。このユーザ ID には、RCL インタフェースを介してデータベース・コマンドを転送するために、**connect source** パーミッションが必要です。
- SysAdmin ユーザ – 管理作業を実行するために、システム管理者またはシステム管理ツールが使用します。作業によっては、このユーザ ID には **sa**、**create object**、または **primary subscribe** パーミッションが必要です。
- メンテナンス・ユーザ – レプリケート・データ・サーバにメッセージを配信するために、Replication Server が使用します。このユーザ ID には、配信されるメッセージのマップ先のコマンドをプライマリ・データベースに対して実行するために必要な、レプリケート・データ・サーバでのパーミッションが必要です。メンテナンス・ユーザが実行する作業は複写されません。
- レプリケート・ユーザ ― プライマリ・データ・サーバにメッセージを配信するために、レプリケート Replication Server が使用します。「要求」メッセージ、つまり、プライマリ・データ・サーバに配信するために選択されたレプリケート・データ・サーバからのメッセージの配信については、Replication Server は、レプリケート・データベースでコマンドを実行するユーザのユーザ ID を使用します。このユーザ ID には、配信されるメッセージのマップ先のコ

マンドを実行するために必要な、プライマリ・データ・サーバでのパーミッションが必要です。

- RSI ユーザ ― 他の Replication Server にログインして、配信されるメッセージを転送するために Replication Server が使用します。このユーザ ID には、レプリケート Replication Server での **connect source** パーミッションが必要です。
- RSSD ユーザ ― 操作可能データを管理する Replication Server システム・データベース (RSSD) にログインするために、Replication Server が使用します。このユーザ ID には、オブジェクトの作成と削除、プロシージャの実行、テーブルの問い合わせと更新を行うための RSSD でのフル・コントロールが必要です。

Replication Agent との関係

Replication Server は、レプリケート・データ・サーバのニーズに合わせて拡張可能 (カスタマイズ可能なファンクション文字列とエラー処理、カスタム・データ型定義、データ型間の変換) ですが、プライマリ・データ・サーバについてのサポートは限られています。

プライマリ・データ・サーバへの Replication Server のインタフェースは、専用のログ転送言語 (LTL) です。プライマリ・データ・サーバからのトランザクションがプライマリ Replication Server に配信されるには、LTL への変換が必要です。そのため、プライマリ・データ・サーバのサポートは、プライマリ・データベースのオペレーションのために LTL への変換を行う Replication Agent が Sybase から提供されているものに限られます。

プライマリ側とレプリケート側の Replication Server のインタフェースは、TDS で実行されている基本の Open Client/Open Server インタフェースによってサポートされます。

LTM ロケータの更新

プライマリ Replication Server は、「ロケータ」値 (LTM ロケータ) を保持しています。これにより、プライマリ Replication Server ですべてのデータが正常に受け取られているトランザクション・ログ内の最後のポイントを特定します。

Replication Agent は、Replication Server コネクションにこの値を定期的に要求して、トランザクション・ログ内の位置を特定します。その後、これを使用して、ログから古いデータを解放または削除できる位置を特定できます。

LTM ロケータの更新を要求する頻度の決定には、パフォーマンスとの兼ね合いが必要です。Replication Server に LTM ロケータ値を頻繁に問い合わせると、複写の速度が遅くなる可能性がある一方 (Replication Agent は、LTM ロケータ値を要求して受け取るために必要な時間、LTL コマンドの送信を停止しなければならない)、データをプライマリ・データベースのトランザクション・ログから解放できる機会が増えます。再起動時には、Replication Agent は、Replication Server から最後の

LTM ロケータ値を受け取った後にログに保持されているすべてのデータを、再送信しなければなりません。

一般に、複写のスループットのパフォーマンスを優先する場合は、十分なログ・リソースを確保し、ログ・トランケーションの頻度と LTM ロケータ値の取得の頻度を少なくしてください。ログ・リソースが少ない場合は、LTM ロケータ値の取得とトランケーションをより頻繁に行わなければならない可能性があります。

LTM ロケータの使用の詳細については、該当する Replication Agent のマニュアルを参照してください。

LTL の生成

Replication Server に送信される情報のバイト数は、複写システムのパフォーマンスに直接影響します。Replication Server が受け取るデータやコマンドが多いほど、必要な作業が増え、処理に時間がかかります。

また、データが多くなると、さらに多くのネットワーク・リソースが必要になります。Replication Agent には、この影響を最小限に抑えるために使用できるさまざまな設定オプションがあります。

- RSSD の使用。RSSD から複写定義を読み取ることによって、Replication Agent は複写定義で指定されたカラム順序でカラム・データを送信できます。これにより、Replication Server は、カラム情報を処理する前にソートを行う必要がなくなります。さらに、カラム名がデータとともに送信されないため、必要な情報のバイト数が減少します。
- 最少カラムの送信。テーブルで更新オペレーションを行うときに、一部のカラムのみを変更すればよい場合があります。変更されたカラムのみの更新前イメージと更新後イメージを送信することによって、Replication Agent が送信する情報が少なくなります。

  注意： レプリケート・データベースのデータにカスタム・ファンクション文字列が含まれる場合は、最少カラムを使用しないでください。
- バッチ・モード。Replication Agent は、Replication Server の限られた量の管理作業を行う LTL にトランザクションを「まとめる」必要があります。バッチ・モードでは、Replication Agent は同一の管理コマンド・セットに複数のコマンドをまとめることができるため、ネットワークと Replication Server によって生成および処理される LTL が全体的に削減されます。

  バッチ・モードに加えて、ほとんどの Replication Agent には “batch timeout” パラメータがあります。このパラメータを使用すると、指定された時間 Replication Agent が待機してもバッチを満たす他のトランザクションを受け取らなかった場合に、バッチの一部が Replication Server に送信されるようにすることができます。

注意： Replication Server のユーザ定義データ型 (UDD) 変換をカラム・レベルまたはクラス・レベルで使用している場合は、Replication Agent のバッチ・モードを使用しないでください。

- オリジン時間。Replication Server に送信される各トランザクションには、オリジン・キュー ID があります。このオリジン・キュー ID に、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た時間が含まれている場合があります。Replication Agent からオリジン時間が送信されなかった場合、処理作業は多少減りますが、Replication Server に送信される LTL の量は同じです。

LTL の出力に影響する Replication Agent の設定パラメータの詳細については、『Replication Agent 管理ガイド』を参照してください。

rs_ticket

一部の Replication Agent では、**rs_ticket** トランザクションを起動できます。

トランザクションは、Replication Server のパフォーマンス、モジュールのハートビート、複写の正常性、およびテーブルレベルのクワイスについてのデータを提供します。『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・コネクション

Replication Server は、プライマリ・データベースとレプリケート・データベースを識別する「コネクション」と、他の Replication Server を識別する「ルート」を使用して、複写システム内の他のコンポーネントを追跡します。

Replication Server は本来、Adaptive Server Enterprise データベースの複写を目的として設計されているので、Replication Server のコネクションの定義は、Sybase の標準である *<サーバ名>.<データベース名>* の形式に準拠しています。たとえば、`ASE1` という名前の Adaptive Server のデータベース `PUBS` への Replication Server コネクションの名前は `ASE1.PUBS` となります。

ASE 以外のプライマリ・データ・サーバに接続するために、Replication Server は、ASE 以外のプライマリ・データベースの代わりに、Replication Agent からコネクションを受け付けます。レプリケート・データベースの場合、Replication Server は、ECDA データベース・ゲートウェイに接続します。ECDA データベース・ゲートウェイは、続いて ASE 以外のレプリケート・データ・サーバに接続します。Oracle の場合、ExpressConnect を使用して Replication Server を直接レプリケート・データ・サーバに接続することもできます。Replication Agent、ECDA ゲートウェイ、および ExpressConnect はデータ・サーバではないので、これらのコンポーネントの Replication Server コネクションのプロパティは、データベース・サーバ・コネクションの場合とは意味が異なる場合があります。

1 つの Replication Server コネクションで、単方向または双方向のデータ・フローをサポートできます。データの受信は、Replication Agent ユーザ・スレッドを介した

Replication Server コネクションによって行われます。データの送信は、データ・サーバ・インタフェース (DSI) スレッドを介した Replication Server コネクションによって行われます。各 Replication Server コネクションでは、送信データ・フローのみ (DSI スレッド経由)、または受信および送信データ・フロー (Replication Agent User スレッドおよび DSI スレッド経由) をサポートできます。

Replication Agent ユーザ・スレッド

Replication Server は、複写されるすべてのデータ変更オペレーションまたはトランザクションを、プライマリ・データ・サーバから、そのデータ・サーバのデータベース・コネクションの Replication Agent User スレッドを介して受け取ります。

複写されるトランザクションを提供するすべてのプライマリ・データベースが、有効な Replication Agent ユーザ・スレッドを持つ Replication Server データベース・コネクションによって表されている必要があります。

プライマリ・データベースが Adaptive Server 内にある場合、Replication Server はそのデータベースと直接コネクションを確立します。プライマリ・データベースが ASE 以外のデータ・サーバにある場合は、プライマリ・データベースに代わって、別の Replication Agent コンポーネントが Replication Agent ユーザ・スレッド・コネクションを使用して Replication Server と通信します。

注意： Replication Server は、コネクションの Replication Agent ユーザ・スレッドには接続しません。Replication Agent ユーザ・スレッドとの通信を開始できるエンティティは、プライマリ・データ・サーバと Replication Agent のみです。

Replication Agent ユーザ・スレッドでは、プライマリ・データ・サーバまたは Replication Agent がクライアント、プライマリ Replication Server がサーバです。

DSI スレッド

Replication Server コネクションの DSI スレッドは、Replication Server によって複写トランザクションが配信される場所にあります。

複写トランザクションを受け取ることが予想されるすべてのレプリケート・データベースが、有効な DSI スレッドを持つ Replication Server コネクションによって表されている必要があります。

レプリケート・データベースが Adaptive Server 内にある場合、Replication Server はそのデータベースと直接コネクションを確立します。レプリケート・データベースが Sybase 以外のデータ・サーバにある場合は、Replication Server は次を使用して通信します。

- ECDA データベース・ゲートウェイ (コネクションの DSI スレッドを介して)、または、

- Oracle レプリケート・データベースと直接コネクションを確立する ExpressConnect

注意： レプリケート・データ・サーバまたはデータベース・ゲートウェイは、コネクションの DSI スレッドには接続しません。DSI スレッドとの通信を開始できるエンティティは、Replication Server だけです。

DSI スレッドでは、Replication Server がクライアント、レプリケート・データ・サーバまたはデータベース・ゲートウェイがサーバです。

メンテナンス・ユーザの目的

メンテナンス・ユーザは複写テーブルでローの挿入、削除、および更新を行い、複写ストアド・プロシージャを実行します。データベース所有者 (またはシステム管理者) は、メンテナンス・ユーザがこれらの作業を実行するために必要なパーミッションを付与する必要があります。

複写データを更新する場合、Replication Server はレプリケート・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。Adaptive Server レプリケート・データベースでは、Sybase Central または **rs_init** によって Replication Server メンテナンス・ユーザのユーザ ID が自動的に作成され、レプリケート・データベースにユーザが追加されます。

メンテナンス・ユーザ ID とパスワードは、レプリケート・データベースに対する Replication Server の **create connection** コマンドによって、Replication Server で自動的に定義されます。データ・サーバでメンテナンス・ユーザ ID のパスワードを変更する場合、Sybase Central、または Replication Server の **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できます。

また、Replication Server メンテナンス・ユーザには、`rs_lastcommit` および `rs_info` システム・テーブルと、それらのテーブルを使用するすべてのストアド・プロシージャにアクセスするためのパーミッションが必要です。

Sybase Central と **rs_init** はどちらも、ユーザ・テーブルとストアド・プロシージャに対するデータベース・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複写ストアド・プロシージャに対するデータベース・パーミッションを付与しておく必要があります。データベースで複写されるテーブル、および複写のために実行されるストアド・プロシージャごとに、次のコマンドを実行してください。

```
grant all on table_name to maint_user
```

また、すべてのレプリケート・オブジェクトに対する必要な権限がデータベース管理者ロールにある場合、メンテナンス・ユーザ ID (*maint_user*) をこのロールに割り当てることもできます。

DDL ユーザの目的

Replication Agent for Microsoft SQL Server や Replication Agent for Oracle では、プライマリ・データベースで入力された DDL コマンドをサブスクライバ・データベースに複写できます。

この機能は、プライマリ・データ・サーバとレプリケート・データ・サーバが同じである場合 (たとえば、Oracle と Oracle) にのみサポートされます。詳細については、『Replication Agent 管理ガイド』を参照してください。

データ型、データ型定義、および制限付きデータ型

特定のデータ・サーバのデータ型のデータ型定義は、データ型クラスにまとめられます。

データ型定義 (ユーザ定義データ型) の詳細については、『Replication Server リファレンス・マニュアル』の「Replication Server システム・テーブル」の「`rs_datatype`」を参照してください。

制限付きデータ型

`rs_address` データ型は、カラム・レベル変換やクラス・レベル変換の変換元データ型および変換先データ型として使用できません。

ASE 以外のデータ・サーバのエラーとファンクション文字列クラス

Sybase では、サポートされている ASE 以外のすべてのレプリケート・データ・サーバについて、ファンクション文字列クラスとそれに関連付けられたファンクション文字列を提供しています。

ASE 以外のエラー・クラスは Replication Server によって作成され、さまざまな ASE 以外のエラー・クラスに対してエラー・アクションが定義されます。適切な接続プロファイルを使用することにより、対応するエラー・クラスを含む 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。

オブジェクトのパブリケーションとサブスクリプションの制限

Sybase 複写システムでのオブジェクトのパブリケーションとサブスクリプションの制限について説明します。

制限事項を次に示します。

- ASE 以外のプライマリ・データベースの複写定義でカラムを宣言するときには、プライマリ・データベースのカラムのデータ型と一致する Replication Server データ型を使用します。一致する Replication Server のネイティブ・デー

タ型がない場合は、プライマリ・データベースのデータ型と一致するデータ型定義を探します。
- カラム・レベル変換に関連するカラムに基づいた述部 **where** 句を使用してサブスクリプションを作成する場合、「宣言された」(つまり変換前の) 形式で述部の値を指定します。

Replication Agent

Replication Agent は、Sybase 複写システム内で ASE 以外のデータ・サーバをプライマリ・データ・サーバとしてサポートすることによって Replication Server の機能を拡張します。

Replication Agent はプライマリ・データに加えられた変更を検出し、RCL (Replication Control Language) のサブセットであるログ転送言語 (LTL) を使用して、プライマリ・データの変更内容をプライマリ Replication Server に送信します。

Replication Agent の動作

Replication Agent は、プライマリ・データベースのトランザクション・ログから情報を取得し、プライマリ Replication Server に対応するようにフォーマットする Replication Server クライアントです。

最初に、Replication Agent 内で必要なプライマリ・テーブルとストアド・プロシージャを複写用にマーク付けします。

Replication Agent は、次のように動作します。

1. Replication Server にログインします。
2. **connect source** コマンドを送信してセッションをログ転送元として特定し、トランザクション情報が転送されるデータベースを指定します。
3. Replication Server から、データベースのメンテナンス・ユーザ名を取得します。
4. Replication Server にデータベースのセカンダリ・トランケーション・ポイントを要求します。
5. トランザクション・ログから、レコードを (セカンダリ・トランケーション・ポイントに続くレコードを起点として) 取得し、情報をログ転送言語 (LTL) コマンドにフォーマットします。

Replication Agent コネクション

Replication Agent は、Replication Server にデータを送信します。Replication Agent は Replication Server にログインして、Replication Server コネクションの Replication Agent ユーザ・スレッドに接続し、そのコネクション上で Replication Server と通信します。

次に、Replication Agent コネクションの影響を示します。

- Replication Server にログインするために Replication Agent が使用する有効なユーザ ID が、Replication Server で定義されている必要があります。
- Replication Agent ユーザ ID に、Replication Server での **connect source** パーミッションが付与されている必要があります。**connect source** パーミッションによって、Replication Agent は Replication Agent User スレッドのみで有効なコマンドを送信できます。
- Replication Agent は、このユーザ ID とそのパスワードを記録する必要があります。
- Replication Agent は、正しい Replication Agent ユーザ・スレッドを特定して接続するために、Replication Server コネクション定義のサーバおよびデータベース部分を記録する必要があります。
- **user_name** と **password** (Replication Server の **create connection** コマンドで定義) は、プライマリ・データベースで有効なユーザ ID とパスワードである場合があります。

  注意： Replication Agent for Oracle、Microsoft SQL Server、および IBM DB2 UDB for UNIX には有効な **user_name** と **password** が指定され、プライマリ・データベースで該当のユーザが見つからない場合は、エラーを報告する必要があります。

  Replication Agent は、コネクションの **user_name** がプライマリ・データベースにあるかどうかを検証します。しかし、Replication Server は、DSI スレッドが使用されるかどうか (またはその時期) を認識していません。そのため、DSI スレッドがアクティブな場合は、ユーザ ID とパスワードが有効でなければなりません。

  注意： 有効なプライマリ・データベースのユーザ ID の必要条件は、Replication Agent によって異なります。一部の Replication Agent は、Replication Server コネクション上の有効なユーザ ID を必要としません (チェックも行いません)。

Interfaces ファイル

Replication Agent と Replication Server 間の対話のために必要なものは Replication Server を識別する `interfaces` ファイルのエントリだけです。

Replication Agent for DB2 UDB には、`interfaces` ファイルは必要ありません。Replication Server と RSSD のロケーションは、必要に応じて `LTMCFG` ファイル内に記録されます。

Replication Agent (UNIX および Windows プラットフォーム上の DB2 UDB 用、Microsoft SQL Server 用、Oracle 用) には、`interfaces` ファイル・エントリは必要ありません。これは、設定パラメータに Replication Server のホスト名とポート番号が記録されるためです。

Replication Agent メンテナンス・ユーザの処理

Replication Agent が Replication Server コネクションに接続する際に、Replication Agent はメンテナンス・ユーザ ID を要求し、プライマリ・データベースにそのユーザ ID があるかどうかを検証する場合があります。

この検証では、Replication Server コネクションで定義されているメンテナンス・ユーザ ID が、そのコネクションが表すデータベースに対して有効でなければなりません。そのコネクションがプライマリ・トランザクションのみ、レプリケート・トランザクションのみ、またはその両方のどれに使用されるかは関係ありません。

Replication Agent は、プライマリ・データベースにログインするときにメンテナンス・ユーザ ID を使用しません。ユーザ ID があるかどうかを検証する以外に、Replication Agent がメンテナンス・ユーザ ID に対して行う唯一の参照は、メンテナンス・ユーザによって作成されたプライマリ・データベース・トランザクションをフィルタすることです。

Replication Agent は、プライマリ・データベースに 1 つのトランザクションが何回も適用されないようにするために、メンテナンス・ユーザのトランザクションをフィルタします。双方向の複写スキームでは、同じデータベース (プライマリとレプリケートの両方の役割を果たす可能性があります) に対して双方向に複写を実行できます。プライマリ・トランザクションがレプリケート・データベースに適用される場合、適用するユーザ ID はレプリケート・データベースのメンテナンス・ユーザです。レプリケート・データベースでトランザクションをスキャンする Replication Agent は、Replication Server メンテナンス・ユーザによって適用されたトランザクションを無視することで、それらのトランザクションが再度送信され、プライマリ・データベースに適用されないようにします。

Replication Agent は、プライマリ・データベースで定義されているユーザ ID (DB2 の場合は DB2 ログ・ファイルにアクセスできるユーザ ID) を使用してデータベースにアクセスします。このユーザ ID は、Replication Server コネクションで定義されているメンテナンス・ユーザとは異なります。プライマリ・データベースへのアクセスに使用される Replication Agent ユーザ ID には、複写トランザクションを適用するために定義されたメンテナンス・ユーザとは異なる役割と目的があります。

また、Replication Agent には、Replication Agent を管理するために使用される別のユーザ ID が定義されている場合もあります。このユーザ ID も、レプリケート・トランザクションを適用する Replication Server メンテナンス・ユーザとは異なります。

Replication Agent では、次の 3 つのユーザを使用できます。

- Replication Agent がプライマリ・データ・サーバにログインし、プライマリ複写オブジェクトの操作やデータベース・トランザクション・ログの読み取りを行うために使用する、プライマリ・データベースで定義されるユーザ ID。
- Replication Agent にログインし、Replication Agent コマンドの発行や Replication Agent パラメータの設定を行うことができるユーザ ID。
- プライマリ・データベースで定義され、プライマリ Replication Server コネクションに記録されるメンテナンス・ユーザ ID。Replication Agent は、Replication Server に代わってこのユーザ ID を検証します。また、このユーザ ID によって作成されるトランザクションを無視するように、Replication Agent を設定できます。

DDL ユーザの処理

DDL ユーザは、DDL 複写を使用できる場合にプライマリ・データベースで定義されます。

このユーザ名は、Replication Agent によって送信されるすべての DDL コマンドに LTL で含められます。Replication Server の DSI スレッドは、このユーザ名を使用して DDL をレプリケート・データベースに適用します。

ASE 以外の Replication Agent

Sybase では、Replication Agent for DB2 UDB、Replication Agent など、ASE 以外の Replication Agent を提供しています。

Replication Agent for DB2 UDB

Replication Agent for DB2 UDB は、IBM z/OS プラットフォームで実行されている DB2 UDB サーバに、プライマリ・データ・サーバのサポートを提供します。

Replication Agent for DB2 UDB 製品は、複写システムで次のように機能します。

- プライマリ・データ・サーバは DB2 UDB で、IBM z/OS においてサブシステムとして実行される。トランザクション・ログは DB2 ログ。
- Replication Agent for DB2 UDB は、IBM z/OS では、開始されたタスクまたはジョブとして実行される。DB2 ログを読み込んで、1 つ以上の DB2 サブシステム用に複写するようにマーク付けされているテーブルについて、関連するアクティブな DB2 ログ・エントリとアーカイブ・ログ・エントリを取得する。このデータを、TCP/IP 通信プロトコルを使用して Replication Server に転送する。

DB2 データ・サーバは、DB2 テーブル内のローが変更されると、そのすべての変更をログに記録します。トランザクション・ログに書き込まれる情報には、変更前後のデータのコピーが含まれます。DB2 では、これらのレコードは “undo” レコードおよび “redo” レコードとして使用されます。制御レコードは、**commits** と

aborts 用に書き込まれます。これらのレコードは、**commit** オペレーションと **rollback** オペレーションに変換されます。

DB2 のログは、一連のデータ・セットで構成されます。Sybase Log Extract は、これらのデータ・セットを使用して DB2 データの変更内容を識別します。DB2 は、変更が行われると変更レコードをアクティブなログに書き込むので、Sybase Log Extract は、変更レコードが入力された直後にログ・レコードを処理できます。

Replication Agent

Replication Agent は、Linux、UNIX、Microsoft Windows プラットフォーム上の DB2 UDB、Microsoft SQL Server、または Oracle プライマリ・データベースのデータベース・トランザクション・ログを読み取る製品です。

Replication Agent は、Java プログラミング言語で実装されています。Replication Agent をインストールすると、Replication Agent ホスト・マシンとして指定されているコンピュータに Java Runtime Environment (JRE) がインストールされます。

Replication Agent は、すべての通信に JDBC (Java Database Connectivity) プロトコルを使用します。Replication Agent は、Sybase JDBC ドライバ (jConnect™ for JDBC™) の 1 つのインスタンスを使用して、Open Client および Open Server アプリケーション (プライマリ Replication Server を含む) へのすべてのコネクションを管理します。プライマリ・データ・サーバの場合、Replication Agent は、適切な JDBC ドライバを使用してプライマリ・データベースに接続します。

Enterprise Connect Data Access

ECDA (Enterprise Connect Data Access) 製品は、DB-Library™ および CT-Library アプリケーション・プログラミング・インタフェース (API)、JDBC および ODBC (Open Database Connectivity) プロトコルをサポートする、Open Server ベースのソフトウェア・ゲートウェイです。

ECDA 製品は、メインフレームおよび LAN ベースの ASE 以外のデータ・ソースへのアクセスを可能にするデータベース・ミドルウェア・アプリケーションの基本要素となります。

ECDA 製品は、以下のサービスを提供します。

- ASE 以外のデータ・ソースへのアクセスを提供するアクセス・サービス
- サーバ側のシステム管理を (DirectConnect Manager を介して) 提供する管理サービス

注意： Replication Server Options 15.5 では、ExpressConnect for Oracle でも Oracle データ・サーバに複写できます。

ECDA の動作

すべての Sybase ECDA オプションは、ASE 以外のデータ・サービスへの基本的なコネクティビティを実現します。特に、アクセス管理、コピー管理、リモート・システム管理を提供します。

各 ECDA オプションは、1 つの DirectConnect サーバおよび 1 つ以上のアクセス・サービス・ライブラリで構成されています。サーバは、サービス・ライブラリが動作するフレームワークを提供します。各アクセス・サービス・ライブラリは、このサーバから DB2 UDB、Microsoft SQL Server、Oracle などの特定のターゲット・データベースのデータにアクセスします。

各アクセス・サービス・ライブラリには、設定プロパティの特定のセットである 1 つ以上のアクセス・サービスが格納されています。アクセス・サービスは、Replication Server とターゲット・データベース間でデータを転送します。

DirectConnect サーバは、言語イベントや RPC などの受信クライアント・コネクションを受信して検証し、許可します。これらのイベントは、データ型変換、ネットワークのコネクティビティ、SQL 変換など、ターゲット固有のコネクティビティ機能を実現するアクセス・サービスを介して、ターゲット・データ・ソース (レプリケート・データベース) にルート指定されます。

Interface ファイル

`interface` ファイルには、ラベルのリストが格納されています。ラベルとは通常、それぞれ対応するホスト名とポート番号を持つサーバ名のことで、サーバ名で表されたサーバは、そのホストとポートでログイン要求を「受信」します。

Replication Server は、Open Server アプリケーションです。別の Open Server アプリケーションの場所 (ホストとポート番号) を特定する方法としては、ファイル内の場所を探す方法が推奨されます。

ECDA データベース・ゲートウェイと Replication Server との対話では、`interface` ファイルが重要です。Replication Server は Replication Server コネクションにおいてサーバ名で表されたサービスにログインしようとするので、そのサービス名が Replication Server の `interfaces` ファイル内になければなりません。また、その `interface` ファイル・エントリは ECDA ゲートウェイ設定ファイルのエントリにもサービス名として存在している必要があります。

1 つの ECDA は、1 つまたは複数の異なるデータベース・インスタンスのゲートウェイとして機能できます。ECDA の設定では、ECDA がアクセスする各データベースは、ユニークなサービス名として設定されます。設定されているサービス名のうち、どのサービス名に接続するかを Replication Server が認識できるように、クライアントはログイン時に渡された server name を使用し、コネクションの確立

に使用する、一致するサービス名を特定します。コネクションが `interfaces` ファイル・エントリと一致している必要があります。Microsoft SQL Server の場合は、データベース名がそのサービスに対して有効なデータベースである必要があります。サービス名の役割とその設定の詳細については、『ECDA Access Service ユーザーズ・ガイド』を参照してください。

Replication Agent と ECDA が共有するコネクション

ECDA ゲートウェイと Replication Agent は、それぞれ別のスレッドで Replication Server に接続するので、1 つの Replication Server コネクションでこれらの両方のコンポーネントをサポートできます。

同じデータベースに対して双方向で情報を複写する場合、データベース・ゲートウェイと Replication Agent の両方に共通のコネクションがあれば、複写システムのネットワーク・トポロジで使用されるリソースを削減することができます。

プライマリとレプリケートの両方の役割を果たすデータベースへの Replication Server コネクションを作成するには、次のように、ECDA データベース・ゲートウェイを適切にサポートするようにコネクションを定義してから、Replication Agent を適切に設定します。

- Replication Server で、**create connection** コマンドを使用して、コネクションの **server_name** と **database_name** を定義します。**server_name** 値は、ECDA で設定されたサービス名と一致している必要があります。
- Replication Agent で、**rs_source_ds** パラメータの値をその **server_name** に、**rs_source_db** パラメータの値を目的の **database_name** に設定します。

ECDA データベース・ゲートウェイ

ECDA データベース・ゲートウェイは、Replication Server からのトランザクションを Sybase 複写システムの ASE 以外のレプリケート・データベースに適用します。

そのために Replication Server は、Replication Server コネクションについて指定された情報を使用して ECDA ゲートウェイにログインします。Replication Server は **user_name** と **password** を使用してサーバにログインし、コネクションで定義されたデータベースに対して **use database** コマンドを発行します。

Replication Server の場合、ECDA ゲートウェイを Adaptive Server レプリケート・データベースと区別するものは何もありません。Replication Server は、通信する DSI スレッドから同じコマンドを配信し、同じ結果を想定します。

これには、以下のような意味があります。

- レプリケート・データベースにログインするために Replication Server が使用する有効なユーザ ID が、Replication Server コネクションで定義されている必要があります。

- このユーザ ID には、レプリケート・テーブルを更新するパーミッションと、レプリケート・プロシージャを実行するパーミッションが付与されている必要があります。
- レプリケート・データベースは RS_LASTCOMMIT テーブルと RS_TICKET_HISTORY テーブルを管理し、**rs_get_lastcommit** 機能をサポートできなければなりません。

  Replication Server には、DB2 UDB、Microsoft SQL Server、Oracle データベースでレプリケート・データベースに必要なテーブルとファンクションを設定するためのサンプル接続プロファイルが用意されています。

  レプリケート・データ・サーバおよびゲートウェイの条件の概要については、『Replication Server デザイン・ガイド』の「Adaptive Server 以外のデータ・サーバへのデータの複写」を参照してください。
- データ型の表現が、レプリケート・データベースのネイティブ・データ型と一致するように変換される必要があります。Replication Server には、DB2 UDB、Microsoft SQL Server、Oracle データ・サーバへの複写をサポートするために必要なファンクション文字列、ファンクション文字列クラス、基本データ型定義、変換を設定するためのサンプル接続プロファイルが用意されています。
- Replication Server の **resume connection** コマンドは、指定されたコネクションの DSI スレッドを使用してアクティビティを開始しようとします。ECDA の場合、これは DirectConnect サーバにログインし、レプリケート・データベース内の RS_LASTCOMMIT テーブルにアクセスした後、レプリケート・データベースにトランザクションを適用するプロセスです。このプロセスに失敗すると、Replication Server ログにエラーとして記録されます。

ECDA Option for ODBC

ECDA Option for ODBC は、DB2 UDB、Microsoft SQL Server、ODBC アクセス可能なデータベースに対する Open Client インタフェースを Replication Server に提供します。

注意： ECDA Option for ODBC の ODBC ドライバ (ターゲットに接続するバックエンド・ドライバ) は、Sybase からは提供されません。各自で入手してインストールと設定を行う必要があります。

ECDA Option for ODBC は、IBM DB2 や Microsoft SQL Server などのターゲット・データベースに合わせて各自で取得した ODBC バックエンド (サーバ側) ドライバを使用して、ASE 以外のデータ・ソースに対するアクセスを実現します。ODBC ドライバをベンダの指示に従って ECDA Option for ODBC と同じサーバにインストールしてから、この ODBC ドライバを使用してデータベースにアクセスするように ECDA Option for ODBC を設定します。

注意： 取得した ODBC ドライバに Sybase のドライバ・マネージャ・ソフトウェアとの互換性があるか、または取得した ODBC ドライバにドライバ・マネージャが含まれていることを確認してください。

ODBC ドライバの機能は広範囲に渡っているので、ASE の提供ではないサード・パーティ製の ODBC ドライバで処理を行う場合には、必要な条件を満たしていることを確認するために統合とテストを注意深く実施する必要があります。

ECDA Option for Oracle

ECDA Option for Oracle は、Oracle データベースに対する Open Client インタフェースを Replication Server に提供します。

ECDA Option for Oracle は、Replication Server からは Oracle SQL を認識する Open Server アプリケーションとして見えます。

注意： ExpressConnect for Oracle を使用しても Oracle データ・サーバを複写できます。ExpressConnect for Oracle は、Replication Server と Oracle 間で直接対話を提供します。

参照：

- ExpressConnect for Oracle (49 ページ)

Mainframe Connect DirectConnect for z/OS Option

Mainframe Connect DirectConnect for z/OS Option は、メインフレーム上で実行されている DB2 に対する Open Client インタフェースを Replication Server に提供します。

ExpressConnect for Oracle

ExpressConnect for Oracle (ECO) は、Oracle を複写するための Replication Server 15.5 以降によってロードされるライブラリです。

ECO は、ECDA で次の利点があります。

- 起動、モニタリング、または管理のために、個別のサーバ・プロセスは必要ありません。
- Replication Server と ECO は同一のプロセス内で実行されるため、これらの間で SSL は不要であり、ECDA for Oracle のグローバル設定パラメータでこれまでに扱われていた設定内容を設定するための要件もありません。
- サーバ接続は Replication Server から **create connection** コマンドと **alter connection** コマンドを使用して設定されるため、ECDA for Oracle の **connect_string** 設定で

個別に同等の設定をする必要はありません。『Replication Server リファレンス・マニュアル』を参照してください。

- `text_chunksize`、`autocommit`、`array_size` など、ECDA for Oracle と同等の特定の設定も設定する必要がありません。これらの設定は Replication Server によって (Replication Agent 入力に基づく場合もあります) 自動的に決められて ECO と通信します。

ECO には、次に示すように ECDA for Oracle と類似した機能が備えられています。

- データ型変換のセットが同じ。
- Sybase データと Oracle データ間での言語および charset の変換。ECO では、`map.cfg` ファイルを使用して設定される。
- ASE プライマリ・データベース内の空の文字列が Oracle レプリケート・データベースに複写されると、Oracle で、1 つまたは複数 (カラムが `varchar` と `fixed char width` のいずれのデータ型であるかによって異なる) の空白で構成される文字列値になる。

ExpressConnect for Oracle でロケーションの透過性を確立するには、`tnsnames.ora` ファイルのみが必要です。ECDA for Oracle のように `interfaces` ファイルは必要ではありません。コネクションの設定のためには、`tnsnames.ora` ファイルで定義されるサービス名を指定する必要があります。

ECO の詳細については、ExpressConnect for Oracle インストールおよび設定ガイドを参照してください。

プライマリ・データ・サーバとしての IBM DB2 for z/OS

Sybase 複写システムにおける、IBM z/OS プラットフォーム上の DB2 UDB サーバに固有のプライマリ・データ・サーバの問題と考慮事項について説明します。

Replication Agent for DB2 UDB

プライマリ・データ・サーバとしての DB2 UDB は、Replication Agent for DB2 UDB と対話します。

Replication Agent for DB2 UDB を使用する際には、次の点を考慮してください。

- Replication Agent はデータ変更オペレーションまたはトランザクションに関する情報を識別し、DB2 UDB プライマリ・データベースからプライマリ Replication Server に転送します。
- Replication Agent は、プライマリ Replication Server と対話します。また、プライマリ Replication Server の RSSD と対話するように設定されている場合は、RSSD とも対話します。

複写の干渉と影響

Replication Agent DB2 ライブラリは、許可プログラム機能 (APF) によって許可されている必要があります。

Sybase 複写システムにおける、DB2 UDB プライマリ・データ・サーバのパフォーマンスと動作は、以下の影響を受ける場合があります。

- DB2 UDB トランザクション・ログは、次のように影響を受けます。
  - 複写には、変更される各ローの更新前イメージと更新後イメージが必要です。プライマリ・テーブルを複写するようマーク付けした場合、テーブルは **DATA CAPTURE CHANGES** 句によって変更されます。複写するようマーク付けされたテーブルの数が増えると、DB2 UDB アクティブ・ログ・データ・セットに必要な DASD 領域も増加します。
  - Replication Agent for DB2 UDB を使用すると、DB2 UDB ログに格納されるデータの量が増加します。増加の規模は、プライマリ・テーブルの数、種類、サイズと、複写されるトランザクションの種類によって異なります。たとえば、**update** トランザクションには更新前イメージと更新後イメージの両方が必要であり、これには、変更されていないカラムも含め、ロー内

のすべてのカラムが含まれます。詳細については、Replication Agent for DB2 UDB のマニュアルを参照してください。

- Replication Agent をインストールすると、次の 2 つの Replication Agent システム・テーブルがプライマリ DB2 UDB に作成されます。
  - `LTMOBJECTS` は、複写するようマーク付けされた各プライマリ・テーブルのローを格納します。サイズは、複写するようマーク付けされたテーブルの数によって異なります。
  - 更新された `LTMMARKER` を使用すると、マテリアライゼーション処理に役立ちます。
- Replication Agent for DB2 UDB で開始されたタスクは、単一の DB2 サブシステムのログ、または DB2 データ共有グループ内のすべてのログを処理できます。この動作は、**LTMCFG** パラメータである **DataSharingOption**、**DataSharingMember**、**Log_identifier**、および **BSDS** によって制御されます。
- プライマリ・データベースの制限事項：
  - LOB 複写がサポートされていません。
  - `char` と `varchar` の最大サイズが 32767 です。
  - DDL とストアド・プロシージャの複写がサポートされていません。
- 次の DB2 UDB ユーティリティは使用しないでください。使用すると、複写の整合性が失われる可能性があります。
  - **LOAD LOG NO**
  - **RECOVER**
  - **REORG with RECOVER**
- **rs_ticket** を Replication Agent DB2 UDB で起動することはできません。Replication Server 15.5 で、**rs_ticket** を Replication Agent DB2 コネクションに「挿入」することはできません。『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**sysadmin issue_ticket**」を参照してください。

DB2 UDB プライマリ・データベースのパーミッション

LTM for z/OS の **LTM_process_maint_uid_trans** 設定パラメータが **Y** でないかぎり、メンテナンス・ユーザによってプライマリ・データベースに適用される更新は無視されます。

次の 2 つのユーザ ID を作成します。

- LTMADMIN ユーザ ― 以下を行う TSO ユーザ。必要に応じて LTMADMIN という名前が付けられます。
  - Replication Agent for DB2 UDB のインストール、起動、停止
  - DB2 UDB 上の Replication Agent システム・テーブルの管理

  LTMADMIN ユーザには、複写するようマーク付けされたすべての DB2 UDB テーブルに対する **ALTER TABLE** 権限が必要です。このユーザ ID は、複写する

ようマーク付けされたプライマリ・テーブルで **ALTER TABLE DATA CAPTURE CHANGES** コマンドを発行します。
また、LTMADMIN ユーザには、DB2 UDB ログ・ファイルに対する **TRACE**、**DISPLAY**、**MONITOR2** の各パーミッションも必要です。

- Replication Server メンテナンス・ユーザ ― Replication Server の **create connection** コマンドで、プライマリ・データベースについて指定されたユーザ ID。

プライマリ・データ・サーバの接続

IBM z/OS 環境でプライマリ DB2 UDB データ・サーバに接続するには、Replication Agent for DB2 UDB に、IBM z/OS に対して定義され、正しい DB2 UDB プランおよびパッケージに対する execute パーミッションが付与されている有効なユーザ ID が必要です。

Replication Agent for DB2 UDB は、このユーザ ID を使用して DB2 UDB にログインします。

Replication Agent for DB2 UDB ジョブの JCL (Job Control Language) が、正しいアカウント、ユーザ ID、DB2 UDB ログ、DB2 UDB サブシステム・ライブラリを使用して実行するように変更されている必要があります。

Replication Server の接続

Replication Agent for DB2 UDB では、Replication Server に接続するために `interface` ファイルを使用しません。Replication Server に接続するために必要な情報は、`LTMCFG` ファイルに含まれています。

Replication Manager を使用して複写オブジェクトを作成しないかぎり、Replication Server の `interface` ファイルに Replication Agent for DB2 のエントリを記述する必要はありません。

Replication Server システム・データベースの接続

Replication Agent for DB2 UDB では、Replication Server System Database (RSSD) へのアクセスは不要です。

ただし、RSSD を使用することにより、Replication Agent for DB2 UDB と Replication Server 間のデータの量を減らすことができます。

LTMCFG パラメータの **Use_repdef** が Y の場合、Replication Agent for DB2 UDB の起動時に複写定義がロードされます。その複写定義を変更する場合は、

Replication Agent が変更を認識できるように Replication Agent を停止し、再起動します。

RSSD に接続するために必要な情報は `LTMCFG` ファイルで提供されます。パラメータはすべて RSSD で始まり、すべてのパラメータを入力する必要があります。ただし、これらは、**Use_repdef** が **N** に設定されている場合は検証されません。

DB2 UDB プライマリ・データベースの設定

Replication Agent は、単一の DB2 サブシステム、または DB2 データ共有グループ内のすべてのログに対して実行できます。LTMCFG パラメータでは、Replication Agent for DB2 UDB の DB2 環境 (**DataSharingOption**、**DataSharing Member**、**Log-identifier**、**BSDS**) を記述します。

Replication Agent for DB2 UDB はメインフレーム z/OS のアプリケーションであり、単一の z/OS アドレス空間で同時に実行される、次の 2 つのタスクで構成されています。

- *Sybase Log Extract* － プライマリ・テーブルのデータ変更オペレーションに関する DB2 UDB のアクティブ・ログとアーカイブ・ログを連続してスキャンします。
- *Replication Agent for DB2 UDB for z/OS* － Sybase Log Extract から複写トランザクションを受け取り、ログ転送言語 (LTL) に変換した後、プライマリ Replication Server に送信します。

DataSharingOption が Multi の場合、Replication Agent for DB2 UDB は **Boot Strap Dataset** (BSDS) パラメータを参照してデータ共有グループ内の各 DB2 メンバの **BSDS** を識別し、Replication Agent for DB2 UDB の位置とグループのメンバごとの DB2 ログを表示します。

Replication Agent のインストールと設定の問題はすべて、『Replication Agent for DB2 UDB Installation Guide』に記載されています。ただし、異機種間複写システムでは次のようになります。

- **rs_source_ds** パラメータと **rs_source_db** パラメータの値では、大文字と小文字が区別されます。Replication Agent と Replication Server の両方のパラメータで大文字と小文字が正しく指定されていない場合は、コネクションに失敗します。
- Replication Agent for DB2 UDB for z/OS の **LTM_process_maint_uid_trans** 設定パラメータは、メンテナンス・ユーザによって実行されたトランザクションを Replication Agent がプライマリ Replication Server に送信するかどうかを制御します。
  双方向複写環境 (同一の DB2 UDB 領域が複写元と複写先となる) では、**LTM_process_maint_uid_trans** パラメータの値を **N** に設定する必要があります。

true に設定していないと、別のサイトに複写されたトランザクションが元のサイトに戻って適用され、無限ループが生成される可能性があります。

DB2 for z/OS におけるプライマリ・テーブルの複写定義

Replication Agent for DB2 UDB for z/OS の **Use_repdef** 設定パラメータは、複写定義で指定されたカラムのみを格納するログ転送言語 (LTL) と、DB2 UDB プライマリ・テーブル内のすべてのカラムを格納するログ転送言語のどちらを Replication Agent が送信するかを制御します。

Use_repdef パラメータの値が **N** に設定されている場合、Replication Agent は DB2 UDB プライマリ・テーブル内のすべてのカラムのデータを含む LTL を送信します。**Use_repdef** パラメータの値が **Y** に設定されている場合は、Replication Agent は、複写定義で指定されたカラムのデータのみを含む LTL を送信します。

複写定義に必要なカラムのデータのみを送信すると、ネットワーク・トラフィックが軽減されるので、パフォーマンスが向上することがあります。

Use_repdef の値を **Y** に設定すると、他のパラメータ (**suppress_col_names**) などを使用して、Replication Agent のパフォーマンスを向上させることができます。『Replication Agent for DB2 UDB Installation Guide』を参照してください。

LTL_table_col_case パラメータは、Replication Agent が Replication Server にテーブル名とカラム名を送信するときの大文字と小文字の区別を制御します。DB2 でのデフォルトは大文字です。ただし、このパラメータを使用して、テーブル名とカラム名を大文字または小文字に変更したり、DB2 で定義されている名前を保持したりできます。

テーブルの名前が Replication Server またはターゲット・データベースの予約語と競合する可能性があります。**with primary table named** 句や **with replicate table named** 句を使用すると、テーブル名を予約できます。ただし、`LTMOBJECTS` テーブルで REPLICATE_NAME オプションを使用すると、Replication Agent for DB2 は、LTL を Replication Server に送信する前にテーブル名を変更します。『Replication Agent for DB2 UDB User and Troubleshooting Guide』の「Replication Agent Setup」の「DB2 Source Table Considerations」の「DB2 Table Names and Reserved Keywords」を参照してください。

DB2 for z/OS プライマリ・データ型の変換

Replication Agent for DB2 UDB for z/OS の **Date_in_char**、**Time_in_char**、**Timestamp_in_char** の各設定パラメータは、Replication Agent が値を文字列として送信するか、Sybase の `datetime` フォーマットに変換するかを制御します。

これらのパラメータの詳細については、『Replication Agent for DB2 UDB Users and Troubleshooting Guide』を参照してください。

注意： 日付または時間に関連するユーザ定義データ型 (UDD) を複写定義で使用する場合は、プライマリ・データベースにとってネイティブなフォーマットで Replication Server にデータを送信するように、Replication Agent を設定することをおすすめします。また、Replication Agent でデータ型変換を実行しないことをおすすめします。

通常、Replication Agent for DB2 UDB はデータ型変換を行いません。ただし、すべてのレプリケート・データ・サーバで同じ変換が必要な場合は、処理時間を節約するために、各レプリケート・データベースの DSI で変換を行うのではなく、Replication Agent で一度に変換を行うようおすすめします。

IBM DB2 UDB では、午前 0 時は 24.00 として表されます。このフォーマットは、他のデータ・サーバと互換性がない場合があります。データ型定義を変更すると、この値を自動的に 24.00 から 00.00 に変更できます。

IBM DB2 UDB では、他のデータ・サーバに対応してない可能性がある年の値を使用できます。IBM DB2 UDB で許容される過去の年がレプリケート・データ・サーバで使用できない場合は、LTMCFG パラメータの *Minimum_year* を設定し、DB2 UDB Replication Agent で、*Minimum_year* パラメータよりも過去の年を *Date_conv_default* パラメータに変更できるようにします。

文字セット

DB2 内のデータは、複数の文字セットを使用してコード化できます。また、Replication Agent for DB2 を使用して、文字セットが Replication Server に送信される前に、複写された文字を Replication Server の文字セットに変換することもできます。

Replication Agent DB2 の文字セット・プロパティを制御するパラメータは、**codepage** と **RS_ccsid** です。これらのパラメータの詳細については、『Replication Agent for DB2 UDB Installation Guide』の「LTM for MVS Configuration Parameters」を参照してください。

マテリアライゼーション

マテリアライゼーションとは、プライマリ・データベースからのデータのコピーでレプリケート・データベースを初めて移植する処理のことです。

DB2 データを使用してターゲットをマテリアライズするには、Replication Agent for DB2 UDB を使用します。DB2 のアンロード・ユーティリティは、データ・ファイルと、そのデータを記述したパンチカード・ファイルを生成します。これらのファイルを Replication Agent for DB2 UDB のマテリアライゼーション機能への入力として使用することにより、複写のターゲットを初期化できます。

『Replication Agent for DB2 UDB User and Troubleshooting Guide』の「Replication Server Setup」の「Task 3:Materializing Replicate Tables」の「Using Replication Agent Materialization」を参照してください。

プライマリ・データ・サーバとしての IBM DB2 for Linux, UNIX, および Windows

Sybase 複写システムにおける、UNIX、Windows、Linux プラットフォーム上の DB2 UDB サーバに固有のプライマリ・データベースの問題と考慮事項について説明します。

Replication Agent for UDB

プライマリ・データ・サーバとしての DB2 UDB は、Replication Agent と対話します。DB2 UDB に対応するように設定された Replication Agent のインスタンスは、「Replication Agent for UDB」と呼ばれます。

Replication Agent for UDB はデータ変更オペレーションまたはトランザクションに関する情報を識別し、DB2 UDB プライマリ・データ・サーバからプライマリ Replication Server に転送します。

注意： トランザクションの複写元のデータベースごとに、個別の Replication Agent for UDB インスタンスが必要です。

Replication Agent は、プライマリ Replication Server と対話します。また、プライマリ Replication Server の RSSD と対話するように設定されている場合は、RSSD とも対話します。

注意： Replication Agent は Java プログラムです。オペレーティング・システムによっては、Java をサポートするためのパッチが必要な場合があります。『Replication Agent 管理ガイド』と『Replication Agent リリース・ノート』を参照してください。

DB2 UDB システムの管理

Replication Agent には、プライマリ・データベースに関するメタデータ情報 (データベース名、テーブル名、プロシージャ名、カラム名など) を返す多くのコマンドがあります。

メタ情報を返すには、メタ情報を返すように設計された特別な JDBC 呼び出しを発行するか、システム・テーブルに直接問い合わせます。

Replication Manager の制限事項

Replication Manager プラグインでは、プライマリ DB2 UDB データ・サーバでの Replication Agent インスタンスの起動も停止もできません。

Replication Agent インスタンスの起動と停止の詳細については、『Replication Agent 管理ガイド』を参照してください。

DB2 UDB での複写の干渉と影響

Sybase 複写システムにおける、DB2 UDB プライマリ・データ・サーバのパフォーマンスと動作は、トランザクション・ログの影響を受ける場合があります。

- **LOGARCHMETH1** 設定パラメータを **LOGRETAIN** または **DISK**:<*path*> に設定する必要があります。ここで、<*path*> は、ログのアーカイブ先のディレクトリです。現在の **LOGARCHMETH1** 設定を指定するには、次の UDB コマンドを使用します。
  ```
  get db cfg for <db-alias>
  ```
- 複写には、変更される各ローの更新前イメージと更新後イメージが必要です。プライマリ・テーブルを複写するようマークを付けた場合、Replication Agent for UDB はテーブルの **DATA CAPTURE** オプションを **DATA CAPTURE CHANGES** に設定します。複写するようマーク付けされたテーブルの数が増えると、DB2 UDB トランザクション・ログに必要な領域も増加します。
- プライマリ・データベースには、ページ・サイズが 8KB 以上の、テンポラリ・ユーザ・システムが管理するテーブル領域が必要です。

DB2 UDB プライマリ・データベースのパーミッションと制限

Replication Agent for UDB には、プライマリ・データベース内のデータにアクセスし、新しいオブジェクトを作成するためのパーミッションを持つ、DB2 UDB ログインが必要です。

プライマリ・データベース・トランザクション・ログにアクセスするには、DB2 UDB のログインに SYSADM または DBADM の権限が必要です。

Replication Agent では、DB2 UDB のストアド・プロシージャまたは DDL の複写はサポートされていません。『Replication Agent プライマリ・データベース・ガイド』を参照してください。

プライマリ・データ・サーバの接続

プライマリ DB2 データ・サーバに接続するには、Replication Agent for UDB でいくつかのタスクを実行する必要があります。

Replication Agent for UDB が DB2 UDB サーバと異なるホスト・マシンにインストールされている場合、Replication Agent ホスト・マシンに DB2 UDB Administration Client をインストールしてください。

Replication Agent for UDB ソフトウェアが DB2 UDB サーバと同じホスト・マシンにインストールされている場合、個別の DB2 UDB Administration Client は必要ありません。

Windows システムでは、Replication Agent for UDB のコネクティビティを設定するときは、DB2 UDB Administration Client で ODBC データ・ソースを設定してから、その ODBC データ・ソースについて指定されているデータベース名とデータベース・エイリアスを使用します。

UNIX システムでは、ODBC を使用する代わりに、UDB のノードとプライマリ・データベースをカタログします。次に、プライマリ・データベースのカタログ時に指定されたデータベース・エイリアスを使用して、データ・ソース Replication Agent の設定パラメータを設定します。

接続の設定方法の詳細については、『Replication Agent インストール・ガイド』の「Replication Agent のインストール」の「プライマリ・データベースのコネクティビティ」を参照してください。

設定する必要がある Replication Agent 設定パラメータの詳細については、『Replication Agent インストール・ガイド』の「インストール前の準備」を参照してください。

Replication Server と RSSD のコネクティビティ

Replication Agent は、TCP/IP と Sybase JDBC ドライバ (Replication Agent インストールに含まれている jConnect for JDBC) を使用して、他の Sybase サーバと通信します。Replication Agent は、コネクティビティ情報を得るのに Sybase `interfaces` ファイルを使用しません。

Replication Agent がプライマリ Replication Server に接続できるようにするために設定する必要がある Replication Agent 設定パラメータについては、『Replication Agent インストール・ガイド』の「インストール前の準備」を参照してください。

Replication Agent オブジェクト

pdb_xlog init を使用して Replication Agent を初期化すると、複写をサポートするオブジェクトがプライマリ・データベースに作成されます。

詳細については、『Replication Agent プライマリ・データベース・ガイド』を参照してください。

Replication Agent for UDB により、`SYBRAUJAR.jar` と `SYBTRUNCJAR.jar` が次のディレクトリにインストールされます。

- Windows では、ファイルは `%DB2DIR%¥SQLLIB¥FUNCTION¥jar ¥`*pds_username* にインストールされます。`%DB2DIR%` は UDB のインストール・パス、*pds_username* は Replication Agent 設定パラメータ *pds_username* によって指定されたプライマリ・データベース・ユーザ名です。
- UNIX では、ファイルは `$HOME/sqllib/function/jar/`*pds_username* にインストールされます。`$HOME` は UDB インスタンス所有者のホーム・ディレクトリ、*pds_username* は Replication Agent 設定パラメータ *pds_username* によって指定されたプライマリ・データベース・ユーザ名です。

これらの Jar ファイルにより、UDB プライマリ・データベースにいくつかの Java プロシージャが実装されます。「トランケーション用の Java プロシージャ」の表は、Replication Agent の初期化時に作成され、ログのトランケーションに使用される Java プロシージャのリストです。

注意： 複数の Replication Agent インスタンスが 1 つの UDB サーバ・インストール (トランザクションが複写される各データベースのインストール) に設定されている場合、各 Replication Agent インスタンスは *pds_username* 設定パラメータで異なるプライマリ・データベース・ユーザ名を指定する必要があります。

トランケーション用の Java プロシージャ

Replication Agent の初期化時に作成され、ログのトランケーションに使用される Java プロシージャをリストします。

プロシージャ	データベース名
現在の LSN が格納されたログ・ファイルの名前を取得する	*prefix*get_log_name_
get_log_name Java クラスのバージョンを取得する	*prefix*get_version_str_
データベース・ログ・ファイルまたはアーカイブ・ログ・ディレクトリのファイルをトランケートする	*prefix*trunc_log_files_

プロシージャ	データベース名
trunc_log_files Java クラスのバージョンを取得する	*prefix*get_trunc_ver_str_

複写オブジェクトの実際の名前の取得

Replication Agent インスタンスによって生成される Replication Agent データベース・オブジェクトの名前を確認します。

Replication Agent 管理ポートで、キーワードを指定せずに **pdb_xlog** コマンドを呼び出します。

```
pdb_xlog
```

pdb_xlog コマンドは、Replication Agent によってプライマリ・データベースに作成されたオブジェクトのリストを返します。

DB2 UDB プライマリ・データベースの設定

異機種間複写に固有の追加の問題について説明します。

プライマリ DB2 UDB データ・サーバのインストールの問題と設定パラメータの詳細はすべて、『Replication Agent for DB2 UDB Installation Guide』に記載されています。

Java Runtime Environment

Replication Agent をインストールするときに、Replication Agent for UDB と互換性がある Java Runtime Environment (JRE) がインストールされます。

Java Runtime Environment に関する特記事項については、『Replication Agent リリース・ノート』を確認してください。

rs_source_ds 設定パラメータと rs_source_db 設定パラメータ

Replication Agent 設定ファイル内の設定パラメータ値はすべて、大文字と小文字を区別します。

Replication Server も大文字と小文字を区別するので、**rs_source_ds** および **rs_source_db** パラメータの値を指定するときには注意が必要です。Replication Agent と Replication Server の両方のパラメータで、大文字と小文字が正しく指定されていない場合は接続できません。

filter_maint_userid 設定パラメータ

Replication Agent の **filter_maint_userid** 設定パラメータは、メンテナンス・ユーザが実行したトランザクションを、Replication Agent がプライマリ Replication Server に転送するかどうかを制御します。

メンテナンス・ユーザの名前は、Replication Server の **create connection** で、プライマリ・データベースに対して定義されます。

双方向複写環境 (同一のデータベースが複写元と複写先になる) では、**filter_maint_userid** パラメータの値を **true** に設定します。true に設定していないと、別のサイトに複写されたトランザクションが元のサイトに戻って適用され、無限ループが生成される可能性があります。

ltl_character_case 設定パラメータ

Replication Agent の **ltl_character_case** 設定パラメータは、Replication Agent がプライマリ Replication Server にデータベース・オブジェクト名を送信するときの大文字と小文字の区別を制御します。

たとえば、複写定義が `all tables named testtab` に対して作成される場合、Replication Agent が送信するテーブル名は `testtab` でなければなりません。そうでない場合、不一致が起こります。Replication Server は大文字と小文字を区別するので、値 `TESTTAB` は値 `testtab` と一致しません。

複写定義を作成する場合は、デフォルト文字を選択し (たとえば、すべて大文字またはすべて小文字で複写定義を作成する)、Replication Agent の **ltl_character_case** パラメータの値を変更することで、大文字と小文字を一致させます。

大文字で格納されるオブジェクト名

DB2 UDB では、オブジェクトの作成時に大文字と小文字の区別が割り当てられていなければ、オブジェクト名はデフォルトですべて大文字で格納されます。そのため、特に設定されていないかぎり、Replication Agent はプライマリ Replication Server に大文字でオブジェクト名を送信します。

ltl_character_case パラメータの詳細については、『Replication Agent 管理ガイド』を参照してください。

DB2 UDB におけるプライマリ・テーブルの複写定義

Replication Agent の **use_rssd** 設定パラメータは、複写定義で指定されたカラムのみを格納するログ転送言語 (LTL) と、プライマリ・テーブル内のすべてのカラムを格納する LTL のどちらを Replication Agent が送信するかを制御します。

use_rssd パラメータの値が **false** の場合、Replication Agent はプライマリ・テーブル内のすべてのカラムのデータを含む LTL を送信します。**use_rssd** パラメータの値が **true** の場合は、Replication Agent は、各プライマリ・テーブルの複写定義で指定されたカラムのデータのみを含む LTL を送信します。

複写定義で指定されたカラムのデータのみを送信すると、ネットワーク・トラフィックが軽減されるので、パフォーマンスが向上することがあります。

また、Replication Agent は複写定義で指定した順番で情報を送信できるので、カラム名とパラメータ名は LTL から削除されます。LTL の **minimal columns** および **structured tokens** オプションは、**use_rssd** パラメータの値が **true** の場合に使用できます。詳細については、『Replication Agent 管理ガイド』を参照してください。

DB2 UDB プライマリ・データ型の変換

Replication Agent では、DB2 UDB の `DATE`、`TIME`、および `TIMESTAMP` カラム値を Replication Server に送信する方法を制御できます。

DB2 UDB データ型マッピングの完全なリストについては、『Replication Agent プライマリ・データベース・ガイド』の「Replication Agent for UDB」の「IBM DB2 固有の考慮事項」の「DB2 データ型の互換性」を参照してください。

プライマリ・データ・サーバとしての Microsoft SQL Server

Sybase 複写システムにおける、Microsoft SQL Server データ・サーバに固有のプライマリ・データベースの問題について説明します。

注意： Replication Agent for Microsoft SQL Server は Microsoft Windows にインストールする必要があります。

Replication Agent for Microsoft SQL Server

プライマリ・データ・サーバとしての Microsoft SQL Server は、Replication Agent と対話します。Replication Agent が Microsoft Windows にインストールされ、Microsoft SQL データベース・ログに直接アクセスできる必要があります。

Replication Agent はデータ変更オペレーションまたはトランザクションに関する情報を識別し、Microsoft SQL Server プライマリ・データベースからプライマリ Replication Server に転送します。

注意： トランザクションの複写元のデータベースごとに、個別の Replication Agent インスタンスが必要です。

Replication Agent は、プライマリ Replication Server と対話します。また、プライマリ Replication Server の RSSD と対話するように設定されている場合は、RSSD とも対話します。

sybfilter ドライバ

Replication Agent でデータを複写するには、sybfilter ドライバを使用して Microsoft SQL Server のログ・ファイルを読み込み可能にする必要があります。

Replication Agent は Microsoft SQL Server のログ・ファイルを読み込むことができる必要があります。ただし、Microsoft SQL Server プロセスは排他的な読み込みパーミッションを使用してこれらのログ・ファイルを開くので、Replication Agent を含む他のプロセスからこのファイルを読み込むことはできません。『Replication Agent プライマリ・データベース・ガイド』を参照してください。

Microsoft SQL Server システムの管理

Replication Agent には、プライマリ・データベースに関するメタデータ情報 (データベース名、テーブル名、プロシージャ名、カラム名など) を返す多くのコマンドがあります。

メタ情報を返すには、メタ情報を返すように設計された特別な JDBC 呼び出しを発行するか、システム・テーブルに直接問い合わせます。

Replication Manager

Replication Manager プラグインでは、Microsoft SQL Server プライマリ・データ・サーバでの Replication Agent インスタンスの起動も停止もできません。

Replication Agent インスタンスの起動と停止の詳細については、『Replication Agent 管理ガイド』を参照してください。

Replication Agent のパーミッション

Microsoft SQL Server にログインするために Replication Agent インスタンスが使用するユーザ ID は、プライマリ・データベースにアクセスできる必要があります。

Replication Agent for Microsoft SQL Server では、複写作業用のデータベース・オブジェクトをプライマリ・データベースで作成します。自動的に付与される必要なパーミッションのリストについては、『Replication Agent プライマリ・データベース・ガイド』を参照してください。

プライマリ・データ・サーバの接続

Replication Agent では、プライマリ・データベースと通信するために JDBC ドライバが必要です。Microsoft SQL Server データベース用の JDBC ドライバはサード・パーティ・データベース・ベンダから提供されています。

使用するデータベース用の JDBC ドライバがまだインストールされていない場合は、ベンダの Web サイトから適切なドライバを入手してください。Microsoft SQL Server の JDBC の最新バージョンについては、『Replication Agent リリース・ノート』を参照してください。

CLASSPATH 環境変数の設定

CLASSPATH 環境変数を設定する方法について説明します。

1. Replication Agent があるホスト・マシンまたは Replication Agent がアクセスできるホスト・マシンに JDBC ドライバをインストールします。
2. JDBC ドライバのロケーションを CLASSPATH 環境変数に追加します。

   [スタート] > [設定] > [コントロール パネル] > [システム] > [環境] を選択し、パス・セパレータとしてセミコロン (;) を使用して、次のロケーションを既存の CLASSPATH 環境変数に追加します。または、[ユーザ環境変数] パネルでパスを作成します。

   ```
   drive:¥path_name¥driver
   ```

   構文の説明は次のとおりです。
   - *drive* は、ドライブ文字です。
   - *path_name* は、JDBC ドライバをインストールした場所です。
   - *driver* は、JDBC ドライバの名前です。Microsoft SQL Server の場合、この名前は `sqljdbc.jar` です。
3. [適用] をクリックし、次に [OK] をクリックします。

設定する必要がある Replication Agent 設定パラメータについては、『Replication Agent インストール・ガイド』の「インストール前の準備」を参照してください。

Replication Server と RSSD のコネクティビティ

Replication Agent は、TCP/IP と Sybase JDBC ドライバ (Replication Agent インストールに含まれている jConnect for JDBC) を使用して、他の Sybase サーバと通信します。Replication Agent は、コネクティビティ情報を得るのに Sybase `interfaces` ファイルを使用しません。

Replication Agent がプライマリ Replication Server に接続できるようにするために設定する必要がある Replication Agent 設定パラメータについては、『Replication Agent インストール・ガイド』の「インストール前の準備」を参照してください。

Replication Agent オブジェクト

Replication Agent は、複写作業用のオブジェクトをプライマリ・データベースで作成します。

Replication Agent オブジェクトは、**pdb_xlog** コマンドを **init** キーワードを指定して呼び出すと自動的に作成されます。既存のプライマリ・データベース・オブジェクトには、複写のマークを付けることができます。

一般的な情報については、『Replication Agent 管理ガイド』を参照してください。

Replication Agent のデータベース・オブジェクト名には、次の 2 つの変数があります。

- *prefix* ― **pdb_xlog_prefix** パラメータの 1 ～ 3 文字の文字列値を表します (デフォルトは **ra**_)。
- *xxx* ― 英数字のカウンタを表します。この文字列は、データベース内でユニークな名前にするためにデータベース・オブジェクト名に追加されます。

pdb_xlog_prefix パラメータの値は、すべての Replication Agent オブジェクト名で使用されるプレフィクス文字列です。**pdb_xlog_prefix_chars** パラメータの値は、**pdb_xlog_prefix** で指定するプレフィクス文字列で使用できる非英数字のリストです。使用できる文字のリストはデータベース固有です。たとえば、Microsoft SQL Server では、データベース・オブジェクト名に使用できる非英数字は $、#、@、_ のみです。

プライマリ・データベース内の Replication Agent トランザクション・ログ・コンポーネントの名前を表示するには、**pdb_xlog** コマンドを使用します。

ログ・オブジェクト名を設定する方法の詳細については、『Replication Agent 管理ガイド』を参照してください。

テーブル、プロシージャ、マーカ、およびトリガ・オブジェクト

Replication Agent のトリガ・オブジェクトと見なされるコマンドに加え、Replication Agent のオブジェクトと見なされるテーブルおよびプロシージャ・オブジェクト、マーカ・プロシージャおよびマーカ隠しテーブルは、『Replication Agent プライマリ・データベース・ガイド』にリストされています。

insert パーミッションと delete パーミッションは、データベース名 prefixddl_trig_xxx の DDL 隠しテーブルでのみ PUBLIC に付与されます。他のテーブルに対するパーミッションは与えられません。

sp_SybSetLogforReplTable プロシージャと **sp_SybSetLogforReplProc** プロシージャは、Microsoft SQL Server の `mssqlsystemresource` システム・データベースに

作成されます。これらのプロシージャの execute パーミッションは PUBLIC に付与されますが、Replication Agent の *pds_username* ユーザだけがプロシージャを正常に実行できます。これは、*pds_username* ユーザだけが `sys.sysschobjs` テーブルに対する **select** パーミッションを付与されているためです。他のプロシージャの作成時に付与されるパーミッションはありません。

Microsoft SQL Server プライマリ・データベースの設定

異機種間複写に固有の追加の問題について説明します。

Microsoft SQL Server プライマリ・データ・サーバのインストールの問題と設定パラメータの詳細については、すべて『Replication Agent インストール・ガイド』に記載されています。

rs_source_ds 設定パラメータと rs_source_db 設定パラメータ

Replication Agent 設定ファイル内の設定パラメータ値はすべて、大文字と小文字を区別します。

Replication Server も大文字と小文字を区別するので、**rs_source_ds** および **rs_source_db** パラメータの値を指定するときには注意が必要です。Replication Agent と Replication Server の両方のパラメータで、大文字と小文字が正しく指定されていない場合は接続できません。

filter_maint_userid 設定パラメータ

maint_user として sysadmin 権限を持つ Microsoft SQL Server ログインを使用する場合、そのログインを対応するデータベースのユーザにマップします。そうしないと、この **maint_user** によって実行されたトランザクションを Replication Agent が正しくフィルタできません。

ltl_character_case 設定パラメータ

Replication Agent の **ltl_character_case** 設定パラメータは、Replication Agent がプライマリ Replication Server にデータベース・オブジェクト名を送信するときの大文字と小文字の区別を制御します。

たとえば、複写定義が `all tables named testtab` に対して作成される場合、Replication Agent が送信するテーブル名は `testtab` でなければなりません。そうでない場合、不一致が起こります。Replication Server は大文字と小文字を区別するので、値 `TESTTAB` は値 `testtab` と一致しません。

複写定義を作成する場合は、デフォルト文字を選択し (たとえば、すべて大文字またはすべて小文字で複写定義を作成する)、Replication Agent の **ltl_character_case** パラメータの値を変更することで、大文字と小文字を一致させます。

以下は、データベースの作成時に指定した照合によって異なります。Microsoft SQL Server データベースでは、オブジェクトの作成時に大文字と小文字の区別が割り当てられていなければ、オブジェクト名はデフォルトで小文字で格納されます。特に設定されていないかぎり、Replication Agent はプライマリ Replication Server に小文字でオブジェクト名を送信します。

ltl_character_case パラメータの詳細については、『Replication Agent 管理ガイド』を参照してください。

Microsoft SQL Server におけるプライマリ・テーブルの複写定義

複写定義で指定されたカラムのデータのみを送信すると、ネットワーク・トラフィックが軽減されるので、パフォーマンスが向上することがあります。

Replication Agent の **use_rssd** 設定パラメータは、複写定義で指定されたカラムのみを格納するログ転送言語 (LTL) と、プライマリ・テーブル内のすべてのカラムを格納する LTL のどちらを Replication Agent が送信するかを次のように制御します。

- **use_rssd** パラメータの値が **false** の場合、Replication Agent はプライマリ・テーブル内のすべてのカラムのデータを含む LTL を送信します。
- **use_rssd** パラメータの値が **true** の場合は、Replication Agent は、各プライマリ・テーブルの複写定義で指定されたカラムのデータのみを含む LTL を送信します。

また、Replication Agent は複写定義で指定した順番で情報を送信できるので、カラム名とパラメータ名は LTL から削除されます。LTL の **minimal columns** および **structured tokens** オプションは、**use_rssd** パラメータの値が **true** の場合に使用できます。『Replication Agent 管理ガイド』を参照してください。

複写定義を変更するには、『Replication Server 管理ガイド 第 1 巻』の「複写テーブルの管理」の「複写定義の修正」の「複写定義の変更」の「複写定義の変更要求プロセス」を参照してください。

Microsoft SQL Server プライマリ・データ型の変換

Microsoft SQL Server のデータ型はすべて、対応する Adaptive Server のデータ型と互換性があります。

`varchar(max)`、`nvarchar(max)`、`varbinary(max)` の各データ型は、Microsoft SQL Server 以外のデータベースに複製することはできません。

プライマリ・データ・サーバとしての Oracle

Sybase 複写システムにおける、Oracle データ・サーバに固有のプライマリ・データベースの問題と考慮事項について説明します。

Replication Agent for Oracle

プライマリ・データ・サーバとしての Oracle は、Replication Agent と対話します。Replication Agent はデータ変更オペレーション (またはトランザクション) に関する情報を識別し、Oracle プライマリ・データ・サーバからプライマリ Replication Server に転送します。

注意： トランザクションの複写元の Oracle データベースごとに、個別の Replication Agent インスタンスが必要です。

Replication Agent は、プライマリ Replication Server と対話します。また、プライマリ Replication Server の RSSD と対話するように設定されている場合は、RSSD とも対話します。

注意： Replication Agent は Java プログラムです。オペレーティング・システムによっては、Java をサポートするためのパッチが必要な場合があります。詳細については、『Replication Agent 管理ガイド』および『Replication Agent リリース・ノート』を参照してください。

Oracle におけるプライマリ・テーブルの複写定義

複写定義で指定されたカラムのデータのみを送信すると、ネットワーク・トラフィックが軽減されるので、パフォーマンスが向上することがあります。

Replication Agent の **use_rssd** 設定パラメータは、複写定義で指定されたカラムのみを格納するログ転送言語 (LTL) と、プライマリ・テーブル内のすべてのカラムを格納する LTL のどちらを Replication Agent が送信するかを制御します。

use_rssd パラメータの値が **false** の場合、Replication Agent はプライマリ・テーブル内のすべてのカラムのデータを含む LTL を送信します。**use_rssd** パラメータの値が **true** の場合は、Replication Agent は、各プライマリ・テーブルの複写定義で指定されたカラムのデータのみを含む LTL を送信します。

また、Replication Agent は複写定義で指定した順番で情報を送信できるので、カラム名とパラメータ名は LTL から削除されます。LTL の **minimal columns** および

structured tokens オプションは、**use_rssd** パラメータの値が **true** の場合に使用できます。『Replication Agent 管理ガイド』を参照してください。

複写定義を変更するには、『Replication Server 管理ガイド 第 1 巻』の「複写テーブルの管理」の「複写定義の修正」の「複写定義の変更」の「複写定義の変更要求プロセス」を参照してください。

Replication Manager の制限事項

Replication Manager プラグインでは、Oracle プライマリ・データ・サーバでの Replication Agent インスタンスの起動も停止もできません。

Replication Agent インスタンスの起動と停止の詳細については、『Replication Agent 管理ガイド』を参照してください。

Oracle システムの管理

Replication Agent には、プライマリ・データベースに関するメタデータ情報 (データベース名、テーブル名、プロシージャ名、カラム名など) を返す多くのコマンドがあります。

メタ情報を返すには、メタ情報を返すように設計された特別な JDBC 呼び出しを発行するか、Oracle システム・テーブルに直接問い合わせます。

注意： Adaptive Server Enterprise とは異なり、Oracle は 1 つのサーバ・インスタンス内では複数のデータベースをサポートしません。

Oracle での複写の干渉と影響

Sybase 複写システムが組み込まれている場合、Oracle プライマリ・データ・サーバのパフォーマンスと動作が影響を受ける場合があります。

Replication Agent は Oracle オンライン・ログとアーカイブ redo ログを読み込んでトランザクション情報を取得する一方で、特定のログ設定を必要とします。必要な情報を提供および管理するには、Oracle で以下の項目を有効にします。

- redo ログのアーカイブ
- プライマリ・キーとユニーク・インデックス・データの補足ロギング

さらに、Replication Agent は、Oracle redo ログに直接アクセスでき、プライマリ Oracle サーバと同じプラットフォームで実行されている必要があります。

Oracle プライマリ・データベースのパーミッション

Replication Agent には、プライマリ・データベース内のデータにアクセスし、新しいオブジェクトを作成するパーミッションを持つ Oracle ログイン ID が必要です。

このような必要なパーミッションを持つ Oracle ログイン ID のリストについては、『Replication Agent プライマリ・データベース・ガイド』を参照してください。

注意： 必要なパーミッションに加えて、Replication Agent for Oracle インスタンスを起動するオペレーティング・システム・ユーザには、Oracle redo ログとアーカイブ・ログの **read** アクセス権が必要です。

プライマリ・データ・サーバの接続

Replication Agent では、Oracle プライマリ・データベースに接続するために JDBC ドライバが必要です。

JDBC ドライバは、Replication Agent ホスト・マシンにインストールされ、CLASSPATH システム変数で参照される必要があります。Java は、CLASSPATH システム変数の内容を参照して Java クラスの検索場所を特定します。Oracle JDBC ドライバの場合、次のように、CLASSPATH 変数に完全なパスとファイル名が含まれていなければなりません。

```
drive:¥<path_name>¥ojdbc14.jar
```

サポートされる JDBC ドライバのバージョンについては、『Replication Agent リリース・ノート』を参照してください。

JDBC のコネクティビティを確保するには、Oracle プライマリ・データ・サーバの TNS Listener プロセスが実行されている必要があります。

設定する必要がある Replication Agent 設定パラメータについては、『Replication Agent インストール・ガイド』の「インストール前の準備」を参照してください。

Replication Server と RSSD のコネクティビティ

Replication Agent は、TCP/IP と Sybase JDBC ドライバ (Replication Agent インストールに含まれている jConnect for JDBC) を使用して、他の Sybase サーバと通信します。Replication Agent は、コネクティビティ情報を得るのに Sybase `interfaces` ファイルを使用しません。

Replication Agent がプライマリ Replication Server に接続できるようにするために設定する必要がある Replication Agent 設定パラメータについては、『Replication Agent インストール・ガイド』の「インストール前の準備」を参照してください。

Replication Agent オブジェクト

Replication Agent は、複写作業用のオブジェクトをプライマリ・データベースで作成します。

Replication Agent のデータベース・オブジェクト名には、次の 2 つの変数があります。

- *prefix* ― **pdb_xlog_prefix** パラメータの 1 ～ 3 文字の文字列値を表します (デフォルトは **ra**_)。
- *xxx* ― 英数字のカウンタを表します。この文字列は、データベース内でユニークな名前にするためにデータベース・オブジェクト名に追加されます。

pdb_xlog_prefix パラメータの値は、**rs_marker** と **rs_dump** を除き、すべての Replication Agent オブジェクト名で使用されるプレフィクス文字列です。

pdb_xlog_prefix_chars パラメータの値は、**pdb_xlog_prefix** で指定するプレフィクス文字列で使用できる非英数字のリストです。使用できる文字のリストはデータベース固有です。たとえば、データベース・オブジェクト名に使用できる非英数字は $、#、_ のみです。

プライマリ・データベース内の Replication Agent トランザクション・ログ・コンポーネントの名前を表示するには、**pdb_xlog** コマンドを使用します。

オブジェクト名を設定する方法の詳細については、『Replication Agent 管理ガイド』を参照してください。

作成されたオブジェクトの名前を調べるには、Replication Agent 管理ポートで、キーワードを指定せずに **pdb_xlog** コマンドを呼び出します。

```
pdb_xlog
```

pdb_xlog コマンドは、すべての Replication Agent オブジェクトのリストを返します。

Oracle プライマリ・データベースの設定

異機種間複写に固有の追加の問題について説明します。

Oracle プライマリ・データ・サーバのインストールの問題と設定パラメータの詳細はすべて、『Replication Agent インストール・ガイド』に記載されています。

Java Runtime Environment

Replication Agent をインストールするときに、Replication Agent と互換性がある Java Runtime Environment (JRE) がインストールされます。

Java Runtime Environment に関する特記事項については、『Replication Agent リリース・ノート』を参照してください。

必要な JDBC ドライバ

プライマリ・データ・サーバへのコネクティビティを確保するために、Replication Agent には JDBC ドライバが必要です。

Sybase では、Oracle データ・サーバ用の JDBC ドライバは提供していません。Oracle データ・サーバ用 JDBC ドライバの入手方法については、『Replication Agent リリース・ノート』を参照してください。

rs_source_ds 設定パラメータと rs_source_db 設定パラメータ

Replication Agent 設定ファイル内の設定パラメータ値はすべて、大文字と小文字を区別します。

Replication Server も大文字と小文字を区別するので、**rs_source_ds** および **rs_source_db** パラメータの値を指定するときには注意が必要です。Replication Agent と Replication Server の両方のパラメータで、大文字と小文字が正しく指定されていない場合は接続できません。

filter_maint_userid 設定パラメータ

Replication Agent の **filter_maint_userid** 設定パラメータは、メンテナンス・ユーザが実行したトランザクションを、Replication Agent がプライマリ Replication Server に転送するかどうかを制御します。

メンテナンス・ユーザの名前は、Replication Server の **create connection** で、プライマリ・データベースに対して定義されます。

双方向複写環境 (同一のデータベースが複写元と複写先になる) では、**filter_maint_userid** パラメータの値を **true** に設定します。true に設定していないと、別のサイトに複写されたトランザクションが元のサイトに戻って適用され、無限ループが生成される可能性があります。

ltl_character_case 設定パラメータ

Replication Agent の **ltl_character_case** 設定パラメータは、Replication Agent がプライマリ Replication Server にデータベース・オブジェクト名を送信するときの大文字と小文字の区別を制御します。

たとえば、複写定義が `all tables named testtab` に対して作成される場合、Replication Agent が送信するテーブル名は `testtab` でなければなりません。そうでない場合、不一致が起こります。Replication Server は大文字と小文字を区別するので、値 `TESTTAB` は値 `testtab` と一致しません。

複写定義を作成する場合は、デフォルト文字を選択し (たとえば、すべて大文字またはすべて小文字で複写定義を作成する)、Replication Agent の **ltl_character_case** パラメータの値を変更することで、大文字と小文字を一致させます。

Oracle データベースでは、オブジェクトの作成時に大文字と小文字の区別が強制されていなければ、オブジェクト名はデフォルトですべて大文字で格納されます。特に設定されていないかぎり、Replication Agent はプライマリ Replication Server に大文字でオブジェクト名を送信します。

ltl_character_case パラメータの詳細については、『Replication Agent 管理ガイド』を参照してください。

Oracle プライマリ・データ型の変換

データ型の変換とデータ型マッピングを実行すると、Oracle データ型のデータ型マッピングの完全なリストが表示されます。

UDD とその使用方法の詳細については、『Replication Server 管理ガイド 第 1 巻』を参照してください。

参照：

- データ型の変換とマッピング (247 ページ)

Automatic Storage Management

Replication Agent for Oracle では、オンライン用の Oracle Automatic Storage Management (ASM) 機能の使用、および redo ログのアーカイブがサポートされています。

ASM は、Oracle データベース環境に対してファイル・システム管理とボリューム管理をサポートします。ASM は、Real Application Cluster (RAC) 環境と RAC 以外の

環境の両方で使用できます。ASM には、RAID (Redundant Array of Independent Disks) または論理ボリューム・マネージャ (LVM) と同様の利点があります。

これらのテクノロジと同様に、ASM では一連の個別のディスクから 1 つのディスク・グループを定義できます。ASM は、ディスク・グループで定義されたすべてのデバイス間で負荷分散を行います。また、ストライピング機能とミラーリング機能も提供します。RAID または LVM と異なり、ASM だけが Oracle データベースで作成され、読み込まれたファイルをサポートします。ASM は汎用的なファイル・システムでは使用できず、バイナリ・ファイルやフラット・ファイルを保管できません。オペレーティング・システムは ASM ファイルにアクセスできません。

Oracle ASM 用の Replication Agent サポートの詳細については、『Replication Agent プライマリ・データベース・ガイド』を参照してください。

Real Application Cluster

Replication Agent は、Oracle 10g および 11g Real Application Cluster (RAC) 環境をサポートします。

Replication Agent for Oracle インスタンスを初期化すると、Oracle データベースに対して、クラスタによってサポートされるノード数を決定するための問い合わせが実行されます。この情報を基に、Replication Agent はすべてのノードから redo ログ情報を処理するように自動的に設定されます。

注意： RAC データベースの複写は、RAC 以外のデータベースの複写と同じです。

Oracle RAC クラスタ内にあるすべてのノードの redo ログ・データを処理するためには、Oracle ノードが各自の redo データの格納に使用する共有領域にアクセスできるロケーションから、Replication Agent を実行する必要があります。Replication Agent は、オンラインの redo ログとアーカイブされた redo ログの両方が存在する共有領域に対し、読み込みアクセスできる必要があります。

必要なホスト、ポート、Oracle SID の値を **pds_host_name**、**pds_port_number**、**pds_database_name** の各設定パラメータに指定することで、1 つの Oracle インスタンスに接続するように Replication Agent を設定できます。Oracle RAC 環境では、ノードで障害が発生したり、使用できない場合に備えて、Replication Agent はクラスタ内のすべてのノードに接続できる必要があります。

複数ノード・ロケーションを設定するため、Replication Agent は、1 つの指定したエントリに対して Oracle `tnsnames.ora` ファイルから必要な情報を取得して、必要なすべての RAC ノードへの接続をサポートします。これにより、すべてのノードに対して個々のホスト名、ポート名、インスタンス名を設定するのではなく、

`tnsnames.ora` ファイルの場所と使用する TNS 接続の名前のみが必要になります。

Oracle RAC 用の Replication Agent サポートの詳細については、『Replication Agent プライマリ・データベース・ガイド』を参照してください。

レプリケート・データ・サーバとしての IBM DB2 for z/OS

ASE 以外のデータ・サーバを含む場合に限った Sybase 複写システムに固有の管理作業について説明します。

複写システムの基本的な管理については、『Replication Server 管理ガイド 第 1 巻』を参照してください。

DB2 UDB for z/OS レプリケート・データ・サーバの環境

ゲートウェイ環境のレプリケート・データ・サーバとしての DB2 UDB for z/OS は、レプリケート Replication Server からのコマンドを受け入れ、それらのコマンドをレプリケート DB2 UDB データベースに適用する Mainframe Connect DirectConnect for z/OS Option データベース・ゲートウェイと対話します。

DB2 UDB for z/OS システム管理

Replication Server バージョン 12.0 における HDS の導入によって、**create connection** コマンドの **dsi_sql_data_style** パラメータは無効になりました。

create connection コマンドの **dsi_sql_data_style** パラメータは、DB2 UDB for z/OS レプリケート・データベースの一部のデータ変換をサポートするために、以前のバージョンの Replication Server で使用されていました。Replication Server バージョン 12.0 以降では、このパラメータを使用しないでください。デフォルト設定は、" " (空白) です。

注意： このシステム管理に関する問題は、レプリケート DB2 UDB for z/OS データ・サーバに固有です。

DB2 UDB for z/OS での複写の干渉と影響

レプリケート DB2 UDB への唯一の重大な干渉または影響は、Replication Server のオペレーションをサポートするために、レプリケート・データベースに 3 つの

テーブルを作成する接続プロファイルによって作成されるデータベース・オブジェクトです。

テーブルの内容は次のとおりです。

- レプリケート・データベースによって使用されるソート順と文字セットに関する情報を格納する RS_INFO

  注意： このテーブルに対する **INSERT** 文で、使用しているデータ・サーバに、適切な文字セットとソート順が指定されていることを確認してください。

  Replication Server バージョン 12.5 以降を使用する場合、レプリケート・データベースのソート順と文字セットは、RS_INFO テーブルに記録されなければなりません。そのためには、Replication Server の **rs_get_charset** および **rs_get_sortorder** ファンクションを使用して、レプリケート・データベース内の RS_INFO テーブルから文字セットとソート順を取得します。
- レプリケート・データベースに適用する複写トランザクションに関する情報を格納する RS_LASTCOMMIT

  RS_LASTCOMMIT テーブルの各ローは、プライマリ・データベースからレプリケート・データベースに配信されコミットされた、最後のトランザクションを示します。Replication Server は、この情報によってすべてのトランザクションが配信されたことを確認します。

  Replication Server の **rs_get_lastcommit** ファンクションは、レプリケート・データベース内の最後にコミットされたトランザクションに関する情報を取得します。ASE 以外のレプリケート・データベースについては、**rs_get_lastcommit** は、データベース固有のファンクション文字列クラスで、レプリケート・データベース内の RS_LASTCOMMIT テーブルへのアクセスに必要なクエリによって置き換えられます。
- RS_TICKET_HISTORY は、Replication Server コマンド **rs_ticket** の実行結果を格納します。プライマリ・データベースに対して **rs_ticket** コマンドを発行し、コマンドがプライマリ・データベースからレプリケート・データベースまで移動するために要する時間を測定できます。この情報を使用して、Replication Server のパフォーマンス、モジュールのハートビート、複写の正常性、テーブルレベルのクワイスをモニタできます。**rs_ticket** の各実行結果は、レプリケート・データベース内にある RS_TICKET_HISTORY テーブルの 1 つのローに保存されます。RS_TICKET_HISTORY テーブルの各ローを問い合わせると、個々の **rs_ticket** 実行結果を取得したり、別のローの結果と比較できます。このテーブルに保存されるデータは複写をサポートするために必要ではなく、手動でトランケートして領域を再利用できます。

  注意： RS_TICKET_HISTORY テーブルは、Replication Server 15.1 以降でのみ使用できます。

DB2 for z/OS レプリケート・データベースのパーミッション

レプリケート・データベースにトランザクションを適用するために、Replication Server では Replication Server の **create connection** コマンドで指定するメンテナンス・ユーザ ID が必要となります。

メンテナンス・ユーザ ID は、DB2 UDB for z/OS データ・サーバに対して定義され、レプリケート・データベースにトランザクションを適用する権限が付与されている必要があります。メンテナンス・ユーザ ID には、レプリケート DB2 UDB データベースにおける、次のパーミッションが必要です。

- Replication Server の処理に使用するテーブルを作成するための **CREATE TABLE** 権限
- すべてのレプリケート・テーブルに対する **UPDATE** 権限と、すべてのレプリケート・ストアド・プロシージャに対する **EXECUTE** 権限

DB2 UDB for z/OS に関するレプリケート・データベースのコネクティビティ

Replication Server データベース・コネクション名は、データ・サーバ名 (**server_name**) とデータベース名 (**db_name**) の 2 つの部分から構成されています。

Mainframe Connect DirectConnect for z/OS Option データベース・ゲートウェイを使用している場合、**server_name** はデータベース・ゲートウェイ・サーバの名前、**db_name** はレプリケート DB2 UDB データベースの名前です。

レプリケート Replication Server は、Replication Server データベース・コネクションで指定されたデータベース・ゲートウェイの **server_name** について、`interface` ファイル・エントリを検索します。レプリケート Replication Server は、データベース・コネクションで指定された **user_name** と **password** を使用して、レプリケート・データ・サーバにログインします。

Replication Server の `interfaces` ファイルにエントリを作成して、Mainframe Connect DirectConnect for z/OS Option データベース・ゲートウェイ・サーバが受信を行うホストとポートを指定してください。`interfaces` ファイル・エントリの名前は、Replication Server データベース・コネクションの **server_name** 部分と一致している必要があります。

DB2 for z/OS におけるレプリケート・データベースの制限事項

Mainframe Connect DirectConnect for z/OS Option では、LOB データ型 (`BLOB` と `CLOB`) の直接の複写がサポートされています。

また、Mainframe Connect DirectConnect for z/OS Option データベース・ゲートウェイによって、DB2 UDB のバイナリ値に対して ASCII から EBCDIC への変換が行われるため、Replication Server は値をバイナリ文字列として送信できません。そのため、DB2 UDB for z/OS に複写されるすべての `binary` または `varbinary` データ型を、`rs_db2_char_for_bit` または `rs_db2_varchar_for_bit` データ型にマップする必要があります。

DB2 for z/OS レプリケート・データベースの設定

Replication Server の異機種データ型サポート (HDS) 機能には、レプリケート Replication Server と DB2 UDB for z/OS レプリケート・データベースで HDS 機能を設定するために使用できる設定情報が用意されています。

この設定情報は、インストール・プロセスにおいて、接続プロファイルの一部として提供されます。

- Replication Server インストール
  - ファンクション文字列、エラー・クラス、ユーザ定義データ型の作成
- 接続プロファイル：
  - RSSD へのクラス・レベル・データ型変換の適用
  - DB2 UDB for z/OS でのオブジェクトの作成
  - 接続プロパティの設定
- 追加設定
  - ECDA の設定
  - 動的 SQL の設定
  - コマンドのバッチ処理の設定

参照：

- RSSD へのクラス・レベル・データ型変換 (88 ページ)
- DB2 UDB for z/OS でのオブジェクトおよび接続プロパティ (88 ページ)
- ECDA の設定 (89 ページ)
- 動的 SQL の設定 (89 ページ)

- コマンドのバッチ処理の設定 (89 ページ)

Replication Server インストール

Replication Server インストールでは、複写をサポートするために必要なファンクション文字列とクラスが自動的にインストールされます。

ファンクション文字列、エラー・クラス、ユーザ定義データ型

ファンクション文字列は、Replication Server のデフォルトの **rs_db2_function_class** に追加されます。

ファンクション文字列は、いくつかのデフォルト Replication Server ファンクション文字列を、DB2 UDB for z/OS レプリケート・データベースと通信し、作成されたテーブルとプロシージャにアクセスするように設計されたカスタム・ファンクション文字列に置き換えます。

接続プロファイル

接続プロファイルを使用すると、事前に定義されたプロパティのセットでコネクションを設定できます。

構文

```
create connection to data_server.database
using profile connection_profile;version
set username [to] user
[other_create_connection_options]
[display_only]
```

パラメータ

data_server – 複写システムに追加するデータベースを持つデータ・サーバです。

database – 複写システムに追加するデータベースです。

connection_profile – コネクションの設定、RSSD の修正、およびレプリケート・データベース・オブジェクトの作成に使用する接続プロファイルを示します。

version – 使用する接続プロファイルのバージョンを指定します。

user – データベースの Replication Server メンテナンス・ユーザのログイン名です。Replication Server は、複写データを管理するのにこのログイン名を使用します。ネットワークベース・セキュリティを有効に設定していない場合には、ユーザ名を指定する必要があります。

other_create_connection_options – プロファイルで指定されない接続オプションの設定 (パスワードの設定など)、またはプロファイルで指定されているオプションの上書き (Replication Server に用意されているファンクション文字列クラスを上書きするカスタム・ファンクション文字列クラスの指定など) を行うには、他の **create**

connection オプションを使用します。**create connection** コマンドの他の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「Replication Server コマンド」の「**create connection**」を参照してください。

display_only – **display_only** は、**using profile** 句とともに使用し、実行されるコマンド、およびそのコマンドを実行するサーバの名前を表示します。**display_only** を使用した結果については、クライアント・ログおよび Replication Server ログを参照してください。

RSSD へのクラス・レベル・データ型変換

クラス・レベル変換によって、プライマリ・データ型とレプリケート・データ型、およびデータの変換先のレプリケート・データ型が特定されます。

たとえば、Oracle の `DATE` は DB2 UDB レプリケート・データベースの `TIMESTAMP` に変換されます。

クラス・レベル変換は、以下に示す適切な名前の接続プロファイルによって DB2 UDB for z/OS レプリケート・データベースに提供されます。

- `rs_ase_to_db2` – Adaptive Server データ型を DB2 UDB データ型に変換します。
- `rs_udb_to_db2` – DB2 UDB (UNIX および Windows 版) データ型を DB2 UDB for z/OS データ型に変換します。
- `rs_msss_to_db2` – Microsoft SQL Server データ型を DB2 データ型に変換します。
- `rs_oracle_to_db2` – Oracle データ型を DB2 UDB データ型に変換します。
- `rs_db2_connection_sample` – DB2 データベースへのコネクションを作成します。(ECDA へのコネクション)。
  接続プロファイルは、Replication Server に付属している事前定義の DB2 UDB for z/OS ファンクション文字列クラスを使用して、レプリケート DB2 UDB for z/OS の Replication Server データベース・コネクションを作成するためのテンプレートを提供します。

DB2 UDB for z/OS でのオブジェクトおよび接続プロパティ

接続プロファイルによって、`RS_INFO`、`RS_LASTCOMMIT`、`RS_TICKET_HISTORY` の各テーブルがレプリケート・データベース内に作成されます。

また、次の接続プロパティも設定されます。

```
set error class rs_db2_error_class
set function string rs_db2_function_class
```

追加設定

複写をサポートするための追加設定について説明します。

設定内容は次のとおりです。

- ECDA の設定
- 動的 SQL の設定
- コマンドのバッチ処理の設定

ECDA の設定

ECDA および DirectConnect アクセス・サービス設定ファイルのプロパティの値について説明します。

ECDA 設定ファイルには、次の設定を使用します。

```
TransactionMode=long
Allocate=connect
SQLTransformation=sybase
```

DB2 UDB for z/OS レプリケート・データベースへの複写に Mainframe Connect DirectConnect for z/OS Option データベース・ゲートウェイを使用している場合は、DirectConnect の `db2.cfg` アクセス・サービス設定ファイルで、次のプロパティを設定します。

```
SQLTransformation=passthrough
TransactionMode=long
```

動的 SQL の設定

ECDA 12.6.1 およびそれ以降では、動的 SQL がサポートされています。

コマンドのバッチ処理の設定

コマンドのバッチ処理を使用すると、Replication Server は複数のコマンドを 1 つのコマンド・バッチとしてデータ・サーバに送信できます。

セミコロン (;) で区切ることによって、言語ファンクション文字列の出力テンプレートに複数のコマンドを入力できます。データベースがコマンド・バッチを使用できるように設定されている場合 (デフォルトの設定) は、Replication Server は、このセミコロンをそのコネクションの DSI コマンド・セパレータ文字に置き換えてから、単一のバッチ内のファンクション文字列としてデータ・サーバに送信します。

セパレータ文字は、**dsi_cmd_separator** オプション (**alter connection** コマンド) に定義されています。データベースへのコネクションがバッチを使用できるように設定されていない場合は、Replication Server は、ファンクション文字列内のコマンド

を一度に 1 つずつデータ・サーバに送信します。データベースのバッチを有効または無効にするには、**alter connection** コマンドを使用します。

コマンドのバッチ処理を使用するには、次のように入力します。

```
set batch = on

set dsi_cmd_separator = ;

set batch_begin = off

use_batch_markers = on
```

『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**alter connection**」を参照し、**batch** オプションおよび **dsi_cmd_separator** オプションを **alter connection** コマンドを使用して設定する方法についての情報を取得します。

Replicate Data Server としての Linux、UNIX、および Windows に関する IBM DB2

ASE 以外のデータ・サーバを含む場合に限った Sybase 複写システムに固有の管理作業を実行します。

複写システムの基本的な管理については、『Replication Server 管理ガイド 第 1 巻』を参照してください。

DB2 UDB レプリケート・データ・サーバ

複写システム内のレプリケート・データ・サーバとしての DB2 UDB は、ECDA Option for ODBC データベース・ゲートウェイと対話します。

ECDA Option for ODBC は、レプリケート Replication Server からコマンドを受け入れ、それらのコマンドを DB2 UDB サーバにあるデータベースに適用します。

DB2 UDB での複写の干渉と影響

DB2 UDB レプリケート・データベースへの唯一の重大な干渉または影響は、Replication Server のオペレーションをサポートするために、レプリケート・データベースに 3 つのテーブルを作成する接続プロファイルによって作成されるデータベース・オブジェクトです。

テーブルの内容は次のとおりです。

- レプリケート・データベースによって使用されるソート順と文字セットに関する情報を格納する `RS_INFO`

  注意： `RS_INFO` に対する **INSERT** 文で、使用している DB2 UDB サーバに、適切な文字セットとソート順が指定されていることを確認してください。

  Replication Server バージョン 12.0 以降を使用する場合、レプリケート・データベースのソート順と文字セットは、`RS_INFO` テーブルに記録されなければなりません。
  Replication Server の **rs_get_charset** および **rs_get_sortorder** ファンクションは、レプリケート・データベース内の `RS_INFO` テーブルから文字セットとソート順を取得します。
- レプリケート・データベースに適用する複写トランザクションに関する情報を格納する `RS_LASTCOMMIT`

RS_LASTCOMMIT テーブルの各ローは、プライマリ・データベースからレプリケート・データベースに配信されコミットされた、最後のトランザクションを示します。Replication Server は、この情報によってすべてのトランザクションが配信されたことを確認します。

Replication Server の **rs_get_lastcommit** ファンクションは、レプリケート・データベース内の最後にコミットされたトランザクションに関する情報を取得します。ASE 以外のレプリケート・データベースについては、**rs_get_lastcommit** 関数は、データベース固有のファンクション文字列クラスで、レプリケート・データベース内の RS_LASTCOMMIT テーブルへのアクセスに必要なクエリによって置き換えられます。

- RS_TICKET_HISTORY は、Replication Server コマンド **rs_ticket** の実行結果を格納します。プライマリ・データベースに対して **rs_ticket** コマンドを発行し、コマンドがプライマリ・データベースからレプリケート・データベースまで移動するために要する時間を測定できます。この情報を使用して、Replication Server のパフォーマンス、モジュールのハートビート、複写の正常性、テーブルレベルのクワイスをモニタします。**rs_ticket** の各実行結果は、レプリケート・データベース内にある RS_TICKET_HISTORY テーブルの 1 つのローに保存されます。RS_TICKET_HISTORY テーブルの各ローを問い合わせると、個々の **rs_ticket** 実行結果を取得したり、別のローの結果と比較できます。このテーブルに保存されるデータは複写をサポートするために必要ではなく、手動でトランケートして領域を再利用できます。

注意： RS_TICKET_HISTORY テーブルは、Replication Server 15.1 以降でのみ使用できます。

DB2 UDB レプリケート・データベースのパーミッションと制限

レプリケート・データベースにトランザクションを適用するために、Replication Server では Replication Server の **create connection** コマンドで指定するメンテナンス・ユーザ ID が必要となります。

メンテナンス・ユーザ ID は DB2 UDB サーバで定義され、レプリケート・データベースにトランザクションを適用する権限が付与されている必要があります。メンテナンス・ユーザ ID には、DB2 UDB レプリケート・データベースにおける、次のパーミッションが必要です。

- Replication Server の処理に使用するテーブルを作成するための **CREATE TABLE** 権限
- すべてのレプリケート・テーブルに対する **UPDATE** 権限

Replication Server から ECDA Option for ODBC への直接の LOB データ型 (BLOB、CLOB、DBCLOB、LONG VARGRAPHIC、および LONG VARCHAR) の複写は、サポートされていません。

DB2 UDB レプリケート・データベースのコネクティビティ

Replication Server データベース・コネクション名は、データ・サーバ名 (**server_name**) とデータベース名 (**db_name**) の 2 つの部分から構成されています。**server_name** は ECDA Option for ODBC データベース・ゲートウェイ・サーバの名前、**db_name** は DB2 UDB レプリケート・データベースの名前です。

レプリケート Replication Server は、Replication Server データベース・コネクションで指定されたデータベース・ゲートウェイの **server_name** について、`interfaces` ファイル・エントリを検索します。レプリケート Replication Server は、データベース・コネクションで指定された **user_name** と **password** を使用して、レプリケート・データ・サーバにログインします。

Replication Server の `interfaces` ファイルにエントリを作成して、ECDA Option for ODBC データベース・ゲートウェイ・サーバが受信を行うホストとポートを指定してください。`interfaces` ファイル・エントリの名前は、Replication Server データベース・コネクションの **server_name** 部分と一致している必要があります。

DB2 UDB レプリケート・データベースの設定

Replication Server の異機種データ型サポート (HDS) 機能には、レプリケート Replication Server と DB2 UDB レプリケート・データベースで HDS 機能を設定するために使用できる設定情報が用意されています。

この設定情報は、インストールにおいて、接続プロファイルの一部として提供されます。

- Replication Server インストール
  - ファンクション文字列、エラー・クラス、ユーザ定義データ型の作成
- 接続プロファイル：
  - RSSD へのクラス・レベル・データ型変換の適用
  - DB2 UDB レプリケート・データベースでのオブジェクトの作成
  - 接続プロパティの設定
- 追加設定
  - ECDA の設定 (必須)
  - 動的 SQL の設定 (オプション)

- コマンドのバッチ処理の設定 (オプション)

参照：

- RSSD へのクラス・レベル・データ型変換 (95 ページ)
- DB2 UDB レプリケート・データベースでのオブジェクトおよび接続プロパティ (95 ページ)

Replication Server インストール

Replication Server インストールでは、複写をサポートするために必要なファンクション文字列とクラスが自動的にインストールされます。

ファンクション文字列、エラー・クラス、ユーザ定義データ型

ファンクション文字列は、Replication Server のデフォルトの **rs_udb_function_class** に追加されます。

ファンクション文字列は、いくつかのデフォルト Replication Server ファンクション文字列を、DB2 UDB レプリケート・データベースと通信し、作成されたテーブルとプロシージャにアクセスするように設計されたカスタム・ファンクション文字列に置き換えます。

エラー・クラスに定義されているエラー・アクションを調べるには、『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_helperror**」を参照してください。

接続プロファイル

接続プロファイルを使用すると、事前に定義されたプロパティのセットでコネクションを設定できます。

構文

```
create connection to data_server.database
using profile connection_profile;version
set username [to] user
[other_create_connection_options]
[display_only]
```

パラメータ

data_server – 複写システムに追加するデータベースを持つデータ・サーバです。

database – 複写システムに追加するデータベースです。

connection_profile – コネクションの設定、RSSD の修正、およびレプリケート・データベース・オブジェクトの作成に使用する接続プロファイルを示します。

version – 使用する接続プロファイルのバージョンを指定します。

user – データベースの Replication Server メンテナンス・ユーザのログイン名です。Replication Server は、複写データを管理するのにこのログイン名を使用します。ネットワークベース・セキュリティを有効に設定していない場合には、ユーザ名を指定する必要があります。

other_create_connection_options – プロファイルで指定されない接続オプションの設定 (パスワードの設定など)、またはプロファイルで指定されているオプションの上書き (Replication Server に用意されているファンクション文字列クラスを上書きするカスタム・ファンクション文字列クラスの指定など) を行うには、他の **create connection** オプションを使用します。**create connection** コマンドの他の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「Replication Server コマンド」の「**create connection**」を参照してください。

display_only – **display_only** は、**using profile** 句とともに使用し、実行されるコマンド、およびそのコマンドを実行するサーバの名前を表示します。**display_only** を使用した結果については、クライアント・ログおよび Replication Server ログを参照してください。

RSSD へのクラス・レベル・データ型変換

クラス・レベル変換によって、プライマリ・データ型とデータの変換先のレプリケート・データ型が特定されます (たとえば、Microsoft SQL Server の `binary` は DB2 UDB の `CHAR FOR BIT DATA` に変換されます)。

以下の接続プロファイルは、DB2 UDB レプリケート・データベースにクラス・レベル変換を提供します。

- `rs_ase_to_udb` – Adaptive Server データ型を DB2 UDB データ型に変換します。
- `rs_db2_to_udb` – DB2 for z/OS データ型を DB2 UDB データ型に変換します。
- `rs_msss_to_udb` – Microsoft SQL Server データ型を DB2 UDB データ型に変換します。
- `rs_oracle_to_udb` – Oracle データ型を DB2 UDB データ型に変換します。

DB2 UDB レプリケート・データベースでのオブジェクトおよび接続プロパティ

接続プロファイルによって、`RS_INFO`、`RS_LASTCOMMIT`、`RS_TICKET_HISTORY` の各テーブルがレプリケート・データベース内に作成されます。

接続プロファイルによって次の接続プロパティが設定されます。

```
set error class rs_udb_error_class
set function string rs_udb_function_class
```

追加設定

複写をサポートするための追加設定について説明します。

設定内容は次のとおりです。

- ECDA の設定 (必須)

  ECDA 設定ファイルには、次の設定を使用します。

  ```
  Transaction Mode = long
  allocate = connect
  ```

  ```
  SQL transformation = Sybase
  ```

- 動的 SQL の設定 (オプション)

  動的 SQL は Replication Server 15.0.1 でサポートされ、DirectConnect UDB 12.6.1 ESD #2 以降を必要とします。

- コマンドのバッチ処理の設定 (オプション)

  コマンドのバッチ処理を使用すると、Replication Server は複数のコマンドを 1 つのコマンド・バッチとしてデータ・サーバに送信できます。セミコロン (;) で区切ることによって、言語ファンクション文字列の出力テンプレートに複数のコマンドを入力できます。データベースがコマンド・バッチを使用できるように設定されている場合 (デフォルトの設定) は、Replication Server は、このセミコロンをそのコネクションの DSI コマンド・セパレータ文字に置き換えてから、単一のバッチ内のファンクション文字列としてデータ・サーバに送信します。セパレータ文字は、**dsi_cmd_separator** オプション (**alter connection** コマンド) に定義されています。

  データベースへのコネクションがバッチを使用できるように設定されていない場合は、Replication Server は、ファンクション文字列内のコマンドを一度に 1 つずつデータ・サーバに送信します。データベースのバッチを有効または無効にするには、**alter connection** コマンドを使用します。

  コマンドのバッチ処理を使用するには、次のように入力します。

  ```
  set batch = on
  ```

  ```
  set dsi_cmd_separator = ;
  ```

  ```
  set batch_begin = off
  ```

  ```
  use_batch_markers = on
  ```

  batch オプションおよび **dsi_cmd_separator** 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「alter connection」を参照してください。

IBM DB2 レプリケート・データベースの並列 DSI スレッド

異機種間複写環境では、並列 DSI を使用すると、レプリケート・データベースのコミット順はプライマリ・データベースと必ず同じになります。デッドロックが発生した場合、DSI によってデッドロック競合が解決され、Replication Server ではトランザクションをロールバックして再度実行します。

Replication Server は、トランザクションのコミット順序を維持し、次のいずれかの方法で、同時に並列して実行されているトランザクションでの更新の競合を検出できます。

- 内部的に、Replication Server の内部テーブルとファンクション文字列を使用する。
- 外部的に、レプリケート・データベースの `rs_threads` システム・テーブルを使用する。

外部コミット制御では、次のルールに従う必要があります。

- 異なるセッションが同じローで動作する場合、セッション 1 の **update** オペレーションによってセッション 2 の **select** オペレーションはブロックされます。
- 異なるセッションが異なるローで動作する場合、セッション 1 の **update** オペレーションによってセッション 2 の **update** はブロックされません。

内部コミット制御メソッドは、依存する条件が少ないため、外部コミット制御よりも優れています。デッドロックが発生した場合、内部コミット制御では Replication Server が単一のトランザクションをロールバックできますが、外部コミット制御ではすべてのトランザクションをロールバックします。

Replication Server で、並列処理が最大限に実行され、トランザクション間の競合が最小限に抑えられるようにする方法は、他にもあります。たとえば、トランザクションの逐次化メソッドを使用すると、システムが競合を起こすことなく処理できる並列度を選択できます。

並列 DSI スレッドを使用する方法の詳細については、『Replication Server 管理ガイド 第 2 巻』の「パフォーマンス・チューニング」を参照してください。

外部コミット制御

Replication Server では、レプリケート・データベースが IBM DB2 UDB の場合、ロー・レベルのロックを使用して **rs_threads** を作成します。

デフォルトでは、ロー・レベルのロックは “on” です。例：

```
create table rs_threads (id int,seq int)
```

```
create unique index thread_index on rs_threads(id) cluster
```

独立性レベルが 3 の場合、次のファンクション文字列を使用する必要があります。

```
select seq from rs_threads where id = ? with cs
```

構文の説明は次のとおりです。

cs はカーソル安定性を示し、IBM DB2 UDB のデフォルトの独立性レベルです。

内部コミット制御

Replication Server は、**rs_dsi_check_thread_lock** ファンクションを使用して、現在の DSI エグゼキュータ・スレッドが別のレプリケート・データベースのプロセスをブロックしているかどうかをチェックします。

例：

```
select count(*) as seq from table(snapshot_lock('',-1))
as T1 where TABLE_NAME!= '' AND AGENT_ID in (SELECT
AGENT_ID FROM TABLE(SNAPSHOT_APPL_INFO('',-1)) as T2
WHERE APPL_ID = (VALUES APPLICATION_ID()))
```

IBM DB2 UDB では、以下を使用して、現在のセッションのロック情報を選択します。

```
select agent_id from table(snapshot_lock('',-1)) as locktable
```

現在のセッション ID を取得するには、以下を使用します。

```
SELECT APPL.AGENT_ID FROM TABLE(SNAPSHOT_APPL_INFO('',
-1)) AS APPL WHERE APPL.APPL_ID = (VALUES APPLICATION_ID())
```

トランザクションの逐次化メソッド

Replication Server には、並列化のレベルを指定するための 3 つの逐次化メソッドが用意されています。複写環境、そして並列スレッド間で予想される競合の度合いに応じて、逐次化メソッドを選択してください。

各逐次化メソッドでは、トランザクションが直前のトランザクションのコミットを待機しなければならなくなる前に、そのトランザクションをどの程度の量だけ開始できるようにするかを定義します。

逐次化メソッドによって割り当てられた並列度を下げずに競合の確率を下げるには、**dsi_partitioning_rule** パラメータを使用します。

逐次化メソッドは、次のとおりです。

- **no_wait**
- **wait_for_start**
- **wait_for_commit**

- **wait_after_commit**

alter connection コマンドと **dsi_serialization_method** パラメータを使用して、データベース・コネクションに逐次化メソッドを選択します。たとえば、次のコマンドを入力して、**wait_for_commit** 逐次化メソッドをコネクションに選択します。このコネクションは pubs2 データベースへのもので、これは SYDNEY_DS データ・サーバにあります。

```
alter connection to SYDNEY_DS.pubs2
 set dsi_serialization_method to 'wait_for_commit'
```

トランザクションは次の 3 つの部分で構成されています。

- 先頭部分。
- トランザクション本体。**insert**、**update**、**delete** などのオペレーションで構成されます。
- トランザクションの末尾部分。コミットまたはロールバックで構成されます。

逐次化メソッドは、コミットの一貫性を保ちながら、トランザクションの先頭部分が直前のトランザクションのコミット準備の完了を待機するかどうか、またはトランザクションの先頭部分がそれよりも前に処理されるようにするかどうかを定義します。

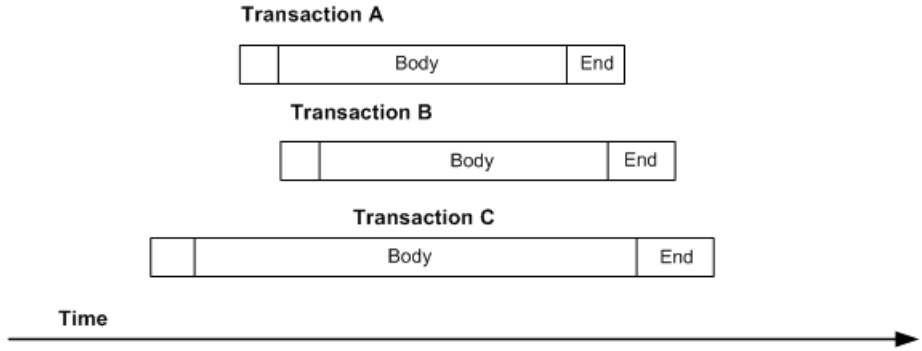
no_wait

no_wait メソッドは、直前のトランザクションのコミットを待機せずに次のトランザクションを開始するよう DSI に指示します。

no_wait を使用する場合、使用するプライマリ・アプリケーションが更新の競合を回避するよう設計されているか、**dsi_partitioning_rule** を効果的に使用して競合を減少させたり、取り除いたりしていることが前提となります。Adaptive Server は、**dsi_isolation_level** が **3** に設定されなければ、更新ロックを保持しません。このメソッドは、並列トランザクション間の競合がほとんどないことを前提とし、結果的には「wait_for_commit 逐次化メソッドによるスレッドのタイミング」図に示すように並列に近い形で実行されます。

注意： **dsi_commit_control** を “on” に設定している場合は、**dsi_serialization_method** は **no_wait** にのみ設定できます。

図 3 : no_wait 逐次化メソッドによるスレッドのタイミング



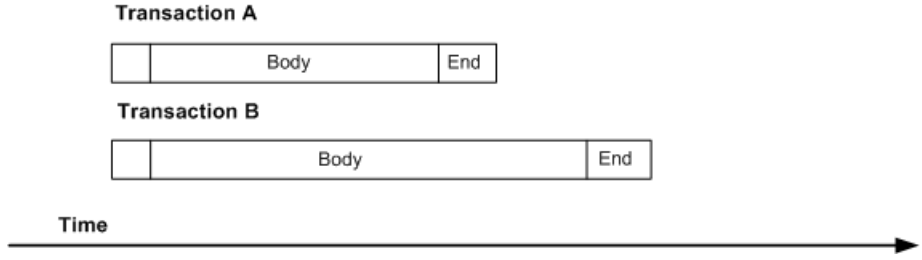
no_wait を指定すると、パフォーマンスが向上する可能性が高くなりますが、競合が発生する危険性も高くなります。

wait_for_start

wait_for_start は、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します。

dsi_serialization_method を **wait_for_start** に設定し、同時に **dsi_commit_control** を **off** に設定することがないようにしてください。

図 4 : wait_for_start 逐次化メソッドによるスレッドのタイミング

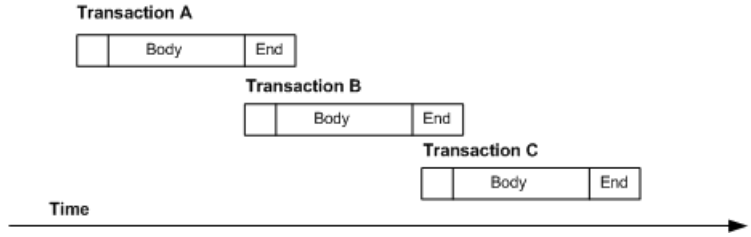


wait_for_commit

wait_for_commit メソッドでは、直前のトランザクションの処理が正常に完了してコミットの送信が開始されるまで、次のスレッドのトランザクション・グループが処理のために送信されることはありません。

デフォルトの設定です。このメソッドは、並列トランザクション間にかなりの競合があることを前提とし、図に示すように実行にずれが生じます。

図 5 : wait_for_commit 逐次化メソッドによるスレッドのタイミング

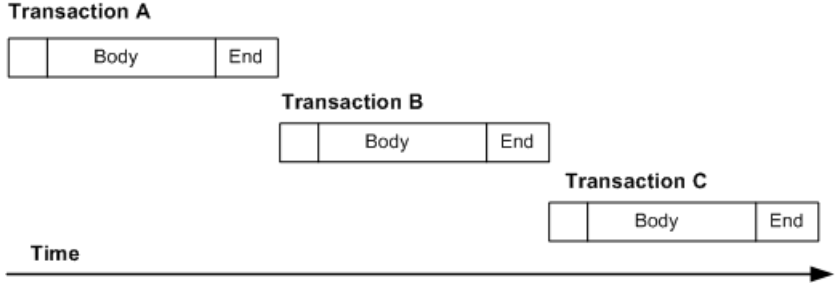


このメソッドでは、1 つのトランザクションのコミットの準備ができるまで待機してから次のトランザクションを開始するように DSI に指示し、トランザクションの逐次化を維持します。最初のトランザクションは必要なロックをすでに保持しているため、最初のトランザクションのコミット中に次のトランザクションをレプリケート・データ・サーバに送信できます。

wait_after_commit

wait_after_commit は、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、トランザクションを開始できないよう指定します。

図 6 : wait_after_commit 逐次化メソッドによるスレッドのタイミング



レプリケート・データ・サーバとしての Microsoft SQL Server

Sybase 複写システムにおける、Microsoft SQL Server データ・サーバに固有のレプリケート・データベースの問題と考慮事項について説明します。

Microsoft SQL Server レプリケート・データ・サーバ

レプリケート・データ・サーバとしての Microsoft SQL Server は、ECDA Option for ODBC データベース・ゲートウェイと対話します。

ECDA Option for ODBC サーバは、レプリケート Replication Server からコマンドを受け入れ、それらのコマンドを Microsoft SQL Server データベースに適用します。

注意： ECDA Option for ODBC では、Replication Server から Microsoft SQL Server データベースへの直接の LOB データ型 (`image`、`ntext`、および `text`) の複写がサポートされています。

Microsoft SQL Server での複写の干渉と影響

Microsoft SQL Server レプリケート・データベースへの唯一の重大な干渉または影響は、Replication Server レプリケート・データベースのオペレーションをサポートするために接続プロファイルによって作成されるデータベース・オブジェクトです。

接続プロファイルによって、Replication Server のオペレーションをサポートする 3 つのテーブルがレプリケート・データベース内に作成されます。

- レプリケート・データベースによって使用されるソート順と文字セットに関する情報を格納する `RS_INFO`。

  注意： **insert** 文 (`RS_INFO` テーブル) で、使用している Microsoft SQL Server データ・サーバに、適切な文字セットとソート順が指定されていることを確認してください。

  Replication Server バージョン 12.0 以降を使用する場合、レプリケート・データベースのソート順と文字セットは、`RS_INFO` テーブルに記録されなければなりません。

Replication Server の **rs_get_charset** および **rs_get_sortorder** ファンクションは、レプリケート・データベース内の RS_INFO テーブルから文字セットとソート順を取得します。

- レプリケート・データベースに適用する複写トランザクションに関する情報を格納する RS_LASTCOMMIT。

  RS_LASTCOMMIT テーブルの各ローは、プライマリ・データベースからレプリケート・データベースに配信されコミットされた、最後のトランザクションを示します。Replication Server は、この情報によってすべてのトランザクションが配信されたことを確認します。

  Replication Server の **rs_get_lastcommit** ファンクションは、レプリケート・データベース内の最後にコミットされたトランザクションに関する情報を取得します。ASE 以外のレプリケート・データベースについては、**rs_get_lastcommit** 関数は、データベース固有のファンクション文字列クラスで、レプリケート・データベース内の RS_LASTCOMMIT テーブルへのアクセスに必要なクエリによって置き換えられます。
- RS_TICKET_HISTORY は、Replication Server コマンド **rs_ticket** の実行結果を格納します。**rs_ticket** コマンドは、コマンドがプライマリ・データベースからレプリケート・データベースまで移動するために要する時間を測定するために、プライマリ・データベースに対して発行できます。この情報を使用して、Replication Server のパフォーマンス、モジュールのハートビート、複写の正常性、テーブルレベルのクワイスをモニタできます。**rs_ticket** の各実行結果は、レプリケート・データベース内にある RS_TICKET_HISTORY テーブルの 1 つのローに保存されます。RS_TICKET_HISTORY テーブルの各ローを問い合わせると、個々の **rs_ticket** 実行結果を取得したり、別のローの結果と比較できます。このテーブルに保存されるデータは、手動でトランケートできます。

注意： RS_TICKET_HISTORY テーブルは、Replication Server リリース 15.1 以降でのみ使用できます。

Microsoft SQL Server におけるレプリケート・データベースの制限事項

Microsoft SQL Server では、サーバの起動オプションによって 28 桁または 38 桁の精度がサポートされます。デフォルトの精度は 28 桁です。

しかし、Replication Server には、デフォルトの 28 桁の精度をサポートするユーザ定義データ型 (UDD) が用意されていません。

サーバの設定を超える精度で Microsoft SQL Server データベースに数値データを複写しようとすると、Replication Server は次のエラーを返します。

```
E. 2007/12/14 11:14:58. ERROR #1028 DSI EXEC(134(1)
   dcm_gabeat70_devdb.devdb)
   - dsiqmint.c(2888)
   Message from server: Message: 30291, State 0,
   Severity 19 --
   '[VENDORLIB] Vendor Library Error: [[Message
   Iteration=1|Data Source Name=mssql70_devdb|
   SQLState=22003|Native Error=1007|Message=
   [Microsoft] [ODBC SQL Server Driver][SQL
   Server]The number
   '9999999999999999999.999999999999999999' is out
   of the range for numeric representation (maximum
   precision 28).
   [Message Iteration=2|SQLState=22003|Native
   Error=|Message=[Microsoft][ODBC SQL Server
   Driver][SQL Server]The number
   '0.99999999999999999999999999999999999999' is out
   of the range for numeric representation (maximum
   precision 28).] <DCA>'
```

Microsoft SQL Server は Adaptive Server Enterprise と同じように identity カラムをサポートしているため、identity への挿入をオフおよびオンに設定する Replication Server のファンクション文字列が、Microsoft SQL Server で正常に動作します。しかし、28 桁の数値精度をサポートするには、Sybase のネイティブ `numeric` データ型が `rs_msss_numeric` データ型に変換されなければなりません。その変換の結果、ID 特性が失われます。

Microsoft SQL Server レプリケート・データベースで 28 桁の精度をサポートするために、`numeric` から `rs_msss_numeric` へのデータ型変換を使用するように選択した場合、レプリケート・テーブルは、そのデータを ID として受け取る数値カラムを宣言できません。

レプリケート Microsoft SQL Server テーブルが、変換されたデータを ID として受け取る数値カラムを宣言した場合、Replication Server は次のエラーを返します。

```
E. 2007/12/14 12:05:39. ERROR #1028 DSI EXEC(134(1)
   dcm_gabeat70_devdb.devdb)
   - dsiqmint.c(2888)
   Message from server: Message: 30291, State 0,
   Severity 19 --
   '[VENDORLIB] Vendor Library Error: [[Message
   Iteration=1|Data Source Name=mssql70_devdb|SQL
   Function=INSERT|SQLState=23000|Native
   Error=544|Message=[Microsoft][ODBC SQL Server
   Driver][SQL Server]Cannot insert explicit value
   for identity column in table 'ase_alltypes' when
   IDENTITY_INSERT is set to OFF.] <DCA>'
```

Microsoft SQL Server レプリケート・データベースのパーミッション

レプリケート・データベースにトランザクションを適用するために、Replication Server では Replication Server の **create connection** コマンドで指定するメンテナンス・ユーザ ID が必要となります。

メンテナンス・ユーザ ID は Microsoft SQL Server データ・サーバで定義され、レプリケート・データベースにトランザクションを適用する権限が付与されている必要があります。メンテナンス・ユーザ ID には、Microsoft SQL Server レプリケート・データベースにおける、次のパーミッションが必要です。

- Replication Server の処理に使用するテーブルを作成するための **create table** 権限
- すべてのレプリケート・テーブルに対する **update** 権限
- すべてのレプリケート・ストアド・プロシージャに対する **execute** 権限

Microsoft SQL Server に関するレプリケート・データベースのコネクティビティ

Replication Server データベース・コネクション名は、データ・サーバ名 (**server_name**) とデータベース名 (**db_name**) の 2 つの部分から構成されています。

server_name は ECDA for ODBC データベース・ゲートウェイ・サーバの名前、**db_name** は Microsoft SQL Server レプリケート・データベースの名前です。

レプリケート Replication Server は、Replication Server データベース・コネクションで指定されたデータベース・ゲートウェイの **server_name** について、`interfaces` ファイル・エントリを検索します。レプリケート Replication Server は、データベース・コネクションで指定された **user_name** と **password** を使用して、レプリケート・データ・サーバにログインします。

Replication Server の `interfaces` ファイルにエントリを作成して、ECDA Option for ODBC データベース・ゲートウェイ・サーバが受信を行うホストとポートを指定してください。`interfaces` ファイル・エントリの名前は、Replication Server データベース・コネクションの **server_name** 部分と一致している必要があります。

Microsoft SQL Server レプリケート・データベースの設定

Replication Server の異機種データ型サポート (HDS) 機能には、レプリケート Replication Server と Microsoft SQL Server レプリケート・データベースで HDS 機能を設定するために使用できる設定情報が用意されています。

この設定情報は、インストールにおいて、接続プロファイルの一部として提供されます。

- Replication Server インストール
  - ファンクション文字列、エラー・クラス、ユーザ定義データ型の作成
- 接続プロファイル：
  - RSSD へのクラス・レベル・データ型変換の適用
  - Microsoft SQL Server データベースでのオブジェクトの作成
  - 接続プロパティの設定
- 追加設定
  - ECDA の設定
  - 動的 SQL の設定
  - コマンドのバッチ処理の設定

参照：

- RSSD へのクラス・レベル・データ型変換 (109 ページ)
- Microsoft SQL Server データベースでのオブジェクトおよび接続プロパティ (109 ページ)

Replication Server インストール

Replication Server インストールでは、複写をサポートするために必要なファンクション文字列とクラスが自動的にインストールされます。

ファンクション文字列、エラー・クラス、ユーザ定義データ型

ファンクション文字列は、Replication Server のデフォルトの **rs_msss_function_class** に追加されます。

ファンクション文字列は、いくつかのデフォルト Replication Server ファンクション文字列を、Microsoft SQL Server と通信し、作成されたテーブルとプロシージャにアクセスするように設計されたカスタム・ファンクション文字列に置き換えます。

エラー・クラスに定義されているエラー・アクションを調べるには、『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_helperror**」を参照してください。

接続プロファイル

接続プロファイルを使用すると、事前に定義されたプロパティのセットでコネクションを設定できます。

構文

```
create connection to data_server.database
using profile connection_profile;version
set username [to] user
[other_create_connection_options]
[display_only]
```

パラメータ

data_server – 複写システムに追加するデータベースを持つデータ・サーバです。

database – 複写システムに追加するデータベースです。

connection_profile – コネクションの設定、RSSD の修正、およびレプリケート・データベース・オブジェクトの作成に使用する接続プロファイルを示します。

version – 使用する接続プロファイルのバージョンを指定します。

user – データベースの Replication Server メンテナンス・ユーザのログイン名です。Replication Server は、複写データを管理するのにこのログイン名を使用します。ネットワークベース・セキュリティを有効に設定していない場合には、ユーザ名を指定する必要があります。

other_create_connection_options – プロファイルで指定されない接続オプションの設定 (パスワードの設定など)、またはプロファイルで指定されているオプションの上書き (Replication Server に用意されているファンクション文字列クラスを上書きするカスタム・ファンクション文字列クラスの指定など) を行うには、他の **create connection** オプションを使用します。**create connection** コマンドの他の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「Replication Server コマンド」の「**create connection**」を参照してください。

display_only – **display_only** は、**using profile** 句とともに使用し、実行されるコマンド、およびそのコマンドを実行するサーバの名前を表示します。**display_only** を使用した結果については、クライアント・ログおよび Replication Server ログを参照してください。

RSSD へのクラス・レベル・データ型変換

クラス・レベル変換によって、プライマリ・データ型とデータの変換先のレプリケート・データ型が特定されます (たとえば、DB2 UDB の `TIMESTAMP` は Microsoft SQL Server の `datetime` に変換されます)。

注意： これらの変換は、Replication Server のパフォーマンスに影響を与える可能性があります。使用している特定のプライマリ・データベースとレプリケート・データベースに必要な変換だけを、RSSD に適用してください。

以下の接続プロファイルは、Microsoft SQL Server レプリケート・データベースにクラス・レベル変換を提供します。

- `rs_db2_to_msss` – DB2 UDB for IBM z/OS データ型を Microsoft SQL Server データ型に変換します。
- `rs_ase_to_msss.sql` – Adaptive Server データ型を Microsoft SQL Server データ型に変換します。
- `rs_udb_to_msss` – DB2 UDB (UNIX および Windows 版) データ型を Microsoft SQL Server データ型に変換します。
- `rs_oracle_to_msss` – Oracle データ型を Microsoft SQL Server データ型に変換します。

Microsoft SQL Server データベースでのオブジェクトおよび接続プロパティ

接続プロファイルによって、`RS_INFO`、`RS_LASTCOMMIT`、`RS_TICKET_HISTORY` の各テーブルがレプリケート・データベース内に作成されます。

接続プロファイルによって次の接続プロパティが設定されます。

```
set error class rs_msss_error_class
set function string rs_msss_function_class
```

追加設定

複写をサポートするための追加設定について説明します。

設定内容は次のとおりです。

- ECDA の設定

  ECDA 設定ファイルには、次の設定を使用します。

  ```
  Transaction Mode = long
  allocate = connect
  ```

  ```
  SQL transformation = Sybase
  ```

  set batch が “on” の場合、以下も指定する必要があります。

  ```
  DelimitSqlRequests = yes
  ```

レプリケート・テーブルに `tinyint` データ型が存在する場合、ECDA Microsoft SQL Server の Microsoft SQL サービスの Datatype Conversion セクションに次のパラメータを追加する必要があります。

```
TinyIntResults=tinyint
```

- 動的 SQL の設定

  動的 SQL は Replication Server 15.0.1 でサポートされ、ECDA Option for ODBC 12.6.1 ESD #2 以降を必要とします。

- コマンドのバッチ処理の設定

  コマンドのバッチ処理を使用すると、Replication Server は複数のコマンドを 1 つのコマンド・バッチとしてデータ・サーバに送信できます。セミコロン (;) で区切ることによって、言語ファンクション文字列の出力テンプレートに複数のコマンドを入力できます。データベースがコマンド・バッチを使用できるように設定されている場合 (デフォルトの設定) は、Replication Server は、このセミコロンをそのコネクションの DSI コマンド・セパレータ文字に置き換えてから、単一のバッチ内のファンクション文字列としてデータ・サーバに送信します。セパレータ文字は、**dsi_cmd_separator** オプション (**alter connection** コマンド) に定義されています。

  データベースへのコネクションがバッチを使用できるように設定されていない場合は、Replication Server は、ファンクション文字列内のコマンドを一度に 1 つずつデータ・サーバに送信します。データベースのバッチを有効または無効にするには、**alter connection** コマンドを使用します。

  コマンドのバッチ処理を使用するには、次のように入力します。

  ```
  batch = on
  ```

  ```
  batch_begin = on or off
  ```

  batch_begin に **on** を使用すると、ネットワーク転送の数が減ります。

  ```
  use_batch_markers = off
  ```

  追加のバッチ・マーカは必要ありません。

  set batch が “on” の場合、以下の設定も指定する必要があります。

  ```
  dsi_cmd_seperator set = ;
  ```

  この設定を指定しない場合、ECDA は各バッチ後のコミットを無視し、DSI コネクションの消失後にすべての複写要求がロールバックされます。

  batch オプションおよび **dsi_cmd_separator** 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「alter connection」を参照してください。

Microsoft SQL Server レプリケート・データベースの並列 DSI スレッド

異機種間複写環境では、並列 DSI を使用すると、レプリケート・データベースのコミット順はプライマリ・データベースと必ず同じになります。

デッドロックが発生した場合、DSI によってデッドロック競合が解決され、Replication Server ではトランザクションをロールバックして再度実行します。

Replication Server は、トランザクションのコミット順序を維持し、次のいずれかの方法で、同時に並列して実行されているトランザクションでの更新の競合を検出できます。

- 内部的に、Replication Server の内部テーブルとファンクション文字列を使用する。
- 外部的に、レプリケート・データベースの `rs_threads` システム・テーブルを使用する。

外部コミット制御では、次のルールに従う必要があります。

- 異なるセッションが同じローで動作する場合、セッション 1 の **update** オペレーションによってセッション 2 の **select** オペレーションはブロックされます。
- 異なるセッションが異なるローで動作する場合、セッション 1 の **update** オペレーションによってセッション 2 の **update** はブロックされません。

内部コミット制御メソッドは、依存する条件が少ないため、外部コミット制御よりも優れています。デッドロックが発生した場合、内部コミット制御では Replication Server が単一のトランザクションをロールバックできますが、外部コミット制御ではすべてのトランザクションをロールバックします。

Replication Server で、並列処理が最大限に実行され、トランザクション間の競合が最小限に抑えられるようにする方法は、他にもあります。たとえば、トランザクションの逐次化メソッドを使用すると、システムが競合を起こすことなく処理できる並列度を選択できます。

並列 DSI スレッドを使用する方法の詳細については、『Replication Server 管理ガイド 第 2 巻』の「パフォーマンス・チューニング」を参照してください。

外部および内部のコミット制御

Replication Server では、レプリケート・データベースが Microsoft SQL Server の場合、ロー・レベルのロックを使用して **rs_threads** を作成します。

デフォルトで、ロー・レベルのロックは “on” で、ページ・レベルのロックも “on” です。外部コミット制御メソッド必要なのは、ロー・レベルのロックのみです。

ロー・レベルのロックをテーブルに適用する場合、ユニークなインデックスまたはプライマリ・キーをテーブルに付与する必要があります。例：

```
create table rs_threads
(id int,seq int CONSTRAINT PK_rs_threads PRIMARY KEY CLUSTERED(id
ASC)
WITH (ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = OFF))
```

独立性レベルが 3 の場合、以下を使用します。

```
Select seq from rs_threads with(nolock) where id =?
```

トランザクションの独立性レベルの選択の詳細については、『Replication Server 管理ガイド 第 2 巻』の「パフォーマンス・チューニング」を参照してください。

Replication Server は、**rs_dsi_check_thread_lock** ファンクションを使用して、現在の DSI エグゼキュータ・スレッドが別のレプリケート・データベースのプロセスをブロックしているかどうかをチェックします。例：

```
select count(*) 'seq' from master..sysprocesses where blocked =
@@spid
```

トランザクションの逐次化メソッド

Replication Server には、並列化のレベルを指定するための 3 つの逐次化メソッドが用意されています。複写環境、そして並列スレッド間で予想される競合の度合いに応じて、逐次化メソッドを選択してください。

各逐次化メソッドでは、トランザクションが直前のトランザクションのコミットを待機しなければならなくなる前に、そのトランザクションをどの程度の量だけ開始できるようにするかを定義します。

逐次化メソッドによって割り当てられた並列度を下げずに競合の確率を下げるには、**dsi_partitioning_rule** パラメータを使用します。

逐次化メソッドは、次のとおりです。

- **no_wait**
- **wait_for_start**
- **wait_for_commit**
- **wait_after_commit**

alter connection コマンドと **dsi_serialization_method** パラメータを使用して、データベース・コネクションに逐次化メソッドを選択します。たとえば、次のコマンドを入力して、**wait_for_commit** 逐次化メソッドをコネクションに選択します。このコネクションは `pubs2` データベースへのもので、これは SYDNEY_DS データ・サーバにあります。

```
alter connection to SYDNEY_DS.pubs2
 set dsi_serialization_method to 'wait_for_commit'
```

トランザクションは次の 3 つの部分で構成されています。

- 先頭部分。
- トランザクション本体。**insert**、**update**、**delete** などのオペレーションで構成されます。
- トランザクションの末尾部分。コミットまたはロールバックで構成されます。

逐次化メソッドは、コミットの一貫性を保ちながら、トランザクションの先頭部分が直前のトランザクションのコミット準備の完了を待機するかどうか、またはトランザクションの先頭部分がそれよりも前に処理されるようにするかどうかを定義します。

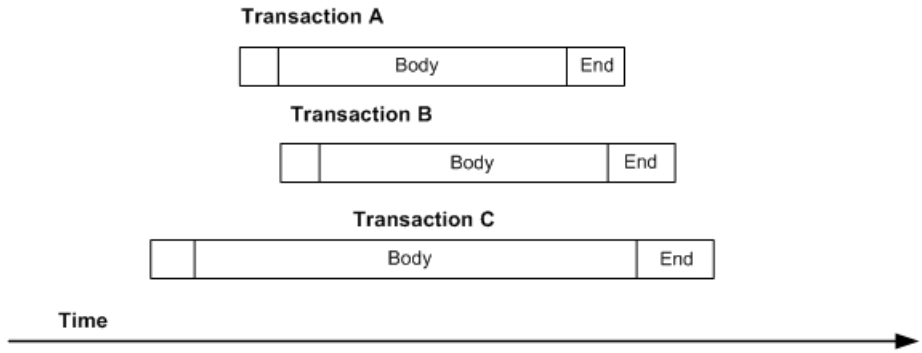
no_wait

no_wait メソッドは、直前のトランザクションのコミットを待機せずに次のトランザクションを開始するよう DSI に指示します。

no_wait を使用する場合、使用するプライマリ・アプリケーションが更新の競合を回避するよう設計されているか、**dsi_partitioning_rule** を効果的に使用して競合を減少させたり、取り除いたりしていることが前提となります。Adaptive Server は、**dsi_isolation_level** が **3** に設定されなければ、更新ロックを保持しません。このメソッドは、並列トランザクション間の競合がほとんどないことを前提とし、結果的には「wait_for_commit 逐次化メソッドによるスレッドのタイミング」図に示すように並列に近い形で実行されます。

注意： **dsi_commit_control** を “on” に設定している場合は、**dsi_serialization_method** は **no_wait** にのみ設定できます。

図 7 : no_wait 逐次化メソッドによるスレッドのタイミング



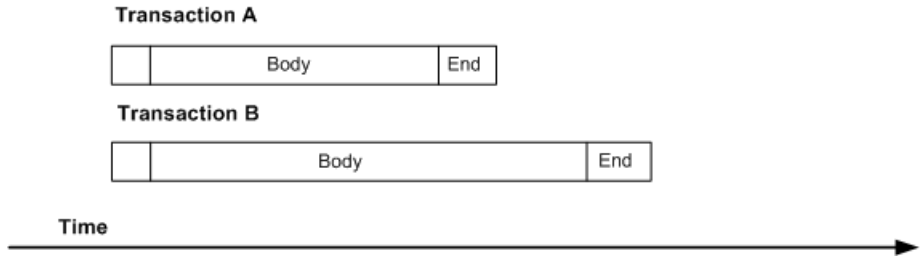
no_wait を指定すると、パフォーマンスが向上する可能性が高くなりますが、競合が発生する危険性も高くなります。

wait_for_start

wait_for_start は、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します。

dsi_serialization_method を **wait_for_start** に設定し、同時に **dsi_commit_control** を **off** に設定することがないようにしてください。

図 8 : wait_for_start 逐次化メソッドによるスレッドのタイミング

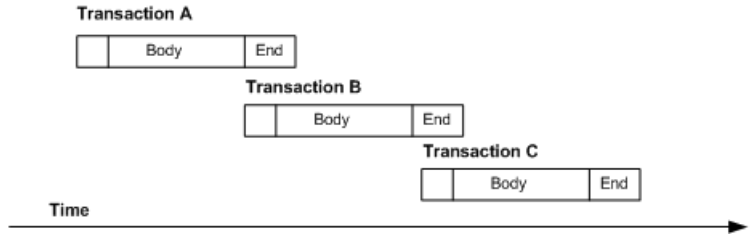


wait_for_commit

wait_for_commit メソッドでは、直前のトランザクションの処理が正常に完了してコミットの送信が開始されるまで、次のスレッドのトランザクション・グループが処理のために送信されることはありません。

デフォルトの設定です。このメソッドは、並列トランザクション間にかなりの競合があることを前提とし、図に示すように実行にずれが生じます。

図 9 : wait_for_commit 逐次化メソッドによるスレッドのタイミング

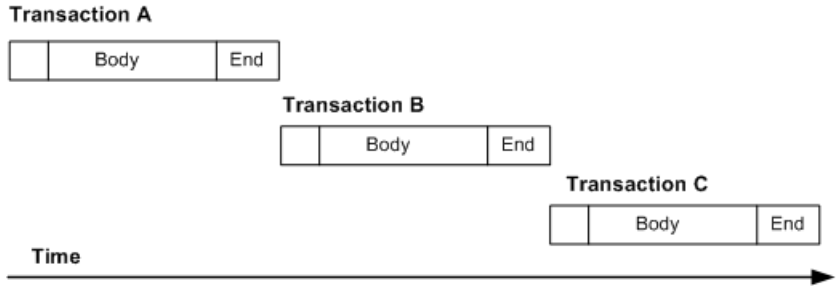


このメソッドでは、1 つのトランザクションのコミットの準備ができるまで待機してから次のトランザクションを開始するように DSI に指示し、トランザクションの逐次化を維持します。最初のトランザクションは必要なロックをすでに保持しているため、最初のトランザクションのコミット中に次のトランザクションをレプリケート・データ・サーバに送信できます。

wait_after_commit

wait_after_commit は、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、トランザクションを開始できないよう指定します。

図 10 : wait_after_commit 逐次化メソッドによるスレッドのタイミング



レプリケート・データ・サーバとしての Oracle

Sybase 複写システムにおける、Oracle データ・サーバに固有のレプリケート・データベースの問題と考慮事項について説明します。

Oracle レプリケート・データ・サーバ

ECDA Option for Oracle データベース・ゲートウェイまたは ExpressConnect for Oracle のいずれを使用しても、Oracle データ・サーバを複写できます。

ECDA Option for Oracle は、レプリケート Replication Server からコマンドを受け入れ、それらのコマンドを Oracle データベースに適用します。

ExpressConnect for Oracle。Replication Server とレプリケート・データ・サーバ間で直接通信を行います。ExpressConnect for Oracle では、ゲートウェイ・サーバのインストールと設定の必要がないため、異機種間複写環境での Oracle データのアクセシビリティが向上します。

Oracle での複写の干渉と影響

Oracle レプリケート・データベースへの唯一の重大な干渉または影響は、Replication Server のオペレーションをサポートするために、レプリケート・データベースに 3 つのテーブルを作成する接続プロファイルによって作成されるデータベース・オブジェクトです。

作成されるテーブルの内容は次のとおりです。

- レプリケート・データベースによって使用されるソート順と文字セットに関する情報を格納する `RS_INFO`。Replication Server バージョン 12.0 以降を使用する場合、レプリケート・データベースのソート順と文字セットは、`RS_INFO` テーブルに記録されなければなりません。

  注意： このテーブルに対する **INSERT** 文で、使用している Oracle データ・サーバに、適切な文字セットとソート順が指定されていることを確認してください。

  Replication Server の **rs_get_charset** および **rs_get_sortorder** ファンクションは、レプリケート・データベース内の `RS_INFO` テーブルから文字セットとソート順を取得します。

- レプリケート・データベースに適用する複写トランザクションに関する情報を格納する RS_LASTCOMMITRS_LASTCOMMIT テーブルの各ローは、プライマリ・データベースからレプリケート・データベースに配信されコミットされた、最後のトランザクションを示します。Replication Server は、この情報によってすべてのトランザクションが配信されたことを確認します。
  Replication Server の **rs_get_lastcommit** ファンクションは、レプリケート・データベース内の最後にコミットされたトランザクションに関する情報を取得します。ASE 以外のレプリケート・データベースについては、**rs_get_lastcommit** 関数は、データベース固有のファンクション文字列クラスで、レプリケート・データベース内の RS_LASTCOMMIT テーブルへのアクセスに必要なクエリによって置き換えられます。
- RS_TICKET_HISTORY は、Replication Server コマンド **rs_ticket** の実行結果を格納します。**rs_ticket** コマンドは、コマンドがプライマリ・データベースからレプリケート・データベースまで移動するために要する時間を測定するために、プライマリ・データベースに対して発行できます。この情報を使用して、Replication Server のパフォーマンス、モジュールのハートビート、複写の正常性、テーブルレベルのクワイスをモニタできます。**rs_ticket** の各実行結果は、レプリケート・データベース内にある RS_TICKET_HISTORY テーブルの 1 つのローに保存されます。RS_TICKET_HISTORY テーブルの各ローを問い合わせると、個々の **rs_ticket** 実行結果を取得したり、別のローの結果と比較できます。データは、手動でトランケートできます。

  注意： RS_TICKET_HISTORY テーブルは、Replication Server バージョン 15.1 以降でのみ使用できます。

Oracle レプリケート・データベースのパーミッション

レプリケート・データベースにトランザクションを適用するために、Replication Server では Replication Server の **create connection** コマンドで指定するメンテナンス・ユーザ ID が必要となります。

メンテナンス・ユーザ ID は Oracle データ・サーバで定義され、レプリケート・データベースにトランザクションを適用する権限が付与されている必要があります。メンテナンス・ユーザ ID には、Oracle レプリケート・データベースにおける、次のパーミッションが必要です。

- Replication Server の処理に使用するテーブルを作成するための **CREATE TABLE** 権限。
- すべてのレプリケート・テーブルに対する **UPDATE** 権限。
- すべてのレプリケート・ストアド・プロシージャに対する **EXECUTE** 権限。

Oracle に関するレプリケート・データベースのコネクティビティ

Replication Server は、ECDA Option for Oracle または ExpressConnect for Oracle (ECO) のいずれを使用しても、Oracle レプリケート・データベースに接続できます。

ECDA の使用

Replication Server のデータベース・コネクション名は、データ・サーバ名 (**server_name**) およびデータベース名 (**db_name**) の 2 つの部分で構成されています。**server_name** は ECDA Option for Oracle データベース・ゲートウェイ・サーバの名前、**db_name** はレプリケート・データベースの Oracle SID の名前です。

レプリケート Replication Server は、Replication Server データベース・コネクションで指定されたデータベース・ゲートウェイの **server_name** について、`interfaces` ファイル・エントリを検索します。レプリケート Replication Server は、データベース・コネクションで指定された **user_name** と **password** を使用して、レプリケート・データ・サーバにログインします。

Replication Server の `interfaces` ファイルにエントリを作成して、ECDA Option for Oracle データベース・ゲートウェイ・サーバが受信を行うホストとポートを指定してください。`interfaces` ファイル・エントリの名前は、Replication Server データベース・コネクションの **server_name** 部分と一致している必要があります。

ExpressConnect for Oracle の使用方法

Replication Server のデータベース・コネクション名は、データ・サーバ名 (**server_name**) およびデータベース名 (**db_name**) の 2 つの部分で構成されています。**server_name** は、`tnsnames.ora` ファイル内のサービス (Oracle インスタンス) の名前です。**db_name** は、Oracle データベース (Oracle SID) のインストールおよび設定時にこのデータベースに与えられた名前です。通常、これはデフォルトで “ORCL” になっています。

ExpressConnect for Oracle は、`tnsnames.ora` ファイル内で、Replication Server データベース・コネクションで指定された **server_name** と一致するエントリを検索します。レプリケート Replication Server は、データベース・コネクションで指定された **user_name** と **password** を使用して、レプリケート・データ・サーバにログインします。ExpressConnect for Oracle を使用して複写する場合、Oracle データ・サーバに必要な interfaces ファイル・エントリはありません。

Replication Server がストアド・プロシージャを複写する方法の指定
ExpressConnect for Oracle (ECO) を使用している場合は、**dsi_proc_as_rpc** を on に設定します。ECO では、リモート・プロシージャ・コール (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ます。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 に設定します。Replication Server Options 15.5 の『Installation and Configuration Guide ExpressConnect for Oracle 15.5』の「Configuring ExpressConnect for Oracle」を参照してください。

ECDA Option for Oracle を使用している場合は、**dsi_proc_as_rpc** を off に設定します。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ません。

Oracle レプリケート・データベースの設定

Replication Server の異機種データ型サポート (HDS) 機能には、レプリケート Replication Server と Oracle レプリケート・データベースで HDS 機能を設定するために使用できる設定情報が用意されています。

この設定情報は、インストールにおいて、接続プロファイルの一部として提供されます。

- Replication Server インストール：
  - ファンクション文字列、エラー・クラス、ユーザ定義データ型の作成
- 接続プロファイル：
  - RSSD へのクラス・レベル・データ型変換の適用
  - Oracle レプリケート・データベースでのオブジェクトの作成
  - 接続プロパティの設定
    接続には、ECDA Server または ExpressConnect for Oracle を使用します。接続に ECDA Server または ExpressConnect for Oracle のどちらを使用するかによって、接続プロファイルのバージョンまたはオプション名がそれぞれ “ecda” または “eco” になります。
- 追加設定
  - ECDA の設定
  - コマンドのバッチ処理の設定
  - 動的 SQL の設定

参照：
- RSSD へのクラス・レベル・データ型変換 (122 ページ)

- Oracle レプリケート・データベースでのオブジェクトおよび接続プロパティ (123 ページ)
- ECDA の設定 (123 ページ)
- コマンドのバッチ処理の設定 (125 ページ)
- 動的 SQL の設定 (127 ページ)

Replication Server インストール

Replication Server インストールでは、複写をサポートするために必要なファンクション文字列とクラスが自動的にインストールされます。

ファンクション文字列、エラー・クラス、ユーザ定義データ型

ファンクション文字列は、Replication Server のデフォルトの `rs_oracle_function_class` に追加されます。

ファンクション文字列は、いくつかのデフォルト Replication Server ファンクション文字列を、Oracle データ・サーバと通信し、テーブルとプロシージャにアクセスするように設計されたカスタム・ファンクション文字列に置き換えます。

警告！ ExpressConnect for Oracle は、text 型と image 型の処理のカスタム・ファンクション文字列の使用をサポートしていません。

接続プロファイル

接続プロファイルを使用すると、事前に定義されたプロパティのセットでコネクションを設定できます。

構文

```
create connection to data_server.database
using profile connection_profile;version
set username [to] user
[other_create_connection_options]
[display_only]
```

パラメータ

data_server – 複写システムに追加するデータベースを持つデータ・サーバです。

database – 複写システムに追加するデータベースです。

connection_profile – コネクションの設定、RSSD の修正、およびレプリケート・データベース・オブジェクトの作成に使用する接続プロファイルを示します。

version – 使用する接続プロファイルのバージョンを指定します。

user – データベースの Replication Server メンテナンス・ユーザのログイン名です。Replication Server は、複写データを管理するのにこのログイン名を使用します。

ネットワークベース・セキュリティを有効に設定していない場合には、ユーザ名を指定する必要があります。

other_create_connection_options – プロファイルで指定されない接続オプションの設定 (パスワードの設定など)、またはプロファイルで指定されているオプションの上書き (Replication Server に用意されているファンクション文字列クラスを上書きするカスタム・ファンクション文字列クラスの指定など) を行うには、他の **create connection** オプションを使用します。**create connection** コマンドの他の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「Replication Server コマンド」の「**create connection**」を参照してください。

display_only – **display_only** は、**using profile** 句とともに使用し、実行されるコマンド、およびそのコマンドを実行するサーバの名前を表示します。**display_only** を使用した結果については、クライアント・ログおよび Replication Server ログを参照してください。

RSSD へのクラス・レベル・データ型変換

クラス・レベル変換によって、プライマリ・データ型とデータの変換先のレプリケート・データ型が特定されます (たとえば、DB2 UDB の `TIMESTAMP` は Oracle の `DATE` に変換されます)。

クラス・レベル変換は、以下に示す適切な名前の接続プロファイルによって Oracle レプリケート・データベースに提供されます。

- `rs_ase_to_oracle` – Adaptive Server データ型を Oracle データ型に変換します。
- `rs_db2_to_oracle` – DB2 UDB for z/OS データ型を Oracle データ型に変換します。
- `rs_udb_to_oracle` – DB2 UDB (UNIX および Windows 版) データ型を Oracle データ型に変換します。
- `rs_msss_to_oracle` – Microsoft SQL Server データ型を Oracle データ型に変換します。

Oracle 複写環境に対して Adaptive Server Enterprise (ASE) の ExpressConnect for Oracle バージョンのプロファイルを使用したスクリプトの例：

```
create connection to oracleSID_name.oracleSID_name
using profile rs_ase_to_oracle;eco
set username rs_maint_user
set password rs_maint_user_pwd
go
```

Oracle レプリケート・データベースでのオブジェクトおよび接続プロパティ

接続プロファイルによって、RS_INFO、RS_LASTCOMMIT、および RS_TICKET_HISTORY の各テーブルがレプリケート・データベース内に作成されるとともに、RS_TRIGGERS_CONTROL パッケージが作成されます。

接続プロファイルによって次の接続プロパティが設定されます。

```
set error class rs_oracle_error_class
set function string rs_oracle_function_class
```

追加設定

複写をサポートするための追加設定について説明します。

設定内容は次のとおりです。

- ECDA の設定
- ExpressConnect の設定
- コマンドのバッチ処理の設定
- トリガの起動の設定
- Oracle Flashback の設定
- 動的 SQL の設定

ECDA の設定

Oracle レプリケート・データ・サーバを使用するときの問題について説明します。

次のような問題があります。

- ECDA Option for Oracle バージョン 12.0 以降では、レプリケート Replication Server は、Oracle レプリケート・データベースにトランザクションを適用するときに、特別なトレース・フラグによってトランザクション・**コミット**の範囲を制御できます。
- ECDA Option for Oracle 設定ファイルで、ECDA の **autocommit** トレース・フラグの値を **0** (ゼロ) に設定すると、Replication Server は **COMMIT** コマンドが Oracle に送信されるタイミングを制御できます。**autocommit** トレース・フラグの値が設定されていない場合、ECDA Option for Oracle は、レプリケート Replication Server によって送信された各オペレーション (**INSERT**、**UPDATE**、**DELETE**) をコミットします。
- ECDA によって各オペレーションがコミットされる場合、複数のオペレーション・トランザクションの途中でエラーが発生すると、レプリケート・データベースで問題が生じます。ECDA が各オペレーションをすでにコミットしていても、レプリケート Replication Server はトランザクション全体を再送信しよう

とします。この問題を回避するには、ECDA の **autocommit** トレース・フラグの値を **0** (ゼロ) に設定します。

- ECDA Option for Oracle 15.0 ESD#3 では、**rep_sparse_parse** 設定パラメータを 1 に設定します。これによって、Oracle 構文の SQL 文が Replication Server から Oracle に直接送信されるとき、ECDA Option for Oracle によって解析されなくなります。この結果、パフォーマンスが改善されるだけでなく、フラッシュバック複写機能を使用することも可能になります。
  rep_sparse_parse 設定パラメータを 0 に設定した場合、Replication Server によって送信される DDL SQL 文および DML SQL 文の一部が ECDA Option for Oracle によって解析され、変更されます。たとえば、ECDA Option for Oracle が Replication Server から DDL 文 **drop table <table_name>** を受け取ると、DDL 文は ECDA Option for Oracle によって解析され、**drop table <table_name> purge** に変更されます。レプリケート・データベースの Recycle Bin とプライマリ・データベースの Recycle Bin を同期させる必要がある場合、DDL 文は変更されないようにする必要があります。この問題は、**rep_sparse_parse** の値を 1 に設定することで回避できます。

ExpressConnect の設定

Replication Server には、Oracle レプリケーション・コネクションの適切なデータベース固有の動作 (データ型変換、コミット処理、**rs_ticket** サポートなど) に必要な設定およびファンクション文字列について Replication Server コネクションに指示する Oracle 接続プロファイルが用意されています。

Oracle に対する Replication Server コネクションを作成または変更する場合、該当する Oracle 接続プロファイル (ASE から Oracle への複写のプロファイルや Oracle から Oracle への複写のプロファイルなど) を使用します。

また、Oracle でストアド・プロシージャを複写する場合、追加の顧客指定のファンクション文字列が必要になる場合があります。デフォルトで、Replication Server によって ASE 構文が生成されますが、ターゲット・データベースで認識されない場合があります。この場合、この構文がターゲット・データベースで認識できるように、ファンクション文字列を追加できます。たとえば、ある文字の型とある通貨の型のパラメータを使用して **econn_test_basic_proc** ファンクション呼び出しを変換する場合、次のようにファンクション文字列を作成する必要があります。

```
create function string econn_test_basic_proc.econn_test_basic_proc
for
rs_oracle_function_class with overwrite output language
'call econn_test_basic_proc(?charcolp!param?, ?moneycolp!param?)'
```

この例では、ファンクション文字列によって、キーワード **call** がすべての複写定義および **rs_oracle_function_class** の **econn_test_basic_proc** というファンクション

の前に配置されます。Oracle で使用可能な構文を生成する別のファンクション文字列の例は次のとおりです。

```
create function string econn_test_basic_proc.econn_test_basic_proc
for
rs_oracle_function_class with overwrite output language 'begin
econn_test_basic_proc(?charcolp!param?, ?moneycolp!param?);; end;;'
```

この例では、ファンクション文字列によって、同じファンクション複写定義とファンクションにキーワード **begin** が付加され、文字列 ";; end;;" が追加されます。

警告！ ExpressConnect for Oracle は、text 型と image 型の処理のカスタム・ファンクション文字列の使用をサポートしていません。

コマンドのバッチ処理の設定

コマンドのバッチ処理を使用すると、Replication Server は複数のコマンドを 1 つのコマンド・バッチとしてデータ・サーバに送信できます。

セミコロン (;) で区切ることによって、言語ファンクション文字列の出力テンプレートに複数のコマンドを入力できます。データベースがコマンド・バッチを使用できるように設定されている場合 (デフォルトの設定) は、Replication Server は、このセミコロンをそのコネクションの DSI コマンド・セパレータ文字に置き換えてから、単一のバッチ内のファンクション文字列としてデータ・サーバに送信します。

セパレータ文字は、**dsi_cmd_separator** オプション (**alter connection** コマンド) に定義されています。データベースへのコネクションがバッチを使用できるように設定されていない場合は、Replication Server は、ファンクション文字列内のコマンドを一度に 1 つずつデータ・サーバに送信します。データベースのバッチを有効または無効にするには、**alter connection** コマンドを使用します。

コマンドのバッチ処理を使用するには、次のように入力します。

```
batch = on
```

```
batch_begin = off
```

set batch が "on" の場合、以下の設定も指定する必要があります。

```
dsi_cmd_seperator set = ;
```

rs_begin ファンクション文字列で使用されるプレースホルダ・コマンドにより、**batch_begin** を "on" に設定すると、DSI を適切に起動できない場合があります。**batch_begin** を "off" に設定すると、**rs_begin** および **rs_commit** コマンドを、コマンドのバッチとは別に送信できるようになるため、転送されたすべてのコマンドで正しい SQL が保証されます。

```
use_batch_markers = on
```

Oracle では、コマンドのバッチに BEGIN マーカと END マーカが必要です。**use_batch_markers** を “on” に設定すると、ファンクション文字列 **rs_batch_start** および **rs_batch_end** から、これらのマーカが自動的に追加されます。『Replication Server 管理ガイド 第 2 巻』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。

トリガの起動の設定

Replication Server では、セッション・レベルまたはコネクション・レベルで Oracle のトリガの実行を無効にすることができます。

トリガの起動は、Replication Server で PL/SQL コマンドがレプリケート・データベースに対して実行されるたびに制御することができます。レプリケート・データベースでのトリガ実行の制御によって、レプリケート・データベース側でのトリガ制御が存在しなかったために発生したデータの重複とデータの誤りのエラーを排除できます。

レプリケート・データベースで制御されるすべてのトリガについては、トリガ・アクションの開始時にトリガを作成し直し、トリガ制御文を追加します。

トリガの起動の制御

RS_TRIGGER_CONTROL パッケージを介してトリガの起動を制御します。このパッケージは、コネクション・プロファイルを通してレプリケート Oracle データベースへのコネクションが作成される際に自動的にインストールされます。

1. コネクション・パラメータ **dsi_keep_triggers** をオフに設定し、Replication Server がレプリケート・データベースへの接続時に、RS_TRIGGERS_CONTROL 有効フラグを設定するようにします。
2. トリガ制御 PL/SQL コードをトリガ・アクションの最初の行に追加します。

   ```
   if RS_TRIGGER_CONTROL.IS_ENABLED then    return;end if;
   ```

   これは、トリガが Replication Server によって起動されることを示し、トリガによる実際のアプリケーション・ロジックの実行を防ぎます。

   『Replication Server リファレンス・マニュアル』を参照してください。

Oracle Flashback の設定

Replication Agent では、テーブル・レベルおよびトランザクション・レベルでの Oracle Flashback がサポートされています。

データベースがオンライン状態のとき、履歴データの問い合わせ、変更分析の実行、および自己サービスによる修復によって、論理的な破損からリカバリするには、Oracle Flashback を使用します。Oracle のユーザは、このフラッシュバックを使用して、以前のデータの変更を元に戻し、貴重なデータの誤削除、誤ったデータの削除、誤ったテーブルの削除などのオペレータまたはユーザによるアプリケーション障害を最小限に抑えることができます。

Replication Agent では、次の 2 種類のフラッシュバックをサポートしています。

- 削除されたテーブルのフラッシュバック。これによって、Oracle を対象とする **drop table**、**flashback table to before drop**、および **purge recyclebin** などのフラッシュバック DDL コマンドが複写されます。**purge dba_recyclebin** を複写するには、DCO 15.0 ESD#3 以降を使用し、**sysdba** 権限を DDL ユーザに割り当てます。
- 特定のタイムスタンプまたは SCN へのテーブルのフラッシュバック。これによって、対象の Oracle データベースに DML の変更が複写されます。

特定のタイムスタンプまたは SCN にテーブルをフラッシュバックするには、次の手順を実行します。

- **pdb_setreptable** コマンドを使用して、特定の状態にフラッシュバックする必要があるテーブルにマークを付けます。

フラッシュバック DDL 文を複写するには、次の手順を実行します。

- 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の Recycle Bin を有効にします。
  ```
  alter system set recyclebin=on
  ```
- ECDA を使用する場合、ECDA Option for Oracle の **rep_sparse_parse** パラメータを 1 に設定します。このパラメータのデフォルト値は、ECDA Option for Oracle 15.0 ESD #3 を使用している場合、0 です。
- **pdb_setrepddl enable** コマンドを使用して、DDL 複写を有効にします。

動的 SQL の設定

動的 SQL は Replication Server 15.0.1 でサポートされ、ECDA Option for Oracle 15.0 以降または ExpressConnect を必要とします。

Oracle レプリケート・データベースの並列 DSI スレッド

異機種間複写環境では、並列 DSI を使用すると、レプリケート・データベースのコミット順はプライマリ・データベースと必ず同じになります。

デッドロックが発生した場合、DSI によってデッドロック競合が解決され、Replication Server ではトランザクションをロールバックして再度実行します。

Replication Server は、トランザクションのコミット順序を維持し、次のいずれかの方法で、同時に並列して実行されているトランザクションでの更新の競合を検出できます。

- 内部的に、Replication Server の内部テーブルとファンクション文字列を使用する。
- 外部的に、レプリケート・データベースの `rs_threads` システム・テーブルを使用する。

外部コミット制御では、次のルールに従う必要があります。

- 異なるセッションが同じローで動作する場合、セッション 1 の **update** オペレーションによってセッション 2 の **select** オペレーションはブロックされます。
- 異なるセッションが異なるローで動作する場合、セッション 1 の **update** オペレーションによってセッション 2 の **update** はブロックされません。

内部コミット制御メソッドは、依存する条件が少ないため、外部コミット制御よりも優れています。デッドロックが発生した場合、内部コミット制御では Replication Server が単一のトランザクションをロールバックできますが、外部コミット制御ではすべてのトランザクションをロールバックします。

Replication Server で、並列処理が最大限に実行され、トランザクション間の競合が最小限に抑えられるようにする方法は、他にもあります。たとえば、トランザクションの逐次化メソッドを使用すると、システムが競合を起こすことなく処理できる並列度を選択できます。

並列 DSI スレッドを使用する方法の詳細については、『Replication Server 管理ガイド 第 2 巻』の「パフォーマンス・チューニング」を参照してください。

外部および内部のコミット制御

Replication Server は、Oracle がレプリケート・データベースの場合、外部コミット制御をサポートしません。

Replication Server は、**rs_dsi_check_thread_lock** ファンクションを使用して、現在の DSI エグゼキュータ・スレッドが別のレプリケート・データベースのプロセスをブロックしているかどうかをチェックします。例：

```
'select count(*) as seq from DBA_BLOCKERS
 where holding_session in (select sid from v$session
 where audsid = userenv('SESSIONID'));'
```

トランザクションの逐次化メソッド

Replication Server には、並列化のレベルを指定するための 3 つの逐次化メソッドが用意されています。複写環境、そして並列スレッド間で予想される競合の度合いに応じて、逐次化メソッドを選択してください。

各逐次化メソッドでは、トランザクションが直前のトランザクションのコミットを待機しなければならなくなる前に、そのトランザクションをどの程度の量だけ開始できるようにするかを定義します。

逐次化メソッドによって割り当てられた並列度を下げずに競合の確率を下げるには、**dsi_partitioning_rule** パラメータを使用します。

逐次化メソッドは、次のとおりです。

- **no_wait**
- **wait_for_start**
- **wait_for_commit**
- **wait_after_commit**

alter connection コマンドと **dsi_serialization_method** パラメータを使用して、データベース・コネクションに逐次化メソッドを選択します。たとえば、次のコマンドを入力して、**wait_for_commit** 逐次化メソッドをコネクションに選択します。このコネクションは pubs2 データベースへのもので、これは SYDNEY_DS データ・サーバにあります。

```
alter connection to SYDNEY_DS.pubs2
 set dsi_serialization_method to 'wait_for_commit'
```

トランザクションは次の 3 つの部分で構成されています。

- 先頭部分。
- トランザクション本体。**insert**、**update**、**delete** などのオペレーションで構成されます。
- トランザクションの末尾部分。コミットまたはロールバックで構成されます。

逐次化メソッドは、コミットの一貫性を保ちながら、トランザクションの先頭部分が直前のトランザクションのコミット準備の完了を待機するかどうか、またはトランザクションの先頭部分がそれよりも前に処理されるようにするかどうかを定義します。

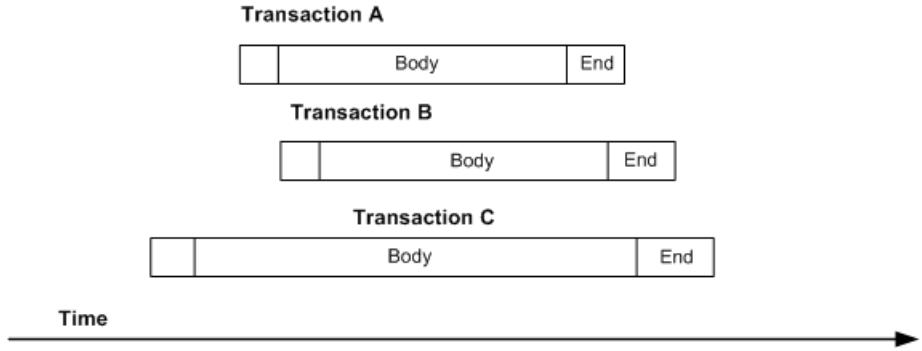
no_wait

no_wait メソッドは、直前のトランザクションのコミットを待機せずに次のトランザクションを開始するよう DSI に指示します。

no_wait を使用する場合、使用するプライマリ・アプリケーションが更新の競合を回避するよう設計されているか、**dsi_partitioning_rule** を効果的に使用して競合を減少させたり、取り除いたりしていることが前提となります。Adaptive Server は、**dsi_isolation_level** が **3** に設定されなければ、更新ロックを保持しません。このメソッドは、並列トランザクション間の競合がほとんどないことを前提とし、結果的には「wait_for_commit 逐次化メソッドによるスレッドのタイミング」図に示すように並列に近い形で実行されます。

注意： **dsi_commit_control** を “on” に設定している場合は、**dsi_serialization_method** は **no_wait** にのみ設定できます。

図 11 : no_wait 逐次化メソッドによるスレッドのタイミング



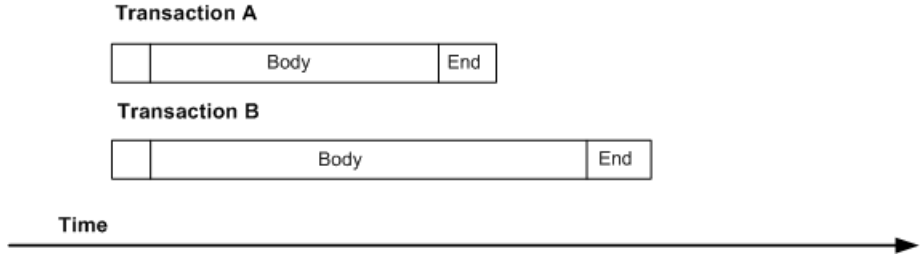
no_wait を指定すると、パフォーマンスが向上する可能性が高くなりますが、競合が発生する危険性も高くなります。

wait_for_start

wait_for_start は、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します。

dsi_serialization_method を **wait_for_start** に設定し、同時に **dsi_commit_control** を **off** に設定することがないようにしてください。

図 12 : wait_for_start 逐次化メソッドによるスレッドのタイミング

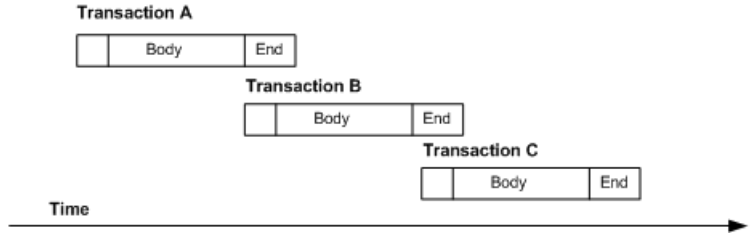


wait_for_commit

wait_for_commit メソッドでは、直前のトランザクションの処理が正常に完了してコミットの送信が開始されるまで、次のスレッドのトランザクション・グループが処理のために送信されることはありません。

デフォルトの設定です。このメソッドは、並列トランザクション間にかなりの競合があることを前提とし、図に示すように実行にずれが生じます。

図 13 : wait_for_commit 逐次化メソッドによるスレッドのタイミング



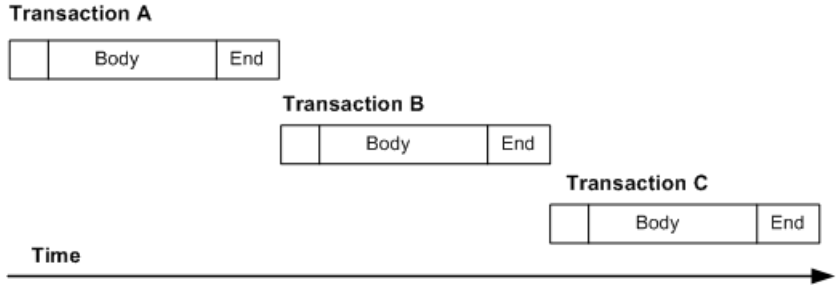
このメソッドでは、1 つのトランザクションのコミットの準備ができるまで待機してから次のトランザクションを開始するように DSI に指示し、トランザクションの逐次化を維持します。最初のトランザクションは必要なロックをすでに保持しているため、最初のトランザクションのコミット中に次のトランザクションをレプリケート・データ・サーバに送信できます。

wait_after_commit

wait_after_commit は、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、トランザクションを開始できないよう指定します。

Oracle など、多版型同時実行制御 (MVCC：Multiversion Concurrency Control) またはオプティミスティック同時制御 (OCC：Optimistic Concurrency Control) を使用するデータベースでは、**wait_after_commit** 逐次化メソッドを使用することをおすすめします。その他のデータベースでは、デフォルトのメソッドとして **wait_for_commit** を使用できます。

図 14 : wait_after_commit 逐次化メソッドによるスレッドのタイミング



レプリケート・データ・サーバとしての Sybase IQ

Sybase 複写システムにおける、Sybase IQ データ・サーバに固有のレプリケート・データベースの問題と考慮事項および Sybase IQ へのレプリケーションの接続および設定方法について説明します。

Sybase IQ は、レポートおよびデータ分析に理想的なプラットフォームです。ただし、より効果的にレポートするには、リアル・タイムのデータが必要です。

Replication Server には、Sybase IQ に直接複写するための Real-Time Loading (RTL) ソリューションが含まれており、ログされた変更をプライマリ・データベースのログ順に従ってレプリケート・データベースに送信する連続複写モードの代わりに使用できます。

コンパイルとバルク適用を実行して Sybase IQ レプリケート・データベースに直接複写する場合、RTL は連続複写モードよりも優れたパフォーマンスを発揮します。

Real-Time Loading ソリューション

RTL はできるだけ多くのコンパイル可能なトランザクションをグループ化して、グループ内のトランザクションをまとめて最終的な変更としてコンパイルしてから、レプリケート・データベースでバルク・インタフェースを使用してその変更をレプリケート・データベースに適用します。

Sybase IQ レプリケート・データベースに複写する際、RTL は次を使用します。

- コンパイル ― 複写データを各テーブル別、各 **insert**、**update**、**delete** オペレーション別にし、それらのオペレーションを最終的なローのオペレーションにコンパイルして整理します。
- バルク適用 ― コンパイルされた最終的な結果に対して最も効率の良いバルク・インタフェースを使用して、最終的な結果をバルク適用します。Replication Server は、メモリ内の最終的な変更が保管されるデータベースを使って最終的なロー変更を保管し、それをレプリケート・データベースに適用します。

RTL では、以下によって、連続複写モードやステージング・ソリューションなどと比べて Sybase IQ への複写パフォーマンスを向上できます。

- 外部コンポーネント数の減少 ― ステージング用のデータベースを使用しないので、メンテナンス・コストとオーバヘッドが削減されます。

- 遅延時間の減少 ― ステージング・ソリューションからのオーバヘッドがなく、Sybase IQ に直接複写されます。
- 利便性の向上 ― RTL の設定には、ファンクション文字列のマップ、DSI のサスペンドとレジューム、ステージング・データベースから Sybase IQ へのデータの移植、ステージング・ソリューション用のアクティビティのスケジューリングのどれも必要ありません。
- コンパイルとバルク適用 ― ログに記録されている個々のオペレーションを送信する代わりに、コンパイルは、グループ化された一連のオペレーションから **insert**、**update**、または **delete** の中間オペレーションを削除し、複写されたトランザクションの最終的なコンパイルされた状態のみを送信します。トランザクション・プロファイルによって異なりますが、これは通常 Replication Server が Sybase IQ に送信して処理させるコマンド数を少なくします。

  Sybase IQ には、SQL 言語モードと比べて **insert** オペレーションのパフォーマンスを向上させるバルク・インタフェースが用意されています。RTL は Sybase IQ のバルク・インタフェースを活用して、**insert** だけでなく **update** と **delete** のオペレーションのパフォーマンスを向上させます。

  Replication Server が大量のトランザクションを組み合わせてコンパイルし 1 つのグループにまとめるので、バルク・オペレーション処理が向上し、複写スループットとパフォーマンスも向上します。グループ・サイズを調整して、バルク適用のためにグループ化されるデータ量を制御できます。

ライセンス

Real-Time Loading (RTL) を使用した Sybase IQ へのレプリケーションは、Real-Time Loading Edition 製品エディションで実行可能です。『Replication Server インストール・ガイド』の「インストールの計画」の「ライセンスの取得」を参照してください。

データベースとプラットフォームのサポート

- Sybase IQ ― Real-Time Loading を使用して Sybase IQ バージョン 12.7 ESD #3 以降に複写できます。Sybase IQ のバージョンとプラットフォームのサポートに関する最新情報については、『Replication Server リリース・ノート』の「製品の互換性」の「Replication Server の相互運用性」を参照してください。
- Adaptive Server – Replication Server は、Adaptive Server バージョン 15.0.3 またはバージョン 15.5 以降から Sybase IQ へのレプリケーションをサポートしています。
- Oracle ― Replication Server は Sybase IQ から Oracle 10g および 11g へのレプリケーションをサポートしています。Replication Server Options 15.5 の『Release Bulletin Replication Agent 15.5』の「Product Summary」の「Compatible Products」を参照してください。

64 ビット・サポート
64 ビットのハードウェア・プラットフォームを使用すると、最適なパフォーマンスを得ることができます。『Replication Server 新機能ガイド』の「Replication Server バージョン 15.5 の新機能」の「64 ビット・コンピューティング・プラットフォームのサポート」を参照してください。

RTL のコンパイルとバルク適用

RTL のコンパイルでは、複写するデータがテーブルごとに **insert**、**update**、**delete** オペレーション別に整理してまとめられ、オペレーションが最終的なローのオペレーションにコンパイルされます。

RTL は複写定義内のプライマリ・キーによって異なったデータ・ローを区別します。複写定義がない場合は、`text` と `image` のカラム以外はすべてプライマリ・キーとみなされます。

通常の複写環境で見られるオペレーションの組み合わせでは、同一のプライマリ・キーを持つテーブルとローがあると、RTL は次のオペレーションの組み合わせルールに従います。

- **insert** の後に **delete** があると結果はオペレーションなしになる。
- **delete** の後に **insert** があると結果のオペレーションは減少しない。
- **update** の後に **delete** があると結果のオペレーションは **delete** になる。
- **insert** の後に **update** があると、2 つのオペレーションは 1 つのオペレーションに集約され、結果のオペレーションは **insert** になる。結果のオペレーションの内容は、最初のオペレーション結果に次のオペレーションの相違点を上書きした結果となる。
- **update** の後にもう 1 つの **update** があると、2 つのオペレーションは 1 つのオペレーションに集約され、結果のオペレーションは **update** になる。結果のオペレーションの内容は、最初のオペレーション結果に次のオペレーションの相違点を上書きした結果となる。

オペレーションのその他の組み合わせでは、コンパイル・ステータスが無効になります。

例 1
これはログ順のローごとの変更例です。この例では、`T` は "**create table** `T(k int , c int)`" コマンドによって以前に作成されたテーブルです。

```
1. insert T values (1, 10)
2. update T set c = 11 where k = 1
3. delete T where k = 1
4. insert T values (1, 12)
5. delete T where k =1
6. insert T values (1, 13)
```

RTL では、1 の **insert** と 2 の **update** を組み合わせて **insert** `T values (1, 11)` に変換できます。変換結果の **insert** と 3 の **delete** は相殺されるので削除できます。4 の **insert** と 5 の **delete** は削除できます。コンパイルされた最終的な RTL オペレーションは、最後の 6 の **insert** の次の値になります。

```
insert T values (1, 13)
```

例 2
ログ順のローごとの変更のもう 1 つの例です。

```
1. update T set c = 14 where k = 1
2. update T set c = 15 where k = 1
3. update T set c = 16 where k = 1
```

RTL では、1 と 2 の **update** をまとめると 2 の **update** になり、2 と 3 の updates をまとめると 3 の **update** になるので、それが k = 1 での最終的なロー変更になります。

Replication Server は最終的な変更を保管するインメモリ・データベース内の **insert**、**delete**、および **update** テーブルを使用して、レプリケート・データベースに適用する最終的なロー変更を保管します。最終的なロー変更がレプリケート・テーブル別およびオペレーションの種類別 (**insert**、**update**、または **delete**) にソートされると、バルク・インタフェースに渡す準備が整います。

RTL は **insert** オペレーションをレプリケート・テーブルに直接ロードします。Sybase IQ は **update** と **delete** のバルク・オペレーションをサポートしないので、RTL は **update** と **delete** オペレーションをテンポラリ・ワークテーブルにロードします。テンポラリ・ワークテーブルは RTL によって IQ のテンポラリ・ストア内に作成されます。次に、RTL は **join-update** または **join-delete** オペレーションをレプリケート・テーブルに対して実行して、最終的な結果を生成します。ワークテーブルは動的に作成され削除されます。

例 2 では、次の処理によってコンパイル結果が `update T set c = 16 where k = 1` になります。

1. RTL は *#rs_uT(k int, c int)* ワークテーブルを作成します。
2. RTL がワークテーブルに対して **insert** を実行します。

   ```
   insert into #rs_uT(k, c) location 'idemo.db' {select * from rs_uT}
   ```
3. RTL が次の **join-update** を実行します。

   ```
   update T set T.c=#rs_uT.c from T,#rs_uT where T.k=#rs_uT.k
   ```

RTL が大量のトランザクションを 1 つのグループにまとめるので、バルク・オペレーション処理が向上し、複写スループットとパフォーマンスも向上します。RTL サイズを設定パラメータで調整することによって、RTL がバルク適用のためにグループにまとめるデータの量を制御できます。

RTL はロー変更を変更がログされた順序で適用しませんが、データ・ロスはありません。

- 異なったデータ・ローでは、ロー変更が適用される順序は結果に影響しません。
- 同じロー内では、コンパイル後、**delete** の後に **insert** を適用することによって整合性を維持します。

最終的な変更のデータベース

Replication Server には、最終的な変更を保管するデータベースがあります。これは、インメモリ・レポジトリとして機能し、トランザクションの最終的なロー変更、つまりコンパイルしたトランザクションを保管します。

各トランザクションに対して 1 つの最終的な変更のデータベース・インスタンスがあります。最終的な変更を保管するデータベース内の各複写テーブルには最高 3 つの追跡テーブルがあります。最終的な変更を保管するデータベースとその中のテーブルを点検することによって RTL 複写のモニタと問題のトラブルシューティングを行うことができます。

参照：

- 最終的な変更のデータベースのサイズ (151 ページ)

最終的な変更のデータベースをモニタする

最終的な変更のデータベース・インスタンスにアクセスしてモニタします。

sysadmin cdb コマンドを使って最終的な変更のデータベースをモニタします。

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

RTL の処理と制限事項

RTL は元のコミット順を維持しながら、トランザクションの最終的なロー変更のみを適用します。それによって中間的なロー変更は省略されますが、トランザクションの整合性は保証されます。

このアプローチには次のような問題が伴います。

- **Insert** トリガが起動されません。これは、RTL プロセスが最終的な新しいローのバルク・ロードをテーブルに対して直接行うからです。Replication Server がコンパイルの最終結果をレプリケート・データベースに適用すると、**update** と **delete** の各トリガは引き続き起動します。ただし、Replication Server がコンパイルし最終結果には含まれないロー変更が、それらのトリガから確認できなくなります。トリガが検出できるのは、最後のロー・イメージのみです。
  Replication Server を使用して、ユーザが変更したテーブルの任意のカラムにそのユーザを関連付けるトリガ・ロジックのあるテーブル・スキーマで、`last_update_user` カラムを使用してユーザ更新を監査するとします。userA がテーブルの `colA` と `colC` を変更した後に、userB が `colB` と `colD` を変更し

た場合、トリガが起動すると、トリガ・ロジックが検出できるのはテーブルを最後に変更したユーザのみです。したがって、トリガ・ロジックはこれら 4 つのカラムを変更したユーザとして userB を関連付けます。ロー変更を個別に検出する必要がある同様のロジックを含むトリガを定義する場合、そのテーブルの RTL コンパイルを無効にする必要がある場合があります。

- RTL はロー変更を変更がログに記録された順序と同じ順序では適用しません。複写テーブルにログ順に変更を適用するには、そのテーブルに対して RTL コンパイルを無効にします。
- 複写テーブルに参照制約がある場合、その参照制約を複写定義の中で指定してください。制約エラーを避けるために、RTL は複写定義に従ってテーブルをロードします。
- RTL では、**wait_for_commit** メソッド以外の並列 DSI 逐次化メソッドをサポートしません。
- RTL では、カスタム・ファンクション文字列をサポートしていないため、コンパイルできないコマンドとして扱います。
- 以下を検出すると、Replication Server は、ログ順にローごとに行う連続複写に戻ります。
  - コンパイルできないコマンド ― ストアド・プロシージャ、SQL 文、システム・トランザクション、Replication Server の内部マーカ。
  - コンパイルできないトランザクション ― コンパイルできないコマンドを含んでいるトランザクション。
  - コンパイルできないテーブル ― RTL が無効にされているテーブル、カスタム・ファンクション文字列を持つテーブル、RTL がコンパイルできないテーブルと参照制約関係があるテーブル。
  - 実行時にコンパイルできないテーブル - これは、対象のテーブルの複写定義に **replicate minimal columns** 句を使用する場合やトランザクションによってプライマリ・キー値が変更された場合など、最小限にパックされた更新がトランザクションに含まれている場合に発生します。
- テーブル複写定義がなく、プライマリ・キーが指定されていないテーブルの場合、Replication Server は、`text` カラムまたは `image` カラムを除き、すべてのカラムをプライマリ・キーとして処理するため、テーブルへの更新をプライマリ・キーの更新に変換します。
- RTL はトランザクションのグループ化を停止できるパラメータ (**dsi_partition_rule** など) を無視します。
- RTL 処理中にエラーが発生すると、Replication Server はコンパイルをリトライします。リトライでは、コンパイルが失敗したトランザクションを特定できるまで、トランザクション・グループを小さなグループに分割していきます。特定されたトランザクションは連続複写を使って適用されます。
- パフォーマンス上の利点を実現するには、プライマリ・データベースとレプリケート・データベースを同期させ、エラー発生による Replication Server への余分な処理オーバヘッドを避けるようにします。**dsi_command_convert** を

i2di,u2di に設定して、データを同期できますが、これも処理オーバヘッドを発生させます。データベースを同期する場合は、**dsi_command_convert** を **none** にリセットします。

- RTL はロー・カウントの検証を行って複写の整合性を確認します。ロー・カウントの検証はコンパイルに基づいて行われます。予期されるロー・カウントはコンパイル後のロー数です。
- 複写定義の中に `identity` データ型のカラムがある場合、Replication Server はレプリケート・データベース内で次の Sybase IQ コマンドを実行します。
  - **set temporary option identity_insert= '***table_name***'** (ID カラムの挿入および更新前)。
  - **set temporary option identity insert= “”** (ID カラムの挿入および更新後)。
- デフォルトでは、Oracle は最小限のロギングを行います。したがって、データベース複写定義を使用している場合、テーブル複写定義を作成するか、完全ロギングを有効にして、**update** コマンドが正しく動作するようにします。テーブル複写定義を作成することにした場合、Replication Agent または Replication Server で次のようにしてその定義を作成できます。
  - Replication Agent for Oracle ― 複写マークの付いたテーブルが存在する場合に Replication Server で複写定義を自動的に作成するには、テーブルに複写マークを付ける前に **pdb_auto_create_repdefs** を **true** に設定するか、テーブルにマークを付けてから **rs_create_repdef** を実行します。詳細については、Replication Server Options の『Replication Agent リファレンス・マニュアル』を参照してください。
  - Replication Server – Replication Server で直接複写定義を作成するには、**create replication definition** に **send standby** 句を指定して実行します。『Replication Server リファレンス・マニュアル』を参照してください。

参照：

- 参照制約のあるテーブル (158 ページ)
- RTL 設定パラメータ (147 ページ)

Sybase IQ のレプリケート・データ・サーバ

レプリケート Replication Server は、Sybase IQ レプリケート・データベース内にロギングし、複写されたトランザクションを適用することによって、レプリケート Sybase IQ データ・サーバと直接交信します。

Sybase IQ での複写の干渉と影響

Sybase IQ レプリケート・データベースに対する唯一の実質的な干渉と影響は、接続プロファイルを通して Sybase IQ レプリケート・データベース内に作成されるシステム・テーブルと、RTL のバルク適用のために Sybase IQ レプリケート・データベース内に作成されるテンポラリ・テーブルです。

システム・テーブル
接続プロファイルは Sybase IQ レプリケート・データベース内に 3 つのテーブルを作成します。

- `rs_threads` ― Replication Server はこのテーブルの情報を使ってデッドロックを検出し、並列 DSI スレッド間でトランザクションの逐次化を実行します。このテーブルのエントリは、トランザクションが開始されたときと、コネクションに対して 2 つ以上の DSI スレッドが定義されたときに更新されます。
- `rs_lastcommit` – レプリケート・データベースに適用される複写トランザクションに関する情報を格納します。`rs_lastcommit` テーブル内の各ローは、プライマリ・データベースからレプリケート・データベースに配信されコミットされた、最後のトランザクションを示します。Replication Server は、この情報によってすべてのトランザクションが配信されたことを確認します。
  Replication Server の **rs_get_lastcommit** 関数は、レプリケート・データベース内の最後にコミットされたトランザクションに関する情報を取得します。ASE 以外のレプリケート・データベースについては、**rs_get_lastcommit** 関数は、データベース固有のファンクション文字列クラスで、レプリケート・データベース内の `rs_lastcommit` テーブルへのアクセスに必要なクエリによって置き換えられます。
- `rs_ticket_history` ― Replication Server コマンド **rs_ticket** の実行結果を格納します。プライマリ・データベースに対して **rs_ticket** コマンドを発行し、コマンドがプライマリ・データベースからレプリケート・データベースまで移動するために要する時間を測定できます。この情報を使用して、Replication Server のパフォーマンス、モジュールのハートビート、複写の正常性、テーブルレベルのクワイスをモニタできます。**rs_ticket** の各実行結果は、レプリケート・データベース内にある `rs_ticket_history` テーブルの 1 つのローに保存されます。`rs_ticket_history` テーブルの各ローを問い合わせると、個々の **rs_ticket** 実行結果を取得したり、別のローの結果と比較できます。必要に応じて、`rs_ticket_history` テーブル内のデータを手動でトランケートします。

ワーク・テーブル
RTL は Sybase IQ データベースの IQ テンポラリ・ストア内にテンポラリ・ワークテーブルを作成して、RTL バルク適用をサポートします。ワークテーブルは動的に作成および削除されます。

テンポラリ・テーブルに必要な Sybase IQ 内の領域は Sybase IQ に複写する予定のデータ量に依存します。テンポラリ・ワークテーブルのために Sybase IQ のテンポラリ・データベースの領域を調整するには、Sybase IQ の **alter dbspace** コマンドを使用します。詳細については、該当するバージョンの Sybase IQ マニュアルを参照してください。Sybase IQ 15.0 以降の例:

```
ALTER DBSPACE dbspace-name ADD FILE FileHist3
'/History1/data/file3' SIZE 500MB
```

Sybase IQ に関するレプリケート・データベースのコネクティビティ

Sybase IQ をレプリケート・データ・サーバとして使用する場合、データベース・ゲートウェイを使用する必要はありません。レプリケート Replication Server は、Sybase IQ レプリケート・データ・サーバに直接接続します。

Replication Server のデータベース・コネクション名は、データ・サーバ名 (**server_name**) およびデータベース名 (**db_name**) で構成されています。レプリケート Replication Server は、データベース・コネクションで指定された Sybase IQ レプリケート・データベースの **server_name** を含む `interfaces` ファイル・エントリを検索します。

dsedit を使って Replication Server の `interfaces` ファイルにエントリを作成し、Sybase IQ レプリケート・データ・サーバが受信するホストとポートの指定に使用します。`interfaces` ファイル・エントリの名前は、Replication Server データベース・コネクションの **server_name** 部分と一致している必要があります。Replication Server を再起動して Replication Server の `interfaces` ファイル内の新しいエントリを有効にします。詳細については、『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」の「新しい Replication Server の設定」で、「interfaces ファイルの編集」を参照してください。

Sybase IQ レプリケート・サーバの `interfaces` ファイル内にレプリケート Replication Server のエントリを作成して、Replication Server が **INSERT ... LOCATION** 文を Sybase IQ に送信したときに、Sybase IQ が Replication Server に接続してデータを取得できるようにします。

Replication Server は、データベース・コネクションで指定された **user_name** と **password** を使用して、Sybase IQ レプリケート・データ・サーバにログインします。Sybase IQ レプリケート・データベースでは、**user_name** と **password** はメンテナンス・ユーザの ID とパスワードです。

Sybase IQ レプリケート・データベースのパーミッション

レプリケート・データベース内のトランザクションを適用するために、Replication Server と Sybase IQ ではメンテナンス・ユーザ ID が必要となります。

複写を開始できるようにするには、Sybase IQ データ・サーバでメンテナンス・ユーザ ID を定義して、その ID にトランザクションをレプリケート・データベースに適用する権限を与える必要があります。メンテナンス・ユーザ ID には、Sybase IQ レプリケート・データベースにおける、次のパーミッションが必要です。

- **RESOURCE** 権限 (ワークテーブルとテンポラリ・インデックスの作成に必要)
- **EXECUTE** パーミッション (**sp_iqwho** ストアド・プロシージャの実行に必要)
- **GRANT ALL** パーミッション (すべての複写テーブルに対して必要)
- **UPDATE** 権限 (すべての複写テーブルに対して必要) および **EXECUTE** 権限 (すべての複写ストアド・プロシージャに対して必要)

メンテナンス・ユーザ ID に権限を与える

単純なセットアップで開始する場合、または Sybase IQ への複写をテストする場合は、DBA と RESOURCE の権限を与えます。

1. Sybase IQ `rssetup.sql` サンプル・スクリプトを使用して必要な権限を持つ Sybase IQ メンテナンス・ユーザを作成します。

   警告！ メンテナンス・ユーザ ID が既に存在する場合、このスクリプトはパスワードをデフォルト・パスワードにリセットします。

```
grant connect to dbmaint identified by dbmaint
grant DBA to dbmaint
grant membership in group rs_systabgroup to dbmaint

-- Create a user for REPSRV to extract -- materialization data,
etc.
-- Give sa user access to any replicated tables
-- Give sa user access to REPSRV schema
grant connect to sa identified by sysadmin
grant DBA to sa
grant membership in group rs_systabgroup to sa

-- Allow sa and dbmaint to reference replicated tables created by
```

```
DBA
grant group to DBA
grant membership in group DBA to dbmaint
grant membership in group DBA to sa
go
```

このスクリプトは Sybase IQ インストール・ディレクトリ内の `scripts` ディレクトリにあります。以下は UNIX プラットフォームでの例です。

- Sybase IQ 15.0 より古いバージョン ― `/$ASDIR/scripts`
- Sybase IQ 15.0 以降 ― `/$IQDIR15/scripts`

ディレクトリの場所については、『Sybase IQ インストールおよび設定ガイド』を参照してください。

2. Sybase IQ データベースが Transact-SQL® (For IQ DBA) と互換性があることを確認します。

   『Sybase IQ リファレンス：文とオプション』の「データベース・オプション」の「Transact-SQL 互換性オプション」、および『Sybase IQ リファレンス：ビルディング・ブロック、テーブル、およびプロシージャ』の「他の Sybase データベースとの互換性」を参照してください。
3. 複写に関係するすべてのテーブルとストアド・プロシージャに対する適切なパーミッションを与えます。

Sybase IQ レプリケート・データベースの設定

Sybase IQ サーバの設定に関する問題について説明します。

Replication Server インストール

Replication Server は必要な接続プロファイルを自動的にインストールすることによって、Sybase IQ への複写をサポートするファンクション文字列とクラスを提供します。

接続プロファイル
接続プロファイルを使用すると、ファンクション文字列クラスとエラー・クラスを設定し、ユーザ定義データ型 (UDD: user-defined datatypes) と Sybase IQ 変換をインストールして、Sybase IQ レプリケート・データベース内に複写に必要なテーブルを作成することによって、事前に定義されたプロパティのセットでコネクションを設定できます。

rs_ase_to_iq や **rs_oracle_to_iq** などの接続プロファイルは、Replication Server のインストール・パッケージの一部であり、Replication Server のインストール時に登録されます。次に接続プロファイルを説明します。

- ファンクション文字列、エラー・クラス、ユーザ定義データ型をカスタマイズします。ファンクション文字列は、いくつかのデフォルト Replication Server ファンクション文字列を、Sybase IQ データ・サーバと通信し、テーブルとプロシージャにアクセスするように設計されたカスタム・ファンクション文字列に置き換えます。これらのファンクション文字列は、Replication Server のデフォルトの **rs_iq_function_class** に追加されます。RTL はカスタム・ファンクション文字列をコンパイルできないコマンドとして扱います。
- クラス・レベル・データ型変換をカスタマイズします。クラス・レベル変換によって、プライマリ・データ型とデータの変換先のレプリケート・データ型が特定されます。クラス・レベル変換は、以下に示す適切な名前の接続プロファイルによって Sybase IQ レプリケート・データベースに提供されます。
  - **rs_ase_to_iq** – Adaptive Server データ型を Sybase IQ データ型に変換します。
  - **rs_oracle_to_iq** – Oracle データ型を Sybase IQ データ型に変換します。
- Sybase IQ レプリケート・データベースに `rs_threads`、`rs_lastcommit`、および `rs_ticket_history` の各テーブルを作成します。
- Sybase IQ へのコネクションを設定するために、次のようにしてデフォルトのファンクション文字列クラスとエラー・クラスのコネクション・プロパティを設定します。

```
set error class rs_iq_error_class
set function string rs_iq_function_class
```

Sybase IQ へのコネクションを作成する

Sybase IQ レプリケート・データベースへのコネクションをセットアップします。

1. **create connection** を **using profile** 句および該当するコネクション・プロファイルを指定して使用し、レプリケート Sybase IQ データ・サーバおよびデータベースを指定します。
   たとえば、Oracle プライマリ・データ・サーバからのコネクションを作成するには、次のようにします。

   ```
   create connection to IQSRVR.iqdb
   using profile rs_oracle_to_iq;standard
   set username to dbmaint
   set password to dbmaint
   go
   ```

   Sybase IQ データベースへのレプリケーション・パスを複数作成して、レプリケーション負荷を分散できます。各パスにユニークなメンテナンス・ユーザ ID を使用します。
2. **admin who** を使って Replication Server が Sybase IQ に正常に接続されたことを確認します。

参照：

- Sybase IQ へのマルチパス・レプリケーション (154 ページ)

Sybase IQ データベース・オプションの設定

rs_session_setting 関数と **create function string** コマンドを組み合わせて使用することで、Sybase IQ レプリケート・データベースに接続している間の Sybase IQ パラメータの値を設定できます。たとえば、パフォーマンスを最適化するパラメータ値を設定できます。

1. 新しいファンクション文字列 (**my_iq_fclass** という名前) を作成して、**rs_iq_function_class** を親クラスとして設定します。

   ```
   create function string class my_iq_fclass
   set parent to rs_iq_function_class
   go
   ```

2. **my_iq_fclass** ファンクション文字列クラスの **rs_session_setting** ファンクション文字列を作成して、設定する Sybase IQ パラメータおよび値を含めます。たとえば、**LOAD_MEMORY_MB**、**MINIMIZE_STORAGE**、および **JOIN_PREFERENCE** Sybase IQ データベース・オプションの値を設定して、パフォーマンスを最適化します。

   ```
   create function string rs_session_setting
   for my_iq_fclass
   output language
   'set temporary option Load_Memory_MB=''200''
   set temporary option Minimize_Storage=''on''
   set temporary option join_preference=5'
   go
   ```

   『Replication Server リファレンス・マニュアル』の「Replication Server システム関数」の「**rs_session_setting**」を参照してください。

3. `IQSRVR` データ・サーバにある `iqdb` データベースへのコネクションを変更して、**my_iq_fclass** ファンクション文字列クラスを使用します。

   ```
   alter connection to IQSRVR.iqdb
   set function string class to my_iq_fclass
   go
   ```

RTL を有効にする

必要なパーミッションを与え、レプリケート Sybase IQ データベースに接続したら、RTL を有効にして Sybase IQ への複写を設定できます。

dsi_compile_enable を使ってそのコネクションの RTL を有効にします。**dsi_compile_enable** を off にした場合、Replication Server はログ順、ローごとの連続複写モードを使用します。たとえば、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、最終的なロー変更のみ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 に設定します。

注意： **dsi_compile_enable** を on に設定すると、Replication Server は **dsi_cmd_prefetch** と **dsi_num_large_xact_threads** を無効にします。

特定のデータベースにのみ影響するように、RTL をデータベース・レベルで有効にして設定するには、次のように入力します。

```
alter connection to IQ_data_server.iq_database
set dsi_compile_enable to 'on'
go
```

RTL をサーバまたはテーブル・レベルで有効にして設定することもできます。

- サーバ・レベル ― Replication Server へのすべてのデータベース・コネクションに影響します。

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

- テーブル・レベル ― 指定した複写テーブルのみに影響します。テーブル・レベルとデータベース・レベルの両方でパラメータを指定している場合は、テーブル・レベルのパラメータがデータベース・レベルのパラメータよりも優先されます。テーブル・レベルでパラメータを指定しなければ、データベース・レベルのパラメータの設定が適用されます。テーブルにパラメータを設定するには、**alter connection** と **for replicate table named** 句を使用します。次に例を示します。

  ```
  alter connection to IQ_data_server.iq_database
  for replicate table named dbo.table_name
  set dsi_compile_enable to 'on'
  ```

  for replicate table name 句の使用によってテーブル・レベルのコネクション設定を変更できます。設定の変更は指定したテーブルのすべてのサブスクリプションからの複写データと複写定義に適用されます。

  注意： テーブル・レベルの設定には、**alter connection** しか使用できません。これは Replication Server が **for** 句を **create connection** に対してサポートしていないためです。

dsi_compile_enable を実行した後、レプリケート Sybase IQ データベースへのコネクションをサスペンドしてレジュームします。

RTL 設定パラメータ

Replication Server は、Sybase が推奨するいくつかのパラメータのデフォルト値を自動的に設定します。これらのパラメータの値を変更すると、レプリケーション・パフォーマンスを微調整できます。

変更するパラメータごとに個別の **alter connection** コマンドを実行する必要があります。**alter connection** を入力した後は、複数のパラメータを入力しないでください。

RTL は、Sybase が推奨する **dsi_cdb_max_size**、**dsi_compile_max_cmds**、**dsi_bulk_threshold**、**dsi_command_convert**、および **dsi_compile_retry_threshold** のデフォルト値を自動的に設定します。ただし、複写環境のパフォーマンスを調整するために独自の値を指定することもできます。

パラメータの詳細な説明については、『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」の「**alter connection**」を参照してください。

dsi_bulk_threshold

dsi_bulk_threshold は、特定のコマンド・タイプのテーブルでコンパイルが行われた後の最終的なロー変更コマンド数を指定します。その数に達したら、それがトリガになって、Replication Server はそのテーブルの同じコマンド・タイプにバルク・コピー・インを使用します。デフォルトの最終的なロー変更コマンド数は 20 です。

デフォルトの最終的なロー変更コマンド数は 20 です。

例：

```
alter connection to IQSRVR.iqdb
set dsi_bulk_threshold to '15'
go
```

dsi_cdb_max_size

dsi_cdb_max_size は、RTL 処理中に Replication Server が生成できる最終的な変更を保管するデータベースの最大サイズ (メガバイト単位) を指定します。

デフォルトは 1024MB です。

例：

```
alter connection to IQSRVR.iqdb
set dsi_cdb_max_size to '2048'
go
```

Replication Server は、Sybase IQ への Real-Time Loading (RTL) にフル・インクリメンタル・コンパイルを使用します。フル・インクリメンタル・コンパイルでは、最終的な変更を保管するデータベースのインスタンス内でコンパイルされたトラ

ンザクション・セグメントのコマンド数が、**dsi_compile_max_cmds** のスレッショルドを超える場合、または最終的な変更を保管するデータベースのインスタンス・サイズが **dsi_cdb_max_size** のスレッショルドを超える場合、Replication Server は、最終的な変更を保管するデータベースのインスタンスに対して、そのトランザクションをレプリケート・データベースに送信し、インスタンスが消費したメモリを解放するように指示します。

dsi_compile_max_cmds

dsi_compile_max_cmds は、Replication Server が 1 つのコンパイルされたトランザクションにコンパイルできるトランザクションとコマンドのグループの最大サイズをコマンド数で指定します。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、RTL は新しいグループを開始します。Replication Server は最終的な変更を保管するデータベースのインスタンスを作成して、コンパイルされたトランザクションを格納します。Replication Server は、**dsi_compile_max_cmds** によってグループに許可されるコマンドの最大数に対応させるため、最終的な変更を保管するデータベースのサイズを増やします。Replication Server がコンパイル中の現在のグループの最大グループ・サイズに達すると、Replication Server は、コンパイルされたトランザクションをレプリケート・データベースのワークテーブルに転送し、最終的な変更を保管する特定のデータベースによって消費されたメモリを開放し、新しいグループを開始して、そのグループのための最終的な変更を保管する新しいデータベースのインスタンスを作成します。

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も、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します。

デフォルトは 10,000 コマンドです。

例：

```
alter connection to IQSRVR.iqdb
set dsi_compile_max_cmds to ‘50000’
go
```

dsi_compile_retry_threshold

dsi_compile_retry_threshold は、グループ内のコマンド数に対するスレッショルド値を指定します。失敗したトランザクションを含むグループ内のコマンド数が **dsi_compile_retry_threshold** の値より小さい場合、Replication Server は RTL モードでそのグループのリトライ処理を行わないので、処理時間を節約してパフォーマンスを向上できます。代わりに、Replication Server は連続複写モードに切り替わります。連続複写モードでは、プライマリ・データベースのログ順に従って変更がレプリケート・データベースに送信されます。

デフォルトは 100 コマンドです。

dsi_compile_retry_threshold を設定するときに、データベース・コネクションをサスペンドしてレジュームする必要はありません。このパラメータはコマンドを実行するとすぐに有効になります。

例：

```
alter connection to IQSRVR.iqdb
set dsi_compile_retry_threshold to '200'
go
```

『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」>「ロー・カウントの検証」>「ロー・カウントの検証を制御する」を参照してください。

dsi_command_convert

dsi_command_convert は、複写コマンドの変更方法を指定します。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ます。

- **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** があります。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" の後ろにあります。例：

- **d2none** – **delete** コマンドを複写しない。このオプションでは、**rs_delete** ファンクション文字列をカスタマイズする必要はありません (**delete** operations オペレーションを複写しない場合)。
- **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。**dsi_row_count _validation** を off にすることによってロー・カウントの検証を無効にする場合、複写時に重複キー・エラーを避け、データベースの自動同期ができるようにするために、**dsi_command_convert** を **i2di,u2di** に設定するようおすすめします。
- **t2none** – **truncate table** コマンドを複写しない。

dsi_command_convert のデフォルトは **none** です。これは、コマンドの変換がないことを意味します。

例：

```
alter connection to IQSRVR.iqdb
set dsi_command_convert to 'i2di,u2di'
go
```

参照：

- メモリ消費の制御 (151 ページ)

リトライ・メカニズムの強化

リトライ・メカニズムの強化を使用すると、Replication Server によるコンパイルおよび一括適用の回数が減らされるため、レプリケーションのパフォーマンスが向上します。

RTL はできるだけ多くのコンパイル可能なトランザクションをグループ化して、グループ内のトランザクションをまとめた最終的な変更としてコンパイルしてから、レプリケート・データベースでバルク・インタフェースを使用してその変更をレプリケート・データベースに適用しようとします。RTL の処理結果から発生するレプリケーションのトランザクションが失敗すると、RTL はリトライ・メカニズムを呼び出します。グループ内のトランザクションが失敗すると、RTL はそのグループを同じサイズの 2 つのグループに分割し、コンパイルとバルク適用を各グループに対して試みます。リトライ・メカニズムは失敗したトランザクションを特定し、Replication Server がエラー・アクションのマッピングを実行できるようにします。また、DSI が停止する場合もあるので、失敗したトランザクションの前にあるすべてのトランザクションが適用されます。

RTL 内の最終的な変更を保管するデータベースは、トランザクションの最終的なロー変更、つまりコンパイルしたトランザクションを保管するインメモリ・レポジトリとして機能します。最終的な変更を保管するデータベースの内容は、複数のプライマリ・トランザクションからのコマンドを集約したものであり、RTL ではログ順に適用されません。したがって、リトライ・メカニズムがないと失敗したトランザクションを特定する方法がありません。グループ内のトランザクションが失敗したら、リトライ・メカニズムはそのグループを分割してコンパイルとバルク適用を繰り返します。このような連続したリトライ・プロセスはパフォーマンスの低下の原因となります。

リトライ・メカニズムの強化によって、RTL でトランザクションが失敗したグループが検出された場合にグループが 3 等分され、失敗したトランザクションを含むグループの特定がより効率的に行われるようになりました。

さらに **dsi_compile_retry_threshold** パラメータを使用してグループ内のコマンド数にスレッショルド値を指定できます。失敗したトランザクションを含むグループ内のコマンド数が **dsi_compile_retry_threshold** の値より小さい場合、Replication Server は RTL モードでそのグループのリトライ処理を行わないので、処理時間を節約してパフォーマンスを向上できます。代わりに、Replication Server は連続複写モードに切り替わります。連続複写モードでは、プライマリ・データベースのログ順に従って変更がレプリケート・データベースに送信されます。

メモリ消費の制御

RTL は、フル・インクリメンタル・コンパイルを使用してメモリ消費を制御するため、最終的な変更を保管するデータベースのサイズを制御してメモリの消費量を削減できます。

RTL の SQT メモリ消費の制御

RTL でトランザクション・プロファイリング中に DSI SQT キャッシュでパックされていないコマンドが消費する最大メモリ量を制御します。

SQT スレッドは、RTL のトランザクション・プロファイリング処理によってアンパックされたコマンドが使用し、DSI SQT キャッシュによって参照されるメモリをモニタします。

Replication Server が RTL を使用して複写を行っている場合、DSI スレッドで使用される最大メモリ量は **dsi_sqt_max_cache_size**、**sqt_max_prs_size**、および **dsi_cdb_max_size** の合計です。**dsi_sqt_max_cache_size**、**sqt_max_prs_size**、および **dsi_cdb_max_size** に小さい値を設定するとメモリ消費は低減しますが、レプリケーション・パフォーマンスは低下します。最適なメモリ消費とパフォーマンスを実現するには、複写環境をチューニングします。パラメータの設定については、『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。

最終的な変更のデータベースのサイズ

最終的な変更を保管するデータベースのメモリ消費を低減します。それには、最終的な変更を保管するデータベースのサイズがスレッショルド・サイズに達したらそのデータベースをトリガして、データをレプリケート・データベースにフラッシュします。

メモリの消費量は、最終的な変更が保管されるデータベースなどの Replication Server データ構造および構造が格納されるデータを参照します。最終的な変更が保管されるデータベースは、インメモリ・データ構造です。最終的な変更が保存されるデータベースのメモリ消費は、Replication Server が多数のカラムを持つテーブルや、大きな `text` および `image` データ型値を持つテーブルに適用されたコマンドをコンパイルすると、劇的に増加することがあります。たとえば、100 のカラムを持つテーブルで 1,000,000 のローをコンパイルすると、10 のカラムを持つテーブルで同数のローをコンパイルするのと比べて約 10 倍のメモリが消費されます。他の処理やモジュールに必要なメモリが不十分な場合、複写パフォーマンスは低下します。

Replication Server は、**dsi_cdb_max_size** および **dsi_compile_max_cmds** に設定した値を使用して、メモリの消費量を制御します。**dsi_cdb_max_size** を使用して、

Replication Server が生成できる最終的な変更を保管するデータベースの最大サイズを制御できます。このサイズが設定されたスレッショルドに達すると、Replication Server は最終的な変更を保管するデータベースに作成中のコンパイル済みトランザクションに新しいコマンドとトランザクションをコンパイルする処理を停止し、コンパイル済みグループをレプリケート・データベースに一括して適用し、最終的な変更を保管するデータベースをクリアし、そのデータベースが消費していたメモリを解放します。

Replication Server が生成する最終的な変更を保管するデータベースのインスタンス数は、**dsi_cdb_max_size** および **memory_limit** で設定する値によって異なります。RTL を使用する複写システムの予測メモリ要件は、複写コネクションの数に **dsi_cdb_max_size** を掛けた値になります。

フル・インクリメンタル・コンパイル

フル・インクリメンタル・コンパイルにより、Real-Time Loading (RTL) のレプリケーション・パフォーマンスが向上しますが、これは多くのコマンドを含む大規模なコンパイル可能なトランザクションの処理中のメモリ消費が低減された結果です。

フル・インクリメンタル・コンパイルでは、**insert**、**delete**、または **update** の混合オペレーションを含む大規模なトランザクションをコンパイルできます。Replication Server では、フル・インクリメンタル・コンパイルを使用して大規模なコンパイル可能なトランザクションをレプリケート・データベースに適用します。その際、最終的な変更を保管する複数のインメモリ・データベース・インスタンスを使用します。フル・インクリメンタル・コンパイルは、大規模なトランザクションをセグメントのシーケンスに分割します。各セグメントはコマンド・グループで構成されます。

Replication Server は各セグメントをコンパイルし、1 つのセグメントを格納するために最終的な変更を保管する専用データベースを作成します。Replication Server は、最終的な変更を保管するデータベース・インスタンスにセグメントをレプリケート・データベースに送信して適用するよう指示します。この後、Replication Server は、最終的な変更を保管するデータベース・インスタンスを閉じ、そのデータベースが消費していたメモリを解放します。Replication Server は、次のトランザクション・セグメントに対して別の最終的な変更を保管するデータベース・インスタンスを作成し、最終的な変更を保管するデータベース・インスタンスをすべてのセグメントに対して順序どおりに引き続き作成してから閉じます。

このため、最終的な変更を保管する大規模なデータベース・インスタンスに対して単一のメモリの大部分を消費して大規模トランザクションを保持するのではなく、フル・インクリメンタル・コンパイルによってメモリ要件は、トランザクションのセグメントのみを含んだ単一の最終的な変更を保管する小さなデータベース・インスタンスが消費するメモリに低減されます。フル・インクリメンタル・コンパイルは、使用した最終的な変更を保管するデータベース・インスタン

スの数でメモリ要件を除算します。たとえば、フル・インクリメンタル・コンパイルが最終的な変更を保管する 10 のデータベース・インスタンスを使用して大規模なトランザクションを適用する場合、メモリ要件はフル・インクリメンタル・コンパイルを伴わない要件の約 10 分の 1 です。

メモリ制御パラメータおよび Replication Server の処理

Replication Server のアクションは、メモリ制御パラメータに対して設定する値によって異なります。

dsi_cdb_max_size の異なる値への設定

以下は、Replication Server が 2 つのテーブルで 100,000 の更新によってトランザクションを適用しているのを示しています。テーブル 1 には約 4GB のメモリを必要とする 100 のカラムがあり、テーブル 2 にはメモリの約 10 分の 1 の 400MB を必要とする 10 のカラムがあります。

dsi_cdb_max_size 値 (MB)	テーブル名	複写処理への影響
1024 (デフォルト)	テーブル 1	前提条件: Replication Server の **memory_limit** を 1GB の最終的な変更を保管するデータベースを作成できる大きさの値に設定する。 Replication Server は 1GB の最終的な変更を保管するデータベースのインスタンスを 4 つ使用して、トランザクションを適用する。
1024 (デフォルト)	テーブル 2	前提条件: Replication Server の **memory_limit** を 400MB の最終的な変更を保管するデータベースを作成できる大きさの値に設定する。
4096	テーブル 1	前提条件: Replication Server の **memory_limit** を 4GB の最終的な変更を保管するデータベースを作成できる大きさの値に設定する。 Replication Server は、4GB の最終的な変更を保管するデータベースのインスタンスを 1 つ使用して、トランザクションを適用する。

dsi_cdb_max_size 値 (MB)	テーブル名	複写処理への影響
4096	テーブル 2	前提条件: Replication Server の **memory_limit** を 400MB の最終的な変更を保管するデータベースを作成できる大きさの値に設定する。 Replication Server は、400MB の最終的な変更を保管するデータベースのインスタンスを 1 つ使用して、トランザクションを適用する。

Sybase IQ へのマルチパス・レプリケーション

レプリケーションのスループットとパフォーマンスを向上させ、遅延と競合を削減するため、Replication Server からレプリケート Sybase IQ データベースに複数のコネクションを作成します。

Adaptive Server または Oracle プライマリ・データベースから Replication Server への複数のコネクションと、Replication Server からレプリケート Sybase IQ データベースへの複数のコネクションを使用して、エンドツーエンドの複数のレプリケーション・パスを作成できます。

データベースのサポート

- プライマリ・データベース
  - Adaptive Server 15.7 以降。
  - Oracle 10g と 11g。Replication Server Options の『Replication Agent リリース・ノート』の「製品の概要」の「製品の互換性」を参照してください。
- レプリケート・データベース – Sybase IQ バージョン 15.1 以降『Replication Server リリース・ノート』の「製品の互換性」の「Replication Server の相互運用性」を参照してください。

ライセンス

マルチパス・レプリケーションは、Advanced Services Option の一部としてライセンスされます。RTL を使用した Sybase IQ へのレプリケーションは、Real-Time Loading Edition (RTLE) で実行可能です。『Replication Server インストール・ガイド』の「インストールの計画」の「ライセンスの取得」を参照してください。

参照：

- 異機種間におけるマルチパス・レプリケーション (181 ページ)
- Adaptive Server から Sybase IQ へのマルチパス・レプリケーション (189 ページ)
- Oracle から Sybase IQ へのマルチパス・レプリケーション (193 ページ)

Sybase IQ への代替レプリケート・コネクションの作成

代替コネクションの作成を **using profile** 句と一緒に使用して、Replication Server からレプリケート Sybase IQ データベースへの代替コネクションを作成します。

前提条件

代替コネクションを作成する前に、レプリケート・データベースへのデフォルトのコネクションを作成します。

手順

コネクション・プロファイルとコネクション・プロファイルのバージョン、およびデフォルトのコネクションと各代替コネクションの一意のメンテナンス・ユーザ名を指定する必要があります。
Sybase IQ レプリケート・データベースへの代替コネクションを作成します。

```
create alternate connection to dataserver.database
named conn_server.conn_db
using profile connection_profile;version
set username [to] user
set password [to] pwd
```

構文の説明は次のとおりです。

- *dataserver*および *database*– レプリケート・データ・サーバとデータベースです。
- *conn_server.conn_db* – データ・サーバ名とコネクション名で構成される代替レプリケート・コネクション。
  - 各レプリケート・コネクション名は、複写システム内でユニークにしてください。
  - *conn_server* が *dataserver* と異なる場合は、interface ファイルに *conn_server* のエントリが必要です。
  - *conn_server* が *dataserver* と同じ場合は、*conn_db* が *database* と異なるようにしてください。
- *connection_profile* – レプリケート・データベースの正しいファンクション文字列クラスとエラー・クラスを指定します。また、クラス・レベル変換の定義とレプリケート・データベース・オブジェクトの作成のサポートを含める場合があります。
  - **rs_ase_to_iq** – Adaptive Server から Sybase IQ へのレプリケーション
  - **rs_oracle_to_iq** – Oracle から Sybase IQ へのレプリケーション

  注意： 作成する Sybase IQ データベースへの各代替コネクションに対し、コネクションのプロファイルを指定する必要があります。
- *version* – 使用する接続プロファイルのバージョン

注意： 作成する Sybase IQ データベースへの各代替コネクションに対し、コネクションのプロファイル・バージョンを指定する必要があります。

- *user* – Sybase IQ データベースの各コネクションの Replication Server メンテナンス・ユーザのログイン名です。Replication Server は、複写データを管理するのにこのログイン名を使用します。ネットワークベース・セキュリティを有効に設定していない場合には、ユーザ名を指定する必要があります。

  注意： 作成する Sybase IQ データベースへの各代替コネクションに対し、異なるメンテナンス・ユーザ名を使用する必要があります。異なる Replication Server から Sybase IQ データベースへのコネクションを作成する場合でも、ユニークなメンテナンス・ユーザ名を使用していることを確認します。ユニークなユーザ名を使用しないと、データが重複する場合があります。複写システムは、別の Replication Server からコネクションを作成するために同じユーザ名を使用しているかどうかを検出できません。

たとえば、プライマリ・データベースが Adaptive Server で、dbmaint2 が `IQSRVR.iqdb_conn2` のメンテナンス・ユーザである `IQSRVR` Sybase IQ データ・サーバ内の `iqdb` レプリケート・データベースに、`IQSRVR.iqdb_conn2` という名前の代替レプリケート・コネクションを作成するには、次のようにします。

```
create alternate connection to IQSRVR.iqdb
named IQSRVR.iqdb_conn2
using profile rs_ase_to_iq;standard
set username to dbmaint2
set password to dbmaint2pwd
go
```

代替レプリケート Sybase IQ コネクションの変更または削除

alter connection コマンドと **drop connection** コマンドを使用して、Sybase IQ へのデフォルトまたは代替コネクションを変更または削除します。

コマンドで指定するデータ・サーバ名とデータベース名は、デフォルトまたは代替レプリケート・コネクション名にすることができます。

代替またはデフォルトのレプリケート・コネクションを設定するときに、**alter connection** で使用可能な設定パラメータを使用することができます。

たとえば、**dsi_bulk_threshold** を `IQSRVR.iqdb_conn2` 代替レプリケート・コネクションに対して 15 に設定する場合、次のように入力します。

```
alter connection to IQSRVR.iqdb_conn2
set dsi_bulk_threshold to '15'
go
```

レプリケート・コネクションに関する情報の表示

replicate パラメータを **admin show_connections** と一緒に使用して、すべてのレプリケート・コネクションに関する情報を表示します。

たとえば、IQSRVR データ・サーバでレプリケート・データベースを制御する Replication Server で、次のように入力します。

```
admin show_connections, 'replicate'
```

次のようなメッセージが表示されます。

```
Connection Name         Server        Database   User
-------------------     ----------    --------   ----------
IQSRVR.iqdb             IQSRVR        iqdb       db_maint
IQSRVR.iqdb_conn2       IQSRVR        iqdb       db_maint2
IQSRVR.iqdb_conn3       IQSRVR        iqdb       db_maint3
```

IQSRVR.iqdb は、コネクション名がデータ・サーバとデータベース名の組み合わせに一致するため、Replication Server と IQSRVR データ・サーバの `iqdb` データベースの間のデフォルトのコネクションになります。

IQSRVR.iqdb_conn2 および IQSRVR.iqdb_conn3 は、コネクション名がデータ・サーバとデータベース名の組み合わせに一致しないため、Replication Server と IQSRVR データ・サーバの `iqdb` データベースの間の代替コネクションになります。

レプリケーション・ロード分散統計

プライマリ・データベースでサポートされている分散モードを使用して、使用可能なレプリケート・パス上のレプリケート・ロードを分散します。

分散モード
オブジェクト・バインド・モードによる Adaptive Server 分散統計を使用して、特定のパスへのオブジェクトをバインドすることにより、複数のパス上でテーブルやストアド・プロシージャなどのオブジェクトを分散できます。プライマリ・コネクションとレプリケート・コネクション名が一致する場合、オブジェクトは、プライマリ・データ・サーバからレプリケート・データ・サーバへのエンドツーエンドのレプリケーション・パスに続きます。『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・レプリケーション」の「複数のプライマリ・レプリケーション・パス」の「レプリケーション・パスへのオブジェクトのバインド」を参照してください。

コネクション・モード別 Adaptive Server 分散では、Adaptive Server RepAgent がさまざまなクライアント・プロセスから発生したトランザクションを使用可能なレプリケーション・パスに割り当てます。時間の経過とともに、使用可能なパス全体でデータ分散のバランスが取れていく傾向があります。使用可能な RepAgent パスがさらにあり、クライアント・プロセスの数が多い場合、レプリケーション・パフォーマンスが向上し、レプリケーション負荷分散はより均一化します。『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「マルチパス・レプリケーション」の「並列トランザクション・ストリーム」の「マルチパス・レプリケーションの分散モード」を参照してください。

Sybase IQ マルチプレックス・ノード
レプリケート・コネクションは、各ノードへのコネクションを作成して関連する `interfaces` ファイル・エントリを作成することによりレプリケーション・ロードを分散するため、Sybase IQ マルチプレックスの異なるノードへ割り当てることができます。『Sybase IQ』の「Sybase IQ Multiplex の使用」を参照してください。

分散モードの設定

プライマリ Adaptive Server データベースから複数のプライマリ・レプリケーション・パスを介してレプリケーションの分散モードを設定します。

前提条件

プライマリ Adaptive Server から Replication Server へのデフォルト・コネクションと代替コネクションを作成し、マルチスレッド RepAgent を有効にします。

手順

オブジェクト・バインド別分散から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、警告を表示します。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、バインドが保持されます。

1. 分散モードを設定します。
   sp_config_rep_agent *データベース*、'multipath distribution model'、{'connection' | 'object'}

   構文の説明は次のとおりです。
   - **multipath distribution model** – **sp_config_rep_agent** の分散モード・パラメータ
   - **connection** – モードをコネクション別分散に設定
   - **object** – モードをオブジェクト・バインド別分散に設定 (デフォルト)
2. Replication Server をクワイス状態にし、RepAgent を再起動します。
   『Replication Server 管理ガイド 第 1 巻』の「複写システムの管理」にある「Replication Server のクワイス」の「複写システムのクワイス」を参照してください。

参照制約のあるテーブル

参照制約 (外部キーその他の検査制約など) のあるテーブルの指定には複写定義を使用できます。それによって RTL にそれらのテーブルの存在が通知されます。

通常は、参照元のテーブルには同じプライマリ・データベース内の参照先テーブルに対する参照制約が含まれています。RTL では複数のプライマリ・データベースからの参照先テーブルをサポートするよう参照制約が拡張されています。

各プライマリ・データベースに対する複写定義内で参照元テーブルを指定できます。ただし、複数の参照制約が互いに競合する場合は、Replication Server によってランダムにテーブルが 1 つ選択されます。

参照：

- RTL の処理と制限事項 (137 ページ)

複写定義の作成と変更

参照制約のあるテーブルの指定には、**references** パラメータを指定して **create replication definition** コマンドを使用します。

create replication definition

```
...
(column_name [as replicate_column_name]
    ...
    [map to published_datatype]] [quoted]
    [references [table_owner.]table_name [(column_name)]] …)
....]
```

参照元のテーブルの追加と変更には、**references** パラメータを指定して **alter replication definition** コマンドを使用します。参照を削除するには、**null** オプションを使用します。

alter replication definition

```
   .....
add column_name [as replicate_column_name]
[map to published_datatype] [quoted]
[references [table_owner.]table_name [(column_name)]
...
| alter columns with column_name references
{[table_owner.]table_name [(column_name)] | NULL}
[, column_name references {[table_owner.]table_name [(column_name)]
| NULL}
    ...
```

alter replication definition と **create replication definition** の両方に **reference** 句を指定すると、Replication Server の動作は次のようになります。

- **reference** 句をカラム・プロパティとして扱う。各カラムはテーブルを 1 つだけ参照できる。
- **reference** 句内の **column_name** パラメータに指定したカラム名を処理しない。
- 循環参照になる参照制約を許可しない。たとえば、元の参照先テーブルは元の参照元テーブルへの参照制約を持つことはできない。

複写プロセスでは、RTL は次のようにロードします。

- 参照先テーブルへの挿入の後で複写定義で指定した参照元テーブルに挿入する。
- 複写定義で指定したテーブルでの削除の後で参照先テーブルで削除する。

場合によっては、両方のテーブルでの更新が競合によって失敗することがあります。RTL が複写処理のリトライをしないようにして、パフォーマンスの低下を防ぐには、以下を行います。

- 更新を削除と挿入に変換するように、**dsi_command_convert** を "**u2di**" に設定してレプリケーションの更新を停止する。
- **dsi_compile_enable** を off にして、影響を受けたテーブルがコンパイルされるのを避ける。

カスタム・ファンクション文字列を持つテーブルと、コンパイルできない既存テーブルへの参照制約を持つテーブルは RTL でコンパイルできません。これらのテーブルにマークを付けることによって、RTL は参照制約エラーによって発生するトランザクションのリトライを避け、複写処理を最適化できます。

RTL 情報の表示

設定パラメータ・プロパティとテーブル参照の情報を表示できます。

設定パラメータ・プロパティの表示

admin config を使用して、例に示されているようなデータベース・レベルとテーブル・レベルの設定パラメータを表示します。

- データベース・レベル
  - NY_DS データ・サーバ (`NY_DS.nydb1`) の `nydb1` データベースへのコネクションに使用するデータベース・レベルの設定パラメータをすべて表示するには、次のように入力します。
    ```
    admin config, “connection”, NY_DS, nydb1
    ```
  - `NY_DS.nydb1` へのコネクションで **dsi_compile_enable** が **on** であることを確認するには、次のように入力します。
    ```
    admin config, “connection”, NY_DS, nydb1,dsi_compile_enable
    ```
  - **dsi_compile_enable** など、名前の一部に "enable" があるデータベース・レベルの設定パラメータをすべて表示するには、次のように入力します。
    ```
    admin config, “connection”, NY_DS, nydb1,"enable"
    ```

    注意： "enable" は Replication Server の予約語なので、引用符で囲む必要があります。『Replication Server リファレンス・マニュアル』の「トピック」の「予約語」を参照してください。
- テーブル・レベル

dsi_command_convert を使用して NY_DS データ・サーバの `nydb1` データベースにある `tb1` テーブルで **d2none** を設定した後、すべての設定パラメータを表示するには、次のように入力します。

```
admin config, "table", NY_DS, nydb1
```

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

テーブル参照の表示

テーブル参照の情報と RTL の情報を表示するには、**rs_helprep** を使用します。これは、Replication Server システム・データベース (RSSD) 上で実行できます。

create replication definition を使用して作成した **authors_repdef** 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_helprep authors_repdef
```

『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。

Replication Server 15.5 のシステム・テーブル・サポート

Replication Server では `rs_tbconfig` テーブルをテーブル・レベルの設定パラメータの保管に使用し、`rs_columns` テーブルの `ref_objowner` カラムと `ref_objname` カラムを参照制約のサポートに使用します。

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』の「Replication Server システム・テーブル」を参照してください。

混合バージョンのサポートと下位互換性

RTL では、複写定義で指定されている参照制約を複写できるのは、アウトバウンド・ルートのバージョンが 15.5 以降の場合のみです。

アウトバウンド・ルートのバージョンが 15.5 より古くても RTL は機能します。しかし、バージョン 15.5 以降の場合は、参照制約情報を Replication Server で使用できません。

連続複写モードはサポートされているすべてのバージョンの Replication Server のデフォルト複写モードです。RTL を使用できるのは Replication Server 15.5 以降のみです。

Sybase IQ への複写シナリオ

このシナリオを使用して、RTL を使った Sybase IQ への複写のセットアップと複写作業のテストの方法について説明します。

Adaptive Server データベース管理者 (ASE DBA)、Oracle データベース管理者 (Oracle DBA)、Sybase IQ データベース管理者 (IQ DBA)、複写システム管理者 (RSA) は、Adaptive Server、Oracle、Replication Server、Sybase IQ を複写用に準備して、Sybase IQ データベースへのコネクションをセットアップする必要があります。

このシナリオでは、*dbo* は、ASE_DS プライマリ Adaptive Server または ORA_DS プライマリ Oracle サーバの `pdb1` データベース内の `testtab` テーブル所有者です。`c1`、`c2`、`c3` は `testtab` 内のカラムであり、データ型はそれぞれ `int`、`int`、`char(10)` で、IQSRVR はレプリケート Sybase IQ データ・サーバであり、`iqdb` データベースが格納されています。

参照：

- Adaptive Server から Sybase IQ へのマルチパス・レプリケーション (189 ページ)
- Oracle から Sybase IQ へのマルチパス・レプリケーション (193 ページ)

interfaces ファイルのエントリの作成

レプリケート Replication Server と Sybase IQ データ・サーバの `interfaces` ファイルに、互いのエントリを作成します。

1. Sybase IQ データ・サーバの `interfaces` ファイル (Windows では `sql.ini` ファイル) に、レプリケート Replication Server のエントリを作成します。

   注意： Sybase IQ データ・サーバの `interfaces` ファイルが、Sybase IQ が使用している $SYBASE ディレクトリ (Windows では %SYBASE% ディレクトリ) にない場合、このファイルを作成します。

2. レプリケート Replication Server の `interfaces` ファイルに、Sybase IQ データ・サーバのエントリを作成します。

   さまざまな Sybase IQ マルチプレックス・ノードへのコネクションを作成している場合、影響を受ける各ノードのエントリをレプリケート Replication Server の `interfaces` ファイルに作成します。

テスト・テーブルの作成

プライマリ・データベースとレプリケート・データベース内にテスト・テーブルを作成し、複写作業をテストするためにメンテナンス・ユーザにそのテーブルに対するパーミッションを与えます。

1. データ・サーバにあるプライマリ・データベース `pdb1` 内に、次の 3 つのカラムを持つ `testtab` という名前のテーブルを作成します。`c1integer`、`c2integer`、および `c3char(10)`。
   たとえば、Adaptive Server で次を実行します。

   ```
   use pdb1
   go
   create table dbo.testtab(c1 int primary key, c2 int,
   c3 char(10))
   go
   ```

   Oracle のマニュアルを参照して Oracle データベースにテーブルを作成してください。

2. Sybase IQ IQSRVR データ・サーバにあるレプリケート・データベース `iqdb` に次のように入力します。

   ```
   use iqdb
   go
   create table dbo.testtab(c1 int primary key, c2 int,
   c3 char(10))
   go
   grant all on dbo.testtab to public
   go
   ```

プライマリ・データベースとレプリケート・データベースへのコネクションの作成

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・コネクションを作成します。

1. プライマリ・データベースへのコネクションを作成します。
   - Adaptive Server – Replication Server の **rs_init** ユーティリティを使用します。『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。
   - Oracle – 詳細については、『異機種間複写ガイド』と、Replication Server Options 製品のマニュアルを参照してください。
2. Sybase IQ レプリケート・データベースへのコネクションを作成します。

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

注意： Sybase IQ へのコネクションの作成には、**rs_init** は使用できません。

この例では、IQSRVR データ・サーバ内の `iqdb` データベースとデフォルト dbmaint の Sybase IQ メンテナンス・ユーザを使用します。

- Adaptive Server の場合

```
create connection to IQSRVR.iqdb
using profile rs_ase_to_iq;standard
set username to dbmaint
set password to dbmaint
go
```

- Oracle の場合

```
create connection to IQSRVR.iqdb
using profile rs_oracle_to_iq;standard
set username to dbmaint
set password to dbmaint
go
```

コマンドが正常に実行されると、次のような出力が表示されます。

```
Connection to 'IQSRVR.iqdb' is created.
```

3. コネクションが機能していることを確認します。

```
admin who
go
```

コネクションが機能していれば、次のような出力が表示されます。

```
Spid   Name       State              Info
----   --------   ----------------   -----------------
 63    DSI EXEC   Awaiting Command   103(1)IQSRVR.iqdb
 62    DSI        Awaiting Message   103 IQSRVR.iqdb
 35    SQM        Awaiting Message   103:0 IQSRVR.iqdb
```

RTL の有効化

データベース・レベルで RTL を有効にします。

1. 指定したデータベースのみに影響するように、データベース・レベルで RTL を有効にして設定するには、次のように入力します。

```
alter connection to IQSRVR.iqdb
set dsi_compile_enable to 'on'
go
```

2. コネクションへの変更を有効にするには、レプリケート Sybase IQ データベースへのコネクションをサスペンドしてレジュームします。

```
suspend connection to IQSRVR.iqdb
go
resume connection to IQSRVR.iqdb
go
```

複写テスト準備のためのテーブルへのマーク付け

プライマリ・データベース内で Sybase IQ データベースへ複写するテーブルにマークを付けます。

下記の例では、dbo は pdb1 プライマリ・データベース内の testtab テーブル所有者です。c1、c2、c3 は testtab 内のカラムで、データ型はそれぞれ int、int、char(10) です。

1. 複写をテストし、挿入が正常に行われることを確認するために、testtab にデータ・ローを挿入します。
   たとえば、Adaptive Server で次を実行します。

   ```
   insert into testtab values(1,1,'testrow 1')
   insert into testtab values(2,2,'testrow 2')
   insert into testtab values(3,3,'testrow 3')
   go
   ```

   挿入が正常に実行されると、次のような出力が表示されます。

   ```
   (1 row affected)
   (1 row affected)
   (1 row affected)
   ```

2. 複写対象の testtab にマークを付けます。
   - Adaptive Server – **sp_setrepdefmode** システム・プロシージャを使用します。
     - Adaptive Server 15.0.3 以降の場合、次を実行します。

       ```
       sp_setrepdefmode testtab,'owner_on'
       go
       ```

     - Adaptive Server 15.0.3. より前の場合、次を実行します。

       ```
       sp_setreptable testtab,'true', 'owner_on'
       go
       ```

   - Oracle – **pdb_setreptable** Replication Agent コマンドを使用して、次を実行します。

     ```
     pdb_setreptable pdb_table, mark, owner
     ```

     使用に関する詳細については、Replication Server Options のマニュアルの『Replication Agent 管理ガイド』の「設定」の「プライマリ・データベース・オブジェクトのマーク付け」の「プライマリ・データベースのテーブルへのマーク付け」を参照してください。

複写定義とサブスクリプションの作成

RTL を有効にして設定したら、Sybase IQ への複写用にマークされているテーブルの複写定義とサブスクリプションを作成します。

1. *repdef_testtab* 複写定義を作成します。RTL をサポートする複写定義に必要なすべての参照制約句を追加します。
   - Adaptive Server の場合：

     ```
     create replication definition repdef_testtab
     with primary at ASE_DS.pdb1
     with primary table named 'testtab'
     with replicate table named dbo.'testtab'
     (c1 int, c2 int, c3 char(10))
     primary key(c1)
     go
     ```

   - Oracle の場合：

     ```
     create replication definition repdef_testtab
     with primary at ORA_DS.pdb1
     with primary table named 'TESTTAB'
     with replicate table named dbo.'testtab'
     (C1 as c1 int, C2 as c2 int, C3 as c3 char(10))
     primary key(C1)
     go
     ```

     注意： デフォルトでは Oracle のオブジェクト名の文字設定はすべて大文字です。例に見られるように、複写定義でオブジェクト名を大文字から小文字に変換できます。Replication Agent for Oracle の設定パラメータ **ltl_character_case** を使用して変換することもできます。詳細については、Replication Server Options のマニュアルの『Replication Agent リファレンス・マニュアル』の「設定パラメータ」の「設定パラメータ・リファレンス」の「**ltl_character_case**」を参照してください。

2. 各テーブルとストアド・プロシージャの複写定義に一致するサブスクリプションを作成します。

   ```
   create subscription sub_testtab for repdef_testtab
   with replicate at IQSRVR.iqdb
   go
   ```

3. Sybase IQ にログインして次を実行し、`testtab` がマテリアライズされたことを確認します。

   ```
   select * from dbo.testtab
   go
   ```

   マテリアライゼーションが正常に行われると、次のような出力が表示されます。

   ```
   c1          c2          c3
   ---------   ---------   ---------
   1           1           testrow 1
   2           2           testrow 2
   3           3           testrow 3
   (3 rows affected)
   ```

参照：

- 参照制約のあるテーブル (158 ページ)

RTL が機能することを検証する

RTL が機能することを確認する方法について説明します。

1. プライマリ・データ・サーバにログインして、testtab に新しいローを挿入するなどのオペレーションを実行します。
   たとえば、Adaptive Server で次を実行します。

   ```
   insert into testtab values(4,4,'testrow 4')
   insert into testtab values(5,5,'testrow 5')
   insert into testtab values(6,6,'testrow 6')
   go
   ```

   次のような出力が表示されます。

   ```
   (1 row affected)
   (1 row affected)
   (1 row affected)
   ```

2. 次を実行して、Sybase IQ にログインし、testtab への変更が Sybase IQ データベースに複写されたことを確認します。

   ```
   select * from dbo.testtab
   go
   ```

   複写が正常に実行されると、次のような出力が表示されます。

   ```
   c1          c2          c3
   ----------  ----------  ----------
   1           1           testrow 1
   2           2           testrow 2
   3           3           testrow 3
   4           4           testrow 4
   5           5           testrow 5
   6           6           testrow 6
   (6 rows affected)
   ```

ステージング・ソリューションから RTL へのマイグレート

Sybase IQ への複写に現在ステージング・ソリューションを使用している場合、Real-Time Loading ソリューションにマイグレートします。

このシナリオでは、pdb がプライマリ・データベース、PRS がプライマリ Replication Server、RRS がレプリケート Replication Server、staging_db がステージング・データベース、iqdb がレプリケート Sybase IQ データベースという複写構成を想定しています。このシナリオでのデータ・フローは、次のとおりです。

```
pdb -----> PRS -----> RRS -----> staging_db -----> iqdb
```

ステージング・ソリューションからのマイグレートの準備

ステージング・ソリューションからマイグレートする前に、いくつかのタスクを実行する必要があります。

1. プライマリおよびレプリケート Replication Servers をバージョン 15.5 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 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。
2. `pdb` に流れ込むトランザクションがないこと、およびマイグレート中は複写システムがクワイス状態にあることを確認します。
   a) Replication Server で実行して、すべてのプライマリ・データベースとシステム・データベースの Replication Agent を停止します。

      ```
      suspend log transfer from all
      ```

   b) Adaptive Server を RSSD として使用している場合は、次のコマンドで RSSD の RepAgent を停止します。

      ```
      sp_stop_rep_agent rssd_name
      ```

   c) 次を実行して、Replication Server のキューが排出され、Replication Server がクワイスされていることを確認します。

      ```
      admin quiesce_check
      ```

      Replication Server がまだクワイスされていない場合は、**admin quiesce_force_rsi** で再試行します。Replication Server がクワイス状態になっていなければ、データが失われる可能性があります。
3. `pdb` と `iqdb` が同期していることを確認します。

   データベース間の再同期は、ステージング・データベースにすべてのデータが複写されてから、`iqdb` にステージング・データベースのデータをロードすることによって行います。データベース間の再同期を行わない場合は、`iqdb` のパージとマテリアライズを行う必要があります。
4. Sybase IQ サーバがレプリケート Replication Server に接続してデータを抽出できるように、Sybase IQ の `interfaces` ファイルにそのレプリケート Replication Server のエントリを追加します。

Real-Time Loading ソリューションにマイグレート

ステージング・ソリューションから RTL にマイグレートします。

1. レプリケート Sybase IQ データ・サーバでメンテナンス・ユーザを作成します。または、既存のメンテナンス・ユーザを使用することもできます。

2. 接続プロファイルと手順 1 の *dbmaint* といったメンテナンス・ユーザを使って、レプリケート Replication Server からレプリケート Sybase IQ データベースへのコネクションを作成します。
   - Adaptive Server の場合：

     ```
     create connection to IQSRVR.iqdb
     using profile rs_ase_to_iq;standard
     set username to dbmaint
     set password to dbmaint
     go
     ```
   - Oracle の場合：

     ```
     create connection to IQSRVR.iqdb
     using profile rs_oracle_to_iq;standard
     set username to dbmaint
     set password to dbmaint
     go
     ```
3. プライマリ・データベースで、*dbo* が所有するテーブルが **owner_on** としてマーク付けされていない場合、*dbo* は Sybase IQ 内に存在しないので、Sybase IQ がそのテーブルを検出できるようにそのテーブルの **owner_on** を有効にする必要があります。
   - Adaptive Server
     - Adaptive Server 15.0.3 以降の場合、次を実行します。

       ```
       sp_setrepdefmode testtab, 'owner_on'
       go
       ```
     - Adaptive Server 15.0.3. より前の場合、次を実行します。

       ```
       sp_setreptable testtab, 'true', 'owner_on'
       go
       ```
   - Oracle

     ```
     pdb_setreptable testtab, mark, owner
     go
     ```
4. Adaptive Server の **owner_on** または Oracle の **owner** を有効にしたので、複写定義を作成し直して所有者情報を含めます。
5. テーブル間に参照制約がある場合、Replication Server がその参照制約の存在を計算に入れてバルク適用を適切な順序で実行できるように、参照制約を定義して複写定義を変更する必要があります。
6. レプリケート・データベースへのコネクションを RTL で有効にします。

   ```
   alter connection to iqserver_name.rdb
   set dsi_compile_enable to 'on'
   ```

   コネクションをサスペンドしてレジュームしたら、接続の変更が有効になります。
7. 各テーブルにサブスクリプションを作成します。プライマリ・データベースとレプリケート・データベースが同期している場合は、**without materialization** 句

をサブスクリプションに含めます。それ以外の場合は、マテリアライゼーション時にオートコレクションを有効にする必要があります。

これで、プライマリ・データ・サーバから Sybase IQ に直接複写できるようになります。

参照：
- 参照制約のあるテーブル (158 ページ)

マイグレーション後のクリーンアップ

RTL を使用して複写を有効にして設定した後、ステージング・ソリューションのシステムをクリーンアップします。

1. ステージング・データベースのサブスクリプションを削除します。
2. 使用しない複写定義を削除します。
3. レプリケート Replication Server からステージング・データベースへのコネクションを削除します。
4. ステージング・データベースから Sybase IQ にデータを抽出する環境を終了します。

Replication Server と Sybase IQ InfoPrimer の統合

Replication Server と Sybase IQ InfoPrimer を統合することで、レプリケート Sybase IQ データベースソースとは異なるスキーマを持つプライマリ Adaptive Server データベース間の複写をサポートします。

Sybase IQ InfoPrimer には、データを変換して Sybase IQ データベースにロードする効果的な機能が搭載されていますが、その抽出機能にはレプリケート Sybase IQ データベースのデータを最新の状態に維持するために必要な Replication Server のリアルタイム・モニタリングがありません。Replication Server Real-Time Loading (RTL) 機能では、バルク・オペレーション処理とコンパイルされたオペレーションを使用して、高パフォーマンスのレプリケーションを達成しますが、Replication Server には Sybase IQ InfoPrimer のデータ変換機能とロード機能がありません。Replication Server と Sybase IQ InfoPrimer を統合することで、ソースとは異なるスキーマを持つレプリケート Sybase IQ データベースで Adaptive Server データのほぼリアルタイムのコピーを維持できます。Replication Server と Sybase IQ InfoPrimer の統合ソリューションは、初期データ・マテリアライゼーションと進行中のデータ処理という 2 つの部分で機能します。

マテリアライゼーション

Replication Server と Sybase IQ InfoPrimer の統合ソリューションは、Adaptive Server プライマリ・データベースからレプリケート Sybase IQ データベースへのデータの

ノンアトミック・バルク・マテリアライゼーションを実行します。このマテリアライゼーションは、Replication Server バルク・マテリアライゼーション・オプションに基づいており、必要に応じてオートコレクションを使用します。

Sybase IQ InfoPrimer は、レプリケート Sybase IQ データベースにステージング・テーブルを作成し、各プライマリ・データベース・テーブルに対してマテリアライゼーション処理のデータ抽出手順を実行します。これらのステージ・テーブルに対して変換ストアド・プロシージャが実行され、結果はベース・テーブルに書き込まれます。ベース・テーブルは、エンドユーザ・テーブルとも呼ばれ、ビジネス分析に使用されます。

進行中のデータ処理
Replication Server は、指定されたテーブルに対してマテリアライゼーション・フェーズで作成された同じステージング・テーブルと変換ストアド・プロシージャを使用します。可能な場合、Replication Server はオペレーションをコンパイルしてステージング・テーブルにロードします。その後、Replication Server は変換ストアド・プロシージャを実行して、ベース・テーブルを更新します。こうして、Replication Server はレプリケート Sybase IQ データベースにほぼリアルタイムのデータ・コピーを維持します。

ライセンス
特別なライセンスの要件は Replication Server と Sybase IQ InfoPrimer の統合に適用されます。『Replication Server 新機能ガイド』の「Replication Server バージョン 15.6 ESD #1 の新機能」の「ライセンス」を参照してください。

Replication Server と Sybase IQ InfoPrimer の統合の使用

Sybase IQ InfoPrimer で Replication Server マテリアライゼーション・メソッドを使用して、データを Sybase IQ にマテリアライズし、プライマリ・データに対して行われた更新を処理するよう Replication Server を設定します。

1. マテリアライゼーション前:

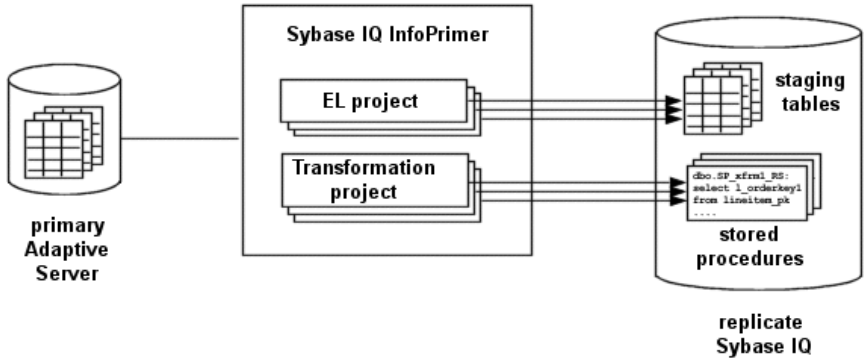
   - Sybase IQ InfoPrimer で EL (抽出、ロード) プロジェクトを作成し、[Replication Server でマテリアライゼーション]を選択します。
     EL プロジェクト・エディタの [RepServer] タブで、プライマリ Replication Server とレプリケート Replication Server (プライマリと異なる場合のみ) のコネクション情報も指定する必要があります。Sybase IQ InfoPrimer によって、[処理] タブにコマンドが追加されます。このコマンドは、変更も削除もしないでください。
     各ソース・テーブルに対して、Sybase IQ InfoPrimer は必要なステージング・テーブル定義を作成します。EL プロジェクト・エディタの [テーブル] タブにある[不足している送信先テーブルを作成]アイコンを選択して、これらの

ステージング・テーブルをレプリケート Sybase IQ データベースに生成します。

注意： 再マテリアライズを試みる場合は、`rs_status` テーブルをクリアする必要があります。

- SQL 変換プロジェクトを作成し、レプリケート Sybase IQ データベースで生成されたステージング・テーブル (挿入、更新、および削除) の各セットの変換をモデル化します。この SQL 変換プロジェクトを使用して、変換の各セットをレプリケート Sybase IQ データベースでストアド・プロシージャとして展開します。

  注意： これらの変換ストアド・プロシージャでは、オペレーションの処理が完了すると、対応するステージング・テーブルがトランケートされます。



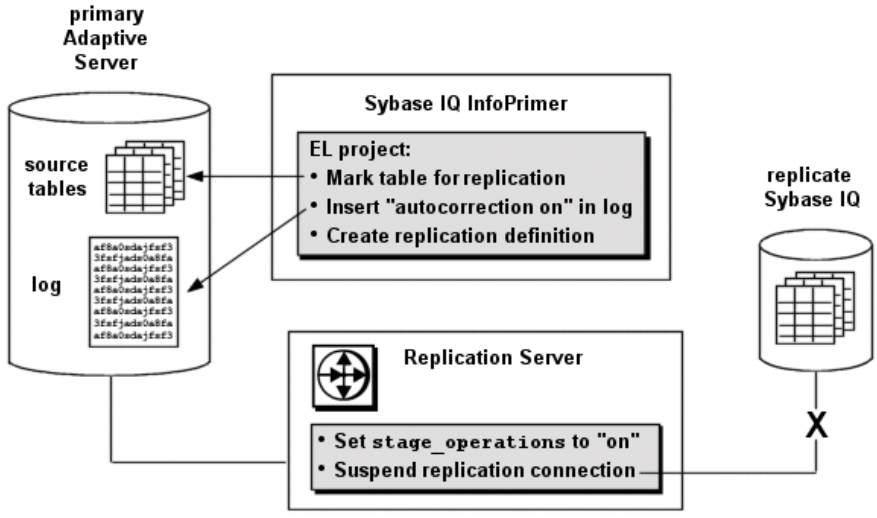
2. Replication Server インスタンスでは、**stage_operations** コネクション・パラメータを使用して、EL プロジェクトで指定されたテーブルのステージ・オペレーションへのレプリケート・データベース・コネクションを設定します。

   注意： **stage_operations** が on に設定されている場合、Replication Server は **dsi_compile_enable** の設定を無視し、そのコネクションの RTL を有効にします。オペレーションはコンパイルされ、**dsi_compile_enable** が on に設定されると、ステージングされます。

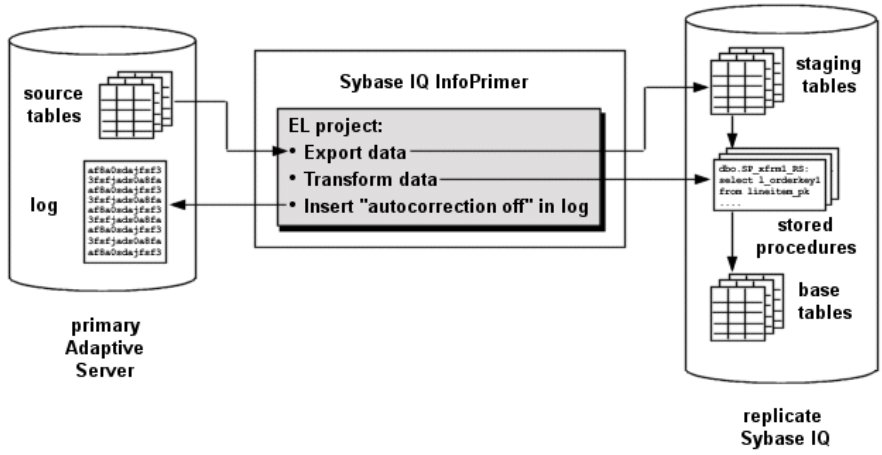
   Sybase IQ InfoPrimer で、EL プロジェクトを実行します。指定された各プライマリ・テーブルについて、EL プロジェクトは以下を実行します。

   a) テーブルをレプリケートするようマーク付けします。

b) プライマリ・データベース・ログに `autocorrection on` レコードを挿入します。その結果、Replication Server レプリケート・データベース・コネクションがサスペンドされます。
c) RSSD でテーブル複写定義を作成します。

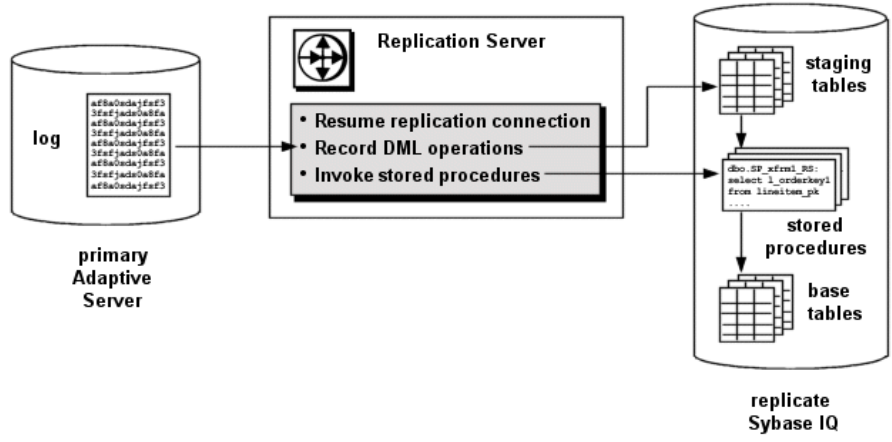


3. Sybase IQ InfoPrimer EL プロジェクトは、各テーブルのプライマリ・データをレプリケート Sybase IQ 上にある対応するステージング・テーブルにエクスポートし、変換ストアド・プロシージャを実行し、`autocorrection off` レコードをプライマリ・データベース・ログに挿入します。



4. Replication Server のレプリケート・データベース・コネクションが再開され、マーク付けされたプライマリ・データベース・テーブルにさらに加えられる変更があれば、Replication Server がレプリケート Sybase IQ データベースでステージング・テーブルと変換ストアド・プロシージャを使用して、それらの変更を処理します。

注意： Sybase IQ InfoPrimer は、データの移行と、ステージング・テーブルおよび変換ストアド・プロシージャの作成にのみ使用されます。Sybase IQ InfoPrimer は、レプリケーションには関与しません。



ベース・テーブル

ベース・テーブルには、レプリケート Sybase IQ データベースでの最終形式でデータが格納されます。

ベース・テーブル・データの発生元は、次のとおりです。

- SQL 変換 ― オペレーションをステージングするよう Replication Server のレプリケート・データベース・コネクションが設定されると、ステージ・テーブルに対して実行される変換ストアド・プロシージャの結果がベース・テーブルに書き込まれます。
- レプリケーション ― あるテーブルがステージングから除外されると、Replication Server はそのステージング・テーブルをバイパスしデータをベース・テーブルに直接レプリケートします。

ステージング・テーブル

プライマリ・テーブルのためにログされるオペレーションをステージングするよう Replication Server のレプリケート・データベース・コネクションを設定すると、これらのオペレーションは可能な場合はコンパイルされ、レプリケート Sybase IQ データベースにあるステージ・テーブルに書き込まれます。

ステージングする各テーブルに対して、3 つのステージング・テーブルが存在し、各ステージング・テーブルは DELETE、INSERT、および UPDATE の各オペレーションに対応します。

- *owner_table_name*_DELETE_RS
- *owner_table_name*_INSERT_RS
- *owner_table_name*_UPDATE_RS

ここで、*owner* と *table_name* は対応するプライマリ・データベース・テーブルの所有者と名前です。これらのテーブルの名前は、EL プロジェクトによって生成され、変更できません。

注意： EL プロジェクトの [テーブル] タブには、insert ステージング・テーブルのみ表示されます。しかし、[テーブル作成] ウィンドウには、指定されたプライマリ・データベース・テーブルに対応する 3 つのステージング・テーブルがすべて表示されます。

Sybase IQ InfoPrimer の EL プロジェクトでどのプライマリ・データベース・テーブルがステージングされるのかを識別する必要があります。ステージングからレプリケート・テーブルを選択的に除外することもできます。ステージングから除外されたテーブルには、対応するステージング・テーブルを作成する必要がないため、データはレプリケート Sybase IQ データベースでプライマリ・テーブルからレプリケート・テーブルにレプリケートされます。

テーブルをステージングするようレプリケート・データベース・コネクションを設定しても、レプリケート Sybase IQ データベースにステージング・テーブルが存

在しない場合は、レプリケート・データベース・コネクションはサスペンドされます。複写定義に identity カラムとして宣言されるカラムが含まれる場合でも、これらのカラムは対応するステージング・テーブルでは identity カラムとして宣言されません。

テーブルのコンパイル
コンパイルできないテーブルでは、コンパイルは実行されません。テーブルがコンパイルできないと見なされるのは、RTL が無効か、ファンクション文字列が変更されているか、または最小カラム・レプリケーションが有効の場合です。コンパイルできないテーブルに対するオペレーションは、順番リストに取得され、コンパイル完了後に対応するレプリケート・テーブルに適用されます。

注意： Replication Server がステージングされたオペレーションをコミットした後、変換ストアド・プロシージャが対応するステージング・テーブルをトランケートします。したがって、Replication Server の **rs_subcmp** ユーティリティを使用してステージング・テーブルを確定化しないでください。

insert ステージング・テーブルの構造
対応する複写定義によって適用される変更とフィルタリングを除き、insert ステージング・テーブルにはプライマリ・テーブルと同じ数のカラムと同じカラム名が格納されます。

delete ステージング・テーブルの構造
delete ステージング・テーブルには、対応する複写定義で指定されたプライマリ・キー・カラムのみが格納されます。

複写定義でプライマリ・キーが指定されない場合、delete ステージング・テーブルには、以下を除くパブリッシュ済みカラムがすべて格納されます。

- 概数値カラム
- 暗号化カラム
- Java カラム
- LOB カラム

注意： テーブル複写定義でプライマリ・キーを指定して、処理を単純化しパフォーマンスを向上させることをお勧めします。

update ステージング・テーブルの構造
update ステージング・テーブルには、対応する複写定義で指定されたプライマリ・キー・カラムごとに 2 つのカラムが格納されます。1 つは変更前のカラム・データ用、もう 1 つは変更後のカラム・データ用です。

update ステージング・テーブルには、複写定義で指定された各非プライマリ・キー・カラムのカラムも格納されます。これらの非プライマリ・キー・カラムの

データに変更が加えられたかどうかを追跡するため、update ステージング・テーブルには 1 個または複数のビットマップ・カラムが格納されます。各ビットマップ・カラムは `int` 型であるため、32 個の非プライマリ・キー・カラムを追跡できます。値 1 はダーティ・ビットであり、データがそのビット位置に対応するカラムで変更されたことを示します。

注意： update ステージング・テーブルの変更前カラムとビットマップ・カラムは、Sybase IQ InfoPrimer の SQL 変換プロジェクトには表示されません。

変換ストアド・プロシージャ

ステージングされるどのプライマリ・データベース・テーブルにも、対応する変換ストアド・プロシージャがレプリケート Sybase IQ データベースに存在します。Replication Server は、ステージング・テーブルに対してこれらのストアド・プロシージャを実行し、結果はベース・テーブルに書き込まれます。

Sybase IQ InfoPrimer の SQL 変換プロジェクトでこれらのストアド・プロシージャによって実行される変換を指定し、それらのストアド・プロシージャをレプリケート Sybase IQ データベースに展開する必要があります。

レプリケート Sybase IQ データベースに存在しないストアド・プロシージャを使用しようとするか、またはストアド・プロシージャの実行に失敗すると、レプリケート・データベース・コネクションがサスペンドされます。

注意： SQL 変換プロジェクトに関与するテーブルをすべて確認できるように、ストアド・プロシージャをレプリケート Sybase IQ データベースに展開する準備が整うまでは、SQL 変換プロジェクトのプロジェクト・プロパティでスキーマを選択しないでください。

パラメータ

Replication Server では、**stage_operations** パラメータと **dsi_stage_all_ops** パラメータを使用して、テーブル・ステージングを制御します。

stage_operations

create connection コマンドまたは **alter connection** コマンドの **stage_operations** パラメータを設定すると、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みます。

レプリケート・データベース・コネクションのステージングを設定できます。
例：

```
create connection to SYDNEY_IQ_RS.iq_db
 using profile rs_ase_to_iq;standard
 set username pubs2_maint
 set password pubs2_maint_pw
 set stage_operations to "on"
```

個々のテーブルについてステージングの有効または無効を指定するには、特定のレプリケート・テーブルに関して **alter connection** コマンドの **stage_operations** パラメータを使用します。例：

```
alter connection to SYDNEY_IQ_RS.iq_db
for replicate table named lineitem_5
set stage_operations to "off"
```

この場合、Replication Server は `lineitem_5` テーブルのオペレーションをステージングしませんが、代わりにオペレーションを通常どおりレプリケートします。

注意： **stage_operations** パラメータは、Sybase IQ レプリケートへのコネクションにのみ設定できます (ここで、**dsi_dataserver_make** パラメータは iq に設定されます)。Sybase IQ コネクション・プロファイルを使用してコネクションを作成すると、**dsi_dataserver_make** コネクション・パラメータが適切に設定されます。

dsi_compile_enable

stage_operations が on に設定されている場合、Replication Server は **dsi_compile_enable** の設定を無視し、そのコネクションの RTL を有効にします。オペレーションはコンパイルされ、**dsi_compile_enable** が on に設定されると、ステージングされます。

dsi_stage_all_ops

alter connection コマンドの **dsi_stage_all_ops** パラメータを使用して、指定したテーブルのオペレーション・コンパイルを回避します。

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 に設定します。例：

```
alter connection to SYDNEY_IQ_RS.iq_db
for replicate table named lineitem_5
set dsi_stage_all_ops to "on"
```

Replication Server のコンポーネント

Replication Server では、Sybase IQ InfoPrimer との統合をサポートするために追加のコンポーネントが必要です。

rs_status テーブル

`rs_status` テーブルは、マテリアライゼーションの進行状況に関する情報を格納します。

カラム	データ型	説明
schema	varchar (255)	マテリアライズされるテーブルの所有者

カラム	データ型	説明
tablename	varchar (255)	マテリアライズされるテーブルの名前
action	varchar (1)	• I ― 初回ロード • A ― オートコレクション・フェーズ • R ― レプリケーション
starttime	timestamp	アクションが開始された時間
endtime	timestamp	アクションが完了した時間
status	varchar (1)	• P ― アクションが進行中 • X ― 実行が完了した • E ― 実行エラー
pid	int	予約済み

たとえば、my_table のオートコレクションが進行中の場合、rs_status には次のようなローが含まれます。

```
schema tablename action starttime                endtime status pid
------ --------- ------ ------------------------ ------- ------ ---
sys    my_table  A      2011-07-11 19:11:25.531          P
```

my_table のオートコレクションが完了すると、rs_status には次のようなローが含まれます。

```
schema tablename action starttime
------ --------- ------ ------------------------
sys    my_table  A      2011-07-11 19:11:25.531
```

```
endtime                  status pid
------------------------ ------ ---
2011-07-11 19:12:14.326 X
```

rs_status データの自動クリーンアップはありません。テーブルの再マテリアライズを試みる前に、rs_status から対応するローを削除する必要があります。

```
delete rs_status where tablename=tablename and schema=owner
```

オートコレクション関数

Replication Server では、**rs_autoc_on**、**rs_autoc_off**、および **rs_autoc_ignore** 関数を使用して、rs_status テーブルを更新します。

『Replication Server リファレンス・マニュアル』の「Replication Server システム関数」を参照してください。

システム変数

rs_autoc_on 関数と **rs_autoc_off** 関数は、`rs_status` テーブルの更新時に 2 つのシステム変数を使用します。

- *rs_deliver_as_name* – オートコレクションの影響を受けるレプリケート・テーブルの名前を指定します。
- *rs_repl_objowner* – オートコレクションの影響を受けるレプリケート・テーブルの所有者を指定します。

デフォルトのデータ型変換

Sybase IQ では、すべての Adaptive Server データ型をネイティブ形式でサポートしているため、Adaptive Server から Sybase IQ へのデータ型変換は必要ありません。

サポートされない機能

Replication Server と Sybase IQ InfoPrimer の統合は、特定の機能とプラットフォームに限定されています。

Replication Server と Sybase IQ InfoPrimer の統合では、以下はサポートされていません。

- Sybase IQ 以外のレプリケート・データベース
- Adaptive Server 以外のプライマリ・データベース
- 複写ストアド・プロシージャ
- カスタム・ファンクション文字列
- RTL によって提供されたもの以外の事前ステージング・オペレーション変換
- レプリケート Sybase IQ データベースで変換ストアド・プロシージャによって実行された変換後に行われる変換

異機種間におけるマルチパス・レプリケーション

複数のレプリケーション・パスを使用して、レプリケーションのスループットとパフォーマンスを向上させ、競合を低減します。

シングルパス・レプリケーション環境では、トランザクションはプライマリ・データベースからレプリケート・データベースに連続して複写されるため、プライマリ・データベースのトランザクションのコミット順が確保され、このためレプリケート・データベースとプライマリ・データベースの一貫性が保たれます。トランザクションをレプリケート・データベースに適用する逐次モードは、複数のアプリケーションがプライマリ・データベースで並列してそれぞれのトランザクションが実行されていたり、複数のプライマリ・データベースから到着するトランザクションがある場合でも変更されません。

同じプライマリ・データベースから生じたすべてのトランザクションを直列化することなく、テーブルのサブセット内でデータの一貫性を維持できる複写環境があります。この環境の典型例は、異なるデータのセットにアクセスする異なるアプリケーションで 1 つのプライマリ・データベースが変更される場合です。特定のアプリケーション内で変更されたテーブルのサブセット内の異なるデータのセットは、引き続き連続して複写されます。異なるテーブルのサブセットのデータは並列して複写することができます。

マルチパス・レプリケーションでは、さまざまなストリームを介したデータのレプリケーションをサポートすると同時に、パス内でのデータ整合性を維持しますが、さまざまなパス間でのコミット順には従いません。

レプリケーション・パスは、Replication Server とプライマリ・データベースまたはレプリケート・データベースの間のコンポーネントとモジュールをすべて含んでいます。マルチパス・レプリケーションでは、プライマリ・データベースから 1 つまたは複数の Replication Server への複数の Replication Agent コネクションのために、複数のプライマリ・レプリケーション・パスを作成できます。また、1 つまたは複数の Replication Server からレプリケート・データベースへのコネクションのために、複数のレプリケート・パスを作成できます。マルチパス・レプリケーションは、ウォーム・スタンバイ環境と Multi-Site Availability (MSA) 環境で設定できます。トランザクションを Replication Server 間の専用ルートで伝達して、共有ルート上での輻輳を回避できます。また、プライマリ・データベースから Replication Server を経由してレプリケート・データベースに至るエンドツーエンドのレプリケーション・パスをオブジェクト (テーブルやストアド・プロシージャなど) 専用にすることができます。

Adaptive Server をマルチパス・レプリケーション・システムのプライマリ・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「マルチパス・レプリケーション」を参照してください。

ライセンス
マルチパス・レプリケーションは、Advanced Services Option の一部としてライセンスされます。『Replication Server インストール・ガイド』の「インストールの計画」の「ライセンスの取得」を参照してください。

システムの稼働条件

表 1 : 異機種間におけるマルチパス・レプリケーション・システムでサポートされているプライマリ・データベースとレプリケート・データベースのペア

プライマリ・データベース	レプリケート・データベース
Adaptive Server	Sybase IQ
Oracle	Sybase IQ
Adaptive Server	Oracle
Oracle	Adaptive Server
Oracle	Oracle

表 2 : サポートされているマルチパス・レプリケーション データベース・バージョン

データベース	サポートされているバージョン
Adaptive Server	15.7 以降
Oracle	Oracle 10g と 11g。Replication Server Options の『Replication Agent リリース・ノート』の「製品の概要」の「製品の互換性」を参照してください。
Sybase IQ	15.1 以降。『Replication Server リリース・ノート』の「製品の互換性」の「Replication Server の相互運用性」を参照してください。

並列トランザクション・ストリーム

マルチパス・レプリケーションは、トランザクションが並行ストリームに分割され、異なるストリーム全体で順番にコミットされない限り、複写パフォーマンスを向上させることができます。

トランザクションを並列レプリケーション・パスに分割して輻輳を削減することにより、複写のパフォーマンスを向上させることができます。トランザクション属性または派生データ値などの並列化ルールに従って、トランザクションを分割できます。たとえば、次の方法を使用できます。

- テーブルまたはストアド・プロシージャなどの特定オブジェクトにパスを割り当てます。オブジェクトをパスにバインドする場合、Replication Agent はパスを介してそのオブジェクトに実行する複写可能なアクションを、複数のレプリケーション・パス設定で定義する Replication Servers に送信します。Adaptive Server の RepAgent および Replication Agent for Oracle では、この複写分散モードがサポートされています。
- プライマリ・データベースでクライアント・コネクションのセッション ID 別にトランザクションを分割します。Adaptive Server の RepAgent は、クライアント・コネクションによるトランザクションの分散をサポートしています。
- 各パスへの Replication Server を使用します。
- オブジェクトをパスにバインドするか、Replication Server 間に専用ルートを作成して、優先度の高い複写に対し専用パスまたは輻輳の少ないパスを割り当てます。

デフォルトおよび代替コネクション

マルチパス・レプリケーションでは、コネクションには、デフォルトおよび 1 つ以上の代替コネクションが含まれます。

Replication Agent からのデータを受け入れるコネクションはプライマリ・コネクションで、データをデータベースに適用するコネクションはレプリケート・コネクションです。デフォルトまたは代替コネクションは、プライマリまたはレプリケート・コネクションのいずれかです。

デフォルトのコネクションは、データベースを複写ドメインに追加する際に、Replication Server から特定のプライマリ・データベースまたはレプリケート・データベースに作成するコネクションです。データ・サーバが Adaptive Server またはサポートされている ASE 以外のサーバであるかに応じて、**rs_init**、Replication Manager の Sybase Central プラグイン、**create connection**、または **create connection ... using profile** を使用して、デフォルトのコネクションを作成できます。

デフォルトのコネクションは、データ・サーバ名およびデータベース名を *dataserver.database* の形式で、コネクション名として使用します。ここで *dataserver* および *database* は、それぞれ実際のデータ・サーバ名およびデータベース名です。

必須のデフォルトのコネクションを作成後、複数の代替コネクションを作成できます。各代替コネクションには、それぞれユニークな名前が必要です。

代替コネクションを作成後、コネクションのプロパティを変更、またはコネクションを削除できます。すべてのコネクションのステータスを表示し、そのコネクションのサブスクリプションを作成することもできます。

代替コネクションを作成する場合、ユーザ ID が有効なユーザであることが必要です。Sybase IQ レプリケート・データベースにコネクションを作成する場合、デフォルトのコネクションおよび各代替コネクションに対し、コネクション・プロファイル、コネクション・プロファイルのバージョン、およびユニークなメンテナンス・ユーザ名を指定する必要があります。

Sybase IQ のファイル要件のインタフェース

Sybase IQ データ・サーバの `interfaces` ファイル (Windows では `sql.ini` ファイル) に、Replication Server のエントリを作成します。単一の Sybase IQ server に対してレプリケートする複数の Replication Servers がある場合、すべての Replication Servers を `interfaces` ファイルに追加する必要があります。

さまざまな Sybase IQ マルチプレックス・ノードへのコネクションを作成している場合、影響を受ける各ノードのエントリをレプリケート Replication Server の `interfaces` ファイルに作成します。

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配します。レプリケート Replication Server に専用ルートを作成して、優先順位の高いトランザクションを複写するか、特定のプライマリ・コネクションのために輻輳の少ないパスを維持できます。

共有ルートは、プライマリ Replication Server から始まるすべてのプライマリ・コネクションのためにトランザクションを分配する、プライマリ Replication Server とレプリケート Replication Server の間です。共有ルートは特定のコネクションにバインドされません。専用ルートにバインドされないコネクションは、使用可能な任意の有効共有ルートを使用します。

専用ルートを作成できるのは、以下の条件が満たされた場合だけです。

- プライマリ Replication Server から送信先 Replication Server への共有ルートが存在し、この共有ルートが直接ルートである。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。
- 共有ルートが有効であり、サスペンドされていない。
- 共有ルートのバージョンが 1570 またはそれ以降である。

専用ルートの作成

create route および **with primary at** 句を使用して、専用ルートを作成します。

たとえば、`NY_DS.pdb1` プライマリ・コネクションのために、RS_NY プライマリ Replication Server と RS_LON レプリケート Replication Server との間の専用ルートを作成するには、RS_NY で次のように入力します。

```
create route to RS_LON
 with primary at NY_DS.pdb1
 go
```

特定のコネクションのために専用ルートを作成すると、そのコネクションから送信先 Replication Server へのトランザクションはすべて、その専用ルートを通るようになります。

専用ルートを管理するコマンド

create route、**drop route**、**resume route**、および **suspend route** を使用して、専用ルートを管理およびモニタします。

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。

『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」で、「**create route**」、「**drop route**」、**suspend route**」、および「**resume route**」を参照してください。

コマンド	構文	コマンドおよびパラメータの変更

create route	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 *route_param* to '*value*']...]
	 [set *security_param* to '*value*'
	  [set *security_param* to '*value*']...]}
		専用ルートの作成時にユーザ ID を使用する場合、ユーザ ID は有効なユーザであることが必要です。

コマンド	構文	コマンドおよびパラメータの変更
drop route	drop route to *dest_replication_server* [with primary at *dataserver.database*] [with nowait]	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。 **警告！ with nowait** 句は、最後の手段としてのみ使用してください。 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。 句を使用する場合、以前の送信先サイトで **sysadmin purge_route_at_replicate** を使用して、送信先のシステム・テーブルからサブスクリプションおよびルート情報を削除します。 『Replication Server 管理ガイド 第 1 巻』の「ルートの管理」の「ルートの削除」の「**drop route** コマンド」を参照してください。
suspend route	suspend route to *dest_replication_server* [with primary at *dataserver.database*]	

コマンド	構文	コマンドおよびパラメータの変更
resume route	`resume route to dest_replication_server [with primary at dataserver.database] [skip transaction with large message]`	

専用ルート情報の表示

admin who を使用して、Replication Server 間の専用ルートの情報を表示します。

次の例では、NY_DS.pdb1 プライマリ・コネクションのために、RS_NY プライマリ Replication Server と RS_LON レプリケート Replication Server との間に専用ルートがあります。2 つの Replication Server に **admin who** と入力すると、次のように表示されます。

- RS_LON では次のように表示されます。

```
Spid Name      State               Info
45 SQT         Awaiting Wakeup     103:1  DIST NY_DS.pdb1
13 SQM         Awaiting Message    103:1 NY_DS.pdb1
32 REP AGENT   Awaiting Command    NY_DS.pdb1
16 RSI         Awaiting Wakeup     RS_LON
11 SQM         Awaiting Message    16777318:0 RS_LON
55 RSI         Awaiting Wakeup     RS_LON(103) /* Dedicated RSI
thread */
53 SQM         Awaiting Message    16777318:103 RS_LON(103) /
*Dedicated RSI outbound queue */
```

- RS_NY では次のように表示されます。

```
Spid Name     State                Info
37 RSI USER   Awaiting Command    RS_NY(103) /*Dedicated RSI user
*/
32 RSI USER   Awaiting Command    RS_NY
```

『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。

異機種間マルチパス・レプリケーションのシナリオ

シナリオを使用して、Adaptive Server、Oracle、および Sybase IQ データベース間に複数のレプリケーション・パスを構築します。

Adaptive Server から Sybase IQ へのマルチパス・レプリケーション

Adaptive Server プライマリ・データベースから Sybase IQ レプリケート・データベースまでのエンドツーエンドの複写のために、プライマリとレプリケートの 2 つのコネクションを持つマルチパス・レプリケーション・システムを設定します。

このシナリオの複写システムは、ASE_DS primary Adaptive Server の `pdb` データベース、IQSRVR レプリケート Sybase IQ データ・サーバ (`iqdb` データベースが含まれている)、PRS プライマリ Replication Server、`testtab1` テーブルと `testtab2` テーブルで構成されます。

1. `pdb` プライマリ・データベースで、2 つのレプリケーション・パスを介して複写するテーブルまたはストアド・プロシージャを 2 セット作成または選択します。
2. `pdb` プライマリ Adaptive Server データベースへのデフォルトのコネクションを作成します。

   create connection または **rs_init** を使用して、デフォルトのコネクションを作成します。『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。
3. マルチスレッド Adaptive Server RepAgent と Adaptive Server RepAgent の複数パスを有効にします。

   また、メモリを設定して Adaptive Server RepAgent に使用可能なバッファを送信することもできます。『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・レプリケーション」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」を参照してください。

   a) マルチスレッド RepAgent を有効にします。
      プライマリ Adaptive Server で、次を入力します。

      ```
      sp_config_rep_agent pdb, 'multithread rep agent', 'true'
      ```

   b) RepAgent へのレプリケーション・パスの数を設定します。
      たとえば、2 つのパスを有効にするには、次を入力します。

      ```
      sp_config_rep_agent pdb, 'max number of replication paths', '2'
      ```
4. `pdb` プライマリ・データベースから PRS Replication Server への代替レプリケーション・パスを作成します。

   『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「マルチパス・レプリケーション」の「複数のプライマリ・レプリケーション・パス」の「複数のプライマリ・レプリケーション・パスの作成」を参照してください。

a) pdb_conn2 という名前の代替レプリケーション・パスを作成します。場所は pdb と PRS Replication Server の間になります。
プライマリ Adaptive Server で、次を入力します。

```
sp_replication_path "pdb", 'add',
"pdb_conn2", "PRS",
"dbmaint2", "dbmaint2pwd"
```

(オプション) 複数の Replication Server の物理パスにバインドされているデータやオブジェクト・バウンドの分散に使用できる論理パスを作成します。『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「マルチパス・レプリケーション」の「複数のプライマリ・レプリケーション・パス」の「論理プライマリ・レプリケーション・パスの作成」を参照してください。

b) Replication Server に対応する代替プライマリ・コネクションを作成し、代替パス名 pdb_conn2 を持つ同じ Adaptive Server データ・サーバ名を使用します。
Replication Server で次のように入力します。

```
create alternate connection to ASE_DS.pdb
named ASE_DS.pdb_conn2
set error class to rs_sqlserver_error_class
set function string class to rs_sqlserver_function_class
set username to dbmaint2
set password to dbmaint2pwd
with primary only
```

複写システムには、2 つのプライマリ・レプリケーション・パスがあります。デフォルトは、ASE_DS.pdb と ASE_DS.pdb_conn2 です。

5. **admin show_connections, 'primary'** を使用して、作成したプライマリ・コネクションを表示します。
6. メンテナンス・ユーザが dbmaint1 の Sybase IQ データベースへのデフォルトのレプリケート・コネクションを作成します。

   コネクション・プロファイルとコネクション・プロファイルのバージョン、およびデフォルトのコネクションと各代替コネクションの一意のメンテナンス・ユーザ名を指定する必要があります。

   ```
   create connection to IQSRVR.iqdb
   using profile rs_ase_to_iq;standard
   set username to dbmaint1
   set password to dbmaint1
   go
   ```
7. コネクションが **admin who** で実行していることを確認します。
8. メンテナンス・ユーザが dbmaint2 で、iqdb Sybase IQ データベースへの IQSRVR.pdb_conn2 という名前の代替レプリケート・コネクションを作成します。

```
create alternate connection to IQSRVR.iqdb
named IQSRVR.pdb_conn2
using profile rs_ase_to_iq;standard
set username to dbmaint2
set password to dbmaint2pwd
go
```

(オプション) 使用可能な Sybase IQ マルチプレックス・ノードへの代替コネクションを作成します。コネクション名がユニークであることを確認します。たとえば、IQSRVR2 Sybase IQ ノードの `iqdb2` データベースへの `pdb_conn3` 代替コネクションを作成するには、次を入力します。

```
create alternate connection to IQSRVR2.pdb_conn3
named IQSRVR2.iqdb2_conn1
using profile rs_ase_to_iq;standard
set username to dbmaint3
set password to dbmaint3pwd
go
```

9. **admin show_connections, 'replicate'** を使用して作成したレプリケート・コネクションを表示します。
10. Sybase IQ へのデフォルトおよび代替レプリケーション・コネクションの RTL を有効にします。
    a) デフォルトのコネクションの RTL を有効にします。

    ```
    alter connection to IQSRVR.iqdb
    set dsi_compile_enable to 'on'
    go
    ```

    b) `IQSRVR.pdb_conn2` 代替コネクションの RTL を有効にします。

    ```
    alter connection to IQSRVR.pdb_conn2
    set dsi_compile_enable to 'on'
    go
    ```

    c) レプリケート Sybase IQ データベースへのコネクションをサスペンドしてレジュームします。

    ```
    suspend connection to IQSRVR.iqdb
    go
    suspend connection to IQSRVR.pdb_conn2
    go
    resume connection to IQSRVR.iqdb
    go
    resume connection to IQSRVR.pdb_conn2
    go
    ```

    configure replication server を使用して RTL を有効にする場合や、すべてのコネクションを同時にサスペンドしてレジュームする場合は、Replication Server を再起動します。
11. RTL の処理を設定し、RTL パフォーマンスを調整するため、オプションとしてパラメータ値を設定します。
12. プライマリ・データベースで、レプリケーションのため、`testtab1` テーブルおよび `testtab2` テーブルにマーク付けします。

```
sp_setreptable testtab1,'true'
go
sp_setreptable testtab2,'true'
go
```

13. プライマリ・データベースで分散モードをオブジェクトのバインド別分散に設定します。

```
sp_config_rep_agent pdb, 'multipath distribution model', 'object'
```

14. Replication Server をクワイス状態にし、RepAgent を再起動します。
『Replication Server 管理ガイド 第 1 巻』の「複写システムの管理」にある「Replication Server のクワイス」の「複写システムのクワイス」を参照してください。
15. testtab2 テーブルを ASE_DS.pdb_conn2 代替コネクションにバインドします。

```
sp_replication_path pdb, 'bind', "table", "dbo.testtab2",
"pdb_conn2"
```

オブジェクトを代替レプリケーション・パスにバインドすることしかできません。testtab1 テーブルなど、代替レプリケーション・パスへバインドしないオプションはすべて、代わりにデフォルトのパスを使用します。

16. マーク付けされたテーブルの複写定義を作成します。RTL をサポートする複写定義に必要なすべての参照制約句を追加します。
testtab1 テーブルの **repdef_testtab1** 複写定義を作成するには、次を入力します。

```
create replication definition repdef_testtab1
with primary at ASE_DS.pdb1
with primary table named 'testtab1'
with replicate table named dbo.'testtab1'
(c1 int, c2 int, c3 char(10))
primary key(c1)
go
```

testtab2 テーブルの **repdef_testtab2** 複写定義を作成するには、次を入力します。

```
create replication definition repdef_testtab2
with primary at ASE_DS.pdb1
with primary table named 'testtab2'
with replicate table named dbo.'testtab2'
(c1 int, c2 int, c3 char(10))
primary key(c1)
go
```

すべての複写定義は、デフォルトのプライマリ・コネクションを参照します。

17. 各テーブルの複写定義に一致し、コネクションを指定するサブスクリプションを作成します。
    a) **repdef_testtab1** 複写定義の **sub_testtab1** サブスクリプションを作成し、レプリケート・トランザクションへのデフォルトのコネクションを指定します。

```
create subscription sub_testtab1 for repdef_testtab1
with replicate at IQSRVR.iqdb
without materialization
go
```

b) **repdef_testtab2** 複写定義の **sub_testtab2** サブスクリプションを作成して、レプリケート・トランザクションへの IQSRVR.iqdb_conn2 代替コネクションを指定します。

```
create subscription sub_testtab2 for repdef_testtab2
with replicate at IQSRVR.pdb_conn2
without materialization
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「マルチパス・レプリケーション」の「複写定義およびサブスクリプション」で「コネクション間でのサブスクリプションの移動」を参照してください。

参照：

- レプリケート・データ・サーバとしての Sybase IQ (133 ページ)
- RTL 設定パラメータ (147 ページ)

Oracle から Sybase IQ へのマルチパス・レプリケーション

Oracle プライマリから Sybase IQ レプリケートへのエンドツーエンドの複写のために、2 つのプライマリとレプリケートのコネクションを持つマルチパス・レプリケーション・システムを設定します。

前提条件

このシナリオでは、次がインストールされ、設定されていることを前提としています。

- LogMiner のあるプライマリ Oracle データベース
- 2 つの Replication Agent インスタンス
- Replication Server
- レプリケート Sybase IQ データベース

手順

1. プライマリ Oracle データベースで、2 つのレプリケーション・パスを介して複写するテーブルを 2 つ作成または選択します。
2. プライマリ Oracle データベースへのデフォルトのコネクションを作成します。

```
create connection to pds.pdb
set error class rs_sqlserver_error_class
set function string class rs_oracle_function_class
set username muser
```

```
set password mpwd
with log transfer on,
dsi_suspended
go
```

構文の説明は次のとおりです。

- *pds* は、Replication Agent で指定された **rs_source_ds** パラメータの値です。
- *pdb* は、Replication Agent で指定された **rs_source_db** の値です。
- *muser* は、プライマリ Oracle データベースのメンテナンス・ユーザです。
- *mpwd* は、メンテナンス・ユーザのパスワードです。

3. プライマリ Oracle データベースへの代替コネクションを作成します。

```
create alternate connection to pds.pdb
named pds.conn2
set error class rs_sqlserver_error_class
set function string class rs_oracle_function_class
set username muser
set password mpwd
with primary only
go
```

4. コネクション・プロファイルを使用して、レプリケート Sybase IQ データベースへのコネクションを作成します。

```
create connection to rds.rdb
using profile rs_oracle_to_iq;standard
set username muser
set password mpwd
go
```

構文の説明は次のとおりです。

- *rds* は、レプリケート Sybase IQ サーバの名前です。
- *rdb* は、レプリケート Sybase IQ データベースです。
- *muser* は、レプリケート Sybase IQ データベースのメンテナンス・ユーザです。
- *mpwd* は、レプリケート Sybase IQ のメンテナンス・ユーザのパスワードです。

5. レプリケート Sybase IQ データベースへの代替コネクションを作成します。

```
create alternate connection to rds.rdb
named rds.conn2
using profile rs_oracle_to_iq;standard
set username muser2
set password mpwd
go
```

- 作成する Sybase IQ データベースへの各代替コネクションに対し、異なるメンテナンス・ユーザ名を使用する必要があります。異なる Replication Server から Sybase IQ データベースへのコネクションを作成する場合でも、ユニークなメンテナンス・ユーザ名を使用していることを確認します。

- 代替コネクションでのレプリケート Sybase IQ サーバに対して、以前使用した名前とは異なる名前を使用する場合、Replication Server の interfaces ファイルへの異なる名前のエントリが必要です。

6. `create object` パーミッションを Replication Server の **rs_username** に付与します。

```
grant create object to rs_username
go
```

ここで、*rs_username* は、Replication Agent が Replication Server へのアクセスに使用するユーザ・ログイン名です。

7. Replication Agent の単一インスタンスでは、**ra_admin_owner**、**ra_admin_prefix**、**ra_admin_instance_prefix**、**rs_source_ds**、および **rs_source_db** パラメータを設定します。

```
ra_config ra_admin_owner, ra_user_1
ra_config ra_admin_prefix, ra_
ra_config ra_admin_instance_prefix, ri1
ra_config rs_source_ds, pds
ra_config rs_source_db, pdb
```

パラメータの説明は次のとおりです。

- `ra_user_1` は、Replication Agent インスタンスが使用する目的で、プライマリ・データベース内での共有オブジェクトおよびインスタンス・オブジェクトの作成に使用されるユーザ名です。このユーザ名は、プライマリ・データ・サーバで定義済みであることが必要です。
- `ra_` は、プライマリ・データベースで共有オブジェクトを識別するために使用されるプレフィクスです。このプレフィクスは、3 文字以下です。
- `ri1` は、レプリケーション・グループ内でこの Replication Agent インスタンスを一意に識別するプレフィクスです。
- **rs_source_ds** の値は **rs_source_db** の値と組み合わされると、この Replication Agent インスタンスが Replication Server への接続に使用するコネクション名を形成します。

8. Replication Agent インスタンスを初期化します。

```
ra_admin init
```

9. Replication Agent の他のインスタンスでは、**ra_admin_owner**、**ra_admin_prefix**、**ra_admin_instance_prefix**、**rs_source_ds**、および **rs_source_db** パラメータを設定します。

```
ra_config ra_admin_owner, ra_user_1
ra_config ra_admin_prefix, ra_
ra_config ra_admin_instance_prefix, ri2
ra_config rs_source_ds, pds
ra_config rs_source_db, conn2
```

パラメータの説明は次のとおりです。

- `ri2` は、レプリケーション・グループ内でこの Replication Agent インスタンスを一意に識別するプレフィクスです。
- **ra_admin_owner** の値と **ra_admin_prefix** の値は、レプリケーション・グループ内の他のすべての Replication Agent インスタンスと同じです。
- **rs_source_ds** の値は **rs_source_db** の値と組み合わされると、この Replication Agent インスタンスが Replication Server への接続に使用するコネクション名を形成します。

10. Replication Agent インスタンスを初期化します。

```
ra_admin init
```

11. 手順 1 からの複写対象の 2 つのテーブルにマークを付けます。`ri1` で識別される Replication Agent インスタンス上では、次のようにします。

```
pdb_setreptable ptab1, mark
go
```

`ri2` で識別される Replication Agent インスタンス上では、次のようにします。

```
pdb_setreptable ptab2, mark
go
```

ここで、`ptab1` および `ptab2` は、複写されるプライマリ・データベース・テーブルです。

12. プライマリ Oracle データベースに対し、2 つの複写定義を作成します。
たとえば、`ptab1` テーブルの `ptab1_repdef` 複写定義を作成するには、次を実行します。

```
create replication definition ptab1_repdef
with primary at pds.pdb
with all tables named 'ptab1'
...
go
```

`ptab2` テーブルの `ptab2_repdef` 複写定義を作成するには、次を実行します。

```
create replication definition ptab2_repdef
with primary at pds.pdb
with all tables named 'ptab2'
...
go
```

注意： これらの複写定義には、デフォルトのプライマリ・コネクション名を使用する必要があります。

プライマリ・コネクションと代替プライマリ・コネクションが異なる Replication Server に存在する場合、それぞれの Replication Server に複写定義を作成します。

13. Replication Agent のインスタンスを再開します。

```
resume
```

Replication Agent インスタンスの再開に失敗した場合、LogMiner がインストールされ設定されているかを確認します。『Replication Agent プライマリ・データベース・ガイド』の「Replication Agent for Oracle」の「Oracle 固有の考慮事項」の「Oracle トランザクションおよびオペレーションのトラブルシューティング」の「ra_dumptran と ra_helpop を使用するための Replication Agent と Oracle の設定」を参照してください。

14. デフォルトのレプリケート・コネクションに対し、サブスクリプションを作成します。
たとえば、`ptab1_repdef` 複写定義の `ptab1_sub` サブスクリプションを作成するには、次を実行します。

```
create subscription ptab1_sub
for ptab1_repdef
with replicate at rds.rdb
without materialization
go
```

15. 代替レプリケート・コネクションに対し、サブスクリプションを作成します。
たとえば、`ptab2_repdef` 複写定義の `ptab2_sub` サブスクリプションを作成するには、次を実行します。

```
create subscription ptab2_sub
for ptab2_repdef
with replicate at rds.conn2
without materialization
go
```

参照：

- レプリケート・データ・サーバとしての Sybase IQ (133 ページ)

Adaptive Server から Oracle へのマルチパス・レプリケーション

Adaptive Server プライマリから Oracle レプリケートまでのエンドツーエンドの複写のために、2 つのプライマリとレプリケートのパスを持つマルチパス・レプリケーション・システムを設定します。

前提条件

このシナリオでは、次がインストールされ、設定されていることを前提としています。

- プライマリ Adaptive Server データベースおよび RepAgent スレッド
- Replication Server
- ExpressConnect for Oracle
- レプリケート Oracle データベース

手順

1. プライマリ Adaptive Server データベースで、2 つのレプリケーション・パスを介して複写するテーブルまたはストアド・プロシージャを 2 セット作成または選択します。これらのトランザクションのセットは、並列レプリケーション・パスに分割可能であることが必要です。
2. プライマリ Adaptive Server データベースへのデフォルトのコネクションを作成します。

   create connection または **rs_init** を使用して、デフォルトのコネクションを作成します。『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。
3. マルチスレッド Adaptive Server RepAgent と Adaptive Server RepAgent の複数パスを有効にします。

   また、メモリを設定して Adaptive Server RepAgent に使用可能なバッファを送信することもできます。『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・レプリケーション」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」を参照してください。

   a) マルチスレッド RepAgent を有効にします。
   プライマリ Adaptive Server で次を入力します。

   ```
   sp_config_rep_agent pdb, 'multithread rep agent', 'true'
   ```

   ここで、*pdb* はプライマリ Adaptive Server のデータベースです。

   b) RepAgent へのレプリケーション・パスの数を設定します。
   たとえば、2 つのパスを有効にするには、次を入力します。

   ```
   sp_config_rep_agent pdb, 'max number of replication paths', '2'
   ```
4. プライマリ・データベースからの代替レプリケーション・パスを作成します。

   『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「マルチパス・レプリケーション」の「複数のプライマリ・レプリケーション・パス」の「複数のプライマリ・レプリケーション・パスの作成」を参照してください。

   a) プライマリ・データベースと Replication Server の間に pdb_conn2 という名前の代替レプリケーション・パスを作成します。
   プライマリ・データベースで、次を入力します。

   ```
   sp_replication_path "pdb", 'add',
   "pdb_conn2", "PRS",
   "muser", "mpwd"
   ```

   説明は次のとおりです。

   - *PRS* は、Replication Server です。

- *muser* は、メンテナンス・ユーザです。
- *mpwd* は、メンテナンス・ユーザのパスワードです。

(オプション) 複数の Replication Server の物理パスにバインドされているデータやオブジェクト・バウンドの分散に使用できる論理パスを作成します。『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「マルチパス・レプリケーション」の「複数のプライマリ・レプリケーション・パス」の「論理プライマリ・レプリケーション・パスの作成」を参照してください。

b) Replication Server に対応する代替プライマリ・コネクションを作成し、代替パス名 `pdb_conn2` を持つ同じ Adaptive Server データ・サーバ名を使用します。
Replication Server で次を入力します。

```
create alternate connection to pds.pdb
named pds.pdb_conn2
set error class to rs_sqlserver_error_class
set function string class to rs_sqlserver_function_class
set username to muser
set password to mpwd
with primary only
```

ここで *pds* はプライマリ Adaptive Server です。

複写システムには、2つのプライマリ・レプリケーション・パス (デフォルトと `pdb_conn2`) が含まれます。

5. **admin show_connections, 'primary'** を使用して、作成したプライマリ・コネクションを表示します。
6. レプリケート Oracle データベースの `tnsnames.ora` ファイルを Replication Server の `RS_installation_directory¥ REP-15_5¥connector¥oraoci¥network¥admin` ディレクトリにコピーして、Replication Server を再起動します。
7. ExpressConnect for Oracle を介して、レプリケート Oracle データベースへのコネクションを作成します。

```
create connection to tns_alias_name.rdb
using profile rs_oracle_to_oracle;eco
set username muser
set password mpwd
set dsi_dataserver_make to 'ora'
set dsi_connector_type to 'oci'
set batch to 'off'
go
```

構文の説明は次のとおりです。

- *tns_alias_name* は、レプリケート Oracle データベースの `tnsnames.ora` ファイルで定義されたレプリケート Oracle データベースのエイリアス名です。
- *rdb* は、`tnsnames.ora` ファイルの上記の **tns_alias_name** と対になるレプリケート Oracle System ID (SID) です。デフォルト値は ORCL です。
- *muser* は、レプリケート Oracle データベースのメンテナンス・ユーザです。
- *mpwd* は、レプリケート Oracle のメンテナンス・ユーザのパスワードです。

デフォルトのコネクションの命名の詳細については、『インストールおよび設定ガイド』を参照してください。

8. ExpressConnect for Oracle を介して、レプリケート Oracle データベースへの代替コネクションを作成します。

```
create alternate connection to tns_alias_name.rdb
named tns_alias_name.conn2
set error class rs_oracle_error_class
set function string class rs_oracle_function_class
set username muser
set password mpwd
set dsi_dataserver_make to 'ora'
set dsi_dataserver_type to 'oci'
set batch to 'off'
set dsi_proc_as_rpc to 'on'
go
```

9. プライマリ・データベースで、`ptab1` テーブルおよび `ptab2` テーブルをレプリケーション用にマーク付けします。

```
sp_setreptable ptab1,'true'
go
sp_setreptable ptab2,'true'
go
```

10. プライマリ・データベースで分散モードをオブジェクトのバインド別分散に設定します。

```
sp_config_rep_agent pdb, 'multipath distribution model', 'object'
```

11. Replication Server をクワイス状態にし、RepAgent を再起動します。

『Replication Server 管理ガイド 第 1 巻』の「複写システムの管理」にある「Replication Server のクワイス」の「複写システムのクワイス」を参照してください。

12. `ptab2` テーブルを `pdb_conn2` 代替コネクションにバインドします。

```
sp_replication_path pdb, 'bind', "table", "dbo.ptab2",
"pdb_conn2"
```

オブジェクトを代替レプリケーション・パスにバインドすることしかできません。`ptab1` テーブルのような代替レプリケーション・パスにバインドしないすべてのオブジェクトは、代わりにデフォルト・パスを使用します。

13. プライマリ Adaptive Server データベースに 2 つの複写定義を作成します。

たとえば、ptab1 テーブルの ptab1_repdef 複写定義を作成するには、次を実行します。

```
create replication definition ptab1_repdef
with primary at pds.pdb
with all tables named 'ptab1'
...
go
```

ptab2 テーブルの ptab2_repdef 複写定義を作成するには、次を実行します。

```
create replication definition ptab2_repdef
with primary at pds.pdb
with all tables named 'ptab2'
...
go
```

注意： これらの複写定義には、デフォルトのプライマリ・コネクション名を使用する必要があります。

プライマリ・コネクションと代替プライマリ・コネクションが異なる Replication Server に存在する場合、それぞれの Replication Server に複写定義を作成します。

14. デフォルトのレプリケート・コネクションに対し、サブスクリプションを作成します。
たとえば、ptab1_repdef 複写定義の ptab1_sub サブスクリプションを作成するには、次を実行します。

```
create subscription ptab1_sub
for ptab1_repdef
with replicate at tns_alias_name.rdb
without materialization
go
```

15. 代替レプリケート・コネクションに対し、サブスクリプションを作成します。
たとえば、ptab2_repdef 複写定義の ptab2_sub サブスクリプションを作成するには、次を実行します。

```
create subscription ptab2_sub
for ptab2_repdef
with replicate at tns_alias_name.conn2
without materialization
go
```

参照：

- レプリケート・データ・サーバとしての Oracle (117 ページ)

Oracle から Adaptive Server へのマルチパス・レプリケーション

Oracle プライマリから Adaptive Server 複写へのエンドツーエンドの複写のために、2 つのプライマリとレプリケートのコネクションを持つマルチパス・レプリケーション・システムを設定します。

前提条件

このシナリオでは、次がインストールされ、設定されていることを前提としています。

- LogMiner のあるプライマリ Oracle データベース
- 2 つの Replication Agent インスタンス
- Replication Server
- レプリケート Adaptive Server データベース

手順

1. プライマリ Oracle データベースで、2 つのレプリケーション・パスを介して複写するテーブルを 2 つ作成または選択します。
2. **rs_init** を使用して、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. プライマリ Oracle データベースへのデフォルトのコネクションを作成します。

   ```
   create connection to pds.pdb
   set error class rs_sqlserver_error_class
   set function string class rs_oracle_function_class
   set username muser
   set password mpwd
   with log transfer on,
   dsi_suspended
   go
   ```

   構文の説明は次のとおりです。
   - *pds* は、Replication Agent で指定された **rs_source_ds** パラメータの値です。
   - *pdb* は、Replication Agent で指定された **rs_source_db** の値です。
   - *muser* は、プライマリ Oracle データベースのメンテナンス・ユーザです。
   - *mpwd* は、メンテナンス・ユーザのパスワードです。
4. プライマリ Oracle データベースへの代替コネクションを作成します。

   ```
   create alternate connection to pds.pdb
   named pds.conn2
   set error class rs_sqlserver_error_class
   set function string class rs_oracle_function_class
   set username muser
   set password mpwd
   ```

```
with primary only
go
```

5. レプリケート Adaptive Server データベースへのコネクションを作成します。

```
create connection to rds.rdb
set error class rs_sqlserver_error_class
set function string class rs_sqlserver_function_class
set username muser
set password mpwd
go
```

構文の説明は次のとおりです。

- *rds* は、レプリケート Adaptive Server のデータ・サーバの名前です。
- *rdb* は、レプリケート Adaptive Server のデータベースです。
- *muser* は、レプリケート Adaptive Server のデータベースのメンテナンス・ユーザです。
- *mpwd* は、メンテナンス・ユーザのパスワードです。

6. レプリケート Adaptive Server のデータベースへの代替コネクションを作成します。

```
create alternate connection to rds.rdb
named rds.conn2
set error class rs_sqlserver_error_class
set function string class rs_sqlserver_function_class
set username muser
set password mpwd
go
```

7. `create object` パーミッションを Replication Server の **rs_username** に付与します。

```
grant create object to rs_username
go
```

ここで、*rs_username* は、Replication Agent が Replication Server へのアクセスに使用するユーザ・ログイン名です。

8. Replication Agent の単一インスタンスでは、**ra_admin_owner**、**ra_admin_prefix**、**ra_admin_instance_prefix**、**rs_source_ds**、および **rs_source_db** パラメータを設定します。

```
ra_config ra_admin_owner, ra_user_1
ra_config ra_admin_prefix, ra_
ra_config ra_admin_instance_prefix, ri1
ra_config rs_source_ds, pds
ra_config rs_source_db, pdb
```

パラメータの説明は次のとおりです。

- `ra_user_1` は、Replication Agent インスタンスが使用する目的で、プライマリ・データベース内での共有オブジェクトおよびインスタンス・オブ

ジェクトの作成に使用されるユーザ名です。このユーザ名は、プライマリ・データ・サーバで定義済みであることが必要です。
- `ra_` は、プライマリ・データベースで共有オブジェクトを識別するために使用されるプレフィクスです。このプレフィクスは、3 文字以下です。
- `ri1` は、レプリケーション・グループ内でこの Replication Agent インスタンスを一意に識別するプレフィクスです。
- **rs_source_ds** の値は **rs_source_db** の値と組み合わされると、この Replication Agent インスタンスが Replication Server への接続に使用するコネクション名を形成します。

9. Replication Agent インスタンスを初期化します。

```
ra_admin init
```

10. Replication Agent の他のインスタンスでは、**ra_admin_owner**、**ra_admin_prefix**、**ra_admin_instance_prefix**、**rs_source_ds**、および **rs_source_db** パラメータを設定します。

```
ra_config ra_admin_owner, ra_user_1
ra_config ra_admin_prefix, ra_
ra_config ra_admin_instance_prefix, ri2
ra_config rs_source_ds, pds
ra_config rs_source_db, conn2
```

パラメータの説明は次のとおりです。
- `ri2` は、レプリケーション・グループ内でこの Replication Agent インスタンスを一意に識別するプレフィクスです。
- **ra_admin_owner** の値と **ra_admin_prefix** の値は、レプリケーション・グループ内の他のすべての Replication Agent インスタンスと同じです。
- **rs_source_ds** の値は **rs_source_db** の値と組み合わされると、この Replication Agent インスタンスが Replication Server への接続に使用するコネクション名を形成します。

11. Replication Agent インスタンスを初期化します。

```
ra_admin init
```

12. 手順 1 からの複写対象の 2 つのテーブルにマークを付けます。`ri1` で識別される Replication Agent インスタンス上では、次のようにします。

```
pdb_setreptable ptab1, mark
go
```

`ri2` で識別される Replication Agent インスタンス上では、次のようにします。

```
pdb_setreptable ptab2, mark
go
```

ここで、`ptab1` および `ptab2` は、複写されるプライマリ・データベース・テーブルです。

13. プライマリ Oracle データベースに対し、2 つの複写定義を作成します。
たとえば、ptab1_repdef 複写定義を ptab1 テーブルに作成します。

```
create replication definition ptab1_repdef
with primary at pds.pdb
with all tables named 'ptab1'
...
go
```

ptab2_repdef 複写定義を ptab2 テーブルに作成するには、次を実行します。

```
create replication definition ptab2_repdef
with primary at pds.pdb
with all tables named 'ptab2'
...
go
```

注意： これらの複写定義には、デフォルトのプライマリ・コネクション名を使用する必要があります。

プライマリ・コネクションと代替プライマリ・コネクションが異なる Replication Server に存在する場合、それぞれの Replication Server に複写定義を作成します。

14. Replication Agent のインスタンスを再開します。

```
resume
```

Replication Agent インスタンスの再開に失敗した場合、LogMiner がインストールされ設定されているかを確認します。『Replication Agent プライマリ・データベース・ガイド』の「Replication Agent for Oracle」の「Oracle 固有の考慮事項」の「Oracle トランザクションおよびオペレーションのトラブルシューティング」の「ra_dumptran と ra_helpop を使用するための Replication Agent と Oracle の設定」を参照してください。

15. デフォルトのレプリケート・コネクションに対し、サブスクリプションを作成します。
たとえば、ptab1_repdef 複写定義の ptab1_sub サブスクリプションを作成するには、次を実行します。

```
create subscription ptab1_sub
for ptab1_repdef
with replicate at rds.rdb
without materialization
go
```

16. 代替レプリケート・コネクションに対し、サブスクリプションを作成します。
たとえば、ptab2_repdef 複写定義の ptab2_sub サブスクリプションを作成するには、次を実行します。

```
create subscription ptab2_sub
for ptab2_repdef
with replicate at rds.conn2
```

```
without materialization
go
```

Oracle から Oracle へのマルチパス・レプリケーション

Oracle プライマリから Oracle レプリケートへのエンドツーエンドの複写のために、2 つのプライマリとレプリケートのコネクションを持つマルチパス・レプリケーション・システムを設定します。

前提条件

このシナリオでは、次がインストールされ、設定されていることを前提としています。

- LogMiner のあるプライマリ Oracle データベース
- 2 つの Replication Agent インスタンス
- Replication Server
- ExpressConnect for Oracle
- レプリケート Oracle データベース

手順

1. プライマリ Oracle データベースで、2 つのレプリケーション・パスを介して複写するテーブルを 2 つ作成または選択します。
2. プライマリ Oracle データベースへのデフォルトのコネクションを作成します。

   ```
   create connection to pds.pdb
   set error class rs_sqlserver_error_class
   set function string class rs_oracle_function_class
   set username muser
   set password mpwd
   with log transfer on,
   dsi_suspended
   go
   ```

   構文の説明は次のとおりです。
   - *pds* は、Replication Agent で指定された **rs_source_ds** パラメータの値です。
   - *pdb* は、Replication Agent で指定された **rs_source_db** の値です。
   - *muser* は、プライマリ Oracle データベースのメンテナンス・ユーザです。
   - *mpwd* は、メンテナンス・ユーザのパスワードです。
3. プライマリ Oracle データベースへの代替コネクションを作成します。

   ```
   create alternate connection to pds.pdb
   named pds.conn2
   set error class rs_sqlserver_error_class
   set function string class rs_oracle_function_class
   set username muser
   set password mpwd
   ```

```
with primary only
go
```

4. レプリケート Oracle データベースの `tnsnames.ora` ファイルを Replication Server の `RS_installation_directory¥ REP-15_5¥connector¥oraoci¥network¥admin` ディレクトリにコピーして、Replication Server を再起動します。
5. ExpressConnect for Oracle を介して、レプリケート Oracle データベースへのコネクションを作成します。

```
create connection to tns_alias_name.rdb
using profile rs_oracle_to_oracle;eco
set username muser
set password mpwd
set dsi_dataserver_make to 'ora'
set dsi_connector_type to 'oci'
set batch to 'off'
go
```

   構文の説明は次のとおりです。
   - *tns_alias_name* は、レプリケート Oracle データベースの `tnsnames.ora` ファイルで定義されたレプリケート Oracle データベースのエイリアス名です。
   - *rdb* は、`tnsnames.ora` ファイルの上記の **tns_alias_name** と対になるレプリケート Oracle System ID (SID) です。デフォルト値は ORCL です。
   - *muser* は、レプリケート Oracle データベースのメンテナンス・ユーザです。
   - *mpwd* は、レプリケート Oracle のメンテナンス・ユーザのパスワードです。

   デフォルトのコネクションの命名の詳細については、『インストールおよび設定ガイド』を参照してください。
6. ExpressConnect for Oracle を介して、レプリケート Oracle データベースへの代替コネクションを作成します。

```
create alternate connection to tns_alias_name.rdb
named tns_alias_name.conn2
set error class rs_oracle_error_class
set function string class rs_oracle_function_class
set username muser
set password mpwd
set dsi_dataserver_make to 'ora'
set dsi_dataserver_type to 'oci'
set batch to 'off'
set dsi_proc_as_rpc to 'on'
go
```

7. `create object` パーミッションを Replication Server の **rs_username** に付与します。

```
grant create object to rs_username
go
```

ここで、*rs_username* は、Replication Agent が Replication Server へのアクセスに使用するユーザ・ログイン名です。

8. Replication Agent の単一インスタンスでは、**ra_admin_owner**、**ra_admin_prefix**、**ra_admin_instance_prefix**、**rs_source_ds**、および **rs_source_db** パラメータを設定します。

```
ra_config ra_admin_owner, ra_user_1
ra_config ra_admin_prefix, ra_
ra_config ra_admin_instance_prefix, ri1
ra_config rs_source_ds, pds
ra_config rs_source_db, pdb
```

パラメータの説明は次のとおりです。

- `ra_user_1` は、Replication Agent インスタンスが使用する目的で、プライマリ・データベース内での共有オブジェクトおよびインスタンス・オブジェクトの作成に使用されるユーザ名です。このユーザ名は、プライマリ・データ・サーバで定義済みであることが必要です。
- `ra_` は、プライマリ・データベースで共有オブジェクトを識別するために使用されるプレフィクスです。このプレフィクスは、3 文字以下です。
- `ri1` は、レプリケーション・グループ内でこの Replication Agent インスタンスを一意に識別するプレフィクスです。
- **rs_source_ds** の値は **rs_source_db** の値と組み合わされると、この Replication Agent インスタンスが Replication Server への接続に使用するコネクション名を形成します。

9. Replication Agent インスタンスを初期化します。

```
ra_admin init
```

10. Replication Agent の他のインスタンスでは、**ra_admin_owner**、**ra_admin_prefix**、**ra_admin_instance_prefix**、**rs_source_ds**、および **rs_source_db** パラメータを設定します。

```
ra_config ra_admin_owner, ra_user_1
ra_config ra_admin_prefix, ra_
ra_config ra_admin_instance_prefix, ri2
ra_config rs_source_ds, pds
ra_config rs_source_db, conn2
```

パラメータの説明は次のとおりです。

- `ri2` は、レプリケーション・グループ内でこの Replication Agent インスタンスを一意に識別するプレフィクスです。
- **ra_admin_owner** の値と **ra_admin_prefix** の値は、レプリケーション・グループ内の他のすべての Replication Agent インスタンスと同じです。
- **rs_source_ds** の値は **rs_source_db** の値と組み合わされると、この Replication Agent インスタンスが Replication Server への接続に使用するコネクション名を形成します。

11. Replication Agent インスタンスを初期化します。

```
ra_admin init
```

12. 手順 1 からの複写対象の 2 つのテーブルにマークを付けます。`ri1` で識別される Replication Agent インスタンス上では、次のようにします。

```
pdb_setreptable ptab1, mark
go
```

`ri2` で識別される Replication Agent インスタンス上では、次のようにします。

```
pdb_setreptable ptab2, mark
go
```

ここで、`ptab1` および `ptab2` は、複写されるプライマリ・データベース・テーブルです。

13. プライマリ Oracle データベースに対し、2 つの複写定義を作成します。
たとえば、`ptab1` テーブルの `ptab1_repdef` 複写定義を作成するには、次を実行します。

```
create replication definition ptab1_repdef
with primary at pds.pdb
with all tables named 'ptab1'
...
go
```

`ptab2` テーブルの `ptab2_repdef` 複写定義を作成するには、次を実行します。

```
create replication definition ptab2_repdef
with primary at pds.pdb
with all tables named 'ptab2'
...
go
```

注意： これらの複写定義には、デフォルトのプライマリ・コネクション名を使用する必要があります。

プライマリ・コネクションと代替プライマリ・コネクションが異なる Replication Server に存在する場合、それぞれの Replication Server に複写定義を作成します。

14. Replication Agent のインスタンスを再開します。

```
resume
```

Replication Agent インスタンスの再開に失敗した場合、LogMiner がインストールされ設定されているかを確認します。『Replication Agent プライマリ・データベース・ガイド』の「Replication Agent for Oracle」の「Oracle 固有の考慮事項」の「Oracle トランザクションおよびオペレーションのトラブルシューティング」の「ra_dumptran と ra_helpop を使用するための Replication Agent と Oracle の設定」を参照してください。

15. デフォルトのレプリケート・コネクションに対し、サブスクリプションを作成します。
    たとえば、ptab1_repdef 複写定義の ptab1_sub サブスクリプションを作成するには、次を実行します。
    ```
    create subscription ptab1_sub
    for ptab1_repdef
    with replicate at tns_alias_name.rdb
    without materialization
    go
    ```
16. 代替レプリケート・コネクションに対し、サブスクリプションを作成します。
    たとえば、ptab2_repdef 複写定義の ptab2_sub サブスクリプションを作成するには、次を実行します。
    ```
    create subscription ptab2_sub
    for ptab2_repdef
    with replicate at tns_alias_name.conn2
    without materialization
    go
    ```

参照：

- レプリケート・データ・サーバとしての Oracle (117 ページ)

Oracle に対する異機種ウォーム・スタンバイ

*ウォーム・スタンバイ・アプリケーション*は、データベースとそのバックアップ・コピーとして機能するデータベースで構成される 1 組のデータベースです。クライアント・アプリケーションは「*アクティブ・データベース*」を更新し、Replication Server はアクティブ・データベースのコピーとして「*スタンバイ・データベース*」を管理します。

アクティブ・データベースで障害が発生した場合、またはアクティブ・データベースやデータ・サーバをメンテナンスする必要がある場合は、スタンバイ・データベースに切り替えると、クライアント・アプリケーションがほとんど中断されることなく処理を再開できます。

スタンバイ・データベースが常にアクティブ・データベースとの一貫性を保つように、Replication Server では、アクティブ・データベースのトランザクション・ログから取得したトランザクション情報が再生成されます。スタンバイ・データベースへのデータの複写には、サブスクリプションは必要ありません。

Oracle に対するウォーム・スタンバイの動作

ウォーム・スタンバイ・アプリケーションでは、Replication Server から 1 つの論理データベースへのコネクションとして、アクティブ・データベースとスタンバイ・データベースが複写システムに示されます。

このウォーム・スタンバイ・アプリケーションでは、次の処理が実行されます。

- クライアント・アプリケーションが、アクティブ・データベースでトランザクションを実行する。
- アクティブ・データベースの Replication Agent が、トランザクション・ログからトランザクションを取得して Replication Server に転送する。
- Replication Server が、スタンバイ・データベースでトランザクションを実行する。
- 場合によっては、Replication Server は送信先データベースやリモート Replication Server にもトランザクションをコピーする。

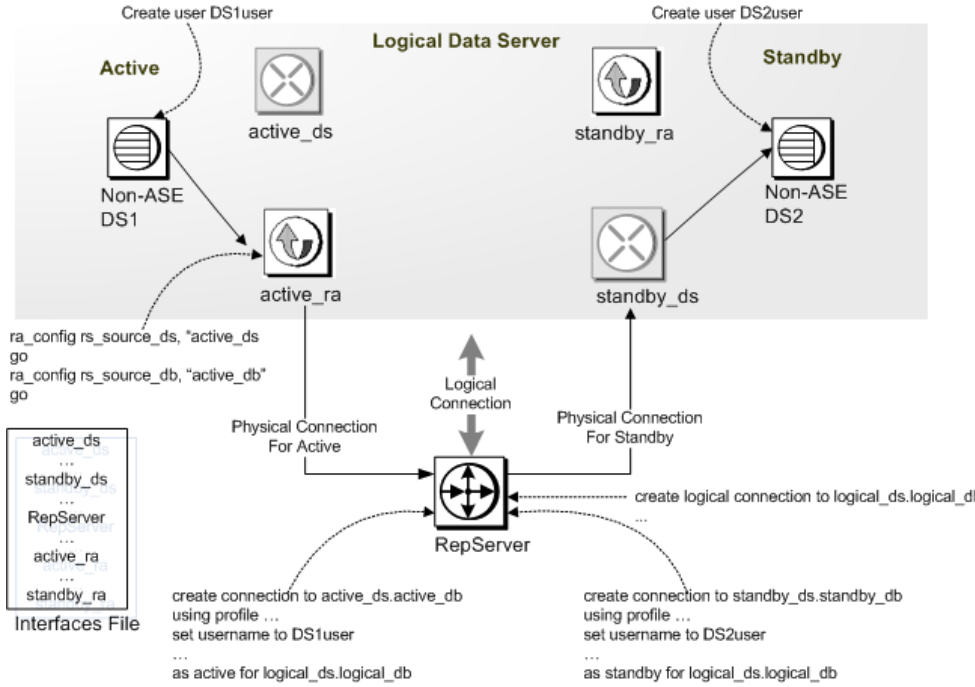
多くの Replication Server アプリケーションでは、次のことが当てはまります。

- プライマリ・データベースは、複写定義とサブスクリプションを使用して他のデータベースにコピーされるデータの送信元である。
- 送信先データベースは、プライマリ (送信元) データベースからデータを受信する。

データベース・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。

ウォーム・スタンバイ・アプリケーション

ウォーム・スタンバイ・アプリケーションの一例における通常のオペレーションを示します。



ウォーム・スタンバイの要件と制限

Replication Server のすべてのウォーム・スタンバイ・アプリケーションに対して適用されるいくつかの要件と制限を理解します。

- 1 つの Replication Server で、アクティブ・データベースとスタンバイ・データベースの両方を管理します。アクティブ・データベースとスタンバイ・データベースはいずれも、同じベンダで作成されたものでなければなりません。
- Replication Server は、クライアント・アプリケーションをスタンバイ・データベースに切り替えません。
- アクティブ・データベースとスタンバイ・データベースのためのデータ・サーバは、それぞれ別のマシンで実行します。アクティブ・データベースとスタン

バイ・データベースを、同じデータ・サーバまたはハードウェア・リソース上で管理すると、ウォーム・スタンバイ機能の利点が損なわれます。
- Replication Server は、異なるプラットフォーム間のウォーム・スタンバイ複写をサポートしていません。
- Sybase では、アクティブ・データベースとスタンバイ・データベース内のテーブルにプライマリ・キーを定義することをお勧めします。

スタンバイ・データベースを管理するためのファンクション文字列

Replication Server は、スタンバイ・データベースへのコネクションであるスタンバイ DSI に対して、システム提供ファンクション文字列クラス **rs_oracle_function_class** を使用します。

ファンクション文字列クラスには、次が含まれます。

- **rs_marker** – スタンバイ・コネクションのために複写を有効化する必要がある、アクティブ・データベースのトランザクション・ログにおけるポイントをマーク付けします。マーカより後のものがすべて複製されている間、マーカより前のものはすべて複製されません。
- **rs_repl_off** – 現在のセッションの複写を無効にします。
- **rs_triggers_reset –** 現在のセッションにトリガを起動するレプリケート・データベースのトリガをすべて無効にします。
- **rs_trunc_set** – Replication Agent for Oracle によって使用されるトランケーション・ポイントをトランザクション・ログの末尾に移動します。

ウォーム・スタンバイ・アプリケーションの複写情報

Replication Agent では、いくつかの方法で Oracle スタンバイ・データベースへの複写を有効にできます。

Replication Server がスタンバイ・データベースにコピーする情報のレベルとタイプは選択するメソッドによって異なりますが、次のいずれかを選択します。

- **pdb_setrepddl** – データベースに格納されているシステム・テーブルを変更する DDL コマンドとプロシージャの複写ができるようにします。DDL コマンドを使用すると、テーブルやビューなどのデータベース・オブジェクトを作成、変更、削除できます。サポートされる DDL システム・プロシージャーは、データベース・オブジェクトに関する情報に影響を与え、DDL ユーザによってスタンバイ・データベースで実行されます。

- **pdb_setreptable** – すべてのユーザ・テーブルまたは指定されたテーブルに複写のマークを付けます。

オプションで、スタンバイ・データベースに複写するユーザ・ストアド・プロシージャを Replication Agent に指示するために **pdb_setrepproc** を使用することもできます。

Replication Agent for Oracle の設定パラメータに関する詳細については、『Replication Agent リファレンス・マニュアル』の「設定パラメータ」を参照してください。

ウォーム・スタンバイ・データベースの設定

ウォーム・スタンバイ・アプリケーションのためのデータベースの設定

前提条件
データベースを設定する前に、次の作業を実行します。

- アクティブ・データベースとスタンバイ・データベースの両方を管理する Replication Server をインストールします。1 つの Replication Server で、アクティブ・データベースとスタンバイ・データベースの両方を管理します。
- Oracle に接続するには、ECDA または ExpressConnect for Oracle を設定します。ECDA を使用する場合、スタンバイ・サイトとプライマリ・サイトのそれぞれで 1 つのコピーを実行し、Oracle データベースと通信するように設定する必要があります。
- Replication Agent を設定し、アクティブ・データベースとスタンバイ・データベースの両方に対して管理モードで実行していることを確認します。
- アクティブ・データベースとスタンバイ・データベースの両方に、DDL ユーザ名を定義し、両方の Replication Agent で構成されていることを確認します。**ddl_username** パラメータは、スタンバイ・データベースまたはターゲット・データベースに DDL コマンドを複製するために、LTL に含まれるデータベースのユーザ名です。このユーザには、ターゲット・データベースで複写されたすべての DDL コマンドを実行するパーミッションが必要です。DDL ユーザは、メンテナンス・ユーザとは異なる必要があります。また、DDL のユーザは、アクティブ・データベースで実行したユーザとして DDL コマンドを実行するには、**alter session** パーミッションも必要です。**ddl_password** パラメータは、データベース・ユーザ名に対応するパスワードです。

手順

1. アクティブ・データベースとスタンバイ・データベースの両方によって使用されるために、1 つの論理コネクションを作成します。
2. Replication Server **create connection** コマンドを使用して、複写システムにアクティブなデータベースを追加します。アクティブ・データベースは、すでに複写システムに追加されている場合は追加する必要はありません。
3. Replication Server **create connection** コマンドを使用して、複写システムにスタンバイ・データベースを追加します。

論理コネクションの作成

ウォーム・スタンバイ・アプリケーションの論理コネクションを作成します。

1. **sa** パーミッションを持つログイン名を使用して、ウォーム・スタンバイ・データベースを管理する Replication Server にログインします。
2. **create logical connection** コマンドを実行します。

   logical_ds.logical_db

   ```
   create logical connection to logical_ds.logical_db
   ```

   データ・サーバ名とデータベース名には、有効なオブジェクト名を指定できます。一般的な値には、論理コネクションを物理 Oracle 実装に関連付けるために、Oracle システム識別子 (SID) が含まれることもあります。

論理コネクション名の指定

logical_ds.logical_db を使用して、論理コネクションに名前を付けます。

論理コネクションに名前を付けるメソッドは、次のように 2 つあります。

- アクティブ・データベースが複写システムに追加されていない場合、論理コネクションには、アクティブ・データベースとは異なる名前を使用します。論理コネクションと物理コネクションにユニークな名前を使用すると、アクティブ・データベースを簡単に切り替えられます。
- アクティブ・データベースがすでに複写システムに追加されている場合、論理コネクション名には、アクティブ・データベースの *data_server* 名と *database* 名を使用します。論理コネクションは、この物理データベースを参照する既存の複写定義とサブスクリプションをすべて継承します。

ウォーム・スタンバイ・アプリケーションに対して複写定義またはサブスクリプションを作成する場合は、物理コネクションではなく論理コネクションを指定します。論理コネクションを指定すると、Replication Server は現在のアクティブ・データベースを参照できます。

アクティブ・データベースへの ReplicationAgent の初期化

isql を使用して、Replication Agent for Oracle (RAO) のインスタンスを開始し、接続します。

1. 次を実行して、ソース Oracle データベースのアーカイブ・ログ・ファイルを設定します。

```
ra_config pdb_include_archives, true
go
ra_config pdb_archive_path, <path-to-oracle-archive-directory>
go
```

2. 次を実行して、Replication Agent のプライマリ・データベースへのコネクションを設定します。

```
ra_config pds_host_name, <the host name of the source oracle>
go
ra_config pds_port_number <the port number of the source oracle>
go
ra_config pds_database_name,<the source oracle database name>
go
ra_config  pds_username, <the oracle user for Replication Agent>
go
ra_config pds_password, <password>
go
test_connection PDS
go
```

   コネクションが正常に確立されると、次のような出力が表示されます。

```
Type   Connection

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

PDS    succeeded
```

3. 次を実行して、Replication Agent の Replication Server へのコネクションを設定します。

```
ra_config rs_host_name, <the host name of the Replication Server>
go
ra_config rs_port_number, <the port number of the
Replication Server>
go
ra_config rs_username, <the Replication Server user for
Replication Agent>
go
ra_config rs_password, <password>
go
ra_config rs_source_ds ',' <the DCON server name>
go
ra_config rs_source_db ',' <the source oracle database name>
go
```

注意： ExpressConnect for Oracle を使用している場合、DirectConnect サーバ名を Oracle インスタンスの名前に置き換えます。例：

```
ra_config rs_source_ds, 'ordb'
go
rs_config rs_source_db, 'ordb'
go
```

4. 次を実行して、Replication Agent の ERSSD へのコネクションを設定します。

```
ra_config rssd_host_name <the host name of the ERSSD>
go
ra_config rssd_port_number, <the port number of the ERSSD>
go
ra_config rssd_username, <the ERSSD user for
Replication Agent>
go
ra_config rssd_password, <password>
go
ra_config rssd_database_name, <the database name of the ERSSD>
go
test_connection RS
go
```

   コネクションが正常に確立されると、次のような出力が表示されます。

```
Type   Connection

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

RS     succeeded
```

5. Replication Server の文字セットが Replication Agent のものと同じではない場合、次を実行して、Replication Server の文字セットを更新します。

```
ra_config rs_charset, <the charset of the Replication Server>
```

6. 次を実行して、複写のマークが付けられた各テーブルに複写定義を作成します。

```
ra_config pdb_auto_create_repdefs, true
go
```

7. 次を実行して、ユーザ・テーブルの自動的なマーク付けを設定します。

```
ra_config pdb_automark_tables, true
go
```

8. 次を実行して、Replication Agent トランザクション・ログを初期化します。

```
pdb_xlog init
```

9. アクティブ・データベースに対して DDL 複写を有効にします。次のように入力します。

```
pdb_setrepddl enable
```

注意： DDL 複写が有効であっても、一部の DDL コマンドはフィルタされます。DDL 複写を有効にする場合、**ddl_password** および **ddl_username** も設定する必要があります。

10. 次を実行して、Replication Agent が初期化される前に、作成されたテーブルに対して複写定義を作成します。

```
rs_create_repdef all
go
```

注意： 複写システムにすでに追加しているデータベースをアクティブ・データベースとして指定する場合、論理コネクションを作成するときにアクティブ・データベースの RepAgent はサスペンドされます。

- 次を実行して、Replication Agent をレジュームします。

```
resume
go
```

複写システムへのアクティブ・データベースの追加

アクティブ・データベースへのコネクションを作成します。

1. 適正なパーミッションを持つログイン名を使って Replication Server にログインします。
2. 次のコマンドを実行します。

```
create connection to active_ds.active_db
using profile …
set username to …
set password to …
with log transfer on
as active for logical_ds.logical_db
```

ExpressConnect for Oracle を使用している場合は、次を実行します。

```
create connection to ordb.ordb/*oracle data server name. database
name*/
using profile rs_oracle_to_oracle;eco
set username to ...
set password to ...
with log transfer on
as active for logical_ds.logical_db
go
```

または、ECDA を使用している場合は、次を実行します。

```
create connection to dco2active.ordb/*dco instance name.database
name*/
using profile rs_oracle_to_oracle;ecda
set username to ...
set password to ...
with log transfer on
```

```
as active for logical_ds.logical_db
go
```

スタンバイ・データベースの初期化

ダンプとロードの手法を使用して、アクティブ・データベースのデータでスタンバイ・データベースを初期化します。

アクティブ・データベースからのデータのダンプ、およびスタンバイ・データベースへのロードの方法については、Oracle のマニュアルを参照してください。

スタンバイ・データベースへの ReplicationAgent の初期化

isql を使用して、Replication Agent for Oracle (RAO) のインスタンスを開始し、接続します。

1. 次を実行して、スタンバイ Oracle データベースのアーカイブ・ログ・ファイルのパスを設定します。

   ```
   ra_config pdb_include_archives, true
   go
   ra_config pdb_archive_path, <path-to-oracle-archive-directory>
   go
   ```

2. 次を実行して、Replication Agent のスタンバイ・データベースへのコネクションを設定します。

   ```
   ra_config pds_host_name, <the host name of the standby oracle>
   go
   ra_config pds_port_number <the port number of the standby oracle>
   go
   ra_config pds_database_name,<the standby oracle database name>
   go
   ra_config  pds_username, <the oracle user for Replication Agent>
   go
   ra_config pds_password, <password>
   go
   test_connection PDS
   go
   ```

   コネクションが正常に確立されると、次のような出力が表示されます。

   ```
   Type   Connection

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

   PDS    succeeded
   ```

3. 次を実行して、Replication Agent の Replication Server へのコネクションを設定します。

   ```
   ra_config rs_host_name, <the host name of the Replication Server>
   go
   ra_config rs_port_number, <the port number of the
   Replication Server>
   go
   ```

```
ra_config rs_username, <the Replication Server user for
Replication Agent>
go
ra_config rs_password, <password>
go
ra_config rs_source_ds ','<the DCON server name>
go
ra_config rs_source_db ','<the standby oracle database name>
go
```

注意： ExpressConnect for Oracle を使用している場合、DirectConnect サーバ名を Oracle インスタンスの名前に置き換えます。例：

```
ra_config rs_source_ds, 'ordb'
go
rs_config rs_source_db, 'ordb'
go
```

4. 次を実行して、Replication Agent の ERSSD へのコネクションを設定します。

```
ra_config rssd_host_name <the host name of the ERSSD>
go
ra_config rssd_port_number, <the port number of the ERSSD>
go
ra_config rssd_username, <the ERSSD user for
Replication Agent>
go
ra_config rssd_password, <password>
go
ra_config rssd_database_name, <the database name of the ERSSD>
go
test_connection RS
go
```

   コネクションが正常に確立されると、次のような出力が表示されます。

```
Type   Connection

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

RS     succeeded
```

5. Replication Server の文字セットが Replication Agent のものと同じではない場合、次を実行して、Replication Server の文字セットを更新します。

```
ra_config rs_charset, <the charset of the Replication Server>
```

6. 次を実行して、複写のマークが付けられた各テーブルに複写定義を作成します。

```
ra_config pdb_auto_create_repdefs, true
go
```

7. 次を実行して、ユーザ・テーブルの自動的なマーク付けを設定します。

```
ra_config pdb_automark_tables, true
go
```

8. 次を実行して、Replication Agent トランザクション・ログを初期化します。

```
pdb_xlog init
```

9. アクティブ・データベースに対して DDL 複写を有効にします。

```
pdb_setrepddl enable
```

注意： DDL 複写が有効であっても、一部の DDL コマンドはフィルタされます。DDL 複写を有効にする場合、**ddl_password** および **ddl_username** も設定する必要があります。

10. スタンバイ・モードで動作するように、Replication Agent を設定します。スタンバイ・モードで動作するように、**ra_standby** 設定パラメータを “true” に設定します。

```
ra_config ra_standby,'true'
go
```

スタンバイ・データベースへのコネクションの作成

スタンバイ・データベース・コネクションの作成

1. 適正なパーミッションを持つログイン名を使って Replication Server にログインします。
2. 次のコマンドを実行します。

```
create connection to standby_ds.standby_db
using profile …
set username to …
set password to …
with log transfer on
as standby for logical_ds.logical_db
```

アクティブ・データベースとスタンバイ・データベースへのコネクションのレジューム

アクティブ・データベースとスタンバイ・データベースへのコネクションをレジュームします。

スタンバイ・データベースの初期化中、Replication Server は、アクティブ・データベースへのコネクションをサスペンドします。

1. 次のコマンドを Replication Server で実行します。

```
resume connection to active_ds.active_db
```

2. アクティブ・データベースとスタンバイ・データベースへのコネクションをレジュームした後、次を実行して、ウォーム・スタンバイのステータスを確認します。

```
admin logical_status [,logical_ds,logical_db]
go
```

アクティブ・データベースとスタンバイ・データベースへの Replication Agents のレジューム

アクティブ・データベースとスタンバイ・データベースの Replication Agents をレジュームして、複写するためにトランザクションのデータベース・ログのスキャンを開始します。
各 Replication Agent では、次を実行します。

```
resume
go
```

アクティブ・データベースとスタンバイ・データベースの切り替え

アクティブ・データベースで障害が発生した場合またはアクティブ・データベースでメンテナンスを実行する場合、スタンバイ・データベースに切り替えます。

1. Replication Server で、次のように入力します。

   ```
   switch active for logical_ds.logical_db
   to standby_ds.standby_db
   ```

   切り替え時の Replication Server の動作については、「内部での切り替え手順」を参照してください。
2. 切り替えの進行状況をモニタするには、次を実行します。

   ```
   admin logical_status, logical_ds, logical_db
   ```

   `Operation in Progress` 出力カラムと `State of Operation in Progress` 出力カラムに、切り替えステータスが表示されます。
3. アクティブ・データベースの切り替えが完了すると、次を実行して、Replication Server のアクティブ・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 active_ds.active_db
   ```
4. まだサスペンドされていない場合は、元のアクティブ・サイトで Replication Agent をサスペンドします。次を実行して、スタンバイ・モードに設定します。

   ```
   ra_config ra_standby,true
   ```
5. 元のスタンバイ・モードで Replication Agent は自動的にサスペンドされ、スタンバイ・モードから複写モードに変更されます。確認するには、次を実行します。

   ```
   ra_config ra_standby
   ```

   戻り値を false に設定する必要があります。

6. アクティブ・サイトとスタンバイ・サイトで両方の Replication Agent をレジュームします。

注意： Replication Server が切り替えの途中で停止した場合は、Replication Server を再起動すると、切り替えがレジュームします。スタンバイ・サイトで Replication Agent をレジュームすると、自動的に Replication Agent システム・データベース (RASD) は更新されます。

参照：

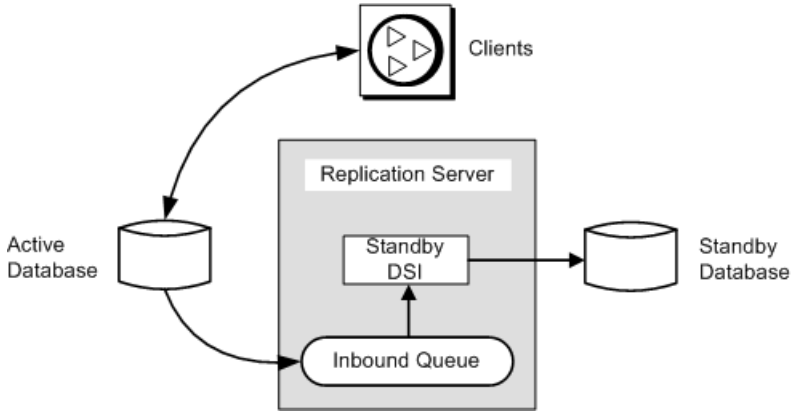
- 内部での切り替え手順 (224 ページ)

アクティブ・データベースとスタンバイ・データベースを切り替える前

ウォーム・スタンバイ・アプリケーション自体のアクティビティを介する場合以外は、複写システムに関与しないデータベースに対するウォーム・スタンバイ・アプリケーションを示します。

アクティブ・データベースとスタンバイ・データベースを切り替える前の、ウォーム・スタンバイ・アプリケーションの通常のオペレーションを示しています。

図 15 : ウォーム・スタンバイ・アプリケーション ― 切り替え前



図：ウォーム・スタンバイ・アプリケーション ― 切り替え前には、次を示す内部詳細が含まれます。

- Replication Server は、アクティブ・データベースから受信したトランザクションを、インバウンド・メッセージ・キューに書き込みます。
- このインバウンド・キューはスタンバイ・データベースの DSI スレッドによって読み取られ、このスレッドによってスタンバイ・データベースでトランザクションが実行されます。

アクティブ・データベースから受信したメッセージは、スタンバイ DSI スレッドがそのメッセージを読み取ってスタンバイ・データベースに適用するまで、インバウンド・キューからトランケートできません。

この例では、トランザクションは、アクティブ・データベースからスタンバイ・データベースに単純に複写されます。論理データベース自体では、次のことは行われません。

- レプリケート・データベースまたはリモート Replication Server に複写されるプライマリ・データの保持
- 別の Replication Server からの複写トランザクションを受信します。

内部での切り替え手順

内部での切り替え手順について説明します。

アクティブ・データベースとスタンバイ・データベースを切り替えると、Replication Server は次を実行します。

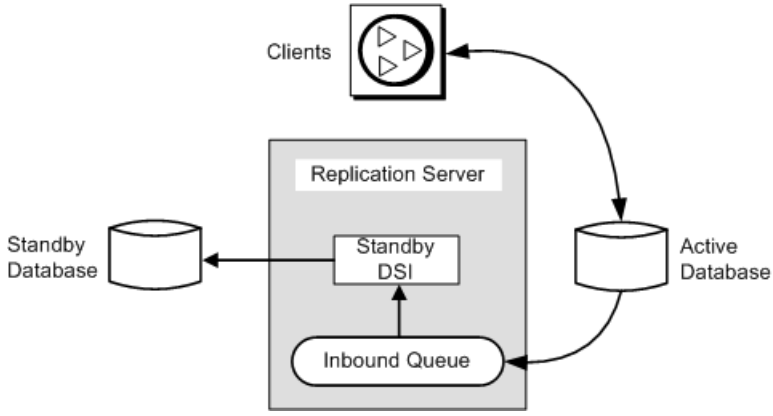
1. アクティブおよびスタンバイのコネクションに対し、**suspend log transfer** コマンドを発行します。
2. インバウンド・キューに残されているすべてのメッセージを読み取り、スタンバイ・データベースに適用します。サブスクリプション・データまたは複写ストアド・プロシージャについては、アウトバウンド・キューに適用します。切り替えが完了する前に、インバウンド・キュー内のコミットされたトランザクションがすべて処理されます。
3. スタンバイ DSI をサスペンドします。
4. 新しいアクティブ・データベースのトランザクション・ログにマーカを設定します。Replication Server はこのマーカを使用して、新しいスタンバイ・データベースや任意のレプリケート・データベースに適用するトランザクションを決定します。
5. 新しいアクティブ・データベースに対するコネクションを停止します。インバウンド・キューをパージし、`rs_oqid` テーブルへの最後の oqid をフラッシュし、`rs_locator` テーブルをリセットします。Replication Server セグメントのフラグをリセットし、新しいスタンバイ DSI をサスペンドします。
6. アクティブ・データベースとスタンバイ・データベースの両方の **ptype** パラメータを更新します。Replication Server は、古いアクティブ・データベースを対象とするサブスクリプションに有効のマークを、新しいアクティブ・データベースのサブスクリプションに無効のマークを付けます。
7. 新しいアクティブ・データベースのコネクションをレジュームし、そのデータベースに対するログ転送をレジュームして新しいメッセージを受信できるようにします。アクティブ・データベースとスタンバイ・データベースの両方の DSI をレジュームします。

アクティブ・データベースとスタンバイ・データベースを切り替えたあと

アクティブ・データベースからスタンバイ・データベースに切り替えたあとの、関連するプロセス、およびウォーム・スタンバイ環境でのコンポーネントのステータスについて説明します。

アクティブ・データベースとスタンバイ・データベースのロールを切り替えると、複写システムは次の図に示すように変更されます。

図 16 : ウォーム・スタンバイ・アプリケーションの例 ― 切り替え後



- 以前のスタンバイ・データベースが新しいアクティブ・データベースになります。クライアント・アプリケーションは、新しいアクティブ・データベースに切り替えられています。
- この例では、以前のアクティブ・データベースが新しいスタンバイ・データベースになります。以前のアクティブ・データベースに対するメッセージは、新しいアクティブ・データベースに適用するためにキューイングされます。

注意： 切り替え後、前のアクティブ・データベースの Replication Agent が停止し、新しいアクティブ・データベースの Replication Agent が起動されます。

ウォーム・スタンバイ・アプリケーションのモニタリング

Replication Server ログ・ファイルまたは **admin** コマンドを使用して、ウォーム・スタンバイ・アプリケーションをモニタできます。

複写を使用するウォーム・スタンバイ・アプリケーション
複写に関与するウォーム・スタンバイ・アプリケーションの場合、論理データベースは、複写システムのプライマリ・データベースまたはレプリケート・データベースとして機能します。

複写を使用するウォーム・スタンバイ・アプリケー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。

複写定義およびサブスクリプション

Replication Agent は、初期化中に自動的に複写定義を作成 (**pdb_xlog init** コマンドの実行) し、Oracle ウォーム・スタンバイのために **pdb_auto_create_repdefs** 設定パラメータを true に設定します。

Replication Agent が論理コネクションを (RSSD に問い合わせることで) 検出する場合、Replication Agent が作成した複写定義は Oracle ウォーム・スタンバイ環境をサポートするようにカスタマイズされます。

ウォーム・スタンバイ環境を非ウォーム・スタンバイ・データベースに複写する特定のシナリオでは、複写予定の各テーブルまたはストアド・プロシージャに対して、2 番目の複写定義を作成する必要があります。

追加のウォーム・スタンバイ・データベースの複写定義

ウォーム・スタンバイ・プライマリ環境からウォーム・スタンバイ環境外のレプリケート・データベースに複写する場合、複写される各テーブルに対して新しい複写定義を作成することができます。

Replication Agent が初期化時に自動的に作成した複写定義には、次のような属性があります。

- Oracle データ型の Replication Server ユーザ定義データ型 (UDD) へのマッピングが提供されます。
- デフォルトで、`clob/blob` カラムを持つテーブルは **always_replicate** 句で定義されます。**auto_create_repdefs** が “on” に設定されている場合、`clob/blob` カラムは **replicate_if_change** 句で定義されます。

注意： **always_replicate** および **replicate_if_change** は、複写定義を作成するための句です。

- **send standby replication definition columns** 句で、複写定義は作成されます。

次のケースの場合、追加の複写定義を作成できます。

- Oracle データ型から Replication Server UDD へのマッピングを提供する。
- clob/blob カラムを持つテーブルのために、**replicate_if_change** を使用する。
- データベース・レベルのサブスクリプションが非ウォーム・スタンバイ・データベースへのサブスクリプション作成のために使用される場合、**send standby all columns** 句を含める。
- プライマリとレプリケート・ファンクションの所有者を指定する。ユーザ・プロシージャのターゲット (スタンバイ・データベース) ファンクションの所有者情報を指定するために、ファンクション文字列をカスタマイズする。

たとえば、ユーザ・テーブル TB1 が COL5 カラムの 1 つで、Oracle のデータ型 date として定義される場合、正常にカラムをスタンバイ・データベースに複写するには、ユーザは次のように複写定義を作成する必要があります。

```
create replication definition rep1
with primary at ordb.pdb
with all tables named 'USER1'.'TB1'
(
"COL1" int,
"COL2" int,
"COL3" int,
"COL4" char(255),
"COL5" rs_oracle_datetime,
)
primary key( "COL1","COL2","COL3")
searchable columns( "COL1","COL2","COL3","COL5")
send standby replication definition columns
replicate minimal columns
go
```

この例では、**create replication definition** コマンドの **send standby replication definition columns** 句は、この複写定義がサブスクリプションを作成するデータベース、およびスタンバイ・データベースのために使用できることを指定します。

参照：

- データ型の変換とマッピング (247 ページ)

ウォーム・スタンバイ・アプリケーションでのサブスクリプション

create subscription コマンドと **define subscription** コマンドは、物理名ではなく、論理データベース名と論理データ・サーバ名を使用します。

アクティブ・データベースからスタンバイ・データベースへの複写にサブスクリプションは使用されませんが、次のことは可能です。

- 論理プライマリ・データベースのデータに対してサブスクリプションを作成する。
- 他のデータベースのデータを論理レプリケート・データベースに複写するためにサブスクリプションを作成する。

複写を使用するウォーム・スタンバイ・アプリケー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。

アップグレードの考慮事項

Oracle ウォーム・スタンバイ機能にとって、アップグレード後に実行する必要があるという、特別な指示は不要です。

Replication Server のバージョンをアップグレードする情報については、ご使用のプラットフォームの『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。

ダウングレードの考慮事項

ダウングレードした後、ファンクション文字列のマッピングやデータ変換が実行できないように、Oracle スタンバイ・コネクションは中断されます。RSSD をダウングレードする前に、スタンバイ・コネクションを削除しなかった場合は、ダウングレード後に削除することができます。

rs_init による RSSD のダウングレードが完了した後、Oracle データ・サーバーへのコネクション (Oracle へのコネクションが **create connection** と **using profile** 句を使用して作成される場合) はダウンする可能性があります。理由は、Replication Server15.5 で提供されている **wait_after_commit** 設定パラメータが使用できなくなったため、複写処理をレジュームする必要があるからです。

ダウングレード後の複写のレジューム

RSSD をダウングレードした後の複写のレジュームについて説明します。

複写をレジュームするには、次を実行します。

1. 次のコマンドを実行します。

```
alter connection to data_server.database
set dsi_serialization_method to 'wait_for_commit'
go
```

2. アクティブ・データベースへのログ転送をレジュームします。
3. Replication Agent for Oracle をレジュームします。

上記の手順を完了したら、アクティブ・データベースから、ダウングレードの前に作成された複写定義とサブスクリプションを持つ他のレプリケート・データベースへの複写を開始することができます。

Oracle レプリケート・データベースの再同期

Replication Server を使用すると、レプリケート・データベースを再同期してマテリアライズできます。また、プライマリ・データベースのクワイスを強いることなく、データの損失や整合性を失うリスクなしで複写をレジュームできます。

データベース再同期化は、信頼されたソースから取得したデータ・ダンプを同期先のデータベースに適用することをベースとしています。

データベースの再同期には、この機能をサポートしているデータベース用の Replication Agent のバージョンが必要です。Replication Agent のための特定のコマンドについては、Replication Agent のマニュアルを参照してください。

製品の互換性

Oracle データベースの再同期をサポートする Oracle、Replication Agent for Oracle、ECDA Option for Oracle、ExpressConnect for Oracle のバージョンを使用します。Replication Server Options 15.5 では ExpressConnect for Oracle が ECDA Option for Oracle の代わりに使用されます。

Replication Server Options のマニュアルを参照してください。

表 3 : Oracle データベースの再同期の製品互換性

データベース・サーバのバージョン	Replication Agent のバージョン	ExpressConnect および ECDA for Oracle バージョン
Oracle Server 10g、11g	15.5	ECDA 15.0 ESD #3、ExpressConnect 15.5 for Oracle

データベースの再同期の設定

Oracle データベースの再同期を設定します。

1. Replication Agent をサスペンドして複写プロセスを停止します。
2. Replication Server を再同期モードにします。再同期モードになると、Replication Server はトランザクションをスキップします。さらに、プライマリ・データベースまたは信頼されたソースから取得したダンプを使ってレプリケート・データベースにデータを再移植する準備として、複写キューから複写データをパージします。

3. プライマリ・データベースからダンプを取得します。
4. Replication Agent を再開して、データベース再同期マーカを Replication Server に送り、再同期処理が進行中であることを示します。

   Replication Server がプライマリ・データベース・ダンプが完了したことを示すダンプ・マーカを検出すると、Replication Server はトランザクションのスキップを停止し、どのトランザクションをレプリケート・データベースに適用するかを判定できるようになります。
5. ダンプをレプリケート・データベースに適用します。
6. レプリケート・データベースを再度、初期化します。
7. 複写をレジュームします。

データベースの再同期では、Replication Server と Replication Agent for Oracle の両方からのコマンドとパラメータを使用する必要があります。

参照：
- データベース再同期化シナリオ (238 ページ)

Replication Server にトランザクションをスキップさせる

resume connection コマンドに **skip to resync** パラメータを指定して、Replication Agent から送られたダンプ・データベース・マーカを受け取り認識するまで、指定されたレプリケート・データベースで DSI アウトバウンド・キュー内のトランザクションをスキップするよう Replication Server に指示します。

レプリケート・データベース内のデータはダンプの内容によって置き換えられることになっているので、Replication Server はアウトバウンド・キュー内のレコードの処理をスキップします。

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

次に示すコマンドを実行します。

```
resume connection to data_server.database
   [skip [n] transaction | execute transaction | skip to resync
marker]
```

警告！ resume connection を **skip to resync marker** オプションを付けて間違ったコネクションで実行すると、レプリケート・データベースのデータが非同期となります。

skip to resync marker を設定すると、Replication Server は Replication Server ログ内またはデータベース例外ログ内でスキップされたトランザクションをログに記録しません。**skip [n] transaction** を設定すると、Replication Server はスキップされたトランザクションをログに記録します。

データベース再同期マーカを Replication Server に送信する

Replication Agent for Oracle を設定または Replication Agent for Oracle に指示して、再同期処理が進行中であることを示すデータベース再同期マーカを Replication Server に送信することができます。

再同期モードで Replication Agent を再開すると、Replication Agent は再同期データベース・マーカを最初のメッセージとして Replication Server へ送信してから、SQL データ定義言語 (DDL: data definition language) またはデータ操作言語 (DML: data manipulation language) のトランザクションを送信します。同じプライマリ・データベースの複数のレプリケート・データベースはそれぞれに DSI アウトバウンド・キューがあるので、すべて同じ再同期マーカを受け取ります。

skip to resync marker パラメータでレジュームする各 DSI に対して、DSI が再同期マーカを受け取ったことが、DSI アウトバウンド・キューによって Replication Server システム・ログに記録されます。また、その時点からダンプ・データベース・マーカを受け取るまで、DSI がコミットされたトランザクションを拒否することも記録されます。

Replication Agent for Oracle では、データベース再同期マーカの送信で各オプションをサポートするために、**resync** または **resync, init** のパラメータを含む **resume** コマンドを使用します。『Replication Agent リファレンス・マニュアル』の「Command Reference」の「**resume**」を参照してください。

再同期マーカの送信

Replication Agent は自動的にトランケーション・ポイントが変更になっているかどうかを判断することができます。

次のような場合、**resume resync** を使用して、再同期マーカを送信することができます。

- トランケーション・ポイントに変更がなく、Replication Agent が最後に処理したところからトランザクション・ログの処理を続けることになっている場合。各アウトバウンド DSI スレッドとキューはデータベース再同期マーカを受け取り処理します。再同期マーカを受け取ったとき、DSI は skip to resync マーカ要求に従って Replication Server システム・ログにレポートを送ります。その後、ダンプ・データベース・マーカを待つ間、DSI はコミットされたトランザクションを拒否します。この時点で、このメッセージと、ダンプ・データベース・マーカを待つ動作の変更によって、レプリケート・データベースにダンプを適用できるようになります。
- プライマリ・データベースのトランケーション・ポイントが移動されている場合。トランケーション・ポイントを手動で変更すると、移動は起きます。

この状況下では、resync marker を送信する前に、Replication Agent レポジトリを初期化する Replication Agent で **ra_init force** を実行します。この初期化では、トランケーション・ポイントを移動し、一部のトランザクション・ログ・レコードの処理をスキップするので、Replication Agent は消失の可能性があるデータベース内の変更すべてを追跡します。
トランケーション・ポイントが変更されると、Replication Server のインバウンド・キュー内にあるオープン・トランザクションは、新しいトランケーション・ポイントから送られたアクティビティと一致しないため、パージされる必要があります。変更されたトランケーション・ポイントが以前のオリジン・キュー ID (OQID) を持つレコードを送信する可能性があるので、Replication Server は重複検出をリセットします。以前のデータがキューからパージされると、Replication Server は Replication Agent からのどの新しいアクティビティも重複アクティビティとして扱いません。したがって、新しいアクティビティが拒否されることはありません。マーカを受け取るまで、Replication Server はアウトバウンド・キュー・コマンドを拒否し続けるので、パージ・オプションは DSI の処理を変更しません。

init コマンドによる再同期マーカの送信

再同期マーカを **init** コマンド付きで送信するには、**resume resync, init** を使用します。これによって、インバウンド・キュー内のすべてのオープン・トランザクションをパージして重複の検出をリセットし、アウトバウンド DSI をサスペンドするよう、Replication Server に指示できます。

このオプションはプライマリ・データベースにレプリケート・データベースと同じダンプを再ロードするときに使用します。プライマリ・データベースから取得したダンプはないので、Replication Agent はダンプ・データベース・マーカを送りません。再同期マーカの後に来るダンプ・データベース・マーカを待つ代わりに、**init** オプションは Replication Server が再同期マーカを処理したらすぐに DSI コネクションをサスペンドします。

DSI がサスペンドされたら、それ以降 DSI を通るすべてのアクティビティは新しいトランザクションのみになります。プライマリで使用したダンプをレプリケート・データベースに再ロードしたら、DSI をレジュームできます。

参照：

- プライマリ・データベースとレプリケート・データベースを同じダンプから再同期 (242 ページ)

データベースのダンプを取得する

dump ユーティリティを使用して、データベースのダンプを取得します。

ダンプが完了した後、管理者として、ダンプが行われたときにプライマリ・データベースから取得された情報を基に、希望するダンプ・ポイントを判断する必要

があります。**dump** ユーティリティはダンプ・ポイントを提供する可能性があります。あとの項のシナリオでは、Oracle RMAN ユーティリティを使用しています。

Oracle では、**backup database plus archivelog** を使用して、プライマリ・データベースをダンプし、**restore database** と **recover database** を使用して、レプリケート・データベースへのダンプを適用します。1 つの RMAN バックアップ・セットからダンプ・ポイントを取得するには、**list backup** Oracle コマンドを使用します。この例は、**list backup** を実行して生成された出力を示しています。

```
RMAN>list backup;
List of Backup Sets
===================


BS Key  Size       Device Type Elapsed Time Completion Time
------- ---------- ----------- ------------ ---------------
8       125.58M    DISK        00:00:04     16-MAY-11
        BP Key: 8   Status: AVAILABLE  Compressed: NO  Tag:
TAG20110516T125049
        Piece Name: /ralinuxsh5/oracle/product/11.1.0/db_2/dbs/
0bmcflp9_1_1

  List of Archived Logs in backup set 8
  Thrd Seq     Low SCN    Low Time  Next SCN   Next Time
  ---- ------- ---------- --------- ---------- ---------
  1    1       1018110    14-MAY-11 1058201    15-MAY-11
  1    2       1058201    15-MAY-11 1103370    15-MAY-11
  1    3       1103370    15-MAY-11 1142662    16-MAY-11
  1    4       1142662    16-MAY-11 1148674    16-MAY-11
  1    5       1148674    16-MAY-11 1150375    16-MAY-11
  1    6       1150375    16-MAY-11 1150477    16-MAY-11

BS Key  Type LV Size       Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ ---------------
9       Full    1.08G      DISK        00:00:15     16-MAY-11
        BP Key: 9   Status: AVAILABLE  Compressed: NO  Tag:
TAG20110516T125054
        Piece Name: /ralinuxsh5/oracle/product/11.1.0/db_2/dbs/
0cmcflpe_1_1
  List of Datafiles in backup set 9
  File LV Type Ckp SCN    Ckp Time  Name
  ---- -- ---- ---------- --------- ----
  1      Full 1150485    16-MAY-11 /work2/oracle11.1/oradata/or11sh1/
system01.dbf
  2      Full 1150485    16-MAY-11 /work2/oracle11.1/oradata/or11sh1/
sysaux01.dbf
  3      Full 1150485    16-MAY-11 /work2/oracle11.1/oradata/or11sh1/
undotbs01.dbf
  4      Full 1150485    16-MAY-11 /work2/oracle11.1/oradata/or11sh1/
users01.dbf
  5      Full 1150485    16-MAY-11 /work2/oracle11.1/oradata/or11sh1/
example01.dbf
```

```
BS Key  Type LV Size       Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ ---------------
10      Full    9.36M      DISK        00:00:04     16-MAY-11
        BP Key: 10   Status: AVAILABLE  Compressed: NO  Tag:
TAG20110516T125054
        Piece Name: /ralinuxsh5/oracle/product/11.1.0/db_2/dbs/
0dmcflq1_1_1
  SPFILE Included: Modification time: 14-MAY-11
  SPFILE db_unique_name: OR11SH1
  Control File Included: Ckp SCN: 1150507      Ckp time: 16-MAY-11

BS Key  Size       Device Type Elapsed Time Completion Time
------- ---------- ----------- ------------ ---------------
11      18.50K     DISK        00:00:04     16-MAY-11
        BP Key: 11   Status: AVAILABLE  Compressed: NO  Tag:
TAG20110516T125118
        Piece Name: /ralinuxsh5/oracle/product/11.1.0/db_2/dbs/
0emcflq6_1_1

  List of Archived Logs in backup set 11
  Thrd Seq     Low SCN    Low Time  Next SCN   Next Time
  ---- ------- ---------- --------- ---------- ---------
  1    7       1150477    16-MAY-11 1150513    16-MAY-11
  1    8       1150513    16-MAY-11 1150568    16-MAY-11
```

必要なダンプ・ポイントは、最後にアーカイブされたログのバックアップ・セットの「次の SCN」カラムの最大値から 1 つ減算されます。この例では、最後のセットがセット 11 で、そのセットでの「次の SCN」の最大値が 1150568 です。したがって、ダンプ・ポイントは 1150567 となります。

RMAN ユーティリティと **list backup** コマンドに関する情報については、Oracle のマニュアルを参照してください。

ダンプ・データベース・マーカを Replication Server に送信する

データ・サーバ・インタフェース・スレッド (DSI) は、**skip to resync marker** を使用してレジュームされた後、再同期モードになっていて、再同期モードで Replication Agent を再起動すると、再同期データベース・マーカの後に受け取ったダンプ・データベース・マーカは DSI をサスペンドし、DSI コネクションの既存の再同期状態を削除します。

同じプライマリ・データベースの複数のレプリケート・データベースは、すべて同じダンプ・データベース・マーカを受け取ります。In Replication Agent では、**oracle scn** パラメータを設定した **lr_dump_marker** コマンドを使用して、ダンプ・データベース・マーカは Replication Agent 設定に基づいて送信されます。Replication Agent のマニュアルを参照してください。

注意： **init** 付き resync を使用して Replication Agent を再起動すると、DSI は、再起動データベース・マーカを受け取った後、すぐにサスペンドされます。DSI は、ダンプ・マーカを待たずに、サスペンドされます。

レプリケート・データベースにダンプを適用したら、手動で DSI をレジュームできます。DSI はコミットされたトランザクションを拒否しなくなります。また、ダンプ・データベース・マーカが示すダンプ・ポイントの後でコミットされたトランザクションはすべて複写されます。

DSI スレッド情報をモニタする

データベースの再同期中に DSI についての情報を提供するには、**admin who** コマンドを使用します。

状態	説明
SkipUntil Resync	**skip to resync** を実行した後 DSI がレジュームする。このステータスは DSI がデータベース再同期マーカを受け取るまで続く。
SkipUntil Dump	DSI がデータベース再同期マーカを受け取った。このステータスは DSI がその後のダンプ・データベース・マーカを受け取るまで続く。

再同期するデータベースにダンプを適用する

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できるのは、メッセージがシステム・ログに表示された後だけです。

メッセージは次のとおりです。

- Replication Server が resync database マーカを受け取ると、次のようなメッセージが出力されます。

  ```
  DSI for data_server.database received and processed
  Resync Database Marker. Waiting for Dump Marker.
  ```

- Replication Server が **init** マーカ付きの再同期データベースを受け取るときのメッセージ。

  ```
  DSI for data_server.database received and processed
  Resync Database Marker. DSI is now suspended. Resume after
  database has been reloaded.
  ```

- Replication Server がダンプ・データベース・マーカを受け取るときのメッセージ。

  ```
  DSI for data_server.database received and processed
  Dump Marker. DSI is now suspended. Resume after database has been
  reloaded.
  ```

再同期するデータベースにダンプをロードする方法の詳細については、Oracle のマニュアルを参照してください。

レプリケート・データベースの再初期化

プライマリ・データベースまたはダンプ・ソースからレプリケート・データベースにダンプを適用したら、レプリケート・データベースを再初期化して、ダンプが削除したユーザ、テーブル、およびパーミッションをリストアします。

1. メンテナンス・ユーザおよび DDL ユーザがプライマリ・データベースに存在しない場合、プライマリ・データベースからダンプを適用した後、レプリケート・データベースにユーザを追加します。
2. `hds_oracle_new_setup_for_replicate.sql` スクリプトをレプリケート・データベースで実行して、関連する Replication Server システム・テーブルをレプリケート・データベースに追加します。

   スクリプトは、レプリケート・データベースに関連する値を挿入し、必要なパーミッションを付与します。

データベース再同期化シナリオ

データベースの再同期手順はそのシナリオによって異なります。再同期手順を完了すると、プライマリ・データベースとレプリケート・データベースはトランザクションの一貫性が保たれた状態になります。

手順の実行には次の要件があります。

- 複写システム管理者であること。
- 正常に稼動する複写環境が存在すること。
- 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コピーするためのメソッドやプロセスがあること。

コマンドの構文については、次を参照してください。

- Replication Agent for Oracle – 『Replication Agent リファレンス・マニュアル』を参照してください。
- Replication Server – 『Replication Server リファレンス・マニュアル』を参照してください。

1 つ以上のレプリケート・データベースをプライマリ・データベースから直接再同期する

1 つ以上のレプリケート・データベースを 1 つのプライマリ・データベースから再同期します。

この手順では、多少の違いはありますが、次のことを実行できます。

- プライマリ・データベースとレプリケート・データベース間の複写の遅延時間が、複写によるデータベースの回復が不可能で、複写データに基づくレポート

の作成が実用的でなくなった場合に、レプリケート・データベースにデータを再移植する。
- プライマリ・データベースから信頼されたデータをレプリケート・データベースに再移植する。
- プライマリ・データベースが複数のレプリケート・データベースのソースになっている場合に、再同期を調整する。
- プライマリ・サイトが一対のウォーム・スタンバイ・データベースで構成されている論理データベースであり、それに 1 つまたは複数のレプリケート・データベースを再同期する場合に、再同期を調整する。ウォーム・スタンバイのペアでは、アクティブ・データベースがプライマリ・データベースとして、スタンバイがレプリケート・データベースとして機能する。したがって、1 つまたは複数のレプリケート・データベースからはアクティブ・データベース (プライマリ・サイトのウォーム・スタンバイ・ペアの 1 つ) がプライマリ・データベースに見える。

プライマリ・データベースから直接再同期する

レプリケート・データベースをプライマリ・データベースから直接再同期します。

1. Replication Agent による複写プロセスを停止します。トランケーション・ポイントを変更しないでください。Replication Agent で次のコマンドを実行します。

   ```
   suspend
   ```

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 Replication Agent 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. トランケーション・ポイントが移動されていない場合は、手順 5 に進んでください。それ以外の場合は、Replication Agent レポジトリを再初期化してからプライマリ・データベースのコンテンツのダンプを取得します。Replication Agent で次のコマンドを実行します。

   ```
   ra_init force
   go
   ```

5. データベースのマニュアルの指示に従ってプライマリ・データベースのコンテンツのダンプを取得します。Recovery Manager (RMAN) for Oracle を使用する場合は、Oracle の **list backup** コマンドを使用して RMAN バックアップの最後の System Change Number (SCN) を取得します。Replication Agent で、次のように **lr_dump_marker** の値としてこの SCN を設定します。

```
lr_dump_marker oracle scn
```

6. 再同期モードで Replication Agent を起動し、再同期マーカを Replication Server に送信します。

   ```
   resume resync
   go
   ```

7. Replication Server システム・ログで、次のメッセージを検索して、DSI が Replication Agent 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.
   ```

   DSI がレプリケート・データベースの再同期マーカを処理すると、レプリケート・データベースにダンプを適用できます。

   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。

8. データベースのマニュアルの指示に従ってプライマリ・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。
9. 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。
10. メンテナンスおよび DDL ユーザがプライマリ・データベースに存在しない場合は、プライマリ・データベースからダンプを適用した後、レプリケート・データベースにこれらのユーザーを追加します。
11. `hds_oracle_new_setup_for_replicate.sql` スクリプトをレプリケート・データベースで実行して、`rs_info` と `rs_lastcommit` のテーブルをレプリケート・データベースに追加します。

    スクリプトは、レプリケート・データベースに関連する値を挿入し、必要なパーミッションを付与します。
12. 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
    ```

サードパーティ・ダンプ・ユーティリティを使用して再同期する

ディスク・スナップショットなどのサードパーティの **dump** ユーティリティを使って、プライマリ・データベースをダンプした後、再同期を調整します。

サードパーティ・ツールでは、プライマリ・データベースとのやり取りをネイティブのデータベース・ダンプ・ユーティリティほど密接には行うことができません。Replication Agent がダンプ・データベース・マーカの生成に使用できるような記録をサードパーティ・ツールがプライマリ・データベースのトランザクション・ログに書き込まない場合は、独自のダンプ・データベース・マーカを生成して再同期処理を完了できるようにします。詳細については、サードパーティ・ツールのマニュアルを参照してください。

1. Replication Agent による複写プロセスを停止します。トランケーション・ポイントを変更しないでください。Replication Agent で次のコマンドを実行します。

   ```
   suspend
   ```

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 Replication Agent 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. トランケーション・ポイントが移動されていない場合は、手順 5 に進んでください。それ以外の場合は、Replication Agent レポジトリを再初期化してからプライマリ・データベースのコンテンツのダンプを取得します。Replication Agent で次のコマンドを実行します。

   ```
   ra_init force
   go
   ```

5. サード・パーティ・ユーティリティを使用してプライマリ・データベースのコンテンツのダンプを取得します。
6. サード・パーティ・ユーティリティからダンプまたは情報を取得したら、プライマリ・データベースからの情報に基づいてダンプ・ポイントを決定します。サード・パーティ・ユーティリティを使用する場合、ユーザはダンプ・ポイントを決定する責任があります。たとえば、ディスク複写ツールを使用する場合、プライマリ・データベースでアクティビティを一時的に停止してディスク・スナップショットから実行中のトランザクションを消去し、ダンプ・データベース・マーカとして「トランザクション・ログの末尾」ポイントを使用できます。
7. 手順 5 で取得したダンプの位置の末尾にマークを付けるには、Replication Agent のプライマリ・データベースでストアド・プロシージャを実行します。

```
lr_dump_marker oracle scn
```

8. 再同期モードで Replication Agent を再起動し、再同期マーカを Replication Server に送信します。

   ```
   resume resync
   go
   ```

   Replication Agent は、手順 6 で取得して手順 7 で設定したダンプ位置の末尾に基づいてダンプ・データベース・マーカを一度に自動的に生成し、ダンプ・データベース・マーカを Replication Server に送信します。

9. Replication Server システム・ログで次のメッセージを検索して、DSI が Replication Agent 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.
   ```

10. データベースとサード・パーティ・ユーティリティのマニュアルの指示に従って、サード・パーティ・ツールからプライマリ・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。

11. 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。

12. メンテナンスおよび DDL ユーザがプライマリ・データベースに存在しない場合は、プライマリ・データベースからダンプを適用した後、レプリケート・データベースにこれらのユーザーを追加します。

13. `hds_oracle_new_setup_for_replicate.sql` スクリプトをレプリケート・データベースで実行して、`rs_info` と `rs_lastcommit` のテーブルをレプリケート・データベースに追加します。

    スクリプトは、レプリケート・データベースに関連する値を挿入し、必要なパーミッションを付与します。

14. 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
    ```

プライマリ・データベースとレプリケート・データベースを同じダンプから再同期

再同期を調整して、プライマリ・データベースとレプリケート・データベースを同じダンプまたはデータのコピーから再ロードします。プライマリ・データベー

スからダンプを取得しないので、ダンプ・データベース・マーカは必要ありません。

1. Replication Agent による複写プロセスを停止します。トランケーション・ポイントを変更しないでください。Replication Agent で次のコマンドを実行します。

   ```
   suspend
   ```

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 data_server.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 Replication Agent 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. プライマリ・データベースに外部ソースからのデータ・ダンプを適用します。
5. プライマリ・データベースのトランザクション・ログの最後までトランケーション・ポイントを移動します。Replication Agent で次のコマンドを実行します。

   ```
   pdb_xlog move_truncpt
   go
   ```

6. プライマリ・データベースからの最新のシステム・データに基づいて Replication Agent レポジトリを再初期化します。

   ```
   ra_init force
   go
   ```

7. Replication Agent に with the **init** オプションによる再同期モードで起動するように指示します。Replication Agent で次のコマンドを実行します。

   ```
   resume resync, init
   ```

8. Replication Server システム・ログで次のメッセージを検索して、DSI が Replication Agent 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. DSI is now suspended. Resume
   after database has been reloaded.
   ```

   Replication Server が **init** マーカを使用して再同期データベースを受信して処理すると、DSI コネクションはサスペンドします。

9. レプリケート・データベースに外部ソースからのデータ・ダンプを適用します。

10. メンテナンスおよび DDL ユーザがプライマリ・データベースに存在しない場合は、プライマリ・データベースからダンプを適用した後、レプリケート・データベースにこれらのユーザーを追加します。
11. `hds_oracle_new_setup_for_replicate.sql` スクリプトをレプリケート・データベースで実行して、`rs_info` と `rs_lastcommit` のテーブルをレプリケート・データベースに追加します。

    スクリプトは、レプリケート・データベースに関連する値を挿入し、必要なパーミッションを付与します。
12. レプリケート・データベースにダンプを適用したら、レプリケート・データベースで DSI をレジュームして Replication Server がプライマリ・データベースからのトランザクションを適用できるようにします。

    ```
    resume connection to data_server.database
    ```

ウォーム・スタンバイ・アプリケーションのアクティブ・データベースとスタンバイ・データベースの再同期

ウォーム・スタンバイ・ペアが単一プライマリ・データベースのレプリケート・サイトになっているときに、ウォーム・スタンバイ環境でアクティブ・データベースとスタンバイ・データベースを再同期します。

このシナリオでは、アクティブ・データベース、スタンバイ・データベース、プライマリ・データベースは、Oracle データベースです。

1. プライマリ・データベース Replication Agent とウォーム・スタンバイ・アクティブ・データベース Replication Agent による複写プロセスを停止します。トランケーション・ポイントを変更しないでください。Replication Agent で次のコマンドを実行します。

   ```
   suspend
   ```
2. アクティブ・データベースとスタンバイ・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. アクティブ・データベースとスタンバイ・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 Replication Agent 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. トランケーション・ポイントが移動されていない場合は、手順 5 に進んでください。それ以外の場合は、Replication Agent レポジトリを再初期化してからプライマリ・データベースのコンテンツのダンプを取得します。プライマリ Replication Agent で次のコマンドを実行します。

```
ra_init force
go
```

5. データベースのマニュアルの指示に従ってプライマリ・データベースのコンテンツのダンプを取得します。Recovery Manager (RMAN) for Oracle を使用する場合は、Oracle の **list backup** コマンドを使用して RMAN バックアップの最後の System Change Number (SCN) を取得します。Replication Agent で、次のように **lr_dump_marker** の値としてこの SCN を設定します。

```
lr_dump_marker oracle scn
```

6. 再同期モードでプライマリ Replication Agent を起動し、再同期マーカを Replication Server に送信します。

```
resume resync
go
```

7. Replication Server システム・ログで次のメッセージを検索して、アクティブ・データベースの DSI がプライマリ・データベース Replication Agent 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.
```

8. Replication Server システム・ログでアクティブ・データベースから次のメッセージを検索して、アクティブ・データベースの Replication Server DSI 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.
```

9. データベースのマニュアルの指示に従ってプライマリ・データベースのダンプをアクティブ・データベースに適用します。
10. アクティブ・データベースのトランザクション・ログの最後までトランケーション・ポイントを移動します。Replication Agent で次のコマンドを実行します。

```
pdb_xlog move_truncpt
go
```

11. アクティブ・データベースからの最新のシステム・データに基づいて Replication Agent レポジトリを再初期化します。

```
ra_init force
go
```

12. **init** オプションを使用してアクティブ・データベースの Replication Agent を再同期モードで起動します。Replication Agent で次のコマンドを実行します。

```
resume resync, init
```

13. Replication Server システム・ログで次のメッセージを検索して、スタンバイ・データベースの DSI がアクティブ・データベース Replication Agent 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. DSI is now suspended. Resume
after database has been reloaded.
```

Replication Server が init マーカを使用して再同期データベースを受信して処理すると、DSI コネクションはサスペンドします。

14. アクティブ・データベースのコンテンツのダンプを取得し、スタンバイ・データベースにダンプを適用します。ダンプにデータベース設定情報が含まれない場合は、手順 5 からプライマリ・データベースのダンプを適用することもできます。
15. 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 をレジュームします。

```
resume connection to data_server.database
```

データ型の変換とマッピング

Replication Server には、サポートされる ASE 以外の各データ・サーバについて、あるデータ型から別のデータ型へのデフォルトのマッピングを定義するクラス・レベル変換が用意されています。

次のデータ型の変換が用意されています。

- Adaptive Server データ型に直接対応しない、ASE 以外のデータ型
- ASE 以外のデータ型に直接対応しない、Adaptive Server データ型
- サポートされる ASE 以外の別のデータ・サーバのデータ型に直接対応しない、ASE 以外のデータ型

注意： 別のデータ・サーバのデータ型に直接対応するデータ型には、クラス・レベル変換は用意されていません。

DB2 データ型

データ型変換に関するこの情報は、メインフレーム環境 (IBM z/OS など)、あるいは UNIX および Microsoft Windows 環境の DB2 UDB に適用されます。

Adaptive Server データ型から DB2 データ型への変換

Adaptive Server データ型から DB2 データ型へのクラス・レベル変換を示します。

Adaptive Server データ型	DB2 データ型
bigdatetime	TIMESTAMP
bigint	BIGINT
bigtime	TIMESTAMP
binary	CHAR FOR BIT DATA
bit	TINYINT
date	DATE (UNIX と Windows のみ)
datetime	TIMESTAMP
decimal	DECIMAL
int	NUMERIC

Adaptive Server データ型	DB2 データ型
money	NUMERIC
numeric	NUMERIC
real	REAL (UNIX と Windows のみ)
smalldatetime	TIMESTAMP
smallint	NUMERIC
smallmoney	NUMERIC
time	TIME (UNIX と Windows のみ)
tinyint	NUMERIC
unsigned bigint	DECIMAL (20,0)
unsigned int	BIGINT
unsigned smallint	INTEGER
unsigned tinyint	SMALLINT
unitext	DBCLOB
varbinary	VARCHAR FOR BIT DATA

DB2 データ型から Adaptive Server データ型への変換

DB2 データ型から Adaptive Server データ型へのクラス・レベル変換を示します。

DB2 データ型	Adaptive Server データ型
CHAR FOR BIT DATA	binary
DATE	datetime
DECFLOAT UDB (UNIX と Windows のみ)	float
DOUBLE (UNIX と Windows のみ)	float
REAL (UNIX と Windows のみ)	real
TIME	datetime
TIMESTAMP	datetime
VARCHAR FOR BIT DATA	varbinary

DB2 データ型から Microsoft SQL Server データ型への変換

DB2 データ型から Microsoft SQL Server データ型へのクラス・レベル変換を示します。

DB2 データ型	Microsoft SQL Server データ型
CHAR FOR BIT DATA	binary
DATE	datetime
DECFLOAT UDB (UNIX と Windows のみ)	float
DOUBLE (UNIX と Windows のみ)	float
REAL (UNIX と Windows のみ)	real
TIME	datetime
TIMESTAMP	datetime
VARCHAR FOR BIT DATA	varbinary

DB2 データ型から Oracle データ型への変換

DB2 データ型から Oracle データ型へのクラス・レベル変換を示します。

DB2 データ型	Oracle データ型
CHAR FOR BIT DATA	RAW
DATE	DATE
DECFLOAT UDB (UNIX と Windows のみ)	FLOAT
DOUBLE (UNIX と Windows のみ)	FLOAT
REAL (UNIX と Windows のみ)	REAL
TIME	DATE (時刻を含む)
TIMESTAMP	DATE (時刻を含む)
VARCHAR FOR BIT DATA	RAW

DB2 に対応する Replication Server のデータ型名

z/OS プラットフォーム上の DB2 データ・サーバの DB2 データ型を表す、Replication Server のユーザ定義データ型 (UDD) 名を示します。

表 4 : DB2 for z/OS データ型に対応する Replication Server UDD 名

DB2 for z/OS データ型	Replication Server の名前
CHAR FOR BIT DATA	*rs_db2_char_for_bit*
DATE	*rs_db2_date*
DECIMAL	*rs_db2_decimal*、*rs_db2_numeric*
TIME	*rs_db2_time*
TIMESTAMP	*rs_db2_timestamp*
VARCHAR FOR BIT DATA	*rs_db2_varchar_for_bit*

UNIX および Microsoft Windows プラットフォーム上の DB2 データ・サーバの DB2 データ型を表す、Replication Server UDD の名前を示します。

表 5 : DB2 for UNIX and Windows データ型に対応する Replication Server UDD 名

DB2 for UNIX/Windows データ型	Replication Server の名前
CHAR FOR BIT DATA	*rs_udb_char_for_bit*
DATE	*rs_udb_date*
DECFLOAT	*rs_udb_decfloat*
DOUBLE	*rs_udb_double*
INTEGER	*rs_udb_bigint*
REAL	*rs_udb_real*
TIME	*rs_udb_time*
TIMESTAMP	*rs_udb_timestamp*
VARCHAR FOR BIT DATA	*rs_udb_varchar_for_bit*

Microsoft SQL Server データ型

Microsoft SQL Server データ型のクラス・レベル変換 (デフォルトのデータ型マッピング) と、Microsoft SQL Server データ型に対応する Replication Server のデータ型名について説明します。

Adaptive Server データ型から Microsoft SQL Server データ型への変換

符号なしデータ型の、Adaptive Server データ型から Microsoft SQL Server データ型へのクラス・レベル変換を示します。

Microsoft SQL Server データ型は Adaptive Server データ型と直接互換性があり、変換を必要としないため、Adaptive Server データ型から Microsoft SQL Server データ型への (または Microsoft SQL Server データ型から Adaptive Server データ型への) その他のクラス・レベル変換は用意されていません。

表 6 : Adaptive Server データ型から Microsoft SQL Server データ型へのクラス・レベル変換

Adaptive Server データ型	Microsoft SQL Server データ型
unsigned bigint	DECIMAL (20,0)
unsigned int	BIGINT
unsigned smallint	INT
unsigned tinyint	SMALLINT
unitext	NTEXT

Microsoft SQL Server データ型から DB2 データ型への変換

Microsoft SQL Server データ型から DB2 データ型へのクラス・レベル変換を示します。

表 7 : Microsoft SQL Server データ型から DB2 データ型へのクラス・レベル変換

Microsoft SQL Server データ型	DB2 データ型
binary	CHAR FOR BIT DATA
bit	TINYINT
datetime	TIMESTAMP
decimal	DECIMAL

Microsoft SQL Server データ型	DB2 データ型
money	NUMERIC
numeric	NUMERIC
smalldatetime	TIMESTAMP
smallmoney	NUMERIC
varbinary	VARCHAR FOR BIT DATA

Microsoft SQL Server データ型から Oracle データ型への変換

Microsoft SQL Server データ型から Oracle データ型へのクラス・レベル変換を示します。

表 8 : Microsoft SQL Server データ型から Oracle データ型へのクラス・レベル変換

Microsoft SQL Server データ型	Oracle データ型
binary	RAW
datetime	DATE (時刻を含む)
money	DECIMAL
smalldatetime	DATE
smallmoney	DECIMAL
varbinary	RAW

Microsoft SQL Server に対応する Replication Server のデータ型名

Microsoft SQL Server データ型を表す Replication Server のユーザ定義データ型 (UDD) 名を示します。

Microsoft SQL Server のデータ型はすべて、対応する Adaptive Server のデータ型と互換性があります。ユーザ定義データ型定義がある Microsoft SQL Server データ型は 1 つだけです。

表 9 : Microsoft SQL Server データ型に対応する Replication Server UDD 名

Microsoft SQL Server データ型	Replication Server の名前
integer	*rs_msss_bigint*

Oracle データ型

Oracle データ型のクラス・レベル変換 (デフォルトのデータ型マッピング) と、Oracle データ型に対応する Replication Server のデータ型について説明します。

Adaptive Server データ型から Oracle データ型への変換

Adaptive Server データ型から Oracle データ型へのクラス・レベル変換を示します。

表 10 : Adaptive Server データ型から Oracle データ型へのクラス・レベル変換

Adaptive Server データ型	Oracle データ型
bigdatetime	TIMESTAMP (9)
bigint	NUMBER
bigtime	TIMESTAMP (9)
binary	RAW
date	DATE
datetime	DATE (時刻を含む)
money	DECIMAL
smalldatetime	DATE
smallmoney	DECIMAL
time	DATE (時刻を含む)
unsigned tinyint	SMALLINT
unsigned smallint	INTEGER
unsigned int	NUMBER
unsigned bigint	NUMBER
unitext	NCLOB
varbinary	RAW

Oracle データ型から Adaptive Server データ型への変換

Oracle データ型から Adaptive Server データ型へのクラス・レベル変換を示します。

表 11 : Oracle データ型から Adaptive Server データ型へのクラス・レベル変換

Oracle データ型	Adaptive Server データ型
RAW	varbinary
DATE	datetime
TIMESTAMP (9)	datetime

Oracle データ型から DB2 データ型への変換

Oracle データ型から DB2 データ型へのクラス・レベル変換を示します。

表 12 : Oracle データ型から DB2 データ型へのクラス・レベル変換

Oracle データ型	DB2 データ型
RAW	CHAR FOR BIT DATA
DATE	DATE
DATE (時刻を含む)	TIMESTAMP
FLOAT	DOUBLE (UNIX と Windows のみ)
INTEGER	INTEGER (UNIX と Windows のみ)
TIMESTAMP (9)	TIMESTAMP (UNIX と Windows のみ)

Oracle データ型から Microsoft SQL Server データ型への変換

Oracle データ型から Microsoft SQL Server データ型へのクラス・レベル変換を示します。

表 13 : Oracle データ型から Microsoft SQL Server データ型へのクラス・レベル変換

Oracle データ型	Microsoft SQL Server データ型
RAW	varbinary
DATE	datetime
TIMESTAMP (9)	datetime

Oracle に対応する Replication Server のデータ型名

Oracle データ型を表す Replication Server のユーザ定義データ型 (UDD) 名を示します。

表 14 : Oracle データ型に対応する Replication Server UDD 名

Oracle データ型	Replication Server の名前
RAW	*rs_oracle_binary*
DATE	*rs_oracle_datetime*
ROWID	*rs_oracle_rowid*
INTEGER	*rs_oracle_int*
INTERVAL	*rs_oracle_interval*
BINARY_FLOAT	*rs_oracle_float*
NUMBER	*rs_oracle_decimal*
TIMESTAMP(n)	*rs_oracle_timestamp9*
TIMESTAMP(n) (ローカル・タイム・ゾーンを含む)	*rs_oracle_timestamptz*
UDD オブジェクト・タイプ	*opaque*

マテリアライゼーション

異機種データ・サーバや ASE 以外のデータ・サーバを含む複写システムを実装するときに考慮する必要がある、サブスクリプション・マテリアライゼーションの問題、および ASE 以外のデータベースでプライマリ・テーブルのサブスクリプションをマテリアライズする方法についても説明します。

マテリアライゼーションとは、サブスクリプションを作成してアクティブ化し、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をコピーすることによって、レプリケート・データベースを初期化することです。

プライマリ・データベースからデータを複写する前に、レプリケート・オブジェクト (テーブルなど) がプライマリ・データベースの状態と一致した状態になるように、各レプリケート・データベースを設定してデータを読み込んでおく必要があります。

マテリアライゼーションの種類

Replication Server では、2 種類のサブスクリプション・マテリアライゼーションがサポートされています。

これらのタイプには次のようなものがあります。

- バルク・マテリアライゼーション ― サブスクリプションを手動で作成してアクティブ化し、複写システムのコントロール外のデータ・アンロード・ユーティリティとデータ・ロード・ユーティリティを使用してレプリケート・データベースにデータを読み込みます。
- 自動マテリアライゼーション ― Replication Server コマンドを使用してサブスクリプションを作成し、レプリケート・データベースにデータを読み込みます。

サブスクリプション・マテリアライゼーション・メソッドの詳細については、『Replication Server 管理ガイド 第 1 巻』の「サブスクリプションの管理」を参照してください。

異機種マテリアライゼーション

適用できる場合、ASE 以外のデータ・サーバでプライマリ・データのサブスクリプションをマテリアライズするには、バルク・マテリアライゼーションまたは自動マテリアライゼーションを使用できます。

バルク・マテリアライゼーション・メソッドの場合、以下のアクティビティを調整し、手動で実行する必要があります。

- サブスクリプションの定義、アクティブ化、および検証を行う (またはマテリアライゼーションを伴わないサブスクリプションを作成する)。
- プライマリ・データベースでサブスクリプション・データをアンロードする。
- アンロードしたデータをレプリケート・データベース・サイトに移動する。
- プライマリ・データをレプリケート・データベース・テーブルにロードする。
- レプリケート・データベースが複写トランザクションを受け取ることができるように、レプリケート Replication Server からレプリケート・データ・サーバへのデータベース・コネクションをレジュームする。
- Replication Agent インスタンスで複写をレジュームする。

バルク・マテリアライゼーションのオプション

ASE 以外のデータベースでのプライマリ・データのサブスクリプションには、2 つのバルク・マテリアライゼーションのオプションがあります。

これらのオプションには次のようなものがあります。

- アトミック・バルク・マテリアライゼーション
  - プライマリ・テーブルに対する更新を停止し、プライマリ・データベースからサブスクリプション・データをダンプする。
  - レプリケート Replication Server で、サブスクリプションを定義する。
  - プライマリ・データベースで、**rs_marker** ファンクションを使用して、**with suspension** オプションを使用しているサブスクリプションを有効にする。このファンクションの適用に関する詳細については、『Replication Server リファレンス・マニュアル』の「Replication Server システム・ファンクション」の「**rs_marker**」を参照してください。
  - レプリケート・テーブルにサブスクリプション・データをロードする。
  - レプリケート Replication Server からレプリケート・データベースへのデータベース・コネクションをレジュームする。
  - レプリケート Replication Server で、サブスクリプションを検証する。
- ノンアトミック・バルク・マテリアライゼーション
  - レプリケート Replication Server では、**set autocorrection** コマンドを使用する。
  - レプリケート Replication Server で、サブスクリプションを定義する。
  - プライマリ・データベースで、**rs_marker** ストアド・プロシージャを使用して、**with suspension** オプションを使用しているサブスクリプションを有効にする。
  - プライマリ・データベースからサブスクリプション・データをダンプする。

- プライマリ・データベースで、**rs_marker** ストアド・プロシージャを使用してサブスクリプションを検証する。
- レプリケート・テーブルにサブスクリプション・データをロードする。
- レプリケート Replication Server からレプリケート・データベースへのデータベース・コネクションをレジュームする。
- すべての Replication Server でサブスクリプションが有効になったら、オートコレクションをオフにする。

プライマリ・データベースからのデータのアンロード

サブスクリプション・マテリアライゼーション・プロセスには、プライマリ・テーブルからサブスクリプション・データをアンロードして、レプリケート・テーブルにロードできるようにするプロセスが含まれます。「サブスクリプション・データ」とは、サブスクリプションによって要求されるプライマリ・テーブル内のデータのことです。

データ・アンロード・ユーティリティは、通常、データ・サーバ・ソフトウェアに付属しています。OEM で提供される任意のデータ・アンロード・ユーティリティを使用することも、データベース・アンロード・ユーティリティを使用することもできます。

注意： プライマリ・データベースからサブスクリプション・データをアンロードした後、レプリケート・データベースにデータをロードする前に、アンロードしたデータに対してデータ型変換を実行しなければならない場合があります。

参照：

- データ型変換 (259 ページ)

データ型変換

アンロード・ユーティリティを使用しておらず、自動マテリアライゼーションを使用している場合は、Replication Server が変換を実行します。

Replication Server の異機種データ型サポート (HDS) 機能を使用して、複写データに対してカラム・レベル変換またはクラス・レベル変換を実行する場合は、マテリアライゼーションのためにプライマリ・データベースからアンロードしたサブスクリプション・データに対してデータ型変換を実行する必要があります。

レプリケート・データベースへのデータのロード

サブスクリプション・マテリアライゼーション・プロセスには、プライマリ・テーブルからレプリケート・テーブルにサブスクリプション・データをロードするプロセスが含まれます。

注意： プライマリ・データベースからサブスクリプション・データをアンロードした後、レプリケート・データベースにデータをロードする前に、アンロードしたデータに対してデータ型変換を実行しなければならない場合があります。

レプリケート・データベースのデータ・サーバとして Adaptive Server Enterprise を使用している場合は、ASE の **bcp** ユーティリティを使用して、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。

レプリケート・データベースのデータ・サーバとして ASE 以外のデータ・サーバを使用している場合は、任意のロード・ユーティリティを使用してレプリケート・データベースにサブスクリプション・データをロードできます。

『Adaptive Server Enterprise ユーティリティ・ガイド』の「ユーティリティ・コマンド・リファレンス」の「**bcp**」を参照してください。

アトミック・バルク・マテリアライゼーション

アトミック・バルク・マテリアライゼーションは、テーブルのコピーが作成される間、プライマリ・テーブルを更新するすべてのアプリケーションをサスペンドできることを前提とします。コピーはその後、レプリケート・サイトにロードされます。

プライマリ・データへの更新を (少なくとも一時的に) サスペンドできる場合は、このアトミック・バルク・マテリアライゼーションを使用してプライマリ・データベースからデータを取得できます。

マテリアライゼーションの準備

アトミック・バルク・マテリアライゼーションの手順を開始する前に、確認する必要があることがいくつかあります。

次のようなことを確認する必要があります。

- プライマリ・テーブルが存在し、格納データがある。
- プライマリ・テーブル (またはプライマリ・テーブルにある複写対象のカラム) に対して、所有権または **select** 権限を持つユーザ ID にアクセスできる。
- レプリケート・テーブルが存在し、適切なカラム、データ型を格納している。

- 複写システムにおいて、すべての Replication Server を適切に設定している。
- プライマリ Replication Server で、複写定義を正しく作成している。
- Replication Agent for a DB2 UDB、Microsoft SQL Server、または Oracle プライマリ・データベースを使用している場合は、次の点を確認する。
  - プライマリ・データベースでいくつかのオブジェクトも作成する Replication Agent を適切に初期化している。
  - プライマリ・データベースのプライマリ・テーブルに、マーク付けされ有効化された複写がある。
  - Replication Agent インスタンスを開始し、そのインスタンスを複写状態にしている。

アトミック・バルク・マテリアライゼーションの実行

アトミック・バルク・マテリアライゼーションの実行について説明します。

1. **isql** を使用して、システム管理者 **(sa)** としてレプリケート Replication Server にログインします。

   ```
   isql -Usa -Psa_password -SRRS_servername
   ```

   構文の説明は次のとおりです。
   - *sa* は、システム管理者のユーザ ID。
   - *sa_password* は、システム管理者のユーザ ID のパスワード。
   - *RRS_servername* は、レプリケート Replication Server のサーバ名。
2. レプリケート Replication Server で、サブスクリプションを定義します。

   ```
   1> define subscription subscription_name
   2> for replication_definition
   3> with replicate at dataserver.database
   4> [where search_conditions]
   5> go
   ```

   dataserver.database は、レプリケート・データベースに対して使用する Replication Server コネクション名と一致している必要があります。
3. 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をチェックします。サブスクリプションのステータスが **DEFINED** であることを確認するには、次のように入力します。

   ```
   1> check subscription subscription_name
   2> for replication_definition
   3> with replicate at dataserver.database
   4> go
   ```
4. プライマリ・トランザクションのアクティビティが行われないように、プライマリ・テーブルをロックします。これにより、マテリアライゼーション実行の間、プライマリ・テーブルに対する更新を避けることができます。

5. サイトの推奨されるデータベース・アンロード方法を使用して、プライマリ・サイトでサブスクリプション・データをアンロードし、プライマリ・テーブルからデータを選択またはダンプします。

   注意： プライマリ・テーブルからサブスクリプション・データをアンロードする場合は、複写定義で指定されているカラムと、サブスクリプションで指定されているローのみを選択するようにしてください。

6. サブスクリプション・データに必要なデータ型変換を実行します。

   このデータの複写定義でカラム・レベル変換が指定されている場合は、複写定義で指定されたデータ型変換を実行します。

   サブスクリプションに対してクラス・レベル変換が指定されている場合は、サブスクリプションに対して指定されたデータ型変換を実行します。

7. レプリケート Replication Server でサブスクリプションをアクティブにします。

```
1> activate subscription subscription_name
2> for replication_definition
3> with replicate at dataserver.database
4> with suspension
5> go
```

8. 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。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 **ACTIVE** であることを確認します。

   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** である場合は、レプリケート・データベースに対するデータベース・コネクションがサスペンドされています。

9. プライマリ・テーブルを読み込みおよび書き込みアクセスに戻します (ロックの解除)。

10. **bcp**、またはサイトで推奨されるデータベース・ユーティリティを使用して、サブスクリプション・データをレプリケート・データベースにロードします。

11. レプリケート Replication Server から、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。

```
1> resume connection
2> to dataserver.database
3> go
```

12. 次を実行して、レプリケート Replication Server でサブスクリプションを確定化します。

```
1> validate subscription subscription_name
2> for replication_definition
3> with replicate at dataserver.database
4> go
```

13. 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有効になるまで待機した後、**check subscription** を実行して、ステータスが **VALID** であ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・データと一致しています。複写は継続中です。

Replication Server の設定とマテリアライゼーション・メソッドの詳細については、『Replication Server リファレンス・マニュアル』の「Replication Server コマンド」および『Replication Server 管理ガイド 第 1 巻』を参照してください。

参照：

- Replicate Data Server としての Linux、UNIX、および Windows に関する IBM DB2 (91 ページ)

ノンアトミック・バルク・マテリアライゼーション

ノンアトミック・バルク・マテリアライゼーションは、テーブルのコピーが作成される間、プライマリ・テーブルを更新するアプリケーションをサスペンドできないことを前提とします。

そのため、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・データベースがプライマリ・データベースと同期されるように、Replication Server の「オートコレクション」機能を使用する必要があります。

注意： プライマリ・テーブルの複写定義に **replicate minimal columns** 機能が設定されている場合、ノンアトミック・マテリアライゼーションを使用することはできません。

マテリアライゼーションの準備

アトミックではないバルク・マテリアライゼーションの手順を開始する前に、確認する必要があることがいくつかあります。

次のことを確認します。

- プライマリ・テーブルが存在し、格納データがある。
- プライマリ・テーブル (またはプライマリ・テーブルにある複写対象のカラム) に対して、所有権または **select** 権限を持つユーザ ID にアクセスできる。
- レプリケート・テーブルが存在し、適切なカラムを格納している。
- 複写システムにおいて、すべての Replication Server を適切に設定している。
- プライマリ Replication Server で複写定義を正しく作成している。
- Replication Agent for a DB2 UDB、Microsoft SQL Server、または Oracle プライマリ・データベースを使用している場合は、次の点を確認する。

- プライマリ・データベースでいくつかのオブジェクトも作成する Replication Agent を適切に初期化している。
- プライマリ・データベースのプライマリ・テーブルに、マーク付けされ有効化された複写がある。
- Replication Agent インスタンスを開始し、そのインスタンスを複写状態にしている。

ノンアトミック・バルク・マテリアライゼーションの実行

アトミックではないバルク・マテリアライゼーションの実行について説明します。

1. **isql** を使用して、システム管理者 **(sa)** としてレプリケート Replication Server にログインします。

   ```
   isql -Usa -Psa_password -SRRS_servername
   ```

   構文の説明は次のとおりです。
   - *sa* は、システム管理者のユーザ ID。
   - *sa_password* は、システム管理者のユーザ ID のパスワード。
   - *RRS_servername* は、レプリケート Replication Server のサーバ名。
2. レプリケート Replication Server でオートコレクション機能をオンにします。

   ```
   1> set autocorrection on
   2> for replication_definition
   3> with replicate at dataserver.database
   4> go
   ```
3. レプリケート Replication Server で **with suspension** オプションを使用して、サブスクリプションを定義します。

   ```
   1> define subscription subscription_name
   2> for replication_definition
   3> with replicate at dataserver.database
   4> with suspension
   5> go
   ```

   dataserver.database は、レプリケート・データベースに対して使用する Replication Server コネクション名と一致している必要があります。
4. プライマリ・データベースで、**rs_marker** ストアド・プロシージャを呼び出して、サブスクリプションをアクティブにします。
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をチェックします。次を実行して、サブスクリプションのステータスが **ACTIVE** であることを確認します。

   ```
   1> check subscription subscription_name
   2> for replication_definition
   3> with replicate at dataserver.database
   4> go
   ```

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** である場合は、レプリケート・データベースに対するデータベース・コネクションがサスペンドされています。

6. サイトの推奨されるデータベース・アンロード方法を使用して、プライマリ・サイトでサブスクリプション・データをアンロードし、プライマリ・テーブルからデータを選択またはダンプします。

   注意： プライマリ・テーブルからサブスクリプション・データをアンロードする場合は、複写定義で指定されているカラムと、サブスクリプションで指定されているローのみを選択するようにしてください。

7. サブスクリプション・データに必要なデータ型変換を実行します。

   このデータの複写定義でカラム・レベル変換が指定されている場合は、複写定義で指定されたデータ型変換を実行します。

   サブスクリプションに対してクラス・レベル変換が指定されている場合は、サブスクリプションに対して指定されたデータ型変換を実行します。

8. プライマリ・データベースで、**rs_marker** ストアド・プロシージャを呼び出して、サブスクリプションを検証します。
9. 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有効になるまで待機した後、**check subscription** を実行して、ステータスが **VALID** であることを確認します。
10. **bcp** ユーティリティ、またはサイトで推奨されるデータベース・ロード・ユーティリティを使用して、サブスクリプション・データをレプリケート・データベースにロードします。
11. レプリケート Replication Server から、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。

```
1> resume connection
2> to dataserver.database
3> go
```

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有効になるまで待機した後、**check subscription** コマンドを実行して、ステータスが **VALID** であることを確認します。

    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 **VALID** の場合、次を実行すると、レプリケート・データベースはプライマリ・データベースと同期化されるので、オートコレクションをオフにすることができます。

```
1> set autocorrection off
2> for replication_definition
3> with replicate at dataserver.database
4> go
```

13. 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・データと一致しています。複写は継続中です。

- 複写コマンド言語 (RCL) 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「Replication Server コマンド」を参照してください。
- Replication Server の設定とマテリアライゼーション・メソッドの詳細については、『Replication Server 管理ガイド 第 1 巻』を参照してください。

も参照してください。

参照：

• Replicate Data Server としての Linux、UNIX、および Windows に関する IBM DB2 (91 ページ)

オートコレクション

Replication Server は **autocorrection** コマンドを複写定義に設定し、複写テーブルの重複しているローによって発生する障害を回避します。

set autocorrection コマンドは、各 **update** または **insert** オペレーションを **delete**、さらに **insert** 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ことのある不一致を修正します。

Replication Agents の API である **ra_set_autocorrection** を使って、マーク付けされたテーブルにオートコレクションを設定すると、**update** 文で変更されたカラムだけを送信するのではなく、Replication Agent はすべてのカラムを Replication Server に送信します。When Replication Agent が 1 つの特定のマーク付けテーブルまたはすべてのテーブルに対してオートコレクションを設定すると、該当する変更はプライマリ・データベースに適用されます。

オートコレクションをサポートするプライマリ・データベースは、次のとおりです。

- MS SQL Server
- IBM DB2
- Oracle – オートコレクション機能は、Oracle の redo ログ・レコードへの制限のために、`LOB`、`LONG`、`LONG RAW` およびユーザ定義型カラムを使用できません。
- ASE

『Replication Agent 15.5 Primary Database Guide』の「Replication Agent for Microsoft SQL」の「Replication Server **set autocorrection** Command」および『Replication Agent

15.5 Reference Manual』の「Command Reference」の「**ra_set_autocorrection**」を参照します。

異機種データベースの調整

異機種間複写システムにある異なるデータベースのデータの比較と調整に関する問題について説明します。

Sybase の rs_subcmp ユーティリティ

rs_subcmp ユーティリティを使用すると、Adaptive Server データベース内のプライマリ・テーブルとレプリケート・テーブルを比較し、差異を調整できます。Sybase では、**rs_subcmp** 実行プログラムを Replication Server に付属して提供しています。

他のデータベース・ベンダでも、自社のデータベースに対して同じ機能を実行できる同様の「比較」ユーティリティを提供している場合がありますが、異なる種類の ASE 以外のデータ・サーバ (Oracle データベース内のテーブルと Microsoft SQL Server データベース内のテーブルの比較など) をサポートする同等のユーティリティはありません。

ASE 以外のデータベースのサポートについては、そのような機能を備えたサード・パーティのツールを購入するか、独自のアプリケーションを作成できます。

データベース比較アプリケーション

rs_subcmp ユーティリティと同じ機能を実行するカスタム・アプリケーションを開発できます。アプリケーションの複雑さは、データ・サーバの種類の数、比較するテーブルの複雑さ、関連するデータ変換の量などによって異なります。

以下に、異機種間複写環境内でデータベース比較アプリケーションが適切に動作するうえで対処する必要がある主要な問題を列挙し説明します。

- コネクティビティ – アプリケーションは、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と通信できなければなりません。複数のデータベース・ベンダが関連している場合、ODBC および JDBC プロトコルが、共通のインタフェースと機能を提供できます。
- ソート順 – データベースごとにデフォルトのソート順が異なる可能性があります。アプリケーションは、比較のパフォーマンスを向上させるためにソート順を強制しなければならない場合があります。
- 文字セット – 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、異なる文字セットで文字データが格納されている可能性があります。カスタ

ム・アプリケーションは、これらの変換をサポートしなければならない場合があります。

- オブジェクトの識別 – プライマリ・テーブルとレプリケート・テーブルの名前が一致しない可能性があります。つまり、まったく同じスキーマ名やカラム名でない可能性があります。比較アプリケーションは、参照されるロケーション、データベース、テーブル名、カラム名に関する非常に明示的な命令を受け入れなければならない場合があります。
- サブセットの比較 ― アプリケーションは、テーブルの一部のみを比較しなければならない可能性があります。プライマリ・テーブルとレプリケート・テーブルの両方に、**where** 句タイプの **select** を指定できることが重要な場合があります。
- 遅延時間 ― 複写システムには、必ず遅延時間 (プライマリ・トランザクションがレプリケート・テーブルで発生するまでにかかる時間) があります。比較アプリケーションには、「そこにない」ローと「現時点ではそこにない」ローを区別する許容性を組み込む必要があります。
- データ変換 – アプリケーションは、Replication Server がクラス・レベル変換をサポートできるように、異なるデータベース間の精度とフォーマットに関する差異を処理できなければなりません。処理を単純にするには、データ型に基づいて特定のカラムが比較プロセスから除外されるようにします (たとえば、異なるデータベース・ベンダの `DATE` データ型を比較しない)。
- ラージ・オブジェクト (LOB) データ – ラージ・オブジェクト (例、`LOB`、`CLOB`、`TEXT`、または `IMAGE`) データ型では、サイズが原因となって追加処理の問題が生じます。引き続き「一致しない」可能性が高い場合は、パフォーマンスを向上させるために、比較に使用するバイト数を制限します。

『Replication Server 管理ガイド』および『Replication Server リファレンス・マニュアル』の「実行プログラム」の「**rs_subcmp**」に、**rs_subcmp** についての詳細が記載されています。

異機種間複写システムのトラブルシューティング

異機種データ・サーバや ASE 以外のデータ・サーバを含む Sybase 複写システムの一般的な問題とトラブルシューティングの手順について説明します。

ステーブル・キューのダンプ、データ・サーバ・インタフェース (DSI) および Replication Server インタフェース (RSI) に関する障害のデバッグ、サブスクリプションに関する問題の診断と修正など、Replication Server の一般的なトラブルシューティング作業については、『Replication Server トラブルシューティング・ガイド』で説明しています。

ASE 以外のプライマリ・データベースとレプリケート・データベースについては、Replication Agent と ECDA ゲートウェイのマニュアルに、データベースごとのトラブルシューティング情報が記載されています。

インバウンド・キューの問題

「インバウンド・キュー」は、Replication Server が (Replication Agent または別の Replication Server を介して) プライマリ・データベースから受け取ったデータを格納する場所です。

プライマリ Replication Server で Replication Server の **admin who,sqm** コマンドを発行した場合に、結果が以下のようであれば、プライマリ・データベースの Replication Server インバウンド・キューは更新されていないと判断できます。

- 該当するコネクションの Replication Server インバウンド・キューに書き込まれるブロックの数が変化しない。
- 検出された重複メッセージの数が増加しない。

インバウンド・キューが更新されていない原因の特定

インバウンド・キューが更新されていない原因を特定する方法について説明します。

1. Replication Server コネクションの Replication Agent ユーザ・スレッドのステータスを確認します。

   該当する Replication Server データベース・コネクションの Replication Agent ユーザ・スレッドのステータスを確認するには、プライマリ Replication Server で **admin who** コマンドを発行します。

- コネクションの Replication Agent ユーザ・スレッドがない場合、そのコネクションは **with log transfer on** 句を使用して作成されていません。必要に応じて、Replication Server データベース・コネクションを変更して、ログ転送処理をオンにします。
- Replication Agent ユーザ・スレッドのステータスが Down (停止) の場合、Replication Agent は Replication Server にアクティブに接続されていません。Down ステータスは、送信する作業がある場合にのみ Replication Server に接続し、一定期間、作業が行われなければ切断する Replication Agent にとって、典型的な状態です。

2. 必要な Replication Agent が実行中であることを確認します。

   必要な Replication Agent がアクティブであり、Replication Agent の **rs_source_ds** および **rs_source_db** 設定パラメータが、該当する Replication Server コネクション名と一致していることを確認してください。

   Replication Agent が実行中であることを確認するためのその他のテストについては、該当する Replication Agent のマニュアルを参照してください。

3. 必要なテーブルまたはプロシージャが、複写するようマーク付けされていることを確認します。

   複写のステータスを確認するために使用可能な Replication Agent のコマンドについては、Replication Agent のマニュアルを参照してください。

   Replication Agent には、マーク付けの他に、複写の「有効／無効」の設定があります。この場合、マーク付けされたオブジェクトが複写に対して有効であることも確認してください。

4. Replication Agent が新しいレコードをスキャンしていることを確認します。

   データベース・オブジェクトが複写するようマーク付けされている場合、Replication Agent のログ・スキャン・プロセスによって、追加情報がスキャンされていることが記録されます。

   新しいレコードがスキャンされていることを確認するには、次の手順に従います。
   - Replication Agent でトレースを開始します。
   - 複写するようマーク付けされたプライマリ・データベース・オブジェクトを更新または実行します。
   - スキャンが行われていることを確認します。

   スキャン・プロセスを検証するために使用できるトレース・フラグを確認するには、該当する Replication Agent のマニュアルを参照してください。

アウトバウンド・キューの問題

「アウトバウンド・キュー」は、Replication Server がレプリケート・サイト (レプリケート・データベースまたは別の Replication Server) に送信する必要があるデータを格納する場所です。

レプリケート Replication Server で Replication Server の **admin who,sqm** コマンドを発行した場合に、結果が以下のようであれば、レプリケート・データベースの Replication Server アウトバウンド・キューは更新されていないと判断できます。

- 該当するコネクションの Replication Server アウトバウンド・キューに書き込まれるブロックの数が変化しない。
- 検出された重複メッセージの数が増加しない。

インバウンド・キューとアウトバウンド・キュー間の問題は、多くの場合、名前の付け方に起因します。

プライマリ Replication Server インバウンド・キューが、データを受け取ることができても、そのデータを複写定義に適用できない場合は、複写定義の名前が、Replication Agent によって作成されたログ転送言語 (LTL) で提供された名前と一致していないためです。デフォルトの大文字と小文字の規則が異なる、Sybase 以外の異なるデータベース・タイプを使用している場合、この可能性はさらに高くなります。

Replication Server の複写コマンドの処理では、大文字と小文字を区別します。ASE 以外のデータ・サーバを含む複写システムでは、Replication Agent によって生成された LTL が、Replication Server コネクション名および複写定義オブジェクト名と一致していることを確認してください。

一部の Replication Agent は、Replication Server (Adaptive Server、DB2 UDB など) と通信するときに、常に小文字の名前を使用します。ただし、最もよい方法は、どちらかの文字 (大文字または小文字) を選択し、すべての Replication Server コネクション、複写定義、およびサブスクリプション名で一貫してそれを使用することです。

大文字と小文字の区別は手動で確認します。**rs_helprep** コマンドを使用すると、複写定義の名前を確認できます。その後、Replication Agent で LTL トレースをオンにし、LTL トレースで提供された名前が、複写定義で指定された名前のスペルおよび大文字と小文字の設定と一致しているかどうかを確認できます。

大文字と小文字の区別が正しくないと思われる場合は、Replication Agent のマニュアルを参照して、デフォルトの文字設定と可能な設定の変更を確認してください。名前のスペルが正しくない場合は、複写定義を削除した後、作成し直してください。

アウトバウンド・キューが更新されていない原因の特定

アウトバウンド・キューが更新されていない原因を特定する方法について説明します。

1. アクティブな Replication Server ルートがあることを確認します。

   プライマリ Replication Server とレプリケート Replication Server 間のルートの検証方法については、『Replication Server トラブルシューティング・ガイド』の「ルートの問題」を参照してください。
2. Replication Server コネクションの DSI スレッドが停止していないことを確認します。

   Replication Server コネクションの DSI スレッドのステータスを確認するには、レプリケート Replication Server で **admin who** コマンドを発行します。

   DSI スレッドのステータスが Down (停止) の場合、Replication Server はレプリケート・データベース (または ECDA ゲートウェイ) に接続されていません。Replication Server ログでエラーを確認し、コネクションをレジュームしてください。
3. レプリケート Replication Server ログで該当する DSI スレッドの “Loss Detected” メッセージを参照して、DSI スレッド・コネクションが “Loss Detected” モードでないことを確認します。

   Replication Server がロスを検出した場合、DSI スレッド・コネクションでは、それ以上メッセージが受け入れられません。

   このエラーのリカバリについては、『Replication Server 管理ガイド』を参照してください。
4. プライマリ複写定義を確認します。

レプリケート・データベースがなぜ更新されないかを特定します。

レプリケート・トランザクションがレプリケート・データベースで適用されない原因を特定する方法を説明します。

Replication Server アウトバウンド・キューが更新されているにもかかわらず、レプリケート・データベースでトランザクション・データが適用されていない場合、原因を特定するには、以下の手順に従います。

1. サブスクリプションに **where** 句が含まれているかどうかを調べます。

サブスクリプション定義で、該当するトランザクション・データが **where** 句を渡すことを確認します。**rs_helpsub** ストアド・プロシージャを使用して、サブスクリプションのテキストを一覧表示します。

2. HDS のインストールを確認します。

   ASE 以外のデータ・サーバとの複写をサポートするために Replication Server HDS を使用している場合は、HDS の接続プロファイルが正しく適用されていることを確認します。

3. `rs_lastcommit` テーブルが正しく設定されていることを確認します。

   ASE 以外のデータ・サーバとの複写をサポートするために Replication Server HDS を使用している場合は、HDS の接続プロファイルが正しく適用されていることを確認します。

4. レプリケート Replication Server ログでエラーを確認します。
5. レプリケート・データベース・ログでエラーを確認します。
6. レプリケート・オブジェクトへの手動でのアクセスを確認します。

   Replication Server コネクションのメンテナンス・ユーザ ID を使用して、レプリケート・データベース (または ECDA ゲートウェイ) にログインし、レプリケート・テーブルまたはプロシージャに対する **update** 権限があることを確認します。

7. レプリケート・データベースに送信されるコマンドを、次のように確認します。
   - レプリケート Replication Server で **DSI_BUF_DUMP** トレース・フラグをオンにし、レプリケート・データベースに送信されるコマンドを Replication Server ログに記録する。
   - これらのコマンドを手動で適用した場合に、想定した結果が生成されることを確認する。

   注意： **DSI_BUF_DUMP** トレース・フラグは、すべての Replication Server で使用できます。これに対して、類似した **DSI_CMD_DUMP** トレース・フラグは、Replication Server の診断バージョンでしか使用できません。Replication Server トレース・フラグについての詳細は、『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。

8. ECDA ゲートウェイでトレースをオンにして、受け取られるコマンドを確認します。

   たとえば、ECDA Option for Oracle 設定ファイル内の次のパラメータによって、ECDA は `DCO.log` ファイルに追加情報を書き込みます。
   - **network_tracing = 1**
   - **traces = 1,2,3,4,5,6,10**

特定のトレースの使用範囲と構文については、該当する ECDA のマニュアルを参照してください。

参照：

- 想定したデータ型変換が行われない (280 ページ)
- rs_lastcommit が更新されない (279 ページ)

HDS の問題と制限事項

この項では、Replication Server の HDS 機能に関する既知の問題と制限事項について説明します。

変換先のデータ型の範囲を超える変換元の値

Sybase によって提供されているデータ型変換では、変換元の値が変換先のデータ型の範囲を超える変換を行おうとするスレッドは停止されます。

これは以下のエラー・メッセージです。

```
E. 2007/12/14 11:14:54. ERROR #32055 DSI EXEC(135(1)
   snickers_dco.ora805) -
   /nrm/nrm.c(7023)
   Class Level translation for column/parameter
   'datetimecol' failed.
   Source DTID is 'datetime'.
   Target DTID is 'rs_oracle_datetime'.
   Function String Class ID 'rs_oracle_function_class'.
   Value length is '21'; Maximum target length is '20';
   The value is '99991231 23:59:59:010'
```

これらは通常、変換元／変換先データ型の組み合わせにも変換される値にも問題がないように見えるため、診断が最も困難な変換の問題です。

このような問題を診断するには、変換されるすべての変換元データ型のデータ型値の範囲を理解しておくことが必要です。たとえば、上記のエラーを診断するには、Oracle の `DATE` 値の上限が 12/31/9999 であることを理解していなければなりません。

データ型変換については、次のようなオプションもあります。

- データ型定義の最大値を使用する。
- データ型定義の最小値を使用する。
- データ型定義のデフォルト値を使用する。

真数値データ型の問題

複写される値がデータ型定義でサポートされている値の範囲 (最大値または最小値) に達すると、真数値データ型に関する問題が生じることがあります。

Microsoft SQL Server では、サーバの起動方法によって 28 桁または 38 桁の精度がサポートされます。デフォルトでは、Microsoft SQL Server は 28 桁の精度をサポートします。

しかし Sybase では、Microsoft のデフォルトである 28 桁の精度をサポートするデータ型定義を提供していません。Replication Server のネイティブな numeric データ型は 72 桁までの精度をサポートするので、データ型定義で 38 桁の精度をサポートする必要はありません。

数値が Microsoft SQL Server レプリケート・データベースの数値精度を超えると、Replication Server は次のエラーを返します。

```
E. 2007/12/14 11:14:58. ERROR #1028 DSI EXEC(134(1)
   dcm_gabeat70_devdb.devdb)
   - dsiqmint.c(2888)
   Message from server: Message: 30291, State 0,
   Severity 19 --
   '[VENDORLIB] Vendor Library Error: [[Message
   Iteration=1|Data Source
   Name=mssql70_devdb|SQLState=22003|Native
   Error=1007|Message=[Microsoft][ODBC SQL Server
   Driver][SQL Server]The number
   '9999999999999999999.9999999999999999999' is out of
   the range for numeric representation (maximum
   precision 28).[Message Iteration=2|SQLState=22003|
   Native Error=|Message=[Microsoft][ODBC SQL Server
   Driver][SQL Server]The number
   '0.99999999999999999999999999999999999999' is out of
   the range for numeric representation (maximum
   precision 28).] <DCA>'
```

numeric データ型の最も困難な問題として、精度と位取りがあります。Replication Server では、decimal データ型の精度と位取りを指定することはできません。データ型定義では、サポートされる最大精度と最大位取りを指定できます。しかし、これが個々のレプリケート・カラムの指定された精度と位取りと一致しない場合、データが範囲値に近づくか範囲値に達すると、問題が生じることがあります。これらの問題は、レプリケート・データ・サーバに応じた方法で報告されます。

たとえば、`decimal (8,5)` (8 桁の精度と 5 桁の位取り) として宣言されたプライマリ・カラムがあるとします。また、レプリケート・データ・サーバでは最大 7 桁の精度と 7 桁の位取りをサポートできるにも関わらず、レプリケート・カラムが `decimal (6,4)` として宣言されているとします。複写定義で、プライマリ・

データ・サーバの decimal データ型の変換を指定します。この変換では、レプリケート・データ・サーバの decimal データ型へのクラス・レベル変換が行われます。両方のデータ型定義で、関連するデータ・サーバの最大の精度と位取りを指定します。

値 999.99999 がプライマリ・データベースから複写される場合、レプリケート・データ・サーバのデータ型定義で丸めが行われるように指定されていると、Replication Server は 1000.000 という値を適用しようとします。この値がレプリケート・データベースの最大精度と位取りの要件を満たしていても、この特定のカラムについて指定された精度と位取りでエラーが発生します。また、レプリケート・データベースのデータ型定義で、データ型定義について指定された最大値で値を置換するように指定した場合、Replication Server は 9999999 という値を適用しようとします。この場合も、この特定のカラムについて指定された精度と位取りでエラーが発生します。

この場合にさまざまなデータ・サーバから表示されるエラー・メッセージは、次のようなものになります。

- DB2 のエラー

```
E. 2007/12/14 15:03:11. ERROR #1028 DSI EXEC(129(1)
   dwm5_via_rct.dwmdbas)
   - dsiqmint.c(2888)
   Message from server: Message: 30291, State 0,
   Severity 19 --
   '[VENDORLIB] Vendor Library Error: [[Message
   Iteration=1|SQLState=22003|Native Error=
   -413|Message=[Sybase][ClearConnect ODBC][DB2]The
   decimal or numeric value had an incorrect wire
   length compared to its specified FDOCA length
   10000000000000000000.00000000000] <DCA>'.
```

- Microsoft SQL Server のエラー

```
E. 2007/12/14 12:29:16. ERROR #1028 DSI EXEC(134(1)
   dcm_gabeat70_devdb.devdb)
   - dsiqmint.c(2888)
   Message from server: Message: 30291, State 0,
   Severity 19 --
   '[VENDORLIB] Vendor Library Error: [[Message
   Iteration=1|Data Source Name=mssql70_devdb|SQL
   Function=INSERT|SQLState=22003|Native Error=
   8115|Message=[Microsoft][ODBC SQL Server Driver]
   [SQL Server]Arithmetic overflow error converting
   numeric to data type numeric.[Message Iteration=
   2|SQLState=01000|Native Error=|Message=
   [Microsoft][SQL Server]The statement has been
   terminated.] <DCA>'
```

Microsoft SQL Server での数値変換と identity カラム

Microsoft SQL Server では、Adaptive Server と同じように identity カラムをサポートしているので、identity の挿入をオフおよびオンに設定する Replication Server のファンクション文字列が機能します。

しかし、Microsoft SQL Server データベースの 28 桁の精度をサポートするには、`numeric` データ型が `rs_msss_numeric` データ型に変換されなければなりません。その結果、ID 特性が失われます。この問題を回避するには、Microsoft SQL Server レプリケート・テーブルが、変換された `numeric` カラムを identity として宣言しないようにしてください。

変換された `numeric` データ型を Microsoft SQL Server で identity カラムに複写しようとすると、次のようなエラーが表示されます。

```
E. 2007/12/14 12:05:39. ERROR #1028 DSI EXEC(134(1)
   dcm_gabeat70_devdb.devdb)
   - dsiqmint.c(2888)
   Message from server: Message: 30291, State 0,
   Severity 19 --
   '[VENDORLIB] Vendor Library Error: [[Message
   Iteration=1 |Data Source Name=mssql70_devdb|SQL
   Function=INSERT|SQLState=23000|Native Error=544
   |Message=[Microsoft][ODBC SQL Server Driver][SQL
   Server]Cannot insert explicit value for identity
   column in table 'ase_alltypes' when IDENTITY_INSERT
   is set to OFF.] <DCA>'
```

特定のエラーのトラブルシューティング

この項では、異機種データ・サーバや ASE 以外のデータ・サーバを含む Sybase 複写システムにおいて、発生する可能性がある特定のエラーのトラブルシューティングについて説明します。

rs_lastcommit が更新されない

ASE 以外のレプリケート・データベースに複写する場合、レプリケート Replication Server は、コネクションがレジュームされると、すぐに `rs_lastcommit` テーブルを更新します。

rs_lastcommit の更新エラーのトラブルシューティング

`rs_lastcommit` テーブルの更新中にレプリケート Replication Server エラー・ログに構文エラーが表示された場合は問題を特定します。

1. テーブルがレプリケート・データベースに存在することを確認します。

2. アクセス権限を確認します。

   そのデータベース・コネクションについて **create connection** コマンドで指定された Replication Server のメンテナンス・ユーザ ID 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。

   このユーザ ID を使用して `rs_lastcommit` table テーブルを更新できることを確認します。ダミー・エントリをエラーの発生なしに挿入および削除できなければなりません。
3. 実際のコマンドをトレースします。

   レプリケート Replication Server (**DSI_BUF_DUMP** トレース) または ECDA ゲートウェイでトレースをオンにし、Replication Server コネクションをレジュームします。

   エラーが発生している文を特定し、必要に応じて修正します。

想定したデータ型変換が行われない

データ型変換エラーの最も一般的な原因は、必要なユーザ定義データ型 (UDD) と変換のインストールが完全でないことです。

UDD と変換のインストールの確認

UDD と変換のインストールを確認する方法を説明します。

1. Replication Server を再起動します。Replication Server は、起動時にすべてのファンクション文字列情報をキャッシュします。

   RSSD に格納されているファンクション文字列に対するそれ以降の変更は、Replication Server が再起動されるまで有効になりません。
2. クラス・レベル変換が、レプリケート Replication Server に適用されていることを確認します。

   Replication Server の接続プロファイルには、レプリケート Replication Server の RSSD にクラス・レベル変換を適用するために必要な、ASE 以外のプライマリ・データベースと ASE 以外のレプリケート・データベースの特定の組み合わせ用の SQL 文が含まれています。

   注意： 接続プロファイルは、ASE 以外のレプリケート・データベースに必要です。たとえば、ASE から Oracle に複写する場合は、`rs_lastcommit` テーブルに対する Replication Server の更新が適切に変換され、レプリケート・データベースに適用されるように、変換用の `rs_ase_to_oracle` 接続プロファイルを適用する必要があります。

これらの接続プロファイルは、再実行してもエラーにはなりません。接続プロファイルのコピーが、RSSD のデータベース名に対する正しい **use** 文で更新されていることを確認します。

3. レプリケート・データベースの Replication Server コネクションが、適切なファンクション文字列クラスと関連付けられていることを確認します。

   クラス・レベル変換を利用するには、レプリケート Replication Server コネクションで、ASE 以外の正しいファンクション文字列クラスが使用されていることが必要です。

   Replication Server の **rs_helpdb** コマンドを使用すると、データベース・コネクションに定義されているファンクション文字列クラスを調べることができます。

   レプリケート・データベースのファンクション文字列クラスは、次のとおりです。

   - Adaptive Server Enterprise — **rs_sqlserver_function_class**
   - IBM z/OS プラットフォーム上の DB2 UDB – **rs_db2_function_class**
   - UNIX および Windows プラットフォーム上の DB2 UDB – **rs_udb_function_class**
   - Microsoft SQL Server — **rs_msss_function_class**
   - Oracle – **rs_oracle_function_class**
   - Sybase IQ – **rs_iq_function_class**

   アクティブなファンクション文字列クラスのリストを表示するには、Replication Server の **admin show_function_classes** コマンドを使用します。

   既存のデータベース・コネクションのファンクション文字列クラスを変更するには、Replication Server の **alter connection** コマンドを使用します。

4. ASE 以外のファンクション文字列クラスが、適切なファンクション文字列で更新されていることを確認します。

   Replication Server の接続プロファイル `rs_xxx_xxx` には、レプリケート Replication Server の RSSD にファンクション文字列を適用するために必要な、ASE 以外の特定のレプリケート・データベース用の SQL 文が含まれています。

   各ファンクション文字列に対して、この接続プロファイルは、**delete** に続いて **insert** を発行します。これらの接続プロファイルは、再実行してもエラーにはなりません。

   接続プロファイルのコピーが、RSSD のデータベース名に対する正しい **use** 文で更新されていることを確認します。

5. Replication Server **admin translate** コマンドを使用します。

admin translate コマンドを使用すると、特定の変換の結果を確認できます。このコマンドを使用して、想定した変換結果が変換エンジンによって提供されることを確認してください。

異機種データ型サポート (HDS) および **admin translate** コマンドの詳細については、『Replication Server 管理ガイド 第 1 巻』の「複写テーブルの管理」の「HDS を使用したデータ型の変換」を参照してください。

ログ転送言語の生成とトレース

この項では、プライマリ Replication Server に送信されるログ転送言語 (LTL) コマンドをトレースする方法と、Replication Agent のその他の重要なトレースについて説明します。

Replication Agent for DB2 UDB for z/OS

この項で説明する設定パラメータを使用すると、Replication Agent for DB2 UDB for z/OS では通常提供されない追加情報を取得できます。

各ログ・レコードのログ・レコード識別子と、DB2 API から受け取った追加のメッセージを印刷するには、**Logtrace** = **Y** を `LTMCFG` に入力します。

注意： 通常、これらのパラメータを使用すると、パフォーマンスに何らかの影響があります。『Replication Agent for DB2 UDB Installation Guide』でパラメータの説明をよく読んでから、使用してください。

- Replication Agent のユーザ Exit に渡された情報のデバッグに使用するために、追加のトレースが必要な場合は、**API_com_test** 設定パラメータの値を **Y** に設定します。このトレースは、ユーザ Exit が使用されない場合にも使用できます。
- **LTL_test_only** 設定パラメータは、LTM for z/OS が Replication Server に接続し、複写のためにトランザクション・オペレーションを送信するかどうかを制御します。**LTL_test_only** パラメータの値が **Y** の場合、通常は Replication Server に送信される LTL が、代わりに `LTLOUT` ファイルに書き込まれます。

  注意： **LTL_test_only** パラメータの値が **Y** の場合、Replication Agent for DB2 UDB は Replication Server に接続されません。
- **trace=LTLebcdic** 設定パラメータは、Replication Server に渡される EBCDIC LTL を `LTLOUT` に書き込みます。ASCII データを含むテーブルを複写する場合は、ASCII 文字が `LTLOUT` データ・セットに書き込まれるように、trace = **LTLASCII** を設定します。このトレースをオンにするには、これらのパラメータの値を **Y** に設定してください。
- **Use_repdef** 設定パラメータを使用すると、LTM for z/OS が複写定義で指定されたカラムのみを格納する LTL を Replication Server に送信するように設定できます。

use_repdef パラメータの値を **N** に設定すると、LTL トレースで提供される情報量が増える可能性があります。

- **suppress_col_names** 設定パラメータは、LTM for z/OS が、Replication Server に送信される LTL で、カラム名を非表示にするかどうかを決定します。

  LTL の出力をトレースしている場合、**suppress_col_names** の値を **N** に設定すると、生成された LTL でカラム名が表示されます。

Replication Agent

この項で説明するトレース・フラグおよび設定パラメータを使用すると、Microsoft SQL Server、Oracle、UDB 用の Replication Agent では通常提供されない追加情報を取得できます。

注意： 通常、これらのトレース・フラグおよびパラメータを使用すると、パフォーマンスに何らかの影響があります。フラグまたはパラメータを使用する前に、『Replication Agent 管理ガイド』の説明をよく読んでください。

トレース・フラグ

通常のトレース出力は、Replication Agent インスタンスのログ・ファイルに送信されます。ただし、**LTITRACELTL** トレース・ポイントからの出力は、別の LTL 出力ログ・ファイル (`LTITRACELTL.log`) に送信されます。

Replication Agent の問題をトラブルシューティングするには、以下のトレース・フラグが特に便利です。

- **LRTRACE** — Log Reader コンポーネントの通常の実行をトレースします。
- **LTITRACE** — Log Transfer Interface コンポーネントの通常の実行をトレースします。
- **LTITRACELTL** — `LTITRACELTL.log` ファイルでの LTL 文のトレースを有効にします。
- **RACONTRC** — コネクションおよびクエリの実行をトレースします。
- **RACONTRCSQL** — プライマリ・データベースに送信される SQL 文をトレースします。

設定パラメータ

Replication Agent の以下の設定パラメータの設定は、トレース情報に影響します。

- **compress_ltl_syntax** – **false** に設定すると、LTL コマンドのより詳細な説明が表示されます。
- **connect_to_rs** – **false** に設定すると、Replication Server への実際の接続や情報の送信を伴わない LTL が生成されます。
- **log_trace_verbose** – **true** に設定されている場合、トレースされるコンポーネントのより詳細な説明を提供します。

- **use_rssd** – **false** に設定されていると、複写定義情報の変更を伴わない LTL コマンド全体の生成を提供します。
- **column_compression** – **false** に設定すると、**update** オペレーションを目的として生成された LTL 内のすべてのカラム情報 (更新後イメージ内のすべてのカラム) が送信されます。

Replication Agent のトレース・フラグと設定の詳細については、『Replication Agent 管理ガイド』を参照してください。

Oracle から Oracle への複製のリファレンス実装

Replication Server には、ご使用の環境で使用可能な製品を使って、Oracle から Oracle への複写のリファレンス実装をすばやくセットアップするためのツールセットが含まれています。

リファレンスとして複写環境を実装すると、Replication Server のさまざまな機能を試すことができます。ツールセットを使用すると、次の手順を実行できます。

1. Replication Server、プライマリ・データベース、レプリケート・データベースを構築します。
2. データベース複写環境を設定します。
3. プライマリ・データベースで単純なトランザクションを実行し、データベース・レベルの複写を使用して変更を複写します。
4. 手順 3 の複写処理から統計とモニタ・カウンタを収集します。
5. リファレンス複写環境をクリーンアップします。

リファレンス実装ツールセットはスクリプトで構成されており、`$SYBASE/REP-15_5/REFIMP-01_0` にあります。

注意： リファレンス実装は、単一の Replication Server、プライマリ・データベース・サーバ、レプリケート・データベース・サーバのみを提供します。複数の複写システム・コンポーネント用のリファレンス環境トポロジは設定できません。

リファレンス環境の要件、手順、設定ファイルのサンプル構築、リファレンス環境の実装で作成されたオブジェクトの詳細については、『Replication Server 管理ガイド 第 2 巻』の「リファレンス複写環境の実装」を参照してください。

プラットフォームのサポート

リファレンス環境は、Linux on IBM p-Series (Linux on Power) 64 ビット版を除く、Replication Server がサポートしているすべてのプラットフォームに実装できます。

ただし、リファレンス環境を Replication Server がサポートする Microsoft Windows プラットフォームでセットアップするには、Cygwin を使ってリファレンス実装スクリプトを実行する必要があります。Cygwin Web サイト：http://www.cygwin.com/ を参照してください。

Oracle のリファレンス実装でサポートされている製品コンポーネントのバージョン

Oracle から Oracle への複写のためのリファレンス実装環境の構築に使用できる、Replication Server、Oracle、Replication Agent for Oracle、および ECDA Option for Oracle のサポートされているバージョンの一覧を示します。

Oracle のリファレンス実装では、これらの製品コンポーネント・バージョンがサポートされています。

- Replication Server 15.5
- Oracle 10.2
- Replication Agent for Oracle 15.2
- 15.0 ESD #3 用の ECDA オプション

たとえば、Oracle 用のリファレンス実装環境の構築には、Replication Server 15.5、Oracle 10.2、Replication Agent 15.2 for Oracle、および ECDA Option for Oracle 15.0 ESD #3 が必要です。

用語解説

複写システムで使用される用語を解説します。

- **アクティブ・データベース** – ウォーム・スタンバイ・アプリケーションにおいて、スタンバイ・データベースに複写されるデータベースです。「*ウォーム・スタンバイ・アプリケーション*」も参照してください。
- **Adaptive Server** – Sybase バージョン 11.5 およびそれ以降のリレーショナル・データベース・サーバです。Replication Server の設定中に RSSD オプションを選択すると、Adaptive Server は RSSD データベースの Replication Server システム・テーブルを管理します。
- **アプリケーション・プログラミング・インタフェース (API)** – ユーザまたはプログラムが相互に通信するために使用する、事前に定義されたインタフェースです。Sybase Open Client および Sybase Open Server は、クライアント／サーバ・アーキテクチャで通信を行う API の 1 つです。RCL (複写コマンド言語) は、Replication Server の API です。
- **適用ファンクション** – ファンクション複写定義に対応する複写ファンクションです。Replication Server によってプライマリ・データベースからファンクション定義のサブスクリプションを持つレプリケート・データベースに配信されます。適用ファンクションがストアド・プロシージャにパラメータ値を渡し、そのストアド・プロシージャがレプリケート・データベースで実行されます。レプリケート・データベースにあるストアド・プロシージャはメンテナンス・ユーザによって実行されます。「*複写ファンクションの配信*」、「*要求ファンクション*」、「*ファンクション複写定義*」も参照してください。
- **アーティクル** – テーブルまたはストアド・プロシージャの複写定義を拡張したもので、パブリケーションの要素となります。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。
- **非同期プロシージャ配信** – プライマリ・データベースまたはレプリケート・データベースで複写するように指定されたストアド・プロシージャを実行できる Replication Server システムの一部です。
- **非同期コマンド** – クライアントが送信するコマンドです。クライアントは、完了ステータスの受信を待たずに、他のオペレーションを継続できます。Replication Server のコマンドの多くは、複写システム内で非同期コマンドとして動作します。
- **アトミック・マテリアライゼーション** – マテリアライゼーション・メソッドの 1 つです。**select** オペレーションを holdlock を指定して使用し、1 つのアトミック・オペレーションでネットワークを介して、プライマリ・データベースからレプリケート・データベースへサブスクリプション・データをコピーします。データの転送が完了するまで、プライマリ・データへの変更は行えません。レ

プリケート・データは、1 つのトランザクションとして、またはレプリケート・データベースのトランザクション・ログが一杯にならないようにトランザクションごとに 10 ローずつ挿入する方法のいずれかを使用して適用できます。アトミック・マテリアライゼーションは、**create subscription** コマンドのデフォルトのメソッドです。「*ノンアトミック・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*」、「*非マテリアライゼーション*」も参照してください。

- **autocorrection** – オートコレクションは、複写定義に適用する機能で、レプリケート・テーブルに、消失ローや重複したローが発生して障害が起こることを防ぎます。**set autocorrection** コマンドを使用して設定します。オートコレクションを有効にすると、Replication Server は各更新オペレーションまたは挿入オペレーションを削除と挿入の連続オペレーションに変換します。オートコレクションは、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を使用している複写定義の場合だけ使用できます。
- **基本クラス** – 親クラスからファンクション文字列を継承しないファンクション文字列クラスです。「*ファンクション文字列クラス*」も参照してください。
- **ビットマップ・サブスクリプション** – ビットマップの比較に基づいてローを複写するサブスクリプションです。`int` データ型のカラムを作成し、複写定義を作成するときには、カラムを `rs_address` データ型として指定します。サブスクリプションを作成するときには、**where** 句でビットマップ比較演算子 (&) を使用し、各 `rs_address` カラムとビットマスクを比較します。サブスクリプションのビットマップと一致するローが複写されます。
- **バルク・コピー・イン** – 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に Replication Server のパフォーマンスを向上させる機能です。Replication Server は、Open Client™ Open Server™ Bulk-Library を使用して、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 Replication Server モジュールであるデータ・サーバ・インタフェース (DSI) にバルク・コピー・インを実装します。

  バルク・コピー・インにより、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も向上します。**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。
- **バルク・マテリアライゼーション** – マテリアライゼーションのメソッドの 1 つです。これは、複写システム以外でレプリケート・データベースのサブスクリプションのデータを初期化します。たとえば、磁気テープ、フロッピー・ディスク、CD-ROM、または光磁気ディスクなどのメディアを使用して、プライマリ・データベースからデータを転送できます。バルク・マテリアライゼーションでは、**define subscription** から始まる一連のコマンドを使用します。バルク・マテリアライゼーションは、テーブル複写定義とファンクション複写定義のどちらのサブスクリプションにも使用できます。「*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*」、「*非マテリアライゼーション*」も参照してください。

- **集中型データベース・システム** – 中央サイトに設置された 1 つのデータベース管理システムでデータを一元管理するデータベース・システムです。
- **クラス** –「*エラー・クラス*」と「*ファンクション文字列クラス*」を参照してください。
- **クラス・ツリー** – 派生クラスと親クラスの複数のレベルから構成されるファンクション文字列クラスのセットです。これは、同じ基本クラスから派生します。「*ファンクション文字列クラス*」も参照してください。
- **クライアント** – クライアント／サーバ・アーキテクチャにおいて、サーバに接続されたプログラムです。ユーザが実行するフロントエンド・アプリケーション・プログラムの場合もあれば、システムの拡張機能として実行されるユーティリティ・プログラムの場合もあります。
- **Client/Server Interfaces (C/SI)** – クライアント／サーバ・アーキテクチャで実行するプログラムのための Sybase のインタフェース標準です。
- **同時実行性** – 複数のクライアントが、データまたはリソースを共有できることを示します。データベース管理システムにおける同時実行性は、あるクライアントが使用中のデータを別のクライアントが変更しようとするときに発生する競合からクライアントを保護するシステムに依存します。
- **接続** – Replication Server からデータベースへ、または LTM から Replication Server への接続です。「*データ・サーバ・インタフェース (DSI)*」と「*論理コネクション*」も参照してください。
- **接続プロファイル** – 接続プロファイルを使用すると、事前に定義されたプロパティのセットでコネクションを設定できます。
- **コーディネート・ダンプ** – 複写システムでファンクション **rs_dumpdb** または **rs_dumptran** を実行することによって、複数のサイト間で同期がとられているデータベース・ダンプ・セット、またはトランザクション・ダンプ・セットです。
- **データベース** – 相互に関連するデータ・テーブルとその他のオブジェクトが、特定の目的に合わせて編成、表現されたものです。
- **データベース世代番号** – データベースおよびデータベースを管理する Replication Server の RSSD に格納されます。データベース世代番号は、各ログ・レコードのオリジン・キュー ID (*qid*) の最初の部分です。オリジン・キュー ID は、Replication Server が重複したレコードを処理していないことを示します。リカバリ・オペレーション中に、データベース世代番号を増やさなければならない場合があります。このようにすることで、Replication Server は、データベースが再ロードされた後に送信されたレコードを無視しません。
- **データベース複写定義** – サブスクリプションを作成できる対象のデータベース・オブジェクト (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) を集めて記述したものです。

テーブル複写定義とファンクション複写定義も作成できます。「*テーブル複写定義*」と「*ファンクション複写定義*」も参照してください。

- **データベース・サーバ** – Sybase Adaptive Server などのサーバ・アプリケーションです。クライアントにデータベース管理サービスを提供します。
- **データ定義言語 (DDL)** – Transact-SQL などのクエリ言語のコマンド・セットであり、データとデータベース内でのデータの関係を記述します。Transact-SQL の DDL コマンドには、**create**、**drop**、**alter** キーワードを使用するものなどがあります。
- **データ操作言語 (DML)** – Transact-SQL などのクエリ言語のコマンド・セットであり、データの操作を行います。Transact-SQL の DML コマンドには、**select**、**insert**、**update**、**delete** があります。
- **データ・サーバ** – Sybase Client/Server Interfaces に準拠のクライアント・インタフェースによって、データベースの複写テーブルの物理表現を管理するのに必要な機能を提供するサーバです。データ・サーバは、通常、データベース・サーバと同じものですが、Replication Server に必要なインタフェースと機能を備えたデータ・リポジトリの場合もあります。
- **データ・サーバ・インタフェース (DSI)** – Replication Server とデータベース間のコネクションに対応している Replication Server スレッドです。DSI スレッドは、DSI アウトバウンド・キューからレプリケート・データ・サーバへトランザクションを送信します。DSI スレッドは 1 つのスケジューラ・スレッドと 1 つまたは複数のエグゼキュータ・スレッドで構成されます。スケジューラ・スレッドは、トランザクションをコミット順にグループ分けしてから、それらをエグゼキュータ・スレッドにディスパッチします。エグゼキュータ・スレッドは、ファンクションをファンクション文字列にマッピングし、レプリケート・データベースのトランザクションを実行します。DSI スレッドは、データベースへの Open Client コネクションを使用します。「*アウトバウンド・キュー*」と「*コネクション*」も参照してください。
- **データ・ソース** – リレーショナル・データ・サーバまたはノンリレーショナル・データ・サーバなどのデータベース管理システム (DBMS) 製品、その DBMS にあるデータベース、および複写システムの他のコンポーネントから DBMS にアクセスするための通信方法の組み合わせからなる概念をデータ・ソースといいます。「*データベース*」と「*データ・サーバ*」も参照してください。
- **意思決定支援アプリケーション** – アドホック・クエリ、レポート、計算などの処理が行え、少数データの更新トランザクションを特徴とするデータベース・クライアント・アプリケーションです。
- **宣言したデータ型** – Replication Agent から Replication Server に配信された値のデータ型です。
  - Replication Agent が `datetime` などの基本 Replication Server データ型を Replication Server に配信する場合、宣言したデータ型は基本データ型です。

  - 上記以外の場合、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に対する UDD でなければなりません。
- **デフォルトのファンクション文字列** – システム提供クラス `rs_sqlserver_function_class` と `rs_default_function_class`、およびこれらのクラスからファンクション文字列を直接的または間接的に継承するクラスに対してデフォルトで提供されるファンクション文字列です。*「ファンクション文字列」*参照。
- **マテリアライゼーション解除** – サブスクリプションが削除されたときに、適宜実行される処理です。これによって、他のサブスクリプションに使用されていない特定のローが、レプリケート・データベースから削除されます。
- **派生クラス** – 親クラスからファンクション文字列を継承するファンクション文字列クラスです。「*ファンクション文字列クラス*」と「*親クラス*」も参照してください。
- **直接ルート** – 中間 Replication Server を使用しないで、送信元 Replication Server から送信先 Replication Server へ直接メッセージを送信するために使用するルートです。「*間接ルート*」と「*ルート*」も参照してください。
- **ディスク・パーティション** –「*パーティション*」を参照してください。
- **分散データベース・システム** – データをネットワーク上にある複数のデータベースに格納するデータベース・システムです。これら複数のデータベースは、同じタイプのデータ・サーバ (たとえば、Adaptive Server) で管理される場合と、異機種のデータ・サーバで管理される場合があります。
- **ディストリビュータ** – インバウンド・キューにある各トランザクションの送信先を決定するために使用する Replication Server スレッド (DIST) です。
- **ダンプ・マーカ** – ダンプの実行時に Adaptive Server がデータベース・トランザクション・ログに書き込むメッセージです。ウォーム・スタンバイ・アプリケーションでは、アクティブ・データベースのデータでスタンバイ・データベースを初期化するときに、Replication Server がダンプ・マーカを使用して、トランザクション・ストリームのどこからトランザクションをスタンバイ・データベースに適用するかを決定するように指定できます。「*ウォーム・スタンバイ・アプリケーション*」も参照してください。
- **Embedded Replication Server システム・データベース (ERSSD)** – Replication Server システム・テーブルを格納する SQL Anywhere (SA) データベースです。Replication Server システム・テーブルを ERSSD と Adaptive Server RSSD のどちらに格納するかを選択できます。「*Replication Server システム・データベース (RSSD)*」も参照してください。
- **Enterprise Connect Data Access (ECDA)** – LAN ベースの ASE 以外のデータ・ソースやメインフレームのデータ・ソースなど、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。

- **ExpressConnect for Oracle** – Replication Server と Oracle データベース間の直接通信に使用できるライブラリのセットです。
- **エラー・アクション** – データ・サーバのエラーに対する Replication Server の応答処理です。Replication Server のエラー・アクションの種類としては、**ignore**、**warn**、**retry_log**、**log**、**retry_stop**、**stop_replication** があります。エラー・アクションは、特定のデータ・サーバ・エラーに割り当てられます。
- **エラー・クラス** – 指定したデータベースで使用するデータ・サーバのエラー・アクションの集まりに名前を付けたものです。
- **例外ログ** – データ・サーバ上で失敗したトランザクションの情報を保存する Replication Server の 3 つのシステム・テーブルがセットになったものです。例外ログに記録されたトランザクションは、ユーザまたはインテリジェント・アプリケーションが処理しなければなりません。例外ログに問い合わせるには、**rs_helpexception** ストアド・プロシージャを使用します。
- **フェールオーバ** – Sybase フェールオーバを使用すると、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、そのサーバのデバイス、データベース、コネクションをセカンダリ・コンパニオンが引き継ぐことができます。

  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。
- **フォールト・トレランス** – 1 つまたは複数のコンポーネントで障害が発生した場合でも、システムが正常に処理を継続できるシステムの機能です。
- **ファンクション** – insert、delete、select、begin transaction などのデータ・サーバのオペレーションを表す Replication Server オブジェクトです。Replication Server は、これらのオペレーションをファンクションとして他の Replication Server に配信します。各ファンクションは、ファンクション名とデータ・パラメータのセットから構成されます。ファンクションを送信先データベースで実行するために、Replication Server はファンクション文字列を使用して、そのファンクションを特定タイプのデータベース用のコマンドまたはコマンド・セットに変換します。「*ユーザ定義のファンクション*」と「*複写ファンクションの配信*」も参照してください。
- **ファンクション複写定義** – 複写ファンクションの配信で使用される、複写ファンクションの記述です。ファンクション複写定義は Replication Server によって管理され、複写されるパラメータと影響を受けるデータのプライマリ・バージョンがあるロケーションを示す情報などからなります。ファンクション複写定義には、適用ファンクション複写定義と要求ファンクション複写定義の 2 つのタイプがあります。「*複写ファンクションの配信*」も参照してください。
- **ファンクションのスコープ** – ファンクションの適用範囲です。ファンクションは、複写定義スコープまたはファンクション文字列クラス・スコープを持ちます。複写定義スコープを持つファンクションは、特定の複写定義に指定され、他の複写定義には適用できません。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は、ファンクション文字列クラスに対して 1 回だけ定義され、そのクラスでのみ使用できます。

- **ファンクション文字列** – Replication Server が、データベース・コマンドをデータ・サーバの API にマップするために使用する文字列です。文字列の最初の部分は入力テンプレートで、**rs_select** および **rs_select_with_lock** ファンクションのファンクション文字列を検索するためにのみ使用します。後半部分は出力テンプレートで、データベース・コマンドを対象のデータ・サーバ用にフォーマットするために使用します。
- **ファンクション文字列クラス** – 指定したデータベース・コネクションで使用される、名前付きのファンクション文字列のコレクションです。ファンクション文字列クラスには、Replication Server によって提供されるものとユーザが作成したものがあります。ファンクション文字列クラスは、ファンクション文字列の継承によってファンクション文字列定義を共有できます。システムで提供されるファンクション文字列クラスには、`rs_sqlserver_function_class`、`rs_default_function_class`、`rs_db2_function_class` の 3 つがあります。「*基本クラス*」、「*クラス・ツリー*」、「*派生クラス*」、「*ファンクション文字列の継承*」、「*親クラス*」も参照してください。
- **ファンクション文字列の継承** – クラス間でファンクション文字列定義を共有する機能です。この機能によって、派生クラスは親クラスからファンクション文字列を継承します。「*派生クラス*」、「*ファンクション文字列クラス*」、「*親クラス*」も参照してください。
- **ファンクション文字列変数** – 実行時に代入される値を表すために、ファンクション文字列内で使用する識別子です。ファンクション文字列内の変数は、疑問符 (?) で囲まれています。この変数は、カラムの値、ファンクションのパラメータ、システム定義の変数、ユーザ定義の変数を表します。
- **ファンクション・サブスクリプション** – ファンクション複写定義に対するサブスクリプションです (適用ファンクションおよび要求ファンクションの配信で使用されます)。
- **ゲートウェイ** – 異なるネットワーク・アーキテクチャを持つ複数のコンピュータ・システム間での通信を可能にする接続ソフトウェアです。
- **世代番号** – 「*データベース生成番号*」を参照してください。
- **異機種データ・サーバ** – 同じ分散データベース・システム内で使用される複数ベンダのデータ・サーバです。
- **ハイバネーション・モード** – Replication Server の状態です。この状態では、**admin** と **sysadmin** コマンドを除くすべての DDL コマンドは拒否され、すべてのルートとコネクション、および DSI、RSI などのほとんどのサービス・スレッドがサスペンドされます。また、RSI と RepAgent ユーザはログオフされログオンできません。ルートのアップグレード中に使用され、Replication Server が問題をデバッグするためにオン状態になることがあります。

- **高可用性 (HA)** – ダウン時間が非常に少ないことです。HA を提供するコンピュータ・システムは、通常、99.999% の可用性 (予定外のダウン時間が、年間約 5 分) を実現しています。
- **High Volume Adaptive Replication (HVAR)** – 最終的な結果とそれ以降のレプリケート・データベースへの最終的な結果のバルク適用を生成する、**insert**、**delete**、**update** の各オペレーションのグループのコンパイルです。
- **ホット・スタンバイ・アプリケーション** – クライアント・アプリケーションを中断したり、トランザクションを失ったりすることなく、スタンバイ・データベースをアクティブに切り替えられるデータベース・アプリケーションです。「*ウォーム・スタンバイ・アプリケーション*」も参照してください。
- **ID サーバ** – 複写システムのいずれか 1 つの Replication Server が、ID サーバとなります。ID サーバは、Replication Server の通常の作業の他に、複写システムにあるすべての Replication Server とデータベースにユニークな ID 番号を割り当て、複写システムのバージョン情報を管理します。
- **インバウンド・キュー** – Replication Agent から Replication Server へのメッセージをスプールするために使用されるステーブル・キューです。
- **間接ルート** – 送信元 Replication Server から送信先 Replication Server へ、1 つ以上の中間 Replication Server を経由してメッセージを送るために使用するルートです。「*直接ルート*」と「*ルート*」も参照してください。
- **interfaces ファイル** – Sybase クライアント／サーバ・アーキテクチャ上のサーバ・プログラムが使用する、ネットワークのアクセス情報を定義するエントリのあるファイルです。サーバ・プログラムには、Adaptive Server、ゲートウェイ、Replication Server、Replication Agent があります。クライアントとサーバは、interfaces ファイルにあるエントリを使用して、ネットワーク上で相互に接続できます。
- **遅延時間** – プライマリ・データベースで最初に適用されたデータ修正オペレーションが、レプリケート・データベースに分配されるまでに要する時間の単位です。この時間には、Replication Agent での処理時間、Replication Server での処理時間、ネットワークのオーバヘッドなどが含まれます。
- **ローカル・エリア・ネットワーク (LAN)** – コンピュータとプリンタや端末などのデバイスを、データやデバイスの共有のためにケーブルで接続したシステムです。
- **ロケータ値** – Replication Server の RSSD の `rs_locater` テーブルに格納されている値です。この値によって、複写中に Replication Server によって受信および確認された、直前の各サイトからの最新のログ・トランザクション・レコードが特定されます。
- **論理コネクション** – Replication Server が、ウォーム・スタンバイ・アプリケーションのアクティブ・データベースとスタンバイ・データベースとのコネクションにマップするデータベース・コネクションです。「*コネクション*」と「*ウォーム・スタンバイ・アプリケーション*」も参照してください。

- **ログイン名** – ユーザまたは Replication Server などのシステム・コンポーネントがデータ・サーバ、Replication Server、または Replication Agent にログインするために使用する名前です。
- **ログ転送言語 (LTL)** – 複写コマンド言語 (RCL) のサブセットです。プライマリ・データベースのトランザクション・ログから取得した情報は、RepAgent などの Replication Agent によって、LTL コマンドを使用して、Replication Server に送信されます。
- **Log Transfer Manager (LTM)** – Sybase SQL Server 用の Replication Agent プログラムです。「*Replication Agent*」と「*RepAgent スレッド*」も参照してください。
- **メンテナンス・ユーザ** – Replication Server がレプリケート・データを管理するために使用するデータ・サーバのログイン名です。ほとんどのアプリケーションでは、メンテナンス・ユーザのトランザクションは複写されません。
- **マテリアライゼーション** – 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、サブスクリプションによって指定されたデータをコピーする処理です。これによって、レプリケート・テーブルが初期化されます。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。「*アトミック・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*」、「*非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*」も参照してください。
- **マテリアライゼーション・キュー** – 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されているサブスクリプションに関連したメッセージをスプールするために使用されるステーブル・キューです。
- **消失ロー** – プライマリ・テーブルには存在するが、そのテーブルの複写コピーには存在しないローです。
- **混合バージョン・システム** – ソフトウェア・バージョンとサイト・バージョンの違いによって異なる機能を持った、ソフトウェア・バージョンの異なる Replication Server がある複写システムです。混合バージョン・サポートは、システム・バージョンが 11.0.2 以降の場合のみ使用できます。

  たとえば、Replication Server バージョン 11.5 以降とバージョン 11.0.2 がある複写システムは、混合バージョン・システムです。バージョン 11.0.2 以降の Replication Server では、特定の新機能の使用がシステム・バージョンによって制限されていますが、それより前のバージョンではこの機能がサポートされていないため、バージョン 11.0.2 より前の Replication Server を持った複写システムは、混合バージョン・システムではありません。「*サイト・バージョン*」と「*システム・バージョン*」も参照してください。
- **カラム数の増加** – 複写定義には 251 ～ 1,024 のカラムを含めることができます。カラム数の増加は、Replication Server バージョン 12.5 以降でサポートされています。
- **Multi-Site Availability (MSA)** – テーブル、ファンクション、トランザクション、システム・ストアド・プロシージャ、DDL などのデータベース・オブジェク

トを、プライマリ・データベースからレプリケート・データベースへ複写する方法です。「*データベース複写定義*」も参照してください。

- **マルチパス・レプリケーション** – 送信元データベースからターゲット・データベースへのデータの並列パスを有効にすることによってパフォーマンスを向上させる Replication Server の機能。マルチパス・レプリケーションは、ウォーム・スタンバイ環境と Multi-Site Availability (MSA) 環境で設定できます。これらの複数のパスではデータが個別に処理され、それらのパス間のトランザクションの一貫性を必要とせずにデータ・セットを並列処理できる場合に適用されます。パス内でのデータ整合性を維持しますが、さまざまなパス間でのコミット順には従いません。
- **ネーム・スペース** – オブジェクト名がユニークでなければならない範囲 (スコープ) です。
- **ノンアトミック・マテリアライゼーション** – マテリアライゼーションのメソッドの 1 つです。これは、holdlock を使わずに 1 つのオペレーションで、ネットワークを介してプライマリ・データベースからレプリケート・データベースへサブスクリプション・データをコピーします。データの転送中もプライマリ・テーブルを変更できるので、レプリケート・データベースとプライマリ・データベース間で一時的に不一致が生じる可能性があります。データは、レプリケート・データベースのトランザクション・ログが満杯にならないように、トランザクションごとに 10 ローずつ挿入する方法を使用して適用されます。ノンアトミック・マテリアライゼーションは、**create subscription** コマンドのオプションのメソッドです。「*オートコレクション*」、「*アトミック・マテリアライゼーション*」、「*非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*」も参照してください。
- **ネットワークベース・セキュリティ** – ネットワーク上のデータを安全に転送することです。Replication Server は、ユーザの認証、統一化ログイン、Replication Server 間の安全なメッセージ転送などのサード・パーティのセキュリティ・メカニズムをサポートします。
- **非マテリアライゼーション** – マテリアライゼーションのメソッドの 1 つです。サブスクリプション・データがレプリケート・サイトにすでに存在する場合、サブスクリプションを作成できます。**without materialization** 句を指定して **create subscription** コマンドを使用してください。このメソッドを使用して、テーブル複写定義とファンクション複写定義へのサブスクリプションを作成できます。「*アトミック・マテリアライゼーション*」と「*バルク・マテリアライゼーション*」も参照してください。
- **オンライン・トランザクション処理 (OLTP) アプリケーション** – データ修正 (挿入、削除、更新) を伴うさまざまなトランザクションを頻繁に実行するデータベース・クライアント・アプリケーションです。
- **オリジン・キュー ID (qid)** – `qid` は、RepAgent によって形成され、Replication Server に渡された各ログ・レコードをユニークに識別します。`date`、

timestamp、およびデータベース世代番号が含まれます。「*データベース生成番号*」も参照してください。

- **孤立したロー** – テーブルの複写コピーにあるローの中で、アクティブなサブスクリプションと一致しないローのことをいいます。
- **アウトバウンド・キュー** – メッセージをスプールするのに使用するステーブル・キューです。DSI アウトバウンド・キューは、レプリケート・データベースへのメッセージを、RSI アウトバウンド・キューは、レプリケート Replication Server へのメッセージをスプールします。
- **並列 DSI** – 単一の DSI スレッドではなく、並列で機能する複数の DSI スレッドを使用して、レプリケート・データ・サーバに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する方法です。「*コネクション*」と「*データ・サーバ・インタフェース (DSI)*」も参照してください。
- **パラメータ** – プロシージャの実行時に提供される値を表す識別子です。ファンクション文字列で使用するパラメータ名は @ 記号で始まります。プロシージャをファンクション文字列から呼び出すと、Replication Server はパラメータ値をそのまま変更しないでデータ・サーバへ渡します。「*サーチャブル・パラメータ*」も参照してください。
- **親クラス** – 派生クラスがファンクション文字列を継承する、ファンクション文字列クラスです。「*ファンクション文字列クラス*」と「*派生クラス*」も参照してください。
- **パーティション** – Replication Server が、ステーブル・キューを格納するために使用するロー・ディスク・パーティションまたはオペレーティング・システムのファイルです(オペレーティング・システムのファイルはテスト環境でのみ使用してください)。
- **物理コネクション** – 「*コネクション*」を参照してください。
- **プライマリ・データ** – 複写システムの中で、複写の対象となるデータ・セットのことです。プライマリ・データは、データ・サーバによって管理されます。このデータ・サーバは、データのサブスクリプションがあるすべての Replication Server で認識されています。
- **プライマリ・データベース** – 複写システムによって別のデータベースに複写されるデータが格納されたデータベースです。
- **プライマリ・フラグメント** – 一連のローのプライマリ・バージョンを保持するテーブルの水平方向セグメントです。
- **プライマリ・キー** – 各ローをユニークに識別するテーブル・カラムのセットです。
- **プライマリ・サイト** – ファンクション文字列クラスまたはエラー・クラスが定義されている Replication Server です。「*エラー・クラス*」と「*ファンクション文字列クラス*」を参照してください。

- **プリンシパル・ユーザ** – アプリケーションを開始するユーザです。ネットワークベース・セキュリティを使用する場合、Replication Server はプリンシパル・ユーザとしてリモート・サーバにログインします。
- **プロファイル** – プロファイルを使用すると、事前に定義されたプロパティのセットでコネクションを設定できます。
- **射影** – テーブルの垂直方向のスライスです。テーブル・カラムのサブセットを表します。
- **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グループです。パブリケーションを使用すると、関連するテーブルかストアド・プロシージャまたはその両方の複写定義を収集して、グループとしてそれらのサブスクリプションを作成できます。送信元 Replication Server のパブリケーション内で、アーティクルとして複写定義を収集し、送信先 Replication Server でパブリケーション・サブスクリプションを使用してそれらにサブスクリプションを作成できます。「*アーティクル*」と「*パブリケーション・サブスクリプション*」も参照してください。
- **パブリケーション・サブスクリプション** – パブリケーションへのサブスクリプションです。「*アーティクル*」と「*パブリケーション*」も参照してください。
- **パブリッシュ・データ型** – レプリケート・データ・サーバにおけるカラム・レベル変換後 (続いてクラス・レベル変換をする場合はその前) のカラムのデータ型です。パブリッシュ・データ型は、Replication Server 基本データ型か、ターゲット・データ・サーバのデータ型に対する UDD のどちらかでなければなりません。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。
- **クエリ** – データベース管理システムで、指定した基準を満たすデータを取得するための要求です。SQL データベース言語では、クエリを指定するときに **select** コマンドを使用します。
- **クワイス状態** – すべての更新がその送信先に送信された状態の複写システムのことをクワイス状態の複写システム (静止している複写システム) といいます。Replication Server コマンドやプロシージャの中には、最初に複写システムをクワイスすることを必要とするものがあります。
- **引用符付き識別子** – スペースや非英数字などの特殊文字が含まれる、アルファベット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は、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。
- **Real-Time Loading (RTL) –** Sybase IQ データベースへの High-Volume Adaptive Replication (HVAR)。HVAR の変更を Sybase IQ レプリケート・データベースに適用するには、関連するコマンドとプロセスを使用します。「*High Volume Adaptive Replication*」を参照してください。
- **リモート・プロシージャ・コール (RPC)** – リモート・サーバに常駐しているプロシージャを実行するための要求です。プロシージャを実行するサーバには、Adaptive Server、Replication Server、または Open Server を使用して構築されたサーバなどがあります。プロシージャの実行要求は、これらのサーバやクライ

アント・アプリケーションから発行できます。RPC 要求のフォーマットは、Sybase Client/Server Interfaces の一部です。

- **RepAgent スレッド –** Adaptive Server データベース用の Replication Agent です。RepAgent は Adaptive Server のスレッドです。プライマリ・データベースから Replication Server にトランザクション・ログ情報を転送して、他のデータベースに分配します。
- **レプリケート・データベース –** 複写システムによって別のデータベースから複写されたデータが格納されたデータベースです。
- **複写ファンクションの配信 –** ファンクション複写定義に対応するストアド・プロシージャを送信元データベースから送信先データベースに複写する方法です。「*適用ファンクション*」、「*要求ファンクション*」、「*ファンクション複写定義*」も参照してください。
- **複写ストアド・プロシージャ – sp_setrepproc** または **sp_setreplicate** システム・プロシージャを使用して、複写するようにマーク付けされた Adaptive Server ストアド・プロシージャです。複写ストアド・プロシージャは、ファンクション複写定義またはテーブル複写定義に関連付けることができます。「*複写ファンクションの配信*」と「*非同期プロシージャ配信*」も参照してください。
- **複写テーブル –** 複数サイトのデータベースで、Replication Server が一部または全部を管理するテーブルです。これらのテーブルのうち、システム・プロシージャの **sp_setreptable** または **sp_setreplicate** を使用して複写するようにマーク付けされた 1 つのバージョンがプライマリ・バージョンで、それ以外のすべてのバージョンは複写コピーです。
- **Replication Agent –** プライマリ・データへの修正を表すトランザクション・ログ情報を、他のデータベースに分配するために、データベース・サーバから Replication Server に転送するプログラムまたはモジュールです。RepAgent は、Adaptive Server データベース用の Replication Agent です。
- **複写コマンド言語 (RCL) –** Replication Server の情報を管理するために使用するコマンドです。
- **複写定義 –** サブスクリプションを作成するためのテーブルの定義です。複写定義は Replication Server によって管理され、この中で複写されるカラムとテーブルのプライマリ・バージョンがあるロケーションが指定されています。

  ファンクションの複写定義も作成できます。複写定義がテーブルに関するものかファンクションに関するものかを区別するために、「テーブル複写定義」という用語を使用することもあります。「*ファンクション複写定義*」も参照してください。
- **Replication Server –** Sybase のサーバ・プログラムです。通常、LAN 上で複写データを管理し、同じ LAN または WAN 上にある別の Replication Server から受け取ったデータのトランザクションを処理します。
- **Replication Server インタフェース (RSI) –** 送信先 Replication Server へログインし、RSI アウトバウンド・ステーブル・キューから送信先 Replication Server へコマンドを転送するスレッドです。プライマリまたは中間 Replication Server か

らコマンドを受け取る送信先 Replication Server ごとに、1 つの RSI スレッドが存在します。「*アウトバウンド・キュー*」と「*ルート*」も参照してください。

- **Replication Monitoring Services (RMS) –** Sybase Unified Agent Framework (UAF) を使用して構築された小さな Java アプリケーションで、複写環境のモニタとトラブルシューティングを行います。
- **複写システム管理者 –** Replication Server の定型作業を管理するシステム管理者です。
- **Replication Server システム・データベース (RSSD) –** Replication Server のシステム・テーブルを格納する Adaptive Server データベースです。Replication Server システム・テーブルを、RSSD と SQL Anywhere (SA) ERSSD のどちらに格納するかを選択できます。「*Embedded Replication Server システム・データベース (ERSSD)*」も参照してください。
- **Replication Server システム Adaptive Server –** Replication Server のシステム・テーブル (RSSD) を格納するデータベースがある Adaptive Server です。
- **複写システム –** 複数のデータベースにデータを複写することで、リモート・ユーザがそれぞれのローカル・データにアクセスできるようにするデータ処理システムです。複写システムは Replication Server を基にして構成され、Replication Agent やデータ・サーバのような他のコンポーネントも含まれています。
- **複写システム・ドメイン –** 同じ ID サーバを使用する複写システムのすべてのコンポーネントです。
- **要求ファンクション –** ファンクション複写定義に対応する複写ファンクションであり、Replication Server によってプライマリ・データベースからレプリケート・データベースに配信されます。要求ファンクションがストアド・プロシージャにパラメータ値を渡し、そのストアド・プロシージャがレプリケート・データベースで実行されます。ストアド・プロシージャは、プライマリ・サイトと同じユーザによってレプリケート・サイトで実行されます。「*複写ファンクションの配信*」、「*要求ファンクション*」、「*ファンクション複写定義*」も参照してください。
- **resync marker –** Replication Agent を再同期モードで再開すると、Replication Agent は、再同期処理が進行中であることを示すデータベース再同期マーカを Replication Server に送信します。Replication Agent は最初のメッセージとして再同期マーカを送信してから、SQL データ定義言語 (DDL: data definition language) またはデータ操作言語 (DML: data manipulation language) のトランザクションを送信します。
- **ルート –** 送信元 Replication Server から送信先 Replication Server への一方向のメッセージ・ストリームです。ルートは、データ修正コマンド (RSSD に対するものを含む) と複写ファンクションまたはストアド・プロシージャを Replication Server 間でやりとりします。「*直接ルート*」と「*間接ルート*」も参照してください。

- **ルート・バージョン** – ルートの送信元と送信先の Replication Server のサイト・バージョン番号のうち、低い方の番号です。Replication Server バージョン 11.5 以降では、レプリケート・サイトに送信するデータを決定するのにルート・バージョン番号を使用します。「*サイト・バージョン*」も参照してください。
- **ロー・マイグレーション** – テーブルのプライマリ・バージョン内のローでカラム値が変更されたとき、テーブルのレプリケート・バージョン内の対応するローも、サブスクリプションの **where** 句内の値の比較に基づいて挿入または削除されるプロセスです。
- **SQL Server** – 11.5 より前の Sybase リレーショナル・データベース・サーバです。
- **SQL 文の複写** – SQL 文の複写では、Replication Server は、個々のローの変更ではなく、プライマリ・データを変更した SQL 文をトランザクション・ログから受け取ります。Replication Server は、SQL 文をレプリケート・サイトに適用します。RepAgent は、SQL データ操作言語 (DML) と個々のローの変更の両方を送信します。設定に応じて、Replication Server が、個々のローの変更によるログの複写または SQL 文の複写のどちらかを選択します。
- **スキーマ** – データベースの構造体です。DDL コマンドとシステム・プロシージャは、データベースに格納されているシステム・テーブルを変更します。Replication Server バージョン 11.5 以降と Adaptive Server バージョン 11.5 以降を使用している場合には、サポートされている DDL コマンドとシステム・プロシージャは、スタンバイ・データベースに複写できます。
- **サーチャブル・カラム** – 複写するローをサイトで制限するために、サブスクリプションまたはアーティクルの **where** 句で指定できる複写テーブル内のカラムです。
- **サーチャブル・パラメータ** – サブスクリプションの **where** 句で指定できる複写ストアド・プロシージャのパラメータです。このパラメータを使用して、ストアド・プロシージャを複写するかどうかを決定します。「*パラメータ*」も参照してください。
- **セカンダリ・トランケーション・ポイント** – 「*トランケーション・ポイント*」を参照してください。
- **サイト** – 最低でも Replication Server、データ・サーバ、データベースで構成され、場合によっては Replication Agent も含まれるインストレーション環境で、通常は地理的に離れた場所にあります。各サイトのコンポーネントは、WAN を介して複写システムにある他のサイトのコンポーネントに接続されます。「*プライマリ・サイト*」も参照してください。
- **サイト・バージョン** – 個々の Replication Server のバージョン番号です。サイト・バージョンが一度あるレベルに設定されると、Replication Server でそのレベル特有の機能が有効になり、レベルをダウングレードすることはできません。「*ソフトウェア・バージョン*」、「*ルート・バージョン*」、「*システム・バージョン*」も参照してください。

- **ソフトウェア・バージョン** – 個々の Replication Server のソフトウェア・リリースのバージョン番号です。「*サイト・バージョン*」と「*システム・バージョン*」も参照してください。
- **ステーブル・キュー・マネージャ (SQM)** – ステーブル・キューを管理するスレッドです。インバウンド・キュー、アウトバウンド・キューのいずれの場合でも、Replication Server がアクセスするステーブル・キューに対して、それぞれ 1 つのステーブル・キュー・マネージャ (SQM) スレッドがあります。
- **ステーブル・キュー・トランザクション (SQT) インタフェース** – コミット順にトランザクション・コマンドを再構築するスレッドです。ステーブル・キュー・トランザクション (SQT) インタフェース・スレッドは、インバウンド・ステーブル・キューを読み取って、トランザクションをコミット順に配列し、それらをディストリビュータ (DIST) スレッドと DSI スレッドのうち、SQT によるトランザクションの並び替えを要求した方に送信します。
- **ステーブル・キュー** – Replication Server が、ルートまたはデータベース・コネクション用のメッセージを格納するための蓄積転送キューです。ステーブル・キューに書き込まれたメッセージは、送信先の Replication Server またはデータベースに配信されるまで、このキューに格納されます。Replication Server は、割り当てられたディスク・パーティションを使用してステーブル・キューを構築します。「*インバウンド・キュー*」、「*アウトバウンド・キュー*」、「*マテリアライゼーション・キュー*」も参照してください。
- **スタンドアロン・モード** – リカバリ処理を開始するために使用する Replication Server の特別な動作モードです。
- **スタンバイ・データベース** – ウォーム・スタンバイ・アプリケーションでは、アクティブ・データベースからデータ変更を受信し、そのデータベースのバックアップとして機能するデータベースのことです。「*ウォーム・スタンバイ・アプリケーション*」も参照してください。
- **ストアド・プロシージャ** – Adaptive Server データベースに名前付きで格納されている SQL 文とオプションのフロー制御文の集まりです。Adaptive Server が提供するストアド・プロシージャは、システム・プロシージャと呼ばれます。Replication Server ソフトウェアには、RSSD に問い合わせるストアド・プロシージャがいくつか組み込まれています。
- **サブスクリプション** – 指定したサイトのレプリケート・データベースにあるテーブルの複写コピー、またはテーブルからのローのセットを管理するために、Replication Server に対して行う要求のことです。適用ファンクション配信を行うために、ファンクション複写定義のサブスクリプションも作成できます。
- **サブスクリプション・マテリアライゼーション解除** – 「*マテリアライゼーション解除*」を参照してください。
- **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*」を参照してください。

- **サブスクリプション・マイグレーション** – 「*ロー・マイグレーション*」を参照してください。
- **Sybase Central** – Sybase および Powersoft 製品を管理する共通のインタフェースを提供するグラフィカルなツールです。Replication Server は、Sybase Central プラグインとして Replication Manager を使用します。「*Replication Monitoring Services (RMS)*」も参照してください。
- **対称型マルチプロセッシング (SMP)** – マルチプロセッサ・プラットフォームで、アプリケーションのスレッドを並列に実行できる機能です。Replication Server は、サーバのパフォーマンスと効率が高められる SMP をサポートしています。
- **同期コマンド** – クライアントが完了ステータスを受信後、初めて完了したとみなすコマンドです。
- **システム・ファンクション** – あらかじめ定義され、Replication Server 製品に組み込まれているファンクションです。**rs_begin** などの複写アクティビティを調整するシステム・ファンクション、または **rs_insert**、**rs_delete**、**rs_update** などのデータ操作のオペレーションを実行するシステム・ファンクションがあります。
- **システム提供クラス** – Replication Server が提供するエラー・クラス `rs_sqlserver_error_class` とファンクション文字列クラス `rs_sqlserver_function_class`、`rs_default_function_class`、`rs_db2_function_class` のことです。ファンクション文字列は、システムで提供されるファンクション文字列クラスとこれらのクラスから直接的または間接的に継承する派生クラス用に自動的に生成されます。「*エラー・クラス*」と「*ファンクション文字列クラス*」も参照してください。
- **システム・バージョン** – リリース 11.0.2 以前の Replication Server に対して、新しい機能が有効なバージョンを表す複写システムのバージョン番号です。このバージョン番号より低いバージョンには、Replication Server をダウングレードまたはインストールできません。Replication Server バージョン 11.5 では、特定の新機能を使用するために、サイト・バージョン 1150 と最低でもシステム・バージョン 1102 が必要です。「*混合バージョン・システム*」、「*サイト・バージョン*」、「*ソフトウェア・バージョン*」も参照してください。
- **テーブル複写定義** – 「*複写定義*」を参照してください。
- **テーブル・サブスクリプション** – テーブル複写定義に対応するサブスクリプションです。
- **スレッド** – Replication Server 内で実行されるプロセスです。Sybase Open Server で構築された Replication Server は、マルチスレッド・アーキテクチャに基づいています。各スレッドは、ユーザ・セッションを管理したり、Replication Agent または別の Replication Server からメッセージを受信したり、メッセージをデータベースに適用したりする特定のファンクションを実行します。「デー

タ・サーバ・インタフェース (DSI)」、「*ディストリビュータ*」、「*Replication Server インタフェース (RSI)*」も参照してください。

- **トランザクション** – 文をグループ化するためのメカニズムです。このメカニズムによって、文はグループ内の単なる構成単位として扱われ、グループ内のすべての文が実行されるか、グループ内の文がまったく実行されないことになります。
- **Transact-SQL** – Adaptive Server で使用するリレーショナル・データベース言語です。これは、標準 SQL (Structured Query Language) をベースとして、Sybase の拡張機能を付加したものです。
- **トランケーション・ポイント** – プライマリ・データを保存している Adaptive Server データベースには、トランザクション・ログ内で Adaptive Server がどこまで処理を完了したかを示すアクティブなトランケーション・ポイントがあります。これをプライマリ・トランケーション・ポイントといいます。

  Adaptive Server データベースの RepAgent が管理するのが、セカンダリ・トランケーション・ポイントで、Replication Server に正常に送信されたログの部分と、まだ送信されていない部分とを分けるために、トランザクション・ログ内の場所にマークを付けます。セカンダリ・トランケーション・ポイントにより、ログの一部がトランケートされる前に、各オペレーションが必ず複写システムへ入力されることが可能になります。
- **ユーザ定義ファンクション** – このファンクションを使用すると、Replication Server を使用して、複写システムのサイト間で複写ファンクションまたは非同期ストアド・プロシージャを配信するカスタム・アプリケーションを作成できます。複写ファンクションの配信では、ファンクション複写定義を作成すると、Replication Server によって自動的にユーザ定義ファンクションが作成されます。
- **変数** – 「*ファンクション文字列変数*」を参照してください。
- **バージョン** – *混合バージョン・システム*

  「*混合バージョン・システム*」、「*サイト・バージョン*」、「*ソフトウェア・バージョン*」、「*システム・バージョン*」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** – Replication Server を使用して、アクティブ・データベースと呼ばれるデータベースに対するスタンバイ・データベースを管理するアプリケーションです。アクティブ・データベースで障害が発生した場合、Replication Server とクライアント・アプリケーションはデータベースをスタンバイ・データベースに切り替えられます。
- **広域ネットワーク (WAN)** – データ通信回線で接続されているローカル・エリア・ネットワーク (LAN) のシステムです。
- **ワイド・カラム** – `char`、`varchar`、`binary`、`varbinary`、`unichar`、`univarchar`、または Java `inrow` データで構成されている、255 バイトより大

きい複写定義のカラムです。ワイド・カラムは、Replication Server バージョン 12.5 以降でサポートされています。

- **ワイド・データ** – データ・サーバのデータ・ページのサイズを上限とする、幅の広いデータ・ローです。Adaptive Server は、2K、4K、8K、16K のページ・サイズをサポートしています。ワイド・データは、Replication Server バージョン 12.5 以降でサポートされています。
- **ワイド・メッセージ** – 複数のブロックにまたがる 16K より大きいメッセージです。ワイド・メッセージは、Replication Server バージョン 12.5 以降でサポートされています。

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

A

B

C

D

E

I

J

L

M

N

O

P

Q

R

S

T

U

Z

あ

い

う

え

お

か

き

く

け

こ

さ

し

す

せ

そ

た

ち

と

な

の

は

ひ

ふ

ま

め

も

ゆ

ら

り

れ

ろ