iAnywhere SOLUTIONS™
A SYBASE COMPANY

SQL Remote®
ユーザーズ・ガイド

パート番号：DC03955-01-0902-01
改訂：2005 年 3 月

版権

Copyright © 2005 iAnywhere Solutions, Inc., Sybase, Inc. All rights reserved.

ここに記載されている内容を iAnywhere Solutions, Inc.、Sybase, Inc. またはその関連会社の書面による事前許可を得ずに電子的、機械的、手作業、光学的、またはその他のいかなる手段によっても複製、転載、翻訳することを禁じます。

Sybase、SYBASE のロゴ、Adaptive Server、AnswerBase、Anywhere、EIP、Embedded SQL、Enterprise Connect、Enterprise Portal、GainMomentum、iAnywhere、jConnect MASS DEPLOYMENT、Netimpact、ObjectConnect、ObjectCycle、OmniConnect、Open ClientConnect、Open ServerConnect、PowerBuilder、PowerDynamo、Powersoft、Quickstart Datamart、Replication Agent、Replication Driver、SQL Anywhere、SQL Central、SQL Remote、Support Plus、SWAT、Sybase IQ、Sybase System 11、Sybase WAREHOUSE、SyBooks、XA-Library は米国法人 Sybase, Inc. の登録商標です。Backup Server、Client-Library、jConnect for JDBC、MainframeConnect、Net-Gateway、Net-Library、Open Client、Open Client/Server、S-Designor、SQL Advantage、SQL Debug、SQL Server、SQL Server Manager、Sybase Central、Watcom、Web.SQL、XP Server は米国法人 Sybase, Inc. の商標です。

Certicom、MobileTrust および、SSL Plus は Certicom Corp. の商標です。Security Builder は Certicom Corp. の登録商標です。

ここに記載されている上記以外の社名および製品名は、各社の商標または登録商標の場合があります。

目次

はじめに ix
SQL Anywhere Studio のマニュアル x
表記の規則 xiv
Adaptive Server Anywhere サンプル・データベース xvii
詳細情報の検索／フィードバックの提供 xviii

1 SQL Remote へようこそ 3
SQL Remote について 4
はじめに 5

2 SQL Remote の概念 7
SQL Remote のコンポーネント 8
パブリケーションとサブスクリプション 12
SQL Remote の機能 15
インストールの例 17

3 SQL Remote の設定 21
設定の概要 22
Adaptive Server Enterprise サーバの準備 24
SQL Remote for Adaptive Server Enterprise のアップグレード 29
SQL Remote のアンインストール 30

4 Adaptive Server Anywhere ユーザのためのチュートリアル 33
概要 34
チュートリアル : Sybase Central を使用した Adaptive Server Anywhere レプリケーション 38
チュートリアル : Interactive SQL と dbxtract を使用した Adaptive Server Anywhere レプリケーション 50
データのレプリケーションの開始 59

サンプル・パブリケーション 65

5 Adaptive Server Enterprise ユーザのためのチュートリアル 67
概要 68
チュートリアル :Adaptive Server Enterprise のレプリケーション 71
データのレプリケーションの開始 83

6 SQL Remote インストール環境の設計 91
設計の概要 92
文のレプリケーション方法 97
データ型がレプリケートされる方法 103
どのサブスクライバにどの文が送信されるか 107
レプリケーション・エラーと競合 109

7 Adaptive Server Anywhere の SQL Remote 設計 113
設計の概要 114
データのパブリッシュ 115
パブリケーションの設計 (Adaptive Server Anywhere)....128
サブスクリプション式を含まないテーブルの分割....132
複数のサブスクリプション間におけるローの共有....141
競合の管理 151
ユニークなプライマリ・キーの確保 162
サブスクリプションの作成....175

8 Adaptive Server Enterprise の SQL Remote 設計.... 179
設計の概要 180
パブリケーションの作成 181
Adaptive Server Enterprise のパブリケーション設計 187
サブスクリプション・カラムを含まないテーブルの分割 190
複数のサブスクリプション間におけるローの共有....200
競合の管理 210
ユニークなプライマリ・キーの確保 222
サブスクリプションの作成....229

9 データベースの配備と同期 233
配備の概要 234
配備前のテスト....235

データベースの同期 238
抽出ユーティリティの使用 241
メッセージ・システムを介したデータの同期 251

10 SQL Remote の管理 253
管理の概要 254
SQL Remote パーミッションの管理 256
メッセージ・タイプの使用 269
Message Agent の実行 287
Message Agent のパフォーマンスのチューニング 294
メッセージのエンコードと圧縮 303
メッセージ・トラッキング・システム 306

11 SQL Remote for Adaptive Server Anywhere の管理 311
Message Agent の実行 312
エラーのレポートと処理 315
トランザクション・ログとバックアップの管理 320
パススルー・モードの使用 335

12 SQL Remote for Adaptive Server Enterprise の管理 339
Adaptive Server Enterprise の Message Agent の動作 340
Message Agent の実行 346
エラーのレポートと処理 349
Adaptive Server Enterprise のトランザクション・ログとバックアップの管理 351
スキーマの変更 355
パススルー・モードの使用 356

13 Replication Server における SQL Remote の使用 359
SQL Remote Open Server を使用する必要がある場合 360
Replication Server/SQL Remote インストール環境のアーキテクチャ 361
SQL Remote Open Server の設定 365
Replication Server の設定 369
その他の問題 372

14 ユーティリティとオプションのリファレンス 375
Message Agent 376
データベース抽出ユーティリティ 389

SQL Remote Open Server 402
SQL Remote のオプション 406
SQL Remote イベント・フック・プロシージャ 413

15 Adaptive Server Anywhere のシステム・オブジェクト 419
SQL Remote システム・テーブル 420
SQL Remote システム・ビュー 427

16 Adaptive Server Enterprise のシステム・オブジェクト 431
SQL Remote システム・テーブル 432
SQL Remote システム・ビュー 442
ステーブル・キュー・テーブル 447

17 Adaptive Server Anywhere コマンド・リファレンス 451
ALTER REMOTE MESSAGE TYPE 文 452
CREATE PUBLICATION 文 453
CREATE REMOTE MESSAGE TYPE 文 454
CREATE SUBSCRIPTION 文 455
CREATE TRIGGER 文 457
DROP PUBLICATION 文 459
DROP REMOTE MESSAGE TYPE 文 460
DROP SUBSCRIPTION 文 461
GRANT CONSOLIDATE 文 462
GRANT PUBLISH 文 463
GRANT REMOTE 文 464
GRANT REMOTE DBA 文 466
PASSTHROUGH 文 467
REMOTE RESET 文 468
REVOKE CONSOLIDATE 文 469
REVOKE PUBLISH 文 470
REVOKE REMOTE 文 471
REVOKE REMOTE DBA 文 472
SET REMOTE OPTION 文 473
START SUBSCRIPTION 文 474
STOP SUBSCRIPTION 文 475
SYNCHRONIZE SUBSCRIPTION 文 476
UPDATE 文 477

18 Adaptive Server Enterprise コマンド・リファレンス 479
sp_add_article プロシージャ 480
sp_add_article_col プロシージャ 482
sp_add_remote_table プロシージャ 484
sp_create_publication プロシージャ 486
sp_drop_publication プロシージャ 487
sp_drop_remote_type プロシージャ 488
sp_drop_sql_remote プロシージャ 489
sp_grant_consolidate プロシージャ 490
sp_grant_remote プロシージャ 493
sp_link_option プロシージャ 496
sp_modify_article プロシージャ 498
sp_modify_remote_table プロシージャ 500
sp_passthrough プロシージャ 502
sp_passthrough_piece プロシージャ 504
sp_passthrough_stop プロシージャ 506
sp_passthrough_subscription プロシージャ 507
sp_passthrough_user プロシージャ 509
sp_populate_sql_anywhere プロシージャ 510
sp_publisher プロシージャ 511
sp_queue_clean プロシージャ 512
sp_queue_confirmed_delete_old プロシージャ 513
sp_queue_confirmed_transaction プロシージャ 514
sp_queue_delete_old プロシージャ 515
sp_queue_drop プロシージャ 516
sp_queue_dump_database プロシージャ 517
sp_queue_dump_transaction プロシージャ 518
sp_queue_get_state プロシージャ 519
sp_queue_log_transfer_reset プロシージャ 520
sp_queue_read プロシージャ 521
sp_queue_reset プロシージャ 522
sp_queue_set_confirm プロシージャ 523
sp_queue_set_progress プロシージャ 524
sp_queue_transaction プロシージャ 525
sp_remote プロシージャ 526
sp_remote_option プロシージャ 528
sp_remote_type プロシージャ 530
sp_remove_article プロシージャ 532

sp_remove_article_col プロシージャ 533
sp_remove_remote_table プロシージャ 534
sp_revoke_consolidate プロシージャ 535
sp_revoke_remote プロシージャ 536
sp_subscription プロシージャ 537
sp_subscription_reset プロシージャ 539

A SQL Remote for Adaptive Server Enterprise と SQL Remote for Adaptive Server Anywhere の違い 543
差異のタイプ 544
機能性の違い 545
アプローチの違い 547
Enterprise 間のレプリケーションの制限 550

B サポートされるプラットフォームとメッセージ・リンク 553
サポートされるメッセージ・システム 554
サポートされるオペレーティング・システム 555

索引 557

はじめに

このマニュアルの内容

このマニュアルでは、モバイル・コンピューティング用の SQL Remote データ・レプリケーション・システムについて、あらゆる角度から説明します。このシステムによって、Adaptive Server Anywhere または Adaptive Server Enterprise の単一データベースと Adaptive Server Anywhere の複数データベースの間で、電子メールやファイル転送などの間接的リンクを使用したデータ共有が可能になります。

対象読者

このマニュアルは、使用している情報システムに SQL Remote レプリケーションを追加したいと考えている Adaptive Server Anywhere ユーザと Adaptive Server Enterprise ユーザを対象としています。

始める前に

SQL Remote と他のレプリケーション・テクノロジとの比較については、『SQL Anywhere Studio の紹介』> 「レプリケーション・テクノロジの概要」を参照してください。

SQL Anywhere Studio のマニュアル

このマニュアルは、SQL Anywhere のマニュアル・セットの一部です。この項では、マニュアル・セットに含まれる各マニュアルと使用法について説明します。

SQL Anywhere Studio のマニュアル

SQL Anywhere Studio のマニュアルは、各マニュアルを 1 つの大きなヘルプ・ファイルにまとめたオンライン形式、マニュアル別の PDF ファイル、および有料の製本版マニュアルで提供されます。SQL Anywhere Studio のマニュアルは、次の分冊マニュアルで構成されています。

- **『SQL Anywhere Studio の紹介』** このマニュアルでは、SQL Anywhere Studio のデータベース管理と同期テクノロジの概要について説明します。また、SQL Anywhere Studio を構成する各部分について説明するチュートリアルも含まれています。
- **『SQL Anywhere Studio 新機能ガイド』** このマニュアルは、SQL Anywhere Studio のこれまでのリリースのユーザを対象としています。ここでは、製品の今回のリリースと以前のリリースで導入された新機能をリストし、アップグレード手順を説明しています。
- **『Adaptive Server Anywhere データベース管理ガイド』** このマニュアルでは、データベースおよびデータベース・サーバの実行、管理、設定について説明しています。
- **『Adaptive Server Anywhere SQL ユーザーズ・ガイド』** このマニュアルでは、データベースの設計と作成の方法、データのインポート・エクスポート・変更の方法、データの検索方法、ストアド・プロシージャとトリガの構築方法について説明します。
- **『Adaptive Server Anywhere SQL リファレンス・マニュアル』** このマニュアルは、Adaptive Server Anywhere で使用する SQL 言語の完全なリファレンスです。また、Adaptive Server Anywhere のシステム・テーブルとシステム・プロシージャについても説明しています。
- **『Adaptive Server Anywhere プログラミング・ガイド』** このマニュアルでは、C、C++、Java プログラミング言語を使用してデータベース・アプリケーションを構築、配備する方法につい

て説明します。Visual Basic や PowerBuilder などのツールのユーザは、それらのツールのプログラミング・インタフェースを使用できます。また、Adaptive Server Anywhere ADO.NET データ・プロバイダについても説明します。

- **『Adaptive Server Anywhere SNMP Extension Agent ユーザーズ・ガイド』** このマニュアルでは、SNMP 管理アプリケーションを使用して Adaptive Server Anywhere データベースを管理する際の Adaptive Server Anywhere SNMP Extension Agent の設定方法について説明します。

- **『Adaptive Server Anywhere エラー・メッセージ』** このマニュアルでは、Adaptive Server Anywhere エラー・メッセージの完全なリストを、その診断情報とともに説明します。

- **『SQL Anywhere Studio セキュリティ・ガイド』** このマニュアルでは、Adaptive Server Anywhere データベースのセキュリティ機能について説明します。Adaptive Server Anywhere 7.0 は、米国政府から TCSEC (Trusted Computer System Evaluation Criteria) の C2 セキュリティ評価を授与されています。このマニュアルには、Adaptive Server Anywhere の現在のバージョンを、C2 基準を満たした環境と同等の方法で実行することを望んでいるユーザにとって役に立つ情報が含まれています。

- **『Mobile Link 管理ガイド』** このマニュアルでは、モバイル・コンピューティング用の Mobile Link データ同期システムについてあらゆる角度から説明します。このシステムによって、Oracle、Sybase、Microsoft、IBM の単一データベースと、Adaptive Server Anywhere や Ultra Light の複数データベースの間でのデータ共有が可能になります。

- **『Mobile Link クライアント』** このマニュアルでは、Adaptive Server Anywhere リモート・データベースと Ultra Light リモート・データベースの設定を行い、これらを同期させる方法について説明します。

- **『Mobile Link サーバ起動同期ユーザーズ・ガイド』** このマニュアルでは、Mobile Link のサーバによって開始される同期について説明します。サーバによって開始される同期とは、統合データベースから同期の開始を可能にする Mobile Link の機能です。

- **『Mobile Link チュートリアル』** このマニュアルでは、Mobile Link アプリケーションの設定方法を示すいくつかのチュートリアルを提供します。
- **『QAnywhere ユーザーズ・ガイド』** このマニュアルでは、Mobile Link QAnywhere について説明します。Mobile Link QAnywhere は、従来のデスクトップ・クライアントやラップトップ・クライアントだけでなく、モバイル・クライアントや無線クライアント用のメッセージング・アプリケーションの開発と展開を可能にするメッセージング・プラットフォームです。
- **『Mobile Link およびリモート・データ・アクセスの ODBC ドライバ』** このマニュアルでは、Mobile Link 同期サーバから、または Adaptive Server Anywhere リモート・データ・アクセスによって、Adaptive Server Anywhere 以外の統合データベースにアクセスするための ODBC ドライバの設定方法について説明します。
- **『SQL Remote ユーザーズ・ガイド』** このマニュアルでは、モバイル・コンピューティング用の SQL Remote データ・レプリケーション・システムについて、あらゆる角度から説明します。このシステムによって、Adaptive Server Anywhere または Adaptive Server Enterprise の単一データベースと Adaptive Server Anywhere の複数データベースの間で、電子メールやファイル転送などの間接的リンクを使用したデータ共有が可能になります。
- **『SQL Anywhere Studio ヘルプ』** このマニュアルには、Sybase Central や Interactive SQL、その他のグラフィカル・ツールに関するコンテキスト別のヘルプが含まれています。これは、製本版マニュアル・セットには含まれていません。
- **『Ultra Light データベース・ユーザーズ・ガイド』** このマニュアルは、Ultra Light 開発者を対象としています。ここでは、Ultra Light データベース・システムの概要について説明します。また、すべての Ultra Light プログラミング・インタフェースに共通する情報を提供します。
- **Ultra Light のインタフェースに関するマニュアル** 各 Ultra Light プログラミング・インタフェースには、それぞれに対応するマニュアルを用意しています。これらのインタフェースは、RAD (

ラピッド・アプリケーション開発) 用の Ultra Light コンポーネントとして提供されているものと、C、C++、Java 開発用の静的インタフェースとして提供されているものがあります。

このマニュアル・セットの他に、PowerDesigner と InfoMaker には、独自のオンライン・マニュアル (英語版) がそれぞれ用意されています。

マニュアルの形式

SQL Anywhere Studio のマニュアルは、次の形式で提供されています。

- **オンライン・マニュアル** オンライン・マニュアルには、SQL Anywhere Studio の完全なマニュアルがあり、SQL Anywhere ツールに関する印刷マニュアルとコンテキスト別のヘルプの両方が含まれています。オンライン・マニュアルは、製品のメンテナンス・リリースごとに更新されます。これは、最新の情報を含む最も完全なマニュアルです。

  Windows オペレーティング・システムでオンライン・マニュアルにアクセスするには、[スタート] ー [プログラム] ー [SQL Anywhere 9] ー [オンライン・マニュアル] を選択します。オンライン・マニュアルをナビゲートするには、左ウィンドウ枠で HTML ヘルプの目次、索引、検索機能を使用し、右ウィンドウ枠でリンク情報とメニューを使用します。

  UNIX オペレーティング・システムでオンライン・マニュアルにアクセスするには、SQL Anywhere のインストール・ディレクトリに保存されている HTML マニュアルを参照してください。

- **PDF 版マニュアル** SQL Anywhere の各マニュアルは、Adobe Acrobat Reader で表示できる PDF ファイルで提供されています。

  PDF 版マニュアルは、オンライン・マニュアルまたは Windows の [スタート] メニューから利用できます。

- **製本版マニュアル** 製本版マニュアルをご希望の方は、ご購入いただいた販売代理店または弊社営業担当までご連絡ください。

表記の規則

この項では、このマニュアルで使用されている書体およびグラフィック表現の規則について説明します。

SQL 構文の表記規則

SQL 構文の表記には、次の規則が適用されます。

- **キーワード** SQL キーワードはすべて次の例に示す ALTER TABLE のように大文字で表記します。

  ALTER TABLE [*owner*.]*table-name*

- **プレースホルダ** 適切な識別子または式で置き換えられる項目は、次の例に示す *owner* や *table-name* のように表記します。

  ALTER TABLE [*owner*.]*table-name*

- **繰り返し項目** 繰り返し項目のリストは、次の例に示す *column-constraint* のように、リストの要素の後ろに省略記号 (ピリオド 3 つ …) を付けて表します。

  ADD *column-definition* [*column-constraint, …*]

  複数の要素を指定できます。複数の要素を指定する場合は、各要素間をカンマで区切る必要があります。

- **オプション部分** 文のオプション部分は角カッコで囲みます。

  RELEASE SAVEPOINT [*savepoint-name*]

  この例では、角カッコで囲まれた *savepoint-name* がオプション部分です。角カッコは入力しないでください。

- **オプション** 項目リストから 1 つだけ選択するか、何も選択しなくてもよい場合は、項目間を縦線で区切り、リスト全体を角カッコで囲みます。

  [**ASC** | **DESC**]

  この例では、ASC と DESC のどちらか 1 つを選択しても、どちらも選択しなくてもかまいません。角カッコは入力しないでください。

- **選択肢**　オプションの中の 1 つを必ず選択しなければならない場合は、選択肢を中カッコで囲み、縦棒で区切ります。

  [**QUOTES** { **ON** | **OFF** }]

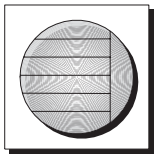
  QUOTES オプションを使用する場合は、ON または OFF のどちらかを選択する必要があります。角カッコと中カッコは入力しないでください。

グラフィック・アイコン

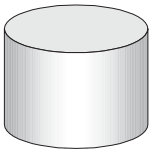
このマニュアルでは、次のアイコンを使用します。

- クライアント・アプリケーション

- Sybase Adaptive Server Anywhere などのデータベース・サーバ



- データベース。高度な図では、データベースとデータベースを管理するデータ・サーバの両方をこのアイコンで表します。



- レプリケーションまたは同期のミドルウェア。ソフトウェアのこれらの部分は、データベース間のデータ共有を支援します。たとえば、Mobile Link 同期サーバ、SQL Remote Message Agent などがあげられます。

- プログラミング・インタフェース

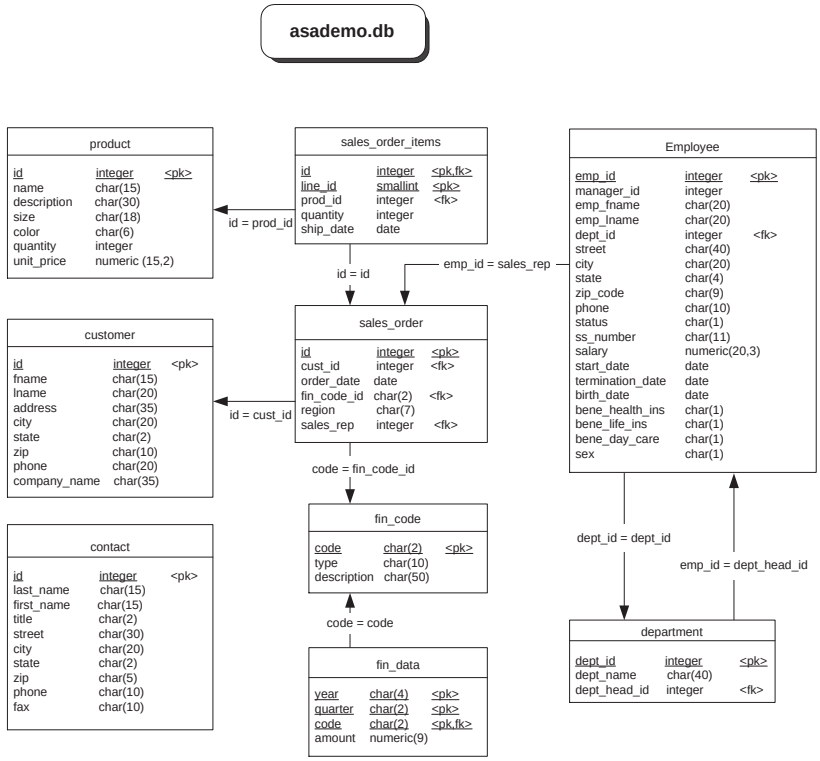
Adaptive Server Anywhere サンプル・データベース

このマニュアルでは、多くの例で Adaptive Server Anywhere サンプル・データベースが使用されています。

サンプル・データベースは、*asademo.db* という名前のファイルに保存され、SQL Anywhere ディレクトリに置かれています。

サンプル・データベースは小規模の企業の例です。データベースには、この企業の内部情報 (従業員、部署、経理) とともに、製品情報や販売情報 (受注、顧客、連絡先) が入っています。データベースに含まれる情報はすべて架空のものです。

次の図は、サンプル・データベース内のテーブルと各テーブル間の関係を示しています。



詳細情報の検索／フィードバックの提供

詳細情報の検索

詳しい情報やリソース (コード交換など) については、iAnywhere Developer Network (http://www.ianywhere.com/developer/) を参照してください。

ご質問がある場合や支援が必要な場合は、次に示す iAnywhere Solutions ニュースグループのいずれかにメッセージをお寄せください。

ニュースグループにメッセージをお送りいただく際には、ご使用の SQL Anywhere Studio バージョンのビルド番号を明記し、現在発生している問題について詳しくお知らせくださいますようお願いいたします。バージョン情報は、コマンド・プロンプトで **dbeng9 -v** と入力して確認できます。

ニュースグループは、ニュース・サーバ *forums.sybase.com* にあります (ニュースグループにおけるサービスは英語でのみの提供となります)。以下のニュースグループがあります。

- sybase.public.sqlanywhere.general
- sybase.public.sqlanywhere.linux
- sybase.public.sqlanywhere.mobilink
- sybase.public.sqlanywhere.product_futures_discussion
- sybase.public.sqlanywhere.replication
- sybase.public.sqlanywhere.ultralite
- ianywhere.public.sqlanywhere.qanywhere

ニュースグループに関するお断り

iAnywhere Solutions は、ニュースグループ上に解決策、情報、または意見を提供する義務を負うものではありません。また、システム・オペレータ以外のスタッフにこのサービスを監視させて、操作状況や可用性を保証する義務もありません。

iAnywhere Solutions のテクニカル・アドバイザとその他のスタッフは、時間のある場合にかぎりニュースグループでの支援を行います。こうした支援は基本的にボランティアで行われるため、解決策や情報を定期的に提供できるとはかぎりません。支援できるかどうかは、スタッフの仕事量に左右されます。

フィードバック

このマニュアルに関するご意見、ご提案、フィードバックをお寄せください。

マニュアルに関するご意見、ご提案は、SQL Anywhere ドキュメンテーション・チームの iasdoc@ianywhere.com 宛てに電子メールでお寄せください。このアドレスに送信された電子メールに返信はいたしませんが、お寄せいただいたご意見、ご提案は必ず読ませていただきます。

マニュアルまたはソフトウェアについてのフィードバックは、上記のニュースグループを通してお寄せいただいてもかまいません。

第 1 部　SQL Remote の概要

第 1 部では、SQL Remote の概念、アーキテクチャ、機能について説明します。SQL Remote for Adaptive Server Anywhere と SQL Remote for Adaptive Server Enterprise の両方に言及します。

第 1 章

SQL Remote へようこそ

この章の内容　この章では、SQL Remote とそのマニュアルについて説明します。

SQL Remote について

SQL Remote は、統合データ・サーバと、一般的には数多くのモバイル・データベースを含む多数のリモート・データベースとの間で双方向レプリケーションを行うためのデータレプリケーション・テクノロジです。

SQL Remote はメッセージを通じてレプリケートするので、サーバ間で直接接続する必要はありません。必要に応じて、ダイヤルアップ接続や電子メールを使用します。

リモート・サイトで必要な管理とリソースは最小限で済みます。統合データベースとリモート・データベース間のタイム・ラグは、分単位、時間単位、または日数単位で設定できます。

Sybase SQL Remote テクノロジは、次の 2 つのパッケージで提供されます。

- **SQL Remote for Adaptive Server Anywhere** Adaptive Server Anywhere 統合データベースと多数のリモート・データベース間のレプリケーションが可能。
- **SQL Remote for Adaptive Server Enterprise** Adaptive Server Enterprise 統合データベースと多数の Adaptive Server Anywhere リモート・データベース間のレプリケーションが可能。

このマニュアルでは、この両方のテクノロジについて説明します。

SQL Remote を使用するには、適切なライセンスを取得した SQL Remote ソフトウェアを、使用する個々のデータベースにインストールする必要があります。

SQL Remote の概念と機能の詳細については、「SQL Remote の概念」7 ページを参照してください。

サポートされているオペレーティング・システムとメッセージ・リンクのリストは、「サポートされるプラットフォームとメッセージ・リンク」553 ページを参照してください。

はじめに

このマニュアルでは、SQL Remote インストール環境の設計、構築、および維持の方法について説明します。

このマニュアルは、次の章で構成されています。

- **SQL Remote の概略**　SQL Remote のレプリケーションの概念と機能
- **SQL Remote のレプリケーションの設計**　SQL Remote インストール環境の設計
- **SQL Remote の管理**　SQL Remote データベースの配備と SQL Remote セットアップの管理
- **リファレンス**　SQL Remote のコマンド、システム・テーブル、参考資料

製品のインストール

ここでは、SQL Remote for Adaptive Server Enterprise のインストール方法について説明します。Adaptive Server Enterprise 以外の製品の一部として SQL Remote を購入した場合は、その製品に付属しているインストール方法の説明に従ってください。

❖ Windows に SQL Remote ソフトウェアをインストールするには、次の手順に従います。

1　CD-ROM ドライブに SQL Remote CD-ROM を挿入します。

2　インストール・プログラムが自動的に起動しない場合は、CD-ROM の *setup* アプリケーションを起動します。

3　インストール・プログラムの指示に従います。

❖ **UNIX に SQL Remote ソフトウェアをインストールするには、次の手順に従います。**

- Adaptive Server Anywhere に付属している『*はじめにお読みください*』で、使用しているオペレーティング・システムに関する指示を参照してください。

SQL Remote for Adaptive Server Enterprise を使用する場合は、レプリケートするデータベースに SQL Remote をインストールする必要があります。

SQL Remote をデータベースにインストールする方法については、「SQL Remote の設定」21 ページを参照してください。

第 2 章

SQL Remote の概念

この章の内容

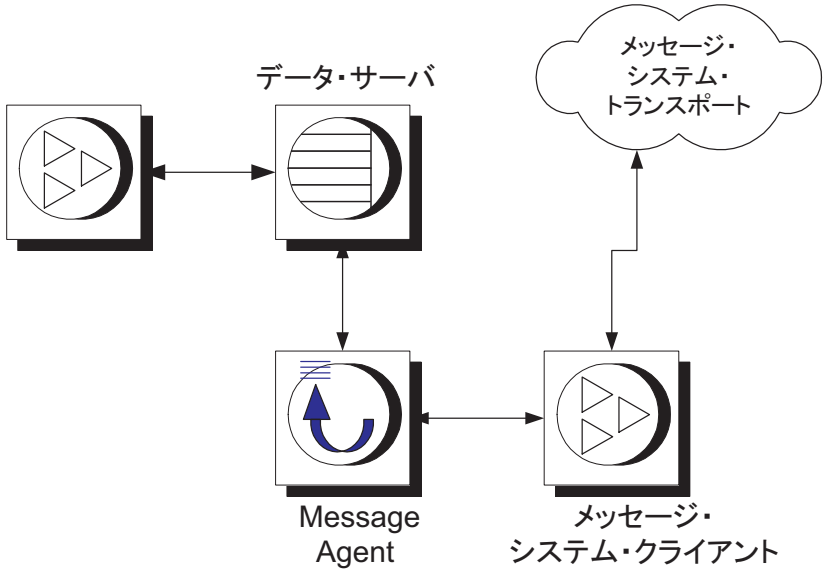
この章では、SQL Remote の概念、設計の目的、機能について説明します。

SQL Remote のコンポーネント

SQL Remote には、以下のコンポーネントが必要です。

- **データ・サーバ** データを管理するために、各サイトに Adaptive Server Anywhere か Adaptive Server Enterprise のいずれかのデータベース管理システムが必要です。
- **Message Agent** SQL Remote メッセージを送受信するために、統合サイトと各リモート・サイトに SQL Remote Message Agent が必要です。

  Message Agent は、クライアント／サーバ接続でデータ・サーバに接続します。Message Agent は、データ・サーバと同じマシンでも別のマシンでも使用できます。
- **データベース抽出ユーティリティ** 開発、テスト、配備の際に、統合データベースからリモート・データベースを作成するには、抽出ユーティリティを使用します。
- **メッセージ・システム・クライアント・ソフトウェア** SQL Remote は、既存のメッセージ・システムを使用して、レプリケーション・メッセージを転送します。SQL Remote に備わっているファイル共有「メッセージ・システム」を使用する場合、クライアント・ソフトウェアは不要です。ファイル共有メッセージ・システム以外を使用して SQL Remote レプリケーションを行う場合は、そのメッセージ・システムを各コンピュータにインストールする必要があります。
- **クライアント・アプリケーション** SQL Remote データベースで使用するのは、標準のクライアント／サーバ・データベース・アプリケーションです。



データ・サーバ

データ・サーバとして使用するのは、Adaptive Server Enterprise サーバまたは Adaptive Server Anywhere サーバです。リモート・サイトでは、Adaptive Server Anywhere パーソナル・サーバをデータ・サーバとして使用するのが普通ですが、Adaptive Server Enterprise サーバや Adaptive Server Anywhere サーバも使用できます。

クライアント・アプリケーション

クライアント・アプリケーションは、データベース内のデータを使用します。クライアント・アプリケーションでは、データ・サーバでサポートされているいずれかのクライアント／サーバ・インタフェースを使用します。

- Adaptive Server Anywhere では、クライアント・アプリケーションは ODBC、Embedded SQL、または Sybase Open Client を使用して Adaptive Server Anywhere を操作します。

- Adaptive Server Enterprise では、クライアント・アプリケーションは Sybase Client Server のいずれかのインタフェース、ODBC、または Embedded SQL を使用します。

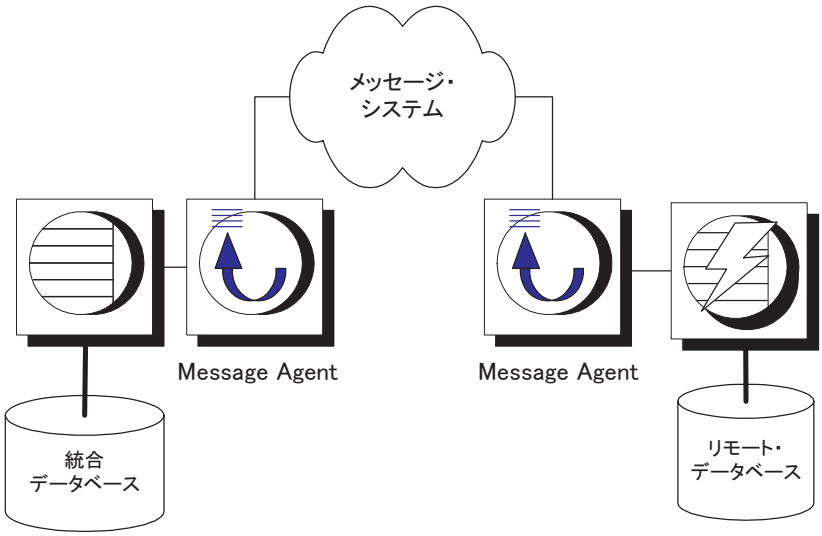
統合データベースとリモート・データベースのどちらを使用しているかを、クライアント・アプリケーション側で認識する必要はありません。クライアント・アプリケーション側から見ると、どちらでも同じだからです。

Message Agent

SQL Remote の **Message Agent** は、レプリケーション・メッセージの送受信を行います。これはデータベース間のメッセージの送受信を行うクライアント・アプリケーションです。統合サイトとリモート・サイトの両方に、Message Agent をインストールする必要があります。

Adaptive Server Anywhere の場合、PC オペレーティング・システムでは *dbremote.exe*、UNIX では *dbremote* というプログラムが Message Agent です。

Adaptive Server Enterprise の場合、PC オペレーティング・システムでは *ssremote.exe*、UNIX では *ssremote* というプログラムが Message Agent です。



メッセージ・システム・クライアント

共有ファイル・メッセージ・システムを使用している場合、メッセージ・システム・クライアントは必要ありません。

電子メールやその他のメッセージ・システムを使用している場合、メッセージを送受信するためには、そのクライアント用のメッセージ・システムが必要です。

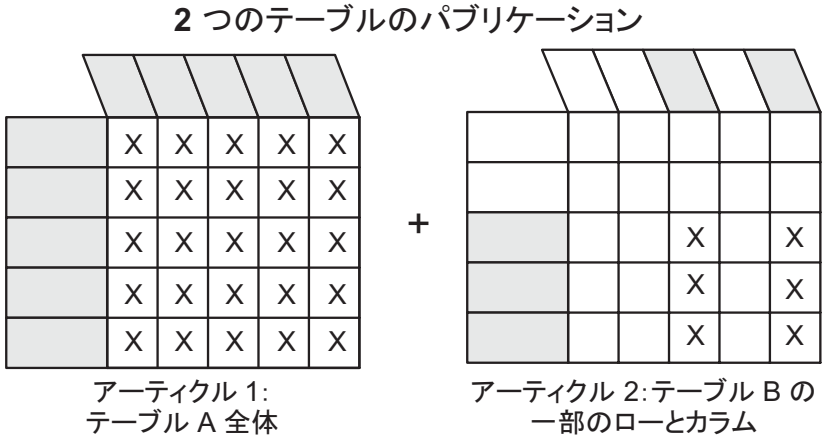
パブリケーションとサブスクリプション

SQL Remote がレプリケートしたデータは、「**パブリケーション**」に記述されます。パブリケーション内の情報を共有する各データベースには、そのパブリケーションに対する「**サブスクリプション**」が必要です。

データはパブリケーションに記述される

「**パブリケーション**」は、レプリケートするデータが記述されたデータベース・オブジェクトです。データベースのリモート・ユーザがパブリケーションを受信するには、パブリケーションに対して「**サブスクリプション**」を作成する必要があります。

1 つのパブリケーションは、複数のデータベース・テーブルから取得したデータを含むことができます。各テーブルからパブリケーションに提供されるデータを、「**アーティクル**」と呼びます。各アーティクルは、1 つのテーブル全体か、1 つのテーブル内のローとカラムのサブセットで構成されます。



データベース内の各パブリケーションに対して変更を加えると、その変更内容はパブリケーションのサブスクライバに定期的にレプリケートされます。このようなレプリケーションを、パブリケーションの「**更新**」と呼びます。

メッセージは常に双方向に送信される

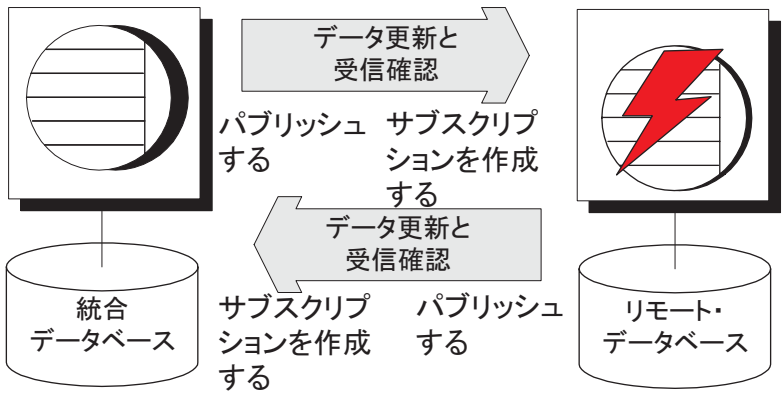
リモート・データベースは、統合データベースからデータを受信できるように、統合データベース上のパブリケーションにサブスクライブします。そのためには、統合データベースで「**サブスクリプション**」を作成し、名前と受信するパブリケーションでサブスクライバを識別します。

SQL Remote では、常に双方向にメッセージが送信されます。統合データベースは、パブリケーション更新を含んだメッセージをリモート・データベースに送信します。リモート・データベースから統合データベースにも、メッセージが送信されます。

たとえば、統合データベースのパブリケーション内でデータが更新されると、更新内容がリモート・データベースに送信されます。また、リモート・データベースでデータが更新されていない場合でも、レプリケーションのステータスを把握するために、リモート・データベースから統合データベースに「**確認メッセージ**」を送り返す必要があります。

両方のデータベースがサブスクリプションを作成する

メッセージは双方向に送信する必要があります。そのため、統合データベースに作成されたパブリケーションに対してリモート・データベースがサブスクリプションを作成するだけでなく、リモート・データベースに作成された対応するパブリケーションに対しても、統合データベースがサブスクリプションを作成する必要があります。



リモート・データベース・ユーザが、自分のマシンにコピーされたデータに変更を加えると、変更内容が統合データベースにレプリケートされます。変更を含んだメッセージを統合データベースで適用すると、その変更が統合データベースのパブリケーションに取り込まれ、

次回にすべてのリモート・サイト (元の変更を加えたサイトを除く) を更新するときに一緒に変更されます。このように、リモート・サイト間でのレプリケーションは、統合データベースを経由して行われます。

リモート・データベースの同期

サブスクリプションを最初に設定したときは、2 つのデータベースに同じ情報が含まれ、レプリケーションを開始できる状態にする必要があります。リモート・データベースと統合データベースとの一貫性を保つように設定するプロセスを、「**同期**」と呼びます。同期は手動でも行えますが、データベース抽出ユーティリティで自動的に行うこともできます。抽出ユーティリティはコマンド・ライン・ユーティリティとして実行できます。また、Adaptive Server Anywhere 統合データベースを使用している場合は、Sybase Central から抽出ユーティリティを実行できます。

SQL Remote データベース抽出ユーティリティでリモート・データベースを作成すると、適切なパブリケーションとサブスクリプションがリモート・データベースに自動的に作成されます。

SQL Remote の機能

SQL Remote の主な機能は以下のとおりです。

多数のサブスクライバのサポート　SQL Remote は、1 つのパブリケーションに対して多数のサブスクライバがある状態でのレプリケーションをサポートしています。

これは、モバイル環境用のアプリケーションにとっては特に重要な機能です。モバイル環境では、オフィスにある 1 つのデータベースから、膨大な数の営業担当者のラップトップ・コンピュータにレプリケートする必要があるからです。

トランザクション・ログ・ベースのレプリケーション　SQL Remote でのレプリケーションは、トランザクション・ログをベースとして行われます。この方法では、更新時に毎回すべてのデータをレプリケートするのではなく、変更のみをレプリケートすることができます。ログ・ベースのレプリケーションは、パフォーマンス面でも他のレプリケーション・システムより優れています。

トランザクション・ログは、データベースに加えた変更がすべて記録されたレポジトリです。SQL Remote は、トランザクション・ログの記録に従って、データベースへの変更をレプリケートします。何らかのパブリケーションに含まれる統合データベースでは、トランザクション・ログに記録されているコミットされた全トランザクションが、定期的にリモート・データベースに送信されます。リモート・サイトでは、トランザクション・ログに記録されているコミットされた全トランザクションが、定期的に統合データベースに送信されます。

SQL Remote では、データのレプリケート中に多少のタイムラグが生じますが、コミットされたトランザクションのみをレプリケートすることによって、レプリケーション設定内でのトランザクションのアトミック性を適正に保ち、レプリケーションに関わるデータベース間で一貫性を維持します。

集中管理　SQL Remote は、統合データベースでの集中管理を考えて設計されています。これは、モバイル環境用アプリケーションでは特に重要なことです。ラップトップ・コンピュータのユーザが、データベース管理タスクを行う必要があってはならないからです。また、小規模なオフィスがいくつかあり、それぞれにサーバはあっても管理タスクを行うほどのリソースがないという状況でのレプリケーションにも、集中管理が必要です。

管理タスクには、パブリケーション、リモート・ユーザ、サブスクリプションの設定や管理のほか、エラーや競合が起きた場合の解決も含まれます。

経済的なリソース要件 SQL Remote の実行に必要なソフトウェアは、お使いの Adaptive Server Anywhere または Adaptive Server Enterprise DBMS 以外には、Message Agent とメッセージ・システムのみです。共有ファイル・リンクを使用していて、メッセージ・ファイルの保管先ディレクトリに各リモート・ユーザ ID でアクセスできる場合は、メッセージ・システム・ソフトウェアも不要です。

レプリケーション・システムのコンポーネントは、どれもそれほど多くのメモリとディスク容量を必要としないので、SQL Remote を実行するためにハードウェアを増設する必要はありません。

マルチプラットフォーム・サポート SQL Remote は、さまざまなオペレーティング・システムとメッセージ・リンクに対応しています。

サポートされている環境のリストについては、「サポートされるプラットフォームとメッセージ・リンク」553 ページを参照してください。

インストールの例

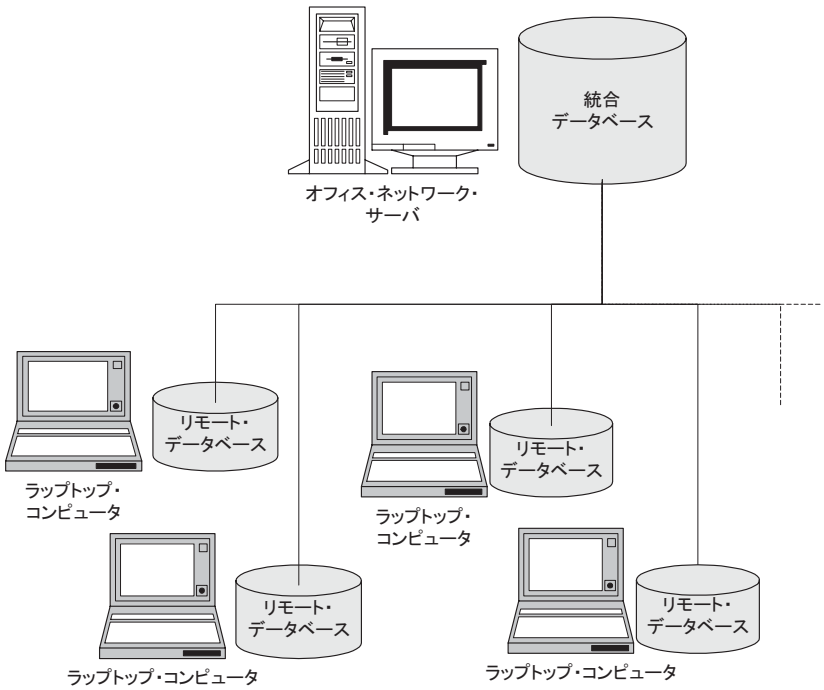
SQL Remote ではさまざまな環境下でレプリケーション・サービスを利用でき、その機能は以下の特徴を意識して設計されています。

- モバイル環境用アプリケーションの場合と同様に、リモート・データベースに管理の負荷を課すことができない場合にも、SQL Remote がソリューションとなること。
- サイト間のデータ通信は、散発的かつ間接的であること。永続的で直接的である必要はない。
- リモート・サイトでのメモリとリソースの要件を重要視する。

以下に、SQL Remote の典型的な設定を示します。

モバイル環境でのサーバ／ラップトップ間レプリケーション

SQL Remote では、オフィス・ネットワーク上のデータベースと、営業担当者のラップトップ・コンピュータ上にあるパーソナル・データベースとの間で、双方向のレプリケーションが可能です。このような設定では、メッセージの伝送手段として電子メールシステムを使用します。



オフィス・サーバが、会社のデータベースを管理するためのサーバを稼働させている場合、会社のデータベースにある Message Agent は、そのサーバのクライアントとして稼働します。

ラップトップ・コンピュータでは、各営業担当者は Adaptive Server Anywhere パーソナル・サーバを使用して、自分のデータを管理します。

営業担当者は、ラップトップ・コンピュータから 1 回電話をかけるだけで、外出先から以下の機能を実行できます。

- 新着メールを受信する。
- 作成済みの電子メールがある場合は送信する。
- オフィス・サーバからパブリケーションの更新を受信する。

- ローカルで行った更新の内容 (新しい注文など) を、オフィス・サーバに送信する。

更新の例としては、その営業担当者が取り扱っている製品の新しい特別割引や、新価格、在庫情報などがあります。これらの情報はラップトップの Message Agent によって読み込まれ、担当者のデータベースに自動的に適用されます。担当者側での操作は一切不要です。

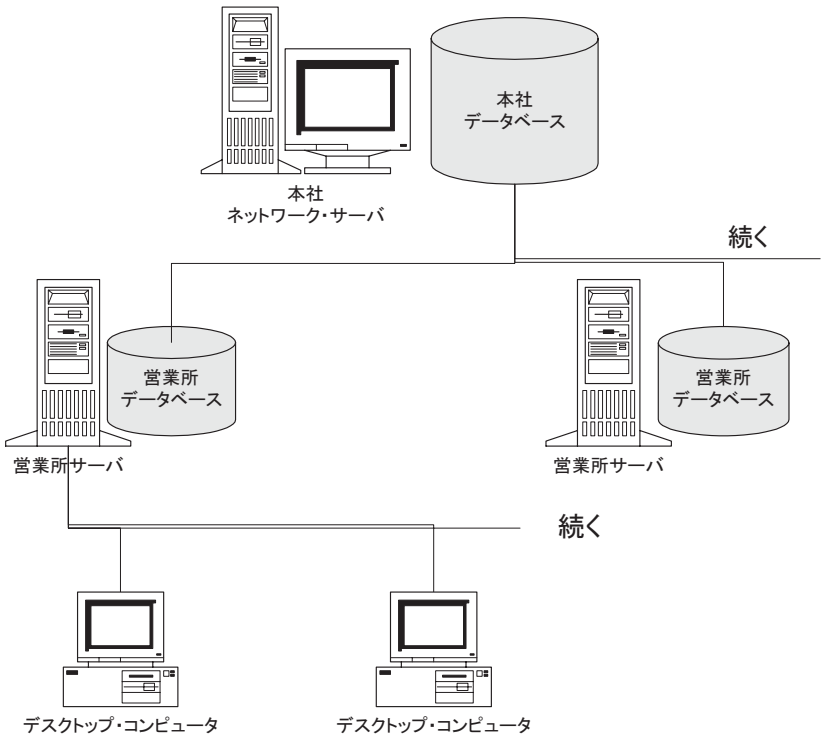
営業担当者が登録した新しい注文も、特別な操作をしなくても自動的にオフィスに送信されます。

複数のオフィスにわたるサーバ間レプリケーション

SQL Remote では、営業所／販売店と本社のデータベース・サーバ間で、双方向レプリケーションが可能です。各営業所では、最初の設定とサーバの管理さえ行うことができれば、それ以上のデータベース管理の経験は必要ありません。

SQL Remote は、各サイトのデータを常に最新に保つような設計にはなっていません。1 時間に 1 回程度の間隔でデータをレプリケートできるような状況に適しています。

その場合、すでに社内に電子メール・システムが構築されていれば、それを使ってレプリケーションを実行できます。あるいは、ダイアルアップ・システムとファイル転送ソフトウェアを随時使用して、FILE メッセージ・システムを実行することもできます。



SQL Remote では、簡単な設定を行うだけで、各オフィスでそれぞれに適したデータ・セットを受信できます。オフィス専用のテーブル (フランチャイズ・オフィスの場合はスタッフ・レコードなど) は、レプリケーション・データと同じデータベース内にあっても機密性を保つことができます。

SQL Remote の階層には、レイヤを追加できます。たとえば、各営業所のサーバを統合データベースとして利用して、その営業所からのリモート・サブスクライバをサポートすることができます。

第 3 章

SQL Remote の設定

この章の内容

この章では、Adaptive Server Enterprise サーバに SQL Remote 機能を追加する方法について説明します。

Adaptive Server Enterprise ユーザ対象

この章の内容は Adaptive Server Enterprise の SQL Remote ユーザのみを対象としています。SQL Remote 機能は、Adaptive Server Anywhere データベースに自動的にインストールされます。

この章では SQL Remote ソフトウェアがすでにマシンにインストールされていることを前提に説明します。

設定の概要

SQL Remote を使ってお互いに情報を交換するデータベースの集合を「**インストール環境**」と呼びます。物理的な観点では SQL Remote インストール環境は、互いに情報を共有する数百から数千のデータベースで構成されています。しかし、SQL Remote は各物理データベースに格納している情報と他の物理データベースにある情報のおおまかな整合性をトランザクション・レベルで維持するのでインストール環境全体を単一の「**分散データベース**」とみなすこともできます。

SQL Remote のインストール環境を大規模に配備するには数多くのマシン上でデータベースを設定する必要があります。稼働中のインストール環境で、変更可能な設計および設定もありますが、設計を注意深く分析し、テストしてから配備することを強くおすすめします。

設定作業

SQL Remote インストール環境の設定には次の作業が含まれます。

- **SQL Remote 用サーバの準備** Adaptive Server Enterprise を SQL Remote サイトとして使用するための設定をいくつか行う必要があります。これには、SQL Remote システム・オブジェクトとステーブル・キューのシステム・オブジェクトのインストールが含まれます。
- **メッセージ・タイプの選択** ファイル共有、電子メール、その他のメッセージ・タイプ、またはこれらのタイプの組合せの中からどの方法で情報を交換するかを決定します。
- **パーミッション設定の確認** インストールする各ユーザには自分自身のデータベースと統合データベース両方のパーミッションが必要です。
- **リモート・データベースの抽出** 統合データベースから各リモート・データベースの初期コピーを抽出する必要があります。

この章では、上記のタスクのそれぞれについて説明します。

すべての管理タスクは統合データベースで実行する

その他の SQL Remote 管理タスクと同様に、諸機能の設定も、データベース管理者またはシステム管理者が統合データベースで実行します。

Sybase のシステム管理者が、すべての SQL Remote 設定タスクを実行するようにしてください。Adaptive Server Enterprise 環境の詳細については、Adaptive Server Enterprise のマニュアルを参照してください。

Adaptive Server Enterprise サーバの準備

始める前に

この項では、次の条件を満たしていることを前提にしています。

- SQL Remote データベースを入れる Adaptive Server Enterprise サーバがインストールされていること。
- コンピュータに SQL Remote ソフトウェアがインストールされていること。SQL Remote ソフトウェアをインストールするには CD-ROM からセットアップ・プログラムを実行します。
- SQL Remote のインストール環境で使用する Adaptive Server Enterprise サーバにデータベースが作成されていること。
- Adaptive Server Enterprise サーバに対するシステム管理者のパーミッションとデータベースに対するデータベース所有者のパーミッションを持っていること。

十分なサイズの TEMPDB

SQL Remote では、TEMPDB データベースを以下の目的で使用します。

- リモート・データベースの作成に使用されるデータベース抽出ユーティリティは、TEMPDB に Adaptive Server Anywhere システム・テーブル・セットを一時的に保持します。
- Message Agent はサーバに接続するとき、**#remote** という名前のテンポラリ・テーブルを作成します。

これらの理由から TEMPDB はデフォルトの 2 MB よりも大きい必要があります。必要なサイズは、SQL Remote インストール環境のテーブル数とカラム数によって異なりますが通常は 10 MB あれば十分です。

SQL Remote システム・オブジェクトのインストール

Adaptive Server Enterprise サーバのデータベースを SQL Remote インストール環境に含めるには、データベースに SQL Remote のシステム・テーブル、ビュー、ストアド・プロシージャをいくつかインストールする必要があります。

❖ SQL Remote システム・オブジェクトのインストールについては、次の手順に従います。

1. SQL Remote インストール・ディレクトリにある SQL Remote 初期化スクリプト *ssremote.sql* を探します。

2. スクリプト・ファイル *ssremote.sql* のバックアップ・コピーを作成します。次の 2 行を *ssremote.sql* の先頭に追加します。

   ```
   use database_name
    go
   ```

   database_name には、SQL Remote レプリケーションに含めるデータベース名を入力します。

   この 2 行は、現在のデータベースを *database_name* に設定し、SQL Remote テーブルを *database_name* データベースに作成するためのものです。SQL Remote テーブルの所有者は、このデータベースの所有者です。

3. Adaptive Server Enterprise サーバに対して、スクリプトを実行します。

   スクリプト・ファイルが入っているディレクトリに移動して次のコマンド・ラインを入力し (すべて 1 行に入力すること)、スクリプトを実行します。

   ```
   isql -S server-name -U login_id -P password -I
   ssremote.sql -o logfile
   ```

ここで、*server-name* には Adaptive Server Enterprise 名、*login_id* と *password* には、データベースを所有しサーバのシステム管理者パーミッションを持つユーザ名とパスワード、*logfile* にはスクリプトのログ情報を格納するログ・ファイル名を指定します。

login_id は、Message Agent で使用したものと同じである必要があります。詳細については、「Message Agent とレプリケーション・セキュリティ」346 ページを参照してください。

4 ログ・ファイルを開きテーブルとプロシージャが正しく作成されていることを確認します。

スクリプトが実行され、データベースに一連の SQL Remote システム・オブジェクトが作成されます。

SQL Remote システム・オブジェクト

スクリプトにより、データベースに以下のオブジェクトが作成されます。

- **SQL Remote システム・テーブル** SQL Remote 情報を維持するために使用するテーブル・セット。これらのテーブル名は **sr_** で始まります。
- **SQL Remote システム・ビュー** SQL Remote 情報をわかりやすい形式で保持する一連のビュー。これらのビュー名は **sr_** で始まり、**s** で終わります。
- **SQL Remote システム・プロシージャ** SQL Remote 設定と管理タスクを実行するのに使用する一連のストアド・プロシージャ。これらのプロシージャ名は **sp_** で始まり、システム管理上の役割を示すものとなっています。

警告：SQL Remote システム・テーブルは編集しないでください
SQL Remote システム・テーブルは、どのような状況下でも決してユーザが変更しないでください。変更するとテーブルが破壊され SQL Remote が正しく動作しなくなることがあります。システム管理タスクは、すべて SQL Remote システム・プロシージャを使って実行してください。

ステーブル・キューのコマンド・ラインからのインストール

「**ステーブル・キュー**」とは、レプリケーション・システムで不要となるまでトランザクションを保持する 1 対のデータベース・テーブルのことです。SQL Remote インストール環境に含まれる Adaptive Server Enterprise データベースには、それぞれにステーブル・キューが 1 つ必要です。

ステーブル・キューの詳細については、「ステーブル・キュー」342 ページを参照してください。

ステーブル・キューの保存場所は、SQL Remote インストール環境に含まれるデータベースと同じでも、別のデータベースでも構いません。ステーブル・キューを別のデータベースに保存すると、バックアップとリカバリ処理が複雑になりますが、別のデバイスや Adaptive Server Enterprise サーバにステーブル・キューの負荷を分散することによって、パフォーマンスを改善できます。

❖ ステーブル・キューのインストールについては、次の手順に従います。

1. SQL Remote インストール・ディレクトリにあるステーブル・キューの初期化スクリプト *stableq.sql* を探します。
2. スクリプト・ファイル *stableq.sql* のバックアップ・コピーを作成します。次の 2 行を *stableq.sql* の先頭に追加します。

```
use database_name
 go
```

database_name には、ステーブル・キューを保持するデータベース名を入力します。

この 2 行は、現在のデータベースを *database_name* に設定し、ステーブル・キューを *database_name* データベースに作成するためのものです。ステーブル・キュー・テーブルの所有者は、データベース所有者です。

3 Adaptive Server Enterprise サーバに対して、スクリプトを実行します。

ステーブル・キューのスクリプト・ファイルが入っているディレクトリに移動して次のコマンド・ラインを入力し (すべて 1 行に入力すること)、スクリプトを実行します。

```
isql -S server-name -U login_id -P password -I
stableq.sql -o logfile
```

ここで、*server-name* には Adaptive Server Enterprise 名、*login_id* と *password* には、データベースを所有しサーバのシステム管理者パーミッションを持つユーザ名とパスワード、*logfile* にはスクリプトのログ情報を格納するログ・ファイル名を指定します。

login_id は、Message Agent で使用したものと同じである必要があります。詳細については、「Message Agent とレプリケーション・セキュリティ」346 ページを参照してください。

4 ログ・ファイルを開きテーブルとプロシージャが正しく作成されていることを確認します。

SQL Remote for Adaptive Server Enterprise のアップグレード

この項では、SQL Remote for Adaptive Server Enterprise をアップグレードする手順について説明します。

SQL Remote インストール環境には大量のデータベースが含まれる場合が多いので通常すべてのマシンのソフトウェアを同時にアップグレードすることはありません。SQL Remote は、差分のみのアップグレード (インクリメンタル・アップグレード) が可能です。アップグレード後のメッセージ形式は前のリリースと互換性があるので、アップグレードを実行する SQL Remote マシンの順序は自由です。

❖ SQL Remote のアップグレードについては、次の手順に従います。

1. 統合データベースとステーブル・キュー・データベース (分離されている場合) のバックアップを作成します。
2. 新規 SQL Remote for Adaptive Server Enterprise ソフトウェアをインストールします。
3. 統合データベースでスクリプト *ssupdate.sql* を実行し、SQL Remote システム・テーブルとプロシージャをアップグレードします。

   スクリプト *ssupdate.sql* は Sybase ディレクトリに保存されます。
4. ステーブル・キュー・データベースでスクリプト *squpdate.sql* を実行し、SQL Remote ステーブル・キュー・テーブルとプロシージャをアップグレードします。

   スクリプト *squpdate.sql* は Sybase ディレクトリに保存されます。

これで、ソフトウェアのアップグレードは完了です。

SQL Remote のアンインストール

この項では、SQL Remote オブジェクトをデータベースからアンインストールする方法と、データベースからステーブル・キューをアンインストールする方法について説明します。

❖ SQL Remote オブジェクトのデータベースからのアンインストールについては、次の手順に従います。

1 dbo パーミッションを持つユーザとして、SQL Remote オブジェクトが入ったデータベースに接続します。

2 ストアド・プロシージャ **sp_drop_sql_remote** を実行し、すべての SQL Remote オブジェクトをこのプロシージャ自体から切り離して削除します。プロシージャ **sp_drop_sql_remote** は、他の SQL Remote オブジェクトと一緒にインストールされています。

```
exec sp_drop_sql_remote
 go
```

3 プロシージャ **sp_drop_sql_remote** を削除して、アンインストールを完了します。

```
drop procedure sp_drop_sql_remote
 go
```

❖ ステーブル・キューのデータベースからのアンインストールについては、次の手順に従います。

1 dbo パーミッションのあるユーザとして、ステーブル・キューを保持するデータベースに接続します。

2 ストアド・プロシージャ **sp_queue_drop** を実行し、プロシージャ自体からすべてのステーブル・キュー・オブジェクトを切り離して削除します。プロシージャ **sp_queue_drop** は他のステーブル・キュー・オブジェクトと一緒にインストールされています。

```
exec sp_queue_drop
 go
```

3　プロシージャ **sp_queue_drop** を削除して、アンインストールを完了します。

```
drop procedure sp_queue_drop
 go
```

第 4 章

Adaptive Server Anywhere ユーザのためのチュートリアル

この章の内容

この章では、Adaptive Server Anywhere を使った簡単なレプリケーション・システムを設定する方法について説明します。

概要

このチュートリアルでは、Adaptive Server Anywhere を使用して簡単な SQL Remote レプリケーション・システムを設定する方法について説明します。

目的

このチュートリアルでは、統合 Adaptive Server Anywhere データベースのシステム管理者として、簡単なレプリケーション・システムを設定します。このレプリケーション・システムは、2 つのテーブルからなる簡単な販売データベースで構成します。

統合データベースはすべてのデータベースを抱合しています。リモート・データベースは同一のデータで構成されますが、いくつかのローは他のリモート・データベースのテーブルに含まれます。

このチュートリアルの手順は次のとおりです。

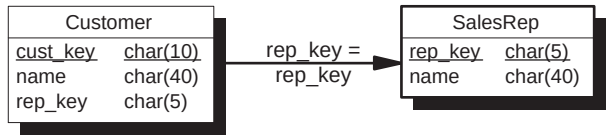
- Adaptive Server Anywhere サーバ上に統合データベースを作成する。
- 1 つの Adaptive Server Anywhere リモート・データベースを含む、ファイル共有レプリケーション・システムを作成する。
- 2 つのデータベース間で、データをレプリケートする。

データベース

このチュートリアルでは、2 つのテーブルで構成される簡単なデータベースを使用します。1 つのテーブルには営業担当の情報が、もう 1 つのテーブルには顧客の情報が入っています。これらのテーブルは実際のデータベースで使用するものよりずっと単純なので、レプリケーションに関する重要な問題だけに注目できます。

データベースのスキーマ

このチュートリアルで使うデータベースのスキーマを次の図に示します。



ここで注意すべき特徴には、次のものがあります。

- 各営業担当は **SalesRep** テーブルの 1 つのローで表されています。
- 各顧客は **Customer** テーブルの 1 つのローで表されています。
- 各顧客は 1 人の営業担当に割り当てられており、この割り当ては Customer テーブルから **SalesRep** テーブルへの外部キーとしてデータベースに組み込まれています。Customer テーブルと **SalesRep** テーブルとの関係は、多対 1 です。

データベース内のテーブル

テーブルの詳細は次のとおりです。

テーブル	説明

SalesRep	会社の営業担当それぞれに 1 つのロー。SalesRep テーブルには、次のカラムがあります。
	• **rep_key**　各営業担当の識別子。これがプライマリ・キーです。
	• **name**　各営業担当の名前。
	このテーブルを作成する SQL 文は次のとおりです。
	`CREATE TABLE SalesRep (`
	`rep_key CHAR(12) NOT NULL,`
	`name CHAR(40) NOT NULL,`
	`PRIMARY KEY (rep_key)`
	`)`

テーブル	説明
Customer	会社と取引がある顧客それぞれに 1 つのロー。Customer テーブルには、次のカラムがあります。 • **cust_key** 各顧客の識別子。これがプライマリ・キーです。 • **name** 各顧客の名前。 • **rep_key** 取引を行う営業担当者の識別子。これが SalesRep テーブルへの外部キーです。 このテーブルを作成する SQL 文は次のとおりです。 `CREATE TABLE Customer ( cust_key CHAR(12) NOT NULL, name CHAR(40) NOT NULL, rep_key CHAR(12) NOT NULL, FOREIGN KEY ( rep_key ) REFERENCES SalesRep (rep_key ), PRIMARY KEY (cust_key) )`

レプリケーションの目標

このレプリケーションの設計上の目的は、各営業担当に次の情報を提供することです。

- 完全な **SalesRep** テーブル。
- 割り当てられている顧客。

このチュートリアルでは、SQL Remote を使用してこの目的を達成する方法について説明します。

Sybase Central またはコマンド・ライン・ユーティリティ

Sybase Central またはコマンド・ラインの使用

チュートリアルには 2 種類あります。最初に、Sybase Central 管理ユーティリティを使用してインストール環境を設定する方法について説明します。次に、コマンド・ライン・ユーティリティを使用してインストール環境を設定する方法について説明します。この場合は、コマンドを自分で入力する必要があります。

次の手順

- Sybase Central を使用したチュートリアルを実行する場合は、「チュートリアル：Sybase Central を使用した Adaptive Server Anywhere レプリケーション」38 ページに進んでください。
- コマンドを明示的に入力するチュートリアルを実行する場合は、「チュートリアル : Interactive SQL と dbxtract を使用した Adaptive Server Anywhere レプリケーション」50 ページに進んでください。

チュートリアル：Sybase Central を使用した Adaptive Server Anywhere レプリケーション

次の項からは、Sybase Central を使用して、Adaptive Server Anywhere で簡単な SQL Remote レプリケーション・システムを設定する方法を説明するチュートリアルです。

Sybase Central を使用して SQL Remote を管理している場合は、SQL 文を入力する必要はありません。Sybase Central へのアクセス権を持たないユーザ、またはコマンド・ライン・ユーティリティを使用するユーザ向けのチュートリアルについては、「チュートリアル：Interactive SQL と dbxtract を使用した Adaptive Server Anywhere レプリケーション」50 ページを参照してください。そのチュートリアルでは、Sybase Central が裏で実行する SQL 文について説明しています。

このチュートリアルでは統合データベースの DBA として操作を行い、ファイル共有メッセージ・リンクを使用して簡単なレプリケーション・システムを設定します。例として、2 つのテーブルからなる営業戦力自動化システムの基本モデルを作成します。1 つのテーブルには営業担当のリストが、もう 1 つのテーブルには顧客のリストが入ります。これらのテーブルは 1 つの統合データベースと 1 つのリモート・データベースを含む設定にレプリケートされます。この例のシステムは、1 台のコンピュータにインストールできます。

このチュートリアルは、Sybase Central についてある程度の知識を持っている方を対象としています。

Sybase Central の概要ついては、『SQL Anywhere Studio の紹介』>「Sybase Central を使用したデータベースの管理」を参照してください。

Sybase Central レプリケーションのチュートリアルの準備

この項では、このチュートリアルの準備に必要な手順について説明します。手順とは次のようなものです。

- チュートリアルに必要なディレクトリとデータベースを作成する。

- 統合データベースにテーブルを追加する。

❖ チュートリアルの準備をする場合は、次の手順に従います。

1. このチュートリアルで作成するファイルを置くディレクトリを作成します。たとえば *c:¥tutorial* とします。

   ```
   mkdir c:¥tutorial
   ```

2. レプリケーション・システム内の 2 つのユーザ ID 用に、それぞれメッセージを置くサブディレクトリを作成します。システム・コマンド・ラインで次の文を使用して、これらのサブディレクトリを作成します。

   ```
   mkdir c:¥tutorial¥HQ
    mkdir c:¥tutorial¥field
   ```

3. **HQ** データベースを作成します。

   - Sybase Central を起動します。
   - 左ウィンドウ枠で、Adaptive Server Anywhere プラグインを選択します。
   - 右ウィンドウ枠で、[ユーティリティ] タブをクリックします。
   - 右ウィンドウ枠にある [データベースの作成] をダブルクリックします。

     [データベース作成] ウィザードが表示されます。
   - ファイル名 *c:¥tutorial¥hq.db* でデータベースを作成します。

     このデータベースのデフォルト設定を使用します。

   Adaptive Server Anywhere データベースは単なるファイルなので、必要であれば他の場所やコンピュータにコピーできます。

次の手順では、統合データベースに 1 組のテーブルを追加します。

❖ 統合データベースにテーブルを追加する場合は、次の手順に従います。

1 Sybase Central から **HQ** データベースに接続します。ユーザ ID には **DBA**、パスワードには **SQL** を使用します。

2 左ウィンドウ枠で、**HQ** データベースの [テーブル] フォルダを選択します。

3 [ファイル] メニューで [新規] ー [テーブル] を選択し、[テーブル作成] ウィザードを使用して **SalesRep** という名前のテーブルを作成します。

4 テーブルに次のカラムを追加します ([ファイル] ー [カラムを追加] を選択してカラムを追加できます)。

キー	カラム	データ型	サイズ／精度
プライマリ・キー	**Rep_key**	char	5
	Name	char	40

カラムのプロパティ・シートを使用する必要はありません。

5 [ファイル] ー [テーブルの保存] を選択するか [Ctrl+S] を押して、テーブルを保存します。

6 [ファイル] メニューで [新規] ー [テーブル] を選択し、次のカラムを持つ **Customer** という名前のテーブルを作成します。

キー	カラム	データ型	サイズ／精度
プライマリ・キー	**Cust_key**	char	10
	Name	char	40
	Rep_key	char	5

ここでも、プロパティ・シートを使う必要はありません。

7　テーブルを保存します。

8　左ウィンドウ枠の [テーブル] フォルダで、Customer テーブルを選択し、右ウィンドウ枠の [外部キー] タブをクリックします。

9　[ファイル] メニューで [新規] ー [外部キー] を選択します。ウィザードを使用して、**SalesRep** テーブルの **Rep_key** カラムに外部キーを追加します。この外部キーにはデフォルト設定を使用できます。

次のチュートリアルに進みましょう。

統合データベースの設定

チュートリアルのこの項では、簡単なレプリケーション・システムの統合データベースを準備する方法について説明します。

レプリケーションに使用する統合データベースを準備する手順は次のとおりです。

1. レプリケーションに使用するメッセージ・タイプを作成します。
2. 出力メッセージのソースを識別できるように、ユーザ ID に PUBLISH パーミッションを付与します。
3. メッセージを受信するすべてのユーザ ID に REMOTE パーミッションを付与します。
4. レプリケートするデータを記述するパブリケーションを作成します。
5. パブリケーションの受信側を記述するサブスクリプションを作成します。

これらの作業を行うには、DBA 権限が必要です。

SQL Remote メッセージ・タイプの追加

レプリケーションの一部として送信されるすべてのメッセージは、メッセージ・タイプを使用します。メッセージ・タイプの記述は、次の 2 つの部分で構成されます。

- SQL Remote でサポートされているメッセージ・リンク。このチュートリアルでは、FILE リンクを使用します。
- 出力メッセージのソースを識別する、メッセージ・リンクのアドレス。

Adaptive Server Anywhere データベースではメッセージ・タイプはすでに作成されていますが、使用するメッセージ・タイプ用のアドレスを指定する必要があります。

❖ メッセージ・タイプにアドレスを追加する場合は、次の手順に従います。

1. Sybase Central から **HQ** データベースに接続します。
2. **HQ** データベースの [SQL Remote ユーザ] フォルダを開きます。
3. 右ウィンドウ枠の [メッセージ・タイプ] タブをクリックします。
4. 右ウィンドウ枠の [FILE] メッセージ・タイプを右クリックして、ポップアップ・メニューで [プロパティ] を選択します。
5. リモート・ユーザ用のリターン・アドレスとなる、パブリッシャ・アドレスを入力します。ここでは、統合データベース (**HQ**) でメッセージを格納するために作成したディレクトリを入力します。

   このアドレスは、SQLRemote 環境変数、またはレジストリ・エントリからの相対アドレスです。アドレスを設定しなかった場合は、Message Agent が実行されているディレクトリからの相対アドレスが設定されます。このアドレスが正しく解釈されるように、チュートリアル・ディレクトリから Message Agent を実行してください。

SQLRemote 値の設定方法については、「メッセージ・タイプ制御パラメータの設定」274 ページを参照してください。

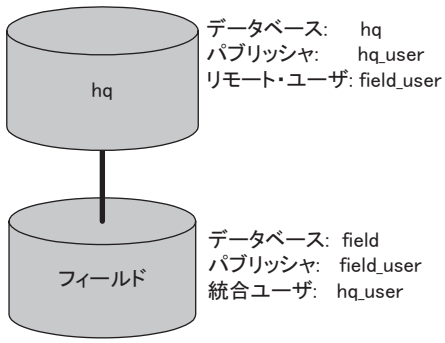
6　[OK] をクリックして、メッセージ・タイプを保存します。

データベースへのパブリッシャとリモート・ユーザの追加

SQL Remote の階層レプリケーション・システムでは、各データベースのすぐ上にデータベースが 0 または 1 つあり (統合データベース)、すぐ下にデータベースが 0 または複数あります (リモート・データベース)。

このチュートリアルでは、現在のデータベースは、2 つのレベルからなるシステムの、統合データベースです。このデータベースの上にはデータベースがなく、下にはリモート・データベースが 1 つだけあります。

次の図は、この 2 つのデータベースを示したものです。



SQL Remote レプリケーション設定のどのデータベースにも、次の 3 つのパーミッションを付与できます。これらのパーミッションは、階層内のデータベースを識別するためのものです。

- **PUBLISH パーミッション**　すべての出力メッセージにある現在のデータベースを識別します。

- **REMOTE パーミッション**　現在のデータベースからメッセージを受信する各データベース (階層で下にあるデータベース) を識別します。
- **CONSOLIDATE パーミッション**　現在のデータベースからメッセージを受信する 1 つのデータベース (階層ですぐ上にあるデータベース) を識別します。

これらのパーミッションは、DBA 権限を持つユーザだけが付与できます。これらの例を実行するには、Sybase Central から **hq** データベースに接続します。ユーザ ID には **DBA**、パスワードには **SQL** を使用します。

データベース・パブリッシャ・ユーザ ID の追加

統合かリモートかにかかわらず、レプリケーション・システム内で他のデータベースに変更を配布するデータベースはすべてパブリッシャ・データベースです。レプリケーション・システムの各データベースは、1 つのユーザ ID で識別されます。この ID は、データベースにパブリッシャを追加することによって設定します。この項では、統合データベース **hq** へのパーミッションの設定について説明します。

まず **hq_user** というユーザ ID を作成します。これがパブリッシャ・ユーザ ID になります。

❖ 新規ユーザをパブリッシャとして作成する場合は、次の手順に従います。

1. [ユーザとグループ] フォルダを選択します。
2. [ファイル] メニューから [新規] ー [ユーザ] を選択します。

   [ユーザ作成] ウィザードが表示されます。
3. **hq_user** という名前を入力し、パスワードに **hq_pwd** を指定して、[完了] をクリックします。
4. [**hq_user**] アイコンを右クリックして、ポップアップ・メニューで [パブリッシャに変更] を選択します。

   1 つのデータベースに設定できるパブリッシャは 1 つだけです。どのユーザがパブリッシャかは、[ユーザとグループ] フォルダを開けばいつでも確認できます。

リモート・ユーザの追加

統合データベースでは、各リモート・データベースは REMOTE パーミッションを持つユーザ ID で識別されます。リモート・データベースがパーソナル・データベース・サーバであっても、多くのユーザがあるネットワーク・サーバであっても、統合データベースに対してそのデータベースを示す 1 つのユーザ ID が必要です。

モバイル・ワークグループの設定で、リモート・ユーザがすでに統合データベースのユーザの場合があります。その場合、新しくユーザを追加する必要はありませんが、リモート・ユーザとして設定する必要はあります。

リモート・ユーザをデータベースに追加するときに、各ユーザが使用するメッセージ・システムとそのメッセージ・システムにおける各ユーザのアドレスを、それぞれのデータベース・ユーザ ID と共に格納する必要があります。

❖ リモート・ユーザを追加する場合は、次の手順に従います。

1. [SQL Remote ユーザ] フォルダを選択します。
2. [ファイル] メニューから [新規] ー [SQL Remote ユーザ] を選択します。

   [新しいリモート・ユーザの作成] ウィザードが表示されます。
3. 次のオプションを指定して、ユーザ ID が **field_user** のリモート・ユーザを作成します。
   - パスワード **field_pwd** を入力します。
   - Remote DBA 権限が選択されていることを確認します。これによって、このユーザが Message Agent を実行できます。
   - メッセージ・タイプ **FILE** を選択し、アドレス **field** を入力します。

     パブリッシャ・アドレスと同様に、リモート・ユーザのアドレスは、SQLREMOTE 環境変数またはレジストリ・エントリからの相対アドレスです。アドレスを設定しなかった場合は、Message Agent が実行されているディレ

クトリからの相対アドレスが設定されます。このアドレスが正しく解釈されるように、チュートリアル・ディレクトリから Message Agent を実行してください。

SQLREMOTE 値の設定方法については、「メッセージ・タイプ制御パラメータの設定」274 ページを参照してください。

- [送信して閉じる] オプションが選択されていることを確認します。

  (多くの運用環境では [送信して閉じる] を選択しませんが、このチュートリアルでは選択すると便利です)。

4 入力がすべて終わったら、[完了] をクリックし、リモート・ユーザを作成します。

以上でこのシステムを使用するユーザの作成が完了しました。

パブリケーションとサブスクリプションの追加

この項では、パブリケーションをデータベースに追加する方法と、そのパブリケーションにユーザのサブスクリプションを追加する方法について説明します。パブリケーションは、**SalesRep** テーブルのすべてのローと、**Customer** テーブルのいくつかのローをレプリケートします。

❖ パブリケーションを追加する場合は、次の手順に従います。

1 左ウィンドウ枠で、[パブリケーション] フォルダを選択します。

2 [ファイル] メニューから [新規] ー [パブリケーション] を選択します。

[パブリケーション作成] ウィザードが表示されます。

3 パブリケーションに **SalesRepData** という名前を付けます。

4 [テーブル] タブで、[使用可能なテーブル] のリストから [**SalesRep**] を選択します。[追加] をクリックします。

右側の [選択したテーブル] リストに、そのテーブルが表示されます。

5　[使用可能なテーブル] のリストから [**Customer**] を選択します。[追加] をクリックします。

6　[SUBSCRIBE BY 制限] タブで Customer テーブルを選択し、**rep_key** 式を入力します。

7　パブリケーションを作成するには [完了] をクリックしてください。

サブスクリプションの追加

パブリケーションに加えられた変更を受信する各ユーザ ID には、そのパブリケーションに対する「**サブスクリプション**」が必要です。サブスクリプションは、有効なリモート・ユーザに対してのみ作成できます。リモート・データベース・ユーザ **field_user** に、**SalesRepData** パブリケーションに対するサブスクリプションを追加する必要があります。

❖ サブスクリプションを追加する場合は、次の手順に従います。

1　[パブリケーション] フォルダを開き、左ウィンドウ枠で **SalesRepData** パブリケーションを選択します。

2　右ウィンドウ枠で、[SQL Remote サブスクリプション] タブをクリックします。

3　[ファイル] メニューで [新規] ー [SQL Remote サブスクリプション] を選択します。

[SQL Remote サブスクリプション作成] ウィザードが表示されます。

4　**field_user** 用にサブスクリプションを作成することを選択します。サブスクリプション値 **rep1** を入力し、[完了] をクリックします。

サブスクリプション値は、パブリケーションの Subscribe By 式に一致する式です。以降の手順で、ユーザ ID **field_user** に **rep1** の値 rep_key が割り当てられます。

以上で統合データベースの設定が完了しました。

Sybase Central でのリモート・データベースの設定

リモート・データベースを作成し、メッセージの送受信を行い SQL Remote 設定に組み込まれるように設定する必要があります。

統合データベースと同じように、リモート・データベースには、出力メッセージのソースを識別するためのパブリッシャ (この例では **field_user** ユーザ ID) が必要です。また、**hq_user** を、統合パーミッションを持つユーザとして識別させる必要があります。リモート・データベースでは、**SalesRepData** パブリケーションを作成する必要があり、**hq_user** ユーザ ID 用に作成されたサブスクリプションが必要です。

リモート・データベースを統合データベースと「**同期**」させる必要もあります。つまり、レプリケーションを開始するには現在のデータのコピーが必要です。この例では、いまのところパブリケーションにデータはありません。

データベース抽出ユーティリティを使用すると、サブスクリプションと必要なユーザ ID がそろったリモート・データベースを作成するのに必要な手順を、すべて実行できます。

リモート・ユーザ **field_user** 用に、統合データベースからデータベースを抽出する必要があります。

❖ データベースを抽出する場合は、次の手順に従います。

1. **HQ** データベースに接続します。
2. データベースを右クリックし、ポップアップ・メニューで [データベースの抽出] を選択します。
3. 次のオプションを指定して、HQ データベースを抽出するように選択します。
   - 独立性レベル 3 で抽出します。

- ユーザ **field_user** に対して [自動的にサブスクリプションを開始] を選択します。
- [新しいデータベースへの抽出と再ロード] を選択します。
- ファイルを再ロードするロケーションは、デフォルトの設定のままにしておきます。
- 構造とデータの両方を抽出するよう選択します。
- 完全に修飾されたパブリケーション定義は抽出しないように選択します。
- ファイル *c:¥tutorial¥field.db* としてデータベースを作成し、[完了] をクリックしてリモート・データベースを作成します。

本来、SQL Remote 設定では、リモート・データベース **field** が、Adaptive Server Anywhere サーバと必要なクライアント・アプリケーションと共に、それを使用するコンピュータにロードされている必要があります。このチュートリアルでは、データベースの場所を変更しないで、Interactive SQL を使用してデータの入力とレプリケートを行います。

field データベースに DBA として接続し、すべてのデータベース・オブジェクトが作成されていることを確認してください。これらのデータベース・オブジェクトには **SalesRep** テーブルと **Customer** テーブル、**SalesRepData** パブリケーション、統合データベース用のサブスクリプションがあります。

次の手順

以上でこのシステムは、レプリケーションの準備が完了しました。

次の手順である、データの挿入とレプリケーションの詳細については、「データのレプリケーションの開始」59 ページの項を参照してください。

チュートリアル：Interactive SQL と dbxtract を使用した Adaptive Server Anywhere レプリケーション

次の項からは、コマンド・ライン・ツールを使用するユーザや、Sybase Central が裏で実行する動作について知りたいユーザを対象に、簡単な SQL Remote レプリケーション・システムの設定方法を説明するチュートリアルです。

このチュートリアルでは、SQL Remote の管理のために Interactive SQL から実行できる SQL 文について説明します。また、統合データベースからリモート・データベースを抽出するための *dbxtract* コマンド・ライン・ユーティリティの使用方法についても説明します。

このチュートリアルでは統合データベースの DBA として操作を行い、ファイル共有メッセージ・リンクを使用して簡単なレプリケーション・システムを設定します。例として、2 つのテーブルからなる営業戦力自動化システムの基本モデルを作成します。1 つのテーブルには営業担当のリストが、もう 1 つのテーブルには顧客のリストが入ります。これらのテーブルは 1 つの統合データベースと 1 つのリモート・データベースを含む設定にレプリケートされます。この例のシステムは、1 台のコンピュータにインストールできます。

レプリケーション・チュートリアルの準備

この項では、このチュートリアルの準備に必要な手順について説明します。手順とは次のようなものです。

- チュートリアルに必要なディレクトリとデータベースを作成する。
- 統合データベースにテーブルを追加する。

❖ チュートリアル用のデータベースとディレクトリを作成する場合は、次の手順に従います。

1. このチュートリアルで作成するファイルを置くディレクトリを作成します。たとえば *c:¥tutorial* とします。

```
mkdir c:¥tutorial
```

2　このチュートリアルでは、*hq.db* という統合データベースと、*field.db* というリモート・データベースの 2 つのデータベースを使用します。チュートリアル・ディレクトリに移動し、コマンド・ラインで次の文を使用して、これらのデータベースを作成します。

```
dbinit hq.db
 dbinit field.db
```

3　レプリケーション・システム内の 2 つのユーザ ID 用に、それぞれサブディレクトリを作成します。コマンド・ラインで次の文を使用して、これらのサブディレクトリを作成します。

```
mkdir c:¥tutorial¥hq
 mkdir c:¥tutorial¥field
```

次の手順では、統合データベースに 1 組のテーブルを追加します。

❖ 統合データベースにテーブルを追加する場合は、次の手順に従います。

1　Interactive SQL から *hq.db* に接続します。ユーザ ID には **DBA**、パスワードには **SQL** を使用します。

2　次の CREATE TABLE 文を実行して、**SalesRep** テーブルを作成します。

```
CREATE TABLE SalesRep (
     rep_key CHAR(12) NOT NULL,
     name CHAR(40) NOT NULL,
     PRIMARY KEY ( rep_key )
 );
```

3　次の CREATE TABLE 文を実行して、**Customer** テーブルを作成します。

```
CREATE TABLE Customer (
     cust_key CHAR(12) NOT NULL,
     name CHAR(40) NOT NULL,
     rep_key CHAR(12) NOT NULL,
     FOREIGN KEY REFERENCES SalesRep,
     PRIMARY KEY ( cust_key )
 );
```

次のチュートリアルに進みましょう。

統合データベースの設定

チュートリアルのこの項では、簡単なレプリケーション・システムの統合データベースを設定する方法について説明します。

この作業を行うには、DBA 権限が必要です。

SQL Remote のメッセージ・タイプの作成

レプリケーションの一部として送信されるすべてのメッセージは、メッセージ・タイプを使用します。メッセージ・タイプの記述は、次の 2 つの部分で構成されます。

- SQL Remote でサポートされているメッセージ・リンク。このチュートリアルでは、FILE リンクを使用します。
- 出力メッセージのソースを識別する、メッセージ・リンクのアドレス。

❖ メッセージ・タイプを作成する場合は次の手順に従います。

- Interactive SQL で、次の文を使用してメッセージ・タイプ file を作成します。

```
CREATE REMOTE MESSAGE
 TYPE file
 ADDRESS 'hq'
```

ファイル・リンクに使うアドレス (**hq**) は、メッセージが入るファイルがあるディレクトリです。これは、SQLRemote 環境変数、またはレジストリ・エントリからの相対アドレスです。アドレスを設定しな

かった場合は、Message Agent が実行されているディレクトリからの相対アドレスが設定されます。このアドレスが正しく解釈されるように、チュートリアル・ディレクトリから Message Agent を実行してください。

SQLRemote 値の設定方法については、「メッセージ・タイプ制御パラメータの設定」274 ページを参照してください。

統合データベースでの PUBLISH と REMOTE の付与

SQL Remote でサポートされる階層レプリケーション・システムでは、各データベースのすぐ上の階層に 1 個の統合データベースがあり、すぐ下の階層に多くのデータベース (リモート・データベース) があります。

PUBLISH パーミッションは出力メッセージの現在のデータベースを識別し、REMOTE パーミッションは現在のデータベースからメッセージを受信する各データベースを識別します。

これらのパーミッションは、DBA 権限を持つユーザだけが付与できます。これらの例を実行するには、Interactive SQL ユーティリティを使用して **hq** に接続します。ユーザ ID には **DBA**、パスワードには **SQL** を使用します。

GRANT PUBLISH を実行して、出力メッセージを識別

レプリケーション・システム内で他のデータベースに変更を配布する各データベースは、パブリッシャ・データベースです。レプリケーション・システムでデータベースに変更を配布する各データベースは、1 つのユーザ ID で識別されます。この ID は、GRANT PUBLISH 文を使用してデータベースに設定します。この項では、統合データベース (*hq.db*) へのパーミッションの設定について説明します。

❖ **データベースにパブリッシャを作成する場合は、次の手順に従います。**

- Interactive SQL を使用してデータベースに接続し、次の文を入力します。

```
GRANT CONNECT
 TO hq_user
 IDENTIFIED BY hq_pwd ;

 GRANT PUBLISH TO hq_user ;
```

CURRENT PUBLISHER 特殊定数を使用すると、データベースのパブリッシュ・ユーザ ID をいつでもチェックできます。

```
SELECT CURRENT PUBLISHER
```

メッセージを送信する各データベースに対して GRANT REMOTE を実行

各リモート・データベースは、GRANT REMOTE 文を使用して識別されます。リモート・データベースがパーソナル・データベース・サーバであっても、多くのユーザがあるネットワーク・サーバであっても、統合データベースに対してそのデータベースを示す 1 つのユーザ ID が必要です。

モバイル・ワークグループの設定で、リモート・ユーザがすでに統合データベースのユーザの場合があります。その場合、新しくユーザを追加する必要はありませんが、リモート・ユーザとして設定する必要はあります。

GRANT REMOTE 文では、アドレスの他に、メッセージを受信者に送信するときに使用されるメッセージ・システムを識別します。

❖ リモート・ユーザを追加する場合は、次の手順に従います。

- Interactive SQL を使用してデータベースに接続し、次の文を実行します。

  ```
  GRANT CONNECT TO field_user
   IDENTIFIED BY field_pwd ;

   GRANT REMOTE TO field_user
   TYPE file ADDRESS 'field' ;
  ```

  アドレス文字列には、メッセージを置くために使用する **field_user** のディレクトリを、一重引用符で囲んで指定します。これは、SQLRemote 環境変数、またはレジストリ・エントリからの相対アドレスです。アドレスを設定しなかった場合は、Message Agent が実行されているディレクトリからの相

対アドレスが設定されます。このアドレスが正しく解釈されるように、チュートリアル・ディレクトリから Message Agent を実行してください。

SQLRemote 値の設定方法については、「メッセージ・タイプ制御パラメータの設定」274 ページを参照してください。

パブリケーションとサブスクリプションの作成

パブリケーションは、CREATE PUBLICATION 文を使用して作成します。この文はデータ定義言語文で、DBA 権限が必要です。このチュートリアルでは、**hq** データベースに接続し (ユーザ ID には **DBA**、パスワードには **SQL** を使用します)、パブリケーションを作成します。

統合データベースでのパブリケーションの設定

SalesRepData というパブリケーションを作成します。これは、**SalesRep** のすべてのローと、**Customer** テーブルのいくつかのローをレプリケートします。

❖ パブリケーションを作成する場合は、次の手順に従います。

- Interactive SQL からデータベースに接続し、次の文を実行します。

```
CREATE PUBLICATION SalesRepData (
     TABLE SalesRep,
     TABLE Customer SUBSCRIBE BY rep_key
 )
```

サブスクリプションの設定

パブリケーションに加えられた変更を受信する各ユーザ ID には、サブスクリプションが必要です。サブスクリプションは、REMOTE パーミッションを持つユーザに対してのみ作成できます。GRANT REMOTE 文には、メッセージを送信するときに使用するアドレスが組み込まれています。

❖ **サブスクリプションを作成する場合は、次の手順に従います。**

- Interactive SQL からデータベースに接続し、次の文を実行します。

```
CREATE SUBSCRIPTION
 TO SalesRepData ('rep1')
 FOR field_user ;
```

値 **rep1** は、**SalesRep** テーブルのユーザ **field_user** に与える **rep_key** の値です。

完全な CREATE SUBSCRIPTION 文を使用すると、サブスクリプション内のデータを制御できます。つまりユーザがパブリケーション内の一部のローだけを受信する、ということが可能になります。詳細については、「CREATE SUBSCRIPTION 文」455 ページを参照してください。

CREATE SUBSCRIPTION 文はサブスクライバを識別し、それらが何を受信するかを定義します。ただしデータの同期は行わず、メッセージ送信の開始も行いません。

リモート・データベースの設定

メッセージの送受信を行い SQL Remote 設定に組み込まれるように、リモート・データベースを設定する必要があります。統合データベースと同じように、リモート・データベースには、出力メッセージのソースを識別するため CURRENT PUBLISHER が必要です。また、統合データベースをサブスクライバとして識別させる必要があります。リモート・データベースでは、パブリケーションを作成する必要があり、統合データベース用に作成されたサブスクリプションも必要です。リモート・データベースを統合データベースと「**同期**」させる必要もあります。つまり、レプリケーションを開始するには現在のデータのコピーが必要です。

dbxtract ユーティリティを使用すると、サブスクリプションと必要なユーザ ID がそろったリモート・データベースを作成するのに必要な手順を、すべて実行できます。

リモート・データベース情報の抽出

hq データベースを実行したままで、チュートリアル・ディレクトリに移動します。

次のコマンドをシステム・コマンド・ラインで (すべて 1 行に) 入力し、ユーザ **field_user** 用のデータベースを統合データベースから抽出します。

```
dbxtract -v -c "dbn=hq;uid=dba;pwd=sql" c:¥tutorial
field_user
```

-v コマンド・ライン・スイッチを使用すると、より冗長な出力が生成されます。これは開発中に使用すると便利です。

このコマンドは、**hq** データベースが現在デフォルト・サーバで実行されていることを前提としています。データベースが実行されていない場合は、次のように接続文字列にデータベース・ファイル・パラメータを入力します。

```
dbf=hq.db
```

これはデータベース名パラメータ **dbn** の代わりです。

dbxtract ユーティリティとそのオプションの詳細については、「抽出ユーティリティ」391 ページを参照してください。

dbxtract コマンドは、*reload.sql* という SQL コマンド・ファイルを現在のディレクトリに、データ・ファイルを *c:¥tutorial* ディレクトリに作成します。リモート・ユーザに対するサブスクリプションも開始します。

次の手順では、これらのファイルをリモート・データベースにロードします。

リモート・データベース情報のロード

❖ **データベース情報をロードする場合は、次の手順に従います。**

1 チュートリアル・ディレクトリで、Interactive SQL からリモート・データベース *field.db* に接続します。ユーザ ID には **DBA**、パスワードには **SQL** を使用します。

2 *reload.sql* コマンド・ファイルを実行します。

```
READ C:¥tutorial¥reload.sql
```

reload.sql コマンド・ファイルは、次のタスクを実行します。

- リモート・データベースでメッセージ・タイプを作成する。
- リモート・データベースと統合データベースに、PUBLISH パーミッションと REMOTE パーミッションをそれぞれ与える。
- データベースにテーブルを作成する。抽出の前にテーブルにデータが入っていた場合、コマンド・ファイルはレプリケートされたテーブルにそのデータのコピーを入れる。
- レプリケートするデータを示すパブリケーションを作成する。
- 統合データベース用のサブスクリプションを作成し、開始する。

field データベースに DBA として接続している間に、次の文を実行して、テーブルが作成されていることを確認します。

```
SELECT * FROM SalesRep ;

SELECT * FROM Customer ;
```

次の手順

以上でこのシステムは、レプリケーションの準備が完了しました。

次の手順である、データの挿入とレプリケーションの詳細については、「データのレプリケーションの開始」59 ページの項を参照してください。

データのレプリケーションの開始

ここまでの作業で、レプリケーション・システムが使えるようになりました。この項では、統合データベースからリモート・データベースに、また、リモート・データベースから統合データベースに、データをレプリケートします。

統合データベースでのデータの入力

最初に、いくつかのデータを統合データベースに入力します。

❖ 統合データベースでデータを入力する場合は、次の手順に従います。

1. Interactive SQL ユーティリティから統合データベース **hq** に接続します。ユーザ ID には **DBA**、パスワードには **SQL** を使用します。
2. 次の文を実行して、**SalesRep** テーブルに 2 つのローを挿入してコミットします。

```
INSERT INTO SalesRep (rep_key, name)
 VALUES ('rep1', 'Field User') ;
 INSERT INTO SalesRep (rep_key, name)
 VALUES ('rep2', 'Another User') ;
 COMMIT ;
```

3. 次の文を実行して、**Customer** テーブルに 2 つのローを挿入してコミットします。

```
INSERT INTO Customer (cust_key, name, rep_key)
 VALUES ('cust1', 'Ocean Sports', 'rep1' ) ;
 INSERT INTO Customer (cust_key, name, rep_key)
 VALUES ('cust2', 'Sports Plus', 'rep2' ) ;
 COMMIT ;
```

4. 次の文を実行して、データが入力されていることを確認します。

```
SELECT *
 FROM SalesRep;

 SELECT *
 FROM Customer;
```

次の手順では、関連のあるローをリモート・データベースに送信します。

統合データベースからのデータの送信

リモート・データベースにローを送信するには、統合データベースで Message Agent を実行してください。*dbremote* プログラムが Adaptive Server Anywhere の Message Agent です。

❖ リモート・データベースにデータを送信する場合は、次の手順に従います。

1. コマンド・プロンプトから、チュートリアル・ディレクトリに移動します。次に例を示します。

   ```
   > c:
    > cd c:¥tutorial
   ```

2. コマンド・ラインで次の文を入力して、統合データベースに対して Message Agent を実行します。

   ```
   dbremote -c "dbn=hq;uid=dba;pwd=sql"
   ```

   このコマンド・ラインは、**hq** データベースが現在デフォルト・サーバで実行されていることを前提としています。データベースが実行されていない場合は、**dbn** パラメータの代わりに **dbf** パラメータを、データベース・ファイル名とともに指定します。

   dbremote オプションの詳細については、「Message Agent」376 ページ を参照してください。

3　メッセージの送信が完了したら、[Message Agent] ウィンドウで [シャットダウン] をクリックして Message Agent を停止します。すべての処理が完了すると、[Message Agent] ウィンドウに「実行が完了しました」というメッセージが表示されます。

リモート・データベースでのデータの受信

リモート・データベースで insert 文を受信するには、リモート・データベースで Message Agent である *dbremote* を実行します。

❖ リモート・データベースでデータを受信する場合は、次の手順に従います。

1　コマンド・プロンプトから、チュートリアル・ディレクトリに移動します。次に例を示します。

```
> c:
 > cd c:¥tutorial
```

2　コマンド・ラインで次の文を入力して、**field** データベースに対して Message Agent を実行します。

```
dbremote -c "dbn=field;uid=dba;pwd=sql"
```

このコマンド・ラインは、**field** データベースが現在デフォルト・サーバで実行されていることを前提としています。

dbremote オプションの詳細については、「Message Agent」376 ページ を参照してください。

3　メッセージの処理が完了したら、[Message Agent] ウィンドウで [シャットダウン] をクリックして Message Agent を停止します。すべての処理が完了すると、[Message Agent] ウィンドウに「実行が完了しました」というメッセージが表示されます。

Message Agent の実行中は、[Message Agent] ウィンドウにステータス情報が表示されます。実際の設定では、この情報をログ・ファイルに出力し記録を保存できます。Message Agent がまず **hq** からメッセージ

を受信し、そのあとでメッセージを送信するのがわかります。このリターン・メッセージには、レプリケーションの更新の受信に成功したことの確認が含まれています。このような確認は、メッセージ・システム・エラーが発生した場合でもメッセージの配信を保証する、SQL Remote メッセージ・トラッキング・システムの一部です。

データの着信の確認

Interactive SQL を使用してリモートの **field** データベースに接続し、**SalesRep** テーブルと **Customer** テーブルを検査して、どのローが受信されたのかを確認してください。

❖ データが着信したことを確認する場合は、次の手順に従います。

1. Interactive SQL を使用して、field データベースに接続します。
2. 次の文を実行し、**SalesRep** テーブルを検査します。

   ```
   SELECT * FROM SalesRep
   ```

   統合データベースで入力したローが両方とも **SalesRep** テーブルにあります。これは、**SalesRepData** パブリケーションに **SalesRep** テーブルからのすべてのデータが含まれていたからです。
3. 次の文を実行し、**Customer** テーブルを検査します。

   ```
   SELECT * FROM Customer
   ```

   Customer テーブルには、統合データベースで入力したローが 1 つだけ (Ocean Sports) しかありません。これは、**SalesRepData** パブリケーションに含まれているのが、サブスクリプションが作成されている営業担当に割り当てられた顧客だけだからです。

リモート・データベースから統合データベースへのレプリケーション

リモート・データベースでデータを入力し、それを統合データベースに送信してみてください。ここではその概略だけを示します。

❖ リモート・データベースから統合データベースにデータをレプリケートする場合は、次の手順に従います。

1　Interactive SQL から **field** データベースに接続します。

2　次の文を実行し、リモート・データベースでローを挿入します。

```
INSERT INTO Customer (cust_key, name, rep_key)
 VALUES ('cust3', 'North Land Trading', 'rep1')
```

3　次の文を実行し、この挿入をコミットします。

```
COMMIT;
```

4　*field.db* データベースの実行中に、コマンド・ラインから *dbremote* ユーティリティを実行して統合データベースにメッセージを送信します。

```
dbremote -c "dbn=field;uid=dba;pwd=sql"
```

5　*hq.db* データベースの実行中に、コマンド・ラインから *dbremote* ユーティリティを実行して統合データベースでメッセージを受信します。

```
dbremote -c "dbn=hq;uid=dba;pwd=sql"
```

6　統合データベースに接続します。次の文を実行し、**Customer** テーブルを表示します。

```
SELECT *
 FROM Customer
```

cust_key	name	rep_key
cust1	Ocean Sports	rep1
cust2	Sports Plus	rep2
cust3	North Land Trading	rep1

この簡単な例では、プライマリ・キーの値が重複したエントリを防ぐようにはなっていません。SQL Remote では、このような重複エントリを防ぐことができます。詳細については、「SQL Remote 設計」の章を参照してください。

サンプル・パブリケーション

コマンド・ファイル *salespub.sql* には、サンプル・データベースにパブリケーションを作成する一連の文が入っています。このパブリケーションには、このチュートリアルのいくつかのポイントが、より詳しく示されています。

❖ サンプル・データベースにパブリケーションを追加する場合は、次の手順に従います。

1　Interactive SQL からサンプル・データベースに接続します。

2　[SQL 文] ウィンドウ枠で、次の文を実行します。

```
READ path¥scripts¥salespub.sql
```

この場合、*path* は SQL Anywhere ディレクトリです。

salespub.sql パブリケーションは、サンプル・データベースのテーブルのいくつかにカラムを追加し、パブリケーションとサブスクリプションを作成します。また、更新の競合が起こった場合にそれを解決するためのトリガも追加します。

第 5 章

Adaptive Server Enterprise ユーザのためのチュートリアル

この章の内容

この章は、Adaptive Server Enterprise データベースと Adaptive Server Anywhere データベースの間で、簡単な SQL Remote レプリケーション・システムを最初から設定する際のチュートリアルになっています。

概要

この章は、SQL Remote インストール環境の設定を行うためのチュートリアルです。インストール環境では、Adaptive Server Enterprise データベース (統合データベース) と Adaptive Server Anywhere データベース (リモート・データベース) の間で、データをレプリケートします。

目的

このチュートリアルでは、統合 Adaptive Server Enterprise データベースのシステム管理者として、簡単なレプリケーション・システムを設定します。このレプリケーション・システムは、2 つのテーブルからなる簡単な販売データベースで構成します。

統合データベースはすべてのデータベースを抱合しています。リモート・データベースは同一のデータで構成されますが、いくつかのローは他のリモート・データベースのテーブルに含まれます。

このチュートリアルの手順は次のとおりです。

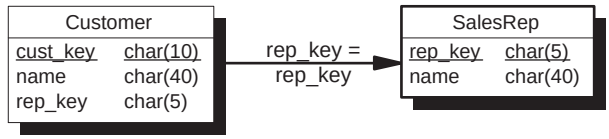
- Adaptive Server Enterprise サーバ上に統合データベースを作成する。
- 1 つの Adaptive Server Anywhere リモート・データベースを含む、ファイル共有レプリケーション・システムを作成する。
- 2 つのデータベース間で、データをレプリケートする。

データベース

このチュートリアルでは、2 つのテーブルで構成される簡単なデータベースを使用します。1 つのテーブルには営業担当の情報が、もう 1 つのテーブルには顧客の情報が入っています。これらのテーブルは実際のデータベースで使用するものよりずっと単純なので、レプリケーションに関する重要な問題だけに注目できます。

データベースのスキーマ

このチュートリアルで使うデータベースのスキーマを次の図に示します。



ここで注意すべき特徴には、次のものがあります。

- 各営業担当は SalesRep テーブルの 1 つのローで表されています。
- 各顧客は Customer テーブルの 1 つのローで表されています。
- 各顧客は 1 人の営業担当に割り当てられており、この割り当ては Customer テーブルから SalesRep テーブルへの外部キーとしてデータベースに組み込まれています。Customer テーブルと SalesRep との関係は、多対 1 です。

データベース内のテーブル

テーブルの詳細は次のとおりです。

テーブル	説明

SalesRep	会社の営業担当それぞれに 1 つのロー。SalesRep テーブルには、次のカラムがあります。
	• **rep_key**　各営業担当の識別子。これがプライマリ・キーです。
	• **name**　各営業担当の名前。
	このテーブルを作成する SQL 文は次のとおりです。
	`CREATE TABLE SalesRep (`
	`rep_key CHAR(12) NOT NULL,`
	`name CHAR(40) NOT NULL,`
	`PRIMARY KEY (rep_key)`
	`)`

テーブル	説明

Customer	会社と取引がある顧客それぞれに 1 つのロー。Customer テーブルには、次のカラムがあります。
	
	• **cust_key** 各顧客の識別子。これがプライマリ・キーです。
	• **name** 各顧客の名前。
	• **rep_key** 取引を行う営業担当者の識別子。これが SalesRep テーブルへの外部キーです。
	
	このテーブルを作成する SQL 文は次のとおりです。
	
	`CREATE TABLE Customer (`
	`cust_key CHAR(12) NOT NULL,`
	`name CHAR(40) NOT NULL,`
	`rep_key CHAR(12) NOT NULL,`
	`FOREIGN KEY ( rep_key )`
	`REFERENCES SalesRep (rep_key ),`
	`PRIMARY KEY (cust_key)`
	`)`

レプリケーションの目標

このレプリケーションの設計上の目的は、各営業担当に次の情報を提供することです。

- 完全な SalesRep テーブル
- 割り当てられている顧客

このチュートリアルでは、SQL Remote を使用してこの目的を達成する方法について説明します。

チュートリアル :Adaptive Server Enterprise のレプリケーション

以下の項では、簡単な SQL Remote レプリケーション・システムを設定する方法を説明します。

このチュートリアルでは、SQL Remote の設定と管理に使用するストアド・プロシージャについて説明します。また、統合データベースからリモート・データベースを抽出するための *ssxtract* ユーティリティと、レプリケーション・システムのデータベース間で情報を送信するための Message Agent の使用方法についても説明します。

このチュートリアルでは統合データベースの管理者として操作を行い、ファイル共有メッセージ・リンクを使用して簡単なレプリケーション・システムを設定します。例として、2 つのテーブルからなる営業戦力自動化システムの基本モデルを作成します。1 つのテーブルには営業担当のリストが、もう 1 つのテーブルには顧客のリストが入ります。これらのテーブルは 1 つの統合データベースと 1 つのリモート・データベースを含む設定にレプリケートされます。この例のシステムは、1 台のコンピュータにインストールできます。

第一段階

ログイン名とパスワードの作成

チュートリアルに従って作業するには、Adaptive Server Enterprise サーバに対するシステム管理者の権限が必要です。このチュートリアルでは、ログイン名が 2 文字の単語 **sa** で、パスワードが **sysadmin** とします。

ここでは Adaptive Server Enterprise の *isql* ユーティリティを使用します。前述のログイン名とパスワードで、次のコマンド・ラインを使用して Adaptive Server Enterprise サーバに接続します。

```
isql -S server-name -U sa -P sysadmin
```

server-name は、接続する Adaptive Server Enterprise サーバの名前です。

このチュートリアルを開始する前に、適切なログイン ID を持っていて、サーバに接続できることを確認してください。

データベースの作成

チュートリアルのデータベースで必要なテーブルとデータを入れるのに十分な領域を確保して、Adaptive Server Enterprise サーバにデータベース **hq** を作成します。領域は 4 MB あれば十分です。

❖ データベースを作成する場合は、次の手順に従います。

1 *isql* を使用して、システム管理者権限を持つユーザとしてサーバに接続します。

```
isql -S server-name -U sa -P sysadmin
```

2 master データベースを使用します。

```
use master
 go
```

3 **hq** というデータベースを作成します。この例では、2 つの異なるデバイスにある、5 MB のログを含む 5 MB のデータベースを使用します。

```
create database hq
 on database_device = 5
 log on log_device = 5
 go
```

データベースを作成し、そのデータベースに領域を割り当てる方法については、Adaptive Server Enterprise のマニュアルを参照してください。

SQL Remote のインストール

hq データベースに SQL Remote をインストールする必要があります。

❖ hq データベースに SQL Remote をインストールする場合は、次の手順に従います。

1 使用しているシステム管理者ログイン名がデフォルトのデータベースとして **hq** データベースを持っていない場合、インストール・ディレクトリから *ssremote.sql* スクリプトのバックアップ・コピーを作成して、スクリプトの最初に次の 2 行を追加します。

```
use hq
 go
```

2　チュートリアル・ディレクトリに移動します。次に *isql* を使用して **hq** データベースを使用しているサーバに接続し、SQL Remote インストール・ディレクトリから *ssremote.sql* スクリプトを実行します。次のコマンドを 1 行で入力してください。

```
isql -S server-name -U sa -P sysadmin -I
ssremote.sql
```

3　使用しているシステム管理者ログイン名がデフォルトのデータベースとして **hq** データベースを持っていない場合、インストール・ディレクトリから *stableq.sql* スクリプトのバックアップ・コピーを作成して、スクリプトの最初に次の 2 行を追加します。

```
use hq
 go
```

4　*isql* を使用して、**hq** データベースを使用しているサーバに接続し、SQL Remote インストール・ディレクトリから *stableq.sql* スクリプトを実行します。次のコマンドを 1 行で入力してください。

```
isql -S server-name -U sa -P sysadmin -I stableq.sql
```

メッセージ用のディレクトリの作成

このチュートリアルで作成されるファイルを保存するディレクトリを作成します。次に例を示します。

```
mkdir c:¥tutorial
```

このチュートリアルを親ディレクトリとして、その下にレプリケーション・システムの 2 つのユーザ用に、ディレクトリをそれぞれ作成します。

```
mkdir c:¥tutorial¥hq
 mkdir c:¥tutorial¥field
```

次の手順では、統合データベースに 1 組のテーブルを追加します。

❖ **統合データベースにテーブルを追加する場合は、次の手順に従います。**

1 *isql* から **hq** データベースに、システム管理者として接続します。

2 **hq** データベースを使用します。

```
use hq
 go
```

3 次の文で、**SalesRep** テーブルを作成します。

```
create table SalesRep (
     rep_key char(12) not null,
     name char(40) not null,
     primary key (rep_key) )
 go
```

4 次の文で、**Customer** テーブルを作成します。

```
create table Customer (
     cust_key char(12) not null,
     name char(40) not null,
     rep_key char(12) not null,
     primary key (cust_key) )
 go
```

5 **Customer** テーブルを変更して、**SalesRep** テーブルに外部キーを追加します。

```
alter table Customer
 add foreign key
 ( rep_key ) references SalesRep
 go
```

次のチュートリアルに進みましょう。

統合データベースの設定

チュートリアルのこの項では、簡単なレプリケーション・システムの統合データベースを準備する方法について説明します。

レプリケーションに使用する統合データベースを準備する手順は次のとおりです。

1. レプリケーションに使用するメッセージ・タイプを作成します。
2. 出力メッセージのソースを識別できるように、ユーザ ID に PUBLISH パーミッションを付与します。
3. メッセージを受信するすべてのユーザ ID に REMOTE パーミッションを付与します。
4. レプリケートするデータを記述するパブリケーションを作成します。
5. パブリケーションの受信側を記述するサブスクリプションを作成します。

これらの作業を行うには、システム管理者の権限が必要です。

メッセージ・リンクとアドレスの作成

このチュートリアルでは、共有ファイル・リンクを使用してメッセージを交換します。メッセージ・タイプ FILE を作成して、統合データベースのパブリッシャのアドレスを入力します。

❖ メッセージ・タイプを作成する場合は次の手順に従います。

- 統合データベースのパブリッシャのアドレスとして hq を使用し、**sp_remote_type** ストアド・プロシージャを実行します。

```
sp_remote_type file, hq
 go
```

ファイル・リンクに使うアドレス (**hq**) は、メッセージが入るファイルがあるディレクトリです。これは、SQLRemote 環境変数、またはレジストリ・エントリからの相対アドレスです。アドレスを設定しなかった場合は、Message Agent が実行されているディレクトリからの相対アドレスが設定されます。このアドレスが正しく解釈されるように、チュートリアル・ディレクトリから Message Agent を実行してください。

SQLRemote 値の設定方法については、「メッセージ・タイプ制御パラメータの設定」274 ページを参照してください。

メッセージ・タイプを定義すると、必要なユーザを作成できます。

必要なユーザとパーミッションの作成

SQL Remote のインストールには、1 組のユーザとパーミッションが必要です。このチュートリアルで必要なものは次のとおりです。

- 名前が **field_user** の、リモート・ユーザまたはサブスクライバ
- **hq_user** というパブリッシャのユーザ名

この項では、各ユーザを作成し、必要なパーミッションを割り当てるのに必要な手順について説明します。

❖ パブリッシャを作成する場合は、次の手順に従います。

1 デフォルトのデータベースが **hq** で、システム管理者のアクセス権を持つ、**hq_user** というログイン名を追加します。

```
exec sp_addlogin hq_user, hq_pwd, hq
 go
 exec sp_role 'grant', sa_role, hq_user
 go
```

2 このログイン名を、hq データベースにユーザ名として追加します。

```
use hq
 go
 exec sp_adduser hq_user
 go
```

3 このユーザを hq データベースのパブリッシャに設定します。

```
exec sp_publisher hq_user
 go
```

リモート・ユーザの追加

統合データベースでは、各リモート・データベースは REMOTE パーミッションを持つユーザ ID で識別されます。リモート・データベースがシングルユーザのサーバであっても、多くのユーザがいるデータベース・サーバであっても、統合データベースに対してそのデータベースを示す 1 つのユーザ ID が必要です。

モバイル・ワークグループの設定で、リモート・ユーザがすでに統合データベースのユーザの場合があります。その場合、新しくユーザを追加する必要はありませんが、リモート・ユーザとして設定する必要はあります。

リモート・ユーザをデータベースに追加するときに、各ユーザが使用するメッセージ・システムとそのメッセージ・システムにおける各ユーザのアドレスを、それぞれのデータベース・ユーザ ID と共に格納する必要があります。

❖ サブスクライバを作成する場合は、次の手順に従います。

1　リモート・ユーザに使用できるログイン名がない場合、ログインを追加します。

```
exec sp_addlogin field_user, field_pwd, hq
 go
```

2　**hq** データベースにユーザを追加します。

```
exec sp_adduser field_user
 go
```

3　ユーザに、REMOTE パーミッションを付与します。ユーザ名に **field_user**、メッセージ・タイプに **file**、アドレスには適切なディレクトリを使用して、**sp_grant_remote** ストアド・プロシージャを実行します。

```
exec sp_grant_remote field_user, file, field
 go
```

パブリッシャ・アドレスと同様に、リモート・ユーザ (**field**) のアドレスは、SQLRemote 環境変数またはレジストリ・エントリからの相対ディレクトリです。アドレスを設定しなかった場合は、Message Agent が実行されているディレクトリから

の相対アドレスが設定されます。このアドレスが正しく解釈されるように、チュートリアル・ディレクトリから Message Agent を実行してください。

SQLRemote 値の設定方法については、「メッセージ・タイプ制御パラメータの設定」274 ページを参照してください。

パブリケーションとサブスクリプションの作成

最後の処理は、レプリケートするデータの定義です。この定義は、使用可能なデータを定義するパブリケーションを最初に作成します。次に、ユーザが共有するデータを定義する **field_user** のサブスクリプションを作成します。

Adaptive Server Enterprise では、空のパブリケーションを作成する **sp_create_publication** プロシージャと、このプロシージャにアーティクルを追加する **sp_add_article** プロシージャを実行して、パブリケーションとサブスクリプションを作成します。また、各テーブルをレプリケーション用にマーク付けしてから、テーブルをパブリケーションに保存してください。

❖ パブリケーションを作成する場合は、次の手順に従います。

1 空のパブリケーションを作成します。

```
exec sp_create_publication SalesRepData
 go
```

2 **SalesRep** テーブルと **Customer** テーブルの両方を、パブリケーション用にマーク付けします。

```
exec sp_add_remote_table SalesRep
 go
 exec sp_add_remote_table Customer
 go
```

3 **SalesRep** テーブル全体を、**SalesRepData** パブリケーションに追加します。

```
exec sp_add_article SalesRepData, SalesRep
 go
```

4　テーブルを分割する **rep_key** カラムを使用して、**SalesRepData** パブリケーションに Customer テーブルを追加します。次の文は、go 以外を 1 行で入力してください。

```
exec sp_add_article SalesRepData, Customer, NULL,
'rep_key'
 go
```

サブスクリプションの追加

パブリケーションに加えられた変更を受信する各ユーザ ID には、そのパブリケーションに対する「**サブスクリプション**」が必要です。サブスクリプションは、有効なリモート・ユーザに対してのみ作成できます。リモート・データベース・ユーザ **field_user** に、**SalesRepData** パブリケーションに対するサブスクリプションを追加する必要があります。

❖ サブスクリプションを作成する場合は、次の手順に従います。

- **field_user** の **SalesRepData** にサブスクリプションを作成します。サブスクリプションの値は **rep1** です。

  ```
  exec sp_subscription 'create', SalesRepData,
  field_user, 'rep1'
   go
  ```

  この段階では、サブスクリプションは「**開始**」されません。つまり、データのやりとりは一切行われません。サブスクリプションはデータベース抽出ユーティリティが開始します。

リモート・データベースの抽出

リモートの Adaptive Server Anywhere データベースの作成には、3 つの段階があります。

- 一連のファイルに、スキーマとデータを抽出する。これは、*ssxtract* ユーティリティを使用します。
- Adaptive Server Anywhere データベースを作成する。
- データベースに、スキーマとデータをロードする。

スキーマとデータの抽出

すべての情報を入力したら、ユーザ **field_user** の Adaptive Server Anywhere データベースを抽出します。次のコマンド・ライン (チュートリアル・ディレクトリで、1 行で入力) で、このプロシージャを実行します。

```
ssxtract -v -c "eng=server-name;
dbn=hq;uid=sa;pwd=sysadmin" C:¥tutorial¥field
field_user
```

オプションは、次のことを意味します。

- **-v** 冗長モード。開発作業では、追加の出力が行われます。
- **-c** 接続文字列オプション。接続文字列は、-c に続けて二重引用符で囲んで入力します。
- **eng=server-name** 抽出ユーティリティが接続するサーバを指定します。
- **dbn=hq** 使用するサーバ上のデータベースを指定します。この場合は **hq** です。
- **uid=sa** データベースへのログオンに使用するログイン ID。
- **pwd=sysadmin** データベースへのログオンに使用するパスワード。
- **C:\tutorial\field** データを格納するファイルを保存するディレクトリ。
- **field_user** データベースの抽出対象となるユーザ ID。

抽出ユーティリティ・オプションの詳細については、「抽出ユーティリティ」391 ページを参照してください。

このコマンドを実行すると、次のファイルが作成されます。

- **再ロード・スクリプト** 再ロード・スクリプトは *reload.sql* という名前で、現在のディレクトリに保存されます。
- **データ・ファイル** データベースにロードするデータを格納するファイル。この例では、ファイルは空です。

Adaptive Server Anywhere データベースの作成

dbinit ユーティリティを使用して、Adaptive Server Anywhere データベースを作成できます。Adaptive Server Enterprise データベースと異なり、簡単な Adaptive Server Anywhere データベースはファイルです。

Adaptive Server Enterprise サーバのオプションをデフォルト以外に設定する場合を除き、Adaptive Server Anywhere データベースは Adaptive Server Enterprise データベースの動作と互換性を持つように作成してください。

❖ field.db という名前のデータベース・ファイルを作成する場合は、次の手順に従います。

- *c:¥tutorial¥field* ディレクトリで、次のコマンドを入力します。

```
dbinit -b -c -k field.db
```

  `-b` オプションは、文字列比較にブランクを強制的に埋め込みます。`-c` オプションは、文字列比較の大文字と小文字を区別します。`-k` オプションは、Adaptive Server Enterprise との互換性をより高めたシステム・カタログを作成します。

データベースへのデータのロード

Adaptive Server Anywhere の Interactive SQL ユーティリティ、または *rtsql* ユーティリティを使用して、データベースにデータをロードできます。*rtsql* は、バッチ処理専用の Interactive SQL に代わるもので、ランタイム・データベースに使用します。

❖ Interactive SQL を使用してデータベースにデータをロードする場合は、次の手順に従います。

1. **field** データベース上で実行する Adaptive Server Anywhere サーバを起動します。

```
dbeng9 field.db
```

2. Interactive SQL ユーティリティを使用して、サーバに接続します。

```
dbisql -c "eng=field;dbn=field;uid=DBA;pwd=SQL"
```

Adaptive Server Anywhere データベースは大文字と小文字を区別するように作成されているため、ユーザ ID とパスワードは大文字で入力してください。

3 READ コマンドを使用して、データをロードします。

```
READ C:¥TUTORIAL¥RELOAD.SQL
```

❖ バッチ処理としてデータベースにデータをロードする場合は、次の手順に従います。

1 **field** データベース上で実行する Adaptive Server Anywhere サーバを起動します。

```
dbeng9 field.db
```

2 Interactive SQL からスクリプトを実行します。

```
dbisql -c "eng=field;dbn=field;uid=DBA;pwd=SQL"
reload.sql
```

Adaptive Server Anywhere データベースは大文字と小文字を区別するように作成されているため、ユーザ ID とパスワードは大文字で入力してください。

次の手順

以上でこのシステムは、レプリケーションの準備が完了しました。

次の手順である、データの挿入とレプリケーションの詳細については、「データのレプリケーションの開始」83 ページの項を参照してください。

データのレプリケーションの開始

ここまでの作業で、レプリケーション・システムが使えるようになりました。この項では、統合データベースからリモート・データベースに、また、リモート・データベースから統合データベースに、データをレプリケートします。

統合データベースでのデータの入力

この項では、統合 (Adaptive Server Enterprise) データベースの **SalesRep** テーブルと **Customer** テーブルにデータを入力し、このデータを Adaptive Server Anywhere データベースにレプリケートします。

❖ Adaptive Server Enterprise データベースにデータを入力する場合は、次の手順に従います。

1　*isql* から Adaptive Server Enterprise サーバに接続します。

```
isql -S server-name -U sa -P sysadmin
```

2　**hq** データベースを使用していることを確認し、一連のローを入力します。

```
use hq
 go
 insert into SalesRep (rep_key, name)
 values ('rep1', 'Field User')
 go
 insert into SalesRep (rep_key, name)
 values ('rep2', 'Another User')
 go
 insert into Customer (cust_key, name, rep_key)
 values ('cust1', 'Ocean Sports', 'rep1')
 go
 insert into Customer (cust_key, name, rep_key)
 values ('cust2', 'Sports Plus', 'rep2')
 go
 commit
 go
```

Ocean Sports は **Field User** に割り当てられており、**Sports Plus** は **Another User** に割り当てられています。SQL Remote はコミットされた変更しかレプリケートしないので、変更をコミットしてください。

統合データベースにデータを入力すると、今度は関連したローをリモート Adaptive Server Anywhere データベースに送信する必要があります。

統合データベースからのデータの送信

リモート・データベースにローを送信するには、統合データベースで Message Agent を実行してください。*ssremote* プログラムが、Adaptive Server Enterprise の Message Agent です。

❖ Adaptive Server Enterprise からデータをレプリケートする場合は、次の手順に従います。

1 次の文を (1 行で) コマンド・ラインに入力し、統合データベースに対して Message Agent を実行します。

```
ssremote -c "eng=server-
name;dbn=hq;uid=sa;pwd=sysadmin"
```

2 メッセージの送信が完了したら、[Message Agent] ウィンドウで [シャットダウン] をクリックして Message Agent を停止します。

リモート・データベースでのデータの受信

リモート・データベースで insert 文を受信するには、リモート・データベースで Message Agent である *dbremote* を実行します。

❖ Adaptive Server Anywhere でデータを受信する場合は、次の手順に従います。

1 データベース・サーバが稼働している状態で、Adaptive Server Anywhere の Message Agent を使用してデータを受信します。

```
dbremote -c "eng=field;dbn=field;uid=DBA;pwd=SQL"
```

dbremote オプションの詳細については、「Message Agent」376 ページ を参照してください。

2　メッセージの処理が完了したら、[Message Agent] ウィンドウで [シャットダウン] をクリックして Message Agent を停止します。

Message Agent の実行中は、[Message Agent] ウィンドウにステータス情報が表示されます。運用設定では、この情報をログ・ファイルに出力し記録を保存できます。

Message Agent は、まず **hq** からのメッセージを受信し、それからメッセージを送信します。このリターン・メッセージには、レプリケーションの更新の受信に成功したことの確認が含まれています。このような確認は、メッセージ・システム・エラーが発生した場合でもメッセージの配信を保証する、SQL Remote メッセージ・トラッキング・システムの一部です。

データの着信の確認

Interactive SQL を使用してリモートの **field** データベースに接続し、**SalesRep** テーブルと **Customer** テーブルを検査して、どのローが受信されたのかを確認してください。

❖ データが着信したことを確認する場合は、次の手順に従います。

1　Interactive SQL を使用して、field データベースに接続します。

2　次の文を入力し、**SalesRep** テーブルを検査します。

```
SELECT * FROM SalesRep
```

統合データベースで入力したローが両方とも **SalesRep** テーブルにあります。これは、**SalesRepData** パブリケーションに **SalesRep** テーブルからのすべてのデータが含まれていたからです。

3　次の文を入力し、**Customer** テーブルを検査します。

```
SELECT * FROM Customer
```

Customer テーブルには、統合データベースで入力したローが 1 つだけ (Ocean Sports) しかありません。これは、**SalesRepData** パブリケーションには、サブスクリプションが作成されている営業担当に割り当てられた顧客のみが含まれるからです。

リモート・データベースから統合データベースへのレプリケーション

リモート・データベースでデータを入力し、それを統合データベースに送信してみてください。ここではその概略だけを示します。

❖ リモート・データベースから統合データベースにデータをレプリケートする場合は、次の手順に従います。

1 Interactive SQL から **field** データベースに接続します。

2 リモート・データベースでローを挿入します。例を示します。

```
INSERT INTO Customer (cust_key, name, rep_key)
 VALUES ('cust3', 'North Land Trading', 'rep1')
```

3 ローをコミットします。

```
COMMIT;
```

4 *field.db* データベースが稼働している状態で、*dbremote* を実行して統合データベースにメッセージを送信します。

```
dbremote -c "eng=field;dbn=field;uid=DBA;pwd=SQL"
```

5 *ssremote* を実行し、統合データベースでメッセージを受信します。

```
ssremote -c "eng=server-
name;dbn=hq;uid=sa;pwd=sysadmin"
```

6 統合データベースに接続して、**Customer** テーブルを表示します。このテーブルには次の 3 つのローがあります。

```
SELECT *
 FROM Customer
```

cust_key	name	rep_key
cust1	Ocean Sports	rep1
cust2	Sports Plus	rep2
cust3	North Land Trading	rep1

この簡単な例では、プライマリ・キーの値が重複したエントリを防ぐようにはなっていません。SQL Remote では、このような重複エントリを防ぐことができます。詳細については、「SQL Remote 設計」の章を参照してください。

第 2 部　SQL Remote のレプリケーション設計

第 2 部では、SQL Remote のレプリケーション設計の問題について説明します。

第 6 章

SQL Remote インストール環境の設計

この章の内容

この章では、SQL Remote インストール環境の設計に関する一般的な問題と原則について説明します。

個々のシステムの詳細な説明については、「Adaptive Server Enterprise の SQL Remote 設計」179 ページ と 「Adaptive Server Anywhere の SQL Remote 設計」113 ページ の各章を参照してください。

設計の概要

この章では、パブリケーションの設計に関する一般的な問題について説明するので、SQL Remote インストール環境の設計時に考慮してください。また、SQL Remote がデータをレプリケートする方法についても説明します。

統合データベースの設計

すべての SQL Remote 管理タスクと同様、設計はデータベース管理者またはシステム管理者が統合データベースで実行します。

Adaptive Server Enterprise のシステム管理者またはデータベース管理者は、SQL Remote のすべての設定タスクを実行してください。

データベースの互換性

SQL Remote を使用するすべてのデータベースは、ソート順、文字セット、およびデータベースのオプション設定について、互いに互換性を持っている必要があります。

Adaptive Server Enterprise と Adaptive Server Anywhere の両方のデータベースで SQL Remote を使用する場合は、Adaptive Server Anywhere のデータベースを Adaptive Server Enterprise のデータベースと互換性を持つように作成します。

Adaptive Server Anywhere データベースを Adaptive Server Enterprise データベースと互換性を持つように作成する方法については、『ASA SQL ユーザーズ・ガイド』> 「Transact-SQL と互換性のあるデータベースの作成」を参照してください。ここでは、簡単に説明します。

❖ **Adaptive Server Enterprise と互換性のある Adaptive Server Anywhere データベース (Sybase Central) を作成するには、次の手順に従います。**

- [データベースの作成] ウィザードには、Adaptive Server Enterprise をエミュレートするために個々の選択項目を設定するボタンがあります。このボタンを使用すると、Transact-SQL と互換性のあるデータベースを非常に簡単に作成できます。

❖ **Adaptive Server Enterprise と互換性のある Adaptive Server Anywhere データベース (コマンド・ライン) を作成するには、次の手順に従います。**

1. **後続ブランクを無視する**　この処理には *dbinit* -b オプションを使用します。
2. **ユーザ ID に dbo を設定する**　すでにユーザ ID に **dbo** が設定されていれば、Adaptive Server Anywhere Transact-SQL システム・ビューの所有権を別のユーザ ID に移転できます。この処理には、*dbinit* -g オプションを使用します。
3. **古いシステム・ビューを削除する**　この処理には *dbinit* -k オプションを使用します。
4. **データベースの大文字と小文字を区別する**　この処理には *dbinit* -c オプションを使用します。

次のコマンドを入力して、現在のディレクトリに *test.db* (大文字と小文字を区別) という名前のデータベースを作成します。現在のユーザ ID である **dbo** を使用して、後続ブランクを無視し、古いシステム・ビューを削除します。

```
dbinit -b -c -k test.db
```

互換性のあるソート順と文字セットの使用

SQL Remote Message Agent は、文字セットを変換しません。

Adaptive Server Anywhere インストール環境の文字セット

Adaptive Server Anywhere では、統合データベースが使用する文字セットと照合順を、リモート・データベースと同じ設定にする必要があります。サポートされている文字セットについては、『ASA データベース管理ガイド』> 「国際言語と文字セット」を参照してください。

Adaptive Server Enterprise インストール環境の文字セット

Open Client/Open Server ライブラリは、LOCALES.DAT 文字セットと Adaptive Server Enterprise の文字セットが異なる場合に、SSREMOTE と Adaptive Server Enterprise 間の文字セットを変換します。Adaptive Server Enterprise サーバにこの両方の文字セットをインストールし、変換をサポートする必要があります。

混合インストール環境の文字セット

locales.dat 設定 (すべての Open Client アプリケーションに使用される) は、リモート Adaptive Server Anywhere の設定と同じにする必要があります。

次の表は、Adaptive Server Enterprise と Adaptive Server Anywhere 間で推奨される文字セットの組み合わせを示したものです。一緒に使用できるすべての文字セットの組み合わせが、このリストで網羅されているわけではありません。

Adaptive Server Anywhere 照合名	**Open Client/ Open Server の名前**	**Open Client/Open Server の大文字と小文字を区別するソート順**	**Open Client/ Open Server の大文字と小文字を区別しないソート順**
デフォルト	cp850	dictionary_cp850	nocase_cp850
437LATIN1	cp437	dictionary_cp437	nocase_cp437
437ESP	cp437	espdict_cp437	espnocs_cp437
437SVE	cp437	bin_cp437	bin_cp437
819CYR	iso_1	bin_iso_1	bin_iso_1
819DAN	iso_1	bin_iso_1	bin_iso_1
819ELL	iso_1	bin_iso_1	bin_iso_1
819ESP	iso_1	espdict_iso_1	espnocs_iso_1
819ISL	iso_1	bin_iso_1	bin_iso_1
819LATIN1	iso_1	dictionary_iso_1	nocase_iso_1
819LATIN2	iso_1	bin_iso_1	bin_iso_1
819NOR	iso_1	bin_iso_1	bin_iso_1
819RUS	iso_1	bin_iso_1	bin_iso_1
819SVE	iso_1	bin_iso_1	bin_iso_1
819TRK	iso_1	bin_iso_1	bin_iso_1
850CYR	cp850	bin_cp850	bin_cp850

Adaptive Server Anywhere 照合名	Open Client/ Open Server の名前	Open Client/Open Server の大文字と小文字を区別するソート順	Open Client/ Open Server の大文字と小文字を区別しないソート順
850DAN	cp850	scandict_cp850	scannocp_cp850
850ELL	cp850	bin_cp850	bin_cp850
850ESP	cp850	espdict_cp850	espnocs_cp850
850ISL	cp850	scandict_cp850	scannocp_cp850
850LATIN1	cp850	dictionary_cp850	nocase_cp850
850LATIN2	cp850	bin_cp850	bin_cp850
850NOR	cp850	scandict_cp850	scannocp_cp850
850RUS	cp850	bin_cp850	bin_cp850
850SVE	cp850	scandict_cp850	scannocp_cp850
850TRK	cp850	bin_cp850	bin_cp850
852LATIN2	cp852	bin_cp852	bin_cp852
852CYR	cp852	bin_cp852	bin_cp852
855CYR	cp855	cyrdict_cp855	cynocs_cp855
857TRK	cp857	bin_cp857	bin_cp857
860LATIN1	cp860	bin_cp860	bin_cp860
866RUS	cp866	rusdict_cp866	rusnocs_cp866
869ELL	cp869	bin_cp869	bin_cp869
932JPN	sjis	bin_sjis	bin_sjis
EUC_JAPAN	eucjis	bin_eucjis	bin_eucjis
EUC_CHINA	eucgb	bin_eucgb	bin_eucgb
EUC_TAIWAN	eucb5	bin_big5	bin_big5

Adaptive Server Anywhere 照合名	Open Client/ Open Server の名前	Open Client/Open Server の大文字と小文字を区別するソート順	Open Client/ Open Server の大文字と小文字を区別しないソート順
EUC_KOREA	eucksc	bin_eucksc	bin_eucksc
UTF8	utf8	bin_utf8	bin_utf8

文のレプリケーション方法

SQL Remote レプリケーションは、トランザクション・ログに基づいて行われ、個々のデータを更新するときに、すべてのデータを更新する代わりに変更部分だけをレプリケートします。SQL Remote がデータをレプリケートすると言う場合、実際は、*SQL Remote が、データを変更する SQL 文をレプリケートする*ことを意味します。

コミットされたトランザクションのみがレプリケートされる

SQL Remote では、データのレプリケート中は多少のタイムラグが生じますが、コミットされたトランザクションのみをレプリケートすることによって、レプリケーション設定時のトランザクションの最小単位を適切に保ち、レプリケーションに関わるデータベース間で一貫性を維持することができます。

プライマリ・キー

UPDATE 文または DELETE 文をレプリケートする場合、SQL Remote はプライマリ・キー・カラムを使用して、更新または削除を行うローをユニークに識別します。レプリケートされるすべてのテーブルは、宣言されたプライマリ・キーを持つか、一意性制約を守るものでなければなりません。ユニーク・インデックスでは十分ではありません。プライマリ・キーのカラムは、レプリケートされた更新や削除の WHERE 句で使用されます。テーブルにプライマリ・キーがない場合、WHERE 句はテーブルのすべてのカラムを参照します。

UPDATE 文は常に UPDATE 文とは限らない

簡単な INSERT 文をあるデータベースに入力すると、その文は、SQL Remote で設定されている他のデータベースに INSERT 文として送信されます。しかし、すべての文がクライアント・アプリケーションで入力されたとおりにレプリケートされるわけではありません。ここでは、SQL Remote が SQL 文をレプリケートする方法について説明します。堅牢な SQL Remote インストール環境を設計するには、このしくみを理解することが重要です。

Message Agent は、文のレプリケーションを実行するコンポーネントです。

挿入と削除のレプリケーション

INSERT 文と DELETE 文は、レプリケーションで最も簡単な例です。SQL Remote は、トランザクション・ログから個々の INSERT または DELETE オペレーションを取り出し、その挿入または削除されたローをサブスクライブするすべてのサイトに送信します。

テーブルのカラムのサブセットだけがサブスクライブされている場合、サブスクライバに送信される INSERT 文にはそのサブセットのみが含まれます。

Message Agent は、文を最初に入力したユーザにそれらの文がレプリケートされないようにします。

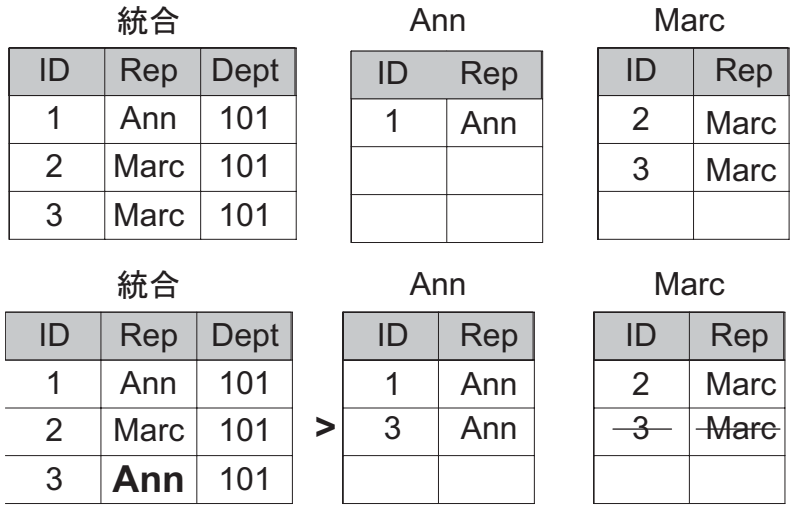
更新のレプリケーション

UPDATE 文は、クライアント・アプリケーションが入力したとおりにはレプリケートされません。ここでは、レプリケートされる UPDATE 文が、入力した UPDATE 文と異なる 2 つの場合を説明します。

UPDATE 文が INSERT 文または DELETE 文としてレプリケートされる場合

UPDATE 文は、いずれかのリモート・ユーザのサブスクリプションからローを削除する場合、DELETE 文としてそのユーザに送信されます。UPDATE 文は、いずれかのリモート・ユーザのサブスクリプションにローを追加する場合、INSERT 文としてそのユーザに送信されます。

次の図は、各サブスクライバが自分の名前を使用してサブスクライブするパブリケーションを示します。



Marc のローの **Rep** 値を Ann に変更する UPDATE 文は、Marc には DELETE 文として、Ann には INSERT 文としてレプリケートされます。

サブスクライバ間のローの再編成は、営業担当者間に顧客を定期的に再割り当てする営業地域自動割り当てアプリケーションに共通の機能で、「**領域の再編成**」とも呼ばれます。

UPDATE 文の競合の検出

UPDATE 文は、既存の 1 つまたは複数のローの値を新しい値に変更します。変更されるローは、UPDATE 文の WHERE 句によって異なります。

SQL Remote が UPDATE 文をレプリケートする場合、複数のシングル・ロー更新をセットにまとめて行います。これらのシングル・ロー文は、次のいずれかの理由で失敗する場合があります。

- **更新するローが存在しない**　個々のローはプライマリ・キーの値によって識別されるため、プライマリ・キーが他のユーザによって変更された場合は、更新すべきローが見つかりません。

  この場合、UPDATE 文は何も更新しません。

- **更新するローが 1 つまたは複数のカラムで異なる** 存在するはずの値の 1 つが他のユーザによって変更された場合、「**更新の競合**」が発生します。

  リモート・データベースでは、ローの値に関わらず更新が実行されます。

  統合データベースでは、SQL Remote によって「**競合解決**」オペレーションが実行されます。競合解決オペレーションは、トリガまたはストアド・プロシージャ内に格納され、競合が検出されると自動的に実行されます。

  Adaptive Server Anywhere では、競合解決トリガは更新の前に実行され、このトリガが終了すると更新が実行されます。Adaptive Server Enterprise では、更新が適用された後に競合解決プロシージャが実行されます。

- **主キー制約または一意性制約のないテーブルはレプリケートされた更新の WHERE 句のカラムをすべて参照する** 2 人のユーザが同じローを更新する場合、レプリケートされた更新は更新作業を行わず、データベースの整合性が失われます。したがって、レプリケートされたすべてのテーブルに主キー制約または一意性制約を持たせ、制約対象のカラムが更新されないようにする必要があります。

プロシージャのレプリケーション

どのようなレプリケーション・システムでも、ストアド・プロシージャ・コールをレプリケートするときに、次の 2 つのオプションのどちらかを選択します。

- **プロシージャ・コールをレプリケートする** 対応するプロシージャがレプリケート・サイトで実行されます。
- **プロシージャの動作をレプリケートする** プロシージャの個別の動作 (INSERT、UPDATE、DELETE など) がレプリケートされます。

*SQL Remote は、プロシージャをレプリケートするときに、プロシージャの動作をレプリケートします。*プロシージャ・コールはレプリケートされません。

トリガのレプリケーション

SQL Remote のトリガのレプリケーションは、Adaptive Server Enterprise Message Agent と Adaptive Server Anywhere Message Agent で異なります。

Adaptive Server Enterprise からのトリガのレプリケーション

Adaptive Server Enterprise から、トリガの動作がレプリケートされます。リモート Adaptive Server Anywhere データベースでは、トリガが起動されないようにしてください。リモート Adaptive Server Anywhere データベースでトリガが起動されると、トリガの動作が 2 回実行されてしまいます。

Adaptive Server Anywhere の FIRE_TRIGGERS データベース・オプションは、トリガの起動を防止します。このオプションを Message Agent が使用するユーザ ID に設定する場合は、このユーザ ID をほかの目的に使用しないでください。

Adaptive Server Anywhere の場合にかぎりますが、トリガの実行を防止する別の方法として、トリガの本文に対して次の条件を指定する方法があります。

```
IF CURRENT REMOTE USER IS NULL
```

この条件を指定すると、現在のユーザが Message Agent である場合にトリガが実行されます。

Adaptive Server Anywhere からのトリガのレプリケーション

デフォルトでは、Message Agent for Adaptive Server Anywhere は、トリガが実行した動作をレプリケートしません。トリガはリモートで定義するように想定されています。これによって、パーミッションに関する問題と各動作が 2 回発生する可能性を回避します。この原則には次のような例外があります。

- **競合解析トリガの動作**　競合解析 (RESOLVE UPDATE) トリガが実行する動作は、統合データベースから、競合を発生させたメッセージを送信したデータベースを含む、すべてのデータベースにレプリケート*されます*。
- **BEFORE トリガのレプリケーション**　BEFORE トリガの中には、SQL Remote を使用しているときに好ましくない結果を発生させるものがあるため、更新中のローを変更する BEFORE トリガの動作は、UPDATE 動作の前にレプリケート*されます*。

インストール環境を設計する場合は、この動作に注意してください。たとえば、ローの更新回数を追跡するカウンタ・カラムをロー内で増加させる BEFORE UPDATE は、レプリケートされると 2 回カウントが行われます。これは、UPDATE がレプリケートされると、BEFORE UPDATE トリガが起動されるためです。この問題を回避するには、サブスクライバ・データベースにトリガが存在しないこと、またはトリガがレプリケートされた動作を実行しないことを確認してください。また、カラムを最終更新時間に設定する BEFORE UPDATE は、UPDATE がレプリケートされた時間を取得します。

トリガの動作をレプリケートするオプション

Adaptive Server Anywhere Message Agent には、メッセージを送信するときにすべてのトリガをレプリケートさせるオプションがあります。それは、*dbremote* `-t` オプションです。

このオプションを使用する場合は、トリガの動作がリモート・データベースで 2 回 (リモート・サイトでトリガを起動するとき、および統合データベースからレプリケートされた動作を明示的に適用するとき) 実行されないようにする必要があります。

トリガの動作が 2 回実行されないようにするには、トリガの本文を IF CURRENT REMOTE USER IS NULL … END IF 文で囲むか、Message Agent のユーザ ID に対して、Adaptive Server Anywhere の Fire_triggers オプションを [OFF] に設定します。

データ定義文のレプリケーション

データ定義文 (CREATE、ALTER、DROP、およびデータベース・オブジェクトを変更するその他の文) は、パススルー・モードで入力した場合を除き、SQL Remote にレプリケートされません。

Adaptive Server Anywhere のパススルー・モードの詳細については、「パススルー・モードの使用」335 ページを参照してください。

データ型がレプリケートされる方法

long binary データや character データ、および datetime データの扱いには特別な注意が必要です。

blob (Binary Large OBject) のレプリケーション

blob は LONG VARCHAR、LONG BINARY、TEXT、および IMAGE データ型で、256 文字より長い値です。

Adaptive Server Anywhere のレプリケーション

SQL Remote には、Adaptive Server Anywhere データベース間で blob をレプリケートする特殊なメソッドがあります。

Message Agent は、レプリケートされている INSERT 文または UPDATE 文の値の代わりに変数を使用します。変数の値は、次のように文を結合して作成します。

```
SET vble = vble || 'more_stuff'
```

この結果、長い値を含んでいる SQL 文が短くなり、1 つのメッセージ内に収まります。SET 文は独立した SQL 文であるため、blob は複数の SQL Remote メッセージに効果的に分割されます。

Adaptive Server Enterprise のレプリケーション

blob は、Message Agent メモリに収まるかぎり、Adaptive Server Enterprise とどちらの方向にもレプリケートできます。

CS_IODESC 構造を取り扱う Sybase Open Client CTLIB アプリケーションでは、**log_on_update** メンバを FALSE に設定しないでください。

Verify_threshold オプションを使用してメッセージ・サイズを最小にする

Verify_threshold データベース・オプションを使用すると、(レプリケートされた UPDATE 文の VERIFY 句による) 確認の対象から長い値を除外できます。このオプションのデフォルト値は 1000 です。カラムのデータ型が閾値より長いと、カラムの古い値は UPDATE がレプリケートされるときに確認されません。このようにして、SQL Remote メッセージのサイズが大きくならないようにしますが、競合する長い値の更新が検出されないという短所もあります。

メッセージのサイズを小さくするために **Verify_threshold** を使用しているときでも、競合を検出する方法があります。"blob" を更新するときに、同じテーブル内の **last_modified** カラムも必ず更新するようにします。これにより、**last_modified** カラムの古い値が確認されるため、競合を検出できます。

ワーク・テーブルを使用して重複する更新を防ぐ

blob が繰り返し更新されるときは、「ワーク」テーブルで更新し、最終バージョンをレプリケートされたテーブルに割り当てるようにします。たとえば、作業中の資料が 1 日に 20 回更新される場合、1 日の終りに Message Agent を 1 回実行すると、20 回の更新すべてがレプリケートされます。資料のサイズが 200 KB の場合は、4 MB のメッセージが送信されます。

これよりも良い方法は、**document_in_progress** テーブルを作成することです。ユーザが資料の変更を終了したときに、アプリケーションがその資料を **document_in_progress** テーブルからレプリケートされたテーブルに移動します。この結果、1 回の更新 (200 KB のメッセージ) で済みます。

blob のレプリケーションの制御

Adaptive Server Anywhere の BLOB_THRESHOLD オプションを使用すると、長い値のレプリケーションを詳細に制御できます。BLOB_THRESHOLD オプションより長い値は、blob としてレプリケートされます。つまり、細かく分割して各部分をレプリケートし、SQL 変数を使用して、受信者サイトで分割された部分を連結して再構成します。

BLOB_THRESHOLD をリモート Adaptive Server Anywhere データベースで高い値に設定すると、blob は細かく分けられず、Message Agent が操作を Adaptive Server Enterprise に適用します。各 SQL 文はメッセージ内に入れる必要があるので、小さな blob のレプリケーションのみ許可されます。

日付と時刻のレプリケーション

日付カラムまたは時刻カラムをレプリケートする場合は、Message Agent は、データベース・オプションの SR_Date_Format、SR_Time_Format、SR_Timestamp_Format の設定を使用して、日付をフォーマットします。

たとえば、次のオプション設定は、Message Agent に 1987 年 5 月 2 日という日付を 1987-05-02 として送信するように命令します。

```
SET OPTION SR_Date_Format = 'yyyy-mm-dd'
```

詳細については、「SQL Remote のオプション」406 ページを参照してください。

日付と時刻をレプリケートするときに、次の点を考慮します。

- 時刻、日付、およびタイムスタンプには、インストール環境全体で一貫したフォーマットを使用します。
- Adaptive Server Anywhere データベースを統合データベースとして使用する場合は、日付とタイムスタンプのフォーマットに使われている年、月、日の順序が、データベース・オプションの DATE_ORDER の設定と同じになるようにします。

  個々の接続の間、DATE_ORDER オプションを変更できます。
- Adaptive Server Enterprise データベースを統合データベースとして使用する場合は、SQL Remote の設定の年、月、日の順序が Adaptive Server Enterprise データベースの dateformat の設定と同じになるようにします。

❖ Adaptive Server Enterprise データベースの日付表示形式の設定を確認する

1. *ssremote* が使用するログイン ID を使用して、*isql* から Adaptive Server Enterprise データベースにログインします。この例では、ログイン ID として **ssr** を使用します。
2. 次のコマンドを発行します。

   ```
   select *
    from master..syslogins
    where name = 'ssr'
    go
   ```

   Adaptive Server Enterprise は、ssr ユーザのデフォルト言語を返します。

3 ssr がデフォルト言語 (us_english) を使用している場合、デフォルトの日付表示形式は YMD です。デフォルト言語と異なる言語を使用している場合は、次のコマンドを入力します。

```
sp_helplanguage language-name
```

ここで、*language-name* は ssr ユーザが使用する言語です。表示される情報には、この言語のデフォルトの日付フォーマットが含まれます。

どのサブスクライバにどの文が送信されるか

テーブルのローに挿入、削除、または更新が行われるたびに、ローにサブスクリプションを作成したユーザにメッセージを送信する必要があります。また、更新の場合はサブスクリプション式が変更されることがあるため、あるサブスクライバには削除文、別のサブスクライバには更新文、また別のサブスクライバには挿入文が送信されます。

送信される文と宛先のサブスクライバの詳細については「文のレプリケーション方法」97 ページを参照してください。サブスクリプションの詳細については、以降の 2 つの章を参照してください。

ここでは、SQL Remote が適切な受信者に適切なオペレーションを送信する方法について説明します。

どのサブスクライバにどのオペレーションを送信するかを決定するタスクは、データベース・サーバと Message Agent が分担します。データベース・サーバはパブリケーションに関係のある作業を処理し、Message Agent はサブスクリプションに関係のある作業を処理します。

Adaptive Server Anywhere の動作

Adaptive Server Anywhere は、パブリケーションの一部であるテーブルを更新する個々のサブスクリプション式を評価します。更新前と更新後に、式の値をログに追加します。

サブスクライバ・リストの評価や記録は行わない

Adaptive Server Enterprise では、サブスクライバのリストの評価やログへの記録は行われません。サブスクリプション式 (パブリケーションのプロパティ) を評価し、記録します。サブスクライバの処理はすべて Message Agent に任されます。

複数のパブリケーションの一部であるテーブルでは、個々のパブリケーションに対して更新前と更新後に、サブスクリプション式が評価されます。

ログに情報を追加すると、以下の場合にパフォーマンスが低下することがあります。

- **高負荷の式**　サブスクリプション式の評価が高い負荷を生む場合は、パフォーマンスが低下することがあります。

- **多数のパブリケーション** テーブルが多数のパブリケーションに属している場合は、多くの式を評価する必要があります。これに対し、サブスクリプションの数はパフォーマンスに影響しません。
- **複数の値が含まれる式** 一部の式には複数の値が含まれています。このような式では、トランザクション・ログの情報が大幅に増加し、パフォーマンスが低下する場合があります。

Adaptive Server Enterprise の動作

Adaptive Server Enterprise パブリケーションの SQL Remote では、サブスクリプション式はカラムでなければなりません。サブスクリプション・カラムには、1 つの値か、またはカンマで区切られた複数の値が含まれます。

サブスクライバ・リストの評価や記録は行わない

Adaptive Server Enterprise では、サブスクライバのリストはログに記録されません。記録するのはカラム値です。サブスクライバの処理はすべて Message Agent に任されます。

テーブルが **sp_add_remote_table** (これは **sp_setreplicate** を呼び出す) を使用してレプリケートするようにマーク付けされている場合、Adaptive Server Enterprise は、削除の場合はロー全体の更新前イメージを、挿入の場合は全体の更新後イメージを、更新の場合は更新前イメージと更新後イメージの両方を、トランザクション・ログに記録します。これは、サブスクリプション・カラムの更新前の値と更新後の値を両方利用できることを意味します。

Message Agent の動作

Message Agent は、トランザクション・ログから評価済みのサブスクリプション式またはサブスクリプション・カラムへのエントリを読み込み、更新前の値と更新後の値をパブリケーションの個々のサブスクライバのサブスクリプション値と照合します。Message Agent は、このようにして適切なオペレーションを個々のサブスクライバに送信します。

サブスクライバの数が非常に多くてもサーバのパフォーマンスは低下しませんが、Message Agent のパフォーマンスは低下する場合があります。サブスクリプション値を大量のサブスクリプション値と照合する作業と、メッセージを送信する作業は、大きな負荷となる場合があります。

レプリケーション・エラーと競合

SQL Remote を使用すると、さまざまなサイトにあるデータベースを更新できます。しかし、特にデータベースの構造が複雑な場合は、レプリケーション・エラーを回避するように注意深くパブリケーションを設計する必要があります。ここでは、レプリケーションの設定で発生する可能性のあるエラーと競合について説明し、次にエラーを回避して競合を処理するようにパブリケーションを設計する方法について説明します。

配信エラーについては説明しません
この項では、メッセージ配信障害に関する問題については説明しません。配信エラーとその解決方法については、「メッセージ・トラッキング・システム」306 ページを参照してください。

レプリケーション・エラー

次のようなレプリケーション・エラーがあります。

- **重複プライマリ・キー・エラー**　2 人のユーザが同じプライマリ・キー値を使用してローを挿入する場合や、あるユーザがプライマリ・キーを更新し、別のユーザが新しい値のプライマリ・キーを挿入する場合に発生します。レプリケーション・システムのデータベースに 2 度目にアクセスしようとすると、プライマリ・キーが 2 つ作成されるため、エラーが発生します。

- **ローが見つからないエラー**　あるユーザがロー (特定のプライマリ・キー値を持つロー) を削除し、別のユーザが異なるサイトで同じローを更新または削除すると発生します。

  この場合は、後者のユーザが UPDATE 文または DELETE 文を送信したときに、ローが見つからないという理由でエラーが発生します。

- **参照整合性エラー**　外部キーを持つカラムがパブリケーションに含まれているにもかかわらず、それと関連付けされているプライマリ・キーが含まれていない場合、抽出ユーティリティはリモート・データベースに外部キーの定義を渡さず、そのリモート・データベースで INSERT 文が失敗しないようにします。

この問題は、適切なデフォルトをテーブル定義に入れることによって解決できます。

また、プライマリ・テーブルに SUBSCRIBE BY 式が含まれているが、それと関連付けされている外部テーブルには含まれていない場合に、参照整合性エラーが発生することがあります。外部テーブルのローはレプリケートされる可能性がありますが、プライマリ・テーブルのローはパブリケーションから除外されることがあるためです。

レプリケーションの競合

レプリケーションの競合は、エラーとは異なります。競合は適切に処理すれば、SQL Remote では問題を起こしません。

- **競合** あるユーザがローを更新するとします。この同じローを別のユーザが他のサイトで更新します。2 番目のユーザのオペレーションが成功すると、SQL Remote によってトリガが起動 (Adaptive Server Anywhere の場合) されるか、プロシージャが呼び出され (Adaptive Server Enterprise の場合)、データが適切に変更されるような方法で競合が解決されます。

  競合は多くのインストール環境で発生します。SQL Remote では、SQL Remote セットアップの通常のオペレーションの一部として、トリガとプロシージャを使用して競合を適切に解決できます。

  競合が発生した場合の SQL Remote の処理方法については、以降の章を参照してください。

SQL エラーのトラッキング

レプリケーションにおける SQL エラーは、セットアップ時にトラッキングできるように設計する必要があります。SQL Remote のオプションを使用して SQL 文のエラーをトラックできますが、このオプションではエラーを解決できません。

Replication_error オプションを設定すると、SQL エラーが発生したときに Message Agent で呼び出すストアド・プロシージャを指定できます。デフォルトではプロシージャを呼び出しません。

❖ Adaptive Server Anywhere で Replication_error オプションを設定する

- 次の文を発行します。

```
SET OPTION
 remote-user.Replication_error
 = 'procedure-name'
```

remote-user は Message Agent コマンド・ラインのユーザ ID、*procedure-name* は SQL エラーが検出されたときに呼び出されるプロシージャです。

❖ Adaptive Server Enterprise で Replication_error オプションを設定する

- 次の文を発行します。

```
exec sp_remote_option Replication_error, procedure-
name
 go
```

procedure-name は SQL エラーが検出されたときに呼び出されるプロシージャです。

レプリケーション・エラー・プロシージャの稼働条件

レプリケーション・エラー・プロシージャでは、データ型が CHAR、VARCHAR、または LONG VARCHAR の引数を 1 つ指定してください。プロシージャは、SQL エラー・メッセージで 1 回、エラーの原因となった SQL 文でもう 1 回呼び出されます。

第 7 章

Adaptive Server Anywhere の SQL Remote 設計

この章の内容

この章では、統合データベースが Adaptive Server Anywhere データベースである場合の、SQL Remote のインストール環境の設計方法について説明します。

Adaptive Server Enterprise の類似ソフトウェア

Adaptive Server Anywhere と Adaptive Server Enterprise では、パブリケーション設計方法の多くが共通していますが、コマンドと機能が異なります。この章は、Adaptive Server Enterprise のユーザを対象とした章 「Adaptive Server Enterprise の SQL Remote 設計」179 ページとの間で多くの部分が重複しています。

設計の概要

SQL Remote インストール環境を設計するには、次の作業を行います。

- **パブリケーションの設計**　パブリケーションは、データベース間で共有する情報を決定します。
- **サブスクリプションの設計**　サブスクリプションは、各ユーザが受信する情報を決定します。
- **設計の実装**　システムの全ユーザ用に、パブリケーションとサブスクリプションを作成します。

すべての管理タスクは統合データベースで実行する

すべての SQL Remote 管理タスクと同様、設計はデータベース管理者またはシステム管理者が統合データベースで実行します。

Adaptive Server Anywhere データベースの管理者は、すべての SQL Remote 設定タスクを実行します。

データのパブリッシュ

この項では、テーブル全体またはテーブルのカラムワイズ・サブセットで構成される、簡単なパブリケーションの作成方法について説明します。これらのテーブルはアーティクルとも呼ばれます。Sybase Central を使用するか、Interactive SQL の CREATE PUBLICATION 文によって、これらのタスクを実行できます。

Sybase Central のパブリケーションはすべて、パブリケーション・フォルダに表示されます。パブリケーション用に作成するアーティクルはすべて、パブリケーションを選択すると右ウィンドウ枠の [アーティクル] タブに表示されます。

各パブリケーションには 1 つまたは複数の完全テーブルが入りますが、部分テーブルも許可されます。テーブルは、カラム、ローまたはその両方によってさらに分割できます。

テーブル全体のパブリッシュ

作成できる最も簡単なパブリケーションは、単一のアーティクルで構成されます。このアーティクルは、1 つまたは複数のテーブルのすべてのローとカラムで構成されます。これらのテーブルをあらかじめ用意しておく必要があります。

❖ 完全テーブルを 1 つ以上パブリッシュするには、次の手順に従います (Sybase Central の場合)。

1. DBA 権限のあるユーザとしてデータベースに接続します。
2. 左ウィンドウ枠の [パブリケーション] フォルダを選択します。
3. [ファイル] メニューから [新規] ー [パブリケーション] を選択します。

   [パブリケーション作成] ウィザードが表示されます。
4. パブリケーションの名前を入力します。[次へ] をクリックします。

5 [テーブル] タブで、[使用可能なテーブル] のリストからテーブルを 1 つ選択します。[追加] をクリックします。選択したテーブルが、右側の [選択したテーブル] リストに表示されます。

6 オプションで、さらにテーブルを追加することもできます。テーブルの順序は重要ではありません。

7 [終了] をクリックします。

❖ 1 つまたは複数の完全テーブルをパブリッシュするには、次の手順に従います (SQL の場合)。

1 DBA 権限のあるユーザとしてデータベースに接続します。

2 CREATE PUBLICATION 文を実行して、新しく作成するパブリケーションの名前とパブリッシュするテーブルの名前を指定します。

例

- 次の文で、customer テーブル全体をパブリッシュするパブリケーションを作成します。

```
CREATE PUBLICATION pub_customer (
     TABLE customer
 )
```

- 次の文は、Adaptive Server Anywhere のサンプル・データベースから、テーブル・セットの各テーブルのすべてのカラムとローを含むパブリケーションを作成します。

```
CREATE PUBLICATION sales (
     TABLE customer,
     TABLE sales_order,
     TABLE sales_order_items,
     TABLE product
 )
```

詳細については、『ASA SQL リファレンス・マニュアル』>「CREATE PUBLICATION 文」を参照してください。

テーブル内の一部のカラムだけをパブリッシュする

Sybase Central では、テーブルのすべてのローと、一部のカラムだけを含むパブリケーションを作成できます。また、CREATE PUBLICATION 文でカラムのリストを指定しても、同様に作成できます。

❖ テーブル内の一部のカラムだけをパブリッシュするには、次の手順に従います (Sybase Central の場合)。

1. DBA 権限のあるユーザとしてデータベースに接続します。
2. 左ウィンドウ枠の [パブリケーション] フォルダを選択します。
3. [ファイル] メニューから [新規] ー [パブリケーション] を選択します。

   [パブリケーション作成] ウィザードが表示されます。
4. 新しく作成するパブリケーションの名前を入力します。[次へ] をクリックします。
5. [テーブル] タブで、[使用可能なテーブル] のリストからテーブルを 1 つ選択します。[追加] をクリックします。選択したテーブルが右側の [選択したテーブル] のリストに追加されます。
6. [カラム] タブで、テーブルのアイコンをダブルクリックし、[使用可能なカラム] のリストを展開します。パブリッシュするカラムを 1 つずつ選択し、[追加] をクリックします。右側の [選択したカラム] リストに、選択したカラムが表示されます。
7. [終了] をクリックします。

❖ テーブル内の一部のカラムだけをパブリッシュするには、次の手順に従います (SQL の場合)。

1. DBA 権限のあるユーザとしてデータベースに接続します。

2 CREATE PUBLICATION 文を実行して、パブリケーション名とテーブル名を指定します。テーブル名の後ろのカッコの中に、パブリッシュするカラムをリストします。

例

- 次の文は、customer テーブルのカラムである id、company_name、city のすべてのローをパブリッシュするパブリケーションを作成します。

```
CREATE PUBLICATION pub_customer (
     TABLE customer (
          id,
          company_name,
          city )
 )
```

詳細については、『ASA SQL リファレンス・マニュアル』>「CREATE PUBLICATION 文」を参照してください。

テーブル内の一部のローだけをパブリッシュする

Sybase Central では、テーブルのすべてのカラムと、一部のローだけを含むパブリケーションを作成できます。Sybase Central と SQL のどちらの場合も、パブリッシュするローにのみ一致する検索条件を書き込んで処理します。

テーブル内の一部のローだけをパブリッシュする方法は、Sybase Central と SQL 言語から 2 とおり提供されますが、Mobile Link との互換性があるのは 1 つだけです。

- **WHERE 句** WHERE 句を使用すると、アーティクルにローのサブセットを入れることができます。このアーティクルを含むパブリケーションのすべてのサブスクライバは、WHERE 句を満たすローを受信します。
- **サブスクリプション式** サブスクリプション式を使用すると、アーティクルを含むパブリケーションに対する別のサブスクリプションに、異なるロー・セットを入れることができます。

アーティクルでは WHERE 句とサブスクリプション式を結合できます。Sybase Central または CREATE PUBLICATION 文で、それらを指定します。

パブリケーションのサブスクライバがそれぞれ、1 つのテーブルから異なるローを受け取る場合には、サブスクリプション式を使用します。サブスクリプション式は、テーブルを分割するための最も強力な手段です。

あるパブリケーションのすべてのサブスクリプションから同じロー・セットを除外する場合には、WHERE 句を使用します。

WHERE 句を使用して一部のローだけをパブリッシュする

WHERE 句を指定して、WHERE 条件を満たすローだけをパブリケーションに入れることができます。

❖ WHERE 句を使用してパブリケーションを作成するには、次の手順に従います (Sybase Central の場合)。

1　DBA 権限のあるユーザとしてデータベースに接続します。

2　左ウィンドウ枠の [パブリケーション] フォルダを選択します。

3　[ファイル] メニューから [新規] ー [パブリケーション] を選択します。

[パブリケーション作成] ウィザードが開きます。

4　新しく作成するパブリケーションの名前を入力します。[次へ] をクリックします。

5　[テーブル] タブで、[使用可能なテーブル] のリストからテーブルを 1 つ選択します。[追加] をクリックします。選択したテーブルが右側の [選択したテーブル] のリストに追加されます。

6　[WHERE 句] タブでテーブルを選択し、下段のボックスに検索条件を入力します。

7　[終了] をクリックします。

❖ WHERE 句を使用してパブリケーションを作成するには、次の手順に従います (SQL の場合)。

1 DBA 権限のあるユーザとしてデータベースに接続します。

2 CREATE PUBLICATION 文を実行して、パブリケーションと WHERE 条件に対象のローを入れます。

例

- 次の文は、status カラムの中でアクティブとマーク付けされた顧客に、customer テーブルの id、company_name、city、state カラムをパブリッシュするパブリケーションを作成します。

```
CREATE PUBLICATION pub_customer (
     TABLE customer (
         id,
         company_name,
         city,
         state )
     WHERE status = 'active'
 )
```

  この場合、status カラムはパブリッシュされません。パブリッシュ対象外のすべてのローにはデフォルト値を設定しておきます。設定しないと、統合データベースから挿入用にダウンロードするときに、エラーが発生します。

- 次の文は、関連する受注情報を営業担当者の Samuel Singer に送信する、単一のアーティクルで構成されたパブリケーションを作成します。

```
CREATE PUBLICATION pub_orders_samuel_singer (
     TABLE sales_order WHERE sales_rep = 856
 )
```

詳細については、『ASA SQL リファレンス・マニュアル』>「CREATE PUBLICATION 文」を参照してください。

SUBSCRIBE BY

create publication 文でも SUBSCRIBE BY 句は有効です。SQL Remote を使って、選択したローだけをパブリッシュするときにも、この句を使用できます。ただし、Mobile Link 同期では無視されます。

サブスクリプション式を使用して一部のローだけをパブリッシュする

サブスクリプション式を指定して、アーティクルを含むパブリケーションの異なるサブスクリプションに、異なるロー・セットを指定できます。

たとえば、モバイル環境において、各営業担当者が自分専用の受注用サブスクリプションを作成する sales パブリケーションがあるとします。これによって、ローカルで受注を更新したり、統合データベースに受注をレプリケートしたりできます。

WHERE 句のモデルを使用すると、営業担当者ごとにパブリケーションが必要です。営業担当者 Samuel Singer 用のパブリケーションは次のとおりです。その他の営業担当者ごとに、同様のパブリケーションが必要です。

```
CREATE PUBLICATION pub_orders_samuel_singer (
     TABLE sales_order
         WHERE sales_rep = 856
 )
```

複数の異なるサブスクリプションを必要とするセットアップに対応するため、SQL Remote では、「**サブスクリプション式**」をアーティクルと関連付けできます。サブスクリプションが受信するローは、指定された式の値によって異なります。

サブスクリプション式の利点

サブスクリプション式を使用すると、より簡潔で理解しやすいパブリケーションを作成できます。また、WHERE 句を使用した複数のパブリケーションを管理するよりも、優れたパフォーマンスが得られます。データベース・サーバは、パブリケーションの数に正比例して、トランザクション・ログに情報を追加し、そのログをスキャンしてメッセージを送信する必要があります。サブスクリプション式を使用すると、複数の異なるサブスクリプションを単一のパブリケーションに関連付けできます。WHERE 句ではこの関連付けができません。

❖ サブスクリプション式を使用してアーティクルを作成するには、次の手順に従います (Sybase Central の場合)。

1　DBA 権限のあるユーザとしてデータベースに接続します。

2　左ウィンドウ枠の [パブリケーション] フォルダを選択します。

3 [ファイル] メニューから [新規] ー [パブリケーション] を選択します。

[パブリケーション作成] ウィザードが表示されます。

4 パブリケーションの名前を入力し、[次へ] をクリックします。

5 [テーブル] タブで、そのテーブルに必要な値を設定します。

6 [SUBSCRIBE BY 制限] タブで、コントロールを使用してサブスクリプション式を作成します。

7 ウィザードの残りの指示に従います。

❖ サブスクリプション式を使用してアーティクルを作成するには、次の手順に従います (SQL の場合)。

1 DBA 権限のあるユーザとしてデータベースに接続します。

2 サブスクリプション式でマッチ式として使用する式を含む CREATE PUBLICATION 文を実行します。

例

- 次の文は、customer テーブルのカラムである id、company_name、city、state をパブリッシュするパブリケーションを作成します。このパブリケーションは、state カラムの値に従ってローとサブスクライバを一致させます。

```
CREATE PUBLICATION pub_customer (
     TABLE customer (
         id,
         company_name,
         city,
         state )
     SUBSCRIBE BY state
 )
```

- 次の文は、パブリケーションに対して 2 人の従業員のサブスクリプションを作成します。Ann Taylor はジョージア州 (GA) の顧客情報を受信し、Sam Singer はマサチューセッツ州 (MA) の顧客情報を受信します。

```
CREATE SUBSCRIPTION
 TO pub_customer ('GA')
 FOR Ann_Taylor ;
 CREATE SUBSCRIPTION
 TO pub_customer ('MA')
 FOR Sam_Singer
```

ユーザは、複数のパブリケーションに対するサブスクリプションを作成できます。また、単一のパブリケーションに対する複数のサブスクリプションも作成できます。

参照

- 『ASA SQL リファレンス・マニュアル』> 「CREATE PUBLICATION 文」
- 「サブスクリプション式を含まないテーブルの分割」132 ページ
- 「サブスクリプションの作成」175 ページ
- 「WHERE 句を使用して一部のローだけをパブリッシュする」119 ページ
- 「既存のパブリケーションの変更」123 ページ

既存のパブリケーションの変更

パブリケーションを作成してから、アーティクルの追加、修正、削除などの変更を加えたり、パブリケーションの名前を変更したりできます。アーティクルを修正する場合、そのアーティクル全体の仕様を入力する必要があります。

Sybase Central を使用するか、Interactive SQL の ALTER PUBLICATION 文によって、これらのタスクを実行できます。

❖ 既存のパブリケーションまたはアーティクルのプロパティを修正するには、次の手順に従います (Sybase Central の場合)。

1　パブリケーションを所有するユーザ、または DBA 権限を持つユーザとしてデータベースに接続します。

2 パブリケーションまたはアーティクルを右クリックし、ポップアップ・メニューで [プロパティ] を選択します。

3 必要なプロパティを設定します。

❖ アーティクルを追加するには、次の手順に従います (Sybase Central の場合)。

1 パブリケーションを所有するユーザ、または DBA 権限を持つユーザとしてデータベースに接続します。

2 左ウィンドウ枠で、[パブリケーション] フォルダを開きます。

3 アーティクルを追加するパブリケーションを選択します。

4 [ファイル] メニューから [新規] ー [アーティクル] を選択します。

[アーティクル作成] ウィザードが表示されます。

5 [アーティクル作成] ウィザードで、次の作業を実行します。

- テーブルを選択して [次へ] をクリックします。
- アーティクルのカラムを選択します。[次へ] をクリックします。
- 必要に応じて、WHERE 句を入力します。[次へ] をクリックします。
- 必要に応じて、SUBSCRIBE BY 制限を作成します。

6 [完了] をクリックし、アーティクルを作成します。

❖ アーティクルを削除するには、次の手順に従います (Sybase Central の場合)。

1 パブリケーションを所有するユーザ、または DBA 権限を持つユーザとしてデータベースに接続します。

2　[パブリケーション] フォルダを開きます。

3　アーティクルを削除するパブリケーションを選択します。

4　削除するアーティクルを右クリックし、ポップアップ・メニューで [削除] を選択します。

❖ **既存のパブリケーションを修正するには、次の手順に従います (SQL の場合)。**

1　パブリケーションを所有するユーザ、または DBA 権限を持つユーザとしてデータベースに接続します。

2　ALTER PUBLICATION 文を実行します。

例

- 次の文は、customer テーブルを pub_contact パブリケーションに追加します。

```
ALTER PUBLICATION pub_contact (
     ADD TABLE customer
 )
```

参照

- 『ASA SQL リファレンス・マニュアル』> 「ALTER PUBLICATION 文」
- 「WHERE 句を使用して一部のローだけをパブリッシュする」119 ページ
- 「サブスクリプション式を使用して一部のローだけをパブリッシュする」121 ページ

パブリケーションの削除

Sybase Central または DROP PUBLICATION 文のいずれかを使用して、パブリケーションを削除できます。パブリケーションを削除すると、そのパブリケーションに対するサブスクリプションもすべて自動的に削除されます。

パブリケーションを削除するには、DBA 権限が必要です。

❖ **パブリケーションを削除するには、次の手順に従います (Sybase Central の場合)。**

1 DBA 権限のあるユーザとしてデータベースに接続します。

2 [パブリケーション] フォルダを開きます。

3 対象のパブリケーションを右クリックし、ポップアップ・メニューで [削除] を選択します。

❖ **パブリケーションを削除するには、次の手順に従います (SQL の場合)。**

1 DBA 権限のあるユーザとしてデータベースに接続します。

2 DROP PUBLICATION 文を実行します。

例 次の文は、パブリケーション pub_orders を削除します。

```
DROP PUBLICATION pub_orders
```

『ASA SQL リファレンス・マニュアル』> 「DROP PUBLICATION 文」も参照してください。

パブリケーションについての注意

- これまでに説明した異なるタイプのパブリケーションを、組み合わせることができます。1 つのパブリケーションで、テーブル・セットのカラム・サブセットをパブリッシュし、WHERE 句を使用して、レプリケートするロー・セットを選択できます。
- パブリケーションの作成と削除には DBA 権限が必要です。
- DBA 権限を持つユーザか、パブリケーションの所有者だけがパブリケーションを変更できます。
- SQL Remote セットアップの実行中にパブリケーションを変更すると、レプリケーション・エラーが発生しやすくなります。レプリケーション・システムのデータが失われる可能性があるので、慎重に行ってください。

- パブリケーションにビューを入れることはできません。
- パブリケーションにストアド・プロシージャを入れることはできません。SQL Remote でプロシージャとトリガをレプリケートする方法については、「プロシージャのレプリケーション」100 ページを参照してください。

パブリケーションの設計 (Adaptive Server Anywhere)

簡単なパブリケーションの作成方法について理解できたら、適切なパブリケーション設計について考える必要があります。最適化設計は、SQL Remote のインストールを成功させる重要な要素です。この項では、Adaptive Server Anywhere の SQL Remote に対する最適な設計方法について説明します。

Adaptive Server Enterprise の類似ソフトウェア

Adaptive Server Anywhere と Adaptive Server Enterprise では、パブリケーション設計方法の多くが共通していますが、コマンドと機能が異なります。この項は、Adaptive Server Enterprise のユーザを対象とした項 「Adaptive Server Enterprise のパブリケーション設計」187 ページとの間で多くの部分が重複しています。

設計に関する問題の概要

各サブスクリプションは、完全なリレーショナル・データベース

リモート・データベースは、統合データベースとサブスクリプションに関する情報を共有します。サブスクリプションは、統合サイトで保持されるリレーショナル・データベースのサブセットであると同時に、リモート・サイトの完全なリレーショナル・データベースでもあります。したがって、サブスクリプション内の情報は、他のリレーショナル・データベースと同様に次の規則に従います。

- **外部キーの関係を有効にする** データベースには、外部キーの各エントリに対応するプライマリ・キーのエントリが必要です。

  データベース抽出ユーティリティを使用すると、リモート・データベースに対する CREATE TABLE 文が、リモートに存在しないテーブルに定義された外部キーを持たないようになります。

- **プライマリ・キーの一意性を管理する** 別のサイトで入力され、まだレプリケートされていない新しいローを検査する方法はありません。異なるサイトのユーザが同じプライマリ・キー値を

使用してローを追加することがないように、設計する必要があります。そうでない場合は、統合データベースにローをレプリケートするときに競合が発生します。

ロックがない状態でトランザクションの整合性を管理する

特定のローに対してシステム全体をロックするメカニズムがなくても、すべてのサイトで更新が行われた場合に、分散データベース (統合データベースとすべてのリモート・データベースで構成される) のデータは、整合性を保つ必要があります。

- **ロック競合を回避または解決する**　SQL Remote のインストール環境には、各ユーザがローを同時に変更しないように、すべてのデータベースに渡ってローをロックする方法はありません。このような競合は、競合が起こらないようにシステムを設計して回避するか、統合データベースで適切な方法で解決する必要があります。

これらリレーショナル・データベースの主要な機能を、パブリケーションとサブスクリプションの設計に組み込む必要があります。この項では、設計方法と技術について説明します。

有効なアーティクルの条件

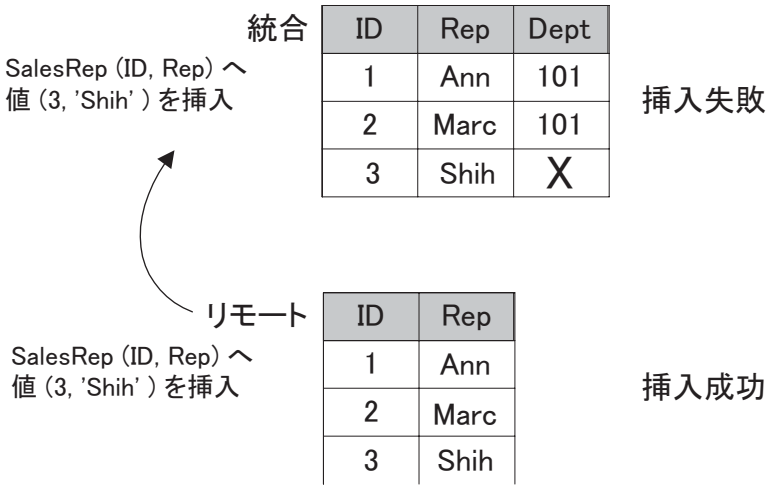
プライマリ・キーの全カラムが、アーティクルに含まれる必要があります。

リモート・データベースでの INSERT のサポート

リモート・データベースの INSERT 文が統合データベースへ正しくレプリケートされるために、有効な INSERT 文にならないカラムだけをアーティクルから除外できます。基本テーブルは次のとおりです。

- NULL を使用できるカラム
- デフォルトのカラム

これらの条件を満たさないカラムを除外すると、リモート・データベースで INSERT 文を実行して統合データベースにレプリケートすると失敗します。



代わりに BEFORE トリガを使用する
統合データベースが Adaptive Server Anywhere データベースである場合、BEFORE トリガを書き込んでおき、INSERT 文に含まれていないカラムを管理します。この方法で、この問題を回避します。

パフォーマンスのための設計ヒント

この項では、ハイパフォーマンスな SQL Remote のインストール環境を設計するための、チェック・リストを示します。

- **パブリケーションの数を小さくする** 特に、1 つの同じテーブルをさまざまなパブリケーションで参照しないようにします。

  データベース・サーバが必要とする作業は、パブリケーションの数に比例します。パブリケーションの数を小さく抑えて、サブスクリプションを有効に利用すれば、データベース・サーバの負荷を軽減できます。

  テーブル上で操作すると、データベース・サーバと Message Agent は、そのテーブルを含むパブリケーションごとにいくつかの作業を行います。リモート・ユーザごとにパブリケーション

が 1 つあると、データベース・サーバの負荷が急激に増加します。SUBSCRIBE BY を使用するパブリケーションをいくつか持ち、リモート・ユーザごとにサブスクリプションを持つ方が、負荷は減ります。パブリケーションに追加するサブスクリプションを増やしても、データベース・サーバの作業は増えません。Message Agent は、大量のサブスクリプションを使用して効率的に作業するように設計されています。

- **パブリケーションを論理的にグループ分けする**　たとえば、価格表などのすべてのリモート・ユーザが必要とするテーブルがある場合、そのテーブル用に別のパブリケーションを作成します。カラム値によって分割できるデータが格納されたテーブルごとに、パブリケーションを 1 つ作成します。

- **サブスクリプションを有効に利用する**　リモート・ユーザが同じような統合データベースのサブセットを受信するたびに、SUBSCRIBE BY 式を組み込むパブリケーションが使用されます。リモート・ユーザごとに別のパブリケーションを作成しないでください。

- **パブリケーション・トリガの更新に注意する**　特に、次の点に注意してください。

  - NEW / OLD SUBSCRIBE BY 構文を使用します。
  - 効率的にデータベースにアクセスするように SELECT 文を調整します。

- **トランザクション・ログ・サイズを監視する**　トランザクション・ログのサイズが大きいければ大きいほど、Message Agent のスキャン時間は長くなります。ログ名は定期的に変更し、DELETE_OLD_LOGS オプションを使用してください。

サブスクリプション式を含まないテーブルの分割

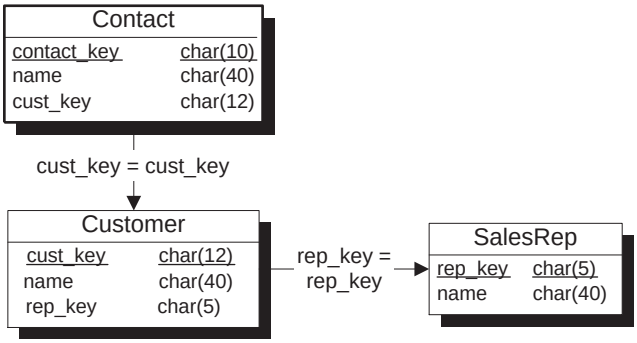
多くの場合、テーブルにサブスクリプション式がなくても、テーブルのローを分割する必要があります。

Contact データベースの例

Contact データベースの例では、サブスクリプション式を含まないテーブルの分割の必要性とその方法について説明します。

例

次に、この問題について説明する簡単なデータベースを示します。



各営業担当者は、複数の顧客に対して販売活動を行います。連絡先が 1 箇所だけの顧客もいれば、複数ある顧客もいます。

データベース内のテーブル

3 つのテーブルの詳細は次のとおりです。

テーブル	説明

SalesRep	社内のすべての営業担当者。**SalesRep** テーブルには、次のカラムがあります。
	
	• **rep_key** 各営業担当の識別子。これがプライマリ・キーです。
	• **name** 各営業担当の名前。
	
	このテーブルを作成する SQL 文は次のとおりです。
	
	`CREATE TABLE SalesRep (`
	`Rep_key CHAR(12) NOT NULL,`
	`Name CHAR(40) NOT NULL,`
	`PRIMARY KEY (rep_key)`
	`)`

Customer	会社と取引があるすべての顧客。**Customer** テーブルには、次のカラムがあります。
	
	• **cust_key** 各顧客の識別子。これがプライマリ・キーです。
	• **name** 各顧客の名前。
	• **rep_key** 取引を行う営業担当者の識別子。これが SalesRep テーブルへの外部キーです。
	
	このテーブルを作成する SQL 文は次のとおりです。
	
	`CREATE TABLE Customer (`
	`Cust_key CHAR(12) NOT NULL,`
	`Name CHAR(40) NOT NULL,`
	`Rep_key CHAR(12) NOT NULL,`
	`FOREIGN KEY REFERENCES SalesRep,`
	`PRIMARY KEY (cust_key)`
	`)`

テーブル	説明
Contact	会社と取引があるすべての個別の連絡先。連絡先はそれぞれ 1 人の顧客に属します。**Contact** テーブルには、次のカラムがあります。 • **contact_key** 各連絡先の識別子。これがプライマリ・キーです。 • **name** 各連絡先の名前。 • **cust_key** 連絡先が関連する顧客の識別子。これが Customer テーブルへの外部キーです。 このテーブルを作成する SQL 文は次のとおりです。 `CREATE TABLE Contact ( Contact_key CHAR(12) NOT NULL, Name CHAR(40) NOT NULL, Cust_key CHAR(12) NOT NULL, FOREIGN KEY REFERENCES Customer, PRIMARY KEY (contact_key) )`

レプリケーションの目標

この設計上の目標は、各営業担当者に次の情報を提供することです。

- 完全な **SalesRep** テーブル
- 営業担当者に割り当てられている顧客 (**Customer** テーブルの情報)
- 関連する顧客の連絡先 (**Contact** テーブルの情報)

Contact データベースの例における Customer テーブルの分割

サブスクリプション式に **rep_key** 値を使用すると、**Customer** テーブルを分割できます。**SalesRep** テーブルと **Customer** テーブルを含むパブリケーションは、次のとおりです。

```
CREATE PUBLICATION SalesRepData (
     TABLE SalesRep
     TABLE Customer SUBSCRIBE BY rep_key
 )
```

Contact データベースの例における Contact テーブルの分割

Contact テーブルも営業担当者の間で分割する必要がありますが、営業担当者の **rep_key** 値への参照は含みません。**rep_key** がテーブルに含まれない場合でも、Message Agent を使用すると、このテーブルのローとサブスクリプション値を同じものにできます。

このためには、**Customer** テーブルの **rep_key** カラムを評価する **Contact** アーティクルで、サブクエリを使用します。このパブリケーションは、次のようになります。

```
CREATE PUBLICATION SalesRepData (
     TABLE SalesRep
     TABLE Customer
         SUBSCRIBE BY rep_key
     TABLE Contact
         SUBSCRIBE BY (SELECT rep_key
             FROM Customer
            WHERE Contact.cust_key = Customer.cust_key
)
 )
```

サブスクリプション式の WHERE 句によって、サブクエリは値を 1 つだけ戻します。これは、**Customer** テーブル内にあるローの 1 つだけが、**Contact** テーブルの現在のローと同じ **cust_key** 値を持つからです。

Adaptive Server Enterprise の統合テーブルでは、この問題の解決策が異なります。詳細については、「サブスクリプション・カラムを含まないテーブルの分割」190 ページを参照してください。

Contact データベースの例における領域の再編成

「**領域の再編成**」を実行すると、サブスクライバ間でローが再割り当てされます。今回の例では、領域の再編成とは、営業担当者の間で **Customer** テーブル (つまり **Contact** テーブルも含む) のローを再割り当てすることを意味します。

新しい営業担当者に顧客が再割り当てされると、**Customer** テーブルが更新されます。UPDATE 文が、以前の営業担当者へ DELETE 文としてレプリケートされて、新しい営業担当者へ INSERT 文としてレプリケートされます。こうして、Customer ローは正しく新しい営業担当者へ移動されます。

Adaptive Server Anywhere と SQL Remote を同時に使用してこのような状況を処理する方法については、「どのサブスクライバにどの文が送信されるか」107 ページを参照してください。

顧客が再割り当てされても、**Contact** テーブルは影響を受けません。**Contact** テーブルには変更が加えられていないため、このテーブルに関するトランザクション・ログにはエントリがありません。この情報がないと、SQL Remote は、**Customer** テーブルだけでなく **Contact** テーブルのローも再割り当てできません。

この失敗によって、参照整合性の問題が発生します。以前の営業担当のリモート・データベースで、**Customer** テーブルと対応しなくなった **cust_key** 値が、**Contact** テーブルに含まれています。

トリガを使用して Contact テーブルを管理する

この問題を解決するには、特殊な UPDATE 文を含むトリガを使用します。これによって、データベース・テーブルをまったく変更しないで、トランザクション・ログにエントリを作成します。このログ・エントリでは、サブスクリプション式に BEFORE、AFTER を指定します。これで、Message Agent を使用して正しくローをレプリケートできます。

ローにおいて、このトリガの BEFORE オペレーションを起動してください。これによって、BEFORE 値を評価してログに配置することができます。また、各文ではなく FOR EACH ROW を起動して、トリガの提供する情報を新しいサブスクリプション式としてください。Message Agent は、この情報を使用して、どのサブスクライバがどのローを受信するかを決定します。

トリガの定義

トリガの定義は次のとおりです。

```
CREATE TRIGGER UpdateCustomer
 BEFORE UPDATE ON Customer
 REFERENCING NEW AS NewRow
    OLD as OldRow
 FOR EACH ROW
 BEGIN
     // determine the new subscription expression
     // for the Customer table
     UPDATE Contact
     PUBLICATION SalesRepData
     OLD SUBSCRIBE BY ( OldRow.rep_key )
     NEW SUBSCRIBE BY ( NewRow.rep_key )
     WHERE cust_key = NewRow.cust_key;
 END;
```

パブリケーションのための特殊な UPDATE 文

このトリガの UPDATE 文は、次のような特殊なフォームになります。

UPDATE *table-name*
　PUBLICATION *publication-name*
　{ **SUBSCRIBE BY** *subscription-expression* |
　　OLD SUBSCRIBE BY *old-subscription-expression*
　　NEW SUBSCRIBE BY *new-subscription-expression* }
　WHERE *search-condition*

- 各句の意味は、次のとおりです。
- *table-name* は、修正する必要がある、リモート・データベースのテーブルです。
- *publication-name* は、サブスクリプションを変更する必要のあるパブリケーションです。
- *subscription-expression* の値は、ローの新しい受信者と既存の受信者を決定するため、Message Agent によって使用されます。代わりに、サブスクリプション式 OLD と NEW を設定することもできます。
- WHERE 句は、サブスクリプションを作成したデータベース間で移動させるローを指定します。

トリガに関する注意事項

- トリガが次の構文を使用する場合は、注意してください。

```
UPDATE table-name
 PUBLICATION pub-name
     SUBSCRIBE BY sub-expression
 WHERE search-condition
```

トリガには、BEFORE トリガを使用する必要があります。この例では、BEFORE UPDATE トリガとなります。他のコンテキストでは、BEFORE DELETE トリガと BEFORE INSERT トリガが必要です。

- トリガが次の代替構文を使用する場合は、注意してください。

```
UPDATE table-name
 PUBLICATION publication-name
     OLD SUBSCRIBE BY old-subscription-expression
     NEW SUBSCRIBE BY new-subscription-expression }
 WHERE search-condition
```

トリガを BEFORE トリガまたは AFTER トリガにすることができます。

- UPDATE 文は、変更が適用されるパブリケーションとテーブルをリストします。文中の WHERE 句は、変更が適用されるローを示します。この UPDATE 文では、テーブルのデータは変更されませんが、トランザクション・ログにエントリが作成されます。
- この例におけるサブスクリプション式は、1 つの値を返します。複数の値を返すサブクエリも使用できます。サブスクリプション式の値は、UPDATE 文を実行したあとの値でなければなりません。

  この場合、ローへのサブスクライバだけが新しい営業担当者になります。新規サブスクライバと既存のサブスクライバが存在するケースについては、「複数のサブスクリプション間におけるローの共有」141 ページを参照してください。

トランザクション・ログ内の情報

ここでは、トランザクション・ログ内の情報について説明します。よく理解して、有効なパブリケーションを設計してください。

- 次のようなデータを想定します。
  - SalesRep テーブル

rep_key	Name
rep1	Ann
rep2	Marc

- Customer テーブル

cust_key	name	rep_key
cust1	Sybase	rep1
cust2	ASA	rep2

- Contact テーブル

contact_key	name	cust_key
contact1	David	cust1
contact2	Stefanie	cust2

- 次のような Update 文を適用して、領域を再編成します。

```
UPDATE Customer
 SET rep_key = 'rep2'
 WHERE cust_key = 'cust1'
```

この文を使用して、トランザクション・ログにエントリを 2 つ作成します。1 つは Contact テーブルの BEFORE トリガに対するエントリ、もう 1 つは Customer テーブルへの実際の UPDATE 文に対するエントリです。

```
SalesRepData - Publication Name
 rep1 - BEFORE list
 rep2 - AFTER list
 UPDATE Contact
 SET contact_key = 'contact1',
     name = 'David',
     cust_key = 'cust1'
```

```
WHERE contact_key = 'contact1'
SalesRepData - Publication Name
rep1 - BEFORE list
rep2 - AFTER list
UPDATE Customer
SET rep_key = 'rep2'
WHERE cust_key = 'cust1'
```

Message Agent は、これらのタグに対するログをスキャンします。この情報に基づいて、どのリモート・ユーザが INSERT、UPDATE、または DELETE を取得するかが決定されます。

この場合、BEFORE list は **rep1**、AFTER list は **rep2** となっています。BEFORE list と AFTER list の値が異なると、UPDATE 文によって影響を受けるローは、あるサブスクライバの値から別のサブスクライバの値へ「移動」します。つまり、Message Agent は、値 **rep1** によって Customer レコードの値 **cust1** に対するサブスクリプションを作成したすべてのリモート・ユーザに DELETE を送信し、値 **rep2** によってサブスクリプションを作成したすべてのリモート・ユーザに INSERT を送信します。

BEFORE list と AFTER list の値が同じ場合は、リモート・ユーザがすでにローを保有しているため、UPDATE が送信されます。

複数のサブスクリプション間におけるローの共有

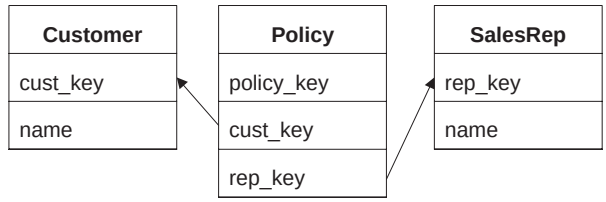
ある 1 つのローを複数のサブスクリプションに含めなければならない場合があります。たとえば、多対多の関係を持つ場合です。この項では、このような状況での処理方法について説明します。

Policy データベースの例

Policy データベースでは、データベースに多対多関係がある場合、テーブルを分割する理由とその方法について説明します。

データベースの例

次に、この問題について説明する簡単なデータベースを示します。



各営業担当者は複数の顧客を担当していますが、複数の営業担当者と取り引きしている顧客もいます。この場合、**Customer** と **SalesRep** の関係は、多対多の関係になります。

データベース内のテーブル

3 つのテーブルの詳細は次のとおりです。

テーブル	説明

SalesRep	社内のすべての営業担当者。**SalesRep** テーブルには、次のカラムがあります。
	
	• **rep_key** 各営業担当の識別子。これがプライマリ・キーです。
	• **name** 各営業担当の名前。
	
	このテーブルを作成する SQL 文は次のとおりです。
	
	`CREATE TABLE SalesRep (`
	`Rep_key CHAR(12) NOT NULL,`
	`Name CHAR(40) NOT NULL,`
	`PRIMARY KEY (rep_key)`
	`);`

Customer	会社と取引があるすべての顧客。**Customer** テーブルには、次のカラムがあります。
	
	• **cust_key** 各顧客の識別子を含んだ、プライマリ・キーのカラム。
	• **name** 各顧客の名前を含んだカラム。
	
	このテーブルを作成する SQL 文は次のとおりです。
	
	`CREATE TABLE Customer (`
	`Cust_key CHAR(12) NOT NULL,`
	`Name CHAR(40) NOT NULL,`
	`PRIMARY KEY (cust_key)`
	`);`

テーブル	説明

Policy	顧客と営業担当者間の多対多の関係を管理する、3 つのカラムで構成されたテーブル。**Policy** テーブルには次のカラムがあります。
	
	• **policy_key**　取り引きの識別子を含んだ、プライマリ・キーのカラム。
	• **cust_key**　取り引きを行う顧客の識別子を含んだカラム。
	• **rep_key**　取り引きを行う営業担当者の識別子を含んだカラム。
	
	このテーブルを作成する SQL 文は次のとおりです。
	
	`CREATE TABLE Policy (`
	`policy_key CHAR(12) NOT NULL,`
	`cust_key CHAR(12) NOT NULL,`
	`rep_key CHAR(12) NOT NULL,`
	`FOREIGN KEY ( cust_key )`
	`REFERENCES Customer ( cust_key )`
	`FOREIGN KEY ( rep_key )`
	`REFERENCES SalesRep (rep_key ),`
	`PRIMARY KEY ( policy_key )`
	`);`

レプリケーションの目標

このレプリケーションの設計上の目的は、各営業担当に次の情報を提供することです。

- **SalesRep** テーブル全体。
- データに対するサブスクリプションが作成されている営業担当者を含む、取り引きを記録した **Policy** テーブルのロー。
- データに対するサブスクリプションが作成されている営業担当者と取り引きする顧客をリストした、**Customer** テーブルのロー。

新しい問題

顧客と営業担当者間の多対多の関係では、適切に情報を共有する上で新しい問題が発生します。

- データを分割するサブスクリプションで使用される、営業担当者の値を参照しないテーブル (この場合は Customer テーブル) があります。

  再びパブリケーションでサブクエリを使用して、この問題に対処します。

- **Customer** テーブルの各ローが、**SalesRep** テーブルの複数のローと関連付けされ、複数の営業担当者のデータベースで共有されている場合があります。

  つまり、「サブスクリプション式を含まないテーブルの分割」132 ページの項に示す **Contact** テーブルのローが、パブリケーションによって別々のセットに分割されます。この項の例では、重複しているサブスクリプションがあります。

レプリケーションの目的を達成するには、1 つのパブリケーションと 1 つのサブスクリプション・セットが再び必要です。この場合、2 つのトリガを使用して、ある営業担当者から別の担当者へ顧客を移動させます。

パブリケーション

1 つのパブリケーションによって、データ共有の基本を指定します。

```
CREATE PUBLICATION SalesRepData (
     TABLE SalesRep,
     TABLE Policy SUBSCRIBE BY rep_key,
     TABLE Customer SUBSCRIBE BY (
         SELECT rep_key FROM Policy
         WHERE Policy.cust_key =
             Customer.cust_key
     ),
 );
```

サブスクリプション文は前述の例と同一です。

パブリケーションの仕組み

パブリケーションには、3 つのテーブルの一部または全部が含まれています。パブリケーションの仕組みを理解するには、次の各アーティクルを参照してください。

- **SalesRep テーブル**　このアーティクルに対応した修飾子はありません。このため、**SalesRep** テーブル全体がパブリケーションに含まれます。

```
...
    TABLE SalesRep,
...
```

- **Policy テーブル**　このアーティクルは、サブスクリプション式を使用して、営業担当者間でデータの分割に使用するカラムを指定します。

```
...
    TABLE Policy
    SUBSCRIBE BY rep_key,
...
```

このサブスクリプション式によって、**rep_key** カラムの値がサブスクリプションで指定された値に一致するテーブルのローだけを、各営業担当者が受信します。

Policy テーブルの分割は「**切断**」です。複数のサブスクライバが共有するローはありません。

Customer テーブル　サブクエリを持つサブスクリプション式は、分割を定義するときに使用します。このアーティクルは、次のように定義されます。

```
...
    TABLE Customer SUBSCRIBE BY (
        SELECT rep_key
        FROM Policy
        WHERE Policy.cust_key =
            Customer.cust_key
    ),
...
```

Customer テーブルの分割は「**非切断**」です。複数のサブスクライバが共有するローがいくつかあります。

複数の値を返すパブリケーションのサブクエリ

Customer アーティクルのサブクエリは、その結果セットに単一のカラム **(rep_key)** を返します。ただし、特定の顧客を担当する営業担当者全員に対応して、複数のローを返す場合があります。サブスクリプション式に複数の値がある場合は、この値のいずれかに一致するサブ

スクリプションを持つすべてのサブスクライバにローがレプリケートされます。複数の値を持つサブスクリプション式の場合、テーブルの分割を非切断にできます。

多対多の関係を使用した領域の再編成

「Contact データベースの例における領域の再編成」136 ページの項で示すように、領域の再編成 (サブスクライバ間におけるローの再割り当て) の問題には特に注意が必要です。

インストール環境全体を通して正しいデータを管理するため、領域の再編成 (サブスクライバ間におけるローの再割り当て) が実行可能な場合は、トリガを書き込む必要があります。

顧客の移動方法

この例では、**Policy** テーブルのローを挿入／削除して顧客を移動させます。

顧客と営業担当者間の取り引きを取り消すには、**Policy** テーブルのローを削除します。この場合、**Policy** テーブルの変更内容は正しくレプリケートされ、営業担当者のデータベースにこのローが表示されなくなります。ただし、**Customer** テーブルは変更されません。このため、**Customer** テーブルに対する変更は、サブスクライバにレプリケートされません。

トリガがないと、**Customer** テーブルに不正確なデータを持つサブスクライバが残されます。**Policy** テーブルに新しいローを追加する場合にも、同様の問題が発生します。

トリガを使用した問題の解決法

Policy テーブルを変更すると起動されるトリガを書き込んで、この問題を解決します。このトリガには、UPDATE 文の特別な構文を入れます。この特別な UPDATE 文によって、データベース・テーブルは変更されません。代わりに、サブスクライバ・データベースでのデータ管理用に SQL Remote が使用するトランザクション・ログに、エントリが作成されます。

BEFORE INSERT トリガ

次のようにトリガを作成し、**Policy** テーブルに対する INSERT を追跡します。これで、リモート・データベースに適切なデータが含まれるようにします。

```
CREATE TRIGGER InsPolicy
 BEFORE INSERT ON Policy
 REFERENCING NEW AS NewRow
 FOR EACH ROW
 BEGIN
     UPDATE Customer
     PUBLICATION SalesRepData
     SUBSCRIBE BY (
         SELECT rep_key
         FROM Policy
         WHERE cust_key = NewRow.cust_key
         UNION ALL
         SELECT NewRow.rep_key
     )
     WHERE cust_key = NewRow.cust_key;
 END;
```

BEFORE DELETE トリガ

次に、**Policy** テーブルに対する DELETE を追跡するトリガを示します。

```
CREATE TRIGGER DelPolicy
 BEFORE DELETE ON Policy
 REFERENCING OLD AS OldRow
 FOR EACH ROW
 BEGIN
     UPDATE Customer
     PUBLICATION SalesRepData
     SUBSCRIBE BY (
         SELECT rep_key
         FROM Policy
         WHERE cust_key = OldRow.cust_key
         AND Policy_key <> OldRow.Policy_key
     )
     WHERE cust_key = OldRow.cust_key;
 END;
```

トリガの機能の一部は、前述の項で示す内容と同じです。主な新機能は、INSERT トリガがサブクエリを含んでいること、およびこのサブクエリに複数の値を使用できることです。

複数の値を返すサブクエリ

BEFORE INSERT トリガのサブクエリは UNION 式であり、複数の値を返すことができます。

```
...
SELECT rep_key
FROM Policy
WHERE cust_key = NewRow.cust_key
UNION ALL
SELECT NewRow.rep_key
...
```

- UNION 式の 2 番目の部分は、INSERT 文から取得した、顧客と取り引きする新しい営業担当者の **rep_key** 値です。
- UNION 式の最初の部分は、Policy テーブルから取得した、顧客と取り引きする既存の営業担当者のセットです。

  つまり、サブスクリプション・クエリの結果セットには、新しい営業担当者だけでなく、ローを受信する営業担当者全員が含まれる必要があるということです。

BEFORE DELETE トリガ内のサブクエリは、複数の値を返します。

```
...
SELECT rep_key
FROM Policy
WHERE cust_key = OldRow.cust_key
AND rep_key <> OldRow.rep_key
...
```

- サブクエリは、**Policy** テーブルから **rep_key** の値を取得します。削除される値 (AND **rep_key** <> **OldRow.rep_key**) を除き、移動される顧客 (WHERE **cust_key** = **OldRow.cust_key**) と取り引きする営業担当者全員のプライマリ・キー値が、その値に含まれます。

  ここでも、サブスクリプション・クエリの結果セットには、DELETE に続くローを受信する営業担当者と一致した値のすべてが含まれる必要があります。

注意

- **Customer** テーブルのデータは、(プライマリ・キーの値などによって) 個々のサブスクライバと関連付けられることはなく、複数のサブスクライバ間で共有されます。このため、レプリケーション・メッセージ間の複数のリモート・サイトでデータが更新される可能性があり、レプリケーションの競合が発生することがあります。(特定のユーザだけに Customer テーブルの

更新権を付与するなどの方法で) パーミッションを使用するか、データベースに RESOLVE UPDATE トリガを追加してプログラム上で競合を処理することによって、この問題に対処できます。

- Policy テーブル上の UPDATE については、ここでは説明していません。そのような操作は避けるか、例で示したように、BEFORE UPDATE トリガで BEFORE INSERT と BEFORE DELETE の機能を組み合わせる必要があります。

多対多の関係における Subscribe_by_remote オプションの使用

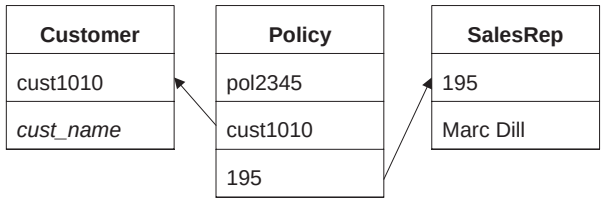
Subscribe_by_remote オプションを ON に設定すると、subscribe by 値が NULL または空の文字列であるローに対して、リモート・データベースから操作が行われた場合、リモート・ユーザがローに対するサブスクリプションを作成するとみなされます。デフォルトでは、Subscribe_by_remote オプションは ON に設定されています。多くの場合、この設定は必要に応じて変更されます。

Subscribe_by_remote オプションは、これを使用しない場合にいくつかのパブリケーション (Policy データベースの例を含む) で発生する問題を解決します。この項では、問題の詳細を説明し、また、オプションを使用してその問題を自動的に回避する方法について説明します。

各顧客が複数の営業担当者に所属できるため、パブリケーションでは、**Customer** テーブルのサブスクリプション式にサブクエリを使用します。

```
CREATE PUBLICATION SalesRepData (
     TABLE SalesRep,
     TABLE Policy SUBSCRIBE BY rep_key,
     TABLE Customer SUBSCRIBE BY (
         SELECT rep_key FROM Policy
         WHERE Policy.cust_key =
             Customer.cust_key
     ),
 );
```

営業担当者 Marc Dill が、新しい顧客との取り引きをデータに入力します。まず、Marc Dill はデータベースに新しい **Customer** ローを挿入し、**Policy** テーブルにもローを挿入して新規の顧客を自分自身に割り当てます。



Customer ローの INSERT は、Message Agent によって統合データベースで実行されるため、INSERT を実行するとき、Adaptive Server Anywhere はトランザクション・ログにサブスクリプション値を記録します。

あとで Message Agent がログをスキャンすると、サブスクリプション式からサブスクライバのリストが構築されます。顧客を割り当てた Policy テーブルでローがまだ適用されていないため、この場合 Marc Dill はリストされません。Subscribe_by_remote オプションが OFF に設定されていると、この新しい顧客は DELETE オペレーションとして Marc Dill に返されます。

Subscribe_by_remote オプションが ON に設定されているかぎり、Message Agent は、ローの所属先はローを挿入した営業担当者であるとみなし、INSERT は Marc Dill にレプリケートされません。また、レプリケーション・システムは影響を受けません。

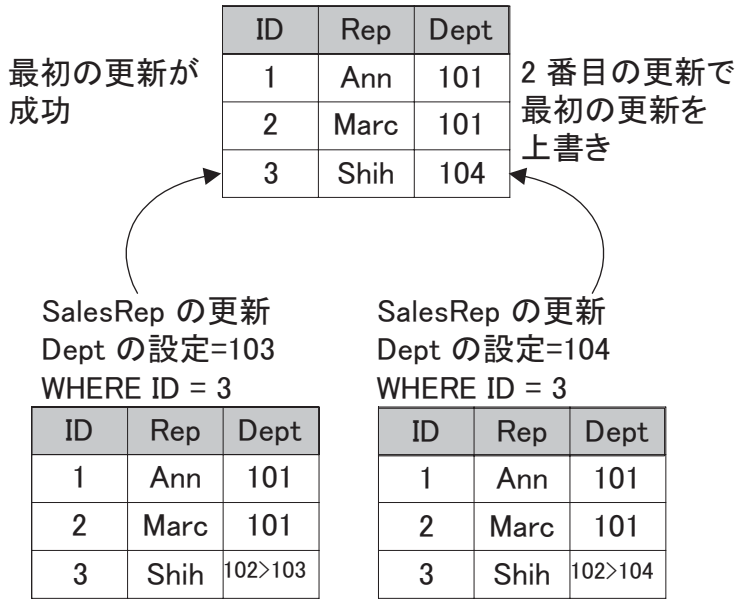
Subscribe_by_remote オプションが OFF に設定されている場合は、必ず Policy ローを挿入してから Customer ローを挿入し、トランザクションの最後までチェックを延期することによって参照整合性違反を回避してください。

競合の管理

UPDATE 競合は、次のような順序でイベントが実行されると発生します。

1. ユーザ 1 が、リモート・サイト 1 でローを更新します。
2. ユーザ 2 が、リモート・サイト 2 で同じローを更新します。
3. ユーザ 1 による更新内容が、統合データベースにレプリケートされます。
4. ユーザ 2 による更新内容が、統合データベースにレプリケートされます。

SQL Remote Message Agent が UPDATE 文をレプリケートするときは、ローごとに別の UPDATE 文をレプリケートします。またメッセージには、比較するために古いローの値が含まれています。ユーザ 2 による更新内容が統合データベースで受信される場合、ローの値はメッセージに記録されたものと異なります。



ID	Rep	Dept
1	Ann	101
2	Marc	101
3	Shih	104

ID	Rep	Dept
1	Ann	101
2	Marc	101
3	Shih	102>103

ID	Rep	Dept
1	Ann	101
2	Marc	101
3	Shih	102>104

デフォルトによる競合解決

デフォルトでは、UPDATE の処理は継続されるため、ユーザ 2 による更新内容 (最終的に統合データベースに到達する) が統合データベースの値になります。また、この値が、ローにサブスクリプションを作成するその他のデータベースすべてにレプリケートされます。

通常、デフォルトの競合解決方法では、最後に実行されたオペレーション (この場合はユーザ 2 によるオペレーション) を正常に処理して、競合の発生がレポートされません。ユーザ 1 による更新内容は失われます。SQL Remote では、変更されるデータにとって意味のある方法で競合を解決するためにトリガを使用して、カスタム競合解決を行うことができます。

プライマリ・キーの更新に競合解決は適用されない
UPDATE 競合は、プライマリ・キーの更新には適用 *されません*。SQL Remote のインストール環境で、プライマリ・キーを更新しないでください。適切な設計をすることによって、プライマリ・キーの競合をインストール環境から取り除く必要があります。

この項では、統合データベースにおける、SQL Remote のインストール環境への競合解決方法の構築について説明します。

SQL Remote の競合処理方法

競合が検出された場合

SQL Remote のレプリケーション・メッセージには、単一ローの更新セットとして UPDATE 文が含まれ、各文には更新前の値を含む VERIFY 句が指定されています。

UPDATE 競合は、データベースのローと一致させるための VERIFY 句の値のエラーとして、データベース・サーバによって検出されます。

統合データベースにおける競合だけが Message Agent によって検出されて解決されます。リモート・データベースからのメッセージで UPDATE 競合が検出されると、そのデータベースで次の 2 つのアクションが実行されます。

1. 競合解決 (RESOLVE UPDATE) トリガが起動されます。
2. UPDATE が適用されます。

RESOLVE UPDATE トリガの有無にかかわらず、UPDATE 文は VERIFY 句の値が一致しなくても適用されます。

競合解決はいくつかの手法を取ることができます。次に例を示します。

- アプリケーションによっては、競合をテーブルにレポートします。
- リモート・サイトでの更新内容よりも、統合データベースでの更新内容を優先して保持します。
- 商品のデリバリや受注時に在庫数を分析するといった、より高度な競合解決をします。

Adaptive Server Enterprise の統合データベースでは、競合解決の方法が異なります。詳細については、「SQL Remote の競合処理方法」211 ページを参照してください。

競合解決の実装

この項では、カスタム競合解決を Adaptive Server Anywhere の SQL Remote で実装するために必要な作業について説明します。Adaptive Server Enterprise の SQL Remote でも概念は同じですが、実装方法が異なります。

SQL Remote では、「**競合解決トリガ**」を定義して UPDATE 競合を処理できます。競合解決トリガは、リモート・ユーザがメッセージを適用する場合に、統合データベースでのみ起動されます。統合データベースで UPDATE 競合が検出されると、次の順序でイベントが実行されます。

1. オペレーションに対して定義された、すべての競合解決トリガが起動されます。
2. UPDATE が実行されます。
3. UPDATE だけでなくそのトリガのすべてのアクションが、リモート・データベース全部にレプリケートされます。この中には、その競合をトリガするメッセージを送信した、リモート・データベースも含まれます。

通常、Adaptive Server Anywhere の SQL Remote は、トリガ・アクションをレプリケートしません。トリガがリモート・データベースにあることを想定しているからです。競合解決トリガは統合データベースでのみ起動されるため、このトリガ・アクションはリモート・データベースにレプリケートされます。

4. リモート・データベースでは、統合データベースからのメッセージに UPDATE 競合が含まれていると、RESOLVE UPDATE トリガは起動されません。

5. UPDATE がリモート・データベースで実行されます。

プロセスの最後では、セットアップを通してデータの一貫性が保たれます。

データが読み取り用に共有されていても、各ロー (プライマリ・キーによって識別される) が 1 つのサイトだけで更新される場合は、UPDATE 競合は発生しません。複数のサイトでデータが更新された場合にのみ、この競合が発生します。

競合解決トリガの使用

この項では、RESOLVE UPDATE トリガまたは「**競合解決**」トリガの使用方法について説明します。

VERIFY 句を持つ UPDATE 文

UPDATE 文の VERIFY 句に含まれる値にエラーが発生すると、競合解決トリガが起動されて、更新前にデータベースの値を一致させます。次に、VERIFY 句を持つ UPDATE 文を示します。

```
UPDATE table-list
    SET column-name = expression, …
    [ VERIFY (column-name, …)
      VALUES ( expression, …) ]
    [ WHERE search-condition ]
```

VERIFY 句が使用できるのは、*table-list* にテーブルが 1 つしか存在しない場合だけです。VERIFY 句は、指定したカラムの値と、予測される値のセットを比較します。この予測値は、UPDATE 文が適用されたときにパブリッシャ・データベースに存在した値です。VERIFY 句を指定すると、テーブルが一度に 1 つずつ更新されます。

VERIFY 句を使用できるのは、単一のローを更新する場合だけです。ただし、このことがクライアント・アプリケーションを記述する場合に制約になることはありません。これは、複数のローの UPDATE 文がデータベースに入力されても、Message Agent によって単一のローに対する UPDATE 文の繰り返しとして解釈されるためです。

競合解決トリガの構文

RESOLVE UPDATE トリガの構文は、次のとおりです。

```
CREATE TRIGGER trigger-name
    RESOLVE UPDATE
    OF column-name ON table-name
    [ REFERENCING [ OLD AS old_val ]
        [ NEW AS new_val ]
        [ REMOTE AS remote_val ] ]
    FOR EACH ROW
    BEGIN
      …
    END
```

RESOLVE UPDATE トリガの起動後に、各ローが更新されます。REFERENCING 句を使用すると、更新するテーブルのロー値 (OLD)、更新用のロー値 (NEW)、VERIFY 句によれば存在するはずのロー (REMOTE) にアクセスできます。REMOTE AS 句の中で参照できるのは、VERIFY 句内のカラムだけです。その他のカラムの場合、「カラムが見つかりません」というエラーが表示されます。

VERIFY_ALL_COLUMNS オプションの使用

データベース・オプション VERIFY_ALL_COLUMNS は、デフォルトでは OFF に設定されています。このオプションを ON に設定すると、レプリケート・データベースですべてのカラムが検証され、カラムが 1 つでも異なる場合は、常に RESOLVE UPDATE トリガが起動されます。設定が OFF の場合は、更新されたカラムだけが検証されます。

このオプションを ON に設定すると、各 UPDATE に対して送信される情報が多くなるため、メッセージが大きくなります。

このオプションは、統合データベースで設定してからリモート・データベースを抽出すると、リモート・データベースでも設定されます。

VERIFY_ALL_COLUMNS オプションは、PUBLIC グループ用、または Message Agent の接続文字列に含まれるユーザ用のいずれかに設定できます。

CURRENT REMOTE USER 特殊定数の使用

CURRENT REMOTE USER 特殊定数は、メッセージを送信するリモート・ユーザの ID を保持します。競合のレポートをテーブルに配置する RESOLVE UPDATE トリガにこの特殊定数を使用して、競合を発生させたユーザを特定できます。

競合解決の例

この項では、RESOLVE UPDATE トリガを使用して競合を処理する方法について説明します。

日付の競合解決

交渉管理システムのテーブルの 1 つに、各顧客との最新の交渉日のデータを保持するカラムがあるとします。

ある担当者が金曜日に顧客と交渉を行いましたが、この担当は、変更を次の月曜日まで統合データベースにアップロードしません。一方、別の担当者が土曜日に同じ顧客と交渉し、変更をその日の夕方に更新しました。

土曜日に UPDATE が統合データベースにレプリケートされても競合は発生しませんが、月曜日に UPDATE が受信された時点で、ローがすでに変更されていることがわかります。

デフォルトでは月曜日の UPDATE が処理され、金曜日を最新の交渉日とする、間違った情報を持つカラムが残されます。

このカラムの UPDATE 競合は、最新の日付をローに挿入して解決してください。

解決の実装

次の RESOLVE UPDATE トリガは、新しい 2 つの値から最新のものを選択して、データベースにその値を入力します。

```
CREATE TRIGGER contact_date RESOLVE UPDATE
 ON contact
 REFERENCING OLD AS old_name
 NEW AS new_name
 FOR EACH ROW
 BEGIN
     IF new_name.contact_date <
```

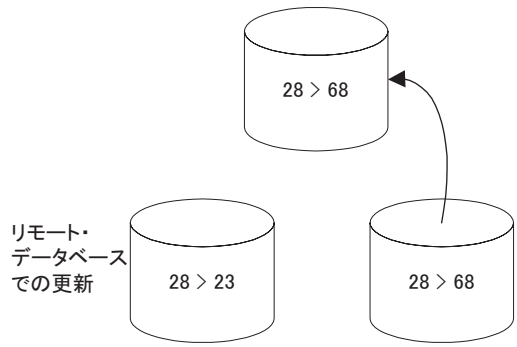
```
                old_name.contact_date THEN
            SET new_name.contact_date
                    = old_name.contact_date
        END IF
    END
```

更新される値が置き換える値よりも新しい場合は、新しい値はリセットされて、変更されないままエントリが残されます。

在庫数の競合解決

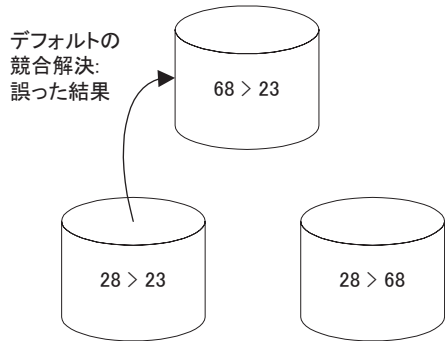
スポーツ用品メーカのための倉庫システムがあるとします。製品情報のテーブルには、各製品の在庫数を記録する **quantity** カラムがあります。このカラムへの更新は、通常は在庫数を引いていくことであり、新しく製品が搬入される場合は、在庫数を追加します。

リモート・データベースで営業担当者が受注を入力し、S サイズのタンクトップ T シャツの在庫を 5 個差し引いて 28 から 23 としました。この在庫数も担当者のデータベースに入力されます。一方、この更新内容が統合データベースにレプリケートされる前に、T シャツが新しく搬入されたため、倉庫では **quantity** カラムの値を 40 追加して 68 になるように入力したとします。

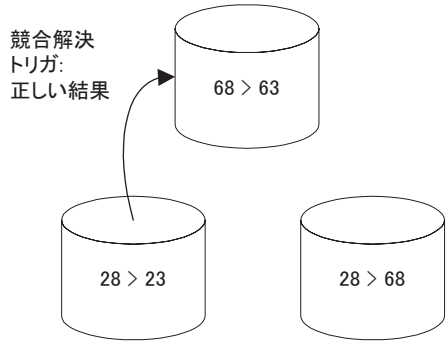


このエントリがデータベースに追加され、現在の **quantity** カラムは、在庫に 68 枚の S サイズのタンクトップ T シャツがあることを示しています。ここで営業担当者からの更新が受信されると、28 から 23 に変更するという内容にもかかわらず、実際のカラム値は 68 であるという競合が Adaptive Server Anywhere で検出されます。

デフォルトでは、最新の UPDATE が処理され、在庫レベルが間違った値 23 に設定されます。



この場合の競合は、データベースの最終的な値が 63 になるように、在庫のカラムに対する変更を合計して最終結果を算出することによって解決してください。



解決の実装

このような状況に適した RESOLVE UPDATE トリガでは、2 つの更新内容による増加分を追加します。次に例を示します。

```
CREATE TRIGGER resolve_quantity
 RESOLVE UPDATE OF quantity
 ON "DBA".product
 REFERENCING OLD AS old_name
 NEW AS new_name
 REMOTE AS remote_name
```

```
FOR EACH ROW
BEGIN
    SET new_name.quantity =    new_name.quantity
                                + old_name.quantity
                                - remote_name.quantity
END
```

統合データベースの以前の値 (68) と、元の UPDATE が実行された時点のリモート・データベースの以前の値 (28) の差が、このトリガによって送信する新しい値に追加されてから、UPDATE が実行されます。したがって、**new_val.quantity** は 63 (= 23 + 68 - 28) となり、この値が **quantity** カラムに入力されます。

リモート・データベースでの一貫性は、次のように管理されます。

1. 元のリモート UPDATE によって、値が 28 から 23 に変更されました。
2. 倉庫システムのエントリがリモート・データベースにレプリケートされますが、以前の値が予測値と一致しないためにエラーが発生します。
3. RESOLVE UPDATE トリガによる変更内容が、リモート・データベースにレプリケートされます。

競合のレポート

場合によっては、SQL Remote による競合解決のデフォルトの方法を変更しないほうがよいときもあります。この場合は、テーブルに競合を格納し、これをレポートするようにできます。こうすると、競合が発生した場合にはその内容を調査でき、必要に応じて競合を解決できます。

参照整合性エラーを回避する設計

リレーショナル・データベースのテーブルは、外部キーの参照で関連付けられています。これらの参照の結果として適用される参照整合性制約により、データベースの一貫性が保たれます。データベースの一部のみをレプリケートする場合は、レプリケート・データベースの参照整合性に問題が発生する場合があります。

パブリケーションの設計中に参照整合性の問題に注意すれば、これらの問題を回避できます。この項では、よくある整合性の問題と、それを回避する方法について説明します。

参照テーブルがレプリケートされないエラー

「テーブル全体のパブリッシュ」115 ページの項で示す **sales** パブリケーションには、**sales_order** テーブルが含まれます。

```
CREATE PUBLICATION pub_sales (
     TABLE customer,
     TABLE sales_order,
     TABLE sales_order_items,
     TABLE product
 )
```

sales_order テーブルには、**employee** テーブルに対する外部キーがあります。営業担当者の ID は、**employee** テーブルのプライマリ・キーを参照する、**sales_order** テーブルの外部キーです。ただし、**employee** テーブルはパブリケーションに含まれていません。

この方法でパブリケーションを作成すると、リモート・データベースの **sales_order** テーブルから外部キー参照が削除されないかぎり、新規受注のレプリケートは失敗します。

抽出ユーティリティを使用してリモート・データベースを作成する場合、外部キー参照はリモート・データベースから自動的に取り除かれ、この問題は発生しません。ただし、データベースには、**sales_order** テーブルの **sales_rep_id** カラムに無効な値が挿入されるのを防ぐ制約はありません。また、このような場合は、統合データベースで INSERT の実行に失敗します。パブリケーションに employee テーブル (または少なくともプライマリ・キー) を含めることによって、この問題を回避できます。

エラーを回避するトリガの設計

トリガによって実行されるアクションはレプリケートされません。SQL Remote セットアップ内のあるデータベースに存在するトリガは、レプリケーション・プロシージャによって、セットアップ内の別のデータベースにも存在しているとみなされます。統合データベースでトリガを起動するアクションが、レプリケート・サイトにレプリ

ケートされると、そのトリガは自動的に起動します。デフォルトでは、データベース抽出ユーティリティがトリガ定義を抽出するため、リモート・データベースにもトリガ定義が配置されます。

パブリケーションにデータベースのサブセットだけが含まれる場合、統合データベースのトリガは、統合データベースでテーブルやローを参照できますが、リモート・データベースでは参照できません。IF 文を使用して条件付きのトリガ・アクションを作成し、このエラーを回避するようなトリガを設計できます。次に、統合データベースとリモート・データベースで機能するトリガの設計方法について示します。

- CURRENT PUBLISHER の値を使用して、条件付きのトリガ・アクションにします。この場合、トリガはリモート・データベース上で特定のアクションを実行しません。
- NULL 値を返さない **object_id** 関数を使用して、条件付きのトリガ・アクションにします。**object_id** 関数は、テーブルやその他のオブジェクトを引数として取得し、そのオブジェクトの ID 番号か NULL 値 (オブジェクトが存在しない場合) を返します。
- ローが存在するかどうか決定する SELECT 文を使用して、条件付きのトリガ・アクションにします。

RESOLVE UPDATE トリガは、UPDATE 競合の解決に使う特殊なトリガ・タイプです。RESOLVE UPDATE トリガについては、「競合解決の例」156 ページの項で説明されています。RESOLVE UPDATE トリガのアクションは、競合を引き起こしたデータベースなど、リモートのデータベースにレプリケートされます。

ユニークなプライマリ・キーの確保

プライマリ・キーの値はユニークである必要があります。すべてのユーザが同じデータベースに接続する場合、ユニークな値の維持に関する問題は発生しません。ユーザが値の再使用を試みる場合に、INSERT 文のエラーが発生します。

レプリケーション・システムにおいては状況が異なります。これは、ユーザが複数のデータベースに接続しているためです。異なるデータベースに接続している 2 人のユーザが、同じプライマリ・キーの値を使用してローを挿入すると問題が発生します。各データベースでは、値がユニークなので各文が正常に実行されます。

ただし、異なるデータベースに接続している 2 人のユーザが、同じプライマリ・キーの値を使用してローを挿入すると、レプリケーション・システムで問題が発生します。レプリケーション・システムの指定されたデータベースで受信される、2 番目の INSERT の実行は失敗します。SQL Remote は不定期で接続するユーザのためのレプリケーション・システムなので、インストール環境のすべてのデータベースに対するロック・メカニズムはありません。したがって、プライマリ・キーの重複エラーが発生しないように SQL Remote のインストール環境を設計する必要があります。

SQL Remote のインストール環境以外で設計されたプライマリ・キー・エラーの場合は、複数のサイトで修正するテーブルのプライマリ・キーがユニークであることを保証する必要があります。この目的を達成する方法はいくつかあります。この章では、一般的で効率的な、信頼性の高い 2 つの方法について説明します。

1. Adaptive Server Anywhere のデフォルトのグローバル・オートインクリメント機能を使用します。

2. プライマリ・キー・プールを使用して、各サイトで未使用のユニークなプライマリ・キー値のリストを管理します。

これらの方法のいずれか一方または両方を使用して、値の重複を回避できます。

グローバル・オートインクリメント・デフォルト・カラム値の使用

Adaptive Server Anywhere では、デフォルトのカラム値を GLOBAL AUTOINCREMENT に設定できます。このデフォルト設定は、ユニークな値を管理するカラムのすべてに適用できますが、特にプライマリ・キーの場合に役立ちます。この機能の目的は、通常 Mobile Link 同期によってデータが複数のデータベース間でレプリケートされるときに、ユニークな値を生成するタスクを簡素化することです。

このデフォルトのグローバル・オートインクリメントを指定すると、そのカラムの値のドメインが分割されます。各分割には同じ数の値が含まれます。たとえば、データベース内の整数カラムの分割サイズを 1000 に設定した場合、1 つの分割が 1001 から 2000 まで拡大します。また、2 つ目の分割は 2001 から 3000 まで拡大し、以降、同じように拡大していきます。

データベースの各コピーに、ユニークなグローバル・データベース ID 番号を割り当てます。Adaptive Server Anywhere では、データベースのデフォルト値は、そのデータベース番号でユニークに識別された分割からのみ設定されます。たとえば、上の例のデータベースに ID 番号 10 を割り当てた場合、このデータベースのデフォルト値は 10001 ～ 11000 の範囲から選択されます。このデータベースの別のコピーで、ID 番号 11 が割り当てられたデータベースからは、11001 ～ 12000 の範囲にある同一カラムのデフォルト値が指定されます。

デフォルトのグローバル・オートインクリメントを宣言する

Sybase Central でカラムのプロパティを選択するか、TABLE 文または ALTER TABLE 文に DEFAULT GLOBAL AUTOINCREMENT 句を組みこむことで、作業データベースにデフォルト値を設定できます。

オプションで、AUTOINCREMENT キーワードの直後にカッコで分割サイズを指定できます。この分割サイズには任意の正の整数を設定できますが、通常、分割サイズは、サイズの値がすべての分割で不足しないように選択されています。

INT または UNSIGNED INT 型のカラムの場合、デフォルトの分割サイズは 2^{16} = 65536 です。他の型のカラムの場合、デフォルトの分割サイズは 2^{32} = 4294967296 です。特にカラムが INT または BIGINT 型以外のときは、これらのデフォルトが適切ではない場合があるため、分割サイズを明示的に指定してください。

たとえば、次の文では 2 つのカラム (顧客 ID 番号を保持する整数カラム、顧客名を保持する文字列カラム) を持つ簡単なテーブルが作成されます。

```
CREATE TABLE customer (
     id   INT          DEFAULT GLOBAL AUTOINCREMENT
(5000),
     name VARCHAR(128) NOT NULL,
     PRIMARY KEY (id)
 )
```

上の例では、選択された分割サイズは 5000 です。

GLOBAL AUTOINCREMENT の詳細については、『ASA SQL リファレンス・マニュアル』> 「CREATE TABLE 文」を参照してください。

Global_database_id 値の設定

アプリケーションを配備するときには、各データベースに対して必ず異なる ID 番号を割り当てる必要があります。ID 番号の作成と配布はさまざまな方法で実行できます。テーブルに値を設定し、ユーザ名など、ユニークなプロパティに基づいて、各データベースに適切なローをダウンロードするのも 1 つの方法です。

❖ グローバル・データベース ID 番号を設定するには、次の手順に従います。

- パブリック・オプション **Global_database_id** の値を設定して、データベースの ID 番号を設定します。ID 番号は正の整数にします。

たとえば、次の文ではデータベースの ID 番号が 20 に設定されます。

```
SET OPTION PUBLIC.Global_database_id = 20
```

特定カラムの分割サイズが 5000 の場合、このデータベースのデフォルト値は 100001 ～ 105000 の範囲から選択されます。

データベース抽出時のユニークなデータベース ID 番号の設定

抽出ユーティリティを使用してリモート・データベースを作成する場合は、ストアド・プロシージャを作成してタスクを自動化できます。sp_hook_dbxtract_begin という名前のストアド・プロシージャを作成すると、抽出ユーティリティによって自動的に呼び出されます。プロシージャが呼び出される前に、次の内容の入った #hook_dict という名前のテンポラリ・テーブルを、抽出ユーティリティが作成します。

名前	値
extracted_db_global_id	抽出されるユーザ ID

sp_hook_dbxtract_begin プロシージャを書き込んでローの value カラムを修正する場合、その値は抽出されたデータベースの GLOBAL_DATABASE_ID オプションとして使用され、GLOBAL DEFAULT AUTOINCREMENT 値のプライマリ・キー値範囲の開始位置にマークを付けます。

例

101 という **user_id** を持つリモート・ユーザ **user2** のために、データベースを抽出するとします。sp_hook_dbxtract_begin プロシージャを定義しない場合、抽出されるデータベースは、**Global_database_id** が **101** に設定されます。

sp_hook_dbxtract_begin プロシージャを定義しても、このプロシージャが #hook_dict のいずれのローも修正しない場合、オプションは **101** に設定されたままになります。

データベースを次のように設定するとします。

```
set option "PUBLIC"."Global_database_id" = '1';
 create table extract_id ( next_id integer not null) ;
 insert into extract_id values( 1 );
 create procedure sp_hook_dbxtract_begin
 as
     declare @next_id  integer
     update extract_id set next_id = next_id + 1000
     select @next_id = (next_id )
```

```
from extract_id
commit
update #hook_dict
set value = @next_id
where name = 'extracted_db_global_id'
```

この設定を使用すると、データベースが抽出されるたびに、**Global_database_id** の値が毎回異なります。1 回目は 1001、2 回目は 2001 という具合です。

プロシージャ・フックのデバッグに役立つように、*dbxtract* を冗長モードで動作するよう設定すると、次のものが出力されます。

- 検出されたプロシージャ・フック
- プロシージャ・フックが呼び出される前の #hook_dict の内容
- プロシージャ・フックが呼び出された後の #hook_dict の内容

デフォルト値の選択方法

各データベース内のパブリック・オプション **Global_database_id** は、ユニークな正の整数に設定してください。特定のデータベースのデフォルト値の範囲は、*pn* + 1 から *p*(*n* + 1) です。ここで、*p* は分割サイズ、*n* はパブリック・オプション **Global_database_id** の値を表します。たとえば、分割サイズを 1000、**Global_database_id** を 3 に設定すると、範囲は 3001 から 4000 になります。

Global_database_id が正の整数に設定されている場合、Adaptive Server Anywhere は次のルールを適用してデフォルト値を選択します。

- カラムに現在の分割の値が含まれていない場合、最初のデフォルト値は *pn* + 1 である。
- カラムに現在の分割の値が含まれていても、そのすべてが *p*(*n* + 1) 未満であれば、この範囲内でこれまで使用した最大値より 1 大きい値が次のデフォルト値になる。

- デフォルトのカラム値は、現在の分割以外のカラムの値の影響を受けない。つまり、*pn*+1 より小さいか *p*(*n* + 1) より大きい数には影響されません。Mobile Link 同期を介して別のデータベースからレプリケートされた場合に、このような値が存在する可能性があります。

public オプション **Global_database_id** がデフォルト値の 2147483647 に設定されると、NULL 値がカラムに挿入されます。NULL 値が許可されていない場合に、ローの挿入を試みるとエラーが発生します。たとえば、テーブルのプライマリ・キーにカラムが含まれている場合に、この状況が発生します。

public オプション **Global_database_id** は、負の値に設定できないため、選択された値は常に正になります。ID 番号の最大値を制限するのは、カラムのデータ型と分割サイズだけです。

デフォルトの NULL 値は、分割で値が不足したときにも生成されます。この場合には、別の分割からデフォルト値を選択できるように、データベースに **Global_database_id** の新しい値を割り当ててください。カラムで NULL が許可されていない場合、NULL 値を挿入しようとするとエラーが発生します。未使用の値が残り少ないことを検出し、このような状態を処理するには、**GlobalAutoincrement** タイプのイベントを作成します。

特定の分割で値が不足する場合は、新しいデータベース ID をそのデータベースに割り当てることができます。方法が適切なものであれば、新しいデータベース ID 番号を割り当てることができます。未使用のデータベース ID 値のプールを管理する方法も、その 1 つです。このプールは、プライマリ・キー・プールと同じ方法で管理されます。

分割で値が不足しそうな場合に、自動的にデータベース管理者へ通知する (またはその他のアクションを実行する) ようにイベント・ハンドラを設定できます。詳細については、『ASA データベース管理ガイド』> 「イベントのトリガ条件の定義」を参照してください。

詳細については、『ASA データベース管理ガイド』> 「GLOBAL_DATABASE_ID オプション [データベース]」を参照してください。

プールの詳細については、「プライマリ・キー・プールの使用」168 ページを参照してください。

プライマリ・キー・プールの使用

「**プライマリ・キー・プール**」は、SQL Remote のインストール環境で、各データベースのプライマリ・キー値のセットを保持するテーブルです。リモート・ユーザは、各自のプライマリ・キー値のセットを受信します。リモート・ユーザは、新しいローをテーブルに挿入する場合、ストアド・プロシージャを使用してプールから有効なプライマリ・キーを選択します。プールは、使用できる値を補充するプロシージャを、統合データベースで定期的に実行することによって管理されます。

営業担当者とその顧客からなる簡単なデータベースの例を用いて、この方法について説明します。これらのテーブルは実際のデータベースで使用するものよりずっと単純なので、レプリケーションに関する重要な問題だけに注目できます。

プライマリ・キー・プールのテーブル

プライマリ・キー・プールは別のテーブルに記録されます。次の CREATE TABLE 文は、プライマリ・キー・プールのテーブルを作成します。

```
CREATE TABLE KeyPool (
     table_name VARCHAR(40) NOT NULL,
     value INTEGER NOT NULL,
     location CHAR(12) NOT NULL,
     PRIMARY KEY (table_name, value),
 );
```

このテーブルの各カラムの意味は、次のとおりです。

カラム	説明
table_name	プライマリ・キー・プールで管理するテーブルの名前を保持する。ここに示す簡単なデータベースの例では、統合データベースだけに営業担当者が新しく追加されると、**Customer** テーブルだけがプライマリ・キー・プールを必要とし、このカラムが重複する。これは、一般的な解決方法を示すために含まれている。
value	プライマリ・キー値のリストを保持する。それぞれの値は、**table_name** にリストされた各テーブルに対してユニークである。
location	受信者の識別子。セットアップによっては、この値が **SalesRep** テーブルの **rep_key** 値と同じになる場合がある。また、営業担当者以外のユーザがいて、この 2 つの識別子が別々でなければならないセットアップもある。

パフォーマンスを向上させるために、次のようにテーブルにインデックスを作成できます。

```
CREATE INDEX KeyPoolLocation
 ON KeyPool (table_name, location, value);
```

プライマリ・キー・プールのレプリケーション

キー・プールは、既存のパブリケーションに組み込むか、または別のパブリケーションとして共有することができます。この例では、プライマリ・キー・プール用に別のパブリケーションを作成します。

❖ プライマリ・キー・プールをレプリケートするには、次の手順に従います (SQL の場合)。

1　プライマリ・キー・プールのデータ用に、パブリケーションを作成します。

```
CREATE PUBLICATION KeyPoolData (
     TABLE KeyPool SUBSCRIBE BY location
 );
```

2 各リモート・データベースのサブスクリプションを、KeyPoolData パブリケーションに作成します。

```
CREATE SUBSCRIPTION
 TO KeyPoolData( 'user1' )
 FOR user1;
 CREATE SUBSCRIPTION
 TO KeyPoolData( 'user2' )
 FOR user2;
 ...
```

サブスクリプションの引数は、location の識別子です。

状況によっては、既存のパブリケーションに KeyPool テーブルを追加し、同じ引数を使用して各パブリケーションにサブスクリプションを作成する方法は有効です。ここでは、location の値と rep_key の値を別にして、より一般的な解決方法について説明しています。

参照

- 『ASA SQL リファレンス・マニュアル』> 「CREATE PUBLICATION 文」
- 「CREATE SUBSCRIPTION 文」455 ページ

キー・プールの入力と補充

ユーザが新しい顧客を追加するたびに、使用できるプライマリ・キーのプールは 1 つずつ減少します。次のようなプロシージャを使用して、統合データベースでプライマリ・キー・プールのテーブルに定期的に補充する必要があります。

```
CREATE PROCEDURE ReplenishPool()
 BEGIN
     FOR EachTable AS TableCursor
     CURSOR FOR
         SELECT table_name
         AS CurrTable, max(value) as MaxValue
         FROM KeyPool
         GROUP BY table_name
     DO
         FOR EachRep AS RepCursor
         CURSOR FOR
```

```
                SELECT location
                AS CurrRep, count(*) as NumValues
                FROM KeyPool
                WHERE table_name = CurrTable
                GROUP BY location
            DO
                // make sure there are 100 values.
                // Fit the top-up value to your
                // requirements
                WHILE NumValues < 100 LOOP
                    SET MaxValue = MaxValue + 1;
                    SET NumValues = NumValues + 1;
                    INSERT INTO KeyPool
                    (table_name, location, value)
                    VALUES
                    (CurrTable, CurrRep, MaxValue);
                END LOOP;
            END FOR;
        END FOR;
    END;
```

このプロシージャによって、各ユーザのプールの値が 100 個まで増えます。指定する値は、ユーザがデータベースのテーブルにローを挿入する頻度によって異なります。

KeyPool テーブルのプライマリ・キー値のプールを再補充するため、**ReplenishPool** プロシージャを統合データベースで定期的に実行する必要があります。

ReplenishPool プロシージャには、各サブスクライバに対して少なくとも 1 つ以上のプライマリ・キー値が必要です。これによって、最大値を検索し、値を追加して次のセットを生成できます。はじめにプールを値で埋めるには、各ユーザに対して値を 1 つ挿入してから、**ReplenishPool** を呼び出して残りを埋めます。次の例では、3 人のリモート・ユーザと単一の統合ユーザ **Office** について示します。

```
INSERT INTO KeyPool VALUES( 'Customer', 40, 'user1' );
 INSERT INTO KeyPool VALUES( 'Customer', 41, 'user2' );
 INSERT INTO KeyPool VALUES( 'Customer', 42, 'user3' );
 INSERT INTO KeyPool VALUES( 'Customer', 43, 'Office');
 CALL ReplenishPool();
```

トリガを使用してキー・プールを補充できない
トリガ・アクションはレプリケートされないため、キー・プールの補充にはトリガを使用できません。

新しい顧客の追加

営業担当者が Customer テーブルに新しく顧客を追加する場合、挿入するプライマリ・キー値は、ストアド・プロシージャを使用して取得します。次の例では、プライマリ・キー値を提供するストアド・プロシージャと、INSERT を実行するストアド・プロシージャについて説明します。

このプロシージャは、営業担当者の識別子がリモート・データベースの CURRENT PUBLISHER であることを利用します。

- **NewKey プロシージャ** **NewKey** プロシージャは、キー・プールにある整数値を提供し、プールからその値を削除します。

```
CREATE PROCEDURE NewKey(
         IN @table_name VARCHAR(40),
         OUT @value INTEGER )
 BEGIN
     DECLARE NumValues INTEGER;
         SELECT count(*), min(value)
     INTO NumValues, @value
             FROM KeyPool
             WHERE table_name = @table_name
             AND location = CURRENT PUBLISHER;
     IF NumValues > 1 THEN
         DELETE FROM KeyPool
         WHERE table_name = @table_name
         AND value = @value;
     ELSE
     // Never take the last value, because
     // ReplenishPool will not work.
     // The key pool should be kept large enough
     // that this never happens.
         SET @value = NULL;
     END IF;
 END;
```

- **NewCustomer プロシージャ**　**NewCustomer** プロシージャは、プライマリ・キーを構成する **NewKey** が取得した値を使用して、テーブルに新しい顧客を追加します。

```
CREATE PROCEDURE NewCustomer(
        IN customer_name CHAR( 40 ) )
 BEGIN
     DECLARE new_cust_key INTEGER ;
     CALL NewKey( 'Customer', new_cust_key );
     INSERT
     INTO Customer (
         cust_key,
         name,
         location
         )
     VALUES (
         'Customer ' ||
         CONVERT (CHAR(3), new_cust_key),
         customer_name,
         CURRENT PUBLISHER
         );
 );
 END
```

NewKey から取得される **new_cust_key** 値をテストしてこの値が NULL でないことを確認し、値が NULL の場合には挿入しないように、プロシージャを強化できます。

プライマリ・キー・プールの概要

プライマリ・キー・プールを使用するには、次のコンポーネントが必要です。

- **キー・プール・テーブル**　インストール環境の各データベースに対する、有効なプライマリ・キー値を保持するテーブル。
- **補充プロシージャ**　値を補充したキー・プール・テーブルを維持するストアド・プロシージャ。
- **キー・プールの共有**　インストール環境の各データベースは、キー・プール・テーブルから取得した、そのデーターベース自体の有効な値のセットに対してサブスクリプションを作成する必要があります。

- **データ入力プロシージャ** 次に有効なプライマリ・キー値をプールから選択し、キー・プールからその値を削除するストアド・プロシージャを使用して、新しいローを入力します。

サブスクリプションの作成

パブリケーションに対するサブスクリプションを作成するには、各サブスクライバに REMOTE パーミッションが付与されている必要があります。また、そのユーザ用にサブスクリプションを作成する必要があります。サブスクリプションの詳細は、パブリケーションにサブスクリプション式を使用するかどうかによって異なります。

Sybase Central におけるサブスクリプションの活用

❖ **Sybase Central でサブスクリプションを作成／管理するには、次の手順に従います。**

1　左ウィンドウ枠で、[パブリケーション] フォルダを開きます。

2　必要なパブリケーションを選択します。Sybase Central では次のタスクを実行できます。

3　右ウィンドウ枠で、[SQL Remote サブスクリプション] タブをクリックします。適切な設定を次のように設定できます。

- パブリケーションにリモート・ユーザのサブスクリプションを作成するには、[ファイル] メニューで [新規] ー [SQL Remote サブスクリプション] を選択し、[新しい SQL Remote サブスクリプションの作成] ウィザードの指示に従います。
- リモート・ユーザのサブスクリプションを削除するには、[サブスクライバ] リストでユーザを右クリックし、ポップアップ・メニューで [削除] をクリックします。
- サブスクリプションを手動で起動、停止、または同期するには、[サブスクライバ] リスト内のユーザを選択し、ポップアップ・メニューで [プロパティ] を選択します。

  [詳細] タブをクリックします。このタブで、サブスクリプションを起動、停止、同期するために、[すぐに起動] [すぐに停止] [すぐに同期化] をそれぞれクリックします。

ボタンをクリックすると、すぐにサブスクリプションが処理されます。プロパティ・シートで [キャンセル] を続けてクリックしても、開始／停止／同期のアクションはキャンセル*されません*。

サブスクリプション式を使用しないサブスクリプション

パブリケーションにサブスクリプション式がない場合に、パブリケーションに対するユーザのサブスクリプションを作成するには、次の情報が必要です。

- **ユーザ ID** このユーザのサブスクリプションがパブリケーションに対して作成されます。このユーザには、REMOTE パーミッションが付与されている必要があります。
- **パブリケーション名** このパブリケーション名でユーザのサブスクリプションが作成されます。

次の文は、ユーザ ID **SamS** のサブスクリプションを **pub_orders_samuel_singer** パブリケーションに作成します。このパブリケーションは WHERE 句を使用して作成されます。

```
CREATE SUBSCRIPTION
 TO pub_orders_samuel_singer
 FOR SamS
```

サブスクリプション式を使用するサブスクリプション

パブリケーションにサブスクリプション式がある場合に、パブリケーションへユーザのサブスクリプションを作成するには、次の情報が必要です。

- **ユーザ ID** このユーザのサブスクリプションがパブリケーションに対して作成されます。このユーザには、REMOTE パーミッションが付与されている必要があります。
- **パブリケーション名** このパブリケーション名でユーザのサブスクリプションが作成されます。
- **サブスクリプション値** パブリケーションのサブスクリプション式に対するテスト用の値です。たとえば、従業員 ID を含むカラム名をパブリケーションがサブスクリプション式として保持する場合、サブスクリプションを作成するユーザの従業員 ID の値が、サブスクリプションに提供されなければなりません。サブスクリプション値は常に文字列です。

次の文は、Samuel Singer (ユーザ ID **SamS**、従業員 ID 856) のサブスクリプションを pub_orders パブリケーションに作成します。このパブリケーションには、サブスクリプション式 **sales_rep** が定義され、Samuel Singer 自身の受注に対してローを要求します。

```
CREATE SUBSCRIPTION
 TO pub_orders ( '856' )
 FOR SamS
```

サブスクリプションの開始

正しく更新を受信して適用するため、各サブスクライバにはデータの初期コピーが必要です。同期の処理については、「データベースの同期」238 ページで説明しています。

詳細については、「CREATE SUBSCRIPTION 文」455 ページを参照してください。

第 8 章

Adaptive Server Enterprise の SQL Remote 設計

この章の内容

この章では、Adaptive Server Enterprise サーバに統合データベースがある場合の SQL Remote インストール環境の設計方法について説明します。

Adaptive Server Anywhere と同様の資料

Adaptive Server Anywhere と Adaptive Server Enterprise では、パブリケーション設計の原則はほとんど同じですが、コマンドと機能に違いがあります。この章は、Adaptive Server Anywhere のユーザを対象とした章 「Adaptive Server Anywhere の SQL Remote 設計」113 ページとの間で多くの部分が重複しています。

設計の概要

SQL Remote インストール環境を設計するには、次の作業を行います。

- **パブリケーションの設計**　パブリケーションは、データベース間で共有する情報を決定します。
- **サブスクリプションの設計**　サブスクリプションは、各ユーザが受信する情報を決定します。
- **設計の実装**　システムの全ユーザ用に、パブリケーションとサブスクリプションを作成します。

すべての管理タスクは統合データベースで実行する

すべての SQL Remote 管理タスクと同様、設計はデータベース管理者またはシステム管理者が統合データベースで実行します。Sybase のシステム管理者が、すべての SQL Remote 設定タスクを実行するようにしてください。

Adaptive Server Enterprise 環境の詳細については、Adaptive Server Enterprise のマニュアルを参照してください。

パブリケーションの作成

この項の内容

この項では、テーブル全体またはテーブルのカラムワイズ・サブセットで構成される簡単なパブリケーションの作成方法について説明します。

「Adaptive Server Enterprise ユーザのためのチュートリアル」67 ページでは、簡単なパブリケーションについても説明します。

テーブル全体のアーティクルの作成

最も簡単な種類のアーティクルには、データベース・テーブルのローとカラムがすべて含まれます。

❖ テーブルのローとカラムのすべてを含むアーティクルを作成する

1. レプリケーション対象としてテーブルにマーク付けします。それには、**sp_add_remote_table** プロシージャを次のように実行します。

   ```
   sp_add_remote_table table-name
   ```

2. テーブルをパブリケーションに追加します。それには、**sp_add_article** プロシージャを次のように実行します。

   ```
   sp_add_article publication-name, table-name
   ```

例

- 次のコマンドでは、テーブル **SalesRep** を **SalesRepData** パブリケーションに追加します。

  ```
  sp_add_remote_table 'SalesRep'
   sp_add_article 'SalesRepData', 'SalesRep'
   go
  ```

テーブル内の一部のカラムを含むアーティクルの作成

テーブル内の一部のカラムだけを含むアーティクルを作成するには、**sp_add_article_col** を使用して対象のカラムをリストする必要があります。カラムをリストしない場合、アーティクルにはテーブルのすべてのカラムが入ります。

❖ テーブル内の一部のカラムとすべてのローを含むアーティクルを作成するには、次の手順に従います。

1 レプリケーション対象としてテーブルにマーク付けします。それには、**sp_add_remote_table** プロシージャを次のように実行します。

```
sp_add_remote_table table-name
 go
```

2 テーブルをパブリケーションに追加します。それには、**sp_add_article** プロシージャを次のように実行します。

```
sp_add_article publication-name, table-name
 go
```

sp_add_article プロシージャを実行すると、テーブルがパブリケーションに追加されます。デフォルトでは、テーブルのすべてのカラムがパブリケーションに追加されます。カラムの一部だけを追加する場合は、**sp_add_article_col** プロシージャを使用してパブリケーションに入れるカラムを指定します。

3 カラムを個々にパブリケーションに追加します。追加するカラムごとに **sp_add_article_col** プロシージャを次のように実行します。

```
sp_add_article_col publication-name,
     table-name,
     column-name
 go
```

例

- 次のコマンドでは、テーブル **SalesRep** の **rep_key** カラムだけを **SalesRepData** パブリケーションに追加します。

```
sp_add_remote_table 'SalesRep'
 sp_add_article 'SalesRepData',
     'SalesRep'
 sp_add_article_col 'SalesRepData',
     'SalesRep',
     'rep_key'
 go
```

テーブル内の一部のローを含むアーティクルの作成

テーブル内の一部のローだけをアーティクルに入れるには、次の 2 つの方法があります。

- **WHERE 句**　WHERE 句を使用して、アーティクルにローのサブセットを入れることができます。このアーティクルを含むパブリケーションのすべてのサブスクライバは、WHERE 句を満たすローを受信します。
- **サブスクリプション・カラム**　サブスクリプション・カラムを使用して、アーティクルを含むパブリケーションに対する別のサブスクリプションに、別のローのセットを入れることができます。

使用できる句

SQL Remote for Adaptive Server Enterprise では、それぞれの場合に次のような制限事項があります。

- **WHERE 句の制限事項**　次の形式の WHERE 句だけがサポートされています。

  ```
  WHERE column-name IS NOT NULL.
  ```

- **サブスクリプション・カラム**　SQL Remote for Adaptive Server Anywhere では、カラム名以外の式をサポートします。Adaptive Server Enterprise の場合、サブスクリプション式がカラム名になります。

WHERE と SUBSCRIBE BY を使用する場合

パブリケーションのサブスクライバがそれぞれ、1 つのテーブルから異なるローを受け取る場合には、サブスクリプション式を使用してください。サブスクリプション式は、テーブルを分割するための最も強力な手段です。

WHERE 句を使用したアーティクルの作成

WHERE 句を使用すると、パブリケーションのすべてのサブスクリプションからローのセットを除外できます。

❖ WHERE 句を使用してアーティクルを作成するには、次の手順に従います。

1 レプリケーション対象としてテーブルにマーク付けしていない場合は、まずマーク付けします。それには、**sp_add_remote_table** プロシージャを次のように実行します。

```
sp_add_remote_table table_name
```

2 テーブルをパブリケーションに追加します。追加するには、**sp_add_article** プロシージャを実行します。このプロシージャの 3 番目の引数に **WHERE** *column* **IS NOT NULL** 句に対応するカラム名を指定します。

```
sp_add_article publication_name,
     table_name,
     column_name
```

IS NOT NULL は指定しないでください。これは暗黙的なものです。カラム名だけを指定してください。

3 テーブルにカラムのサブセットだけを入れる場合は、**sp_add_article_col** プロシージャを使用してカラムを指定します。WHERE 句で指定したカラムをアーティクルに入れてください。

例

- 次の一連の文では、1 つのアーティクルを含むパブリケーションを作成します。これには、カラム **col_1** が NULL ではない **test_table** のローだけが入ります。

```
sp_create_publication test_pub
 sp_add_remote_table test_table
 sp_add_article test_pub, test_table, col_1
 go
```

サブスクリプション・カラムを使用したアーティクルの作成

サブスクリプション・カラムは、多数のリモート・データベース間でローを共有する場合に使用されます。

❖ サブスクリプション・カラムを使用してアーティクルを作成するには、次の手順に従います。

1　レプリケーション対象としてテーブルにマーク付けしていない場合は、まずマーク付けします。それには、**sp_add_remote_table** プロシージャを次のように実行します。

sp_add_remote_table *table_name*

2　テーブルをパブリケーションに追加します。追加するには、**sp_add_article** プロシージャを実行します。このプロシージャの 4 番目の引数にサブスクリプション式として使用するカラム名を指定します。

sp_add_article *publication_name*,
　table_name,
　NULL,
　column_name

NULL エントリを入れて、WHERE 句が追加されないようにします。

3　テーブルにカラムのサブセットだけを入れる場合は、**sp_add_article_col** プロシージャを使用してカラムを指定します。サブスクリプション式で指定したカラムをアーティクルに入れてください。

例

- 次の一連の文では、1 つのアーティクルを含むパブリケーションを作成します。これは、カラム **col_1** の値に基づいてサブスクリプションをサポートします。

```
sp_create_publication test_pub
 sp_add_remote_table test_table
 sp_add_article test_pub,
     test_table,
     NULL,
     col_1
 go
```

アーティクルに関する注意

- アーティクルでは WHERE 句とサブスクリプション式を結合できます。
- プライマリ・キーのカラムはすべてアーティクルに入れる必要があります。
- 次の場合以外はカラムのサブセットをアーティクルに入れないでください。
  - 残りのカラムがデフォルト値の場合、または NULL を使用できる場合。
  - リモート・データベースでは挿入が実行されない場合。プライマリ・キーの値を変更する更新が行われなければ、問題は発生しません。

  このような場合以外にカラムのサブセットをアーティクルに入れると、統合データベースでの INSERT 文は失敗します。

Adaptive Server Enterprise のパブリケーション設計

簡単なパブリケーションの作成方法がわかったら、パブリケーションの適切な設計方法について考える必要があります。この項では、パブリケーションの設計に関する問題と堅牢な設計を行う方法について説明します。

設計に関する問題の概要

各サブスクリプションは、完全なリレーショナル・データベース

リモート・データベースは、統合データベースとサブスクリプションに関する情報を共有します。サブスクリプションは、統合サイトで保持されるリレーショナル・データベースのサブセットであると同時に、リモート・サイトの完全なリレーショナル・データベースでもあります。したがって、サブスクリプション内の情報は、他のリレーショナル・データベースと同様に次の規則に従います。

- **外部キーの関係を有効にする**　データベースには、外部キーの各エントリに対応するプライマリ・キーのエントリが必要です。

  データベース抽出ユーティリティを使用すると、リモート・データベースに対する CREATE TABLE 文が、リモートに存在しないテーブルに定義された外部キーを持たないようになります。

- **プライマリ・キーの一意性を管理する**　別のサイトで入力され、まだレプリケートされていない新しいローを検査する方法はありません。異なるサイトのユーザが、同じプライマリ・キー値を使用してローを追加しないように設計してください。そうしないと、統合データベースにローをレプリケートする場合に競合が発生します。

ロックがない状態でトランザクションの整合性を管理する

特定のローに対してシステム全体をロックするメカニズムがなくても、すべてのサイトで更新が行われた場合に、分散データベース (統合データベースとすべてのリモート・データベースで構成される) のデータは、整合性を保つ必要があります。

- **ロック競合を回避または解決する**　SQL Remote のインストール環境には、各ユーザがローを同時に変更しないように、すべてのデータベースに渡ってローをロックする方法はありません。

このような競合は、競合が起こらないようにシステムを設計して回避するか、統合データベースで適切な方法で解決する必要があります。

これらリレーショナル・データベースの主要な機能を、パブリケーションとサブスクリプションの設計に組み込む必要があります。この項では、最適な設計方法と技術について説明します。

有効なアーティクルの条件

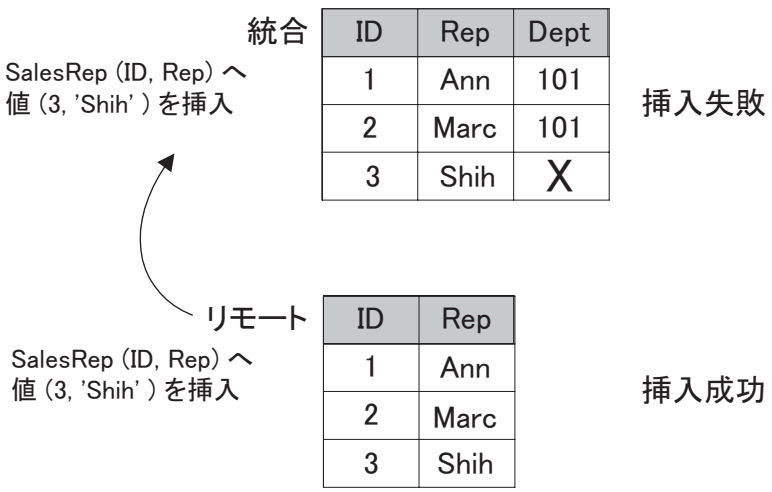
プライマリ・キーの全カラムが、アーティクルに含まれる必要があります。

リモート・データベースでの INSERT のサポート

リモート・データベースの INSERT 文が統合データベースへ正しくレプリケートされるために、有効な INSERT 文にならないカラムだけをアーティクルから除外できます。基本テーブルは次のとおりです。

- NULL を使用できるカラム。
- デフォルトのカラム。

これらの条件を満たさないカラムを除外すると、リモート・データベースで INSERT 文を実行して統合データベースにレプリケートすると失敗します。



ローの条件

一部のローだけをパブリケーションに入れるには、次の 2 つの方法があります。

- **WHERE 句**　WHERE 句を使用して、アーティクルにローのサブセットを入れることができます。このアーティクルを含むパブリケーションのすべてのサブスクライバは、WHERE 句を満たすローを受信します。

  SQL Remote for Adaptive Server Enterprise では、次の WHERE 句だけがサポートされています。

  ```
  WHERE column-name IS NOT NULL
  ```

- **サブスクリプション・カラム**　サブスクリプション・カラムを使用して、アーティクルを含むパブリケーションに対する別のサブスクリプションに、別のローのセットを入れることができます。

ローの制限の詳細については、「テーブル内の一部のローを含むアーティクルの作成」183 ページを参照してください。

サブスクリプション・カラムを含まないテーブルの分割

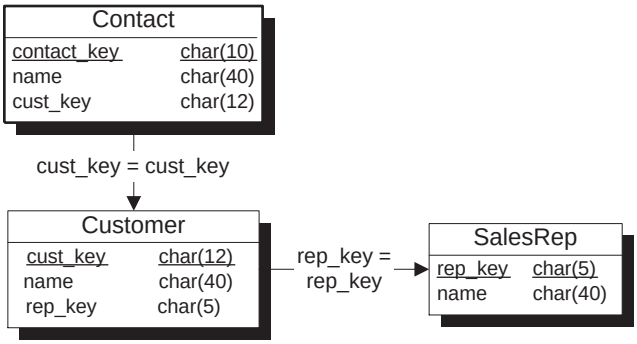
多くの場合、サブスクリプション・カラムがテーブルにないときでもテーブルのローを分割する必要があります。この項では、例を使ってこのような場合の処理方法について説明します。

Contact データベースの例

Contact データベースを例に、サブスクリプション・カラムを含まないテーブルを分割する理由とその方法について説明します。

例

次に、この問題について説明する簡単なデータベースを示します。このデータベースを Contact データベースと呼びます。これは、このデータベースにこの章ですでに説明した 2 つのテーブルに加え Contact テーブルが含まれているためです。



各営業担当者は、複数の顧客に対して販売活動を行います。連絡先が 1 箇所だけの顧客もいれば、複数ある顧客もいます。

データベース内のテーブル

3 つのテーブルの詳細は次のとおりです。

テーブル	説明

SalesRep	社内のすべての営業担当者。SalesRep テーブルには、次のカラムがあります。
	
	• **rep_key**　各営業担当の識別子。これがプライマリ・キーです。
	• **name**　各営業担当の名前。
	
	このテーブルを作成する SQL 文は次のとおりです。
	
	`CREATE TABLE SalesRep (`
	`rep_key CHAR(12) NOT NULL,`
	`name CHAR(40) NOT NULL,`
	`PRIMARY KEY (rep_key)`
	`)`
	`go`

Customer	会社と取引があるすべての顧客。Customer テーブルには、次のカラムがあります。
	
	• **cust_key**　各顧客の識別子。これがプライマリ・キーです。
	• **name**　各顧客の名前。
	• **rep_key**　取引を行う営業担当者の識別子。これが SalesRep テーブルへの外部キーです。
	
	このテーブルを作成する SQL 文は次のとおりです。
	
	`CREATE TABLE Customer (`
	`cust_key CHAR(12) NOT NULL,`
	`name CHAR(40) NOT NULL,`
	`rep_key CHAR(12) NOT NULL,`
	`FOREIGN KEY ( rep_key )`
	`REFERENCES SalesRep,`
	`PRIMARY KEY (cust_key)`
	`)`
	`go`

テーブル	説明

Contact	会社と取引があるすべての個別の連絡先。連絡先はそれぞれ 1 人の顧客に属します。Contact テーブルには、次のカラムがあります。
	
	• **contact_key** 各連絡先の識別子。これがプライマリ・キーです。
	• **name** 各連絡先の名前。
	• **cust_key** 連絡先が関連する顧客の識別子。これが Customer テーブルへの外部キーです。
	
	このテーブルを作成する SQL 文は次のとおりです。
	
	`CREATE TABLE Contact (`
	`contact_key CHAR(12) NOT NULL,`
	`name CHAR(40) NOT NULL,`
	`cust_key CHAR(12) NOT NULL,`
	`FOREIGN KEY (cust_key)`
	`REFERENCES Customer,`
	`PRIMARY KEY (contact_key)`
	`)`
	`go`

レプリケーションの目標

この設計上の目標は、各営業担当者に次の情報を提供することです。

- 完全な SalesRep テーブル
- 営業担当者に割り当てられている顧客 (Customer テーブルの情報)
- 関連する顧客の連絡先 (Contact テーブルの情報)
- 営業担当領域が再編成された場合の適切な情報管理

Contact データベースの例における領域の再編成

「**領域の再編成**」を実行すると、サブスクライバ間でローが再割り当てされます。この例での領域の再編成とは、営業担当者の間で顧客を再割り当てすることを意味します。領域の再編成は、**Customer** テーブルの **rep_key** カラムを更新すると実行されます。

UPDATE 文が、以前の営業担当者には DELETE 文として、新しい営業担当者へは INSERT 文としてレプリケートされ、Customer ローは正しく新しい営業担当者へ移動されます。

領域の再編成時に Contact テーブルのログ・エントリはない

顧客が再割り当てされても、**Contact** テーブルは影響を受けません。**Contact** テーブルには変更が加えられていないため、このテーブルに関するトランザクション・ログにはエントリがありません。この情報がないと、SQL Remote は、**Customer** テーブルだけでなく **Contact** テーブルのローも再割り当てできません。この失敗によって、参照整合性の問題が発生します。以前の営業担当のリモート・データベースで、**Customer** テーブルと対応しなくなった **cust_key** 値が、**Contact** テーブルに含まれています。

この項では、**Contact** テーブルのローを再割り当てする方法について説明します。

Contact データベースの例における Customer テーブルの分割

サブスクリプション・カラムに **rep_key** 値を使用すると、**Customer** テーブルを分割できます。**SalesRep** テーブルと **Customer** テーブルを含むパブリケーションは、次のとおりです。

```
exec sp_add_remote_table 'SalesRep'
 exec sp_add_remote_table 'Customer'
 go
 exec sp_create_publication 'SalesRepData'
 go
 exec sp_add_article 'SalesRepData', 'SalesRep'
 exec sp_add_article SalesRepData,
             Customer, NULL,
             'rep_key'
 go
```

Contact テーブルへの subscription-list カラムの追加

Contact テーブルも営業担当者の間で分割する必要がありますが、営業担当者の **rep_key** 値への参照は含みません。

subscription-list カラムを追加する

Adaptive Server Enterprise でこの問題を解決するには、ローに対してカンマで区切られたサブスクリプション値のリストを含むカラムを **Contact** テーブルに追加します (今回の例では、サブスクリプション値は 1 つだけになります)。カラムはトリガを使用して管理できるので、データベースに対するアプリケーションはカラムが存在していても影響を受けることはありません。このカラムのことを「**subscription-list カラム**」と呼びます。

Customer テーブルのローが挿入、更新、または削除されると、トリガによって **Contact** テーブルのローが更新されます。特に、トリガによって subscription-list カラムが更新されます。**Contact** テーブルはレプリケートするようマーク付けされているので、ローの更新前と更新後のイメージがログに記録されます。

ログ・エントリは、サブスクライバではなく値である
この場合、入力された値はサブスクライバに対応していますが、ログに入力されているのはサブスクライバのリストではありません。サーバはパブリケーションに関する情報だけを処理し、Message Agent はサブスクライバに関する全情報を処理します。ログに入力された値は、各サブスクリプションのサブスクリプション値との比較に使用されます。たとえば、テーブルのローが、営業担当者の間で地方や県単位で分けられた場合、地方や県の値がトランザクション・ログに入力されます。

「**subscription-list カラム**」は、カンマで区切られたサブスクライバのリストを保持するためだけにテーブルに追加されます。今回の例では、**Contact** テーブルの各ローに対するサブスクライバは 1 つだけなので、subscription-list カラムには 1 つの値だけが保持されます。

subscription-list カラムで複数の値が保持できるケースについては、「複数のサブスクリプション間におけるローの共有」200 ページを参照してください。

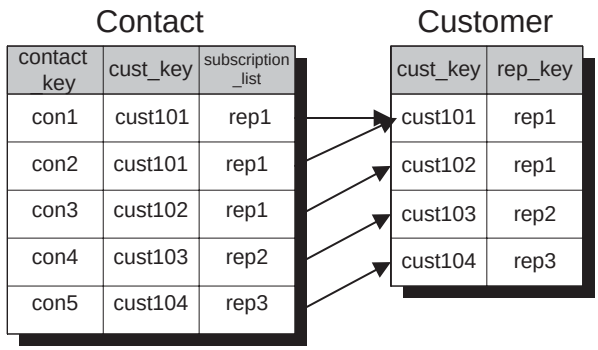
Contact テーブルの定義

Contact テーブルの場合、テーブル定義は次のように変更されます。

```
CREATE TABLE Contact (
     contact_key CHAR( 12 ) NOT NULL,
     name CHAR( 40 ) NOT NULL,
     cust_key CHAR( 12 ) NOT NULL,
     subscription_list CHAR( 12 ) NULL,
     FOREIGN KEY ( cust_key )
     REFERENCES Customer ( cust_key ),
     PRIMARY KEY ( contact_key )
 )
 go
```

NULL を使用できるカラムが追加で作成されるので、既存のアプリケーションは何も変更を加えずにデータベースに対して引き続き機能します。

subscription_list カラムには、Customer テーブルのプライマリ・キー値 **cust_key** を持つローに対応している **rep_key** 値が入っています。**subscription_list** カラムの管理は、一連のトリガによって処理されます。



Adaptive Server Anywhere の統合データベースでは、この問題の解決方法が異なります。詳細については、「サブスクリプション式を含まないテーブルの分割」132 ページを参照してください。

subscription-list カラムの管理

subscription_list カラムを最新の状態に保つには、次のオペレーションでトリガが必要になります。

- Contact テーブルでの INSERT
- Contact テーブルでの UPDATE
- Customer テーブルでの UPDATE

Customer テーブルの UPDATE 文は「**領域の再編成**」の問題に対処しています。この場合、顧客は異なる営業担当者に割り当てられます。

Contact テーブルの INSERT トリガ

Contact テーブルの INSERT トリガでは、subscription_list 値を Customer テーブルの対応する **rep_key** 値に設定します。

```
CREATE TRIGGER set_contact_sub_list
 ON Contact
 FOR INSERT
 AS
 BEGIN
     UPDATE Contact
     SET Contact.subscription_list = (
             SELECT rep_key
             FROM Customer
            WHERE Contact.cust_key = Customer.cust_key
)
     WHERE Contact.contact_key IN (
             SELECT contact_key
             FROM inserted
     )
 END
```

このトリガによって、挿入されるローの **subscription_list** カラムが更新されます。これらのローはサブクエリで識別されます。

```
SELECT contact_key
 FROM inserted
```

Contact テーブルの UPDATE トリガ

Contact テーブルの UPDATE トリガでは、**cust_key** カラムが変更されているか、**subscription_list** カラムが更新されているかを確認します。

```
CREATE TRIGGER update_contact_sub_list
 ON Contact
 FOR UPDATE
 AS
 IF UPDATE ( cust_key )
     BEGIN
         UPDATE Contact
         SET subscription_list = Customer.rep_key
         FROM Contact, Customer
         WHERE Contact.cust_key=Customer.cust_key
     END
```

トリガはジョインを使用して作成しますが、サブクエリも使用できます。

Customer テーブルの UPDATE トリガ

次のトリガでは、顧客の UPDATE を処理して、顧客を新しい営業担当者に移します。

```
CREATE TRIGGER transfer_contact_with_customer
 ON Customer
 FOR UPDATE
 AS
 IF UPDATE ( rep_key )
 BEGIN
     UPDATE Contact
     SET Contact.subscription_list = (
         SELECT rep_key
         FROM Customer
         WHERE Contact.cust_key = Customer.cust_key )
     WHERE Contact.contact_key IN (
             SELECT cust_key
         FROM inserted
     )
 END
```

抽出パフォーマンスのチューニング

ユーザを抽出または同期するとき、*subscription-list* カラムは完全にテーブルをスキャンする必要があるためパフォーマンス上の問題が発生することがあります。

多くのユーザを対象にデータベースを抽出していて、パフォーマンスに問題がある場合、「**サブスクリプション・ビュー**」を使用するとパフォーマンスが向上します。ビューにはサブクエリが必ず含まれま

す。このサブクエリは、抽出と同期のみに使用され、ログのスキャン中は無視されます。ただし、*subscription-list* カラムを管理するために、関連するテーブルでは、トリガを定義しておく必要があります。

❖ サブスクリプション・ビューを作成するには、次の手順に従います。

1 サブクエリを使用するクエリを設計して、テーブルからサブスクリプションに適したローを選択します。

これまでの例を基に説明します。次のクエリでは、rep_key 値 **rep5** によってサブスクリプションが作成されたユーザの Contact テーブルのローを選択します。

```
SELECT *
 FROM Contact
 WHERE 'rep5' = (SELECT rep_key
                      FROM Customer
                    WHERE cust_key = Contact.cust_key
)
```

2 このサブクエリを含むビューを作成します。次に例を示します。

```
CREATE VIEW Contact_sub_view AS
 SELECT *
 FROM dbo.Contact
 WHERE 'repxx' = ( SELECT rep_key
     FROM dbo.Customer
     WHERE cust_key = dbo.Contact.cust_key )
```

このビュー定義では、WHERE 句の左側にどのような値を使用するかは、重要ではありません (上の例では repxx)。レプリケーション・ツールでは、抽出と同期のためだけにサブクエリを使用します。SUBSCRIBE BY 値がサブクエリの結果セットと等しいローが、抽出または同期されます。

3 ビューの名前を sp_add_article または sp_modify_article のパラメータとして指定します。

```
exec sp_add_remote_table 'Contact'
 go
 exec sp_add_article SalesRepData,
                          'Contact',
                          NULL,
                          'subscription_list',
                          'Contact_sub_view'
```

subscription_list カラムはログのスキャンに使用され、サブクエリは抽出と同期に使用されます。

詳細については、「共有ローの抽出パフォーマンスのチューニング」207 ページ、「sp_add_article プロシージャ」480 ページ、および 「sp_modify_article プロシージャ」498 ページを参照してください。

複数のサブスクリプション間におけるローの共有

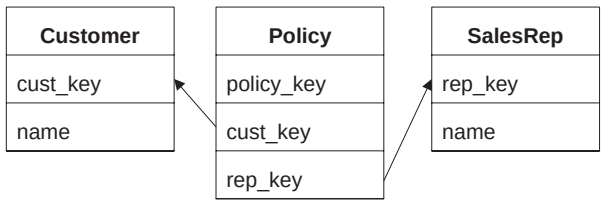
ある 1 つのローを複数のサブスクリプションに含めなければならない場合があります。たとえば、これまで見てきた顧客と営業担当者間の多対 1 関係だけではなく、多対多関係があることがあります。

Policy データベースの例

Policy データベースでは、データベースに多対多関係がある場合、テーブルを分割する理由とその方法について説明します。

データベースの例

次に、この問題について説明する簡単なデータベースを示します。



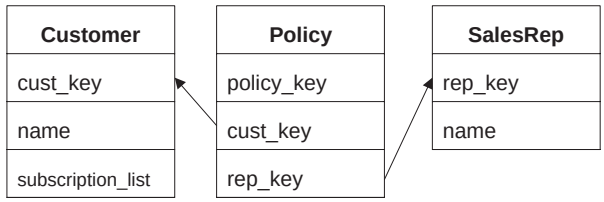
Policy テーブルには、各 Policy のセットごとに 1 つのローがあります。顧客に対する Policy はそれぞれ特定の営業担当者によって作成されます。顧客と営業担当者の間に多対多関係があり、さらにある特定の営業担当者／顧客間に複数の Policy が作成されることがあります。

Customer テーブルのローは、営業担当者と共有する必要がない場合もあれば 1 人または複数の営業担当者と共有する必要がある場合もあります。

問題の解決

このような場合には、トリガを作成して、カンマで区切った値のリストを構築して、Customer テーブルの重複 subscription-list カラムに格納します。次に、Customer テーブルをパブリケーションに追加するときに、サブスクリプション・カラムとしてこのカラムを入れる必要があります。ローは、サブスクリプション値が subscription-list カラムの値と一致するサブスクリプションと共有されます。

subscription-list カラムが入ったデータベースは、次のようになります。



Adaptive Server Enterprise VARCHAR カラムは 255 文字までに制限されています。これにより、カンマで区切られたリストに格納可能な値の数が制限されます。

テーブル定義

テーブル定義は次のとおりです。

```
CREATE TABLE SalesRep (
     rep_key CHAR( 12 ) NOT NULL,
     name CHAR( 40 ) NOT NULL,
     PRIMARY KEY ( rep_key )
)
go
CREATE TABLE Customer (
     cust_key CHAR( 12 ) NOT NULL,
     name CHAR( 40 ) NOT NULL,
     subscription_list VARCHAR( 255 ) NULL,
     PRIMARY KEY ( cust_key )
)
go
CREATE TABLE Policy (
     policy_key    INTEGER NOT NULL,
     cust_key CHAR( 12 ) NOT NULL,
     rep_key CHAR( 12 ) NOT NULL,
     FOREIGN KEY    ( cust_key )
     REFERENCES Customer (cust_key ),
     FOREIGN KEY (rep_key )
     REFERENCES SalesRep ( rep_key ),
     PRIMARY KEY    (policy_key)
)
```

注意：

- Customer テーブルの **subscription_list** カラムでは NULL を使用できるので、**subscription_list** カラムには営業担当者が指定されていない顧客を追加できます。

パブリケーション

このデータベースのパブリケーションは、次の一連の文を使って作成できます。

```
//Mark the tables for replication
 exec sp_add_remote_table 'SalesRep'
 exec sp_add_remote_table 'Policy'
 exec sp_add_remote_table 'Customer'
 go

// Create an empty publication
 exec sp_create_publication 'SalesRepData'

//Add the Sales Rep table to the publication
 exec sp_add_article 'SalesRepData', 'SalesRep'

//Add the Policy table to the publication
 exec sp_add_article 'SalesRepData', 'Policy',
       NULL, 'rep_key'

// Add the Customer table to the publication.
 // Subscribe by the subscription_list column
 // Exclude the subscription_list column
 exec sp_add_article 'SalesRepData', 'Customer',
       NULL, 'subscription_list'
 exec sp_add_article_col 'SalesRepData', 'Customer',
'cust_key'
 exec sp_add_article_col 'SalesRepData', 'Customer',
'name'
 go
```

このパブリケーションのサブスクリプションは、次のようなフォームになります。

```
exec sp_subscription 'create',
             'SalesRepData',
             'userID',
             'rep_key'
 go
```

userID はサブスクライバを識別します。*rep_key* はサブスクリプション・カラムであり、これは **SalesRep** テーブルの **rep_key** カラムの値になります。

subscription-list カラムの管理

Customer テーブルに追加された subscription-list カラムを管理するには、プロシージャと一連のトリガを作成する必要があります。この項では、これらの作成方法について説明します。

ストアド・プロシージャ

次のプロシージャを使用して subscription-list カラムを構築します。このプロシージャは、subscription_list カラムを管理するトリガから呼び出されます。

```
CREATE PROCEDURE SubscribeCustomer @cust_key CHAR(12)
 AS
 BEGIN
     --  Rep returns the rep list for customer @cust_key
     DECLARE Rep CURSOR FOR
         SELECT DISTINCT RTRIM( rep_key )
         FROM Policy
         WHERE cust_key = @cust_key
     DECLARE @rep_key    CHAR(12)
     DECLARE @subscription_list   VARCHAR(255)
         -- build comma-separated list of rep_key
     -- values for this Customer
     OPEN Rep
     FETCH Rep INTO @rep_key
     IF @@sqlstatus = 0 BEGIN
         SELECT @subscription_list = @rep_key
         WHILE 1=1 BEGIN
             FETCH Rep INTO @rep_key
             IF @@sqlstatus != 0 BREAK
             SELECT @subscription_list =
                 @subscription_list + ',' + @rep_key
         END
     END
     ELSE BEGIN
         SELECT @subscription_list = ''
     END
         -- update the subscription_list in the
```

```
        -- Customer table
        UPDATE Customer
        SET subscription_list = @subscription_list
        WHERE cust_key = @cust_key
    END
```

注意：

- プロシージャは、入力引数として Customer キーをとります。
- Rep とは、顧客が連絡を取る各営業担当者をリストするクエリのカーソルです。
- WHILE ループによって、カンマで区切られた営業担当者のリストを保持する VARCHAR(255) 変数が構築されます。

トリガ

次のトリガでは、Policy テーブルにローが挿入されるときに Customer テーブルの **subscription_list** カラムを更新します。

```
CREATE TRIGGER InsPolicy
 ON Policy
 FOR INSERT
 AS
 BEGIN
     -- Cust returns those customers inserted
     DECLARE Cust CURSOR FOR
         SELECT DISTINCT cust_key
         FROM inserted
     DECLARE @cust_key CHAR(12)
         OPEN Cust
     -- Update the rep list for each Customer
     -- with a new rep
     WHILE 1=1 BEGIN
         FETCH Cust INTO @cust_key
         IF @@sqlstatus != 0 BREAK
         EXEC SubscribeCustomer @cust_key
     END
 END
```

次のトリガでは、Policy テーブルからローが削除されるときに Customer テーブルの **subscription_list** カラムを更新します。

```
CREATE TRIGGER DelPolicy
 ON Policy
 FOR DELETE
 AS
 BEGIN
     -- Cust returns those customers deleted
     DECLARE Cust CURSOR FOR
         SELECT DISTINCT cust_key
         FROM deleted
     DECLARE @cust_key CHAR(12)
         OPEN Cust
     -- Update the rep list for each Customer
     -- losing a rep
     WHILE 1=1 BEGIN
         FETCH Cust INTO @cust_key
         IF @@sqlstatus != 0 BREAK
         EXEC SubscribeCustomer @cust_key
     END
 END
```

パブリケーションから subscription-list カラムを除外する

subscription-list カラムはレプリケートされて頻繁に更新されるので、このカラムをパブリケーションから除外してください。

たとえば、顧客ごとに複数の Policy があるとどうなるかを考えてみます。新しい営業担当者が顧客に割り当てられると、トリガが起動して、Customer テーブルの subscription-list カラムが更新されます。subscription-list カラムがパブリケーションの一部である場合、その顧客に割り当てられている全営業担当者に対して Policy ごとに 1 つの更新がレプリケートされます。

統合データベース専用のトリガ

subscription-list カラムの値は、トリガを使用して管理します。挿入または更新のトリガが Message Agent によって適用される場合、トリガは統合データベースで起動します。トリガは存在しないカラムを管理するので、リモート・データベースからは除外します。

sp_user_extraction_hook プロシージャを使用すると、抽出時にリモート・データベースから特定のトリガだけを除外できます。このプロシージャは、抽出処理の最終部分として呼び出されます。デフォルトでは、空です。

❖ 抽出プロシージャをカスタマイズして特定のトリガを除外するには、次の手順に従います。

1 **quoted_identifier** オプションに ON が設定されていることを確認します。

```
set quoted_identifier on
 go
```

2 プロシージャで参照されるテンポラリ・テーブルが存在しないと、CREATE PROCEDURE 文は失敗します。次のプロシージャで参照されるテンポラリ・テーブルは、*ssremote.sql* スクリプトで入手できます。このスクリプトから必要なテーブル定義をコピーしてから CREATE TABLE 文を実行すれば、プロシージャの作成前に現在の接続上に存在するようになります。

3 次のプロシージャを作成します。

```
CREATE PROCEDURE sp_user_extraction_hook
 AS
 BEGIN
   -- We do not want to extract the INSERT and
   -- DELETE triggers created on the Policy table
   -- that maintain the subscription_list
   -- column, since we do not include that
   -- column in the publication.
   -- If these objects were extracted the
   -- INSERTs would fail on the remote database
   -- since they reference a column
   -- ( subscription_list ) that does not exist.
   DELETE FROM #systrigger
   WHERE table_id = object_id( 'Policy' )
   -- Do not create any procedures
   DELETE FROM #sysprocedure
   WHERE proc_name = 'SubscribeCustomer'
 END
 go
```

共有ローの抽出パフォーマンスのチューニング

ユーザを抽出または同期するとき、*subscription-list* カラムは完全にテーブルをスキャンする必要があるためパフォーマンス上の問題が発生することがあります。

多くのユーザを対象にデータベースを抽出していて、パフォーマンスに問題がある場合、「**サブスクリプション・ビュー**」を使用するとパフォーマンスが向上します。ビューにはサブクエリが必ず含まれます。このサブクエリは、抽出と同期のみに使用され、ログのスキャン中は無視されます。ただし、*subscription-list* カラムを管理するために、関連するテーブルでは、トリガを定義しておく必要があります。

❖ サブスクリプション・ビューを作成するには、次の手順に従います。

1　サブクエリを使用するクエリを設計して、テーブルからサブスクリプションに適したローを選択します。

　これまでの例を基に説明します。次のクエリでは、rep_key 値 **rep5** によってサブスクリプションが作成されたユーザの Contact テーブルのローを選択します。

```
SELECT *
 FROM Contact
 WHERE 'rep5' = (SELECT rep_key
                      FROM Customer
                   WHERE cust_key = Contact.cust_key
)
```

2　このサブクエリを含むビューを作成します。次に例を示します。

```
CREATE VIEW Customer_sub_view AS
 SELECT *
 FROM dbo.Customer
 WHERE 'repxx' IN ( SELECT rep_key
     FROM dbo.Policy
     WHERE dbo.Policy.cust_key =
dbo.Customer.cust_key )
```

このビュー定義では、WHERE 句の左側にどのような値を使用するかは、重要ではありません (上の例では repxx)。レプリケーション・ツールでは、抽出と同期のためだけにサブクエリを使用します。サブクエリの結果セット内に SUBSCRIBE BY 値があるローを抽出または同期します。

3 ビューの名前を sp_add_article または sp_modify_article のパラメータとして指定します。

```
exec sp_add_article SalesRepData,
                            'Customer',
                            NULL,
                            'subscription_list',
                            'Customer_sub_view'
```

subscription_list カラムはログのスキャンに使用され、サブクエリは抽出と同期に使用されます。

詳細については、「抽出パフォーマンスのチューニング」197 ページ、「sp_add_article プロシージャ」480 ページ、および「sp_modify_article プロシージャ」498 ページを参照してください。

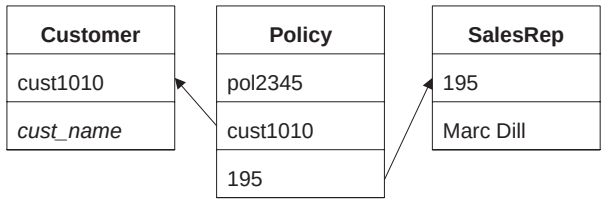
多対多の関係における Subscribe_by_remote オプションの使用

SUBSCRIBE_BY_REMOTE オプションを ON に設定すると、Subscribe by 値が NULL または '' であるローに対して、リモート・データベースから操作が行われた場合、リモート・ユーザがローにサブスクリプションを作成したとみなされます。デフォルトでは、SUBSCRIBE_BY_REMOTE オプションは ON に設定されています。多くの場合、この設定は必要に応じて変更されます。

SUBSCRIBE_BY_REMOTE オプションは、パブリケーション (Policy データベースの例を含む) で起こり得る問題を解決します。この項では、オプションが問題を自動的に回避する方法について説明します。

各顧客が複数の営業担当者に所属できるので、データベースでは、Customer テーブルの subscription-list カラムを使用します。

営業担当者 Marc Dill が、新しい顧客との取り引きをデータに入力します。まず、Marc Dill はデータベースに新しい Customer ローを挿入します。次に、Policy テーブルにローを挿入して、新しい顧客を Marc Dill に割り当てます。subscription-list カラムがパブリケーションに含まれないとした場合、Marc のリモート・データベースでのオペレーションは次のようになります。



Customer ローの INSERT は、Message Agent によって統合データベースで実行されるため、INSERT を実行するとき、Adaptive Server Enterprise はトランザクション・ログにサブスクリプション値を記録します。

あとで Message Agent がログをスキャンすると、ログに保管されたサブスクリプション値を使用して新しいローに対するサブスクライバのリストが構築されます。この場合 Marc Dill はリストされません。SUBSCRIBE_BY_REMOTE オプションが OFF に設定されていると、新しい Customer ローは DELETE オペレーションとして Marc Dill に返されます。

SUBSCRIBE_BY_REMOTE オプションが ON に設定されているかぎり、Message Agent では、subscription-list カラムが NULL なので、ローはそれを挿入した営業担当者に所属するとみなされます。このため、INSERT は Marc Dill にレプリケートされず、レプリケーション・システムは影響を受けません。

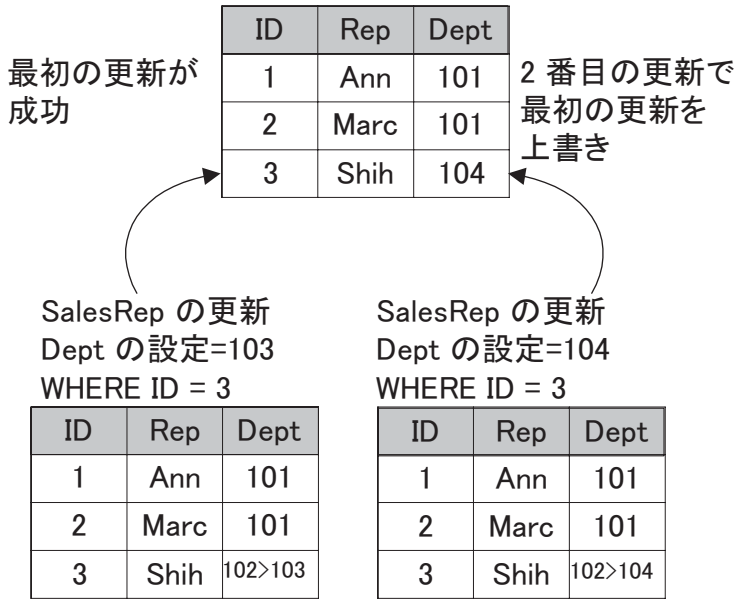
ユーザは、INSERT の後にトリガを実行して、subscription-list カラムを管理することができます。

競合の管理

UPDATE 競合は、次のような順序でイベントが実行されると発生します。

1. ユーザ 1 が、リモート・サイト 1 でローを更新します。
2. ユーザ 2 が、リモート・サイト 2 で同じローを更新します。
3. ユーザ 1 による更新内容が、統合データベースにレプリケートされます。
4. ユーザ 2 による更新内容が、統合データベースにレプリケートされます。

SQL Remote Message Agent が UPDATE 文をレプリケートするときは、ローごとに別の UPDATE 文をレプリケートします。またメッセージには、比較するために古いローの値が含まれています。ユーザ 2 による更新内容が統合データベースで受信される場合、ローの値はメッセージに記録されたものと異なります。



デフォルトによる競合解決

デフォルトでは、UPDATE の処理は継続されるため、ユーザ 2 による更新内容 (最終的に統合データベースに到達する) が統合データベースの値になります。また、この値が、ローにサブスクリプションを作成するその他のデータベースすべてにレプリケートされます。通常、デフォルトの競合解決方法では、最後に実行されたオペレーション (この場合はユーザ 2 によるオペレーション) を正常に処理して、競合の発生がレポートされません。ユーザ 1 による更新内容は失われます。

SQL Remote では、データの変更手段における競合を解決するストアド・プロシージャを使用して、競合解決の方法をカスタム設定できます。

競合はプライマリ・キーに適用されない

UPDATE 競合は、プライマリ・キーの更新には適用されません。変更されているカラムがプライマリ・キーである場合に、ユーザ 2 による更新内容が統合データベースで受信されると、ローは更新されません。

この項では、統合データベースにおける、SQL Remote のインストール環境への競合解決方法の構築について説明します。

SQL Remote の競合処理方法

競合が検出された場合

SQL Remote のレプリケーション・メッセージには、単一のロー更新のセットとして UPDATE 文が含まれています。それぞれの文には更新前の値が含まれています。

UPDATE 競合は、データベースのローに値を一致させるときのエラーとして、データベース・サーバによって検出されます。

統合データベースにおける競合だけが Message Agent によって検出されて解決されます。リモート・データベースからのメッセージで UPDATE 競合が検出されると、そのデータベースで次の 2 つのアクションが実行されます。

1. UPDATE が適用されます。
2. 競合解決のプロシージャが呼び出されます。

RESOLVE UPDATE プロシージャの有無にかかわらず、UPDATE 文は VERIFY 句の値が一致しなくても適用されます。

Adaptive Server Anywhere の統合データベースでは、競合解決の方法が異なります。詳細については、「SQL Remote の競合処理方法」152 ページを参照してください。

競合解決の実装

この項では、SQL Remote でカスタム設定した競合解決を実装するために必要な作業について説明します。

必須オブジェクト

競合を解決するテーブルごとに、解決を処理するための次の 3 つのデータベース・オブジェクトを作成します。

- **古い値のテーブル** 競合メッセージを受信したときに、テーブルに保管されていた値を保持します。
- **リモート値のテーブル** メッセージの指定どおり、競合する更新内容が適用されたときに、リモート・データベースのテーブルに保管されていた値を保持します。
- **ストアド・プロシージャ** 競合を解決するアクションを実行します。

これらのオブジェクトは、競合解決が実行される統合データベースだけに必要です。オブジェクトをパブリケーションに含めないでください。

オブジェクトの命名

sp_add_remote_table ストアド・プロシージャまたは **sp_modify_remote_table** ストアド・プロシージャを使用して、レプリケートするテーブルにマークを付けている場合は、オプションのパラメータを使用して競合解決オブジェクトの名前を指定します。

sp_add_remote_table ストアド・プロシージャと **sp_modify_remote_table** ストアド・プロシージャは 1 つの必須引数を取ります。この引数は、レプリケートするためにマーク付けされているテーブルの名前になります。また、さらに 3 つの引数を取りますが、それらは競合の解決に使用されるオブジェクトの名前になります。たとえば、**sp_add_remote_table** の構文は次のとおりです。

exec sp_add_remote_table *table_name*
[, *resolve_procedure*]
[, *old_row_table*]
[, *remote_row_table*]

3 つのオブジェクト *resolve_procedure*、*old_row_table*、*remote_row_table* をそれぞれ作成します。これらのオブジェクトについて順に説明します。

- **old_row_table**　このテーブルには、テーブル *table_name* と同じカラム名およびデータ型が必要ですが、外部キーは指定できません。競合が発生すると、更新される *table_name* のローの値がある *old_row_table* にローが挿入されてから、UPDATE が適用されます。*resolve_procedure* が実行されると、挿入されたローは削除されます。

  Message Agent は単一のロー更新のセットとして更新を適用するので、テーブルには単一のローだけが含まれます。

- **remote_row_table**　このテーブルには、テーブル *table_name* と同じカラム名およびデータ型が必要ですが、外部キーは指定できません。競合が発生すると、リモート・データベースからの *table_name* のローの値がある *remote_row_table* にローが挿入されてから、UPDATE が適用されます。*resolve_procedure* が実行されると、挿入されたローは削除されます。

  Message Agent は単一のロー更新のセットとして更新を適用するので、テーブルには単一のローだけが含まれます。

- **resolve_procedure**　このプロシージャでは、競合の解決に必要なアクションがすべて実行されます。アクションには、ローの値の変更や別のテーブルへの値のレポートなどが含まれます。

これらのオブジェクトを作成したら、**sp_add_remote_table** ストアド・プロシージャまたは **sp_modify_remote_table** ストアド・プロシージャを実行して、テーブルの競合解決オブジェクトとして通知します。

制限事項

- Adaptive Server Enterprise データベースでは、VERIFY_ALL_COLUMNS オプションが ON に設定されている間は、カラムが 129 以上あるテーブルでは競合解決は機能しませ

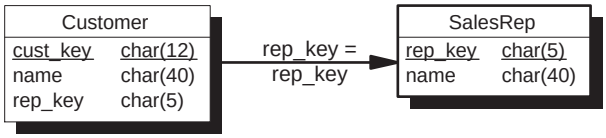
ん。VERIFY_ALL_COLUMNS オプションが OFF に設定されていても、UPDATE 文が 129 以上のカラムを更新する場合には、競合解決は機能しません。

競合解決の例 1

この例では、チュートリアルで使用している 2 テーブルからなるデータベースの内、Customer テーブルで発生した競合が、テーブルにレポートされ、後で競合について検証できます。

データベース

2 テーブルのデータベースは次のとおりです。



競合解決の目的

競合解決によって、Customer テーブルの **name** カラムに対する更新時の競合が、**ConflictLog** という名前の別のテーブルにレポートされます。

競合解決オブジェクト

競合解決テーブルは、次のように定義されます。

```
CREATE TABLE OldCustomer(
     cust_key CHAR( 12 ) NOT NULL,
     name CHAR( 40 ) NOT NULL,
     rep_key CHAR( 5 ) NOT NULL,
     PRIMARY KEY ( cust_key )
 )
 CREATE TABLE RemoteCustomer(
     cust_key CHAR( 12 ) NOT NULL,
     name CHAR( 40 ) NOT NULL,
     rep_key CHAR( 5 ) NOT NULL,
     PRIMARY KEY ( cust_key )
 )
```

各テーブルのカラムとデータ型は、Customer テーブル自体とまったく同じです。これらの定義で異なる点は、**SalesRep** テーブルに対する外部キーを持たないことだけです。

競合解決プロシージャによって、**ConflictLog** という名前のテーブルに競合がレポートされます。このテーブルの定義は次のとおりです。

```
CREATE TABLE ConflictLog (
     conflict_key numeric(5, 0) identity not null,
     lost_name char(40) not null ,
     won_name char(40) not null ,
     primary key ( conflict_key )
 )
```

競合解決プロシージャは、次のとおりです。

```
CREATE PROCEDURE ResolveCustomer
 AS
 BEGIN
     DECLARE @cust_key CHAR(12)
     DECLARE @lost_name CHAR(40)
     DECLARE @won_name CHAR(40)

 // Get the name that was lost
     // from OldCustomer
     SELECT @lost_name=name,
         @cust_key=cust_key
     FROM OldCustomer

 // Get the name that won
     // from Customer
     SELECT @won_name=name
     FROM Customer
     WHERE cust_key = @cust_key

 INSERT INTO ConflictLog ( lost_name, won_name )
     VALUES ( @lost_name, @won_name )
 END
```

この解決プロシージャでは、**RemoteCustomer** テーブルは使用されません。

競合解決の動作

ストアド・プロシージャは、競合解決にとって重要です。次のように機能します。

1. **OldCustomer** テーブルから **@lost_name** 値を取得し、さらにプライマリ・キー値を取得して、実際のテーブルにアクセスできるようにします。

@lost_name 値は、競合の原因となる UPDATE によって上書きされた値です。

2. Customer テーブル自体から **@won_name** 値を取得します。この値によって **@lost_name** が上書きされます。ストアド・プロシージャは更新*後*に実行されますが、これは **@won_name** 値が Customer テーブルにあるためです。この動作は SQL Remote for Adaptive Server Anywhere とは異なります。その場合、競合解決は BEFORE トリガに実装されます。

3. **@lost_name** 値と **@won_name** 値がある **ConflictLog** テーブルにローを追加します。

4. プロシージャが実行されると、**OldCustomer** テーブルと **RemoteCustomer** テーブルのローが Message Agent によって削除されます。この例は簡単なものなので、**RemoteCustomer** ローは使用されていません。

例のテスト

❖ 例をテストするには、次の手順に従います。

1 統合データベースにテーブルとプロシージャを作成し、競合解決オブジェクトとして Customer テーブルに追加します。

2 統合データベースで変更を挿入し、その変更内容をコミットします。次に例を示します。

```
UPDATE Customer
 SET name = 'Sea Sports'
 WHERE cust_key='cust1'
 go
 COMMIT
 go
```

3 リモート・データベースの同じ行に別の変更内容を挿入してコミットします。次に例を示します。

```
UPDATE Customer
 SET name = 'C Sports'
 WHERE cust_key='cust1'
 go
 COMMIT
 go
```

4　変更内容をリモート・データベースから統合データベースにレプリケートします。まずリモート・データベースで Message Agent を実行してメッセージを送信し、次に統合データベースで Message Agent を実行してそのメッセージを受信し適用して実行します。

5　統合データベースで **Customer** テーブルと **ConflictLog** テーブルを表示します。Customer テーブルには、リモート・データベースからの値が次のように入っています。

cust_key	name	rep_key
cust1	C Sports	rep1

ConflictLog テーブルには、競合を示すローが 1 つあります。

conflict_key	lost_name	won_name
1	Sea Sports	C Sports

競合解決の例 2

この例では、競合を解決するより複雑な例について説明します。この例は、「競合解決の例 1」214 ページで説明した、前述の例と同じ状況に基づいています。

競合解決の目的

この場合の競合解決の目的は、次のとおりです。

- リモート・データベースから更新できないようにする。前述の例では、更新は可能でした。
- 更新に失敗したリモート・ユーザの名前を、消失した名前と取得した名前とともにレポートする。

競合解決オブジェクト

この場合、**ConflictLog** テーブルに追加のカラムがあり、リモート・ユーザのユーザ ID が記録されます。テーブルは次のとおりです。

```
CREATE TABLE ConflictLog (
     conflict_key numeric(5, 0) identity not null,
     lost_name char(40) not null ,
     won_name char(40) not null ,
     remote_user char(40) not null ,
     primary key ( conflict_key )
 )
```

ストアド・プロシージャはより複雑になります。更新は前述の例とは異なり不可能なので、**lost_name** 値はメッセージに送信される値を参照します。その値は、始めは適用されますが、後で競合解決プロシージャによって前に存在した値に置き換わります。

ストアド・プロシージャは、テンポラリ・テーブル **#remote** からの値を使用します。テンポラリ・テーブルを参照するプロシージャを作成するには、まずそのテンポラリ・テーブルを作成する必要があります。そのための文は次のとおりです。

```
CREATE TABLE #remote (
     current_remote_user varchar(128),
     current_publisher varchar(128)
 )
```

このテーブルは TEMPDB に作成され、現在のセッション中だけ存在します。Message Agent は、接続時に固有の #remote テーブルを作成し、プロシージャの実行時にそのテーブルを使用します。

```
CREATE PROCEDURE ResolveCustomer
 AS
 BEGIN
     DECLARE @cust_key CHAR(12)
     DECLARE @lost_name CHAR(40)
     DECLARE @won_name CHAR(40)
     DECLARE @remote_user varchar(128)

  -- Get the name that was present before
     -- the message was applied, from OldCustomer
     -- This will "win" in the end
     SELECT @won_name=name,
         @cust_key=cust_key
     FROM OldCustomer
```

```
-- Get the name that was applied by the
    -- Message Agent from Customer. This will
    -- "lose" in the end
    SELECT @lost_name=name
    FROM Customer
    WHERE cust_key = @cust_key

-- Get the remote user value from #remote
    SELECT @remote_user = current_remote_user
    FROM #remote

-- Report the problem
    INSERT INTO ConflictLog ( lost_name,
        won_name, remote_user )
    VALUES ( @lost_name, @won_name, @remote_user )

-- Disallow the update from the Message Agent
    -- by resetting the row in the Customer table
    UPDATE Customer
    SET name = @won_name
    WHERE cust_key = @cust_key
END
```

注意

ここで注意しなければならないことは、次のとおりです。

- リモート・ユーザのユーザ ID は、Message Agent によってテンポラリ・テーブル **#remote** の **current_remote_user** カラムに保管されます。
- Message Agent からの UPDATE は、プロシージャの実行前に適用されるので、プロシージャでは値を明示的に置き換える必要があります。これは SQL Remote for Adaptive Server Anywhere の場合とは異なります。その場合、競合解決は BEFORE トリガによって実行されます。

例のテスト

❖ **例をテストするには、次の手順に従います。**

1. 統合データベースにテーブルとプロシージャを作成し、競合解決オブジェクトとして Customer テーブルに追加します。
2. 統合データベースで変更を挿入し、その変更内容をコミットします。次に例を示します。

```
UPDATE Customer
 SET name = 'Consolidated Sports'
 WHERE cust_key='cust1'
 go
 COMMIT
 go
```

3 リモート・データベースの同じ行に別の変更内容を挿入してコミットします。次に例を示します。

```
UPDATE Customer
 SET name = 'Field Sports'
 WHERE cust_key='cust1'
 go
 COMMIT
 go
```

4 変更内容をリモート・データベースから統合データベースにレプリケートします。まずリモート・データベースで Message Agent を実行してメッセージを送信し、次に統合データベースで Message Agent を実行してそのメッセージを受信し適用して実行します。

5 統合データベースで **Customer** テーブルと **ConflictLog** テーブルを表示します。Customer テーブルには、統合データベースからの値が次のように入っています。

cust_key	name	rep_key
cust1	Consolidated Sports	rep1

ConflictLog テーブルには、競合を示し、リモート・データベースで入力された値を記録するローが 1 つあります。

conflict_key	lost_name	won_name	remote_user
1	Field Sports	Consolidated Sports	field_user

6 リモート・データベースで Message Agent を再度実行します。これにより、訂正された更新内容が統合データベースから受信されるので、顧客の名前にはここでも Consolidated Sports が設定されます。

参照整合性エラーを回避する設計

リレーショナル・データベースのテーブルは、外部キーの参照で関連付けられています。これらの参照の結果として適用される参照整合性制約により、データベースの一貫性が保たれます。データベースの一部のみをレプリケートする場合は、レプリケート・データベースの参照整合性に問題が発生する場合があります。

参照整合性エラーによるレプリケーションの停止
リモート・データベースが、参照整合性制約エラーによって実行できない文を含むメッセージを受信すると、以降のメッセージをデータベースに適用できなくなります (これは、適用されていないメッセージが残っているためです)。これには、メッセージ・キューにあるパススルー文も含まれます。

パブリケーションの設計中に参照整合性の問題に注意すれば、これらの問題を回避できます。この項では、よくある整合性の問題と、それを回避する方法について説明します。

参照テーブルがレプリケートされないエラー

次の **SalesRepData** パブリケーションを考えてみます。

```
exec sp_add_remote_table 'SalesRep'
 exec sp_create_publication 'SalesRepData'
 exec sp_add_article 'SalesRepData', 'SalesRep'
 go
```

SalesRep テーブルに、パブリケーションに入らなかった別のテーブル (たとえば **Employee**) に対する外部キーがある場合は、リモート・データベースから外部キー参照が削除されていないかぎり **SalesRep** テーブルへの挿入または更新を実行するとレプリケートに失敗します。

抽出ユーティリティを使用してリモート・データベースを作成する場合、外部キー参照はリモート・データベースから自動的に取り除かれ、この問題は発生しません。ただし、データベースには、**SalesRep** テーブルの **rep_id** カラムに無効な値が挿入されるのを回避する制約はありません。また、このような場合は、統合データベースで INSERT の実行に失敗します。この問題を回避するには、パブリケーションに **Employee** テーブル (または少なくともプライマリ・キー) を含めます。

ユニークなプライマリ・キーの確保

物理的にまったく異なるサイトのユーザは、各自がテーブルに新しいローを挿入できるので、プライマリ・キー値をユニークに保つ場合に問題になります。

2 人のユーザが同じプライマリ・キー値を使用してローを挿入する場合、レプリケーション・システムにある指定されたデータベースに着信する 2 番目の INSERT は失敗します。SQL Remote は不定期に接続するユーザのためのレプリケーション・システムなので、インストール環境内のすべてのデータベースに対するロック・メカニズムはありません。したがって、プライマリ・キーのエラーが発生しないように SQL Remote のインストール環境を設計する必要があります。

SQL Remote のインストール環境外でプライマリ・キー・エラーが発生しないようにするには、複数のサイトで修正されるテーブルのプライマリ・キーが、ユニークであることを保証する必要があります。この目的を達成する方法はいくつかあります。この章では、インストール環境のサイトごとにプライマリ・キー値のプールを使用する、一般的で効率的な、信頼性の高い方法について説明します。

プライマリ・キー・プールの概要

「**プライマリ・キー・プール**」は、SQL Remote のインストール環境で、各データベースのプライマリ・キー値のセットを保持するテーブルです。リモート・ユーザは、各自のプライマリ・キー値のセットを受信します。リモート・ユーザは、新しいローをテーブルに挿入する場合、ストアド・プロシージャを使用してプールから有効なプライマリ・キーを選択します。プールは、使用できる値を補充するプロシージャを、統合データベースで定期的に実行することによって管理されます。

営業担当者とその顧客からなる簡単なデータベースの例を用いて、この方法について説明します。これらのテーブルは実際のデータベースで使用するものよりずっと単純なので、レプリケーションに関する重要な問題だけに注目できます。

プライマリ・キー・プール

プライマリ・キー・プールは別のテーブルに記録されます。次の CREATE TABLE 文は、プライマリ・キー・プールのテーブルを作成します。

```
CREATE TABLE KeyPool (
     table_name VARCHAR(40) NOT NULL,
     value INTEGER NOT NULL,
     location VARCHAR(6) NOT NULL,
     PRIMARY KEY (table_name, value),
 )
 go
```

このテーブルの各カラムの意味は、次のとおりです。

カラム	説明
table_name	プライマリ・キー・プールで管理するテーブルの名前を保持する。ここに示す簡単なデータベースの例では、統合データベースだけに営業担当者が新しく追加されると、Customer テーブルだけがプライマリ・キー・プールを必要とし、このカラムが重複する。これは、一般的な解決方法を示すために含まれている。
value	プライマリ・キー値のリストを保持する。それぞれの値は、**table_name** にリストされた各テーブルに対してユニークである。
location	セットアップによっては、この値が **SalesRep** テーブルの **rep_key** 値と同じになる場合がある。また、営業担当者以外のユーザがいて、この 2 つの識別子が別々でなければならないセットアップもある。

パフォーマンスを向上させるために、次のようにテーブルにインデックスを作成できます。

```
CREATE INDEX KeyPoolLocation
 ON KeyPool (table_name, location, value)
 go
```

プライマリ・キー・プールのレプリケーション

キー・プールは、既存のパブリケーションに組み込むか、または別のパブリケーションとして共有することができます。この例では、プライマリ・キー・プール用に別のパブリケーションを作成します。

❖ **プライマリ・キー・プールをレプリケートするには、次の手順に従います。**

1 プライマリ・キー・プールのデータ用に、パブリケーションを作成します。

```
sp_create_publication 'KeyPoolData'
 go
 sp_add_remote_table 'KeyPool'
 go
 sp_add_article 'KeyPoolData', 'KeyPool',
     NULL, 'location'
 go
```

2 各リモート・データベースのサブスクリプションを、**KeyPoolData** パブリケーションに作成します。

```
sp_subscription 'create',
     KeyPoolData,
     field_user,
     rep1
 go
```

サブスクリプションの引数は、location の識別子です。

状況によっては、既存のパブリケーションに **KeyPool** テーブルを追加し、同じ引数を使用して各パブリケーションにサブスクリプションを作成する方法は有効です。ここでは、location の値と **rep_key** の値を別にして、より一般的な解決方法について説明しています。

キー・プールの入力と補充

ユーザが新しい顧客を追加するたびに、使用できるプライマリ・キーのプールは 1 つずつ減少します。次のようなプロシージャを使用して、統合データベースでプライマリ・キー・プールのテーブルを定期的に補充する必要があります。

```
CREATE PROCEDURE ReplenishPool AS
 BEGIN
     DECLARE @CurrTable  VARCHAR(40)
     DECLARE @MaxValue   INTEGER
     DECLARE EachTable   CURSOR FOR
                 SELECT table_name, max(value)
```

```
                FROM KeyPool
                GROUP BY table_name
    DECLARE @CurrLoc    VARCHAR(6)
    DECLARE @NumValues  INTEGER
    DECLARE EachLoc        CURSOR FOR
                SELECT location, count(*)
                FROM KeyPool
                WHERE table_name = @CurrTable
                GROUP BY location

OPEN EachTable
    WHILE 1=1 BEGIN
        FETCH EachTable INTO @CurrTable, @MaxValue
        IF @@sqlstatus != 0 BREAK
        OPEN EachLoc
        WHILE 1=1 BEGIN
            FETCH EachLoc INTO @CurrLoc, @NumValues
            IF @@sqlstatus != 0 BREAK
            -- make sure there are 10 values
            WHILE @NumValues < 10 BEGIN
                SELECT @MaxValue = @MaxValue + 1
                SELECT @NumValues = @NumValues + 1
                INSERT INTO KeyPool
                          (table_name, location, value)
              VALUES (@CurrTable, @CurrLoc, @MaxValue)
            END
        END
        CLOSE EachLoc
    END
    CLOSE EachTable
END
go
```

このプロシージャは、各ユーザのプールに 10 個までの値を入力します。実際の運用環境では、より大きい値を使用することがあります。指定する値は、ユーザがデータベースのテーブルにローを挿入する頻度によって異なります。

KeyPool テーブルのプライマリ・キー値のプールを再補充するため、**ReplenishPool** プロシージャを統合データベースで定期的に実行する必要があります。

ReplenishPool プロシージャには、各サブスクライバに対して少なくとも 1 つ以上のプライマリ・キー値が必要です。これによって、最大値を検索し、値を追加して次のセットを生成できます。はじめにプー

ルを値で埋めるには、各ユーザに対して値を 1 つ挿入してから、**ReplenishPool** を呼び出して残りを埋めます。次の例では、3 人のリモート・ユーザと単一の統合ユーザ Office について示します。

```
INSERT INTO KeyPool VALUES( 'Customer', 40, 'rep1' )
 INSERT INTO KeyPool VALUES( 'Customer', 41, 'rep2' )
 INSERT INTO KeyPool VALUES( 'Customer', 42, 'rep3' )
 INSERT INTO KeyPool VALUES( 'Customer', 43, 'Office')
 EXEC ReplenishPool
 go
```

トリガを使用してキー・プールを補充できない
トリガを使用してキー・プールを補充することはできません。これは、元のオペレーションを実行するリモート・データベースには、トリガ・アクションなどのアクションがレプリケートされないためです。

新しい顧客の追加

営業担当者が Customer テーブルに新しく顧客を追加する場合、挿入するプライマリ・キー値は、ストアド・プロシージャを使用して取得します。次の例では、プライマリ・キー値を提供するストアド・プロシージャと、INSERT を実行するストアド・プロシージャについて説明します。

このプロシージャは、営業担当者の識別子がリモート・データベースの CURRENT PUBLISHER であることを利用します。

- **NewKey プロシージャ** **NewKey** プロシージャは、キー・プールにある整数値を提供し、プールからその値を削除します。

```
CREATE PROCEDURE NewKey
            @TableName VARCHAR(40),
            @Location VARCHAR(6),
            @Value INTEGER OUTPUT AS
     BEGIN
         DECLARE @NumValues INTEGER
         SELECT @NumValues = count(*),
                  @Value = min(value)
         FROM KeyPool
         WHERE table_name = @TableName
```

```
        AND location = @Location
        IF @NumValues > 1
            DELETE FROM KeyPool
            WHERE table_name = @TableName
            AND value = @Value
        ELSE
            -- Never take the last value,
            -- because RestorePool will not work.
            -- The key pool should be kept large
            -- enough so this never happens.
            SELECT @Value = NULL
    END
```

- **NewCustomer プロシージャ**　**NewCustomer** プロシージャは、プライマリ・キーを構成する **NewKey** が取得した値を使用して、テーブルに新しい顧客を追加します。

```
CREATE PROCEDURE NewCustomer @name VARCHAR(40),
            @loc VARCHAR(6) AS
 BEGIN
     DECLARE @cust  INTEGER
     DECLARE @cust_key  VARCHAR(12)
         EXEC NewKey 'Customer', @loc, @cust output
     SELECT @cust_key = 'cust' +
         convert( VARCHAR(12), @cust )
     INSERT INTO Customer (cust_key, name, rep_key
)
     VALUES ( @cust_key, @name, @loc )
 END
```

NewKey から取得される **@cust** 値をテストしてこの値が NULL でないことを確認し、値が NULL の場合には挿入しないように、プロシージャを強化できます。

キー・プールのテスト

❖ プライマリ・キー・プールをテストするには、次の手順に従います。

1　field_user ユーザ ID を使用して、リモート・データベースを再抽出します。

2　次のサンプルの INSERT を、リモート・サイトと統合サイトで実行してみます。

```
EXEC NewCustomer 'Great White North', rep1
```

プライマリ・キー・プールの概要

プライマリ・キー・プールを使用するには、次のコンポーネントが必要です。

- **キー・プール・テーブル**　インストール環境で、各データベースの有効なプライマリ・キー値を保持するテーブル。
- **補充プロシージャ**　値を補充したキー・プール・テーブルを維持するストアド・プロシージャ。
- **キー・プールの共有**　インストール環境の各データベースは、キー・プール・テーブルから取得した、そのデーターベース自体の持つ有効な値のセットに対してサブスクリプションを作成する必要があります。
- **データ入力プロシージャ**　次に有効なプライマリ・キー値をプールから選択し、キー・プールからその値を削除するストアド・プロシージャを使用して、新しいローを入力します。

サブスクリプションの作成

パブリケーションに対するサブスクリプションを作成するには、各サブスクライバに REMOTE パーミッションが付与されている必要があります。また、そのユーザ用にサブスクリプションを作成する必要があります。サブスクリプションの詳細については、パブリケーションがサブスクリプション・カラムを使用するかどうかによって異なります。

サブスクリプション・カラムを使用しないサブスクリプション

パブリケーションにサブスクリプション・カラムがない場合に、パブリケーションへユーザのサブスクリプションを作成するには、次の情報が必要です。

- **ユーザ ID**　このユーザのサブスクリプションがパブリケーションに対して作成されます。このユーザには、REMOTE パーミッションが付与されている必要があります。
- **パブリケーション名**　ユーザのサブスクリプションを作成するパブリケーション名。

次の文は、ユーザ ID **SamS** のサブスクリプションを **pub_orders_samuel_singer** パブリケーションに作成します。このパブリケーションは、サブスクリプション・カラムを使用しないで作成したものです。

```
sp_subscription 'create',
     'pub_orders_samuel_singer',
     'SamS'
```

サブスクリプション・カラムを使用するサブスクリプション

パブリケーションにサブスクリプション・カラムがある場合に、パブリケーションへユーザのサブスクリプションを作成するには、次の情報が必要です。

- **ユーザ ID**　このユーザのサブスクリプションがパブリケーションに対して作成されます。このユーザには、REMOTE パーミッションが付与されている必要があります。
- **パブリケーション名**　ユーザのサブスクリプションを作成するパブリケーション名。

- **サブスクリプション値** パブリケーションのサブスクリプション・カラムに対するテスト用の値です。たとえば、従業員 ID を含むカラム名をパブリケーションがサブスクリプション・カラムとして保持する場合、サブスクリプションを作成するユーザの従業員 ID の値が、サブスクリプションに提供されていなければなりません。サブスクリプション値は常に文字列です。

次の文は、Samuel Singer (ユーザ ID **SamS**、従業員 ID 856) のサブスクリプションを **pub_orders** パブリケーションに作成します。このパブリケーションは、サブスクリプション・カラム **sales_rep** で定義され、Samuel Singer 自身の営業実績に対してローを要求します。

```
sp_subscription create,
     pub_orders,
     SamS,
     '856'
```

サブスクリプションの開始

正しく更新を受信して適用するため、各サブスクライバにはデータの初期コピーが必要です。同期の処理については、「データベースの同期」238 ページで説明しています。

第 3 部　SQL Remote の管理

第 3 部では、SQL Remote の配備と管理について説明します。

第 9 章

データベースの配備と同期

この章の内容

この章では、SQL Remote レプリケーション・インストールを配備して同期させるために必要な作業を説明します。

配備の概要

SQL Remote システムの設計段階が完了したら、次にリモート・データベースとアプリケーションを作成して配備します。

配備作業

場合によっては、配備作業は主要な作業となります。たとえば、営業支援システムに多数のリモート・ユーザがいる場合、配備作業は次のようになります。

1. 各リモート・ユーザ用の Adaptive Server Anywhere データベースとそのデータの最初のコピーを構築します。
2. 各ユーザのマシンに、Adaptive Server Anywhere データベース・サーバ、SQL Remote Message Agent、クライアント・アプリケーションとともにデータベースをインストールします。
3. ユーザ名、Message Agent 接続文字列、パーミッションなどが正しく、システムが適切に設定されていることを確認します。

大規模な配備の場合、リモート・サイトには一般的に Adaptive Server Anywhere データベースが使用されます。この章では、このことを前提に説明を進めます。

この章のトピック

この章のトピックは、次のとおりです。

- **リモート・データベースの作成** 各リモート・サイト用のリモート・データベースを作成してから、SQL Remote システムを配備します。

  主に、リモートの Adaptive Server Anywhere データベースの作成について説明します。

- **データの同期** データベースを同期させるには、データの最初のコピーをリモート・データベースに設定します。

配備前のテスト

SQL Remote システムを配備する前に、特に多数のリモート・サイトがある場合には、そのシステムを十分にテストしてください。

設計および設定段階にある場合は、SQL Remote 設定のあらゆる部分を変更できます。パブリケーションやメッセージ・タイプを変更したり、トリガを作成して更新の競合を解決することが簡単にできます。

SQL Remote アプリケーションを配備した後は、状況が異なります。SQL Remote 設定は、単一の「**分散データベース**」として認識できます。データベースは多数のサイトに分散しており、緩やかな一貫性が保持されます。設定内のすべてのデータベースのデータは、常にまったく同じ状態にはなりません。しかし時間が経つと、すべてのデータ変更は完全なトランザクションとしてシステム全体にレプリケートされます。SQL Remote 設定の一貫性は、入念なパブリケーション設計と、更新の競合が発生したときにそれが解決されることによって成立します。

アップグレードと再同期

SQL Remote 設定を配備して実行した後に修正することは容易ではありません。SQL Remote インストールに対するアップグレードは、最初の配備と同じくらい注意して行う必要があります。Adaptive Server Enterprise や Adaptive Server Anywhere データベース・ソフトウェアのメンテナンス・リリースをアップグレードする場合も同様です。これらのソフトウェアのアップグレードについては、配備前に互換性をテストする必要があります。

システム内の 1 つのデータベースでデータベース・スキーマを変更すると、データベース・オブジェクトに互換性がないために障害が発生する恐れがあります。パススルー・モードによって、スキーマの変更を SQL Remote 設定の一部またはすべてのデータベースに送信することが可能です。しかし、パススルー・モードの使用には注意と計画が必要です。

分散データベースの緩やかな一貫性とは、更新が常に進行中であるという意味です。通常、すべてのデータベースに実行中の変更を停止して、データベース・スキーマの一部を変更し、再起動することはできません。

計画を入念に行わないと、データベース・スキーマの変更によってインストールでエラーが発生し、すべてのサブスクリプションを停止して再同期させなければならない場合があります。再同期させるには、各リモート・サーバにデータの新しいコピーをロードする必要があります。サブスクライバが多数ある場合は、この処理に時間がかかり、作業の中断やデータの消失が発生する恐れがあります。

実行中のシステムに行ってはならない変更

次に、配備済みで実行中の SQL Remote 設定に行ってはならない変更の例を示します。「**許容できる**」と記述されている変更は、通常は許容できる変更です。「**制限のある**」と記述されている変更は、行うべきでない変更です。

以下に条件が記述されている場合を除き、次の変更は行わないでください。

- 統合データベースのパブリッシャの変更。
- カラムの削除や NULL 値を許容しないようにカラムを変更することなど、テーブルに対して行う制限のある変更。カラムや NULL 値に関連する変更は、メッセージによって SQL Remote 設定全体に送信済みである可能性があります。変更は失敗します。
- パブリケーションの変更。パブリケーション定義は、ローカル・サイトとリモート・サイトの両方で管理する必要があります。古いパブリケーション定義に従って行われた変更は、すでにメッセージによって SQL Remote 設定全体に送信済みである可能性があります。

  新規テーブルや新規カラムの追加など許容できる変更を行うことは可能です。ただし、パススルーによって、新規テーブルや新規カラムがリモート・データベース内およびリモート・データベースのパブリケーション内に存在するようにしている場合にかぎります。
- サブスクリプションの削除。パススルーの削除機能を使用してリモート・サイトのデータを削除する場合のみ、これを行うことは可能です。
- Adaptive Server Anywhere データベースのアンロードと再ロード。

Adaptive Server Anywhere データベースがレプリケーションに関係している場合は、データベースを同期し直さなければ、そのデータベースのアンロードと再ロードはできません。レプリケーションはトランザクション・ログに基づいていますが、データベースをアンロードして再ロードすると、古いトランザクション・ログが使えなくなります。このため、レプリケーションが関係するときは、バックアップをきちんと実行することが特に重要です。

システムで処理が実行されていないで、トランザクション・ログが完全にスキャンされている場合のみ、Adaptive Server Enterprise データベースをアンロードして再ロードできます。ステーブル・キューの **sr_queue_state** テーブル内にある **page_id** ローと **row_id** ローをリセットしてください。

データベースの同期

同期とは何か

SQL Remote レプリケーションは、トランザクション・ログの情報を使用して実行されます。しかし、リモート・データベースと統合データベースという 2 つの状況があるので、SQL Remote は、パブリケーションの一部を成すリモート・データベースのテーブルにあるすべての既存のローを削除して、パブリケーションの全内容を統合データベースからリモート・サイトにコピーします。この、リモート・データベースと統合データベースを同じにする処理を、「**同期**」といいます。

同期の時期

同期は、次の場合に使用されます。

- 統合データベースでサブスクリプションが作成されると、リモート・データベースが統合データベースと同じ状態で開始できるように、同期が実行されます。
- リモート・データベースが破損したり、リモート・データベースと統合データベースの処理ステップが揃わなくなり、SQL パススルー・モードを使用して修復できない場合は、同期によってリモート・サイト・データベースと統合データベースの処理ステップが揃うようにします。

同期の方法

リモート・データベースの同期は、次の方法で実行できます。

- **データベース抽出ユーティリティの使用** このユーティリティによって、リモートの Adaptive Server Anywhere データベースのスキーマが作成され、リモート・データベースが同期されます。通常は、この方法をおすすめします。
- **手動による同期** PowerBuilder パイプラインや他のツールを使用してファイルからロードすることによって、リモート・データベースを手動で同期させます。
- **メッセージ・システムを介した同期** SYNCHRONIZE SUBSCRIPTION 文 (Adaptive Server Anywhere の場合) または **sp_subscription 'synchronize'** プロシージャ (Adaptive Server Enterprise の場合) を使用して、メッセージ・システムを介してリモート・データベースを同期させます。

警告
SYNCHRONIZE SUBSCRIPTION または **sp_subscription 'synchronize'** をリモート・データベースで実行しないでください。

複数のオペレーティング・システムとデータベース抽出

多くの場合、統合サーバのオペレーティング・システムとリモート・データベースのオペレーティング・システムは異なります。

Adaptive Server Anywhere データベースは、1 つのファイル／オペレーティング・システムから、別のファイル／オペレーティング・システムにコピーできます。これにより、データベースを最初に同期させる作業が柔軟に行えます。

例

たとえば、統合データベースが格納されている UNIX システム上で Adaptive Server Enterprise サーバを実行しているが、Windows などが使用されているラップトップ・コンピュータにリモート・データベースを配備したい場合があります。

この場合、データベースの抽出を行うプラットフォームとして、次のような選択肢があります。ただし、必要なソフトウェアがある場合にかぎります。

- UNIX で抽出ユーティリティを実行して、再ロード・スクリプトとデータ・ファイルを作成します。スクリプトとデータ・ファイルを Windows マシンにコピーします。Adaptive Server Anywhere データベースを作成して、スキーマとデータとともに Windows でロードします。
- UNIX で抽出ユーティリティを実行して、再ロード・スクリプトとデータ・ファイルを作成します。Adaptive Server Anywhere データベースを作成して、同じ UNIX プラットフォーム上でスキーマとデータとともにロードします。次に、データベース・ファイルを配備するために Windows マシンにコピーします。
- Windows で抽出ユーティリティを実行し、データベースの作成とその他の作業すべてを Windows で行います。

同期と抽出に関する注意事項

- 多数のサブスクリプションを抽出したり、サブスクリプションを大規模で使用頻度の高いテーブルに同期させたりすると、他のユーザがデータベースにアクセスする際の処理速度が低下します。データベースへのアクセスが少ないときにサブスクリプションを抽出することをおすすめします。SEND AT 句を使用してアクセスが少ない時刻を指定すると、この処理は自動的に実行されます。
- 同期は、サブスクリプション全体に適用されます。現時点では、1 つのテーブルを同期させる簡単な方法はありません。

subscription-list カラムを使用している Adaptive Server Enterprise ユーザ用のパフォーマンスに関するヒントは、「抽出パフォーマンスのチューニング」197 ページと「共有ローの抽出パフォーマンスのチューニング」207 ページを参照してください。

抽出ユーティリティの使用

抽出ユーティリティは、リモートの Adaptive Server Anywhere データベースを作成するのに役立ちます。Adaptive Server Enterprise データベースの作成には使用できません。

抽出ユーティリティの実行

抽出ユーティリティには次の方法でアクセスできます。

- Sybase Central からアクセスする方法 (統合データベースが Adaptive Server Anywhere の場合)。
- コマンド・ライン・ユーティリティとしてアクセスする方法。*dbxtract* ユーティリティ (Adaptive Server Anywhere の場合) または *ssxtract* ユーティリティ (Adaptive Server Enterprise の場合) です。

警告
抽出ユーティリティの実行中は、Message Agent を実行しないでください。予測できない結果を生じる可能性があります。

再ロード・ファイルからのデータベースの作成

コマンド・ライン・ユーティリティによって、名前を付けたサブスクライバ用のリモートの Adaptive Server Anywhere データベースを構築するのに適したデータベース・スキーマとデータをアンロードします。これにより、*reload.sql* というデフォルト名が付いた SQL コマンド・ファイルと、データ・ファイルのセットが生成されます。これらのファイルを使用して、リモートの Adaptive Server Anywhere データベースを作成できます。

reload.sql の編集が必要な場合があります。
データベース抽出ユーティリティはリモート・データベースの準備を支援するためのものですが、すべての場合においてブラック・ボックス・ソリューションとはなりません。リモート・データベースを作成するときは、必要に応じて *reload.sql* コマンド・ファイルを編集してください。

❖ 再ロード・ファイルからリモート・データベースを作成するには、次の手順に従います。

1 次のいずれかの方法で、Adaptive Server Anywhere データベースを作成します。

- Sybase Central の [データベースの作成] ウィザード ([ツール] メニューから [Adaptive Server Anywhere 9] － [データベースの作成] を選択)
- *dbinit* ユーティリティ

2 Interactive SQL ユーティリティからデータベースに接続して、*reload.sql* コマンド・ファイルを実行します。[SQL 文] ウィンドウ枠に次の文を入力して、*reload.sql* コマンド・ファイルを実行します。

```
read path¥reload.sql
```

path には、再ロード・コマンド・ファイルのパスを指定します。

Sybase Central から使用する場合、抽出ユーティリティでは、*dbxtract* と同様にデータベースをアンロードするタスクが実行され、さらに新規データベースの作成処理が実行されます。

抽出ユーティリティはメッセージ・システムを使用しません。再ロード・ファイル (*ssxtract*/*dbxtract* の場合) またはデータベース (Sybase Central の場合) が、現在のマシンからアクセス可能なディレクトリに作成されます。メッセージ・リンクを介して多数のサブスクリプションを同期させると、メッセージ・トラフィックの量が増えます。また、メッセージ・システムが信頼性に欠ける場合は、すべてのメッセージがリモート・サイトで正常に受信されるのに多少時間がかかることがあります。

データベースを抽出する前に

統合データベースで抽出ユーティリティを使用する前に、次の作業を完了してください。

- レプリケーションのメッセージ・タイプを作成する。

- データベースにパブリッシャ・ユーザ ID を追加する。
- データベースにリモート・ユーザを追加する。
- データベースにパブリケーションを追加する。
- リモート・ユーザのサブスクリプションを作成する。
- メッセージ・リンク・パラメータを指定する必要がある場合は、それを指定する。

上記の作業を実行する方法については、「Adaptive Server Anywhere ユーザのためのチュートリアル」33 ページの章を参照してください。メッセージ・リンク・パラメータの設定方法については、「メッセージ・タイプ制御パラメータの設定」274 ページを参照してください。

抽出ユーティリティを使用してリモート・データベースを作成すると、そのデータベースを使用するユーザには、統合データベースで所有しているパーミッションと同じパーミッションが与えられます。さらに、そのユーザが統合データベースのグループのメンバである場合は、そのグループ ID が、ユーザが統合データベースで所有しているパーミッションが存在するリモート・データベースに作成されます。

Sybase Central からの抽出ユーティリティの使用

この項では、現在の統合データベースからリモート・ユーザ用のデータベースを抽出する方法を説明します。この項は、Adaptive Server Anywhere 統合データベースだけに適用されます。

[データベースの抽出] ウィザードによって、マシンでは次のことが実行されます。

- リモート・データベースの作成
- 統合データベースからファイルへの、関連する構造やデータの抽出 (アンロード)
- 新規に作成されたリモート・データベースへのファイルのロード

❖ **リモート・ユーザ用のデータベースを抽出するには、次の手順に従います (Sybase Central)。**

1 左ウィンドウ枠で、Adaptive Server Anywhere プラグインを選択します。

2 右ウィンドウ枠で、[ユーティリティ] タブをクリックします。

3 右パネルの [データベースの抽出] をダブルクリックします。

4 ウィザードの指示に従います。

注意

- このウィザードには、[ツール] － [Adaptive Server Anywhere] － [データベースの抽出] の順にクリックしてもアクセスできます。
- 実行されていないデータベースをウィザードを使用して抽出する場合は、自分用の構造とデータのアンロードだけを行えます。リモート・データベースの作成と再ロードはできません。このため、必ず Sybase Central で接続している統合データベースから抽出することをおすすめします。
- 特定のデータベースまたは特定のリモート・ユーザ用に抽出ウィザードを起動することもできます。Sybase Central によって、ウィザードへの適切な入力が自動的に行われます。
- 抽出ウィザードは、必ず WITH SYNCHRONIZATION オプションを使用してリモート・サーバを抽出 (同期) します。このオプションを使用したくない場合は、代わりに *dbxtract* ユーティリティを使用してください。
- [データベースの抽出] ウィザードには、[所有者別にオブジェクトをフィルタ] ダイアログで選択したユーザのテーブルだけが表示されます。特定のデータベース・ユーザが所有するテーブルを表示するには、アンロードしているデータベースを右クリックし、ポップアップメニューで [所有者別にオブジェクトをフィルタ] を選択し、表示されたダイアログで目的のユーザを選択します。

関連項目

抽出ユーティリティのオプション (コマンド・ライン・オプションとして、または抽出ウィザードの選択肢として使用可能) については、「抽出ユーティリティのオプション」394 ページを参照してください。

効率的な抽出プロシージャの設計

リモート・データベースごとに抽出ユーティリティを実行して、多数のデータベースを作成する処理は、非常に非効率的です。この処理を、より効率的にすることができます。この項では、処理を効率的にする方法を説明します。

subscription-list カラムを使用している Adaptive Server Enterprise ユーザ用のパフォーマンスに関するヒントは、「抽出パフォーマンスのチューニング」197 ページと「共有ローの抽出パフォーマンスのチューニング」207 ページを参照してください。

大規模な抽出処理が非効率的になる原因には、次のものがあります。

- 抽出ユーティリティは、各ユーザ用のスキーマとデータを含め、一度にデータベースを 1 つだけ処理します。通常、多数のユーザが共通のスキーマを共有し、データだけが異なります。各ユーザについて抽出ユーティリティを実行する単調な方法では、大量の処理が不必要に繰り返されます。スキーマとデータを別々に抽出すると、この問題を解決できます。

- Sybase Central から抽出ユーティリティを実行すると、各ユーザ用の新しいデータベースが作成されます。サブスクライバがスキーマを共有している場合は、スキーマはあるがデータのないデータベースを 1 つ作成し、ファイルをコピーできます。

- デフォルトでは、抽出ユーティリティは独立性レベル 0 で実行されます。アクティブなサーバからデータベースを抽出する場合は、独立性レベル 3 (「抽出ユーティリティのオプション」394 ページを参照) で実行し、抽出されたデータベース内のデータがサーバ上のデータと一致するようにします。

  独立性レベル 3 で実行すると、多数のロックが必要になるため、サーバ上の他のユーザのターンアラウンド・タイムに影響が出ることがあります。サーバへのアクセスが少ないときに抽出ユーティリティを実行するか、データベースのコピーに対して抽出ユーティリティを実行することをおすすめします。

多数のデータベースを抽出する効率的な方法

上記の問題を回避する方法の 1 つは、次のとおりです。

1. 統合データベースのコピーを作成して、同時にライブ・データベースからサブスクリプションを開始します。サブスクライバへのメッセージの送信が開始されます。ただし、サブスクライバにはまだデータベースがなく、メッセージは受信されません。

   1 つのトランザクション内でいくつかのサブスクリプションを開始するには、REMOTE RESET 文 (Adaptive Server Anywhere の場合) または **sp_remote** プロシージャ (Adaptive Server Enterprise の場合) を使用します。

2. データベースのコピーからリモート・データベースを抽出します。データベースはコピーであるため、ロックと同時性の問題は発生しません。データベースが多数ある場合は、この処理は数日かかることがあります。

3. リモート・データベースが作成されると、その情報は古いものになります。しかし、ユーザはライブ統合データベースから送信されたメッセージを受信して適用し、そのリモート・データベースを最新の情報に更新できます。

この解決法によって運用データベースに影響が出るのは、最初の手順においてのみです。データベースが使用中で多数のロックを使用している場合は、コピーを独立性レベル 3 で作成する必要があります。また、コピー作成と同時にサブスクリプションを開始する必要があります。コピー作成とサブスクリプション開始の間に発生したオペレーションはすべて失われ、これが原因でリモート・データベースでのエラーが発生する恐れがあります。

グループの抽出

リモート・ユーザがグループ・ユーザ ID である場合、抽出ユーティリティはグループのメンバのユーザ ID すべてを抽出します。異なるユーザ ID を使用して、カスタム抽出処理なしで、この機能を各リモート・データベースの複数のユーザすべてに使用できます。

ユーザ用にデータベースが抽出されるときには、ユーザおよびそのユーザがメンバであるグループの、すべてのメッセージ・リンク・パラメータが抽出されます。

抽出ユーティリティ使用の制限

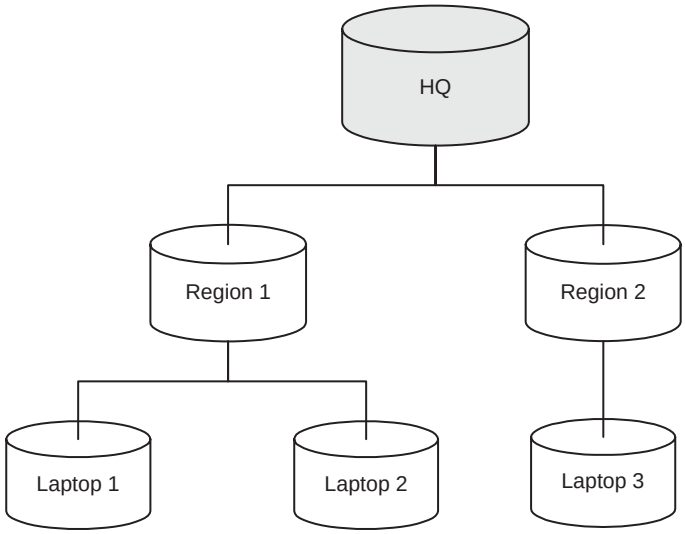
統合データベースからリモート・データベースを作成して同期させる適切な方法は、抽出ユーティリティを使用することですが、場合によっては抽出ユーティリティを使用できず、リモート・データベースを手動で同期させる必要があります。この項では、そのような場合をいくつか説明します。

- **Adaptive Server Enterprise リモート・データベースを作成できない**　抽出ユーティリティは、Adaptive Server Anywhere リモート・データベースのみに使用できます。
- **リモート・データベースの追加テーブル**　レプリケーションに関係しないテーブルであれば、統合データベースにないテーブルをリモート・データベースに追加できます。当然、抽出ユーティリティはこのようなテーブルを統合データベースから抽出できません。
- **Adaptive Server Enterprise と Adaptive Server Anywhere の機能の違い**　Adaptive Server Enterprise の一部の機能は、Adaptive Server Anywhere にはありません。抽出ユーティリティは同様の機能へのマッピングを実行しますが、マッピングは完全なものとはなりません。

  Adaptive Server Enterprise と Adaptive Server Anywhere に関する問題の詳細については、「Adaptive Server Enterprise の抽出ユーティリティの使用」248 ページを参照してください。
- **プロシージャとビューの抽出**　デフォルトでは、抽出ユーティリティはすべてのストアド・プロシージャとビューをデータベースから抽出します。ビューとプロシージャには、リモート・サイトで必要なものと必要でないものがあります。必要でないビューとプロシージャは、データベースの、リモート・サイトに含まれない部分のみを参照するものです。

  抽出ユーティリティを実行後に再ロード・スクリプトを実行して、不必要なビューとプロシージャを削除してください。
- **マルチ層の設定で抽出ユーティリティを使用する**　マルチ層配置での抽出ユーティリティの役割について理解するには、3 層の SQL Remote 設定を考慮してください。

次の図は、この設定を表わしたものです。



最上位レベルの統合データベースから抽出ユーティリティを使用して、第 2 レベルのデータベースを作成できます。次に、第 2 レベルのデータベースにリモート・ユーザを追加して、第 2 レベルのデータベースから抽出ユーティリティを使用してリモート・データベースを作成します。ただし、最上位レベルの統合データベースから第 2 レベルのデータベースを再抽出する必要がある場合は、作成したリモート・ユーザをそのサブスクリプションとパーミッションとともに削除して、ユーザを再構築する必要があります。例外はデータのみを再同期させる場合で、抽出ユーティリティを使用して、データベースのスキーマを置き換えないで、データを置き換えることができます。

Adaptive Server Enterprise の抽出ユーティリティの使用

Adaptive Server Enterprise の抽出ユーティリティは、Adaptive Server Enterprise のデータベース・スキーマを取得し、Adaptive Server Anywhere データベースを生成します。このツール特有の制限とテクニックがいくつかあります。

Adaptive Server Anywhere でサポートされていない Adaptive Server Enterprise の機能

Adaptive Server Enterprise の機能の一部には、Adaptive Server Anywhere でまったくサポートされていないものや、一部分だけがサポートされているものがあります。抽出ユーティリティでは、このような機能は部分的に扱われるか、まったく扱われません。

Adaptive Server Enterprise と Adaptive Server Anywhere の互換性の詳細については、『*Adaptive Server Anywhere ユーザーズ・ガイド*』の「*Transact-SQL の互換性*」を参照してください。

ssxtract でサポートされていない機能は、次のとおりです。

- **グループ化されたプロシージャ**　Adaptive Server Anywhere はプロシージャ・グループをサポートしません。プロシージャ・グループは *ssxtract* によって抽出されません。
- **名前を付けた制約とデフォルト**　Adaptive Server Anywhere は、名前を付けた制約とデフォルトをサポートしません。このようなオブジェクトは、1 つのオブジェクトに適用される制約とデフォルトとして直接抽出され、その名前は失われます。
- **ロール**　*ssxtract* は、Adaptive Server Anywhere のグループの概念を使用して、ロールを抽出します。名前の付いたロールのあるグループを作成し、それをユーザに割り当てます。
- **パスワード**　抽出中のデータベースのユーザに SYSLOGINS のエントリがない場合は、パスワードは抽出されません。ユーザがログイン ID を所有している場合は、ダミーのパスワードが抽出されます。
- **NCHAR と NVARCHAR**　これらのデータ型は、CHAR と VARCHAR として抽出され、NULL が許容されます。
- **timestamp カラム**　Adaptive Server Anywhere は timestamp カラムを提供しますが、これは Adaptive Server Enterprise のものとは異なるデータ型です。timestamp カラムは抽出されません。

システム・テーブルのカスタマイズ

Adaptive Server Anywhere データベースにロードされるオブジェクトは、システム・カタログに記述されます。Adaptive Server Enterprise の抽出ユーティリティは、まず Adaptive Server Anywhere システム・テーブルのセットを TEMPDB に作成して、Adaptive Server Enterpries カタログのデータをそれらのテーブルに格納します。次に、このテーブルのセットをアンロードして、Adaptive Server Anywhere データベースを構築する再ロード・スクリプトを提供します。

TEMPDB 内の Adaptive Server Anywhere システム・テーブルの内容を変更したい場合があります。SQL Remote では、この変更が可能です。

TEMPDB 内に Adaptive Server Anywhere システム・オブジェクトを作成して、それにデータを格納するストアド・プロシージャを、**sp_populate_sql_anywhere** といいます。最後のオペレーションとして、このプロシージャは **sp_user_extraction_hook** というプロシージャを呼び出します。デフォルトでは、このプロシージャはなにも処理を実行しません。抽出プロシージャをカスタマイズする場合は、適切な **sp_user_extraction_hook** プロシージャを記述します。

メッセージ・システムを介したデータの同期

サブスクリプションの作成

サブスクリプションは、最初の引数に **create** を指定した **sp_subscription** プロシージャを使用して、Adaptive Server Enterprise 統合データベースにて作成します。

サブスクリプションの作成によって、受信されるデータが定義されます。サブスクリプションの同期 (データの最初のコピーの提供) または開始 (メッセージの交換) は行われません。

サブスクリプションの同期

サブスクリプションの同期により、Message Agent はサブスクリプション内のすべてのローのコピーをサブスクライバに送信します。ただし、適切なデータベース・スキーマが設定されていることが必要です。Adaptive Server Anywhere 統合データベースでは、SYNCHRONIZE SUBSCRIPTION 文を使用して、サブスクリプションが同期されます。Adaptive Server Enterprise 統合データベースでは、最初の引数に synchronize を指定した **sp_subscription** プロシージャを使用して、サブスクリプションが同期されます。

同期メッセージがサブスクライバ・データベースで受信されると、Message Agent はデータベースの現在の内容を新しいコピーと置き換えます。サブスクリプションの一部を成し、統合データベースにレプリケートされなかったサブスクライバのデータは、失われます。同期が完了すると、Message Agent は START SUBSCRIPTION 文、または最初の引数に **start** を指定した **sp_subscription** プロシージャを使用して、サブスクリプションを開始します。

大量のメッセージの生成

メッセージ・システムを介してデータベースを同期させると、大量のメッセージが生成される可能性があります。多くの場合、メッセージ・システム内に大量のメッセージを保存せずに、抽出処理を使用してローカルにおいてデータベースを同期させることが望ましい方法です。

オペレーション中のサブスクリプションの同期

リモート・データベースと統合データベースの処理ステップが揃わなくなり、SQL Remote の SQL パススルー機能を使用して元に戻せない場合は、サブスクリプションを同期させると、リモート・データベー

スの内容を介してサブスクリプションのローが統合データベースからコピーされ、リモート・データベースと統合データベースの処理ステップが揃うようになります。

同期でのデータの消失

サブスクリプションの一部であっても、統合データベースにレプリケートされなかったリモート・データベースのデータは、サブスクリプションが同期されると消失してしまいます。データベースを同期させる前には、Sybase Central、または Adaptive Server Anywhere の場合は *dbunload* ユーティリティを使用して、リモート・データベースをアンロードまたはバックアップできます。

第 10 章

SQL Remote の管理

この章の内容

この章では、稼働中の SQL Remote のインストール環境を管理する上での一般的な問題、および原則について説明します。

個々のシステムの詳細な説明については、「SQL Remote for Adaptive Server Enterprise の管理」339 ページ と 「SQL Remote for Adaptive Server Anywhere の管理」311 ページ の各章を参照してください。

管理の概要

この章では、SQL Remote のインストール環境を管理する上で発生する問題点について解説します。

配備され稼働している SQL Remote の設定は、統合データベースで管理します。

- **パーミッション** SQL Remote のインストール環境にはさまざまな物理データベースが多数あるので、リモート・データベースと統合データベースのパーミッションを持っているユーザに対して一貫したスキームが必要になります。この章には、ユーザにパーミッションを割り当てるときに考慮すべき点を説明している項があります。

- **メッセージ・システムの設定** SQL Remote のインストール環境で使用される各メッセージ・システムでは、制御パラメータやその他の設定についてセット・アップします。この章ではこれらの設定について説明します。

- **Message Agent** Message Agent の役割は、メッセージの送信と受信です。Message Agent の動作と設定オプションの細かい点には、Adaptive Server Anywhere と Adaptive Server Enterprise とで異なるものがありますが、概念や方法についてはこれらと共通しているものもあります。共通の機能について、このマニュアルで解説します。

- **メッセージ・トラッキング** SQL Remote のインストール環境を管理することは、数多くのデータベース間でやりとりされる大量のメッセージを処理することを意味します。メッセージの内容、送信されるタイミング、適用方法などを理解するため、この章には SQL Remote のメッセージ・トラッキング・システムについて説明している項があります。

- **ログ管理** SQL Remote は、トランザクション・ログから送信するデータを取得します。したがって、SQL Remote のインストール環境をスムーズに稼働させるには、トランザクション・ログの適切な管理とバックアップが不可欠です。手順の詳細については稼働中のサーバによって異なりますが、一般的な問題はこの章で説明します。

- **パススルー・モード**　統合データベースからリモート・サイトに直接介入するための方法です。この方法については、この章で説明します。

SQL Remote パーミッションの管理

SQL Remote のレプリケーションに含まれるデータベースのユーザは、次のパーミッション・セットのいずれかで識別されます。

- **PUBLISH** データベース内にある 1 つのユーザ ID が、そのデータベースのパブリッシャとして識別されます。パブリケーションの更新と受信確認を含め、送信されるすべての SQL Remote メッセージは、パブリッシャ・ユーザ ID によって識別されます。SQL Remote 設定内のすべてのデータベースには、メッセージを送信するため、パブリッシャ・ユーザ ID が 1 つずつ必要です。
- **REMOTE** 現在のデータベースに対してメッセージを送受信するユーザのうち、SQL Remote の階層の中で現在のデータベースのすぐ下位に位置するユーザには、REMOTE パーミッションが付与されます。
- **CONSOLIDATE** 1 つのデータベース内に CONSOLIDATE パーミッションを付与されるユーザ ID は 1 つしかありません。CONSOLIDATE パーミッションは、SQL Remote 設定内の現在のデータベースの 1 段上位にあるデータベースを識別します。各データベースの 1 段上位には、統合データベースを 1 つしか配置できません。

これらのパーミッションについては SQL Remote のシステム・テーブルに定義されていて、他のデータベース・パーミッションからは独立しています。

PUBLISH パーミッションの付与と取り消し

データベースからメッセージが送信されると、そのデータベースを示すユーザ ID がメッセージに含まれるため、受信者側でメッセージの送信元を識別できます。このユーザ ID が、データベースの「**パブリッシャ・ユーザ ID**」です。1 つのデータベースに設定できるパブリッシャは 1 つだけです。Adaptive Server Anywhere データベースのパブリッシャとして割り当てられたユーザについては、Sybase Central で [ユーザとグループ] フォルダを開けばすぐにわかります。

レプリケーション・システム内にある読み込み専用のリモート・データベースにもパブリッシャは必要です。これは、レプリケーションのステータスについての情報を管理するため、これらのデータベースからも統合データベースに確認メッセージが送信されるためです。リモートの Adaptive Server Anywhere データベースの GRANT PUBLISH 文は、データベース抽出ユーティリティによって自動的に実行されます。

Sybase Central からの PUBLISH パーミッションの付与と取り消し

Sybase Central から、Adaptive Server Anywhere データベースの PUBLISH パーミッションを付与できます。それには、フル・システムまたはデータベース管理者のパーミッションを持つユーザとしてデータベースに接続する必要があります。

❖ 新規ユーザをパブリッシャとして作成するには、次の手順に従います (Sybase Central の場合)。

1. 左ウィンドウ枠で、[ユーザとグループ] フォルダを選択します。
2. [ファイル] メニューから [新規] ー [ユーザ] を選択します。

   [ユーザ作成] ウィザードが表示されます。
3. ウィザードの指示に従います。そのユーザがパスワードを持ち、Remote DBA 権限が付与されていることを確認します。この権限によって、このユーザ ID で Message Agent を起動できます。
4. [完了] をクリックすると、ユーザが作成されます。
5. [ユーザとグループ] フォルダで、作成したユーザを右クリックしてポップアップ・メニューで [パブリッシャに変更] を選択します。

❖ 既存のユーザをパブリッシャにするには、次の手順に従います (Sybase Central の場合)。

- [ユーザとグループ] フォルダで、ユーザを右クリックしてポップアップ・メニューで [パブリッシャに変更] を選択します。

Sybase Central から PUBLISH パーミッションを取り消すこともできます。

❖ PUBLISH パーミッションを取り消すには、次の手順に従います (Sybase Central の場合)。

- [ユーザとグループ] フォルダで PUBLISH パーミッションを付与されているユーザをダブルクリックし、ポップアップ・メニューで [パブリッシャの取り消し] を選択します。

PUBLISH パーミッションの付与と取り消し (Adaptive Server Anywhere の場合)

Adaptive Server Anywhere では、次の GRANT PUBLISH 文を使用して PUBLISH パーミッションを付与します。

```
GRANT PUBLISH TO userid ;
```

userid は、現在のデータベースで CONNECT パーミッションを持つユーザです。たとえば、次の文では **S_Beaulieu** というユーザに PUBLISH パーミッションを付与します。

```
GRANT PUBLISH TO S_Beaulieu
```

次のように REVOKE PUBLISH 文を使用して、現在のパブリッシャの PUBLISH パーミッションを取り消します。

```
REVOKE PUBLISH FROM userid
```

PUBLISH パーミッションの付与と取り消し (Adaptive Server Enterprise の場合)

Adaptive Server Enterprise では、次の **sp_publisher** プロシージャを使用して PUBLISH パーミッションを付与します。

```
sp_publisher userid
```

userid は、現在のデータベースで CONNECT パーミッションを持つユーザです。たとえば、次の文では **S_Beaulieu** というユーザに PUBLISH パーミッションを付与します。

```
exec sp_publisher 'S_Beaulieu'
 go
```

データベースがパブリッシャを持たないように設定するには、次のように引数を入れずに sp_publisher プロシージャを実行します。

```
exec sp_publisher
 go
```

PUBLISH パーミッションについての注意

- Sybase Central 以外で Adaptive Server Anywhere データベースのパブリッシャ・ユーザ ID を確認するには、特殊定数 CURRENT PUBLISHER を使用します。次の文で、「**パブリッシャ・ユーザ ID**」を取得します。

  ```
  SELECT CURRENT PUBLISHER
  ```

- Adaptive Server Enterprise データベースのパブリッシャ・ユーザ ID を確認するには、次の文を使用します。

  ```
  SELECT name
   FROM sysusers
   WHERE uid = ( SELECT user_id
                        FROM sr_publisher )
   go
  ```

- GROUP パーミッションを持つユーザ ID に PUBLISH パーミッションを付与しても、PUBLISH パーミッションはグループのメンバには継承されません。
- PUBLISH パーミッションには、送信メッセージのパブリッシャを識別する以外の権限はありません。
- 現在のデータベースから送信されたメッセージをユーザが受信して処理するには、パブリッシャ・ユーザ ID が受信するデータベースに対して REMOTE パーミッションまたは CONSOLIDATE パーミッションが必要です。
- データベースのパブリッシャ・ユーザ ID には、そのデータベースの REMOTE パーミッションまたは CONSOLIDATE パーミッションを付与できません。これにより、送信メッセージの送信者とメッセージの受信者を両方とも識別できます。
- リモート・データベースでパブリッシャのユーザ ID を変更すると、情報が失われるなど、そのデータベースを含むサブスクリプションのいずれかに深刻な問題が発生するおそれがあります。リモート・ユーザを最初から再同期するつもりでない限り、リモート・データベースのパブリッシャを絶対に変更しないでください。

- SQL Remote 設定の操作中に統合データベースのパブリッシャのユーザ ID を変更すると、データが失われるなどの深刻な問題が発生するおそれがあります。SQL Remote 設定を閉鎖してすべてのリモート・ユーザを再同期するつもりでないかぎり、統合データベースのパブリッシャのユーザ ID は絶対に変更しないでください。

REMOTE パーミッションと CONSOLIDATE パーミッションの付与と取り消し

REMOTE パーミッションと CONSOLIDATE パーミッションは非常に似ています。現在のデータベースからメッセージを受信するデータベースには、現在のデータベースに関連したユーザ ID が必要で、そのユーザ ID には REMOTE パーミッションまたは CONSOLIDATE パーミッションが付与されていなくてはなりません。パーミッションが付与されたユーザ ID は、そのデータベースが現在のデータベース内の受信データベースであることを示します。

SQL Remote の階層内で、現在のデータベースの 1 段下位にあるデータベースには REMOTE パーミッションが付与され、上位にある多くても 1 つのデータベースには、CONSOLIDATE パーミッションが付与されます。

REMOTE パーミッションと CONSOLIDATE パーミッションの設定

Adaptive Server Anywhere では、GRANT REMOTE 文と GRANT CONSOLIDATE 文を使ってメッセージ・システムとレプリケーション・メッセージの送信先アドレスを特定します。

Adaptive Server Enterprise では、**sp_grant_remote** プロシージャで REMOTE パーミッションを設定し、**sp_grant_consolidate** プロシージャで CONSOLIDATE パーミッションを設定します。

CONSOLIDATE パーミッションを読み込み専用リモート・データベースから統合データベースへも付与する必要があります。これは、リモート・データベースから統合データベースに受信確認メッセージが送信されるためです。リモートの Adaptive Server Anywhere データベースの GRANT CONSOLIDATE 文は、データベース抽出ユーティリティによって自動的に実行されます。

REMOTE パーミッションの付与

各リモート・データベースには、統合データベースにそれを表す単一のユーザ ID が必要です。ユーザ ID とアドレスがパブリケーションのサブスクライバとして識別されるために、ユーザ ID には REMOTE パーミッションを付与する必要があります。

REMOTE パーミッションを付与すると、次に挙げるいくつかのタスクが実行されます。

- ユーザ ID をリモート・ユーザとして識別する。
- このユーザ ID でメッセージを交換するときに使用するメッセージ・タイプを指定する。
- メッセージの送信先アドレスを指定する。
- メッセージがリモート・ユーザに送信される頻度を指示する。

REMOTE パーミッションの付与は、データベースへのリモート・ユーザの追加にも関連します。

Sybase Central の場合

Sybase Central を使用して、データベースにリモート・ユーザを追加することができます。リモート・ユーザとグループは、Sybase Central の [ユーザとグループ] フォルダと [SQL Remote ユーザ] フォルダの 2 ヵ所に表示されます。この項は、Adaptive Server Anywhere データベースだけに適用されます。

デフォルトでは、リモート・ユーザは作成時から remote DBA 権限を持っています。リモート・データベースにアクセスするための Message Agent でこの権限が必要なので取り消さないでください。

データベース内で最低 1 つのメッセージ・タイプが定義されるまで、新規リモート・ユーザを作成することはできません。

グループには REMOTE パーミッションを付与できますが、テーブル・パーミッションなどとは異なり、REMOTE パーミッションがグループ内のユーザに自動的に適用されることはあり*ません*。グループ内のユーザにパーミッションを適用するには、各ユーザに REMOTE パーミッションを明示的に付与する必要があります。この操作を行わないと、リモート・グループはリモート・ユーザとまったく同じような動作をし、リモート・ユーザとして分類されます。

❖ **新規ユーザをリモート・ユーザとしてデータベースに追加するには、次の手順に従います (Sybase Central の場合)。**

1 左ウィンドウ枠で、[SQL Remote ユーザ] フォルダを選択します。

2 [ファイル] メニューから [新規] ー [SQL Remote ユーザ] を選択します。

[新しいリモート・ユーザの作成] ウィザードが表示されます。

3 ウィザードの指示に従います。

❖ **既存のユーザをリモート・ユーザまたはリモート・グループにするには、次の手順に従います (Sybase Central の場合)。**

1 [ユーザとグループ] フォルダを開きます。

2 リモートに設定するユーザを右クリックして、ポップアップ・メニューで [ユーザをリモート・ユーザに変更] を選択します。

3 表示されるダイアログ・ボックスでリストからメッセージ・タイプを選択した後、アドレスを入力してメッセージを送信する頻度を選択し、[OK] をクリックします。すると、そのユーザがリモート・ユーザとして設定されます。

このユーザが、[ユーザとグループ] フォルダと [SQL Remote ユーザ] フォルダの両方に表示されます。

Adaptive Server Anywhere の場合

次の文では、**S_Beaulieu** というユーザに、以下のオプションを持つ REMOTE パーミッションを付与します。

- SMTP 電子メール・システムの使用
- 電子メール・アドレス **s_beaulieu@acme.com** へのメッセージの送信
- 毎日午後 10 時にメッセージを送信

```
GRANT REMOTE TO S_Beaulieu
 TYPE smtp
 ADDRESS 's_beaulieu@acme.com'
 SEND AT '22:00'
```

Adaptive Server Enterprise の場合

次の文では、**S_Beaulieu** というユーザに、以下のオプションを持つ REMOTE パーミッションを付与します。

- ファイル共有システムを使用してメッセージを交換する。
- アドレスのルート・ディレクトリの下にある **beaulieu** というディレクトリにメッセージを置く。

  アドレスのルート・ディレクトリ (Adaptive Server Anywhere および Adaptive Server Enterprise のいずれにも共通) は、SQLREMOTE 環境変数で指定されます。ただし、この変数が設定されている場合に限ります。環境変数が設定されていない場合は、メッセージ制御パラメータ FILE (レジストリまたは INI ファイルに入っている) のディレクトリ設定で指定されます。
- 12 時間ごとにメッセージを送信する。

```
exec sp_grant_remote 'S_Beaulieu',
     'file',
     'beaulieu',
     'SEND EVERY',
     '12:00'
 go
```

送信頻度の選択

メッセージを送信する頻度の設定には、選択肢が 3 つあります。それは次のとおりです。

- **SEND EVERY**　頻度を時、分、秒 (HH:MM:SS フォーマット) で指定できます。

  SEND EVERY を設定したユーザにメッセージを送信すると、同じ頻度が設定されているすべてのユーザにもメッセージが送信されます。たとえば、12 時間ごとに更新内容を受信するリモート・ユーザ全員に、時間をずらすことなく同時に更新内容が送信されます。これにより、Adaptive Server Anywhere のトランザ

クション・ログや Adaptive Server Enterprise のステーブル・キューを処理する回数を減らすことができます。あるユーザに固有な頻度を設定することは、できるだけ避けてください。

- **SEND AT** 1 日の特定の時刻 (時間と分)。

  毎日、指定した時刻に更新します。送信時刻をずらすよりも効率的に、できるだけ少ない回数で更新を行うことができます。また、データベースがビジーでない時刻を選択すると、他のユーザとの干渉を最小限にすることができます。

- **デフォルト設定 (SEND 句なし)** SEND AT 句または SEND EVERY 句が設定されていないユーザには、Message Agent が起動されるたびにメッセージが送信され停止します。Message Agent はバッチ・モードで起動します。

Sybase Central での送信頻度の設定

Sybase Central では、次の方法で送信頻度を設定できます。

- 既存のユーザまたはグループをリモートに設定するときに指定する。詳細については、「REMOTE パーミッションの付与」261 ページを参照してください。
- リモート・ユーザまたはグループのプロパティ・シートの [SQL Remote] タブで指定する。プロパティ・シートは、リモート・ユーザまたはグループを右クリックして、ポップアップ・メニューで [プロパティ] を選択すると開きます。

CONSOLIDATE パーミッションの付与

リモート・データベースでは、パブリッシュ・ユーザ ID とサブスクライブ・ユーザ ID が統合データベースの場合と逆になります。統合データベースのサブスクライバ (リモート・ユーザ) は、リモート・データベースではパブリッシャになります。統合データベースのパブリッシャは、リモート・データベースからのパブリケーションのサブスクライバになり、CONSOLIDATE パーミッションが付与されます。

各リモート・データベースでは、統合データベースに CONSOLIDATE パーミッションを付与しなくてはなりません。データベース抽出ユーティリティを起動してリモート・データベースを生成すると、GRANT CONSOLIDATE 文がリモート・データベースで自動的に実行されます。

Adaptive Server Anywhere の場合

Adaptive Server Anywhere では、VIM 電子メール・システムを使用して、次の文でユーザ ID **hq_user** に CONSOLIDATE パーミッションを付与します。

```
GRANT CONSOLIDATE TO hq_user
 TYPE vim
 ADDRESS 'hq_address'
```

この文には SEND 句がないためデフォルト設定が適用され、Message Agent が起動されるたびにメッセージが統合データベースに送信されます。

Adaptive Server Enterprise の場合

Adaptive Server Enterprise では、ファイル・メッセージ・リンクを使用して、次の文でユーザ ID **hq_user** に CONSOLIDATE パーミッションを付与します。

```
exec sp_grant_consolidate 'hq_user', 'file', address
 go
```

REMOTE パーミッションと CONSOLIDATE パーミッションの取り消し

ユーザの REMOTE パーミッションを取り消すと、SQL Remote のインストール環境からそのユーザを削除できます。ユーザまたはグループから REMOTE パーミッションを取り消すと、通常のユーザまたはグループに戻ります。また、それらのサブスクリプションもすべてのパブリケーションから自動的に削除できます。

Sybase Central からのパーミッションの取り消し

Sybase Central から、Adaptive Server Anywhere データベースの REMOTE パーミッションを取り消せます。

❖ **REMOTE パーミッションを取り消すには、次の手順に従います (Sybase Central の場合)。**

1. [ユーザとグループ] フォルダまたは [SQL Remote ユーザ] フォルダを開きます。
2. リモート・ユーザまたはグループを右クリックし、ポップアップ・メニューで [リモートの取り消し] を選択します。

Adaptive Server Anywhere でのパーミッションの取り消し

REVOKE 文を使用して、ユーザから REMOTE パーミッションと CONSOLIDATE パーミッションを取り消すことができます。ユーザ **S_Beaulieu** から REMOTE パーミッションを取り消す文は次のとおりです。

```
REVOKE REMOTE FROM S_Beaulieu
```

REMOTE アクセス権または CONSOLIDATE アクセス権を取り消すには、DBA 権限が必要です。

Adaptive Server Enterprise でのパーミッションの取り消し

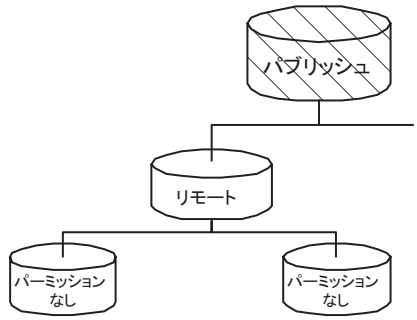
sp_revoke_remote プロシージャを使用して、ユーザから REMOTE パーミッションを取り消すことができます。このプロシージャに必要な引数は、ユーザのユーザ ID のみです。ユーザ **S_Beaulieu** から REMOTE パーミッションを取り消す文は次のとおりです。

```
exec sp_revoke_remote 'S_Beaulieu'
 go
```

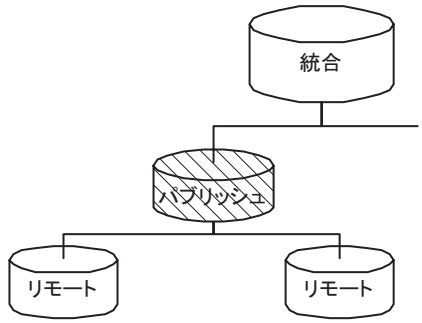
多層インストール環境でのパーミッションの割り当て

多層インストール環境でパーミッションを割り当てるには、特別に考慮すべきことがあります。3 層の SQL Remote の設定でのパーミッションについて、次の図にまとめました。各図のデータベースの 1 つにグレーの影を付けてあります。その他の各データベースには、そのグレーのデータベースで生成する必要がある、ユーザ ID に対するパーミッションを図示しています。「パーミッションなし」は、グレーのデータベースではそのデータベースに対し何のパーミッションも付与しないことを意味します。

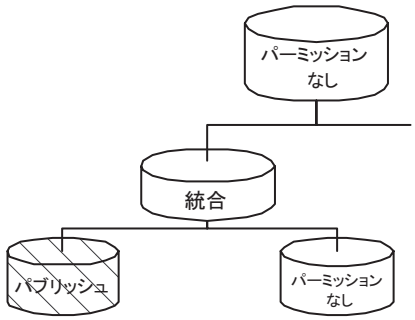
次の図は、3 層インストール環境の統合サイトで付与されている SQL Remote パーミッションを示しています。



次の図は、3 層インストール環境の内部サイトで付与されている SQL Remote パーミッションを示しています。



次の図は、3 層インストール環境の内部サイトで付与されている SQL Remote パーミッションを示しています。



リモート・データベースでの適切な PUBLISH パーミッションと CONSOLIDATE パーミッションの付与は、データベース抽出ユーティリティが自動的に行います。

メッセージ・タイプの使用

SQL Remote は、さまざまなメッセージ交換システムに対応しています。SQL Remote がサポートするメッセージ・システムは、次のとおりです。

- **file**　共有ファイル・システムのディレクトリにメッセージ・ファイルを格納して、他のデータベースからも読み込めるようにする。
- **ftp**　ファイル転送プロトコル (ftp) リンクでアクセス可能なディレクトリにメッセージ・ファイルを保管する。
- **mapi**　Microsoft Mail や他の電子メール・システムで使用する、Microsoft のメッセージング API (MAPI) リンク。
- **smtp**　インターネット電子メールで使用する、インターネット・メール転送プロトコル (SMTP/POP)。
- **vim**　Lotus Notes と cc:Mail で使用する、Lotus の Vendor Independent Messaging (VIM)。

データベースでは、使用可能なメッセージ・システムを 1 つまたは複数使用してメッセージを交換できます。

オペレーティング・システムによって使用可能なメッセージ・システム

SQL Remote が使用可能なすべてのオペレーティング・システムが、すべてのメッセージ・システムをサポートしているわけではありません。Windows オペレーティング・システムでは、リンクは DLL として実装されています。

各オペレーティング・システムが対応しているメッセージ・システムの一覧については、「サポートされるプラットフォームとメッセージ・リンク」553 ページを参照してください。

関連項目

- **file** メッセージ・システムの詳細については、「FILE メッセージ・システム」276 ページを参照してください。
- **ftp** メッセージ・システムの詳細については、「FTP メッセージ・システム」278 ページを参照してください。

- **smtp** メッセージ・システムの詳細については、「SMTP メッセージ・システム」280 ページを参照してください。
- **mapi** メッセージ・システムの詳細については、「MAPI メッセージ・システム」283 ページを参照してください。
- **vim** メッセージ・システムの詳細については、「VIM メッセージ・システム」285 ページを参照してください。

メッセージ・タイプの処理

各メッセージ・タイプの定義には、タイプ名 (**file**、**ftp**、**smtp**、**mapi**、**vim**) とそのメッセージ・タイプにおけるパブリッシャのアドレスも含まれます。リモート・データベースの作成時にデータベース抽出ユーティリティが、統合データベースでのパブリッシャ・アドレスを返信アドレスとして使用します。また Message Agent でも、**file** システムの受信メッセージを探す場所を識別するためにパブリッシャ・アドレスを使用します。

メッセージ・タイプの定義に入力されるアドレスには、データベースのパブリッシャ ID と密接なつながりがあります。有効なアドレスについては以降の項で説明します。

メッセージ・システムを使用する前に、必ずパブリッシャのアドレスを設定してください。

Sybase Central を使用したメッセージ・タイプの処理

Sybase Central では、メッセージ・タイプを作成したり変更したりできます。メッセージ・タイプは、[SQL Remote ユーザ] フォルダが選択されているときに、右ウィンドウ枠の [メッセージ・タイプ] タブに表示されます。この項は、Adaptive Server Anywhere データベースだけに適用されます。

メッセージ・タイプの作成および変更には、DBA 権限が必要です。

❖ メッセージ・タイプを追加するには、次の手順に従います (Sybase Central の場合)。

1. データベースに接続します。
2. 左ウィンドウ枠で、そのデータベースの [SQL Remote ユーザ] フォルダを開きます。
3. 右ウィンドウ枠の [メッセージ・タイプ] タブをクリックします。
4. [ファイル] メニューから [新規] ー [メッセージ・タイプ] を選択します。

   [新しい SQL Remote メッセージ・タイプの作成] ウィザードが表示されます。
5. [新しい SQL Remote メッセージ・タイプの作成] ウィザードで、メッセージ・タイプ名を入力します。入力する名前は、お使いの Adaptive Server Anywhere ディレクトリにすでにインストールされているメッセージ・タイプ DLL と一致させてください。[次へ] をクリックします。
6. パブリッシャ・アドレスを入力し、[完了] をクリックしてデータベースに定義を保存します。

パブリッシャのアドレスを変更するには、メッセージ・タイプを変更します。既存のメッセージ・タイプの名前を変更することはできないので、メッセージ・タイプを削除してから、新しい名前で新しいメッセージ・タイプを作成してください。

❖ メッセージ・タイプを変更するには、次の手順に従います (Sybase Central の場合)。

1. 左ウィンドウ枠で、データベースの [SQL Remote ユーザ] フォルダを開きます。
2. 右ウィンドウ枠の [メッセージ・タイプ] タブをクリックします。

3 右ウィンドウ枠で、変更するメッセージ・タイプを右クリックし、ポップアップ・メニューで [プロパティ] を選択します。

4 プロパティ・シートで、さまざまなオプションを設定します。

インストール環境からメッセージ・タイプを削除することもできます。

❖ **メッセージ・タイプを削除するには、次の手順に従います (Sybase Central の場合)。**

1 左ウィンドウ枠で、データベースの [SQL Remote ユーザ] フォルダを開きます。

2 右ウィンドウ枠の [メッセージ・タイプ] タブをクリックします。

3 右ウィンドウ枠で、変更するメッセージ・タイプを右クリックし、ポップアップ・メニューで [削除] を選択します。

Windows CE で使用するメッセージ・タイプの作成

Windows CE サービスがインストールされている場合は、[Sybase Central ユーティリティ] タブ (左ウィンドウ枠で [Adaptive Server Anywhere 9] を選択し、右ウィンドウ枠の [ユーティリティ] タブをクリック) で、SQL Remote を ActiveSync 同期用に設定するオプションを選択できます。これにより、FILE メッセージ・リンクのメッセージのフォルダが、ActiveSync フォルダとして設定されます。Windows CE マシンをデスクトップ・マシンにドッキングすると、ActiveSync は、デスクトップ・マシンの [ActiveSync] フォルダにあるファイルと、Windows CE の [ActiveSync] フォルダにあるファイルの同期をとります。

コマンドを使用したメッセージ・タイプの処理

❖ **メッセージ・タイプを作成するには、次の手順に従います (SQL の場合)。**

1 作成するメッセージ・タイプでのパブリッシャのアドレスが決定されていることを確認します。

2　CREATE REMOTE MESSAGE TYPE コマンドを実行します。

Adaptive Server Anywhere では、CREATE REMOTE MESSAGE TYPE 文は次の構文になっています。

CREATE REMOTE MESSAGE TYPE *type-name*
ADDRESS *address-string*

Adaptive Server Enterprise では、**sp_remote_type** プロシージャを使用します。このプロシージャには次のような引数が必要です。

sp_remote_type *type-name*, *address-string*

この文では、*type-name* は SQL Remote が対応しているメッセージ・システムのひとつで、*address-string* はそのメッセージ・システムでのパブリッシャのアドレスです。

パブリッシャのアドレスを変更するには、メッセージ・タイプを変更します。

❖ メッセージ・タイプを変更するには、次の手順に従います (SQL の場合)。

1　変更するメッセージ・タイプでのパブリッシャの新しいアドレスが決定されていることを確認します。

2　ALTER REMOTE MESSAGE TYPE 文を実行します。

Adaptive Server Anywhere では、ALTER REMOTE MESSAGE TYPE 文は次の構文になっています。

ALTER REMOTE MESSAGE TYPE *type-name*
ADDRESS *address-string*

Adaptive Server Enterprise では、メッセージ・タイプの作成と同じ方法で **sp_drop_remote_type** プロシージャを使用します。このプロシージャには次のような引数が必要です。

sp_remote_type *type-name*, *address-string*

この文では、*type-name* は SQL Remote が対応しているメッセージ・システムのひとつで、*address-string* はそのメッセージ・システムでのパブリッシャのアドレスです。

使用中のインストール環境ですでに使用されなくなったメッセージ・タイプを削除することもできます。メッセージ・タイプを削除すると、その定義からパブリッシャのアドレスが削除されます。

❖ メッセージ・タイプを削除するには、次の手順に従います (SQL の場合)。

- DROP REMOTE MESSAGE TYPE 文を実行します。

  Adaptive Server Anywhere では、DROP REMOTE MESSAGE TYPE 文は次の構文になっています。

  DROP REMOTE MESSAGE TYPE *type-name*

  Adaptive Server Enterprise では、メッセージ・タイプの作成と同じ方法で **sp_drop_remote_type** プロシージャを使用します。このプロシージャには次のような引数が必要です。

  sp_drop_remote_type *type-name*

  この文では、*type-name* は SQL Remote がサポートしているメッセージ・システムのひとつです。

参照

- 「CREATE REMOTE MESSAGE TYPE 文」454 ページ
- 「ALTER REMOTE MESSAGE TYPE 文」452 ページ
- 「DROP REMOTE MESSAGE TYPE 文」460 ページ

メッセージ・タイプ制御パラメータの設定

メッセージ・リンクごとにその動作を管理するパラメータがいくつかあります。パラメータはメッセージ・システムによって異なりますが、管理方法はすべて同じです。

特定のメッセージ・リンクで Message Agent を初めて使用する場合、リンクの動作を制御するパラメータを設定するダイアログ・ボックスが表示されます。これらのパラメータには、メッセージ・システムのユーザ ID、ftp メッセージが保管されるホスト名などがあります。入力したパラメータは、Message Agent によって保存されます。これらのパラメータを明示的に設定することもできます。

メッセージ・リンク・パラメータのデータベースへの保存

メッセージ制御パラメータは、データベース内に保存されます。オプションの設定方法は次のとおりです。

❖ **メッセージ制御パラメータを設定するには、次の手順に従います (Adaptive Server Anywhere の場合)。**

- 次の文を実行します。

```
SET REMOTE link-name OPTION
    [username.]option-name = option-value
```

❖ **メッセージ制御パラメータを設定するには、次の手順に従います (Adaptive Server Enterprise の場合)。**

- 次の文を実行します。

```
exec sp_link_option link-name, [username],
    option-name, option-value
```

sys.sysremoteoptions ビュー (Adaptive Server Anywhere の場合) または sr_remoteoptions ビュー (Adaptive Server Enterprise の場合) のクエリを実行すると、現在のメッセージ・リンク・パラメータが表示されます。

メッセージ・リンク・パラメータのディスクへの保存

このソフトウェアの古いバージョンでは、メッセージ・リンク・パラメータはデータベースの外に保存されていました。今でもこの方法を使用できますが、特別な理由があるとき以外は、データベース内にパラメータを保管することをおすすめします。

メッセージ・リンク制御パラメータは、次の場所に格納されます。

- **Windows**　レジストリ内の次の場所に格納されます。

```
¥¥HKEY_CURRENT_USER
    ¥Software
        ¥Sybase
            ¥SQL Remote
```

各メッセージ・リンクのパラメータは、メッセージ・リンクの名前 (4、*smtp* など) を付けられて、SQL Remote キーの下のキーに入ります。

- **NetWare** FILE システム・ディレクトリ設定を保存するために、*sys:¥system* ディレクトリにファイル *dbremote.ini* を作成してください。このファイルは Windows 形式の INI ファイルではないので、ディレクトリ名のみが入った単一行にします。

  たとえば、ディレクトリが *user:¥dbr43* の場合、*dbremote.ini* ファイルには次の行が入ります。

  ```
  user:¥dbr43
  ```

- **UNIX** FILE システム・ディレクトリ設定は、SQLREMOTE 環境変数に保存されます。

sqlremote 環境変数には、ファイル共有システムの制御パラメータのうち、その代替のひとつとして使用されるパスが保存されます。

各メッセージ・システムで使用できるパラメータについては、以降の項で説明します。各項ごとに 1 つのメッセージ・システムについて説明します。

Message Agent がメッセージ・リンクをロードすると、リンクは現在のパブリッシャの設定を使用します。設定が指定されていない場合は、そのパブリッシャが属するグループの設定を使用します。Windows では、メッセージ・リンク・パラメータのデータベースへの保存をサポートするバージョンの Message Agent を初めて起動すると、リンクのオプションがレジストリからデータベースにコピーされます。

FILE メッセージ・システム

file メッセージ・システムを使用すれば、メッセージ・システムが適切な場所になくても SQL Remote を使用できます。

FILE メッセージ・システムのアドレス

file メッセージ・システムは、単純なファイル共有システムです。リモート・ユーザの **file** アドレスは、すべてのメッセージが書き込まれるサブディレクトリです。「受信ボックス」からメッセージを取得するには、アプリケーションがユーザのファイルを格納しているディレクトリからメッセージを読み込みます。返信メッセージは、統合データベースのアドレスに送信 (ディレクトリに書き込み) されます。

NT サービスとして起動中の場合は、Message Agent が稼働しているアカウントに、必要なすべてのディレクトリに対する読み込みと書き込みのパーミッションがあることを確認してください。ネットワーク・ドライブにアクセスするときに、パーミッションの所持がよく問題になります。

アドレスのルート・ディレクトリ

file システムのアドレスは、一般的に共有ディレクトリのサブディレクトリです。このサブディレクトリは、モデムまたはローカル・エリア・ネットワークからアクセスする SQL Remote ユーザ全員が使用できます。各ユーザには、レジストリのエントリ、初期化ファイルのエントリ、または共有ディレクトリを示す SQLREMOTE 環境変数が必要です。

file システムを使用して、統合マシンまたはリモート・マシンのディレクトリにメッセージを入れることもできます。これにより、簡易ファイル転送メカニズムを使用してファイルを定期的に交換し、レプリケーションを実行できます。

FILE メッセージ制御パラメータ

FILE メッセージ・システムでは、次の制御パラメータを使用します。

- **Directory**　メッセージが保存されるディレクトリを設定します。この設定は、SQLREMOTE 環境変数の代替となります。
- **Debug**　これは YES か NO に設定されます (デフォルト設定では NO)。設定が YES の場合、FILE リンクによるすべてのファイル・システム呼び出しが表示されます。
- **Unlink_delay**　前回のファイル削除試行が失敗した場合、この秒数だけ待ってから次の削除を試行します。unlink_delay に対して値が定義されていない場合、デフォルト動作では、最初の試行失敗の後に 1 秒間、2 回目の試行失敗の後に 2 秒間、3 回目の試行失敗の後に 3 秒間、4 回目の試行失敗の後に 4 秒間一時停止します。

NetWare では、ディレクトリ設定を保存するために、*sys:¥system* ディレクトリにファイル *dbremote.ini* を作成してください。

FTP メッセージ・システム

FTP のアドレス

FTP メッセージ・システムでは、メッセージは FTP ホストのルート・ディレクトリの下位ディレクトリに保存されます。FTP ホストとルート・ディレクトリは、レジストリまたは初期化ファイルに保存されているメッセージ・システム制御パラメータによって指定されます。またメッセージが保存されるサブディレクトリが各ユーザのアドレスになります。

FTP をサポートしているオペレーティング・システムの一覧については、「サポートされるオペレーティング・システム」555 ページを参照してください。

FTP メッセージ制御パラメータ

FTP メッセージ・システムでは、次の制御パラメータを使用します。

- **host** メッセージが保存されるコンピュータのホスト名。ホスト名 (**ftp.ianywhere.com** など) または IP アドレス (192.138.151.66 など) を使用できます。
- **user** ftp ホストにアクセスするためのユーザ名。
- **password** ftp ホストにアクセスするためのパスワード。
- **root_directory** メッセージが保存される、ftp ホスト・サイトのルート・ディレクトリ。
- **port** 通常は必要ありません。これは、FTP 接続に使用する IP ポート番号です。
- **debug** YES または NO に設定されます。デフォルトでは NO です。YES を設定すると、デバッグ出力が表示されます。
- **active_mode** YES または NO に設定されます。デフォルトは NO (受動モード) です。

FTP 問題のトラブルシューティング

FTP メッセージ・リンクにおける問題のほとんどは、ネットワーク設定の問題です。この項では、問題に対処するためのテストについて説明します。

DEBUG メッセージ制御パラメータの設定　デバッグ出力を調査すると、FTP サーバに接続しているかどうかが分かります。接続している場合は、どの FTP コマンドに問題があるかが示されます。

FTP サーバの ping　FTP リンクが FTP サーバに接続できない場合は、システム・ネットワーク設定をテストしてみてください。システムに **ping** コマンドがある場合は、次のコマンドを入力してみます。

```
ping ftp-server-name
```

サーバの IP アドレスと、サーバへの ping (往復) 時間を示している出力を確認してください。ネットワーク設定に問題があるというよりサーバの ping が実行できない場合は、ネットワーク管理者に問い合わせてください。

受動モードの動作確認　FTP リンクが FTP サーバに接続しているのに、データ接続を開けない場合は、FTP クライアントが受動モードを使用してサーバからデータを転送できることを確認します。

受動モードは推奨の転送モードで、FTP メッセージ・リンクにデフォルトとして設定されています。受動モードでは、すべてのデータ転送接続はクライアント (この場合はメッセージ・リンク) から開始されます。アクティブ・モードでは、サーバがすべてのデータ接続を開始します。FTP サーバが正しく設定されていないファイアウォールの保護を受けていると、デフォルトの受動転送モードを使用できない場合があります。この場合、ファイアウォールは FTP 制御ポート以外のポート上の FTP サーバへのソケット接続をブロックします。

転送モードを **active** か **passive** に設定できる FTP ユーザ・プログラムを使用する場合は、転送モードを受動に設定して、ファイルのアップロードやダウンロードを行ってください。使用しているクライアントが、アクティブ・モードを使用しないとファイルを転送できない場合は、受動モードでの転送ができるようにファイアウォールと FTP サーバを設定し直すか、active_mode メッセージ制御パラメータに YES を設定してください。すべてのネットワーク設定でアクティブ・

モード転送が動作するとはかぎりません。次に例を示します。クライアントが IP マスカレードが機能しているゲートウェイの後方に位置している場合、ゲートウェイ・ソフトウェアによっては受信接続で障害が発生する可能性があります。

パーミッションとディレクトリ構造の確認 FTP サーバが接続中で、ディレクトリのリストの取得やファイルの操作に問題がある場合は、パーミッションが正しく設定されていることと、必要なディレクトリが存在していることを確認します。

FTP プログラムを使用して FTP サーバにログインします。ディレクトリを、root_directory パラメータに保存されている場所に変更します。必要なディレクトリが表示されない場合は、root_directory 制御パラメータが間違っているか、ディレクトリが存在しない可能性があります。

メッセージ・ディレクトリのファイルを取り出してパーミッションをテストし、統合データベース・ディレクトリにファイルをアップロードします。FTP サーバでエラーが発生する場合は、パーミッションが正しく設定されていません。

SMTP メッセージ・システム

メール転送プロトコル (SMTP) は、インターネット電子メール製品で使用されています。

SQL Remote では、SMTP システムを使用してインターネット・メールでメッセージを送信します。メッセージはテキスト形式にエンコードされ、電子メール・メッセージとしてターゲット・データベースに送信されます。メッセージは SMTP サーバを介して送信され、POP サーバから受信されます。これは、多くの電子メール・プログラムがメッセージの送受信に使用する方法です。

SMTP をサポートしているオペレーティング・システムの一覧については、「サポートされるオペレーティング・システム」555 ページを参照してください。

SMTP アドレスとユーザ ID

SQL Remote と SMTP メッセージ・システムを使用するには、設定に関係する各データベースで、SMTP アドレス、POP3 ユーザ ID、パスワードが必要です。これらは別々の識別子です。SMTP アドレスは各

メッセージの送信先で、POP3 ユーザ ID とパスワードは、ユーザが自分のメールボックスに接続するときに入力する名前とパスワードです。

個別の電子メール・アカウントを推奨
SQL Remote メッセージには、POP 電子メール・アカウントを個別に設定することをおすすめします。

トラブルシューティング

SMTP リンクが動作しない場合は、Message Agent が稼働している同じマシンから、同じアカウントとパスワードを使用して SMTP/POP3 サーバに接続してみます。SMTP/POP3 をサポートするインターネット電子メール・プログラムを使用して、SMTP メッセージ・リンクが動作することがわかったら、必ずそのプログラムを無効にしてください。

SMTP メッセージ制御パラメータ

Message Agent がメッセージ・システムに接続してメッセージを送受信する前に、ユーザは制御パラメータのセットを自分のコンピュータにあらかじめ設定しておくか、ウィンドウで必要な情報を入力しなくてはなりません。この情報が必要となるのは、初回接続時のみです。この情報は保存され、以後の接続でデフォルトのエントリとして使用されます。

SMTP メッセージ・システムでは、次の制御パラメータを使用します。

- **local_host**　ローカル・コンピュータの名前。SQL Remote がローカル・ホスト名を決定できないコンピュータで設定すると便利です。ローカル・ホスト名は、任意の SMTP サーバとのセッションを開始するのに必要です。ほとんどのネットワーク環境では、ローカル・ホスト名が自動的に決定されるため、このエントリは不要です。
- **TOP_supported**　受信メッセージを列挙するときに、SQL Remote は POP3 コマンド TOP を使用します。TOP コマンドは、すべての POP サーバでサポートされているわけではありません。このエントリを NO に設定すると、RETR コマンドを使用します。このコマンドは TOP よりも効率は落ちますが、すべての POP サーバで動作します。デフォルトは YES です。

- **smtp_authenticate** SMTP リンクがユーザを認証するかどうかを決定します。デフォルト値は YES です。SMTP 認証を実行しない場合は、NO に設定します。

- **smtp_userid** SMTP 認証のユーザ ID。デフォルトでは、このパラメータは **pop3_userid** パラメータと同じ値を取ります。**smtp_userid** は、ユーザ ID が POP サーバ上のユーザ ID と異なる場合にのみ設定する必要があります。

- **smtp_password** SMTP 認証のパスワード。デフォルトでは、このパラメータは **pop3_password** パラメータと同じ値を取ります。**smtp_password** は、ユーザ ID が POP サーバ上のユーザ ID と異なる場合にのみ設定する必要があります。

- **smtp_host** SMTP サーバが稼働しているコンピュータの名前。SMTP/POP3 ログイン・ダイアログの SMTP ホスト・フィールドに対応しています。

- **pop3_host** POP ホストが稼働しているコンピュータの名前。一般的には SMTP ホストと同じです。SMTP/POP3 ログイン・ダイアログの POP3 ホスト・フィールドに対応しています。

- **pop3_userid** メールの受信に使用します。POP ユーザ ID は、SMTP/POP3 ログイン・ダイアログのユーザ ID フィールドに対応しています。必ず POP ホスト管理者からユーザ ID を取得してください。

- **pop3_password** メールの受信に使用します。SMTP/POP3 ログイン・ダイアログのパスワード・フィールドに対応しています。これら 5 つのフィールドがすべて設定されると、ログイン・ダイアログは表示されなくなります。

- **Debug** YES に設定すると、SMTP と POP3 のすべてのコマンドと応答が表示されます。SMTP/POP のサポート問題のトラブルシューティングに便利です。デフォルトは NO です。

SMTP/POP アドレスの共有

各データベースには、SQL Remote メッセージ用の電子メール・アカウントが必要です。このアカウントは、個人的な電子メール・メッセージを読み込むものとは区別します。これは、多くの電子メール読者が次の方法で電子メールを取得するためです。

1. POP ホストに接続してすべてのメッセージをダウンロードする。
2. POP ホストからメッセージをすべて削除する。
3. POP ホストとの接続を切断する。
4. ローカル・ファイルまたはメモリからメールを読む。

これは、電子メール・プログラムがすべての SQL Remote の電子メール・メッセージを個人のメッセージと同様にダウンロードしたり削除したりするため、問題が発生します。お使いの電子メール・プログラムが未読メッセージを POP ホストから削除しないことが確実な場合は、データベースのメッセージを削除したり変更したりしないかぎり、電子メール・アドレスをデータベースと共有できます。

SQL Remote メッセージはランダムなテキストの行から構成されているように見えるため、簡単に識別できます。

MAPI メッセージ・システム

Message Application Programming Interface (MAPI) は、Microsoft Mail や最新の Lotus cc:Mail など、いくつかの有名な電子メール・システムで使用されています。

MAPI をサポートしているオペレーティング・システムの一覧については、「サポートされるオペレーティング・システム」555 ページを参照してください。

MAPI アドレスとユーザ ID

SQL Remote と MAPI メッセージ・システムを使用するには、その設定に関連する各データベースに MAPI ユーザ ID とアドレスが必要です。これらは別々の識別子です。MAPI アドレスは各メッセージの送信先で、MAPI ユーザ ID は、ユーザが自分のメールボックスに接続するために入力する名前とパスワードです。

MAPI メッセージと電子メール受信ボックス

SQL Remote のメッセージは読み込み用電子メールと同じメールボックスで受信しますが、一般的に電子メール受信ボックスには表示されません。

SQL Remote はアプリケーションで定義されたメッセージを送信しようとしますが、メールボックスを開くと、そうしたアプリケーション定義のメッセージを MAPI が識別して非表示にします。この方法では、ユーザは自分の個人的な電子メールとデータベースのアップデートを同じ電子メール・アドレスと同じ接続で受信することになりますが、SQL Remote のメッセージが読み込み用メールを干渉することはありません。

ただし、インターネット経由でメッセージをルーティングする場合、またはオペレーティング・システムに特定のパッチが適用されている場合は、特殊なメッセージ・タイプ情報が失われる可能性があります。そのようなメッセージは、受信者のメールボックスに表示されます。

MAPI メッセージ制御パラメータ

MAPI メッセージ・システムでは、次の制御パラメータを使用します。

- **Debug** YES に設定すると、すべての MAPI 呼び出しとリターン・コードが表示されます。MAPI のサポート問題のトラブルシューティングに便利です。デフォルトは NO です。
- **Force_Download** (デフォルトは YES) MapiLogon を呼び出したときに MAPI_FORCE_DOWNLOAD フラグが設定されているかどうかを制御します。このフラグが設定されているとダイヤルするリモート・メール・ソフトウェアを使用するときに便利です。
- **IPM_Receive** YES か NO に設定します (デフォルトは YES)。NO を設定すると、MAPI リンクは IPC メッセージを受信しますが、そのメッセージはメールボックスに表示されません。YES を設定すると、MAPI リンクは IPM メッセージと IPC メッセージを受信します。前者はメールボックスに表示されますが、後者は表示されません。IPM メッセージと IPC メッセージの両方が確実に SQL Remote によって受信されるように、この値はデフォルトである YES のままにしておくことをおすすめします。

- **IPM_Send**　YES か NO に設定します (デフォルトは NO)。YES を設定すると、MAPI リンクが IPM メッセージを送信し、メールボックスに表示します。NO を設定すると、MAPI リンクは IPC メッセージを送信しようとますが、そのメッセージはメールボックスに表示されません。インターネット経由でメッセージをルーティングする場合、またはオペレーティング・システムに特定のパッチが適用されている場合、SQL Remote は IPC メッセージを送信できないことがあります。
- **Profile**　指定の Microsoft Exchange プロファイルを使用します。Message Agent をサービスとして実行している間は、これは使用しないでください。

VIM メッセージ・システム

Vendor Independent Messaging システム (VIM) は、Lotus Notes と一部のバージョンの Lotus cc:Mail で使用されています。

SQL Remote と VIM メッセージ・システムを使用するには、その設定に関連する各データベースに VIM ユーザ ID とアドレスが必要です。これらは別々の識別子です。VIM アドレスは各メッセージの送信先で、VIM ユーザ ID は、ユーザが自分のメールボックスに接続するときに入力する名前とパスワードです。

VIM をサポートしているオペレーティング・システムの一覧については、「サポートされるオペレーティング・システム」555 ページを参照してください。

VIM メッセージ制御パラメータ

VIM メッセージ・システムでは、次の制御パラメータを使用します。

- **Path**　cc:Mail のログイン・ダイアログの [パス] フィールドに対応します。Lotus Notes では適用されないため、無視されます。
- **Userid**　cc:Mail のログイン・ダイアログの [ユーザ ID] フィールドに対応します。
- **Password**　cc:Mail のログイン・ダイアログの [パスワード] フィールドに対応します。Path、Userid、Password をすべて設定すると、ログイン・ダイアログが表示されなくなります。

- **Debug** YES を設定すると、すべての VIM 呼び出しとリターン・コードが表示されます。VIM のサポート問題のトラブルシューティングに便利です。デフォルトは NO です。

- **Receive_All** YES を設定すると、Message Agent がすべてのメッセージをチェックして、SQL Remote のメッセージであるかどうか確認します。NO (デフォルト) を設定すると、Message Agent はアプリケーション定義型 **SQLRemoteData** のメッセージだけを探します。このため、Notes のパフォーマンスが向上します。

  Receive_All に YES を設定すると、メッセージ・タイプが失われたとき、リセットされたとき、または設定されていないときに便利です。ここには cc:Mail メッセージ、またはインターネットを介したメッセージを含む設定が入っています。

- **Send_VIM_Mail** YES を設定すると、Adaptive Server Anywhere の 5.5.01 より古いバージョンと cc:Mail に対して互換性のあるメッセージが Message Agent から送信されます。YES を設定している場合は、**Receive_All** にも YES を設定していることを確認してください。

Message Agent の実行

SQL Remote Message Agent は、SQL Remote レプリケーションの主要コンポーネントです。Message Agent では、メッセージの送受信処理を行います。Message Agent の機能は次のとおりです。

- 受信メッセージを処理して、データベースに適切な順序で適用する。
- 各パブリッシャ・データベースのトランザクション・ログまたはステーブル・キューをスキャンして、ログのエントリをサブスクライバへのメッセージに変換する。
- ログのエントリを、固定最大サイズ (デフォルトで 50,000 バイト) 以下のメッセージに分散し、サブスクライバに送信する。
- システム・テーブルのメッセージ・トラッキング情報を保持し、保証済みの転送メカニズムを管理する。

実行可能な名前

Windows オペレーティング・システムでは、Adaptive Server Enterprise に対する Message Agent 名は *ssremote.exe* で、Adaptive Server Anywhere に対する Message Agent 名は *dbremote.exe* です。UNIX オペレーティング・システムでは、それぞれ名前が *ssremote* と *dbremote* に設定されています。

Adaptive Server Enterprise の Message Agent はステーブル・キューを使用して、必要がなくなるまでトランザクションを保持します。ステーブル・キューの詳細については、「Adaptive Server Enterprise の Message Agent の動作」340 ページを参照してください。

Message Agent のバッチ・モードおよび継続モード

Message Agent は次のいずれかのモードで動作します。

- **バッチ・モード**　バッチ・モードでは、Message Agent は起動し、送受信できるメッセージをすべて処理し、そして停止します。

バッチ・モードは、たまに接続するリモート・サイトにおいて有用です。接続したときにだけ、統合データベースとメッセージを交換できるためです。たとえば、リモート・サイトがメイン・ネットワークにダイヤルアップするようなときです。

- **継続モード** 継続モードでは、Message Agent は各リモート・ユーザのプロパティで指定された時刻に、定期的にメッセージを送信します。メッセージを送信していないときは、メッセージが到着すると受信します。

  継続モードは、メッセージが随時送受信される統合サイトにおいて有用です。負荷を分散させて迅速なレプリケーションを確実にするためです。

使用可能なオプションは、リモート・ユーザが選択した送信頻度オプションによって異なります。送信頻度オプションについては、「送信頻度の選択」263 ページを参照してください。

❖ 継続モードで Message Agent を実行するには、次の手順に従います。

1. すべてのユーザが送信頻度を指定していることを確認します。送信頻度は、GRANT REMOTE 文 (Adaptive Server Anywhere の場合) または **sp_grant_remote** プロシージャ (Adaptive Server Enterprise の場合) の SEND AT または SEND EVERY オプションで指定します。
2. -b オプションを使用しないで、Message Agent を起動します。

❖ バッチ・モードで Message Agent を実行するには、次の手順に従います。

- 次のいずれかを行います。
  - リモートのプロパティに SEND AT オプションも SEND EVERY オプションも設定していないリモート・ユーザを最低 1 人設定する。
  - -b オプションを使用して Message Agent を起動する。

Message Agent で使用される接続

Message Agent は、データベース・サーバへの接続をいくつか使用します。それらは次のとおりです。

- 汎用的な接続。Message Agent の起動中は常に接続されています。
- ログをスキャンするための接続。スキャンのフェーズ中のみ接続されています。
- ログスキャン・スレッドからコマンドを実行するための接続。スキャンのフェーズ中のみ接続されています。
- ステーブル・キューのための接続 (Adaptive Server Enterprise のみ)。スキャン中と送信フェーズ中のみ接続されています。
- 同期サブスクリプション要求を処理するための接続。送信フェーズ中のみ接続されています。
- 各ワーカ・スレッドのための接続。受信フェーズ中のみ接続されています。

レプリケーション・システムのリカバリの手順

SQL Remote のレプリケーションは、統合データベース・サイトにおいてデータ・リカバリの実行を新しい稼働条件としています。標準的なバックアップとリカバリ処理により、システムまたはメディアの障害からデータをリカバリできます。レプリケーションのインストールでは、リカバリ処理をしていても、リカバリされたデータベースがリモート・データベースと同期していないことがあります。そこでリモート・データベースの完全な再同期が必要になりますが、インストール環境に大量のデータベースが含まれている場合、再同期は膨大なタスクになる可能性があります。

つまり、統合サイトでの障害から統合データベースをリカバリしても、レプリケーションのインストール全体をリカバリするタスクの一部に過ぎないのです。

レプリケーション・システムをメディア障害から保護する方法は、2 つあります。

- **バックアップとログ管理** 統合データベース・サーバの連続バックアップ・プロシージャとログ管理プロシージャは、リカバリ・プランの中で不可欠な部分です。バックアップ・プロシージャの実行は、データベース・デバイス上のメディア障害から保護します。トランザクション・ログ・ミラーの使用は、トランザクション・ログ・デバイス上のメディア障害から保護します。

  バックアップとログの管理手順については、「トランザクション・ログとバックアップの管理」320 ページと 「Adaptive Server Enterprise のトランザクション・ログとバックアップの管理」351 ページの項を参照してください。

- **Message Agent 設定** Message Agent のコマンド・ライン・オプションによって、Message Agent の動作を各自のバックアップやリカバリ稼働条件に一致するようにチューニングできます。

  Message Agent の設定については、以降のページで説明します。

バックアップしたトランザクションのみレプリケート

デフォルトでは、Message Agent はコミットされたトランザクションをすべて処理します。Message Agent が -u オプションを使用して稼働しているときは、データベース・バックアップ・コマンドによってバックアップされたトランザクションのみが処理されます。

Adaptive Server Anywhere では、トランザクション・ログのバックアップを Sybase Central もしくは、*dbbackup* ユーティリティを使用して実行するか、ログ・ファイルのオフライン・コピーと名前の変更によって実行します。Adaptive Server Enterprise では、トランザクション・ログのバックアップを **dump transaction** 文を使用して実行します。

バックアップしたトランザクションのみを送信することで、トランザクション・ログ上のメディア障害からレプリケーションのインストールを保護できます。ミラーされたトランザクション・ログを管理することでも、レプリケーションのインストールを保護できます。

さらに、他のサイトでもバックアップを実行すれば、-u オプションによってサイト全体の障害を防ぐこともできます。

一貫した Message Agent 設定の確認

Message Agent 設定には、インストール全体を通して同じでなくてはならないものがあります。Message Agent を配備する前に、これらを必ず同じになるように設定してください。この項では、そのような設定について説明します。

- **メッセージの最大長**　SQL Remote ではメッセージの最大長に、デフォルト値 50 K が設定されています。この値は、Message Agent の -l オプションを使用して変更できます。ただし、1 つのインストール環境内にある各 Message Agent のメッセージの最大長は、すべて同じ値にしてください。またメッセージ長は、オペレーティング・システムのメモリ割り当て制限によって制限されることがあります。

  制限より長い受信メッセージは、矛盾したメッセージとして削除されます。

  この設定の詳細については、「Message Agent」376 ページを参照してください。

Message Agent とレプリケーション・セキュリティ

SQL Remote Message Agent で送信されるメッセージには、不意なスヌーピングから保護するために非常に簡単な暗号がついています。しかし、この暗号化スキームでは、強力な暗号解読方法からメッセージを完全に保護することはできません。

リモート・サイトでのトラブルシューティング・エラー

管理者が、統合サイトにのみアクセスして、リモート・サイトで発生したエラーのトラブルシューティングを行うには、明らかな障害があります。この問題に関する支援として、リモート・サイトの出力ログの一部を統合サイトに送信してファイルに書き込むように、SQL Remote を設定できます。この 1 つのファイルに、システムの一部またはすべてのサイトのログ情報が含まれます。

ログ情報を収集するように SQL Remote を設定するには、リモート・サイトと統合サイトの両方で設定します。

❖ 統合データベースにログ情報を送信するようにリモート・データベースを設定するには、次の手順に従います。

1 エラーが発生したときにログ情報を送信するようにリンクのオプションを設定します。

リモート・データベースに対して、次のコマンドを実行します。

```
SET REMOTE link-name OPTION
 PUBLIC.OUTPUT_LOG_SEND_ON_ERROR = 'YES'
```

このオプションを設定すると、エラーを示す "E" で始まるメッセージが発生した場合に SQL Remote が統合サイトにログ情報を送信します。

詳細については、『ASA SQL リファレンス・マニュアル』>「SET REMOTE OPTION 文 [SQL Remote]」を参照してください。

2 統合サイトに送信される情報の量を制限するようにリンクのオプションを設定します。この手順はオプションです。

リモート・データベースに対して、次のコマンドを実行します。

```
SET REMOTE link-name OPTION
 PUBLIC.OUTPUT_LOG_SEND_LIMIT = 'nnn'
```

このオプションの値は、統合データベースに送信されるバイト数で、出力ログの最後に表示されます (つまり、最後のエントリです)。*nnn*K としてキロバイトを指定できます。デフォルト設定値は `"5 K"` です。

指定した値が大きすぎて最大メッセージ・サイズに対して適切でない場合、SQL Remote はこのオプションの値を上書きし、適切な大きさのメッセージだけを送信します。

OUTPUT_LOG_SEND_NOW オプションを YES に設定すると、エラーがなくてもログ情報を送信できます。この場合、SQL Remote は次のポーリング時に出力ログ情報を送信し、ログの送信後にオプションが "NO" にリセットされます。

❖ **ログ情報を受信するように統合サイトを設定するには、次の手順に従います。**

- Message Agent の `-ro` または `-rt` オプションを使用します。

  詳細については、「Message Agent」376 ページを参照してください。

Message Agent のパフォーマンスのチューニング

この項の対象読者
自分のサイトのパフォーマンスに問題がない場合は、この項を読む必要はありません。

Message Agent のパフォーマンスをチューニングするためのオプションが、いくつかあります。この項では、それらのオプションについて説明します。

メッセージの送信と受信は、独立した 2 つの処理です。パフォーマンスの主要な問題は、それぞれの処理により異なります。

- **レプリケーションのスループット** SQL Remote サイトの全体のスループットにおける主なボトルネックは、一般的に、多くのリモート・サイトからメッセージを受信して、それを統合サイトのデータベースに適用することです。この作業は、統合サイトにおける Message Agent の受信処理をチューニングすることにより制御できます。
- **レプリケーションのターンアラウンド** あるサイトにデータが入力されてから他のサイトに表示されるまでのタイム・ラグが、レプリケーションのターンアラウンド・タイムです。このタイムラグは制御できます。

Message Agent スレッドの制御によるスループットのチューニング

この項では、統合サイトにおいて継続モードで稼働している Message Agent のパフォーマンスをチューニングしていると仮定します。

Message Agent では、ワーカ・スレッドを使用してリモート・ユーザからの受信メッセージを適用します。この場合、メッセージを直列でなく並列に適用できるようにするとスループットが向上します。

ワーカ・スレッド数の設定

ワーカ・スレッド数は、Message Agent のコマンド・ラインで、-w オプションを使用して設定します。次のコマンド・ラインでは、メッセージを適用する 20 のワーカ・スレッドを持つ Adaptive Server Enterprise の Message Agent を起動します。

```
ssremote -c "eng=..." -w 20
```

デフォルトではワーカ・スレッドを使用しない設定になっているので、すべてのメッセージは直列に適用されます。ワーカ・スレッドの最大数は 50 です。

ワーカ・スレッドからのパフォーマンス上の利点

Adaptive Server Anywhere の Message Agent では、分散されたドライブ・アレイがあるシステム上にサーバがあるときに、パフォーマンス上の利点が最も顕著に表れます。

Adaptive Server Enterprise では、複数のエンジンが設定されたサーバを使用すると、Message Agent のパフォーマンスがさらに向上します。

並列で適用されるメッセージ

ワーカ・スレッドを使用中のときは、さまざまなリモート・ユーザからのメッセージが並列で適用されます。リモート・ユーザ 1 人からのメッセージは直列で適用されます。たとえば、リモート・ユーザ 1 人から 10 件のメッセージが送信されると、そのメッセージは 1 つのワーカ・スレッドで適切な順序で適用されます。

ロールバックされたトランザクションを後で再適用することで、デッドロックが処理されます。

メッセージ・システムからのメッセージの読み込みは、1 つのスレッドになります。メッセージを読み込んで、ヘッダ情報を検索 (リモート・ユーザと正しい適用順を決定するため) してから、適用するワーカ・スレッドにメッセージを渡します。

メッセージの構築と送信は、1 つのスレッドです。

Open Client のバージョン

Adaptive Server Enterprise の Message Agent で複数のワーカ・スレッドを使用するには、Open Client バージョン 11.1 以降を使用してください。

11.1 より古いバージョンを使用すると、Message Agent ではメッセージの出力後にワーカ・スレッドを使用しません。Open Client のバージョンは、Message Agent 出力の最初の数行に表記されています。

メッセージのキャッシュによるスループットのチューニング

Message Agent は受信メッセージを読み込むと、メモリの設定可能エリアにそのメッセージをキャッシュします。

メッセージ・キャッシュ・サイズの指定

メッセージ・キャッシュのサイズは、-m オプションを使用して、Message Agent のコマンド・ラインで指定します。

-m オプションは、Message Agent がメッセージの構築に使用するメモリの最大容量を指定します。使用できるサイズは、*n* (バイト)、*n*K、*n*M で指定します。デフォルトは 2048 K (2 M) です。

例

次のコマンド・ラインは、12 メガバイトのメモリをメッセージ・キャッシュとして使用して Adaptive Server Anywhere の Message Agent を起動します。

```
dbremote -c "eng=..." -m 12M
```

メッセージをキャッシュする方法

トランザクションが非常に大きいか、メッセージが正しく到着しない場合、メッセージは適用されるまで Message Agent がメモリに保存します。このようにメッセージをキャッシュすると、規模の大きいインストール環境においてパフォーマンスを低下させるおそれがある、メッセージ・システムからの不正なメッセージの再読み込みを防止できます。これは、メッセージを WAN (リモート・アクセス・サービスやモデム経由の POP3 など) 上で読み込む場合、特に重要になります。また、メッセージの内容がキャッシュされるため、メッセージを読み込む (単一スレッド・タスク) ワーカ・スレッド間の競合を防ぐこともできます。

-m オプションで指定したメモリ使用量を超過すると、最低使用頻度 (LRU) 方式でメッセージがフラッシュされます。

このオプションは、数千のリモート・データベースに対して単一統合データベースを検討している顧客に主に提供されます。

受信メッセージのポーリングのチューニング

Message Agent を継続モードで実行している場合は、一般的に、受信メッセージをポーリングする頻度や不正なメッセージが到着してから再送を要求するまでの待機時間を統合データベース・サイトで制御できます。これらの動作をチューニングすることで、状況によってはパフォーマンスの向上に大きな成果をもたらします。

注意点

メッセージ受信処理をチューニングするときに注意する点は、メッセージ送信処理をチューニングするときの注意点と類似しています。

- **通常のメッセージ**　Message Agent がリモート・データベースから受信メッセージをポーリングする頻度を選択します。
- **再送要求**　不正なメッセージが到着してから再送を要求するまで待機するポーリング数を制御できます。
- **受信メッセージの処理**　メッセージの受信頻度と比較して受信メッセージのポーリング間隔が長すぎる場合、キューに入っているメッセージを終了して、処理されるまで待機できます。ポーリング間隔が短すぎる場合は、キューにメッセージがないときもポーリングするため、リソースが無駄になります。

メッセージ送信プロセスの詳細については、「メッセージ送信処理のチューニング」300 ページを参照してください。

ポーリング間隔

デフォルトでは、継続モードで実行中の Message Agent は前回のポーリングが終了した 1 分後にポーリングを行って、新着メッセージの有無を確認します。ポーリング間隔は、-rd オプションを使用して設定できます。

ポーリングが終了してから次回のポーリングが開始されるまでのポーリング間隔のデフォルトは、1 分です。次のコマンド・ラインのように、秒単位の値を使用してポーリング頻度を上げることもできます。

```
dbremote -rd 30s
```

逆に、ポーリングの頻度を下げることもできます。次のコマンド・ラインでは、ポーリング間隔を 5 分に設定します。

```
dbremote -rd 5
```

間隔を非常に短く設定すると、システムのスループット全体に悪影響を及ぼすことがあります。それは次のような理由によります。

- 電子メールを使用している場合は、メール・サーバをポーリングするごとにメッセージ・システムに負荷がかかります。あまり頻繁にポーリングを行うと、メッセージ・システムに悪影響を及ぼし、利点はまったくありません。
- 不正なメッセージが失われたとみなされるまでの間、Message Agent の待機時間を変更せずに再送を要求すると、その要求でシステムがいっぱいになる場合があります。

一般的には、メッセージへの素早い応答が要求される特別な場合でないかぎり、ポーリング間隔はあまり短くしないでください。

長めの間隔を設定すると、各メッセージを適用するまで多少長く待機するので、システムのメッセージ・スループット全体が向上します。SQL Remote の多くのインストール環境では、ターンアラウンド・タイムの最適化はあまり重要ではありません。

再送要求

Message Agent で受信メッセージをポーリング中に、あるメッセージがシーケンスから欠落している場合、Message Agent はそのメッセージの再送を即座には要求しません。Message Agent には、1 回のポーリングにデフォルトの「**待機時間**」が設定されています。

6 番であるはずの次のメッセージが 7 番であっても、Message Agent は次のポーリングまで何もしません。その後、そのユーザ宛の新しいメッセージが検出されなければ、再送要求が発行されます。

要求を送信するまでに Message Agent が待機するポーリングの数は、`-rp` オプションを使用して変更できます。このオプションは、ポーリング間隔を設定する `-rd` オプションとともに使用されることが多くあります。

たとえば、ポーリング間隔が非常に短く、メッセージの到着順を維持しないメッセージ・システムを使用している場合、一般的には、非同期メッセージが到着するのは 2 ～ 3 回のポーリングが完了してからになります。このような場合は、`-rp` の値を増やして、再送要求を送信

するまでの待機時間が長くなるように Message Agent を設定してください。これを設定しないと、不要な再送要求が大量に送信されてしまいます。

例

user1 と **user2** という 2 人のリモート・ユーザがいて、Message Agent のコマンド・ラインが次のようであると仮定します。

```
dbremote -rd 30s -rp 3
```

次の一連の操作によってメッセージは *userX.n* とマーク付けされ、**user1.5** が user1 からの 6 番目のメッセージになります。Message Agent では、メッセージは両方のユーザについて 1 番から開始するものと考えます。

0 秒後

1. Message Agent が user1.1 と user2.4 を読み込みます。
2. Message Agent が user1.1 を適用します。
3. 順序不整合のメッセージが user2 から到着したため、この時点の Message Agent の待機時間は user1 が N/A、user2 が 3 です。

30 秒後

1. Message Agent でのメッセージの読み込み：新着メッセージなし
2. Message Agent での適用：なし
3. 現時点の Message Agent の待機時間：user1：N/A、user2：2

60 秒後

1. Message Agent でのメッセージの読み込み：user1.3
2. Message Agent での適用：新着メッセージなし
3. Message Agent の待機時間：user1：3、user2：1

90 秒後

1. Message Agent でのメッセージの読み込み：user1.4
2. Message Agent での適用：なし
3. Message Agent の待機時間：user1：3、user2：0
4. Message Agent が、user2 に再送を要求します。

ユーザが新着メッセージを受信すると、それが予期していたメッセージでなくても Message Agent の待機時間はリセットされます。

メッセージ送信処理のチューニング

レプリケーションのターンアラウンド・タイムは、各サイトからメッセージが送信される頻度と、各サイトが受信メッセージをポーリングする頻度によって制御されます。データ入力とデータ・レプリケーションの間のタイム・ラグを小さくするには、Message Agent の `-sd` オプションの値を小さく設定します。これにより、それ以上のデータの送信の必要性を確認するポーリングの頻度が制御されます。

注意点

メッセージ送信処理をチューニングするときに注意すべきことは、受信メッセージのポーリング頻度をチューニングするときの注意点に類似しています。

- **通常のメッセージ**　リモート・データベースにアップデートを送信する頻度を選択します。
- **再送要求**　リモート・ユーザがメッセージの再送要求を発行すると、Message Agent では特別な処理を実行する必要があるため、通常のメッセージ送信を中断してしまう可能性があります。再送要求の緊急度は制御できます。
- **メッセージの数とサイズ**　非常に頻繁にメッセージを送信すると、小さいメッセージが送信されることが多くなります。メッセージの送信頻度を下げると、より多くの命令を 1 つのメッセージにグループ化できます。お使いのメッセージ・システムで小さいメッセージを大量に送信しなければならない場合は、ポーリング間隔をあまり短くしないでください。

受信メッセージのポーリングのチューニングについては、「受信メッセージのポーリングのチューニング」297 ページを参照してください。

ポーリング間隔

トランザクション・ログからの追加データを送信するポーリングの待機間隔は、-sd オプションを使用して制御します。デフォルトは 1 分です。次の例では、ポーリング間隔を 30 秒に設定します。

```
dbremote -sd 30s ...
```

逆に、ポーリングの頻度を下げることもできます。次のコマンド・ラインでは、ポーリング間隔を 5 分に設定します。

```
dbremote -sd 5
```

間隔を非常に短く設定すると、システムのスループット全体に悪影響を及ぼすことがあります。それは次のような理由によります。

- あまり頻繁にポーリングを行うと、短いメッセージがたくさん作成されます。メッセージのロードによりメッセージ・システムに負荷がかかると、スループットに悪影響が及びます。

長めの間隔を設定すると、各メッセージを適用するまで多少長く待機するので、システムのメッセージ・スループット全体が向上します。SQL Remote の多くのインストール環境では、ターンアラウンド・タイムの最適化はあまり重要ではありません。

メッセージの再送

ユーザがメッセージの再送を要求すると、メッセージがトランザクション・ログの最初のほうから取り出されます。このメッセージを取り出し、送信するためにトランザクション・ログに戻ると、Message Agent によって通常の送信処理が中断されます。最適なパフォーマンスを得るために SQL Remote のインストール環境をチューニングする場合は、メッセージの再送要求の緊急度と通常のメッセージ処理の優先度のバランスを取ってください。

-ru オプションでは、再送要求の緊急度を制御します。パラメータの値は分単位 (数字の最後に s や h を付ければ他の単位も可能) で設定します。デフォルトはゼロです。

通常のメッセージ送信処理を中断する前に、他のメッセージが到着するまで Message Agent の再送要求の処理を遅らせるには、このオプションの設定時間を長くします。

次のコマンド・ラインでは、再送要求を処理するまで 1 時間待機します。

```
dbremote -ru 1h ...
```

-ru オプションを指定しないと、Message Agent は、データの再送を要求したユーザの送信間隔に基づいてデフォルト値を選択します。ユーザの再送要求を受信してからログを再スキャンするまでの経過時間が、そのユーザの送信間隔の半分を超えることはありません。

メッセージのエンコードと圧縮

メッセージは電子メールや他のメッセージ・システムを介して渡されるため、メッセージが壊れる危険性があります。たとえば、一部のメッセージ・システムでは、特定の文字や文字の組み合わせを制御文字として使用しています。

メッセージのサイズは、メッセージがシステムを介して渡されるときの効率に影響を与えます。圧縮したメッセージは、圧縮していないメッセージよりも効率的にメッセージ・システムで処理されます。一方、圧縮の実行自体にもかなりの時間がかかります。

SQL Remote でのエンコードと圧縮

SQL Remote では、メッセージのエンコードと圧縮スキームが Message Agent に構築されています。このスキームの機能は次のとおりです。

- **互換性**　システムは、ソフトウェアの古いバージョンと互換性を持つように設定されています。
- **圧縮**　メッセージの圧縮レベルを選択できます。
- **エンコード**　SQL Remote ではメッセージをエンコードして、メッセージが破壊されずにメッセージ・システムを介して確実に渡されるようにします。エンコード・スキームをカスタマイズすると、特別な機能も使用できるようになります。

互換性の設定

ソフトウェアの古いバージョンと互換性を持たせるには、バージョン 6 のソフトウェアが動作しているそれぞれのデータベースで、データベース・オプション COMPRESSION を -1 (マイナス 1) に設定します。すると、古いバージョンと互換性のあるフォーマットでメッセージが送信されるようになります。

SQL Remote のアップグレード

統合データベースの Message Agent を最初にアップグレードするには、データベース・オプション COMPRESSION を -1 に設定します。レプリケーション・システムの各リモート・サイトをバージョン 6 にアップグレードすると、COMPRESSION オプションの設定を 0 (圧縮なし) ～ 9 (最大圧縮率) に変更できます。これにより、統合データベースに送信するメッセージに対して圧縮機能を利用できます。すべてのリモート・サイトをアップグレードしたら、統合データベースの Message Agent の COMPRESSION オプションに -1 以外の値を設定できます。

また、このオプションに -1 以外の値を設定すると、バージョン 6 で改良されたメッセージのエンコード機能も利用できます。

エンコード・スキーム

SQL Remote のメッセージのエンコード機能は、デフォルトで次のように動作します。

- バイナリ形式のメッセージが使用できるメッセージ・システムでは、エンコードは実行されません。
- SMTP、VIM、MAPI などの一部のメッセージ・システムでは、テキストベースのメッセージ形式を使用してください。このようなシステムでは、エンコード DLL (Adaptive Server Anywhere では *dbencod.dll*、Adaptive Server Enterprise では *ssencod.dll*) によって、メッセージがテキスト・フォーマットに変換されてから送信されます。このメッセージ形式は、同じ DLL を使用する受信側ではエンコードされません。
- カスタム・エンコード・スキームの使用を SQL Remote に指示できます。カスタム・エンコード・スキームを構築するためのツールは、次の項で解説します。
- データベース・オプション COMPRESSION に -1 を設定すると、すべてのメッセージ・システムでバージョン 5 と互換性のあるエンコードが実行されます。

カスタム・エンコード・スキームの作成

カスタム・エンコード DLL を構築すると、カスタム・エンコード・スキームを実装することができます。この DLL を使用して、特定のメッセージ・システムで必要な特殊機能を適用したり、各ユーザに送信されたメッセージの数やバイト数などの統計を取ったりすることができます。

インストール・ディレクトリの *h* サブディレクトリにインストールされているヘッダ・ファイル *dbrmt.h* は、そのようなスキームを構築するためのアプリケーション・プログラミング・インタフェースを提供します。

特定のメッセージ・システムで自分の DLL を使用するように SQL Remote に指示するには、そのシステムにレジストリ・エントリを作成する必要があります。レジストリ・エントリは、次の場所に作成してください。

```
Software
     ¥Sybase
          ¥SQL Remote
               ¥message-system
                    ¥encode_dll
```

ここでの message-system とは、SQL Remote のメッセージ・システム (**file**、**smtp** など) です。このレジストリ・エントリを自分のエンコード DLL 名に設定してください。

エンコードと復号化には互換性が必要

カスタム・エンコードを実装する場合、その DLL が受信側にもあり、自分のメッセージを正確に復号化できるようになっていることを必ず確認してください。

メッセージ・トラッキング・システム

SQL Remote にはメッセージ・トラッキング・システムが用意されていて、レプリケートされたすべてのオペレーションが正しい順序で適用され、欠落したオペレーションや二重に適用されているオペレーションがないか確認できます。

メッセージ・システムに障害があると、レプリケーション・メッセージが送信先に到着しなかったり、または壊れた状態で到着したりするおそれがあります。また、送信した順序と異なる順序で送信先に到着する可能性もあります。この項では、メッセージ・システム・エラーの検出と修正、およびメッセージの正常な適用を確認する SQL Remote のシステムについて説明します。

電子メール・メッセージ・システムを使用している場合、SQL Remote のメッセージが適切に送受信されないときは、2 台のコンピュータの間で電子メールが正しく動作していることを確認してください。

SQL Remote のメッセージ・トラッキング・システムは、SQL Remote の **remoteuser** システム・テーブルで管理されているステータス情報に基づいています。このテーブルは、Message Agent が管理しています。サブスクライバ・データベースの Message Agent はパブリッシャ・データベースに確認メッセージを送信し、サブスクリプションの最後で **remoteuser** が適切に管理されていることを確認します。

Adaptive Server Anywhere の **remoteuser** テーブルは、**sys.sysremoteuser** システム・テーブルです。Adaptive Server Enterprise では、**sr_remoteuser** テーブルです。

remoteuser テーブルのステータス情報

SQL Remote の **remoteuser** システム・テーブルには各サブスクライバに対応したローがあり、そこにはそのサブスクライバが送受信するメッセージのステータス情報が示されています。統合データベースでは、**remoteuser** に各リモート・ユーザに対応したローがあります。各リモート・データベースでは、**remoteuser** に統合データベースの情報

を管理するローが 1 つあります。(統合データベースは、リモート・データベースからパブリケーションにサブスクリプションを作成します)。

それぞれのサブスクリプションの最後にある SQL Remote の **remoteuser** システム・テーブルは、Message Agent が管理しています。

トランザクション・ログ・オフセットによるメッセージのトラッキング

メッセージ・トラッキングのステータス情報は、パブリッシャ・データベースとサブスクライバ・データベースのトランザクション・ログにあるオフセットの形式になっています。トランザクション・ログにある各 COMMIT は、十分に定義されたオフセットでマーク付けされています。トランザクションの順序は、オフセット値を比較して決定されます。

メッセージの順序

メッセージを送信すると、前回のメッセージの最後の COMMIT のオフセットによって、メッセージが順に並べられます。トランザクションがいくつかのメッセージにわたっている場合は、メッセージを正しい順序で並べるための通し番号がトランザクションにあります。デフォルトの最大メッセージ・サイズは 50,000 バイトですが、Message Agent の `-l` オプションを使用すると、この設定を変更できます。

メッセージの送信

log_sent カラムには、サブスクライバに送信された最新メッセージのローカル・トランザクション・ログのオフセットが入ります。Message Agent はメッセージを送信すると、そのメッセージの最後の COMMIT のオフセットを **log_sent** 値に設定します。メッセージが受信されてサブスクリプションを作成したデータベースに適用されると、パブリッシャには確認メッセージが返送されます。パブリッシャの Message Agent が確認メッセージを受信すると、そのサブスクライバが **confirm_sent** カラムにローカル・トランザクション・ログのオフセットとともに設定されます。**log_sent** と **confirm_sent** は、両方ともローカル・データベースのトランザクション・ログにあるオフセットで、**confirm_sent** は **log_sent** より後のオフセットにはなりません。

メッセージの受信

サブスクライバ・データベースの Message Agent がレプリケーションのアップデートを受信して適用すると、そのメッセージの最後の COMMIT のオフセットによって **log_received** カラムが更新されます。したがって、すべてのサブスクライバ・データベースの **log_received**

カラムには、パブリッシャ・データベースのトランザクション・ログにあるトランザクション・ログのオフセットがあります。オペレーションが受信され適用されると、Message Agent はパブリッシャ・データベースに確認メッセージを返送し、ローカルの SYSREMOTEUSER テーブルにある **confirm_received** 値も設定します。すべてのサブスクライバ・データベースの **confirm_received** カラムには、パブリッシャ・データベースのトランザクション・ログにあるトランザクション・ログのオフセットが入ります。

サブスクリプションの 2 通りの方法

SQL Remote のサブスクリプションには、2 通りの操作があります。各リモート・データベースは統合データベースのパブリケーションへのサブスクライバで、統合データベースは各リモート・データベースの一致するパブリケーションへサブスクライブします。したがって、統合データベースおよびリモート・データベースにある SQL Remote の **remoteuser** システム・テーブルには、補完的な情報が入っています。

Message Agent はトランザクションを適用し、**log_received** の値を自動的にアップデートします。メッセージに複数のトランザクションが含まれ、メッセージの適用中に障害が発生する場合は、**log_received** 値はすでに適用されコミットされている値と正確に一致します。

メッセージの再送

SQL Remote の **remoteuser** テーブルには、メッセージの再送を処理する 2 つのカラムがさらに含まれています。**resend_count** カラムと **rereceive_count** カラムは再試行の回数で、何らかの理由でメッセージが失われたり削除されたりすると増加します。

一般的に、**log_send** カラムには **log_sent** カラムと同じ値が入っています。しかし、**log_send** の値が **log_sent** より大きい場合、Message Agent は次の実行時にサブスクライバに直ちにメッセージを送信します。

消失または壊れたメッセージの取り扱い

サブスクライバ・データベースでメッセージが受信されると、Message Agent はそれを正しい順序 (ログのオフセットで決定される) で適用し、確認メッセージをパブリッシャに送信します。メッセージが欠落している場合、Message Agent は **rereceive_count** のローカル値を増加させ、再送を要求します。その他の現在あるメッセージや送信途中のメッセージは適用されません。

サブスクライバからのメッセージの再送要求によってパブリッシャ・データベースの **resend_count** 値が増加し、パブリッシャの **log_sent** 値も **confirm_sent** の値に設定されます。この方法で **log_sent** 値を設定し直すと、オペレーションが再送されます。

ユーザは log_sent をリセットできない
log_sent 値はシステム・テーブルにあるため、ユーザがリセットすることはできません。

メッセージの識別

各メッセージは次の 3 つの値で識別されます。

- メッセージの **resend_count**
- 前のメッセージの最後の COMMIT のトランザクション・ログ・オフセット
- 複数のメッセージにわたるトランザクションについては、トランザクションの通し番号

rereceive_count より **resend_count** の値が小さいメッセージは、適用されずに削除されます。これにより、オペレーションが 2 回以上適用されることが確実になくなります。

第 11 章

SQL Remote for Adaptive Server Anywhere の管理

この章の内容

この章では、Adaptive Server Anywhere を統合データベースとして使用している SQL Remote の管理者を対象に、設定と管理の重要な点について説明します。

Message Agent の実行

この項では、Adaptive Server Anywhere の Message Agent を実行する方法について説明します。

Adaptive Server Anywhere と Adaptive Server Enterprise に共通する Message Agent の機能については、「SQL Remote の管理」253 ページを参照してください。

Message Agent の起動

Message Agent には、それ自体の動作を制御する一連のオプションがあります。Message Agent を実行するために最低限必要なオプションは、接続パラメータ・オプション (-c) です。

接続パラメータの詳細については、『ASA データベース管理ガイド』>「接続パラメータ」を参照してください。

冗長キーワード	省略名	引数
DatabaseFile	DBF	string
DatabaseName	DBN	string
DatabaseSwitches	DBS	string
EngineName	ENG	string
Password	PWD	string
Start	Start	string
Userid	UID	string

サービスとしての Message Agent の実行

Message Agent をバッチ・モードではなく連続モードで実行している場合、サーバの稼動中は常に Message Agent を実行させておく必要があります。

このためには、Message Agent を Windows 上で「**サービス**」として実行します。サービスは、現在使用しているユーザがログ・アウトしても実行し続け、オペレーティング・システムが起動され次第すぐに起動するように設定できます。

プログラムをサービスとして実行する方法については、『ASA データベース管理ガイド』>「データベース・サーバの実行」を参照してください。

Message Agent とレプリケーション・セキュリティ

前の章のチュートリアルでは、DBA パーミッションの付与されたユーザ ID を使用して Message Agent を実行しました。メッセージ内のオペレーションは、Message Agent の接続文字列で指定されたユーザ ID から実行されます。**DBA** というユーザ ID を使用すると、あらゆる変更を加えるパーミッションをそのユーザに持たせることができます。

DBA のユーザ ID とパスワードをすべてのリモート・データベース・ユーザに配布することは、多くの場合、セキュリティとデータのプライバシー上の問題からおすすめできません。SQL Remote によるソリューションでは、メッセージに含まれる変更を実行するために Message Agent がデータベースに完全にアクセスでき、かつセキュリティ上の問題を起こさずに済むようになります。

特別なパーミッションである REMOTE DBA には、以下のような性質があります。

- **Message Agent 以外からの接続では特別なパーミッションはなし**　REMOTE DBA 権限を付与されたユーザ ID は、Message Agent 以外からの接続に対して特別な権限を持ちません。したがって、REMOTE DBA ユーザのユーザ ID とパスワードを大勢に配布しても、セキュリティの問題は発生しません。そのデータベースで CONNECT より上のパーミッションが付与されたユーザ ID を使用しないかぎり、データベース内のデータにアクセスすることはできません。

- **Message Agent からの接続では完全な DBA パーミッション**
  Message Agent から接続している場合、REMOTE DBA 権限を持つユーザ ID には、データベースに対する完全な DBA パーミッションが付与されます。

REMOTE DBA パーミッションの使用

おすすめしたいのは、統合データベースに対する REMOTE DBA 権限を、パブリッシャと各リモート・ユーザに付与するという方法です。リモート・データベースが抽出されると、リモート・ユーザはリモート・データベースのパブリッシャになり、統合データベースで付与されたのと同じパーミッションを付与されます。このパーミッションには REMOTE DBA 権限が含まれており、リモート・ユーザはこのユーザ ID を Message Agent 接続文字列で使用できます。このプロシージャを使用すると、管理するユーザ ID が余計に生じないので、各リモート・ユーザはデータベースに接続するためのユーザ ID を 1 つだけ知っていればいいということになります。このことは、Message Agent からの接続 (完全な DBA 権限が付与される) でも、他のクライアント・アプリケーションからの接続 (REMOTE DBA 権限から特に追加のパーミッションは付与されない) でも同様です。

REMOTE DBA パーミッションの付与

REMOTE DBA パーミッションを **dbremote** というユーザ ID に付与するには、次のように指定します。

```
GRANT REMOTE DBA
 TO dbremote
 IDENTIFIED BY dbremote
```

Adaptive Server Anywhere では、REMOTE DBA 権限をリモート・ユーザに付与するには、リモート・ユーザのプロパティ・シートの [権限] タブでそのオプションをオンにします。

エラーのレポートと処理

この項では、Message Agent がエラーをレポートし処理する方法について説明します。

デフォルトのエラー処理

デフォルトでは、エラーが発生すると、Message Agent はそのエラーをログ出力に記録します。Message Agent は、ログ出力をウィンドウまたは操作を記録するログ・ファイルに送信します。デフォルトでは、ログ出力はウィンドウにのみ送信されます。-o オプションを使用すると、出力がログ・ファイルにも送信されます。

Message Agent は、情報をウィンドウよりも出力ログに多く出力することがあります。Message Agent ログには、次のものが含まれます。

- 適用されたメッセージのリスト
- 失敗した SQL 文のリスト
- その他のエラーのリスト

エラーではない UPDATE 競合

UPDATE 競合はエラーではないので、Message Agent 出力にはレポートされません。

ログ・ファイルの詳細については、「Message Agent」376 ページを参照してください。

エラーの無視

SQL 文の適用中に Message Agent がエラーを検出した場合、それをレポートしないようにしたいという例外的ケースもあります。これは、エラーが発生する状況を理解しており、データの不一致が発生しないことが確実で、その結果を無視しても大丈夫な場合です。

エラーがレポートされないようにするには、既知のエラーの原因となる動作に BEFORE トリガを作成します。このトリガは、REMOTE_STATEMENT_FAILED SQLSTATE (5RW09) または SQLCODE (-288) 値を通知する必要があります。

たとえば、参照先カラムが見つからないためテーブルの INSERT 文が失敗した場合に、エラーをレポートしないようにするには、参照先カラムが存在しない場合に REMOTE_STATEMENT_FAILED SQLSTATE を通知する BEFORE INSERT トリガを作成します。INSERT 文は失敗しますが、この失敗は Message Agent ログにはレポートされません。

エラー処理プロシージャの実装

SQL Remote では、エラーが発生した場合、メッセージのログを記録するだけでなく他の処理も実行できます。Replication_error データベース・オプションを使用すると、エラーが発生したときに Message Agent で呼び出すストアド・プロシージャを指定できます。デフォルトではプロシージャを呼び出しません。

プロシージャには、データ型が CHAR、VARCHAR、または LONG VARCHAR の引数を 1 つ指定してください。プロシージャは、2 回呼び出されます。1 回はエラー・メッセージによる呼び出し、もう 1 回はエラーの原因となった SQL 文による呼び出しです。

このオプションを使用すると、レプリケーションでのエラーの追跡とモニタができますが、その場合でもやはり、エラーが起こらないように設計することが必要です。このオプションは、そのようなエラーの解決を意図したものではありません。

たとえば、プロシージャで、現在の時刻とリモート・ユーザ ID を付記してエラーをテーブルに挿入できます。その後この情報を、統合データベースにレプリケートして戻すことができます。エラーが表示されたとき、統合データベースのアプリケーションでレポートを作成したり、管理者に電子メールを送信したりできます。

REPLICATION_ERROR オプションの設定については、「SQL Remote のオプション」406 ページを参照してください。

例：エラーの通知を電子メールで送信

Message Agent でエラーが発生したときは、統合データベースで通知を受信する必要があります。この項では、エラー発生時に管理者に電子メールでメッセージを送信する方法を説明します。

ストアド・プロシージャ

この例のストアド・プロシージャは、**sp_LogReplicationError** という名前で、ユーザ **cons** が所有しています。エラー発生時にこのプロシージャを呼び出すには、Interactive SQL または Sybase Central を使用して **Replication_error** データベース・オプションを設定します。

```
SET OPTION PUBLIC.Replication_error =
     'cons.sp_LogReplicationError'
```

この通知は、以下のストアド・プロシージャで実装できます。

```
CREATE PROCEDURE cons.sp_LogReplicationError
     (IN error_text LONG VARCHAR)
 BEGIN
     DECLARE current_remote_user CHAR(255);
     SET current_remote_user = CURRENT REMOTE USER;

 // Log the error
     INSERT INTO cons.replication_audit
         ( remoteuser, errormsg)
     VALUES
         ( current_remote_user, error_text);
     COMMIT WORK;

 //Now notify the DBA an error has occurred
    // using email. We only want this information if
// the error occurred on the consolidated database
    // We want the email to contain the error strings
// the Message Agent is passing to the procedure
     IF CURRENT PUBLISHER = 'cons' THEN
         CALL sp_notify_DBA( error_text );
     END IF
 END;
```

このストアド・プロシージャは、電子メールの送信を管理するために、以下のストアド・プロシージャを呼び出します。

```
CREATE PROCEDURE sp_notify_DBA( in msg long varchar)
 BEGIN
     DECLARE rc INTEGER;
     rc=call xp_startmail( mail_user='davidf' );
     //If successful logon to mail
     IF rc=0 THEN
     rc=call xp_sendmail(
             recipient='Doe, John; John, Elton',
             subject='SQL Remote Error',
             "message"=msg);
```

```
        //If mail sent successfully, stop
            IF rc=0 THEN
                call xp_stopmail()
            END IF
        END IF
    END;
```

監査テーブル

以下は、監査テーブルの定義の例です。

```
CREATE TABLE replication_audit (
     id          INTEGER DEFAULT AUTOINCREMENT,
     pub         CHAR(30) DEFAULT CURRENT PUBLISHER,
     remoteuser     CHAR(30),
     errormsg     LONG VARCHAR,
     timestamp      DATETIME DEFAULT CURRENT TIMESTAMP,
     PRIMARY KEY (id,pub)
 );
```

各カラムの意味は、次のとおりです。

カラム	説明
pub	データベースの現在のパブリッシャ (どのデータベースで挿入されたのかを示す)
remoteuser	メッセージを適用しているリモート・ユーザ (どのデータベースから受け取ったのかを示す)
errormsg	Replication_error プロシージャに渡されたエラー・メッセージ

以下は、上記のエラーでテーブルに挿入される内容の例です。

```
INSERT INTO cons.replication_audit
     (    id,
          pub,
          remoteuser,
          errormsg,
          "timestamp")
 VALUES
     (    1,
          'cons',
```

```
            'sales',
            'primary key for table ''reptable'' is not
   unique (-193)',
            '1997/apr/21 16:03:13.836')
    COMMIT WORK
```

Adaptive Server Anywhere はストアド・プロシージャからの外部 DLL の呼び出しをサポートするので、電子メールを使用する代わりにページング・システムを構築することもできます。

エラーの例

たとえば、リモートで挿入されたローと同じプライマリ・キーを使用して統合データベースでローを挿入すると、Message Agent に次のエラーが表示されます。

```
"cons" (0-0000000000-0) メッセージを受信しました。

SQL 文が失敗しました：(-193) テーブル 'reptable' のプライマリ・
キーがユニークではありません。

INSERT INTO cons.reptable(id,text,last_contact)

VALUES (2,'dave','1997/apr/21 16:02:38.325')

COMMIT WORK
```

Doe, John と Elton, John に電子メールで届くメッセージは、それぞれ "SQL Remote Error" という件名になります。

```
テーブル 'reptable' のプライマリ・キーがユニークではありません (-
193)。

INSERT INTO cons.reptable(id,text,last_contact) VALUES
(2,'dave','1997/apr/21 16:02:52.605')
```

トランザクション・ログとバックアップの管理

バックアップをしっかり取る習慣の重要性

レプリケーションは、トランザクション・ログ内のオペレーションへのアクセスに依存して行われ、古いトランザクション・ログへのアクセスが必要になることがあります。この項では、統合データベースとリモート・データベースでバックアップ・プロシージャを設定して古いトランザクション・ログに正しくアクセスする方法を説明します。

SQL Remote の統合データベース・サイトでは、しっかりバックアップを取る習慣が大切です。トランザクション・ログを失ったときは、リモート・ユーザを再抽出しなくてはならないからです。そのため、統合データベース・サイトでは、トランザクション・ログ・ミラーリングをおすすめします。

トランザクション・ログ・ミラーとその他のバックアップ・プロシージャの詳細については、『ASA データベース管理ガイド』> 「バックアップとデータ・リカバリ」を参照してください。

古いトランザクションへのアクセスの確保

トランザクション・ログはすべて、レプリケーション・システムで必要とされなくなるまで保管しておく必要があります。

多くの場合、リモート・データベースのユーザは毎日のようにオフィス・サーバから更新を受信します。ただし、消失または削除したメッセージを、メッセージ・トラッキング・システムから再送しなければならない場合には、数日前に加えられた変更の内容が必要になることがあります。また、リモート・ユーザの休暇中にメッセージが失われた場合は、数週間も前の変更の内容が必要になります。トランザクション・ログのバックアップを毎日行っている場合、そのような以前の変更が含まれるログは、もうサーバで使用されていないことになります。

トランザクション・ログは増え続けるので、領域に関する問題が発生します。トランザクション・ログ・サイズのイベント・ハンドラを使用すると、指定サイズに達した場合にログ名を変更できます。変更したら、DELETE_OLD_LOGS オプションを使用して、不要になったログ・ファイルをクリーン・アップできます。

トランザクション・ログ・サイズの制御に関する詳細については、『ASA SQL リファレンス・マニュアル』> 「BACKUP 文」を参照してください。

トランザクション・ログ・ディレクトリの設定

現在のログ以外のトランザクション・ログをスキャンする必要がある場合、Message Agent は指定の「**トランザクション・ログ・ディレクトリ**」に保管されているすべてのトランザクション・ログ・ファイルを調べます。そのディレクトリは、Message Agent のコマンド・ラインで指定されています。

例

たとえば、次のコマンド・ラインは、ディレクトリ *e:¥archive* を調べて古いトランザクション・ログを見つけるように、Message Agent に指示します。このコマンドは、すべて 1 行で入力してください。

```
dbremote -c "eng=server_name;uid=DBA;pwd=SQL"
e:¥archive
```

ログ名は重要ではない

Message Agent は、トランザクション・ログ・ディレクトリにあるすべてのファイルを開いてログ・ファイルを見つけます。そのため、ログ・ファイルの実際の名前は重要ではありません。

この項では、バックアップ・プロシージャを設定して、このようなディレクトリを適切な形で維持する方法を説明します。

バックアップ・ユーティリティのオプション

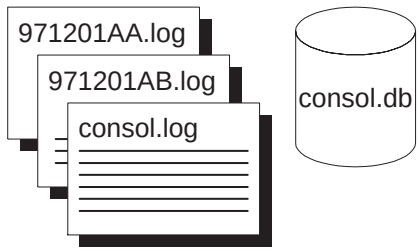
Adaptive Server Anywhere のバックアップ・ユーティリティには、動作を制御するためのオプションがいくつかあります。これらのオプションは、Sybase Central ウィザードで選択するか、*dbbackup* オプションを使って指定できます。

この項では、SQL Remote 統合データベース・バックアップでバックアップ・ユーティリティを使用する 2 つの方法について説明します。バックアップによって、Message Agent での使用に適した一連のトランザクション・ログを、常に保管しておく必要があります。

ライブ・ディレクトリをトランザクション・ログ・ディレクトリとして使用する

統合データベースとリモート・データベースのトランザクション・ログをバックアップするときは、トランザクション・ログを名前変更して再作成するオプションの使用をおすすめします。*dbbackup* ユーティリティの場合、これは `-r` オプションです。

次の図は、データベース *consol.db* とトランザクション・ログ *consol.log* が同じディレクトリにある様子を表しています。これは通常、実際の運用環境では安全なこととは言えませんが、わかりやすくするために、ここではログがデータベースと同じディレクトリにあると仮定します。ディレクトリの名前は、*c:¥live* です。



バックアップ・コマンド・ライン

以下は、名前変更と再作成のオプションを使用してデータベースをバックアップするコマンド・ラインです。

```
dbbackup -r -c "uid=DBA;pwd=SQL" c:¥archive
```

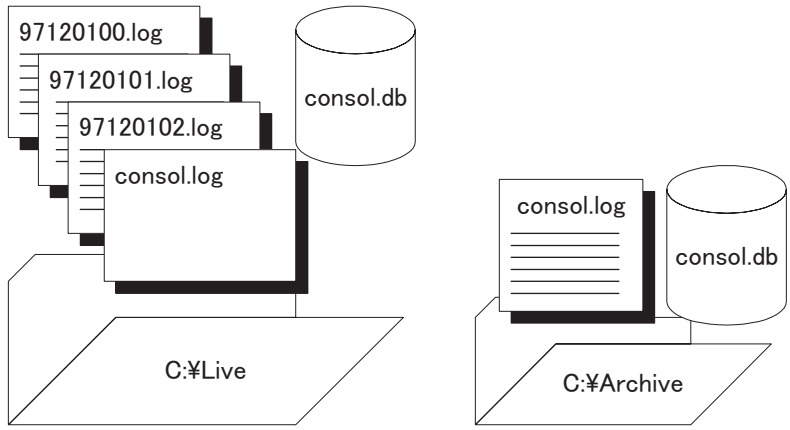
接続文字列のオプションは、データベースによって異なります。

バックアップの影響

名前の変更と再起動のオプションを使用してトランザクション・ログをディレクトリ *c:¥archive* にバックアップすると、バックアップ・ユーティリティが以下のタスクを実行します。

1. トランザクション・ログ・ファイルをバックアップし、バックアップ・ファイル *c:¥archive¥consol.log* を作成する。
2. 既存のトランザクション・ログ・ファイルの名前を *971201xx.log* (*xx* は AA から ZZ までの連続した順序の英字) に変更する。
3. *consol.log* という名前で新しいトランザクション・ログを作成する。

バックアップを何回か行うと、ライブ・ディレクトリに連続した名前の一連のトランザクション・ログができます。



Message Agent コマンド・ライン

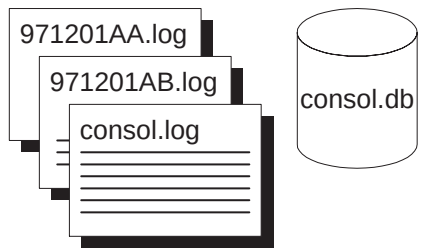
これらのログ・ファイルにアクセスしながら Message Agent を実行するには、次のコマンド・ラインを使用します。

```
dbremote -c "dbn=hq;..." c:¥live
```

バックアップ・ディレクトリをトランザクション・ログ・ディレクトリとして使用する

もう 1 つの方法では、バックアップ・ディレクトリをトランザクション・ログ・ディレクトリとして使用します。

次の図も、データベース *consol.db* がトランザクション・ログ *consol.log* と同じディレクトリにある様子を表しています。これは通常、実際の運用環境では安全なこととは言えませんが、わかりやすくするために、ここではログがデータベースと同じディレクトリにあると仮定します。ディレクトリの名前は、*c:¥live* です。



バックアップ・コマンド・ライン

次のコマンド・ラインは、名前変更と再作成のオプションを使用してデータベースをバックアップします。また、トランザクション・ログのバックアップ・ファイルの名前を変更するオプションも使用します。

```
dbbackup -r -k -c "uid=DBA;pwd=SQL" c:¥archive
```

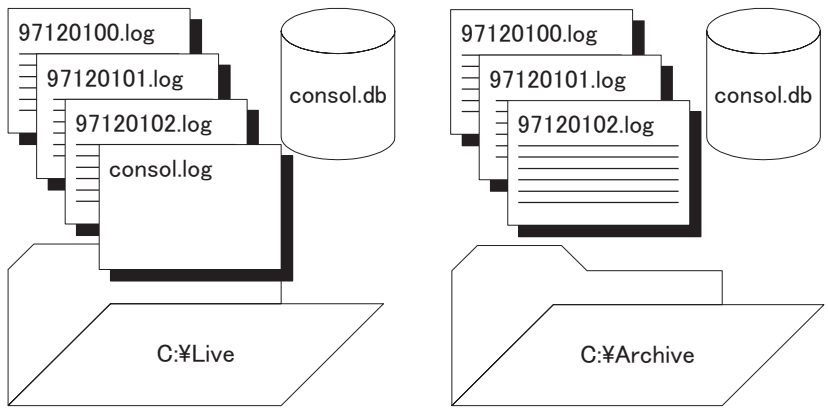
接続文字列のオプションは、データベースによって異なります。

バックアップの影響

名前の変更と再起動のオプションと、ログの名前変更オプションを使用して、ディレクトリ *c:¥archive* にトランザクション・ログをバックアップすると、バックアップ・ユーティリティが以下のタスクを実行します。

1. 既存のトランザクション・ログ・ファイルのファイル名を *971201xx.log*（*xx* は *AA* から *ZZ* までの連続した順序の英字) に変更する。
2. トランザクション・ログ・ファイルをバックアップ・ディレクトリにバックアップし、バックアップ・ファイル *971201xx.log* を作成する。
3. *consol.log* という名前で新しいトランザクション・ログを作成する。

バックアップを何回か行うと、live ディレクトリと archive ディレクトリに連続した名前の一連のトランザクション・ログができます。



Message Agent コマンド・ライン

これらのログ・ファイルにアクセスしながら Message Agent を実行するには、次のコマンド・ラインを使用します。

```
dbremote -c "dbn=hq;..." c:¥archive
```

8.0.1 より前の古いログ・ファイル名は異なる

リリース 8.0.1 より前の Adaptive Server Anywhere では、古いログ・ファイルの名前が、*yymmdd01.log*、*yymmdd02.log* のようになっています。名前を変更したのは、古いログをより多く保存できるようにするためです。Message Agent は、指定されたディレクトリ内で、ファイル名に関係なく全ファイルをスキャンするので、ログ・ファイル名が変わっても既存のアプリケーションに影響はありません。

古いトランザクション・ログの管理

トランザクション・ログはすべて、レプリケーション・システムで必要とされなくなるまで保管しておく必要があります。不要になったら廃棄してかまいません。

レプリケーション・システムがログを必要としなくなるのは、ログ・ファイルに含まれるメッセージを、すべてのリモート・データベースが受信して正常に適用し終わったときです。統合データベースからのメッセージを正常に受信したことがリモート・データベースで確認されると、統合データベースの SQL Remote テーブルに値が設定されます (「メッセージ・トラッキング・システム」306 ページを参照してください)。この受信確認をすべてのリモート・データベースから受け取った時点で、統合データベースの古いトランザクション・ログは SQL Remote で必要とされなくなります。

Delete_old_logs オプションの使用

Delete_old_logs データベース・オプションを統合データベースで使用すると、古いトランザクション・ログを自動的に管理できます。

DELETE_OLD_LOGS データベース・オプションは、デフォルトでは OFF に設定されています。ON に設定すると、不要になった古いトランザクション・ログが自動的に削除されます。古いログが不要になるのは、そのログ・ファイルに記録されている変更をすべて受信したという確認を、すべてのサブスクライバから受け取った時点です。

DELETE_OLD_LOGS オプションは、PUBLIC グループに対して、または Message Agent の接続文字列に含まれるユーザのみに対して設定できます。

例

- 以下は、PUBLIC グループに対して DELETE_OLD_LOGS を設定する文です。

```
SET OPTION PUBLIC.DELETE_OLD_LOGS = 'ON'
```

統合データベースのデータベース・メディア障害からのリカバリ

この項では、統合データベースでデータベース・デバイスのメディア障害からリカバリする方法について説明します。

リカバリの手順が最も簡単なのは、トランザクション・ログ・ファイルが 1 つしかない場合です。統合データベースでは通常は複数のトランザクション・ログ・ファイルがあり、手順も複雑になりますが、ここではまず 1 つしかない場合について説明します。その後で、複数のトランザクション・ログ・ファイルがある場合について説明します。

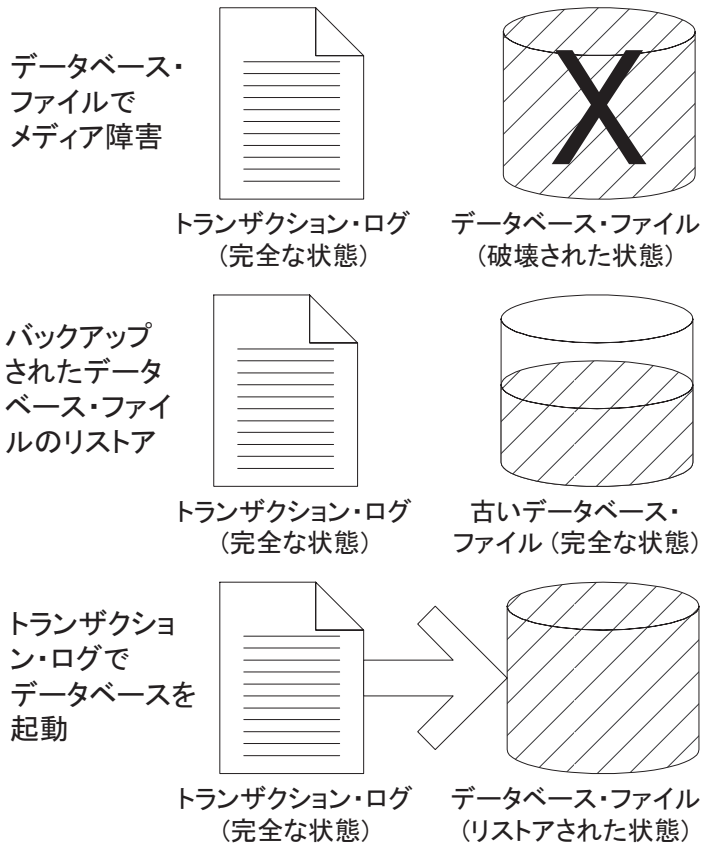
トランザクション・ログが 1 つの場合のリカバリ

ここでは、トランザクション・ログ・ファイルは 1 つしかないと仮定します。このトランザクション・ログ・ファイルは、データベースが作成されてからずっと存在しているものです。また、データベース・ファイルは前もってバックアップされており、テープなどに保存されていると仮定します。

❖ データベースをリカバリするには、次の手順に従います。

1. データベース・ファイルとログ・ファイルのコピーを作成します。
2. データベース (*.db*) ・ファイルをテープからテンポラリ・ディレクトリにリストアします。ログ・ファイルはリストア*しません*。
3. 既存のトランザクション・ログと -a オプションを使用してデータベースを起動し、トランザクションを適用してデータベース・ファイルを最新の状態に更新します。

4　データベースを通常どおり起動します。新しいアクティビティは、現在のトランザクション・ログに追加されます。



例

この例では、ミラーリングされたトランザクション・ログを使用したリカバリについて説明します。

consol.db という統合データベース・ファイルがディレクトリ *c:¥dbdir* にあり、トランザクション・ログ・ファイル *c:¥logdir¥consol.log* が *d:¥mirdir¥consol.mlg* にミラーリングされるとします。

❖ C ドライブのメディア障害からリカバリするには、次の手順に従います。

1 ミラーリングされたトランザクション・ログ *d:¥mirdir¥consol.mlg* をバックアップします。

2 障害の起きたハードウェアを交換し、影響を受けたすべてのソフトウェアを再インストールします。

3 リカバリを実行するテンポラリ・ディレクトリを作成します。ここでは *c:¥recover* とします。

4 データベース・ファイル *consol.db* の最新のバックアップを *c:¥recover¥consol.db* にリストアします。

5 ミラーリングされたトランザクション・ログ *d:¥mirdir¥consol.mlg* を、リカバリ・ディレクトリにコピーします。拡張子は *.log* に変更し、*c:¥recover¥consol.log* とします。

6 次のコマンド・ラインを使用してデータベースを起動します。

```
dbeng9 -a c:¥recover¥consol.log
c:¥recover¥consol.db
```

7 データベース・サーバを停止します。

8 リカバリされたデータベースとトランザクション・ログを *c:¥recover* からバックアップします。

9 *c:¥recover* から実際に使用する適切なディレクトリに、ファイルをコピーします。

- *c:¥recover¥consol.db* を *c:¥dbdir¥consol.db* にコピーします。
- *c:¥recover¥consol.log* を *c:¥dbdir¥consol.lOG* と *d:¥mirdir¥consol.mlg* にコピーします。

10 システムを通常どおり再起動します。

トランザクション・ログが複数の場合のリカバリ

複数のトランザクション・ログがある場合は、リカバリ手順が異なります。データベース・ファイルは前もってバックアップされており、テープなどに保存されていると仮定します。

❖ データベースをリカバリするには、次の手順に従います。

1. データベース・ファイルとログ・ファイルのコピーを作成します。
2. データベース (*.db*) ・ファイルをテープからテンポラリ・ディレクトリにリストアします。ログ・ファイルはリストア*しません。*
3. テンポラリ・ディレクトリで、データベースを起動します。このとき、-a オプションを使用して古いログを適用し、指定したトランザクション・ログを適切な順序で適用します。
4. 現在のトランザクション・ログと -a オプションを使用してデータベースを起動し、トランザクションを適用してデータベース・ファイルを最新の状態に更新します。
5. データベースを通常どおり起動します。新しいアクティビティは、現在のトランザクション・ログに追加されます。

例

c:¥dbdir¥cons.db という統合データベースがあると仮定します。トランザクション・ログ・ファイル *c:¥dbdir¥cons.log* は、*d:¥mirdir¥cons.mlg* にミラーリングされます。

毎週実行するフル・バックアップに加え、次のコマンドを使用してインクリメンタル・バックアップを毎日実行しているものとします。

```
dbbackup -c "uid=DBA;pwd=SQL" -r -t E:¥BACKDIR
```

このコマンドは、トランザクション・ログ *cons.log* をディレクトリ *e:¥backdir* にバックアップします。トランザクション・ログの名前は *datexx.log* (*date* はそのときの日付、*xx* はアルファベット順の次の英字) に変更され、新しいトランザクション・ログが作成されます。その後、ディレクトリ *e:¥backdir* がサード・パーティ・ユーティリティを使用してバックアップされます。

ここでは、名前を変更したトランザクション・ログ・ファイルを示すディレクトリを任意で指定して Message Agent を実行しているものとします。Message Agent のコマンド・ラインは次のとおりです。

```
dbremote -c "uid=DBA;pwd=SQL" C:¥DBDIR
```

毎週実行しているバックアップの 3 日後に、ディスク・ブロック破損のためにデータベース・ファイルが破壊されてしまったとします。

❖ C ドライブのメディア障害からリカバリするには、次の手順に従います。

1. ミラーリングされたトランザクション・ログ *d:¥mirdir¥cons.mlg* をバックアップします。
2. リカバリを実行するテンポラリ・ディレクトリを作成します。ここでは *c:¥recover* とします。
3. データベース・ファイル *cons.db* の最新のバックアップを *c:¥recover¥cons.db* にリストアします。
4. 名前を変更したトランザクション・ログを、次の順序で適用します。

   ```
   dbeng9 -a C:¥DBDIR¥date00.LOG C:¥RECOVER¥CONS.DB
    dbeng9 -a C:¥DBDIR¥date01.LOG C:¥RECOVER¥CONS.DB
   ```

5. 現在のトランザクション・ログ *c:¥dbdir¥cons.log* をリカバリ・ディレクトリにコピーし、*c:¥recover¥cons.log* を作成します。
6. 次のコマンドを使用してデータベースを起動します。

   ```
   dbeng9 C:¥RECOVER¥CONS.DB
   ```

7. データベース・サーバを停止します。
8. リカバリされたデータベースとトランザクション・ログを *c:¥recover* からバックアップします。
9. *c:¥recover* から実際に使用する適切なディレクトリに、ファイルをコピーします。
   - *c:¥recover¥cons.db* を *c:¥dbdir¥cons.db* にコピーします。

- *c:¥recover¥cons.log* を *c:¥dbdir¥cons.log* と *d:¥mirdir¥cons.mlg* にコピーします。

10　システムを通常どおり再起動します。

リモート・データベースでのバックアップ・プロシージャ

リモート・データベースでは、バックアップ手順は統合データベースの場合ほど重要ではありません。データのバックアップを、統合データベースへのレプリケーションに頼る方法もあります。ただし、メディア障害が発生した場合は、統合データベースからリモート・データベースを再抽出しなければならず、レプリケートされていないオペレーションは失われます (失われたオペレーションは、ログ変換ユーティリティを使用してリカバリできます)。

レプリケーションを使ってリモート・データベースのデータを保護する場合でも、トランザクション・ログが大きくなりすぎるのを防ぐために、リモート・データベースで定期的にバックアップを実行する必要があります。統合データベースで使用するのと同じオプション (ログの名前変更と再起動) を使用して Message Agent を実行し、Message Agent が名前変更されたログ・ファイルにアクセスできるようにします。リモート・データベースで DELETE_OLD_LOGS オプションを ON に設定すると、不要になった古いログ・ファイルが Message Agent によって自動的に削除されます。

自動的にトランザクション・ログの名前を変更

-x Message Agent オプションを使用すると、データベース・サーバが停止しても、リモート・コンピュータでトランザクション・ログの名前を変更する必要がなくなります。-x オプションを指定すると、トランザクション・ログで出力メッセージをスキャンした後に、名前が変更されます。

統合データベースのアップグレード

この項では、SQL Remote 環境で統合データベースをアップグレードするときの注意点について説明します。Sybase Replication Server でプライマリ・サイトとして使用されている Adaptive Server Anywhere データベースにも、同じ注意事項が当てはまります。

新しいソフトウェアをインストールしても、必ずしも新しい機能が使用可能になるわけではありません。多くの場合、新しい機能を使用可能にするには、データベースでアップグレード・ユーティリティを実行する必要があります。アップグレード・ユーティリティは、新しい機能を使用可能にするのに必要な情報をシステム・カタログに書き込みます。アップグレード・ユーティリティを実行すると、トランザクション・ログをアーカイブすることを求めるメッセージが表示されます。これは、アップグレード・ユーティリティが新しいトランザクション・ログを新しいファイル・フォーマットで作成するからです。

SQL Remote または Replication Server を使用しているときは、Message Agent と Replication Agent のトランザクション・ログを別々に保管する必要があります。アップグレード・ユーティリティを実行した後は、エンジンを停止し、ログの名前を変更してください。ログは Message Agent によって自動的に削除されます。ログは、バックアップ用にもアーカイブしてください。

アップグレード・ユーティリティの詳細については、『ASA データベース管理ガイド』> 「アップグレード・ユーティリティ」を参照してください。

レプリケーションに参加しているデータベースのアンロードと再ロード

レプリケーションされているデータベースがある場合、データベースのアンロードと再ロードには特に注意が必要です。

データベースのアンロードと再ロードについては、『SQL Anywhere Studio 新機能ガイド』> 「データベース・ファイル・フォーマットのアップグレード」を参照してください。上記の項に記載されている指示は、SQL Remote のレプリケーションで必要なデータベースにも適用されます。

この項では、データベースのアンロードと再ロードを手動で行う方法について説明します。ここで説明する方法は、スキーマの変更やデータベースでのその他の重要な変更など、これまで参照してきた自動的な手順では実行できない特殊な環境で使用してください。

レプリケーションは、トランザクション・ログに基いて行われます。データベースをアンロードして再ロードすると、古いトランザクション・ログは使用できなくなります。このため、レプリケーションが関係するときは、バックアップをきちんと実行することが特に重要です。

❖ 統合データベースをアンロードして再ロードするには、次の手順に従います (手動の場合)。

1　既存のデータベースを停止します。

2　データベース・ファイルとトランザクション・ログ・ファイルを安全なロケーションにコピーしてオフラインでフル・バックアップを実行します。

3　*dbtran* ユーティリティを実行してデータベースの現在のトランザクション・ログ・ファイルの開始オフセットと終了オフセットを表示します。終了オフセットは後で使用するために記録しておきます。

4　現在のトランザクション・ログ・ファイルの名前を変更して、アンロード処理中に修正されないようにします。このファイルをオフライン・ディレクトリに置きます。

5　既存のデータベースを起動します。

6　データベースをアンロードします。

7　既存のデータベースを停止します。このデータベースと、この手順と前の手順で作成されたログ・ファイルは、もう必要ありません。

8　新しいデータベースを初期化します。

9　データを新しいデータベースに再ロードします。

10　新しいデータベースを停止します。

11　新しいデータベース用に現在のトランザクション・ログ・ファイルを消去します。

12 新しいデータベースで *dblog* を実行します。手順 3 で記録した終了オフセットを -z オプションに指定し、相対オフセットを 0 に設定します。

```
dblog -x 0 -z 137829 database-name.db
```

13 Message Agent を実行するときは、コマンド・ラインに元のオフライン・ディレクトリのパスを入力します。

パススルー・モードの使用

統合データベースのパブリッシャは、パススルー・モードを使用してリモート・サイトに直接介入できます。パススルー・モードでは、標準の SQL 文をリモート・サイトに渡すことができます。デフォルトでは、パススルー・モード文はローカル (統合) データベースでも実行されますが、オプションのキーワード指定によって、ローカルで実行しないようにすることもできます。

警告
パススルー・オペレーションは、リモート・データベースのサブスクリプションが作成されているテスト・データベース上で必ずテストしてください。テストを行っていないパススルー・スクリプトを、運用データベースに対して実行しないでください。

パススルーの開始と停止

パススルー・モードの開始と停止には、PASSTHROUGH 文を使用します。開始の PASSTHROUGH 文とパススルー・モードを終了させる PASSTHROUGH STOP 文の間に入力された文は、構文エラーのチェックを受け、現在のデータベースで実行されます。さらに、識別されたサブスクライバに渡され、サブスクライバのデータベースでも実行されます。開始のパススルー文と停止のパススルー文の間にある一連の文を、「**パススルー・セッション**」といいます。

次の文を使用すると、ローカルのデータベースで実行されずに、指定した 2 つのサブスクライバのリストにパススルー文を渡すパススルー・セッションが開始されます。

```
PASSTHROUGH ONLY
 FOR userid_1, userid_2;
```

パススルー文の指示

次の文は、指定したパブリケーションのすべてのサブスクライバに文を渡すパススルー・セッションを開始します。

```
PASSTHROUGH ONLY
 FOR SUBSCRIPTION TO [owner].pubname [ ( string ) ] ;
```

パススルー・モードは、加算的です。次の例では、**statement_1** は **user_1** に送られ、**statement_2** は **user_1** と **user_2** の両方に送られます。

```
PASSTHROUGH ONLY FOR user_1 ;
 statement_1 ;
 PASSTHROUGH ONLY FOR user_2 ;
 statement_2 ;
```

次の文は、パススルー・モードを終了します。

```
PASSTHROUGH STOP ;
```

PASSTHROUGH STOP を実行すると、すべてのリモート・ユーザへのパススルー・モードが終了します。

パススルー文の適用順序

パススルー文は、通常のレプリケーション・メッセージと一緒に、文がログに記録された順序で連続してレプリケートされます。

パススルーは通常、データ定義言語の文を送信するために使用します。この場合、レプリケートされた DML 文は、パススルーの前に *before* スキーマを使用し、パススルーの後に *after* スキーマを使用します。

パススルー・モードの使用上の注意

- パススルー・オペレーションは常に、リモート・データベースのサブスクリプションが作成されているテスト・データベース上でテストしてください。テストを行っていないパススルー・スクリプトを、運用データベースに対して実行しないでください。
- オブジェクト名は必ず所有者名で修飾してください。PASSTHROUGH 文は、同じユーザ ID のリモート・データベースでは実行されません。したがって、所有者名の修飾子を持たないオブジェクト名は正しく解析されないことがあります。

パススルー・モードの使用法と制限事項

パススルー・モードは強力なツールなので、使用するときは注意が必要です。文 (特にデータ定義文) の中には、実行中の SQL Remote が失敗する原因となるものがあります。SQL Remote は、1 つの環境内の各データベースが同じオブジェクトを持っているものとして動作します。あるテーブルが一部のサイトのみで変更されている場合、データの変更をレプリケートしようとしても失敗します。

また、デフォルトの設定では、パススルー・モードはローカル・データベースでも文を実行するということを忘れないでください。文をローカルで実行せずにリモート・データベースに送るには、ONLY キーワードが必要です。パススルー・セクションにストアド・プロシージャへの呼び出しが含まれている場合、そのストアド・プロシージャは、パススルー・コマンドを発行するサーバ上に存在していなければなりません (これらのストアド・プロシージャの中には、サーバ上では実行されないものもあります)。次の一連の文では、リモート・データベースだけでなく統合データベースでもテーブルを削除します。

```
-- Drop a table at the remote database
 -- and at the local database
 PASSTHROUGH TO Joe_Remote ;
 DROP TABLE CrucialData ;
 PASSTHROUGH STOP ;
```

リモート・データベースのみでテーブルを削除する構文は、次のとおりです。

```
-- Drop a table at the remote database only
 PASSTHROUGH ONLY TO Joe_Remote ;
 DROP TABLE CrucialData ;
 PASSTHROUGH STOP ;
```

稼動中の SQL Remote で行えるタスクは、以下のとおりです。

- 新しいユーザを追加する。
- ユーザを再同期する。
- インストール環境からユーザを削除する。
- リモート・ユーザのアドレス、メッセージ・タイプ、または頻度を変更する。
- テーブルにカラムを追加する。

その他のスキーマ変更の多くは、稼動中の SQL Remote で行うと重大な問題を引き起こす可能性があります。

階層の 1 つのレベルのみで動作するパススルー

多層 SQL Remote インストール環境では、現在レベルのすぐ下のレベルにあるデータベースでパススルー文が動作することが重要になります。多層インストール環境では、統合データベースで、その下のレベルを動作の対象としてパススルー文を入力してください。

パススルー・モードでレプリケートされないオペレーション

パススルー・モードで特に注意が必要な文があります。

プロシージャの呼び出し

パススルー・モードで CALL 文または EXEC 文を使用してストアド・プロシージャを呼び出すと、CALL 文自体がレプリケートされ、プロシージャ内の文はレプリケートされません。レプリケート側のプロシージャは適切な作用をしていると仮定しています。

フロー制御文とカーソル処理

IF や LOOP などのフロー制御文と、すべてのカーソル処理は、パススルー・モードではレプリケートされません。ループ構造または制御構造内の文は、レプリケートされます。

カーソルの処理はレプリケートされません。カーソルによるローの挿入、カーソル内でのローの更新、またはカーソルによるローの削除は、パススルー・モードではレプリケートされません。

静的な Embedded SQL の SET OPTION 文はレプリケートされません。つまり、次の文はパススルー・モードではレプリケートされません。

```
EXEC SQL SET OPTION . . .
```

しかし、次の動的 SQL 文はレプリケートされます。

```
EXEC SQL EXECUTE IMMEDIATE "SET OPTION . . . "
```

バッチ

バッチ文 (BEGIN と END に囲まれた一連の文) は、パススルー・モードではレプリケートされません。パススルー・モードでバッチ文を使おうとすると、エラー・メッセージが表示されます。

第 12 章

SQL Remote for Adaptive Server Enterprise の管理

この章の内容

この章では、Adaptive Server Enterprise を統合データベースのサーバとして使用する SQL Remote 管理者を対象とし、設定と管理上の問題について説明します。

Adaptive Server Enterprise の Message Agent の動作

この項では、Adaptive Server Enterprise における Message Agent がどのように機能するかについて説明します。Adaptive Server Enterprise における Message Agent と、Adaptive Server Anywhere における Message Agent の機能には、大きな違いがあります。これらの機能は、この 2 つのサーバの異なる役割にそれぞれ対応しています。

Adaptive Server Anywhere と Adaptive Server Enterprise に共通する Message Agent の機能については、「Message Agent の実行」287 ページを参照してください。

Message Agent の実体は ssremote プログラム

Adaptive Server Enterprise における Message Agent は、次の実行プログラムです。

- Windows オペレーティング・システム上では、Message Agent は *ssremote.exe*
- UNIX オペレーティング・システム上では、Message Agent は *ssremote*

トランザクション・ログのスキャン

リモート・データベースに送信するトランザクションを収集するため、Message Agent は Adaptive Server Enterprise のトランザクション・ログをスキャンします。収集されたトランザクションは、「**ステーブル・キュー**」に保存されます。

ステーブル・キューの詳細については、「ステーブル・キュー」342 ページを参照してください。Message Agent によるステーブル・キューの使用方法については、「Message Agent の操作フェーズ」343 ページを参照してください。

SQL Remote Message Agent は、Adaptive Server Enterprise Log Transfer Manager (LTM) と同じトランザクション・ログ・スキャン・インタフェースを使用します。Adaptive Server Enterprise は、「**トランケーション・ポイント**」を管理します。トランケーション・ポイントとは、トランザクション・ログ内の最も古いページに対する識別子のことで、レプリケーション・システムで必要なものです。

トランザクションがトランザクション・ログからスキャンされ、ステーブル・キューでコミットされるとすぐに、SQL Remote Message Agent はトランケーション・ポイントを設定します。これにより、**dump transaction** コマンドを使用してトランザクション・ログ内の領域をできるだけ迅速に再利用することが可能になります。Message Agent は、リモート・データベースからの確認の受信を待たずにトランケーション・ポイントを設定します。

Message Agent を実行して、ログ領域を再利用してください
トランザクション・ログで領域不足が発生しないように、Message Agent を頻繁に実行してください。**dump transaction** コマンドは、トランケーション・ポイント以降のページから領域を再利用するものではありません。

Replication Server と SQL Remote

Replication Server セットアップに関連している Adaptive Server Enterprise データベース上で SQL Remote を使用するには、別の注意事項があります。データベースに対して Replication Agent (LTM) が実行されている場合、SQL Remote Open Server を追加コンポーネントとして使用してください。次のいずれかの場合に、Adaptive Server Enterprise データベースに対して Replication Agent が実行されています。

- データベースが、プライマリ・データベースとして Replication Server セットアップに関連している
- データベースが Replication Server セットアップに関連していて、非同期プロシージャ・コールを使用している

データベースが Replication Server セットアップにレプリケート・サイトとして関連していて、非同期プロシージャ・コールが使用されていない場合、SQL Remote Open Server は必要ありません。

SQL Remote Open Server の詳細については、「Replication Server における SQL Remote の使用」359 ページを参照してください。

ステーブル・キュー

Adaptive Server Enterprise における Message Agent は、トランザクションを削除できるようになるまで「**ステーブル・キュー**」に格納します。ステーブル・キューは、レプリケーション・システムでまだ必要な可能性のあるメッセージを格納する、データベース・テーブルの組み合わせです。

Adaptive Server Anywhere における SQL Remote は、ステーブル・キューを使用しません。

ステーブル・キューと Replication Server ステーブル・キューは異なるものです
Sybase Replication Server もまた、ステーブル・キューをレプリケーション・メッセージの記憶領域として使用します。Replication Server ステーブル・キューと SQL Remote ステーブル・キューは、ほぼ同様の機能を実行しますが、同じものでは*ありません*。

ステーブル・キューの保存場所

ステーブル・キューは、レプリケートされるテーブルと同じデータベースに保存されるか、または、別のデータベースに保存されます。ステーブル・キューを別のデータベースに保存すると、バックアップとリカバリ処理が複雑になりますが、別のデバイスや Adaptive Server Enterprise サーバにステーブル・キューの負荷を分散することによって、パフォーマンスを改善できます。

ステーブル・キューを直接修飾しないてください
ステーブル・キューは、Message Agent によって管理されています。ステーブル・キューを直接修正しないでください。

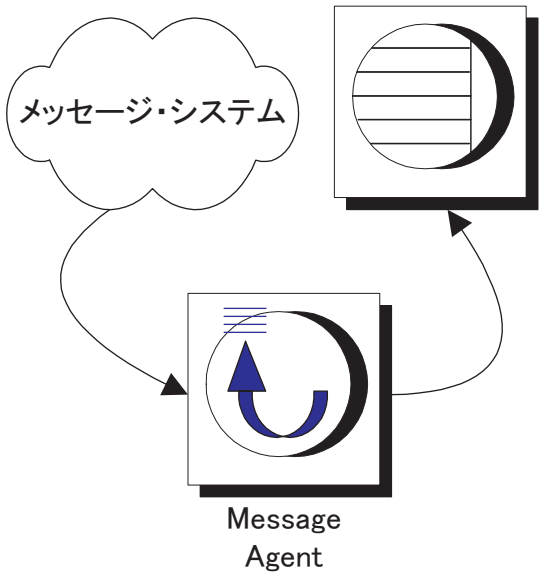
ステーブル・キューは、トランザクション・ログからスキャンされた全トランザクションの情報を保存する、一連のテーブルで構成されます。

これらのテーブルの各カラムについては、「ステーブル・キュー・テーブル」447 ページを参照してください。

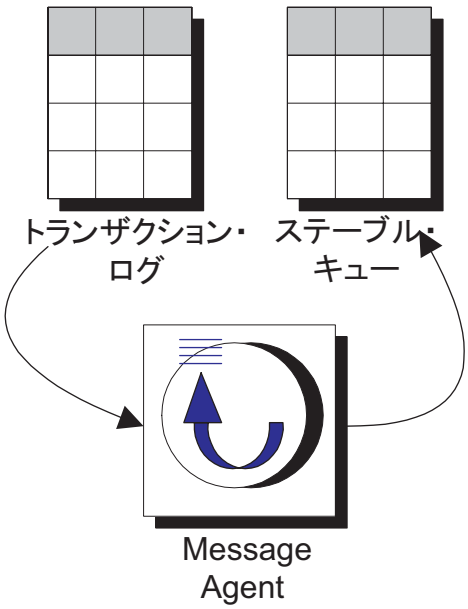
Message Agent の操作フェーズ

Message Agent には、次の実行フェーズがあります。

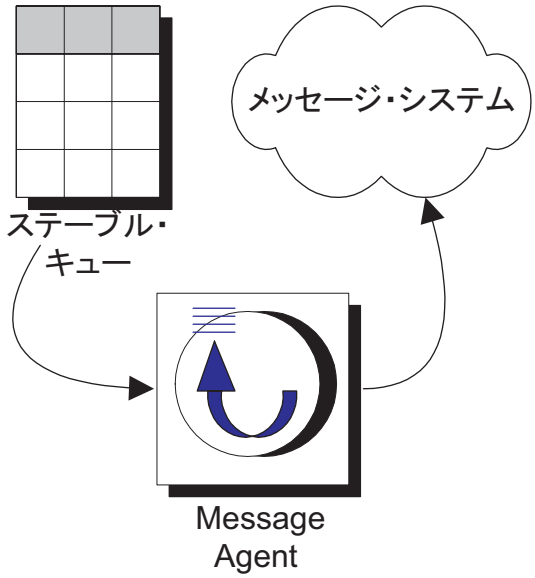
- **メッセージの受信**　このフェーズでは、Message Agent は入力メッセージを受信し、そのメッセージを Adaptive Server Enterprise サーバに適用します。



- **ステーブル・キューの移植**　このフェーズでは、Message Agent は Adaptive Server Enterprise のトランザクション・ログをスキャンしてステーブル・キューに保存します。



- **メッセージの送信** このフェーズでは、Message Agent はステーブル・キューから出力メッセージを作成します。



すべてのリモート・データベースからの確認を受信するまで、トランザクションはステーブル・キューに保存されたままです。確認を受信すると、トランザクションは Message Agent によって自動的にステーブル・キューから削除されます。

Message Agent は、ステーブル・キューの保存されているデータベースが SQL Remote システム・テーブルのデータベースと異なる場合、そのデータベースのトランザクション・ログをスキャンしません。

複数の Message Agent を実行して、これらのタスクを実行する方法の詳細については、「複数の Message Agent の実行」347 ページを参照してください。

Message Agent の実行

この項では、Adaptive Server Enterprise における Message Agent の実行方法について説明します。Adaptive Server Anywhere と Adaptive Server Enterprise に共通する Message Agent の機能については、「Message Agent の実行」287 ページを参照してください。

Message Agent とレプリケーション・セキュリティ

このマニュアルのこれまでのチュートリアルでは、システム管理者パーミッションのあるユーザ ID を使用して Message Agent を実行しました。メッセージ内の操作は、Message Agent の接続文字列で指定されたユーザ ID から実行されます。システム管理者ユーザ ID を使用すれば、すべての変更を行うことのできるパーミッションがユーザに確実にあることになります。

実際にはそのようなユーザ ID は使用しませんが、Message Agent の実行には、レプリケーションの役割を持ったユーザ ID の使用が必要です。レプリケーションの役割は、次の文で付与できます。

```
sp_role 'grant', replication_role, user_name
```

テーブルに対して変更を適用するため、Message Agent のユーザには、すべてのレプリケートされたテーブルに対する挿入、更新、削除のパーミッションが必要です。また、レプリケーション・エラー・プロシージャも Message Agent ユーザ ID で作成してください。

Adaptive Server Enterprise データベースをセットアップする場合、Message Agent で使用するユーザ名と同じ名前で、スクリプト *ssremote.sql* とスクリプト *squeue.sql* を実行してください。

設定手順については、「SQL Remote の設定」21 ページを参照してください。

Message Agent ユーザ ID のパスワードを非表示にするには、*ssremote* コマンド・ライン・オプションをファイルに保存し、*@data* パラメータを指定して *ssremote* スクリプトを実行します。ファイル・システム・セキュリティを使用して、このファイルに対する不正なアクセスを防げます。

複数の Message Agent の実行

Message Agent の 3 つの操作フェーズについては、「Message Agent の操作フェーズ」343 ページの項で説明しました。3 つのフェーズを要約すると、次のようになります。

- メッセージの受信
- トランザクション・ログのスキャン
- メッセージの送信

必要に応じて、これらのフェーズを別々の Message Agent で実行できます。Message Agent コマンド・ラインで、実行するフェーズを Message Agent に対して指定できます。

実行するフェーズの指定

コマンド・ライン・オプションは次のとおりです。

- **受信**　-r コマンド・ライン・オプションを使用して、Message Agent が実行している間、メッセージを受信するように指示します。有効なメッセージの受信後に Message Agent を停止するには、`-r` の後ろに `-b` オプションを指定します。
- **ログのスキャン**　-I コマンド・ライン・オプションを使用して、Message Agent が実行している間、トランザクション・ログをスキャンして、ステーブル・キューに保存するように指示します。
- **送信**　-s コマンド・ライン・オプションを使用して、Message Agent が実行している間、メッセージを送信するように指示します。
- **複数のフェーズ**　`-r`、`-I`、`-s` のいずれも指定しない場合、Message Agent は 3 つのフェーズすべてを実行します。それ以外の場合、指定されたフェーズのみ実行されます。

次のように、複数の Message Agent の実行が必要な場合がいくつかあります。

- **トランザクション・ログで領域不足を発生させない**　トランザクション・ログが満杯にならないように注意してください。それには、トランザクション・ログを頻繁にスキャンして、

SQL Remote で必要なすべてのエントリがステーブル・キューに確実に保存されるようにします。このため、メッセージの送受信をバッチ・モードのみで実行していても、Message Agent を継続的に実行してトランザクション・ログをスキャンしてください。

- **オペレーティング・システムの混在** ひとつのオペレーティング・システムでサポートされるメッセージ・リンクを使用する場合は、そのプラットフォーム上で Message Agent を使用して、メッセージの送受信を行ってください。ログのスキャンを UNIX マシン上で実行している場合でも、2 つの Message Agent を実行することでこれは可能になります。

Message Agent の同期方法

複数の Message Agent の操作は、**sr_marker** と呼ばれるテーブルによって同期されます。このテーブルは、**marker** という、**datetime** データ型の単一カラムからなります。

トランザクションがスキャンされてステーブル・キューに保存されるのを待つ場合、Message Agent は **sr_marker** を更新して、システムに操作が渡されるのを待ちます。**sr_queue_state** テーブル内のカラム marker には、トランザクション・ログからスキャンされる最新のマーカが含まれます。

エラーのレポートと処理

この項では、Message Agent がエラーをレポートし処理する方法について説明します。

デフォルトのエラー処理

デフォルトでは、エラーが発生すると、Message Agent はそのエラーをログ出力に記録します。Message Agent は、ログ出力をウィンドウまたは操作を記録するログ・ファイルに送信します。デフォルトでは、ログ出力はウィンドウにのみ送信されます。-o コマンド・ライン・オプションを使用すると、出力がログ・ファイルにも送信されます。

Message Agent は、情報をウィンドウよりも出力ログに多く出力することがあります。Message Agent ログには、次のものが含まれます。

- 適用されたメッセージのリスト
- 失敗した SQL 文のリスト
- その他のエラーのリスト

エラーではない UPDATE 競合

UPDATE 競合はエラーではないので、Message Agent 出力にはレポートされません。

エラー処理プロシージャの実装

SQL Remote では、エラーが発生した場合、メッセージのログを記録するだけでなく他の処理も実行できます。REPLICATION_ERROR データベース・オプションを使用すると、エラーが発生したときに Message Agent で実行するストアド・プロシージャを指定できます。デフォルトでは、プロシージャは実行されません。

プロシージャには、データ型が CHAR または VARCHAR の引数を 1 つ指定してください。プロシージャは、すべてのエラー・メッセージ、エラーの原因となった SQL 文で呼び出されます。

このオプションを使用すると、レプリケーションでのエラーの追跡とモニタができますが、その場合でもやはり、エラーが起こらないように設計することが必要です。このオプションは、そのようなエラーの解決を意図したものではありません。

たとえば、プロシージャで、現在の時刻とリモート・ユーザ ID を付記してエラーをテーブルに挿入できます。その後この情報を、統合データベースにレプリケートして戻すことができます。エラーが表示されたとき、統合データベースのアプリケーションでレポートを作成したり、管理者に電子メールを送信したりできます。

REPLICATION_ERROR オプションの設定については、「SQL Remote のオプション」406 ページを参照してください。

Adaptive Server Enterprise のトランザクション・ログとバックアップの管理

リモート・データベースにレプリケートされたトランザクションが失われるのを防止する必要があります。すでにリモート・データベースにレプリケートされたトランザクションが失われた場合、リモート・データベースと統合データベースとの一貫性が失われます。この場合、リモート・データベースをすべて再抽出する必要があります。

トランザクション・ログのメディア障害からの保護

トランザクション・ログでメディア障害が発生すると、コミットされたトランザクションが失われる原因となります。トランザクション・ログがスキャンされて、すでにサブスクライバ・データベースに送信された場合、パブリッシュされたデータベースから失われたトランザクションが、サブスクリプションを作成したデータベースに含まれ、これらのデータベースは一貫性のない状態になります。

トランザクション・ログの必要性

すべてのエントリがスキャンされてステーブル・キューに保存された後でも、データベース・ファイルのメディア障害を防止するため、トランザクション・ログは必要です。データベースが失われた場合、リモート・データベースへ送信されたすべてのトランザクションがそろっていたポイントに復旧してください。

この復旧を実行するには、データベース・ダンプをリストアし、トランザクション・ダンプをロードして、データベースを最新の状態にします。最後にリストアされたトランザクション・ダンプとは、障害が発生した時点のアクティブなトランザクション・ログのダンプです。

トランザクション・ログの消失の防止

トランザクション・ログのメディア障害による一貫性の喪失を防止するには、次の 2 つの方法があります。

- **トランザクション・ログのミラーリング**　デバイスがミラーリングされると、そのデバイスへのすべての書き込みは別の物理デバイスにコピーされます。
- **バックアップしたトランザクションのみレプリケート**　あるコマンド・ライン・オプションを使用して、トランザクションがバックアップされるまで、Message Agent がトランザクションを送信しないように指定します。

トランザクション・ログのミラーリング

トランザクション・ログのメディア障害を防止する唯一の方法は、トランザクション・ログのミラーリングです。

ディスクをミラーリングすると、メディア障害が発生したときでも、すぐにリカバリすることができます。**disk mirror** コマンドは、Adaptive Server Enterprise データベース・デバイスを複製します。つまり、このデバイスへのすべての書き込みは別の物理デバイスにコピーされます。1 つのデバイスに障害が発生した場合でも、もう 1 つのデバイスにすべてのトランザクションの最新のコピーが含まれます。

Adaptive Server Enterprise におけるディスクのミラーリングの詳細については、『*Adaptive Server Enterprise システム管理ガイド*』の「データベース・デバイスのミラーリング」の章を参照してください。

バックアップしたトランザクションのみレプリケート

Message Agent には、バックアップされたトランザクションのみ送信するという、コマンド・ライン・オプション (-u) もあります。これは Adaptive Server Enterprise では、最新の **dump database** コマンドまたは **dump transaction** コマンドの前に、トランザクションが完了することを意味します。

アプローチの選択

この方式の目的は、リモート・データベースの再抽出が必要となる可能性を、許容できるレベルまで減少させることです。大規模なセットアップの場合、再抽出 (つまり、停止している時間) にかかるコストはかなり高くなるため、その可能性をかぎりなくゼロに近づけることが必要となります。

- 統合データベースの全トランザクションのリカバリが不要で、ミラーリングが高負荷になりすぎるような場合に、トランザクションのミラーリングの代わりに、Message Agent の -u コマンド・ライン・オプションを使用できます。この方法は、小規模なセットアップや、統合データベース上にローカル・ユーザがまったくないセットアップの場合でも、有効です。
- Message Agent の -u コマンド・ライン・オプションを使用すると、ミラーリングのほかに、サイト全体の障害や二重メディア障害に対する保護も追加できます。

ステーブル・キューのリカバリ問題

ステーブル・キューを別のデータベースに保存すると、2 つのデータベースを一貫したバージョンに復旧する必要があるため、バックアップとリカバリ処理が複雑になります。

通常のリカバリでは自動的に 2 つのデータベースが一貫した状態にリストアされますが、メディア障害からのリカバリでは、いくつかの処理が必要です。データベース・ダンプとトランザクション・ダンプをリストアする場合、ステーブル・キューを一貫性のあるポイントまでリカバリすることが重要です。

次の 2 つのステーブル格納データベースのプロシージャを使用すると、メディア障害からのリカバリに役立ちます。

- **sp_queue_dump_database**　このプロシージャは、dump database がトランザクション・ログからスキャンされるときに呼び出されます。
- **sp_queue_dump_transaction**　このプロシージャは、dump transaction がトランザクション・ログからスキャンされるときに呼び出されます。

ステーブル格納データベースで **dump database** コマンドと **dump transaction** コマンドを発行するように、このストアド・プロシージャを修正できます。

トランザクション・ログの管理

Adaptive Server Enterprise の「**ログ転送**」インタフェースを使用すると、Message Agent で Adaptive Server Enterprise のトランザクション・ログをスキャンできます。このインタフェースを使用すると、トランザクション・ログに「**トランケーション・ポイント**」が設定されます。トランケーション・ポイントは、SSREMOTE がスキャンする前に、Adaptive Server Enterprise がトランザクション・ログのページを再利用するのを防止します。したがって、DUMP TRANSACTION を使用して、最初に開いたトランザクションの前のトランザクション・ログ・ページを解放する必要はありません。DUMP TRANSACTION は、「トランケーション・ポイント」以降のトランザクション・ログ・ページを解放しません。

トランケーション・ポイントの初期化

SQL Remote の設定スクリプト (*ssremote.sql*) は、次のコマンドでトランケーション・ポイントを初期化します。

```
dbcc settrunc( 'ltm', 'valid' ).
```

トランケーション・ポイントは、次のコマンドでリセットします。

```
dbcc settrunc( 'ltm', 'ignore' ).
```

これにより、トランケーション・ポイントを無視し、トランケーション・ポイント以降のトランザクション・ログ・ページを再利用のために解放するよう、Adaptive Server Enterprise に通知します。データベースの SQL Remote レプリケーションの必要がなくなり、DUMP TRANSACTION コマンドでトランザクション・デバイスの領域を再利用する場合のみ、このコマンドを使用してください。トランケーション・ポイントを無視した後も SQL Remote を継続すると、すべてのトランザクションのレプリケートに失敗します。これは、Message Agent でスキャンされず、DUMP TRANSACTION で解放されたトランザクション・ログ・ページで発生します。

スキーマの変更

SQL Remote でレプリケートされるテーブルへのスキーマ変更は、「**クワイエット・システム**」上で行います。クワイエット・システムとは、次のことを意味します。

- **トランザクションがまったくレプリケートされない**　テーブルを変更するトランザクションがまったくレプリケートされません。テーブルを変更するすべてのトランザクションがトランザクション・ログからスキャンされ、ステーブル・キューに保存されてから、スキーマを変更します。これは、Message Agent を普通に実行するか、`-I -b` オプションを使用することで行われます。Message Agent の完了後、スキーマを変更できます。

- **Message Agent**　スキーマを変更するときは、Message Agent を停止します。

- **SQL Remote Open Server**　SQL Remote Open Server を使用している場合、スキーマを変更するときはそれを停止します。

スキーマの変更には、アーティクルの追加や修正などの、パブリケーションの変更が含まれます。ただし、サブスクリプションの作成や削除、リモート・ユーザの追加や削除は、クワイエット・システム上で実行する必要はありません。

Adaptive Server Enterprise のトランザクション・ログには、テーブル構造の変更情報は記録されません。SQL Remote ログのスキャン処理は、Adaptive Server Enterprise のシステム・テーブルからテーブル構造を取得します。その結果 Message Agent は、古いテーブル構造に対して発生した操作を、トランザクション・ログからスキャンできません。

スキーマが変更される前にステーブル・キューに保存された情報は古いテーブル定義を使用しており、スキーマ変更以後に保存された情報は新しいテーブル定義を使用します。

パススルー・モードをスキーマの変更と同時に使用して、リモート・データベースでのスキーマの変更が正しい順序で発生することを確認できます。

パススルー・モードの使用

統合データベースのパブリッシャは、パススルー・モードを使用してリモート・サイトに直接介入できます。パススルー・モードでは、標準の SQL 文をリモート・サイトに渡すことができます。

パススルー文の受信者の決定

パススルーの宛先は、**sp_passthrough_user** と **sp_passthrough_subscription** によって決定されます。これらのプロシージャのどちらを実行しても、後続のパススルー文すべてに対する受信者が決定されます。

sp_passthrough_user と **sp_passthrough_subscription** のどちらを実行しても、受信者の現在のリストに追加します。**sp_passthrough_stop** プロシージャは、パススルーをリセットします。つまり、受信者のリストをリセットして空にします。

Adaptive Server Enterprise では、**sp_passthrough** は統合データベースで文を実行することはありません。パススルー SQL 文は、リモート・データベースに対してのみ適用されます。

パススルー文

パススルー SQL 文をレプリケートするには、**sp_passthrough** を呼び出します。

Adaptive Server Enterprise の VARCHAR(255) 制限のため、長い SQL 文は分割して作成してください。**sp_passthrough_piece** を呼び出すと、1 つの SQL 文を作成します。最後の文で **sp_passthrough** を呼び出すと、作成された文がレプリケートされます。

パススルー・モードの使用上の注意

- パススルー・オペレーションは常に、リモート・データベースのサブスクリプションが作成されているテスト・データベース上でテストしてください。テストを行っていないパススルー・スクリプトを、運用データベースに対して実行しないでください。
- オブジェクト名は必ず所有者名で修飾してください。PASSTHROUGH 文は、同じユーザ ID のリモート・データベースでは実行されません。したがって、所有者名の修飾子を持たないオブジェクト名は正しく解析されないことがあります。

スキーマの変更

Adaptive Server Enterprise のログ転送インタフェースには、テーブルのカラムの数とデータ型の情報は含まれていません。SSREMOTE はこの情報を、Adaptive Server Enterprise のシステム・テーブルから直接取得します。このため、テーブルを変更し、ALTER TABLE の前にスキャン操作が発生すると、エラーとなります。SSREMOTE が、レプリケートされたテーブルでの全操作以降に「トランケーション・ポイント」を設定してから、スキーマの変更を行ってください。レプリケートされたテーブルに対する操作が、SSREMOTE の実行からスキーマの変更の間に行われないようにしてください。

第 13 章

Replication Server における SQL Remote の使用

この章の内容

この章では、Replication Server のインストール環境に加わる Adaptive Server Enterprise データベースで SQL Remote を使用するために必要な、追加コンポーネントについて説明します。

SQL Remote Open Server を使用する必要がある場合

「ステーブル・キュー」342 ページの項に示すように、Adaptive Server Enterprise の Message Agent は、Adaptive Server Enterprise のトランザクション・ログをスキャンし、ステーブル・キューに移植します。SQL Remote メッセージは、ステーブル・キューのトランザクションから構築されます。

Message Agent は、Adaptive Server Enterprise の Replication Agent と同じインタフェースを使用してトランザクション・ログをスキャンします。したがって、Replication Server 設定のプライマリ・サイト (またはプライマリ・データへの非同期更新を許可するレプリケート・サイト) である Enterprise データベースのトランザクション・ログを、Message Agent はスキャンできません。

使用する Adaptive Server Enterprise データベースに対して実行中の Replication Agent がある場合は、追加コンポーネントとして SQL Remote Open Server を使用する必要があります。この場合、Replication Server がステーブル・キューへの移植を行うように SQL Remote を設定します。SQL Remote Message Agent はトランザクション・ログをスキャンしません。代わりに、SQL Remote Open Server が Replication Server からのトランザクションを受信します。受信されたトランザクションは、SQL Remote Open Server によって解析され、SQL Remote のステーブル・キューに格納されます。

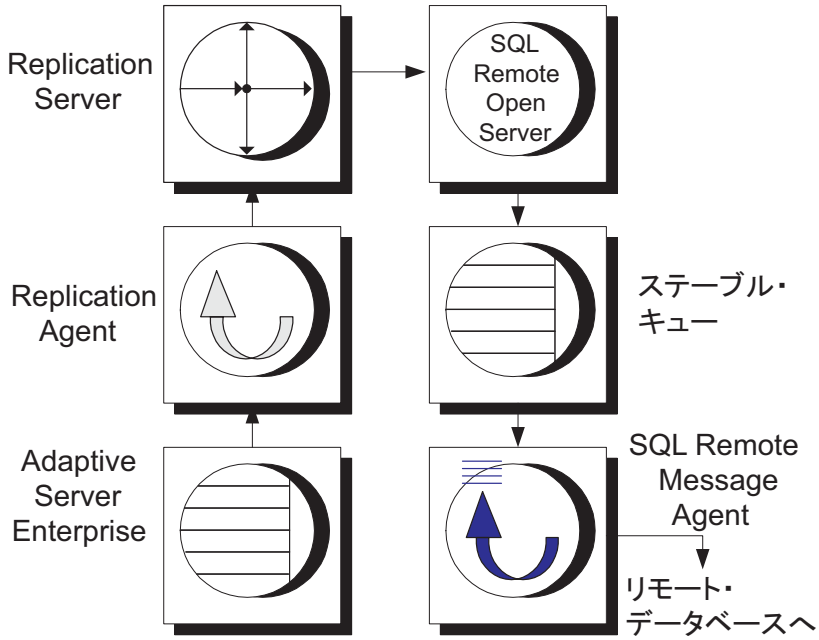
この章では、Replication Server の知識があることを前提としています。詳細については、Replication Server のマニュアルを参照してください。

Open Server ランタイム・コンポーネントが必要

SQL Remote には Open Server ランタイム・コンポーネントが入っていません。SQL Remote Open Server を使用するため、Sybase から別途 Open Server ランタイム・コンポーネントを入手する必要があります。

Replication Server/SQL Remote インストール環境のアーキテクチャ

Replication Server のプライマリ・サイトとして、および SQL Remote データベースとして、データベースを使用するための準備について次の図に示します。この図は、ステーブル・キューがレプリケートされるデータとは異なるデータベースに保持される場合について示しています。また、ステーブル・キューをレプリケートされるデータと同じデータベースに保持することもできます。すべての接続はクライアント／サーバ接続です。このため、同じまたは異なるマシン上でコンポーネントを実行できます。

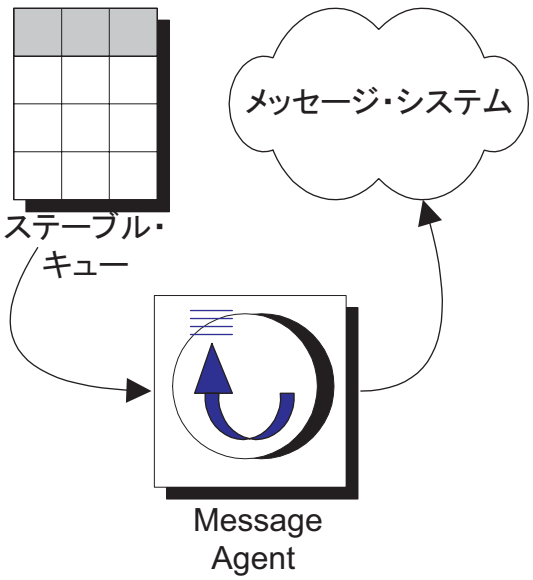


各部分を組み合わせる方法

SQL Remote Open Server は、Replication Server 設定のレプリケート・データベースとして動作します。このため、Adaptive Server Enterprise データベースでは、SQL Remote レプリケーションに関連するすべてのテーブルといくつかの SQL Remote システム・テーブルについて、レプリケーションの定義とサブスクリプションが必要です。

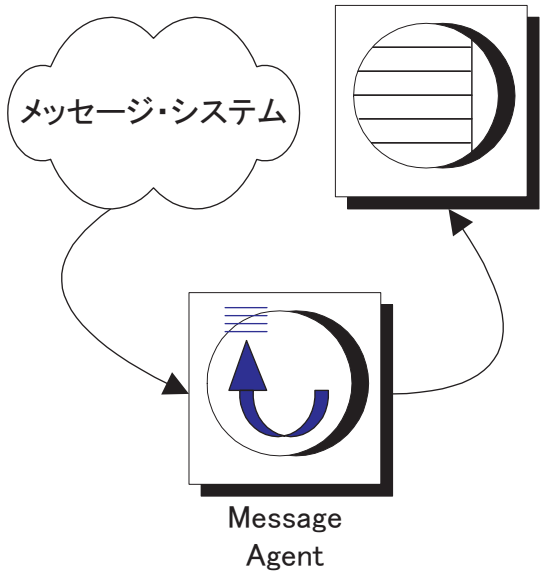
ステーブル・キューの内容

オペレーションはすべて SQL Remote Open Server にレプリケートされ、そのステーブル・キューに格納されます。ステーブル・キューは、レプリケートされるテーブルのコピーを持ちません。挿入、更新、削除のオペレーションを解析して、トランザクションを構築します。トランザクションはすべて、単一のテーブルの image カラムに格納されます。Message Agent はこれらのトランザクションを使用し、SQL Remote メッセージを構築します。



入力メッセージ

Message Agent は常に、入力 SQL Remote メッセージを直接 Adaptive Server Enterprise へ適用します。Replication Server にはオペレーションを送信しません。ステーブル・キューへの移植方法に関らず、入力メッセージは直接統合データベースへ適用されます。競合解決も同じ方法で実行されます。



Replication Server と SQL Remote

SQL Remote では、統合データベースとリモート・データベースの間で双方向レプリケーションを実行できます。Replication Server は、統合データベースから SQL Remote Open Server への単方向レプリケーションを実行します。Replication Server 側から見ると、リモート SQL Remote データベースで生じたトランザクションが、統合 SQL Remote データベースで生じたトランザクションとして表示されます。

SQL Remote システム・テーブル

SQL Remote Open Server は、SQL Remote システム・テーブルからのパブリケーションとサブスクリプションに関する情報を必要とします。Open Server は、この情報を保持する Adaptive Server Enterprise データベースへの接続を使用して、起動時にそれを取得します。

Open Server の実行中に SQL Remote システム・テーブルが更新される場合は、正確な時間に SQL Remote Open Server がこの情報を受信する必要があります。このため、SQL Remote システム・テーブルの一部をレプリケーション用にマーク付けする必要があります。これについては、「SQL Remote Open Server の設定」365 ページで説明しています。

SQL Remote Open Server 実行プログラム

SQL Remote Open Server の実行プログラムは、次のとおりです。

- *ssqueue.exe* (Windows オペレーティング・システムの場合)
- *ssqueue* (UNIX オペレーティング・システムの場合)

SQL Remote Open Server の設定

この項では、SQL Remote Open Server を使用して、SQL Remote インストール環境を設定する方法について説明します。SQL Remote ステーブル・キューが、レプリケートされるテーブルとは別の Adaptive Server Enterprise データベースに保持されるか、同じ Adaptive Server Enterprise データベースに保持されるかによって、手順が異なります。

ステーブル・キューのロケーションの詳細については、「ステーブル・キュー」342 ページを参照してください。

データの初期コピー

設定手順では、抽出ユーティリティを使用して、各リモート・データベースのデータの初期コピーを作成すると仮定しています。Replication Server のマテリアライゼーション機能を、この目的で使用しないでください。

SQL Remote Open Server の設定手順には、次の 2 つの段階があります。

- **SQL Remote 設定の準備**　SQL Remote インストール環境がすでに存在するかどうかによって、この段階は異なります。
- **SQL Remote Open Server の設定への追加**　以前のインストール環境に関係なく、この段階は同一です。

❖ SQL Remote 設定の準備をする (SQL Remote インストール環境がすでに存在する場合) には、次の手順に従います。

1. クワイエット・プライマリ・データベースで、Message Agent を使用して、残りのトランザクションをステーブル・キューにスキャンします。

   クワイエット・データベースとは、Message Agent も SQL Remote Open Server も動作していない、どのトランザクションもレプリケートされていないデータベースのことです。

2. 統合サイトで SQL Remote ソフトウェアをアップグレードするには、「SQL Remote for Adaptive Server Enterprise のアップグレード」29 ページの項の手順に従ってください。

3 次のコマンドを使用して、統合データベースで Message Agent のトランケーション・ポイントを無効にします。

```
dbcc settrunc( 'ltm', 'ignore' )
```

4 ステーブル・キュー・データベースで、ストアド・プロシージャ **sp_queue_log_transfer_reset** を実行します。

❖ SQL Remote 設定の準備をする (SQL Remote インストール環境がない場合) には、次の手順に従います。

1 「SQL Remote の設定」21 ページで示すように、SQL Remote を設定します。

2 この時点で SQL Remote のパブリケーションとサブスクリプションを設定します。この手順の詳細については、「Adaptive Server Enterprise の SQL Remote 設計」179 ページを参照してください。

3 リモート・データベースを抽出します。この手順の詳細については、「抽出ユーティリティの使用」241 ページを参照してください。

これで、SQL Remote Open Server を設定するための準備が完了しました。

❖ SQL Remote Open Server を設定する場合には、次の手順に従います。

1 SQL Remote のステーブル・キューが別のデータベースに存在する場合は、次を実行します。

- ステーブル・キュー・データベースを、Replication Server 設定内のレプリケート・データベースとして設定します。これによって、Replication Server が必要とするテーブルやプロシージャ (**rs_lastcommit** など) が作成されます。
- Replication Server のステーブル・キュー・データベースへの接続を削除します。

2　SQL Remote Open Server の interfaces ファイルにエントリを追加します。SQL Remote Open Server のコマンド・ラインで使用されるデフォルト名は、**SSQueue** です。

3　SQL Remote Open Server を起動します。

4　Replication Server の SQL Remote Open Server への接続を作成します。この接続に使うユーザ ID とパスワードは、ステーブル・キューの接続に使う SQL Remote Open Server コマンド・ライン (-cq オプション、または -cq が指定されていなければ -c スイッチ) で指定されたユーザ ID とパスワードに一致する必要があります。

この時点で Replication Server を設定する

この時点で、接続する Replication Server を設定します。詳細については、「Replication Server の設定」369 ページを参照してください。

5　SQL Remote テーブル **sr_marker**、**sr_remoteuser**、**sr_subscription**、**sr_passthrough** に対する、Replication Server のレプリケーション定義とサブスクリプションを作成します。スクリプト **ssremote.rs** は、このタスクを実行するサンプル・スクリプトです。スクリプトでサーバとデータベースの名前を編集し、実際に使用している名前と一致させる必要があります。

SQL Remote システム・テーブルにデータが 1 つでもある場合は、マテリアライゼーションが起こらないようにレプリケーション定義を作成します。

マテリアライゼーションを使用しないレプリケーション定義の作成に関する詳細については、『*Replication Server 管理ガイド*』を参照してください。「*第 10 章 サブスクリプションの管理*」の「バルク・マテリアライゼーション」の項では、リモート・データベースにデータが存在する場合の Replication Server の設定方法について説明しています。

6 SQL Remote がレプリケートする必要があるデータベースのテーブルに対する、レプリケーション定義とサブスクリプションを作成します。これらは、マテリアライゼーションを使用しないで作成する必要があります。

Replication Server の設定

この項では、SQL Remote Open Server と同時に使用する、Replication Server の設定方法について説明します。

Replication Server の SQL Remote Open Server への接続には、いくつかの設定パラメータ・セットが必要です。

dsi_xact_group_size パラメータの設定

デフォルトでは、Replication Server は複数のトランザクションを大きなトランザクションにグループ化します。**dsi_xact_group_size** パラメータは、グループ化されたトランザクションの最大サイズを制御します。

トランザクションのグループ化を無効にするには、**dsi_xact_group_size** パラメータを、−1 に設定する必要があります。SQL Remote 設定の別のリモート・データベースから生じたトランザクションは、同じグループにしないでください。

パラメータの設定方法

次の文を使用してパラメータを設定できます。

```
CONFIGURE CONNECTION TO  "ssqueue_server"
 SET dsi_xact_group_size TO '-1'
```

dsi_num_threads パラメータの設定

SQL Remote Open Server は、パラレル DSI スレッドをサポートしません。Replication Server を設定し、SQL Remote の接続でパラレル DSI スレッドを使用しないようにする必要があります。

SQL Remote のデータに対するレプリケーション定義の作成

SQL Remote がレプリケートするテーブルのレプリケーション定義には、ある特性が必要です。この項では、これらの特性について説明します。

状況によっては、SQL Remote は UPDATE オペレーションを INSERT または DELETE としてレプリケートします (「更新のレプリケーション」98 ページを参照してください)。これは、Replication Server のマニュアルで「**サブスクリプション・マイグレーション**」と呼ばれます。UPDATE を INSERT としてレプリケートするには、SQL Remote ではローの更新前イメージ全部が必要です。つまり Replication Server は、INSERT としてレプリケートされる必要のあるテーブルに対する UPDATE の WHERE 句に全カラム値を指定する必要があります。

これらを指定する最も簡単な方法は、レプリケーション定義の PRIMARY KEY にすべてのカラムをリストすることです。これによって、Replication Server は、すべての更新の WHERE 句に全カラムを入れるようになります。REPLICATE MINIMAL COLUMNS をこれらのレプリケーション定義で使用すると、更新の SET 句に全カラムがリストされてしまうことを防げます。

text カラムと image カラム

Replication Server は、レプリケーション定義のプライマリ・キーの TEXT カラム、または IMAGE カラムを受け入れません。TEXT と IMAGE 以外のすべてのカラムは、レプリケーション定義のプライマリ・キーのリストに含めます。TEXT カラムと IMAGE カラムは、すべて ALWAYS_REPLICATE 句に指定します。レプリケーション定義では、REPLICATE MINIMAL COLUMNS の代わりに REPLICATE ALL COLUMNS を使用する必要があります。これによって、Replication Server は、更新時は常に TEXT カラムと IMAGE カラムの更新前イメージを SQL Remote Open Server へ送信するようになります。

dsi_sql_data_style データ・スタイルの使用

Replication Server 11.5 には、SQL Remote 用の新しい **dsi_sql_data_style** があります。このデータ・スタイルは、すべての UPDATE 文の WHERE 句へカラム全部を自動的に含めます。レプリケーション定義のプライマリ・キーに、すべてのカラムをリストする必要はありません。REPLICATE MINIMAL COLUMNS を使用したレプリケーション定義は、更新するローの完全な更新前イメージを Replication Server が維持できないようにします。このため、SQL Remote の **dsi_sql_data_style** は、REPLICATE MINIMAL COLUMNS が使用されていると動作しません。

接続のサスペンドとレジューム

Replication Server の SQL Remote Open Server への接続を設定したら、この接続をサスペンド／レジュームし、パラメータの設定を有効にする必要があります。次のコマンドを使用して、このタスクを実行します。

```
suspend connection to ssqueue_server
 go
 resume connection to ssqueue_server
 go
```

その他の問題

この項では、SQL Remote を Replication Server とともに使用する方法について、その他の問題をリストします。

Message Agent の実行 Message Agent は、受信と送信のコマンド・ライン・オプション (-r と -s) を使用して実行してください。これによって、Message Agent によるトランザクション・ログのスキャンを防止します。Replication Agent の実行中に Message Agent がトランザクション・ログをスキャンすると、「ログ転送コンテキスト」の予約を試みてエラーが発生します。

SQL Remote Open Server におけるプロシージャの呼び出し
SQL Remote Open Server は、Replication Server から受信するプロシージャの呼び出しすべてを、ステーブル・キュー・データベースへ渡します。たとえば、**rs_get_lastcommit** と **rs_update_lastcommit** はステーブル・キュー・データベースで実行されます。

コーディネート・ダンプ Replication Server には、メイン・データベースとステーブル・キュー・データベースの間の、データベース・ダンプとトランザクション・ログ・ダンプを調整するメカニズムがあります。ファンクション文字列 **rs_dumpdb** と **rs_dumptran** は、ステーブル・キュー・データベースのコーディネート・ダンプを実行する場合に使用できます。詳細については、Replication Server のマニュアルを参照してください。

スキーマ変更 SQL Remote のインストール環境に対してスキーマを変更する場合は、クワイエット・システムで実行しなければなりません。これには、SQL Remote Open Server の停止も含まれます。

第 4 部　リファレンス

第 4 部では、SQL Remote のリファレンス情報を紹介します。

第 14 章

ユーティリティとオプションのリファレンス

この章の内容

この章では、SQL Remote のユーティリティと SQL Remote のデータベース・オプションのリファレンス項目について説明します。

また、レプリケーション処理のカスタマイズに使用できるクライアント・イベント・フック・プロシージャについても説明します。

Message Agent

内容

SQL Remote メッセージの送信と適用、およびメッセージの配信を確認するメッセージ・トラッキング・システムの管理を行います。

構文

{ **dbremote** | **ssremote** } [*options*] [*directory*]

オプション

オプション	説明
@data	指定された環境変数または設定ファイルからオプションを読み込む。
-a	受け取ったトランザクションを適用しない。
-b	バッチ・モードで実行する。
-c *"keyword=value; …"*	データベース接続パラメータを指定する。
-cq *"keyword=value; …"*	ステーブル・キューのデータベース接続パラメータを指定する (Adaptive Server Enterprise のみ)。
-dl	画面にログ・メッセージを表示する。
-ek *key*	暗号化キーを指定する。
-ep	暗号化キーを入力するよう要求する。
-e *locale-string*	ロケールを設定する (Adaptive Server Enterprise のみ)。
-fq	メッセージを送信するときにステーブル・キューの完全スキャンを行う (Adaptive Server Enterprise のみ)。
-g *n*	*n* 個未満のオペレーションで構成されるグループ・トランザクション
-l	トランザクション・ログからトランザクションをステーブル・キューにスキャンする (Adaptive Server Enterprise のみ)。
-k	完了してウィンドウを閉じる。
-l *length*	メッセージの最大長

オプション	説明
-m *size*	メッセージの作成に使用する最大メモリ・サイズ
-ml *directory*	オフライン・ミラー・ログのロケーションを指定する。
-o *file*	メッセージをファイルに出力する。
-os *size*	出力メッセージを記録するファイルの最大サイズ
-ot *file*	ファイルをトランケートし、出力メッセージを記録する。
-p	メッセージをパージしない。
-q	最小化ウィンドウで実行する。
-r	メッセージを受信する。
-rd *minutes*	入力メッセージのポーリング頻度
-ro *filename*	ファイルにリモート出力を記録する。
-rp *number*	メッセージを消失したと判断するまでのポーリング受信回数
-rt *filename*	トランケートし、ファイルにリモート出力を記録する。
-ru *time*	再送信の宛先のログを再スキャンする待ち時間
-s	メッセージを送信する。
-sd *time*	送信ポーリング間隔
-t	すべてのトリガをレプリケートする (Adaptive Server Anywhere のみ)。
-u	バックアップしたトランザクションだけを処理する。
-ud	UNIX プラットフォーム上でデーモンとして実行する。
-v	冗長オペレーション
-w *n*	入力メッセージを適用するワーカ・スレッドの数 (NetWare と Windows CE 以外)

オプション	説明
-x [*size*]	トランザクション・ログの名前を変更し、再起動する (Adaptive Server Anywhere のみ)。
directory	古いトランザクション・ログが保存されるディレクトリ (Adaptive Server Anywhere のみ)

説明

Message Agent は、SQL Remote レプリケーションのメッセージの送信と適用、およびメッセージの配信を確認するメッセージ・トラッキング・システムの管理を行います。

Message Agent の実行プログラム名は次のとおりです。

- **dbremote** Adaptive Server Anywhere の Message Agent
- **ssremote** Adaptive Server Enterprise の Message Agent

推奨されなくなった機能

SQL Anywhere Studio の次のメジャー・リリースでは、SQL Remote for Adaptive Server Enterprise はなくなります。Mobile Link が、Adaptive Server Enterprise と Adaptive Server Anywhere データベース間のデータ同期を実現する、より柔軟でスケーラブルなソリューションを提供します。

DBTools ライブラリに呼び出すと、自分のアプリケーションから Message Agent を実行することもできます。詳細については、SQL Remote インストール・ディレクトリの h サブディレクトリにある *dbrmt.h* ファイルを参照してください。

Adaptive Server Anywhere では、Message Agent コマンドのユーザ ID には REMOTE DBA か DBA 権限が必要です。Adaptive Server Enterprise のユーザ ID には、レプリケーションの役割が必要です。

オプションの *directory* パラメータは、古いトランザクション・ログが保存されているディレクトリを指定します。これにより、Message Agent は現在のログが開始される前のイベントにアクセスできます。

Message Agent はいくつかのデータベース接続を使用します。詳細については、「Message Agent で使用される接続」289 ページを参照してください。

REMOTE DBA 権限の詳細については、「Message Agent とレプリケーション・セキュリティ」313 ページを参照してください。

オプションの詳細

@data　指定された環境変数または設定ファイルからオプションを読み込みます。指定された環境変数と設定ファイルが両方とも存在する場合は、環境変数が使用されます。

環境変数には、あらゆるオプションのセットを格納できます。たとえば、次の文の組み合わせは、4 MB のキャッシュ・サイズで起動し、メッセージだけを受信し、**myserver** というサーバの **field** というデータベースに接続するデータベース・サーバ用に、オプションのセットを格納する環境変数を設定します。**set** 文は 1 行で入力してください。

```
set envvar=-m 4096 -r
     -c "eng=myserver;dbn=field;uid=sa;pwd=sysadmin"
 ssremote @envvar
```

設定ファイルには、改行や任意のオプションを指定できます。たとえば、次のコマンド・ファイルには、4 MB のキャッシュ・サイズで起動し、メッセージだけを送信し、**myserver** というサーバの **field** というデータベースに接続する Message Agent 用のオプションのセットが格納されています。

```
-m 4096
 -s
 -c "eng=myserver;dbn=field;uid=sa;pwd=sysadmin"
```

この設定ファイルを *c:¥config.txt* として保存すると、コマンドで次のように使用できます。

```
ssremote @c:¥config.txt
```

または

```
dbremote @c:¥config.txt
```

-a　受信したメッセージ (受信ボックス内にあるもの) を、データベースに適用しないで処理します。このオプションを使用するときに、-v (冗長出力) と -p (メッセージがパージされない) の両方を指

定すると、入力メッセージの問題を検出しやすくなります。このオプションを使用するときに -p を指定しないと、メッセージを適用しないで受信ボックスがパージされるため、サブスクリプションを再開する場合に便利です。

-b バッチ・モードで実行します。このモードでは、Message Agent は入力メッセージを処理し、トランザクション・ログを 1 回スキャンし、出力メッセージを処理して停止します。

-c "parameter=value; …" 接続パラメータを指定します。Adaptive Server Anywhere では、このオプションを指定しない場合、環境変数 SQLCONNECT が使用されます。

たとえば、次の文ではデータベース・ファイル *c:Program Files¥Sybase¥SQL Anywhere 9¥asademo.db* で *dbremote* を実行します。ユーザ ID は **DBA**、パスワードは **SQL** を使用しています。

```
dbremote -c "uid=DBA;pwd=SQL;dbf=c:¥Program
Files¥Sybase¥SQL Anywhere 9¥asademo.db"
```

Message Agent を実行するユーザには、REMOTE DBA 権限または DBA 権限が必要です。

REMOTE DBA 権限の詳細については、「Message Agent とレプリケーション・セキュリティ」313 ページを参照してください。

Adaptive Server Anywhere の Message Agent は、Adaptive Server Anywhere 接続パラメータの全種類をサポートします。Adaptive Server Enterprise の Message Agent は、次の接続パラメータをサポートします。

パラメータ	説明
UID	ログイン ID
PWD	パスワード
DBN	データベース名 (オプション)。このパラメータを指定しないと、デフォルトとして、ログイン ID のデフォルト・データベースに接続されます。
ENG	Adaptive Server Enterprise の名前

-cq "parameter=value; …"　ステーブル・キューの接続パラメータを指定します。このオプションは、Adaptive Server Enterprise にのみ適用されます。指定しないと、デフォルトとして -c の値が使用されます。

-dl　[Message Agent] ウィンドウまたはコマンド・プロンプトにメッセージを表示します。指定されている場合はログ・ファイルにも出力します。

暗号化キーを指定する (-ek)　このオプションを使用すると、強力に暗号化されたデータベースの暗号化キーをコマンド・プロンプトで直接指定できます。強力に暗号化されたデータベースを扱う場合には、データベースやトランザクション・ログ (オフライン・トランザクション・ログなど) を使用するのに、常に暗号化キーを使用する必要があります。強力に暗号化されたデータベースの場合、-ek または -ep のどちらかを指定してください。両方同時には指定できません。強力に暗号化されたデータベースでは、キーを指定しないとコマンドが失敗します。

暗号化キーを入力するよう要求する (-ep)　このオプションを使用すると、暗号化キーの入力を求めるプロンプトを表示するよう指定できます。このオプションでは、暗号化キーを入力するためのダイアログ・ボックスが表示されます。クリア・テキストでは暗号化キーを見ることができないようにすることで、高いセキュリティが得られます。強力に暗号化されたデータベースの場合、-ek または -ep のどちらかを指定してください。両方同時には指定できません。強力に暗号化されたデータベースでは、キーを指定しないとコマンドが失敗します。

-e locale-string　このオプションは、Adaptive Server Enterprise にのみ適用されます。Adaptive Server Enterprise のロケール情報を指定します。ロケール文字列のフォーマットは次のとおりです。

```
"language_name,charset_name[,sort_order]"
```

デフォルトでは、Message Agent はデフォルト・ロケールを使用します。このロケールは、ファイル *sybase¥locales¥locales.dat* に定義されています。

language_name と *charset_name* を指定しないと、Message Agent は Adaptive Server Enterprise からこれらの値を取得します。*sort_order* を指定しないと、バイナリ・ソート順 (バイト値の順にソート) が使用されます。

-fq　このオプションは、Adaptive Server Enterprise でだけ使用します。メッセージの送信時にステーブル・キューの完全なスキャンを許可します。スキャンは sr_remoteuser テーブルの最も古い **confirm_sent** 値から開始します。

この機能は、大きなステーブル・キューを時々クリーニングするためのものです。たとえば、1 人のユーザがだいぶ以前からメッセージの受信を確認していない場合、そのステーブル・キューは非常に大きくなることがあります。ただし、`-fq` を実行すると、より新しいユーザから確認済みのエントリを削除できます。また、これらのユーザが、デフォルトでエントリが削除された時点のカットオフ値より新しい場合も同じです。

-g *n* 個未満のオペレーションのトランザクションを、後に続くトランザクションとまとめるように Message Agent に指示します。デフォルトのオペレーション数は 20 です。*n* の値を大きくするとコミットが少なくなるため、入力メッセージの処理速度が向上します。ただし、トランザクションのサイズが大きくなると、デッドロックやブロックの原因になることもあります。

-I　トランザクション・ログからトランザクションをステーブル・キューにスキャンします。このオプションは、Adaptive Server Enterprise でだけ使用できます。トランザクション・ログのスキャンとメッセージの送受信のために、Message Agent のコピーを別々に実行するときに使用します。

`-r`、`-I`、`-s` のいずれも指定しない場合、Message Agent は 3 つのフェーズすべてを実行します。それ以外の場合、指定されたフェーズのみ実行されます。

詳細については、「複数の Message Agent の実行」347 ページを参照してください。

-k　`-o` パラメータと併用すると、完了時にウィンドウを閉じます。

-l length　送信する各メッセージの最大長をバイトで指定します。長いトランザクションは複数のメッセージに分割されます。デフォルトは 50000 バイトで、最小長は 10000 バイトです。

警告
メッセージの最大長は、同一のインストール環境内のすべてのサイトで必ず同じ長さにしてください。

メモリの割り付けに制限のあるプラットフォームの場合、この値はオペレーティング・システムのメモリ割り付けの最大値より小さい値にしてください。

-m size　メッセージ作成と入力メッセージのキャッシュに、Message Agent が使用する最大メモリ・サイズを指定します。使用できるサイズは、*n* (バイト)、*n*K、*n*M で指定します。デフォルトは 2048 K (2 M) です。

すべてのリモート・データベースが、レプリケートされるオペレーションのユニーク・サブセットを受信する場合、それぞれのリモート・データベースにメッセージが同時に作成されます。同じオペレーションを受信するリモート・ユーザのグループには、メッセージが 1 つだけ作成されます。使用されるメモリが -m 値を超えると、メッセージの送信後に最大サイズ (-l オプションで指定したサイズ) に達します。

メッセージが届くと、適用されるまで Message Agent によってメモリ内に格納されます。このようなメッセージのキャッシュによって、適切でないメッセージをメッセージ・システムが再読み込みしないようにできますが、大規模なインストール環境ではパフォーマンスが低下することもあります。-m オプションを使用して指定したメモリ使用量が超えると、メッセージは、最近使用した方式でフラッシュされます。

-ml　このオプションを指定すると、次のどちらかの場合に、dbremote が古いミラー・ログ・ファイルを削除できます。

- オフライン・ミラー・ログが、ミラー・トランザクション・ログと異なるディレクトリに存在する。
- dbremote がリモート・データベース・サーバと異なるマシン上で実行されている。

通常の設定では、アクティブなミラー・ログと名前が変更されたミラー・トランザクション・ログは同じディレクトリ内に存在し、dbremote はリモート・データベースと同じマシン上で実行されるた

め、このオプションを指定しなくても、古いミラー・ログ・ファイルは自動的に削除されます。このディレクトリ内のトランザクション・ログが影響を受けるのは、DELETE_OLD_LOGS データベース・オプションが ON または DELAY に設定されている場合だけです。

-o 出力をログ・ファイルに追加します。デフォルトでは、画面に出力が表示されます。

-os 出力メッセージをロギングするファイルの最大サイズを指定します。使用できるサイズは、*n* (バイト)、*n***K** (KB)、または *n***M** (MB) で指定します。デフォルトによる制限はなく、最小制限は 10000 バイトです。

SQL Remote では、現在のファイル・サイズをチェックしてから、出力メッセージをファイルに記録します。ログ・メッセージによって、ファイル・サイズが指定したサイズより大きくなると、SQL Remote は出力ファイルの名前をそれぞれ *yymmddxx.dbr* (dbremote の場合) と *yymmddxx.ssr* (ssremote の場合) に変更します。xx は AA 〜 ZZ のアルファベット順の英字で、*yymmdd* は現在の年月日を表します。

Message Agent を継続モードで長時間実行する場合、このオプションを使用して、手動で古いログ・ファイルを削除し、ディスク領域を解放できます。

-ot ログ・ファイルをトランケートし、このファイルに出力メッセージを追加します。デフォルトでは、画面に出力が表示されます。

-p メッセージをパージしないで処理します。

-q ウィンドウ操作のオペレーティング・システムの場合にのみ、Message Agent を最小化ウィンドウで起動します。

-r メッセージを受信します。`-r`、`-I`、`-s` のいずれも指定しない場合、Message Agent は 3 つのフェーズすべてを実行します。それ以外の場合、指定されたフェーズのみ実行されます。

-r で呼び出された場合、Message Agent は継続モードで動作します。メッセージ受信後に Message Agent を停止するには、`-r` オプションと `-b` オプションを使用します。

-rd time　デフォルトでは、Message Agent は入力メッセージを 1 分ごとにポーリングします。このオプション (rd は「**受信遅延 (receive delay)**」の意味) によってポーリング頻度を設定できます。これは、ポーリングが高負荷である場合に便利です。

頻繁にポーリングする場合は、秒数を表す数の後にサフィックス **s** を付けると便利です。次に例を示します。

```
dbremote -rd 30s
```

このように指定すると、30 秒ごとにポーリングされます。

ポーリングの詳細については、「受信メッセージのポーリングのチューニング」297 ページを参照してください。

-ro　このオプションは統合サイトで使用します。統合データベースに出力ログ情報を送信するようにリモート・データベースを設定する場合、このオプションを使用すると情報がファイルに書き込まれます。このオプションを指定すると、管理者がリモート・サイトでエラーのトラブルシューティングを行う場合に役立ちます。

詳細については、「リモート・サイトでのトラブルシューティング・エラー」291 ページを参照してください。

-rp　Message Agent を継続モードで実行すると、メッセージが一定の間隔でポーリングされます。設定回数 (デフォルトでは 1 回) のポーリングが行われた後にメッセージが見つからないと、Message Agent はそのメッセージが失われたと判断して、再送信を要求します。このため、メッセージ・システムの処理速度が遅い場合、必要のない再送信要求が多く出されることがあります。このオプションを使用して、再送信要求が出される前に行うポーリング回数を指定すると、再送信要求の数を少なくできます。

このオプションの設定方法については、「受信メッセージのポーリングのチューニング」297 ページを参照してください。

-rt　このオプションは統合サイトで使用します。ファイルが起動時にトランケートされることを除いて、-ro オプションと同じです。

-ru　「**再送信緊急度 (resend urgency)**」を制御します。再送信要求の検出から、Message Agent がこの要求の処理を開始するまでの時間を指定します。このオプションを使用すると、Message Agent が複数の

ユーザの再送信要求を集めてからログの再スキャンを行うことができます。指定できる時間の単位は {s = 秒、m = 分、h = 時間、d = 日数 } のいずれかです。

-s メッセージを送信します。-r、-I、-s のいずれも指定しない場合、Message Agent は 3 つのフェーズすべてを実行します。それ以外の場合、指定されたフェーズのみ実行されます。

-sd time 「**送信遅延 (send delay)**」を制御します。これは、ポーリングとポーリングの間に送信されるトランザクション・ログ・データを待つ時間です。

-t すべてのトリガ・アクションがレプリケートされます。このオプションを使用する場合は、リモート・データベースでトリガ・アクションが 2 回実行されないことを確認してください。この 2 回とは、リモート・サイトで起動されるトリガによる 1 回と、統合データベースからレプリケートされたアクションの明示的な適用による 1 回です。

トリガ・アクションが 2 回実行されないようにするには、トリガの本文を IF CURRENT REMOTE USER IS NULL … END IF 文で囲みます。このオプションは、Adaptive Server Anywhere でのみ使用できます。

-u バックアップされたトランザクションだけを処理します。このオプションを指定すると、最後にバックアップされた以降のトランザクションは処理されません。これにより、出力トランザクションと入力トランザクションの確認とが、バックアップされてから送信されるようになります。

Adaptive Server Anywhere の場合、名前が変更されたログのトランザクションだけが処理されます。Adaptive Server Enterprise の場合は、最後の **dump database** 文または **dump transaction** 文の前にコミットされたトランザクションだけが処理されます。

-ud UNIX プラットフォーム上では、-ud オプションを指定して、Message Agent をデーモンとして実行できます。

Message Agent をデーモンとして実行する場合は、-o または -ot オプションも指定して、出力情報のログを取ってください。

Message Agent をデーモンとして実行し、FTP または SMTP メッセージ・リンクを使用する場合は、データベースにメッセージ・リンク・パラメータを格納してください。これは、Message Agent をデーモンとして実行していると、これらのオプションの入力をユーザに要求しないためです。

メッセージ・リンク・パラメータの詳細については、「メッセージ・タイプ制御パラメータの設定」274 ページを参照してください。

-v　冗長出力です。このオプションによって、メッセージに含まれる SQL 文が画面に表示されます。-o または -ot オプションを指定するとログ・ファイルに出力されます。

-w n　受信メッセージの適用に使用されるワーカ・スレッドの数です。デフォルトは 0 です。この場合、すべてのメッセージがメインの (1 つだけの) スレッドによって適用されます。1 の値を指定すると、1 つのスレッドがメッセージ・システムからメッセージを受信し、1 つのスレッドがメッセージをデータベースに適用します。

-w オプションを指定すると、ハードウェアのアップグレードによって、受信メッセージのスループットを増加できます。多くのオペレーションを同時に実行できるデバイスに統合データベースを配置すると、入力メッセージのスループットが向上されます。このような統合データベースの配置方法をストライプ論理ドライブの RAID アレイといいます。また、Message Agent を実行するコンピュータにマルチ・プロセッサを搭載しても、入力メッセージのスループットは向上します。

多くのオペレーションを同時に実行できないハードウェアでは、-w オプションを使用してもパフォーマンスはそれほど向上しません。

単一のリモート・データベースからの入力メッセージを複数のスレッドに適用できません。単一のリモート・データベースからのメッセージは、常に適切な順序で逐次適用されます。

-x　出力メッセージがスキャンされてから、トランザクション・ログの名前を変更し、再起動します。場合によっては、リモート・データベースのバックアップが実行されたり、データベース・サーバをシャットダウンするときにトランザクション・ログの名前を変更する代わりに、統合データベースにデータがレプリケートされます。このオプションは、Adaptive Server Anywhere でのみ使用できます。

オプションの *size* 修飾子が指定された場合、トランザクション・ログは、指定されたサイズよりも大きい場合にのみ名前を変更されます。使用できるサイズは、*n* (バイト)、*n*K、*n*M で指定します。デフォルトは 0 です。

メッセージ・システム制御パラメータ

SQL Remote は、複数のレジストリ設定を使用してメッセージ・リンク動作を制御します。

メッセージ・リンク制御パラメータは、次の場所に格納されます。

- **Windows** レジストリ内の次の場所に格納されます。

```
¥¥HKEY_CURRENT_USER
    ¥Software
        ¥Sybase
            ¥SQL Remote
```

- **NetWare** FILE システム・ディレクトリ設定を保存するために、*sys:¥system* ディレクトリにファイル *dbremote.ini* を作成してください。

レジストリ設定のリストについては、「メッセージ・タイプの使用」 269 ページの各メッセージ・システムの項を参照してください。

データベース抽出ユーティリティ

次の方法で、リモート・データベース抽出ユーティリティにアクセスできます。

- 対話的に使うには、Sybase Central からアクセスする。
- *ssxtract* または *dbxtract* ユーティリティを使って、システム・コマンド・プロンプトからアクセスする。バッチまたはコマンド・ファイルへの組み込みには、このユーティリティが便利です。

  ssxtract は Adaptive Server Enterprise の抽出ユーティリティで、*dbxtract* は Adaptive Server Anywhere の抽出ユーティリティです。

デフォルトでは、抽出ユーティリティは独立性レベル 0 で実行されます。アクティブなサーバからデータベースを抽出する場合は、独立性レベル 3 (「抽出ユーティリティのオプション」394 ページを参照) で実行し、抽出されたデータベース内のデータがサーバ上のデータと一致するようにします。独立性レベル 3 で実行すると、多数のロックが必要になるため、サーバ上の他のユーザのターンアラウンド・タイムに影響が出ることがあります。サーバへのアクセスが少ないときに抽出ユーティリティを実行するか、データベースのコピーに対して抽出ユーティリティを実行することをおすすめします (「効率的な抽出プロシージャの設計」245 ページを参照)。

dbo によって所有されるオブジェクト

dbo ユーザ ID は、Adaptive Server Anywhere データベースに Adaptive Server Enterprise 互換のシステム・オブジェクトを保有します。

Adaptive Server Anywhere では、データベースの作成中に **dbo** ユーザ ID 用に作成されたオブジェクトは抽出ユーティリティでアンロードされません。データをアンロードするとき、システム・プロシージャの再定義など、これらのオブジェクトに加えられた変更は失われます。抽出ユーティリティでアンロードされるため、データベースの初期化以降に **dbo** ユーザ ID によって作成されたオブジェクトは保存されます。

Sybase Central でのリモート・データベースの抽出

Sybase Central から抽出ユーティリティを実行すると、SQL Remote サブスクリプションの作成と同期に関連する次のタスクを実行できます。

- 指定したパブリケーションのデータのコピーを含むリモート・データベースを構築するため、コマンド・ファイルを作成する。
- メッセージ・タイプ、パブリッシャ・ユーザ ID およびリモート・ユーザ ID、パブリケーションおよびサブスクリプションなどの SQL Remote オブジェクトを作成する。これらのオブジェクトは、統合データベースとの間でメッセージの送受信を行うため、リモート・データベースで必要なものです。
- 統合データベースとリモート・データベースの両方でサブスクリプションを開始する。

注意

[データベースの抽出] ウィザードには、[所有者別にオブジェクトをフィルタ] ダイアログで選択したユーザのテーブルだけが表示されます。特定のデータベース・ユーザが所有するテーブルを表示するには、アンロードしているデータベースを右クリックし、ポップアップメニューで [所有者別にオブジェクトをフィルタ] を選択し、表示されたダイアログで目的のユーザを選択します。

❖ 実行中のデータベースからリモート・データベースを抽出する場合は、次の手順に従います (Sybase Central)。

1. データベースに接続します。
2. データベースを右クリックし、ポップアップ・メニューから [データベースの抽出] を選択します。
3. ウィザードの指示に従います。

❖ **データベース・ファイルまたは実行中のデータベースからリモート・データベースを抽出する場合は、次の手順に従います (Sybase Central)。**

1. 左ウィンドウ枠で、Adaptive Server Anywhere プラグインを選択します。
2. 右ウィンドウ枠で、[ユーティリティ] タブをクリックします。
3. 右パネルの [データベースの抽出] をダブルクリックします。
4. ウィザードの指示に従います。

抽出ユーティリティ

内容

Adaptive Server Anywhere リモート・データベースを、Adaptive Server Enterprise または Adaptive Server Anywhere の統合データベースから抽出します。

構文

{ **ssxtract** | **dbxtract** } [*options*] [*directory*] *subscriber*

オプション	説明
@data	設定ファイルからオプションを読み出す。『ASA データベース管理ガイド』> 「@data サーバ・オプション」を参照してください。
-an *database*	アンロードするデータベースと同じ設定でデータベース・ファイルを作成し、そのデータベース・ファイルを自動的に再ロードする。
-ac *"keyword=value; …"*	接続文字列で指定したデータベースに接続して、再ロードする。
-b	サブスクリプションを開始しない。
-c *"keyword=value; …"*	データベース接続パラメータを指定する。
-d	データのみアンロードする。
-e *language,charset*	使用するロケールを指定する。

オプション	説明
-ea *alg*	新しいデータベースで使用する暗号化アルゴリズムを指定する。
-ek *key*	新しいデータベースで使用する暗号化キーを指定する。
-ep	新しいデータベースで使用する暗号化キーの入力を求める。
-f	完全に修飾されたパブリケーションを抽出する。
-ii	内部アンロード、内部再ロード
-ix	内部アンロード、外部再ロード
-j *count*	ビュー作成文の対話回数
-l *level*	指定した独立性レベルですべての抽出オペレーションを実行する。
-k	完了してウィンドウを閉じる。
-n	スキーマ定義のみ抽出する。
-o *file*	メッセージをファイルに出力する。
-p *character*	エスケープ文字
-q	クワイエット・モードで処理を実行し、メッセージ出力またはウィンドウ表示を行わない。このオプションは -y オプションと一緒に指定してください。そうしないと操作は失敗します。
-r *file*	生成された再ロード Interactive SQL コマンド・ファイルの名前を指定する (デフォルトは *reload.sql*)。
-u	データを順序付けしない。
-v	冗長メッセージ
-x	外部テーブル・ロードを使用する。
-xf	外部キーを除外する。
-xi	外部アンロード、内部ロード
-xp	ストアド・プロシージャを除外する。

オプション	説明
-xt	トリガを除外する。
-xv	ビューを除外する。
-xx	外部アンロード、外部ロード
-y	確認メッセージを表示せずにコマンド・ファイルを上書きする。
directory	ファイルが書き込まれたディレクトリ。-an または -ac を使用する場合は不要。
subscriber	データベースを抽出するサブスクライバ

説明

ssxtract は Adaptive Server Enterprise の抽出ユーティリティです。Adaptive Server Enterprise に対して実行し、リモートの Adaptive Server Anywhere データベース用のコマンド・ファイルを作成します。

dbxtract は Adaptive Server Anywhere の抽出ユーティリティです。Adaptive Server Anywhere データベースに対して実行し、リモートの Adaptive Server Anywhere データベース用のコマンド・ファイルを作成します。

リモートの Adaptive Server Enterprise データベースを作成する抽出ユーティリティはありません。

抽出ユーティリティは、コマンド・ファイルと、一連の関連データ・ファイルを作成します。新しく初期化した Adaptive Server Anywhere データベースに対してコマンド・ファイルを実行し、データベース・オブジェクトを作成してリモート・データベース用のデータをロードできます。

デフォルトのコマンド・ファイル名は *reload.sql* です。

リモート・ユーザがグループの場合、グループのメンバのユーザ ID すべてを抽出します。これにより、リモート・データベースの複数のユーザに対して異なるユーザ ID を使用できます。カスタム抽出処理は必要ありません。

SSXtract の注意事項

推奨されなくなった機能
SQL Anywhere Studio の次のメジャー・リリースでは、SQL Remote for Adaptive Server Enterprise はなくなります。Mobile Link が、Adaptive Server Enterprise と Adaptive Server Anywhere データベース間のデータ同期を実現する、より柔軟でスケーラブルなソリューションを提供します。

Adaptive Server Enterprise オブジェクトは、Adaptive Server Anywhere のすべてのオブジェクトには対応していません。*ssxtract* ユーティリティには次の制限事項があります。

- **単一データベース** 抽出されたオブジェクトはすべて、1 つの Adaptive Server Enterprise データベースに格納される。
- **パスワード** 抽出されたユーザ ID のパスワードは、ユーザ ID そのものである。
- **パーミッション** 抽出されたユーザ ID には REMOTE DBA 権限が付与される。
- **名前付き制約** Adaptive Server Anywhere CHECK 制約として抽出される。
- **システム・テーブル** **sp_populate_sql_anywhere** SQL Remote プロシージャは、Adaptive Server Enterprise システム・テーブルから TEMPDB に、一連の Adaptive Server Anywhere システム・テーブルを作成する。抽出されたスキーマは、これらのテンポラリ・システム・テーブルから取得される。

抽出ユーティリティの各オプションの詳細については、「抽出ユーティリティのオプション」394 ページを参照してください。

抽出ユーティリティのオプション

ライト・ファイル・ユーティリティのオプション (@data) 指定された環境変数または設定ファイルからオプションを読み込みます。指定された環境変数と設定ファイルが両方とも存在する場合は、環境変数が使用されます。

設定ファイルの詳細については、『ASA データベース管理ガイド』>「設定ファイルの使用」を参照してください。

設定ファイル内のパスワードおよびその他の情報を保護する場合は、ファイル非表示ユーティリティを使用して、設定ファイルの内容を難読化できます。

詳細については、『ASA データベース管理ガイド』>「dbfhide コマンド・ライン・ユーティリティを使用してファイル内容を隠す」を参照してください。

再ロード用のデータベースを作成する (-an)　このオプションを使用すると、データベースのアンロード、新規データベースの作成、データのロードを組み合わせて実行できます。

たとえば、次のコマンド (1 行で入力) は、新規のデータベース・ファイル *asacopy.db* を作成し、そこに *asademo.db* の field_user サブスクライバのスキーマとデータをコピーします。

```
dbxtract -c "uid=dba;pwd=sql;dbf=asademo.db"
 -an asacopy.db field_user
```

このオプションを使用した場合、データのコピーはディスク上に作成されないため、コマンドでアンロード用ディレクトリを指定する必要はありません。これによりデータのセキュリティは高まりますが、パフォーマンスは多少悪くなります。

既存データベースにデータを再ロードする (-ac)　このオプションを使用すると、データベースのアンロード処理と、既存データベースへの結果の再ロード処理を組み合わせることができます。

たとえば、次のコマンド (1 行で入力) は、field_user サブスクライバのデータのコピーを既存のデータベース・ファイル *newdemo.db* にロードします。

```
dbxtract -c "uid=dba;pwd=sql;dbf=asademo.db"
 -ac "uid=dba;pwd=sql;dbf=newdemo.db" field_user
```

このオプションを使用した場合、データのコピーはディスク上に作成されないため、コマンドでアンロード用ディレクトリを指定する必要はありません。これによりデータのセキュリティは高まりますが、パフォーマンスは多少悪くなります。

サブスクリプションを自動的に開始しない (-b) このオプションを選択した場合は、統合データベース (リモート・データベース用) とリモート・データベース (統合データベース用) で、レプリケーションを開始するために START SUBSCRIPTION 文を使用して、サブスクリプションを明示的に開始しなければいけません。

接続パラメータ (-c) 一連の接続パラメータの文字列です。

- **dbxtract 接続パラメータ** ユーザはデータベースの全テーブル上にパーミッションを持っている必要があるため、**user ID** には DBA 権限を持つユーザ ID を指定してください。

  たとえば、次の文 (1 行で入力) は、ユーザ ID が DBA、パスワードが SQL で接続している **sample_server** サーバ上で実行している *asademo* データベースから、リモート・ユーザ ID **joe_remote** のデータベースを抽出します。データは *c:¥unload* ディレクトリにアンロードされます。

  ```
  ssxtract -c "eng=sample_server;dbn=asademo;
   uid=dba;pwd=sql" c:¥extract joe_remote
  ```

  接続パラメータを指定しない場合、SQLCONNECT 環境変数が設定されていると、SQLCONNECT 環境変数からの接続パラメータを使用します。

- **ssxtract 接続パラメータ** サポートされている接続パラメータは、次のとおりです。

パラメータ	説明
UID	ログイン ID
PWD	パスワード
DBN	データベース名 (オプション)。このパラメータを指定しないと、デフォルトとして、ログイン ID のデフォルト・データベースに接続されます。
ENG	Adaptive Server Enterprise の名前

  ssxtract ではパスワードを抽出できません。パスワードは、ユーザ ID と同一に設定されます。

データのみをアンロードする (-d)　このオプションを選択すると、スキーマ定義はアンロードされません。また、リモート・データベースに対するパブリケーションとサブスクリプションも作成されません。このオプションは、適切なスキーマのあるリモート・データベースがすでに存在していて、データを格納するためだけに使用します。

指定したロケールを使用する (-e)　このオプションは、Adaptive Server Enterprise にのみ適用されます。

Adaptive Server Enterprise のロケール情報を指定します。ロケール文字列のフォーマットは次のとおりです。

```
"language_name,charset_name[,sort_order]"
```

デフォルトでは、Message Agent はデフォルト・ロケールを使用します。このロケールは、ファイル *sybase¥locales¥locales.dat* に定義されています。

language_name と *charset_name* を指定しないと、Message Agent は Adaptive Server Enterprise からこれらの値を取得します。*sort_order* を指定しないと、バイナリ・ソート順 (バイト値の順にソート) が使用されます。

暗号化アルゴリズムを指定する (-ea)　このオプションにより、新しいデータベースの暗号化に強力な暗号化アルゴリズムを選択できます。AES (デフォルト) または AES_FIPS (FIPS 承認のアルゴリズム) のどちらかを選択できます。AES_FIPS は個別のライブラリを使用するため、AES との互換性はありません。アルゴリズム名の大文字と小文字は区別されません。-ea オプションを指定する場合は、-ep または -ek のどちらかのオプションも一緒に指定する必要があります。

詳細については、『SQL Anywhere Studio セキュリティ・ガイド』>「高度な暗号化」を参照してください。

別途ライセンスを取得できるオプションが必要
AES_FIPS を使用する強力なデータベース暗号化では、別途ライセンスの SQL Anywhere Studio セキュリティ・オプションを入手する必要があります。このセキュリティ・オプションは、輸出規制対象品目です。

このコンポーネントの注文方法については、『SQL Anywhere Studio の紹介』>「別途ライセンスが入手可能なコンポーネント」を参照してください。

暗号化キーを指定する (-ek) このオプションを使用すると、コマンドに暗号化キーを直接指定することで、強力に暗号化されたデータベースを作成できます。データベースの暗号化に使用されるアルゴリズムは、-ea オプションで指定した AES または AES_FIPS です。-ek オプションを指定して、-ea オプションを指定しないと、AES アルゴリズムが使用されます。

警告
キーは保護してください。キーのコピーは、安全な場所に保管してください。キーを紛失すると、データベースにまったくアクセスできなくなり、リカバリも不可能になります。

次のキーはデータベース暗号化キーに使用できません。

- 空白スペース、一重引用符、または二重引用符で始まるキー
- 空白スペースで終わるキー
- セミコロンを含むキー

暗号化キーを入力するよう要求する (-ep) このオプションを使用すると、ダイアログ・ボックスに暗号化キーを入力することで、強力に暗号化されたデータベースを作成するように指定できます。クリア・テキストでは暗号化キーを見ることができないようにすることで、高いセキュリティが得られます。

暗号化キーは、正確に入力されたことを確認するために 2 回入力してください。キーが一致しない場合は、初期化は失敗します。

詳細については、『SQL Anywhere Studio セキュリティ・ガイド』>「高度な暗号化」を参照してください。

完全に修飾されたパブリケーションを抽出する (-f) ほとんどの場合、完全に修飾されたパブリケーション定義をリモート・データベース用に抽出する必要はありません。通常、すべてのローはレプリケートされ、統合データベースに戻されます。

しかし、多層の設定や、統合データベースにないローがリモート・データベースにある設定では、完全に修飾されたパブリケーションが必要な場合もあります。

内部アンロード、内部ロード (-ii)　このオプションを使用すると、再ロード・スクリプトは、データのアンロードとロードそれぞれに対して、Interactive SQL の OUTPUT 文と INPUT 文ではなく、内部の UNLOAD 文と LOAD TABLE 文を強制的に使用します。

このオペレーションの組み合わせがデフォルトの動作です。

データ・ファイルのパスには、外部オペレーションでは *dbxtract* の現在の作業ディレクトリからの相対パスを使用し、内部の文ではサーバからの相対パスを使用します。

内部アンロード、外部ロード (-ix)　このオプションを使用すると、再ロード・スクリプトは、データのアンロードに対して内部の UNLOAD 文を強制的に使用し、新しいデータベースへのデータのロードに対して Interactive SQL の INPUT 文を強制的に使用します。

データ・ファイルのパスには、外部オペレーションでは *dbxtract* の現在の作業ディレクトリからの相対パスを使用し、内部の文ではサーバからの相対パスを使用します。

ビューの対話回数 (-j)　統合データベースにネストされたビューがある場合は、このオプションを使用してビューの抽出時の最大対話回数を指定します。

指定された独立性レベルで抽出を実行する (-l)　デフォルト設定では、独立性レベルは 0 です。アクティブなサーバからデータベースを抽出する場合は、独立性レベル 3 (「抽出ユーティリティのオプション」 394 ページを参照) で実行し、抽出されたデータベース内のデータがサーバ上のデータと一致するようにします。独立性レベルを大きくすると、抽出ユーティリティが多数のロックを使用することになり、他のユーザによるデータベースの使用が制限される可能性があります。

スキーマ定義のみアンロードする (-n)　この定義を指定すると、データはアンロードされません。再ロード・ファイルには、データベース構造体だけを構築する SQL 文が記述されています。SYNCHRONIZE SUBSCRIPTION 文を使用すると、メッセージ・システム全体のデー

タをロードできます。パブリケーション、サブスクリプション、PUBLISH パーミッション、SUBSCRIBE パーミッションは、スキーマの一部です。

メッセージをファイルに出力する (-o) 抽出処理時のメッセージを、後で検討できるようにファイルに出力します。

エスケープ文字 (-p) このオプションを使用して、デフォルトのエスケープ文字 (\) を別の文字に置き換えることができます。

クワイエット・モードで処理を実行する (-q) エラー以外のメッセージを表示しません。このオプションは他の環境からは使用できません。コマンド・ライン・ユーティリティからのみ使用できます。

再ロード・ファイル名 (-r) 再ロード・コマンド・ファイルのデフォルト名は、現在のディレクトリの *reload.sql* です。このオプションを使用して異なるファイル名を指定できます。

データを順序付けせずに出力する (-u) デフォルトでは、各テーブルのデータはプライマリ・キーを基準に順序付けられます。-u オプションを使用するとアンロード処理は高速になりますが、リモート・データベースへのデータのロード処理は遅くなります。

冗長モード (-v) アンロードされているテーブル名とアンロードされたロー数が表示されます。使用された SELECT 文も表示されます。

外部キー定義を除外する (-xf) リモート・データベースに統合データベース・スキーマのサブセットがあり、いくつかの外部キー参照がない場合に、このオプションを使用できます。

外部アンロード、内部ロード (-xi) データベースのアンロードでは、デフォルトの動作として UNLOAD 文を使用します。これはデータベース・サーバが実行します。外部アンロードを選択すると、*dbxtract* は UNLOAD 文の代わりに OUTPUT 文を使用します。OUTPUT 文はクライアントで実行されます。

データ・ファイルのパスには、外部オペレーションでは *dbxtract* の現在の作業ディレクトリからの相対パスを使用し、内部の文ではサーバからの相対パスを使用します。

ストアド・プロシージャを除外する (-xp) データベースからストアド・プロシージャを抽出しません。

トリガを除外する (-xt)　データベースからトリガを抽出しません。

ビューを除外する (-xv)　データベースからビューを抽出しません。

外部アンロード、外部ロード (-xx)　データをアンロードする場合には、OUTPUT 文が使用されます。また、新規データベースにデータをロードする場合には、INPUT 文が使用されます。

アンロードのデフォルト動作では UNLOAD 文が使用され、ロードのデフォルト動作では LOAD TABLE 文が使用されます。内部の UNLOAD 文と LOAD TABLE 文を使用すると、OUTPUT 文と INPUT 文よりも高速で処理します。

データ・ファイルのパスには、外部オペレーションでは *dbxtract* の現在の作業ディレクトリからの相対パスを使用し、内部の文ではサーバからの相対パスを使用します。

確認メッセージを表示することなく処理を実行する (-y)　このオプションを指定しないと、既存のコマンド・ファイルを置き換えるときに、確認メッセージが表示されます。

SQL Remote Open Server

内容

Replication Server からレプリケーション・データを取得し、それを SQL Remote ステーブル・キューに適用します。このユーティリティが必要なのは、Replication Server (Replication Agent を使用) および SQL Remote レプリケーションの両方に関係するデータベースだけです。

推奨されなくなった機能

SQL Anywhere Studio の次のメジャー・リリースでは、SQL Remote for Adaptive Server Enterprise はなくなります。Mobile Link が、Adaptive Server Enterprise と Adaptive Server Anywhere データベース間のデータ同期を実現する、より柔軟でスケーラブルなソリューションを提供します。

構文

ssqueue [*options*] [*open-server-name*]

オプション

オプション	説明
open-server-name	オープン・サーバ名。インターフェイス・ファイルで宣言する。
-c *"keyword=value; …"*	データベース接続パラメータを指定する。
-cq *"keyword=value; …"*	ステーブル・キューのデータベース接続パラメータを指定する。
-dl	メッセージをウィンドウに表示する。
-k	完了してウィンドウを閉じる。
-o *file*	メッセージをファイルに出力する。
-os *file*	出力メッセージを記録するファイルの最大サイズ
-ot *file*	ファイルをトランケートし、出力メッセージを記録する。
-q	最小化ウィンドウで実行する。
-ud	デーモンとして実行する (UNIX)。

オプション	説明
-v	冗長オペレーション

説明

SQL Remote Open Server を使用すると、Adaptive Server Enterprise データベースが SQL Remote レプリケーションに参加でき、一方で Replication Server のインストール環境でのプライマリ・サイト (または非同期プロシージャ・コールを使用するレプリケート・サイト) として動作します。

実行プログラム名は次のとおりです。

- **ssqueue.exe**　Windows オペレーティング・システム
- **ssqueue**　UNIX オペレーティング・システム

オプションの詳細

open-server-name　Replication Server は SQL Remote Open Server に接続する必要があるため、オープン・サーバ名が必要になります。このオープン・サーバ名はコマンド・プロンプトで設定し、SQL Remote Open Server を実行するマシンのインターフェイス・ファイル内のマスタ・エントリとクエリ・エントリ、および Replication Server を実行するマシンのインターフェイス・ファイルにあるクエリ・エントリに対応させます。

インターフェイス・ファイルの名前は、Windows オペレーティング・システム上では *sql.ini*、UNIX 上では *interfaces* です。

オープン・サーバ名のデフォルト値は **SSQueue** です。

-c　レプリケートされるデータが格納されているデータベースへの接続パラメータを指定します。この接続は、SQL Remote Open Server が SQL Remote システム・テーブルへアクセスするために必要です。

接続パラメータは、次のリストにあるいずれかを使用します。

パラメータ	説明
UID	ログイン ID
PWD	パスワード

パラメータ	説明
DBN	データベース名 (オプション)。このパラメータを指定しないと、デフォルトとして、ログイン ID のデフォルト・データベースに接続されます。
ENG	サーバ名

-cq ステーブル・キューの接続パラメータを指定します。指定しないと、デフォルトとして -c の値が使用されます。

-dl ウィンドウまたはコマンド・プロンプトにメッセージを表示します。ログ・ファイルにも出力します。

-k 完了後、ウィンドウを閉じます。

-o 出力をログ・ファイルに追加します。デフォルトでは、画面に出力が表示されます。

-os 出力メッセージをロギングするファイルの最大サイズを指定します。使用できるサイズは、*n* (バイト)、*n***K** (Kb)、または *n***M** (Mb) で指定します。デフォルトによる制限はなく、最小制限は 10000 バイトです。

SQL Remote では、現在のファイル・サイズをチェックしてから、出力メッセージをファイルに記録します。ログ・メッセージによって、ファイル・サイズが指定したサイズより大きくなると、SQL Remote は出力ファイルの名前をそれぞれ *yymmddxx.dbr* (dbremote の場合) と *yymmddxx.ssr* (ssremote の場合) に変更します。xx は AA 〜 ZZ のアルファベット順の英字で、*yymmdd* は現在の年月日を表します。

Message Agent を継続モードで長時間実行する場合、このオプションを使用して、手動で古いログ・ファイルを削除し、ディスク領域を解放できます。

-ot ログ・ファイルをトランケートし、このファイルに出力メッセージを追加します。デフォルトでは、画面に出力が表示されます。

-q ウィンドウ操作のオペレーティング・システムの場合にのみ、Message Agent を最小化ウィンドウで起動します。

-ud UNIX プラットフォーム上では、-ud オプションを指定して、SQL Remote Open Server をデーモンとして実行できます。

デーモンとして実行する場合は、-o または -ot オプションも指定して、出力情報のログを取ってください。

-v　冗長出力です。このオプションによって、メッセージに含まれる SQL 文が画面に表示されます。-o オプションを指定するとログ・ファイルに出力されます。

SQL Remote のオプション

機能

レプリケーション・オプションは、レプリケーション動作を制御するためのデータベース・オプションです。

Adaptive Server Anywhere 構文

SET [**TEMPORARY**] **OPTION**
[*userid*. | **PUBLIC**.]*option_name* = [*option_value*]

Adaptive Server Enterprise 構文

exec sp_remote_option *option-name*, *option-value*

パラメータ

引数	説明
option_name	変更されるオプションの名前
option-value	オプションの設定が含まれる文字列

説明

次のオプションを使用できます。

オプション	値	デフォルト
Blob_threshold	**整数 (kb)**	**256**
Compression	**-1 ～ 9**	**6**
Delete_old_logs	**ON**、**OFF**	**OFF**
External_remote_options	**ON**、**OFF**	**OFF**
Qualify_owners	**ON**、**OFF**	**ON**
Quote_all_identifiers	**ON**、**OFF**	**OFF**
Replication_error	*プロシージャ名*	**NULL**
Save_remote_passwords	**ON**、**OFF**	**ON**
SR_Date_Format	*日付文字列*	**yyyy/mm/dd**
SR_Time_Format	*時刻文字列*	**hh:nn:ss.Ssssss**

オプション	値	デフォルト
SR_Timestamp_Format	*タイムスタンプ文字列*	**yyyy/mm/dd hh:nn:ss.Ssssss**
Subscribe_by_remote	**ON、OFF**	**ON**
Verify_threshold	*整数*	**256**
Verify_all_columns	**ON、OFF**	**OFF**

これらのオプションは Message Agent が使用します。Message Agent のコマンドで指定されたユーザ ID 用に設定してください。一般的な public の使用にも設定できます。

次にオプションの説明を示します。

Blob_threshold オプション　このオプションより長い値は、blob としてレプリケートされます。つまり、細かく分割して各部分をレプリケートし、SQL 変数を使用して、受信者サイトで分割された部分を連結して再構成します。

Adaptive Server Enterprise のインストール環境で blob をレプリケートする場合、レプリケートされる最大の blob より大きい値が Blob_threshold に設定されていることを確認してください。

blob のレプリケーションと Adaptive Server Enterprise の詳細については、「blob (Binary Large OBject) のレプリケーション」103 ページを参照してください。

Compression オプション　メッセージの圧縮レベルを設定します。-1 ～ 9 の値を指定します。各値の意味は次のとおりです。

- **-1**　メッセージをバージョン 5 フォーマットで送信します。SQL Remote の以前のバージョンの Message Agents (*dbremote* と *ssremote* の両方) では、バージョン 6 フォーマットで送信されたメッセージを読み込めません。使用しているシステムのすべての Message Agent をバージョン 6 にアップグレードするまでは、COMPRESSION を -1 に設定してください。
- **0**　圧縮しません。

- **1 ～ 9** 圧縮率を高めます。大きい圧縮率でメッセージを作成すると、小さい圧縮率を使用した場合より時間がかかります。

Delete_old_logs オプション このオプションは、SQL Remote と Adaptive Server Anywhere Replication Agent で使用します。デフォルト設定は OFF です。ON に設定すると、トランザクション・ログに含まれているすべての変更が送信され、受信が確認されたときに、Message Agent (DBREMOTE) はそれぞれの古いトランザクション・ログを削除します。

External_remote_options このオプションは SQL Remote が、メッセージ・リンク・パラメータをデータベースに格納するか (OFF)、外部に格納するか (ON) を指定するために使用します。デフォルトでは、OFF に設定されています。

Qualify_owners オプション SQL Remote によってレプリケートされる SQL 文で、修飾されたオブジェクト名を使用するかどうかを指定します。デフォルト設定は、Adaptive Server Anywhere では ON で、Adaptive Server Enterprise では OFF です。

Adaptive Server Enterprise 設定で所有者を修飾する必要はありません。一般的に、オブジェクトは **dbo** によって所有されているためです。Adaptive Server Anywhere 設定で修飾が必要ない場合、このオプションを OFF にした場合よりもメッセージは少し小さくなります。

Quote_all_identifiers オプション SQL Remote によってレプリケートされる SQL 文で、引用符で囲んだ識別子を使用するかどうかを指定します。デフォルトは OFF です。

このオプションが OFF の場合、*dbremote* は Adaptive Server Anywhere による引用符が必要な識別子を (通常行われているように) 引用符で囲み、*ssremote* は識別子を引用符で囲みません。このオプションが ON の場合、すべての識別子が引用符で囲まれます。

Replication_error オプション SQL エラーが発生したときに、Message Agent が呼び出すストアド・プロシージャを指定します。デフォルトではプロシージャを呼び出しません。

レプリケーション・エラー・プロシージャでは、データ型が CHAR、VARCHAR、または LONG VARCHAR の引数を 1 つ指定してください。プロシージャは、SQL エラー・メッセージで 1 回、エラーの原因となった SQL 文でもう 1 回呼び出されます。

このオプションを使用すると、レプリケーションでの SQL エラーの追跡とモニタができますが、その場合でもやはり、エラーが起こらないように設計することが必要です。このオプションは、そのようなエラーの解決を意図したものではありません。

DEFAULT CURRENT REMOTE USER のテーブルを使用して、エラーが発生したリモート・サイトを記録できます。

Save_remote_passwords オプション　最初の接続時にメッセージ・リンクのダイアログ・ボックスにパスワードを入力すると、そのパラメータ値が保存されます。デフォルトでは、Save_remote_passwords は ON に設定され、パスワードが保存されます。メッセージ・リンク・パラメータをデータベースではなく外部に保存する場合、パスワードが保存されないように設定できます。このオプションを NO に設定すると、パスワードは保存されません。

SR_Date_Format オプション　日付を保存するカラムをレプリケートするときに Message Agent が使用します。このオプションは、次の記号で構成される文字列です。

記号	説明
yy	2 桁の年
yyyy	4 桁の年
mm	2 桁の月
mmm	月を表す文字フォーマット
dd	2 桁の日

各記号は、レプリケートされる日付に置き換えられます。

フォーマット記号 **mm** を大文字で設定すると、対応する文字も大文字になります。

数字フォーマットの場合、このオプション設定が大文字か小文字かによって埋め込みが制御されます。記号を DD のように大文字か小文字にそろえると、0 (ゼロ) で埋められます。Mm のように大文字と小文字を混在させると、0 (ゼロ) にはなりません。

SR_Time_Format オプション 時間を保存するカラムをレプリケートするときに Message Agent が使用します。このオプションは、次の記号で構成される文字列です。

記号	説明
hh	2 桁の時間 (24 時間表示)
nn	2 桁の分
mm	コロンの後の場合は、2 桁の分 (hh:mm など)
ss[.s…]	2 桁の秒とオプションの小数

フォーマット文字列で大文字と小文字を混在させて使用すると、数字の前にゼロが付きません。

SR_Timestamp_Format このオプションは、Message Agent で日付情報をレプリケートするときに使用します。Adaptive Server Anywhere では、timestamp、datetime、smalldatetime のデータ型があります。Adaptive Server Enterprise では、datetime と smalldatetime のデータ型があります。

フォーマット文字列は SR_Date_Format と SR_Time_Format の設定から取得します。

デフォルト設定は、SR_Date_Format 設定、次に SR_Time_Format 設定の順番です。

Subscribe_by_remote オプション ON に設定すると、NULL または空の文字列である subscribe by 値のローに対してリモート・データベースから操作が行われた場合、リモート・ユーザがローにサブスクリプションを作成したとみなします。OFF に設定すると、リモート・ユーザはローにサブスクリプションを作成していないとみなします。

このオプションでの唯一の制約は、リモート・ユーザが NULL 値のローまたは空のサブスクリプション式 (情報が統合データベースにだけ格納されているもの) を、実際に INSERT (または UPDATE) しようとした場合にエラーが発生することです。この問題はそれほど深刻ではなく、インストール環境で、リモート・ユーザに属さないサブスクリプション値を割り当てることで対処できます。

このオプションの詳細については、「多対多の関係における Subscribe_by_remote オプションの使用」149 ページと「多対多の関係における Subscribe_by_remote オプションの使用」208 ページを参照してください。

Verify_threshold オプション　カラムのデータ型が閾値より長い場合、カラムの古い値は UPDATE がレプリケートされるときに確認されません。デフォルト設定値は 1000 です。

このオプションで、SQL Remote メッセージのサイズが大きくならないようにしますが、長い値の重複する更新が検出されないという短所もあります。

Verify_all_columns オプション　デフォルト設定は OFF です。ON に設定すると、ローカル・データベースがパブリッシュした更新を含むメッセージはすべてのカラム値を含んで送信され、カラムの重複はサブスクライバ・データベースで RESOLVE UPDATE トリガを起動します。

Adaptive Server Enterprise の抽出ユーティリティを使用して Adaptive Server Anywhere リモート・データベースにパブリック・オプションを設定し、Adaptive Server Enterprise データベースの設定と一致させます。

例

- 次の文は、Adaptive Server Anywhere で、すべてのユーザに対して Verify_all_columns オプションを OFF に設定します。

```
SET OPTION PUBLIC.Verify_all_columns = 'OFF'
```

- 次の文は、Adaptive Server Enterprise で、Verify_all_columns オプションを OFF に設定します。

```
exec sp_remote_option Verify_all_columns, 'OFF'
 go
```

Adaptive Server Enterprise では、レプリケーション・オプションは SQL Remote でのみ使用します。

SQL Remote イベント・フック・プロシージャ

以下のストアド・プロシージャ名と引数は、SQL Remote データベースで同期をカスタマイズするためのインタフェースを提供します。

注意

特に明記しないかぎり、イベント・フック・プロシージャには次の条件が適用されます。

- ストアド・プロシージャには、DBA 権限 (Adaptive Server Anywhere の場合) または dbo 権限 (Adaptive Server Enterprise の場合) が必要です。
- プロシージャでは、オペレーションのコミットとロールバックはできません。また、暗黙的なコミットを実行するアクションも実行できません。プロシージャのアクションは、呼び出し側のアプリケーションによって自動的にコミットされます。
- Message Agent の冗長モードをオンにすると、フックのトラブルシューティングができます。
- *dbremote* と *ssremote* の場合、フックの違いは名前だけです。

#Hook_dict テーブル

#hook_dict テーブルは、次の CREATE 文を使用して、フックが呼び出される直前に作成されます。

```
CREATE table #hook_dict(
 name VARCHAR(128) NOT NULL UNIQUE,
 value VARCHAR(255) NOT NULL )
```

Message Agent は #hook_dict テーブルを使用して、値をフック関数に渡します。フック関数は #hook_dict テーブルを使用して、値を Message Agent に返します。

sp_hook_dbremote_begin と sp_hook_ssrmt_begin

機能

このストアド・プロシージャを使用すると、レプリケーション処理の開始時にカスタム・アクションを追加できます。

#hook_dict テーブルのロー

名前	値の範囲	説明
send	**true** または **false**	処理がレプリケーションの送信フェーズを実行中かどうかを示す
receive	**true** または **false**	処理がレプリケーションの受信フェーズを実行しているかどうかを示す

説明

この名前のプロシージャが存在する場合、プロシージャは、Message Agent が起動するときに呼び出されます。

sp_hook_dbremote_end と sp_hook_ssrmt_end

機能

このストアド・プロシージャを使用すると、Message Agent の終了直前にカスタム・アクションを追加できます。

#hook_dict テーブルのロー

名前	値の範囲	説明
send	**true** または **false**	処理がレプリケーションの送信フェーズを実行中かどうかを示す
receive	**true** または **false**	処理がレプリケーションの受信フェーズを実行しているかどうかを示す
exit code	**integer**	0 以外の終了コードはエラーを示す

説明

この名前のプロシージャが存在する場合、プロシージャは、Message Agent が停止する前の最後のイベントとして呼び出されます。

sp_hook_dbremote_shutdown と sp_hook_ssrmt_shutdown

機能

このストアド・プロシージャを使用すると、Message Agent のシャットダウンを開始できます。

#hook_dict テーブルのロー

名前	値の範囲	説明
send	**true** または **false**	処理がレプリケーションの送信フェーズを実行中かどうかを示す
receive	**true** または **false**	処理がレプリケーションの受信フェーズを実行しているかどうかを示す
shutdown	**true** または **false**	プロシージャが呼び出されるときにはこのローは **false** です。プロシージャがこのローを **true** に更新すると、Message Agent がシャットダウンされます。

説明　この名前のプロシージャが存在する場合、プロシージャは、Message Agent がメッセージを送受信していないときに呼び出され、Message Agent のフックで開始されるシャットダウンを可能にします。

sp_hook_dbremote_receive_begin と sp_hook_ssrmt_receive_begin

機能　このストアド・プロシージャを使用すると、レプリケーションの受信フェーズの開始前にアクションを実行できます。

#hook_dict テーブルのロー　なし

sp_hook_dbremote_receive_end と sp_hook_ssrmt_receive_end

機能　このストアド・プロシージャを使用すると、レプリケーションの受信フェーズの終了後にアクションを実行できます。

#hook_dict テーブルのロー　なし

sp_hook_dbremote_send_begin と sp_hook_ssrmt_send_begin

機能　このストアド・プロシージャを使用すると、レプリケーションの送信フェーズの開始前にアクションを実行できます。

#hook_dict テーブルのロー なし

sp_hook_dbremote_send_end と sp_hook_ssrmt_send_end

機能 このストアド・プロシージャを使用すると、レプリケーションの送信フェーズの終了後にアクションを実行できます。

#hook_dict テーブルのロー なし

sp_hook_dbremote_message_sent と sp_hook_ssrmt_message_sent

機能 このストアド・プロシージャを使用すると、任意のメッセージが送信された後にアクションを実行できます。

#hook_dict テーブルのロー

名前	値の範囲
remote user	メッセージの送信先

sp_hook_dbremote_message_missing と sp_hook_ssrmt_message_missing

機能 このストアド・プロシージャを使用すると、Message Agent がリモート・ユーザからの 1 つ以上のメッセージが見つからないと判断した場合にアクションを実行できます。

#hook_dict テーブルのロー

名前	値の範囲
remote user	メッセージを再送する必要があるリモート・ユーザの名前

sp_hook_dbremote_message_apply_begin と sp_hook_ssrmt_message_apply_begin

機能

このストアド・プロシージャを使用すると、Message Agent がユーザからの一連のメッセージを適用する直前にアクションを実行できます。

#hook_dict テーブルのロー

名前	値の範囲
remote user	適用されるメッセージを送信したリモート・ユーザの名前

sp_hook_dbremote_message_apply_end と sp_hook_ssrmt_message_apply_end

機能

このストアド・プロシージャを使用すると、Message Agent がユーザからの一連のメッセージを適用した直後にアクションを実行できます。

#hook_dict テーブルのロー

名前	値の範囲
remote user	適用されたメッセージを送信したリモート・ユーザの名前

第 15 章

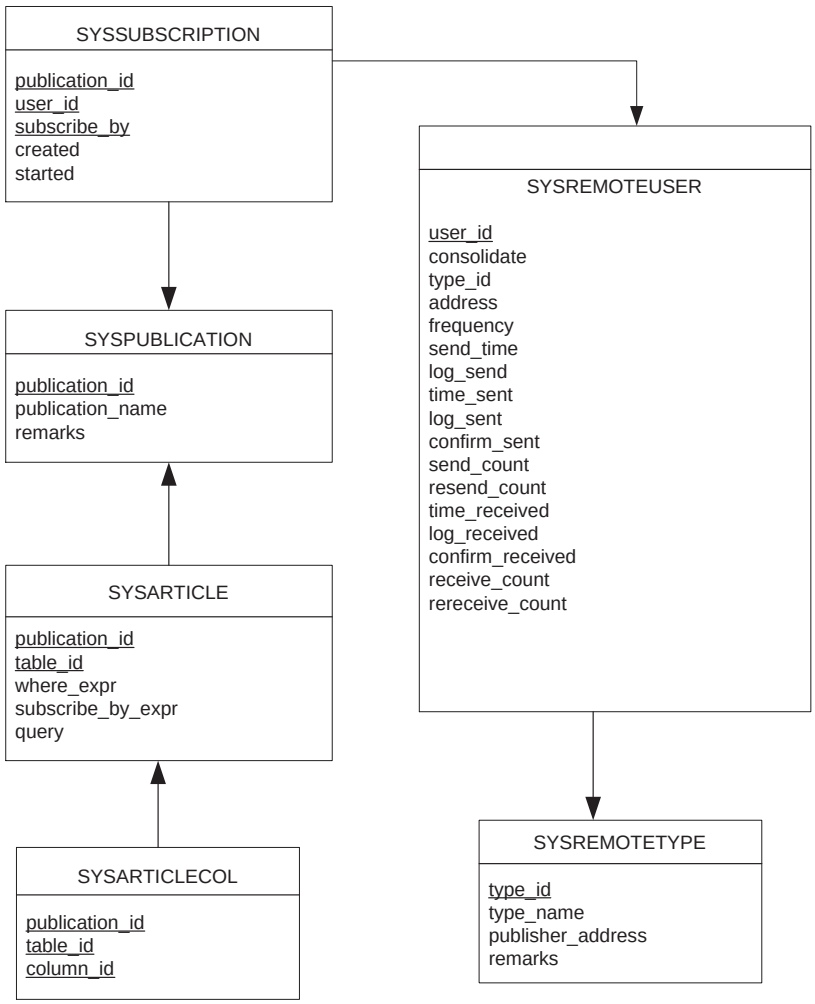
Adaptive Server Anywhere のシステム・オブジェクト

この章の内容

SQL Remote 固有のシステム情報は、Adaptive Server Anywhere カタログに保持されます。この情報をより分かりやすい形にしたものが、一連のシステム・ビューに保持されます。

SQL Remote システム・テーブル

この項では、SQL Remote 情報の定義および管理のために SQL Remote が使用するシステム・テーブルについて説明します。次の図では、矢印はテーブル間の外部キー関係を示します。矢印は、外部テーブルからプライマリ・テーブルへ示されています。



これらのテーブルについては、次の項で詳しく説明します。

SYSARTICLE テーブル

機能

SQL Remote パブリケーション内の 1 アーティクルについて、各ローで示します。

カラム

カラム	データ型	説明
publication_id	UNSIGNED INT	このアーティクルが含まれるパブリケーション
table_id	UNSIGNED INT	各アーティクルは、単一のテーブルのカラムとローから構成されます。このカラムには、このテーブルの ID が含まれます。
where_expr	LONG VARCHAR	WHERE 句で定義されたローのサブセットを含むアーティクルの場合、このカラムに探索条件が含まれます。
subscribe_by_expr	LONG VARCHAR	SUBSCRIBE BY 式で定義されたローのサブセットを含むアーティクルの場合、このカラムに式が含まれます。
query	CHAR(1)	SUBSCRIBE BY 式を、複数の値を返すサブクエリとすることができます。このカラムには、その式がサブクエリであるか (Y) そうでないか (N) を示す値、Y または N が含まれます。このカラムは、抽出ユーティリティによって使用されます。

SYSARTICLECOL テーブル

機能

各ローで、アーティクル内のカラムを 1 つ特定します。それには、カラム、カラムが属するテーブル、カラムを含むパブリケーションから特定します。

カラム

カラム	データ型	説明
publication_id	UNSIGNED INT	カラムを含むパブリケーションのユニークな識別子
table_id	UNSIGNED INT	カラムが属するテーブル
column_id	UNSIGNED INT	SYSCOLUMN システム・テーブルのカラム識別子

SYSPUBLICATION テーブル

機能 SQL Remote パブリケーションについて、各ローに示します。

カラム

カラム	データ型	説明
publication_id	UNSIGNED INT	パブリケーションのユニークな識別子
creator	UNSIGNED INT	パブリケーションを所有するユーザ ID
publication_name	VARCHAR(128)	パブリケーションの名前
remarks	LONG VARCHAR	コメント

SYSREMOTEOPTION テーブル

機能 SQL Remote のメッセージ・リンク・パラメータの値について、各ローに示します。

カラム

カラム	データ型	説明
option_id	UNSIGNED INT	メッセージ・リンク・パラメータの ID 番号

カラム	データ型	説明
user_id	UNSIGNED INT	パラメータが設定されているユーザ ID
"setting00"	VARCHAR(255)	メッセージ・リンク・パラメータの値

SYSREMOTEOPTIONTYPE テーブル

機能

SQL Remote のメッセージ・リンク・パラメータについて、各ローで示します。

カラム

カラム	データ型	説明
option_id	UNSIGNED INT	メッセージ・リンク・パラメータの ID 番号
type_id	SMALLINT	このパラメータを使用するメッセージ・タイプの ID 番号
"option"	VARCHAR(128)	メッセージ・リンク・パラメータの名前

SYSREMOTETYPE テーブル

機能

SQL Remote のメッセージ・タイプについて 1 つずつ、パブリッシャ・アドレスを含めて、各ローで示します。

カラム

カラム	データ型	説明
type_id	SMALLINT	メッセージ・タイプの ID 番号

カラム	データ型	説明
type_name	VARCHAR(128)	メッセージ・タイプ。以下のタイプごとに別のローが存在します。 • FILE • MAPI • VIM • SMTP
publisher_address	VARCHAR(128)	メッセージ・タイプ **type_name** 用のパブリッシャ・アドレス。SQL Remote は、このアドレスからメッセージを受信します。
remarks	LONG VARCHAR	コメント

SYSREMOTEUSER テーブル

機能

REMOTE パーミッションを持つユーザ ID (サブスクライバ) について、各ローで示します。そのユーザが送受信した SQL Remote メッセージのステータスについても、合わせて示します。

カラム

カラム	データ型	説明
user_id	UNSIGNED INT	REMOTE パーミッションを持つユーザの ID
consolidate	CHAR(1)	このカラムには、REMOTE パーミッションを付与されたユーザを示す N か、CONSOLIDATE パーミッションを付与されたユーザを示す Y のいずれかが含まれます。
type_id	SMALLINT	このユーザへメッセージを送信する場合に使用される、メッセージ・システムの ID

カラム	データ型	説明
address	LONG VARCHAR	SQL Remote メッセージが送信されるアドレス。アドレスは **address_type** に対して適切でなければいけません。
frequency	CHAR(1)	SQL Remote メッセージが送信される頻度。P は定期的、A は随時を表します。
send_time	TIME	次にメッセージがこのユーザへ送信される時刻
log_send	NUMERIC(20, 0)	**log_send** が **log_sent** より大きい場合、Message Agent は次に実行されたときにサブスクライバへすぐにメッセージを再送信します。
time_sent	TIMESTAMP	このサブスクライバに、最後にメッセージが送信された時刻
log_sent	NUMERIC(20, 0)	このサブスクライバに最後に送信されたオペレーションに対する、ローカルのログ・オフセット
confirm_sent	NUMERIC(20, 0)	このサブスクライバから最後に確認されたオペレーションに対する、ログ・オフセット
send_count	INT	このサブスクライバへ送信された SQL Remote メッセージの数
resend_count	INT	サブスクライバ・データベースで、メッセージが一度だけ適用されたことを確認するためのカウンタ
time_received	DATETIME	最後にこのサブスクライバからメッセージを受信した時刻
log_received	NUMERIC(20, 0)	現在のデータベースで最後に受信されたオペレーションに対する、サブスクライバ・データベース内のログ・オフセット

カラム	データ型	説明
confirm_received	NUMERIC(20, 0)	確認メッセージが送信された最後のオペレーションに対する、サブスクライバ・データベース内のログ・オフセット
receive_count	INT	このサブスクライバから受信したメッセージの数
rereceive_count	INT	現在のデータベースで、メッセージが一度だけ適用されたことを確認するためのカウンタ

SYSSUBSCRIPTION テーブル

機能

REMOTE パーミッションを持つあるユーザ ID からの、あるパブリケーションに対するサブスクリプションについて、各ローで示します。

カラム

カラム	データ型	説明
publication_id	UNSIGNED INT	ユーザ ID のサブスクリプションが作成されているパブリケーションの識別子
user_id	UNSIGNED INT	パブリケーションに対するサブスクリプションが作成されているユーザ ID
subscribe_by	VARCHAR(128)	SUBSCRIBE BY 式を使用するパブリケーションの場合、このカラムにはこのサブスクリプションと一致する値が保持されます。
created	NUMERIC(20, 0)	トランザクション・ログ内の、サブスクリプションが作成されたオフセット
started	NUMERIC(20, 0)	トランザクション・ログ内の、サブスクリプションが開始されたオフセット

SQL Remote システム・ビュー

この項では、SQL Remote 情報を表示および要約するために、SQL Remote が使用するデータベース・ビューについて説明します。

SYSARTICLES ビュー

機能　各ローのリストにアーティクルを 1 つ示します。

カラム

カラム	説明
publication_name	このアーティクルが含まれるパブリケーション
table_name	各アーティクルは、単一のテーブルのカラムとローから構成されます。このカラムには、このテーブルの名前が含まれます。
where_expr	WHERE 句で定義されたローのサブセットを含むアーティクルの場合、このカラムに探索条件が含まれます。
subscribe_by_expr	SUBSCRIBE BY 式で定義されたローのサブセットを含むアーティクルの場合、このカラムに式が含まれます。

SYSARTICLECOLS ビュー

機能　あるアーティクル内の 1 カラムについて、各ローで示します。

カラム

カラム	説明
publication_name	カラムが含まれるパブリケーションの名前
table_name	カラムが属するテーブルの名前
column_name	カラムの名前

SYSPUBLICATIONS ビュー

機能 すべてのパブリケーションの名前をリストします。

カラム

カラム	説明
publication_name	パブリケーションの名前
creator	パブリケーションの所有者
remarks	コメント

SYSREMOTEOPTIONS ビュー

機能 システム・テーブル SYSREMOTEOPTION と SYSREMOTEOPTIONTYPE に格納されている SQL Remote のメッセージ・リンク・パラメータとその値を、より読みやすい形でリストします。

カラム

カラム	説明
type_name	メッセージ・リンク・タイプ
"option"	オプションの名前
setting	オプションの値

SYSREMOTEUSERS ビュー

機能 リモート・ユーザとそのステータスについての情報をリストします。

カラム

カラム	説明
user_name	REMOTE パーミッションを持つユーザの ID

カラム	説明
consolidate	このカラムには、REMOTE パーミッションを持つユーザを示す N か、CONSOLIDATE パーミッションを持つユーザを示す Y のいずれかが含まれます。
type_name	このユーザへメッセージを送信する場合に使用されるメッセージ・タイプの名前
address	SQL Remote メッセージが送信されるアドレス。アドレスは **address_type** に対して適切でなければいけません。
frequency	SQL Remote メッセージが送信される頻度
send_time	次にメッセージがこのユーザへ送信される時刻
next_send	次にメッセージがこのユーザへ送信される時刻を、より分かりやすい形式で表したもの
log_send	メッセージは、**log_send** が **log_sent** よりも大きいサブスクライバだけに送信されます。
time_sent	このサブスクライバに、最後にメッセージが送信された時刻
log_sent	最後に送信されたオペレーションに対する、トランザクション・ログ・オフセット
confirm_sent	このサブスクライバから最後に確認されたオペレーションに対する、トランザクション・ログ・オフセット
send_count	SQL Remote メッセージが送信された回数
resend_count	サブスクライバ・データベースで、メッセージが一度だけ適用されたことを確認するためのカウンタ
time_received	最後にこのサブスクライバからメッセージを受信した時刻
log_received	現在のデータベースで最後に受信されたオペレーションに対する、サブスクライバ・データベース内のログ・オフセット
confirm_received	確認メッセージが送信された最後のオペレーションに対する、サブスクライバ・データベース内のログ・オフセット

カラム	説明
receive_count	メッセージが受信された回数
rereceive_count	現在のデータベースで、メッセージが一度だけ適用されたことを確認するためのカウンタ

SYSSUBSCRIPTIONS ビュー

機能

サブスクリプションについての情報を、各ローでリストします。

カラム

カラム	説明
publication_name	ユーザ ID のサブスクリプションが作成されているパブリケーションの名前
user_name	パブリケーションに対するサブスクリプションが作成されているユーザ ID
subscribe_by	SUBSCRIBE BY 式を使用するパブリケーションの場合、このカラムにはこのサブスクリプションと一致する値が保持されます。
created	トランザクション・ログ内の、サブスクリプションが作成されたオフセット
started	トランザクション・ログ内の、サブスクリプションが開始されたオフセット

第 16 章

Adaptive Server Enterprise のシステム・オブジェクト

この章の内容

SQL Remote 固有のシステム情報は、SQL Remote システム・テーブルという一連のテーブルに保存されます。この情報をより分かりやすい形式にしたものが、SQL Remote システム・ビューという一連のビューに保持されます。

SQL Remote システム・テーブル

この項では、SQL Remote 情報の定義および管理のために SQL Remote が使用するデータベース・テーブルについて説明します。

警告
これらのテーブルは SQL Remote 専用です。これらのテーブルやその内容を直接変更しないでください。

#remote テーブル

機能

このテンポラリ・テーブルは、現在のリモート・ユーザの名前と現在のパブリッシャの名前を保持するために Message Agent によって作成されます。このテーブルは、Adaptive Server Enterprise にだけ存在します。

カラム

カラム	データ型	説明
current_remote_user	VARCHAR(128)	現在のリモート・ユーザ (Message Agent のコマンド・ラインから)
current_publisher	VARCHAR(128)	現在のパブリッシャ

説明

これはシステム・テーブルではありません。Message Agent for Adaptive Server Enterprise はサーバに接続するときに、現在のリモート・ユーザ ID の値と現在のパブリッシャの値を **#remote** テーブルに保存します。このテンポラリ・テーブルは、TEMPDB に格納されます。

#remote テーブルの値は、競合解決のプロシージャで使用します。

このテーブルの CREATE TABLE 文は次のとおりです。

```
CREATE TABLE #remote (
     current_remote_user varchar(128),
     current_publisher varchar(128)
 )
```

このテーブルのローは 1 つです。

sr_article テーブル

機能

SQL Remote パブリケーション内の 1 アーティクルについて、各ローで示します。

カラム

カラム	データ型	説明
publication_id	INT	このアーティクルが含まれるパブリケーション
table_id	INT	各アーティクルは、単一のテーブルのカラムとローから構成されます。このカラムには、このテーブルの ID が含まれます。
where_expr	VARCHAR(128)	WHERE 句で定義されたローのサブセットを含むアーティクルの場合、このカラムに探索条件が含まれます。
subscribe_by_expr	VARCHAR(128)	SUBSCRIBE BY 式で定義されたローのサブセットを含むアーティクルの場合、このカラムに式が含まれます。
subscribe_by_view	VARCHAR(128)	ビューで定義されたローのサブセットを含むアーティクルの場合、このカラムにビューの名前が含まれます。

sr_articlecol テーブル

機能

各ローで、アーティクル内のカラムを 1 つ特定します。それには、カラム、カラムが属するテーブル、カラムを含むパブリケーションから特定します。

カラム

カラム	データ型	説明
publication_id	INT	カラムが含まれるパブリケーションのユニークな識別子
table_id	INT	カラムが属するテーブル
column_id	INT	SYSCOLUMN システム・テーブルのカラム識別子

sr_marker テーブル

機能

Message Agent が受信したメッセージを同じセッション内のリモート・データベースにも送信したかを確認するためのテーブルです。

カラム

カラム	データ型	説明
marker	DATETIME	最新のメッセージが適用された時刻を示す値

説明

統合データベースで 2 つの Message Agent (ステーブル・キューへのデータの格納 (`-I`) 用とメッセージの送受信 (`-r -s`) 用を使用する場合は、**sr_marker** テーブルの 1 つのローを使用して統合データベースで受信し適用したメッセージが送信されたことをマークしてから、Message Agent を終了します。

sr_object テーブル

機能

SQL Remote オブジェクトのリストを保持します。抽出ユーティリティが SQL Remote システム・オブジェクトを抽出しないようにする必要があります。SQL Remote オブジェクトのリストは、TEMPDB に Adaptive Server Anywhere のシステム・テーブル・セットを作成する **sp_populate_sql_anywhere** プロシージャが **sr_object** テーブルから取得します。

カラム

カラム	データ型	説明
name	VARCHAR(128)	オブジェクトの名前
type	CHAR(1)	次のいずれか • **U** ユーザ定義テーブル • **V** ビュー • **P** プロシージャ

sr_option テーブル

機能

SQL Remote で使用するレプリケーション・オプションを各ローに示します。

カラム

カラム	データ型	説明
option	VARCHAR(128)	オプションの名前
値	VARCHAR(128)	オプションの設定

説明

使用可能なオプションの詳細については、「SQL Remote のオプション」406 ページを参照してください。

sr_passthrough テーブル

機能

各ローに、ユーザやパブリケーションのサブスクライバに送信するパススルー・オペレーションを 1 つ示します。

カラム

カラム	データ型	説明
operation	VARCHAR(20)	**sp_passthrough** または **sp_passthrough_piece** を使用して入力された 1 つのパススルー・オペレーションまたはパススルー・オペレーションの一部
value	VARCHAR(255)	オペレーションを受信するユーザを示すサブスクリプション・カラム値
id	INT	オペレーションを受信するユーザ

sr_publication テーブル

機能 SQL Remote パブリケーションについて、各ローに示します。

カラム

カラム	データ型	説明
publication_id	INT	パブリケーションの識別子
publication_name	VARCHAR(128)	パブリケーションの名前

sr_publisher テーブル

機能 ローは、パブリッシャのユーザ ID を保持します。

カラム

カラム	データ型	説明
user_id	INT	パブリッシャのユーザ ID

sr_remoteoption テーブル

機能 SQL Remote のメッセージ・リンク・パラメータの値について、各ローに示します。

カラム

カラム	データ型	説明
option_id	整数	メッセージ・リンク・パラメータの ID 番号
user_id	整数	パラメータが設定されているユーザ ID
"setting00"	VARCHAR(255)	メッセージ・リンク・パラメータの値

sr_remoteoptiontype テーブル

機能

SQL Remote のメッセージ・リンク・パラメータについて、各ローで示します。

カラム

カラム	データ型	説明
option_id	整数	メッセージ・リンク・パラメータの ID 番号
type_id	整数	このパラメータを使用するメッセージ・タイプの ID 番号
"option"	VARCHAR(128)	メッセージ・リンク・パラメータの名前

sr_remotetable テーブル

機能

SQL Remote を使用してレプリケーション用にマーク付けされるテーブルを 1 つずつ各ローに示します。

カラム

カラム	データ型	説明
table_id	INT	テーブルの ID

カラム	データ型	説明
resolve_name	VARCHAR(128)	競合が発生した場合に実行されるストアド・プロシージャの名前
old_row_name	VARCHAR(128)	古いローの名前を保持するテーブル
remote_row_name	VARCHAR(128)	リモート・ローの名前を保持するテーブル

sr_remotetype テーブル

機能

SQL Remote のメッセージ・タイプについて 1 つずつ、パブリッシャ・アドレスを含めて、各ローで示します。

カラム

カラム	データ型	説明
type_id	INT	メッセージ・タイプの ID 番号
type_name	VARCHAR(128)	メッセージ・タイプ。以下のタイプごとに別のローが存在します。 • FILE • MAPI • VIM • SMTP
publisher_address	VARCHAR(128)	メッセージ・タイプ **type_name** 用のパブリッシャ・アドレス

sr_remoteuser テーブル

機能

REMOTE パーミッションを持つユーザ ID (サブスクライバ) について、各ローで示します。そのユーザが送受信した SQL Remote メッセージのステータスについても、合わせて示します。

カラム

カラム	データ型	説明
user_id	INT	REMOTE パーミッションを持つユーザの ID
consolidate	CHAR(1)	このカラムには、REMOTE パーミッションを付与されたユーザを示す N か、CONSOLIDATE パーミッションを付与されたユーザを示す Y のいずれかが含まれます。
type_id	INT	このユーザへメッセージを送信する場合に使用される、メッセージ・システムの ID
address	VARCHAR(128)	SQL Remote メッセージが送信されるアドレス。アドレスは **address_type** に対して適切でなければいけません。
frequency	CHAR(1)	SQL Remote メッセージが送信される頻度
send_time	DATETIME	次にメッセージがこのユーザへ送信される時刻
log_send	NUMERIC(20, 0)	メッセージは、**log_send** が **log_sent** よりも大きいサブスクライバだけに送信されます。
time_sent	DATETIME	このサブスクライバに、最後にメッセージが送信された時刻
log_sent	NUMERIC(20, 0)	最後に送信されたオペレーションのログ・オフセット
confirm_sent	NUMERIC(20, 0)	このサブスクライバから最後に確認されたオペレーションに対する、ログ・オフセット
send_count	INT	SQL Remote メッセージが送信された回数

カラム	データ型	説明
resend_count	INT	サブスクライバ・データベースで、メッセージが一度だけ適用されたことを確認するためのカウンタ
time_received	DATETIME	最後にこのサブスクライバからメッセージを受信した時刻
log_received	NUMERIC(20, 0)	現在のデータベースで最後に受信されたオペレーションに対する、サブスクライバ・データベース内のログ・オフセット
confirm_received	NUMERIC(20, 0)	確認メッセージが送信された最後のオペレーションに対する、サブスクライバ・データベース内のログ・オフセット
receive_count	INT	このサブスクライバから受信したメッセージ数
rereceive_count	INT	現在のデータベースで、メッセージが一度だけ適用されたことを確認するためのカウンタ
filler1	CHAR(255)	予約
filler2	CHAR(255)	予約
filler3	CHAR(255)	予約
filler4	CHAR(255)	予約

sr_subscription テーブル

機能

REMOTE パーミッションを持つあるユーザ ID からの、あるパブリケーションに対するサブスクリプションについて、各ローで示します。

カラム

カラム	データ型	説明
publication_id	INT	ユーザ ID のサブスクリプションが作成されているパブリケーションの識別子
user_id	INT	パブリケーションに対するサブスクリプションが作成されているユーザ ID
subscribe_by	VARCHAR(128)	SUBSCRIBE BY 式を使用するパブリケーションの場合、このカラムにはこのサブスクリプションと一致する値が保持されます。
created	NUMERIC(20, 0)	トランザクション・ログ内の、サブスクリプションが作成されたオフセット
started	NUMERIC(20, 0)	トランザクション・ログ内の、サブスクリプションが開始されたオフセット
operation	VARCHAR(20)	

SQL Remote システム・ビュー

この項では、SQL Remote 情報を表示および要約するために、SQL Remote が使用するデータベース・ビューについて説明します。

sr_articles ビュー

機能

各ローのリストにアーティクルを 1 つ示します。

カラム

カラム	説明
publication_name	このアーティクルが含まれるパブリケーション
table_name	各アーティクルは、単一のテーブルのカラムとローから構成されます。このカラムには、このテーブルの名前が含まれます。
where_expr	WHERE 句で定義されたローのサブセットを含むアーティクルの場合、このカラムに探索条件が含まれます。
subscribe_by_expr	SUBSCRIBE BY 式で定義されたローのサブセットを含むアーティクルの場合、このカラムに式が含まれます。
subscribe_by_view	ビューで定義されたローのサブセットを含むアーティクルの場合、このカラムにはビューの名前が含まれます。

sr_articlecols ビュー

機能

あるアーティクル内の 1 カラムについて、各ローで示します。

カラム

カラム	説明
publication_name	カラムが含まれるパブリケーションの名前

カラム	説明
table_name	カラムが属するテーブルの名前
column_name	カラムの名前

sr_publications ビュー

機能　すべてのパブリケーションの名前をリストします。

カラム

カラム	説明
publication_name	パブリケーションの名前

sr_remoteoptions ビュー

機能　システム・テーブル remoteoption と remoteoptiontype に格納されている SQL Remote のメッセージ・リンク・パラメータとその値を、より読みやすい形でリストします。

カラム

カラム	説明
type_name	メッセージ・リンク・タイプ
"option"	オプションの名前
setting	オプションの値

sr_remotetables ビュー

機能　**remotetable** システム・テーブルに格納されている、SQL Remote レプリケーション用にマーク付けされたテーブルをより読みやすい形でリストします。

このテーブルは、Adaptive Server Enterprise にだけ存在します。

カラム

カラム	説明
table_name	テーブルの名前
resolve_name	競合が発生した場合に実行されるストアド・プロシージャの名前
old_row_name	古いローの名前を保持するテーブル
remote_row_name	リモート・ローの名前を保持するテーブル

sr_remotetypes ビュー

機能 **remotetype** システム・テーブルに格納されているメッセージ・タイプをリストします。

カラム

カラム	説明
type_id	メッセージ・タイプの ID 番号
type_name	メッセージ・タイプ。以下のタイプごとに別のローが存在します。 • FILE • MAPI • VIM • SMTP
publisher_address	メッセージ・タイプ **type_name** 用のパブリッシャ・アドレス

sr_remoteusers ビュー

機能 リモート・ユーザとそのステータスについての情報をリストします。

カラム

カラム	説明
user_name	REMOTE パーミッションを持つユーザの ID
consolidate	このカラムには、REMOTE パーミッションを付与されたユーザを示す N か、CONSOLIDATE パーミッションを付与されたユーザを示す Y のいずれかが含まれます。
type_name	このユーザにメッセージを送信するときに使用されるメッセージ・システムの名前
address	SQL Remote メッセージが送信されるアドレス。アドレスは address_type に対して適切でなければいけません。
frequency	SQL Remote メッセージが送信される頻度
send_time	次にメッセージがこのユーザへ送信される時刻
next_send	次にメッセージがこのユーザへ送信される時刻を、より分かりやすい形式で表したもの
log_send	メッセージは、log_send が log_sent よりも大きいサブスクライバだけに送信されます。
time_sent	このサブスクライバに、最後にメッセージが送信された時刻
log_sent	最後に送信された操作のログ・オフセット
confirm_sent	このサブスクライバから最後に確認されたオペレーションに対する、ログ・オフセット
send_count	SQL Remote メッセージが送信された回数
resend_count	サブスクライバ・データベースで、メッセージが一度だけ適用されたことを確認するためのカウンタ
time_received	最後にこのサブスクライバからメッセージを受信した時刻
log_received	現在のデータベースで最後に受信されたオペレーションに対する、サブスクライバ・データベース内のログ・オフセット

カラム	説明
confirm_received	最後に確認メッセージが送信されたオペレーションに対する、サブスクライバ・データベース内のログ・オフセット
receive_count	メッセージが受信された回数
rereceive_count	現在のデータベースで、メッセージが一度だけ適用されたことを確認するためのカウンタ

sr_subscriptions ビュー

機能

サブスクリプションについての情報を、各ローでリストします。

カラム

カラム	説明
publication_name	ユーザ ID のサブスクリプションが作成されているパブリケーションの名前
user_name	パブリケーションに対するサブスクリプションが作成されているユーザ ID
subscribe_by	SUBSCRIBE BY 式を使用するパブリケーションの場合、このカラムにはこのサブスクリプションと一致する値が保持される。
created	トランザクション・ログ内の、サブスクリプションが作成されたオフセット
started	トランザクション・ログ内の、サブスクリプションが開始されたオフセット

ステーブル・キュー・テーブル

この項では、ステーブル・キュー情報を定義し管理するために SQL Remote が使用するデータベース・テーブルについて説明します。ステーブル・キューは、SQL Remote データベースと同じデータベースに保存されるか、または別のデータベースに保存されます。

ステーブル・キューは、SQL Remote for Adaptive Server Enterprise だけが使用します。

sr_queue_state テーブル

機能

ステーブル・キューのステータスについての永続的なグローバル情報を格納する 1 つのローから構成されたテーブルです。

カラム

カラム	データ型	説明
version	INT	ステーブル・キューのバージョン番号
page_id	INT	スキャンされた最後のエントリのトランザクション・ログの **page_id**
row_id	INT	スキャンされた最後のエントリのトランザクション・ログの **row_id**
confirm_offset	NUMERIC(20,0)	すべてのリモート・ユーザから受信した確認オフセットの最小値。Message Agent はこの値を使用して、ステーブル・キューから削除できるトランザクションを決定します。
commit_offset	NUMERIC(20,0)	最も古い未完了トランザクションよりも前に完了した、最新トランザクションのログ・オフセット

カラム	データ型	説明
backup_offset	NUMERIC(20,0)	前回の **dump database** コマンドまたは **dump transaction** コマンドのトランザクション・ログ・オフセット。 この情報は、-u コマンド・ライン・スイッチ (バックアップ・トランザクションのみをレプリケート) を使用して Message Agent を実行した場合に使用されます。
marker	DATETIME	ステーブル・キューへスキャンされた最新の受信メッセージ。メッセージが Adaptive Server Enterprise サーバに適用されると、sr_remoteuser テーブルに time_received カラムが設定されます。トランザクション・ログがスキャンされ、そのメッセージからのトランザクションがステーブル・キューへスキャンされると、sr_queue_state テーブルの time_received カラムが設定されます。 このカラムの目的は、トランザクション・ログを連続的にスキャンする Message Agent と、バッチ・モードでメッセージの送受信を行う別の Message Agent との間の調整です。バッチ・モードの場合は、Message Agent はメッセージを受信し、それらのメッセージがステーブル・キューにスキャンされるまで待機し、その後メッセージを送信します。待機は、**sr_queue_state** テーブルの **time_received** カラムを参照することによってデータベース全体で行われます。

カラム	データ型	説明
confirmed_id	NUMERIC(20,0)	送信スレッドは、この値より小さい **confirmed_id** を持つローを **sr_confirmed_transaction** テーブルから削除します。

sr_transaction テーブル

機能

このテーブルには、ステーブル・キュー内の各トランザクションに対応するローが 1 つあります。

カラム

カラム	説明
offset	トランザクション用コミット・オペレーションのログ・オフセット。この値をもとに、各トランザクションがユニークに識別されます。
user_id	トランザクションを発生させたリモート・ユーザ。リモート・ユーザからトランザクションが発生しなかった場合は、このカラムに NULL が入ります。 **user_id** カラムは、アクションが入力元のリモート・サイトにレプリケートされないようにするために使用されます。
data	トランザクション自体を内部表現にしたもの

sr_confirmed_transaction テーブル

機能

各ローは、**sr_transaction** 内の対応するローをマーク付けします。

カラム

カラム	データ型	説明
confirmed_id	NUMERIC(20,0)	ユニークな ID

カラム	データ型	説明
offset	NUMERIC(20,0)	**sr_transaction** にある削除対象のローをマーク付けするときに使用するオフセットのコピー

sr_queue_coordinate テーブル

機能

SQL Remote のログ・スキャン用スレッドと送信用スレッドを調整し、ステーブル・キューおよび関連テーブルにアクセスできるようにする、単一ローです。

カラム

カラム	データ型	説明
status	CHAR(1)	SQL Remote がステーブル・キューを使用していない場合は N。SQL Remote のログ・スキャン用スレッドと送信用スレッドがキューにアクセスした場合は I と S。

第 17 章

Adaptive Server Anywhere コマンド・リファレンス

この章の内容

この章では、SQL Remote コマンドを実行する際に使用する SQL 文と、SQL Remote インストール環境とその状態に関する情報を格納するためのシステム・テーブルについて説明します。

ALTER REMOTE MESSAGE TYPE 文

機能

この文は、特定のメッセージ・システム、または作成したメッセージ・タイプに対する、パブリッシャのメッセージ・システム、またはパブリッシャのアドレスの変更に使用します。

構文

ALTER REMOTE MESSAGE TYPE *message-system*
    ADDRESS *address*

message-system: **FILE** | **FTP** | **MAPI** | **SMTP** | **VIM**

address: *string*

パラメータ

パラメータ	説明
message-system	SQL Remote がサポートするメッセージ・システムの名前。次のいずれかの値を指定します。
address	指定したメッセージ・システムに対して有効なアドレスを含む文字列。

パーミッション

DBA 権限が必要です。

関連する動作

オートコミット。

参照

『ASA SQL リファレンス・マニュアル』> 「ALTER REMOTE MESSAGE TYPE 文 [SQL Remote]」

「CREATE REMOTE MESSAGE TYPE 文」454 ページ

「sp_remote_type プロシージャ」530 ページ

CREATE PUBLICATION 文

機能

この文は、パブリケーションを作成するために使用します。
SQL Remote では、統合データベースやリモート・データベース内のレプリケートされたデータはパブリケーションによって識別されます。

構文

CREATE PUBLICATION [*owner*.]*publication-name*
(**TABLE** *article-description*, …)

owner, *publication-name* : *identifier*

article-description:
table-name [**(** *column-name*, … **)**]
[**WHERE** *search-condition*]
[**SUBSCRIBE BY** *expression*]

参照

『ASA SQL リファレンス・マニュアル』> 「CREATE PUBLICATION 文」

CREATE REMOTE MESSAGE TYPE 文

機能

この文は、データベースからの出力メッセージのメッセージ・リンクとリターン・アドレスの識別に使用します。

構文

CREATE REMOTE MESSAGE TYPE *message-system*
ADDRESS *address*

message-system: **FILE** | **FTP** | **MAPI** | **SMTP** | **VIM**

address: *string*

パラメータ

パラメータ	説明
message-system	サポートしているメッセージ・システムの名前。
address	指定したメッセージ・システムのアドレス。

パーミッション

DBA 権限が必要です。

関連する動作

オートコミット。

参照

『ASA SQL リファレンス・マニュアル』> 「CREATE REMOTE MESSAGE TYPE 文 [SQL Remote]」

「GRANT PUBLISH 文」463 ページ

「GRANT REMOTE 文」464 ページ

「GRANT CONSOLIDATE 文」462 ページ

「sp_remote_type プロシージャ」530 ページ

「メッセージ・タイプの使用」269 ページ

CREATE SUBSCRIPTION 文

機能

この文は、パブリケーションに対するユーザのサブスクリプションの作成に使用します。

構文

CREATE SUBSCRIPTION
    TO *publication-name* [(*subscription-value*)]
    FOR *subscriber-id*

publication-name: *identifier*

subscription-value, *subscriber-id*: *string*

subscriber-id: *string*

パラメータ

パラメータ	説明
publication-name	ユーザのサブスクリプションを作成するパブリケーション名。パブリケーションの所有者を含めることもできます。
subscription-value	パブリケーションのサブスクリプション式と照合される文字列。サブスクライバは、サブスクリプション式がサブスクリプション値と一致するすべてのローを受信します。
subscriber-id	パブリケーションに対するサブスクライバのユーザ ID。ユーザには、REMOTE パーミッションが必要です。

パーミッション

DBA 権限が必要です。

関連する動作

オートコミット。

参照

『ASA SQL リファレンス・マニュアル』>「CREATE SUBSCRIPTION 文 [SQL Remote]」

「DROP SUBSCRIPTION 文」461 ページ

「GRANT REMOTE 文」464 ページ

「SYNCHRONIZE SUBSCRIPTION 文」476 ページ

「START SUBSCRIPTION 文」474 ページ

「sp_subscription プロシージャ」537 ページ

CREATE TRIGGER 文

機能

この文は、データベースへの新しいトリガの作成に使用します。特に SQL Remote が使用するために設計されたトリガのフォームが 1 つ用意されています。

構文

CREATE TRIGGER *trigger-name trigger-time*
trigger-event, …
[**ORDER** *integer*] **ON** *table-name*
[**REFERENCING** [**OLD AS** *old-name*]
[**NEW AS** *new-name*]]
[**REMOTE AS** *remote-name*]]
[**FOR EACH** { **ROW** | **STATEMENT** }]
[**WHEN** (*search-condition*)]
[**IF UPDATE** (*column-name*) **THEN**
[{ **AND** | **OR** } **UPDATE** (*column-name*)] …]
compound-statement
[**ELSEIF UPDATE** (*column-name*) **THEN**
[{ **AND** | **OR** } **UPDATE** (*column-name*)] …
compound-statement
END IF]]

trigger-time:
BEFORE | **AFTER** | **RESOLVE**

trigger-event:
DELETE | **INSERT** | **UPDATE**
| **UPDATE OF** *column-name* [**,** *column-name***,** …]

パラメータ

trigger-time　ロー・レベルのトリガを定義して、挿入、更新、または削除の後 (BEFORE) または前 (AFTER) に実行できます。文レベルのトリガは、文の後 (AFTER) に実行されます。RESOLVE トリガ時間は、SQL Remote と共に使用します。これは、ロー・レベルの UPDATE または UPDATE OF カラム・リストの前だけで起動されます。

BEFORE UPDATE トリガは、ローを対象に UPDATE が実行されるたびに起動されます。この場合、新しい値が古い値と異なるかどうかは問題ではありません。一方、AFTER UPDATE トリガは、新しい値が古い値と異なる場合にのみ起動されます。

トリガ・イベント　トリガは次のイベントの 1 つまたは複数によって起動できます。

- **DELETE**　関連するテーブルのローが削除されると、呼び出されます。
- **INSERT**　このトリガに関連するテーブルに新しいローが挿入されると、呼び出されます。
- **UPDATE**　関連するテーブルのローが更新されると、呼び出されます。
- **UPDATE OF column-list**　関連するテーブルのローが更新され、*column-list* のカラムが修正されると、呼び出されます。

使用法

特に制限なし。

パーミッション

RESOURCE 権限またはテーブルに対する ALTER パーミッションが必要です。または、DBA 権限が必要です。CREATE TRIGGER はテーブルにテーブル・ロックを設定するので、テーブルを排他的に使用してください。

関連する動作

オートコミット。

参照

『ASA SQL リファレンス・マニュアル』> 「CREATE TRIGGER 文 [SQL Remote]」

「UPDATE 文」477 ページ

DROP PUBLICATION 文

機能　この文は、パブリケーションを削除するときに使用します。
SQL Remote では、統合データベースやリモート・データベース内のレプリケートされたデータはパブリケーションによって識別されます。

構文　**DROP PUBLICATION** [*owner*.]*publication-name*

owner, *publication-name* : *identifier*

参照　『ASA SQL リファレンス・マニュアル』> 「DROP PUBLICATION 文」

DROP REMOTE MESSAGE TYPE 文

機能

この文は、データベースからのメッセージ・タイプ定義の削除に使用します。

構文

DROP REMOTE MESSAGE TYPE *message-system*

message-system: **FILE** | **FTP** | **MAPI** | **SMTP** | **VIM**

パラメータ

パラメータ	説明
message-system	SQL Remote がサポートするメッセージ・システムの名前。

パーミッション

DBA 権限が必要です。指定されたメッセージ・タイプに REMOTE または CONSOLIDATE パーミッションを持つユーザがいると、そのタイプは削除できません。

関連する動作

オートコミット。

参照

『ASA SQL リファレンス・マニュアル』> 「DROP REMOTE MESSAGE TYPE 文 [SQL Remote]」

「CREATE REMOTE MESSAGE TYPE 文」454 ページ

「ALTER REMOTE MESSAGE TYPE 文」452 ページ

「sp_drop_remote_type プロシージャ」488 ページ

「メッセージ・タイプの使用」269 ページ

DROP SUBSCRIPTION 文

機能

この文は、パブリケーションからユーザのサブスクリプションを削除するのに使用します。

構文

DROP SUBSCRIPTION TO *publication-name* [(*subscription-value*)]
　　FOR *subscriber-id*, …

subscription-value: *string*

subscriber-id: *string*

パラメータ

パラメータ	説明
publication-name	ユーザのサブスクリプションを作成するパブリケーション名。パブリケーションの所有者を含めることもできます。
subscription-value	パブリケーションのサブスクリプション式と照合される文字列。ユーザは単一のパブリケーションに対して複数のサブスクリプションを持つことがあるので、この値は必須です。
subscriber-id	パブリケーションに対するサブスクライバのユーザ ID。

パーミッション

DBA 権限が必要です。

関連する動作

オートコミット。

参照

『ASA SQL リファレンス・マニュアル』> 「DROP SUBSCRIPTION 文 [SQL Remote]」

「CREATE SUBSCRIPTION 文」455 ページ

「DROP SUBSCRIPTION 文」461 ページ

GRANT CONSOLIDATE 文

機能

この文は、SQL Remote 階層において、現在のデータベースのすぐ上にあり、現在のデータベースからメッセージを受信するデータベースの識別に使用します。

構文

GRANT CONSOLIDATE
TO *userid, …*
TYPE *message-system, …*
ADDRESS *address-string, …*
[**SEND** { **EVERY** | **AT** }*'hh:mm: ss'*]

message-system: **FILE** | **FTP** | **MAPI** | **SMTP** | **VIM**

address: *string*

パラメータ

パラメータ	説明
userid	パーミッションを付与されるユーザのユーザ ID。
message-system	SQL Remote がサポートするメッセージ・システムの名前。
address	指定したメッセージ・システムのアドレス。

パーミッション

DBA 権限が必要です。

関連する動作

オートコミット。

参照

『ASA SQL リファレンス・マニュアル』>「GRANT CONSOLIDATE 文 [SQL Remote]」

「GRANT REMOTE 文」464 ページ

「REVOKE CONSOLIDATE 文」469 ページ

「GRANT PUBLISH 文」463 ページ

「sp_grant_consolidate プロシージャ」490 ページ

GRANT PUBLISH 文

機能　この文は、現在のデータベースのパブリッシャの識別に使用します。

構文　**GRANT PUBLISH TO** *userid*

パーミッション　DBA 権限が必要です。

関連する動作　オートコミット。

参照

『ASA SQL リファレンス・マニュアル』> 「GRANT PUBLISH 文 [SQL Remote]」

「GRANT REMOTE 文」464 ページ

「GRANT CONSOLIDATE 文」462 ページ

「REVOKE PUBLISH 文」470 ページ

『ASA SQL リファレンス・マニュアル』> 「CREATE PUBLICATION 文」

「CREATE SUBSCRIPTION 文」455 ページ

「sp_publisher プロシージャ」511 ページ

GRANT REMOTE 文

機能

この文は、SQL Remote 階層において、現在のデータベースのすぐ下にあり、現在のデータベースからメッセージを受信するデータベースの識別に使用します。これらはリモート・ユーザといいます。

構文

GRANT REMOTE TO *userid, …*
    TYPE *message-system, …*
    ADDRESS *address-string, …*
    [**SEND** { **EVERY** | **AT** } '*send-time*']

パラメータ

パラメータ	説明
userid	パーミッションを付与されるユーザのユーザ ID。
message-system	SQL Remote がサポートするメッセージ・システムの名前。次のいずれかの値を指定します。 • FILE • FTP • MAPI • SMTP • VIM
address-string	指定したメッセージ・システムに対して有効なアドレスを含む文字列。
send-time	*hh*:*mm*:*ss* のフォームで時刻を指定した文字列。

パーミッション

DBA 権限が必要です。

関連する動作

オートコミット。

参照

『ASA SQL リファレンス・マニュアル』> 「GRANT REMOTE 文 [SQL Remote]」

「GRANT CONSOLIDATE 文」462 ページ

「REVOKE REMOTE 文」471 ページ

「GRANT PUBLISH 文」463 ページ

「sp_grant_remote プロシージャ」493 ページ

「REMOTE パーミッションと CONSOLIDATE パーミッションの付与と取り消し」260 ページ

GRANT REMOTE DBA 文

機能

この文は、ユーザ ID に対して DBA 権限を付与するために使用します。ただし、Message Agent から接続したときのみ有効です。

構文

GRANT REMOTE DBA
TO *userid, …*
IDENTIFIED BY *password*

パーミッション

DBA 権限が必要です。

関連する動作

オートコミット。

参照

『ASA SQL リファレンス・マニュアル』> 「GRANT REMOTE DBA 文 [SQL Remote]」

「Message Agent とレプリケーション・セキュリティ」313 ページ

「REVOKE REMOTE DBA 文」472 ページ

PASSTHROUGH 文

機能　この文は、SQL Remote 管理のパススルー・モードの起動または停止に使用します。構文 1 と 2 はパススルー・モードを起動し、構文 3 はパススルー・モードを停止します。

構文 1　**PASSTHROUGH** [**ONLY**] **FOR** *userid, …*

構文 2　**PASSTHROUGH** [**ONLY**] **FOR SUBSCRIPTION**
TO [(*owner*)].*publication-name* [(*constant*)]

構文 3　**PASSTHROUGH STOP**

パーミッション　DBA 権限が必要です。

関連する動作　なし。

参照　『ASA SQL リファレンス・マニュアル』> 「PASSTHROUGH 文 [SQL Remote]」

「sp_passthrough プロシージャ」502 ページ

REMOTE RESET 文

機能

この文は、カスタム・データベース抽出プロシージャで、単一トランザクションでの 1 リモート・ユーザのすべてのサブスクリプションの起動に使用します。

構文

REMOTE RESET *userid*

パーミッション

DBA 権限が必要です。

関連する動作

この文のオートコミットはありません。

参照

『ASA SQL リファレンス・マニュアル』> 「REMOTE RESET 文 [SQL Remote]」

「START SUBSCRIPTION 文」474 ページ

REVOKE CONSOLIDATE 文

機能　この文は、このデータベースからの SQL Remote メッセージを統合データベースが受信するのを中止させます。

構文　**REVOKE CONSOLIDATE FROM** *userid, …*

パーミッション　DBA 権限が必要です。

関連する動作　オートコミット。指定したユーザのすべてのサブスクリプションを削除します。

参照　『ASA SQL リファレンス・マニュアル』>「REVOKE CONSOLIDATE 文 [SQL Remote]」

「GRANT CONSOLIDATE 文」462 ページ

「sp_revoke_consolidate プロシージャ」535 ページ

REVOKE PUBLISH 文

機能

この文は、名前を指定したユーザ ID を CURRENT パブリッシャとして識別するのを中止します。

構文

REVOKE PUBLISH FROM *userid*

パーミッション

DBA 権限が必要です。

関連する動作

オートコミット。

参照

『ASA SQL リファレンス・マニュアル』> 「REVOKE PUBLISH 文 [SQL Remote]」

「GRANT PUBLISH 文」463 ページ

「REVOKE REMOTE 文」471 ページ

『ASA SQL リファレンス・マニュアル』> 「CREATE PUBLICATION 文」

「CREATE SUBSCRIPTION 文」455 ページ

「sp_publisher プロシージャ」511 ページ

REVOKE REMOTE 文

機能　この文は、このデータベースからの SQL Remote メッセージをユーザが受信できないようにします。

構文　**REVOKE REMOTE FROM** *userid, …*

パーミッション　DBA 権限が必要です。

関連する動作　オートコミット。指定したユーザのすべてのサブスクリプションを削除します。

参照　『ASA SQL リファレンス・マニュアル』> 「REVOKE REMOTE 文 [SQL Remote]」

「sp_revoke_remote プロシージャ」536 ページ

REVOKE REMOTE DBA 文

機能

この文は、ユーザ ID に対して DBA 権限を付与するために使用します。ただし、Message Agent から接続したときのみ有効です。

構文 1

REVOKE REMOTE DBA
FROM *userid, …*

パーミッション

DBA 権限が必要です。

関連する動作

オートコミット。

参照

『ASA SQL リファレンス・マニュアル』> 「REVOKE REMOTE DBA 文 [SQL Remote]」

「Message Agent とレプリケーション・セキュリティ」313 ページ

「GRANT REMOTE DBA 文」466 ページ

SET REMOTE OPTION 文

機能

この文は、SQL Remote メッセージ・リンクのメッセージ制御パラメータの設定に使用します。

構文

SET REMOTE *link-name* **OPTION**
[*userid*.| **PUBLIC**.]*link-option-name* = *link-option-value*

パラメータ

link-name:
file | **ftp** | **mapi** | **smtp** | **vim**

link-option-name:
file-option | *ftp-option* | *mapi-option* | *smtp-option* | *vim-option*

file-option:
debug | **directory** | **unlink_delay**

ftp-option:
active_mode | **debug** | **host** | **password**
| **port** | **root_directory** | **user**

mapi-option:
debug | **force_download** | **ipm_receive**
| **ipm_send** | **profile**

smtp-option:
debug | **local_host** | **pop3_host**
| **pop3_password** | **pop3_userid**
| **smtp_host** | **top_supported**

vim-option:
debug | **password** | **path** | **receive_all**
| **send_vim_mail** | **userid**

link-option-value:
string

パーミッション

DBA 権限が必要です。パブリッシャは自分自身のオプションを設定できます。

関連する動作

オートコミット。

参照

「sp_link_option プロシージャ」496 ページ

START SUBSCRIPTION 文

機能

この文は、パブリケーションに対するユーザのサブスクリプションの開始に使用します。

構文

START SUBSCRIPTION
TO *publication-name* [**(** *subscription-value* **)**]
FOR *subscriber-id, …*

パラメータ

パラメータ	説明
publication-name	ユーザのサブスクリプションを作成するパブリケーション名。パブリケーションの所有者を含めることもできます。
subscription-value	パブリケーションのサブスクリプション式と照合される文字列。サブスクライバは単一のパブリケーションに対して複数のサブスクリプションを持つことがあるので、この値は必須です。
subscriber-id	パブリケーションに対するサブスクライバのユーザ ID。このユーザは、パブリケーションへのサブスクリプションを持っている必要があります。

パーミッション

DBA 権限が必要です。

関連する動作

オートコミット。

参照

『ASA SQL リファレンス・マニュアル』>「START SUBSCRIPTION 文 [SQL Remote]」

「CREATE SUBSCRIPTION 文」455 ページ

「REMOTE RESET 文」468 ページ

「SYNCHRONIZE SUBSCRIPTION 文」476 ページ

「sp_subscription プロシージャ」537 ページ

STOP SUBSCRIPTION 文

機能

この文は、パブリケーションに対するユーザのサブスクリプションを停止します。

構文

STOP SUBSCRIPTION
　TO *publication-name* [(*subscription-value*)]
　FOR *subscriber-id, …*

パラメータ

パラメータ	説明
publication-name	ユーザのサブスクリプションを作成するパブリケーション名。パブリケーションの所有者を含めることもできます。
subscription-value	パブリケーションのサブスクリプション式と照合される文字列。サブスクライバは単一のパブリケーションに対して複数のサブスクリプションを持つことがあるので、この値は必須です。
subscriber-id	パブリケーションに対するサブスクライバのユーザ ID。このユーザは、パブリケーションへのサブスクリプションを持っている必要があります。

パーミッション

DBA 権限が必要です。

関連する動作

オートコミット。

参照

『ASA SQL リファレンス・マニュアル』> 「STOP SUBSCRIPTION 文 [SQL Remote]」

「CREATE SUBSCRIPTION 文」455 ページ

「SYNCHRONIZE SUBSCRIPTION 文」476 ページ

SYNCHRONIZE SUBSCRIPTION 文

機能

この文は、パブリケーションに対するユーザのサブスクリプションの同期に使用します。

構文

SYNCHRONIZE SUBSCRIPTION
TO *publication-name* [(*subscription-value*)]
FOR *remote-user, …*

パラメータ

パラメータ	説明
publication-name	ユーザのサブスクリプションを作成するパブリケーション名。パブリケーションの所有者を含めることもできます。
subscription-value	パブリケーションのサブスクリプション式と照合される文字列。サブスクライバは単一のパブリケーションに対して複数のサブスクリプションを持つことがあるので、この値は必須です。
remote-user	パブリケーションに対するサブスクライバのユーザ ID。このユーザは、パブリケーションへのサブスクリプションを持っている必要があります。

パーミッション

DBA 権限が必要です。

関連する動作

オートコミット。

参照

『ASA SQL リファレンス・マニュアル』> 「SYNCHRONIZE SUBSCRIPTION 文 [SQL Remote]」

「CREATE SUBSCRIPTION 文」455 ページ

「START SUBSCRIPTION 文」474 ページ

UPDATE 文

機能

この文は、データベース内のデータの修正に使用します。

構文 1

UPDATE *table-list*
　SET *column-name* = *expression*, …
　[**VERIFY** (*column-name*, …) **VALUES** (*expression*, …)]
　[**WHERE** *search-condition*]
　[**ORDER BY** *expression* [**ASC** | **DESC**], …]

構文 2

UPDATE *table*
　PUBLICATION *publication*
　{ **SUBSCRIBE BY** *expression* |
　OLD SUBSCRIBE BY *expression*
　NEW SUBSCRIBE BY *expression* }
　WHERE *search-condition*

expression: *value* | *subquery*

使用法

構文 1 と構文 2 は SQL Remote のみに適用されます。

構文 2 で OLD SUBSCRIBE BY 式と NEW SUBSCRIBE BY 式を付けない場合は BEFORE トリガで使用する必要があります。

構文 2 で OLD SUBSCRIBE BY 式と NEW SUBSCRIBE BY 式を付ける場合はどこでも使用できます。

パーミッション

修正対象のカラムに対する UPDATE パーミッションが必要です。

関連する動作

なし。

参照

『ASA SQL リファレンス・マニュアル』> 「UPDATE 文 [SQL Remote]」

「CREATE TRIGGER 文」457 ページ

第 18 章

Adaptive Server Enterprise コマンド・リファレンス

この章の内容

この章では、SQL Remote コマンドの実行に使用する、SQL Remote ストアド・プロシージャについて説明します。

sp_add_article プロシージャ

内容

アーティクルをパブリケーションに追加します。

構文

sp_add_article *publication_name*,
table_name,
where_expr,
subscribe_by_expr,
subscribe_by_view

引数	説明
publication_name	アーティクルを追加するパブリケーションの名前。
table_name	アーティクルがあるテーブル。
where_expr	オプションの引数。カラム名または NULL を指定。ここで指定されたカラムの値が NULL ではないローだけが、パブリケーションに含まれます。 デフォルト値は NULL です。この場合、パブリケーションにはすべてのローが含まれます。
subscribe_by_expr	サブスクリプションごとにパブリケーションに含めるローを定義する、新しいサブスクリプション式。 式は *table_name* にあるカラム名とします。デフォルト値は NULL です。
subscribe_by_view	パブリケーションに含めるカラムとローを定義するビュー。 詳細については、「抽出パフォーマンスのチューニング」197 ページと「共有ローの抽出パフォーマンスのチューニング」207 ページを参照してください。

参照

「sp_add_remote_table プロシージャ」484 ページ

「sp_create_publication プロシージャ」486 ページ

「sp_remove_article プロシージャ」532 ページ

『ASA SQL リファレンス・マニュアル』> 「CREATE PUBLICATION 文」

説明

sp_add_article を実行すると、アーティクルをパブリケーションに追加できます。**sp_add_remote_table** を使用して、テーブルをレプリケーション用にマーク付けしてから、パブリケーションに追加してください。マーク付けしないと、エラーとなります。

sp_add_article を呼び出すと、テーブルの全カラムがパブリケーションに追加されます。テーブルのカラムの一部だけをパブリケーションに含めるには、まず **sp_add_article** を実行し、その次に **sp_add_article_col** を呼び出します。

他のデータ定義変更と同様に、運用環境では、クワイエット SQL Remote インストール環境でのみ、このプロシージャを実行してください。

クワイエット・システム要件の詳細については、「スキーマの変更」355 ページを参照してください。

例

- 次の文は、SalesRep テーブルを SalesRepData という名前のパブリケーションに追加します。

```
sp_add_article 'SalesRepData', 'SalesRep'
 go
```

sp_add_article_col プロシージャ

内容

カラムをパブリケーション内のアーティクルに追加します。

構文

sp_add_article_col *publication_name*,
table_name,
column_name

引数	説明
publication_name	アーティクルを追加するパブリケーションの名前。
table_name	アーティクルがあるテーブル。
column_name	パブリケーション内のアーティクルに追加するカラム。

参照

「sp_add_article プロシージャ」480 ページ

「sp_remove_article プロシージャ」532 ページ

『ASA SQL リファレンス・マニュアル』> 「ALTER PUBLICATION 文」

説明

sp_add_article_col を実行すると、パブリケーション内のアーティクルにカラムが追加されます。まず、「sp_add_article プロシージャ」480 ページを使用して、パブリケーションにテーブルを追加してください。

テーブルの全カラムをパブリケーションに追加する場合は、sp_add_article_col を使用する必要はありません。sp_add_article だけを呼び出してください。

テーブルのカラムの一部だけをパブリケーションに追加するには、まず sp_add_article を呼び出し、その次にそのパブリケーションに含めるカラムごとに **sp_add_article_col** を呼び出します。

他のデータ定義変更と同様に、運用環境では、クワイエット SQL Remote インストール環境でのみ、このプロシージャを実行してください。

クワイエット・システム要件の詳細については、「スキーマの変更」355 ページを参照してください。

例

- 次の文は、employee テーブルのカラム emp_id と emp_lname を、Personnel という名前のパブリケーションに追加します。

```
sp_add_article 'Personnel', 'employee'
 sp_add_article_col 'Personnel', 'employee',
'emp_id'
 sp_add_article_col 'Personnel', 'employee',
'emp_lname'
 go
```

sp_add_remote_table プロシージャ

内容

SQL Remote レプリケーション用にテーブルをマーク付けします。

構文

sp_add_remote_table *table_name*,
[*resolve_procedure*,]
[*old_row_name*,]
[*remote_row_name*]

引数	説明
table_name	SQL Remote レプリケーション用にマーク付けするテーブル。
resolve_procedure	競合が発生した場合にアクションを実行するストアド・プロシージャの名前。
old_row_name	競合が発生した場合に値を保持するテーブルの名前。
remote_row_name	競合の原因となる UPDATE 文が適用された場合に、リモート・データベースで値を保持するテーブルの名前。

権限

このプロシージャを実行するには、システム管理者の権限が必要です。

参照

「sp_modify_remote_table プロシージャ」500 ページ

「sp_remove_remote_table プロシージャ」534 ページ

「競合の管理」210 ページ

説明

データベースの各テーブルは、**sp_add_remote_table** を使用してレプリケーション用にマーク付けしてから、SQL Remote パブリケーションに含めます。**sp_add_remote_table** を実行したら、「sp_add_article プロシージャ」480 ページと 「sp_add_article_col プロシージャ」482 ページを使用して、テーブルをパブリケーションに追加できます。

sp_add_remote_table プロシージャでは、テーブルにレプリケーション用のフラグを設定する **sp_setreplicate** が呼び出されます。これにより、トランザクション・ログに詳細情報を入れるよう、Adaptive Server Enterprise に通知されます。この情報には、ローの前後のイメージ全体が含まれます。

1 番目の引数には、レプリケーション用にマーク付けするテーブルの名前を指定します。

残りの 3 つの引数はオプションです。ここに指定するオブジェクト名は、カスタム競合解決のためだけに必要なものです。カスタム競合解決を実装する場合は、2 つのテーブルとストアド・プロシージャの名前を指定します。**sp_add_remote_table** プロシージャでは、競合解決の引数で指定されたオブジェクトが存在するかどうかはチェックされません。オブジェクトを作成するのは、テーブルをレプリケーション用にマーク付けする前でも、マーク付けした後でも、どちらでも構いません。

2 つのテーブルには、*table_name* で指定したテーブルと同じカラムとデータ型がなくてはなりません。

例

- 次の文は、Customer テーブルをレプリケーション用にマーク付けし、デフォルトの競合解決を使用するよう設定します。

  ```
  exec sp_add_remote_table Customer
  ```

- 次の文は、Customer テーブルをレプリケーション用にマーク付けし、競合解決用に Customer_Conflict という名前のストアド・プロシージャを使用するよう設定します。古いローとリモートのローはそれぞれ、old_Customer テーブルと remote_Customer テーブルに格納されます。

  ```
  exec sp_add_remote_table Customer,
  Customer_Conflict, old_Customer, remote_Customer
  ```

sp_create_publication プロシージャ

内容

パブリケーションを作成します。

構文

sp_create_publication *publication_name*

引数	説明
publication_name	パブリケーションの名前。

参照

「sp_drop_publication プロシージャ」487 ページ

『ASA SQL リファレンス・マニュアル』> 「CREATE PUBLICATION 文」

説明

sp_create_publication を実行すると、パブリケーションが作成されます。ただし、その内容は指定されていません。パブリケーションが作成されたら、「sp_add_remote_table プロシージャ」484 ページと「sp_add_article プロシージャ」480 ページを使用してアーティクルを追加する必要があります。

例

- 次の文は、**SalesRepData** という名前のパブリケーションを作成します。

```
sp_create_publication 'SalesRepData'
go
```

sp_drop_publication プロシージャ

内容

データベースからパブリケーションを削除します。

構文

sp_drop_publication *publication_name*

引数	説明
publication_name	削除するパブリケーションの名前。

参照

「sp_create_publication プロシージャ」486 ページ

『ASA SQL リファレンス・マニュアル』>「DROP PUBLICATION 文」

説明

sp_drop_publication を実行すると、データベースからパブリケーションを削除できます。パブリケーションを構成する全アーティクルと、パブリケーションに対するサブスクリプションも、削除されます。

例

- 次の文は、SalesRep という名前のパブリケーションを削除します。

```
sp_drop_publication 'SalesRep'
 go
```

sp_drop_remote_type プロシージャ

内容

データベースからメッセージ・タイプを削除します。

構文

sp_drop_remote_type *type_name*

引数	説明
type_name	削除するメッセージ・タイプ。次のいずれかの文字列を指定します。 • file • ftp • smtp • mapi • vim

参照

「sp_remote_type プロシージャ」530 ページ

「DROP REMOTE MESSAGE TYPE 文」460 ページ

説明

データベースから指定されたメッセージ・タイプを削除します。

例

- 次の文は、データベースから MAPI メッセージ・タイプを削除します。

```
sp_drop_remote_type mapi
go
```

sp_drop_sql_remote プロシージャ

内容

データベースから SQL Remote システム・オブジェクトを削除します。

構文

sp_drop_sql_remote

参照

「sp_queue_drop プロシージャ」516 ページ

説明

SQL Remote システム・オブジェクトが SQL Remote インストール環境で機能しないように、データベースから削除します。

SQL Remote オブジェクトのうち、**sp_drop_sql_remote** プロシージャだけは削除されません (プロシージャはデータベースから自分自身を削除できません)。SQL Remote オブジェクトの削除を完了するには、sp_drop_sql_remote を、呼び出した後明示的に削除します。

sp_drop_sql_remote プロシージャは、データベースからステーブル・キュー・オブジェクトを削除しません。ステーブル・キューを削除するには、「sp_queue_drop プロシージャ」516 ページを使用してください。

例

- 次の文は、データベースから SQL Remote システム・オブジェクトを削除します。

```
sp_drop_sql_remote_type
 go
 drop procedure sp_drop_sql_remote
 go
```

sp_grant_consolidate プロシージャ

内容

SQL Remote 階層で、現在のデータベースのすぐ上にあるデータベースを識別します。このデータベースは現在のデータベースからメッセージを受信します。このプロシージャは、リモート・データベースとして動作している Adaptive Server Enterprise データベースにのみ、適用されます。

構文

sp_grant_consolidate *user_name*,
type_name,
address
[, *frequency*]
[, *send_time*]

引数	説明
user_name	SQL Remote メッセージを受信できるように設定するユーザ ID。
type_name	使用されるメッセージ・タイプ。次のいずれかを指定します。 • file • ftp • smtp • mapi • vim
address	このユーザへのレプリケーション・メッセージの送信先のアドレス文字列を、指定したメッセージ・タイプに応じて指定します。
frequency	次のいずれかの文字列。 • **SEND EVERY** *send_time* に指定した頻度でメッセージを送信。 • **SEND AT** *send_time* に指定した時刻にメッセージを送信。

引数	説明
send_time	時刻の文字列。次のいずれかを意味します。 • *frequency* が SEND EVERY の場合、メッセージを送信する間隔。 • *frequency* が SEND AT の場合、メッセージを送信する時刻。 頻度を指定しないと、Message Agent はメッセージ送信後に停止します。

参照

「sp_grant_remote プロシージャ」493 ページ

「sp_revoke_consolidate プロシージャ」535 ページ

「GRANT CONSOLIDATE 文」462 ページ

説明

Adaptive Server Enterprise サーバが SQL Remote インストール環境でリモート・データベースとして動作している場合、**sp_grant_consolidate** プロシージャを使用して、現在のデータベースの上の階層の単一のデータベースに CONSOLIDATE パーミッションを付与します。

統合ユーザはメッセージ・システムにより識別され、統合ユーザに対するメッセージの送受信方法が識別されます。address-name には、メッセージ・システムに対して有効なアドレスを、一重引用符で囲んで指定します。

sp_grant_consolidate プロシージャは、リモート・データベースのメッセージ受信に必要です。ただし、このプロシージャでは、データに対するリモート・ユーザのサブスクリプションは作成されません。データに対するサブスクリプションを作成するには、現在のデータベース内のいずれかのパブリケーションに対して、このユーザ ID のサブスクリプションを作成します。

オプションの引数 *frequency* には、メッセージを送信する頻度を指定します。*send_time* 引数には、メッセージを送信する間隔 (frequency が SEND EVERY の場合)、またはメッセージを送信する時刻 (frequency が SEND AT の場合) を指定します。SEND AT の場合、メッセージは 1 日に 1 回送信されます。

frequency 引数を指定しないと、Message Agent はメッセージを処理した後停止します。Message Agent を継続的に実行するには、リモートまたは CONSOLIDATE パーミッションを持つ各ユーザに、頻度を必ず指定します。

例

- 次の文は、CONSOLIDATE パーミッションをユーザ **hq_user** に付与し、ファイル共有システムを使用し、メッセージをアドレス **hq_dir** に送信するよう設定します。頻度の引数は指定しません。Message Agent はバッチ・モードで実行します。

```
sp_grant_consolidate
     @user_name=hq_user,
     @address=hq_dir,
     @type_name=file
 go
```

sp_grant_remote プロシージャ

内容

SQL Remote 階層で、現在のデータベースのすぐ下にあるデータベースを識別します。このデータベースは現在のデータベースからメッセージを受信します。これらはリモート・ユーザといいます。

構文

sp_grant_remote *user_name*,
type_name,
address
[, *frequency*]
[, *send_time*]

引数	説明
user_name	SQL Remote メッセージを受信できるように設定するユーザ ID。
type_name	使用されるメッセージ・タイプ。次のいずれかを指定します。 • file • ftp • smtp • mapi • vim
address	このユーザへのレプリケーション・メッセージの送信先のアドレス文字列を、指定したメッセージ・タイプに応じて指定します。
frequency	次のいずれかの文字列。 • **SEND EVERY** *send_time* に指定した頻度でメッセージを送信。 • **SEND AT** *send_time* に指定した時刻にメッセージを送信。

引数	説明
send_time	オプション。時刻の文字列。次のいずれかを意味します。 • *frequency* が SEND EVERY の場合、メッセージを送信する間隔。 • *frequency* が SEND AT の場合、メッセージを送信する時刻。 頻度を指定しないと、Message Agent はメッセージ送信後に停止します。

参照

「sp_revoke_remote プロシージャ」536 ページ

「GRANT REMOTE 文」464 ページ

説明

SQL Remote インストール環境では、**sp_grant_remote** プロシージャを使用して、現在のデータベースからのメッセージを受信する各データベースに REMOTE パーミッションを付与します。

リモート・ユーザはメッセージ・システムにより識別され、統合ユーザに対するメッセージの送受信方法が識別されます。address-name には、メッセージ・システムに対して有効なアドレスを、一重引用符で囲んで指定します。

sp_grant_remote プロシージャは、リモート・データベースのメッセージ受信に必要です。ただし、このプロシージャでは、データに対するリモート・ユーザのサブスクリプションは作成されません。データに対するサブスクリプションを作成するには、現在のデータベース内のいずれかのパブリケーションに対して、このユーザ ID のサブスクリプションを作成します。

オプションの引数 *frequency* には、メッセージを送信する頻度を指定します。*send_time* 引数には、メッセージを送信する間隔 (frequency が SEND EVERY の場合)、またはメッセージを送信する時刻 (frequency が SEND AT の場合) を指定します。SEND AT の場合、メッセージは 1 日に 1 回送信されます。

frequency 引数を指定しないと、Message Agent はメッセージを処理した後停止します。Message Agent を継続的に実行するには、REMOTE パーミッションを持つ各ユーザに、頻度を必ず指定します。

統合データベースの多くでは、すべてのリモート・データベースに *frequency* 引数が指定されていれば、Message Agent が継続的に実行すると予想されます。ラップトップ・ユーザに毎日 (SEND AT)、リモート・サーバに 1、2 時間ごとに (SEND EVERY) メッセージを送信するというのが、一般的な設定です。効率的に運用するためには、この設定の時間をできるだけ変えずに使用してください。

例

- 次の文は、リモート・パーミッションをユーザ **SamS** に付与し、MAPI E-mail システムを使用し、アドレス **Singer, Samuel** へ 2 時間ごとにメッセージを送信するよう設定します。

```
exec sp_grant_remote 'SamS',
     'mapi',
     'Singer, Samuel',
     'SEND EVERY',
     '02:00'
 go
```

sp_link_option プロシージャ

内容

SQL Remote メッセージ・リンク用のメッセージ制御パラメータを設定します。

構文

sp_link_option *link-name*,
userid,
option-name,
option-value

パラメータ

link-name:
file | **ftp** | **mapi** | **smtp** | **vim**

link-option-name:
file-option | *ftp-option* | *mapi-option* | *smtp-option* | *vim-option*

file-option:
debug | **directory** | **unlink_delay**

ftp-option:
active_mode | **debug** | **host**
| **password** | **port** | **root_directory** | **user**

mapi-option:
debug | **force_download**
| **ipm_receive** | **ipm_send** | **profile**

smtp-option:
debug | **local_host** | **pop3_host** | **pop3_password**
| **pop3_userid** | **smtp_host** | **top_supported**

vim-option:
debug | **password** | **path** | **receive_all**
| **send_vim_mail** | **userid**

link-option-value:
string

パーミッション

DBA 権限が必要です。パブリッシャは自分自身のオプションを設定できます。

関連する動作

オートコミット。

参照

「SET REMOTE OPTION 文」473 ページ

説明

メッセージ・リンクの初回使用時に、ユーザがメッセージ・リンク・パラメータをメッセージ・リンクのダイアログ・ボックスで入力すると、Message Agent はそのパラメータを保存します。この場合、このプロシージャを明示的に使用する必要はありません。このプロシージャは、多数のデータベースの抽出用に統合データベースを準備する場合に、最も有効です。

オプション名では、大文字と小文字が区別されます。オプションの値で大文字と小文字が区別されるかどうかは、オプションによって異なります。ブール値の場合、大文字と小文字は区別されません。パスワード、ディレクトリ名、その他の文字列では、ファイル・システム (ディレクトリ名の場合) またはデータベース (ユーザ ID とパスワードの場合) で大文字と小文字を区別するかどうかに従います。

userid　*userid* を指定しない場合、現在のパブリッシャが使用されます。

オプション値　オプション値は、メッセージリンクによって異なります。詳細については、次を参照してください。

- 「FILE メッセージ・システム」276 ページ
- 「FTP メッセージ・システム」278 ページ
- 「MAPI メッセージ・システム」283 ページ
- 「SMTP メッセージ・システム」280 ページ
- 「VIM メッセージ・システム」285 ページ

例

次の文は、ユーザ myuser 用の ftp リンクに対し、FTP ホストを *ftp.mycompany.com* に設定します。

```
exec sp_link_option ftp, myuser,
     host, 'ftp.mycompany.com'
```

sp_modify_article プロシージャ

内容

プロシージャ内でアーティクルの記述を変更します。

構文

sp_modify_article
publication_name,
table_name,
[*where_expr*,]
[*subscribe_by_expr*]
[*subscribe_by_view*]

引数	説明
publication_name	アーティクルを修正するパブリケーションの名前。
table_name	アーティクルがあるテーブル。
where_expr	オプションの引数。カラム名または NULL を指定。指定されたカラム名が NULL でないローだけが、パブリケーションに含まれます。 デフォルト値は NULL です。この場合、パブリケーションにはすべてのローが含まれます。
subscribe_by_expr	サブスクリプションごとにパブリケーションに含めるローを定義する、新しいサブスクリプション式。 デフォルト値は NULL です。
subscribe_by_view	パブリケーションに含めるローとカラムを定義するビュー。デフォルト値は NULL です。 詳細については、「抽出パフォーマンスのチューニング」197 ページと「共有ローの抽出パフォーマンスのチューニング」207 ページを参照してください。

参照

「sp_add_article プロシージャ」480 ページ

「sp_remove_article プロシージャ」532 ページ

『ASA SQL リファレンス・マニュアル』> 「ALTER PUBLICATION 文」

説明

パブリケーション内のアーティクルの記述を変更します。WHERE 式、サブスクリプション式、サブスクリプション・ビューをそれぞれ変更できます。

他のデータ定義変更と同様に、運用環境では、クワイエット SQL Remote インストール環境でのみ、このプロシージャを実行してください。

クワイエット・システム要件の詳細については、「スキーマの変更」355 ページを参照してください。

例

次の文は、**Customer** テーブルから情報を取得する **SalesRepData** パブリケーション内のアーティクルを、サブスクリプション式を持たないように変更します。

```
sp_modify_article SalesRepData, Customer
 go
```

次の文は、**Customer** テーブルから情報を取得する **SalesRepData** パブリケーション内のアーティクルを、**rep_key** カラムのサブスクリプション式を持つように変更します。

```
sp_modify_article SalesRepData,     Customer,
     NULL, rep_key
 go
```

sp_modify_remote_table プロシージャ

内容

SQL Remote レプリケーション用にマーク付けされたテーブルの解決オブジェクトを変更します。

構文

sp_modify_remote_table *table_name*,
[*resolve_name*,]
[*old_row_name*,]
[*remote_row_name*]

引数	説明
table_name	SQL Remote レプリケーション用にマーク付けされたテーブル。
resolve_procedure	競合が発生した場合にアクションを実行する、新しいストアド・プロシージャの名前。
old_row_name	競合が発生した場合に値を保持する、新しいテーブルの名前。
remote_row_name	競合の原因となる UPDATE 文が適用された場合に、リモート・データベースで値を保持する、新しいテーブルの名前。

参照

「sp_add_remote_table プロシージャ」484 ページ

「sp_remove_remote_table プロシージャ」534 ページ

「競合の管理」210 ページ

説明

データベースの各テーブルは、**sp_add_remote_table** を使用してレプリケーション用にマーク付けしてから、SQL Remote パブリケーションに含めます。

sp_modify_remote_table を使用すると、このテーブルで発生した更新の競合に対して競合解決を実行する方法を変更できます。

引数にはテーブル名のほかに、カスタム競合解決に必要なオブジェクトの名前があります。カスタム競合解決を実装する場合は、2 つのテーブルとストアド・プロシージャの名前を指定します。**sp_modify_remote_table** プロシージャでは、競合解決の引数で指定さ

れたオブジェクトが存在するかどうかはチェックされません。オブジェクトを作成するのは、テーブルをレプリケーション用にマーク付けする前でも、マーク付けした後でも、どちらでも構いません。

2 つのテーブルには、*table_name* で指定したテーブルと同じカラムとデータ型がなくてはなりません。

例

次の文は、SQL Remote に対して、resolve_Cust プロシージャ、old_Cust テーブル、remote_Cust テーブルを使用して **Customer** テーブル上の更新の競合を解決するように設定します。

```
sp_modify_remote_table Customer, resolve_Cust,
     old_Cust, remote_Cust
 go
```

sp_passthrough プロシージャ

内容

パススルー・モードで SQL 文を送信します。

構文

sp_passthrough *statement*

引数	説明
statement	パススルー・モードで実行する文の文字列。

参照

「sp_passthrough_piece プロシージャ」504 ページ

「sp_passthrough_stop プロシージャ」506 ページ

「sp_passthrough_subscription プロシージャ」507 ページ

「sp_passthrough_user プロシージャ」509 ページ

「PASSTHROUGH 文」467 ページ

説明

パススルー・オペレーションを送信します。パススルー文の受信者は、事前の **sp_passthrough_user** と **sp_passthrough_subscription** の呼び出しによって決まります。

文字列は、最大 255 文字までとします。255 文字を超える SQL 文の場合は、**sp_passthrough_piece** プロシージャの一連の呼び出しを実行し、文の最後の部分に対して **sp_passthrough** を実行してレプリケーションを発生させます。

警告
パススルー・オペレーションは常に、リモート・データベースのサブスクリプションが作成されているテスト・データベース上でテストしてください。テストを行っていないパススルー・スクリプトを、運用データベースに対して実行しないでください。

例

- 次の文は、create table 文をパススルー文の現在の受信者に送信します。

```
exec sp_passthrough
     'CREATE TABLE simple (
         id integer NOT NULL,
         name char(50) )'
 go
```

sp_passthrough_piece プロシージャ

内容

パススルー用に長い SQL 文を構築します。

構文

sp_passthrough_piece *string*

引数	説明
string	パススルー・モードで実行する文の一部分。

参照

「sp_passthrough プロシージャ」502 ページ

「sp_passthrough_stop プロシージャ」506 ページ

「sp_passthrough_subscription プロシージャ」507 ページ

「sp_passthrough_user プロシージャ」509 ページ

「PASSTHROUGH 文」467 ページ

説明

「sp_passthrough プロシージャ」502 ページは、リモート・ユーザの集合に文を直接送信するために使用します。255 文字を超える文は、部分ごとに作成する必要があります。

長い SQL 文を作成して送信するには、文の最後の部分以外に対して **sp_passthrough_piece** を呼び出してから、最後の部分に対して **sp_passthrough** を呼び出します。これで完了し、文がレプリケートされます。

パススルー文の構成要素はすべて 1 つのトランザクション内で作成してください。

例

- 次の文は、長いパススルー文を、現在のパススルー受信者のリストに送信します。

```
begin transaction
 go
 exec sp_passthrough_piece 'CREATE TABLE
     DBA.employee
     (
     emp_id  integer NOT NULL,
```

```
    manager_id  integer NULL,
    emp_fname  char(20) NOT NULL,
    emp_lname  char(20) NOT NULL,'
go
exec sp_passthrough_piece '
    dept_id  integer NOT NULL,
    street  char(40) NOT NULL,
    city  char(20) NOT NULL,
    state  char(4) NOT NULL,
    zip_code  char(9) NOT NULL,
    phone  char(10) NULL,'
go
exec sp_passthrough_piece 'status  char(1) NULL,
    ss_number  char(11) NOT NULL,
    salary  numeric(20,3) NOT NULL,
    start_date  date NOT NULL,
    termination_date  date NULL,
    birth_date  date NULL,'
go
exec sp_passthrough '
    bene_health_ins  char(1) NULL,
    bene_life_ins  char(1) NULL,
    bene_day_care  char(1) NULL,
    sex  char(1) NULL,
    PRIMARY KEY (emp_id),
    )'
go
commit
go
```

sp_passthrough_stop プロシージャ

内容

パススルー・モードをリセットします。

構文

sp_passthrough_stop

参照

「sp_passthrough プロシージャ」502 ページ

「sp_passthrough_subscription プロシージャ」507 ページ

「sp_passthrough_user プロシージャ」509 ページ

「PASSTHROUGH 文」467 ページ

説明

sp_passthrough_stop プロシージャを使用して、パススルー文の受信者リストを空にリセットし、現在構築中の文をすべてクリアします。

例

- 次の文は、パススルー受信者のリストを空にリセットします。

```
exec sp_passthrough_stop
 go
```

sp_passthrough_subscription プロシージャ

内容

指定したパブリケーションに対するサブスクライバを、パススルー文の受信者リストに追加します。

構文

sp_passthrough_subscription *publication_name*,
subscribe_by

引数	説明
publication_name	パブリケーションの名前。
subscribe_by	受信者がパススルー文を受信するためのサブスクリプション値。

参照

「sp_passthrough プロシージャ」502 ページ

「sp_passthrough_piece プロシージャ」504 ページ

「sp_passthrough_stop プロシージャ」506 ページ

「sp_passthrough_user プロシージャ」509 ページ

「PASSTHROUGH 文」467 ページ

説明

パススルー文の受信者リストに追加する方法は 2 つあります。このプロシージャはそのうちの 1 つで、もう 1 つは、「sp_passthrough_user プロシージャ」509 ページです。

sp_passthrough_subscription プロシージャを呼び出すと、サブスクリプション値 *subscribe_by* でパブリケーション *publication_name* に対しサブスクリプションを作成している全ユーザが、受信者リストに追加されます。

subscribe_by のデフォルト値は NULL です。この場合、パブリケーションに対する全サブスクライバがパススルー文を受信します。

例

- 次の文は、サブスクリプション値「rep1」を使用する、**SalesRepData** パブリケーションに対するサブスクライバを、パススルー受信者のリストに追加します。

```
sp_passthrough_subscription SalesRepData, rep1
```

sp_passthrough_user プロシージャ

内容

パススルー文の受信者リストに、指定されたユーザを追加します。

構文

sp_passthrough_user *user_name*

引数	説明
user_name	受信者リストに追加するユーザ。

参照

「sp_passthrough プロシージャ」502 ページ

「sp_passthrough_piece プロシージャ」504 ページ

「sp_passthrough_stop プロシージャ」506 ページ

「sp_passthrough_subscription プロシージャ」507 ページ

「PASSTHROUGH 文」467 ページ

説明

パススルー文の受信者リストに追加する方法は 2 つあります。このプロシージャはそのうちの 1 つで、もう 1 つは、「sp_passthrough_subscription プロシージャ」507 ページです。

sp_passthrough_user プロシージャは、名前付きユーザをパススルー文の受信者リストに追加します。「sp_passthrough_stop プロシージャ」506 ページを使ってリセットするまで、このリストは有効なままです。

例

- 次の文は、ユーザ field_user をパススルー文の受信者リストに追加します。

```
sp_passthrough_user 'field_user'
 go
```

sp_populate_sql_anywhere プロシージャ

内容

TEMPDB に、Adaptive Server Anywhere システム・テーブルのコピーを作成します。このプロシージャは、抽出ユーティリティ *ssxtract* が使用します。

構文

sp_populate_sql_anywhere

説明

リモート Adaptive Server Anywhere データベース用に、Adaptive Server Anywhere システム・テーブルのセットを TEMPDB に作成します。この情報を抽出ユーティリティが使用して、Adaptive Server Enterprise 統合データベース内のパブリケーションのセットから、Adaptive Server Anywhere データベース・スキーマを構築します。

このプロシージャは、*ssxtract* 抽出ユーティリティが使用します。直接呼び出さないでください。

sp_publisher プロシージャ

内容

現在のデータベースのパブリッシャの設定または削除を行います。

構文

sp_publisher [*user_name*]

引数	説明
user_name	データベースのパブリッシャであることを識別するためのユーザ ID。

参照

「SQL Remote パーミッションの管理」256 ページ

「GRANT PUBLISH 文」463 ページ

説明

SQL Remote インストール環境内の各データベースは、出力メッセージ内でユーザ ID により識別され、「**パブリッシャ**」と呼ばれます。**sp_publisher** プロシージャは、これらの出力メッセージに関連付けられたパブリッシャ・ユーザ ID を設定します。

各データベースに設定できるパブリッシャは 1 つだけです。パブリッシャがすでに存在している場合は、**sp_publisher** によってパブリッシャの名前が変更されます。

user_name 引数を指定しないと、現在のパブリッシャは削除され、データベースのパブリッシャは指定されません。パブリッシャとなるパーミッションが削除されるだけで、ユーザ ID はデータベースから削除されません。

例

- 次の例は、ユーザ ID **joe** を現在のデータベースのパブリッシャとして識別する文です。

```
sp_publisher joe
 go
```

- 次の文は、現在のデータベースにパブリッシャがまったくない状態に設定します。

```
sp_publisher
 go
```

sp_queue_clean プロシージャ

内容

このプロシージャは SQL Remote Message Agent が使用します。直接呼び出さないでください。

構文

sp_queue_clean

説明

このプロシージャは SQL Remote Message Agent が使用します。直接呼び出さないでください。このプロシージャは、前回のログのスキャン時に完了していなかった最も古いトランザクションの開始後に完了したトランザクションをすべて、ステーブル・キューから削除します。

sp_queue_confirmed_delete_old プロシージャ

内容

このプロシージャは SQL Remote Message Agent が使用します。直接呼び出さないでください。

構文

sp_queue_confirmed_delete_old

説明

このプロシージャは SQL Remote Message Agent が使用します。直接呼び出さないでください。このプロシージャは、オフセットが **sr_confirmed_transaction** に表示されるトランザクションをすべて、ステーブル・キューから削除します。

sp_queue_confirmed_transaction プロシージャ

内容

このプロシージャは SQL Remote Message Agent が使用します。直接呼び出さないでください。

構文

sp_queue_confirmed_transaction *offset*

説明

このプロシージャは SQL Remote Message Agent が使用します。直接呼び出さないでください。このプロシージャは、指定されたオフセットを **sr_confirmed_transaction** に追加します。SQL Remote は、オフセットがこのオフセットと一致するトランザクションをすべて、ステーブル・キューから削除します。

sp_queue_delete_old プロシージャ

内容

このプロシージャは SQL Remote Message Agent が使用します。直接呼び出さないでください。

構文

sp_queue_delete_old

説明

このプロシージャは SQL Remote Message Agent が使用します。直接呼び出さないでください。このプロシージャは、すべてのリモート・データベースで確認されたトランザクションをすべて、ステーブル・キューから削除します。

sp_queue_drop プロシージャ

内容

データベースからステーブル・キュー・オブジェクトを削除します。

構文

sp_queue_drop

参照

「sp_drop_sql_remote プロシージャ」489 ページ

説明

データベースからステーブル・キュー・システム・オブジェクトを削除します。これ以降、このデータベースは SQL Remote ステーブル・キューをサポートしません。

ステーブル・キュー・オブジェクトのうち、**sp_queue_drop** プロシージャだけは削除されません (プロシージャはデータベースから自分自身を削除できません)。ステーブル・キュー・オブジェクトの削除を完了するには、**sp_queue_drop** を、呼び出した後明示的に削除します。

sp_queue_drop プロシージャでは、SQL Remote システムのオブジェクトはデータベースから削除されません。SQL Remote システム・オブジェクトを削除するには、「sp_drop_sql_remote プロシージャ」489 ページを使用します。

例

- 次の文は、データベースからステーブル・キュー・オブジェクトを削除します。

```
sp_queue_drop
 go
 drop procedure sp_queue_drop
 go
```

sp_queue_dump_database プロシージャ

内容

ステーブル・キューが SQL Remote オブジェクトとは別のデータベースにある場合に、メディア障害からの復旧を容易にします。

構文

sp_queue_dump_database

参照

「sp_queue_dump_transaction プロシージャ」518 ページ

「ステーブル・キューのリカバリ問題」353 ページ

説明

ステーブル・キューを別のデータベースに保存すると、2 つのデータベースを一貫したバージョンに復旧する必要があるため、バックアップとリカバリ処理が複雑になります。

通常のリカバリでは自動的に 2 つのデータベースが一貫した状態にリストアされますが、メディア障害からのリカバリでは、いくつかの処理が必要です。データベース・ダンプをリストアする場合、ステーブル・キューを一貫したポイントへ復旧することが重要です。**sp_queue_dump_database** プロシージャは、メディア障害からの復旧に役立ちます。このプロシージャは、**dump database** がスキャンされるときに必ず呼び出されます。

このプロシージャはそのまま使用しても何の操作も行いません。ステーブル格納データベースで **dump database** コマンドを発行するように、このストアド・プロシージャを修正できます。

sp_queue_dump_transaction プロシージャ

内容

ステーブル・キューが SQL Remote オブジェクトとは別のデータベースにある場合に、メディア障害からの復旧を容易にします。

構文

sp_queue_dump_transaction

参照

「sp_queue_dump_database プロシージャ」517 ページ

「ステーブル・キューのリカバリ問題」353 ページ

説明

ステーブル・キューを別のデータベースに保存すると、2 つのデータベースを一貫したバージョンに復旧する必要があるため、バックアップとリカバリ処理が複雑になります。

通常のリカバリでは自動的に 2 つのデータベースが一貫した状態にリストアされますが、メディア障害からのリカバリでは、いくつかの処理が必要です。データベース・ダンプをリストアする場合、ステーブル・キューを一貫したポイントへ復旧することが重要です。**sp_queue_dump_transaction** プロシージャは、メディア障害からの復旧に役立ちます。このプロシージャは、**dump transaction** がスキャンされる時に必ず呼び出されます。

このプロシージャはそのまま使用しても何の操作も行いません。ステーブル格納データベースで **dump transaction** コマンドを発行するように、このストアド・プロシージャを修正できます。

sp_queue_get_state プロシージャ

内容

このプロシージャは SQL Remote Message Agent が使用します。直接呼び出さないでください。

構文

sp_queue_get_state

説明

このプロシージャは SQL Remote Message Agent が使用します。直接呼び出さないでください。このプロシージャは、ステーブル・キューの現在の状態の記述を返します。

sp_queue_log_transfer_reset プロシージャ

内容

このプロシージャは SQL Remote Message Agent が使用します。直接呼び出さないでください。

構文

sp_queue_log_transfer_reset

説明

このプロシージャは SQL Remote Message Agent が使用します。直接呼び出さないでください。このプロシージャは、**sr_queue_state** テーブル内のページとロー ID を 0 にリセットします。

sp_queue_read プロシージャ

内容

このプロシージャは SQL Remote Message Agent が使用します。直接呼び出さないでください。

構文

sp_queue_read *start_offset*,
stop_offset

説明

このプロシージャは、ステーブル・キューからトランザクションを読み込みます。Message Agent のみが使用します。

sp_queue_reset プロシージャ

内容

サーバをステーブル・キューが空のポイントにリセットします。

構文

sp_queue_reset

説明

このプロシージャは SQL Remote Message Agent が使用します。運用環境では、直接呼び出さないでください。このプロシージャはステーブル・キュー **sr_transaction** テーブルから全ローを削除し、**sr_queue_state** テーブルをリセットして、新しい SQL Remote セットアップに備えます。

このプロシージャは、開発段階でサーバをリセットするのに便利です。

sp_queue_set_confirm プロシージャ

内容

このプロシージャは SQL Remote Message Agent が使用します。直接呼び出さないでください。

構文

sp_queue_set_confirm *confirm_offset*

説明

このプロシージャは SQL Remote Message Agent が使用します。直接呼び出さないでください。このプロシージャは、**sr_queue_state** テーブル内の全リモート・ユーザからの最小確認オフセットを設定します。

sp_queue_set_progress プロシージャ

内容

このプロシージャは SQL Remote Message Agent が使用します。直接呼び出さないでください。

構文

sp_queue_set_progress *page_id*,
row_id,
commit_offset,
backup_offset,
marker

説明

このプロシージャは SQL Remote Message Agent が使用します。直接呼び出さないでください。このプロシージャは、**sr_queue_state** テーブル内のトランザクション・ログ・スキャン進行値を設定します。

sp_queue_transaction プロシージャ

内容

このプロシージャは SQL Remote Message Agent が使用します。直接呼び出さないでください。

構文

sp_queue_transaction *offset*,
user_id

説明

このプロシージャは SQL Remote Message Agent が使用します。直接呼び出さないでください。このプロシージャは、ステーブル・キューに新しいトランザクションを追加します。

sp_remote プロシージャ

内容

このプロシージャは SQL Remote Message Agent が使用します。この後説明する場合を除いて、直接呼び出さないでください。このプロシージャは、**sr_remoteuser** テーブル内のローを管理します。

構文

sp_remote *operation*,
user_name
[, *offset*]
[, *confirm*]

引数	説明
operation	アクションの名前。ユーザが使用できる値は **reset** のみで、他の値はすべて Message Agent が使用。
user_name	リセットされているリモート・ユーザの名前。
offset	未使用。
confirm	未使用。

説明

このプロシージャは SQL Remote Message Agent が使用します。**reset** 呼び出しを除いて、直接呼び出さないでください。このプロシージャは、**sr_remoteuser** テーブル内のメッセージ・トラッキング情報を管理します。

カスタム・データベース抽出処理の作成時に、特別に次のように直接使用できます。

```
sp_remote reset, remote_user
```

remote_user にはリモート・ユーザの名前を指定します。

このコマンドは、リモート・ユーザのサブスクリプションをすべて、1 つのトランザクション内で開始します。**sr_remoteuser** テーブル内の **log_sent** 値と **confirm_sent** 値を、トランザクション・ログ内の現在の位置に設定します。また、**sr_subscription** 内で作成され開始された値を、このリモート・ユーザの全サブスクリプション用のトランザクション・ログ内の、現在の位置に設定します。このプロシージャはコミットを行いません。この呼び出しの後、明示的にコミットを実行してください。

ライブ・データベース上で安全な抽出処理を書き込むには、サブスクリプションが開始されたのと同じトランザクション内で、データを独立性レベル 3 で抽出します。

sp_remote_option プロシージャ

内容

SQL Remote オプションを設定します。

構文

sp_remote_option *option_name*,
option_value

引数	説明
option_name	SQL Remote オプションの名前 (1 つのみ)。
option_value	オプションに設定する値。

参照

「SQL Remote のオプション」406 ページ

説明

SQL Remote オプションで、レプリケーションの動作を制御します。Adaptive Server Enterprise では、次のオプションが使用可能です。

オプション	値	デフォルト
Blob_threshold	**整数** (単位 **K**)	**256**
Compression	**-1 ～ 9**	**6**
Delete_old_logs	**ON**、**OFF**	**OFF**
Qualify_owners	**ON**、**OFF**	**ON**
Quote_all_identifiers	**ON**、**OFF**	**OFF**
Replication_error	*プロシージャ名*	**NULL**
SR_Date_Format	*時刻文字列*	**hh:mm:ss.Ssssss**
SR_Time_Format	*日付文字列*	**yyyy/mm/dd**
SR_Timestamp_Format	*タイムスタンプ文字列*	**yyyy/mm/dd hh:nn:ss.Ssssss**
Subscribe_by_remote	**ON**、**OFF**	**ON**
Verify_threshold	*整数*	**256**

オプション	値	デフォルト
Verify_all_columns	**ON、OFF**	**OFF**

これらのオプションの完全なリストについては、「SQL Remote のオプション」406 ページを参照してください。

例

- 次の文は、Verify_all_columns オプションを OFF に設定します。この結果、Message Agent が適用した update 文の古い値は、全カラムに対して自動的にはチェックされません。

```
sp_remote_option Verify_all_columns, OFF
 go
```

sp_remote_type プロシージャ

内容

SQL Remote メッセージ・タイプの作成または修正を行います。

構文

sp_remote_type *type_name publisher_address*

引数	説明
type_name	作成または変更するメッセージ・タイプ。次のいずれかを指定。 • file • ftp • smtp • mapi • vim
publisher_address	指定したメッセージ・タイプに応じた、パブリッシャのアドレス。

参照

「sp_drop_remote_type プロシージャ」488 ページ

「ALTER REMOTE MESSAGE TYPE 文」452 ページ

説明

Message Agent は、サポートされたメッセージ・リンクのうちの 1 つを使用して、データベースから出力メッセージを送信します。指定されたリンクを使用しているユーザへのリターン・メッセージは、リモート・データベースが抽出ユーティリティで作成されているかぎり、指定されたアドレスへ送信されます。Message Agent は、リンク用のリモート・ユーザがある場合にのみ、それらのリンクを開始します。

アドレスには、指定したメッセージ・システムに応じた、パブリッシャのアドレスを指定します。電子メール・システムの場合、アドレス文字列には有効な電子メール・アドレスを指定します。ファイル共有システムの場合、アドレス文字列には、SQLREMOTE 環境変数またはレジストリ・エントリに設定されたディレクトリのサブディレクトリか、設定されていない場合は現在のディレクトリのサブディレクトリを指定します。

例

次は、データベース用に FILE メッセージ・タイプを作成し、パブリッシャのアドレスとして、*publisher* という名前の SQLREMOTE ロケーションのサブディレクトリを指定する例です。

```
sp_remote_type file, publisher
 go
```

sp_remove_article プロシージャ

内容

パブリケーションからアーティクルを削除します。

構文

sp_remove_article *publication_name*,
table_name

引数	説明
publication_name	アーティクルを削除するパブリケーションの名前。
table_name	アーティクルがあるテーブル。

参照

「sp_add_article プロシージャ」480 ページ

『ASA SQL リファレンス・マニュアル』> 「ALTER PUBLICATION 文」

説明

sp_remove_article を実行すると、パブリケーションからアーティクルを削除できます。指定されたテーブルの一部を含んでいるアーティクルはすべて、パブリケーションから削除されます。

例

- 次の文は、**SalesRep** テーブルの一部を使用するアーティクルをすべて、**SalesRepData** という名前のパブリケーションから削除します。

```
sp_remove_article SalesRepData, SalesRep
 go
```

sp_remove_article_col プロシージャ

内容

パブリケーション内のアーティクルからカラムを削除します。

構文

sp_remove_article_col *publication_name*,
article_name,
column_name

引数	説明
publication_name	アーティクルが属するパブリケーションの名前。
article_name	カラムを削除するアーティクル。
column_name	アーティクルから削除するカラム。

参照

「sp_add_article_col プロシージャ」482 ページ

「sp_remove_article プロシージャ」532 ページ

『ASA SQL リファレンス・マニュアル』> 「ALTER PUBLICATION 文」

説明

sp_remove_article_col を使用して、パブリケーションからカラムを削除できます。

sp_remove_article_col を使用してカラムを削除するには、「sp_add_article_col プロシージャ」482 ページを使用して、パブリケーションにそのカラムを明示的に追加しておく必要があります。sp_add_article_col ではなく 「sp_add_article プロシージャ」480 ページを使用すると、テーブル内のすべてのカラムを一括してパブリケーションに追加できます。ただし、そのパブリケーションから、sp_remove_article_col を使用して特定のカラムだけを削除することはできません。

例

- 次の文は、employee テーブルの emp_lname カラムを、Personnel という名前のパブリケーションから削除します。

```
sp_remove_article_col 'Personnel', 'employee',
'emp_lname'
 go
```

sp_remove_remote_table プロシージャ

内容

SQL Remote レプリケーション用に使用できないものとして、テーブルをマーク付けします。

構文

sp_remove_remote_table *table_name*

引数	説明
table_name	SQL Remote レプリケーション用に使用できないものとしてマーク付けするテーブル。

参照

「sp_add_remote_table プロシージャ」484 ページ

「sp_modify_remote_table プロシージャ」500 ページ

説明

レプリケーション用に使用できないものとして、テーブルをマーク付けします。このプロシージャを呼び出した後は、このテーブル内のデータを SQL Remote を使用している他のデータベースと共有することはできません。

例

- 次の文は、**employee** テーブルをレプリケーション用に使用できないものとしてマーク付けします。

```
sp_remove_remote_table employee
 go
```

sp_revoke_consolidate プロシージャ

内容

ユーザがこのデータベースからの SQL Remote メッセージを受信できないように設定します。

構文

sp_revoke_consolidate *user_name*

引数	説明
user_name	統合データベースとして動作できないように設定するユーザ ID。

参照

「sp_grant_consolidate プロシージャ」490 ページ

説明

sp_revoke_consolidate プロシージャは、現在のデータベースからのメッセージを受信するユーザのリストから、統合データベース・ユーザ ID を削除します。

例

- 次の文は、ユーザ **hq_user** の CONSOLIDATE パーミッションを取り消します。

```
sp_revoke_consolidate hq_user
 go
```

sp_revoke_remote プロシージャ

内容

ユーザがこのデータベースからの SQL Remote メッセージを受信できないように設定します。

構文

sp_revoke_remote *user_name*

引数	説明
user_name	SQL Remote メッセージを受信できないように設定するユーザ ID。

参照

「sp_grant_remote プロシージャ」493 ページ

説明

sp_revoke_remote プロシージャは、現在のデータベースからメッセージを受信するユーザのリストから、ユーザ ID を削除します。

例

- 次の文は、ユーザ **Field User** のリモート・パーミッションを取り消します。

```
sp_revoke_remote 'Field user'
 go
```

sp_subscription プロシージャ

内容

サブスクリプションを管理します。

構文

sp_subscription *operation*,
publication_name,
user_name,
[*subscribe_by*]

引数	説明
operation	実行する操作。次のいずれかを指定。 • **create**　指定されたパブリケーションに対するユーザのサブスクリプションを作成。 • **drop**　指定されたパブリケーションに対するユーザのサブスクリプションを削除。 • **start**　名前付きパブリケーションに対するサブスクリプションを開始。 • **stop**　名前付きパブリケーションに対するサブスクリプションを停止。 • **synchronize**　名前付きパブリケーションに対するサブスクリプションを同期。
publication_name	サブスクリプションが参照するパブリケーションの名前。
user_name	パブリケーションに対するサブスクリプションを作成されているユーザ ID。
subscribe_by	サブスクリプション値。

参照

「サブスクリプションの作成」229 ページ

説明

sp_subscription プロシージャを使用して、サブスクリプションを管理します。このプロシージャの最初の引数 (*operation*) には、プロシージャの作成、削除、開始、停止、同期のいずれを行うかを指定します。

通常、サブスクリプションの開始と同期は、抽出ユーティリティを使用して行われます。

例

- 次の文は、ユーザ **SalesRep1** のサブスクリプションを、**SalesRepData** パブリケーションに対して作成します。サブスクリプション式はありません。

```
sp_subscription create,
     SalesRepData,
     SalesRep1
 go
```

sp_subscription_reset プロシージャ

内容　全リモート・ユーザの全 SQL Remote 情報をリセットします。

構文　**sp_subscription_reset**

説明　このプロシージャは、**sr_remote_user** テーブルと **sr_subscription** テーブル内の全エントリを、0 または NULL にリセットします。

第 5 部　付録

付録では、アプリケーションを日常的に使用するときには必要のない補足情報について説明します。

付録 A

SQL Remote for Adaptive Server Enterprise と SQL Remote for Adaptive Server Anywhere の違い

この付録の内容

この付録では、SQL Remote for Adaptive Server Enterprise と SQL Remote for Adaptive Server Anywhere との違いについて概略を説明します。

ここでは、これらのテクノロジのバージョン間の主な違いについて説明します。

差異のタイプ

ソフトウェアのバージョン間には、次の違いがあります。

- **機能性** 2 つのバージョンのうち、一方では実行できるのに、もう片方では実行できないタスクがあります。
- **アプローチ** 同様の結果が得られるのに、各バージョンで異なるアプローチが必要となる場合があります。見かけ上異なる実行方法が複数あるのに、結果は同じになるタスクも含まれます。
- **サーバ間の差異** バックアップの管理など SQL Remote に関連するタスクでは、2 つのサーバに対するタスクが異なります。これらの違いについては、ここでは説明しません。

この付録では、Adaptive Server Anywhere をリモート・データベースとして使用するレプリケーションについてのみ説明します。Adaptive Server Enterprise をリモート・サーバとして使用する場合は、制限事項が追加されます。

機能性の違い

SQL Remote for Adaptive Server Enterprise (SRE) と SQL Remote for Adaptive Server Anywhere (SRA) との主な違いは次のとおりです。

- **スキーマの変更** SRE では、スキーマの変更を必ず「**クワイエット**」システム上で行います。クワイエット・システムとは、次のことを意味します。
  - **トランザクションはレプリケートされない** 変更対象のテーブルを修正するトランザクションがレプリケートされることはありません。テーブルを変更するすべてのトランザクションがトランザクション・ログからスキャンされ、ステーブル・キューに保存されてから、スキーマを変更します。この操作は、Message Agent を通常どおり実行するか、`-I -b` オプションを使用して行います。Message Agent の完了後、スキーマを変更できます。
  - **Message Agent を停止する** スキーマを変更するときは、Message Agent を停止します。
  - **SQL Remote Open Server** SQL Remote Open Server の使用中にスキーマを変更する場合には、その間、SQL Remote Open Server を停止します。
- **トリガ・アクションのレプリケーション** SRE では、トリガ・アクションがレプリケートされます。SRA の場合は、トリガ・アクションをレプリケートするかどうかを選択できます。デフォルト設定ではレプリケートされません。トリガ・アクションのレプリケーションを行うとき、SRE のユーザはトリガがリモート・データベースで起動されていないことを確認してください。
- **プラットフォームの可用性** SRA は SRE に比べ、より広範な種類のプラットフォームで実行できます。このことは 2 つのサーバのプラットフォーム可用性を反映しています。

- **パブリケーション定義**　SRA でのパブリケーションは、SRE に比べ選択の幅がより広くなります。たとえば、SRA では WHERE 句にどのような値を使用してもかまいません。SRE の場合、WHERE 句で使用できる値は IS NULL 条件と IS NOT NULL 条件だけです。

アプローチの違い

SQL Remote の機能の中には、SRE と SRA では異なる方法でアプローチしなければならないものがあります。

- **サブスクリプション式を含まないテーブルの分割** SRA では、パブリケーションにサブクエリを含むことができます。したがって、テーブルに分割式がなくても、サブスクライバ間に適切に配信されます。SRE では、分割式のないテーブルにカラムを追加してください。カラムにはサブスクライバのリストを含め、カラムを管理するためにトリガを記述してください。このカラムのサイズは最大 255 文字です。

  詳細については、「サブスクリプション式を含まないテーブルの分割」132 ページと「サブスクリプション・カラムを含まないテーブルの分割」190 ページを参照してください。

- **競合解決** SRA では、競合解決は特別なトリガ構文を使用して実行されます。SRE でこのタスクを実行するにはストアド・プロシージャを記述してください。

  詳細については、「競合の管理」151 ページと「競合の管理」210 ページを参照してください。

- **送信前にメッセージを格納する** SRE では、「**ステーブル・キュー**」というテーブルを別に使用して変更内容を保管してからレプリケーションが実行されます。SRA では、ステーブル・キューがないので、そのかわりに現在使用しているトランザクション・ログのファイルや古いファイルからメッセージが検索されます。

- **コマンド** パブリケーションを作成するなどの SQL Remote のタスクは、SRA では SQL 文を使用して実行されますが、SRE では、システム・ストアド・プロシージャを使用して実行されます。

Adaptive Server Enterprise のプロシージャと Adaptive Server Anywhere の文

SQL Remote for Adaptive Server Anywhere では、SQL 文はタスクを実行するために使用されます。このタスクは、ストアド・プロシージャが Adaptive Server Enterprise で実行します。次の表に、SQL Remote のプロシージャと、対応する Adaptive Server Anywhere の SQL 文を示します。

Adaptive Server Enterprise のプロシージャ	対応する Adaptive Server Anywhere の文
sp_remote_type	CREATE REMOTE MESSAGE TYPE
sp_remote_type	ALTER REMOTE MESSAGE TYPE
sp_drop_remote_type	DROP REMOTE MESSAGE TYPE
sp_grant_remote	GRANT REMOTE
sp_revoke_remote	REVOKE REMOTE
sp_publisher	GRANT PUBLISH
sp_publisher	REVOKE PUBLISH
sp_create_publication	CREATE PUBLICATION
sp_add_article	
sp_add_article_col	
sp_add_article	ALTER PUBLICATION
sp_remove_article	
sp_add_article_col	
sp_remove_article_col	
sp_drop_publication	DROP PUBLICATION
sp_subscription 'create'	CREATE SUBSCRIPTION
sp_subscription 'drop'	DROP SUBSCRIPTION

Adaptive Server Enterprise のプロシージャ	対応する Adaptive Server Anywhere の文
sp_subscription 'start'	START SUBSCRIPTION
sp_subscription 'stop'	STOP SUBSCRIPTION
sp_subscription 'synchronize'	SYNCHRONIZE SUBSCRIPTION
sp_passthrough_user	PASSTHROUGH FOR USERID
sp_passthrough_subscription	PASSTHROUGH FOR SUBSCRIPTION
sp_passthrough_stop	PASSTHROUGH STOP

Enterprise 間のレプリケーションの制限

Adaptive Server Enterprise のデータベースと Adaptive Server Anywhere のリモート・データベース間ではなく、Adaptive Server Enterprise のデータベース間で SQL Remote を使ってレプリケートする場合、次の制限事項に注意してください。

- **データベースの抽出** 抽出ユーティリティは、Adaptive Server Anywhere のリモート・データベースを構築するために RELOAD.SQL スクリプトとデータ・ファイルを作成します。リモート ASE データベースのセットアップには、顧客が作成した抽出プロセスが必要です。

  抽出プロセスの作成については、「sp_remote プロシージャ」526 ページを参照してください。

- **参照整合性エラー** Adaptive Server Enterprise では、参照整合性は常時迅速に確認されます。一方、Adaptive Server Anywhere では、参照整合性の確認時に制御を行う WAIT_FOR_COMMIT オプションが用意されています。これにより、領域の再編成を行うときなどに、リモート・データベース間でローを移動することが困難になります。

  たとえば、**Order** テーブルが **Customer** テーブルに外部キーを持っていて、その Customer テーブルが **SalesRep** テーブルに外部キーを持っているとします。Customer テーブルは、営業担当者によってサブスクリプションが作成されています。**Order** テーブルも営業担当者によってサブスクリプションが作成されています (テーブルはトリガによって維持される冗長カラムを持っています)。

  Customer テーブルのローが新しい営業担当者を指すように更新された場合、トリガが起動して **Order** テーブルの営業担当者のカラムを更新します。**Customer** テーブルの更新は、古い営業担当者の削除と新しい担当者の挿入としてレプリケートされます。同様に、トリガが起動した **Order** テーブルの更新も、古い営業担当者の削除と新しい担当者の挿入としてレプリケートされます。

SQL Remote は、発生順に操作をレプリケートするので問題が生じます。つまり、**Order** のローが削除される前に **Customer** のローが削除されてしまいます。このため参照整合性エラーが発生します。

- **スキーマのアップグレード** 統合データベースとリモート・データベースの両方が Adaptive Server Enterprise のデータベースである場合、スキーマのアップグレードは管理が難しくなります。リモート Adaptive Server Enterprise データベースへのパススルーは実行困難です。

  この問題の原因は、クワイエット・システムにスキーマのアップグレードが必要なためです (「機能性の違い」545 ページを参照)。パススルーは、スキーマ・アップグレード文を標準のメッセージ・ストリームに配置します。(同じメッセージまたは前のメッセージにある) スキーマのアップグレードに先だって行うオペレーションは、スキーマの変更が実行される前にトランザクション・ログからステーブル・キューへスキャンされていない可能性があります。

- **サブスクリプションの同期** これは、Adaptive Server Enterprise のリモート・データベースには実装されていません。

付録 B

サポートされるプラットフォームとメッセージ・リンク

この付録の内容

この付録では、SQL Remote がサポートするプラットフォームとメッセージ・リンクを概説します。

サポートされるメッセージ・システム

SQL Remote は、基本となるメッセージ・システムを使用して、データベース間のデータ交換を行います。SQL Remote では、次のメッセージ・システムをサポートしています。

- **ファイル共有**　他のソフトウェアを必要としない簡単なシステム
- **FTP**　インターネット・ファイル転送プロトコル
- **SMTP/POP**　インターネット電子メール転送プロトコル
- **MAPI**　Microsoft Messaging Application Programming Interface。Microsoft 製品と cc:Mail リリース 8 以降で使用されています。
- **VIM**　Vendor Independent Messaging。Lotus Notes と Lotus cc:Mail の一部のバージョンで使用されています。

すべてのシステムがすべてのオペレーティング・システムでサポートされるわけではありません。ファイル共有システム以外のすべてのシステムについては、SQL Remote を機能させるための適切なメッセージ・システム・ソフトウェアを購入してインストールしておく必要があります。SQL Remote には、基本となるメッセージ・システム・ソフトウェアが含まれていません。

サポートされるオペレーティング・システム

SQL Remote for Adaptive Server Enterprise

SQL Remote for Adaptive Server Enterprise は、次のオペレーティング・システムとメッセージ・リンクで使用できます。

- **Windows NT/2000/XP** すべてのメッセージ・プロトコル
- **Sun Microsystems Solaris/Sparc** ファイル共有、FTP、SMTP/POP のみ

SQL Remote for Adaptive Server Anywhere

SQL Remote for Adaptive Server Anywhere は、次のオペレーティング・システムで使用できます。

- **Windows 95/98/Me** すべてのメッセージ・リンク
- **Windows NT/2000/XP** すべてのメッセージ・リンク
- **Windows CE** FILE リンク、FTP リンク、SMTP/POP リンク。ファイル・リンクの場合、*dbremote* は *¥My Documents¥Synchronized Files* を調べます。デスクトップ・マシン上では、FILE リンク用の SQLREMOTE 環境変数または directory メッセージ・リンク・パラメータを次のように設定してください。

  ```
  %SystemRoot%¥Profiles¥userid¥Personal¥ce-machine-name¥
  Synchronized Files
  ```

  ここで、*userid* と *ce-machine-name* に適切な値を設定してください。ActiveSync はこの設定を使用して、デスクトップと CE システム間でメッセージ・ファイルを自動的に同期します。

  [モバイル・デバイス] ― [ツール] ― [ActiveSync] オプションをチェックして、ファイルの同期がアクティブ化されていることを確認します。

  メッセージ・リンク・パラメータの設定方法については、「FILE メッセージ・システム」276 ページを参照してください。

- **Sun Microsystems Solaris/Sparc** ファイル共有、FTP、SMTP/POP のみ
- **Novell NetWare** ファイル共有、FTP、SMTP/POP のみ

- **Linux** ファイル共有、FTP、SMTP/POP のみ

サポートされる UNIX オペレーティング・システムのバージョンについては、SQL Anywhere Studio に付属している『*はじめにお読みください (UNIX 版)*』を参照してください。

索引

記号

#hook_dict テーブル
  dbremote 413
  SQL Remote for ASA のユニークなプライマリ・キー 165
#remote テーブル
  説明 432

A

ActiveSync
  Windows CE レプリケーション 555
Adaptive Server Anywhere
  ASE と互換性のあるデータベースの作成 92
  SQL Remote のチュートリアル 33
  トリガのレプリケート 101
Adaptive Server Anywhere のシステム・オブジェクト
  SQL Remote 419
Adaptive Server Enterprise
  SQL Remote for ASE のアンインストール 30
  SQL Remote の設定 24
  SQL Remote のチュートリアル 67
  トリガのレプリケート 101
Adaptive Server Enterprise のシステム・オブジェクト 431
article
  SQL Remote for ASA のシステム・テーブル 421
  SQL Remote for ASE のシステム・テーブル 433
articlecol SQL Remote テーブル
  ASE 433
  定義 421
articlecols SQL Remote ビュー
  ASA 427
  ASE 442
article SQL Remote テーブル
  ASA 421
  ASE 433
articles SQL Remote ビュー
  ASA 427
  ASE 442

B

BLOB
  レプリケート 103
BLOB_THRESHOLD オプション
  レプリケーション・オプション 407
BLOBs
  レプリケーション 407
-b オプション
  Message Agent 288

C

ccMail
  SQL Remote 269
COMMIT 文
  イベント・フック・プロシージャ 413
  レプリケート 97
COMPRESSION オプション
  レプリケーション・オプション 407
CONSOLIDATE パーミッション
  管理 256
  取り消し 260
  付与 260, 264
contact データベースの例
  SQL Remote for ASE 190
CREATE SUBSCRIPTION 文

SQL Remote for ASA サブスクリプションの作成 175
SQL Remote の使用 251
CREATE 文
レプリケート 102
CURRENT PUBLISHER
システム・テーブル 432
使用 53
レプリケーション 44
CURRENT REMOTE USER
SQL Remote for ASE の競合解決 217
システム・テーブル 432
特殊定数 155

D

dbcc settrunc
使用 353
dbo ユーザ
システム・オブジェクト 389
dbremote
#hook_dict テーブル 413
概要 10
コマンド 376
セキュリティ 313
説明 287, 376
データのレプリケーション 59
dbunload ユーティリティ
レプリケーション 332
dbxtract ユーティリティ
sp_hook_dbxtract_begin プロシージャ 165
SQL Remote for ASA のリモート・データベースの抽出 57
SQL Remote for ASA レプリケーションの使用 50
SQL Remote レプリケーションの使用 241
オプション 394
構文 391
説明 238, 391
DDL
SQL Remote for ASE 355
レプリケート 102
debug 制御パラメータ
FTP メッセージ・タイプ 277, 278
MAPI メッセージ・タイプ 284
SMTP メッセージ・タイプ 281
VIM メッセージ・タイプ 285
DELETE_OLD_LOGS オプション
トランザクション・ログの管理 325
レプリケーション・オプション 408
DELETE 文
レプリケート 97, 98
directory 制御パラメータ
FILE メッセージ・タイプ 277
DROP PUBLICATION 文
SQL Remote for ASA の使用 125
dsi_num_threads
Replication Server のパラメータ 369
dsi_sql_data_style パラメータ
設定 370

E

EXTERNAL_REMOTE_OPTIONS オプション
レプリケーション・オプション 408

F

FILE メッセージ・タイプ
概要 276
使用 269
制御パラメータ 277
説明 276
FIRE_TRIGGERS オプション
トリガ・アクション 102
Force_Download 制御パラメータ
MAPI メッセージ・タイプ 284
FTP メッセージ・タイプ
概要 278
使用 269
制御パラメータ 278
説明 278

トラブルシューティング　279

G

GLOBAL_DATABASE_ID オプション
SQL Remote for ASA の設定　164
GRANT CONSOLIDATE 文
SQL Remote の設定　260
GRANT PUBLISH 文
SQL Remote for ASA の設定　43
SQL Remote の設定　256
統合データベース　53
GRANT REMOTE 文
SQL Remote for ASA の設定　43
SQL Remote の設定　260
統合データベース　53

H

host 制御パラメータ
FTP メッセージ・タイプ　278

I

IF 文
パススルー・モード　338
レプリケーション　338
IMAGE データ型
レプリケート　103
INSERT 文
レプリケート　97, 98
IPM_Receive 制御パラメータ
MAPI メッセージ・タイプ　284
IPM_Send 制御パラメータ
MAPI メッセージ・タイプ　284

L

LONG BINARY データ型
レプリケート　103
LONG VARCHAR データ型
レプリケート　103
LOOP 文
パススルー・モード　338
Lotus Notes
SQL Remote　269, 285
LTM
SQL Remote for ASE　341
-l オプション
Message Agent　291

M

MAPI メッセージ・タイプ
概要　283
使用　269
制御パラメータ　283, 284
説明　283
marker SQL Remote テーブル
ASE　434
Message Agent
-l オプション　291
-m オプション　296
-rd オプション　297
-ro オプション　291
-rp オプション　297, 298
-rt オプション　291
SQL Remote for ASA の出力　315
SQL Remote for ASA のセキュリティ　313
SQL Remote for ASA のデータのレプリケーション　59
SQL Remote for ASE の出力　349
SQL Remote for ASE のセキュリティ　346
SQL Remote for ASE のデータのレプリケーション　83
SQL Remote の管理　254
-u オプション　290
エラーのレポート　315
概要　10
起動　312
継続モード　287

コマンド　376
サービスとしての実行　312
再送要求　297
サブスクリプション処理　108
実行　312
スキーマの変更　355
スレッド　294
セキュリティ　291
接続　289
設定　291
説明　287
デーモン　386
トランザクション・ログの管理　320, 325, 331, 351
トリガのレプリケーション　102
バッチ・モード　287
パフォーマンス　294, 300
ポーリング　297
メッセージ・トラッキング　306, 307, 308
メッセージの配信　306, 308
ユーザ ID　346
ワーカ・スレッド　294
Microsoft Exchange
プロファイル　284
-m オプション
Message Agent　296

N

NCHAR データ型
抽出ユーティリティ　249
NetWare
SQL Remote　277
サポートしている SQL Remote メッセージ・タイプ　269
Notes
SQL Remote　269, 285
Novell NetWare
SQLRemote　555
NVARCHAR データ型
抽出ユーティリティ　249

O

object SQL Remote テーブル
ASE　434
option SQL Remote テーブル
ASE　435
OUTPUT_LOG_SEND_LIMIT リモート・オプション
使用　291
OUTPUT_LOG_SEND_NOW リモート・オプション
使用　291
OUTPUT_LOG_SEND_ON_ERROR リモート・オプション
使用　291

P

passthrough SQL Remote テーブル
ASE　435
PASSTHROUGH 文
使用　335
多層インストール環境　337
password 制御パラメータ
FTP メッセージ・タイプ　278
VIM メッセージ・タイプ　285
Path 制御パラメータ
VIM メッセージ・タイプ　285
permissions
REMOTE　43
policy データベースの例
SQL Remote for ASA パブリケーション　141
pop3_host 制御パラメータ
SMTP メッセージ・タイプ　281
pop3_password 制御パラメータ
SMTP メッセージ・タイプ　281
pop3_userid 制御パラメータ
SMTP メッセージ・タイプ　281
port 制御パラメータ
FTP メッセージ・タイプ　278
publication SQL Remote テーブル
ASA　422
ASE　436

publications SQL Remote ビュー
ASA 428
ASE 443
publisher SQL Remote テーブル
ASE 436
PUBLISH パーミッション
REMOTE パーミッション 43, 53
管理 256
取り消し 256
付与 43, 256

Q

QUALIFY_OWNERS オプション
レプリケーション・オプション 408
QUOTE_ALL_IDENTIFIERS オプション
レプリケーション・オプション 408

R

-rd オプション
Message Agent 297
receive_all 制御パラメータ
VIM メッセージ・タイプ 285
remoteoption SQL Remote テーブル
ASA 422
remoteoptions SQL Remote ビュー
ASA 428
ASE 443
remoteoptiontype SQL Remote テーブル
ASA 423
remotetables SQL Remote ビュー
ASE 443
remotetable テーブル
ASE 437
remotetype SQL Remote テーブル
ASA 423
ASE 438
remotetypes SQL Remote ビュー
ASE 444
remoteuser SQL Remote テーブル
ASA 424
ASE 438
使用 306
メッセージ・トラッキング 307
remoteusers SQL Remote ビュー
ASA 428
ASE 444
Remote パーミッション
Sybase Central 261
管理 256
取り消し 260, 265
付与 260
Remote パーミッションの取り消し 265
REPLICATION_ERROR オプション
SQL エラーのトラッキング 110
エラー処理 349
エラー処理プロシージャ 316
レプリケーション・オプション 408
Replication Agent
SQL Remote 360, 361
SQL Remote for ASE 341
Replication Server
SQL Remote 359, 360, 369, 402
SQL Remote for ASE 341
SQL Remote アーキテクチャ 361
ssqueue 365
接続の再開 371
設定 369
RESOLVE UPDATE トリガ 156
説明 154
REVOKE CONSOLIDATE 文
SQL Remote の設定 260
REVOKE PUBLISH 文 256
REVOKE REMOTE 文 260, 265
ROLLBACK 文
イベント・フック・プロシージャ 413
root 制御パラメータ
FTP メッセージ・タイプ 278
-ro オプション
Message Agent 291
-rp オプション
Message Agent 297, 298
rs_dumpdb
使用 372

rs_dumptran
使用　372
-rt オプション
Message Agent　291

S

salespub.sql
SQL Remote for ASA のサンプル・パブリケーション　65
SAVE_REMOTE_PASSWORDS オプション
レプリケーション・オプション　409
send_vim_mail 制御パラメータ
VIM メッセージ・タイプ　285
SEND AT
頻度の設定　263
SEND EVERY
頻度の設定　263
SMTP/POP
アドレス　283
smtp_authenticate 制御パラメータ
SMTP メッセージ・タイプ　281
smtp_host 制御パラメータ
SMTP メッセージ・タイプ　281
smtp_password 制御パラメータ
SMTP メッセージ・タイプ　282
smtp_userid 制御パラメータ
SMTP メッセージ・タイプ　282
SMTP メッセージ・タイプ
SQL Remote　280
概要　280
使用　269
制御パラメータ　281
説明　280
電子メール　280
sp_add_article_col プロシージャ
構文　482
sp_add_article プロシージャ
構文　480
sp_add_remote_table プロシージャ
構文　484
sp_create_publication プロシージャ
構文　486
sp_drop_publication プロシージャ
構文　487
sp_drop_remote_type プロシージャ
構文　488
sp_drop_sql_remote プロシージャ
SQL Remote for ASE のアンインストール　30
構文　489
sp_grant_consolidate プロシージャ
構文　490
sp_grant_remote プロシージャ
構文　493
sp_hook_dbremote_begin ストアド・プロシージャ
SQL 構文　413
sp_hook_dbremote_end ストアド・プロシージャ
SQL 構文　414
sp_hook_dbremote_message_apply_begin ストアド・プロシージャ
SQL 構文　417
sp_hook_dbremote_message_apply_end ストアド・プロシージャ
SQL 構文　417
sp_hook_dbremote_message_missing ストアド・プロシージャ
SQL 構文　416
sp_hook_dbremote_message_sent ストアド・プロシージャ
SQL 構文　416
sp_hook_dbremote_receive_begin ストアド・プロシージャ
SQL 構文　415
sp_hook_dbremote_receive_end ストアド・プロシージャ
SQL 構文　415
sp_hook_dbremote_send_begin ストアド・プロシージャ
SQL 構文　415
sp_hook_dbremote_send_end ストアド・プロシージャ
SQL 構文　416

sp_hook_dbremote_shutdown ストアド・プロシージャ
SQL 構文 414
sp_hook_dbxtract_begin プロシージャ
使用 165
ユニークなプライマリ・キー 165
sp_hook_ssrmt_begin ストアド・プロシージャ
SQL 構文 413
sp_hook_ssrmt_end ストアド・プロシージャ
SQL 構文 414
sp_hook_ssrmt_message_apply_begin ストアド・プロシージャ
SQL 構文 417
sp_hook_ssrmt_message_apply_end ストアド・プロシージャ
SQL 構文 417
sp_hook_ssrmt_message_missing ストアド・プロシージャ
SQL 構文 416
sp_hook_ssrmt_message_sent ストアド・プロシージャ
SQL 構文 416
sp_hook_ssrmt_receive_begin ストアド・プロシージャ
SQL 構文 415
sp_hook_ssrmt_receive_end ストアド・プロシージャ
SQL 構文 415
sp_hook_ssrmt_send_begin ストアド・プロシージャ
SQL 構文 415
sp_hook_ssrmt_send_end ストアド・プロシージャ
SQL 構文 416
sp_hook_ssrmt_shutdown ストアド・プロシージャ
SQL 構文 414
sp_link_option プロシージャ
構文 496
sp_modify_article プロシージャ
構文 498
sp_modify_remote_table プロシージャ
構文 500
sp_passthrough_piece プロシージャ
構文 504
説明 356
sp_passthrough_stop プロシージャ
構文 506
説明 356
sp_passthrough_subscription プロシージャ
構文 507
説明 356
sp_passthrough_user プロシージャ
構文 509
説明 356
sp_passthrough プロシージャ
構文 502
説明 356
sp_populate_sql_anywhere プロシージャ
構文 510
説明 250
sp_publisher プロシージャ
構文 511
sp_queue_clean プロシージャ
構文 512
sp_queue_confirmed_delete_old プロシージャ
構文 513
sp_queue_confirmed_transaction プロシージャ
構文 514
sp_queue_delete_old プロシージャ
構文 515
sp_queue_drop プロシージャ
SQL Remote ステーブル・キューのアンインストール 30
構文 516
sp_queue_dump_database プロシージャ
構文 517
sp_queue_dump_transaction プロシージャ
構文 518
sp_queue_get_state プロシージャ
構文 519
sp_queue_log_transfer_reset プロシージャ
構文 520

sp_queue_read プロシージャ
構文 521
sp_queue_reset プロシージャ
構文 522
sp_queue_set_confirm プロシージャ
構文 523
sp_queue_set_progress プロシージャ
構文 524
sp_queue_transaction プロシージャ
構文 525
sp_remote_option プロシージャ
構文 528
sp_remote_type プロシージャ
構文 530
sp_remote プロシージャ
構文 526
sp_remove_article_col プロシージャ
構文 533
sp_remove_article プロシージャ
構文 532
sp_remove_remote_table プロシージャ
構文 534
sp_revoke_consolidate プロシージャ
構文 535
sp_revoke_remote プロシージャ
構文 536
sp_setreplicate プロシージャ
sp_add_remote_table 484
sp_subscription_reset プロシージャ
構文 539
sp_subscription プロシージャ
構文 537
使用 251
説明 229
sp_user_extraction_hook
SQL Remote for ASE の例 205
sp_user_extraction_hook プロシージャ
説明 250
SQLANY.INI
SQL Remote 274
SQL Anywhere Studio
マニュアル x
SQL Remote
Adaptive Server Enterprise のチュートリアル 67
ASA コマンド・リファレンス 451
ASA システム・オブジェクト 419
ASA システム・テーブル 420
ASA チュートリアル 33
ASA と ASE の違い 543, 545
ASA 統合データベースの設定 41
ASA の設計概要 114
ASA プロシージャと対応する ASE 文 548
ASE コマンド・リファレンス 479
ASE のアップグレード 29
ASE のインストール 5
ASE の設計概要 180
ASE の設定 22
dbremote 59
dbxtract コマンド・ライン・ユーティリティ 391
DDL 文のレプリケート 102
Message Agent の概要 10
Message Agent のパフォーマンス 294
PUBLISH パーミッションの付与 43, 53
REMOTE パーミッションの付与 43, 53
SQL Remote for ASA の article システム・テーブル 421
ssremote 83
ssxtract ユーティリティ 391
TEMPDB 24
Windows CE で使用するメッセージ・タイプ 272
アンインストール 489, 516
オプション 406
概念 7
管理 15, 254, 336
競合検出 99
ケース・スタディ 17
更新のレプリケート 98
コンポーネント 8
サーバ間レプリケーション 19
サーバ／ラップトップ間レプリケーション 17

削除のレプリケート 98
サブスクライバ 15
サブスクリプション 12
サポートされるプラットフォーム 553
サポートされるメッセージ・システム 553
時刻のレプリケート 104
設計の原則 91
設定の例 17
説明 4
挿入のレプリケート 98
多層インストール環境 337
データ型のレプリケート 103
データベース抽出前の作業 242
データベースのアップグレード 331
データベースのアンロード 332
統合データベースの設定 52, 74
トランザクション・ログの管理 320, 325, 331, 351
トリガのレプリケート 101
配備の概要 234
バックアップ・プロシージャ 320, 325, 331, 351
パブリケーション 12
パブリケーションの作成 46
日付の競合解決 156
日付のレプリケート 104
プロシージャのレプリケート 100
メッセージ・トラッキング 306, 308
メッセージの配信 306, 307, 308
モバイル環境 15, 17
ユーザ ID 346
ユーティリティとオプションのリファレンス 375
リモート・データベースの設定 56
リモート・データベースの設定 (チュートリアル) 48
レプリケーション・システム・リカバリ・プロシージャ 289
SQL Remote Open Server
ASE のスキーマの変更 355
アーキテクチャ 361
コマンド・ライン 402
設定 365
必要な場合 360
SQLRemote Open Server
プロシージャ 372
SQLREMOTE 環境変数
代替 277
メッセージ制御パラメータの設定 274
SQL Remote システム・テーブル
#remote (ASE) 432
article (ASA) 421
articlecol (ASA) 421
publication (ASA) 422
remoteoption (ASA) 422
remoteoptiontype (ASA) 423
remotetype (ASA) 423
remoteuser (ASA) 424
sr_article (ASE) 433
sr_articlecol (ASE) 433
sr_marker (ASE) 434
sr_object (ASE) 434
sr_option (ASE) 435
sr_passthrough (ASE) 435
sr_publication (ASE) 436
sr_publisher (ASE) 436
sr_remoteoption (ASE) 437
sr_remoteoptiontype (ASE) 437
sr_remotetable (ASE) 437
sr_remotetype (ASE) 438
sr_remoteuser (ASE) 438
sr_subscription (ASE) 440
subscription (ASA) 426
SQL Remote システム・ビュー
articlecols (ASA) 427
articlecols (ASE) 442
articles (ASA) 427
articles (ASE) 442
publications (ASA) 428
subscriptions (ASA) 430
publications (ASE) 443
remoteoptions (ASA) 428
remoteoptions (ASE) 443
remotetables (ASE) 443
remotetypes (ASE) 444
remoteusers (ASA) 428
remoteusers (ASE) 444

subscriptions (ASE) 446
SQL Remote の管理 253
SQL Remote の設定
ssremote.sql 25
stableq.sql 27
SQL Remote プロシージャ
sp_add_article 480
sp_add_article_col 482
sp_add_remote_table 484
sp_create_publication 486
sp_drop_publication 487
sp_drop_remote_type 488
sp_drop_sql_remote 489
sp_grant_consolidate 490
sp_grant_remote 493
sp_link_option 496
sp_modify_article 498
sp_modify_remote_table 500
sp_passthrough 502
sp_passthrough_piece 504
sp_passthrough_stop 506
sp_passthrough_subscription 507
sp_passthrough_user 509
sp_populate_sql_anywhere 510
sp_publisher 511
sp_queue_clean 512
sp_queue_confirmed_delete_old 513
sp_queue_confirmed_transaction 514
sp_queue_delete_old 515
sp_queue_drop 516
sp_queue_dump_database 517
sp_queue_dump_transaction 518
sp_queue_get_state 519
sp_queue_log_transfer_reset 520
sp_queue_read 521
sp_queue_reset 522
sp_queue_set_confirm 523
sp_queue_set_progress 524
sp_queue_transaction 525
sp_remote 526
sp_remote_option 528
sp_remote_type 530
sp_remove_article 532
sp_remove_article_col 533
sp_remove_remote_table 534
sp_revoke_consolidate 535
sp_revoke_remote 536
sp_subscription 537
sp_subscription_reset 539
SQL 文
レプリケート 97, 102
squpdate.sql
ステーブル・キューのアップグレード 29
sr_articlecol テーブル
説明 433
sr_article テーブル
説明 433
sr_confirmed_transaction テーブル
ASE 449
SR_DATE_FORMAT オプション
レプリケーション・オプション 409
sr_marker テーブル
説明 434
sr_object テーブル
説明 434
sr_option テーブル
説明 435
sr_passthrough テーブル
説明 435
sr_publication テーブル
説明 436
sr_publisher SQL Remote テーブル
説明 436
sr_queue_coordinate テーブル
ASE 450
sr_queue_state テーブル
説明 447
sr_remoteoption SQL Remote テーブル
ASE 437
sr_remoteoptiontype SQL Remote テーブル
ASE 437
sr_remotetable テーブル
ASE 437
sr_remotetype テーブル
ASE 438
sr_remoteuser テーブル
ASE 438
sr_subscription テーブル
ASE 440

SR_TIME_FORMAT オプション
レプリケーション・オプション 410
SR_TIMESTAMP_FORMAT オプション
レプリケーション・オプション 410
sr_transaction テーブル
ASE 449
ssqueue
アーキテクチャ 361
コマンド・ライン 402
設定 365
説明 359, 402
必要な場合 360
ssremote.sql
SQL Remote for ASE の設定 25
ssremote
Message Agent 83
概要 10
コマンド 376
セキュリティ 346
説明 287, 346, 376
ssupdate.sql
SQL Remote for ASE のアップグレード 29
ssxtract ユーティリティ
オプション 394
構文 391
使用 241, 248, 249
説明 238, 391
stableq.sql
SQL Remote for ASE の設定 27
SUBSCRIBE_BY_REMOTE オプション
SQL Remote for ASA での多対多関係 149
SQL Remote for ASE の多対多関係 208
レプリケーション・オプション 410
subscription-list カラム
SQL Remote for ASE 190
SQL Remote for ASE での管理 196, 203
SQL Remote for ASE の管理 205
SQL Remote for ASE の使用 194
SQL Remote for ASE のトリガ 196, 203
subscription SQL Remote テーブル
ASA 定義 426
ASE 440
subscriptions SQL Remote ビュー
ASA 430
ASE 446
Sun Solaris
SQL Remote 555
Sybase Central
抽出ユーティリティ 243
メッセージ・タイプ 272
SYNCHRONIZE SUBSCRIPTION 文
説明 251
SYSREMOTEUSER テーブル 307
SYSREMOTEUSER 文
使用 306

T

TEMPDB
SQL Remote for ASE の条件 24
TEXT データ型
レプリケート 103
TIMESTAMP
抽出ユーティリティ 249

U

UNIX
サポートしている SQL Remote メッセージ・タイプ 269
unlink_delay 制御パラメータ
FILE メッセージ・タイプ 277
UPDATE 競合
SQL Remote for ASE の管理 210
UPDATE の競合
SQL Remote for ASA での管理 151
UPDATE 文
SQL Remote for ASA でのレプリケーション 136
SQL Remote for ASA のパブリケーション 137
SQL Remote for ASE でのレプリケーション 193
SQL Remote の競合検出 99

競合検出　103
領域の再編成　98
レプリケート　97, 98
Userid 制御パラメータ
VIM メッセージ・タイプ　285
user 制御パラメータ
FTP メッセージ・タイプ　278
-u オプション
Message Agent　290

V

VERIFY_ALL_COLUMNS オプション
SQL Remote for ASA での使用　155
レプリケーション・オプション　411
VERIFY_THRESHOLD オプション
メッセージ・サイズ　103
レプリケーション・オプション　411
VIM メッセージ・タイプ
概要　285
使用　269
制御パラメータ　285
説明　285

W

WHERE 句
ASA の SQL Remote パブリケーション　119
ASE の SQL Remote パブリケーション　184
Windows
SQL Remote　555
サポートしている SQL Remote メッセージ・タイプ　269
Windows CE
ActiveSync レプリケーション　555
サポートしている SQL Remote メッセージ・タイプ　272
レプリケーション　555

あ

アーティクル
SQL Remote for ASA での作成　115
SQL Remote for ASA での追加　123
SQL Remote for ASA のプロパティ　115
SQL Remote for ASA の有効性　129
SQL Remote for ASE のカラムワイズ分割　182
SQL Remote for ASE のテーブル全体　181
SQL Remote for ASE の有効性　188
SQL Remote for ASE のローワイズ分割　183
[アーティクル作成] ウィザード
SQL Remote for ASA のアーティクルの追加　124
アイコン
マニュアルで使用　xv
アップグレード
COMPRESSION オプション　303
SQL Remote　303
SQLRemote　235
SQL Remote for ASA　331
SQL Remote for ASE　29
レプリケーション　331
アドレス
FTP　278
SMTP　280
SMTP/POP　283
パブリッシャの設定　270
ファイル共有　277
アンインストール
SQL Remote for ASE　30
SQL Remote ステーブル・キュー　30
データベースの SQL Remote システム・オブジェクト　489
データベースのステーブル・キュー・オブジェクト　516
暗号化
SQL Remote のメッセージ　291
アンロード
SQL Remote for ASA 統合データベース　332

い

イベント・フック
sp_hook_dbremote_begin ストアド・プロシージャ 413
sp_hook_dbremote_end ストアド・プロシージャ 414
sp_hook_dbremote_message_apply_begin ストアド・プロシージャ 417
sp_hook_dbremote_message_apply_end ストアド・プロシージャ 417
sp_hook_dbremote_message_missing ストアド・プロシージャ 416
sp_hook_dbremote_message_sent ストアド・プロシージャ 416
sp_hook_dbremote_receive_begin ストアド・プロシージャ 415
sp_hook_dbremote_receive_end ストアド・プロシージャ 415
sp_hook_dbremote_send_begin ストアド・プロシージャ 415
sp_hook_dbremote_send_end ストアド・プロシージャ 416
sp_hook_dbremote_shutdown ストアド・プロシージャ 414
sp_hook_ssrmt_begin ストアド・プロシージャ 413
sp_hook_ssrmt_end ストアド・プロシージャ 414
sp_hook_ssrmt_message_apply_begin ストアド・プロシージャ 417
sp_hook_ssrmt_message_apply_end ストアド・プロシージャ 417
sp_hook_ssrmt_message_missing ストアド・プロシージャ 416
sp_hook_ssrmt_message_sent ストアド・プロシージャ 416
sp_hook_ssrmt_receive_begin ストアド・プロシージャ 415
sp_hook_ssrmt_receive_end ストアド・プロシージャ 415
sp_hook_ssrmt_send_begin ストアド・プロシージャ 415
sp_hook_ssrmt_send_end ストアド・プロシージャ 416
sp_hook_ssrmt_shutdown ストアド・プロシージャ 414
コミット禁止 413
同期 413
ロールバック禁止 413
インターネット
SQL Remote 280
電子メール 280

う

ウィザード
アーティクル作成 124
データベースの抽出 243, 390
パブリケーション作成 115, 117, 119, 121
メッセージ・タイプの作成 270
失われたメッセージ
説明 298

え

エラー
Message Agent のレポート 315
SQL Remote for ASA の処理 316
SQL Remote for ASA のレポート 315
SQL Remote for ASE の処理 349
SQL Remote for ASE のレポート 349
SQL 文とレプリケーション 110
エラーではない競合 315
競合はエラーではない 349
通知 316, 349
プライマリ・キー・レプリケーション 109
無視 315
レプリケーションでの発生の種類 109
レポート 349
エラー処理
SQL Remote for ASA 316
SQL Remote for ASA のエラーの無視 315
SQL Remote for ASA のデフォルト 315

SQL Remote for ASE 349
SQL Remote for ASE のデフォルト 349
エラーのレポート 349
Message Agent 315
エンコード
カスタム 304
説明 304

お

オプション
BLOB_THRESHOLD 406
COMPRESSION 406
DELETE_OLD_LOGS 406
EXTERNAL_REMOTE_OPTIONS 406
QUALIFY_OWNERS 406
QUOTE_ALL_IDENTIFIERS 406
REPLICATION_ERROR 406
SAVE_REMOTE_PASSWORDS 406
SQL Remote 406
SQL Remote for ASE で使用する SUBSCRIBE_BY_REMOTE 208
SR_DATE_FORMAT 406
SR_TIME_FORMAT 406
SR_TIMESTAMP_FORMAT 406
SUBSCRIBE_BY_REMOTE 149, 406
VERIFY_ALL_COLUMNS 406
VERIFY_THRESHOLD 103, 406
抽出ユーティリティ 411
オフセット
トランザクション・ログ 307

か

カーソル
パススルー・モード 338
レプリケーション 338
外部キー
ASA の SQL Remote パブリケーション 128
ASE の SQL Remote パブリケーション 187, 190
SQL Remote for ASA のパブリケーション 132
SQL Remote for ASA の領域の再編成 136
SQL Remote for ASE の領域の再編成 193
カラム
SQL Remote for ASA の選択カラムのパブリッシュ 117
環境変数
SQLREMOTE 274
環境変数オプション
Message Agent 379
管理
SQL Remote 15, 254
SQL Remote for ASA 311, 312
SQL Remote for ASE 339

き

規則
表記 xiii
起動
Message Agent 312
キャッシュ
メッセージ 296
競合
Message Agent 出力にない競合 315, 349
SQL Remote for ASA での解決 153, 156
SQL Remote for ASA での管理 151
SQL Remote for ASA での処理 152
SQL Remote for ASA でのレポート 159
SQL Remote for ASA の解決方法 157
SQL Remote for ASA のプライマリ・キー 162, 168
SQL Remote for ASA のロック 129
SQL Remote for ASE でのロック 188
SQL Remote for ASE の #remote 217
SQL Remote for ASE の解決 212
SQL Remote for ASE の管理 210
SQL Remote for ASE のプライマリ・キー 222
SQL Remote for ASE の例 214
SQL Remote の処理 110
SQL Remote を使った回避 109
SQL Remote を使った検出 110
VERIFY_ALL_COLUMNS オプション 155

エラーではない　349
エラーではない競合　315
レプリケーション　109
競合解決
SQL Remote for ASA での実装　153
SQL Remote for ASA のトリガ　153, 154, 156
SQL Remote for ASA の方法　157
SQL Remote for ASE の #remote　217
SQL Remote for ASE の実装　212
SQL Remote for ASE の制限事項　213
SQL Remote for ASE のトリガ　212
SQL Remote for ASE の例　214
競合検出
SQLRemote　99
長いデータ型　103
競合の検出
SQL Remote for ASA　152
SQL Remote for ASE　211

く

グループ
抽出　246, 393
グローバル・オートインクリメント
SQL Remote for ASA のユニークな値の生成　163
クワイエット・システム
定義　355

け

継続モード
Message Agent　287

こ

更新
トランザクション・ログの情報　107
互換性
SQL Remote レプリケーションの ASE と ASA　92, 248, 249, 250
SQL Remote レプリケーションのデータベース間　92
コマンド・ライン
Message Agent　379
環境変数　379
[壊れたメッセージを削除しています。] エラー
説明　303

さ

サーバ間レプリケーション
説明　19
サービス
Message Agent の実行　312
再送要求
説明　298
メッセージ　297
再ロード・ファイル
SQL Remote のデータベース抽出　241
削除
SQL Remote for ASA のパブリケーション　125
メッセージ・タイプ　272, 274
作成
SQL Remote for ASA のアーティクル　115, 123
SQL Remote for ASA のカラムワイズ分割のパブリケーション　117
SQL Remote for ASA のサブスクリプション　175
SQL Remote for ASA のパブリケーション　46, 115
SQL Remote for ASA のローワイズ分割のパブリケーション　118
SQL Remote for ASE のカラムワイズ分割のアーティクル　182
SQL Remote for ASE のカラムワイズ分割のパブリケーション　182
SQL Remote for ASE のサブスクリプション　229
SQL Remote for ASE のテーブル全体のパブリケーション　181

SQL Remote for ASE のパブリケーション 181
SQL Remote for ASE のローワイズ分割のアーティクル 183
SQL Remote for ASE のローワイズ分割のパブリケーション 183
SQL Remote のサブスクリプション 79
メッセージ・タイプ 270, 272
サブクエリ
SQL Remote for ASA のパブリケーション 132
サブスクリプション
Message Agent 108
sp_subscription_reset プロシージャ (SQL Remote) 539
sp_subscription プロシージャ (SQL Remote) 537
SQL Remote for ASA での管理 175
SQL Remote for ASA での作成 175
SQL Remote for ASA での設定 175
SQL Remote for ASE での作成 229
SQL Remote for ASE の設定 229
SQL Remote の作成 79
SQL Remote の同期 241, 251
SQL Remote のパフォーマンス 108
SQL Remote レプリケーション 12
サブスクリプション式
SQL Remote for ASA のサブクエリ 132
SQL Remote for ASA の使用 121
SQL Remote for ASE 185
SQL Remote トランザクション・ログ 107
SQL Remote の評価のコスト 107
多くの値を含む SQL Remote 107
サブスクリプション・ビュー
SQL Remote for ASE での使用 207
サポート
ニュースグループ xviii
参照整合性
SQL Remote for ASA 159
SQL Remote for ASE 221
レプリケーション 159
サンプル
SQL Remote for ASA の policy データベースの例 141

し

時間
レプリケーション 104, 410
システム・オブジェクト
dbo ユーザ 389
システム・テーブル
SQL Remote for ASA 420
ステーブル・キュー 447
実行
Message Agent 312
照合順
SQL Remote 93

す

スキーマの変更
SQL Remote for ASE 355
スキーマ変更
SQLRemote Open Server 372
ステーブル・キュー
Replication Server 362
SQL Remote Open Server 362
ssqueue 362
クリーニング 381
システム・テーブル 447
設定 27
ストアド・プロシージャ
SQLRemote Open Server 372
SQL Remote のパススルー・モードで使用 336
パススルー・モード 338
レプリケート 100
スレッド
Message Agent 294

せ

制御文
レプリケーション 338
制限
ASE 間レプリケーション 550

制限事項
SQL Remote for ASE の競合解決 213
生成
SQL Remote for ASA のユニークなカラム値 162
制約
抽出ユーティリティ 249
設計
ASE の SQL Remote パブリケーション 187
SQLRemote for ASA 113
SQL Remote for ASA の多対多関係 141
SQL Remote for ASA の多対多関係の例 146
SQL Remote for ASA のパフォーマンス 130
SQL Remote for ASA のパブリケーション 128
SQL Remote for ASE 179
SQL Remote for ASE でのロック 188
SQL Remote for ASE の多対多関係 200
SQL Remote for ASE の多対多関係の例 200
SQL Remote の原則 91
レプリケーション・エラーとパブリケーション 109
レプリケーション競合とパブリケーション 109
接続
Message Agent 289
設定
SQL Remote for ASA のサブスクリプション 175
SQLRemote for ASA のパブリケーション 115
SQL Remote for ASA のリモート・データベース 56
SQL Remote for ASA のリモート・データベース (チュートリアル) 48
SQL Remote for ASE 21
SQL Remote for ASE の TEMPDB 24
SQL Remote for ASE のステーブル・キュー 27
SQL Remote Open Server 365
SQL Remote Open Server に対する Replication Server 369
SQL Remote の ASA 統合データベース 41
SQL Remote の統合データベース 52, 74
設定ファイル
Message Agent のオプション 379

そ

送信頻度
Message Agent 287
選択 263

た

ターンアラウンド・タイム
レプリケーションのパフォーマンス 294
待機時間
Message Agent 298
多層インストール環境
SQL Remote for ASA のパススルー文 337
SQL Remote のパーミッション 266
多対多関係
SQL Remote for ASA でのトリガ 146
SQL Remote for ASA の SUBSCRIBE_BY_REMOTE オプション 149
SQL Remote for ASA のパブリケーション設計 141
SQL Remote for ASA の領域の再編成の例 146
SQL Remote for ASA の例 144
SQL Remote for ASE の SUBSCRIBE_BY_REMOTE オプション 208
SQL Remote for ASE のパブリケーション設計 200
SQL Remote for ASE の例 200
ダンプ
コーディネート 372

ち

違い
SQL Remote バージョン 545
抽出

SQL Remote for ASE のパフォーマンス 197, 207
SQL Remote データベース 233, 238
SQL Remote の再ロード・ファイル 241
Sybase Central の使用 243
多くのデータベース 245
カスタム・プロシージャ 245
抽出前の作業 242
複数のオペレーティング・システムと SQL Remote 239
プロシージャの設計 245
抽出ユーティリティ
ASE 248
SQL Remote 238, 391
SQL Remote for ASE の使用 249
SQL Remote の使用前の作業 242
SQL Remote レプリケーションの使用 241
Sybase Central からの使用 243
オプション 394, 411
グループ 246
構文 391
制限 247
目的 247
チュートリアル
SQL Remote for Adaptive Server Enterprise 67
SQL Remote for ASA 33

つ

追加
SQL Remote for ASA のアーティクル 123

て

データ移動テクノロジ
SQL Remote レプリケーション 3
データ型
レプリケート 103
データベース
SQL Remote の再ロード・ファイルからの作成 241
SQL Remote の抽出前の作業 242
レプリケート 57
[データベース作成] ウィザード
ASE と互換性のあるデータベースの作成 92
[データベース抽出] ウィザード
SQL Remote の使用 243
データベース抽出ユーティリティ [dbxtract]
SQL Remote 391
[データベースの抽出] ウィザード
Sybase Central でのリモート・データベースの抽出 390
データベースのロード
SQL Remote 241
データ・リカバリ
SQL Remote 289
テーブル
SQL Remote for ASA でのパブリッシュ 115
SQL Remote for ASA のカラムワイズ分割 117
SQL Remote for ASA のローワイズ分割 118
SQL Remote for ASE でのパブリッシュ 181
SQL Remote for ASE のカラムワイズ分割 182
SQL Remote for ASE のローワイズ分割 183
テーブル全体
SQL Remote for ASA でのパブリッシュ 115
デーモン
dbremote 386
Message Agent 386
ssremote 386
テクニカル・サポート
ニュースグループ xviii
テスト
SQL Remote 235
デフォルト
抽出ユーティリティ 249
電子メール
MAPI 269
SMTP 269
VIM 269

と

同期

dbxtract の使用 241
SQL Remote の使用 238
SQL Remote を使用するデータベース 233
ssxtract の使用 241
イベント・フック 413
カスタマイズ 413
複数のオペレーティング・システムと SQL Remote 239
メッセージ・システムを使用した SQLRemote データベース 251
統合データベース
SQL Remote としての Adaptive Server Enterprise 74
SQL Remote としての ASA 41, 52
トラブルシューティング
SQL Remote エラー 291
トランケーション・ポイント
設定 353
トランザクション・ログ 353
トランザクション
レプリケート 97
トランザクション・ログ
Message Agent 376
SQL Remote 15, 97, 376
SQL Remote for ASE 353
SQL Remote Open Server のスキャン 360
SQL Remote パブリケーション 107
update 文 107
オフセット 307
管理 353
メッセージ・トラッキング 307
レプリケーション 15, 97
トランザクション・ログ・ミラーリング
レプリケーション 320, 351
トリガ
RESOLVE UPDATE 154, 156
SQL Remote 101
SQL Remote for ASA 160
SQL Remote for ASA での競合解決 153, 154, 156
SQL Remote for ASA の領域の再編成 137
SQL Remote for ASA を使用するレプリケーション 137
SQL Remote for ASE の subscription-list カラム 196, 203
SQL Remote for ASE の競合解決 212
レプリケーション 101, 160
レプリケーション・オプション 102
取り消し
CONSOLIDATE パーミッション 260
PUBLISH パーミッション 256
REMOTE パーミッション 260

な

名前を付けた制約
抽出ユーティリティ 249
名前を付けたデフォルト
抽出ユーティリティ 249

に

ニュースグループ
テクニカル・サポート xviii

は

パーミッション
CONSOLIDATE 260
CONSOLIDATE の付与 264
PUBLISH 43, 53, 256
REMOTE 53
Remote 260
Remote の取り消し 265
SQL Remote の管理 254
多層 SQL Remote インストール環境 266
バイナリ・ラージ・オブジェクト
レプリケーション 407
レプリケート 103
配備
SQL Remote データベース 233, 235
パススルー・モード
SQL Remote for ASA 335

SQL Remote for ASA の使用　336
SQL Remote for ASE　356
バッチ　338
レプリケートされないオペレーション　338
パスワード
抽出ユーティリティ　249
保存　409
バックアップ
SQL Remote　290
SQL Remote for ASA　320
SQL Remote for ASE　351
リモート・データベース　331
レプリケーション　325
バッチ
パススルー・モード　338
バッチ・モード
Message Agent　287
パフォーマンス
ASE の ssxtract　197
Message Agent　294
SQL Remote　294
SQL Remote for ASA の設計ヒント　130
SQL Remote for ASE の ssxtract　207
SQL Remote のサブスクリプション数　108
SQL Remote のパブリケーション　107
受信メッセージ　297
スレッド　294
データベース抽出　245
メッセージ送信　300
レプリケーションのスループット　294
レプリケーションのターンアラウンド・タイム　294
パブリケーション
SQL Remote for ASA からの削除　125
SQL Remote for ASA サブスクリプションの管理　175
SQL Remote for ASA での作成　115
SQL Remote for ASA での設計　159
SQL Remote for ASA での注意　126
SQL Remote for ASA での変更　123
SQL Remote for ASA の WHERE 句の使用　119
SQL Remote for ASA の外部キー　128, 132
SQL Remote for ASA のカラムワイズ分割　117
SQL Remote for ASA のサブクエリ　132
SQL Remote for ASA の参照整合性　159
SQL Remote for ASA のサンプル　65
SQL Remote for ASA の設計　128
SQLRemote for ASA の設定　115
SQL Remote for ASA の多対多関係　141, 144, 146
SQL Remote for ASA のテーブル全体　115
SQL Remote for ASA のトランザクション　129
SQL Remote for ASA のパフォーマンス　130
SQL Remote for ASA のプライマリ・キー　128, 160, 162, 168
SQL Remote for ASA のプロパティ　115
SQL Remote for ASA のローワイズ分割　118
SQL Remote for ASA のロック　129
SQL Remote for ASE での作成　181
SQL Remote for ASE での設計　187
SQL Remote for ASE でのロック　188
SQL Remote for ASE の WHERE 句の使用　184
SQL Remote for ASE の外部キー　187, 190
SQL Remote for ASE のカラムワイズ分割　182
SQL Remote for ASE の参照整合性エラー　221
SQL Remote for ASE の参照整合性エラーの回避　221
SQL Remote for ASE の設定　181
SQL Remote for ASE の多対多関係　200
SQL Remote for ASE の多対多関係の例　200
SQL Remote for ASE のトランザクション　188
SQL Remote for ASE のプライマリ・キー　187, 222
SQL Remote for ASE のローワイズ分割　183
SQL Remote の設計　92, 109
SQL Remote のパフォーマンス　107
SQL Remote レプリケーション　12
多くの SQL Remote パブリケーションのテーブル　107
[パブリケーション作成] ウィザード
SQL Remote for ASA でのサブスクリプション式を使用したアーティクルの作成　121
SQL Remote for ASA の WHERE 句の使用　119
SQL Remote for ASA のカラムワイズ分割　117
SQL Remote for ASA パブリケーションの作成

115
パブリケーション設計
SQLRemote for ASA 113
SQL Remote for ASA でのサブスクリプション式の使用 121
SQL Remote for ASE 179
SQL Remote for ASE のサブスクリプション式の使用 185
多くの SQL Remote for ASA サブスクライバ 121
多くの SQL Remote for ASE サブスクライバ 185
パブリッシャ
アドレス 270
作成 256
説明 256
チュートリアル 44
データベースへの追加 53
パブリッシュ
SQL Remote for ASA の選択カラム 117

ひ

日付
SQL Remote for ASA での競合解決 156
レプリケーション 104, 409
表記
規則 xiii
頻度
送信 263

ふ

フィードバック
提供 xviii
マニュアル xviii
付与
CONSOLIDATE パーミッション 260
PUBLISH パーミッション 256
REMOTE パーミッション 260
プライマリ・キー
ASA の SQL Remote パブリケーション 128
ASE の SQL Remote パブリケーション 187
SQL Remote for ASA 160
SQL Remote for ASA のプールを使用したユニークな値の生成 168
SQL Remote for ASA のユニークな値の生成 162
SQL Remote for ASE 221
SQL Remote for ASE の一意性 222
SQL Remote パブリケーション 97
レプリケーション 160, 163, 221
レプリケーション・エラー 109
プライマリ・キー・プール
SQL Remote for ASA での補充 170
SQL Remote for ASA のデフォルトのグローバル・オートインクリメントを使用したユニークな値の生成 163
SQL Remote for ASE での補充 224
SQL Remote for ASE でのレプリケーション 223
SQL Remote for ASE の使用 222
レプリケーション 169
プラットフォーム
SQL Remote がサポートするオペレーティング・システム 553
プロシージャ
SQL Remote 100
SQLRemote Open Server 372
対応する文 548
パススルー・モード 338
レプリケート 100
プロシージャ・グループ
抽出ユーティリティ 249
プロパティ
SQL Remote for ASA のアーティクル 115
SQL Remote for ASA のパブリケーション 115
メッセージ・タイプのプロパティ 270
プロファイル
Microsoft Exchange 284
文
レプリケート 97
分割
SQL Remote for ASA のカラムワイズ 117

SQL Remote for ASA のローワイズ　118
SQL Remote for ASE のカラムワイズ　182
SQL Remote for ASE のローワイズ　183

へ

変更
SQL Remote for ASA のパブリケーション　123
メッセージ・タイプ　270, 272

ほ

ポーリング
メッセージ　297

ま

マニュアル
SQL Anywhere Studio　x

め

メッセージ
SQL Remote　306, 308
SQL Remote for ASA の受信　61
SQL Remote for ASA の送信　60
SQL Remote の受信　84
SQL Remote の送信　84
SQL Remote のデータベースの同期　251
圧縮　303, 407
エンコード　303
カスタム・エンコード　304
キャッシュ　296
サイズの制御　103
再送　297
トラッキング　306, 307, 308
配信　306, 308
メッセージ・システム
サポートされる　554
メッセージ・タイプ
FTP　278
MAPI　283, 284
SMTP　280, 281
SQL Remote の管理　254
VIM　285
削除　272, 274
作成　270, 272
処理　270
説明　269
パラメータ　274
ファイル共有　276, 277
プロパティの編集　270
変更　270, 272
メッセージ・タイプ作成ウィザード
Sybase Central のメッセージ・タイプの追加　270
メッセージ・トラッキング
SQL Remote の管理　254
メッセージの受信
SQL Remote　84
SQL Remote for ASA　61
メッセージの送信
SQL Remote　84
SQL Remote for ASA　60
メッセージ・リンク
サポートされる　554
メッセージ・リンク・パラメータ
EXTERNAL_REMOTE_OPTIONS レプリケーション・オプション　408
設定　274
メディア障害
SQL Remote　289
メンテナンス・リリース
SQL Remote のアップグレード　235

も

文字セット
SQL Remote の変換　93
SQL Remote レプリケーションの互換性　92
モバイル環境

SQL Remote 15, 17
SQL Remote for ASA のパブリケーション設計 121
SQL Remote for ASE のパブリケーション設計 185

ゆ

ユーザ ID
Message Agent 346
グループの抽出 246
ユニークな値
SQL Remote for ASA のデフォルトのグローバル・オートインクリメントを使用した生成 163
SQL Remote for ASA のプールを使用した生成 168
ユニークなカラム値
SQL Remote for ASA での生成 162

ら

ラップトップ・コンピュータ
SQL Remote レプリケーション 17

り

リカバリ
SQL Remote 289
リセット
サブスクリプション 539
リモート・データベース
Remote パーミッション 261
SQL Remote for ASA の設定 (チュートリアル) 48
領域の再編成
SQL Remote for ASA でのトリガ 146
SQL Remote for ASA の外部キー 136
SQL Remote for ASA の多対多関係 146
SQL Remote for ASE 196
SQL Remote for ASE の subscription-list カラム 194
SQL Remote for ASE の外部キー 193
UPDATE のレプリケート 98

れ

例
SQL Remote for ASE 190
レジストリ
SQL Remote 274
レプリケーション
ASA のパブリケーション設計 128
ASE の制限 550
ASE の設計 187
blob 103
dbremote 376
Message Agent 376
SQL Remote for ASA の参照整合性エラー 159
SQL Remote for ASA のトリガ 160
SQL Remote for ASA のプライマリ・キー 162
SQL Remote for ASA のプライマリ・キー・エラー 160
SQL Remote for ASA のプライマリ・キー・プール 168
SQL Remote for ASE の参照整合性エラー 221
SQL Remote for ASE のプライマリ・キー 222
SQL Remote for ASE のプライマリ・キー・エラー 221
SQL 文 97, 335
ssqueue 402
ssremote 376
オプション 406
カーソル処理 338
カーソル文 338
管理 15
競合 109
ケース・スタディ 17
サーバ間 19
サーバ／ラップトップ間レプリケーション 17
サブスクリプション 12
ストアド・プロシージャ 338

制御文　338
設定の例　17
データ型　103
データ定義文　102
データベースのアップグレード　331
データ・リカバリ　289
同期　391
トランザクション・ログ　15, 97
トランザクション・ログの管理　320, 325, 331, 351
トリガ　101
パススルー・モード　335
バックアップ・プロシージャ　320, 325, 331, 351
パブリケーション　12
複数のオペレーティング・システム　239
プロシージャ　100
モバイル環境　17
レプリケーション・オプション
BLOB_THRESHOLD　407
COMPRESSION　407
DELETE_OLD_LOGS　408
EXTERNAL_REMOTE_OPTIONS　408
QUALIFY_OWNERS　408
QUOTE_ALL_IDENTIFIERS　408
REPLICATION_ERROR　110, 316, 349, 408
SAVE_REMOTE_PASSWORDS　409
SR_DATE_FORMAT　409
SR_TIME_FORMAT　410
SR_TIMESTAMP_FORMAT　410
SUBSCRIBE_BY_REMOTE　410
VERIFY_ALL_COLUMNS　411
VERIFY_THRESHOLD　411
レプリケーション定義
Replication Server　369
レプリケーションの競合
SQL Remote for ASA での管理　151
SQL Remote for ASE の管理　210
レプリケーションの役割
付与　346
レポート
SQL Remote for ASA の競合　159
エラー　349

ろ

ロール
抽出ユーティリティ　249
ログ管理
SQL Remote　290
ログ転送インタフェース
Message Agent　353
ロック
SQL Remote for ASA　129
SQL Remote for ASA のパブリケーション　168
SQL Remote for ASA のパブリケーション設計　162
SQL Remote for ASE のパブリケーション設計　222
レプリケーション・システム　188

わ

ワーカ・スレッド
Message Agent　294